INFORMATION TO USERS

This manuscript has been reproduced from the microfilm master. UMI films the text directly from the original or copy submitted. Thus, some thesis and dissertation copies are in typewriter face, while others may be from any type of computer printer.

The quality of this reproduction is dependent upon the quality of the copy submitted. Broken or indistinct print, colored or poor quality illustrations and photographs, print bleedthrough, substandard margins, and improper alignment can adversely affect reproduction.

In the unlikely event that the author did not send UMI a complete manuscript and there are missing pages, these will be noted. Also, if unauthorized copyright material had to be removed, a note will indicate the deletion.

Oversize materials (e.g., maps, drawings, charts) are reproduced by sectioning the original, beginning at the upper left-hand corner and continuing from left to right in equal sections with small overlaps.

Photographs included in the original manuscript have been reproduced xerographically in this copy. Higher quality 6" x 9" black and white photographic prints are available for any photographs or illustrations appearing in this copy for an additional charge. Contact UMI directly to order.

Bell & Howell Information and Learning
300 North Zeeb Road, Ann Arbor, MI 48106-1346 USA

UMI®
800-521-0600

INTER-LAN SERVICES OVER BROADBAND SATELLITE SYSTEMS

Akbar Garjani

A Thesis

in

The Department

of

Electrical and Computer Engineering

Presented in Partial Fulfilment of the Requirements

for the Degree of Master of Applied Science at

Concordia University

Montreal, Quebec, Canada

April 2000

ABSTRACT

INTER-LAN SERVICES OVER BROADBAND SATELLITE SYSTEMS

AKBAR GARJANI

In this thesis we examine the feasibility of providing interconnections of Local Area Networks (LANs) in various remote plants of the same company in a mesh topology to form a Private Business Network (PBN) over a broadband satellite communications system. Such a PBN consists of several User Terminals (UTs); each connected to the router of a LAN in a given plant. A satellite system supports a number of PBNs by dynamically sharing the satellite resources. We considered Internet Protocol (IP) based LANs using IPv6 and Resource Reservation Protocol (RSVP) to support both real-time and non-real time traffic. We propose a Medium Access Control (MAC) protocol for broadband satellite Inter-LAN services with a special emphasis on the convergence sublayer. In order to efficiently utilize the satellite resources in serving bursty multimedia traffic, the Combined Free- and Demand -Assignment Multiple Access (CFDAMA) techniques are used in conjunction with Multiple-Frequency Time-Division Multiple-Access (MF-TDMA) format for Dynamic Capacity Allocation (DCA). Functions of the UT and Master Control Station (MCS) are described. We describe the translation of Quality of Service (QoS) requirements to service categories and the handling of aggregated requests. We provide a formal model of the MAC layer using the Specification and Description Language (SDL). Moreover, the proposed SDL model is simulated and validated using the ObjectGEODE tool.

Acknowledgement

I would like to express my deepest respect and gratitude to my supervisors Dr. Tho Le-Ngoc and Dr. Ferhat Khendek without whose expert guidance and encouragement the quality and extend of this research would not have been possible. This work was partially supported by Canadian Institute for Telecommunications Research (CITR).

I also wish to express my sincere thanks to my colleagues Tallal Elshabrawy and Mohammad Ashour who provided helpful and constructive discussions about the various aspects of this thesis.

Finally, I would like to acknowledge the friendship, support and encouragement I have received from all my friends at Concordia University.

To My Beloved Mother

Table of Contents

LIST OF FIGURES

LIST OF TABLES

LIST OF ACRONYMS

ARP	Address Resolution Protocol
ATM	Asynchronous Transfer Mode
BSA	Broadband Satellite Access
BTS	Base Transceiver Station
CDMA	Code-Division Multiple Access
CFDAMA	Combined Free- and Demand- Assignment Multiple Access
CODE	Cooperative Olympus Data Experiment
CSMA	Carrier Sense Multiple Access
CSMA/CD	Carrier Sense Multiple Access with Collision Detection
CSU	Channel Service Unit
DAMA	Demand -Assigned Multiple Access
DCA	Dynamic Capacity Allocation
DHCP	Dynamic Host Configuration Protocol
DSU	Data Service Unit
ESA	European Space Agency
ES-IS	End System to Intermediate System
ESTELLE	Extended Finite State Machine Language
FBA/DAMA	Fixed Bandwidth Assignment/ Demand Assigned Multiple Access
FDMA	Frequency Division Multiple Access
FDT	Formal Description Technique
FIFO	First In First Out
FTP	File Transfer Protocol

GEO	Geostationary Earth Orbith
HTTP	Hypertext Transfer Protocol
IGMP	Internet Group Management Protocol
IP	Internet Protocol
IPv6	Internet Protocol version 6
LAN	Local Area Network
LEO	Low Earth Orbit
LLC	Logical Link Control
LOTOS	Language of Temporal Ordering Specification
MAC	Medium Access Control
MASPnet	Multiple Access- Type Satellite Packet Network
MCS	Master Control Station
MF-TDMA	Multiple-Frequency Time-Division Multiple Access
MSC	Message Sequence Chart
NV	Network Video
PAR	Peak-to-Average Ratio
PBN	Private Business Network
PCM	Pulse Code Modulation
PDU	Protocol Data Unit
PSTN	Public Switched Telephone Network
QCM	Queue and Capacity Manager
QoS	Quality of Service
RA/DAMA	Random Access/ Demand -Assigned Multiple Access

RSVP	Resource Reservation Protocol
SATINE	SATellite Internetworking Experiment
SAT-MAC	Satellite Medium Access Control
SDL	Specification and Description Language
SDCS	Satellite Digital Communication Service
SNMP	Simple Network and Management Protocol
STS	Subscriber Transceiver Station
SWAN	Satellite Wide Area Network
TCP/IP	Transmission Control Protocol/Internet Protocol
TDMA	Time Division Multiple Access
TFTP	Trivial File Transfer Protocol
UCD	Upper Channel Descriptor
UT	User Terminal
VBR	Variable Bit Rate
VSAT	Very Small Aperture Terminal
WAN	Wide Area Network
WWW	World Wide Web

Chapter 1

Introduction

Globalization of business has led to numerous gigantic companies with several offices spread in all over the world. Requirements for an efficient and fast decision-making mechanism in such widespread companies necessitate the interconnection of the LANs in remote offices. Furthermore, the needs for security and to privately control the network have increased the demands for some sort of Private Business Networks (PBN).

PBNs are usually required for high volume and long distance data traffic. Interconnections of LANs to build PBNs are implemented by using various mediums. Advancements in satellite technology and its inherent advantages, such as broadcast capability and global connectivity make the satellite communications system a better choice in building a PBN. However, satellite system faces some challenges stemmed from its longer delay and its power and bandwidth constraints.

Nowadays, Internet is increasingly becoming a popular multimedia and application-rich environment, led by the huge popularity of the World Wide Web (WWW). Interconnected LANs are based on IP to provide both real-time services such as videoconferencing, web browsing, and non-real time best-effort services such as File Transfer Protocol (FTP). However, bursty traffic characteristics of the multimedia traffic

and its different QoS requirements according to the various types of services causes other challenges to provide QoS guarantees in PBNs over broadband satellite systems.

The challenge for satellite communications systems in providing Inter-LAN services is to have a suitable MAC and DCA to support various QoS requirements of bursty traffic while efficiently utilizing the limited satellite communications systems resources.

1.1 Scope and Contributions

In this thesis, we consider interconnections of various remote plants of the same company in a mesh topology to form a PBN over satellite communication systems. We present the system configuration and describe its components. A protocol architecture for this configuration to support IPv6 [1] and multimedia QoS requirements is proposed. Our special emphasis is to provide a MAC protocol capable of efficient access to the satellite medium and dynamic capacity allocation to support various QoS requirements. Regarding this issue our main contributions in this thesis are:

i. Provision of the structure, functionality and specifications of the required MAC layer.

ii. Formal description of the convergence sub-layer on top of an abstract model of the SAT-MAC sub-layer using SDL [4,11].

iii. SDL model simulation and validation against general and specific properties using ObjectGEODE tool from Verilog [5].

In order to achieve a proper MAC protocol for Inter-LAN services over broadband satellite systems we introduce the followings in our proposed model:

i. A mechanism to support RSVP [2] as an upper layer control in order to provide the various QoS requirements is proposed.

ii. CFDAMA [3] dynamic capacity allocation technique is used to support the QoS.

iii. Translation of QoS requirements to service categories and aggregation of connection requests are performed by the UTs to alleviate the processing loads in the MCS.

iv. Dynamic service establishment mechanism in UT and MCS is provided.

v. Admission and policy control functions are implemented.

1.2 Thesis Outline

The rest of the thesis is structured as follows:

Chapter 2 gives an introduction to the Wide Area Network (WAN) configurations and their concepts to support Inter-LAN services. WAN's functionality and its potential problems in controlling and maintaining privacy, and reachability problems to remote and inaccessible areas are discussed. The advantages and constraints of the broadband satellite communications systems in comparison to other types of interconnected LAN systems are explained. Challenges for satellite systems in providing Inter-LAN connections capable of QoS support for real-time and non-real-time traffic are discussed. Necessity of a special MAC protocol capable of providing QoS guarantees for bursty

real-time traffic is reasoned. The advantages of CFDAMA technique using MF-TDMA format in comparison to other available medium access techniques are explained. Finally, the general configuration of Inter-LAN connections over broadband satellite systems, its components, implementing CFDAMA with MF-TDMA , translation of QoS requirements to service categories, support of two level scheduling, aggregation of requests in UTs and provision of the possibility to apply prediction techniques for future frequency-time slot requests are presented.

Chapter 3 provides the informal description of the requirements of the Inter-LAN connections over broadband satellite systems. Furthermore, the protocol stack for the proposed system configuration is introduced. The functionality of each layer is described. Functionality of Satellite Medium Access Control (SAT-MAC) and SAT-MAC Convergence sub-layers are explained in details. A block structure for SAT-MAC Convergence and SAT-MAC sub-layer to show the building components of these two sub-layers are provided. The end of this chapter is dedicated to MAC frames required to handle the negotiations between MCS and UT. Using Message Sequence Chart [11] (MSC) the negotiation between UT and MCS for initialization and dynamic service are presented.

Chapter 4 gives a brief introduction to Formal Description Techniques (FDTs) [7]. Using SDL language, the formal descriptions of SAT-MAC and SAT-MAC Convergence sub-layers along with model assumptions are provided. The interaction between SAT-MAC Convergence sub-layer and RSVP are discussed. Building blocks, processes, their functionality, behavior and related messages are described. For better clarification, MSC will present the interactions of the processes through exchanging the messages.

Verification and Validation results against general and specific properties for each sub-layer are presented.

Finally, in Chapter 5, the conclusions and future works are discussed.

Appendix A provides an introduction to RSVP and Appendix B briefly describes SDL.

Chapter 2

Inter-LAN Connections Over Broadband Satellite Systems

Interactive and secure connections among geographically separated LANs are to provide real-time and non-real-time services are one of the important issues for a big business. A satellite system can support a number of interconnected LANs to build PBNs by sharing the satellite bandwidth/capacity. This system supports real-time and non-real-time services over satellite shared bandwidth. However, satellite medium longer delay and its power and bandwidth constraints bring about challenges and design issues, which necessitates a special MAC and DCA to support the QoS requirement of a bursty traffic. In this Chapter, the advantages and constraints of satellite systems to provide Inter-LAN connections, in comparison to other types of systems are explained. Challenges stemmed from these constraints in providing multimedia services with QoS support and necessity of a special MAC protocol are described. Finally, the system configuration and our main approaches in devising such a MAC protocol are provided.

2.1 Wide Area Networks: Demands and Constraints

To provide interconnections among remote LANs, various WAN configurations and protocol were devised. WAN uses dedicated or switched connections to solve the problem of connecting a LAN to a distant workstation or another remote LAN when the

distances exceed cable media specifications or when physical cable connections are not possible.

When a WAN connects a company's branch offices and divisions, it becomes a PBN. A private network requires its own communication equipment. It may also use its own carrier services, or it may lease them from public networks or other private networks that have built their own communication lines. Although the private network is expensive, in companies where tight security and control over data traffic are paramount, private networks are the only guarantee of a high level of service.

The nature of a WAN is determined primarily by transmission medium, topology and MAC protocol. In design of a WAN, all these three factors affect each other. To implement WANs, the following transmission media can be used:

i. The public switched telephone network (PSTN)

ii. High speed, high bandwidth dedicated leased circuits

iii. High speed fiber-optic cable

iv. Microwave transmission links

v. Satellite links

vi. Wireless radiated media (radio frequencies)

Most often, WANs use high-speed leased telephone circuits (such as T1) to provide the links between different network sites.

A typical WAN and the equipment required for WAN connections are shown in Figure 2.1. A router sends traffic addressed to a remote location from the local network over the wide area connection to the remote destination. The router is connected either to an analog line or a digital line. Routers are connected to analog lines via modems or to digital lines via Channel Service Unit/Data Service Units (CSU/DSUs). The type of carrier service determines the exact type of equipment the WAN will need to function.

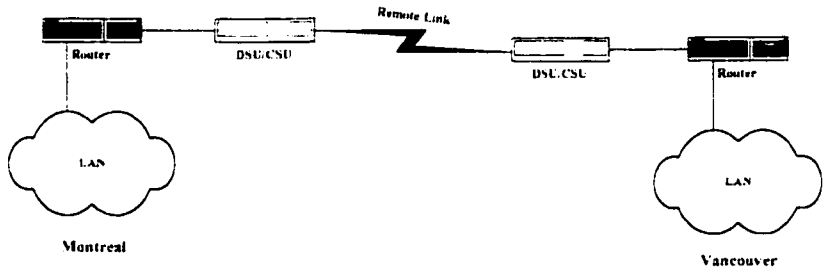

Figure 2.1 A Wide Area Network

WANs can include either dedicated or switched lines. A dedicated line service, shown in Figure 2.2, is a permanent connection between two points that is usually leased on a monthly basis. A switched line service, shown in Figure 2.3, does not require permanent connections between two points. Instead, it lets users establish temporary connections among multiple points that last only for the duration of the data transmission.

Although wired WANs provide reliable Inter-LAN connections, they suffer from a number of deficiencies:

i. Wired WANs encounter major reachability problems to remote and inaccessible areas. In some cases this problem causes the wired WAN to be completely out of consideration.

ii. In regions with infrastructure defficiencies wired WAN is not cost effective to be implemented.

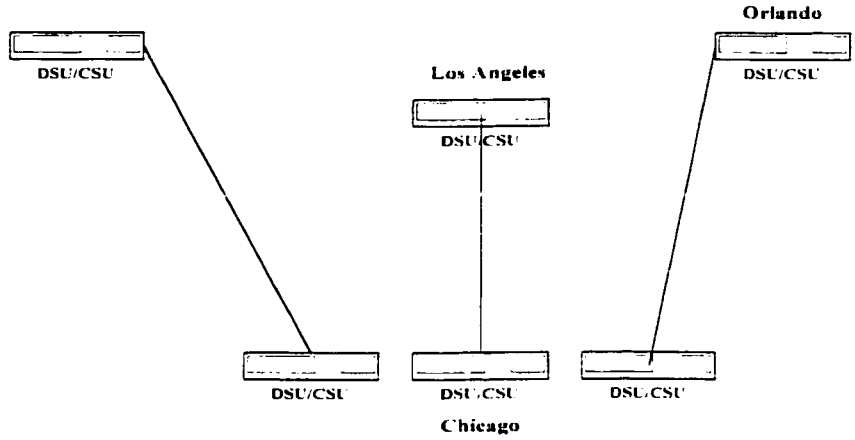

Figure 2.2 A Dedicated Line Service

iii. Wired WANs consist of several switching and routing devices in the way from a source to destination. This type of configuration and components can cause potential problems in controlling and maintaining privacy.

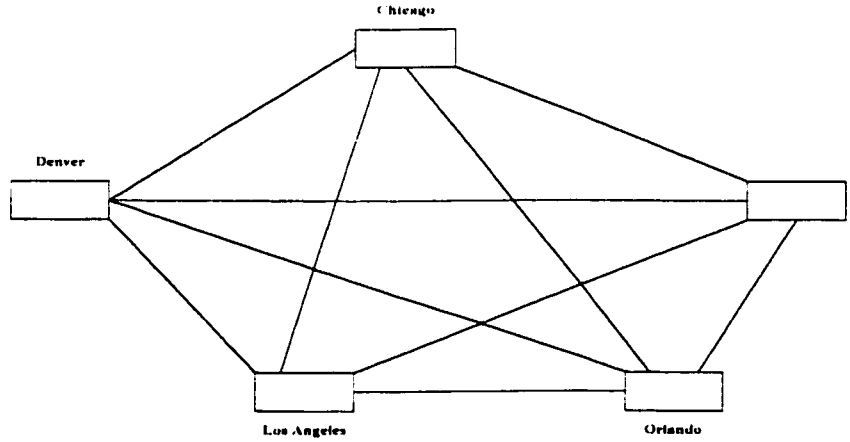

Figure 2.3 A Switched Line Service

iv. WANs may experience sequencing problems over large distances. This

necessitates special schemes in order to solve these problems.

2.2 Satellite Systems: Advantages and Constraints

The increasing interest in broadband satellite communications technology has positioned satellite networks to play an indispensable role in the global information infrastructure. Satellite communications networks can be an integral part of the newly emerging national and global information infrastructures provided the issues of interoperability of satellite and terrestrial networks are resolved. The rapidly evolving information infrastructure will play a critical role in realizing the "global village" concept of the world. The interest in extending packet switched terrestrial networks to include satellite communication links brings many potential benefits and presents some system design challenges. Satellites can be attractive alternatives to conventional wired networks for the following reasons:

i. Satellites can provide wide geographic coverage to remote and inaccessible areas not connected to the terrestrial fabric. This enables users to connect to terrestrial WANs from any location on the globe and allows connections where wired networks are not possible or economically infeasible. Satellite networks will be invaluable in providing ubiquitous connectivity for disaster recovery applications.

ii. The inherent broadcasting capability of satellites facilitates cost-effective broadcast/multi-point communications and services.

iii. Satellite communications networks have a very flexible bandwidth on demand capability. This facilitates the implementing of different DCA techniques.

iv. Global connectivity over single hop in satellite communication networks eliminates the switching and routing delays as well as sequencing problems experienced in wired WANs.

v. Connection over single hop enables satellite communication network to provide real private network.

However, satellite systems have several inherent constraints:

i. The large delays in Geostationary Earth Orbit (GEO) systems and delay variations in Low Earth Orbit (LEO) systems affect both real-time and non-real-time applications.

ii. Power and bandwidth constraints in satellite networks imposes some limitations in implementing these systems.

iii. The resources of satellite communications network, especially the satellite and the earth station, are expensive and typically have low redundancy.

2.3 Related Projects and Background

Activities for LAN interconnections by satellite started in the early 1980's, and currently there is still a considerable research efforts being undertaken. Reported European projects include French NADIR project, SATellite Internetworking Experiment (SATINE) [19, 20, 21, 22, 23], experiments performed by the European Space Agency (ESA) [24], Cooperative Olympus Data Experiment (CODE), CHEOPS and the European COST226 research project [25,26]. Projects for LAN interconnections undertaken in Japan consists of Multiple Access- Type Satellite Packet Network (MASPnet) [27], Satellite Digital

Communication Service (SDCS) and NEC projects [28]. North American projects include LAN Interconnection by Very Small Aperture Terminal (VSAT) [29,30] and Satellite Wide Area Network (SWAN) [31].

Considering the key features of these projects, special attention is given to the access schemes to the shared satellite medium. Here, the major problem is the long round trip delay of about half a second, which makes an efficient allocation scheme difficult to realize. Some early or simple experiments have used a fixed allocation scheme, whereas more sophisticated solutions implemented a demand assignment control scheme. In the context of the European SATINE-2 project, a special protocol hardware and software was developed, which is called FIFO Ordered Demand Assignment (FODA)-TDMA access scheme. It is intended to handle mixed digital traffic (stream and datagram) coming from a heterogeneous network environment by means of a centralized control. Nowadays more hybrid access techniques such as Fixed Bandwidth Assignment DAMA (FBA/DAMA), Random Access combined with DAMA (RA/DAMA) and CFDAMA are considered.

In the early projects of interconnections of LANs over broadband satellite system the experiments has been accomplished using non-standard LANs, but most of the recent experiments use Ethernet LANs with the TCP/IP protocol stack. Both commercial and experimental satellites were used, with different frequencies in different bands.

The overview of the Inter-LAN connections over broadband satellite systems shows that the most important challenge topics in this field is to provide a satellite medium access control scheme capable of QoS support for various data traffic while efficiently utilizing the satellite resources. The interaction of these schemes with upper layers necessitates a

convergence sub-layer which is responsible to translate and map the upper layer constructs to the MAC construct. In next section, these challenges are discussed.

2.4 Challenges for Satellite Systems in Providing Inter-LAN Services

As we move further into the age of multimedia communications, demands for real-time services such as videoconferencing, multimedia and non-real-time services such as FTP are increased. Multimedia traffic has a bursty characteristic and various types of multimedia services require different QoS. Furthermore, many currently developed real-time applications are extremely delay sensitive and require QoS guarantee. Considering the satellite longer delay and its power and bandwidth constraints, provision of such Inter-LAN services with QoS guarantees necessitates a suitable and special MAC and DCA for efficient utilization of the satellite limited and shared resources.

MAC protocols are designed to enable communicating hosts to regulate the movement of their packets and manage network bandwidth in order to utilize the network resources as efficiently as possible. Different operating environments and user requirements have led to a variety of MAC protocols. Part of the explanation for having many different MAC protocols is that protocol, which are suitable for some applications, often do not meet the requirements for other applications. All commonly used high-level protocols such as File Transport Protocol (FTP), Hypertext Transfer Protocol (HTTP), Network Video (NV) protocol for video conferencing, Trivial File Transfer Protocol (TFTP), Transmission Control Protocol/Internet Protocol (TCP/IP), and Asynchronous Transfer mode (ATM) use one or more low-level MAC protocols .

The space environment possesses some major constraints that eliminate a large number of some classes of MAC protocols from consideration. First, the performance impact of the long propagation delay on multiple access channel limits the applicability of some classes of MAC protocols, such as some classes of protocols proposed for LANs and WANs. Second, because of the difference in propagation delay in satellite and terrestrial links, the impact on any previously calculated performance of a MAC protocol could be significant; hence, these protocols need to be reevaluated for satellite communications. Third, physical changes to the controllers in space are limited if not impossible, and this necessitates a simple control mechanism for the protocols under consideration. Finally, to provide fault tolerance and network survivability, a MAC protocol for satellite communications is expected to easily accommodate topological changes and network reconfigurability in case of failures by adding, deleting, activating or deactivating a station from the network [14].

Fundamental architectural objectives in the design of MAC protocols for satellite communications are high channel throughput, low transmission delay, channel stability, protocol scalability, channel reconfigurability, and low complexity of the control algorithm.

The Inter-LAN services over broadband satellite systems should satisfy the different end-to-end QoS requirements and efficiently utilize satellite resources. The proposed MAC protocol for this configuration must maximize the utility of the satellite bandwidth in serving highly bursty traffic by implementing DCA and traffic control strategies.

In shared channel system, many medium access techniques have been introduced. MAC protocols for satellite communication can be classified based on their functionality with respect to the static or dynamic nature of the channel, the centralized or distributed control mechanism for channel assignments and the adaptive behavior of the control algorithm. The following classification is also described in [3]. These techniques vary from random access to fixed bandwidth assignment with different capability of QoS guarantees:

i. Random access techniques

ii. Fixed assignment techniques

iii. Demand assignment techniques

iv. Hybrid random access and reservation techniques

A concise explanation and comparison of these techniques are provided.

i. Random Access Techniques

Random access techniques such as Pure ALOHA, Selective-Reject ALOHA, SLotted ALOHA, Carrier Sense Multiple Access (CSMA) and CSMA with Collision Detection (CSMA/CD) allow users to transmit data in an uncoordinated manner. In this scheme, all the time slots are made available to all earth terminals. Whenever traffic arrives, the terminal transmits its traffic over one or more randomly selected time slots without making any request. If collision occurs, retransmission is required. At low traffic load, collision and hence retransmission are negligible. Successfully transmitted data packets

experience the shortest delay, equivalent to one round-trip propagation delay. However, at medium and high loads, the collision rate becomes unacceptably high. It is well known that the throughput of a random-access scheme is remarkably low. Furthermore, possible retransmission eliminates the possibility to support real-time services using random-access techniques. It must also be noted that random-access schemes require reliable collision detection, which is difficult to implement in a multi-beam satellite environment.

Therefore, this scheme can give no QoS guarantees because collision-free bandwidth reservations are not possible. They do however have benefits in terms of ease of implementation. Since there is no central control, signaling for channel access and algorithmic processing overhead is not incurred and the set-up phase is eliminated [15].

ii. Fixed-Assignment Techniques

In fixed-assignment multiple access techniques, the allocation of channel bandwidth to a station is a static assignment and is independent of station's activities. In this case there is no MAC, except at connection set-up.

Fixed-assignment is done by partitioning the bandwidth space into slots, each assigned to a station. Channel assignment is tightly controlled and is not adaptive to traffic changes. Fixed-assignment multiple access can be classified as orthogonal fixed-assignment techniques such as Time Division Multiple Access (TDMA) and Frequency Division Multiple Access (FDMA), or quasi-orthogonal fixed-assignment techniques such as Code-Division Multiple Access (CDMA). Fixed-assignment techniques are the most effective techniques for satellite networks composed of a small number of stations (less than ten) with stable and predictable traffic patterns.

With fixed bandwidth assignment, QoS guarantees can be delivered, but at the expense of network capacity. If the peak rate of the source is known, then simply acquiring channel bandwidth greater than or equal to this rate will give a strong upper bound on the delays seen by packets entering the network. The problem with this technique is that for Variable Bit Rate (VBR) sources often times significant channel bandwidth goes unused. If a traffic source has a high peak-to-average ratio and if the bandwidth allocated is the peak rate, then when the source is producing traffic below this peak, valuable network bandwidth is wasted. This poor link utilization leads to low network capacity, i.e., the number of terminals that can be supported within a given amount of up-link bandwidth.

iii. Demand Assignment Techniques

Demand Assigned Multiple Access (DAMA) techniques try to address the capacity issue by using instantaneous bandwidth demands to statistically multiplex many VBR sources on one channel. DAMA schemes have been widely used in satellite communications for voice services. DAMA techniques are two-phase capacity allocation techniques. The first or *set-up (or request)* phase deals with the request/allocation of the required bandwidth between the requesting user terminal and allocating agent (in a satellite network the allocating agent resides either at the satellite or at a terrestrial master control station). This phase could be itself random access. In the second or *transfer (or data)* phase, in case of successful bandwidth reservation, actual data is transferred. In this scheme bandwidth proportional to the need of an individual terminal is allocated, subject to any shortage. In the context of TDMA systems this means that time is set aside, either in designated control slots or in a piggybacking manner, for the terminal to reserve time slots for transmission in future TDMA time frames. We should note that DAMA does not

presuppose any particular physical layer transmission format. It can be implemented using TDMA, FDMA, and CDMA. What this MAC technique seeks to do is to move any possible collisions to the request phase and assign channel resources in the data phase to those terminals with packets to send. This MAC scheme results in a system wherein the time varying bandwidth needs of any terminal can be accommodated. The only time insufficient allocated bandwidth may be when several terminals are producing traffic at close to their peak rates. When this occurs, some terminals may not get all of the channel bandwidth they requested. Because of this possibility, the QoS guarantees, in terms of delay bounds, are not strong as the fixed bandwidth allocation case.

In DAMA scheme, capacity is allocated to the UT on a *per-call* basis. Delay in traffic transfer includes the call set-up time and the actual transmission, which is equivalent to three round-trip delays. QoS is guaranteed when the allocated bandwidth is equal to the *peak* rate. For Pulse Code Modulation (PCM) voice communications, the Peak-to-Average Ratio (PAR) is 1. In other words, the transmission rate is *constant* for the entire duration of the *call* (or connection). DAMA is therefore an efficient dynamic capacity allocation for stream-type traffic with a PAR of 1 and relative long transfer time, to achieve a very high channel throughput. Unfortunately, for *bursty* multimedia services with large PAR, DAMA can provide guaranteed QoS at the expense of unacceptably inefficient resource utilization. This inefficiency limits the number of supported user-terminals. Furthermore, the long set-up time (equivalent to two round-trip delays) is not desired for transmission of short data packets since the overhead of connection set-up is far more longer than the actual packet transmission. Bursty multimedia services contain both real-time and non-real-time traffic with different QoS requirements, PAR as well as

transfer duration. Furthermore, required capacity significantly varies during a *call* or *connection.* This calls for a *more* dynamic capacity allocation during a *connection*, which can provide both required QoS and high throughput. Hybrid forms of demand assigned multiple access schemes address this issue.

iii. Hybrid Random Access and Reservation Techniques

There are many hybrid forms of demand assigned multiple access. DAMA can be combined with fixed bandwidth allocation. In Fixed Bandwidth assignment DAMA (FBA/DAMA), terminals are always guaranteed to have a fixed amount of up-link bandwidth. The remaining bandwidth in the up-link channel is assigned in a DAMA manner. The scheme seeks to exploit the fact that the bit rate of most real-time applications never drops to zero. Another hybrid of DAMA combines demand-assigned bandwidth and random access transmission (RA/DAMA). This method attempts to achieve lower end-to-end delays at the expense of light packet loss. RA/DAMA assigns channel resources in a demand assigned manner and the remaining bandwidth is accessed using random access.

Another hybrid scheme is Combined Free/Demand Assigned Multiple Access (CFDAMA) [3,6,13] has been shown to be suitable to support multimedia traffic. The CFDAMA first allocates the bandwidth to UTs according to their demands and certain predefined priorities, similarly to the DAMA. However, if the resources are still available, the CFDAMA will continue to *freely* allocate the remaining bandwidth to the UTs according to a given strategy. Since the *free*-assigned capacity can be available to the UTs at the time traffic arrives, traffic can be instantaneously transmitted without

making a reservation. Hence, the set-up phase is removed in this case and traffic is transferred at a *minimum* of one round-trip delay, similarly to the case of random access. Performance evaluation results in [6.13] show that the CFDAMA provides a high throughput, equivalent to that of DAMA, as well as short traffic transfer time at both low and medium traffic load.

Our proposed model for MAC protocol will support CFDAMA using MF-TDMA format to combine their advantages. Following Section provides the detailed explanation of the configuration and building components of the proposed system and how we strengthen the CFDAMA scheme using *prediction* and *aggregation* techniques.

2.5 Inter-LAN Connections Over Broadband Satellite Systems Configuration

Figure 2.4 shows the proposed general configuration for the Inter-LAN connections over broadband satellite systems. This configuration consists of several UTs, routers, a satellite and a ground station called MCS. This system supports mesh topology in which the LANs are interconnected through the satellite.

LANs are connected to UTs through routers. UTs are intermediate components between routers and the satellite system. They are responsible to communicate with MCS to establish a QoS contract to satisfy user requests. To accomplish this duty they are able to have access to the satellite resources. UTs use allocated up-link frequency-time slots to send their packets. On one hand, UTs are capable of communicating with users to handle

their QoS requirements, and on the other hand, they can translate these QoS requirements to service categories and slot requests that are understandable by MCS.

Satellite system (satellite and MCS) supports a number of Inter-LAN connections (PBNs) by sharing the satellite bandwidth/capacity. MCS provides the required scheduling services to enable UTs to have efficiently access to the satellite resources. Allocation of up-link frequency-time slots to UTs is performed by the scheduler located in the MCS in case of bent-pipe satellite or on board in case of OBP satellite. According to predefined strategies, the main responsibilities of MCS are policy control, connection admission control and bandwidth allocation.

In this system, down-link is a broadcast and up-link is a shared link. The satellite up-link uses MF-TDMA [3] format to support the capacity allocation in a shared bandwidth.

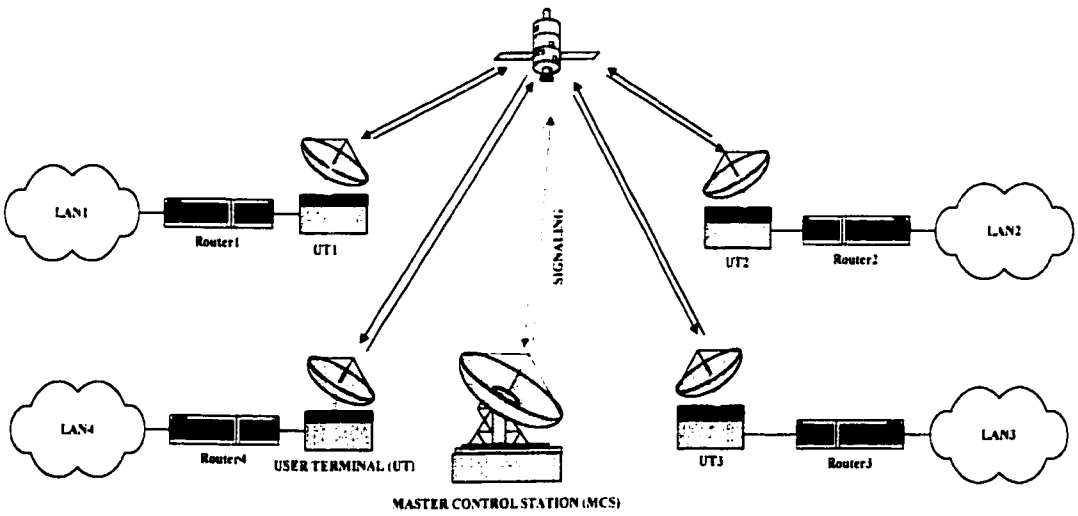

Figure 2.4 General Configuration for Inter-LAN Connections over Broadband Satellite Systems

TDMA scheme has been widely used in satellite communications to provide high bandwidth and power efficiencies, simple connectivity, great flexibility and compatibility with digital transmission [3]. TDMA scheme is quite efficient for high-capacity trunking applications in which satellite links are used to provide international connections between high-load nodes of a digital communication network. However, the high transmission bit rate in TDMA system requires large and expensive earth-terminals, which are not cost-effective for user-oriented applications.

Basically, MF-TDMA is a hybrid FDMA/TDMA scheme. The up-link bandwidth is first divided into a number of frequency slots. Each frequency slot is then time-shared by UTs. A channel in a MF-TDMA format is presented by a frequency-time slot in a two-dimensional frame as shown in Figure 2.5. The MF-TDMA scheme has a single carrier transmitter with a carrier-hopping capability to access any frequency-time slot. The transmission bit rate (and consequently the bandwidth of the frequency slot) is selected according to the required transmission capacity, size and cost of the earth-terminal [3].

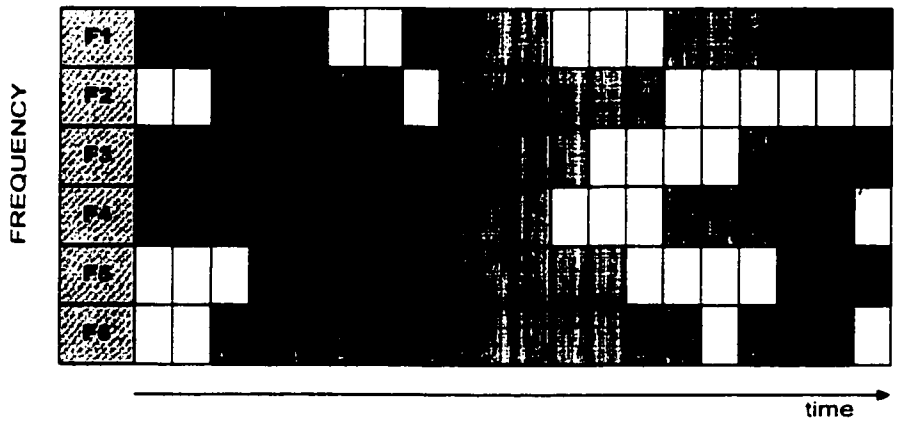

Figure 2.5 An Example of a MF-TDMA Structure

While maintaining the advantages of TDMA. MF-TDMA format uses a much lower transmission bit rate in order to reduce both size and cost of UTs.

Capacity allocation in MF-TDMA format can be presented by a frequency-time map indicating the frequency-time slots assigned to each UT. If the individual traffic load per connection is high and the interconnection patterns are static a fixed capacity allocation can be used. In this case. the frequency-time map is slowly changed. However. traffic in real-time applications has different types and is rather bursty. Each terminal demands satellite resources infrequently; but when it does. relatively high capacity and rapid response are required. In this case. capacity allocation must be more dynamic to cope with the real-time traffic demands [3]. In this proposed structure CFDAMA using MF-TDMA format is implemented to respond to such demands. In request phase UTs will request for network resources and MCS will reserve the requested bandwidth. The remaining frequency-time slots in the frame are *free*-assigned to the UTs using a given strategy by the MCS. If the UT obtaining a free frequency-time slot has a short message then it can send this message right away.

Implementing some *prediction* techniques in UTs for requesting the time slots for the incoming and stored packets in the data queues and some decision-making strategies in MCS to allocate *free* frequency-time slots will result in a situation that those packets will not experience the time needed for access. Prediction can be based on the traffic profiles and queue indicator.

The scheduler in MCS will have the responsibility of giving capacity grants to all requesting UTs. Thus, the scheduler will receive periodically the requests (both

instantaneous and predicted) of all UTs. Based on all currently received requests, the service class needs of each request, and predefined scheduling strategies the scheduler will give capacity grants to each UT. To ease the duty of MCS, we must try to distribute the processing duties as much as possible among the user terminals. For this reason, instead of sending the user terminals' requests as a set of *QoS parameters*, we can provide a mechanism to cast the requested QoS parameters in some predefined service categories and proportional slot requests. Using this mechanism, user terminals will request MCS only for a predefined service category and some calculated slot requests. The proposed model supports this mechanism. This characteristic of the model will enable the MAC protocol to adjust itself to the new possibilities in IPv6 for defining service classes other than best efforts, guaranteed and controlled-load service.

Even more, to reduce the processing load in MCS, each UT can aggregate its requests. MCS will process a bunch of requests in one message, will keep a record of it, and will allocate frequency-time slots for this aggregated request. UT itself must keep a record of each request and according to some predefined rescheduling techniques, must reschedule locally the received frequency-time slots. This strategy will require a collaboration between UT and MCS in scheduling.

Chapter 3

Proposed Protocol Structures for Inter-LAN Services Over Broadband Satellite Systems

The global requirement of Inter-LAN connections over broadband satellite systems is to provide a transparent bi-directional transfer of IP and multimedia traffic between two UTs while sustaining the required end-to-end QoS over the satellite segment.

Considering the detailed decomposition of this general requirement, in this chapter a protocol stack for the proposed Inter-LAN connections over broadband satellite systems are introduced. The functionality of each layer with a special emphasis on SAT-MAC and SAT-MAC Convergence sub-layers is explained. Using these details a block structure consisting of required components for SAT-MAC and SAT-MAC Convergence is presented. Finally, MAC frames required to communicate between MCS and UT are provided.

3.1 Service Requirements

Inter-LAN connections over broadband satellite systems necessitates the following services from the MAC layer:

i. Interaction With a Resource Reservation Protocol to Acquire the Requested QoS:

- MAC layer must interact with a resource reservation protocol such as RSVP in order to receive the requested QoS parameters and establish a QoS related contract between UT and MCS.

- MAC layer must provide all the necessary information to the RSVP messages. It must be able to respond to all required RSVP-MAC interface messages explained in RFC 1205 [2].

- RSVP requests are issued using QoS parameters. MAC layer must be able to respond to these requests as per QoS parameters.

ii. A Signaling Function for Dynamical Establishment of QoS-enabled Service Categories:

- MAC layer must be able to handle the negotiation between UT and MCS in order to provide the requested dynamic services.

- Dynamic service negotiation between UT and MCS must be done through dynamic service related signaling.

- Dynamic service messages may carry the service categories and an amount of time-frequency slots corresponding to the requested QoS.

- Mapping the requested QoS parameters to service categories requires the translation of QoS parameters to service categories.

- UT may aggregate the requested services and negotiate for a bunch of requests in one dynamic service message.

- UT may predict the future required time-frequency slots and may send requests for these predicted time-frequency slots in addition to the actual aggregated requests.

iii. Policy Control:

- MAC layer must be able to apply the policy control on each individual UT. Policy control must be agreed between UT and MCS in the registration phase. Only the UTs that follow the policies agreed in the registration phase will be allowed to use the shared medium.

- Policy control must be applied on each dynamic service request.

- If a request fails in policy control check, it must be rejected.

- MAC layer in MCS must always have the updated information about the amount of resources that each individual UT uses.

iv. Admission Control:

- MAC layer must be able to apply admission control on each individual request if that request succeeds policy control check.

- Admission control may accept, reject or modify the requested amount of time-frequency slots for a specific service category according to the availability of the resources.

v. Scheduling Strategies:

- MAC layer in MCS must use macro scheduling strategies in order to allocate efficiently the requested resources and its admission control must use updated scheduling information in order to respond the requests.

- MAC layer in UT may use micro scheduling in order to distribute the dedicated time-frequency slots to the requests. If aggregation and prediction are implemented micro scheduling in UTs is required.

vi. **Power-on Initialization, Registration, Synchronizing, Scanning, Up-link Selection, Ranging.**

- Power-on initialization, registration, synchronizing, scanning, up-link selection, and ranging are required to make and maintain a UT as a part of the network. This requires negotiations between UTs and MCS.

- Negotiations between UT and MCS to accomplish these purposes must be done through MAC frames containing the required information.

- A message called Upper Channel Descriptor must be periodically transmitted by the MCS to define the characteristics of possible UTs up-link channels.

- A synchronizing message must be periodically transmitted by MCS to establish the required synchronization.

- A MAP message must be regularly sent by the MCS in order to organize and regulate bandwidth allocations on up-link.

- Ranging related messages must be exchanged between UT and MCS, at initialization and periodically, to determine network delay and request for time, power and frequency adjustment.

- Registration related messages must be exchanged between UT and MCS during the initialization phase in order to register UT to the satellite network.

- Data traffic must and can not be transmitted prior to a successful registration of a UT, except for TFTP and DHCP commands.

- Collision of initial ranging packets requires contention resolution.

- A change in UCD parameters that does not suit the UTs capabilities requires re-initialization.

- Generally a UT must stay registered if it attains synchronizing, UCD and ranging requirements.

- Channel changing procedures may require re-ranging and even sometimes re-initialization depending on MCS instructions.

3.2 Protocol Architecture

In this Section the overall protocol stack in UT and MCS to support the requirements of the Inter-LAN connections over broadband satellite systems are presented. Furthermore, a detailed explanation of the functionality of the MAC sub-layers in UT and MCS are provided.

3.2.1 Protocol Stack

Figure 3.1 depicts the overall protocol stack structure for Inter-LAN connections over

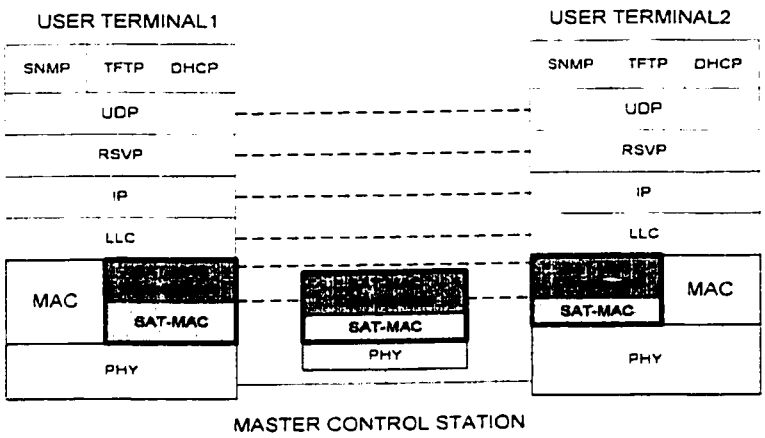

Figure 3.1 Protocol Architecture for Inter-LAN Connections over Broadband Satellite Systems

broadband satellite system.

Each UT includes the TCP/IP suite of protocols. Simple Network and Management Protocol (SNMP) is responsible for network management. Dynamic Host Configuration Protocol (DHCP) and TFTP are used during UTs initialization and registration.

IPv6 provides routing services. The proposed model for MAC protocol are flexible and extendable to support the IPv6 new features of eight level transmit and delivery priority and flow label characteristics.

RSVP over IPv6 is used to enable UTs to communicate with each other and MCS through MAC layer in order to set up necessary contracts to support required QoS. Many currently developed real-time applications are delay-sensitive and the best effort delivery scheme of IPv4 can be inadequate even under modest network loads. The necessity to

provide many applications with additional service classes offering enhanced QoS with regard to the bandwidth, packet queuing delay, and loss has led to the introduction of several reservation protocols. These reservation protocols are supposed to enable the senders, receivers and routers along the path to communicate with each other in order to set up the necessary network nodes' states to support the required QoS. The most important IP reservation protocols are RSVP, ST2, ST2+ and Differentiated Services. As an illustrative example, the proposed model will consider RSVP and IPv6 on top of the MAC layer to use the RSVP characteristics to provide the end-to-end QoS. RSVP has been designed to operate with current and future unicast and multicast routing protocols such as IPv6. RSVP is not a routing protocol: it is used to reserve resources along the route whichever underlying routing protocol is in place. RSVP is only concerned with the QoS of those packets that are forwarded in accordance with routing. RSVP is able to cope with the resulting routing changes and to automatically re-establish the resource reservation along the new paths as long as adequate resources are available. Furthermore, in a wide area inter-network, receivers, as well as the paths used to reach the receivers, can have very different properties from one another. RSVP has been designed to provide the ability for heterogeneous receivers to make reservations specifically tailored to their own needs. In order to accommodate efficiently large groups, dynamic membership, and heterogeneous receiver requirements, RSVP makes receivers responsible for requesting a specific QOS. The characteristic of dynamic membership in a multicast group removes the burdensome re-initiation process in the case of joining a new member or leaving an existing member. RSVP has been designed to allow end users to specify their application needs so that the aggregate resources reserved for a multicast group can more accurately

reflect the resources actually needed by that group. Finally, RSVP overhead limitation and modularity makes RSVP deployable in many contexts. RSVP is compatible with UTs MAC layer requirements of QoS support and capacity allocation. Using RSVP, UTs MAC layer is able to make reservation of network resources upon the acceptance of a call and their consumption after the generation of actual traffic.

In LLC Layer of UT, appropriate error and flow control techniques (such as go-back-N) are applied. These techniques are applied only on non-real-time packets. This layer is transparent to real-time packets.

The MAC layer of UTs consists of two sides. One side is connected to routers and hence can use standard MAC protocols. The other side, which is responsible for provision of access to satellite medium and QoS support, must use a special MAC protocol. This special MAC layer functionality is split over two sub-layers namely the *SAT-MAC Convergence* and *the SAT-MAC* sub-layers.

SAT-MAC Convergence translates upper layer QoS parameters into SAT-MAC constructs. It also maps upper layer's addresses into the corresponding SAT-MAC addresses.

SAT-MAC layer guarantees efficient data transmission over the satellite medium.

Physical layer is responsible for physical transmission of bit stream bursts over satellite channels and has the MF-TDMA format as explained in Section 2.4.

MCS consists of the TCP/IP suit protocols, SAT-MAC Convergence, SAT-MAC and Physical layer. The TCP/IP suit protocols are required in MCS for registering the UTs,

downloading registration file and management. For simplicity and to emphasize on *SAT-MAC* and *SAT-MAC convergence*, these protocols are not shown in MCS in the Figure 3.1.

3.2.2 Functionality of SAT-MAC Convergence and SAT-MAC Sub-layers

In this thesis, we focus on MAC layer responsible for provision of access to satellite medium and QoS support.

Functions of SAT-MAC convergence in UTs and MCS are not the same. However, they negotiate to provide the requested QoS and dynamic service establishment. SAT-MAC Convergence specific functions in UTs are:

i. SAT-MAC Convergence must be capable of interaction with RSVP through an interface. It provides information to RSVP about the actual available network resources (i.e., bandwidth) and negotiates with MCS SAT-MAC Convergence in order to reserve network resources for requested QoS.

ii. The SAT-MAC Convergence sub-layer must support categorized dynamic service negotiations between UT and MCS to dynamically establish, modify, and delete connections according to the UT current services demands. This necessitates the translation of QoS requirements to category services and frequency-time slot requests to ease the processing efforts in MCS.

iii. SAT-MAC Convergence may support aggregation of requests to avoid sending the requests per connection in order to simplify the processing of the requests in MCS.

iv. SAT-MAC Convergence must distribute the granted frequency-time slots for aggregated requests to respond to individual requests.

SAT-MAC *Convergence* specific functions in MCS are:

i. *Policy control* to determine whether the user has administrative permission to make the reservation.

ii. *Connection admission control* to determine whether the satellite has sufficient available resources to supply the requested QOS.

Although *SAT-MAC* in UTs and MCS are different they cooperate to accomplish the following functions:

i. SAT-MAC is responsible for overall management of the MAC layer.

ii. Power-on initialization, synchronizing, scanning, up-link selection, ranging, registration, SAT-MAC PDU handling and data transmission are accomplished by SAT-MAC sub-layer.

iii. SAT-MAC is responsible for actual data transmissions in which efficient use of bandwidth allocated by the MCS is achieved through piggybacking, fragmentation, concatenation or compression, ...etc of data packets.

iv. SAT-MAC in UTs uses prediction techniques to ask for additional capacity in anticipation of arriving traffic.

v. Macro scheduling to guarantee the efficient data transmission over the satellite medium is the most important duty of the MCS SAT-MAC. To accomplish this

responsibility CFDAMA technique along with MF-TDMA format is used. Applying prediction techniques in UTs and decision making schemes in MCS help to provide an efficient macro scheduling in MCS.

3.3 Structure of MAC Layer

This Section proposes components to the MAC layer of the UT and MCS that will satisfy the requirements of the MAC layer for Inter-LAN connections over broadband satellite system. System description for the MAC layer of UT and MCS are separately provided.

3.3.1 Structure of MAC Layer in UT

Figure 3.2 depicts a block structure and highlights the required components for the MAC layer of the UT [12]. This structure consists of:

i. **Interface to Resource Reservation Protocol:** This will provide the interface between MAC layer and the resource reservation protocol.

ii. **QoS to Service Category Translator:** It has the responsibility of translating requested upper layer QoS parameters into service categories containing local parameters recognized by local Policy and Admission Control.

iii. **Local Policy and Admission Control:** It defines the UT policy and handles admission of all connection requests from upper layer. Translated upper layer requests are handled as certain service categories. If the local resources completely exhausted or the connection violates the terminal capabilities Local Policy and Admission process might immediately refuse a connection request locally.

Admission mechanisms might also aggregate requests over an existing contract in case it is under utilized. Otherwise, the admission must negotiate with the MCS. To do this negotiation Dynamic Service-related requests for the corresponding category are issued. These requests accompanied by traffic parameters related to the specific category will be transmitted over the MAC to MCS.

Local Policy and Admission process uses updated information received from Queue and Capacity Manager (QCM) to make decision. The Local policy and Admission process also update the QCM with the current admitted resources status to facilitate the prediction process. It will also give information about the various connections to the traffic selector to help it monitor complying traffic.

iv. **Dynamic Service:** The Dynamic Service process handles the negotiations between UT and MCS to provide a contract for requested service categories.

v. **Traffic Selector:** The traffic selector separates the different types of the user traffic and forwards them into the data queues of the corresponding category. By interleaving over a time frame traffic from different local connections before forwarding them into the queues it prevents queues from being monopolized by a single connection.

vi. **Data Queues:** For each service category there is a data queue. All connections of the same category transfer their data packets to the corresponding data queues. Data queues send instantaneous status updates to prediction process to help in decision making.

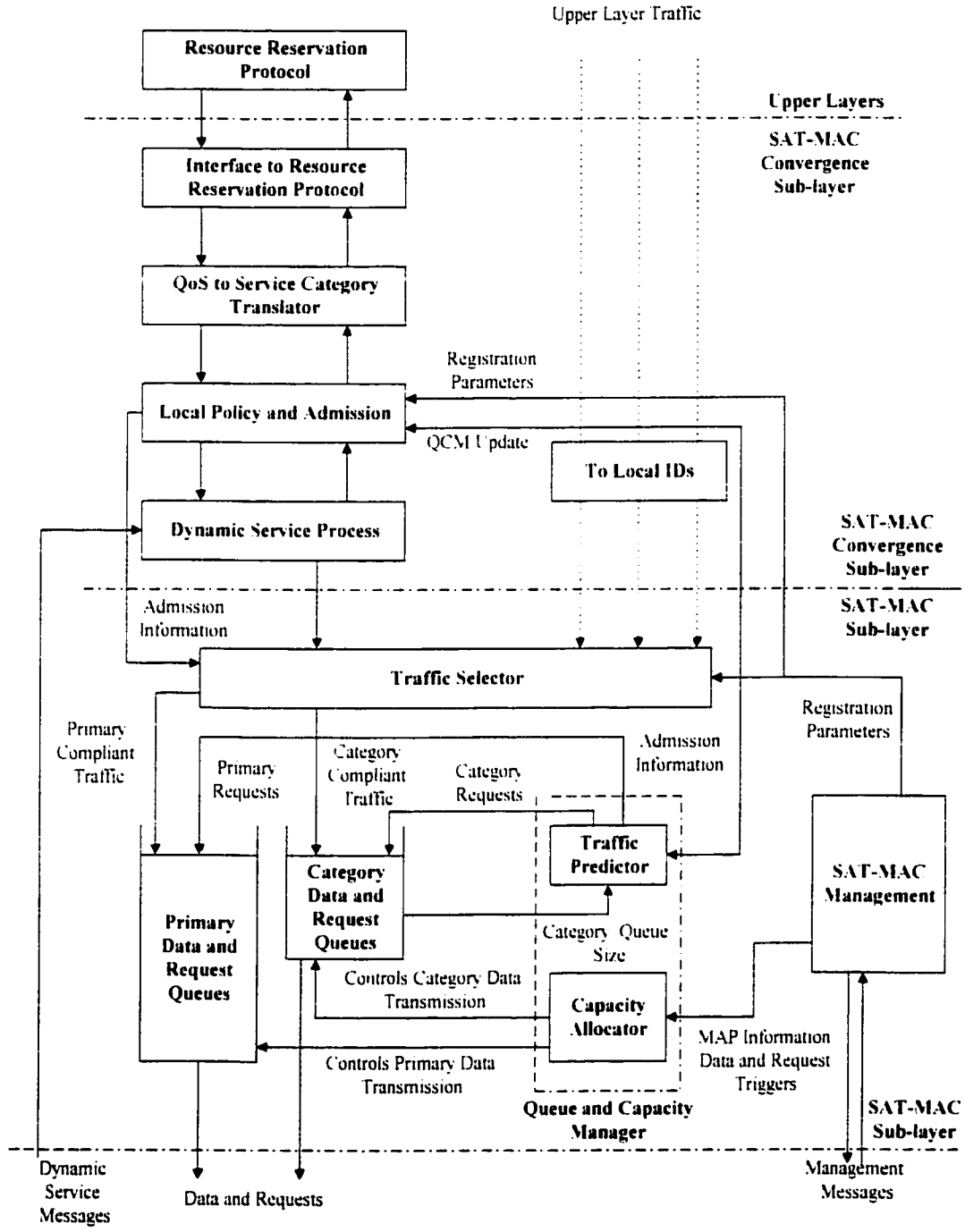

Figure 3.2 Components of the UT *SAT-MAC Convergence* and *SAT-MAC*

vii. **Queue and Capacity Manager:** It has the responsibility of managing and distributing

granted time-frequency slot from the MCS. Based on queue status and admission information the QCM predicts and request the required bandwidth and allocate granted allocations.

The queue and capacity manager consists of:

- **Traffic Predictor:** The traffic predictor has the responsibility of requesting the necessary capacity from the MCS to send the packets stored in the data queues. The traffic predictor periodically receives updates from the data queues. The predictor also maintains a record of and receives updates from Local Policy and Admission process on the traffic parameters of currently accepted connections. All provided information will be used in predicting the anticipated number of data requests after two round trips.

- **Capacity Allocator:** It receives description of the frequency-time-slot grants from MAC management and updates the predictor by the allocated grants. It also manages the queues and their timing. In case of multiple queues, the allocator may decide to borrow allocations of one category to the other (real-time over non-real-time for example). Excess bandwidth if any will also be shuffled based on local algorithms as well as instantaneous sizes of the different queues.

viii. **SAT-MAC Management:** SAT-MAC management is the center of the MAC layer structure. It is responsible to control the main functions of the SAT-MAC, i.e., power-on initialization, registration, synchronizing, scanning, up-link selection, ranging and handling the MAP message.

3.3.2 Structure of MAC layer in MCS

Figure 3.3 depicts a block structure that highlights the required components for the MAC layer of the MCS. This structure consists of:

i. **Policy Control:** It carries policy information acquired during registration. If a dynamic service request violates the policy control information, it will be immediately rejected. Otherwise, the request will be passed to the Admission Control. MCS can itself initiate a new dynamic service request to UT to add, change or delete a service category.

ii. **Admission Control:** It decides to accept, reject or accept with modification the requests according to the current network status and regular updates received from Scheduler. It also updates the scheduler with the latest admission information.

iii. **Scheduler:** It has the responsibility of allocating frequency-time slot grants to all requesting UTs. UTs requests represent their overall requirements per category. The scheduler regularly receives admission information and uses it to monitor complying stations. It is also frequently polled from MAC management to allocate management opportunities to specific UTs. Scheduling instructions are forwarded to the MAC management to be transmitted as MAP messages.

iv. **SAT-MAC Management:** It is responsible to control the main functions of the SAT-MAC, i.e., power-on initialization, registration, synchronizing, scanning, up-link selection, ranging and handling the MAP message.

v. **Output Queue:** It stores all the dynamic service responses to be forwarded to UTs.

vi. **Dynamic Service:** It has the same responsibility of its counterpart in UT.

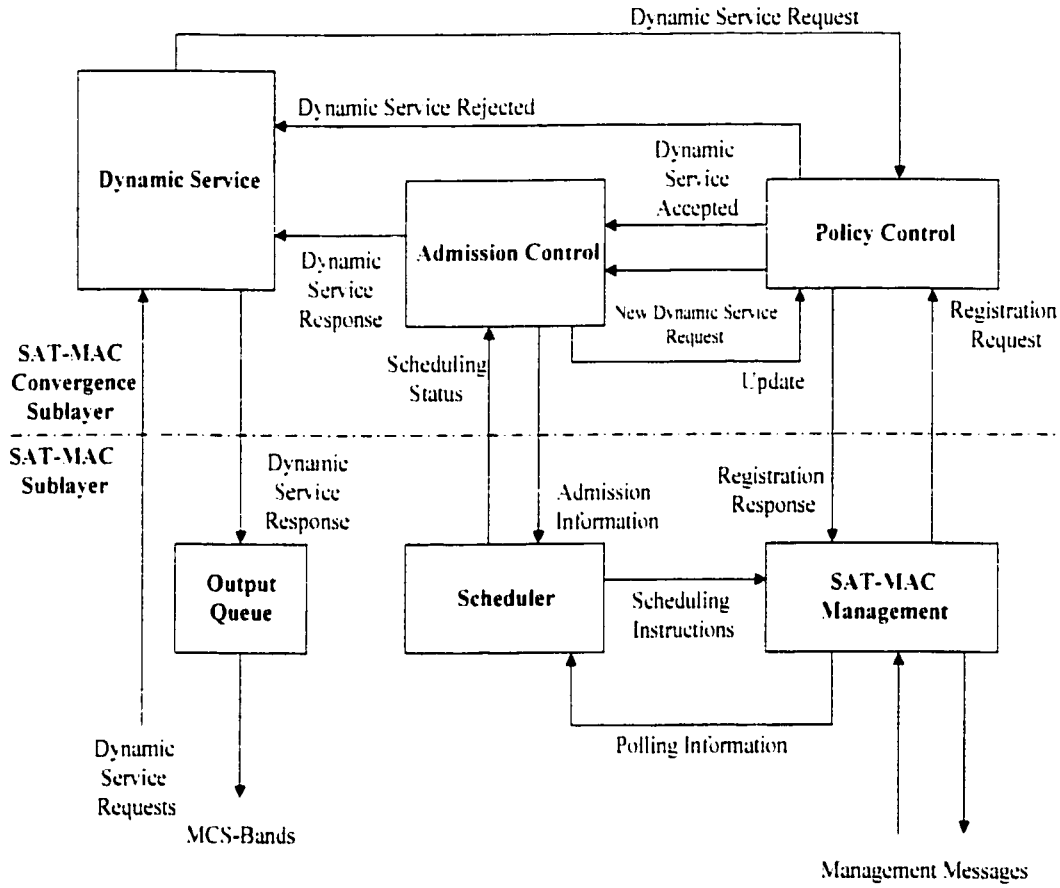

Figure 3.3 Components of the MCS *SAT-MAC Convergence* and *SAT-MAC*

3.4 SAT-MAC Frames

UT and MCS must negotiate to do the explained duties. *SAT-MAC* frames contain the information exchanged between *SAT-MAC* entities. *SAT-MAC* frames may be one of the following types:

- **Data Frames:** Data frames may carry an IP packet or a variable number of ATM cells.

- ***SAT-MAC* Specific Frames:** Specific frames may be Request frames used for reservation requests or MAC management frames, which carry the MAC management messages.

3.4.1 SAT-MAC Management Messages

SAT-MAC management is the heart of the MAC protocol. It controls the SAT-MAC main functions such as power-on initialization, synchronizing, scanning, up-link selection, ranging, registration, SAT-MAC Protocol Data Unit (PDU) handling and data transmission. Management messages must be encapsulated in LLC information frames, which in turn are encapsulated within SAT-MAC framing. The reason is that SAT-MAC management messages naturally carry sensitive control data and therefore cannot function properly within erroneous conditions. The different management messages chosen coincide with DOCSIS1.1 [10] and are described as follows:

i. **SYNC (Synchronization):** A *SYNC* message is transmitted periodically by the MCS. UTs use it to establish MAC sub-layer timing. The MCS broadcasts inside this message the current state at transmission of a clock inside the MCS to offer a global timing reference.

ii. **UCD (Upper Channel Descriptor):** A *UCD* message is transmitted periodically by the MCS. It defines the characteristics of possible UTs up-link channels. Typical descriptors would include frequency-time slot size, symbol rate, modulation type and other physical parameters.

iii. **MAP (Allocation Map):** A *MAP* message is regularly sent by the MCS in order to organize and regulate bandwidth allocations on up-link. Based on scheduling decisions, it describes time as a variable number of variable-length transmit opportunities over a band of frequencies. Opportunities include request, data, initial maintenance (ranging), station maintenance (ranging) registration and data pending information in case the scheduler is unable to satisfy a data grant in that *MAP*.

iv. **RNG_REQ (Ranging Request):** *RNG_REQ* messages are transmitted by UTs at initialization and periodically on request from MCS to determine network delay and request for time, power and frequency adjustments.

v. **RNG_RSP (Ranging Response):** A *RNG_RSP* message must be transmitted by the MCS in response to a received *RNG_REQ*. Typical carried parameters will include timing and power adjustment as well as fine frequency tuning.

vi. **REG_ REQ (Registration Request):** A *REG_REQ* message is transmitted by a UT during the initialization phase. It carries UT configuration parameters as downloaded from the MCS. These parameters are used by the MCS in registering the UT to the satellite network.

vii. **REG_RSP (Registration Response):** A *REG_RSP* must be transmitted by the MCS in response to *REG_REQ*. The MCS may modify some of the *REG_REQ* parameters in the response.

viii. **REG_ACK (Registration Acknowledgement):** A *REG_ACK* message is transmitted by the UT in response to a *REG_RSP* from the MCS. It confirms the

reception and acceptance of the UT to the QoS parameters as reported by the MCS in the *REG_RSP*.

3.4.2 An Illustrative Example for UT Initialization

The UT power-on initialization will be accomplished in the following stages as shown in Figure 3.4.

MCS periodically sends *SYNC* messages. UT can not sense these messages before getting power-on.

i. *Scanning for appropriate down-link channel:* UT gets *POWER_ON* signal and starts for scanning for an appropriate down-link channel.

ii. *Synchronization*: UT receives two *SYNC* management messages and is synchronized with MCS.

iii. *Obtaining Upper Channel Descriptor Parameters*: After synchronization, UT must wait for a *UCD* message from the MCS in order to retrieve a set of transmission parameters for a possible up-link channel. These messages are transmitted periodically from the MCS for all available up-link channels and are addressed to the SAT-MAC broadcast address. The UT must determine whether it can use the up-link channel from the description parameters. UT must collect all *UCD*s and check them to find a proper up-link channel. After a suitable timeout period if no channel can be found, then UT must continue the scanning to find another down-link channel.

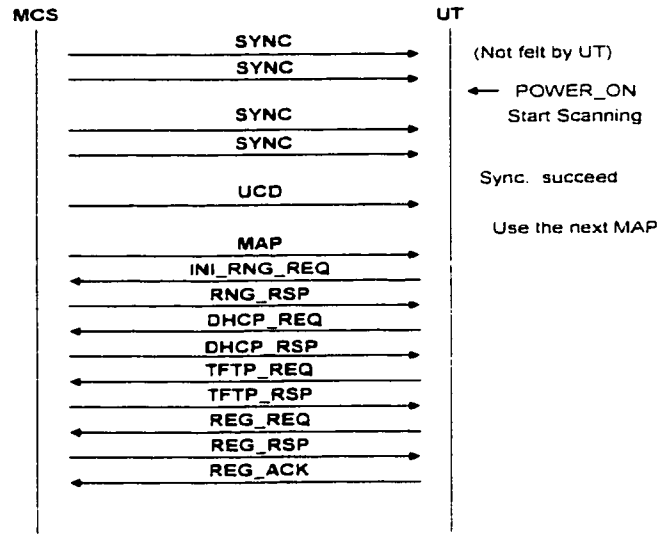

Figure 3.4 Successful UT Initialization

iv. *Sending RNG_REQ* (Initial Maintenance): Using the initial ranging opportunity, provisioned in the MAP, UT sends a *RNG_REQ* to the MCS. In this message it uses the Broadcast Service Flow ID.

v. *Receiving RNG_RSP:* Upon receiving the *RNG_REQ*, MCS sends a *RNG_RSP* using a temporary ID.

vi. *Sending RNG_REQ* (Station Maintenance): UT receives a MAP and sends a *RNG_REQ* using the temporary ID.

vii. *Ranging Accomplishment*: MCS sends a *RNG_RSP* indicating the successful ranging. It should be noted that UT must perform initial ranging at least once per getting a power-on message. If the initial ranging is not successful, then the next channel ID is selected, and the procedure restarted from *UCD* extraction. When there are no more

channel IDs to try, then the UT must continue scanning to find another down-link channel.

viii. *IP connectivity:* After finishing the ranging, UT must obtain IP connectivity using *DHCP*. *DHCP* allows UT to obtain an IP address as well as the name of the file to be downloaded for configuration parameters (registration parameters). In this step MCS has to recognize a UT via its MAC address.

ix. *Downloading the required parameter file*: Using TFTP UT downloads the required parameter file from MCS.

x. *Registration*: UT must be authorized to forward traffic into the network once it is initialized and configured. The UT is authorized to forward traffic into the network via registration. To register with a MCS, the UT must forward its configured class of service and any other operational parameters in the configuration file to the MCS as part of a Registration Request. To do this UT waits for a MAP and sends the *REG_REQ*. It waits for a Registration Response (*REG_RSP*) from MCS. UT will acknowledge the *REG_RSP* by sending *REG_ACK*. At this stage UT is authorized to forward traffic to the network.

3.5 SAT-MAC Convergence Dynamic Service Messages

SAT-MAC Convergence sub-layer in UT and MCS need to exchange dynamic service related messages through *SAT-MAC* to enable the dynamic service negotiation. These messages are carried trough SAT-MAC sub-layer as a data frame. UT and MCS can initiate a request for addition, change or deletion of a dynamic service. These messages are comparable to those proposed in DOCSIS 1.1. However, in this architecture, these

messages do not carry the *individual* QoS parameters requested by RSVP for each connection. Instead, they carry translated QoS parameters into the service categories and the amount of frequency-time slots required for a specific service category. Dynamic service related messages are:

i. **Dynamic Service Addition Request (DSA_REQ)**: A Dynamic Service Addition Request may be initiated by a UT or MCS to create a new service category.

ii. **Dynamic Service Addition Response (DSA_RSP):** A Dynamic Service Addition Response must be generated in response to a received *DSA_REQ*.

iii. **Dynamic Service Addition Acknowledgement (DSA_ACK)**: A Dynamic Service Addition Acknowledgement must be generated in response to a received *DSA_RSP*.

iv. **Dynamic Service Change Request (DSC_REQ):** A Dynamic Service Change Request may be initiated by a UT or MCS to dynamically change the reserved resources for a service category.

v. **Dynamic Service Change Response (DSC_RSP)**: *DSC_RSP* is generated by a UT or MCS in response to the *DSC_REQ*.

vi. **Dynamic Service Change Acknowledgement (DSC_ACK)**: It is generated by a UT or MCS in response to the received *DSC_RSP*.

vii. **Dynamic Service Deletion Request (DSD_REQ):** *DSD_REQ* may be initiated by a UT or MCS to delete an existing service category.

viii. **Dynamic Service Deletion Response (DSD_RSP):** It is sent by a UT or MCS in response to the received *DSD_REQ*.

Figure 3.5, 3.6, 3.7 show the UT initiated addition, change, and deletion dynamic service. To distinguish the dynamic service request, response and acknowledgement messages issued from MCS an extra "S" is added to the beginning of the message.

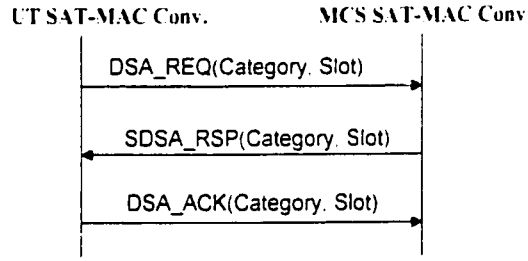

Figure 3.5 UT Initiated Dynamic Addition Request

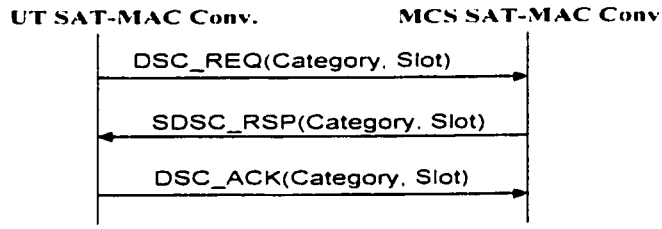

Figure 3.6 UT Initiated Dynamic Change Request

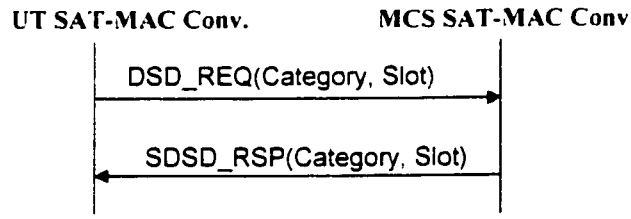

Figure 3.7 UT Initiated Dynamic Deletion Request

Chapter 4

Formal Description and Validation of Proposed MAC Structure

Over the years protocol specifications for communication systems were informal. Using informal methods in description of complex systems and standards may lead to ambiguities and undetectable errors. Though final products are released after numerous debugging iterations, still many errors were experienced through practical use.

During the last two decades, Formal Description Techniques (FDT) [4.7.11] have been successfully devised to cope with these deficiencies. Formal approaches are particularly justified for systems, such as communication systems, which are complex, concurrent, quality-critical, safety-critical, security-critical and standardized. FDTs are intended to be used in international standards, specifically in communication area, to define unambiguous and consistent standards. FDTs are used in system development, to specify the required behavior, to design, to simulate and generate an optimal system implementation and to document the provided behavior. Furthermore, formal approaches are used to verify and validate the proposed system prior to the implementation. Validation of the design and behavior of the system prior to the implementation, as it is shown in Figure 4.1, alleviates the burden of end product debugging. In traditional methods of software development the debugging of the end product needs to repeat a cycle through implementation and sometimes it leads even to modification of the design.

But in FDT we guarantee that our design is correct before starting to generate the code. FDTs are usually seen as tools to improve the quality of the end product.

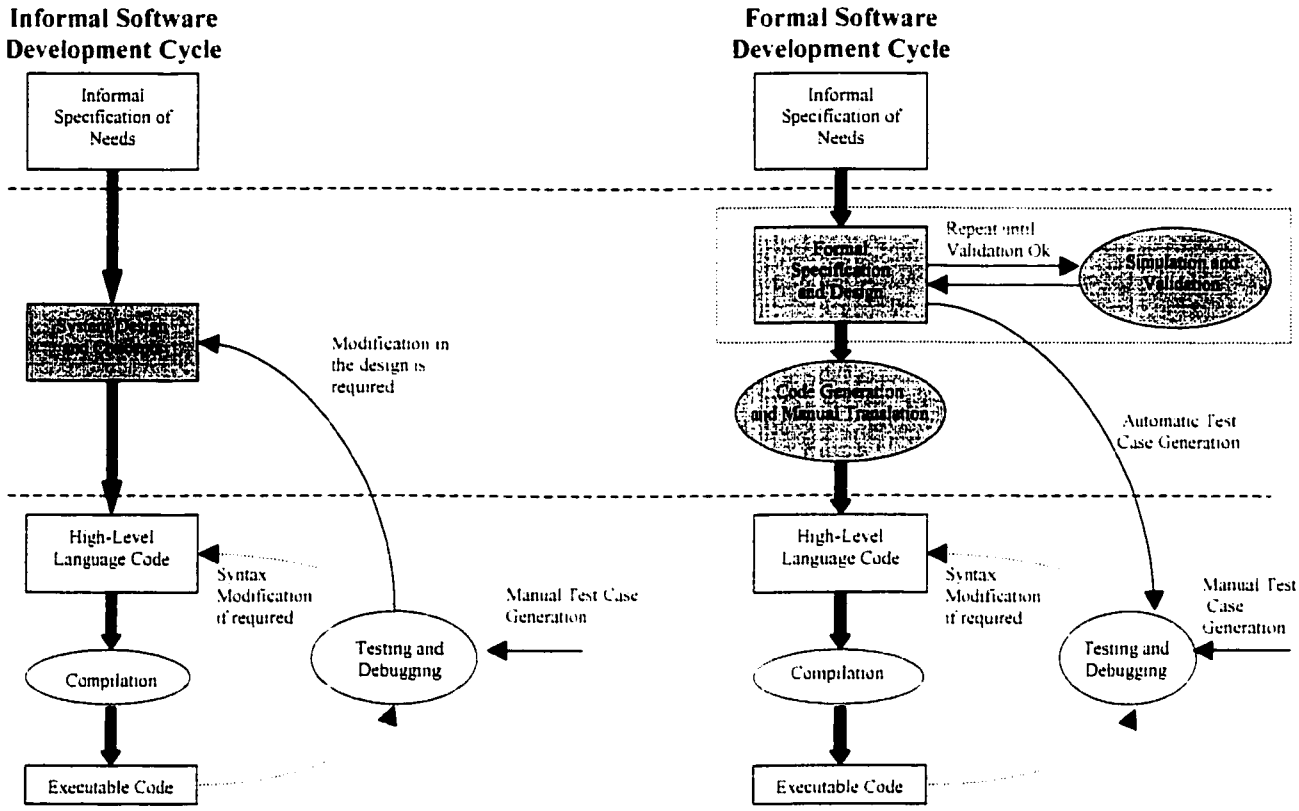

Figure 4.1 Formal Description Technique in Comparison to Traditional Methods

The most important languages to apply FDT are ESTELLE (Extended Finite State Machine Language), LOTOS (Language of Temporal Ordering Specification) and SDL. In this thesis we have chosen SDL as the formal language to model the proposed MAC protocol for Inter-LAN connections over satellite system. Furthermore, the devised model using ObjectGEODE tool is simulated and verified against general and specific properties.

4.1 SDL

SDL has been introduced and being maintained by ITU-T SG10. SDL is a behavioral specification language based on the extended finite-state machine model, supplemented by features for specifying data and structures. SDL provides constructs for describing structures, behaviors, interfaces and communication links. From the structural point of view, SDL describes a *system* in a hierarchical manner as a set of *blocks* communicating with each other through *channels*. *Blocks* can be decomposed in a recursive manner into lower level *blocks*.

The dynamic behavior of an SDL system is described by *processes*, which are given in the lowest block level. *SDL processes* communicate with their *environment* and with each other by messages, called *signals*. *Transitions* in *processes* from one *state* to another are triggered by the reception of *signals*. In each *process*, the reaction to triggering *signals* and the behavior of the *process* are defined. Each *process instance* has an unbound first-in-first-out *input queue* to receive incoming *signals*.

SDL is the most popular formal language in telecommunications. This is mainly due to the graphical representation, in addition to the texual one, and the availability of powerful commercial tools such ObjectGEODE, which consists of many features for creating, simulating and validation specifications, as well as features for code generation and for testing the end product. An introduction to SDL is provided in Apendix B.

4.2 Model Assumptions

In order to provide a model for the MAC layer of Inter-LAN connections over satellite

systems in SDL following strategies are taken:

i. Functionality of *SAT-MAC Convergence* and *SAT-MAC* sub-layers are quite separable. This characteristic makes it possible to separately model them.

ii. *SAT-MAC Convergence* sub-layer is modeled on top of an abstract model of *SAT-MAC*.

iii. Detailed *SAT-MAC* model is the same as the *SAT-MAC* model provided for Broadband Satellite Access in [12] with some modifications. The required modifications are explained in Section 4.5.

To provide the *SAT-MAC convergence model* the following assumptions are made:

i. To simplify the model, we assume two UTs are connected to the network to use the shared satellite medium in order to carry their packets.

ii. It is assumed that initialization and registration has already been done.

iii. *SAT-MAC Convergence* model models the dynamic service negotiations between UT and MCS in order to show how UTs can reserve the satellite resources, and modify or delete the previously reserved resources. It does not model the transfer of data between two UTs. After reservation of the satellite resources and receiving the required frequency-time slots, it will be the duty of *SAT-MAC* to handle the data packet transmission.

iv. It must be noted that RSVP reserves the network resources for the link to the first *next hop*. In this model when UT RSVP receives a reservation request it sends the

request to the next UT without any reservation and it is the next UT which reserves the network resources for the link between itself and the satellite.

v. The behavior of the system in response to all of the RSVP messages, except for *TC_Preempt* message, are modeled. This decision is made to avoid unnecessary complication of the model. This assumption does not affect the general behavior of the model. To model the behavior of the system against this message it is required to have a data base of all previously accepted reservation requests and a decision-making policy to preempt one or more existing reservations.

vi. The aggregation of the requests in UTs is only implemented for service category modification requests.

4.3 Inter-LAN SAT-MAC Convergence Model

Figure 4.2 depicts the devised *SAT-MAC Convergence* model for the Inter-LAN connections over broadband satellite systems[1]. It contains two USER_TERMINAL_ SAT_MAC_CONVERGENCE blocks as instances of the same block type and MASTER_CONTROL_STATION_SAT-MAC_CONVERGENCE block. For simplicity, we call them *UT_CONV* and *MCS_CONV*, respectively. *SAT-MAC* is an abstract model of *SAT_MAC* sub-layers of the UT and MCS and the satellite medium.

UT_CONV receives the RSVP messages [2.8.9] from the environment and provides proper responses to them to support the requested QoS requirements. To describe the *UT-CONV* functions and messages, to explain how RSVP collects the information about the

[1] Refer to: www.ece.concordia.ca/Research/citr/citr_major/projects/2-1/members/a_garjan/main.html

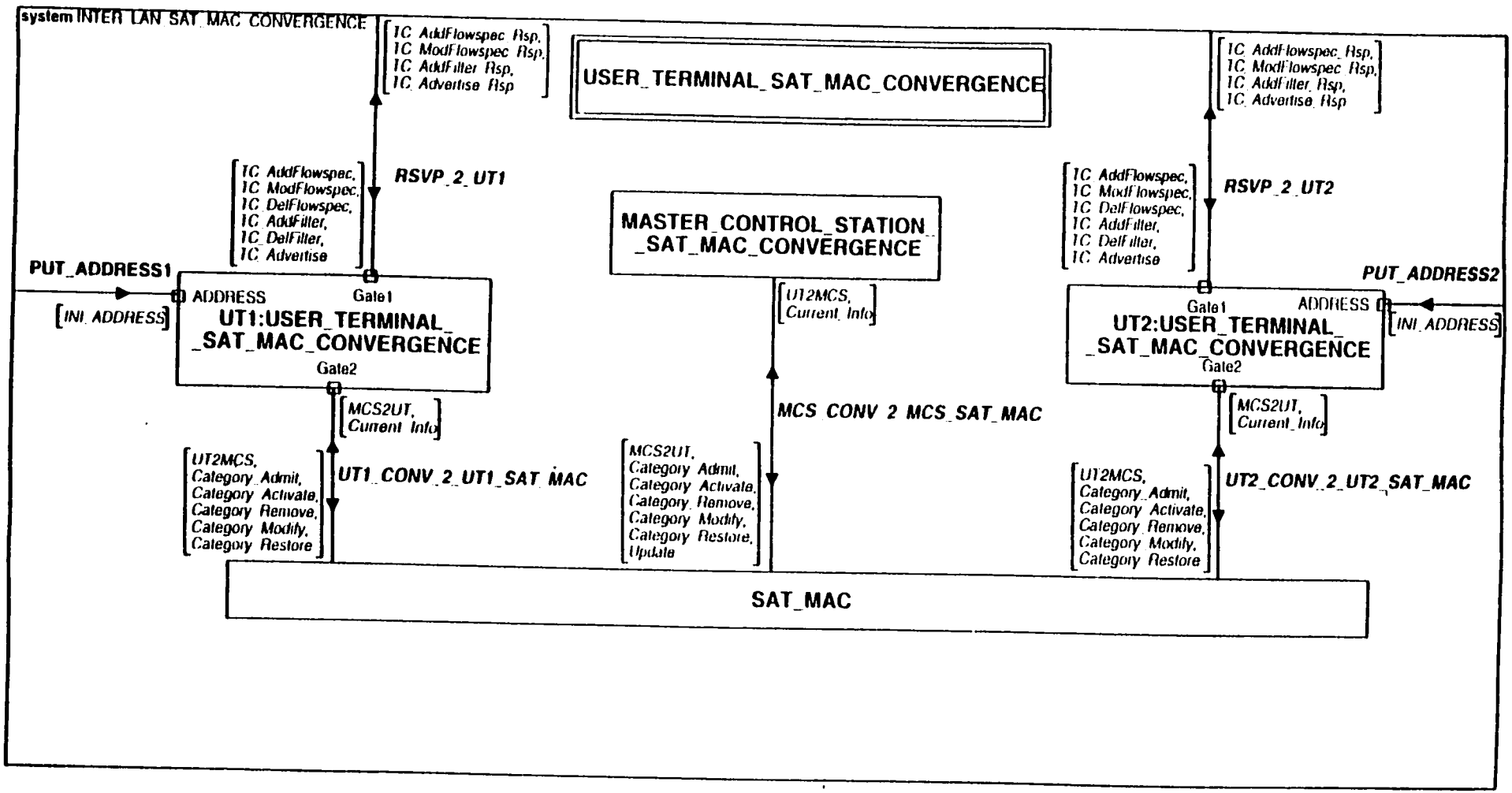

Figure 4.2 Inter-LAN SAT-MAC Convergence System

available network resources and how reserves the requested bandwidth a simplified scenario is depicted in Figure 4.3. We simply assume Sender host is located in LAN1 and Receiver host is in LAN2 as shown in Figure 2.4. Router1 and UT1 are connected to LAN1 and Router2 and UT2 are connected to LAN2. After receiving the *Path* message from Sender host through Router1, UT1-RSVP on top of the *UT1_CONT* sub-layer passes the *Adspec* parameters, as the "arriving" ones, to the *UT1_CONT*. These parameters carry information about the actual available network resources, i.e. bandwidth, before arriving at UT1. *UT1-CONT* collects the corresponding "current" information about the actual available satellite resources. By using some routines explained in RSVP documents, the "arriving" and "current" parameters will be combined to produce the "result" parameters to be passed to UT1-RSVP. In turn, UT1 sends the RSVP *Path* message carrying the result parameters as new "arriving" ones to UT2. UT2 and Router2 do the same actions and finally the Receiver host receives the *Path* message After analyzing the "arriving" parameters, Receiver host decides to send a *Resv* message to the Sender host to reserve the required network resources in the path to achieve the required QoS. This message contains the Receiver's desired QoS and traffic parameters, which are determined by analyzing the total "arriving" parameters in the *Path* message. After receiving the *Resv* message, Router2 reserves the requested QoS for the link between Router2 and UT2 and passes the *Resv* message to UT2. UT2 without reserving passes the *Resv* message to UT1-RSVP. UT1-RSVP must negotiate with MCS through

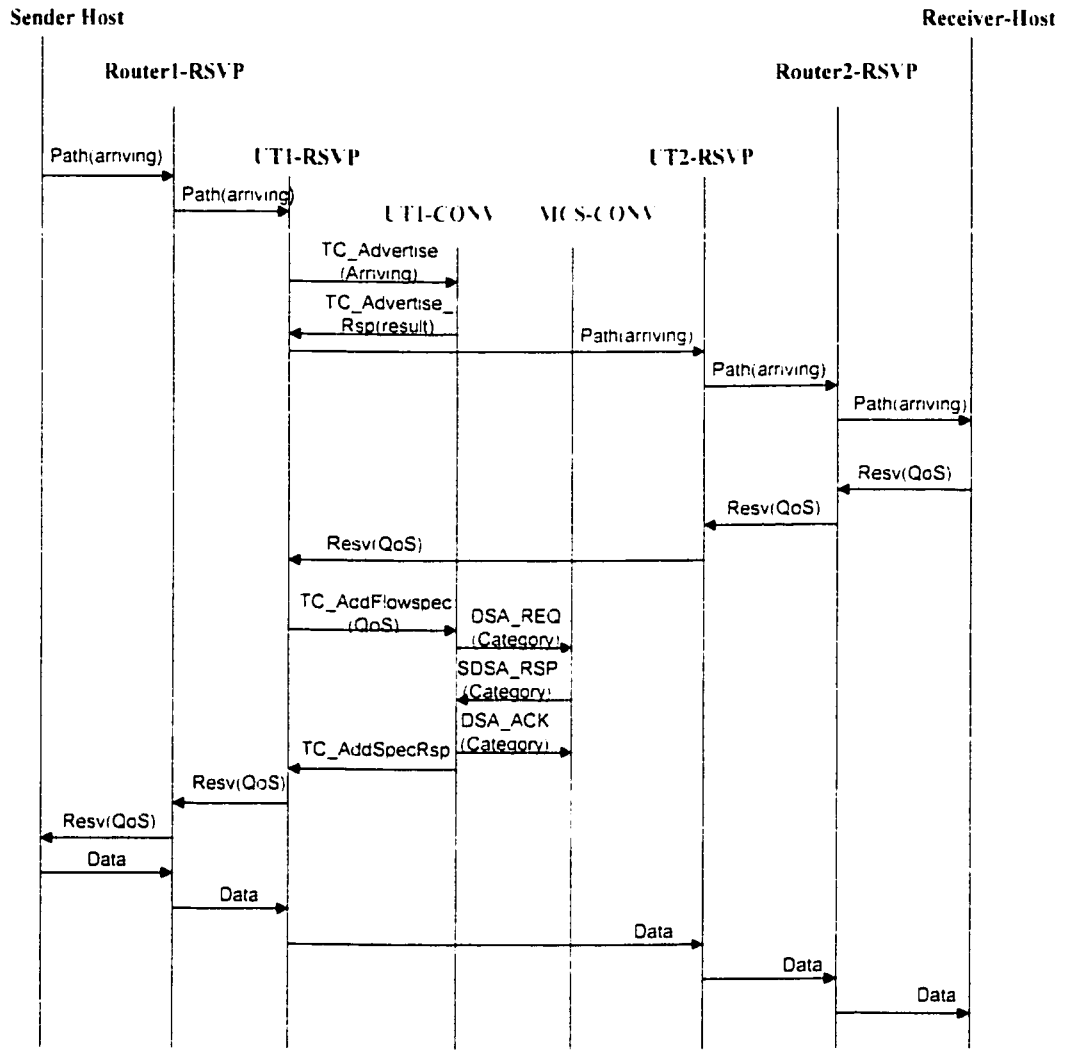

Figure 4.3 RSVP Path and Resv Message

SAT_MAC Convergence sub-layers to reserve the requested resources. These negotiations are done through dynamic service messages explained in this Section. By successful negotiation, the set-up phase is concluded and MCS is committed to provide the requested slots in data phase. Upon reserving the required bandwidth in satellite medium, UT1-RSVP will reserve the bandwidth for the next link to Router1 and will pass the *Resv* message through Router1 to Sender host and data phase will start.

More detailed explanations about RSVP and RSVP_MAC interface messages can be found in [2] and Appendix A. The messages received by a UT from environment (i.e., RSVP) and relevant *UT_CONF* responses in system level are as follows:

i. ***TC_AddFlowspec*:** This message is received from RSVP to ask for a new reservation request for specified QoS parameters. This message triggers the RSVP_MAC_INTERFACE to start a reservation request negotiation.

ii. ***TC_AddFlowspec_Rsp*:** Depending on the availability of the network resources and in response to *TC_AddFlowspec*, RSVP is answered back by *TC_AddFlowspec_Rsp* message positively, negatively or a positive answer with some modification to the requested QoS parameters. If the answer is positive, it returns a pointer to the accepted reservation. Otherwise, it returns a code indicating negative response. If the reservation request is accepted by some modification, a pointer to the modified *Flowspec* is returned

iii. ***TC_ModFlowspec*:** It is used to modify a reserved *Flowspec*. This message causes the same scenario as *TC_AddFlowspec* does.

iv. ***TC_ModFlowspec_Rsp:*** This message returns the modification request result. If it is successful the requested Flowspec is modified and a pointer to the modified *Flowspec* is returned to RSVP.

v. ***TC_DelFlowspec*:** It is used to delete a reserved *Flowspec*.

vi. ***TC_AddFilter:*** This message adds a new *Filterspec* to the session. This message causes the addition of a *Filterspec* to the *Filter-list* in the RSVP_MAC_INTERFACE.

vii. ***TC_AddFilter_Rsp:*** It returns a pointer to added *Filterspec.*

viii. ***TC_DelFilter:*** It removes a *Filterspec* from the session.

ix. ***TC_Advertise:*** Carrying the 'arriving' *Adspec*, it triggers the RSVP_MAC _INTERFACE to collect the 'current' *Adspec* parameters.

x. ***TC_Advertise_Rsp:*** It carries the "result" *Adspec* parameters to RSVP.

In system level, UTs and MCS *SAT_MAC Convergence* blocks negotiate through ***UT2MCS*** and ***MCS2UT*** messages. These messages are transferred through the *SAT-MAC* sub-layer and satellite medium to exchange dynamic service related signals explained in Section 3.5. Using these messages UT and MCS can make a request for addition, change or deletion of a dynamic service. To accomplish these negotiations, *UTs* and *MCS_CONV* must know the MAC address of each other. We assumed the registration has already been completed, and *UTs* and *MCS_CONV* know each other. UTs addresses are given to them through *INI_ADDRESS* messages in the beginning of the simulation.

To update the category related information in UT and MCS *SAT-MAC* the following messages are sent from *UT* and *MCS-CONV* to the UT and MCS *SAT-MAC*. It must be noted that these messages are local messages and are not transferred through medium:

i. ***Category-Admit***: When *UT* or *MCS-CONV* admits a dynamic service addition request, received from its counterpart, it updates the category records in its own *SAT-MAC* sub-layer.

ii. ***Category-Activate****:* By receiving a dynamic service addition acknowledgement. *UT* or *MCS-CONV* triggers this message to update its own *SAT-MAC* sub-layer category records.

iii. ***Category-Modify:*** Acceptance of a dynamic change request by *UT* or *MCS-CONV* causes this message to be issued to update the category records in UT and MCS *SAT-MAC*.

iv. ***Category-Remove***: Acceptance of a dynamic deletion request or any scenarios which leads to the cancellation of a service category triggers the *UT* and *MCS-CONV* to update the category records in its own *SAT-MAC* sub-layer.

v. **Category-Restore:** A negative dynamic change acknowledgement causes *UT* or *MCS-CONV* to issue this message to its *SAT-MAC* in order to restore the modified category.

vi. ***Current-Info:*** UT and MCS *SAT-MAC* updates their own *Convergence* sub-layer by issuing this message and sending the current information about the service categories.

4.3.1 UT SAT_MAC Convergence

Figure 4.4 shows the UT_SAT_MAC_CONVERGENCE block type. This block contains the following processes:

i. **RSVP_MAC_INTERFACE:** This process is responsible to interact with RSVP protocol.

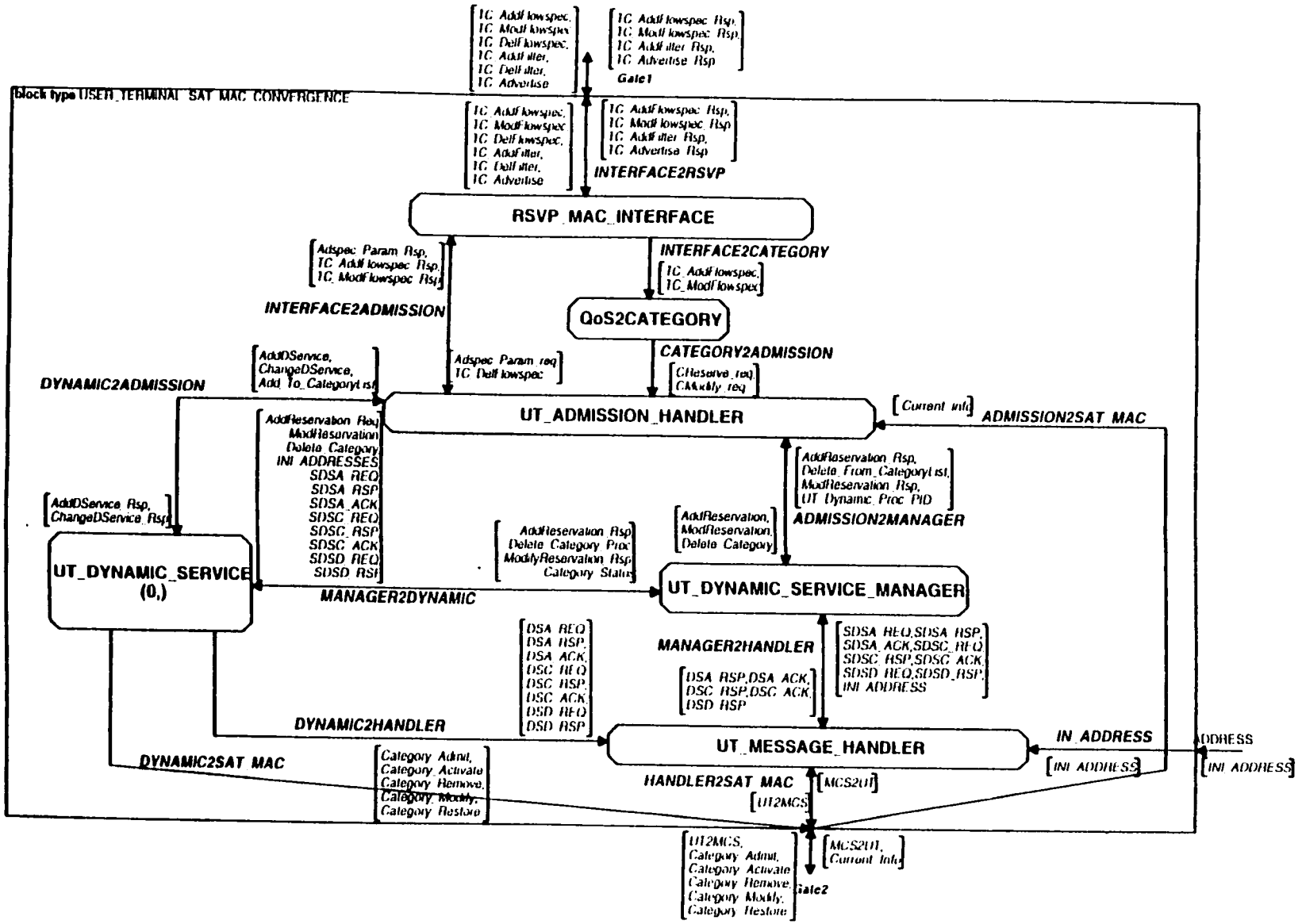

Figure 4.4 UT SAT-MAC Convergence Block Type

It receives the RSVP messages *TC_AddFlowspec* and *TC_ModFlowspe* per connection in order to reserve or modify a *Flowspec*. These messages carry the information about the requested QoS parameters. These messages are passed to QoS2Category process in order to translate the requested QoS parameters to a specific service category and an amount of required time-frequency slots. Responses to these requests, *TC_AddFlowspec_Rsp* and *TC_ModFlowspec_Rsp* are passed to the RSVP through this process.

RSVP requests for deletion of a previously reserved *Flowspec*, i.e., *TC_DelFlowspec*, is passed to UT_ADMISSION_HANDLER to be processed there. This message carries a pointer to a previously reserved *Flowspec* to be deleted.

Upon reception of *TC_Advertise*, this process issues *Adspec_Param_req* to UT_ADMISSION_HANDLER in order to collect all information about "current" available network resources. This interface by using the RSVP predefined routines, combines the "current" information with "arriving" parameters to provide the "result" network capability.

This process builds a dynamic list of all participants in a session by handling the *TC_AddFilter*. The response to this message, *TC_AddFilter_Rsp* carries the information about the added participant. Request for deletion of one participant in a session, by receiving *TC_DelFilter*, modifies the *Filter-list*.

ii. **QoS2CATEGORY:** This process supports the translation from the *individual* QoS requirements to the predefined service categories and amount of required time-frequency slots.

This process receives *TC_AddFlowspec* and *TC_ModFlowspec* carrying the requested QoS parameters and issues *CReserve_req* and *CModify_req* carrying service category and an amount of time-frequency slot. This process is flexible to adjust itself with IPv6 possibilities in defining new service classes.

iii. **UT_ADMISSION_HANDLER:** This process contains all updated information about all activated, admitted, removed, modified and restored service categories. It has the whole updated information about the capacity of the UTs data queues. Using this information, on one hand, it reacts to the received requests from RSVP-MAC-INTERFACE and QoS2Category processes and on the other hand decides about the requests related to dynamic service addition, change and deletion issued by *MCS-CONF*. It may answer negatively or positively to the dynamic addition or modification requests of MCS but it must accept the deletion request of MCS. This process may aggregate the service category modification requests to alleviate the processing load in MCS. Upon receiving the response to aggregated requests, it answers back to the RSVP individual requests.

This process receives *Current_Info* from UT *SAT-MAC* containing all updated information about the capacity of the UTs data queues and service category records. It uses this information in processing the received requests.

By receiving *Creserve_req* from QoS2Category in order to reserve a specified amount of frequency-time slot for a particular service category, it issues *AddReservation* or *ModReservation* requests, according to the existence or nonexistence of the requested service category, respectively. These messages are issued to UT_DYNAMIC_

SERVICE_MANAGER to start a negotiation with MCS through DYNAMIC_ SERVICE to get the necessary responses.

TC_DelFlowspec received from RSVP_MAC_INTERFACE causes this process to issue a *Delete_Category* or *ModReservation* depending on the available information about the service category. If execution of *TC_DelFlowspec* removes all the reservations for a service category it issues *Delete_Category* .or if its execution causes a decrease in the amount of the already reserved frequency-time slots it issues *ModReservation.*

CModify_req carries a modification request for a previously reserved service category. This process may send a *ModReservation* for individual requests or aggregate the requests for modifications in a single request.

AddReservation_Rsp received from DYNAMIC_SERVICE_MANAGER carries the result of negotiation with MCS in response to the *Addreservation.* Positive response causes the addition of the accepted service category to the list of active service categories and the insertion of a pointer to a list indicating the acceptance of this request. Both positive and negative answers trigger this process to issue a *TC_AddFlowspec_Rsp* to the RSVP_MAC_INTERFACE.

ModResevation_Rsp may be received in response to an individual or aggregated *CModify_req*, *Creserve_req* or *TC_DelFlowspec*. This response is analyzed by this process and depending on the type of the response necessary modifications is applied to the current information inside this process.

AddDService and *ChangeDService* negotiate the addition and modification request issued by *MCS_CONF*. This process may give positive or negative response to these requests through *AddDService_Rsp* and *ChangeDService_Rsp*.

Successful establishment of a service category adds a new service category to the category list through receiving *Add_To_CategoryList* or positive *AddReservation_Rsp*. Unsuccessful effort to add a service category or a category service deletion request issued by MCS deletes a service category from the category list.

iv. **UT_DYNAMIC_SERVICE_MANAGER:** This process has the responsibility to create the UT_DYNAMIC_SERVICE processes and hold a dynamic list of all the created processes. Termination of each created UT_DYNAMIC_SERVICE processes deletes that process from process and category list held in UT_DYNAMIC_MANAGER. This process is responsible to transfer the received messages from the processes inside the *UT_CONF* and dynamic service related messages from the *MCS_CONF* to the right destination. This process contains a dynamic list of each dynamic addition, deletion and change transactions between UT and MCS. Furthermore, this process acts like a monitor to decide to pass the messages to the created UT_DYNAMIC_SERVICE processes or reject them. To do this it assigns some indicators to each created UT_DYNAMIC_SERVICE processes. When a UT_DYNAMIC_SERVICE process is processing an addition or change request, it sets a *flag* to prevent of sending any new addition or change request destined to this process. It is important to note that deletion request can be forwarded to the UT_DYNAMIC_SERVICE even in the creation period. However, when a process is in

deletion cycle. UT_DYNAMIC_SERVICE_MANAGER sets the *Dflag* and prevents any messages to be transferred to the process in deletion phase.

This process composes of two different states. free and Wait_For_Address states. At the start. the process waits to receive the UT MAC address and then goes to free state to be able to receive other messages.

All dynamic service related messages issued by *MCS_CONF* and messages related to addition. modification and deletion of service categories issued by *UT_CONF* upper layers are passed and monitored in this process. Based on the received *AddReservation* or *SDSA_REQ* requests from UT_ADMISSION_ HANDLER or *MCS_CONF* for service categories. this process creates a UT_DYNAMIC_SERVICE process for each requested service category. All modification and deletion requests for a specific service category must be transferred to the process created for that service category.

v. **UT_DYNAMIC_SERVICE:** Corresponding to each requested service category an instance of this process is created. Created process instances are responsible to communicate with their own counterpart in MCS to handle the negotiation for dynamic service addition. change and deletion requests. For this purpose. this process instances are responsible to set their timer after sending each dynamic addition. change or deletion request and repeat it in the case of no response. After receiving the response. only in the case of addition and change request. it must be acknowledged. This process assign a transaction ID for each addition. change and deletion request which is initiated by *UT_CONF* and delete it after successful or unsuccessful termination of the negotiation. By implementing this scenario. the communication between dynamic

service processes in MCS and UT will seamlessly be accomplished. It must be noted that each instance of this process each time can respond only to dynamic addition or change request. In the other word, when this process is doing a dynamic addition request and the transaction for this request has not been finished yet, it can not start another addition or change request. However, this process can receive and execute the dynamic deletion request anytime.

This process consists of following states:

At the beginning, it waits for receiving the UT address and its own ID. This process uses UT address and its ID in communication with MCS and other processes. After receiving this information, the process goes to idle state and waits for *AddReservation_Req* or *SDSA_REQ* from the UT_DYNAMIC_SERVICE_MANAGER or MCS_CONV, respectively.

Receiving *AddReservation_Req* triggers the process to set a timer and send a DSA_REQ to *MCS_CONV* and wait in DSA_RSP_PENDING state to receive the MCS response. In this state, if it does not receive the response in a proper time it will try the DSA_REQ again. Successful positive response causes a transition to SC_Operational state, which means service category is activated and a DSA_ACK is sent to MCS_CONV after setting a transaction timer. Negative or five unsuccessful timeouts cause the termination of the process. In SC_Operational, transaction timeout indicates the establishment of the requested category service. Figure 4.5 shows a successful service category addition scenario after receiving AddReservation.

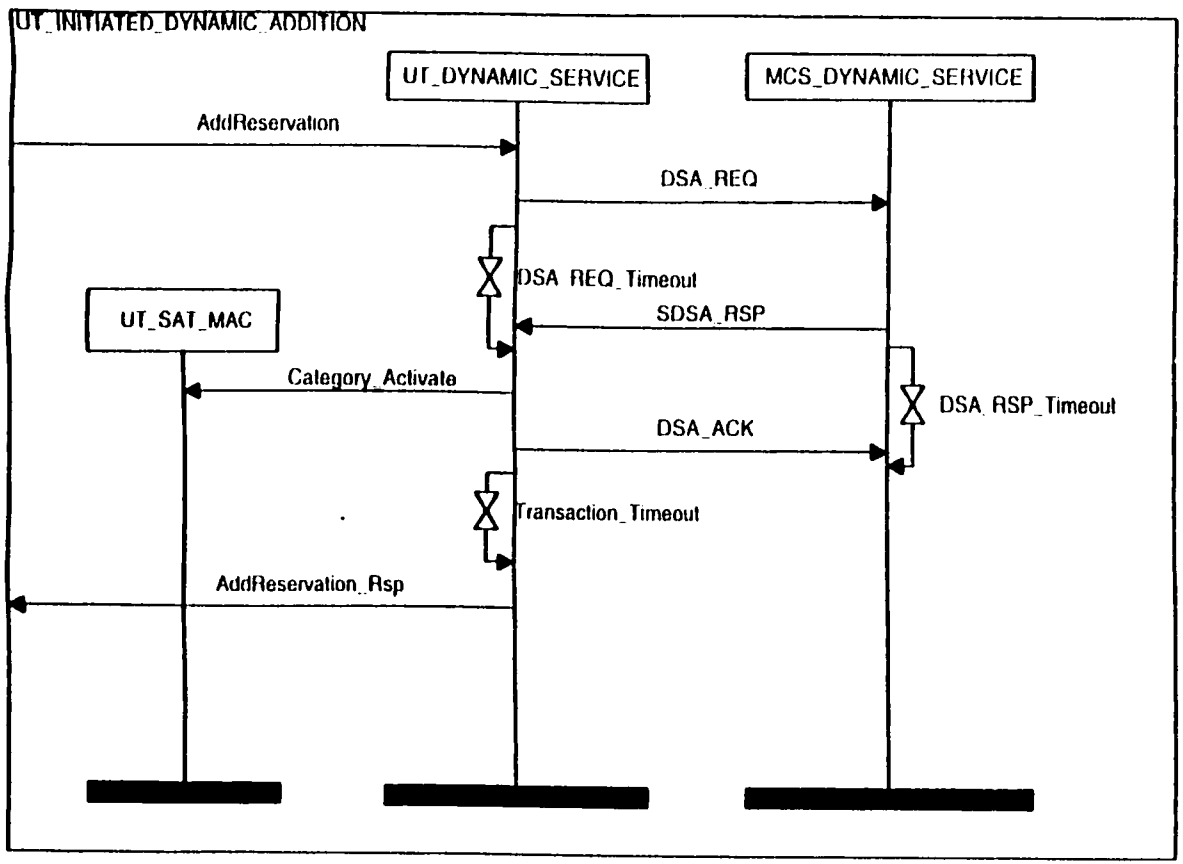

Figure 4.5 Successful UT Initiated Dynamic Addition

Receiving SDSA_REQ from *MCS_CONF* in idle state causes the process to issue the *AddDService* to UT_ADMISSION_HANDLER to get the response for this dynamic addition request. Process waits for the response in Wait_For_UT_Admission state and after getting the response it sends Category_Admit to UT_SAT_MAC (if the response is positive) and sets a timer. It issues the DSA_RSP to *MCS_CONF* and waits for SDSA_ACK in DSA_ACK_Pending. In this state receiving a positive acknowledgement causes the service category to be activated and to be transferred to SC_Operational. Negative response or five times unsuccessful trying to receive the SDSA_ACK will terminate the process. Figure 4.6 depicts a successful service category addition after receiving a SDSA_REQ from *MCS_CONF*.

In SC_Operational state the process can start modification transactions by receiving SDSC_REQ or ModReservation from *MCS_CONF* or *UT_CONF* processes, respectively.

ModReservation message will cause a timer to be set. DSC_REQ to be sent to *MCS_CONF* and the process must wait for response in DSC_RSP_Pending state. In this state if it does not receive the response in a proper time it will try the DSC_REQ again. Successful positive response causes a transition to SC_Operational state, which means service category is modified. Negative or five unsuccessful timeouts cause the process to go to SC_Operational state and terminate the transaction. Figure 4.7 shows a successful service category modification scenario after receiving ModReservation.

Receiving SDSC_REQ from *MCS_CONF* in SC_Operational state causes the process to issue the *ChangeDService* to UT_ADMISSION_HANDLER to get the response for

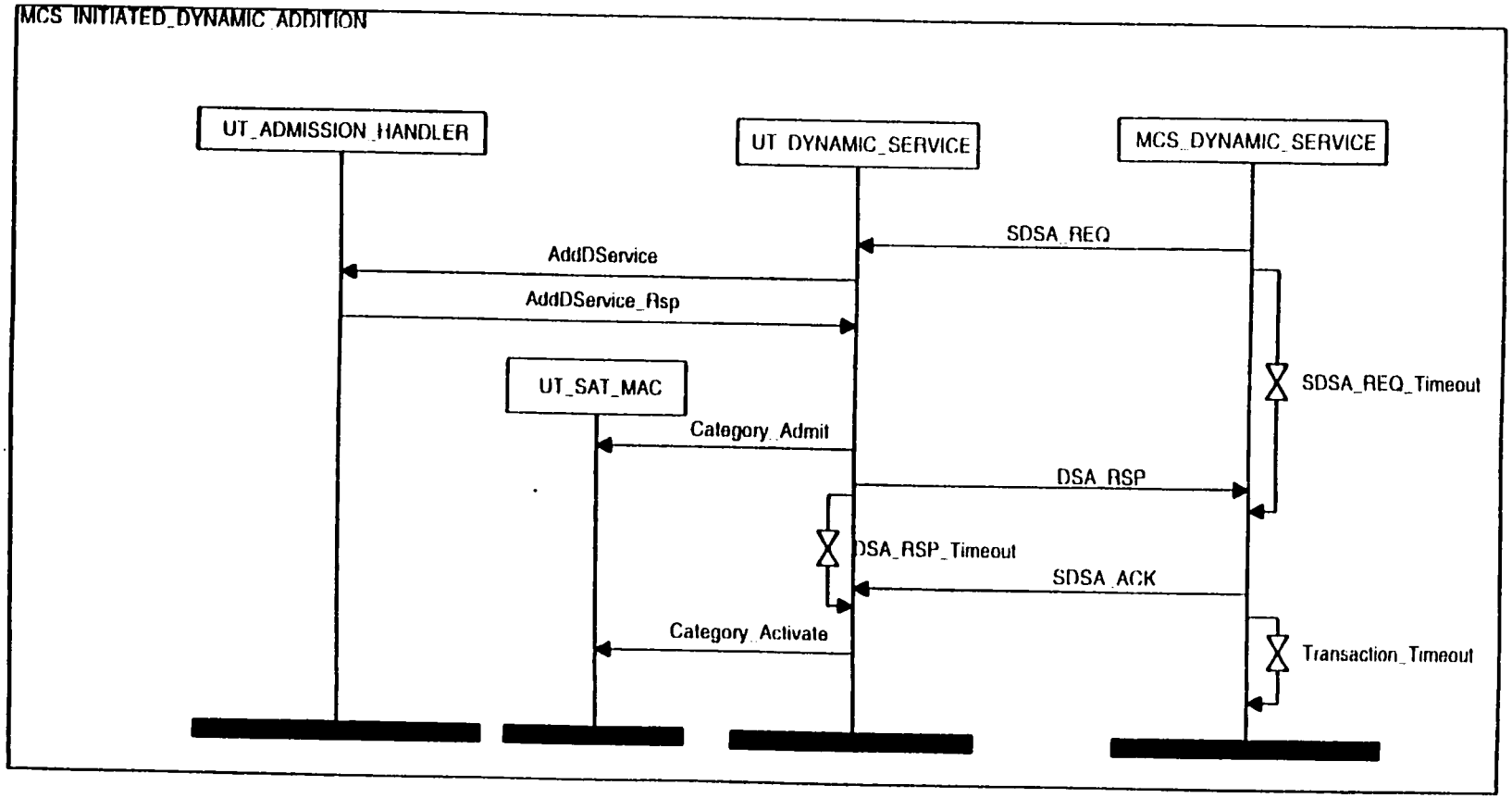

Figure 4.6 Successful MCS Initiated Dynamic Addition

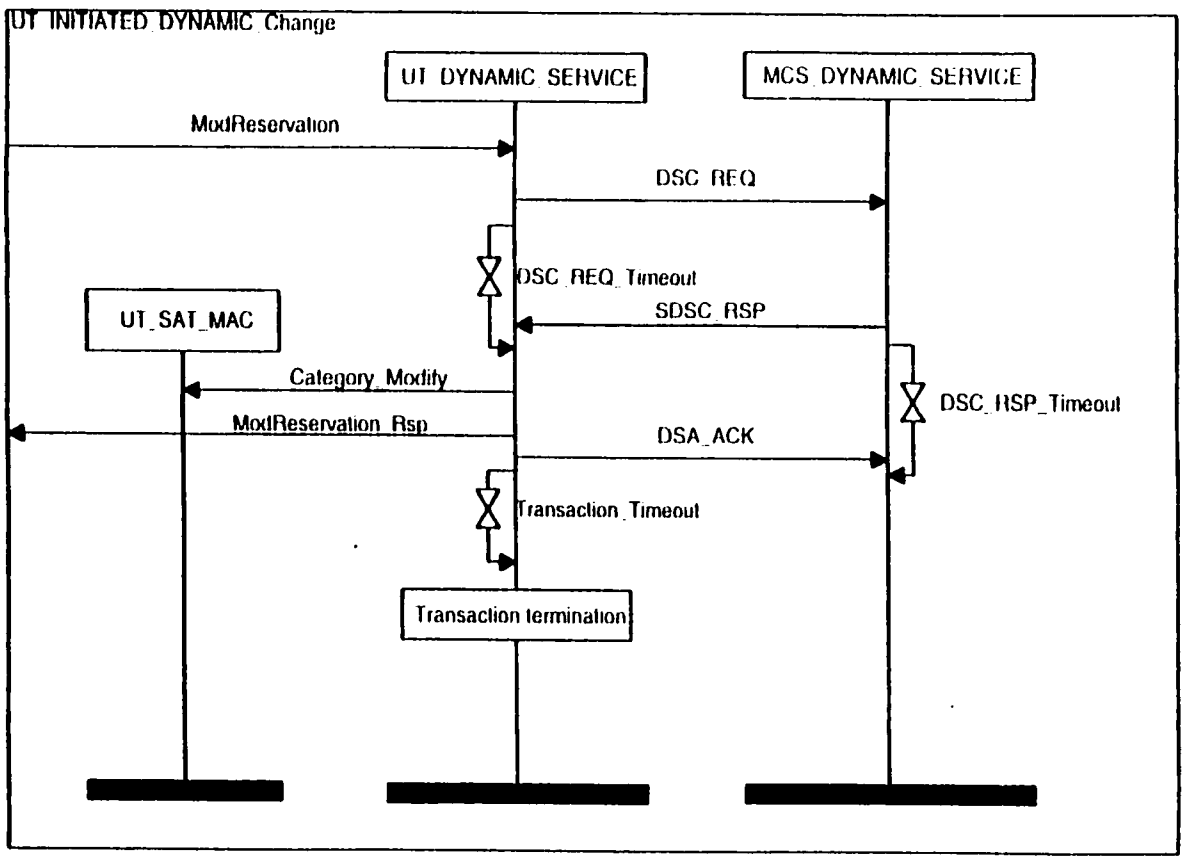

Figure 4.7 Successful UT Initiated Dynamic Change

this dynamic modification request. Process waits for the response in Wait_For_UT_Admission state and after getting the response it sends a Category_Modify to UT_SAT_MAC (if the response is positive) and sets a timer. It issues the DSC_RSP to *MCS_CONV* and waits for SDSC_ACK in DSC_ACK_Pending state. In this state receiving a positive acknowledgement causes the service category to remain modified and the negative acknowledgement or five times unsuccessful trying to receive the SDSC_ACK will cause to send a Restore_Category message to UT_SAT_MAC to restore the modified category. In each case the process will be transferred to SC_Operational state. Figure 4.8 depicts a successful service category modification after receiving a SDSC_REQ from *MCS_CONV*.

SDSD_REQ and Delete_Category may be received in each states of this process except idle and Wait_For_Address states. It means that each UT_DYNAMIC_SERVICE process instance can be deleted even when an addition or change transaction is in the middle of negotiation.

Receiving a SDSD_REQ from *MCS_CONV* in each state triggers a DSD_RSP to MCS and a Category_Remove to UT_SAT_MAC to update the information. The process will be terminated after sending these messages. Figure 4.9 depicts this scenario.

Delete_Category received from UT_DYNAMIC_MANAGER triggers a Category_Remove to UT_SAT_MAC in order to update the information. Then it sends a DSD_REQ to *MCS_CONV* and sets a timer and waits for response in DSD_RSP_

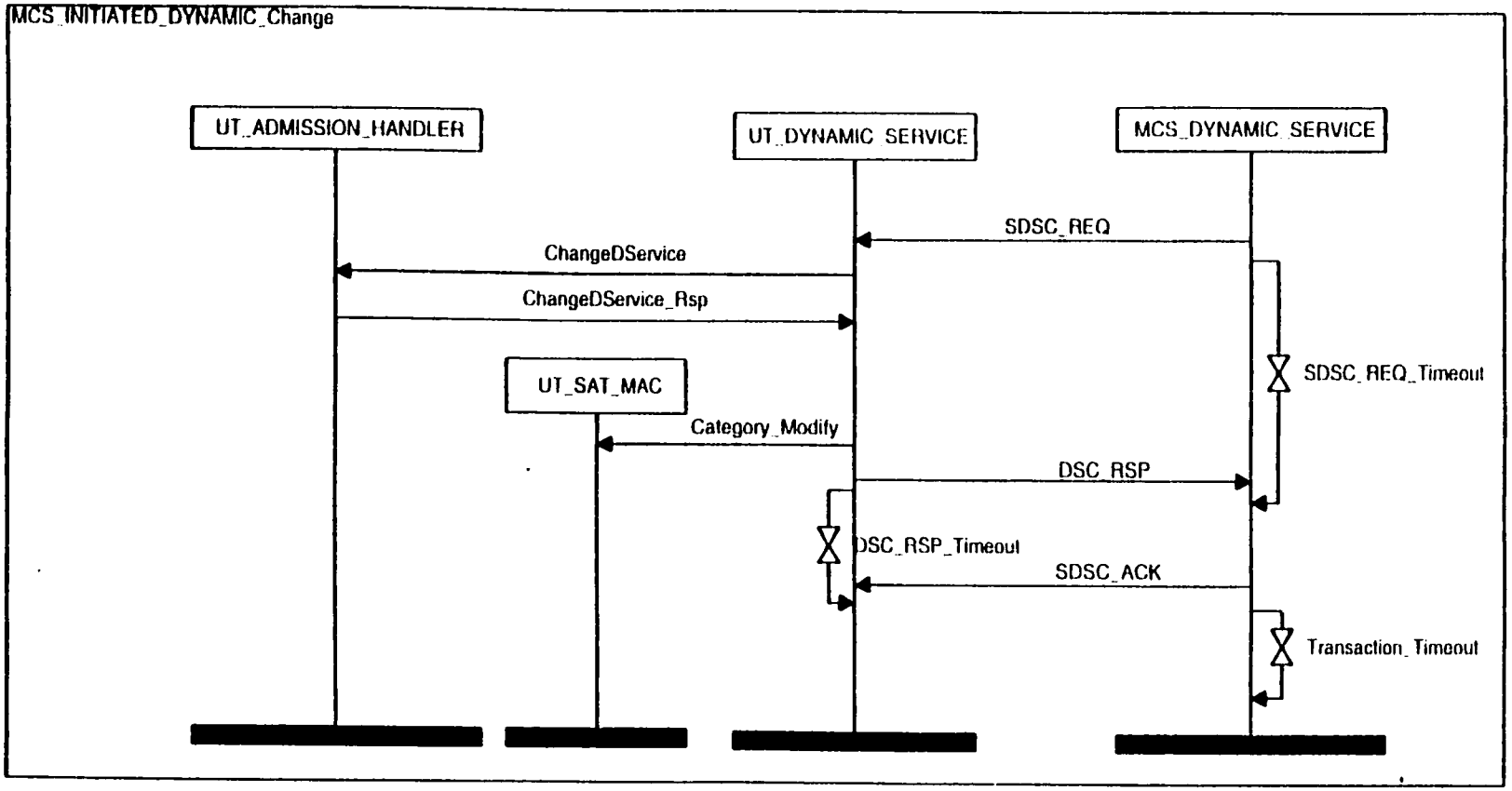

Figure 4.8 Successful MCS Initiated Dynamic Change

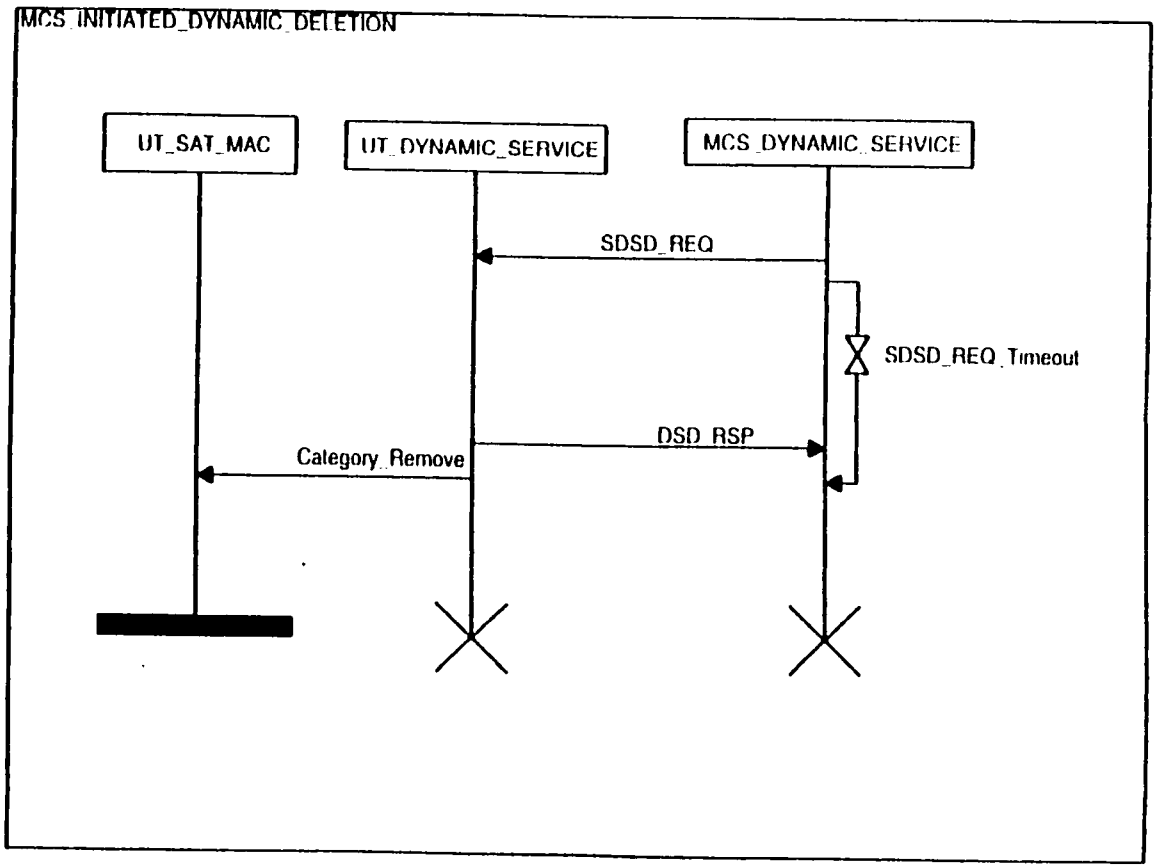

Figure 4.9 Successful MCS Initiated Dynamic Deletion

Pending state. In DSD_RSP_Pending after receiving the SDSD_RSP or five timeouts the process will be terminated. Figure 4.10 shows this scenario.

It must be noted that in dynamic deletion transactions we do not need any acknowledgements.

vi. **UT_MESSAGE_HANDLER:** This process is responsible to unpack and pack the MCS2UT and UT2MCS messages. Receiving MCS2UT message, it unpacks the message and transfers it to UT_DYNAMIC_SERVICE_MANAGER. It packs all the dynamic service related messages received from UT_DYNAMIC_SERVICE_ MANAGER and UT_DYNAMIC_SERVICE processes in UT2MCS message and pass it to the SAT_MAC block.

4.3.2 MCS SAT-MAC Convergence

Figure 4.11 shows the MCS_SAT_MAC_CONVERGENCE block. This block contains the following processes:

i. **POLICY_CONTROL:** This process checks if the UTs have administrative permission to use a specific service category. In registration phase, each UT will be allowed to use the satellite resources according to the subscription contract. POLICY_CONTROL have to keep a record of this contract and control any violation. Each dynamic service request initiated by UTs must be checked to prevent any violation. By POLICY_ CONTROL's positive processing result, the request will be passed to MCS_ADMISSION_ CONTROL to check the availability of satellite

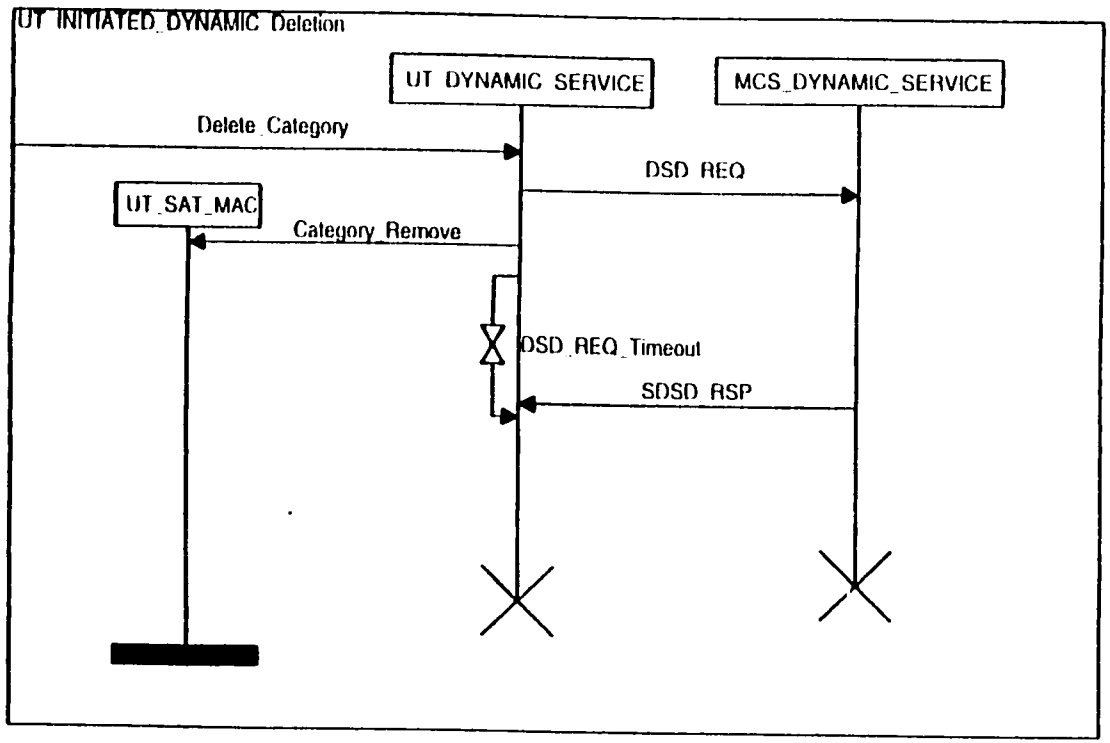

Figure 4.10 Successful UT Initiated Dynamic Deletion

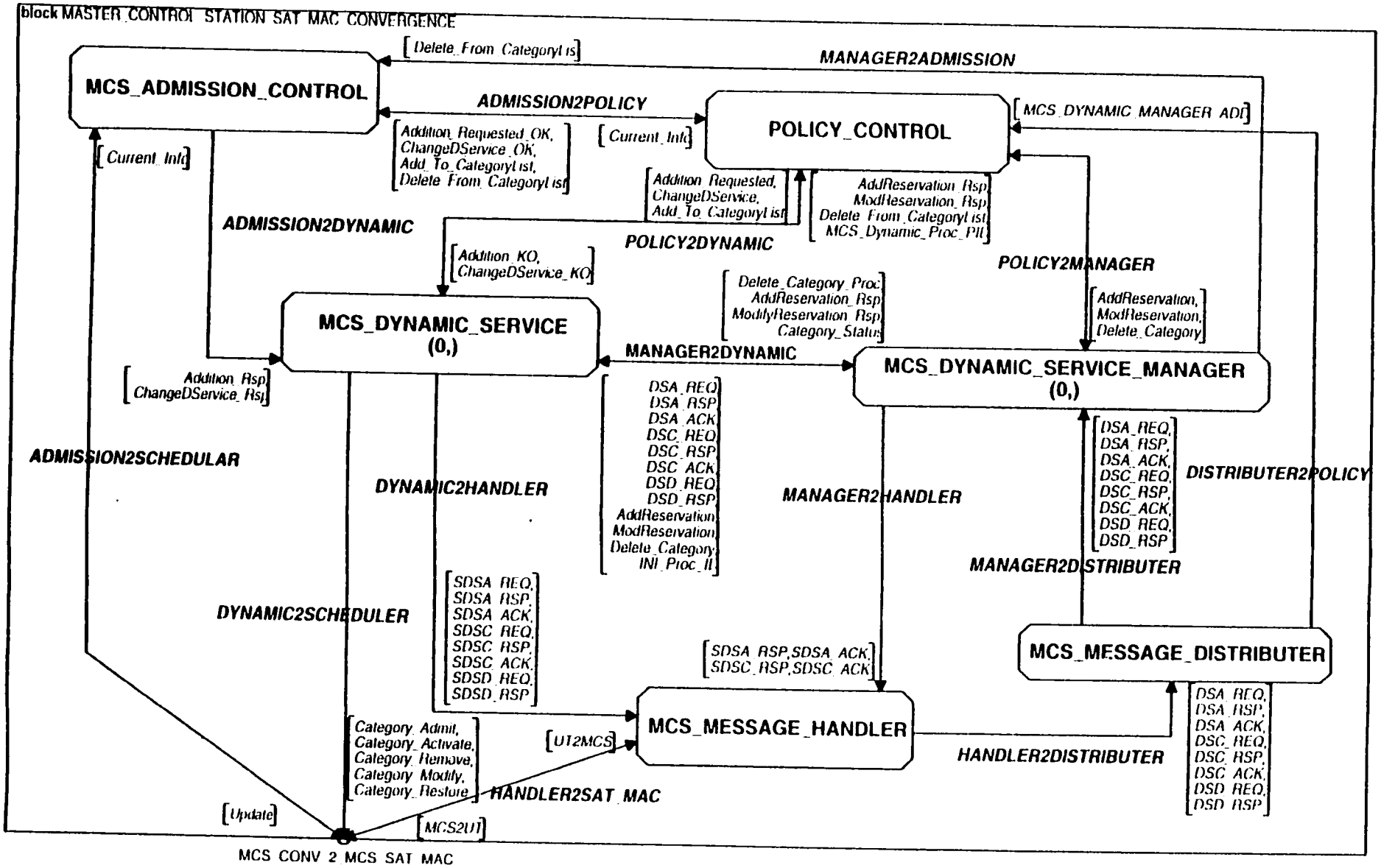

Figure 4.11 MCS SAT-MAC Convergence Block

resources. POLICY_CONTROL may initiate a dynamic addition. change or deletion request for a specific UT.

This process keeps a list of all UTs connected to the network and a dynamic list of service categories for each of UTs. Upon receiving *Addition_Requested* or *ChangeDService* from MCS_DYNAMIC_SERVICE (i.e.. UT initiated dynamic addition or change request) it checks if the UT has permission to use the requested service category. The negative response will be passed directly to the MCS_DYNAMIC_ SERVICE and positive response is transferred to MCS_ADMISSION _CONTROL to be checked against the availability of the network resources.

Successful addition or deletion of a service category adds or deletes a category from the category list of each connected UT by receiving *Add_To_CategoryList* or *Delete_From_CategoryList.* respectively.

Upon receiving the current information from MCS_ADMISSION_CONTROL. this process can initiate a dynamic addition. change or modification for a specific UT and service category (i.e.. MCS initiated dynamic service request). These requests are triggered by *AddReservation. ModReservation* and *Delete_Category* messages. Positive *Addreservation_Rsp* adds and Delete_Category deletes a new service category to and from the category list of the corespondent UT.

ii. **MCS_ADMISSION_CONTROL:** This process receives all accepted requests from POLICY_CONTROL and must decide to give the proper answer to all these requests considering the available satellite resources. Decision making strategies are implemented in this process by processing all current information received from

SCHEDULER. MCS_ADMISSION_CONTROL may have the following decisions on requests for dynamic service addition or change: accept. refuse or accept with a reduced commitment. In the latter case. the MCS_ADMISSION_CONTROL is allowing for negotiation to the UTs. This process must update POLICY_CONTROL and SCHEDULER by its new decisions.

iii. **MCS_DYNAMIC_SERVICE_MANAGER:** This process has the responsibilities similar to its counterpart in UT_CONV. But it must be noted that in MCS_CONV for each connected UT we must create a MCS_DYNAMIC_ SERVICE_MANAGER process. This process is created by MCS_MESSAGE_ DISTRIBUTER.

This process has the responsibility to create the MCS_DYNAMIC_SERVICE processes and hold a dynamic list of all the created processes. Termination of each created MCS_DYNAMIC_SERVICE processes deletes that process from process and category list held in MCS_DYNAMIC_MANAGER. This process is responsible to transfer the received messages from the processes in the *MCS_CONV* and dynamic service related messages from the *UT_CONV* to the right destination. This process contains a dynamic list of each dynamic addition. deletion and change transactions between UT and MCS. Furthermore. this process acts like a monitor to decide to pass the messages to the created MCS_DYNAMIC_SERVICE processes or reject them. To do this it assigns some indicators to each created MCS_DYNAMIC_SERVICE processes. When a MCS_DYNAMIC_SERVICE process is implementing an addition or change request. it sets a *flag* to prevent of sending any new addition or change request destined to this process. It is important to note that deletion request can be forwarded to the

MCS_DYNAMIC_SERVICE even in the creation period. However, when a process is in deletion cycle, MCS_DYNAMIC_SERVICE_MANAGER sets the *Dflag* and prevents any messages, even a new deletion request, to be transferred to the process.

All dynamic service related messages issued by POLICY_CONTROL and messages related to addition, modification and deletion of service categories issued by *UT_CONV* are passed and monitored in this process. Based on the received *AddReservation* or *DSA_REQ* requests from POLICY_CONTROL or *UT_CONV*, this process creates a MCS_DYNAMIC_SERVICE process for each requested service category. All modification and deletion requests for a specific service category must be transferred to the process created for that service category.

iv. **MCS_DYNAMIC_SERVICE:** This process has the same responsibilities and states of its counterpart in UT_CONV.

To describe the relation between MCS_DYNAMIC_SERVICE, POLICY_CONTROL and MCS_ADMISSION_CONTROL Figure 4.12 and 4.13 depicts the MSC messages to show the successful UT initiated dynamic addition and change requests processed in MCS_CONV.

As we can notice from these MSCs every addition or change request initiated by UT must pass both POLICY_CONTROL and MCS_ADMISSION_CONTROL check. If POLICY_CONTROL check passes successfully the request is transferred to MCS_ADMISSION_CONTROL check (as the scenario shows), otherwise POLICY_ CONTROL will negatively answer back to the MCS_DYNAMIC_ SERVICE. In depicted scenarios, MCS_ADMISSION_CONTROL responds positively and two

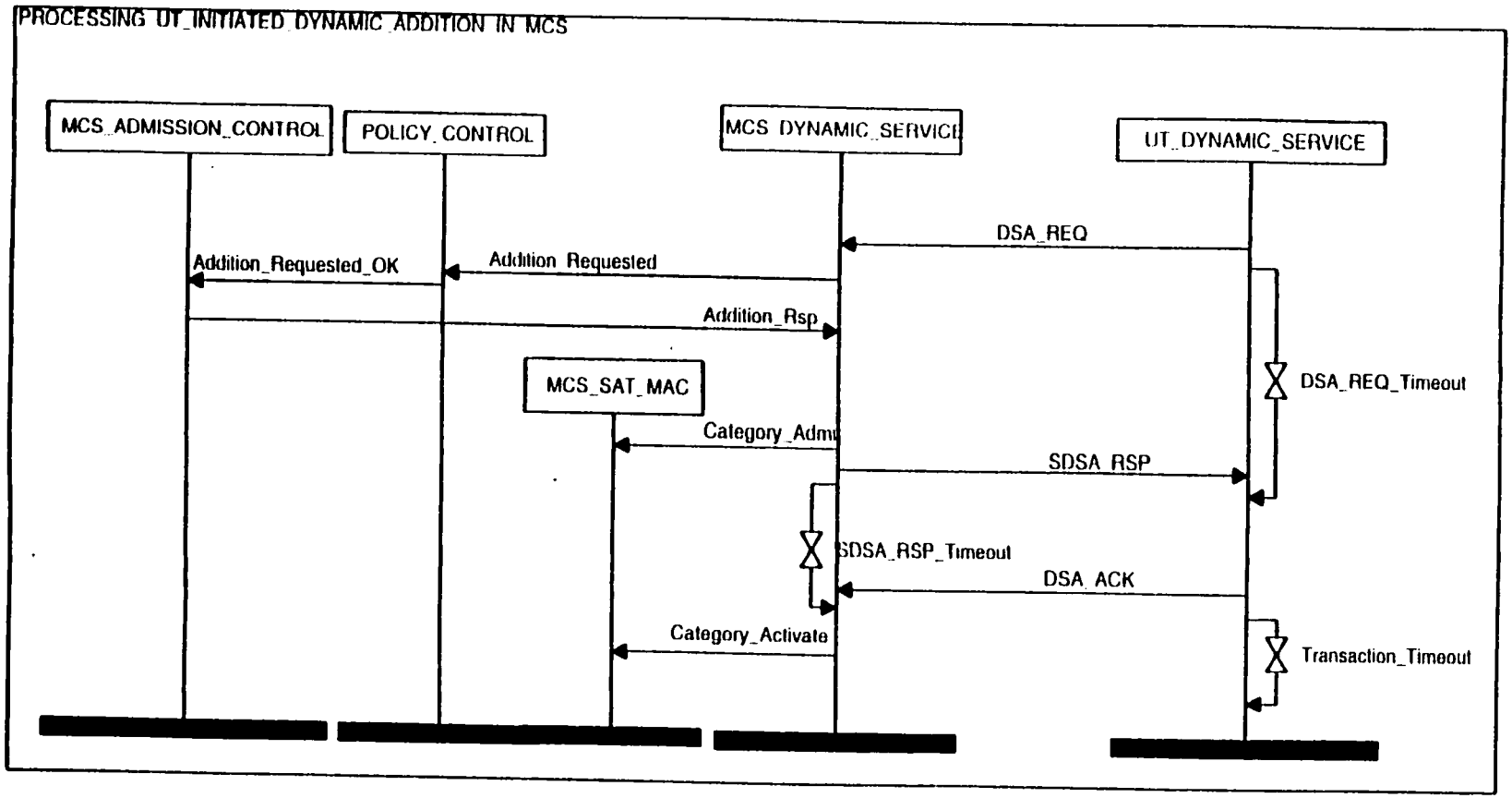

Figure 4.12 Successful UT Initiated Dynamic Addition Processed in MCS-CONV

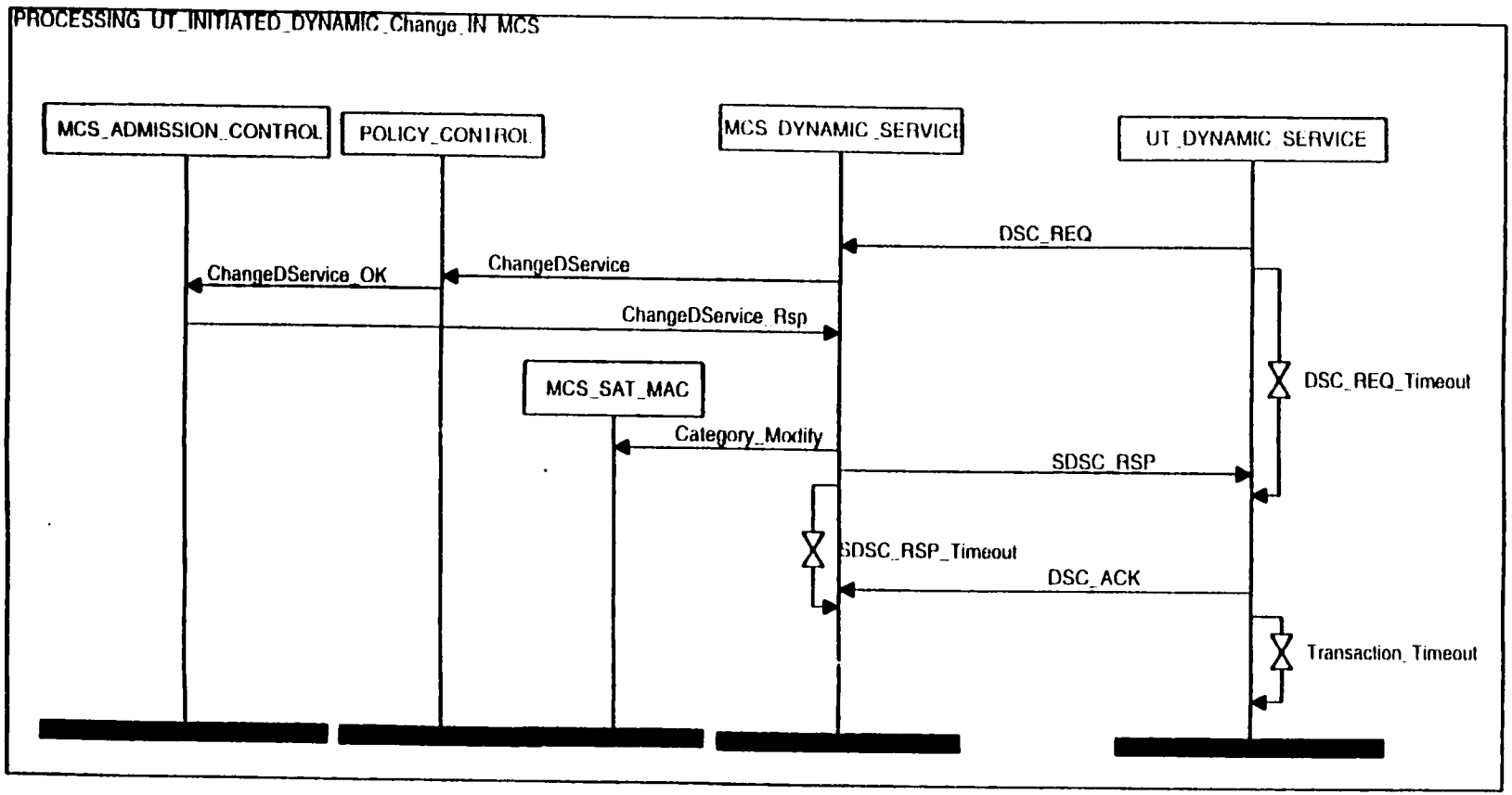

Figure 4.13 Successful UT Initiated Dynamic Change Processed in MCS-CONV

counterpart dynamic service processes follow the negotiations successfully similar to the cases explained in UT_SERVICE_DYNAMIC description.

v. **MCS_MESSAGE_DISTRIBUTER:** This process has the responsibility to create a MCS_DYNAMIC_MANAGER specific for each connected UT and to transfer the received messages from UTs to the related MCS_DYNAMIC_MANAGER. To do this a dynamic list of connected UTs and their corespondent MCS_DYNAMIC_SERVICE_MANAGER is held in this process. Upon receiving a message the sender UT and the corespondent MCS_DYNAMIC_SERVICE_MANAGER are extracted from the list and the message is forwarded to the right destination.

vii. **MCS_MESSAGE_HANDLER:** This process is responsible to unpack and pack the UT2MCS and MCS2UT messages. Receiving UT2MCS message, it unpacks the message and transfers it to MCS_MESSAGE_DISTRIBUTER. It packs all the dynamic service related messages received from MCS_DYNAMIC_SERVICE_MANAGER and MCS_DYNAMIC_SERVICE processes in MCS2UT message and pass it to the SAT_MAC block.

4.3.3 SAT-MAC

This block abstracts the behavior of satellite medium and SAT_MAC sub-layer in MCS and UT. Figure 4.14 shows that this block contains the sub-block of MCS_SAT_MAC and two sub-blocks of block type UT_SAT_MAC.

i. **UT_SAT_MAC block:** UT_SAT_MAC block type shown in Figure 4.15 abstracts two behavior of the UT_SAT_MAC: 1) It transfers, discards or transfers with delay the received UT2MCS messages to *MCS_CONV* without violation of the

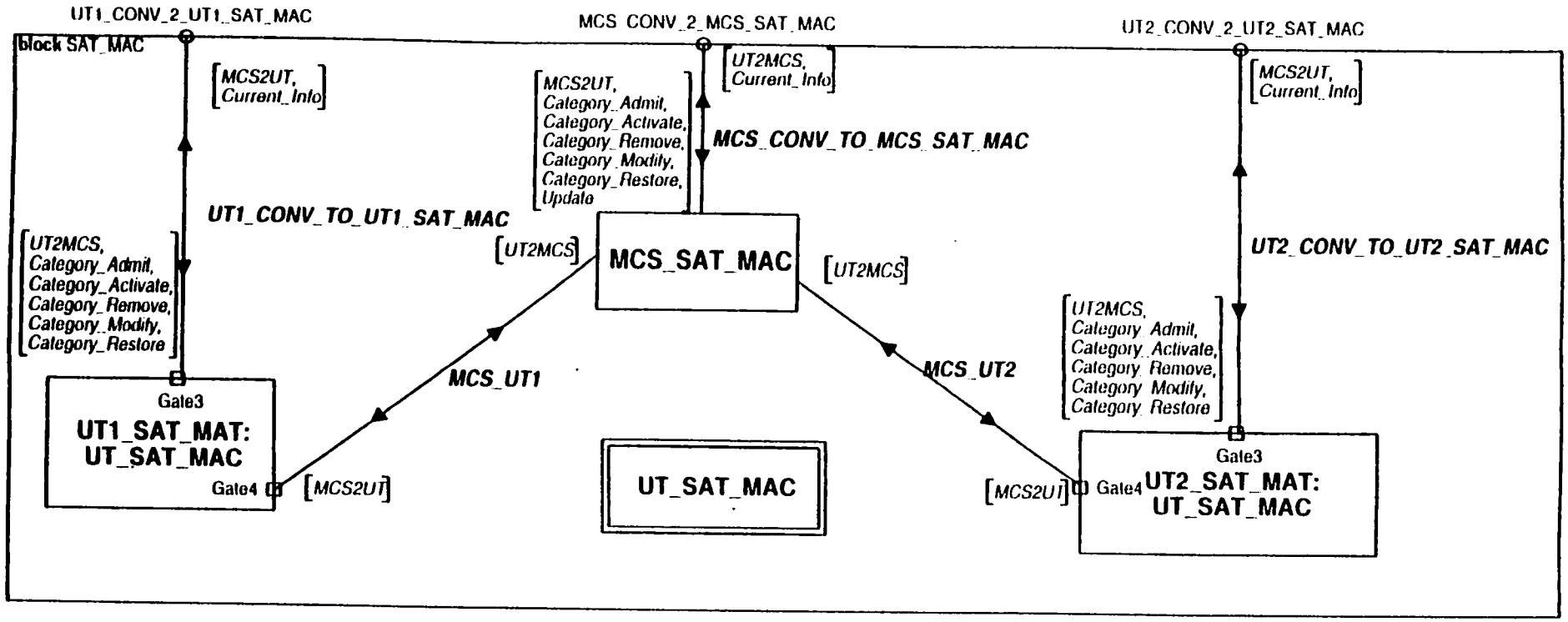

Figure 4.14 SAT-MAC Block

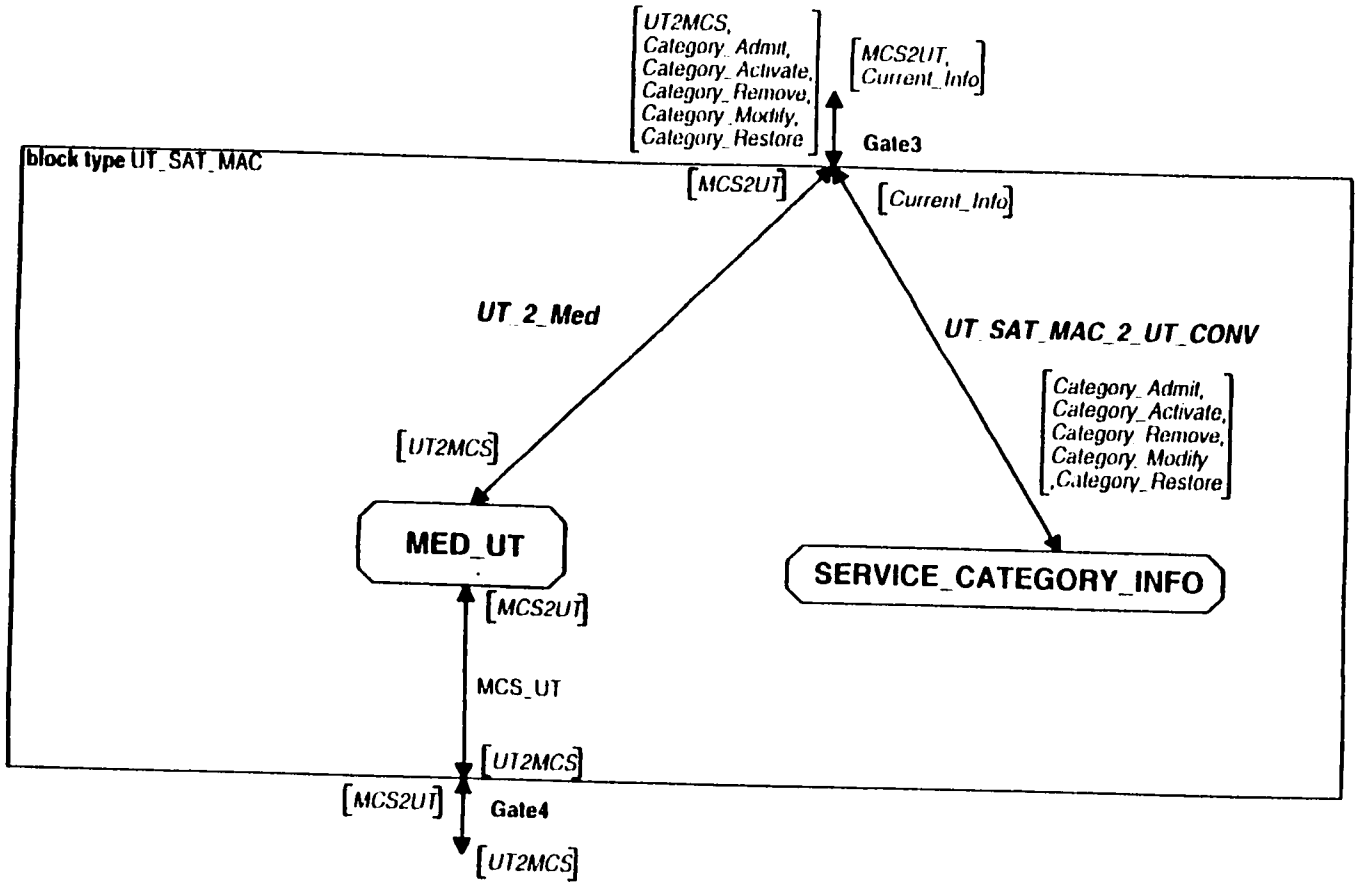

Figure 4.15 UT SAT-MAC Block Type

order of the received messages. It transfers the MCS2UT messages to UT upper layer without any delay. 2) It interacts with *UT_CONV* by receiving the information related to the service categories from *UT_CONV* and transferring the current information to the *UT_CONV* to be used in UT_ADMISSION_HANDLER. This block contains following processes:

- **MED_UT** receives UT2MCS messages in idle state and send them right away to MCS, discard them or save them in a FIFO queue and send them with a delay. To implement the FIFO queue the procedure GET_FROM_QUEUE is used. This procedure is used to get the delayed message from the head of queue and rearrange it. MED_UT passes the received MCS2UT messages without any delay to UT upper layer.

- **SERVICE_CATEGORY_INFO:** This process must hold all the updated information about each service category available in UT and their status. It receives all instantaneous information about the service categories from UT_DYNAMIC_ SERVICE process instances and sends the information about the categories status to the UT_ADMISSION_HANDLER to help in its decision making process. The depicted process only shows the messages and for simplicity does not model the book keeping and information processing.

ii. **MCS_SAT_MAC block:** MCS_SAT_MAC block type, shown in Figure 4.16, abstracts two behavior of the UT_SAT_MAC: 1) It transfers the received MCS2UT messages to *UT_CONV* without violation of the order of the received messages and delay. It passes the received UT2MCS messages right away to MCS upper layer. 2) It

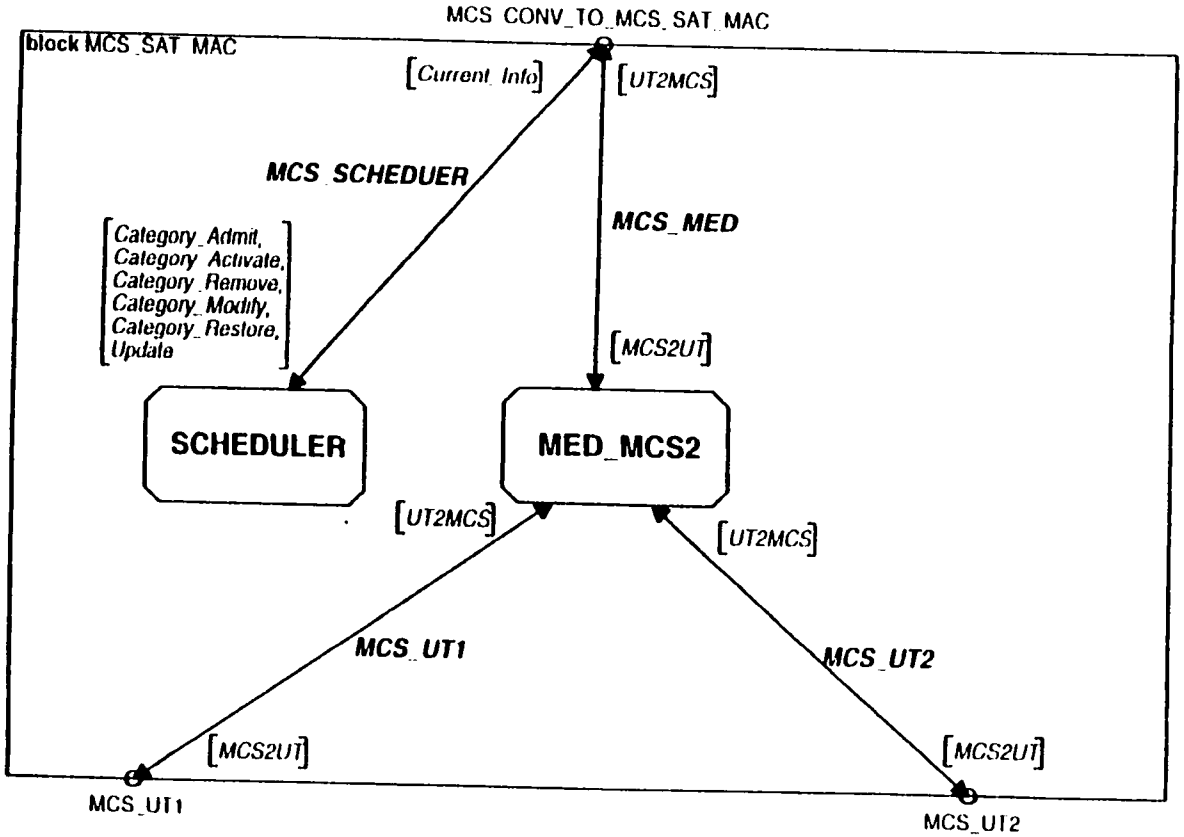

Figure 4.16 MCS SAT-MAC Block

interacts with MCS_*CONV* by receiving the information related to the service categories from *MCS_CONV* and transferring the current information to the *MCS_CONV* to be used in MCS_ADMISSION_CONTROL. This block contains two processes:

- **MED_MCS: I**t receives MCS2UT and UT2MCS messages in idle state and sends them right away to UT_CONV or MCS_CONV uplayer without any delay.

- **SCHEDULER:** In this model, we have not modeled the behavior of the scheduler. We only show the received and send messages to and from this process. This process receives updated information from MCS_DYNAMIC_SERVICE process instances and sends the current information to MCS_ADMISSION_CONTROL to help this process in its decision making.

4.4 SAT-MAC Convergence Verification and Validation

Verification and validation are so interdependent that the acronym V&V is commonly used to refer to both. A more colloquial definition from software engineering for verification and validation is " Am I building the product right?" and "Am I building the right product?", respectively.

Like any communication model, the SAT-MAC Convergence model should be verified and validated against general and specific properties, respectively. The designed model must be free of design errors such as deadlocks, unspecified receptions and livelocks. These properties are referred to as general properties. Deadlocks happen when no process can execute its transition and there is no message in the queue to be consumed. Unspecified reception occurs when a process is in a state in which it can not consume the

message located at the head of its queue. A livelock ocures when all processes are operating but there is no progress. Specific properties are the properties that cover the expected behavior of the protocols specified in the requirements.

In order to have a full confidence in the system design, the model must be verified and dynamic check must cover all of the model's behavior. The ObjectGEODE Simulator exhaustive mode makes this possible. The tool generates a complete system state graph and checks the resulting behavior against design SDL rules and expressed properties that are input to the simulation. A system state graph is composed of all states a system can reach and all possible transitions going from one system state to another.

We have verified our model against the general properties in exhaustive mode and proved that our SAT-MAC Convergence model is free of such errors. It must be noted in large models exhaustive simulation and verification needs more memory space and reaching all states takes more time. The best way to use the tool for exhaustive verification is applying it in different stages of the model implementation. For example, in implementing SAT_MAC Convergence SDL model, first we verified the model consisting of one UT, MCS and only one service category. By adding new components in each stage the system was verified against general properties. Applying exhaustive verification on complete SAT_MAC Convergence model has a sample result as follows:

Number of states: 100000

Number of transitions:257485

Maximum depth reached: 30

Maximum breath reached: 33800

Duration: 16 mn 45 s

Number of exceptions: 0

Number of deadlocks: 0

Number of stop conditions:0

Transaction coverage rate: 52.92 (194 transactions not covered)

States coverage rate:76.27 (14 states not covered)

In order to validate further our design against more specific requirements, we have chosen the most critical specific properties that cover the major portion of the protocol behavior. These properties are translated into Message Sequence Chart (MSC) [11] observers. Formal validation is achieved by running the exhaustive verification against these MSC observers. The specific properties are as followings:

i. Each User Terminal requesting for a reservation of a specific QoS eventually must receive a positive or negative answer.

To support this property, for a TC_AddFlowspec issued by a User Terminal eventually a positive or negative TC_AddFlowspec_Rsp must be received by the RSVP of the User Terminal. MSC leaves shown in Figure 4.17 and 4.18 are linked by MSC high level operator, OR, to translate this property to MSC observer.

ii. Each request for a modification of a previously reserved QoS initiated by a User Terminal must eventually be answered back by a positive or negative response.

To support this property, for a TC_ModFlowspec issued by the RSVP of a User Terminal eventually a TC_ModFlowspec_Rsp must be returned to the RSVP. To verify this property, MSC leaves shown in Figure 4.19 and 4.20 are linked by MSC operator OR to build the required MSC observer.

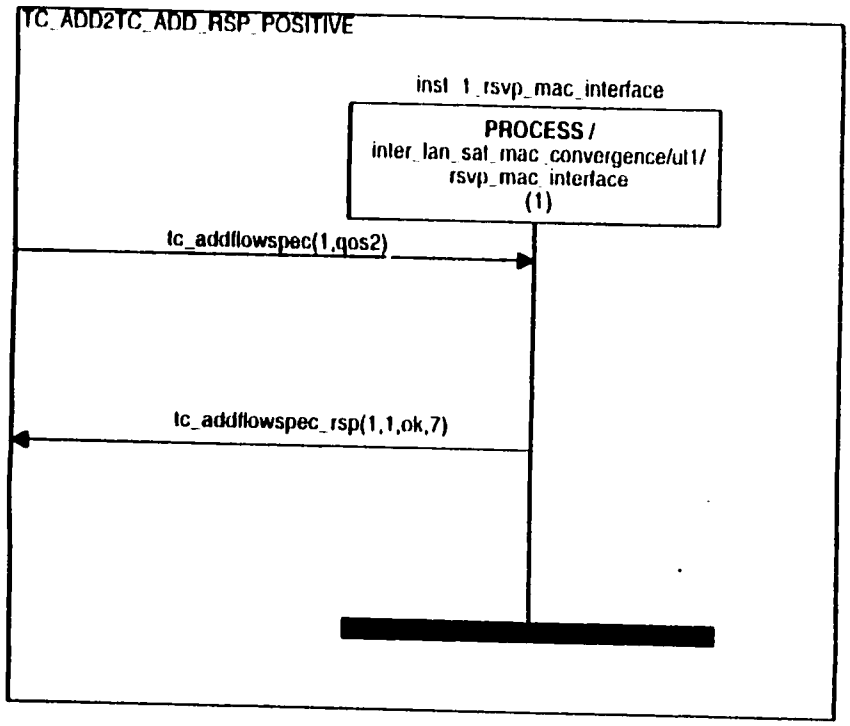

Figure 4.17 MSC Observer for Positive Response to TC_AddFlowspec

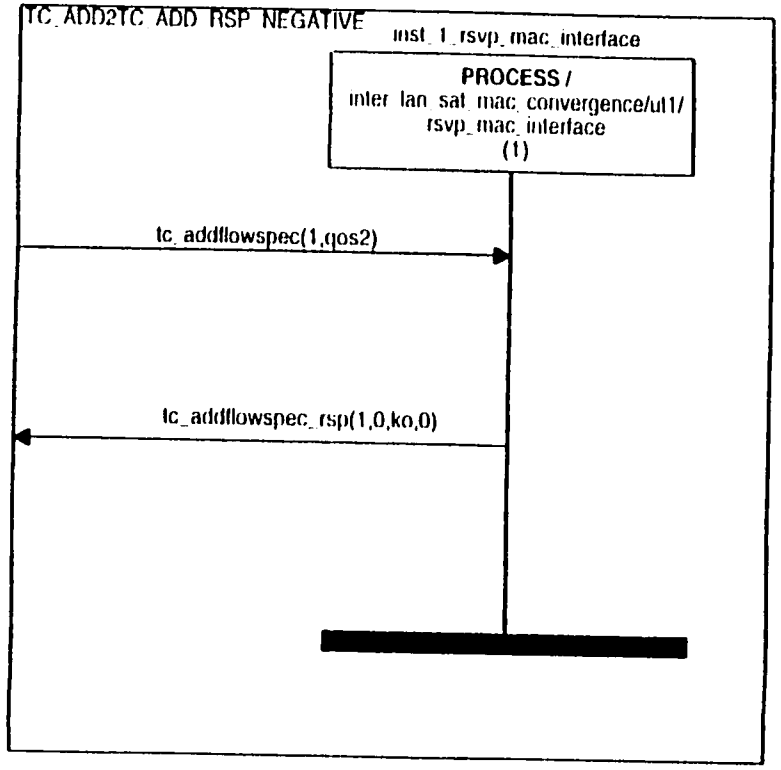

Figure 4.18 MSC Observer for Negative Response to TC_AddFlowspec

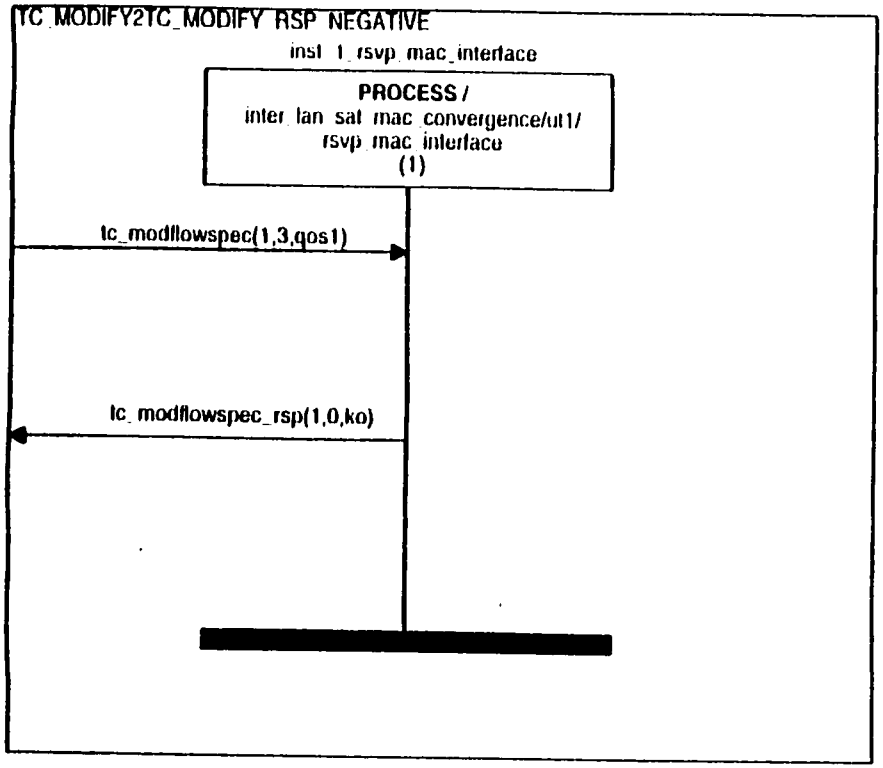

Figure 4.19 MSC Observer for Negative Response to TC_ModFlowspec

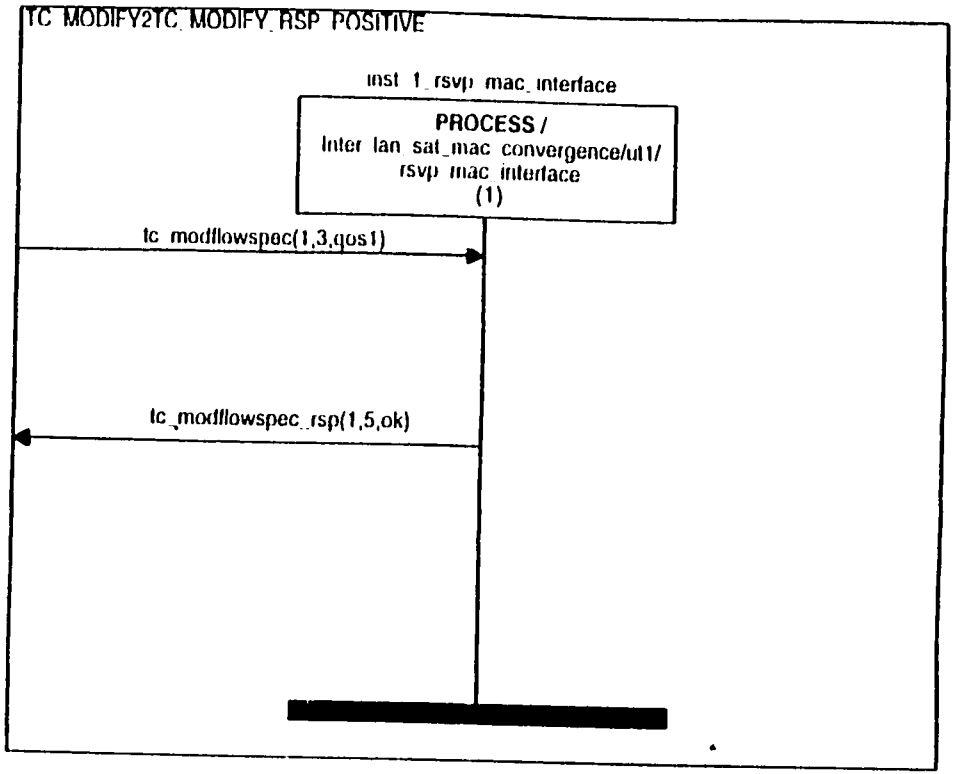

Figure 4.20 MSC Observer for Positive Response to TC_ModFlowspec

iii. Each User Terminal always must be able to get the information about the available network resources before sending its reservation request for a specific set of QoS parameters.

To support this property for each *TC_Advertise* request received from RSVP must always a *TC_Advertise_Rsp* be produced. The MSC observer for this property is shown in Figure 4.21.

iv. To alleviate the processing load in MCS during dynamic service negotiation between UT and MCS, each reservation request for a specific set of QoS initiated by a UT must always be translated to a specific service category.

To support this property a received TC_AddFlowspec for a specific set of QoS must be translated to a Creserve_Req for a specific service category and an amount of frequency-time slot. The MSC observer for this property is depicted in Figure 4.22.

v. For alleviation of the processing load in MCS, User Terminals may aggregate the requests for modification.

Two MSC leaves depicted in Figure 4.23 are linked by OR operator to translate this property.

All derived properties were verified by running the verification process against each equivalent MSC observers. Non of the properties were violated. This concludes the correctness of our structure.

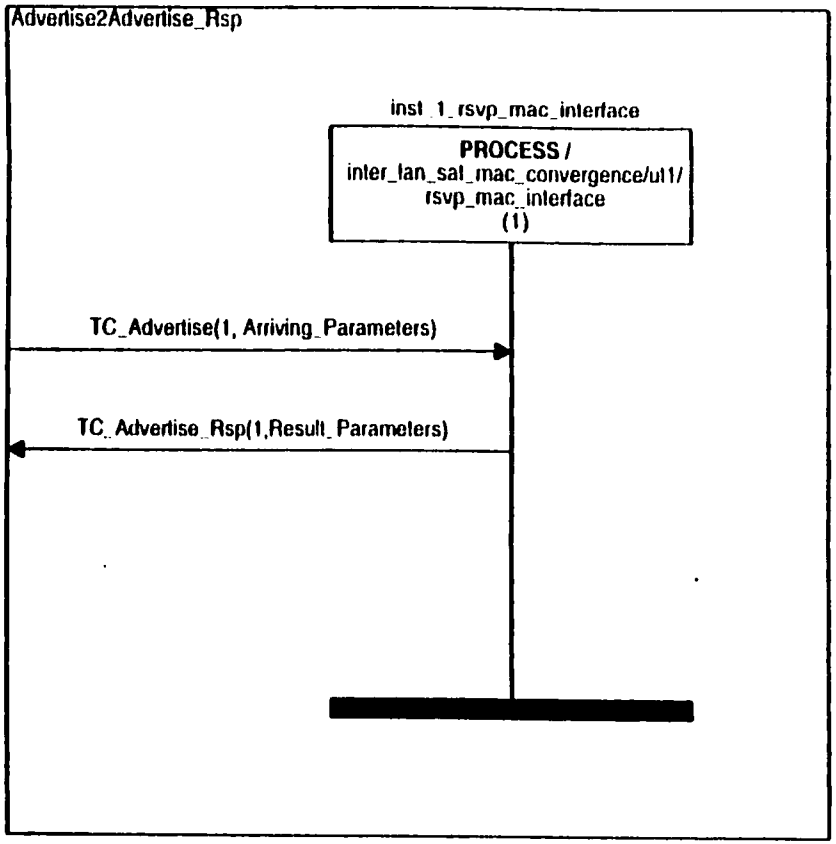

Figure 4.21 MSC Observer for Receiving The Information about the Available Network Resources

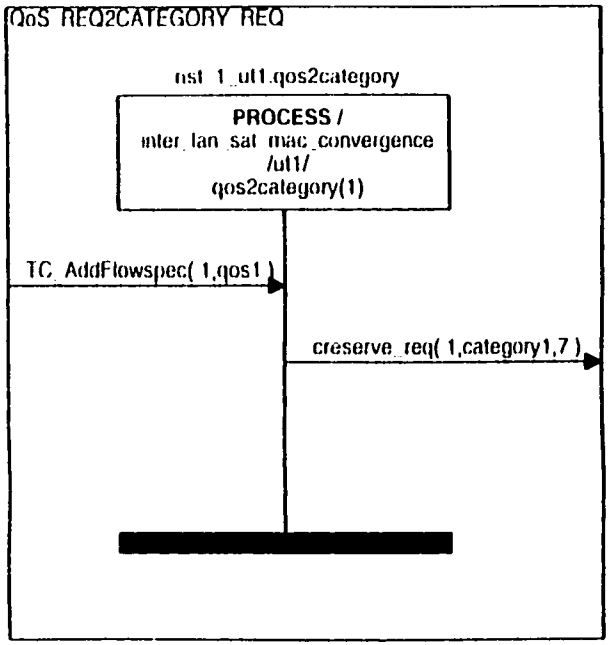

Figure 4.22 MSC Observer for Translation of QoS to Service Category

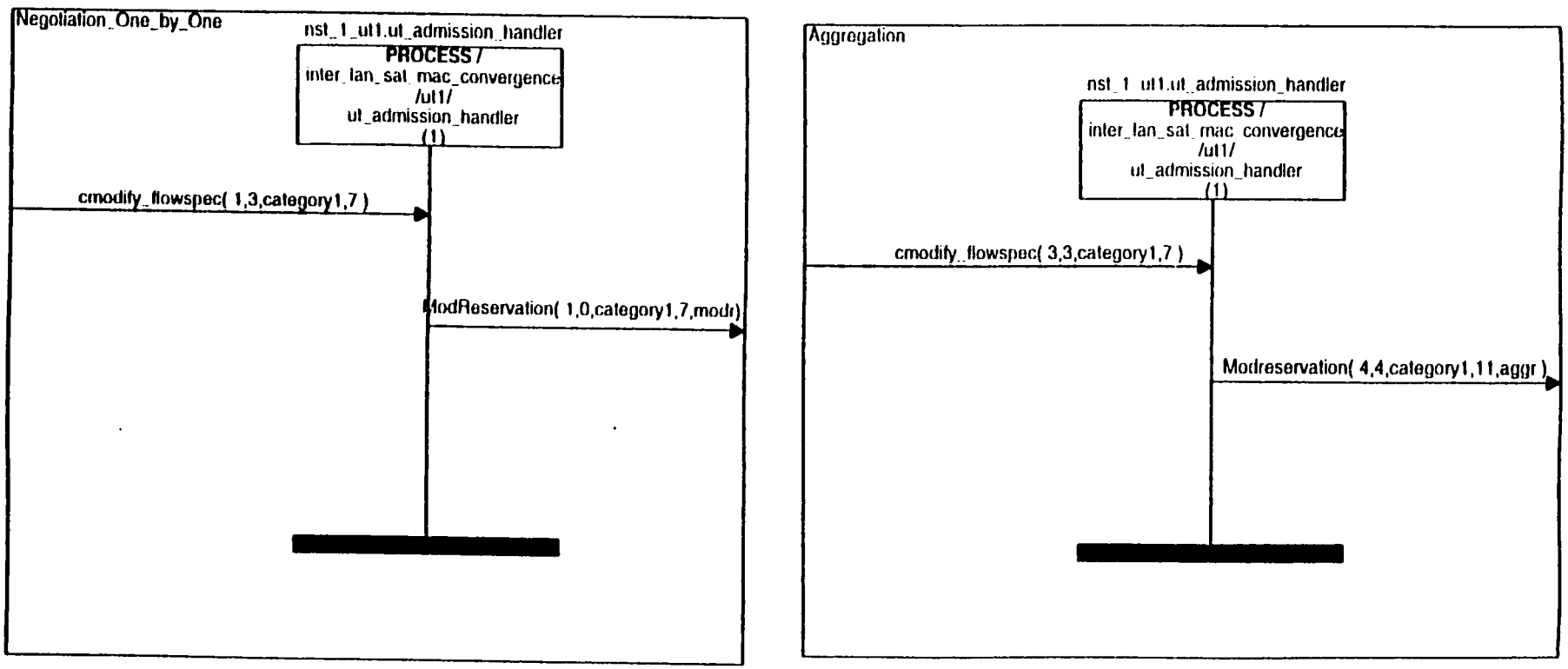

Figure 4 23 MSC Observer for Aggregation of Modification Requests

4.5 SAT_MAC Detailed Model

An abstract model for SAT_MAC block was provided in our model. To provide a detailed model for SAT_MAC sub-layer we can adapt the MAC model proposed for Broadband Satellite Access (BSA) system in [12] with some modifications. The following required modifications stem from the different system configuration:

i. BSA system constitutes of three main stations: 1) Base Transceiver Station (BTS) which represents the gateway to the backbone network. Subscribers information and registration files to be downloaded during the Subscriber Transceiver Station (STS) initialization are loaded over there. 2) STS which is connected to the user premises equipment. 3) MCS which regulates the access to the satellite medium.

In Inter-LAN connections over Broadband Satellite Systems we do not need any BTS. Therefore, in Inter-LAN system the initial registration is completely done by MCS while in BSA the BTS and MCS are both involved. DHCP and TFTP in Inter-LAN are at the MCS while in BSA they are at the BTS.

ii. In Inter-LAN system power-on is used only in initialization phase but in BSA it is used quite often.

iii. In BSA the MAC destination address (next hop) in upstream can be implicitly specified. Therefore, an independent destination field in the MAC header is not necessary. In Inter-LAN the destination addresses must be explicitly added. In Inter-LAN, the mapping of IPv6 addresses is accomplished by neighbor discovery procedures [16], a compromise between Address Resolution Protocol (ARP) and End System to

Intermediate System (ES-IS) routing protocol. In the BSA model this address mapping is not covered but we can assume the model is provided with these address mapping by ARP and ES-IS protocols in case of Inter-LAN.

iv. Concatenation in BSA and Inter-LAN is different. In BSA the next hop is the same: thus, implementing the concatenation is simple. In Inter-LAN the next hop is not the same. We must search the data queue, choose the packets destined to the same hop and concatenate them.

v. In BSA to send data from STS to STS it must pass through BTS but in Inter-LAN we can send data directly from UT to UT.

Applying these modifications to the proposed MAC protocol in [12], a detailed model for SAT-MAC sub-layer in Inter-LAN connections over broadband satellite systems will be achieved.

Chapter 5

Conclusions

In this thesis, we analyzed the advantages of the satellite technology in building Inter-LAN connections among remote LANs. The broadcast nature of satellite networks and their ability to support mesh configuration allow for cost effective deployment of broadcast/multipoint services. Feasibility of global connectivity over single hop to provide coverage for wide areas removes the switching, routing and sequencing problems experienced by other technologies. Fast configuration/re-configuration, distant-insensitive connectivity, capability of flexible bandwidth on demand, which is critical for end-to-end QoS provisioning, and having the characteristics of a truly private business network are among those interesting advantages. However, satellite system faces some challenges stemmed from its longer delay and its power and bandwidth constraints.

We highlighted the challenges stemmed from satellite constraints in providing QoS guarantees for bursty and delay sensitive real-time traffic. We reasoned that to provide Inter-LAN services, to support various QoS requirements for real-time and non-real-time services and to efficiently utilize the shared bandwidth of the satellite a special MAC and DCA is required.

We described the configuration of Business Private Networks and proposed a protocol architecture with emphasis on the MAC layer to support different QoS requirements over satellite communications. Based on the CFDAMA and MF-TDMA techniques, the

functions of the UT and MCS to provide an efficient traffic scheduling and dynamic allocation strategy were explained.

We proposed the use of RSVP and IPv6, as an illustrative example, on top of the MAC layer and devised an architecture in MAC protocol to give services to RSVP in order to provide the QoS guarantees.

A structure for translation of the QoS requirements requested by RSVP to service categories and an amount of correspondent frequency-time slots was introduced. Using this method, the processing loads are distributed among the UTs and MCS.

We devised a structure in UT to support the aggregation of requests for further alleviation of the processing loads in MCS. This characteristic enables UTs to ask for a group of requests. Furthermore, we proposed a mechanism to negotiate these requests between UTs and MCS through a suitable architecture for policy and admission control.

The building blocks of MAC layer structure, their functionality and relations were explained in details. Furthermore using MSC the interaction of SAT-MAC and SAT-MAC convergence sub-layer between UTs and MCS for initialization and dynamic service negotiation were presented.

Using SDL, we have provided a formal model for the proposed MAC protocol and verified its correctness against general and specific properties using the ObjectGEODE tool. Modeling and verifying the proposed MAC architecture proved the suitability, flexibility and capability of SDL and ObjectGEODE and related tools for unambiguous and consistent descriptions of large and challenging protocols.

Future work will focus on modeling the prediction and traffic monitoring processes to include the suitable micro and macro scheduling strategies in the proposed MAC architecture.

REFERENCES

[1] RFC 2460: Internet Protocol Version 6 (IPv6) Specification. Dec 1998.

[2] RFC 2205: RSVP Resource Reservation Protocol. RSVP Version 1. Functional Specification. Sept. 1997.

[3] T. Le-Ngoc. "Dynamic Resource Allocation Schemes for Multimedia Satellite Communications". *The 4th International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC' 93)*. Yokohama. Japan. Sept. 8-11.1993. pp.552-556.

[4] ITU-T. "Specification and Description Language (SDL)". Z.100. Geneva. 1996.

[5] Verilog. ObjectGEODE. Toulouse. France. 1996.

[6] T. Le-Ngoc. S.V. Krishnamurthy. "Performance of Combined Free/Demand Assignment Multiple-Access Schemes in Satellite Communications". *International Journal of Satellite Communications*. VOL.14. 1996. pp.11-21.

[7] K. J. Turner. Using Formal Description Techniques. An Introduction to Estelle. Lotos and SDL. John Wiley & Sons. 1993.

[8] RFC 2210: The Use of RSVP with IETF Integrated Services. Sept. 1997.

[9] RFC 2215: General Characterization Parameters for Integrated Service Network Elements. Sept. 1997.

[10] DOCSIS: Data -Over-Cable Service Interface Specifications. Radio Frequency Interface Specification, SP-RFIv1.1-101-990311, March 1999.

[11] ITU-T "Recommendation Z.120. Message Sequence Charts (MSC)", Geneva, 1996.

[12] T. Elshabrawy, MAC Architecture for Broadband Satellite Access Systems, M.A.Sc Thesis, Concordia University, April 2000.

[13] T. Le-Ngoc, I.J. Mohammed, "Performance Analysis of CFDAMA-PB Protocol for Packet Satellite Communications", *IEEE Transactions on Communications*, Vol. 46, No.9, September 1998, pp.1206-1214.

[14] H. Peyravi, " Medium Access Control Protocols Performance in Satellite Communications ", *IEEE Communications Magazine*, March 1999, p.62-70.

[15] D.P Connors, B. Ryu, S. Dao, " Modelling and Simulation of Broadband Satellite Networks- Part 1: Medium Access Control for QoS Provisioning ", *IEEE Communications Magazine*, March 1999, p.72-79.

[16] C.Huitema, IPv6, the new internet protocol, Prentice Hall PTR, 1998, p.138.

[17] A.Olsen, O.Faergemand, B.M.Pedersen, R.Reed, J.R.W.Smith.System Engineering Using SDL-92, Elsevier Science B.V, 19994.

[18] J.Ellsberger, D.Hogrefe, A.Sarma, SDL Formal Object-oriented Language for Communicating Systems, Prentice Hall, 1997

[19] M. Hine, C. Adams, J. Burren, N. Celandroni, E. Ferro, O. Koudelka, W. Riedler, T. Waibel, "SATINE-2: an experiment in high speed wide area networking. Part 1: overview," *Proceedings of the European Teleinformatics Conference- EUTECO '88*, 1988, p.577-605

[20] N. Celandroni, E. Ferro, "FODA-TDMA satellite access scheme: description, implementation and environment simulation ", *Proceedings of the Third Tirrenia International Workshop*, 1988, p.353-61

[21] N. Celandroni, E. Ferro, " Fade Detection for the FODA-TDMA Access Scheme", *Proceedings of the Olympus Utilization Conference*, 1989, p.177-184

[22] N. Celandroni, E. Ferro, " Protocol between the FODA system and the outside environment", *CNUCE Report*, August 1990, C90-03

[23] N. Celandroni, E. Ferro, " The FODA-TDMA satellite access scheme: presentation, study of the system and results ", *IEEE Transactions on Communications, December 1991*

[24] C. Garrido, P. Viau, J. Stjernevi, " Computer Networking via High-Speed Satellite Links ", *ESA bulletin*, November 1990, pp.98-101,

[25] COST 226 Project, " Integrated Space/Terrestrial Networks ", *First Annual Report, EUCO-COST*, Brussels, Sept. 1991

[26] O. Koudelka, R.A. Harris, " LAN interconnection by satellite and the Cost226 project", Computer Networks and ISDN systems, June 1991, p.285-92, vol.21, No.4

[27] A. Ito, S.Kato, H.Mastsumoto, S. Fujii, " Multiple Access- Type Satellite Packet Network (MASPnet) - An Experiment for a 'LAN type' Wide Area Network ", Journal of the Communications Research Laboratory, Japan, March 1991 p.7-21

[28] K. et. al. Kobayashi, " A Satellite LAN Interconnection System ", *SBT IEEE Intl. Telecommunications Symposium, ITS'90*, 1990, p.17.4.1-5

[29] E.R Cacciamani, C.A. Politi, " VSATs take LANs airborne ", Networking Management, Nov.1991, p.55-58

[30] C.A., Politi, J.A Stein, "VSATs Give Corporate Networks a Lift ", Data communication, Feb.1991, p.89-94

[31] Yang, O.W.W; Yao, X.X., Murthy, K.M.S. " Modeling and Performance Analysis of File Transfer in a Satellite Wide Area Network," *IEEE Journal on Selected Areas in Communications*, 1992, p.428-36

Appendix A

An Introduction to RSVP

As we move further into the age of multimedia communications, many real-time applications are being developed that are delay sensitive to the point where the best effort delivery model of IPv4 can be inadequate even under modest network loads. The necessity to provide many applications with additional service classes offering enhanced quality of service with regard to the bandwidth, packet queuing delay, and loss has led to the introduction of several reservation protocols. These reservation protocols are supposed to enable the senders, receivers and routers along the path to communicate with each other in order to set up the necessary network nodes' states to support the required QOS.

The most important IP reservation protocols are RSVP, ST-II, ST-II+ and Differentiated Services. In this appendix the RSVP design goals and principles are explained. A detailed explanation of RSVP messages and their parameters are introduced. RSVP interface to the link layer is explained.

A.1 RSVP Design Goals

RSVP has been designed according to the several goals:

i. **Unicast and Multicast capabilities:** RSVP has been designed to operate with current and future unicast and multicast routing protocols. RSVP is not a routing protocol: it is used to reserve resources along the route whichever underlying routing protocol is in place. RSVP is only concerned with the QOS of those packets that are forwarded in accordance with routing. RSVP is able to cope with the resulting routing changes and it is able to reestablish automatically the resource reservation along the new paths as long as adequate resources are available.

ii. **Heterogeneous receivers support:** In a wide area inter-network, receivers, as well as the paths used to reach the receivers, can have very different properties from one another. RSVP has been designed to provide the ability for heterogeneous receivers to make reservations specifically tailored to their own needs. In order to accommodate efficiently large groups, dynamic membership, and heterogeneous receiver requirements, RSVP makes receivers responsible for requesting a specific QOS.

iii. **Dynamic membership in a multicast group:** The membership in a multicast group can be dynamic. The basic proposal would have to reinitiate the reservation protocol every time a new member joined or an existing member left the multicast group. Reinitiation of the reservation protocol is particularly burdensome for large group because the larger the group size, the more frequent are the changes in the group

membership. So one of the design goals of RSVP has been to deal gracefully with changes in the multicast group membership.

iv. **Source and sub-stream filtering capabilities**: A receiver should be able to control which packets are carried on its reserved resources, not only what gets displayed on its local screen. RSVP provides this characteristic.

v. **Efficient use of resources**: RSVP has been designed to allow end users to specify their application needs so that the aggregate resources reserved for a multicast group can more accurately reflect the resources actually needed by that group.

vi. **Protocol overhead limitation:** The RSVP protocol carries the reservation request to all the nodes (routers and hosts) along the reserve data paths to the data sources, but only as far as the router where the receiver's data path joins the multicast distribution tree. AS a result RSVP reservation overhead is in general logarithmic rather than linear in the number of receivers.

vii. **Modularity:** RSVP has been designed in a way that it is independent of the routing and underlying traffic and admission control algorithm. This design makes RSVP deployable in many contexts.

A.2 RSVP Design Principles

In order to reach the previous goals the following design principles have been adopted:

i. **Receiver initiated reservation :**

- A reservation is requested, maintained and torn down by the receiver(s).

- Each receiver decides the proper amount of resources to reserve for its QOS needs.

- Receivers of a multicast group can select different levels of QOS.

- It adapts to multicast reservation requests and to heterogeneity of receivers.

ii. Soft-state:

- It is opposite to hard-state choice typically used in connection oriented networks.

- Path and reservation states stored in routers along the path expire after a given time interval and need periodical refresh before they time-out.

- Periodical refresh is performed by host(s).

- The choice of soft-state adapts to dynamic changes of routing paths.

- It is also coherent with the Internet philosophy of moving complexity towards the end systems.

iii. Reservation styles and merging :

- It defines how reservations for the same multicast group are aggregated at intermediate routers.

- This is done for efficient resource usage and protocol overhead control.

iv. Opaque information transport :

- RSVP ignores the contents of its messages (modularity). It means that the contents of the messages are opaque to the RSVP itself. Its messages only carry this information.

v. **Independence from underlying routing protocol:**

- RSVP is NOT a routing protocol, since it relies on underlying protocols to perform routing (e.g. IPv4 or IPv6)

- This allows datagrams to be transparently treated in non-RSVP routers (but QOS cannot be assured in this case)

- Routing policy does not necessarily choose a path containing only RSVP routers, even if such a path exists. This choice is motivated in order to keep existing standards unchanged.

A.3 RSVP Procedures

RSVP requests resources for simplex flows, i.e. it requests resources in only one direction. Therefore, RSVP treats a sender as logically distinct from a receiver, although the same application process may act as both a sender and receiver at the same time. It must be noted that in this introduction we will use "upstream" as the direction from receiver to Sender and "downstream" vice versa. "Previous hop" is the node after the current node on the downstream and "next hop" is the next hop to the current hop on the upstream. All the parameters are explained in details in the section "RSVP Messages".

Resource reservation setup using RSVP requires the following steps:

i. The Receiver host sends IGMP messages to the Sender host to join a unicast or multicast group.

ii. Receiving IGMP messages, the sender host application will request Application-RSVP interface for a session. This request leads to a SESSION call which initiates RSVP processing for a session, defined by DestAddress together with Protocol Id and possibly a port number DestPort. RSVP will answer to this call by returning a local session identifier Session_Id, to the application process. This Session_Id will be used in subsequent calls, to the application process.

iii. After getting Session_Id, the application process sends the sender information using the identified session to the sender's Application-RSVP interface.

iv. Sender's Application-RSVP interface uses SENDER call to define the attributes of the data flow. The first SENDER call for the session registered as 'Session-Id' will cause RSVP to begin sending Path messages for this session. This call will carry Source_Address, Source_Port, Sender_Template, Sender_Tspec, Adspec, Data_TTL.

v. After getting the first SENDER call for the registered Session-Id, RSVP will send a path message to the next RSVP hop. Path message contains a SENDER_TEMPLATE object defining the format of the data packets and a SENDER_TSPEC object specifying the traffic characteristics of the flow. Optionally, it may contain an ADSPEC object carrying advertising data for the flow.

The sender host will initialize the ADSPEC parameters as the initial-Adspec and will send them by path message to the next RSVP node.

vi. When a RSVP node receives a Path message:

a. It creates or updates a path state associated to the corresponding session, containing Phop.Sender_Tspec and Adspec.

b. It restarts the cleanup timer associated to this path state (i.e. refresh).

c. It updates Phop and Adspec objects and then forwards the Path message towards the next node. To update the Adspec parameters, RSVP must negotiate with the link layer through the RSVP-MAC layer interface. RSVP uses TC_Advertise call to stimulate the RSVP-MAC layer interface for collecting the current information related to Adspec parameters from the MAC layer. TC_Advertise carries the 'arriving' Adspec information from the previous node to the RSVP-MAC interface. RSVP-MAC interface collects the 'current' Adspec information from the MAC layer using Adspec_Param_req and Adspec_Param_Rsp and submits them to the RSVP-MAC interface. This interface uses some well defined routines to modify the Adspec. The result Adspec will be sent to RSVP through TC_Advertise_Rsp. RSVP will send the resulted Adspec parameters in Path message to the next node.

d. This scenario is repeated along the route from the sender host to the receiver host.

It must be noted that Senders can explicitly initiate a PathTear (path teardown) message, which will travel downstream towards all receivers. PathTear message can be initiated by path state timeout in any node. Receipt of a PathTear message deletes matching path state.

Errors in processing the Path messages will cause a PathErr message. This message travels upstream towards the senders and reports the error to the senders application.

vii. After receiving the first path message, RSVP in Receiver host will initiate an UPCALL indicating a PATH_EVENT to the receiver host application which means there is at least one active sender. The UPCALL will be transferred to the application process through the RSVP-Application interface.

viii. Upon receiving the UPCALL indicating a PATH-EVENT, the receiver host sends a RESERVE call to the Application-RSVP interface. A receiver uses this call to make or to modify a resource reservation for the session registered as session-id. The first RESERVE call will initiate the periodic transmission of Resv messages. A later RESERVE call may be given to modify the parameters of the earlier call.

ix. RSVP-Application interface will be stimulated by the RESERVE call to initiate a Resv message. The Resv message will carry reservation requests hop-by-hop from receivers to senders, along the reserve paths of data flow for the session. This message will contain the Receiver's desired QoS and Traffic parameters which are determined by analyzing the total "arriving" parameters in the Path message.

x. When a RSVP node receives a Resv message the following steps are necessary for a reservation setup:

During reservation setup an RSVP QoS request is passed to two local decision modules, "admission control" and "Policy control".

a) Policy control determines whether the user has administrative permission to make the reservation. If the policy control check fails the reservation setup is impossible.

b) After succeeding in policy control check. admission control determines whether the node has sufficient available resources to supply the requested QOS. To make the reservation. RSVP sends TC_AddFlowspec to the RSVP-MAC interface. This message carries TC_Flowspec. which defines the desired effective QOS to the admission control.

c) RSVP-MAC interface negotiates with MAC layer. which contains admission control. and provides the TC_AddFlowspec_Rsp to the RSVP. If it is successful it means that admission control has committed to reserve the requested QOS.

d) If both checks (policy and admission control) succeed. parameters are set in the packet classifier and in the packet scheduler to obtain the desired QoS. If either check fails. the RSVP program returns an error notification to the application process that originated the request.

e) It sets the reservation state and restarts the cleanup timer associated to the Resv state (i.e. refresh).

f) It forwards Resv message to the previous node of the path (Phop).

This scenario is repeated along the route from the receiver host to the sender host.

It must be noted if Resv message carries a RESV_CONFIRM parameter each RSVP after successful reservation will send a ResvConf message to the reserve initiating receiver host. If there is an error in processing Resv messages or a spontaneous disruption of a reservation, ResvErr message will report the error to the receivers by travelling downstream towards the appropriate receiver. Furthermore. a ResvTear message can be initiated explicitly by receivers or by any node in which the reservation state has timed

out. It travels upstream and receipt of a ResvTear message deletes matching reservation state.

xi. Upon receiving the UPCALL indicating a RESERVE-EVENT, the sender host will start to send the data.

A.4 RSVP Messages

In this section RSVP messages and their parameters are explained in details. RSVP has 7 messages: Path, Resv, PathTear, PathErr, ResvTear, ResvErr, ResvConf.

A.4.1 Path message: Path(Session, Phop, Time_Values, Policy_Data, Sender_ Template, Sender_Tspec, Adspec)

i. Each sender host periodically sends a path message for each data flow that it originates.

ii. It is sent downstream towards receiver(s).

iii. It provides information about sender(s) Tspec.

iv. It creates path states in each RSVP node along the path.

v. It provides end-to-end path characteristics using Adspec parameters. Adspec in each network element is modified, then it is passed to the next RSVP node downstream. Path message coming from the previous hop contains the "arriving values" in Adspec. Each hop contains "local values" corresponding to Adspec parameters. In each hop, using several composition rules in the RSVP interface the

"result" values of Adspec is calculated and passed to the next hop through the Path message.

Path message contains the following objects:

i. **Session:**

- A session is defined as the destination of a data flow.

- A session is identified by the destination IP address (DestAddress), destination port and IP protocol type identifier (e.g. UDP or TCP).

ii. **TIME_VALUES:** It shows the value for the refresh period.

iii. **Phop:** Address of previous RSVP node .It also carries a logical interface handle (LIH).

iv. **POLICY_DATA:** It carries the Information that will allow a local policy module to decide whether an associated reservation is administratively permitted.

v. **SENDER_TEMPLATE:** A sender IP address and some demultiplexing information to identify a sender

vi. **SENDER_TSPEC(r,b,p,m,M):** The RSVP SENDER_TSPEC object carries information about a data source's generated traffic. This TSPEC carries traffic information usable by either the Guaranteed or Controlled-Load QoS control services.

This TSPEC contains: **Token bucket rate (r)** and **Token bucket size (b)** which are set to reflect the sender's view of its generated traffic. **Sender's peak traffic generation rate**

(p) may be set to the sender's peak traffic generation rate (if known or controlled), the physical interface line rate(if known), or positive infinity (if no better value is available). **Minimum policed unit parameter (m)** should generally be set equal to the size of the smallest packet generated by the application. This packet size includes the application data and all protocol headers at or above the IP level. The size given does not include any link level headers, because these headers will change as the packet crosses different portions of the inter-network. The [m] parameter is used by nodes within the network to compute the maximum bandwidth overhead needed to carry a flow's packets over the particular link-level technology, based on the ratio of [m] to the link level header size. This allows a correct amount of bandwidth to be allocated to the flow at each point in the net. A [m] value of zero is illegal. A value of zero would indicate that no data or IP header is present, and would give an infinite amount of link level overhead. **Maximum packet size (M)** should be set to the size of the largest packet the application might wish to generate. This value must be equal or larger than the value of [m].

vii. Adspec (Default General parameters fragment (always present), Guaranteed Service Fragment (Present if application might use Guaranteed Service), Controlled-Load Service Fragment (present if application might use Controlled-Load service)**):**

Adspec carries information needed by receivers to choose a service and determine the reservation parameters. The sender application constructs an initial RSVP ADSPEC object. This Adspec carries information about the QOS control capabilities and requirements of the sending application itself, and forms the starting point for the accumulation of the path properties. The Adspec is modified by subsequent network nodes as the RSVP Path message moves from the sender to the receiver(s). At each

network element the Adspec is passed from RSVP to the traffic control module (in MAC layer). The traffic control module updates the Adspec, which may contain data for several QOS control services, by identifying the services mentioned in the Adspec and calling each such service to update its portion of the Adspec. If the traffic control module discovers a QOS control service mentioned in the Adspec but not implemented by the network element, a flag is set to report this to the receiver. The updated Adspec is then returned to the RSVP for delivery to the next hop along the path.

Data in the Adspec is divided into blocks or fragments, each of which is associated with a specific service.

a) Default General parameters fragment (per-service-header, Global Break bit, Number-IS-hops, Available-Bandwidth, Minimum-path-latency, composed path MTU)

- **per-service-header or service number**: It indicates the type of services and for Default General Fragment is 1.

- **Global Break bit:** This parameter provides information about the presence of network elements which do not implement QOS control services along the data path. The local value of the parameter is 1 if the network element does not implement the relevant QOS control service or knows that there is a break in the chain of elements which implement the service. The local parameter is 0 otherwise. The composition rule for this parameter is the OR function. A composed parameter value of 1 arriving at the end of a path indicates that at least one point along the path does not offer the integrated services.

- **Number-IS-hops**: IS stand for "integrated services aware". In each node of the path this parameter will be incremented if that node is integrated-service aware.

- **Available-Bandwidth**: This parameter provides information about the bandwidth available along the path followed by a data flow. The local parameter is an estimate of the bandwidth the network element has available for packets following the path. Computation of the value of this parameter should take into account all information available to the network element about the path, taking into consideration administrative and policy controls on bandwidth, as well as physical resources. This parameter should reflect as closely as possible, the actual bandwidth available to packets following a path. In circumstances where the parameter can not be provided accurately, the network element should make the best attempt is possible, but it is acceptable to overestimate the available bandwidth by a significant amount. The composition rule for this parameter is the MIN function. The composed value is the minimum of the network element's value and the previously composed value. This quantity when composed end-to-end, informs the endpoint of the minimal bandwidth link along the path from the sender to receiver.

- **Minimum-path-latency**: The local parameter is the latency of the packet forwarding process associated with the network element, where the latency is defined to be the smallest possible packet delay added by the network element. This delay results from speed-of-light propagation delay, from packet processing limitations or both. It does not include any variable queuing delay, which may be present. The purpose of this parameter is to provide a baseline minimum path latency for use with services which provide estimates or bounds on additional path delay, such as Guaranteed services. Note that the quantity characterized by this parameter is the absolute smallest possible value for the

packet processing and transmission latency of the network element. This value is the quantity required to provide the end hosts with jitter bounds. The parameter does not provide an upper-bound estimate of minimum latency, which might be of interest for best effort traffic and QOS control services which don't explicitly offer delay bounds. In other words, the parameter will always underestimate, rather than overestimate, latency, particularly in multicast and large cloud situations.

- **Composed path MTU**: This parameter computes the maximum transmission unit (MTU) for packets following a data path. This value is required to invoke QOS control services, which require that IP packet size be strictly limited to a specific MTU. The local characterization parameter is the IP MTU, where the MTU of a network element is defined to be the maximum transmission unit the network element can accommodate without fragmentation, including IP and upper-layer protocol header but not including link level headers. The composition rule is to take the minimum of the network element's MTU and the previously composed value. This quantity, when composed end-to-end, informs the endpoint of the maximum transmission unit that can traverse the path from sender to receiver without fragmentation.

b) Guaranteed Service Fragment(Per-service-header, Break-bit, Composed Ctot, Composed Dtot, Composed Csum, Composed Dsum)

- **Per-service-header**: For Guaranteed Service Fragment is 2.

- **Break-bit:** If the network element implements the Guaranteed Service the local value will be 0 otherwise it will be 1. The composition rule for the Break-bit is OR function.

- C and D parameters: The guaranteed service uses the RSVP Adspec to carry data needed to compute the C and D terms passed from the network to the application. The error term C is the rate–dependent error term. It represents the delay a datagram in the flow might experience due to the rate parameters of the flow. An example of such an error term is the need to account for the time taken serializing a datagram broken up into ATM cells, with the cells sent at a frequency of 1/r. The error term C is mesured in units of bytes.

The error term D is the rate-independent, per-element error term and represents the worst case non-rate-based transit time variation through the service element. An example of D is a slotted network, in which guaranteed flows are assigned particular slots in a cycle of slots. Some part of the per-flow delay may be determined by which slots in the cycle are allocated to the flow. In this case, D would measure the maximum amount of time a flow's data, once ready to be sent, might have to wait for a slot. This value can be computed before slots are assigned and thus can be advertised. The error term D is measured in units of one microsecond.

If the composition functions is applied along the entire path to compute the end to end sum of C and D (Ctot and Dtot) and the resulting values are then provided to the end nodes, the end nodes can compute the maximal datagram queuing delay. Moreover, if the partial sums (Csum and Dsum) from the most recent reshaping point downstream towards receivers are handed to each network element then these network elements can compute the buffer allocations necessary to achieve no datagram loss. The proper use and provision of Guaranteed Service requires that the quantities Ctot and Dtot, and the quantities Csum and Dsum be computed.

- **Composed Ctot**: end-to-end composed value of C.

- **Composed Dtot**: end-to-end composed value of D.

- **Composed Csum**: The since-last-shaping point composed value of C.

- **Composed Dsum**: The since-last-shaping point composed value of D.

c) **Controlled-Load Service Fragment** (Per-service-header, Break-bit)

The control-Load service does not require extra Adspec data.

- **Per-service-header:** For Controlled-Load this parameter is 5.

- **Break-bit:** The only Adspec data specific to the Controlled-load service is the controlled-load break bit. When a path message Adspec with a controlled-load service header encounters a network element implementing Controlled-load service, the network element makes no changes to the service header. When it encounters a network element that supports RSVP but does not implement Controlled-Load service, the network element sets the break bit in the Controlled-Load service header.

A.4.2 Resv message: Resv(Session, Nhop, Time_Values, Resv_Confirm, Scope, Policy_Data, Style, Flow_Descriptor)

i. It is sent upstream towards sender(s) .

ii. It carries reservation requests to the RSVP nodes along the distribution tree.

iii. Resv messages originating from receivers of the same multicast group are merged together before being forwarded upstream.

iv. The IP destination address of a Resv message is the unicast address of a previous –hop node. obtained from the path state.

v. The IP source address is an address of the node that sent the message.

Resv message contains the following objects:

i. **Session, Time_Values, and Policy_Data:** These parameters were explained in Path message.

ii. **Nhop:** It is the RSVP hop which contains the IP address of the interface through which the Resv message was sent and the LHI for the Logical interface on which the reservation is required.

iii. **Resv_Confirm:** The appearance of a Resv_Confirm object signals a request for a reservation confirmation and carries the IP address of the receiver to which the Resv_Confirm should be sent.

iv. **Scope:** It carries an explicit list of sender hosts towards which the information in the message is to be forwarded.

v. **Style:** It defines the reservation style plus style-specific information that is not in Flowespec or Filterspec objects.

Multicast applications may have different service requirements. Therefore, various ways of aggregating requests from different receivers of the same session at intermediate routers have to be provided. Reservation Styles refer to the way of reserving bandwidth on outgoing links of each router and Merging is performed by each router along the

multicast distribution tree before sending upstream Resv messages. The merging is done in a specific interface using the information carried by arrived Resv messages to each RSVP node.

Reservation Styles and Merging are justified by two reasons: Better use of available resources and limitation of protocol overhead due to periodic refreshes

Reservation styles and Merging are defined by specifying reservation attributes and sender selection.

Reservation attributes may be distinct or shared. In distinct reservation attributes each receiver makes a separate reservation request for each sender of the multicast group but in shared reservation attribute each receiver makes a common reservation request to be shared among all specified senders.

Sender selection can be explicit or wildcard. Explicit means each receiver provides an explicit list of specified senders but in wildcard each receiver implicitly chooses all senders of the session.

Considering the reservation attribute and sender selection three styles are defined: Fixed Filter, Shared Explicit, and Wildcard Filter Style. Table A.1 shows the relation between styles, reservation attributes, and sender selection.

In **Fixed Filter style** separate reservation for each sender is done on outgoing interfaces (senders are explicitly listed). In this style the bandwidth requested in Resv messages sent upstream is the maximum among the bandwidth requests coming from receivers along

the distribution tree (this is done separately for each sender). A Fixed Filter (FF) style is shown in Figure A.1.

Reservation-Attribute / Sender Selection	Distinct	Shared
Explicit	Fixed Filter Style(FF)	Shared Explicit Style (SE)
Wildcard	Not Defined	Wildcard Filter Style

Table A.1 Reservation Styles

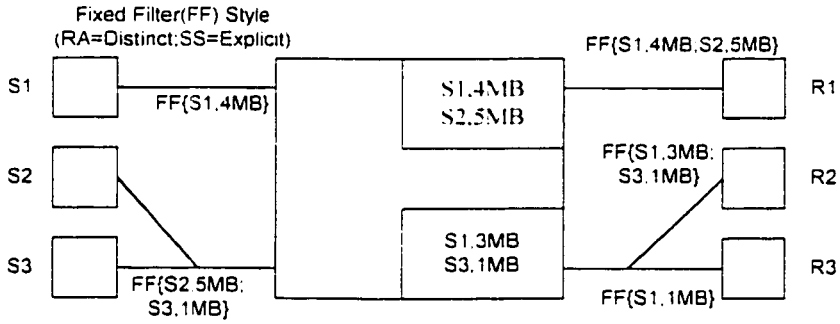

Figure A.1 Fixed Filter Style

In **Shared Explicit** style shared reservation for all senders is listed in the cumulative FilterSpec on each outgoing interface. The bandwidth requested in Resv messages sent upstream is the maximum among the bandwidth requests coming from receivers along the distribution tree for all senders reachable via each incoming interface. A Shared Explicit style is shown in Figure A.2

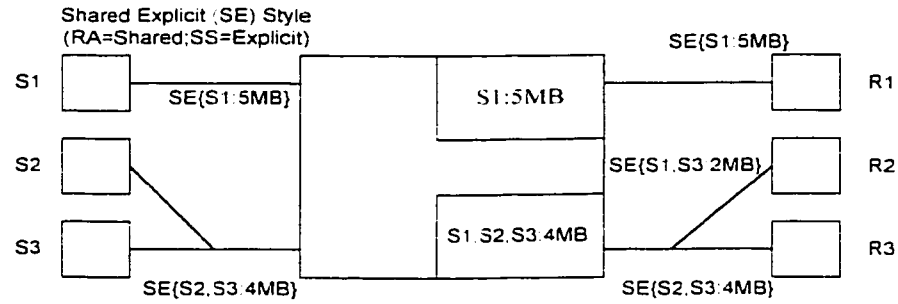

Figure A.2 Shared Explicit Style

In **Wildcard Filter** style shared reservation for all senders is listed in the cumulative FilterSpec on each outgoing interface. The bandwidth requested in Resv messages sent upstream is the maximum among the bandwidth requests coming from all receivers along the distribution tree. Figure A.3 shows a Wildcard Filter style.

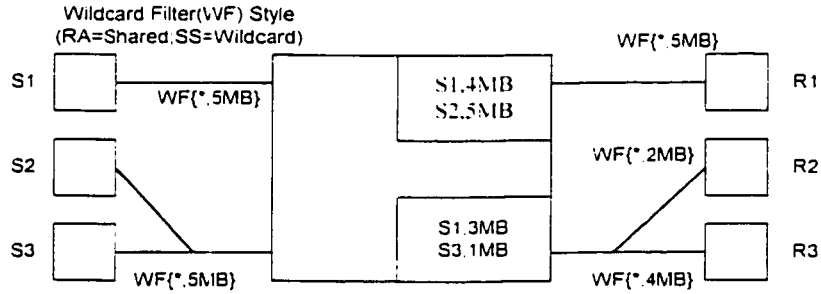

Figure A.3 Wildcard Filter (WF) Style

vi. **Flow-Descriptor** (FLOWSPEC, FILTERSPEC):

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

The QOS control service requested is indicated by the service-number in the FLOWSPEC's per-service header.

a) FLOWSPEC object when requesting **Controlled-Load service: FLOWSPEC** (SERVICE-HEADER,TSPEC)

SERVICE-HEADER for Controlled-Load service is 5 and **TSPEC** (TOKEN-BUCKET-RATE, TOKEN-BUCKET-SIZE, PEAK-DATA-RATE, MAXIMUM-PACKET -SIZE, MINIMUM-POLICED-UNIT) contains the following parameters:

- **TOKEN-BUCKET-RATE (r), TOKEN-BUCKET-SIZE (b), and PEAK-DATA-RATE (p)** are the TSPEC parameters r, b and p which reflect the traffic parameters of the receiver's desired reservation (the reservation TSPEC comparable to Sender Tspec in Path message). **MAXIMUM-PACKET-SIZE (M)** should be set to the value of the smallest path MTU, which the receiver learns from information in arriving RSVP ADSPEC objects. If the application has a fixed, known minimum packet size, then that value should be used for **MINIMUM-POLICED-UNIT (m)**.

For shared reservation style, the receiver may choose between two options or pick some intermediate point between them. If the receiver chooses a large value for [m], then the reservation will allocate less overhead for link-level headers. However if a new sender with a smaller SENDER-TSPEC [m] joins the session later, an already installed reservation may fail at that time. If the receiver chooses a value of [m] equal to the smallest value, which might be used by any sender, then the reservation will be forced to allocate more overhead for link-level headers. However, it will not fail later if a new sender with a smaller SENDER-TSPEC [m] joins the session. For a Fixed-Filter

reservation style. if no application-specific value is known the receiver should simply use the value of [m] arriving in each sender's SENDER-TSPEC for its reservation request to that sender.

b) FLOWSPEC object when requesting **Guaranteed service**: **FLOESPEC**(SERVICE-HEADER. TSPEC. MAXIMUM-PACKET-SIZE. MINIMUM-POLICED-UNIT. RSPEC)

SERVICE-HEADER for Guaranteed Service is 2 and **TSPEC**(TOKEN-BUCKET-RATE. TOKEN-BUCKET-SIZE. PEAK-DATA-RATE. MAXIMUM-PACKET-SIZE. MINIMUM-POLICED-UNIT) carries parameters that were explained in Controlled-Load service.

- **RSPEC**(Rate. Slack-Term) is generated by a receiver for a Resv message to be sent for a guaranteed service reservation request. It must include a slack term. S. as well as the amount of bandwidth R to be installed in each router along the path.

- **Rate(R)** must be greater than or equal to Token Bucket Rate. The rate R is measured in bytes of IP datagrams per second and has the same range and suggested representation as the bucket and the peak rate. The RSPEC Rate can be bigger than the TSPEC rate because higher rates will reduce queuing delay.

- **Slack-Term** signifies the differences between the desired delay and the delay obtained by using a reservation level R. This Slack Term can be utilized by the network element to reduce its resource reservation for this flow. When a network element chooses to utilize some of the Slack in the RSPEC. it must follow specific rules in updating the R and S

fields of the RSPEC; these rules are related to merging and ordering. The Slack Term is in microseconds and must be nonnegative.

A.4.3 PathTear message: PathTear(SESSION. RSVP_HOP. SENDER-DESCRIPTOR)

i. It is initiated explicitly by senders or by path state timeout in any node.

ii. It travels downstream towards all receivers.

iii. Receipt of a PathTear message deletes matching path states. Matching states must match the SESSION. SENDER-TEMPLATE. and PHOP object. A Path Tear message for a multicast session can only match path state for the incoming interface on which the PathTear arrived.

iv. If there is no matching path state. a PathTear message should be discarded and not forwarded.

A PathTear message contains following objects:

- **SESSION** and **RSVP_HOP** which have the same definition in Path message.

- **SENDER-DESCRIPTOR** which contains **SENDER_TEMPLATE, SENDER_ TSPEC and ADSPEC.** All these parameters were defined in Path message. **SENDER_TSPEC** and **ADSPEC** can be ignored in PathTear message.

A.4.4 ResvTear message: ResvTear(SESSION. RSVP_HOP. SCOPE. STYLE. FLOW_ DESCRIPTOR)

i. It is initiated explicitly by receivers or by any node in which reservation state has timed out.

ii. It travels upstream towards all matching senders.

iii. Receipt of a ResvTear message deletes matching reservation state. Matching reservation state must match the SESSION, STYLE, and FILTERSPEC and as well as the LHI in the RSVP_HOP object.

iv. If there is no matching it should be discarded.

v. A ResvTear message may tear down any subset of the FILTERSPECs in Fixed Filter and Shared Filter style.

All the parameters of this message have been explained in Resv message.

A.4.5 PathErr message: PathErr(SESSION, ERROR_SPEC, POLICY_DATA, SENDER_DESCRIPTOR)

i. It reports errors in processing path messages.

ii. It travels upstream towards senders and is routed hop by hop using the path state.

iii. It doesn't modify the state of any node through which it passes.

iv. It is only reported to the Sender.

All the parameters have been explained in Path message except ERROR_SPEC. **ERROR_SPEC** specifies the error and includes the IP address of the node that detected the error. SENDER_DESCRIPTOR is copied from the message in error.

A.4.6 ResvErr message: ResvErr (SESSION, RSVP_HOP, ERROR_SPEC, SCOPE, POLICY_DATA.STYLE, ERROR_ FLOW_DESCRIPTOR)

i. It reports errors in processing the Resv messages.

ii. It travels downstream towards the appropriate receivers.

iii. The node that detects an error in a reservation request sends a ResvErr message to the next hop node from, which the erroneous reservation come.

All the parameters have been explained in Resv message. **ERROR_SPEC** object specifies the error and includes the IP address of the node that detected the error.

A.4.7 ResvConf message: ResvConf (SESSION, ERROR_SPEC, RESV_CONFIRM, STYLE, FLOW_DESCRIPTOR)

i. ResvConf messages are sent to probabilistically acknowledge reservation requests.

ii. A ResvConf message is sent as the result of the appearance of a RESV_ CONFIRM object in a Resv message.

iii. It is sent to the unicast address of a receiver host.

All parameters have been explained in Resv and ResvErr messages.

Figure A.4 shows the directions of the RSVP messages.

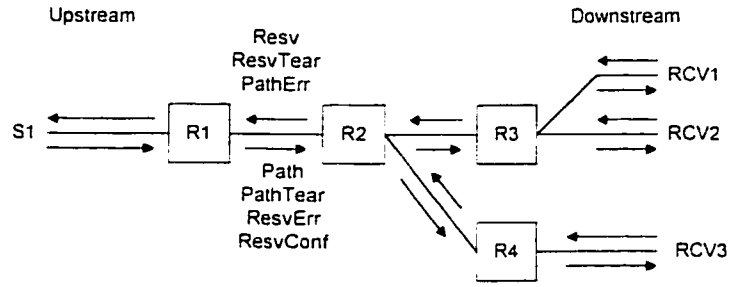

Figure A.4 Direction of the RSVP Messages

A.5 RSVP-MAC Interface

All the parameters in RSVP messages are opaque to RSVP itself. This means that to modify a parameter RSVP needs to use a special kind of interfaces according to the message type. RSVP in router and User Terminal has an interface to routing (network layer) and to traffic control (MAC layer). RSVP in a host has an interface to the application and also an interface to the traffic control. In this section we only describe the RSVP-MAC layer interface messages and parameters.

To modify the Adspec parameters in Path message, to make a reservation for a specific QOS, and to modify a reservation, RSVP needs to negotiate with MAC layer. This negotiation is done through the RSVP-MAC interface. In this section all the messages and related parameters from RSVP to the RSVP-MAC interface are described.

i. **TC_AddFlowspec** (Interface, TC_Flowspec, TC_Tspec, Police_Flags)

RSVP triggers this call to the interface by receiving a Resv message. This call carries the desired effective QOS to the admission control. If this call is successful it establishes a new reservation channel corresponding to Rhandle; otherwise it returns an error code. If

the traffic control service updates the Flowspec it will return the updated object as Fwd_Flowspec. TC_AddFlowspec contains the following parameters:

- **Interface:** Defines the interface that triggers this call.

- **TC_Flowspec**: Defines the desired effective QOS to the admission control. Its value is computed as the maximum over the Flowspecs of different next hops.

- **TC_Tspec:** Defines the effective sender Tspec.

- **Police_Flags:** carries the three flags. E_Police_Flag. M_Police_Flag. and B_Police_Flag

- **E_Police_Flag:** This flag is set in the first-hop RSVP node that implements traffic control.

- **M_Police_Flag:**This flag should be set on for a reservation using a shared style (WF or SE) when flows from more than one sender are being merged.

- **B_Police_Flag:**This flag should be set on when the Flowspec being installed is smaller than, or incomparable to, a Flowspec in place on any other interface, for the same Filter_Spec and Session.

After negotiation with MAC layer. RSVP-MAC interface will respond to this call in three possible ways: If the call is successful, the MAC layer will accept to reserve the requested QOS. and the interface will return a pointer. Rhandle. to this reservation. The caller will use Rhandle for subsequent references to this reservation. If the call is

successful but the traffic control service updates the Flowspec, the call will return the updated object as Fwd_Flowspec. If the call is unsuccessful it will return an error code.

ii. **TC_ModFlowspec**(Interface, Rhandle, TC_Flowspec, TC_Tspec, TC_Adspec, Police_Flags)

This call is used to modify an existing reservation. TC_Flowspec is passed to Admission control for implementing the modification. After negotiation with MAC layer, RSVP-MAC interface will respond to this call in the following possible ways:

If the request for modification is rejected the current Flowspec is left in force and a message indicating the rejection will be sent from the interface to the RSVP. If the call is successful, the Flowspec will be updated in the MAC layer and the interface will receive the updated object as Fwd_Flowspec. The interface will pass this updated object as a parameter in its response message to RSVP.

iii. **TC_DelFlowspec** (Interface, Rhandle)

This call deletes an existing reservation, including the Flowspec and all associated FilterSpecs. Rhandle points to the reserved Flowspec that is supposed to be deleted. This call doesn't require any response.

iv. TC_AddFiter (Interface, Rhandle, Session, FilterSpec)

This call associates an additional FilterSpec with the reservation specified by the Rhandle.

In response to this message a FilterSpec is added a pointer is returned to this Filter. which is called Fhandle.

v. **TC_DelFilter** (Interface. Fhandle)

This call removes a specific Filter. specified by Fhandle. This call dosen't require any response.

vi. **TC_Advertise** (Interface. Adspec. Non_RSVP_Hop_Flag)

This call is used to compute the New_Adspec. TC_Advertise sends the 'arriving' Adspec to the RSVP-MAC interface. The interface checks the 'arriving' Adspec parameters. According to the data in Adspec. the interface sends a probe message to the MAC layer to get the current 'local' information relevant to the requested parameters. MAC layer will collect the required information and sends them by a response message to the interface. Using specific routine. RSVP-MAC interface computes the 'result' Adspec and sends it as the New_Adspec to RSVP. **Non_RSVP_Hop_Flag:** should be set whenever the RSVP detects that the previous RSVP hop included one or more non-RSVP-capable nodes.

vii. **TC_Preempt ( )**

This is an upcall from MAC-RSVP interface to the RSVP. In order to grant a new reservation request. the admission control and/or policy control modules may preempt one or more existing reservations. This call is triggered by MAC layer. MAC layer will pass the pointer Rhandle and a reason code to the interface to declare the reservation to be preempted. RSVP-MAC interface will send TC_Preempt upcall to RSVP.

Appendix B

An Introduction to SDL

In two decades starting in 1970 there was a significant change of telecommunication systems from electromechanical systems with simple signaling methods to complex computer controlled systems with sophisticated signaling protocols. During this period, the CCITT started the development of Specification Description Language (SDL), and it was put into use by a substantial proportion of the world's switching system engineers, as ITU (International Telecommunication Union) recommendation Z.100.

The language has been evolving since the first Z.100 recommendation in 1980. It was further refined in 1981 and most of the current language was defined by 1984. Between 1988 and 1992 the language was extended with mechanisms supporting object orientation, parameterized types and packages. These features were extended in the latest version (SDL-2000) to give better support for object modeling and for code generation. For SDL-2000, the opportunity was taken to remove some features that were not strongly supported by tools. Object modeling in SDL was strengthened and better support given

for programming directly in SDL. In particular the data model was revised to give such features as global data and referenced data objects. Support for ASN.1 was strengthened so that the use of ASN.1 modules with SDL no longer requires any change main body of the language.

B.1 SDL: Objectives

SDL is intended for following objectives [17]:

i. Use of SDL in specification: SDL provides an unambiguous specification of systems. Specification written in SDL to describe what a system should provide can be analyzed and interpreted unambiguously because the meaning of SDL is described formally. The focus of a specification is on the interfaces of the system, and the relation between stimuli and responses.

ii. Use of SDL in design: SDL is used in design of the system to describe how a system should perform its functions. SDL design documents reflects the structure and the behavior of the system, which must completely conform to the system specification. The use of SDL leads to design by transformation of the specification and by addition of design information.

iii. Use of SDL in implementation: SDL design documents are used for achieving a workable system. Since most parts of the SDL design document is executable, in this phase it is only required to optimize parts of the final design document and add run-time information.

iv. Use of SDL in documentation: The graphical representation (GR) of SDL and the close connection between the SDL description and the executable code makes SDL valuable for controlling maintenance costs of large software systems.

B.2 Key features of the SDL

The key features of the SDL language are:

i. The ability to be used as a wide spectrum language from requirements to implementation.

ii. Suitability for real-time, stimulus-response system.

iii. Presentation in a graphical (SDL/GR) and textual (SDL/PR) form. (In this introduction we only emphasize on graphical form).

iv. A model based on communicating processes.

v. Object-oriented description of SDL components.

B.3 Overview of the Language

i. **System and environment**: SDL sees the world as divided into two parts: the system and its environment. Everything outside the system is considered to be part of the environment. The specification defines how the system under consideration reacts to events in the environment that are communicated by *signals* sent to the system. Figure B.1 shows the interaction of an SDL system with its environment via signals.

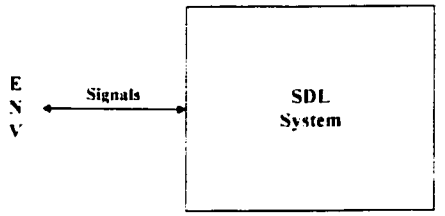

Figure B.1 Interaction between System and Environment

ii. **System behavior**: In SDL the behavior of a system is the joint behavior of all *process* instances contained in the system, as shown in Figure B.2. The process instances may communicate both with each other and with the environment via signals.

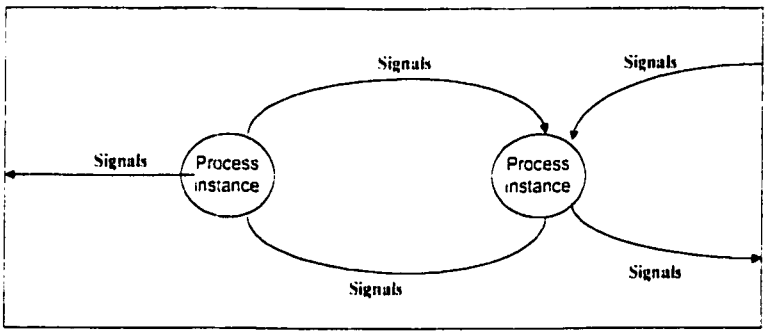

Figure B.2 System Behavior

iii. **Finite-state and extended finite-state machines:** A finite state machine always has one of a finite number of states. A finite-state machine may change its state by receiving one of the signals defined for it in its input signals. An extended finite-state machine introduces variables in addition to the explicit states of the process instances. These variables become implicit states. The complete set of states in an extended finite-state machine is the union of the implicit and explicit states. An SDL process instance is a communicating extended finite-state machine. These processes are concurrent and

run in parallel. Each process has an unbounded buffer called the input queue to receive incoming signals. This queue is normally a FIFO queue. However, it must be noted that the SDL process may extend the finite state machine model in a way, that the queue associated with the process may implement some sort of priority. This means the process queue is not necessarily FIFO [18].

B.4 Architecture Description of a System Instance

SDL provides mechanisms to structure systems to be able to cope with complexity. This idea is achieved by splitting the system into different levels of hierarchy [18]. *System level* of description includes the basic building blocks used to structure a system known as *blocks*. Blocks may contain sub-blocks. If we have many hierarchical levels of blocks, we get a block tree with the system as the root. Only the lowest level of blocks in any branch may contain *processes*. In a very simple system, a block contains one or more process sets that communicate with each other and via *signal routes* that connect the process sets with each other and the environment of the *block*. Blocks communicate with each other and the system environment via *channels*. Channels connected to a block are connected to signal routs within the block. Figure B.3 and B.4 shows the partitioning a

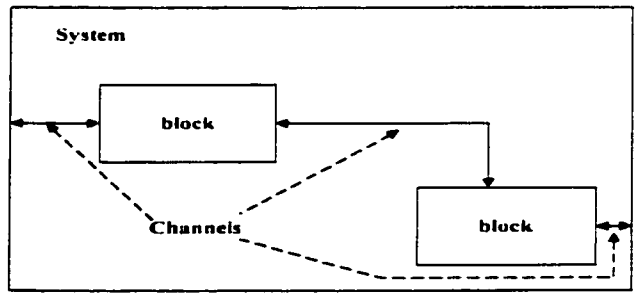

Figure B.3 Partitioning a System into Blocks

system into blocks and block into processes, respectively.

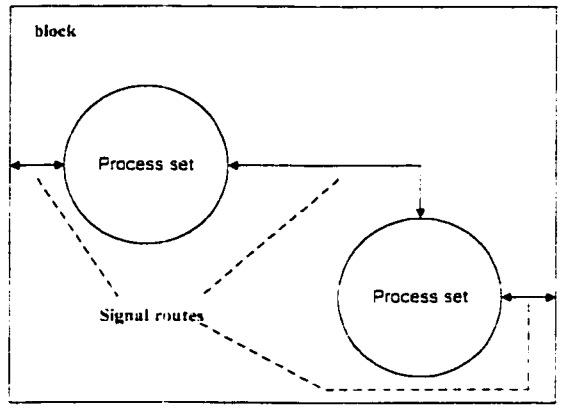

Figure B.4 Partitioning a Block into Process Sets

B.4.1 System definition:

At the first level of decomposition, a specification of a system in SDL, as shown in the

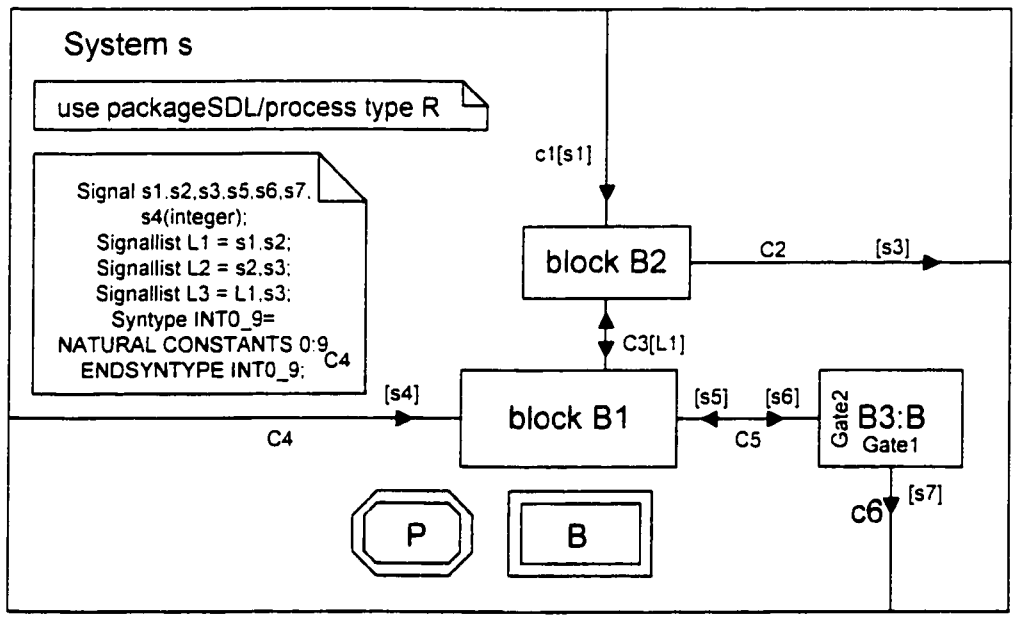

Figure B.5 Specification of a System. Block B1 and B2 are "referenced", block B3 is instantiated.

sample Figure B.5, includes the following points:

i. System declaration: The system declaration area can be contained in one or more sub-areas each delimited by a text symbol shown in Figure B.6.

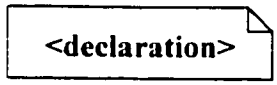

Figure B.6 Text Symbol

ii. Naming the system: Each system must be given a name in system level. In Figure B.5, the system has been named as S.

iii. Definition of the blocks and channels that make up the system and the interaction part: The block interaction part defines the architecture of the system at its highest level. It introduces the blocks and channels of the system. Channels connect blocks to each other and with the environment of the system. Normally, instead of using the specification of the block directly within the system specification, a reference to the block is used. The block specification itself is found somewhere else. The reference shows the external connections to the channels. The block is reduced to a pointer to a remote specification. The block interaction part thus contains block reference (Figure B.7), block

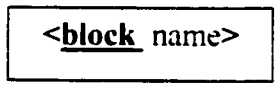

Figure B.7 Syntax of a Block Reference

instantiations (Figure B.8) and delaying and non-delaying channel specification (Figure B.9 and B.10). For the system specification, this implies that there is either a block type specification or reference at this level, or that the type specification is imported from an external package.

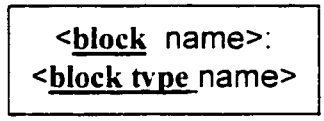

Figure B.8 Instantiation of a Block type

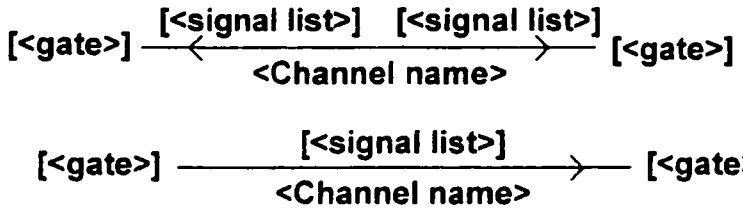

Figure B.9 Syntax of the Specification of Delaying Channels

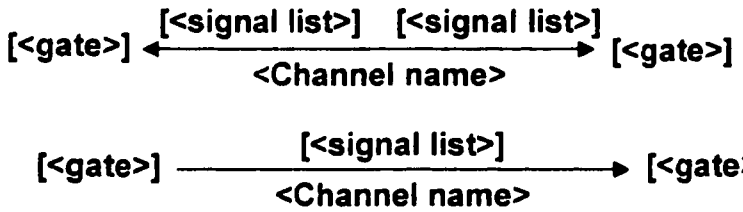

Figure B.10 Syntax of a Specification of non-delaying Channels

It must be noted system specification part may contain type specifications such as block and process types. Block types that are remotely specified at system level may be used or instantiated by blocks in the block interaction part. Remote specifications use a block type reference instead of a block type specification. Figure B.11 shows a block type reference.Process types may not be instantiated at system level. as the block interaction part may not contain process instances. The reason for specifying process type at system level could be that it is desirable to have enclosed units. such as blocks. use the same

process type definition. This way we avoid placing a reference to a process type repeatedly in enclosed scope units. Figure B.12 illustrates a process type reference.

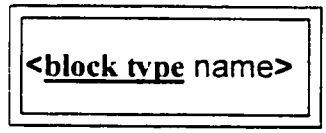

Figure B.11 Syntax of a Block Type Reference

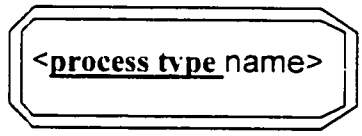

Figure B.12 Syntax of a Process Type Reference

iv. Definition of the *signals*: The signal specification identifies the name of the signal type and the sorts of the parameters to be carried by the signal. These formal parameters in the signal specification are replaced by the actual parameters or values of the data sorts when the real signal is sent by one process to another. The value of the actual parameters must correspond to the sorts of the signal specification. Figure B.13 shows the signal definition.

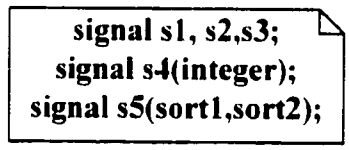

Figure B.13 Signal Definition

v. Definition of the *signallists:* Signallist groups together the signals and/or signallists, to facilitates the writing of the specification. It includes a name for the signallist and the list of signals and/or signallists. Figure B.14 illustrates the definition of the signallists.

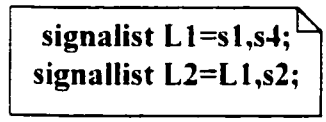

Figure B.14 Signallist Definition

vi. Definition of the data types: In SDL, data types are abstract data types (ADTs). This means that each data type has an interface part (*a signature*) definning how and which literals and operators can be used to obey the language rules and a behavior part defining the semantics of the literals and operators. The concept of abstract data types defines data in an implementation-independent way, both for the predefined data and for data that may be defined by the user. Data types in SDL are called *sorts* to avoid being confused with the abstract data types. Data types defined in system level are visible throughout the specification. SDL includes features, which free the user from defining often used data types from scratch. In particular SDL includes:

a) Predefined sorts such as *BOOLEAN, CHARACHTER, CHARSTRING, NATURAL* (positive), *INTEGER* (signed), *REAL, PID, DURATION* and *TIME.*

b) Predefined generators for constructing sets, strings, and arrays. To define these data types the *Powerset, String and Array* generators are used, respectively

c) Record types: As known from programming languages, record types can also be specified in SDL. SDL includes a special construct, a so-called *struct*, to define such a type.

d) Syntype concept is used for two purposes: To define a more meaningful name for an existing data type and to limit the range of values that variables of an existing data type can have. Figure B.5 contains examples of defining some data types.

vii. Packages: SDL has offered the possibility to place reusable components defined as types into libraries that are called packages. This allows the type specification to be used in more than a single system specification. A system specification imports an external type specification defined in a package with the *use* clause. Figure B.5 shows an example of calling a package in system level.

B.4.2 Block definition

Blocks are the structuring elements in SDL. There are three different kinds of blocks in SDL: Terminal block, which consists exclusively of SDL processes, shown in Figure B.15. Non-terminal blocks, which consists exclusively of blocks, shown in Figure B.16. and referenced block, which is defined in another module, as shown in Figure B.7.

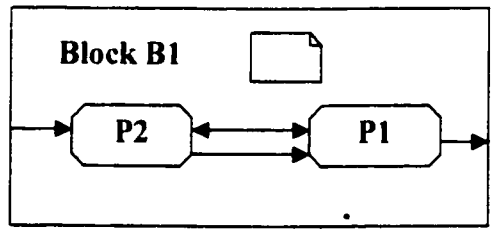

Figure B.15 Terminal Block

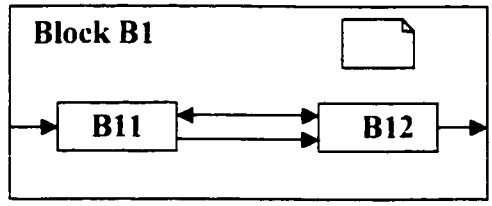

Figure B.16 Non-Terminal Block

Here we emphasize on the specification of a terminal block. as illustrated in the sample Figure B.17. A terminal block specification describes process sets that communicate with each other and with explicit or implicit channels in the environment of the block. This communication is via channels carried in signal routes that may be explicit or implicit.

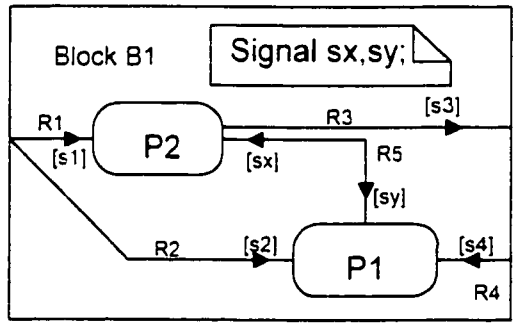

Figure B.17 SDL Terminal Block Specification

The signals themselves may carry parameters with additional values. Signals are transported in signal routes without any delay. Specification of a terminal block includes:

i. Name of the block

ii. Definition of SIGNALLISTs, data type, and signals: Signallists, data types, and signals that have already been defined at system level need not be defined again at block level.

iii. Definition of the PROCESSES and SIGNALROUTEs of the block: Processes describe the behavior of the block and signal routes connect process sets to each other and with the environment of the block. The individual process instances of a process set all share the same connections via signal route.

The process sets of a block must be defined either locally or remotely. As the case of system specifications, we use the remote type of specification here. This call for a process reference instead of a process specification inside the block specification. A process reference symbol is shown in Figure B18.

Figure B.18 Syntax of a Process Reference

Process instance sets can also be instantiated from process type. Figure B.19 shows a process instantiation. It must be noted that instantiation of process types necessitates the definition of the gates of the process type to which the instantiated process set is connected.

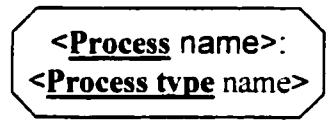

Figure B.19 Syntax of a Process Set Instantiation

SIGNALROUTEs connect the processes of the block among themselves and with the environment, and they carry the specified set of signals. Signal routes may be both uni- and bi-directional. Figure B.20 shows a signal route specification.

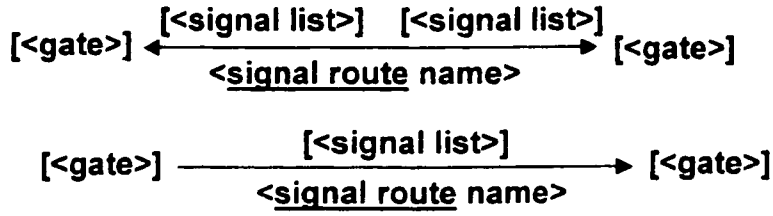

Figure B.20 Syntax of a Signal Route Specification

It is important to note that processes may be referenced but signal routes are completely specified when they are introduced for the first time. They may also only be used as instances, never as types.

iv. Definition of the CONNECTIONs between channels exterior to the block and signalroutes interior to the block: Figure B.21 shows the connections between channels and signal routes in both remote and local specification.

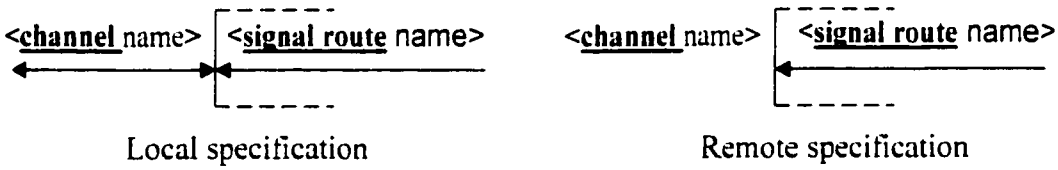

Figure B.21 Differences in Connecting Channels and Signal Routes

B.4.3 Process Definition

This part of specification concerns, the dynamic behavior of processes of a system. As previously mentioned, the behavior of the blocks and systems is not directly specified. It is derived from the behavior of the process instances.

Processes act as extended finite state machines. It means besides explicit states, SDL also allows the use of data or variables in a process. When a new, different value is assigned to a variable of a process, the global state of the process changes. Though not as visible as the explicit states, variables may influence the behavior of a process.

Instances of a process can exist from the creation of the system, or can be created after a request from another process. An arbitrary number of a process can be created and exist simultaneously.

A process can have an unlimited lifetime that means the process effects a certain number of actions and returns to an initial state. Furthermore, a process can stop after a certain time by moving to a final state. Specification of a process includes:

i. Name of the process

ii. A pair of integer numbers are used to specify the number of instances: The first integer specifies the number of processes created at the initialization of the system (default value 1) and the second gives the maximum number of simultaneous instances of the process (default value unlimited)

iii. Formal parameters: This defines the information provided at the moment of creation of the process.

iv. Definition of data types: Data types declared in a process specification is only visible inside the process. Data types that have already been defined at system and block level need not be defined again.

v. Definition of timers: Timers are objects that belong to a process instance. They are in a position to create timer signals under certain conditions. These signals are then put into the input queue of the owning process in the same way that all other signals are put into the input queue. Timers must be declared before they are used. Timers exist from the time the process instance is created to the time the process stops. Figure B.22 illustrates the declaration of a timer.

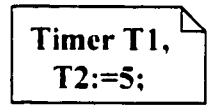

Figure B.22 Definition of Timers: This declares a timer T1 with indeterminate duration and timer T2 with duration of 5

vi. Declaration of variables: Inside the process the variables that are used in the body of the process must be declared. These variables are local and can be seen inside the process. Figure B.23 shows an example of the variable declaration.

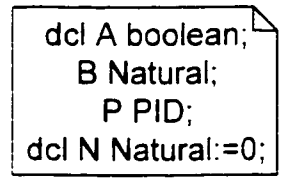

Figure B.23 Declaration of Variables

vii. Body of the process: The process body describes the behavior of a process set and is called the process graph in the graphical representation. The specification of the behavior is based on the concept of extended finite-state machines. Figure B.24 illustrates the different parts of the body of a process.

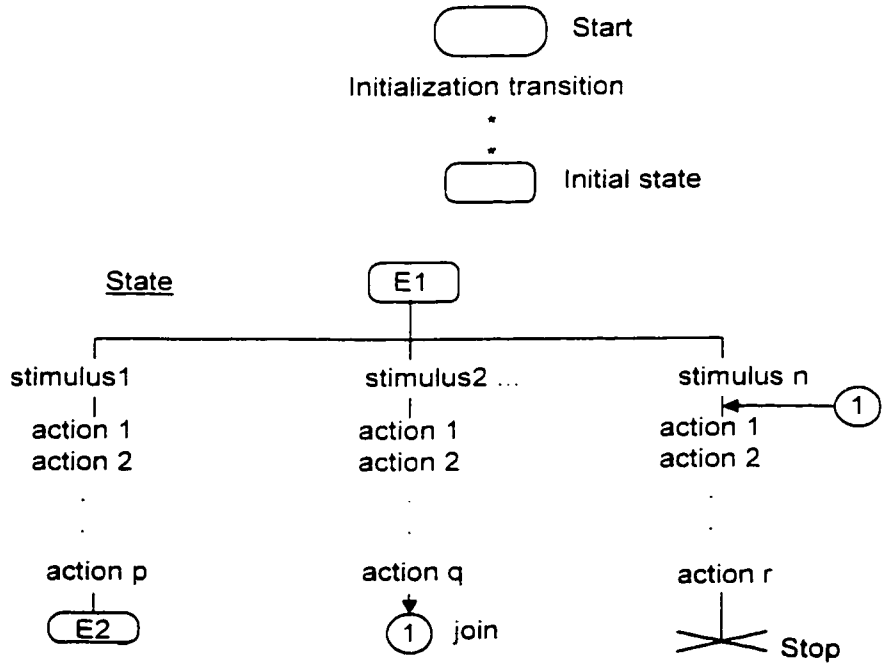

Figure B.24 Body of a Process

The body of a process consists of the followings:

a) Initialization transition (start): It brings the process to an initial state.

b) Set of states: Each process consists of different states. Each state must have its own name.

c) A set of stimuli possible in each state such as followings:

c-1) Input: An input is the acceptance of a signal by a process in a certain state. When a signal is accepted it is also consumed. An input signal connects a state to the actions which the process shall take after consuming the signal mentioned in the input symbol. Figure B.25 shows an input symbol.

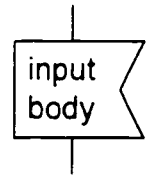

Figure B.25 Input Symbol

c-2) Priority input: In some cases handling of some signals have higher priority than handling of others. Signals which take priority in a state are placed inside priority input symbol, as shown in Figure B.26. If a state has both ordinary and priority inputs, the first signal in the input port mentioned in a priority input is consumed, even if it is not the first signal in the input port.

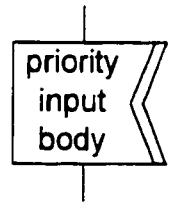

Figure B.26 Priority Input Symbol

c-3) Continuous signal: In some cases it is more convenient to trigger transitions by some

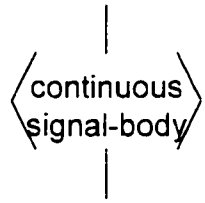

Figure B.27 Continuous Signal Symbol

conditions being fulfilled than by the reception of signals. The continuous signal symbol connects a state to the actions taken if the condition is true. Figure B.27 illustrates the continuous signal symbol.

c-4) Postponing triggers by SAVE: Signals which should not be dealt with in a certain state are mentioned in a save symbol, as shown in Figure B.28. They are retained in the input port. The retained signals are available for input in a consecutive state.

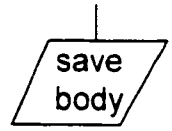

Figure B.28 Save Symbol

d) The transitions that the process effects upon reception of each stimulus: A transition is composed of a set of actions such as TASK, OUTPUT, CREATE REQUEST, DECISION, SET and RESET. A transition may be terminated by the arrival in a new state, a JOIN, or the termination of the process (STOP).

d-1) Create Request: The execution of the system depends on the existence of initial

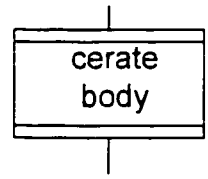

Figure B.29 Create Symbol

processes, but in addition dynamic process creation can be used to represent dynamic population. Dynamic process creation is represented by the create request symbol, as shown in Figure B.29.

d-2) Task: The action called task allows data to be manipulated locally and to be utilized for influencing the behavior of the process. A simple example is the assignment of the values to variables of a process instance. Figure B.30 shows the task symbol.

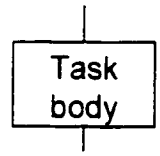

Figure B.30 Task Symbol

d-3) Decision: Decisions allow the execution of a process to be influenced by data values. A decision splits a transition into several branches. The decision symbol is shown in Figure B.31. It must be noted that there is not any limitation to the number of the possible

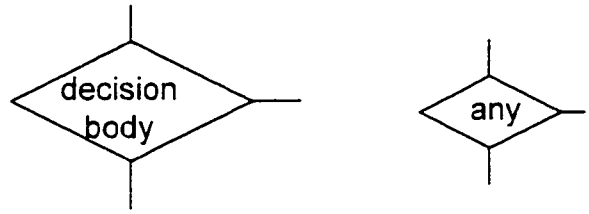

Figure B.31 Decision Symbol

outlets. Decision body is a test to decide which branch in the transition should be taken. Decision body can be "any" which implies an arbitrary decision. When execution reaches an arbitrary decision, an arbitrary exit is chosen.

d-4) Join and connection: Branches of transitions can directly merge, as shown in Figure B.32, or if this is inconvenient for drawing, out-connectors and in-connectors can be used as shown in Figure B.33. An out-connector corresponds to a "goto" in a programming language, whereas an in-connector corresponds to a "lable".

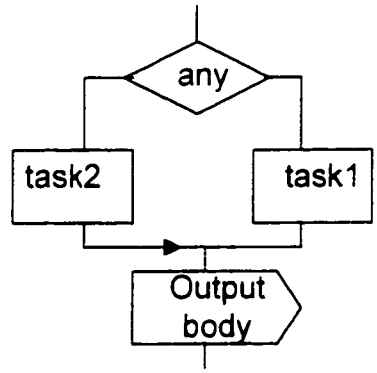

Figure B.32 Direct Merging of Transitions

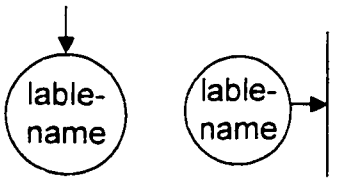

Figure B.33 Out and In-connector

d-5) Output: Output is the basic communication mechanism from a process to other processes or to the environment of the system. A signal instance begins its life when an output is executed. It is then sent via communication paths to receiving process instance. which stores it in the input port until it is consumed. The output symbol is shown in Figure B.34.

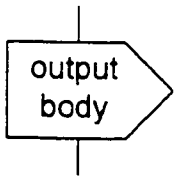

Figure B.34 Output Symbol

The text inside the output symbol contains the names of signals possibly with actual parameters. followed by a possible address and a possible path which must be followed when the signals are output.

d-6) Stop: A process can only be stopped by itself. This occurs by interpreting the stop node. shown by stop symbol in Figure B.35. After interpreting this node. the process instance ceases to exist and with it all its components. its variables and pre-defined expressions.

The stop node has no parameters and may be repeated any number of times in any particular number of transitions.

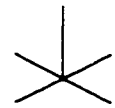

Figure B.35 Stop Symbol

d-7) Set and Reset of timers: A timer can be set explicitly to a certain value or implicitly to the optional default value specified in the timer declaration. When a timer is set, it always resets first and the corresponding timer signal is removed from the input queue, if applicable. The syntax of a set is shown in Figure B.36. Most often we wish to set a

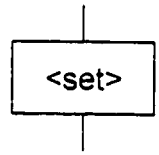

Figure B.36 Setting a Timer

relative time, not an absolute time. One method is to use the default duration for the timer. The other is to use the predefined expression *now* and to add to it the required duration, which results in the absolute time required in the specification of the timer.

A timer can also be reset during a transition. An active time is then made inactive. If at

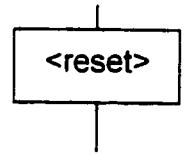

Figure B.37 Resetting a Timer

this time a timer signal of the timer is still in the input queue of the process instance, it is removed from the input queue. Figure B.37 illustrates the syntax of reset.