

Dublin City University

School of Electronic Engineering

**Simulation and Performance Analysis of
A Telecommunication System based on
Advanced Intelligent Network
Architecture**

A thesis submitted as a requirement for the degree of Master of Engineering in
Electronic Engineering

July 1993



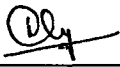
I declare that this thesis is entirely of my own work and has not been submitted as an
exercise to any other university

Supervisor: Dr. T. Curran

**Manish Gulyani
B.E.**

Declaration

I hereby certify that this material, which I now submit for assessment on the programme of study leading to the award of Master of Engineering is entirely my own work and has not been taken from the work of others save and to the extent that such work has been cited and acknowledged within the text of my work

Signed. 
(Manish Gulyani)

Date 27/9/13

Date. _____

Abstract

Title: Simulation and Performance Analysis of a Telecommunication System based on Advanced Intelligent Network Architecture

Author: Manish Gulyani

This thesis describes the modelling and simulation of a telecommunication system based on Advanced Intelligent Network (AIN) architecture for provision of telecom services. It also presents an analysis of the performance of the Service Control Point under overload conditions with the implementation of Automatic Code Gap (ACG) controls.

A telecommunications system was developed to demonstrate the operation of an intelligent network. Further, a set of services including Call Forward, Call Back, Call Transfer and Automatic Call Distribution were implemented on the network. This was done to exhibit the relative ease with which new services can be developed and implemented on the AIN network in a system independent manner. Further, to evaluate the performance of the Service Control Point (SCP) under overload conditions another system was developed. The functionality for implementing ACG controls on the system was also realised. In addition, an algorithm for selection of controls was developed.

An extensive set of simulations were performed on the system to determine the effectiveness of the controls in preventing overloading at the SCP. A considerable improvement in SCP performance was observed with the deployment of ACG controls. OPNET, a network simulation tool, was used for developing the system.

Acknowledgements

I would like to thank my supervisor, Dr T Curran, for his guidance and support throughout the course of my research I would like to also thank him for giving me an opportunity to pursue the research program

I am very grateful to Prof C McCorkell for his help and encouragement A special word of thanks for the technicians Dave Condell and Paul Wogan, for their assistance.

I am thankful to Dan Keane, for the time he took out to guide me Thanks to Shane Sexton for helping me with the project and proof-reading my thesis

I am greatly indebted to my fiance' Anuradha Sahgal, for always supporting me and assisting me in times of trouble

I would also like to thank my family for their affection and constant support, even though they were so far away from here Finally, I thank all my friends and colleagues for their help and patience.

Table of Contents

Chapter 1 Introduction	
1.1 Introduction	1
1.2 Research Objectives	2
1.3 Structure of Thesis	2
Chapter 2 Intelligent Networks	
2.1 Introduction	4
2.2 Intelligent Networks - CCITT Recommendation	5
2.2.1 Intelligent Networks Conceptual Model (INCM)	6
2.2.2 Functional Architecture	8
2.2.3 Physical Architecture	10
2.3 Bellcore's Advanced Intelligent Networks (AIN)	10
2.3.1 Functional Architecture	10
2.3.1.1 Functional Entities (FEs)	11
2.3.1.1.1 Network Access (NA) FE	11
2.3.1.1.2 Service Switching (SS) FE	12
2.3.1.1.3 Service Logic and Control (SL&C) FE	12
2.3.1.1.4 Information Management (IM) FE	12
2.3.1.1.5 Service Assistance (SA) FE	12
2.3.1.1.6 Automatic Message Accounting (AMA) FE	12
2.3.1.1.7 Operations (OP) FE	13
2.3.1.2 Functional Groups	13
2.3.2 Physical Architecture	13
2.3.2.1 AIN Switching System	14
2.3.2.2 Service Control Point (SCP)	16
2.3.2.3 Adjunct	16
2.3.2.4 Service Node (SN)	17
2.3.2.5 Intelligent Peripheral (IP)	17
2.3.2.6 Operations Systems (OS)	17
2.4 AIN Call Processing	18
2.4.1 AIN Basic Call Processing	18
2.4.2 AIN Call Processing Requiring Network Resources	19
2.4.3 Call Model	19
2.4.3.1 Connection View	20
2.4.3.2 Basic Call State Models (BCSMs)	22
2.5 An AIN Service Example - Freephone Service	25
2.6 Comparison between Bellcore AIN and CCITT IN	27
Chapter 3 System design and Implementation	
3.1 Introduction	29
3.2 Overview of OPNET	30
3.3 AIN Network Model	33
3.4 AIN Node Models	34
3.4.1 Switching System Node Model	34
3.4.2 Service Logic Node Model	36

3.4.3 Intelligent Peripheral Model	37
3.4.4 User Model	38
3.5 AIN Process Models	38
3.5.1 Switching System Process Models	39
3.5.1.1 Process Controller	39
3.5.1.2 Switching Fabric	40
3.5.1.3 Call State Models	42
3.5.2 Service Logic Node Process Models	45
3.5.2.1 Service Manager	45
3.5.2.2 Service Logic Program	45
3.5.3 Intelligent Peripheral Node Process Model	46
3.5.3.1 Resource Controller	47
3.6 Communication between Network Nodes	47
3.6.1 ASC-SLEE Interface	47
3.6.1.1 ASC-to-SLEE Communication	47
3.6.1.2 SLEE-to-ASC Communication	48
3.6.2 ASC-RCEE Interface	49
Chapter 4 AIN Services Realisation	
4.1 Introduction	51
4.2 Basic Call Handling	51
4.3 AIN Services	53
4.3.1 Call Forwarding	54
4.3.2 Call Transfer	56
4.3.3 Call Back	58
4.3.4 Automatic Call Distribution	61
4.4 Results	65
Chapter 5 SCP Overload Control - Modelling and Simulation	
5.1 Congestion in Networks	66
5.2 Network Traffic Management	66
5.2.1 NTM Controls	67
5.3 Overload in Intelligent Networks	68
5.4 NTM at Service Control Point	69
5.4.1 Network Functionality	71
5.4.2 SCP Overload Control	72
5.4.3 SMS Originated Code Control	72
5.4.4 ACG Control Mechanism	73
5.5 Overload Control Simulation Model	74
5.6 Network Model	76
5.7 Node Models	77
5.7.1 Switch Side Node Model	77
5.7.2 Service Logic Node Model	78
5.8 Process Models	79
5.8.1 Switch Side Node Process Models	80
5.8.1.1 Poisson Generator Model	80
5.8.1.2 Bursty Generator Model	81
5.8.1.3 Switch Process Model	82

5.8 2 Service Logic Node Process Models	84
5.8.2 1 SLEE Process Model	85
5 8.2 2 SLEE Transmitter Process Model	86
5 8 2.3 SLP Process Model	87
5.8 2 4 SMS Process Model	88
5.9 Calculation of Thresholds	89
5.10 SMS Control Selection Algorithm	91
5.11 Simulation Parameters and Results	94
5.11.1 SCP Performance Without ACG Controls	95
5 11 2 SCP Performance With SMS Originated ACG Controls	98
Chapter 6 Conclusion	
6.1 Summary and Conclusions	109
6.2 Directions for Future Work	111
References	R 1
Appendices	
Appendix A List of Acronyms	A 1
Appendix B Message Set	B 1
Appendix C Flow Chart of Control Selection Algorithm	C.1

Chapter 1

Introduction

1.1 Introduction

Rapid growth has been seen in the telecommunications industry for the last few decades. The sizes of telephone networks have multiplied to provide service to an ever increasing number of subscribers. Simultaneously, the range of services offered on these telephone networks have also expanded. As early as in 1965, services like Call Waiting and Centrex were provided to the subscribers by deploying stored program control (SPC) technology in the telephone networks. As networks grew bigger and more complex, SPC technology was also used for developing network management systems[17]

With further technological advancements more sophisticated services like Freephone and Calling Card were introduced. These services were realised by adding Network Control Point (NCP) systems to the existing telephone networks comprising of SPC switches. Large volumes of data necessary for the provision of these services are stored in the databases located at the NCPs. These databases are accessed by the SPC switches via the Common Channel Interoffice Signalling (CCIS) network. This approach made possible the introduction of services which were otherwise impractical to provide by eliminating the need for managing data at each SPC switch.

But as competition in the telephone industry becomes fiercer, there arises an urgent need for network operators to respond quickly to the subscriber's demands for more specific service capabilities. Since software written for implementing a service on the SPC switch is system dependent, services can be modified or added only by the switch vendors themselves. This makes the telecom operators totally dependent on the switch vendors for providing specialised services to their subscribers, and consequently slows down their response to subscriber needs. Therefore, there is a need for the introduction of a network architecture that would allow the operators to quickly and economically create and modify telecommunications services for their subscribers.

The Advanced Intelligent Network (AIN) architecture is designed to fulfil these needs of the telecom operators. In an intelligent network the service control logic is decoupled from the underlying switching fabric, thereby allowing easier introduction of new services through changes to the service software by the operators themselves[19]. Further, to allow multiple vendors to supply the network elements, it standardises the interfaces between systems, and defines the functionality of each system. As an additional feature, it introduces the concept of automated subscriber interactions with the network.

Separation of service logic from the switch creates extensive signalling requirements in an AIN network. Messages are exchanged between the service control element and the switch for providing services to the subscribers. Where the service control element is realised in a Service Control Point (SCP), messages are exchanged between the SCP and the switch over an SS7 signalling network. This introduces delays in providing services to the subscribers. In some cases of services where extensive messaging is required between the SCP and the switch, it is possible that delays are longer as compared to the delays with switch-based implementation of the services. Therefore, the performance of the SCP and the signalling network is going to play an important role in determining the success of the AIN architecture.

1.2 Research Objectives

The subject of our research was Advanced Intelligent Networks (AIN). The aim of the project was to model and simulate a telecommunication system based on the AIN architecture. In addition, to demonstrate the ease with which new services could be created and implemented on the network, a set of standardised telecom services were realised in this system.

As performance of the SCP plays such a critical role in the quality of service offered by an intelligent network, the aim of the second part of the project was to analyse the performance of the SCP under overload conditions. Further, to evaluate the effectiveness of controls in limiting the load on the SCP, and thus preventing congestion in the network, another system was simulated.

1.3 Structure of Thesis

Chapter 2 offers an overview of Intelligent Networks (IN). A brief introduction about IN is followed by the description of the IN functional and physical architecture as proposed by CCITT. Bellcore is making their recommendations under the name of Advanced Intelligent Network (AIN). The architecture proposed by them is also presented in this

chapter. Subsequently, the call processing mechanism performed in an AIN is explained, and further illustrated with the help of an AIN service example. Finally, the differences between Bellcore's AIN and CCITT's IN are brought out in this chapter.

A network simulation CAE tool, OPNET, was used in the development of the AIN system. An overview of the tool, and the methodology for simulating a communications system model with this tool, is described in Chapter 3. Subsequently, a detailed description of the model developed to simulate an AIN network is also presented.

On the simulated AIN network, basic call handling services and a few AIN supplementary services were implemented. These services included Call Forwarding, Call Back, Call Transfer, and Automatic Call Distribution. How these services were realised using the AIN architecture and protocol, is discussed in chapter 4 of the thesis. In the service descriptions, operation of the elements of an AIN network, and the role played by each one of them to provide the service is also explained.

In chapter 5, the possible reasons for congestion developing in Intelligent Networks are examined. Some of the NTM systems that may be employed to control it are then discussed briefly. This is followed by an indepth description of the Automatic Code Gap control scheme suggested by Bellcore for preventing congestion at the SCP. To analyse the performance of the SCP under overload conditions, another system was simulated. On this system, ACG controls were implemented. A detailed description of this system's design and implementation is provided in this chapter. Finally, the performance of the SCP is analysed by interpreting the results obtained from running simulations on the developed system.

Chapter 6 concludes this thesis. It gives a summary of the systems that were developed, and the results that were obtained by performing simulations on these systems. A comparison between the results obtained and the results expected, is also presented. To conclude, some suggestions are made for further enhancing the developed systems.

Chapter 2

Intelligent Networks

2.1 Introduction

Intelligent Network (IN) is a telecommunications network services control architecture. It provides a framework so that the Network Operator can introduce, control and manage services more effectively, economically and rapidly than the current network architecture allows[10]

The intelligent network's main advantage from the telecom operator's point of view is the ability to provide service from a small set of IN nodes, the Service Control Points, which are connected to switches, either directly or via Signalling System No 7. In the existing service creation and control environment, services are not centralised in this way. Operators must agree with their switch vendors on the services to be provided. The switch vendor must develop the software, and then the software must be loaded and tested on each switch separately. Thus, the operator is entirely dependent on the switch vendor for new services.

This is a time consuming and costly process, and prone to errors. With the IN architecture, these problems are diminished because the switch is effectively decoupled from the service creation and control environment. The Service Control Point need not be provided by the operator's main switch supplier, it could be provided by another vendor, or by a computer vendor. Moreover, in this standardised environment, services can be created, written, and an input made by the service operators themselves, or contracted out to either the equipment manufacturers or software houses.

The key objectives for IN [9] can therefore be summarised as follows

- Increase service velocity to enable the market-driven, rapid introduction of new services, from conception to deployment

- Broaden the range of services to go beyond the traditional voice and data bearer services to a much broader range, including information services, broadband and multimedia bearer capabilities
- Enable a multivendor, competitive environment to ensure that the services will work correctly and consistently on any vendor's equipment, and across several vendors' equipment
- Evolve from existing networks; the deployable technology must interwork with and evolve from existing networks since these cannot be replaced overnight.

Work on IN is being carried out, both, in the U S and Europe. The CCITT is producing a standard called Capability Set 1(CS-1), Bellcore is working on the Advanced Intelligent Network(AIN) standard in a series of Releases, starting with 0 0 and reaching a goal of 1 0, through several interims, of which the first is 0 1, to provide a migration path. CS-1 should be approved by CCITT in 1993, and has been designed as a subset of AIN Release 1 to ensure convergence between the two standards. CCITT has adopted a Chinese box strategy, whereby CS-1 will be in turn, a subset of CS-2, and so on. Both AIN and CS-1 assume IN architectures that can be described in both logical and physical network terms, the elements are similar, but not identical [19]. A brief description of CCITT's IN architecture has been given in section 2.2. It is followed by a more detailed description of Bellcore's AIN architecture in section 2.3. AIN call processing has been dealt with in section 2.4. We have developed our telecom system based on the latter because of the availability of more detailed information on this architecture.

2.2 Intelligent Network - CCITT Recommendation

In order to achieve the above mentioned objectives, IN is defined in CCITT's draft recommendation I.312/Q.1201 [5] as an architectural concept for the operation and provision of new services which is characterised by.

- extensive use of information processing techniques;
- efficient use of network resources,
- modularisation and reusability of network functions,
- integrated service creations and implementation by means of the modularised reusable functions,
- flexible allocation of network functions to physical entities;
- portability of network functions among physical entities,
- standardised communication between network functions via service independent interfaces,
- service subscriber control of some subscriber-specific service attributes,
- service user control of some user-specific service attributes,
- standardised management of service logic

The specification and deployment of networks that will meet all the objectives of the IN target architecture will take many years, therefore a phased standardisation process has been recommended. The recommendations will be made in the form of Capability Sets (CSs), starting from CS 1. CS 1 is a realistic initial set of IN capability, which is both technically implementable and commercially deployable [6].

2.2.1 Intelligent Network Conceptual Model (INCM)

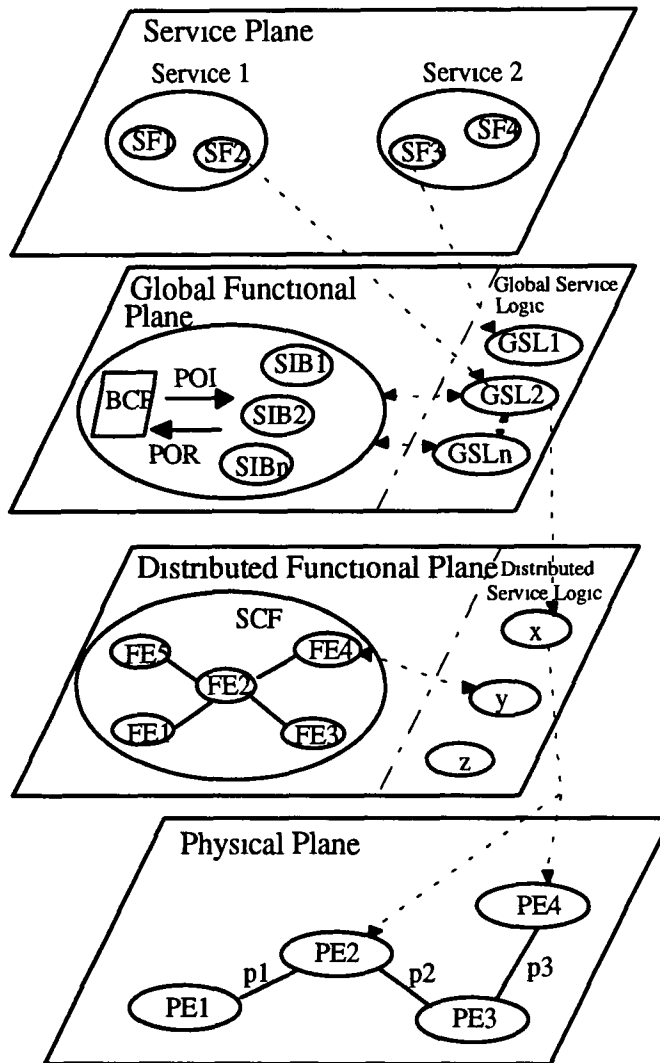
To design the Intelligent Network, the IN Conceptual Model was defined [5][9], to serve as a modelling tool. This model has a four plane representation which shows IN as an integrated framework, within which all other IN concepts are identified, characterised and related to each other. The four planes of this conceptual model, shown in fig 2.1, represent different levels of an IN's abstraction. These planes address service aspects, global functionality, distributed functionality and physical aspects of an IN.

The *Service Plane (SP)* represents an exclusively service oriented view. In this plane, services are described to the user only in terms of how they behave. It does not give the user any information on how the service has been realised on the network. The services are composed of a set of generic blocks called Service Features (SFs). Examples of services include Freephone, and Automatic Call Distribution.

The *Global Functional Plane (GFP)* provides a view of the distributed functionality of an IN network. In this plane, the network is considered as a single entity, and this plane contains a call-processing model and service independent building blocks (SIBs). SIBs can be defined as meaningful units of reusable network functions used to design services and service features. The actual physical location of a SIB need not be considered, since it resides in the GFP, and hence is completely independent from any physical architecture considerations. Examples of SIBs include Translation, Call Control, and User Interaction.

The *Distributed Functional Plane (DFP)* provides a view of the distributed functions of IN. The functional entities (FEs), their actions (FEAs), and their relationships are identified in this plane. These FEs are service independent, so that they can support a wide range of services. Call Control Function FE and Service Control Function FE are some examples of FEs.

The *Physical Plane (PP)* models the physical aspects of the IN-structured network. It identifies the different types of physical entities, the functional entities they realise, and the protocols by which they communicate. Service Switching Point is an example of a physical entity.



Legend

SF	Service Feature	SIB	Service Independent Building Block
FE	Functional Entity	PE	Physical Entity
GSL	Global Service Logic	DSL	Distributed Service Logic
SMF	Service Management Function	SCF	Service Control Function
SSF	Service Switching Function	POI	Point of Initiation
POR	Point of Return	I	Interface
BCP	Basic Call Process		

Fig 2 1 IN Conceptual Model

Service logic is represented on these planes in different ways. In the Global Functional plane, for each Service Feature there is a Global Service Logic (GSL). Each GSL is built of one or more SIBs. Again each GSL in the Global Functional plane, comprises of one or more Distributed Service Logic (DSL) programs in the Distributed Functional plane. The DSLs reside in the Service Control Function FE. These service logic programs may be loaded and executed in the physical entity containing the Service Control Function FE (e.g. Service Control Point), in the Physical plane.

There exist relationships between the adjacent planes of the INCM. The service features within the Service plane are realised in the Global Functional plane by a combination of Global Service Logic and SIBs. Each SIB, in turn, is mapped by one or more FEs in the Distributed Functional plane. The FEs identified in the Distributed Functional plane are further mapped by Physical Entities (PEs) in the Physical plane. Fig 2.1 illustrates the INCM, the representation of service logic on different planes, and the relationship between adjacent planes.

From this IN Conceptual Model, the functional and physical architecture have been derived. The functional architecture has been described in section 2.2.2, while the physical architecture has been illustrated in section 2.2.3.

2.2.2 Functional Architecture

The IN functional architecture [9] is composed of Functional Entities (FEs). Fig 2.2 illustrates the FEs that compose the functional architecture, and the relationships between FEs. The FEs can be broadly categorised into the following functional groups:

A. Call Control Related FEs

They include Call Control Function, Service Switching Function, Call Control Agent Function, and the Specialised Resource Function.

The *Call Control Function (CCF)* provides the connection and basic call related functionality. It does not deal with IN related capabilities.

The *Call Control Agent Function (CCAF)* provides user access to the network.

The *Service Switching Function (SSF)* is responsible for determining which call requires IN service processing, and reports it to the SCF. It interacts with the service logic as well as the call processing for these calls.

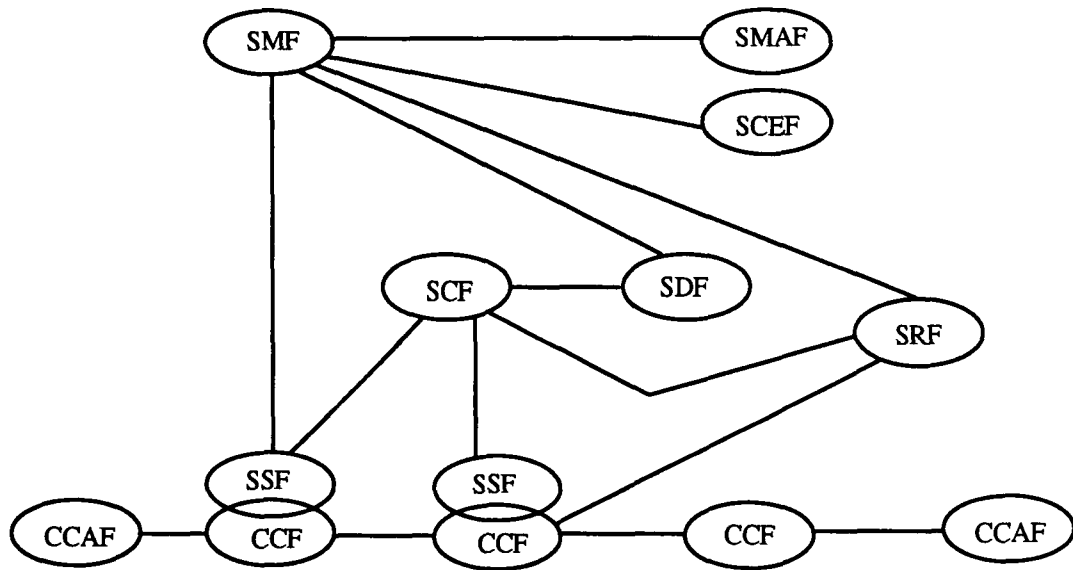
The *Specialised Resource Function (SRF)* provides user-interaction related services to the network. These services include digit collection, speech recognition and synthesis, and announcements among others.

B. Service Control Related FEs

Service Control related FEs include the Service Control Function and the Service Data Function.

The *Service Control Function (SCF)* contains functionality for providing service logic processing. It interprets service requests from the SSF, and provides the necessary service.

The *Service Data Function (SDF)* manages the service and subscriber related data, and provides a consistent logical view to the SCF in real time.



Legend

- | | | | |
|------|---------------------------------------|------|------------------------------------|
| CCAF | Call Control Agent Function | CCF | Call Control Function |
| SCF | Service Control Function | SDF | Service Data Function |
| SRF | Specialised Resource Function | SSF | Service Switching Function |
| SMF | Service Management Function | SMAF | Service Management Access Function |
| SCEF | Service Creation Environment Function | | |

Fig 2 2 IN Functional Architecture

C. Service Management Related FEs

Service Management FEs include the Service Management Function, Service Management Access Function, and the Service Creation Environment Function

The *Service Management Function (SMF)* provides the real time operations functionality for service management, billing, memory administration, testing, traffic management, etc

The *Service Management Access Function (SMAF)* provides the operator access to the service management functions.

The *Service Creation Environment Function (SCEF)* provides the service designer an interface to create and test new services, or modify existing services

2.2.3 Physical Architecture

The functional entities mentioned above must be mapped on to physical components to realise the IN network. A mapping of the FEs on to the physical entities is given below.

Service Switching Point (SSP) contains SSF and CCF functionality. It may provide CCAF functionality if subscribers are directly connected to it.

Service Control Point (SCP) and Adjunct contain SCF and SDF functionality. The difference in the systems lies in the connection between them and the SSP. In case of SCP, the connection is established over a Signalling system 7 network. The Adjunct is connected to the SSP with a direct high speed link.

The Intelligent Peripheral (IP) contains SRF functionality.

The Service Node (SN) contains SCF, SRF and SDF functionality.

The Service Management System (SMS) contains SMF functionality, and optionally may contain SMAF functionality.

Since the IN physical architecture is similar to the AIN physical architecture, the systems mentioned above are described in greater detail in sections detailing the AIN physical architecture.

2.3 Bellcore's Advanced Intelligent Network (AIN)

The Advanced Intelligent Network architecture proposed by Bellcore is conceptually similar to the CCITT IN Recommendation. Although no planar model depicting AIN as an integrated network has been suggested by Bellcore, the functional as well as physical architecture have many common features. At present the terminology for the two architectures differs, but in future releases of AIN and IN Capability Sets it is planned to converge. A detailed description of the functional architecture has been given in section 2.3.1, while the physical architecture is illustrated in section 2.3.2. The AIN Call Model is given in section 2.4, where the call handling mechanism, and the Call State Models have been described.

2.3.1 Functional Architecture

The functional architecture outlines a view of the capabilities required from an AIN network to realise its objectives. In order to provide a network which allows for easy and

quick introduction of new services by the telecom operator, a new approach of modularising the functionality has been advocated in this architecture

The functionality required by the AIN is contained in *Functional Groups* and in *Operations Applications (OAs)* [3] These functional groups are composed of one or more Functional Entities (FEs), an FE can be defined as a set of functions that provide one or more specified capabilities The functional groups are contained in physical systems which may be deployed to realise an intelligent network

The operations functions provide the capabilities needed to support the AIN architecture They exist in the FEs and OAs of the architecture The operations functions in the FEs manage operations within one functional group and interact with the OAs

2.3.1.1 Functional Entities (FEs)

An FE is composed of characteristic functions that uniquely define the FE type A set of FEs grouped in functional groups may be required to realise any component of the network. Therefore to provide the desired functionality, the FEs need to collaborate with member FEs of the functional groups they belong to. To allow the FEs to interact interrelationships between FEs and between FEs and the users of the network have been defined These relationships can be of three types, control, user access, and transport Control relationships exist between FEs within the architecture Functional users communicate with the network through user access relationships The transport relationship exists only between two functional users, or between a user and a Service Assistance FE

Seven FE types have been identified in the AIN functional architecture They include the Network Access FE, Service Switching FE, Service Logic & Control FE, Information Management FE, Service Assistance FE, Automatic Message Accounting FE, and Operations FE They are briefly described below

2.3.1.1.1 Network Access (NA) FE

The Network Access FE is responsible for providing the users access to the network It acts as an interface between the user and the network by providing a consistent view of the network to the user and vice-versa When a user wishes to use the network, this FE is responsible for detecting the network access request and presenting it to the system It may provide bearer capability information to the Switching System FE if required For example, if a user keys in the telephone number of the user he wants to call, this information is forwarded to the SS FE in the form of a Information Collected message From the network side if a request to provide the user a ringing tone is received, it is

responsible for interpreting the request and sending an activate ringer signal to the user equipment

2.3.1.1.2 Service Switching (SS) FE

The SS FE provides generic call processing capabilities. It monitors call related processing to detect requests for AIN services. It maintains relationships with the Network Access (NA) FE, Service Logic and Control (SL&C) FE, and the Information Management (IM) FE to detect service requests. It performs this function at specified points in the call, from information it receives from the NA FE and the trigger criteria information it receives from the IM FE. If a service request is detected, it informs the SL&C FE of the event and sends the appropriate information to it. After receiving a response to its query, it responds with the required actions to provide the service.

2.3.1.1.3 Service Logic and Control (SL&C) FE

The SL&C FE forms the heart of the service logic control systems. It provides the logical control applied to a call that requires AIN service processing. It interacts with the SS FE to provide this functionality. On receiving a service request from the SS FE, it interprets the request. According to the requirements, it invokes the appropriate Service Logic Program (SLP), which it receives from the IM FE. It then sends the instructions to the SS FE to enable it to provide the service. (An SLP is a programmed set of activities that result in the performance of specific tasks, for example analysing digits)

2.3.1.1.4 Information Management (IM) FE

A Large volume of persistent information is required to be stored and accessed by different functional entities. Examples of persistent information include subscriber information, network usage records, SLPs etc. This data handling functionality is provided by the Information Management FE. It provides the other FEs with a consistent data view.

2.3.1.1.5 Service Assistance (SA) FE

The SA FE is responsible for providing participant-interaction with the network. It manages the resources required for these interactions. For example, allocation of announcement devices, and digit collectors.

2.3.1.1.6 Automatic Message Accounting(AMA) FE

The main responsibility of the AMA FE is generation of records from the data it receives from other FEs. The network usage data delivered to it by the SL&C FE and SS FE is organised in predetermined formats, and delivered to the billing entities. The AMA FE may generate records for any other FE requiring its services.

2.3.1.1.7 Operations (OP) FE

The OP FE provides real-time operations functionality for memory administration, surveillance, testing, traffic management, and data collection. The FE acts as interface between the network and the OAs. It monitors the resources for occurrence of any abnormal events, and reports them to the Operations Applications (OAs). It also communicates with other FEs to send information regarding event detection and error reporting. It performs actions to quantify and protect service resources by organising event data from other FEs, monitoring for alarm conditions, identifying error conditions, and organising data to send to the OA.

2.3.1.2 Functional Groups

The above mentioned functional entities are organised into *Functional Groups*. These functional groups reside in different physical systems. Some of these systems may be deployed to realise an AIN network. The FEs are grouped into four functional groups. They are as follows:

1. The *AIN Switch Capabilities (ASC)* contains the functionality included in the NA, SS, IM, SA, and OP FEs. AIN Switching Systems contain ASC functionality.
2. The *Service Logic Execution Environment (SLEE)* contains the functionality included in the SL&C, IM, AMA, and OP FEs. SLEE functionality can reside in Service Control Points and Adjuncts.
3. The *Service Node Execution Environment (SNEE)* contains functionality equivalent to the SLEE, i.e. the SL&C, IM, AMA, and OP FEs. Service Nodes provide SNEE functionality.
4. The *Resource Control Execution Environment (RCEE)* contains the functionality included in the SA, IM and OP FEs. The RCEE functionality is deployed in Intelligent Peripherals and Service Nodes.

2.3.2 Physical Architecture

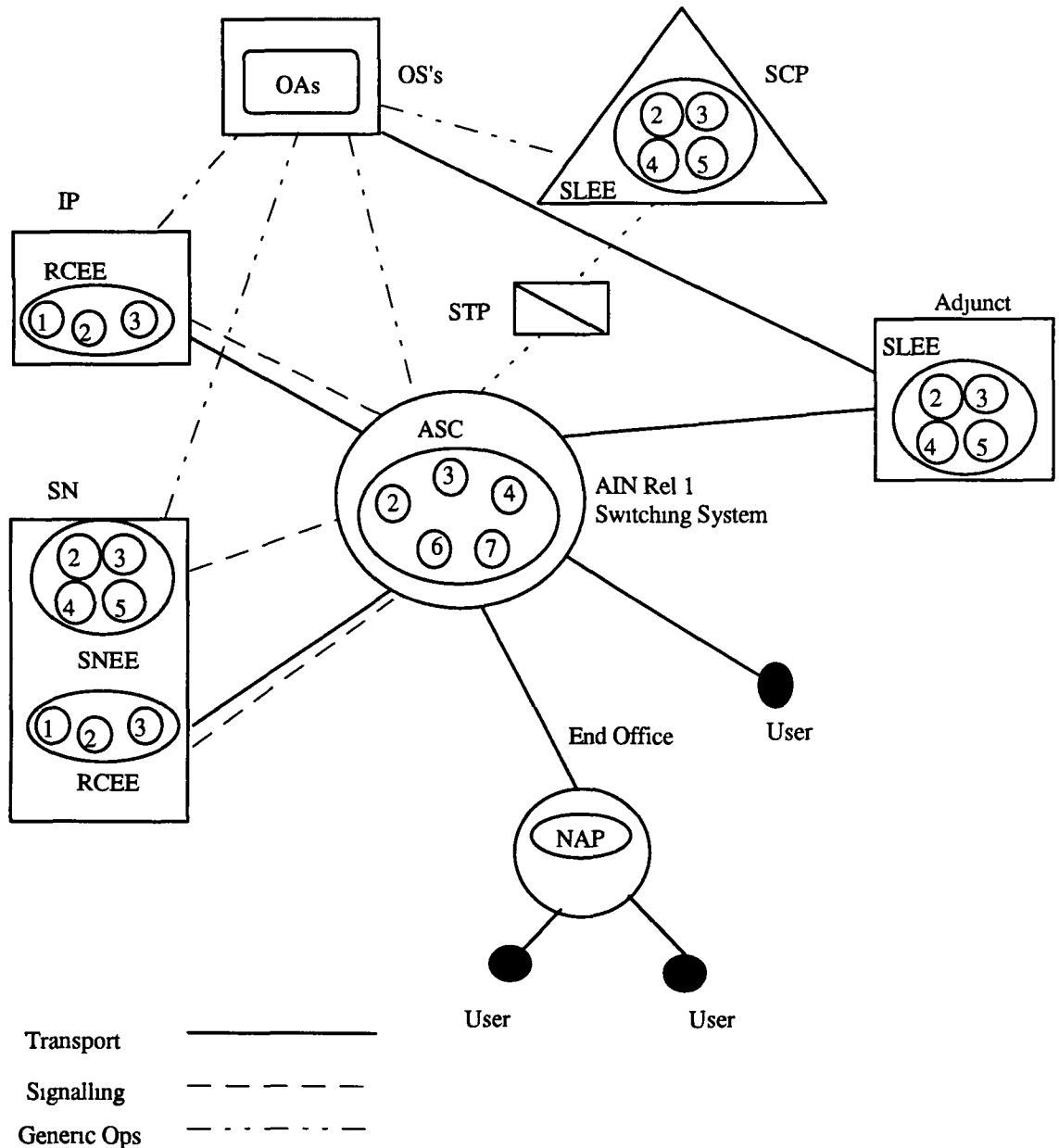
The physical architecture is an implementation of the functional architecture. The elements identified in the physical architecture are characterised by the functional groups that reside in them. Not all of these elements need to be necessarily deployed in realising a network. Depending on various factors, including business demands, types of service required, and existing network characteristics, the telecom operator may select the components for deployment [17][18]. A sample physical architecture has been illustrated in Fig 2.3, it also shows the FEs that reside as part of functional groups in each network element.

While individual network implementations may vary, all share some common characteristics. All networks must deploy the network elements in such a way that there exists at least one instance of each functional group on the network. Therefore there must be at least one system supporting ASC functionality, another supporting SLEE/SNEE functionality, and finally one supporting RCEE functionality. As a result many configurations are possible. The characteristics found in all network implementations include the following [18]

- AIN services are provided through interactions between switching systems and the systems supporting AIN service logic. If the switching system detects the need for AIN service processing in a call (i.e. it encounters a "trigger"), it formulates a query, and sends it to the AIN service logic support system. On receiving a response to the query, it takes necessary action.
- SLEE/SNEE capabilities may reside in more than one type of system. Depending on various factors, AIN service logic may be provided at a *Service Control Point (SCP)*, *Adjunct system*, *Service Node (SN)*, or within a programmable platform provided at the switching system.
- To provide interaction between users and the network, the RCEE functionality may be deployed in the switching system or in an *Intelligent Peripheral (IP)*. The interaction may involve the provision of service-specific announcements to a user or the collection of digits input by a user. A Service Node may be used to provide participant interaction capabilities and AIN service logic in a single network system.

2.3.2.1 AIN Switching System

An AIN Switching System is any switching system containing AIN Switching Capabilities (ASC) functionality. It forms the hub of the AIN architecture, responsible for interacting with all associated elements of the network. It contains the functionality to identify calls requiring AIN service logic processing. On detection of such events in a call, it formulates a query for the AIN system containing service logic, and suspends further processing for that call. After receiving instructions from the service logic, it responds with the requested actions and then resumes call processing. Other functions performed by an AIN Switching System include providing network access to users, generic call processing capabilities to the service logic, routing of messages to other network systems, storage and management of information required for its functioning, monitoring and manipulation of its resources, and gathering data (e.g. network usage).



Legend

1	Service Assistance (SA) FE	2	Operations (OP) FE
3	Information Management (IM) FE	4	Automatic MessageAccounting(AMA) FE
5	Service Logic and Control (SL&C) FE	6	Service Switching (SS) FE
7	Network Access (NA) FE		
SCP	Service Control Point	IP	Intelligent Peripheral
SN	Service Node	OS	Operations Systems
NAP	Network Access Point	STP	Signal Transfer Point
SLEE	Service Logic Execution Environment	SNEE	Service Node Execution Environment
ASC	AIN Switch Capabilities	RCEE	Resource Control Execution Environment

Fig 2 3 AIN Architecture

The AIN Switching System capabilities may be deployed in an access tandem, local tandem or end office. If acting as an access tandem or local tandem, it is able to provide limited AIN services to users connected to subtending switching systems. However, only those users directly connected to an AIN Switching System can access the full complement of AIN services.

Switching systems that are not equipped with ASC capabilities may or may not be equipped with *Network Access Point (NAP)* capability. Those that are NAP-equipped can provide access to certain AIN originating services for their subtending subscribers. A NAP is capable of detecting AIN service requests in calls, and routing those calls to the AIN Switching System that serves it [1].

An AIN Switching System may also contain a SLEE. Some switch vendors are interested in integrating the Switch and Service Control Point into one element - the Service Switching and Control Point (SSCP) [19].

2.3.2.2 Service Control Point (SCP)

The Service Control Point provides a Service Logic Execution Environment (SLEE). It responds to the queries sent by the ASC, by executing the service logic, and sending messages to the ASC to perform actions necessary for providing the requested services. The SCP may also invoke service logic as a result of some internal request generated by some other service logic program (SLP). It is also capable of managing resources (e.g. allocating digit collectors), and data (e.g. measuring, collecting, formatting and outputting subscriber network-usage data to billing entities).

It is interconnected with the AIN Switching Systems through *Signal Transfer Points (STPs)*, and communicates over the *Common Channel Signalling system number 7 (SS7)* network, to process AIN calls. The Transaction Capabilities Application Part (TCAP) is used as the application layer protocol for communicating over the network. Because of this characteristic, SCPs are well suited to support network support capabilities such as those required for 800/Freephone and Person Locator services [18].

2.3.2.3 Adjunct

The Adjunct like the SCP also provides SLEE functionality, however it has a direct communication link to the AIN Switching System. Communication between an Adjunct and an AIN switching system is over a high-speed interface operating at 45 Mb/s. This system may be used for supporting services that require quicker responses from the system (e.g. for services that control provision of dial tone to user).

2.3.2.4 Service Node (SN)

The Service Node provides a Service Node Execution Environment (SNEE), as well as a Resource Control Execution Environment (RCEE). The SNEE much like the SLEE, supports execution of AIN SLPs. However, the Service Node may be specialised to support a specific service or set of services rather than supporting the full range of network functions being defined for the SLEE. The SN communicates with the AIN Switching System directly by means of ISDN access links. The RCEE supports user interaction, that includes collecting dialled digits or spoken input from users and providing customised announcement to users.

The AIN service logic can request the AIN Switching System to connect a user to a resource located in an SN that is connected to the AIN Switching System from which the service was detected. The service logic can also request the AIN Switching System to connect a user to a resource located in an SN that is connected to another AIN Switching System.

2.3.2.5 Intelligent Peripheral (IP)

The Intelligent Peripheral contains RCEE capability, that controls and manages IP resources such as voice synthesis, voice recognition and DTMF digit collection. The AIN Switching System routes a call to the IP, as necessary, to support the request of such resources by AIN service logic. In establishing the call connection to an IP, the AIN switching system also passes the message parameters that instruct the IP to perform specific user-interaction functions. Then it returns any information collected from the user to the AIN Service Logic Node, which made the request, via the Switching System.

Additionally, the IP provides basic message handling functions for all incoming and outgoing messages related to services and operations. It monitors its resources for errors, and reports their status to the SLEE. It is also responsible for collecting usage information for billing entities.

2.3.2.6 Operations Systems (OS)

An Operations System is a software system that implements Operations Applications, which perform operations tasks in the network (e.g. network traffic management for the network systems). The OSs have interfaces to the SCPs, Adjuncts, AIN Switching Systems as well as have interfaces to the other OSs. They provide a network view of the services, which is used for subscriber trouble shooting, service negotiations, planning and engineering processes. An example of an OS is a Service Management System (SMS).

2.4 AIN Call Processing

This section describes how the AIN architecture manages call-processing events. It describes basic AIN Release 1 call processing and AIN Release 1 call processing that requires network resources.

2.4.1 AIN Release 1 Basic Call Processing

When an AIN switching system receives a call, it examines the internal information (i.e. trigger criteria) and information received from the user (e.g. digits) to determine if the call involves an AIN service request. It detects these requests at predetermined points in the call processing known as *Trigger Check Points (TCPs)*. If such a request is detected, then it sends a message to the SLEE in an SCP, Adjunct or Service Node identified for the trigger encountered. The message is populated with information that provides a logical view of the call to the service logic. The Switching System then suspends processing events for the call if it expects a reply from the AIN service logic. On receiving a response from the SLEE, it executes the requested actions, and resumes call processing. The logical view of the call presented to the SLEE, includes the connectivity attributes of the connection view and the call processing information that the *Basic Call State Model (BCSM)* contains.

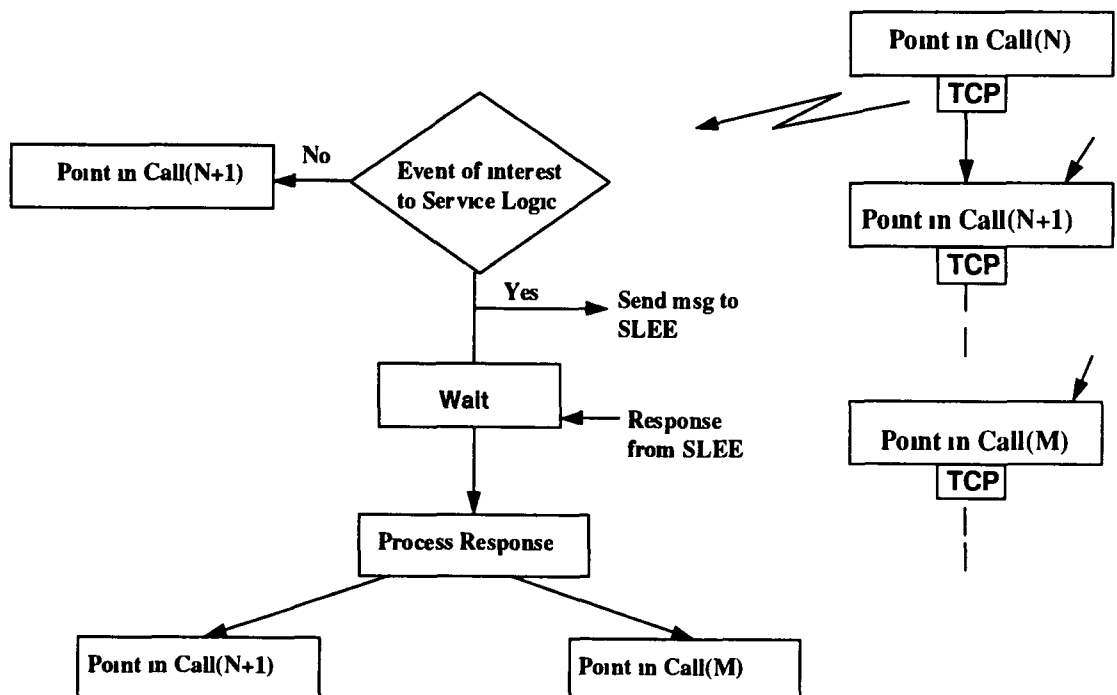


fig 2.4 - Basic Mode of Operation for AIN Call Processing

On the SLEE side, when a message is received from the Switching System, it checks if an instance of a Service Logic Program (SLP) for that feature (defined as an Feature

Service Logic Program) has been already invoked for the requesting user. If not, then it invokes the appropriate FSLP, and forwards the essential information to the FSLP. The Feature service logic, then specifies the actions it wants the switch to perform in order to provide the service to the user. This call processing mechanism has been illustrated in fig 2.4.

2.4.2 AIN Release 1 Call Processing Requiring Network Resources

In processing a call, there might be a need to connect a user to a resource (e.g. announcements or digit collection) so that the user and the network can exchange information. The SLEE maintains information on the available resources and their location on the network. Therefore if a particular resource is required for user interaction, it requests the Switching System to connect the user to the resource. The Switching System, in turn requests the RCEE in the IP/SN which controls the required resource, to set up an interaction with the user. In case the resource resides in an IP/SN not directly connected to the Switching System where the service was requested, it requests the Switching System to route the call to the remote Switching System to which that IP/SN is connected. After information has been exchanged with the user, the RCEE returns the collected information to the Switching System. The Switching System then clears the connection between the user and the resource and returns the collected information to the SLEE.

2.4.3 Call Model

The call model is an abstraction of the aspects of call processing that determine the SLP operation. It defines the call processing functionality of the architecture and the relationship that exists between the SLEE in the SCP or Adjunct and the ASC in the Switching System. It needs to be maintained during the duration of the call.

The Switching System is viewed as having two functionally separate sets of call processing logic that co-ordinate call processing activities to create and maintain a basic two-party call. One set of logic provides call origination processing in response to a call origination request and controls the establishment of that portion of the call referred to as the originating call portion. The other set of logic provides call termination processing for the intended recipient of the call and controls the establishment of the terminating call portion. During call setup information is exchanged between the originating and terminating call processing logic in the Switching System. Aside from this necessary exchange of information, originating and terminating call processing logic operate independently of one another with respect to the calling and called parties, respectively.

The AIN call model consists of two components: Connection View and Basic Call State Model (BCSM) The following sections give a description of the Connection View, and the Originating and Terminating Basic Call State Models

2.4.2.1 Connection View

A connection view is a generic representation of the call processing resources accessible to a service logic program at the SLEE It is provided to the SLEE in the form of a *Connection Segment (CS)* A Connection Segment is either the originating or terminating portion of a call within an ASC The two connection segments together form a connection view for a two-party call

The connection view represents the connectivity attributes of a CS - the connection points and legs The view consists of a connection point, which represents the interconnection of legs, and one or more legs, which represent communication paths towards some addressable entity Legs are of two types Controlling Leg and Passive Leg A *Controlling Leg* represents the user for whom the service is invoked The ASC always assigns a Leg ID of 0 to the controlling leg There is only one controlling leg per CS This leg must be present and connected to the controlling point for the user to communicate with other parties of the CS Leg 0 represents the signalling and transport path to the user For the terminating leg, this leg is not present until the terminating party is alerted. A *Passive Leg* represents the communication path to a terminating access of the Switching System

The *attributes* assigned to the connection point include the CS ID, number of legs and bearer capability. Attributes of the leg include the Leg ID, Leg Status (connected or unconnected), and Leg State The Leg State is the current state of the BCSM that includes the leg

When ASC detects a trigger, it sends a message to the SLEE, in which it provides the SLEE with a connection view and its attributes The ASC assigns the CS ID, which is used in all messages between the ASC and SLEE that pertain to a specific CS The ASC only creates a connection view if it determines that a SLEE should be involved in the processing of a given CS.

Fig 2 5(a) represents an originating CS that is being set up Its connection view has a Leg 0 and a connection point, with Leg 0 pointing towards the originating party When the call progresses beyond set-up, the connection view has a Leg 0, a Leg 1, and a connection point, as shown in fig 2 5(b) Leg 0 represents the calling or called party, depending on whether the CS is originating or terminating The connection view for a

terminating CS that is being set up has a Leg 1 and a connection point, where Leg 1 represents the Switching System trying to terminate the call on the called party (see fig. 2.5(c)).

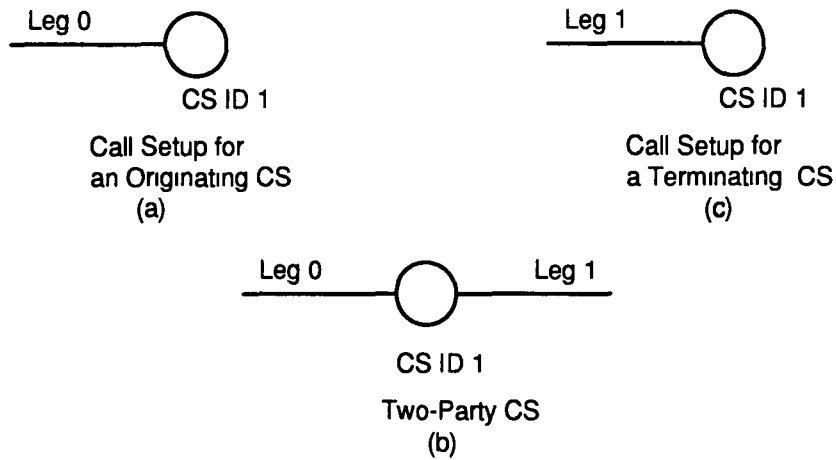


fig 2.5 Connection Views

Several connection views may be made visible to SLPs for a single two-party call. For intraoffice calls, there is only one originating segment and one terminating segment. For interoffice calls, two or more originating and terminating segments may exist, one for each ASC. As shown in fig. 2.6 there are four independent connection views to represent a two-party interoffice call. The control of each segment of a call by an SLEE is independent of the control of other segments of the call by the same or a different SLEE.

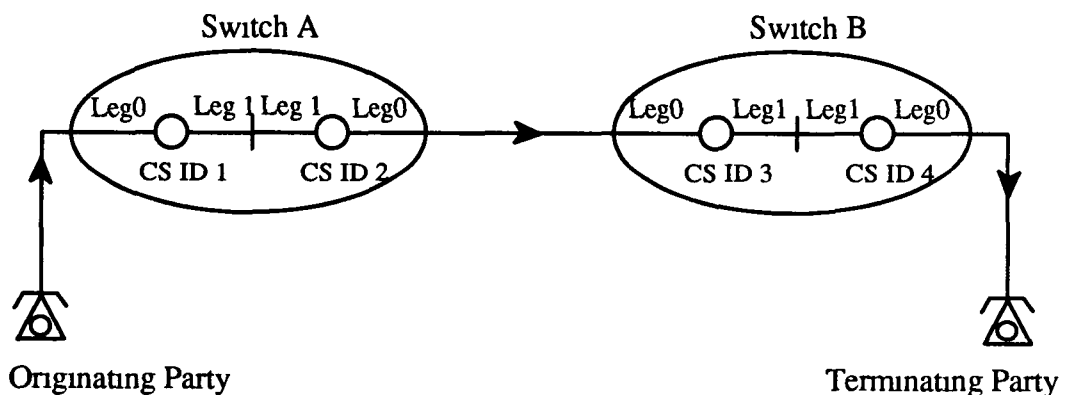


fig 2.6 Illustration of a Two-Party, Interoffice Connection

Associated Connection Segments

The call models permits association between CSs - a relationship between two CSs that share a common Leg 0. SLPs can affect the two CSs independently. The association

allows an SLP to affect both the CSs concurrently. Only two special actions that affect the two CSs are permitted: merging the two CSs or moving legs between the CSs. Only two CSs can be associated at a time. One CS can be multiway (i.e. more than one passive leg), but the other must be a two-party CS (i.e. only one passive leg).

To explain the use of an association, let's take the case of the working of Three-Way Calling (see fig. 2.7(a)). CS ID 1 denotes a stable two-party CS. The subscriber represented by Leg 0 flashes a switch-hook to obtain a dial-tone for a second call (CS ID 2). The ASC sets up CS 2 when requested by the SLP, and associates CSs 1 and 2. Leg 0 represents the subscriber in control of both the CSs. The subscriber flashes again after CS 2 is set up to request the CSs to be merged. When the switch-hook flash occurs on CS 2, the connection view for CSs 1 and 2 is included in the message the ASC sends the SLEE. The SLP that requested the set up of CS 2 requests the ASC to merge the CSs to form a single CS with one connection point and three legs. When the ASC merges the two CSs into one CS, it must renumber the legs in the connection view. One way is to renumber the passive leg of the CS that is merged into the other CS. Here, CS 2 is merged into CS 1. Thus Leg 1 of CS 2 is renumbered as Leg 2 of CS 1. (See fig. 2.7(b)).

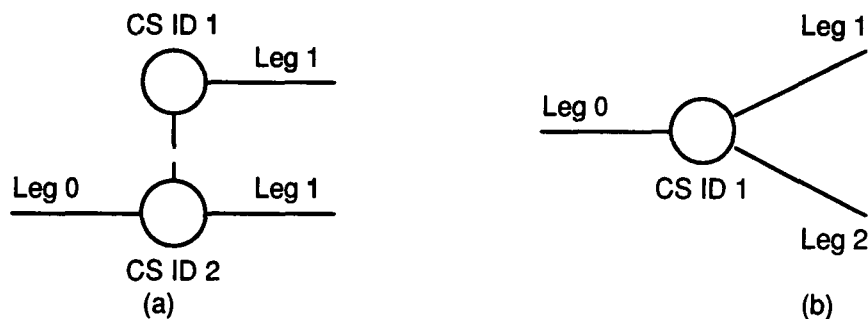


fig. 2.7 Connection Views (a) Associated CSs (b) Three-Way CS

2.4.2.2 Basic Call State Model (BCSM)

The Basic Call State Model represents the states and identifies the events that are encountered at which AIN services may be invoked. The BCSM is split into the originating and terminating parts. The originating part models the processing related to the originating CS. The terminating part models the processing on the terminating CS.

The switch-based processing considered essential to establish, maintain and clear a two-party call are represented by *Points in Call (PICs)*. The points where call processing can be interrupted to notify the SCP of a given event and allow it to influence subsequent call processing, are called *Trigger Check Points (TCPs)*. They lie between the PICs. The originating and terminating BCSMs are shown in the fig. 2.8 and fig. 2.9, respectively.

For the originating BCSM, call setup is represented by the first six PICs. For the terminating BCSM, call set-up is represented by the first four PICs. A stable PIC is one that has progressed beyond the call setup phase. Mid-call events occur while a CS is stable, and allow call processing to return to the same PIC from which the events were detected. Timer expirations and feature requests are examples of these events. These events interrupt call processing and are immediately reported to the SLEE.

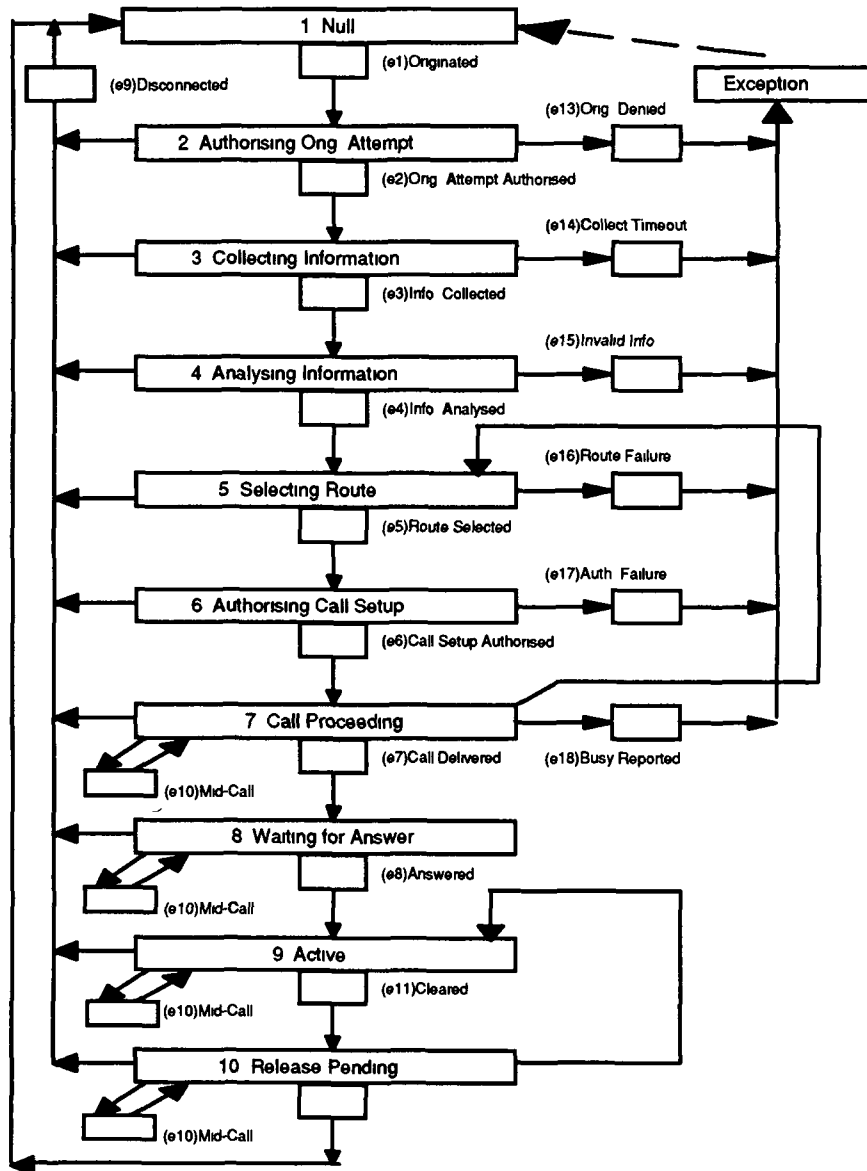


fig 2 8 Originating BCSM

Events that cause the call processing associated with a PIC to be initiated are called Entry Events. After entering a PIC, call processing related with that PIC is performed. Events that signify normal completion of the call processing associated with this PIC, are

regarded as Exit Events for the PIC Exit Events related to incomplete calls include the events that lead to the Exception box and the Disconnect and Cleared events, when they are received before a call is answered

Interaction between the Originating and Terminating BCSM first takes place at Call Proceeding PIC, and then continues throughout the life of the call A detailed description of the call models can be found in [3]

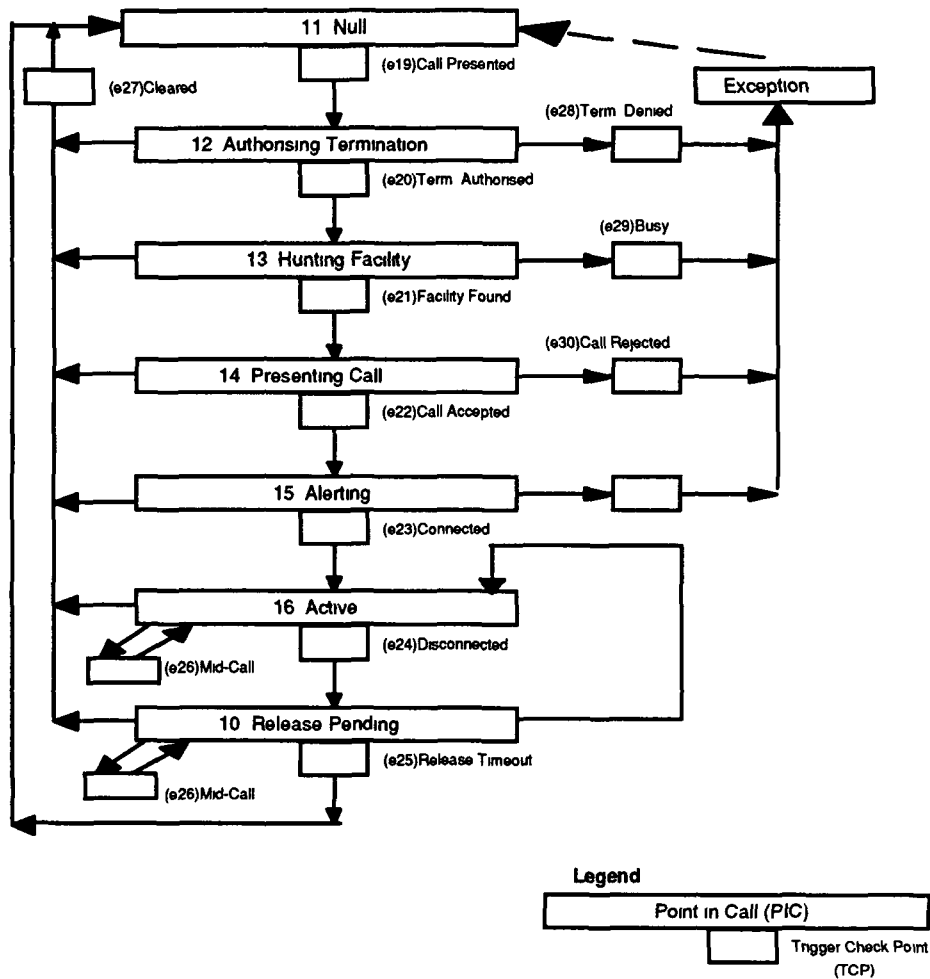


fig. 2.9 Terminating BCSM

The degree to which any given call can be broken down and controlled by the service is determined largely by the number of TCPs supported by the call model. In theory, a large number of TCPs would enable an extensive set of services to be created and controlled by the services. This has resulted in the call models prescribed by the AIN Release 1 architecture. However, the necessary additional complexity in the call model consumes a commensurate increase in real time processing power affecting the capacity of the switch. Therefore the goal for initial deployment is to identify and implement a modest

set of TCPs that still enable a rich variety of services. The TCPs identified for deployment in AIN Release 0.1 [4] include Origination Attempt, Collected Information, Analysed Information, Route Select Failure and Termination Attempt. In the system developed, Information Collected, Busy, Mid Call and Termination Authorised TCPs have been implemented, to provide a few of the AIN services.

2.5 AIN Service Example - Freephone Service

A Freephone telephone number is not a real number. It is a logical number assigned to a subscriber, which allows the callers to ring the called party without knowing their actual number, and at the same time not be charged for the call [21]. The subscribers of the service pay for the calls. This service stimulates business of the subscribers by increasing sales and reaching new customers inexpensively. It also gives them flexibility to direct calls to the best answering location, and in general improve their efficiency.

On existing networks, telecom operators provide this service to their subscribers, by placing number translation tables on each switch of the network. Whenever a freephone call is detected, the translation tables are looked up to determine the real number. As the service control is not centralised, if new entries have to be added, or old entries modified in these tables, the changes have to be made at every switch providing the service. This being a very cumbersome and tedious process prevents the telecom operator from allowing subscribers to make changes frequently.

The AIN architecture is well suited for providing such services as control of services is centralised. Its suitability can be illustrated by taking the example of Freephone service. When a request for the freephone service (e.g. 1-800-NXX-XXX numbers in Ireland) is detected, a query is sent to the SLEE. The SLEE looks up the translation table in the database to determine the real number. In this network, if the subscriber wishes to modify the number for directing calls, the telecom operator can modify the translation table in the database at only one location. This makes their task much easier. The AIN architecture also allows them to enhance this basic service. It may allow the subscriber to direct the calls to different locations depending on the time of day, place origin of call, and additional information received from caller, among other factors.

A more detailed description of how the service may be realised over the network has been given below. It exhibits the interactions that take place between different systems of the network, and between the user and the network.

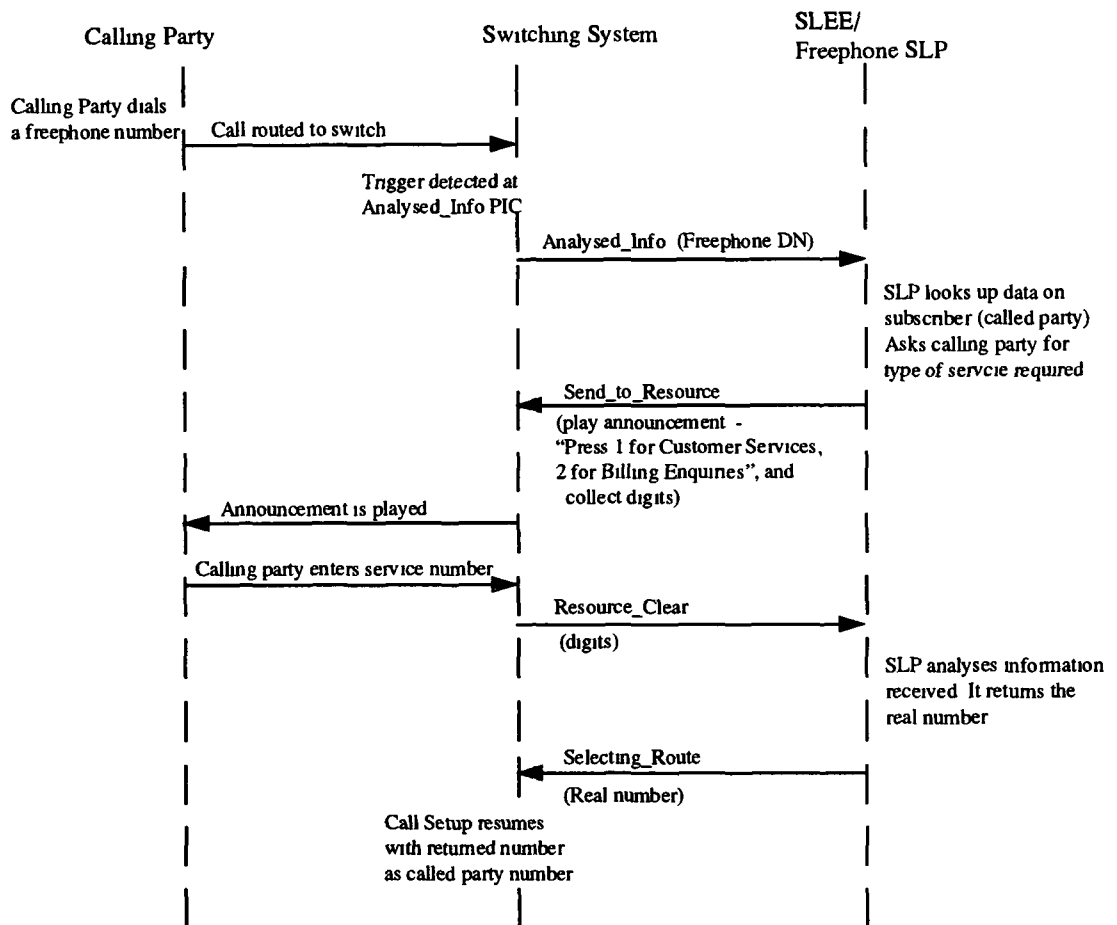


Fig 2 10 Freephone Service

When a call origination attempt is detected by the switch, an instance of the Originating BCSM is invoked for the call. All call related processing required for the call is handled by this call model. When the calling party keys in the digits of the called party, the collected information is analysed at the Analysing_Information PIC of the Originating BCSM. If it contains a special access code (e.g 1-800), a trigger is detected. An Info_Analysed message is formulated and populated with call related information, which provides a connection view of the call to the service logic. The BCSM then suspends any processing related to this call. The formulated message is sent to the SLEE (e.g SCP or Adjunct), which provides the service. At the SLEE, the Freephone SLP looks up the data for the called party. If the called party requires additional information from the calling party to determine where the call should be directed, the SLEE sends a request in a Send_to_Resource message to the switch, to prompt the user with an announcement and subsequently collect information. For example, a "Press 1 for Customer Services, 2 for Billing Enquiries" announcement may be played to the calling party. The calling party then enters his request, which is returned to the SLEE in a Resource_Clear message. The SLP in the SLEE analyses the received information and decides according to the subscriber's specifications, what the real number should be. It returns this number to the

switch in a `Select_Route` message. When the Originating BCSM receives this message, it resumes call setup in a normal manner at the `Selecting_Route` PIC, with the returned number as the destination number. Fig 2.10 illustrates this mechanism. The messages used above are explained in Appendix B.

2.6 Comparison between Bellcore AIN and CCITT IN

The aim and objectives of the two architectures are similar, but the modeling in functional terms and realisation in physical terms differs. The IN architecture recommended by CCITT is described using a four planar model, in which the IN concepts are identified, characterised and related to each other. From this model the functional and physical architecture are derived. Bellcore has not represented their AIN architecture in planar form. Instead their architecture is described in terms of the functional and physical architecture.

From the point of view of the functional architecture, the concept of functional entities exists in both architectures. An approximate mapping of the Functional Entities in IN and AIN is shown in table 2.1.

AIN FEs	IN FEs
SS	SSF, CCF
NA	CCAF
SL&C	SCF
IM	SDF
OP	SME, SCEF, SMAF
SA	SRF
AMA	No direct mapping. Functionality present with each FE

Table 2.1 Possible Mapping of AIN FEs to IN FEs

The terminology used in realising the physical architecture is almost identical. The only difference in terminology of network elements is that in AIN the Service Switching Point (SSP) is called the AIN Switching Capability (ASC). The ASC is richer in functionality as compared to the SSP, in that it can provide limited AIN services to non-AIN switches or Network Access Points (NAPs) connected to it [19].

The provision of standard reusable network capabilities in IN is in the form of Service Independent building blocks (SIBs) SIBs are nested in service logic to create service features SIBs have very well defined input and output interfaces and are used as monolithic building blocks providing a single complete activity For service creation purposes, IN tends to suggest the use of high level GUIs where SIBs are manipulated as screen icons This approach puts some limitations on the service designers They can modify services by manipulating data parameters, rather than create original service ideas This goes against one of the main principles of IN, that is to allow for introduction of customised services for its clients AIN offers the service designer an Application Programming Interface (API) in the form of Functional Components (FCs). These FCs are more primitive and allow for lower level manipulation as they are imbedded in C-language programs to realise new services It gives the service designer greater freedom to design more specific applications But this may in itself pose as a problem for the switch vendors. Incorrect manipulation of resources by the service designer may lead to operational problems

Bellcore originally planned to support circuit-switched voice and circuit-switched data services on the AIN networks There was no provision for supporting broadband or multi-media services. But with the change in trends of technology Bellcore plans to look into the provision of multi-media services on AIN systems In the CCITT recommendations for IN, a proposal for providing bearer services, teleservices and broadband interactive services has been made Bearer services include circuit-mode speech, circuit-mode audio, packet switched data services, and circuit-switched data services among others [5] Telephony, telefax and videotex are some of the examples cited for Teleservices Additionally, messaging and retrieval services are some of the broadband interactive services suggested

IN access capabilities also go beyond the AIN access capabilities IN capabilities that are being enhanced in the Capability Sets (CSs), going from CS1 to CSn, foresee the provision of access to fixed networks (e.g. PSTNs, ISDN, PSPDNs), private networks, mobile networks, and broadband networks (e.g. ATM, STM) [5] AIN access capabilities seem to be restricted to fixed networks

Chapter 3

AIN System Design and Implementation

3.1 Introduction

A "system" is usually defined as an ordered set of interrelated physical (or abstract) objects. In order to analyse, design, control, or improve understanding/performance of a specific system, a model may be developed. A "model", therefore, can be defined as a reflection of the modeller's understanding of the system, its components, and their interrelations. It allows the modeller, through simulation, to reproduce the behaviour of the system, and perform the above mentioned activities. But when modeling very large and complex systems, it may not be possible for the modeller to represent the entire system. Instead he may cut the vast subject down to manageable proportions[11]

Simulation is the process of building and experimenting with the system model such that a specific purpose of the study is achieved through observing the model's behaviour under assumptions defined by the modeller[15]. For example, simulation may be performed to check and optimise the design of the system before its construction, thus helping to avoid costly design errors and ensuring safe designs.

We have attempted to simulate a telecommunication system based on the Advanced Intelligent Network architecture, in order to understand the architecture better, and to demonstrate its operation. The AIN Switching System, and the components providing the Service Logic Execution Environment, and Resource Control Execution Environment have been simulated to realise the system. This system is capable of providing basic call control functionality, and a subset of AIN services to the users. The simulated network is described in this chapter, while realisation of the Call Control Mechanism, and the AIN services, have been described in Chapter 4.

OPNET, a network simulation Computer Aided Engineering (CAE) tool, was used for this purpose. OPNET runs on SUN SPARC and HP APOLLO workstations, in a UNIX environment. Before describing the network design, a brief overview of OPNET has been given.

3.2 Overview of OPNET

OPNET, or Optimised Network Engineering Tools, is a hierarchical object oriented simulation tool, designed specifically for the development and analysis of communication networks[13]. Some examples of possible applications include local area networks, mobile packet radio networks, and ISDN architectures. It provides a graphical interface to the user, for specification of models. The models of protocols and algorithms employ a hybrid approach by allowing the user to embed 'C' language code within a graphically laid out finite state machine. The specification of processes in 'C' is facilitated by an extensive library of support functions which provide a wide range of simulation services. It also provides a set of analysis tools to interpret the simulation results in graphical form.

OPNET simulations are based on four separate modeling domains called Network, Node, Process and Link. The dependencies between these modeling domains are shown in the diagram below. As the fig 3.1 illustrates, network models rely on the definition of the node models which in turn incorporate process models. In addition, link models are used to characterise links within the network domain.

In the **Network Domain**, node models are instantiated and each instance may be assigned independent attributes including identification and position, and user-defined attributes. Nodes which are designed to attach to physical links may be interconnected to form arbitrary network topologies.

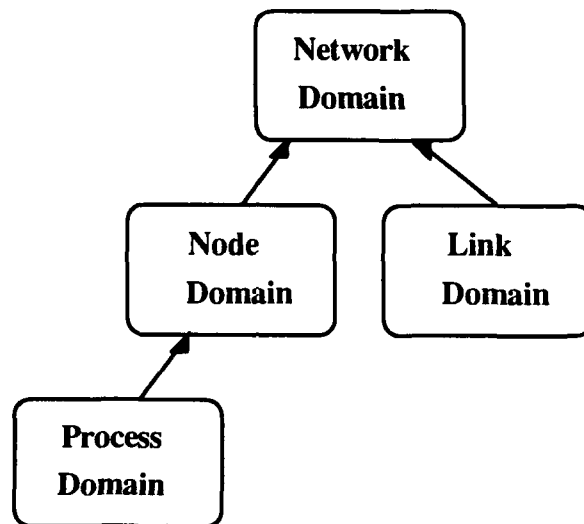


Fig 3.1 OPNET Modelling Domains

The **Link Domain** allows incorporation of custom or user-specific link models within OPNET simulations. These models are specified in C and are linked into the simulation.

Point-to-point links are represented by lines between source and destination nodes. The point-to-point links are unidirectional, therefore a duplex link is represented by two links, one for each direction. The point-to-point links have a number of built-in attributes which can be specified by the user. They include transit delay incurred by packets/frames forwarded over the link, bit error rate which is the probability of bit errors in packets/frames transmitted over the link.

In the **Node Domain**, the internal structure of the nodes is defined. The internal structure of the nodes consists of modules which can generate, process, store, receive and transmit packets and manage resources according to a user defined process. These modules can be interconnected to form arbitrary complex node architectures.

The *Ideal Generator* module provides a convenient stochastic packet source. The frequency of packet arrivals and the length of packets can be controlled by probability distribution. The packets generated can also have a packet format specified.

The *Queue* module executes a process model which defines the communications process that the queue module is required to perform. The process model incorporates 'C' code and simulation kernel procedures to model processing functions of the node. In this way, the queue module's behaviour can be completely specified. The queue module may contain a number of subqueues, each of which can hold a list of packets. The queuing discipline used and the number of subqueues needed in a particular queue module, and the capacity of each subqueue can also be specified. Subqueues are accessed by the process model using subqueue indices in the kernel procedures.

The *transmitter* and *receiver* modules are used for communicating between nodes. A transmitter module of one node is connected to a corresponding receiver module at the destination node via point-to-point links. The maximum data rate for each of these modules can be specified.

Process Models are specified using a graphical editor which captures the structure of the process in the form of a finite state machine. The *finite state machine (FSM)* models a communications process by responding to changes in its inputs, modifying its state and producing new outputs. No loss of generality occurs when using the graphical FSM approach to represent the process model as the state-transition diagram can contain fully general C language code. Process models may make use of a library of kernel procedures which support access to packets, network variables, statistic collection, packet communication and other simulation services.

The two fundamental components of an FSM state are states and transitions. States can be used to represent the significant modes of the process and may have certain actions associated with them. An FSM implements these actions either on entering or on leaving the state. All states can be thought of as being decomposed into three phases of traversal by the FSM, as shown in fig 3.2. The first phase is the enter executives which are always implemented upon arrival in the state. The second is a possible resting phase where the FSM returns from its invocation. And the third phase is the implementation of the exit executives.

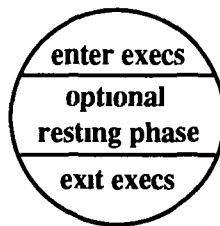


Fig 3 2 FSM Representation

Two types of states are distinguished in implementing OPNET process models: forced and unforced states. *Forced states* bypass the second phase rather than return from the process model invocation. *Unforced states*, on the other hand, always cause the FSM to return from invocation and block immediately after implementation of the enter executives. An FSM will return in the rest phase until a new interrupt is delivered to the process model, causing a new invocation. In fact, interrupts are always delivered to process models when their FSMs are in a blocked condition, and thus necessarily occupying an unforced state. The FSM will continue to execute until the rest phase is implemented by entering an unforced state. In hard copy output, forced states are drawn black, while unforced states are represented in white, as shown in fig 3 3.

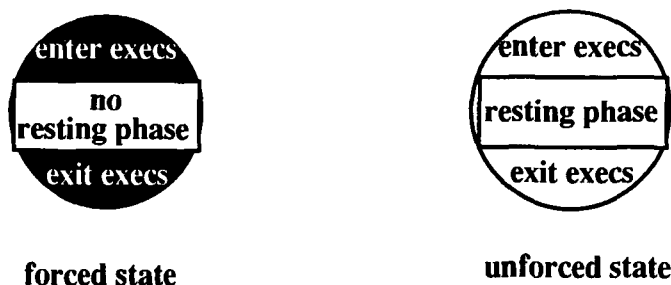


Fig 3 3 Forced/Unforced State Representation

The *transitions* represent possible ways in which the process can migrate from one state to another. These expressions, evaluated as Booleans determine whether a particular

transition will be followed and a new state entered. Since the finite state machine should occupy only one state at a time, only one transition statement should evaluate true at any one time

3.3 AIN Network Model

The network model of the AIN network, as shown in the fig 3.4, consists of the Switching System node, the Intelligent Peripheral node, the Service Logic node (e g Adjunct) and five User nodes The Switching System node forms the hub of the network to which all other nodes are connected The Service Logic node is connected to the Switching System via a 45 Mbps duplex link Two duplex data links, and a duplex signalling link connect the IP to the Switching System The number of data channels between the IP and the Switch can be increased, if required by the IP, provided the IP has the capability to manage additional lines, and if lines are available from the Switching System

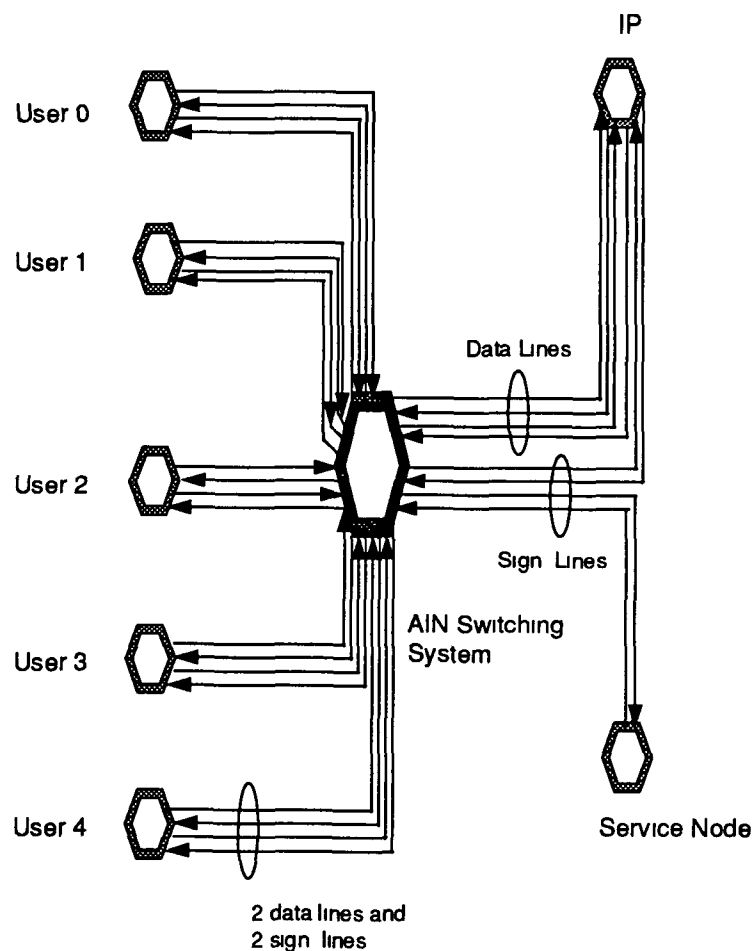


fig. 3.4 - AIN Network Model

The Switching System uses out of band signalling, to exchange signalling information with the user node. In real-time systems, there exists only one physical link, which is subdivided into separate channels for signalling and data. These two channels are simulated by providing two duplex links between each user node and the Switching System. Again, the number of users connected to the Switching System can be increased, by simply adding more user nodes. The Switching System is capable of supporting additional users. The number of users has been kept small, to keep the model simple.

3.4 AIN Node Model

For each component of the developed system there exists a node model. In the following sections the description and working of node models for the Switching System node, Service Logic Node, and Intelligent Peripheral node has been given.

3.4.1 Switching System(SS) Node Model

The Switching System is responsible for providing AIN Switch Capabilities(ASC) functionality. ASC functionality provided by this node model is as follows:

- Connection Control.
- Call Control.
- Reporting of requested events and triggers to the SLEE.
- Communication with the SLEE, within the Service Logic Node.
- Communication with the RCEE, within the Intelligent Peripheral.
- Data Management.
- Resource Status Checking.

The node model, as shown in the fig 3.5, consists of queue modules for the Process Controller and the Switching Fabric, process modules for the Originating and Terminating Call State models, and transmitter and receiver modules for linking the nodes. The abbreviations ASC and SS, are used to refer to the Switching System in this document.

The Process Controller is the master controller of the switching node. It is connected to all the other blocks via packet streams. Within it runs a process which queues the information received from all the sources, removes inserted information from a sub-queue when free, processes it, and if required, sends relevant information, in the form of packets, to the connected modules.

The Switching Fabric is responsible for establishing and releasing connections between two or more users, or between users and resources. It does so on instruction from the

Process Controller It also forwards information received on data lines, on to other data lines, in accordance with the call connection information stored, and thus provides circuit switching functionality

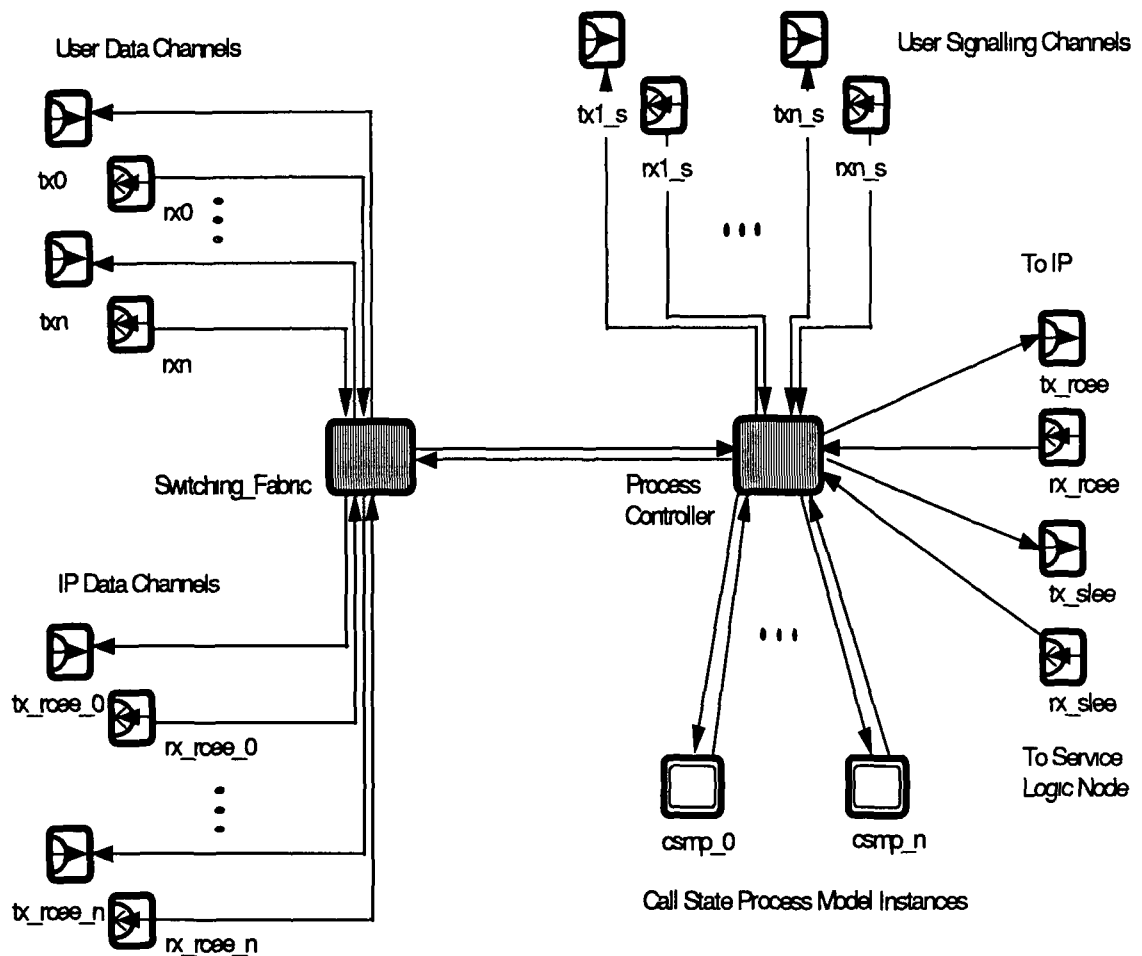


fig 3 5 - Switching System Node Model

The Call State process models are deployed for call processing. They simulate the Basic Call State Models(BCSMs), which are a generic representation of ASC call processing that identify points where call processing may be interrupted or resumed. Each call, which has progressed beyond the setup stage has at least one instance of the Originating and one instance of the Terminating Call State Model invoked for its processing. In case of feature invocation by the user, there may be more than one Call State Model of each type invoked for the call. The number of instances of Call State Models limit the number of calls the Switching System can handle. They can be increased taking availability of memory into consideration.

There are transmitter and receiver modules connected to the Process Controller for communicating with the SLEE, RCEE and users via its signalling paths. The transmitter

and receiver modules used for connecting to the user data streams and RCEE data streams are linked to the Switching Fabric via packet streams

3.4.2 Service Logic(SL) Node Model

The Service Logic node model is responsible for providing Service Logic Execution Environment (SLEE) functionality. Only a small subset of the SLEE functions have been supported by the Service Logic node that we have simulated. It does not strictly adhere to the architecture specified by Bellcore for SCP/Adjunct implementation, but it is conceptually similar. The SLEE functionality that has been provided includes

- Invocation and execution of Service Logic Programs (SLPs)
- Communication with the ASC, resident in the Switching System
- Management of resources
- Management of feature interactions
- Provision of a service creation environment

The Service Logic node model, as shown in the fig. 3 6, consists of a queue module for the Service Manager, process modules for the Service Logic Programs, and a transmitter and receiver module for communicating with the Switching System. The services supported include Call Transfer, Call Forwarding and Call Back.

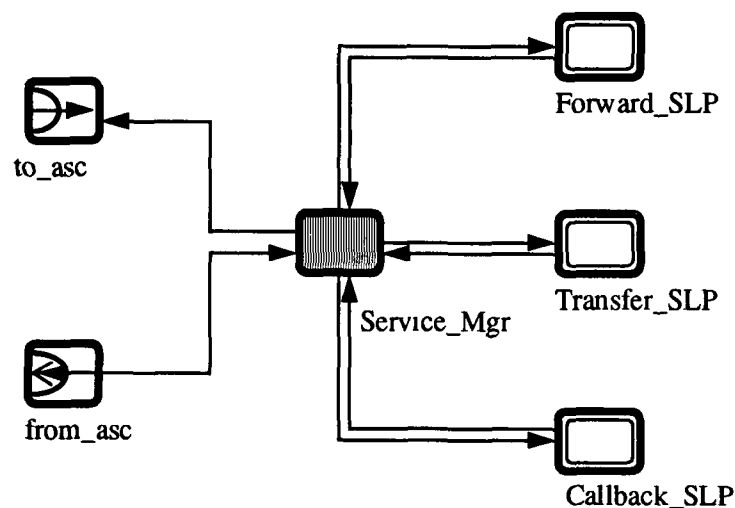


fig 3 6 - Service Logic Node Model

The Service Manager is connected to the SLPs via internal packet streams, and to the Switching System via a physical link. It acts as a message handler. It forwards messages received from the Switching System to the appropriate SLP, and vice-versa.

The SLPs are responsible for management of individual services, and maintenance of feature related information. The SLPs work independently of each other. If any feature interaction exists, it is managed by the Service Manager.

Alternatively to support Automatic Call Distribution (ACD) services, the node model as shown in fig 3.7, consists of a queue module for the ACD service logic program and a receiver and transmitter module for communicating with the Switching System. Only one of the two sets of services can be operational at any given time. Therefore an assumption has been made that all users belong to a single group and subscribe to the same set of services.

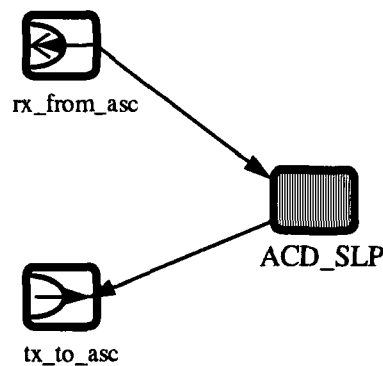


fig 3.7 - Service Node Model, for ACD

3.4.3 Intelligent Peripheral (IP) Node Model

The Intelligent Peripheral node is responsible for providing Resource Control Execution Environment (RCEE) functionality. RCEE functions include:

- Control and management of resources
- Communication with the ASC, resident in the Switching System
- Interaction with the users.

The node model, as shown in fig 3.8, consists of a queue module which manages the resources, and transmitter and receiver modules connecting its signalling and data channels to the Switching System. The number of data lines connected to the Switching Fabric within the switching system depend on the number of resources supported and availability of connections from the switch. In every interaction with a user, resources are allocated by the IP before entering into conversation with the user. After the setup is completed, data may be exchanged between the IP and the user. Finally the IP returns the collected information to the ASC, and releases the resources allocated for this interaction.

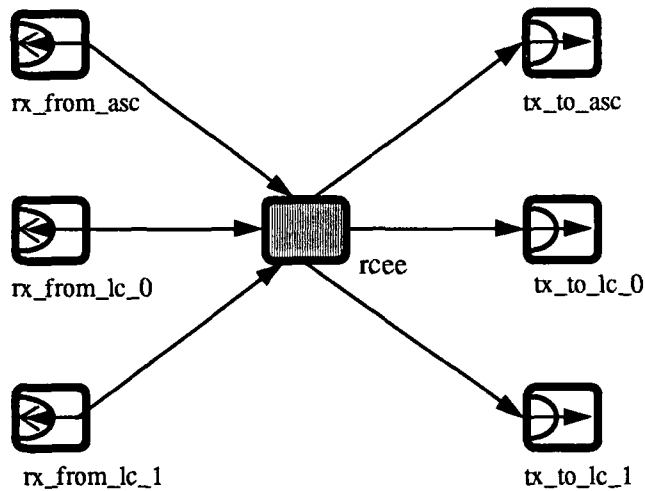


fig 3 8 - Intelligent Peripheral Node Model

3.4.4 User Node Model

The node model for the user node consists of a process module and transmitter and receiver modules. The process module is responsible for initiating or answering a call, and exchanging information with the Switching System. One transmitter-receiver pair forms the signalling path connected to the Process Controller, while the other pair forms a data path connected to the Switching Fabric within the Switching System. The User node model is illustrated in fig 3 9.

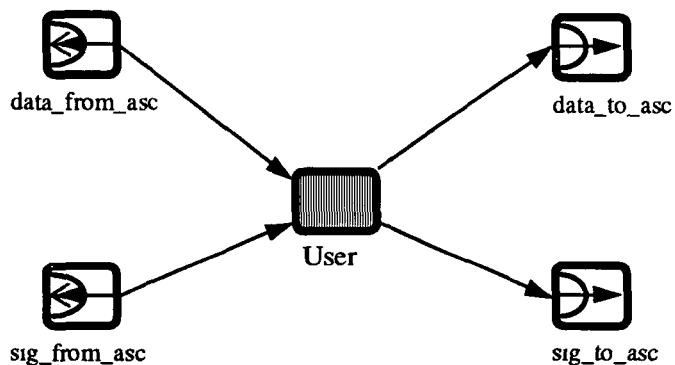


fig 3 9 - User Node Model

3.5 The Process Model

In this section the detailed working and functionality of all process models that have been developed, is illustrated. The finite state machine (fsm) representation of each process model has been depicted graphically.

3.5.1 Switching System Process Models

The Switching System comprises of the following process models

- Process Controller
- Switching Fabric
- Originating Call State Model
- Terminating Call State Model

There are multiple instances of the Originating and Terminating Call State Models

3.5.1.1 Process Controller

The process controller forms the heart of the switching system It performs the following functions

- Maintains persistent and dynamic information related with users and calls
- Invokes the call state models to originate/terminate a call
- Allows the SLEE to activate and deactivate triggers.
- Provides a resource monitoring service to the SLEE
- Reports the detection of triggers to the SLEE
- Performs actions requested by the SLEE
- Enters a dialogue with the RCEE to initiate user-interaction with the system
- Interacts with the switching fabric to manage connections

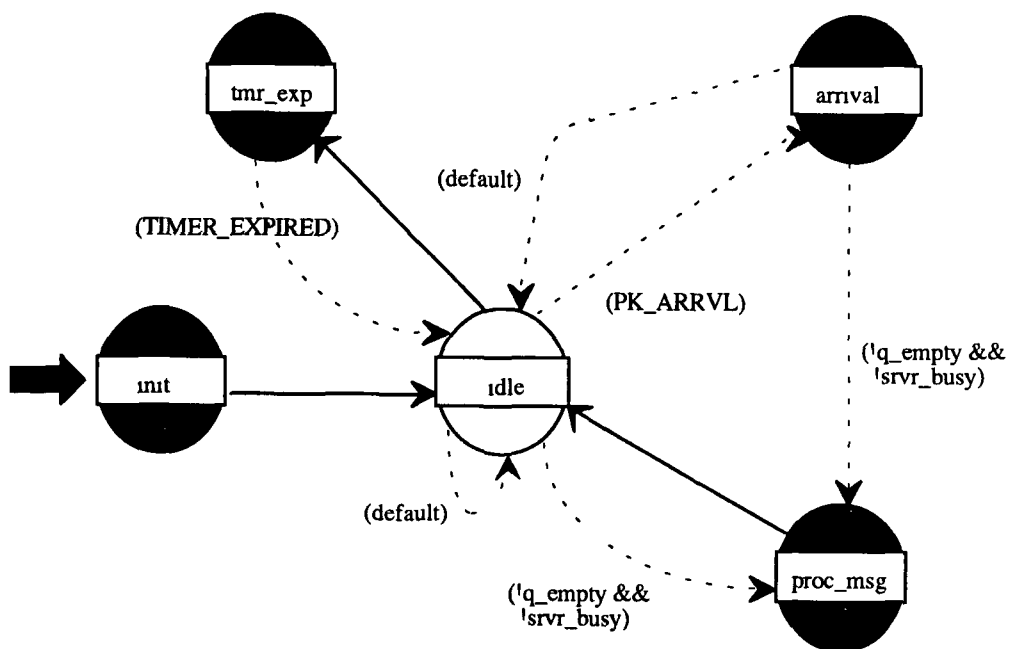


fig 3 10 - Process Controller Queue Model

Finite State Machine Description

The Finite State Machine (FSM) of the queue model is shown in fig 3.10. The process is initiated at the INIT state, where data structures for maintaining call, line and call state model related information are allocated and initialised. The process then unconditionally transitions to the IDLE state. Upon receiving any information on any of its packet streams, the PK_ARRVL macro evaluates true, resulting in the process transitioning to the ARRIVAL state. In the ARRIVAL state, it determines the line on which the packet arrived, procures it and inserts it in the queue. If the process is free to service the queue and if the queue is not empty, it transitions to the PROC_MSG state. Otherwise it goes back to the IDLE state and waits there till the process becomes free or receives another packet. When the process becomes free, and the queue has packets inserted in it, which need processing, a transition to the PROC_MSG state takes place. In the PROC_MSG state, it removes a packet from the head of the queue, determines the packet type, and takes relevant action. It also sets the busy flag, and schedules a self interrupt for the period equivalent to the service time. After that it unconditionally transitions to the IDLE state. Here if it receives intimation of a self scheduled interrupt, it checks the interrupt code number. If the interrupt indicates a timer expiry condition, it transitions to the TMR_EXP state and performs executives necessary to service this condition. After doing so, it returns to the IDLE state. Otherwise the occurrence of a self-scheduled interrupt in the IDLE state results in the process being made free.

3.5.1.2 Switching Fabric

The data lines of the users and the resources from the RCEE are connected to the Switching Fabric. The fabric provides a connection between users involved in a call by forwarding information sent by one participant of the call to other participants. The fabric is also connected by packet streams to the Process Controller. It executes commands issued by the Process Controller to .

- Make a connection between two parties
- Clear an existing connection
- Temporarily split a party from an existing connection
- Reconnect a split party
- Move a party between two calls by splitting it from an active connection and reconnecting it to a split connection

Finite State Machine Description

The Finite State Machine (FSM) of the Switching Fabric queue model is shown in fig 3.11. The process is initiated at the INIT state, where data structures for maintaining line and call related information are allocated and initialised. The process then unconditionally transitions to the IDLE state. Upon receiving any information on any of

it's packet streams, it transitions to the ARRIVAL state. In the ARRIVAL state, it determines on which line a packet arrived. If the packet arrived on a stream representing a data line, it looks up the call related information to determine the destination stream and the status of the call. If the connection has been split, it does not forward information on the destination stream, instead destroys it. Otherwise it forwards it on the destination stream(s). If the packet arrives on the stream connecting it to the Process Controller, it procures it and inserts it in the queue. If the process is free to service the queue and the queue is not empty, in that case it transitions to the PROC_MSG state. Otherwise it goes back to the IDLE state and waits there till the process becomes free or it receives another packet. In the PROC_MSG state, it removes a packet from the head of the queue and processes it. It also sets the busy flag, and schedules a self interrupt for the period equivalent to the service time. After that it unconditionally transitions to the IDLE state. Here if it receives intimation of a self scheduled interrupt, it make the process free by resetting the busy flag.

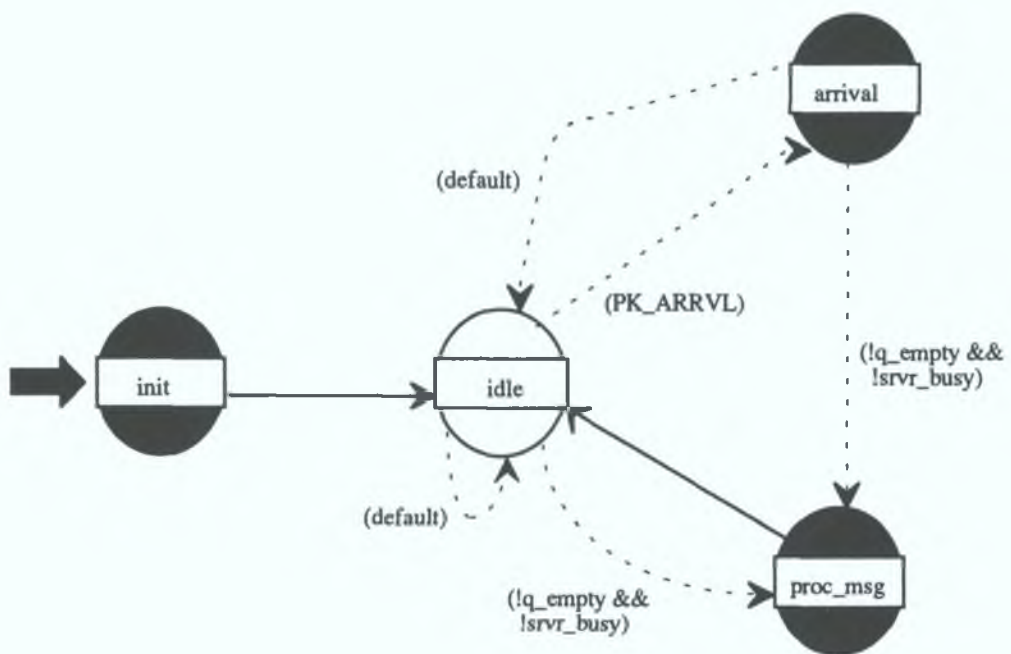


fig. 3.11 - Switching Fabric Queue Model

This model has been used as a process model providing basic queue handling functionality, for many Node Process Models. Although the FSM remains the same, the executives are written separately for each protocol being modelled. Therefore, each one of them performs different actions in the same state. The Resource Controller within the IP, and the Service Manager and ACD SLP within the Service Node have employed this FSM for their process models.

3.5.1.3 Call State Models

The Call State Model Processes (CSMPs) have been based entirely on AIN Basic Call State Models (BCSMs). Each state in the FSM represents either a Point in Call (PIC) or a Trigger Check Point (TCP). Every time a new call has to be setup an instance of the Originating CSMP is invoked. When a call has to be terminated on the called party, the Process Controller invokes a Terminating CSMP. The first six PICs in the Originating CSMP and the first four PICs in the Terminating CSMP represent the setup stages of a call. A stable call is one that has progressed beyond the setup stage. Each instance of the CSMP is connected to the Process Controller by a pair of packet streams providing a duplex data path between them.

Within the process, after performing the actions related with each PIC state, it transitions to the subsequent TCP state. There it first checks if that trigger is activated or not. If the trigger is deactivated it transitions to the next state, otherwise it suspends processing, builds a message indicating the detection of the trigger, adds relevant information in it, and then sends it to the Process Controller. It resumes processing only on receiving a response from the Process Controller. It then transitions to the next state or the state specified by the Process Controller.

A vast range of transitions from a TCP to PICs are allowed. However only a few transitions necessary to support the services that have been deployed, have been implemented. The model has been designed to allow for any future addition in transitions.

Finite State Machine Description

The Finite State Machines for Originating CSMP (OCSMP) and Terminating CSMP (TCSMP) are shown in fig 3.12 and fig 3.13 respectively. When the system is started, the process enters INIT state, where it allocates and initialises data structures and variables required for maintaining information for this process. It then unconditionally transitions to the NULL state. The process waits for the arrival of any instruction or message from the Process Controller. The Process Controller can start an OCSMP and request it to transition to a particular state by specifying it in a message. For e.g. it can specify ANALYSING_INFO as the starting state, in that case it provides the Directory Number (DN) of the called party as a parameter of the message. If the Process_Controller sends the OCSMP a DISCONNECT message, the OCSMP, irrespective of its present state, returns to the NULL state and resets all call related information. Within the TCSMP, if a CLEAR message is received, it also immediately returns to the NULL state. In an active call, if the called party clears the call, a timer is started at the CSMPs. If the party goes off-hook again within that time, the timer is

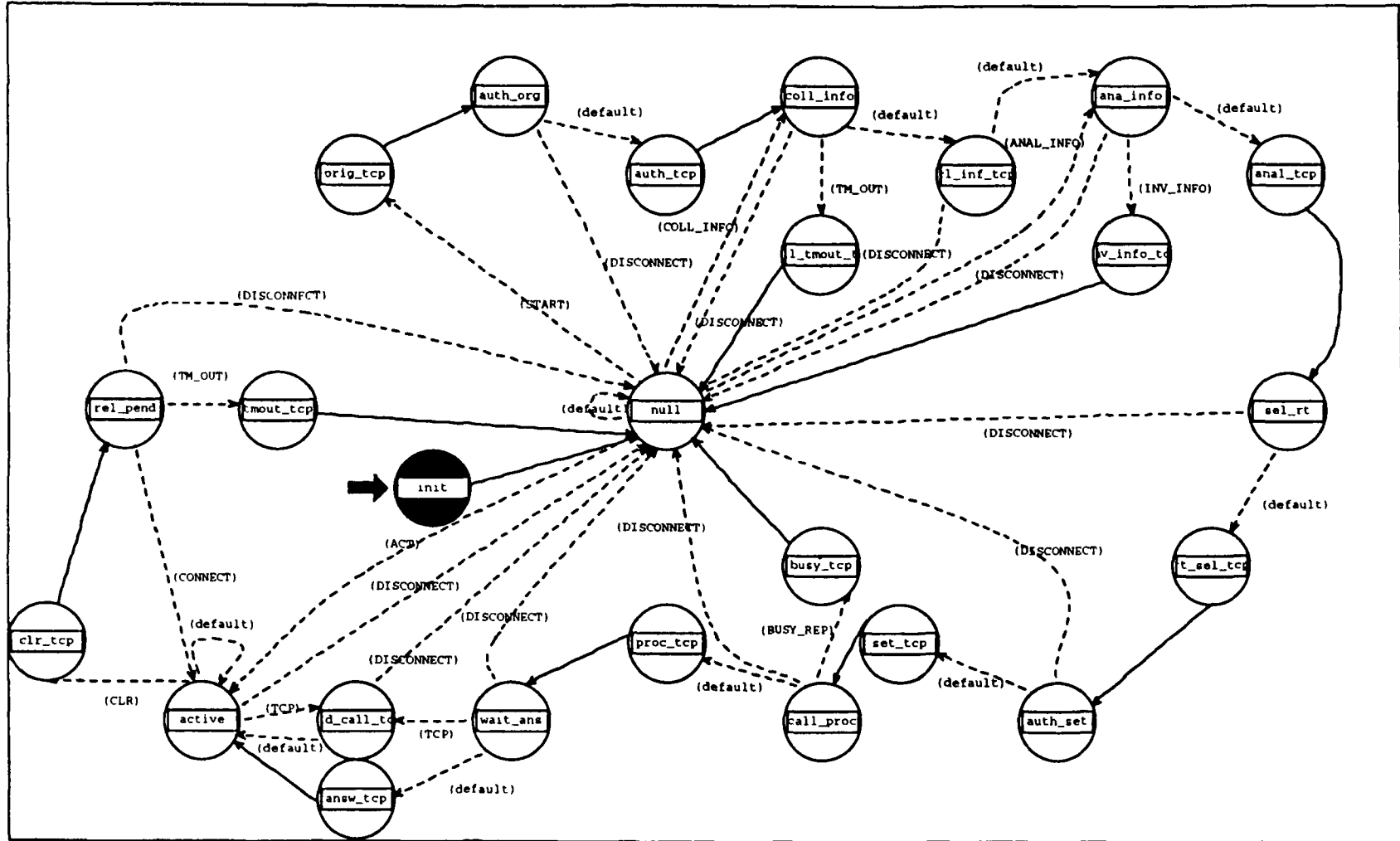


Fig 3.12 Originating Call State Process Model

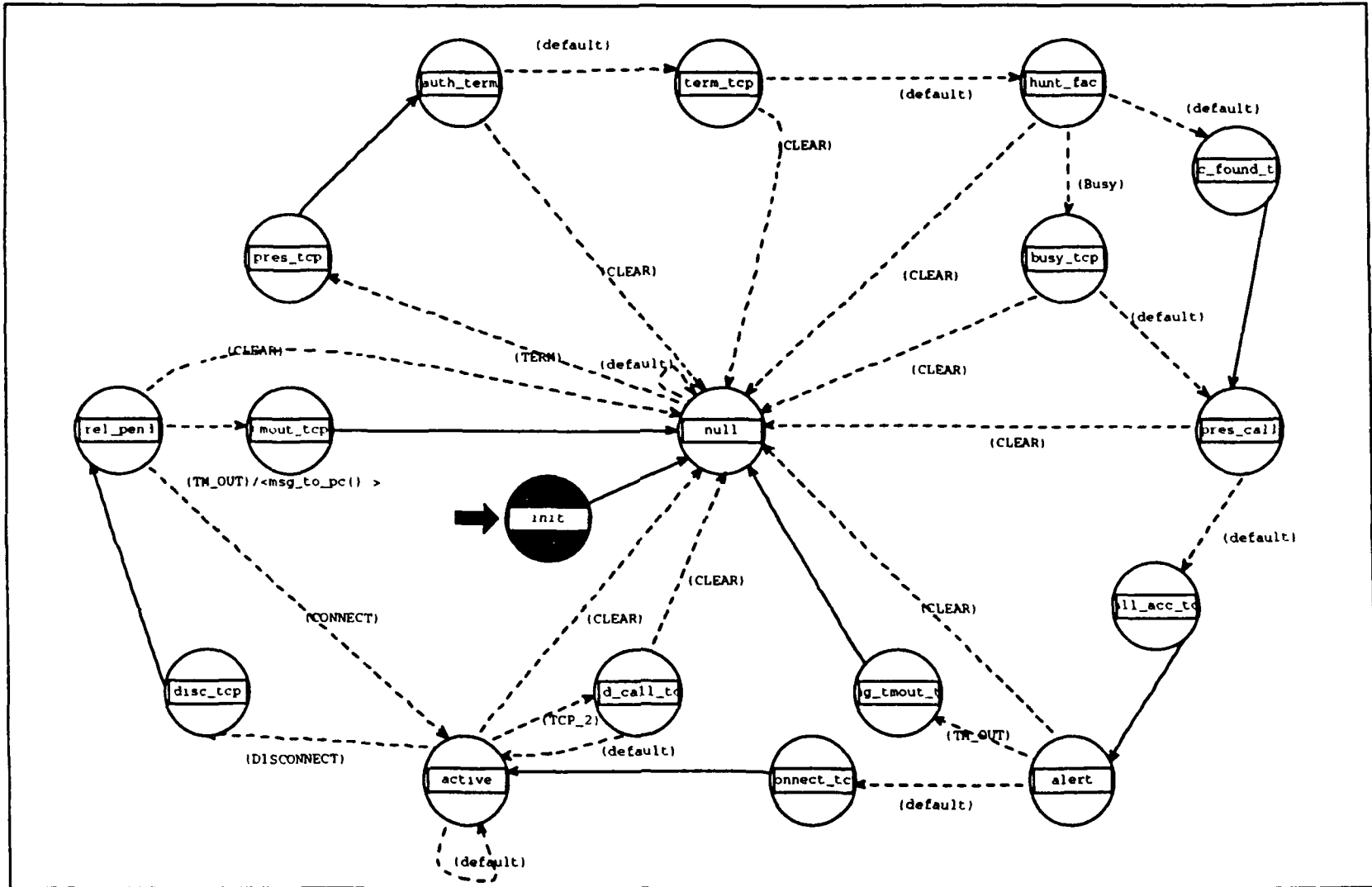


Fig 3 13 Terminating Call State Process Model

stopped and the call is reconnected. Otherwise, if the timer expires, the TCSMP sends a timeout message to the Process Controller, which results in the call being cleared. The detailed working of the model is made clearer by explaining the basic call setup mechanism in section 4.2.

3.5.2 Service Logic Node Process Models

There are two Service Logic node configurations supported. The first configuration supports Call Forwarding, Call Transfer and Call Back services. The process models developed for this configuration are .

- Service Manager
- Call Forwarding Service Logic Program
- Call Transfer Service Logic Program
- Call Back Service Logic Program

The other configuration supports the provision of Automatic Call Distribution (ACD) set of services. The ACD Service Logic Program queue model provides this service.

3.5.2.1 Service Manager

The Service Manager queue model acts as an interface between the SLP processes and the ASC. It is responsible for routing the messages received from the ASC to the appropriate SLP. If there are any feature interactions, it is responsible for managing them. The Service Manager developed only performs routing function, but can be easily upgraded to provide feature interaction management.

Finite State Machine Description

The Finite State Machine for the Service Manager is shown in fig 3.11. The Switching Fabric queue model has been employed here as it provides the basic functionality required for message queue handling and processing. Though the transition conditions are the same, the actions performed within the states are different. When a message arrives at this node, the process transitions to the ARRIVAL state from the IDLE state. If the message is from the feature SLPs, it is forwarded to the Switch. Otherwise it is inserted in the queue. In the PROC_MSG state, the message is removed from the queue, analysed and routed to the appropriate feature SLP.

3.5.2.2 Service Logic Programs

Service Logic Programs (SLPs) are implementations of the AIN services offered to the subscribers. For each feature, there exists an SLP, which is responsible for providing that service to the users. The detailed working of each SLP is explained in sections illustrating that particular service.

The FSM used for all SLPs is the same, but the executives within the states differ for each service. It is shown in fig 3 14

Finite State Machine Description

When the AIN system is started, the process is invoked in the INIT state. It executes commands to initialise the data structures required to maintain information for the feature. It then unconditionally transitions to the IDLE state, and waits for the arrival of feature requests or event reports relevant to the feature. When a message arrives, it transitions to the PROC_MSG state, where it processes the received message and takes necessary action. After completing the processing of the message it jumps back to the IDLE state.

For ACD SLP the FSM of the Switching Fabric queue model is used to provide basic queue functionality. The detection of triggers and occurrence of events which require the attention of the service logic result in messages being sent from the ASC to the SLEE. The messages received from the ASC are processed within the PROC_MSG state, and action taken accordingly. It provides services that include agent logon, agent logoff, agent make busy, agent make unbusy and call distribution.

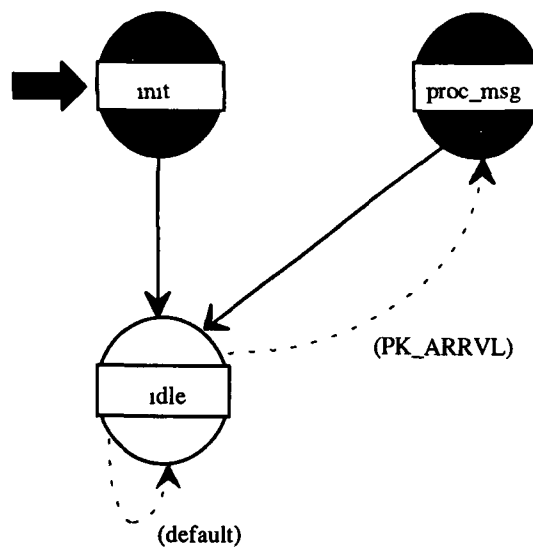


fig 3 14 - Service Logic Program Process Model

3.5.3 Intelligent Peripheral (IP) Node Process Model

Within the Intelligent Peripheral node resides the Resource Controller queue model. It is connected to the Process Controller within the Switching System via a transmitter-receiver pair.

3.5.3.1 Resource Controller

The FSM for the resource controller is similar to the Switching Fabric queue model FSM. The process provides RCEE functionality and controls and manages resources such as customised voice announcements and DTMF digit collection for interaction between users and the network. The ASC routes a call to the RCEE in response to a message from the SLEE. This message contains parameters the ASC passes to the RCEE to perform certain user interaction functions. When the RCEE completes the requested functions, it returns any information collected from the user to the SLEE via the ASC. All processing of received messages is done in the PROC_MSG state of the FSM. How a user-resource interaction is setup and cleared is illustrated in section 3.6.2.

3.6 Communication between Network Nodes

The components of the AIN Network are independent entities, which may be provided by multiple vendors. In order to have compatibility between these components, there is a need to have standard interfaces between them. Therefore Bellcore has proposed mechanisms for communication between these components. The AIN Rel. 1 Network and Operation Plan defines a message set to communicate between the ASC and other network systems. A subset of this message set defined for ASC-SLEE communication, has been implemented to provide a platform for the development of some AIN services. This section lists that subset of messages. A brief description of each one of these messages has been given in Appendix B. Additionally, the communication mechanism between the ASC and RCEE that has been implemented in our model, has also been described here.

3.6.1 ASC-SLEE Interface

The message set used for communication between the ASC and SLEE can be categorised into two major categories:

- ASC-to-SLEE Communication.
- SLEE-to-ASC Communication.

3.6.1.1 ASC-to-SLEE Communication

There are two types of events at the ASC that are of interest to the SLEE and may interrupt normal call processing : triggers and requested events. A *trigger* is the occurrence of an event and the satisfaction of the conditions set for a particular trigger check point. A trigger can be activated or deactivated by the ASC on instruction from the SLEE. A *requested event* is an event specified by the SLEE that influences the call. The SLEE requests the ASC to report the event unconditionally to it on it's occurrence. Every requested event has a parameter specifying whether it is a *report-only* or

response-required event For report-only events the ASC informs the SLEE about the event occurrence and continues normal call processing For response-required events, the switch reports the event occurrence, and suspends call processing awaiting further instructions from the SLEE on how to proceed These messages indicating the detection of requested events or triggers, are used for ASC to SLEE communication

ASC messages are grouped into messages that can be either *initiating or subsequent* messages, and messages that can only be *subsequent* event messages

Initiating or Subsequent Event Messages

Busy_Detected

Feature_Requested

Info_Collected

Termination_Authorised

Subsequent Event Messages Only

Create_Call_Failure

Create_Call_Success

Entity_Status

Join_Leg_Failure

Join_Leg_Success

Merge_Call_Failure

Merge_Call_Success

Move_Leg_Failure

Move_Leg_Success

Resource_Clear

Split_Leg_Failure

Split_Leg_Success

3.6.1.2 SLEE-to-ASC Communication

Messages sent from the SLEE to the ASC are used for controlling calls requesting services, and for non-call related functions They can be grouped into two categories Connection Segment (CS) related and non-CS related *CS related* messages affect call processing In particular, they affect real-time control of the CSs The CS and BCSM control, and participant interaction describe these messages There are no restrictions precluding the SLEE from requesting the ASC to move a leg of a CS past a TCP or PIC which may not be the successive state within the Basic Call State Model *Non-CS related* messages are used for ASC interactions not involved with a CS These messages include entity status checking and information revision messages These messages may require a

response from the ASC, but never require the ASC to suspend the processing of any events associated with a CS

CS and BCSM control

Continue
Create_Call
Disconnect_S
Forward_Call
Join_Leg
Merge_Call
Move_Leg
Originate_Call
Split_Leg
Transfer_Call

Participant Interaction

Send_to_Resource
Cancel_Resource

Entity Status Checking

Continuous_Monitor
Monitor_for_Change
Cancel_Monitor

Information Revision Requests

Update_Request

3.6.2 ASC - RCEE Interface

The SLEE may request that a user be connected to a resource that resides at the triggering AIN Switching System, an IP/Service Node(SN) directly connected to the AIN Switching System, or an IP/SN connected to another AIN Switching System in the network. In the developed model, it has been assumed that all resources reside in an IP directly connected to the AIN Switching System. This section describes how the ASC interprets and acts on a SLEE request to establish a user-resource interaction.

Fig 3.15 illustrates an example of the message flows that take place between an ASC and an RCEE in an IP that is directly connected to the Switching System. The SLEE sends a *Send_to_Resource* message to the ASC to request a resource. The ASC interworks the resource type and the variable length parameter block of the

Send_to_Resource message into the parameter block of a *Setup_Res* message and sends it to the RCEE. If the resources are available to handle the call, the RCEE sends a *Make_Conxn_to_Res* message to the ASC, requesting it make a connection between the specified data line of the IP and the user. The ASC tries to establish the requested connection, and if it is successful, it sends a *Make_Conxn_to_Res_Success* message to the IP. Else it indicates failure by sending a *Make_Conxn_to_Res_Failure* message to the IP. If a message indicating success is received at the RCEE, it begins interactions with the user. When the user interactions are completed, the RCEE sends a *Disconnect_Res* message to the ASC, which includes any information that has been collected from the user, in the parameter block of the message. Otherwise if the RCEE receives a *Make_Conxn_to_Res_Failure* message from the ASC, it sends a *Disconnect_Res* message to the ASC with a parameter specifying the failure cause. The ASC then passes the collected information, completion status and indication of error, if it occurred, to the SLEE in a *Resource_Clear* message.

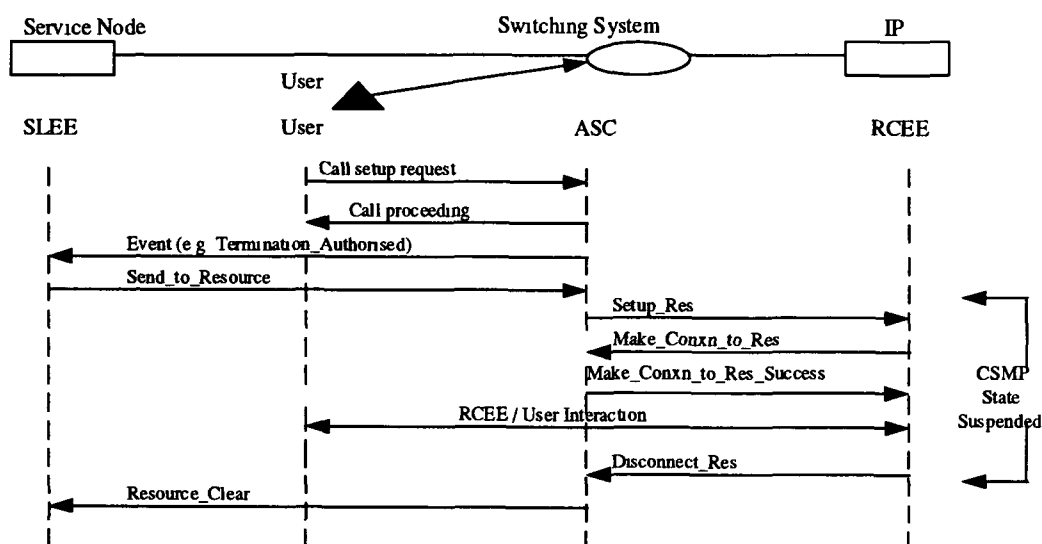


fig 3.15 - Message flow diagram - ASC interworking for accessing resources in an IP

This chapter has given a brief overview of the development tool, OPNET, used for simulating the AIN Network, and a detailed description of the model developed for that purpose. In the next chapter a description of how the AIN basic and supplementary services that have been designed and implemented on this model, has been given.

Chapter 4

AIN Services Realisation

4.1 Introduction

Setting up a basic call in an AIN system requires use of various resources within the Switching System. The actions of each of these resources have to be co-ordinated by a central controller, to manage the call. The mechanism deployed in our system, for handling a basic call has been described in section 4.2. A few services including Call Forwarding, Call Back, Call Transfer, and Automatic Call Distribution (ACD) have been realised on the system. A detailed description of how these services have been realised has been given in the subsequent sections.

4.2 Basic Call Handling

In this section, the internal working of the Switching System has been explained, taking Basic Call set up and handling as an example. Figure 4.1 illustrates the message flow between the Process Controller, Switching Fabric and Originating and Terminating Call State Modules within the Switching System, and the calling party A and called party B, for the setup and termination of a basic call. A "basic call" can be best described as one in which no AIN feature is requested by the users. As a result, no interaction takes place between the ASC and SLEE or ASC and RCEE.

Assumptions

- All users have authorisation to originate calls or can have calls terminated on them.
- All users subscribe to all the features supported by the system.
- No exception conditions occur, i.e. all resources required for call setup are available, and the terminating user is free.

When User A wishes to make a call, he goes Off-hook. An *Init_Call* message is sent to the ASC. The Process Controller (PC) within the ASC receives the message. It looks for a free Originating Call State Model process (OCSMP). If it is available, it sends an *Init_Call* message with the line-id of the user as a parameter in the message, to the OCSMP.

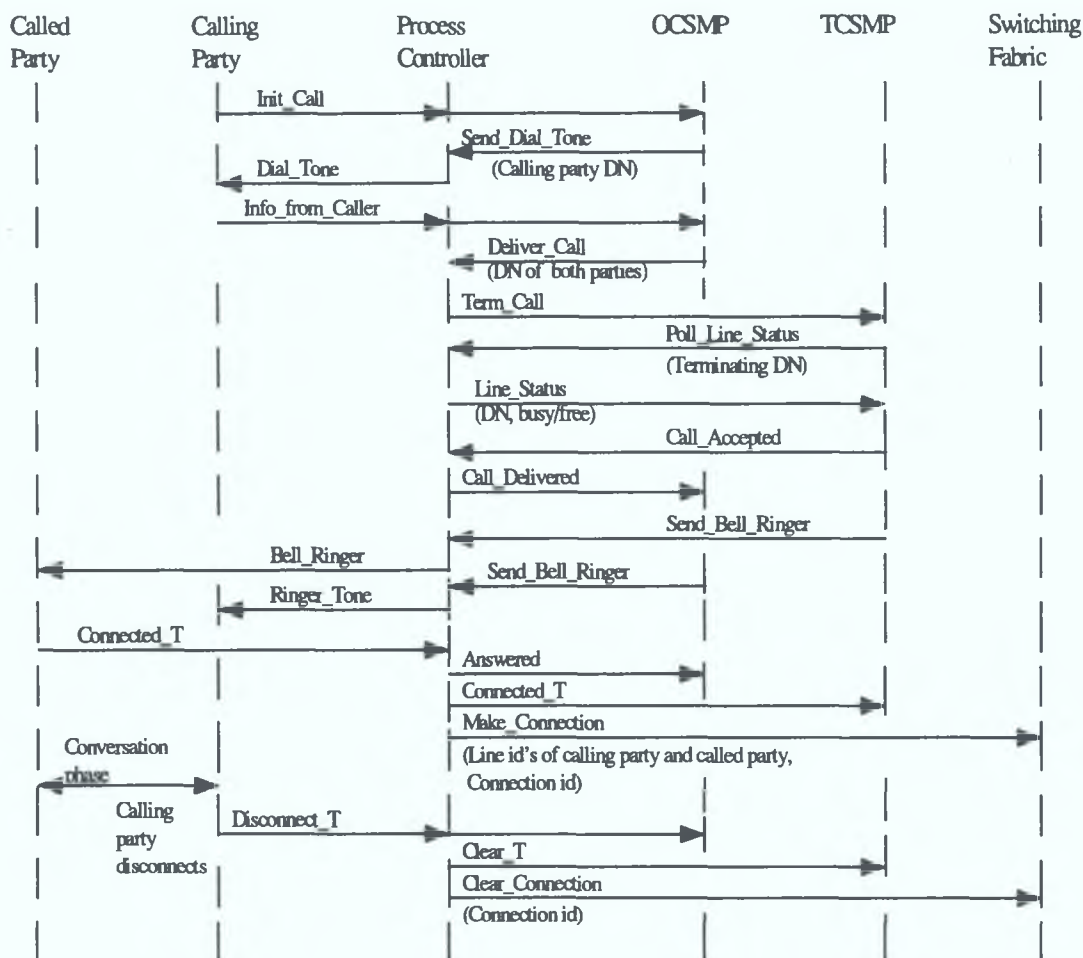


fig. 4.1 - Message flow diagram - Basic Call Handling

At the OCSMP the user information is looked up to determine if the user is authorised to initiate a call. As it has been assumed that all users can originate calls, the OCSMP sends a *Send_Dial_Tone* message to the PC, requesting the PC to provide dial tone to the calling party A. The PC, in turn, sends a *Dial_Tone* message to the user A. On receiving a *Dial_Tone* message, the user A keys in the DN of user B he wants to call. The information is returned to the PC in an *Info_from Caller* message. The PC forwards the message to the OCSMP it had invoked earlier for this call. The OCSMP processes the information. If the digits returned in the message are valid, it sends a *Deliver_Call* message to the PC, with information (i.e. digits) on the called user, to request for call termination. The PC looks for a free Terminating CSMP (TCSMP), and if it available, sends it a *Term_Call* message. Before the call can be terminated on user B, the TCSMP needs to know the status of the terminating user B. It requests the status in a *Poll_Line_Status* message to the PC. The PC returns the status of user B to the TCSMP in a *Line_Status* message. If the TCSMP gets a message indicating that user B is free, it

then sends a *Call_Accepted* message to the PC. The PC translates this message into a *Call_Delivered* message for the OCSMP. The CSMPs request the PC to send ringing and ringer tones to the originating and terminating party, respectively, in a *Send_Bell_Ringer* message. The PC completes the requests by sending a *Ringer_Tone* message to user A and *Bell_Ringer* message to user B. When user B goes off-hook, a *Connected_t* message is sent to the PC from the user B process. The PC sends an *Answered* message to the OCSMP, and a *Connected_t* message to the TCSMP. It also sends a request to the Switching Fabric to establish a connection between user A and B, in a *Make_Connection* message. At this stage, the call goes into the Active state, and the information sent by one user is forwarded to the other user, entering the conversation phase.

The active call can be disconnected by either party. The case, when originating user A disconnects call, is described first. When user A goes On-hook, the user process sends a *Disconnect_t* message to the PC. The PC then sends a *Disconnect_t* message to the OCSMP, and a *Clear_t* message to the TCSMP. It also instructs the Switching Fabric to clear the connection, with a *Clear_Connection* message. On receiving the *Disconnect_t* message, the OCSMP clears all call related information and returns to the NULL state. The TCSMP does the same on receiving the *Clear_t* message from the PC.

In the other case, when the terminating user B goes On-hook, the user process sends a *Disconnect_t* message to the PC. The PC then sends a *Clear_t* message to the OCSMP and a *Disconnect_t* message to the TCSMP. At both the CSMPs, the process enters the *Release_Pending* state and starts a timer. If the user B goes Off-hook before the timer expires, a *Reconnect* message is sent to the PC from the user process. It results in a *Reconnect* message being sent to both the CSMPs. At the CSMPs, the process transitions to the Active state, after cancelling the timer. Otherwise, when the timer expires, the TCSMP sends a *Rel_Pending_Tmout* message to the PC, and transitions to the Null state. The OCSMP also transitions to the Null state, clearing all call related information. The PC then requests the connection to be torn down by sending a *Clear_Connection* request to the Switching Fabric.

4.3 AIN Services

In this section, the modelling technique employed for user-system interaction process has been explained. In addition, a detailed description of each of the AIN services implemented on the system has been given. The services supported by this telecommunication system are

- Call Forwarding
- Call Transfer
- Call Back
- Automatic Call Distribution

User-Network Interaction Mechanism

All user interaction with the network is simulated with the help of a user process written to interact with the system in a pre-determined sequence. For the user to initiate or answer a call, or to invoke any feature, certain information exchanges with the network are required. Therefore a specific user process model is developed for invoking each feature. For example, an Off-hook condition for a user is simulated by sending an Init_Call message from the user process to the ASC. In response to a certain input, E.g. on receiving a Dial_Tone message from the ASC, the user process simulates the action of keying in digits, by sending digits in an Info_from Caller message. To invoke a feature, for e.g. Forward, the user process sends a Forward message with the destination DN as a part of the parameter block. The action of a user pressing the Forward button, entering digits, and pressing Forward again will result in the sending of the Forward message. Therefore the user process simulates the interaction of a user with the network.

4.3.1 Call Forwarding

Description of basic service

The service allows a subscriber of the service to unconditionally forward all incoming calls to a particular destination specified by the user.

Benefits

In case the subscriber is not going to be available to receive calls at his DN, and wants all his calls to be forwarded to a specific DN, he can invoke the Forward Call feature. By doing so, he need not miss any calls that are received in his absence.

The user who dials a forwarded number, can reach the called party without knowing that the call has been terminated at a DN other than the dialled DN. Alternatively, the service can be upgraded to inform the calling party that his call is being forwarded to another destination. As a result of which the user does not have to call back later or make another call at the other number.

Service Invocation Mechanism

The subscriber of the service can invoke this feature by performing the following actions.

- 1 Without going off-hook, press the Forward button
- 2 Key in the destination number, where you want the calls to be forwarded to

3 Press Forward button again

Result. The LED against the Forward button lights up and the service is activated

The subscriber can cancel the invoked Forward Call feature by performing the following action

1 Press the Forward button

Result The LED against the Forward button goes off, and the service is deactivated

Calls to be terminated on this DN are forwarded to the destination DN specified by the subscriber.

Service Logic Description

When a service subscriber wants to forward all incoming calls, the user model sends a *Diversion* message to the ASC, with the destination DN as a parameter of the message. When the PC receives this message it initiates an OCSMP, with *Collecting_Info* as the starting state. Progressing through the OCSMP, it detects that the *Info_Collected* trigger is activated, and the information received represents a feature request. It sends an *Info_Collected* message to the PC, with the information received. The PC creates a connection view of this call and sends an *Info_Collected* message to the SLEE, indicating the detection of a trigger. The Service Manager within the Service Logic node forwards the message to the *Forward_Call* SLP. The SLP updates the user information, making an entry of the destination DN specified by the subscriber. It then sends an *Update_Request* message to the ASC, via the Service Manager, requesting the ASC to light up the LED at the user indicating activation of the forward call service, and to update its trigger related information. The PC sends a *Lamp_On* message to the user to activate the LED. The message flows are illustrated in fig 4.2

When a user makes a call to the forwarded DN, and the PC attempts to terminate the call, the *Termination_Authorised* trigger is detected at the TCSMP. The TCSMP sends a *Termination_Authorised* message to the PC. The PC creates a connection view for the call, and sends the same message with additional parameters, including the id of the trigger that resulted in this message, to the SLEE. The SLP within the SLEE looks up the entry for the called user, and returns the destination DN to the ASC in a *Forward_Call* message. The ASC enters the originating BCSM at the *Analysing_Information* PIC, which is specified as the starting state parameter of the message. The ASC progresses through the OCSMP, and it results in the setting up of a call between the calling party and the user at the destination DN.

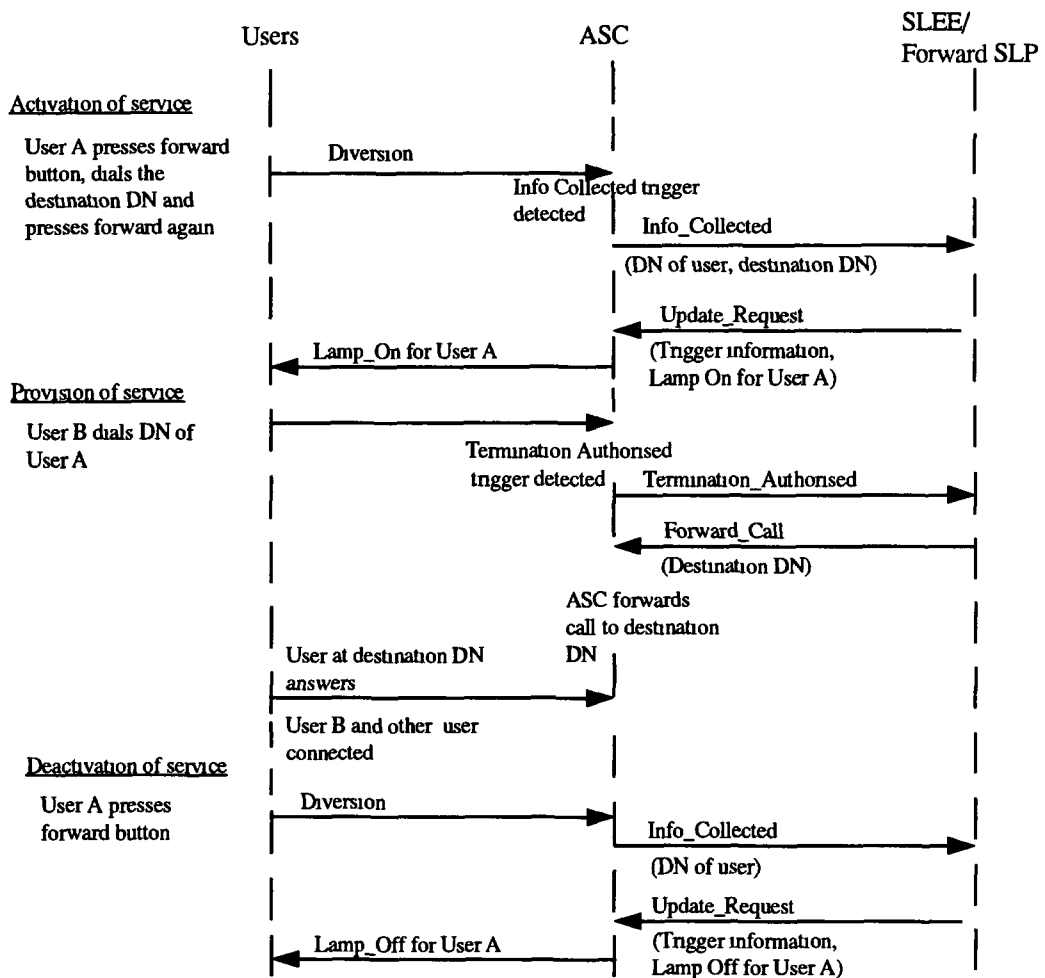


Fig 4 2 - Message Flow Diagram - Forward Call

When the subscriber wants to deactivate the forward call service, the user model sends a *Diversion* message to the ASC. The PC starts an OCSMP, with *Collecting_Info* as the starting state. At the OCSMP, the *Info_Collected* trigger results in an *Info_Collected* message being sent to the SLEE via the PC. When the Forward Call SLP receives the message, it resets the destination DN information, and sends an *Update_Request* message to the ASC. On receiving this message, the ASC updates the trigger information, and sends a *Lamp_Off* message to the user, that results in the LED, indicating activation of the *Forward_Call* feature, being turned off.

4.3.2 Call Transfer

Description of basic service

The service allows a subscriber of the service to make an enquiry call, placing the existing call on hold, and then transfer the original call to the user to whom the enquiry call was made.

Benefits

The service provides the subscriber with a mechanism to forward the call to a third party. It allows him to make a consultation call, and if he wants the two parties to communicate directly, he can transfer the call, thus making himself free to make or receive any other call.

The user who initiates the original call can reach the party he may be interested in being connected to, without having to make another call. For example, in offices, when the reception number is called, the caller is connected to a particular user, by transferring the call to the user's extension.

Service Invocation Mechanism

The subscriber of the service can invoke this feature by performing the following actions:

1. Press the Enquiry button, when involved in another call.
2. On receiving dial-tone, key in the destination number.
3. Press Transfer button to transfer the call.

Result: The subscriber gets disconnected, while the other two parties get connected.

Service Logic Description

Two cases are possible for invoking transfer. One possibility is that the Originating party wants to invoke transfer. The second possibility is that the Terminating party in the first call wants to invoke transfer. Both these cases are discussed below.

When the originating party wishes to make an enquiry call, the user process sends a *Transfer* message to the ASC. The PC, in turn, sends a *Feature_Req* message to the OCSMP. The OCSMP detects a *Mid_Call* trigger, and sends a *Feature_Requested* message to the PC. In case the terminating party wishes to make an enquiry call, the user process sends a *Transfer* message to the ASC. The PC sends a *Feature_Req* message to the TCSMP. The TCSMP detects the *Mid_Call* trigger, and sends a *Feature_Requested* message to the PC. In both cases, the PC creates a connection view of the call, populates the parameters, and sends the *Feature_Requested* message to the SLEE. At the SLEE, the Service Manager forwards the message to the Transfer SLP. At the SLP, the logic interprets the request, and sends an *Originate_Call* request to the ASC, via the Service Manager, to initiate an enquiry call at the requesting user. At this stage, the PC requests the Switching Fabric to split the requesting user from the active call, with a *Split_Line* message. It then starts an OCSMP and progresses through it for setting up the enquiry call.

After the enquiry call has proceeded beyond the setup state, if the requesting user then invokes *Transfer*, the user process again sends a *Transfer* message to the ASC. The

entire process is repeated, and a *Feature_Requested* message is sent to the SLEE. The Transfer SLP detects that transfer has been invoked, and requests the ASC to transfer the call, by sending a *Transfer_Call* message. The PC then requests the Switching Fabric to clear connections for both the calls, and setup a new connection for the final call. The requesting user is therefore, split from both the calls, and a third new call set up between the other two parties. The message flows for handling the service, as shown in fig 4.3, are the same irrespective of whether the originating or terminating user invokes the feature, but the processing within the ASC varies.

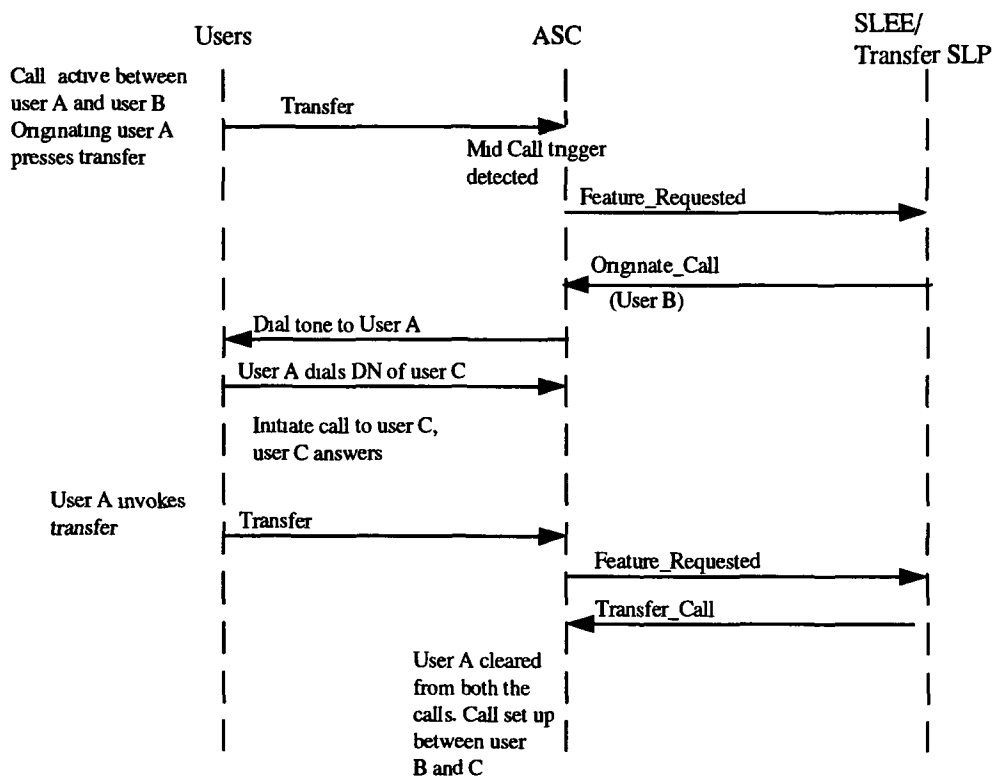


fig. 4.3 - Message Flow Diagram - Call Transfer

4.3.3 Call Back Service

Description of basic service

The network maintains a list of parties who called the subscriber of the service, and requested to be called back, while he was busy with another call. When his line goes idle, the network sends the subscriber a special ringer, requesting the setup of a call back call. If the subscriber responds by going off-hook within a certain span of time, the network attempts to initiate a call to the first user in its list of callers who had requested call back. In the same way it tries to set up calls with all users placed on the call back list of that subscriber.

Benefits

The service prevents the subscriber of the service from losing calls made to him, while he is busy in another call. It also eliminates the need for the subscriber to respond to the calls manually, and therefore the need to know the number of the caller.

If the user finds that the party he wants to get connected to, is busy, he need not try again and again, to try and get a free line. Instead, when the called party becomes free, the network itself tries to set up a call between the two parties.

Service Invocation Mechanism

The user calling up the subscriber has to interact with the system to request call back. The following dialogue is required between the user and the system:

1. If the called subscriber is busy, an announcement stating "Line is Busy, Press 0 for Call Back, 1 otherwise", is sent to the caller.
2. On receiving the announcement, the user has to key in a digit indicating his choice.
3. If he requests call back, an announcement stating "Will call back when line is free", is played.
4. If he does not request call back, an announcement stating "Thank you for calling, please try later on", is played to the caller.

When the subscriber becomes free, an attempt is made by the system, to set up a call between the user and the subscriber.

Service Logic Description

In the process of setting up a basic call, the TCSMP sends a *Poll_Line_Status* message to the PC to determine if the terminating user is busy or free. If the user is busy, the PC returns Busy status in a *Line_Status* message to the TCSMP. On detecting this condition, the Busy trigger is detected, and a *Busy* message is sent to the PC. The PC in turn, creates a connection view of the call, and sends the Busy message to the SLEE. The Service Manager forwards the message to the Call Back SLP. The SLP sends a *Send_to_Resource* request to the ASC, specifying the announcement that has to be played at the caller, and the dialling plan template. The PC sets up a dialogue with the RCEE with a *Setup_Res* message, and requests user-system interaction (the detailed mechanism of how this is achieved, has already been described in Section 3.5.2). The RCEE returns the collected information to the ASC in a *Disconnect_Res* message. The PC sends this information back to the SLEE in a *Resource_Clear* message. If the caller has requested call back, the SLP adds the DN of the caller in the call back list, and requests that a monitor be placed on the subscriber. This is done by sending a *Continuous_Monitor* request to the ASC. The SLP then sends a *Send_to_Resource*

message instructing the ASC to send an announcement to the caller that he will be called back later, and then to disconnect him. Otherwise the SLP sends an announcement to the caller, telling him to try later on, and then requests the ASC to disconnect him.

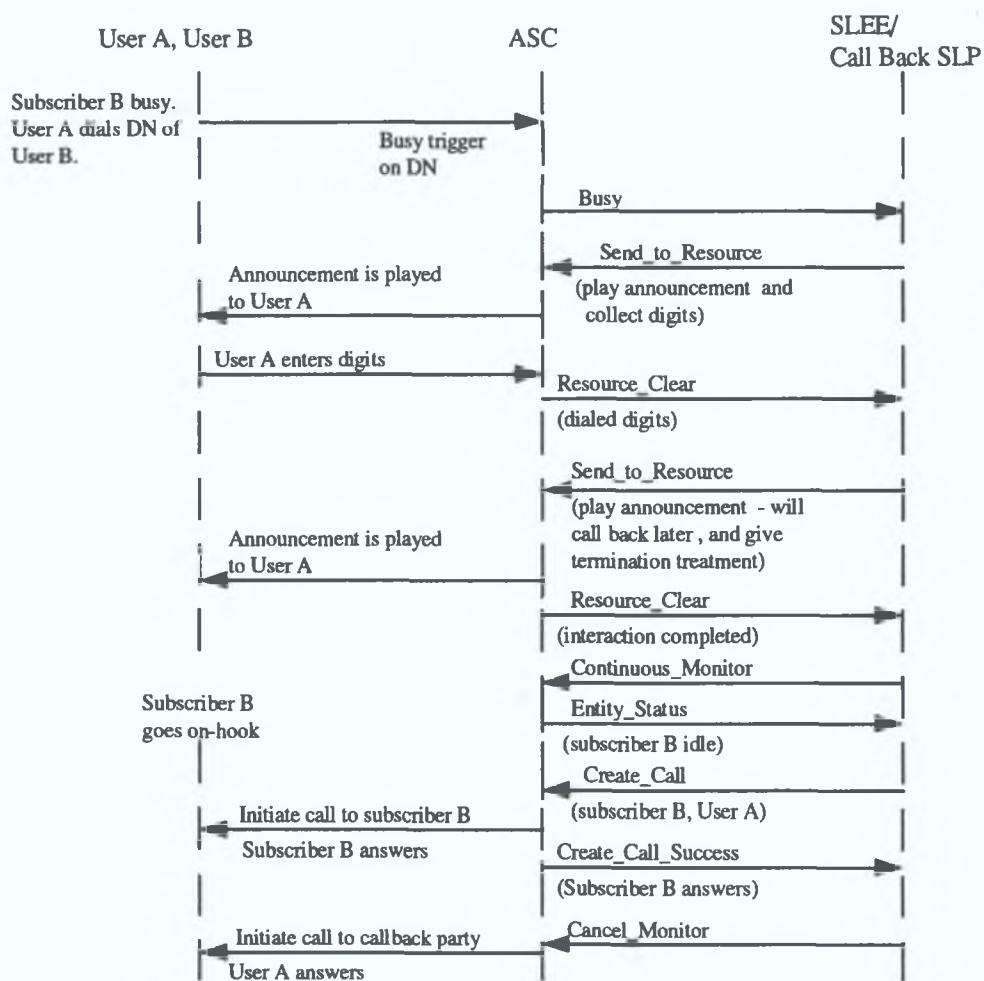


fig. 4.4 - Message Flow Diagram - Call Back

When the subscriber goes on-hook, and the line becomes idle, the ASC returns the status of the subscriber to the SLEE in an *Entity_Status* message. The SLP then sends a *Create_Call* message with the destination DN of the caller and the DN of the subscriber to the ASC. The PC sends a special call back ringer to the subscriber and starts a timer. If the subscriber goes off-hook before the timer expires, the PC cancels the timer, and starts an OCSMP at the *Analysing_Info* state, to initiate call back. In this case the PC sends a *Create_Call_Success* message to the SLEE. Otherwise, if the timer expires, the PC sends a *Create_Call_Failure* message to the SLEE. At the SLP, if a *Create_Call_Success* message is received, that entry from the call back list is deleted. In case of failure, it tries again after some time. If there are more entries in the call back list, it awaits the arrival of an *Entity_Status* message indicating that the subscriber is free,

before requesting the setup of another call back call. If all the entries in the call back list have been serviced, the SLP sends a *Cancel_Monitor* request to the ASC, to stop monitoring the subscriber's line. The message flow diagram for this service is illustrated in figure 4.4.

4.3.4 Automatic Call Distribution (ACD)

Description of basic service

The ACD set of services allow subscribers to automatically distribute incoming calls to non-busy subscriber's agents who answer the call. This feature allows the incoming calls to be distributed among the agents using a pre-determined algorithm. One possible algorithm that has been implemented routes calls to the next available agent. Fairness can be introduced by implementing an algorithm which provides call routing to the agent who is idle the longest.

To provide call distribution, the network must know how many agents are logged on, and what is their status. To support it, the following sub-features have to be supported:

Agent Log On

Agent Log Off

Agent Make Busy

Agent Make Unbusy

Benefits

The service allows the subscriber to have multiple agent terminations on one published DN. This eliminates the need for the subscriber to have a separate DN for each termination available. It also permits the subscriber's agents to make themselves available or busy to allow them to perform their work-related tasks without interruption and to take personal breaks. Additionally, it gives the subscriber flexibility to direct calls to the best answering location. Therefore the subscriber can customise his network, and provide services from different locations in a manner which is transparent to the caller.

For the user, the system is quicker and easier to use. There is no need to go through the operator, and the same number is applicable everywhere. The system connects the user to an agent, who can best service the user's needs, thus providing faster access.

Service Invocation Mechanism

Agent Interaction with network to log on:

1. Agent rings up the published DN.
2. It receives an announcement stating "Please Enter PIN "

3. Agent keys in identification number
4. If PIN entered is correct, the agent receives an announcement stating "You have logged on" Else an "Invalid PIN" announcement is played, and dialogue re-enters step 2

Agent Interaction with network to make busy/unbusy or log off

1. Agent rings up the published DN
2. It receives an announcement stating "Please Enter Service Request Number"
3. Agent keys in service number
4. He receives an announcement stating "Agent made busy", "Agent made unbusy" or "You have logged off" according to the selection made
5. If an invalid service number is entered, an "Invalid service type entered, Please re-enter request" announcement is played, and dialogue re-enters step 2

When a user calls the published ACD number, he is connected to an available agent, else termination treatment is applied. The user is sent an announcement stating "All lines are busy, please try later on", and is then disconnected.

Service Logic Description

For the service to be provided to the subscriber, the SLP must know the number of subscriber's agents available for incoming calls at any given time. This is accomplished by keeping a record of the agents logging on, serving calls, logging off, making busy or making unbusy. The description for call distribution and other agent related services is given below.

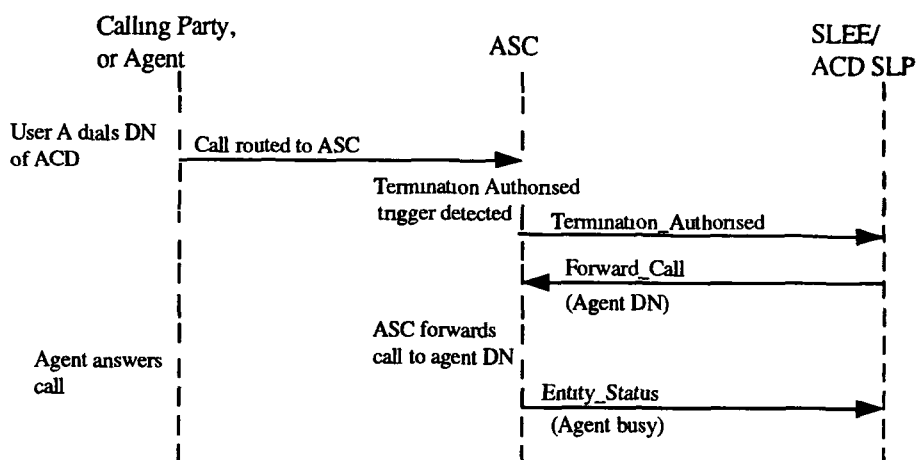


fig 4 5 - Message Flow Diagram - ACD - Call Distribution

When the ASC tries to terminate a call on the ACD subscriber's DN, within the TCSMP the *Termination_Authorised* trigger is detected. The TCSMP then sends a *Termination_Authorised* message to the PC with the terminating user's DN, as a parameter in the message. The PC creates a connection view of the call, and sends a *Termination_Authorised* message to the SLEE. The ACD SLP looks up the subscriber's agent list, and in accordance with a call distribution algorithm, selects the agent to whom the call must be forwarded. The SLP returns the destination DN of the agent in a *Forward_Call* message to the ASC. The ASC starts a new OCSMP with *Analysing_Info* as the starting state, and a call between the calling party and the agent is set up. When the caller and the agent are connected, the PC sends an *Entity_Status* message to the SLP, indicating that the agent has become busy. At the SLP the agent status is marked Busy, and no calls are directed towards this agent till he becomes free again. In case there are no available agents, the SLP sends a *Send_to_Resource* message to the ASC, requesting it to send an announcement stating "All lines are busy, please try later on" to the calling party, and then disconnect the calling party. The message flows for ACD Call Distribution are shown in fig 4 5

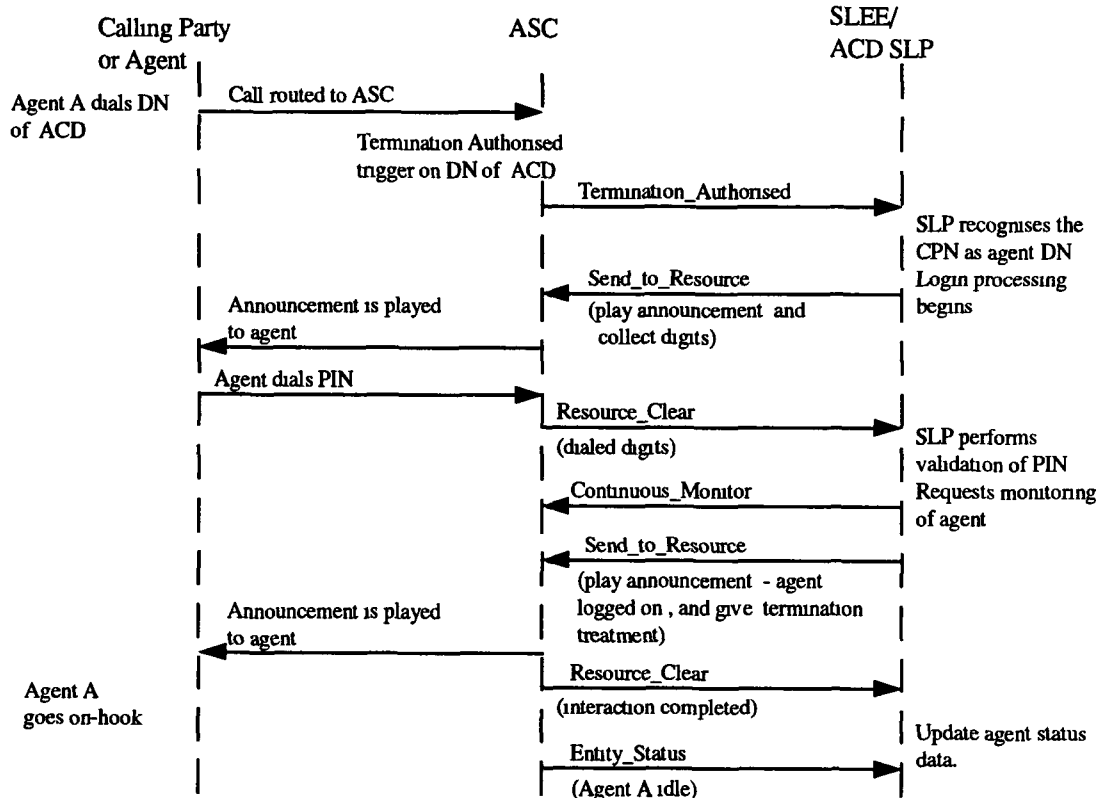


fig 4 6 - Message Flow Diagram - ACD - Agent Log On

When an agent wants to log on, he dials the subscriber's published DN. While trying to terminate the call, the TCSMP detects the *Termination_Authorised* trigger, and sends the

message to the PC. The PC, in turn, sends the *Termination_Authorised* message to the SLEE, with the calling party's number (CPN), as a parameter of the message. If the SLP finds this CPN among its list of subscriber's agents, it sends a *Send_to_Resource* message to the PC, instructing it to send an announcement to the caller to enter PIN to logon, and collect any information returned by the caller. The ASC initiates a dialogue with the RCEE to do so. The information collected by the RCEE, is returned to the ASC, who forwards it to the SLEE in a *Resource_Clear* message. The SLP validates the information returned, and accordingly sends another announcement to the caller, declaring if the agent logged on successfully or not. If the agent log on is successful, he is given termination treatment after playing the announcement. The SLP then requests the ASC to monitor the agent, by sending a *Continuous_Monitor* request. Otherwise the agent is told to re-enter the PIN. The message flows for agent logon are illustrated in fig 4.6

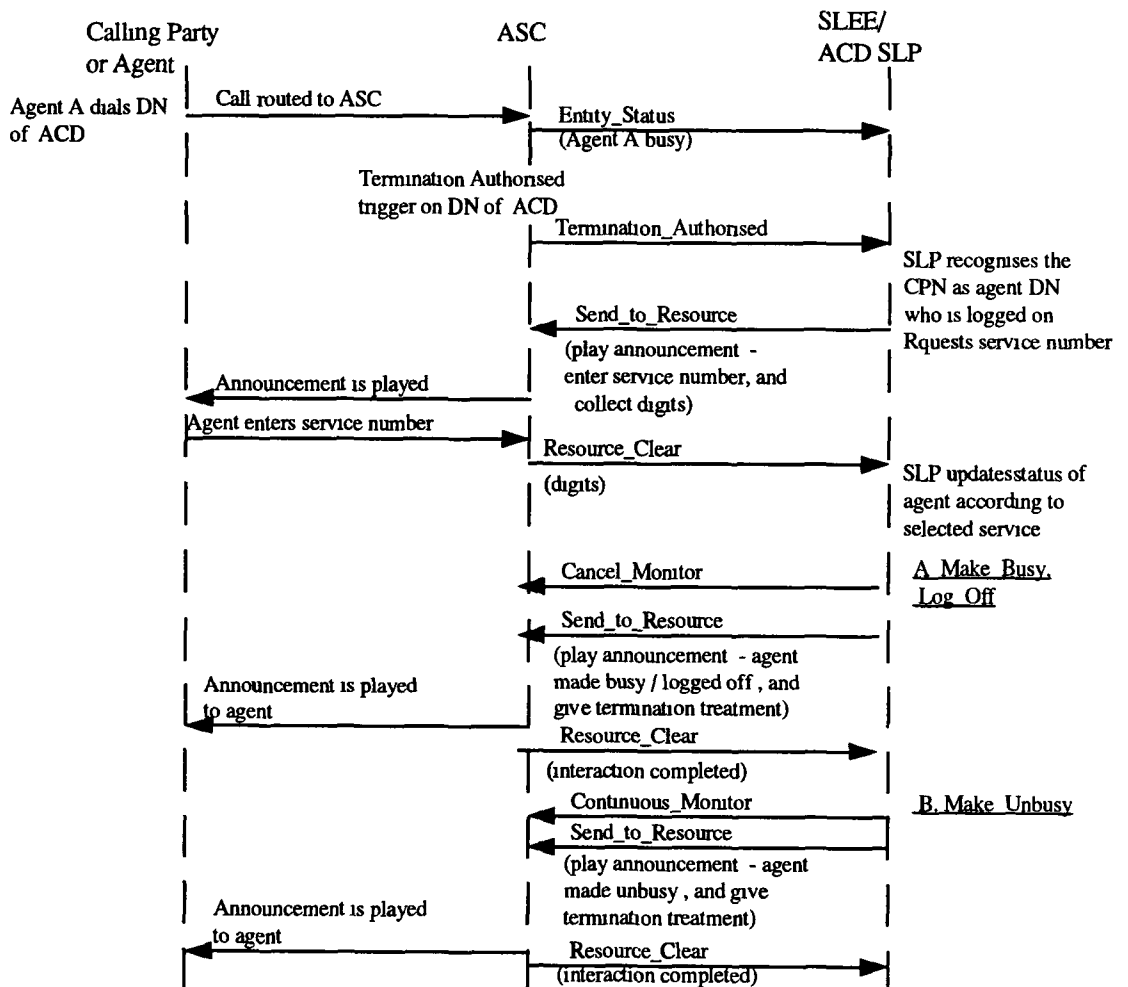


fig 4.7 - Message Flow Diagram - ACD - Make Busy/Log Off

In case the agent wants to make himself busy, he dials the published DN again. Processing similar to the case mentioned above, is performed and a *Term_Authorised* message is sent to the SLEE. The ACD SLP requests the ASC to send an announcement requesting the agent to enter the service number, by sending a *Send_to_Resource* message to it. The data collected by the RCEE is returned to the SLEE, via the ASC, in a *Resource_Clear* message. If the agent enters digits to make himself busy or log off, the SLP sends a *Cancel_Monitor* request to the ASC. If the agent requests himself to be made unbusy, the SLP again requests the ASC for a *Continuous_Monitor* on the agent. In case an invalid service is selected, the SLP sends a message to the agent, indicating that an invalid service type has been requested, and again requests the agent to enter the service number. The message flows for agent logoff, make busy/unbusy are illustrated in fig 4.7

4.4 Results

We could realise all the services described above, with the exception of the Transfer Call service, by using messages defined in the standard set for communicating between the Switching System and the Service Logic Node. In case of the Transfer Call service, no message in the standard set could request the Switching System to perform tasks that would result in its realisation. Therefore, an extra message, "Transfer_Call", was created, and used for implementing this service. The AIN services implemented on the system were then tested extensively to ensure that they worked in accordance with the system specifications. Test plans were drawn up, and conditions created to perform a check for each test case. The responses received for these requests were then compared with the expected results. Establishment of basic calls between users was successfully demonstrated. Further, provision of AIN services to users of the system was also exhibited.

Simulation of this system gave us an opportunity to further enhance our understanding of the intelligent networks. It also allowed us demonstrate the advantages of separating service logic control from the switch in the AIN architecture. Services could be easily created and implemented on the network without making any modifications at the switch.

After successfully simulating this system, the research effort was concentrated on performance analysis of the Service Control Point under overload conditions. The next chapter of the thesis provides a brief background on traffic management in intelligent networks. Subsequently, overload control strategies for the Service Control Point are discussed. Finally a detailed description of the system developed for analysing the SCP's performance, and the results obtained from it, are presented.

Chapter 5

Traffic Management in Intelligent Networks

5.1 Congestion in Telecom Networks

The network requires resources for setting up of a call, maintaining it for the duration of the call, and eventually for clearing the call. A call therefore needs a temporary association of network resources, which can be reused. As long as call traffic is below the engineered capacity of the network, it is in a position to allocate the necessary resources, and can successfully complete calls. But sometimes when there is a sudden surge in demand, or when a network element fails, some resources in the network begin to get overloaded (e.g. a trunk group). Queues of requests for these resources begin to form thus leading to an introduction of delay in call processing. As these delays increase, it eventually results in failure of calls either by time-out or by abandonment by the subscribers.

In these circumstances the call carrying capacity of the network also decreases, causing degradation of service. This situation of congestion is further worsened by the subscriber's behaviour. Due to an increase in delays, subscribers tend to abandon calls, and retry. This reaction is detrimental to the network, as more calls come in to setup phase, as compared to active calls. In contrast with processing for an active call, an additional burden falls on the already overloaded resources causing severe congestion, since processing involved in setting up a call is more resource intensive.

5.2 Network Traffic Management

To maintain an acceptable grade of service in times of network stress, prevent the spread of congestion, and maximise the call carrying capacity of the network, *Network Traffic Management (NTM)* functionality is required [2][8]. NTM functionality allows the network operator to maintain as high a degree of efficiency as possible during traffic overloads and failures.

NTM realises its objectives by *Surveillance* and *Control*. It monitors usage of resources in real-time in a service-independent as well as a service-specific manner. By doing so it can detect the onset of congestion in any element of the network. The collected information may include status reports of network elements, quality of service (QoS) measurements, instantaneous traffic information, and the status of existing controls. It may be collected by other operations entities present on the network, or by itself. For example, in AIN the Network Surveillance operations application collects the required information and delivers it to the NTM operations application. The NTM then analyses the data to detect the evolution and development of overload. If overloading is detected, then it invokes necessary controls to modify the treatment of calls to prevent or reduce congestion.

5.2.1 NTM Controls

NTM controls are of two types - Protective control and Expansive control. A *Protective control* on a call restricts a call from one or more of its normal routing options. This may be used to prevent the spread of congestion by preventing calls from being routed to a congested portion of a network. On the other hand, an *Expansive control* on a call results in the re-routing of the call to other uncongested portions of the network capable of handling the call when the normal routes are busy or have failed. This control strategy makes possible the utilisation of the network's idle capacity.

The choice of the control strategy deployed on the network depends on the configuration of the network as well as the service demands from it. In general, if the overloaded network element is deployed as a stand alone system, protective controls are used. For example, if service logic control is realised in a stand-alone Adjunct, then this control method has to be used. On the other hand, if the switching system used for routing a call is congested, and alternate switching systems are available for routing the call, then expansive control measures may be used. Therefore, for different network elements, different control schemes may be implemented.

The NTM controls used may be either manually activated or automatic[2]. *Manual controls* are activated by network managers either at a central Operations Systems site or at the network element itself. The managers activate these controls in response to the congestion messages they may receive from the surveillance functions in other Operations Applications monitoring the network. Alternately, *Automatic controls* are activated automatically in the network elements on receiving signals from their own built in NTM functions, or from other systems. As no human intervention is required, they respond quickly to changing conditions in the network, thus providing an improvement

over manual controls. The automatic controls can be manually adjusted if the network manager wishes to do so.

For different network elements different controls are implemented. Some of the standard controls used for controlling congestion are given below.

Automatic Congestion Control (ACC) is an automatic control used to manage congestion at the switching system. When the NTM capabilities in the switch detect the onset of congestion, they send signals to the neighbouring switching systems indicating its presence. On receiving this signal, the connected switches reduce the number of calls to the overloaded switch. The blocked calls are completed by routing over alternate paths. When the overloaded switch gets out of congestion, it informs the neighbouring switches which then remove the controls.

Circuit Reservation (CR) is also an automatic control used to reduce trunk congestion. It restricts traffic in a selective manner, allowing direct routed traffic through, and stopping traffic which can be alternately routed. It is activated when the number of free trunks in a trunk group falls below a pre-specified threshold. Conversely it is deactivated when the number of free trunks rises above the threshold value.

Code Controls can be used as automatic as well as manual controls. They may be used to address focused loads where a particular element(s) gets congested. *Percentage Call Blocking* is one code control strategy which blocks a percentage of calls in an arbitrary manner to reduce traffic. In *Call Gapping* an upper limit is fixed for the number of calls that can be forwarded to a specific destination in a given time interval. The latter scheme provides better congestion control. Call gapping is discussed in greater detail in section 5.4.3.

In the following sections issues related to overload and its control in intelligent networks are addressed briefly. It is followed by a more in depth treatment of the SCP overload scenario, and the Automatic Code Gapping control strategy for it.

5.3 Overload in Intelligent Networks

For the provision of different services supported by the IN architecture, varied resource requirements arise. If any service is used in a manner which creates demands for resources that exceed the network's engineered capacity, overloading occurs. For example, on Christmas Day the number of calls greatly exceed the normal traffic, leading to *General Network Overload*. Similarly for televoting type of services, special numbers

are set up to log votes, resulting in mass call-ins. As these calls require IN service logic processing, heavy query traffic is generated between the switches and the SCP/Adjunct which provides the service. It results in increased SCP response times, which may cause call failures by timeout. *SCP Overloading* caused by one service may lead to the deterioration of all other services supported by it. Freephone services offered by companies for their customers lead to congestion at specific terminating switches. This kind of overloading is termed as *Focused Overload*. Services like ACD require user interaction with the network. For each call a digit collection device needs to be reserved. Again if usage levels for this service are abnormally high, it causes *Switching System Overload*. Calls wait in queues for resources. This may lead to call failure either by abandon or by timeout. In a manner similar to Switching System overload or Focused overload, *Trunk Group Overload* may occur too [8].

The standard control schemes suggested in section 5.2 may be used to overcome overloading problems in network systems. We have focused attention on SCP overloading and its control. Consequently, a detailed description of the SCP overload scenario, along with the control strategy suggested by Bellcore has been given in section 5.4.

5.4 Traffic Management of the Service Control Point

When a trigger is detected at the switch, a query is sent to the SCP, and a response is expected within a time-out period. Under normal load conditions the response is received well within the stipulated time period. But as query traffic between the switch and the SCP rises beyond the capacity of the SCP, congestion takes place resulting in an increase in response times. If query response times exceed the time-out value, the switch routes the calls to a reorder tone or an announcement indicating call failure.

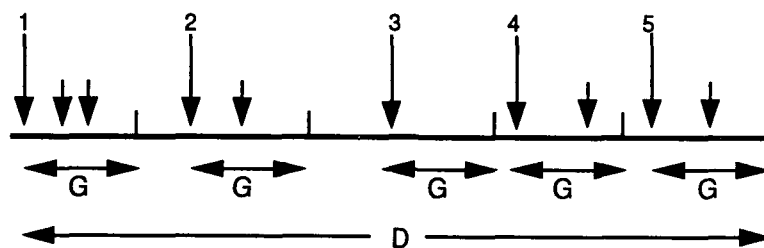
Therefore to minimise the impact of overload on the service quality, there is a need to monitor response times and accordingly control the message traffic between the Switching System and the SCP. The Network Traffic Management (NTM) function in the SCP is responsible for providing this functionality. The main objectives of the NTM function [3] can be outlined as

- i. To ensure that response to queries routed to the SCP are received within tolerable limits, i.e. within the time-out period the switch enforces, and that the waiting times are not much larger than those under periods of normal load.
- ii. To prevent degradation of service quality of other services, as a result of a particular service dominating access to SCP resources.

- iii. To maintain good throughput characteristics under periods of overload by maximising the efficiency in which existing resources are used.

Expansive and Protective control schemes are available for overload control. The *Expansive Control* scheme suggests employing mated SCPs to allow routing of excessive load from the overloaded SCP to the other SCP. But if the SLEE is implemented at an Adjunct or Service Node(SN), this technique cannot be implemented as these systems are not deployed as mated pairs. The *Protective Control* technique suggests that if the NTM function detects overloading, then the SLEE should restrict the switch from sending excessive query traffic to it. As general practise, protective control techniques are used before resorting to expansive controls. Therefore we have put more emphasis on the protective control strategy.

Automatic Code Gapping (ACG), a type of protective control is enforced by the SCP/SMS on the switch to control congestion at the SCP. The ACG control scheme has been illustrated in Fig 5.1. When overload is detected at the SCP, an ACG message is sent to the switch. It specifies the source to be controlled, a *gap interval 'g'*, and a *gap duration 'd'*. Upon receipt of this control message, the switch responds by blocking further queries from this source until gap interval 'g' is exceeded. The next query is then allowed followed by another gap interval of length 'g' during which further attempts are blocked. This pattern continues until the duration timer 'd' expires or until another ACG message for the source is received overriding the existing control[14][16]. In the figure, numbered long arrows refer to accepted queries, and unnumbered short arrows refer to rejected queries. Section 5.4.3 describes the ACG scheme in greater detail.



Legend

- Numbered Long Arrows - Calls Accepted
- Unnumbered Short Arrows - Calls Rejected
- G - Gap Interval
- D - Gap Duration

Fig 5.1 Call Gapping Mechanism

In order to meet the first requirement of NTM objectives, ACG controls are taken in a service independent fashion, to limit the query traffic received at the SCP, and thus ensure that response times are within the time-out values. This control mechanism is called *SCP Overload Control*.

To meet the second requirement, i.e. to prevent one service from dominating access to SCP resources, several strategies can be employed. One way could be to limit all services to a predefined call rate. These predefined call rates determine the mix of traffic that the SCP carries when fully loaded, but is still able to meet the performance requirements. When traffic for a service reaches its maximum threshold value, an ACG control is sent to maintain the service query rate at a consistent level. These call rates may be changed according to a schedule to accommodate different service busy hours. However, situations may occur in which one service is offered traffic above its allowable call rate, while other services receive traffic levels well below their allowed call rates. Thus it may be desirable to dynamically change the call rate threshold, based on factors such as time of day, or on current traffic loads. A service in overload could be allowed extra calls when other competing services are at low traffic levels. As traffic levels increase for these competing services, the SCP sends ACG messages to the ASC to bring the overloaded service back to its predefined call rate, freeing up SCP query handling capacity for other services. Another strategy could be to partition the SCP capacity only when overload occurs.

Because the strategy cannot be fixed when the SLEE is deployed, an Operations System, for example a Service Management System (SMS), is used to provide this functionality. It allows the telecom operators to choose the way services are prioritised and treated in overload conditions. SMS reads performance data provided to it by the SLEE's overload detection functionality, and sends ACG messages to control service traffic levels. This control mechanism, called the *SMS Originated Code Control (SOCC)* provides service selective control.

The functionality required by the elements of the network for providing overload control is described in section 5.4.1; the SCP overload control and SOCC mechanism is detailed in section 5.4.2 and section 5.4.3 respectively; and the ACG Control mechanism is illustrated in section 5.4.4.

5.4.1 Network Functionality

The SCP monitors the overall response times for queries in a service-independent fashion. Response time averages, i.e. the time a query spends in the SCP, from entry to exit, can be calculated over time intervals and used as one measure of SCP performance.

Overload levels may vary from 1 to n. Each level of overload is associated with some action to help improve response time. There are two possible recovery schemes mapped to each overload level. One being the SCP Overload Control scheme, and the other being the SOCC scheme. In the second scheme the SCP sends a message to the SMS as it detects each level of overload.

The Service Management System (SMS) holds the responsibility of selecting code gapping controls when it gets a message from the SCP that indicates that congestion is developing. These controls are then sent to the switch in the form of ACG messages.

The AIN switch provides the functionality that allows the SCP/SMS to place ACG controls on queries that contain a number of switch parameters. AIN switches apply ACG controls based on parameters such as Number Plan Area (NPA) codes, Directory Numbers (DNs), Automatic Number Identification (ANI), subscriber ID, Service Access Codes (SACs) and trigger check point with specific trigger criteria.

5.4.2 SCP Overload Control

The SCP Overload control provides one recovery scheme. In this case, generic requirements specify the appropriate recovery actions taken at each overload level. The recovery action is service independent. When the SCP is overloaded, it sends an ACG request to the switch for every query received during the overload condition. The ACG request contains the first six digits of the originating/terminating number and instructs the switch to cut back future calls having the same first six digits. The ACG request also contains the control *Gap Interval* (Table 5.1) that indicates how severe the cutback should be, the control *Gap Duration* level (Table 5.2), and the reason for call gapping, i.e. the *Control Cause Indicator*.

When the switch receives an ACG request with the Control Cause indicator "SCP Overload", it places the six digit code on an SCP overload control list, and restricts the queries matching the control code. The mechanism employed to do so is detailed in section 5.4.4. The switch can control up to 64 codes on the SCP overload control list simultaneously.

5.4.3 SMS Originated Code Control

SOCC is a flexible control that may be specified as a 3,6,7,8,9 or 10 digit code control, i.e. NPA, SAC, SAC-NXX, NPA-NXX, NPA-NXX-X, etc., used for more selective controls. With the ACG request, the switch receives the *Gap Interval* and *Gap Duration*, and the *Control Cause Indicator* set as "SMS Initiated". It places the code in the SOCC control list and routes all calls blocked by this control to a reorder tone or announcement.

The switch can maintain upto 64 codes on the SOCC control list, in addition to the separate 64 SCP overload controls

5.4.4 ACG Control Mechanism

This section describes the mechanism that applies to both SCP overload and SOCC control. When an ACG control is initiated, a duration timer is set to mark the duration of control. Along with a duration timer a timer for the gap interval is also set. There are 13 possible control duration levels that apply to both the controls. Table 5.2 lists the values of these control duration levels. In case of SCP overload control there are 17 possible gap interval levels, while for SOCC control there are 16 possible levels (Table 5.1)

Gap Interval Level	Average Gap Interval (Seconds)	
	SCP Overload Control	SOCC
0	0	Remove Code
1	3	0.0
2	4	0.1
3	6	0.25
4	8	0.5
5	11	1.0
6	16	2.0
7	22	5.0
8	30	10.0
9	42	15.0
10	58	30.0
11	81	60.0
12	112	120.0
13	156	300.0
14	217	600.0
15	300	infinity
16	Remove Code	-

Table 5.1 Gap Interval Levels

All subsequent calls which match any control code on either lists are blocked by the switch, and no query sent to the SCP, until the gap timer expires. The next call to arrive after the gap timer expires is not blocked, instead is processed normally. The timer is

reset to start another blocking period. This cycle continues until either the switch is requested by the SCP to remove the code item from the control list, or the duration timer expires. To explain further, if gap interval level 3 is selected, then that implies that a gap interval timer of 0.25 sec is started. This would result in restricting the calls to a maximum of 4 calls a second.

On receiving a control removal message from the SCP, the switch removes the appropriate control item from the control list. It is received in the same form as an ACG message with the gap interval set to 16 or 0, for the SCP overload control and SOCC control respectively.

Gap Duration Parameter	Control Duration (Seconds)
1	1
2	2
3	4
4	8
5	16
6	32
7	64
8	128
9	256
10	512
11	1024
12	2048
13	Stop All Calls

Table 5.2 Gap Duration Levels

The functionality required from the network for realising the ACG control scheme and the protocol for requesting placement of controls at the switch has been described in the first part of the chapter. In the next part of the chapter a description of the simulation model has been given.

5.5 Overload Control Simulation Model

In order to analyse the performance of the Service Control Point under service-specific overload conditions a system has been simulated. The system has the functionality to

allow the reproduction of service-specific overload conditions. Further, it has been equipped with the functionality to provide SMS Originated Code Control (SOCC). An algorithm for selection of appropriate controls has also been devised, and implemented at the SMS.

In the original system described in chapter 3, each call is actually set up, maintained for the required duration and eventually cleared down. AIN service requests in these calls are detected at Trigger Check Points (TCPs) between Points in Call (PICs) in the Call State Models. These service requests result in queries being sent over to the SCP, which constitute the load on the SCP. Because the entire call mechanism is negotiated for each call, memory requirements are very high for simulating large traffic volumes. For performance evaluation of the SCP, call set up at the switch is not necessary. Therefore this model is not suitable, instead another model which is functionally much simpler, and modelled to meet our requirements, has been implemented.

The new model generates traffic for the SCP comprising of queries that would be created as a result of invocation of AIN services. The queries are generated for those services that were deployed in the original telecom system, namely Call Forward, Call Back, Call Transfer and ACD. In a simulation run, while load is generated for most of these services in a poisson distribution, random bursty traffic is provided for one or two services. This allows the modeller to analyse the performance of the SCP under variable load conditions.

Performance of the SCP is measured in terms of calls carried by the SCP versus calls offered to it, that require AIN service processing, and response times for queries sent to the SCP from the switch. Response time has been defined in AIN as the interval that begins when the last bit of the query enters the SCP, and ends when the last bit of the response to the query leaves the SCP [3].

In the following sections the realisation of the model using OPNET has been described in detail. Section 5.6 describes the network model, followed by the node model description in section 5.7. The finite state machines for the process models are described in section 5.8. Since the SMS has the responsibility of selecting controls, it has to make decisions regarding allocation of processing capacity of the SCP between services. How it determines the maximum usage values for each service is explained with the help of an example in section 5.9. Further, the algorithm used by the SMS for selecting the controls is described in section 5.10. Finally a description of parameters influencing the selection of controls, and an analysis of the results obtained, is illustrated in section 5.11.

5.6 Network Model

The network is partitioned into two major blocks - the Switch side, and the Service Logic side. Each side is represented as a node of the network. A black box approach has been adopted. The operations within one node are not visible to the other node, and vice-versa. Take the example of the Service Logic side node. Messages received from the switch belong to a standard message set. It need not know how these messages are generated by the switch, or how the response it sends to the messages is used by the switch. To this node the entire network from the switch side, including users connected to the switch, the resource environment, and the switch itself, appears as a single entity. Similarly for the switch, elements connected to the SCP are not visible. Therefore, the switching system, users, and other elements, if any, have been grouped together to form the *Switch side node*. The SCP, and the SMS, have been represented together as the *Service logic side node*.

In real systems, the SCP is connected to the switch via Signal Transfer Points (STPs). The two systems communicate using Signalling System 7 (SS7) TCAP part of the protocol. As there is usually more than one SCP, the STPs route the queries to the appropriate SCP which provides the required service processing. Again, to keep the model size manageable and simple, we have assumed SS7 related processing to be transparent to the switch and SCP. Also, by assuming the deployment of a stand-alone SCP, we have eliminated the need for an STP node. As a result, the Switch side node and the Service Logic side node have been directly connected via a 64 kbps full duplex link. Fig 5.2 illustrates the network model.

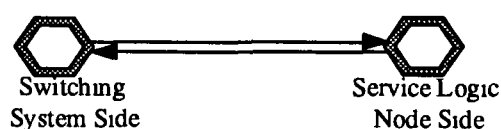


Fig 5.2 Network Model

The ACG control mechanism has been realised by deploying the necessary functionality in the switch, the SCP and the SMS. The SCP is responsible for detecting the onset of congestion and reporting it to the SMS. In response to the alarms sent by the SCP, the SMS sends out ACG messages to the switch via the SCP. In turn the switch is responsible for implementing the controls requested by the SMS/SCP to prevent the enhancement of congestion at the SCP.

5.7 Node Models

For each side of the network there exists a node model. In the following sub-sections the description and working of the Switch side node model and the Service Logic node model has been given.

5.7.1 Switch Side Node Model

The node model as shown in fig 5.3, consists of a queue module for the Switch, process modules for the initiating query generators, and a transmitter and receiver module for linking with the Service Logic node. There are 10 query generators, one for each of the following sub-services - Call Forward Activation, Call Forward Deactivation, Call Forward Servicing, Call Back Servicing, Call Transfer Service, ACD Agent Logon, ACD Agent Logoff, ACD Agent Make Busy, ACD Agent Make Unbusy, and finally, ACD Call Distribution.

The switch is responsible for providing the following functionality:

- Initiating query traffic for SCP
- Creating subsequent queries required for service request completion
- Activation and deactivation of ACG controls requested by SCP/SMS
- Filtering of queries to be sent to the SCP, in accordance with active controls
- Monitoring Offered Traffic and Accepted Traffic in terms of calls and queries

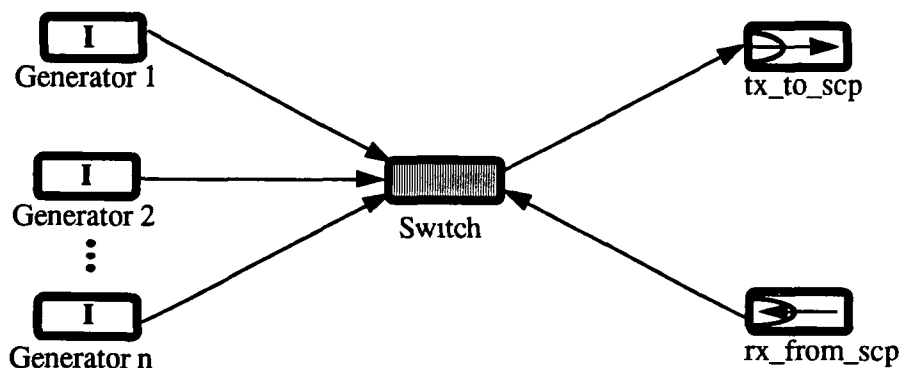


Fig. 5.3 Switch Side Node Model

The query generator process is required to generate initiating queries. Instead of a query being generated by the switch on detection of a trigger while processing a call, the generator creates that query. For example, when a call is to be terminated on a line on which a call forward request has been placed, a *Termination Authorised* message is sent by the switch to the SCP. In the model, this action is simulated by the message generator allocated for the Forward service. If any subsequent queries are to be sent to the SCP,

after the SCP responds to the *Termination Authorised* query, then they are sent by the switch process itself. In this way service specific queries corresponding to new service requests are sourced by the generator processes. The traffic mix is manipulated by controlling the query generation rate for each generator individually.

Two kinds of generators have been used. Since we are interested in generating overload as a result of one or two services at a time, poisson generators are used for services which are not responsible for creating congestion. They generate traffic in an exponential manner, at a mean rate specified separately for each one of them, reaching a steady state. The other generator type - the bursty generator process provides random traffic output. Therefore, if the effect of the control mechanism is to be observed when a particular service goes into overload, this generator is employed. Out of the ten sub-services mentioned above, four of them, namely Call Back service, ACD distribution service, Transfer service and Call Forward service are most likely to be used at rates where congestion might occur. Therefore for one or two of these services a bursty generator is used, while for the others a poisson generator is employed.

The Switch process, in addition to generating subsequent query messages, is also responsible for providing the ACG control functionality. It maintains code control lists, and checks the list to verify if any control is active on that query type, before forwarding it to the SCP. Monitoring load offered by the generators and load that is eventually carried to the SCP, is also the responsibility of this process.

5.7.2 Service Logic Node Model

The Service Logic node model comprises queue modules for the SLEE process, SLEE Transmitter process, and Call Forward, Call Back, Call Transfer, and ACD Service Logic Programs (SLPs). Additionally, there is a process module for the Service Management System (SMS), and a transmitter and receiver module for communicating with the switch side node. These modules, put together, are responsible for providing the following functionality:

- Invoking and executing Service Logic Programs, that provide AIN service processing
- Monitoring query response times, and detecting onset of congestion
- Periodically reporting SCP status to the SMS
- Selecting appropriate ACG controls, to provide acceptable grades of service in overload conditions

The Service Logic Execution Environment (SLEE) process, much like the Service Manager described in chapter 3, acts as a manager for the SCP. Queries received from

the switch are analysed here to determine which SLP(s) should process the request. The query is then passed on to the appropriate SLP. The response sent by the SLP may result in the SLEE sending a message to the switch, or another SLP, if required. If a message needs to be sent to the switch, it forwards it to the SLEE Transmitter process.

The SLEE Transmitter process enqueues the responses, and sends them whenever the transmitter is free. It is therefore in a position to determine the actual time when the last bit of the response leaves the Service Logic Node model and so is given the responsibility to monitor the response times. It reports the usage information on a per service basis along with the current status in terms of overload to the SMS periodically. If service independent controls are employed, then it selects the controls to be placed in case congestion occurs.

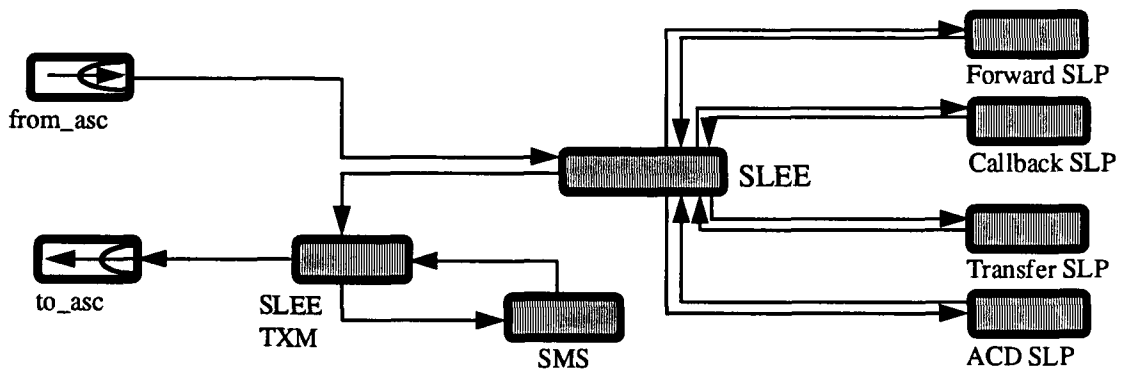


Fig 5.4 Service Logic Node Model

The SLPs for all services basically provide the response to the queries they receive. Their functionality is similar to the SLPs described in Chapter 3.

The SMS is responsible for interpreting messages received from the SLEE Transmitter process. Then, taking into account factors such as current traffic mix, pre-defined thresholds for a service, already active controls, and other business decisions, it selects the best suitable ACG control. An ACG message is then sent to the switch requesting it to press this control into action. Fig 5.4 illustrates the Service Logic node model.

5.8 Process Models

The process models have been described under two major headings - Switch side node process models, and Service logic side node process models. The functionality derived

from these process models, and their working has been described in the following sub-sections.

5.8.1 Switch Side Node Process Models

The process models comprising the Switch side node model include

- Poisson Generator
- Bursty Generator
- Switch

5.8.1.1 Poisson Generator Process Model

The generator process model provides a query generation source. To specify the type of query to be generated by this process, "Service", "Service Type", and "Message Type" attributes have been defined. By assigning different sets of values to these attributes different queries can be created. Therefore, to simulate requests for all services multiple instances of the generator are employed and appropriate sets of values assigned to these attributes. For example, when a Call Back service request is detected by the switch, a BUSY message is sent to the SCP. This action can be simulated by using a generator process model and specifying the service attribute as *Callback*, service type as *Service*, and message type as *Busy*.

The best way to simulate load in a realistic manner is to have arrivals distributed in an exponential fashion. Therefore the frequency of query arrivals is controlled using an exponential distribution as the probability distribution function (pdf) for packet generation. The value assigned to the "Calls per Sec" attribute is used to specify the mean value for the exponential pdf.

The size of an SS7 TCAP message may vary between 120 bits and 2232 bits. A "Packet Length" attribute has been defined to allow for specification of query size. In our system this attribute has been assigned 1200 bits as the default value for all queries.

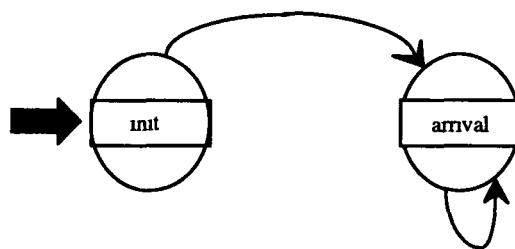


fig 5.5 - Poisson Generator Process Model

Finite State Machine Description

The FSM of the Poisson generator process model is shown in fig 5 5 When the simulation starts, the process is invoked in the INIT state It reads the attributes specified for that instance of the generator An exponential distribution is then loaded, using the value of the "Calls per Sec " attribute The outcome from this distribution is used as the inter arrival time between queries For example, if the "Calls per Sec " attribute is assigned a value of 5 calls/sec , then an exponential distribution is loaded with 1/5 (i e 0.2 sec) as the mean value If an outcome of this distribution is, say 0.3, an interrupt for generation of the next query will be scheduled to occur 0.3 sec from that instant After completing the initialisation, the process then transitions unconditionally to the ARRIVAL state Here a formatted packet is created and the fields of the packet are set in accordance with the service, service type, and message type attribute values This query is then sent to the switch, and another outcome of the loaded distribution taken to schedule the generation of the next query The process waits in the resting stage of the ARRIVAL state till it receives a signal indicating the occurrence of the self scheduled interrupt At this point it re-enters the ARRIVAL state and repeats the actions

5.8.1.2 Bursty Generator Process Model

If a Poisson generator creates queries at a rate above the message handling capacity of the SCP, it will cause a continuous overload condition In that case, the same controls will be implemented by the SMS continuously in an attempt to restrict the query traffic to an acceptable level. But in more realistic systems, overload takes place in bursts and does not last forever. Therefore, in order to evaluate the performance of the ACG mechanism under bursty overload conditions, a bursty generator is needed It is deployed for those services whose traffic levels need to be driven into overload

The bursty generator process is quite similar to the poisson generator process It has all the attributes of the other generator, except that generator frequency is specified in a different way. Instead of specifying it in terms of "Calls per Sec ", a minimum and maximum range is specified Also, the range for burst duration is specified as an attribute of the generator

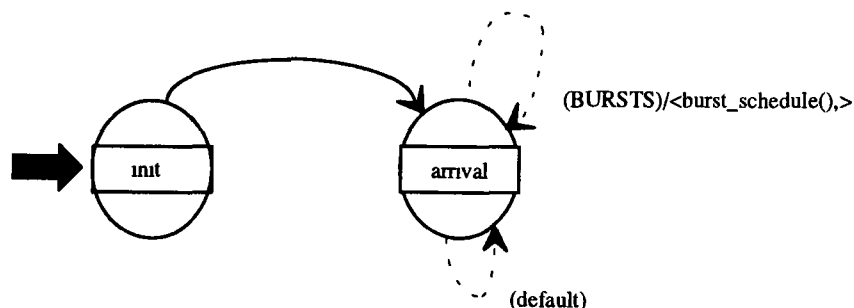


fig 5 6 - Burst Generator Process Model

Finite State Machine Description

The FSM of the bursty generator process model, shown in fig 5 6, is very similar to the poisson one. The process, when invoked, enters the INIT state. The values specified for the attributes are obtained. Accordingly distributions are loaded for obtaining values for burst durations and generation frequencies. The outcome of the burst duration distribution is taken to determine the length of the burst period. Then an outcome of the generation frequency distribution is taken to determine average number of queries to be generated in this burst duration. As in the case of the previous generator process, with the second value an exponential distribution is loaded. This distribution governs the generation of queries in the burst period. The process then transitions to the ARRIVAL state. Here a query is created and the fields of the packet are set in accordance with the service, service type, and message type attribute values. The query is sent to the switch, and an interrupt is scheduled for the generation of the next query. The process then sits idle in this state waiting for the occurrence of any interrupt it had scheduled earlier. When it does receive notification of an event, it checks for event type. If the interrupt indicates a request for generation of a query, the process re-enters the ARRIVAL process, and the above actions are repeated. Otherwise if the event indicates the end of the burst duration, the conditional statement marked "BURSTS" evaluates true. The executives specified in the *burst_schedule* function are performed. The next outcome of the query rate distribution is taken and a new exponential distribution for scheduling message generation is loaded. Also, the next outcome of the burst duration distribution is used to determine the length of the subsequent bursty period. The process then re-enters the ARRIVAL state. In this manner random bursty traffic is created. A sample output from such a generator is depicted in fig. 5 6.

5.8.1.3 Switch Process Model

The Switch process is modelled assuming that it has infinite processing capacity and never gets overloaded. As a result, the switch offers no processing delay for a query. Its main functions include implementing ACG controls requested by the SMS, query handling and traffic monitoring. It is capable of supporting ACG controls requested by the SCP Overload Control (SOC) function as well as the SMS Originated Code Control (SOCC) function.

Finite State Machine Description

Like all fsms, this process when invoked enters the INIT state. It creates the control lists, reads the values of the attributes, and loads the required distributions. It then sits in the IDLE state waiting for the arrival of packets from the generators or the Service Logic side. On receiving a message the process transitions to the ARRIVAL state, and enqueues it for processing. If the processor is free and there are queries waiting to be

processed, the system then proceeds to the PROC_MSG state. It extracts a message from the head of the queue, and sets a busy flag indicating that the server is now busy. After completing message related processing, the process returns to the IDLE state. On receiving an interrupt indicating that server is free, if there are still some messages queued, it re-enters the PROC_MSG state. The messages are processed in this state according to their type and source in the following way.

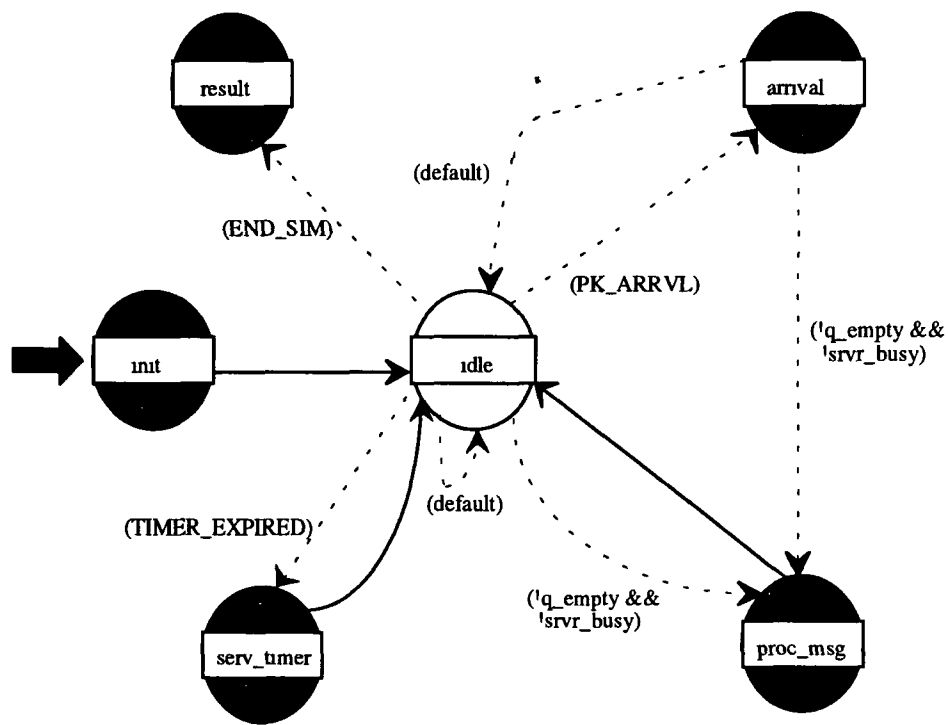


fig 5.7 - Switch Process Model

ACG message It reads the *Control Cause Indicator* value to determine whether the request is issued by the SCP itself or by the SMS. It then checks if the request is to place a new control code, update an existing control code or remove a control code from the list. Accordingly it updates the appropriate control list and takes necessary action described later on in this section.

Query from Generators It checks the SMS control list for any controls active on that particular service type. If found, it then applies termination treatment on the query. Else it scans the SCP Control list for control. If it is found, it gives it termination treatment. Otherwise, if required, it populates the query with the requisite information and sends it to the SCP.

Response from SCP On receiving a response to a query from the SCP, it may be necessary to send subsequent queries to complete the service request. For example, in

case of Call Back, a *Send_to_Resource* is received in response to a *Busy* query sent to the SCP. The *Send_to_Resource* message must be responded to by sending a *Resource_Clear* message. This job is done by the Switch process

One important point to note here is that no controls are exercised on messages that act as responses to messages received by the SCP. This is because the service requests that have been previously accepted by the system must be successfully completed. Therefore if congestion begins to build up at the SCP, control placed at the switch limits only new service requests from being accepted.

The control mechanism in accordance with the ACG scheme explained in section 5.4 is deployed in this way. When a new control is activated, timers for the Gap interval and Gap duration are allocated. The Gap duration and Gap interval level values are obtained from the received ACG message. The time period corresponding to the Gap interval level is obtained from the Table 5.1, and to the Gap duration from Table 5.2. The timers are then started and assigned a special code. When any active timer expires, the process transitions from the IDLE state to the SERV_TIMER state. Here it obtains the code of the expired timer to determine which control it was allocated to. It marks the timer status field as RESET and returns to IDLE state. Now, if the process needs to determine whether the query for whom a control is active, should be sent or given termination treatment, it first looks at the status of the Gap duration timer. If it has expired, it allows the query to be sent along with a tag indicating that though the control code is in the list, it has expired. Otherwise, if it is still running, it checks the status of the Gap interval timer. If this timer is also still running, it gives termination treatment to the query. Otherwise, it restarts the Gap interval timer, and forwards the query to the SCP.

The Switch keeps an account of the load offered by the generators, and the load that is actually carried through to the SCP. It writes these results periodically. To provide this functionality an interrupt is scheduled. On its occurrence, the MONITOR conditional statement leaving the IDLE state evaluates true. The statistics are written to an output vector file, a new interrupt is scheduled, and then the process returns to the IDLE state.

At the end of the simulation, the process transitions to the RESULT state, and writes the final results. Fig. 5.7 illustrates the fsm of this process model.

5.8.2 Service Logic Node Process Models

The process models associated with modules of this node include:

- Service Logic Execution Environment (SLEE)
- SLEE Transmitter

- Service Logic Programs (SLPs)
- Service Management System (SMS)

Before describing these process models, factors influencing the response times of queries have to be identified. From the description of the working of an SCP given in section 5.7.2, it can be inferred that the response time of a query entering the SCP is mostly determined by the processing and queuing delays offered by the SLEE process and the SLPs. In addition, the queuing delay and transmission delay offered by the SLEE Transmitter process also contributes to the eventual response time. While the extent to which processing delays contribute to the response time is fairly consistent, it varies for queuing delays. The latter depends entirely on the load on the SCP. Therefore, in the modelling of Service Logic node processes the processing and queuing delays have been considered.

5.8.2.1 SLEE Process Model

The Service Logic Execution Environment (SLEE) process forms the heart of the SCP. It is responsible for routing queries from the switch to the appropriate SLP, and from the SLP to the switch. To allow the modeller to define query processing times, a "Service Rate" attribute has been defined for the SLEE process. The value of this attribute determines its processing speed in bits per second. For example, if a value of 120,000 bits/sec is assigned to this attribute, and the length of the query received is 1200 bits, then the SLEE would take 0.01 sec, i.e. 10 ms to process this query. Therefore, by changing the "Service Rate" different processing times can be obtained.

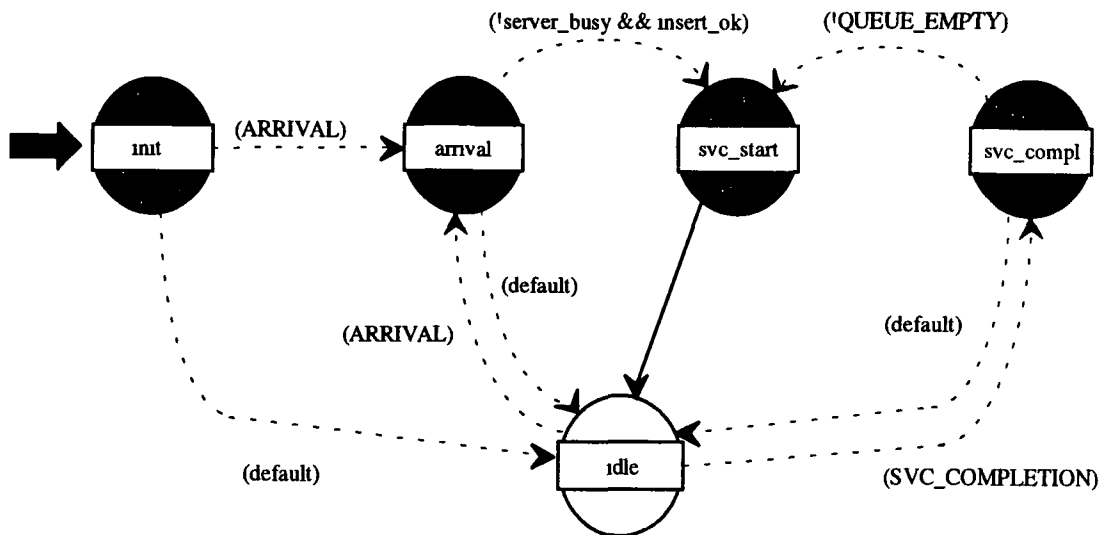


fig 5.8 - SLEE Process Model

Finite State Machine Description

The fsm shown in fig. 5.8, is the representation of a standard first-in first-out single server queue model. The fsm and the protocol related with the provision of queuing functionality has been used by all service logic program (SLP) processes. To this basic model process specific code is added to achieve the desired functionality. The queue functionality is achieved in the following manner

The process initialises the variables by performing the actions specified in the INIT state, reads the value of the "Service Rate" attribute, and transitions to the IDLE state. When a message is received by this process, it transitions to the ARRIVAL state and inserts the message at the tail-end of the queue. If the server processing the messages is free, the process transitions to the SVC_START state, else it jumps back to the IDLE state. In the SVC_START state the message at the head of the queue is accessed to determine its length. The processing time for this query is determined and an interrupt is scheduled to occur after a period equal to the processing time. For this period the server is marked *Busy*, and no other query is processed. The process then returns to the IDLE state. Detection of the scheduled interrupt results in the transition of the process to the SVC_COMPL state. Here the message is removed from the head of the queue, and forwarded on the outgoing stream. After completing this action, it marks the server *Free*, and checks if there are any more entries in the queue waiting to be processed. If there are entries in the queue, it enters SVC_START state again, else returns to the IDLE state.

5.8.2.2 SLEE Transmitter Process Model

The transmitter module provided in the OPNET package is used for linking nodes. The channel capacity for this module can be assigned by the modeller. As a result, when a message is sent to the transmitter, it takes a transmission time equivalent to a period obtained by dividing the message length by the channel capacity. Then, if information is sent to it at a rate higher than its channel capacity, it just queues the information in its buffer. This feature of the transmitter prohibits the modeller from knowing the actual time of transmission of every message. In order to calculate the response times knowledge of transmission time is necessary. Therefore, to get the transmission times the SLEE Transmitter process has been implemented. The SLEE Transmitter enqueues the messages and sends them whenever the transmitter module is free. As a result the process knows exactly when the message is transmitted.

Finite State Machine Description

The FSM for this process is shown in fig 5.9. The working of this queue model is very similar to that of the Switch process. Besides enqueueing the messages, it is given the responsibility to monitor the response times, so that the onset of congestion can be

process model is employed. The value assigned to the "Service Rate" attribute then determines the processing time per message. Therefore, if we specify the "Service Rate" as 40,000 bits/sec, and take message length to be fixed at 1200 bits, then the processing time will be 0.03 sec, i.e. 33 ms.

Each service has a separate SLP process model, as the processing for each service is different. But the FSM, shown in fig. 5.8, has been used for all of them. For FSM description refer to section 5.8.2.1.

5.8.2.4 Service Management System Process Model

This process is responsible for selecting the most effective ACG controls and thus get the best possible SCP performance under overload conditions. For the SMS to make the right selection of controls, information regarding the desired traffic mix, SCP processor capacities, link capacity, etc. has to be provided to it.

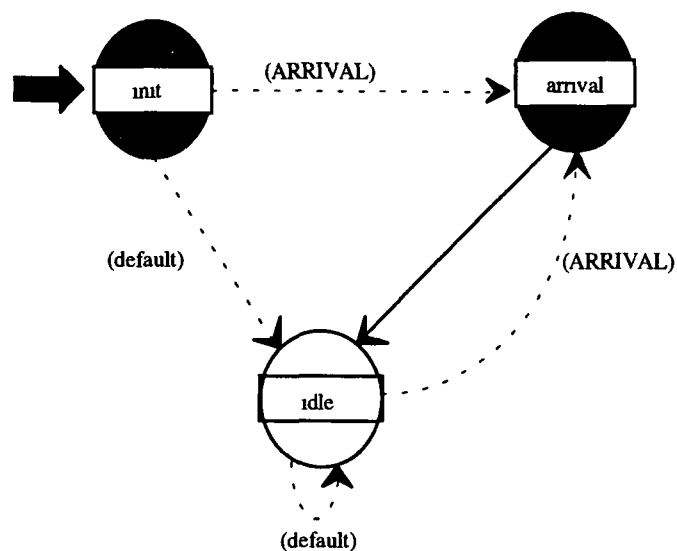


fig 5.10 - SMS Process Model

Finite State Machine Description

The FSM for this process is illustrated in fig. 5.9. The process when initiated, enters the INIT state, reads the persistent information and calculates the usage threshold levels for every service. It also initialises a control list, which is later used to keep a record of the controls it requests the switch to activate. Then it transitions to the IDLE state, where it waits for the arrival of monitor reports from the SLEE Transmitter. On receiving a message from the SLEE Transmitter it transitions to the ARRIVAL state. Here it executes the control selection algorithm to determine the controls to be applied at the switch. It formulates an ACG message, populates it with values for the control digits, gap interval level, gap duration level, and Control cause indicator parameters. It then

sends it to the switch via the SLEE Transmitter. After completing these operations it transitions back to the IDLE state. The method used for the calculation of thresholds is described with the help of an example in section 5.9. It is followed by an illustration of the control selection algorithm in section 5.10.

5.9 Calculation of Thresholds

Thresholds are defined in terms of maximum number of queries that the SCP can handle for each service. Before actually calculating thresholds the difference in the meaning of the terms "Queries" and "Messages" has to be explained. Each message sent from the switch to the SCP that requires a reply may be considered as a "Query". On the other hand, any packet that is processed by the SLEE is a "Message". Therefore message counts are determined from all the messages that are processed by the SLEE. As performance is analysed in terms of response time per query, besides comparing Carried Load against Offered Load, thresholds are defined in terms of queries.

Although response time of a query entering the SCP is determined by many factors, the most crucial among them is the part played by the SLEE. This can be attributed to the fact that every message to and from the SCP passes through it. Therefore for preventing development of congestion in the SCP, the SLEE capacity has to be partitioned on a per service basis. If processing time per message is taken as 10 ms, it implies that at most 100 messages can be processed in a second, provided they arrive at regular intervals. But since message arrival is random and bursty, 100 messages cannot be processed without suffering some queuing delay. To prevent the response times from exceeding the maximum acceptable value, queuing delays must be controlled. To achieve this, thresholds are decided taking a maximum of approximately 75 messages per second as the SLEE capacity.

From the service realisation message flow diagrams shown in chapter 4, it is obvious that all services do not use the same number of messages to provide a service. Therefore it is necessary to pre-define the desired traffic mix before deciding the thresholds. Different traffic mixes will result in different message volumes at the SCP. As a result, for different simulation runs, if the desired traffic mix is changed, then thresholds need to be recalculated. The calculation method is described with the help of the following example.

If the operator, from past experience and usage statistics, decides that at a particular time of the day, the SCP should be partitioned in such a way that if overload occurs, the SCP should restrict usage of each service to a maximum level. A sample desired traffic mix is given below.

<i>Service</i>	<i>Maximum Percentage of Total Service Requests</i>
Call forward	13.33%, (Activation - 3.33%, Deactivation - 3.33%, Servicing - 6.67%)
Call back	40.00%,
Call transfer	13.33%,
ACD	33.34%, (Call Distribution - 26.67%, (Agent Logon, Logoff, Make Busy, Make Unbusy - 1.67% each)

Let us assume that the SLEE gets service requests at the rate of 60 calls per second, and that the calls arrive in the above mentioned mix. The SLEE usage is then worked out in the following manner. Taking the example of Call Transfer service requests - if it constitutes 13.33% of the traffic, then it implies that 8 of the 60 calls are requests for Transfer. From fig. 4.3 (in chapter 4), we can see that each transfer request results in 2 queries, and in this case, 4 messages. Therefore to complete 8 transfer requests in a second, the SLEE has to process 32 messages. Similarly for all services, this calculation has to be made.

While for all services except Call back, the number of queries and messages can be easily accounted for by looking at the message flow diagrams in chapter 4. In Call back, only the detection of Busy status of the called party, and interaction with the user to determine if call back is desired, has been considered. As a result, only the first 6 messages illustrated in fig. 4.4 are simulated and accounted for. The usage details are worked out in the table 5.3 given below:

Service Type	Queries per Call (A)	Messages per Call (B)	Calls out of 60 calls (C)	Total queries (A*C)	Total Messages (B*C)
Forward Activate	1	2	2	2	4
Forward Deactivate	1	2	2	2	4
Forward Servicing	1	2	4	4	8
Transfer	2	4	8	16	32
Call Back	3	6	24	72	144
ACD Agent Logon	2	7	1	2	7
ACD Agent Logoff	2	7	1	2	7
ACD Make Busy	2	7	1	2	7
ACD Make Unbusy	2	7	1	2	7
ACD Distribution	1	3	16	16	48
			Total	60	120
					268

Table 5.3 SLEE Usage Calculations

Therefore, with the specified desired traffic mix the SLEE will process 268 messages (120 queries) to complete 60 service requests. As the maximum capacity of the SLEE is taken as 75 messages a second, then that implies that a maximum of $(75/268)*60$ or 16.8 service requests can be handled per second. Taking other background processing into account the engineered capacity for this traffic mix is taken at 15 service requests a second. This throughput level should be achieved at times of overload.

Taking the ratios of individual service processing requirement to total processing requirements, we can finally arrive at the thresholds. Since Forward Servicing, Call Back, Transfer, and ACD Distribution are of main interest, thresholds for only those services are calculated.

If 268 messages are processed, as a result of 120 queries, then if the maximum message capacity of the SLEE is 75 messages/sec, the maximum query capacity of the SLEE will be $(120 / 268) * 75 = 33.6$ queries/sec.

This maximum query handling capacity has to be allocated between the services taking into consideration their processing requirements. If for Forward service, a maximum of 8 out of the total 268 messages are required to be processed, then the threshold for this service should be $(8 / 268) * 33.6 = 1$ query/sec.

Similarly threshold for each service will be.

Threshold for Forward servicing	$(8 / 268) * 33.6$	= 1.00 query/sec
Threshold for Call back service	$(144 / 268) * 33.6$	= 18.00 queries/sec
Threshold for Transfer service	$(32 / 268) * 33.6$	= 4.00 queries/sec
Threshold for ACD Distribution	$(48 / 268) * 33.6$	= 6.00 queries/sec

The remaining query capacity is utilised for the remainder of the services.

5.10 Control Selection Algorithm

When a monitor report is received by the SMS, it first looks up the control list to see if any control has expired. Each expired message is deleted from the list, and an ACG message is sent to the switch requesting it to do the same. Next, the overload level reported in the message is examined. The overload level could be greater than zero or equal to zero. Sub-section A describes the details of the algorithm for the former case, while sub-section B describes the algorithm for the latter one. The flow chart given in Appendix A.3 illustrates the entire algorithm.

A. Overload level greater than zero

If the overload level is greater than zero, the usage levels of the Call Forwarding Service, Call Back service, Call Transfer service, and ACD Distribution service are compared with their corresponding threshold values. The reason for comparing only four of the 10 sub-services is that since these are the major services, only they could possibly cause overloading. For each one that exceeds the threshold, an ACG message needs to be sent. The levels for the gap interval and gap duration for the ACG message should be selected in a manner that the system throughput does not fall below an optimum value. Additionally, the response time for the accepted service requests should not be much different from the service response time in normal conditions. When the switch receives an ACG message for an existing control, it replaces the old control with the new control.

The gap levels are selected for each service exceeding its threshold value in the following way. The control list is scanned to determine if a control for the service is already active. The possibility of having and not having a control active for the service is dealt with below:

Case 1 - Control for the service not found in list

If there is no control active for that service a new control must be applied. To select the gap levels the threshold level for this service is examined. Since for each type of service request different number of queries have to be processed, same threshold values for two different services may require different controls. For example, if Transfer and ACD Call Distribution service are both assigned a threshold value of 10 queries a second, 5 Transfer service requests per sec will be allowed, as compared to 10 service requests per sec for Call Distribution. That is because, for each Transfer service request 2 queries are processed by the SLEE, as compared to 1 query for the other (Refer Table 5.3). By choosing a gap interval time of 0.2 sec the Transfer queries can be restricted to 5 per second. Similarly for Call distribution, 0.1 sec gap interval would limit the queries to 10 per second. Therefore, to determine the maximum call rate for a service, the threshold value is divided by the number of queries required per call for that service, giving the *adjusted threshold value*. Then the *Gap Level* corresponding to the adjusted threshold value is selected. *Gap interval level* and *Gap duration level* values have been assigned to each gap level. (The adjusted threshold values, and corresponding gap levels, with gap interval levels and gap duration levels are illustrated in Table 5.4). These values are selected for implementing control at the switch. But before finally deciding the values, the extent of congestion is also taken into account. If the overload level is very high, then the gap level selected before is incremented by one. This will cause overgapping in the next monitor period, but it give the SLEE an opportunity to clear its backlog. The final gap values are then sent to the switch in an ACG message.

Adjusted Threshold Values (Calls/sec)		Gap Level	Gap Interval Level	Gap Duration Level
Minimum	Maximum			
4 0	10 0	1	2	3
2 0	4.0	2	3	4
1 0	2 0	3	4	5
0.50	1 0	4	5	6
0 20	0.50	5	6	7
0 10	0.20	6	7	7
0.0667	0 10	7	8	8
0 0333	0.0667	8	9	9
0 0167	0 0333	9	10	10
0 0	0 0167	10	11	10

Table 5 4 Threshold to Gap Level Mapping

Case 2 - Active Control for the service found in list

In case a control is already present for this service, it compares the current overload level with the overload level that was detected when the control was last updated. Three possibilities arise:

1. *Previous overload level is the same as current overload level* In that case the same values for the gap levels are taken and the ACG message sent.
2. *Previous overload level is less than current level* If the previous gap interval level is less than the maximum value, the previous gap level value is incremented by one and selected as the new value. Otherwise the old values are used for the ACG message.
3. *Previous overload level is greater than current overload level* If the old gap interval level is greater than 2, then the value is decremented by one, and a new ACG message with the reduced gap interval value is sent to the switch. Otherwise the gap interval level is not reduced any further. The same values are used for the ACG message.

B. Overload level equal to zero

If the overload level is detected to be zero, indicating absence of congestion, then if any controls are active, they should be changed. One approach could be to remove the active controls. But if the burst of service requests is not yet over, sudden removal in all controls could cause very high congestion. To prevent the system from being driven into congestion, the level of gapping applied for these controls is gradually reduced. For each active control the overload level associated with it is examined. If the previous overload level was very high then the existing gapping levels applied must also be high. Therefore

the gaps interval level is reduced by two levels. If the previous overload level was not very high, the gap interval level is decremented by one. If for any of the active controls, the gap interval level was 2, which is equivalent to the smallest gapping level, then it cannot be decremented any further. In this eventuality the control is removed from the list. New values for the existing controls are selected and sent in an ACG message to the switch.

Table 5.1 gives the mapping of the gap interval levels to the actual gapping periods

5.11 Simulation Parameters and Results

To analyse the SCP performance under overload conditions, an AIN system has been simulated. Before presenting the results obtained from this system, the parameters that need to be specified for a simulation run, have been described below:

SCP Service Rates

Service rate values are assigned to the "Service Rate" attributes of the SLEE and SLP processes within the SCP. These parameters determine the SCP's query handling capacity, and play a critical role in the calculation of service usage thresholds.

A service rate of 120,000 bps has been taken for the SLEE, and 40,000 bps for an SLP. Since the message size has been taken at a fixed value of 1200 bits, it implies that the service time for each message is 10 ms at the SLEE, and 33.3 ms at the SLP. A large SLP service time has been taken because SLPs would require to lookup databases to complete service related processing. Please note that because of lack of information regarding the actual processing delays offered by existing SCP platforms, these values have been selected taking into consideration the maximum specified response time of 150 ms [3]. Therefore, they may or may not be close to processing speeds of real systems.

Monitor Periods

The effectiveness of SMS control depends on the monitoring capabilities of the SCP, and the rate at which reports are presented to it by the SCP. If the monitor periods are too long, detection of overload may occur after congestion has reached dramatic levels. In that case the SCP might get totally choked, and as a consequence, performance may fall to very poor and unacceptable levels.

Similarly, if the load on the SCP falls from very high overload levels to normal levels, then this change in load should be reported quickly to the SMS. If monitor periods are long, this change in status will not get reported to the SMS for some time. Then because

of the controls previously implemented by the SMS, service requests may be turned down by the switch even though the SCP is now free to process them. Therefore, to achieve the best possible SCP performance, monitor periods have been kept small. The actual size of the monitor period is determined experimentally.

Network Configuration

For each simulation run, different generator types may be selected for separate generator modules within the switch side node model. Selecting different generator types alters the nature of traffic generated for the SCP by the system. For example, if an overload condition is to be created as a result of excessive use of Call Back service, a bursty generator process is employed for Call Back service. For other services, instances of a poisson generator process are invoked. Multiple instances of the bursty generator process may be used for creating overloading by more than one service.

Traffic Mix and Call Generation Rates

Maximum limits for usage of each service are defined by the system manager, after deciding the traffic mix desired for that time. A different set of values may be used at different hours of the day. These would be decided based on previous usage patterns and other business decisions. To simulate this behaviour, different sets of traffic mix values have been used.

For each desired traffic mix selected for the SCP, the values of the following parameters are calculated:

- Total call handling capacity of the SCP.
- Threshold values for each service.
- Call generation rates for each service, to achieve the traffic mix.

Taking the example mentioned in section 5.9, if call handling capacity is calculated at 16 calls/sec, then the call generation rate for a service is calculated as follows.

Call Forwarding service is allocated a maximum 6.67% of total traffic. Therefore the generator can create a maximum of $(16 * 6.67) / 100 \cong 1.07$ calls/sec. This value is assigned to the "Calls per sec" attribute of the Call Forwarding service poisson generator. Similarly, for all services, a value close to the calculated value is used as the generation rate. The method for calculation of thresholds has already been explained in section 5.9.

For services that employ bursty generators, a range of generation rates has to be specified. The minimum generation rate is taken at 1 call/sec, while the maximum goes up to 2 times more than the calculated value for that service. The burst duration range has been taken as 30 - 90 seconds.

We have executed many sets of simulations, changing one parameter at a time. The results that were obtained are analysed in section 5.11.2. But first, the performance of the SCP in the absence of any controls has been evaluated, and the results are presented in section 5.11.1.

5.11.1 SCP Performance without ACG controls

The desired traffic mix, as specified in section 5.9, has been taken as

<i>Service</i>	<i>Maximum Percentage of Total Service Requests</i>
Call forward	13.33%, (Activation - 3.33%, Deactivation - 3.33%, Servicing - 6.67%)
Call back	40.00%,
Call transfer	13.33%,
ACD	33.34%, (Call Distribution - 26.67%, (Agent Logon, Logoff, Make Busy, Make Unbusy - 1.167% each)

SCP call handling capacity is calculated to be 16 calls/sec, for this traffic mix. To drive the load at near maximum capacity, the call generation rates used are.

Call Forward Service	1.07 calls/sec,
Call Forward Activate, Call Forward Deactivate	0.53 calls/sec,
Call Back	6.40 calls/sec,
Call Transfer	2.13 calls/sec,
ACD Call Distribution	4.27 calls/sec,
ACD Agent Logon, Agent Logoff,	0.26 calls/sec,
ACD Agent Make Busy, Agent Make Unbusy	0.26 calls/sec

To provide bursty traffic, a bursty generator has been employed for Call Back service. Minimum and maximum values of 1 and 20 calls/sec respectively, have been specified as the range for call generation rates. The burst durations range between 30 and 90 seconds. For other services, a poisson generator has been used.

As no controls are implemented, the entire offered call traffic is sent from the switch to the SCP, therefore throughput is 100%. This is the reason why the Offered Load and Carried Load curves, drawn in fig. 5.10(a), overlap. But, when the load on the SCP exceeds the maximum capacity, queues of queries waiting to be processed, begin to develop. As a consequence, the response times rise sharply. In this case they are observed to rise as high as 45 seconds, whereas the maximum acceptable values lie in the 150 to 200 milliseconds range. The response time plot obtained for this simulation is shown in fig. 5.10(b).

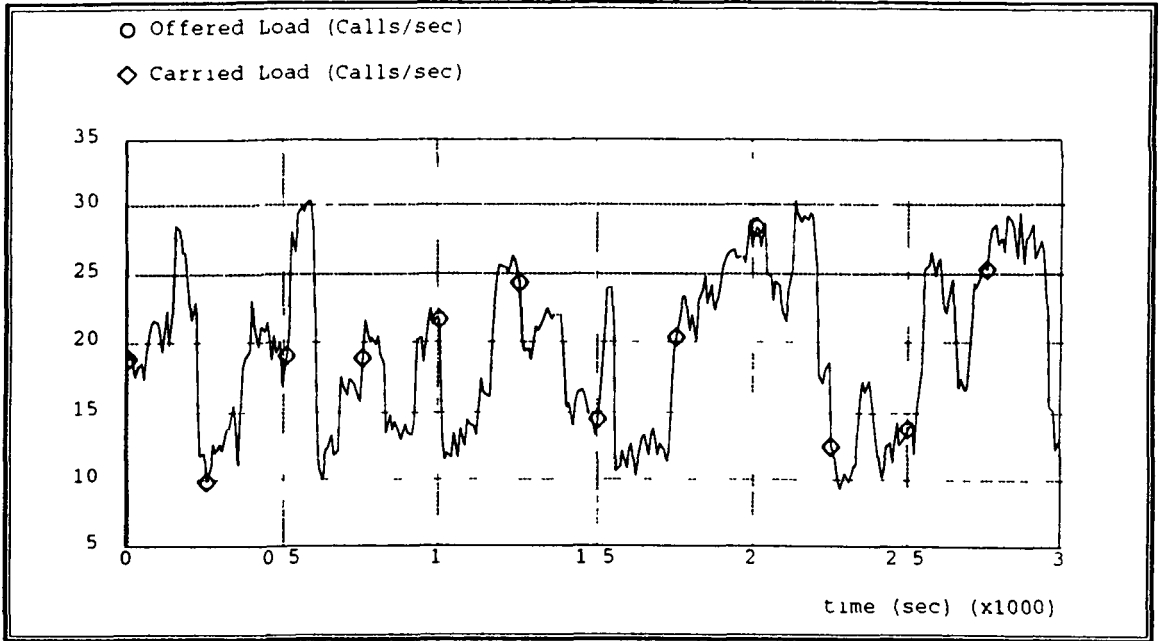


Fig 5 10(a) Offered Load and Carried Load Vs Time

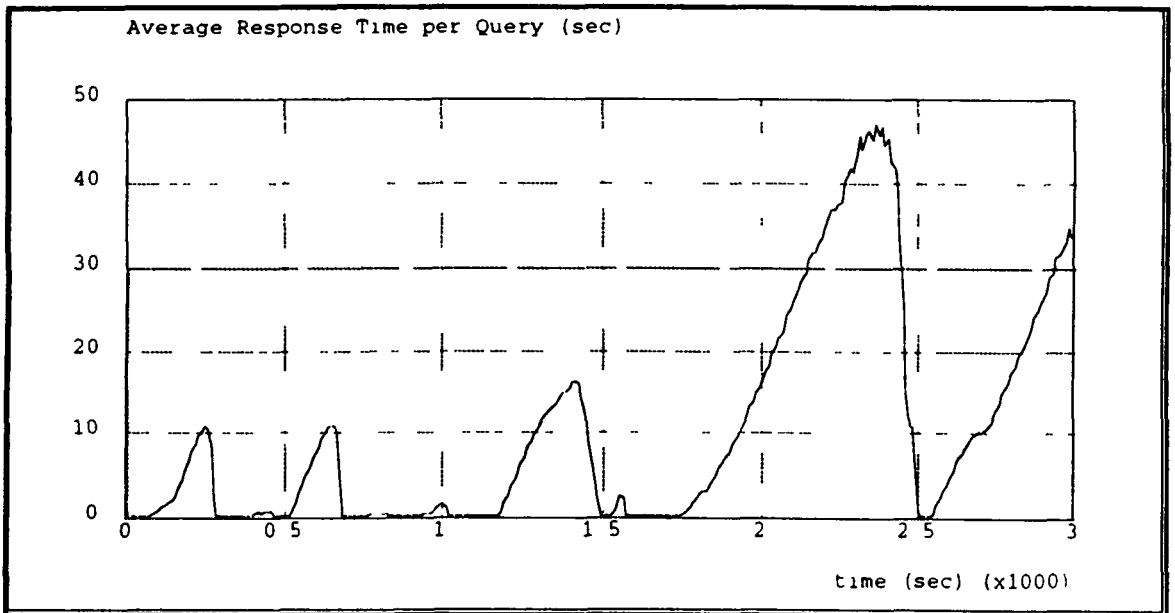


Fig 5 10(b) Average Response Time Vs Time
(Without ACG Controls)

5.11.2 SCP Performance with SMS Originated ACG Controls

At times of overload, when SMS selected ACG controls are implemented at the switch, then for the same traffic mix, a marked improvement in the performance of the SCP is expected. This NTM system is expected to maintain good SCP throughput under overload, and ensure that requests accepted by the system are serviced within reasonable time periods.

To determine its performance, the network with the same configuration and parameters is used. Additional parameters that need to be determined include the threshold values, and the monitor periods. The threshold values are calculated to be 1.00 query/sec for Forward servicing, 18.00 queries/sec for Call back service, 4.00 queries/sec for Transfer service, and 6.00 queries/sec for ACD Distribution. The monitor period is taken to be 1 second.

The graphs obtained for this simulation show that the average throughput at times of overload is around 12.5 calls/sec, which is over 75% of maximum capacity. The load and throughput curves are illustrated in fig. 5.11(a). A section of the response time curve obtained, is also plotted in fig. 5.11(b). Although, for some queries the response time exceeds the maximum limit, for a very large proportion of them, it is within acceptable limits. In fact, the response time for 90% of the queries accepted, is found to be less than 100 ms. Another 9% of the queries are seen to have response times less than the maximum limit of 150 ms. The cumulative distribution plot of the response times, shown in fig. 5.11(d), ratifies this observation. A moving average plot of the response times, drawn in fig. 5.11(c), further confirms. Therefore, for subsequent simulation runs, the average response time plot has been shown.

Another feature that is observed from the throughput and response time plots, is their unsteady nature. This behaviour can be attributed to the restrictive nature of ACG controls. The gap intervals used to limit the queries do not provide very precise control. In some cases, the desired call rates are such that no gap interval value permits the queries to be limited to that call rate. Then, the gap intervals that can limit the call rate to near that value are used. Taking the threshold value of Call Back service for this simulation as an example. It has been calculated at 6 calls/sec (i.e. 18 queries/sec). If Call Back service requests are generated at a rate higher than the threshold value, ACG controls are selected. The possible choices of gap intervals that could limit the call rate to around the threshold value are 0.1 sec. and 0.25 sec. The former gap interval provides undergapping, while the latter causes overgapping of the Call Back calls. Normally, to control congestion, overgapping is preferred over undergapping, therefore the 0.25 sec. value would be selected at first. This control limits the Call Back service usage rate to 4

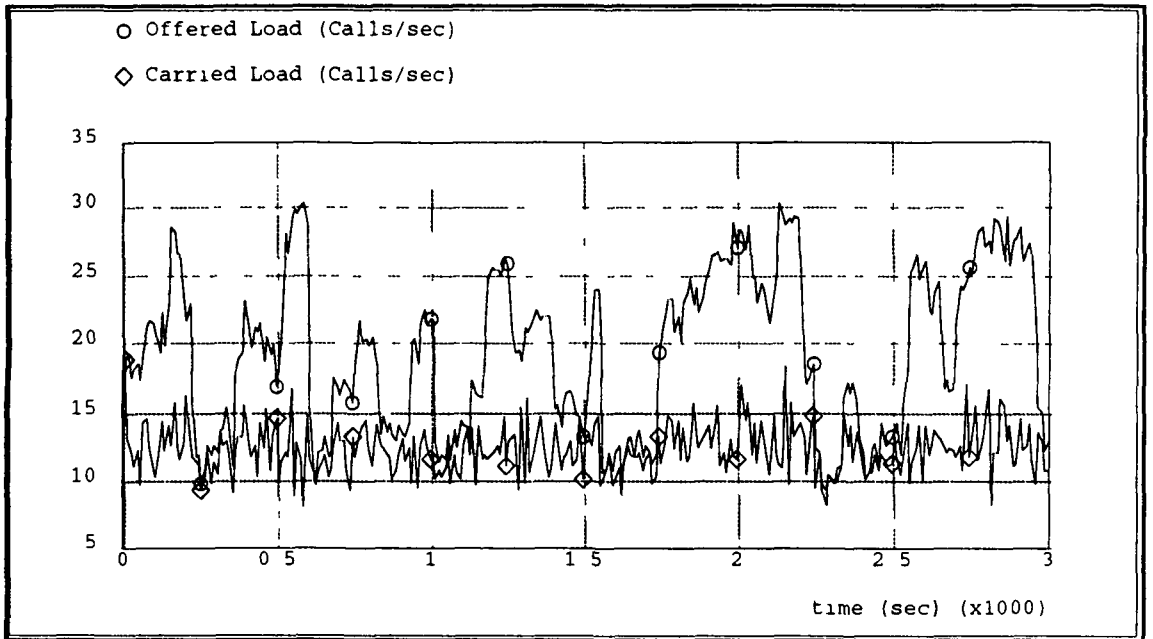


Fig 5 11(a) Offered Load and Carried Load Vs Time

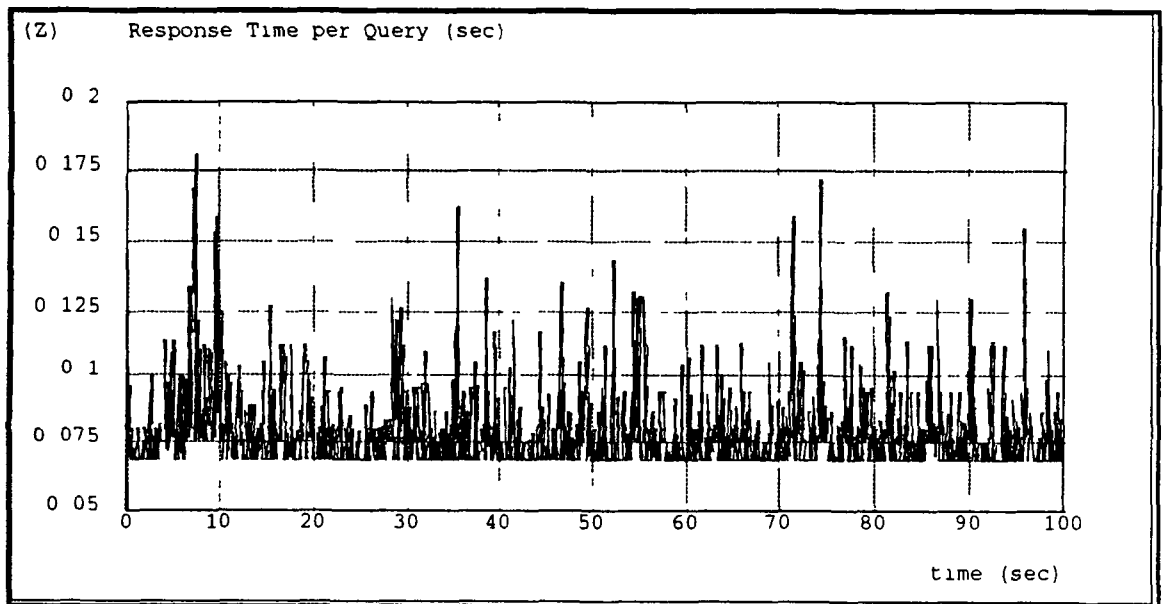


Fig 5 11(b) Response Time Vs Time
(With ACG Controls, Call Back Service Load Bursty, Monitor Period 1 sec)

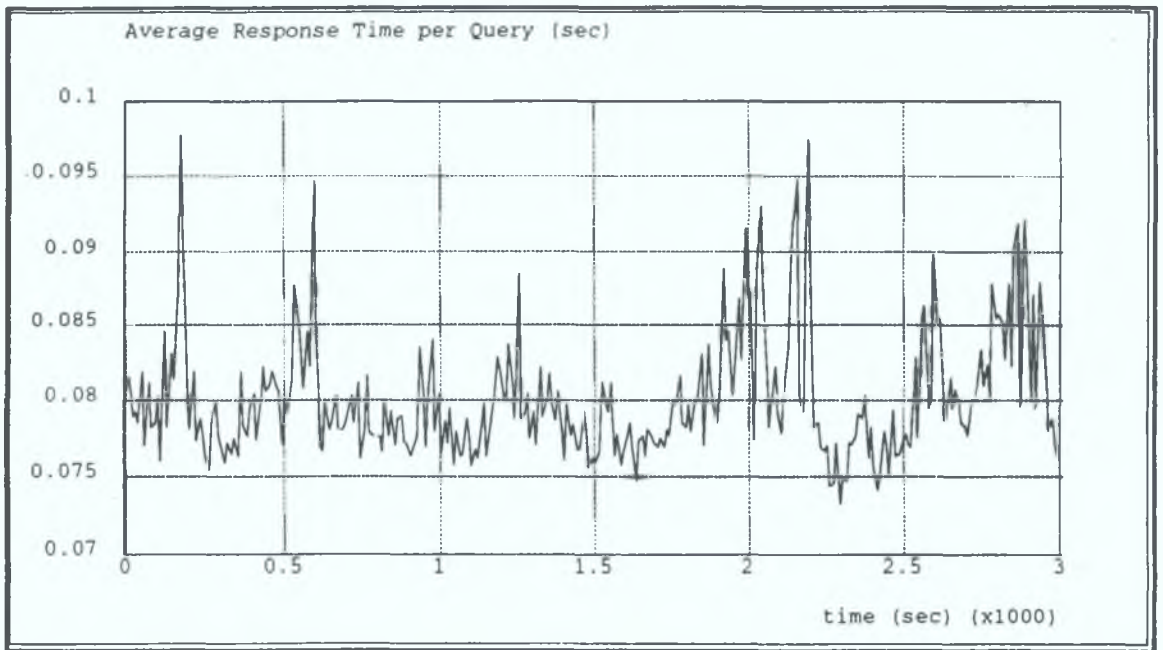


Fig. 5.11(c) Average Response Time Vs. Time

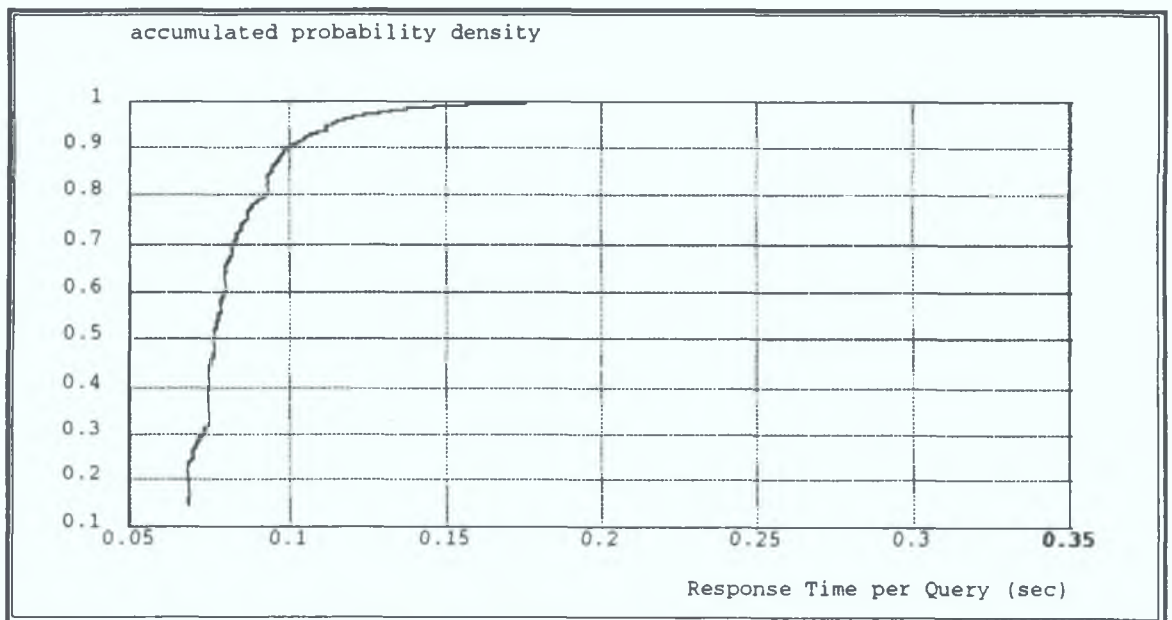


Fig. 5.11(d) Cumulative Probability Distribution of Response Time
(With ACG Controls, Call Back Service Load Bursty, Monitor Period 1 second)

calls/sec , even though the maximum allowed limit is 6 calls/sec. Because the system is overgapped in this monitor period, the response time improves, and therefore the need for updating the controls arises. Subsequently, the 0.1 sec gap interval will be used in the next monitor period. This would allow too many requests to be accepted, thereby causing an increase in response times. As a result, both throughput and response time values are found to periodically increase and decrease, instead of remaining at steady levels.

The results obtained by varying the network parameters have been discussed below.

Results obtained by varying monitor periods

To determine the effect of changing monitor periods, a 5 second long monitor period was taken. At times, when the load suddenly became very high, the throughput rose above the maximum capacity of the SCP. As a consequence, there was a sharp increase in query response time. This can be attributed to the fact that congestion may have been detected and reported some time after its actual onset. This results in delayed deployment of controls, and subsequently, a degradation in performance. Therefore, a small monitor period of 1 second has been taken for further simulation runs. The load and throughput curves are shown in fig 5.12(a), while the average response time plot is given in fig 5.12(b). The cumulative probability curves obtained for response times, with a 1 second and 5 second monitor period, are illustrated in fig 5.12(c). It can be observed that for a 1 second monitor period, 99% of the queries have a response time less than 150 ms, whereas in the other case, response time of only 89% of the queries could be kept below the 150 ms value.

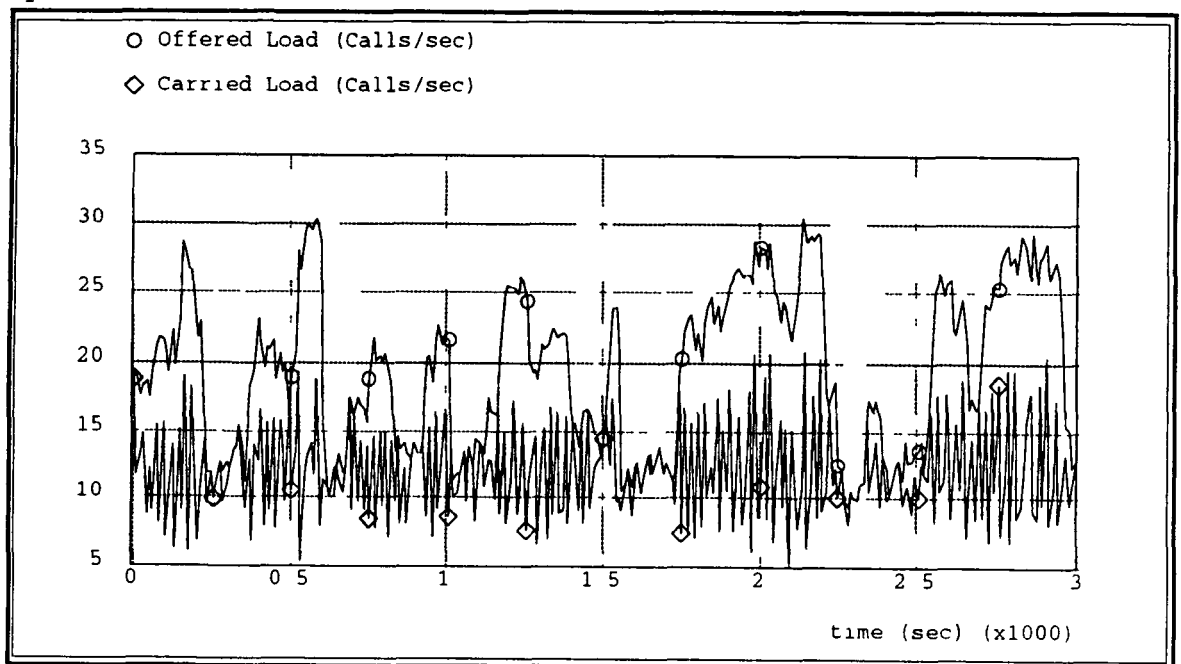


Fig 5.12(a) Offered Load and Carried Load Vs Time

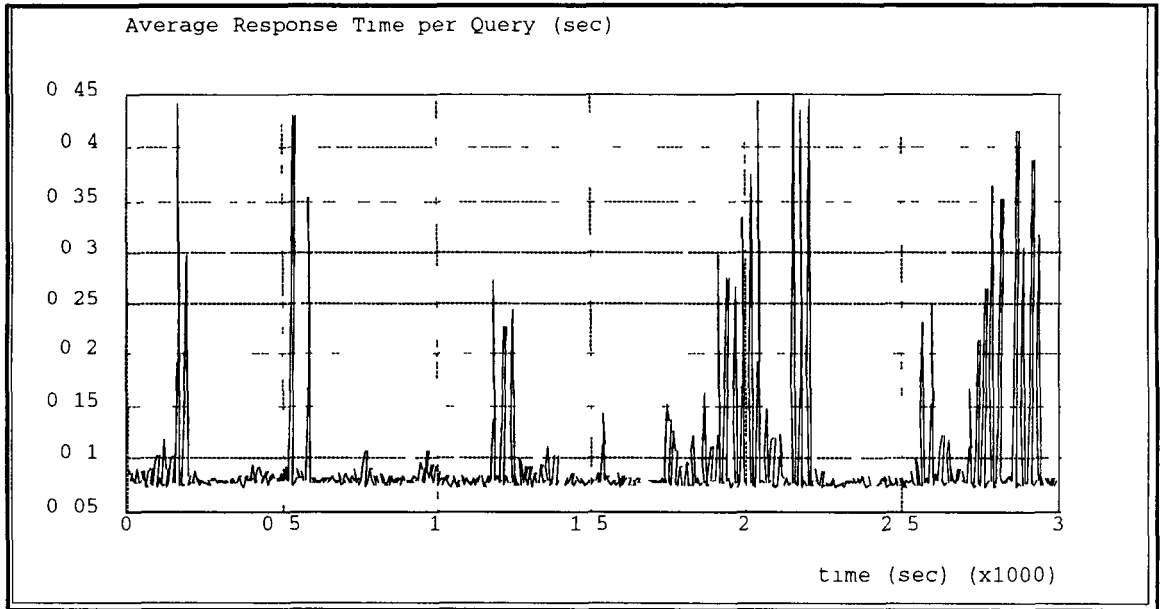


Fig 5.12(b) Average Response Time Vs Time
(Call Back Service Load Bursty, Monitor Period 5 sec.)

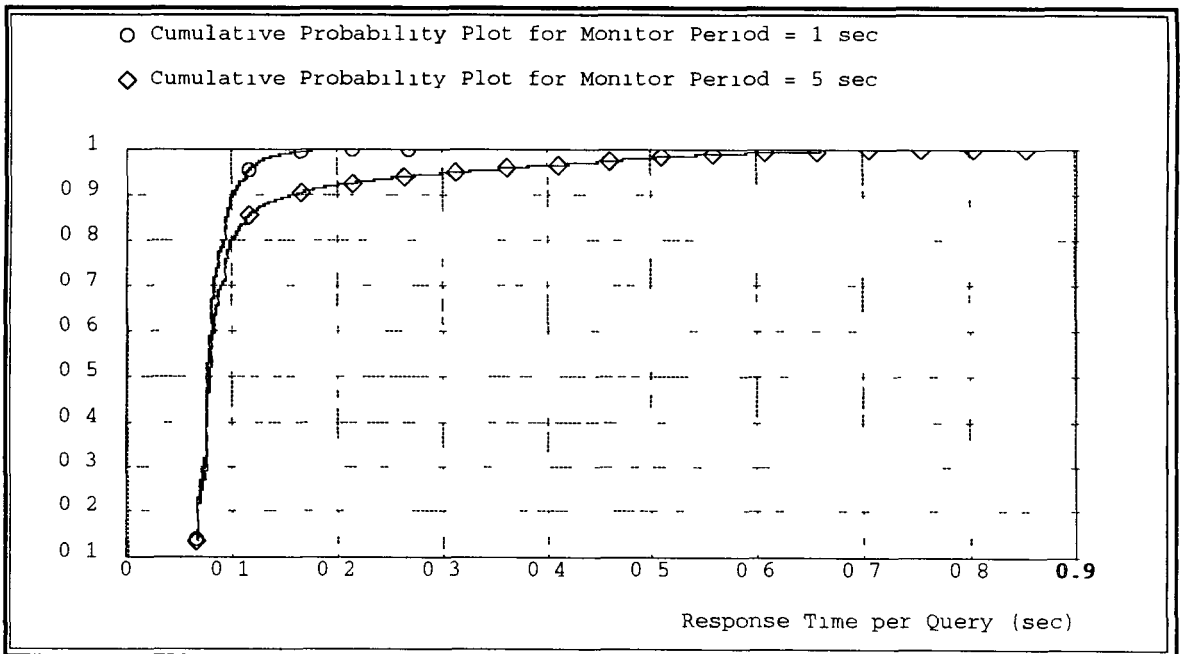


Fig. 5 12(c) Comparison of Cumulative Probability Distributions of Response Time for Monitor Periods of 1 sec. & 5 sec. (Call Back Service Load Bursty)

Results obtained by varying the network configuration

If the bursty generator for Call Back service is replaced by a poisson generator, and the poisson generator for ACD Call Distribution service replaced with the bursty type, a different set of results are obtained. As explained earlier in the chapter, Call Back service requires more resources in terms of number of queries that need to be processed for each request, as compared to the ACD Call Distribution service. As a result, with the same offered load, the average carried load for this configuration is found to be slightly more. The results for this simulation are illustrated in fig 5.13(a) and fig 5 13(b)

In yet another configuration, bursty generators are implemented for both Call Back and ACD Distribution service. The graphs obtained are illustrated in fig 5 14(a) and 5 14(b)

Results obtained by varying the desired traffic mix and call generation rates

Another set of simulations were done with a different desired traffic mix. The traffic mix is as follows.

<i>Service</i>	<i>Maximum Percentage of Total Service Requests</i>
Call forward	20.00%, (Activation - 2.00%, Deactivation - 2.00%, Servicing - 16.0%)
Call back	20.00%,
Call transfer	30.00%,
ACD	30.00%, (Call Distribution - 26.00%, (Agent Logon, Logoff, Make Busy, Make Unbusy - 1.00% each)

The call handling capacity of the SCP for this traffic mix is calculated to be approximately 18 calls/sec. To achieve this traffic mix the call generation rates are calculated to be.

Call Forward Service	2.88 calls/sec ,
Call Forward Activate, Call Forward Deactivate	0.36 calls/sec ,
Call Back	3.60 calls/sec ,
Call Transfer	6.00 calls/sec ,
ACD Call Distribution	5.28 calls/sec.,
ACD Agent Logon, Agent Logoff	0.18 calls/sec.,
ACD Agent Make Busy, Agent Make Unbusy	0.18 calls/sec

The threshold values are calculated as 2.80 query/sec for Forward servicing, 10.05 queries/sec. for Call back service, 10.05 queries/sec for Transfer service, and 6.80 queries/sec for ACD Distribution

In the first case, a bursty generator has been employed for Call Back service. Minimum and maximum values of 1 and 12 calls/sec respectively, have been specified as the range for call generation rates. The burst durations range between 30 and 90 seconds. For other services, a poisson generator has been used. Fig. 5.15(a) displays the Offered Load and Carried Load curves, whereas fig. 5.15(b) shows the average response time per query.

In the second set of plots, both Call Back and ACD Distribution service loads are made bursty. The results obtained are shown in figs. 5.16(a) and 5.16(b).

From these results we can conclude that with the algorithm designed for selecting appropriate ACG controls, and implemented at the SMS, provides effective overload control for regulating the flow of messages between the switch and the SCP. With the help of the ACG control system implemented on the AIN network, the SCP can provide acceptable grades of service in overload conditions. The response to most queries are received within acceptable time limits, so that calls do not fail on account of time-out occurring. The control strategy also prevents any one service from degrading the service quality for other services, as it restricts their levels of usage to pre-defined maximum levels. In addition, at times of overload, the SCP throughput is prevented from falling to very low levels, in fact it operates between 75% to 90% of maximum capacity.

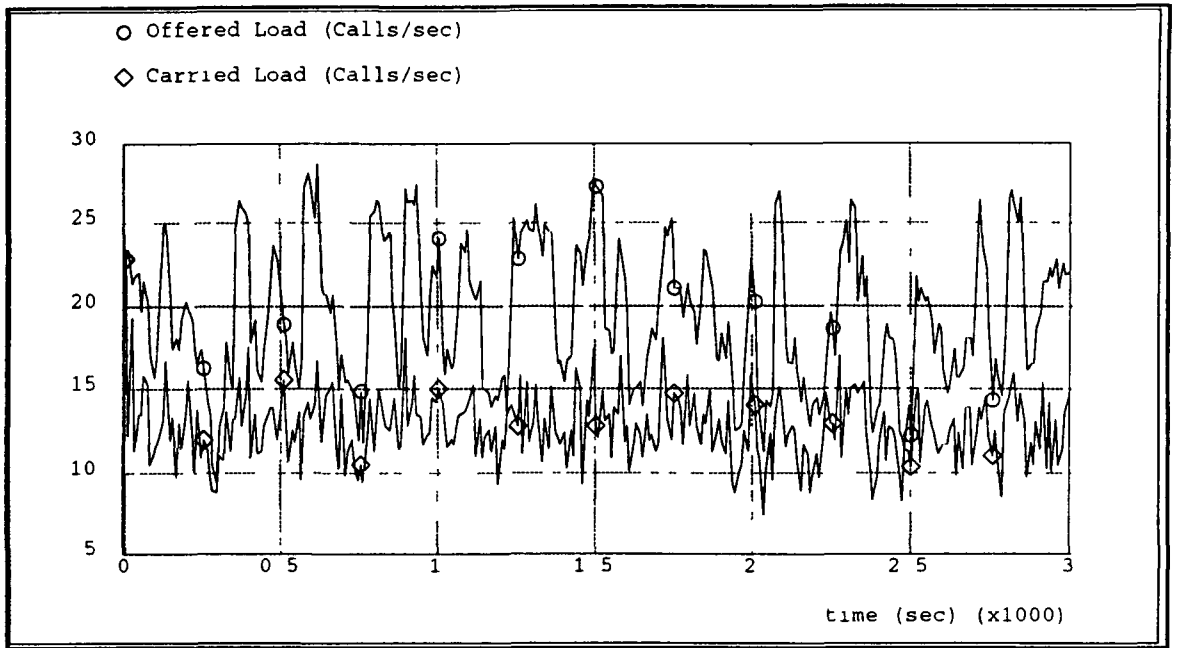


Fig. 5 13(a) Offered Load and Carried Load Vs. Time

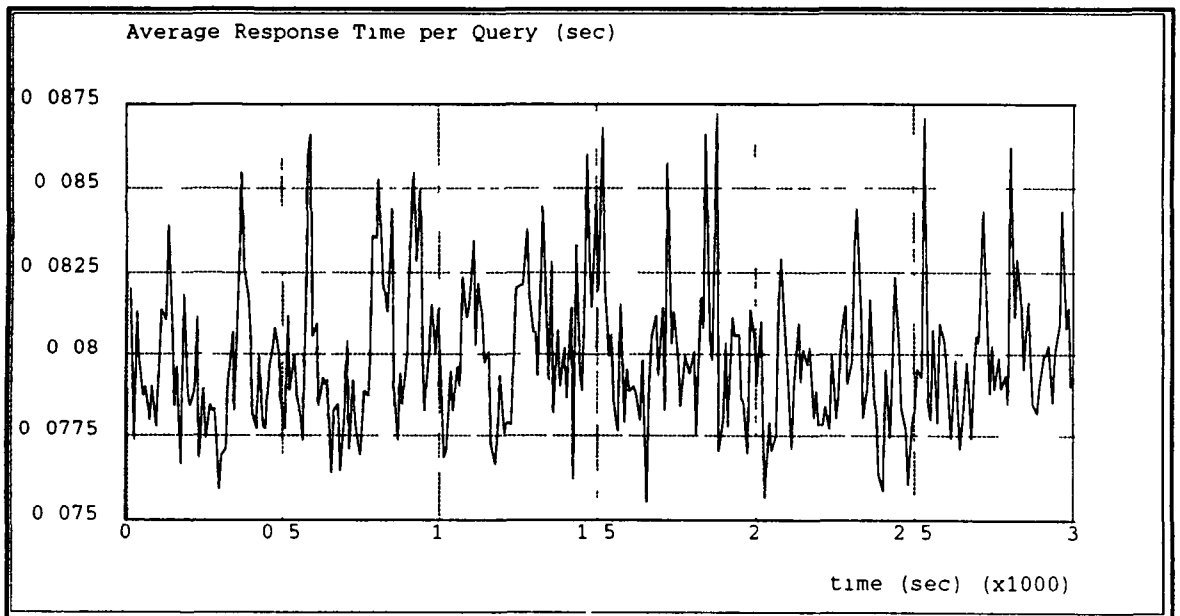


Fig. 5.13(b) Average Response Time Vs. Time
(ACD Distribution Service Load Bursty)

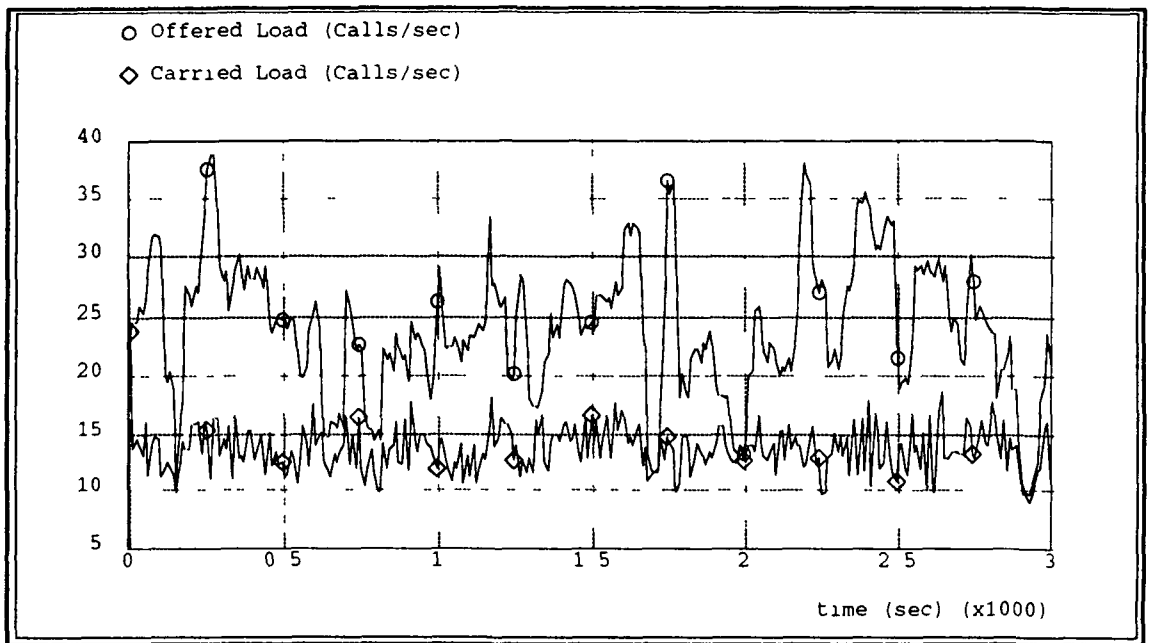


Fig 5 14(a) Offered Load and Carried Load Vs Time

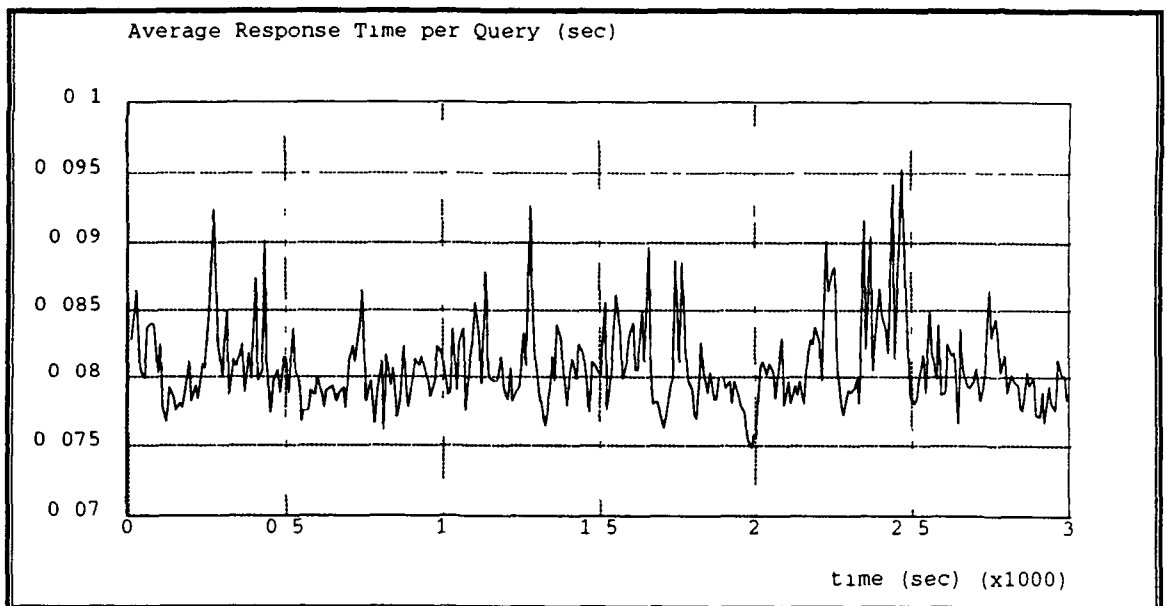


Fig 5 14(b) Average Response Time Vs. Time
(Call Back and ACD Call Distribution Service Load Bursty)

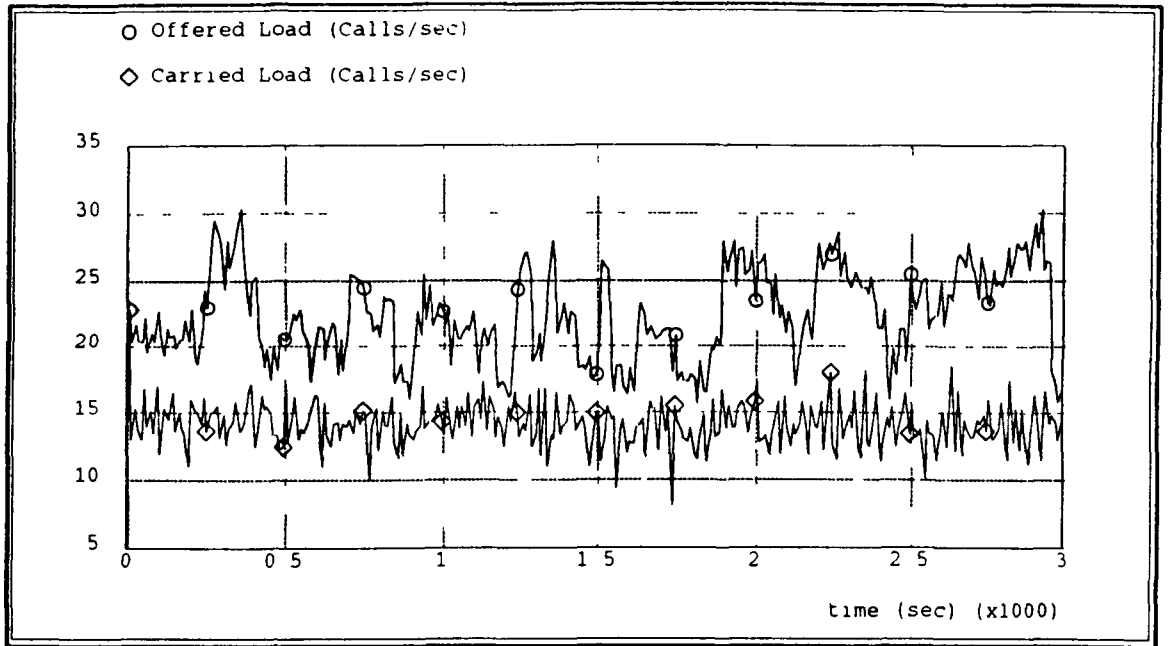


Fig 5 15(a) Offered Load and Carried Load Vs Time

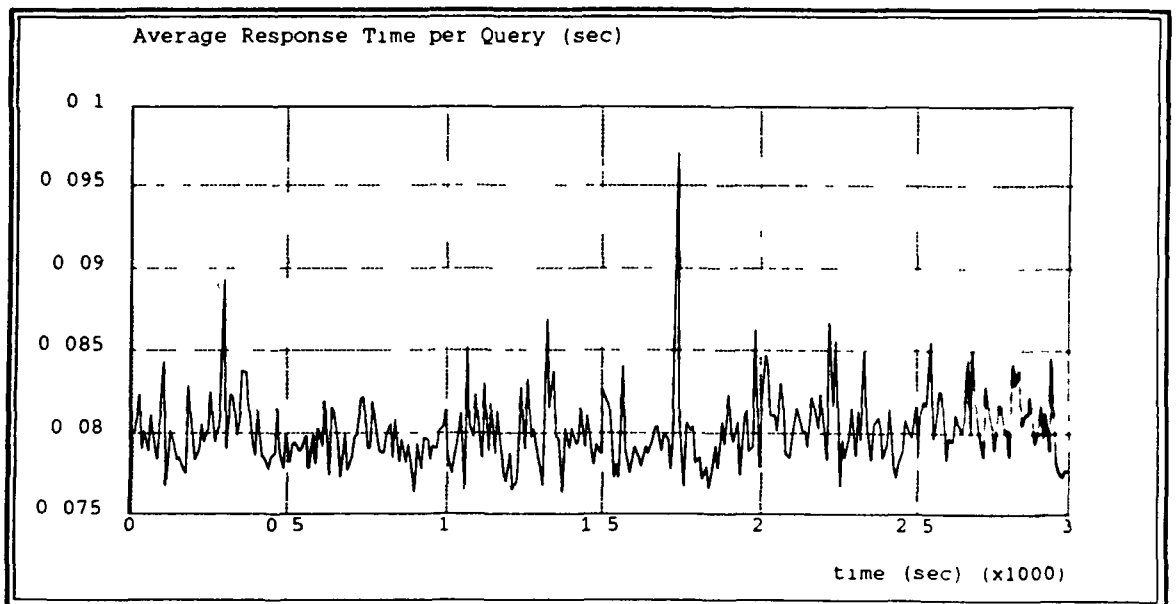


Fig 5 15(b) Average Response Time Vs Time
(Call Back Service Load Bursty, Second Traffic Mix)

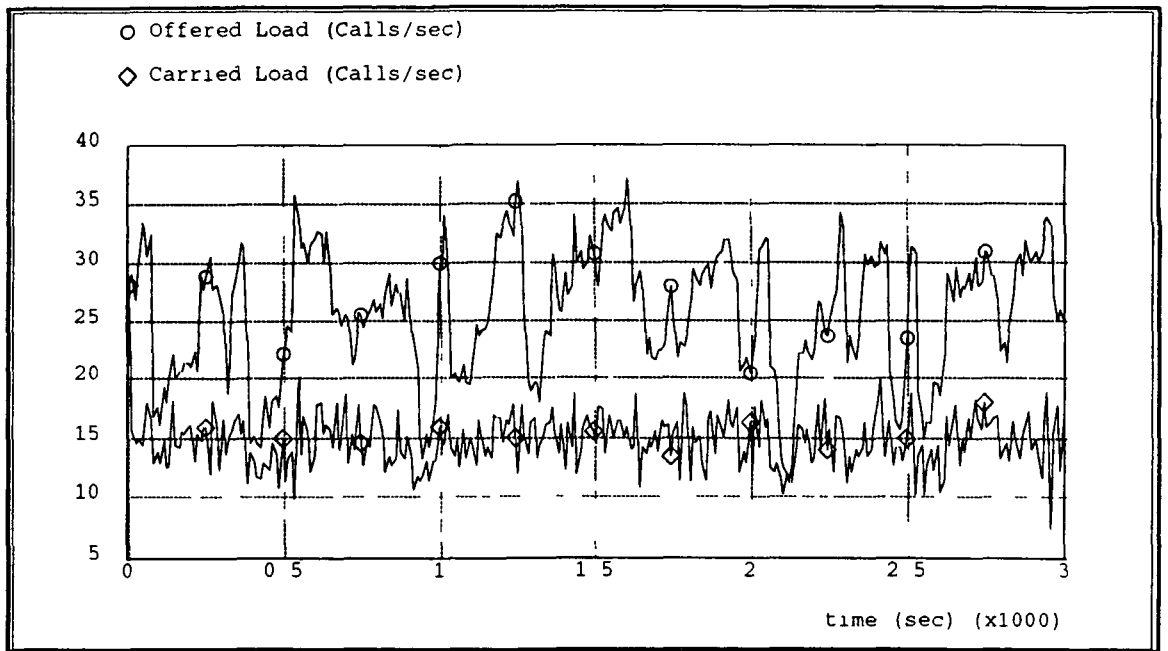


Fig. 5 16(a) Offered Load and Carried Load Vs Time

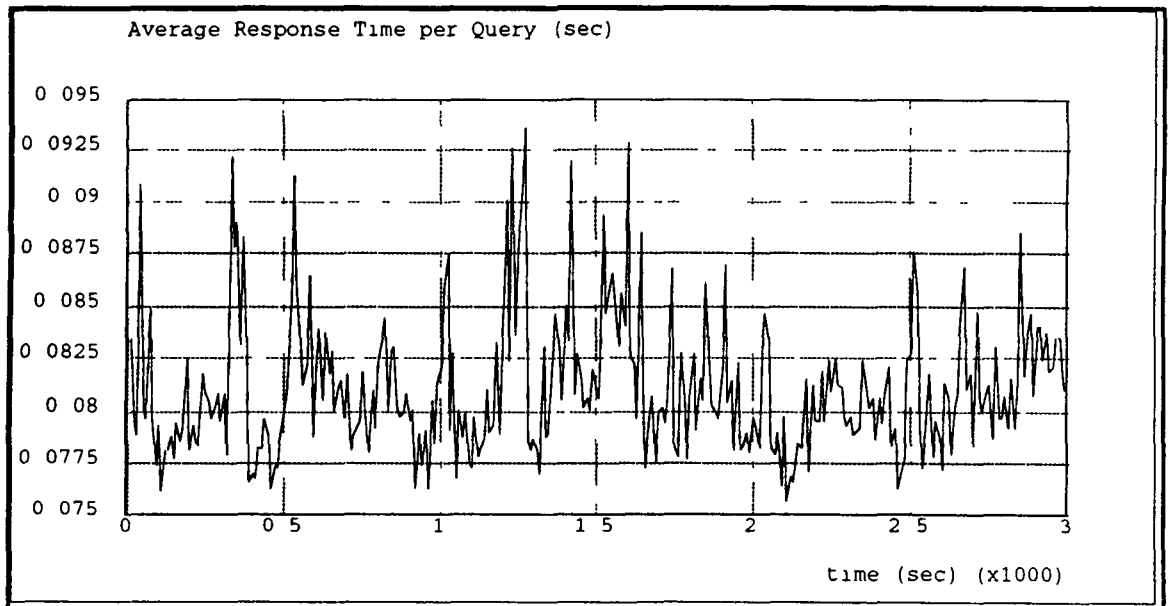


Fig 5 16(b) Average Response Time Vs. Time
(Call Back and ACD Call Distribution Service Load Bursty, Second Traffic Mix)

Chapter 6

Conclusions

6.1 Summary and Conclusions

The objective of this research effort was to model and simulate a telecommunication system based on the Advanced Intelligent Network (AIN) architecture for the provision of telecom services. In addition, the performance of the Service Control Point under overload conditions was to be evaluated.

To simulate the telecom system, an in-depth study of the AIN architecture was undertaken. With our understanding of the network's operation, a system was modelled. The components implemented to realise the network included the AIN Switching System, the Service Control Node and the Intelligent Peripheral node. These components were equipped with necessary functionality to allow them to interact with each other using standard interfaces, defined by Bellcore. In addition, each node was realised to furnish a certain set of capabilities to contribute to the operation of the network.

AIN Switch Capabilities were implemented in the Switching System node. The Switch was equipped with functionality to setup, maintain and clear calls between participating users. In addition, the AIN Call Model was implemented on the Switch. It provides the functionality to detect requests for AIN service logic processing, report these requests to the Service Control node, and perform actions as directed by the Service Control node to complete the service requests.

The Service Logic Execution Environment was implemented in the Service Control node. It is responsible for responding to queries sent by the Switching System, and providing AIN service logic processing. The Intelligent Peripheral node was implemented to provide user-network interaction. It manages resources for playing announcements to users, and collecting information from them.

After simulating the entire system, a few telecom services were deployed to demonstrate its operation. Call Forwarding, Call Transfer, Call Back and Automatic Call Distribution (ACD) services were selected for implementation on the network. With the exception of the Transfer Call service, each of these services were designed to use only those messages that belonged to the standard message set, to communicate with the Switch. In case of Transfer service, a non-standard message, "Transfer Call", was used. This was done because no appropriate set of messages were found that could request the Switching System to perform the operations necessary for the provision of this service. We would therefore recommend the addition of a message to the standard set, which demands equivalent actions from the Switching System for completion of Transfer service requests. Alternatively, this service may be implemented as a switch-based service, as it is in existing systems.

To ensure correct operation of the simulated system, it was extensively tested. Numerous test programs were written to determine how the system responds to requests for different services. These test programs generated correct as well as incorrect requests for the system. The responses received for these requests were then compared with the results expected. We were successful in demonstrating the setup of calls between users, the actual exchange of information between them while the calls were active, and the release of allocated resources, with requests for clearing the calls. The provision of services to the requesting users was also exhibited. Thus, we achieved our objective of demonstrating the service creation and control capabilities of the AIN architecture.

For analysing the performance of the Service Control Point under service-specific overload conditions, the existing system was found inappropriate. This system has been designed for complete call setup, and therefore maintains call related information for the duration of the call. Using this system for generating large volumes of load, would create enormous memory requirements. Moreover, for evaluating performance of the SCP, call set up at the Switch is not necessary. Therefore, a functionally simpler system was implemented. This system was designed to reproduce network stress conditions. Further, the Automatic Code Gap (ACG) control based SMS Originated Code Control (SOCC) scheme was implemented on the system. The functionality provided in the various components of the system included monitoring and overload detection capabilities at the SCP, control selection capabilities at the Service Management System, and implementation of ACG controls at the Switching System. An algorithm for selection of appropriate controls was also devised, and implemented at the SMS.

Simulations were performed to analyse the SCP performance, where overload conditions were created. In the absence of any control scheme, the grade of service offered by the

SCP was observed to be totally unacceptable. The response to queries sent from the switch were received well outside their time-out periods, resulting in call failure. Subsequently, the controls were activated, while maintaining the same network condition. The controls were effective in controlling development of congestion at the SCP, and thus providing acceptable grades of service at times of overload. More simulations were performed for different load conditions, created by varying the network parameters. Results for all simulations were observed to be well within acceptable limits.

The algorithm deployed at the SMS, for selection of controls, restricted the level of usage of each service to pre-defined maximum limits, and therefore prevented any one service from degrading the quality of other services. It also maintained good throughput levels, and kept response time for queries low. The only problem encountered while using this control scheme, was the fluctuation seen in the throughput and response time plots. That occurs because the gap interval timer values are explicitly defined in the standards[4]. As a result, they force the control management system to select controls that may not be very accurate. However, these values have to be standardised to meet the conceptual requirements of AIN, which is to provide standard interfaces between components of the network so that they can be supplied by different vendors. Overall, the control system measured up to its expectations and achieved the objectives of a Network Traffic Management system.

6.2 Directions for Future Work

We have implemented the ACG controls on the system for preventing overload at the SCP. Schemes for managing traffic and controlling congestion at the Switching System, could also be investigated and implemented, to obtain good Switching System performance.

As our primary objective was to demonstrate the overall operation of the network, and the relationships that exist between the Switching System and associated nodes, we have implemented a very simple Service Control node. As a consequence, the Application Programming Interface (API) defined for creating services was not implemented. Since the provision of a powerful service creation environment is an important part of the AIN concept, the Service Control node could be enhanced by developing the API on it.

References

- 1 Ameritech Technical Report AM-TR-OAT-000042, "Ameritech Service Switching Point Functional Specifications", Issue 1, July 1989
- 2 Andrews, M Tignol, "Network Traffic Management and its application for Intelligent Networks", Trends in Telecommunications, Vol. 6, No 1, June 1990
3. Bellcore Special Report SR-NPL-001263, "Advanced Intelligent Network Release 1 Network and Operations Plan", Issue 1, June 1990
- 4 Bellcore Technical Report TR-NWT-001284, "Advanced Intelligent Network Release 0 1 Switching System Generic Requirements", Issue 1, August 1992
- 5 CCITT Draft Recommendation I 1312/Q 1201, Principles of Intelligent Network Architecture
- 6 CCITT Recommendation Q 1211, Introduction to IN Capability Set 1
- 7 James Aitken, "Experiences of implementation using Advanced Intelligent Network Release 1 architecture", Software Engineering for Telecommunication Systems and Services, (SETSS 92), IEE, 30 March - 1 April 1992
- 8 Joaquin Dawis, Gary Loberg, "New services operations support in an intelligent network", Globecom '90, Dec. 2-5 1990
- 9 Jose M Duran, John Visser, "International Standards for Intelligent Networks", IEEE Communications Magazine, February 1992
10. Keith Mallinson, "Future of IN", The European IN Conference, London, May 1992
- 11 Naim A Kheir, *Systems Modeling and Computer Simulation*, Marcel Dekker Inc
12. Nicola Gatti, "State of the art and future perspectives of IN standards", The European IN Conference, London, May 1992
- 13 OPNET Modeling Manual, MIL 3 INC

14. P.M D Turner, P.B. Key, "A new call gapping algorithm for network traffic management", *Teletraffic and Datatraffic in a period of change*, Proceeding of the Thirteenth International Teletraffic Congress (ITC-13), Copenhagen, Denmark, Vol 14, June 19-26 1991.
15. Paul Bratley, Bennet L Fox, Linus E Schrage, *A Guide to Simulation*, Springer-Verlag
- 16 Richard A Farel, Mohan Gawande, "Design and analysis of overload control strategies for transaction network databases", *Teletraffic and Datatraffic in a period of change*, Proceeding of the Thirteenth International Teletraffic Congress (ITC-13), Copenhagen, Denmark, Vol. 14, June 19-26 1991
- 17 Roger K Berman, John H. Brewster, "Perspectives on the AIN Architecture", *IEEE Communications Magazine*, February 1992
18. Roger K Berman, S F. Knapp, R.F Baruzzi, "Evolvability of the Advanced Intelligent Network", *IEEE International Conference on Communications '91 (ICC '91)*.
19. Tim Hallis, "Intelligent Switching", *Communications International*, December 1992
- 20 Vapheas, B.A Polonsky, A M Gopin, R J Wojcik, "Advanced Intelligent Network. Evolution", *IEEE International Conference on Communications '91 (ICC '91)*
- 21 W D Ambrosch, A Maher, B Sasscer, *The Intelligent Network - A joint study by Bell Atlantic, IBM and Siemens*, Springer-Verlag

Appendix A

Acronyms

ACD	Automatic Call Distribution
ACG	Automatic Code Gap
AIN	Advanced Intelligent Network
AMA	Automatic Message Accounting
ANI	Automatic Number Identification
API	Application Programming Interface
ASC	AIN Switch Capabilities
ASLP	Administrative Service Logic Program
ATM	Asynchronous Transfer Mode
BCP	Basic Call Process
BCSM	Basic Call State Model
BRI	Basic Rate Interface
CCAF	Call Control Agent Function
CCF	Call Control Function
CCIS	Common Channel Interoffice Signalling
CCITT	International Telegraph and Telephone Consultative Committee
CPN	Calling Party Number
CS	Connection Segment
CS1	Capability Set 1
CSMP	Call State Model Process
DFP	Distributed Function Plane
DN	Directory Number
DTMF	Dual Tone Multifrequency
FC	Functional Component
FE	Functional Entity
FEA	Functional Entity Action
FSLP	Feature Service Logic Program
FSM	Finite State Machine
GFP	Global Functional Plane
GSL	Global Service Logic
IEE	Institute of Electrical Engineers
IEEE	Institute of Electrical and Electronics Engineers
IM	Information Management
IN	Intelligent Network
INCM	Intelligent Network Conceptual Model

IP	Intelligent Peripheral
ISDN	Integrated Services Digital Network
ISO	International Standards Organisation
Kbps	Kilobits per Second
Mbps	Megabits per Second
NA	Network Access
NAP	Network Access Point
NCP	Network Control Point
NPA	Numbering Plan Area
NTM	Network Traffic Management
OA	Operations Application
OCSMP	Originating Call State Model Process
OP	Operations
OPNET	Optimised Network Engineering Tools
OS	Operations System
OSI	Open Systems Interconnection
PBX	Private Branch Exchange
PE	Physical Entity
PIC	Point in Call
PIN	Personal Identification Number
POI	Point of Initiation
POR	Point of Return
PRI	Primary Rate Interface
PSPDN	Public Switched Packet Data Network
PSTN	Public Switched Telephone Network
QoS	Quality of Service
RCEE	Resource Control Execution Environment
SA	Service Assistance
SAC	Service Access Code
SCEF	Service Creation Environment Function
SCF	Service Control Function
SCP	Service Control Point
SDF	Service Data Function
SDL	Specification and Description Language
SF	Service Feature
SIB	Service Independent Building Blocks
SL	Service Logic
SL&C	Service Logic and Control
SLEE	Service Logic Execution Environment

SLP	Service Logic Program
SMAF	Service Management Access Function
SMF	Service Management Function
SMS	Service Management System
SN	Service Node
SNEE	Service Node Execution Environment
SOC	SCP Overload Control
SOCC	SMS Originated Code Control
SP	Service Provisioning
SPC	Stored Program Control
SR	Special Report
SRF	Specialised Resource Function
SS7	Signalling System 7
SSCP	Service Switching and Control Point
SSF	Service Switching Function
SSP	Service Switching Point
STM	Service Traffic Management
STP	Signal Transfer Point
TCAP	Transaction Capabilities Application Part
TCP	Trigger Check Point
TCSMP	Terminating Call State Model Process
TMN	Telecommunication Management Network
TR	Technical Report

Appendix B

Message Set

Bellcore has defined a message set for communication between the ASC and SLEE. We have implemented a subset of that message set in our system. A brief description of each message of that subset has been given in this appendix.

The messages are broadly categorised in two categories:

A ASC-to-SLEE Communication

B SLEE-to-ASC Communication

A. ASC-to-SLEE Communication

ASC messages are grouped into messages that can be either *initiating or subsequent* messages, and messages that can only be *subsequent* event messages.

A.1 Initiating or Subsequent Event Messages

Busy_Detected

The ASC sends this message to the SLEE when all facilities in the terminating resource group (e.g. a line for a single user) are busy. For example, this message may be sent to the SLEE, if Call back service has been subscribed by the busy terminating party. The Call back SLP responds with a Send_to_Resource message, instructing the ASC to play an announcement to the originating party indicating that the terminating party is busy, and prompt the party to indicate if call back is desired.

Feature_Requested

The ASC sends this message to the SLEE when it detects that a user depressed a flash-hook or a feature activator (e.g. the forward button). Forward and Transfer feature activators are supported by the system that has been implemented. A Feature_Requested event can occur in almost any state of a stable call. For example, Transfer may be requested in either the Waiting for Answer state or Active state.

Info_Collected

The ASC sends this message to the SLEE when it collects initial complete information from the user according to the appropriate dialling plan. This message is created by the ASC if the Info_Collected TCP is activated, or if a request on the event has been made by the SLEE.

Termination_Authorised

The ASC sends this message to the SLEE when it determines that the call is authorised to be routed to a terminating user. This message is created by the ASC if the Termination_Authorised TCP is activated, or if a request on the event has been made by the SLEE. For example, when a call has to be routed to a terminating user who has invoked the Forward_Call feature, in that case, the Termination_Authorised message is sent to the SLEE with the terminating user's DN as a parameter in the message. On receiving this message, the Forward SLP at the SLEE may request the ASC to route the call to the destination DN specified by the terminating user.

A.2 Subsequent Event Messages Only

Create_Call_Failure

The ASC sends this message to report that it failed in attempting to create a call that the SLEE had requested in a prior Create_Call message. For example, create call failure may take place when SLEE requests the ASC to setup a new call between two parties, and one of the parties is busy.

Create_Call_Success

The ASC sends this message to report that it was successful in creating a call that the SLEE had requested in a prior Create_Call message.

Entity_Status

The ASC sends this message to inform the SLEE of the entity status or change in status of a resource, as requested in a Monitor_for_Change or Continuous_Monitor message. Each message sent by the ASC contains a response_destination parameter along with the busy or free status, so that the SLEE can route the information to the SLP which requested it.

Join_Leg_Failure

The ASC sends this message to report that it failed to join a leg to a CS that the SLEE had requested in a prior Join_Leg message.

Join_Leg_Success

The ASC sends this message to report that it successfully joined a leg to a CS that the SLEE had requested in a prior Join_Leg message.

Merge_Call_Failure

The ASC sends this message to report that it failed in merging the legs of two CSs into one CS that the SLEE had requested in a prior Merge_Call message.

Merge_Call_Success

The ASC sends this message to report that it succeeded in merging the legs of two CSs into one CS that the SLEE had requested in a prior Merge_Call message

Move_Leg_Failure

The ASC sends this message to report that it failed in moving a leg from one CS to another, as requested by the SLEE in a prior Move_Leg message

Move_Leg_Success

The ASC sends this message to report that it succeeded in moving a leg from one CS to another, as requested by the SLEE in a prior Move_Leg message

Resource_Clear

After the action requested in the Send_to_Resource message is completed, the ASC returns any collected information, the completion status, or the reason for failure, to the SLEE in a Resource_Clear message. Information could include digits, or an announcement completed message.

Split_Leg_Failure

The ASC sends this message to report that it failed to split a leg from one CS to another that the SLEE had requested in a prior Split_Leg message

Split_Leg_Success

The ASC sends this message to report that it succeeded in splitting a leg from one CS to another that the SLEE had requested in a prior Split_Leg message

B. SLEE-to-ASC Communication

SLEE messages fall into two categories: Connection Segment (CS) related and non-CS related. *CS related* messages affect call processing. In particular, they affect real-time control of the CSs. The CS and BCSM control, and participant interaction describe these messages. There are no restrictions precluding the SLEE from requesting the ASC to move a leg of a CS past a TCP or PIC which may not be the successive state within the Basic Call State Model. *Non-CS related* messages are used for ASC interactions not involved with a CS. These messages include entity status checking and information revision messages. These messages may require a response from the ASC, but never require the ASC to suspend the processing of any events associated with a CS.

B.1 CS and BCSM control

When the ASC detects a trigger, it creates a connection view for the CS, and notifies the SLEE of the trigger's occurrence. The SLEE then, may request the ASC to modify the CSs to influence call processing.

Continue

The SLEE sends this message to acknowledge the receipt of an event message, and requests the ASC to perform normal operation. This message is sent after the SLEE determines, based on the event, that it does not need to explicitly affect call processing and that normal processing of the event be performed.

Create_Call

The SLEE sends a Create_Call message to the ASC to originate a new CS and a controlling leg at the specified user (e.g. Directory Number). The SLEE can send the message and originate the CS without having knowledge of the current status or activity for the specified user. The ASC originates a call to the user that the message specifies. The ASC provides an alerting tone to the originating DN based on a message parameter, starts a timer, and awaits answer from the originating point. The attempt fails if the timer expires before the ASC receives an answer. If the attempt to create a call fails, the ASC sends a Create_Call_Failure message to the SLEE that details the cause (e.g. time-out or busy). The ASC sends a Create_Call_Success message when the originating party answers. If the originating party answers the call, the ASC moves the controlling leg of the CS to the starting BCSM state the message specifies. The message provides destination information that corresponds to the designated call state. For e.g. if the message specifies Analysing_Info as the starting state, it provides the destination as digits.

Disconnect_S

The SLEE sends this message to the ASC to delete a leg from a CS or to delete all legs of a CS. It results in the clearing of the CSMPs associated with the corresponding legs that are to be deleted from a CS.

Forward_Call

The SLEE sends this message to instruct the ASC to forward the call to a user specified in the message. The ASC assigns a new CS and creates an association between it and the original CS. The ASC places the controlling leg of the new CS in the starting state the message specifies, and uses the destination information the message provides to set up the call. For example, the message may provide a DN and specify Analysing_Info as the starting state.

Join_Leg

The SLEE sends this message to the ASC to join a leg to a CS. The ASC connects an existing split (i.e. unconnected) leg to the designated CS. For e.g. If the user places a call on hold, it results in the splitting of a leg from the CS. A reconnect request, results in a request to join the split leg to a CS. The ASC reports the success or failure of *Join_Leg* to the SLEE with *Join_Leg_Success* and *Join_Leg_Failure* messages, respectively.

Merge_Call

The ASC can merge two CSs that refer to a single user (i.e. with a common Leg 0) into one CS if the ASC has an association between the two CS IDs. The ASC merges the two CSs by combining the legs of the two designated CSs onto the connection point of the second CS. The ASC renumbers the legs of the CSs to eliminate any duplication. The ASC reports the success or failure of *Merge_Call* with *Merge_Call_Success* or *Merge_Call_Failure* messages, respectively.

Move_Leg

The SLEE sends this message to the ASC to move a leg from one CS to another. The CSs must be associated. The ASC moves the specified leg from one CS to the second CS. The remaining legs of the first CS remain in their original states. For e.g. when a user is participating in two calls, one placed on hold, and the other active, then his request to connect to the held call and to place the active call on hold, results in a request to move a leg between two CSs. The ASC reports the success or failure of *Move_Leg* with *Move_Leg_Success* or *Move_Leg_Failure* messages, respectively.

Originate_Call

The SLEE sends this message to the ASC to originate an additional CS from a user who already has a CS and Leg 0. The ASC assigns a CS ID for the new CS and moves the active leg of the new CS to the starting state in the BCSM that the message specifies. The message also provides destination information that is related to the starting state information. If the message specifies *Null* or *Authorising_Origination_Attempt* as the starting state, it does not provide a destination. In that case the ASC proceeds from the beginning of the BCSM (e.g. providing an alerting tone and collecting digits). If it specifies *Analysing_Info* as the starting state, it provides destination information as digits(DN). This may be used to provide an Enquiry Call facility to the users.

Split_Leg

The SLEE sends this message to the ASC to split a leg from a CS. The ASC separates one or more legs from a connection point, depending on the parameters the message provides. Leg splits from the CS are in the stable state. The ASC reports the success or

failure of *Split_Leg* to the SLEE with *Split_Leg_Success* and *Split_Leg_Failure* messages, respectively

Transfer_Call

The SLEE sends this message to the ASC to split the user requesting Transfer feature from the two calls he is participating in, and to try and establish a connection between the other participants of the calls. The ASC clears the CSs associated with the two calls, and sets up a new CSs with the user associated with the second call set up as the controlling leg 0

B.2 Participant Interaction

The ASC and RCEE can provide information(e.g. announcements and tones) to a user and collect information(e.g. DTMF digits) from a user

Send_to_Resource

The SLEE uses the *Send_to_Resource* message to request the exchange of information between call participants and the network. For example, the network provides an announcement to a call participant and the call participant enters digits, which the network collects

The *Send_to_Resource* action occurs within the current call state in the ASC. The ASC cannot change states while the user is connected to the resource. A *Cancel_Resource* message can interrupt the function. The *Send_to_Resource* action can take place in any state, but is most common to the *Collecting_Information*, *Analysing_Information*, *Active*, *Authorising_Termination* and *Presenting_Call* states. Connection to a resource is not considered a separate connection point within the call model. This message only results in the ASC only connecting a single leg of the CS to the resource. The remaining legs are not affected.

The message specifies the type of resource that the ASC uses(e.g. announcements) and parameters for the resource. The *leg_id*, if specified, indicates the leg the ASC should connect to the resource. If the resource type is announcement, the parameter block includes the *resource_id* that explicitly identifies the announcement. For digit collection, the message specifies the digit template. The *Send_to_Resource* message may optionally specify that the ASC should disconnect the CS after the action is completed. Typically, this is done when an announcement is given to a call participant as terminating treatment, and the ASC is to move the CS to the Null state after the announcement is completed.

Cancel_Resource

The SLEE sends this message to the ASC to abort a connection between a user and a resource that a *Send_to_Resource* message created. Since the ASC receives this message while it is processing the interaction between the user and the resource, it sends the SLEE a *Resource_Clear* message with collected information and a status indicator, which indicates that the SLEE requested cancellation of the resource.

B.3 Entity Status Checking

The SLEE sends messages to request the ASC to perform a status check on various entities and to report the status of the identified entity back to the SLEE. These requests by the SLEE do not affect any ASC call processing on any CS. Each message sent by the SLEE contains a *response_destination* parameter that the ASC must return in the response message back to the SLEE. This parameter allows the SLEE to internally route the returned message to the SLP that made the request and allows the SLP to correlate the returned message to the appropriate entity.

Continuous_Monitor

The SLEE sends this message to request the SLEE to report all changes to the status of a specified entity. The ASC monitors the specified entity and sends the *Entity_Status* message to the SLEE each time the entity changes state from busy to idle or from idle to busy. If multiple monitor requests are made for a single entity, the ASC sends multiple responses.

Monitor_for_Change

This message requests the ASC to inform the SLEE when a specified event takes place. The ASC monitors the specified entity, and reports when the next transition to the specified state occurs. The specified state can be busy or idle.

Cancel_Monitor

When the ASC receives a *Cancel_monitor* message, a previously invoked *Continuous_Monitor* or *Monitor_for_Change* request is cancelled and no further state changes are reported to the SLEE. The *response_destination* parameter identifies the monitor request to be cancelled.

B.4 Information Revision Requests

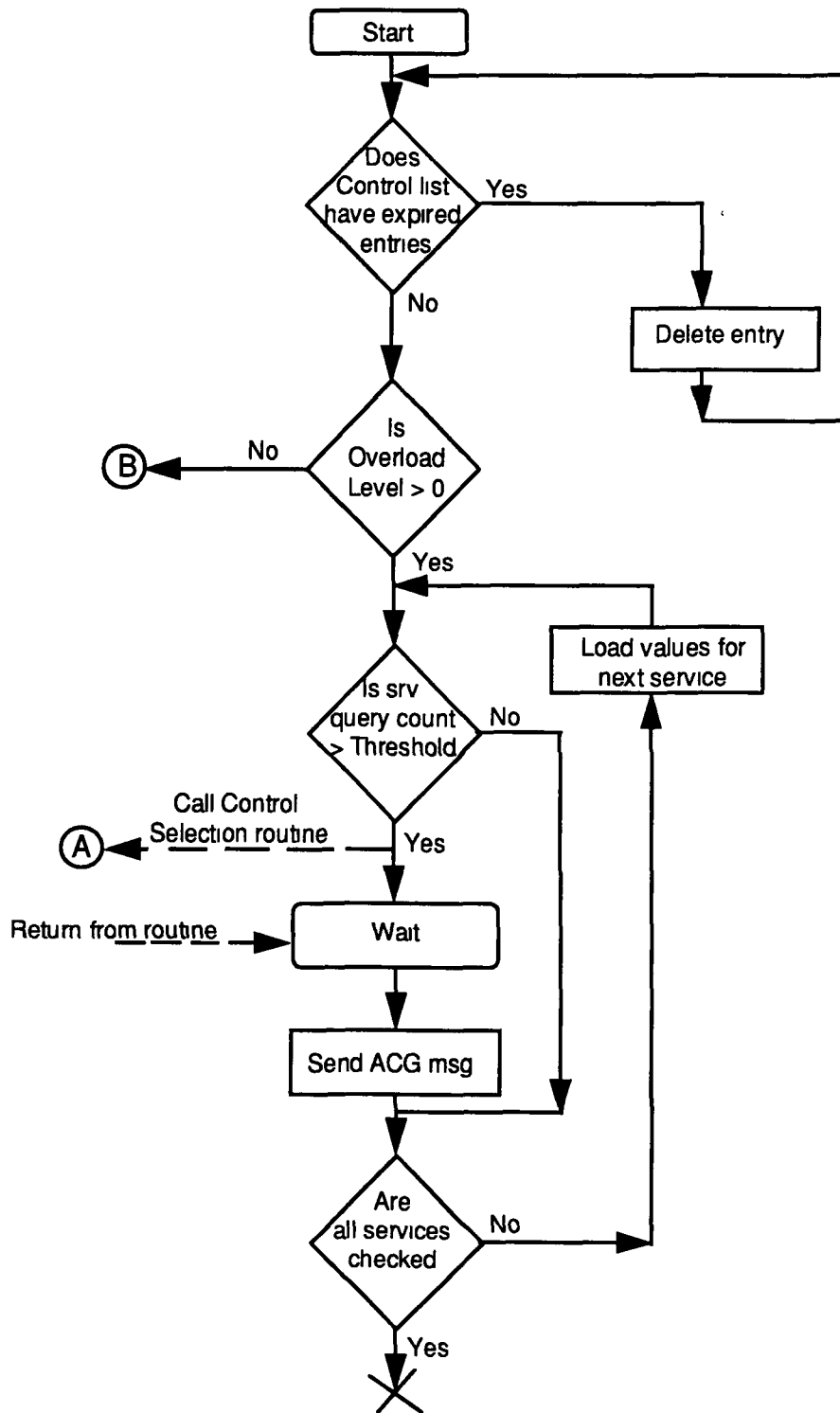
Messages described below are sent from the SLEE to the ASC to update information stored at the ASC.

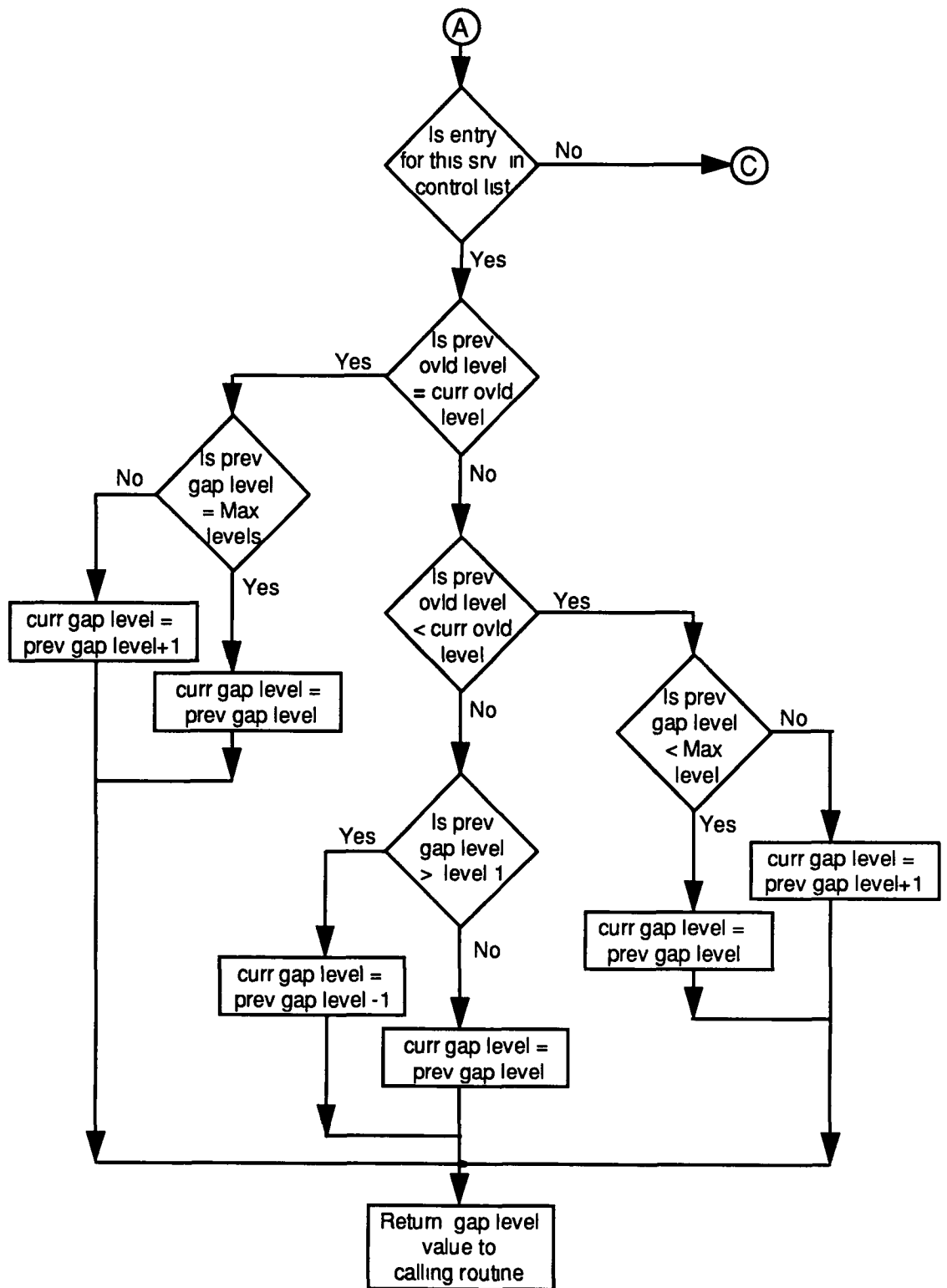
Update_Request

The SLEE sends this message to request the ASC to update information it has stored. The SLEE provides the `information_id` and new information. The data that can be changed includes Triggers activated/deactivated status flags, and flags to Turn on/off a lamp indicating call forwarding. It may be used when call forwarding is activated, to activate the `Termination_Authorised` trigger, and request the ASC to light up the call forwarding lamp at the requesting user's instrument.

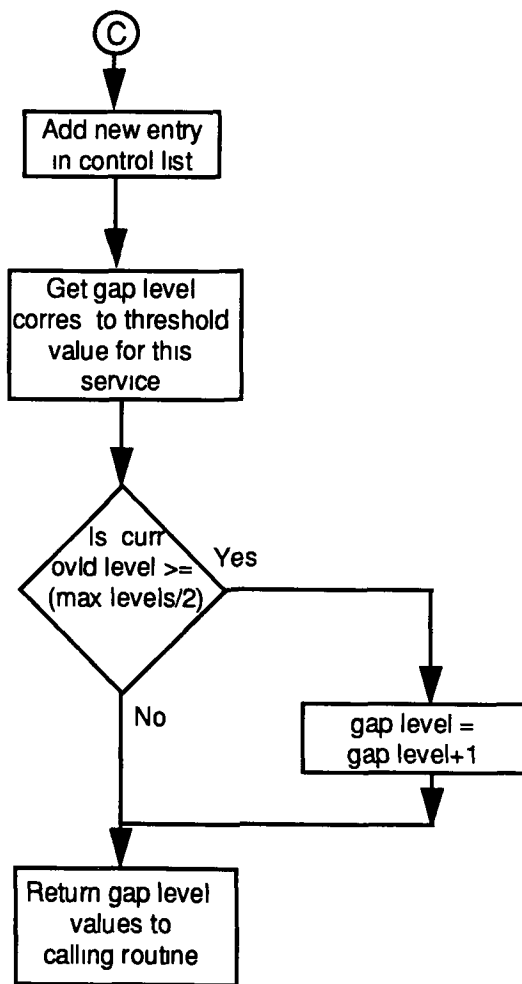
Appendix C

Flow Chart of ACG Control Selection Algorithm

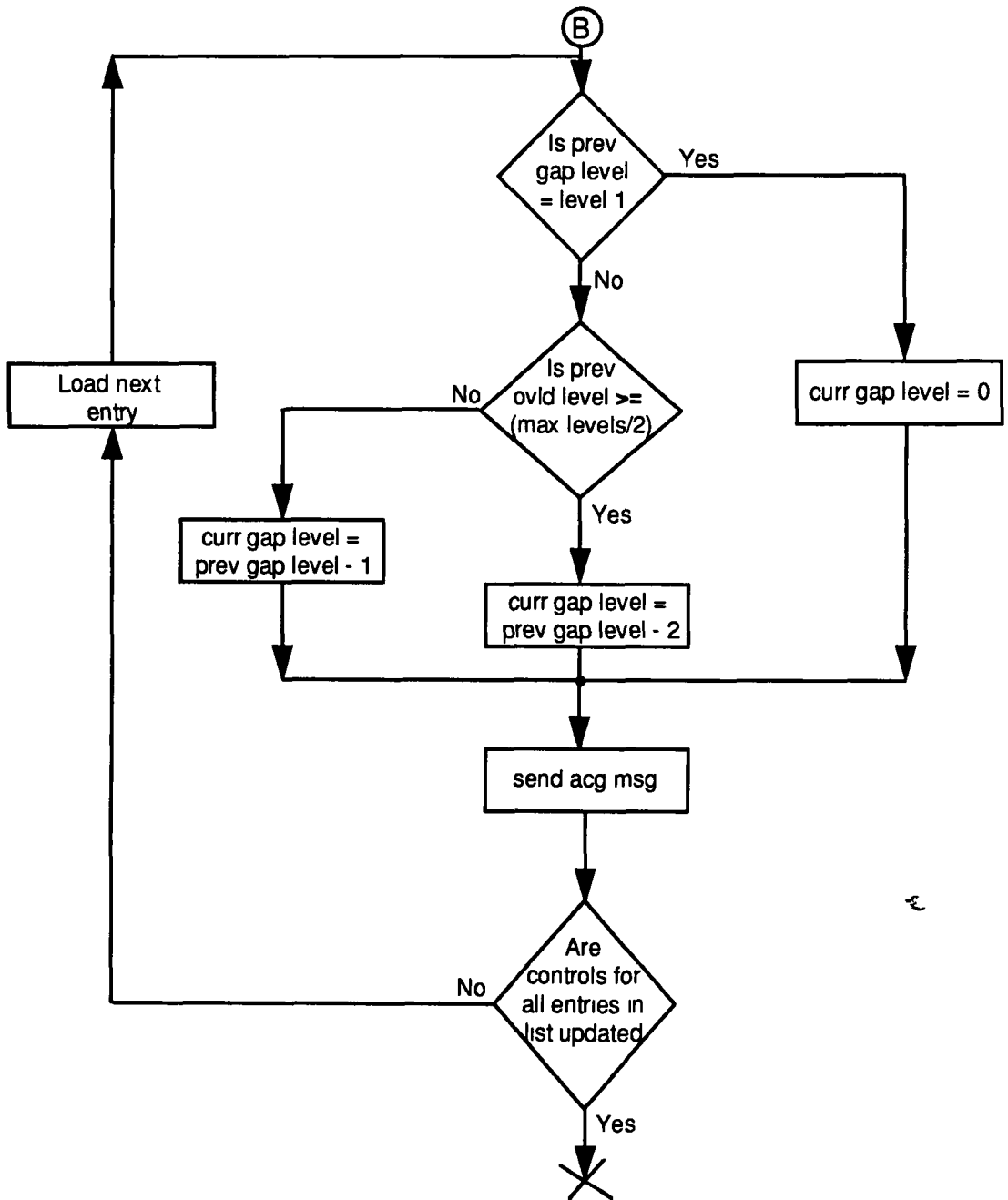




Control Selection Routine



Routine for adding a new entry



Routine for updation of control list in case overload level is detected to be zero