

INFORMATION TO USERS

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

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

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

Oversize materials (e.g., maps, drawings, charts) are reproduced by sectioning the original, beginning at the upper left-hand corner and continuing from left to right in equal sections with small overlaps. Each original is also photographed in one exposure and is included in reduced form at the back of the book.

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

UMI

A Bell & Howell Information Company
300 North Zeeb Road, Ann Arbor MI 48106-1346 USA
313/761-4700 800/521-0600

NOTE TO USERS

The original manuscript received by UMI contains pages with indistinct print. Pages were microfilmed as received.

This reproduction is the best copy available

UMI



Université d'Ottawa • University of Ottawa

Intelligent Network Service Control Point: Design, Modeling and Evaluation

by

Michel Ouellette, B.A.Sc.

A thesis submitted to the
School of Graduate and Research
in partial fulfillment of the requirements for the degree of

Master of Applied Science

in Electrical Engineering

Ottawa-Carleton Institute of Electrical and Computer Engineering
School of Information Technology and Engineering

Faculty of Engineering
University of Ottawa

April, 1998

©1998, Michel Ouellette, Ottawa, Canada



National Library
of Canada

Acquisitions and
Bibliographic Services

395 Wellington Street
Ottawa ON K1A 0N4
Canada

Bibliothèque nationale
du Canada

Acquisitions et
services bibliographiques

395, rue Wellington
Ottawa ON K1A 0N4
Canada

Your file Votre référence

Our file Notre référence

The author has granted a non-exclusive licence allowing the National Library of Canada to reproduce, loan, distribute or sell copies of this thesis in microform, paper or electronic formats.

The author retains ownership of the copyright in this thesis. Neither the thesis nor substantial extracts from it may be printed or otherwise reproduced without the author's permission.

L'auteur a accordé une licence non exclusive permettant à la Bibliothèque nationale du Canada de reproduire, prêter, distribuer ou vendre des copies de cette thèse sous la forme de microfiche/film, de reproduction sur papier ou sur format électronique.

L'auteur conserve la propriété du droit d'auteur qui protège cette thèse. Ni la thèse ni des extraits substantiels de celle-ci ne doivent être imprimés ou autrement reproduits sans son autorisation.

0-612-32549-0

Abstract

This thesis examines and suggests solutions to some of the technical problems and performance problems encountered when designing, modeling, and evaluating a Service Control Point (SCP) architecture. The SCP is known as the “*heart*” of the Intelligent Network (IN). It is essentially a transaction processor-based computer system designed to execute IN services under stringent performance requirements and Quality of Service (QoS). Expected future problems for the SCP are mainly due to the introduction and increasing complexity of new, and already deployed IN services (e.g., freephone service, calling card service, televoting, virtual private network, universal access number, etc.). The challenges in designing the SCP architecture will involve issues such as performance, evaluation of the system, quality of service definition, estimating the capacity and delays/blocking, traffic rate measurements, availability, reliability, flow control, management activities, interworking, etc.

We present and study an SCP architecture by means of queuing models and simulations. Our study differs from previous research as we consider in detail the internal SCP architecture. The proposed SCP architecture is designed as a multiprocessor-based system. It provides scalability while maintaining the requirements of the three most important performance metrics: SCP throughput, SCP response time and SCP call sampling rate. Scalability insures that IN services and SCP call sampling growth rates can be handled without impacting the performance of the SCP. The architecture also lends itself to discuss a multitude of performance and modeling issues such as service logic program storage, IN call processing activity, real-time traffic rate, main memory constraints, blocking probability, buffer dimensioning, bottleneck problems, management and administrative activities, statistical distributions, and specialized processors.

Acknowledgments

I would like to express my sincere gratitude to my supervisor, Dr. Luis Orozco-Barbosa. Thank you for giving me the opportunity to participate and cooperate in the Intelligent Network research project. I really enjoyed working under your guidance, leadership and supervision.

Many thanks to Dr. N. D. Georganas (Director of the MCRLab) and Dr. Luis Orozco-Barbosa, for procuring me with more than adequate research facilities. Dr. Orozco-Barbosa and Dr. Georganas were the principal investigators of the Intelligent Network research project.

I am also grateful to the Stentor Resource Center Inc., for given me the opportunity to study, learn and acquire knowledge in the field of Intelligent Networks.

I would like to thank my parents, François and Monique Ouellette, for providing me with full support and love throughout my studies.

Many thanks to some of my working colleagues, especially: Mr. Jeff Brinskelle (formerly the MCRLab manager), Mr. Wayne Robbins (a Ph.D. colleague of the MCRLab), Mr. Jesus Bedoy (colleague from the Center of Scientific Research of Mexico), Mr. Nik Swamy (from the Stentor Resource Center Inc.) and Dr. Ahmed Karmouch (professor at the University of Ottawa). Thank you all for your comments, help, exchange of knowledge and friendliness throughout the course of my studies. It was really appreciated.

List of Acronyms

ABS	Alternate Billing Service
ACC	Account Card Calling
AD	Adjunct
AIN	Advanced Intelligent Network
ANI	Automatic Number Identification
BCC	Basic Call Model
BCP	Basic Call Process
BCSM	Basic Call State Model
Bellcore	Bell Communication Research
CCAF	Call Control Access Function
CCC	Credit/Calling Card
CCF	Call Control Function
CCIS	Common Channel Interoffice Signaling
CCS	Common Channel Signaling
CCSS7	Common Channel Signaling System Number 7
CPU	Central Processing Unit
CP	Call Processor Station
CS	Capability Set
DB	Database
DBDB	Disk-Based Database / Disk-Based Service Database
DFP	Distributed Functional Plane
DMP	Data Management Processor
DP	Data Processor

DPT	Detection Point
DPC	Destination Point Code
DTMF	Dual Tone Multifrequency
EF	Elementary Function
FC	Functional Component
FE	Functional Entity
FEA	Functional Entity Action
FPH	Freephone
FSM	Finite State Machine
GFP	Global Functional Plane
GTT	Global Title Translation
GUI	Graphical User Interface
IDE	Integrated Drive Electronics
IF	Information Flow
IN	Intelligent Network
INCM	Intelligent Network Conceptual Model
IP	Intelligent Peripheral
ISDN	Integrated Service Digital Network
ISUP	Integrated Service Digital Network User Part
ITU	International Telecommunication Union
LAN	Local Area Network
MAC	Medium Access Control
MMDB	Main Memory Database / Main Memory Service Database
MTP	Message Transfer Part
MVI	Multi-Vendor Interaction

NCP	Network Control Point
NIU	Network Interface Unit
NTA	Near Term Architecture
OPNET	OPTimized Network Engineering Tool
P	Protocol
PE	Physical Entity
PIC	Point In Call
PIN	Personal Identification Number
POI	Point Of Initiation
POR	Point Of Return
PP	Physical Plane
PSPDN	Public Switched Packet Data Network
PSTN	Public Switched Telephone Network
QoS	Quality of Service
RBOC	Regional Bell Operating Company
RV	Random Variable
SCCP	Signaling Connection Control Part
SCEF	Service Creation Environment Function
SCF	Service Control Function
SCSI	Small Computer System Interface
SCP	Service Control Point
SDF	Service Data Function
SF	Service Feature
SG	Study Group
SIB	Service Independent Building Block

SL	Service Logic
SLP	Service Logic Program
SMAF	Service Management Access Function
SMF	Service Management Function
SMS	Service Management System
SP	Service Plane
SPC	Stored Program Control
SRF	Service Resource Function
SS7	Signaling System Number 7
SSA	Storage System Architecture
SSF	Service Switching Function
SSP	Service Switching Point
STD	State Transition Diagram
STP	Signaling Transfer Point
TCAP	Transaction Capability Application Part
VPN	Virtual Private Network

List of Figures

Figure 2.1 Intelligent Network Conceptual Model	19
Figure 2.2 Distributed Functional Plane Model.....	21
Figure 2.3 Intelligent Network Physical Architecture.....	23
Figure 2.4 Intelligent Network Service Decomposition Model.....	29
Figure 3.1 High Level Model of the Intelligent Network.....	32
Figure 3.2 Intelligent Network Service Call Flow	33
Figure 3.3 SCP Response Time Requirements.....	39
Figure 3.4 SCP Throughput Requirements	39
Figure 3.5 Geographical Deployment of SCPs	50
Figure 3.6 Traffic Rate to the Toronto's SCP.....	51
Figure 3.7 Traffic Rate Trace for the Simulations.....	53
Figure 3.8 Normal Probability Density Function	54
Figure 3.9 Exponential Probability Density Function	55
Figure 4.1 SCP Architecture for Configuration 1	60
Figure 4.2 Service Control Point Model.....	62
Figure 4.3 Token Ring Interface Model	64
Figure 4.4 Call Processor Station Model.....	64
Figure 4.5 Call Processor Model.....	67
Figure 4.6 Service Switching Point Model	69
Figure 4.7 Data Management Processor Model.....	69
Figure 4.8 Response Time Histogram for IN Call Scenario 1	74
Figure 4.9 CPU Queue Size and CPU Utilization.....	76
Figure 4.10 IN Calls Response Time	78

Figure 4.11 SCP Throughput for IN Call Scenario 1.....	78
Figure 4.12 Response Time Histogram for IN Call Scenario 2	81
Figure 4.13 CPU Queue Size and CPU Utilization.....	82
Figure 4.14 IN Calls Response Time	83
Figure 4.15 SCP Throughput for IN Call Scenario 2.....	83
Figure 4.16 Response Time Histogram for IN Call Scenario 3	85
Figure 4.17 CPU Queue Size and CPU Utilization (Buffer Size = 150).....	87
Figure 4.18 CPU Queue Size and CPU Utilization (Buffer Size = 300).....	87
Figure 4.19 IN Calls Response Time (Buffer Size = 150)	88
Figure 4.20 IN Calls Response Time (Buffer Size = 300)	88
Figure 4.21 SCP Throughput for IN Call Scenario 3.....	90
Figure 4.22 Disk-based Service Database Utilization	91
Figure 4.23 IN Call Processing and SCP Call Sampling Distribution.....	93
Figure 4.24 CPU Queue Size and CPU Utilization.....	96
Figure 4.25 IN Calls Response Time	97
Figure 4.26 SCP Throughput for Sampling Scenario 1.....	98
Figure 4.27 Sampling Disk Utilization.....	98
Figure 4.28 CPU Queue Size and CPU Utilization.....	100
Figure 4.29 IN Calls Response Time	101
Figure 4.30 SCP Throughput for Sampling Scenario 2.....	102
Figure 4.31 CPU Queue Size and CPU Utilization (Buffer Size = 150).....	103
Figure 4.32 CPU Queue Size and CPU Utilization (Buffer Size = 250).....	104
Figure 4.33 IN Calls Response Time (Buffer Size = 150)	105
Figure 4.34 IN Calls Response Time (Buffer Size = 250)	105
Figure 4.35 SCP Throughput for Sampling Scenario 3.....	106

Figure 4.36 SCP Throughput versus SCP Call Sampling Rate.....	108
Figure 4.37 SCP Response Time versus SCP Call Sampling Rate.....	109
Figure 5.1 SCP Architecture for Configuration 2.....	111
Figure 5.2 Service Control Point Model.....	112
Figure 5.3 Data Processor Model.....	114
Figure 5.4 Call Processor Model.....	115
Figure 5.5 Call Processor CPU Queue Size and CPU Utilization.....	119
Figure 5.6 IN Calls Response Time	120
Figure 5.7 SCP Throughput	121
Figure 5.8 Data Processor CPU Utilization.....	122
Figure 5.9 Data Processor Incoming Calls.....	123
Figure 5.10 Data Processor Throughput.....	124
Figure 5.11 Data Processor Throughput (Doubling CPU processing power)	125
Figure 5.12 Data Processor Throughput (Quadrupling CPU processing power).....	126
Appendix A.1 OPNET Modeler GUI Environment.....	144
Appendix A.2 Finite State Machine of an M/M/1 Queue.....	144
Appendix A.3 Partial C Code and Kernel Procedures	145
Appendix A.4 Simulation Script File.....	146
Appendix B.1 Originating Basic Call State Model.....	148
Appendix B.2 Terminating Basic Call State Model.....	149
Appendix C.1 Signaling System Number 7 Protocol Stack.....	152

List of Tables

Table 2.1 ITU-T Recommendation Framework Structure	15
Table 2.2 Intelligent Network Capability Set-1 Services	26
Table 3.1 SCP Service Database Characteristics	45
Table 3.2 IN Call Scenario Example	47
Table 4.1 IN Call Scenarios Simulated	71
Table 4.2 SCP Parameters for Configuration 1	71
Table 4.3 Simulation Results for IN Call Scenario 1	73
Table 4.4 95% Quantile Response Time for IN Call Scenario 1	75
Table 4.5 Simulation Results for IN Call Scenario 2	80
Table 4.6 95% Quantile Response Time for IN Call Scenario 2	81
Table 4.7 Simulation Results for IN Call Scenario 3	85
Table 4.8 95% Quantile Response Time for IN Call Scenario 3	86
Table 4.9 SCP Call Sampling Scenarios Simulated	93
Table 4.10 Simulation Results of Sampling Scenario 1	94
Table 4.11 95% Quantile Response Time for Sampling Scenario 1	95
Table 4.12 Simulation Results of Sampling Scenario 2	99
Table 4.13 95% Quantile Response Time for Sampling Scenario 2	99
Table 4.14 Simulation Results of Sampling Scenario 3	102
Table 4.15 95% Quantile Response Time for Sampling Scenario 3	103
Table 5.1 Additional SCP Parameters for Configuration 2	116
Table 5.2 Simulation Results for Configuration 2	117
Table 5.3 95% Quantile Response Time for Configuration 2	117

Contents

Abstract	ii
Acknowledgments	iii
List of Acronyms	iv
List of Figures	viii
List of Tables	xi
Chapter 1. Introduction.....	1
1.1 Background	1
1.2 Statement of the Challenges and Problems	4
1.3 Research Contributions.....	7
1.4 Outline of the Thesis	8
Chapter 2. Overview of the Intelligent Network	10
2.1 Introduction	10
2.1.1 Evolution of the Intelligent Network.....	11
2.2 ITU-T Intelligent Network Standards	14
2.2.1 Objectives of the ITU-T Intelligent Network	15
2.3 Intelligent Network Conceptual Model.....	17
2.3.1 Service Plane	17
2.3.2 Global Functional Plane.....	18
2.3.3 Distributed Functional Plane.....	18

2.3.4 Physical Plane.....	22
2.4 Intelligent Network Services.....	25
2.4.1 Intelligent Network Capability Set-1 Services	25
2.4.2 IN Service Decomposition.....	26
Chapter 3. SCP Modeling and Performance Issues	30
3.1 Introduction	30
3.2 System Operation.....	31
3.3 SCP Architecture, Performance and Activity Issues	34
3.3.1 SCP Throughput and SCP Response Time Requirements.....	36
3.3.2 SCP Call Sampling Activity	40
3.3.3 SCP Polling Activity	42
3.3.4 SCP House-Keeping Activity	43
3.4 Service Control Point Memory Issues.....	44
3.4.1 Main Memory-based Service Database.....	45
3.4.2 Disk-based Service Database.....	47
3.5 Real-Time Network Traffic Measurements.....	49
3.6 Statistical Distributions	53
Chapter 4. Simulation Results /1	59
4.1 Introduction	59
4.2 Configuration 1	60
4.3 First Set of Simulations.....	70
4.3.1 Simulation Results for IN Call Scenario 1	72
4.3.2 Simulation Results for IN Call Scenario 2	79
4.3.3 Simulation Results for IN Call Scenario 3	84

4.4 Second Set of Simulations	92
4.4.1 Simulation Results for Sampling Scenario 1	94
4.4.2 Simulation Results for Sampling Scenario 2	99
4.4.3 Simulation Results for Sampling Scenario 3	102
4.5 Summary	106
Chapter 5. Simulation Results /2	110
5.1 Introduction	110
5.2 Configuration 2	111
5.3 First Set of Simulations.....	116
5.3 Data Processor Modeling and Performance Issues.....	126
5.5 Summary	129
Chapter 6. Conclusions.....	131
References	136
Appendix A. OPNET Modeler	143
Appendix B. IN Basic Call State Model.....	147
Appendix C. Signaling System Number 7 Protocol	150
Appendix D. Freephone and Calling Card Service	153

Chapter 1. Introduction

1.1 Background

The world of telecommunications is evolving at a very fast pace. The explosion of information and development of new technologies are fueling a huge variety of new telecommunication requirements and customer demands. Nowadays, most of the industrialized countries benefit from an efficient *Public Switched Telephone Network* (PSTN). The PSTN [2,3,12,13] is a highly specialized network designed to support point-to-point telephony services such as voice communications. The deployment of such a network represents a huge investment in terms of equipment and man-power. Due to the telecommunication sector evolution and the large number of business and residential customers, the volume of traffic carried by the PSTN has greatly increased in the last decade. This had a major influence on the service growth rate. It turns out that this huge equipment and man-power investment is relevant for users use (i.e., for voice communications) and profitable for businesses (i.e., for revenue-generating and profits), or else a network of this size would cease to exist.

Many decades ago, when the PSTN was in its infancy stage, the primary idea was to develop switching facilities. For several decades (until the 80's) the telephone companies and equipment vendors dedicated their research and development on the physical switching network [3,53,54]. The ultimate goal was to provide an efficient and less costly connection between distant sites. This resulted in a worldwide network, which gave users the possibility to communicate with people in any other country.

During that period, little attention was given to meet future needs. A dramatic change took place in the telephone network in 1965: it was called software [4]. This introduced the first *Stored Program Control* (SPC) switch - the AT&T's No.1 ESS [1,2,5] and consequently, the widespread deployment of SPC technology in the PSTN. Soon came a family of services such as call forwarding, speed dialing, and three-way calling. The SPC also provided intelligence in order to support network management and maintenance. It was needed in order to deal with the increasing complexity of the network and services offered to the customers. The integration of the SPC technology was exclusively done by the switch vendors, since all services had to be implemented in every switch. With this approach, the service deployment was a lengthy and costly process, mostly at the expense of the telephone companies and customers. It used to take at least a couple of years, and in some cases up to five years for switch vendors to specify, implement and deploy new services. Even the smallest change to a service would affect the whole switching network, in terms of investment, system reorganization, time, and man-power. Denis Smith, an executive at US West Advanced Technologies stated that *"It takes up to five years to develop some services, at a cost of up to \$5 million for each development. With Advanced Intelligent Networks, US West hopes to reduce the time for development to six months and the cost to less than \$1 million"*.

With the years flowing by, there came a time in the mid 80's, where issues such as deregulation, increasing competition and evolving markets for new types of services raised a number of potential problems. At that time the network operators (i.e., telephone companies) bought their equipment from more than one vendor. If a new service had to be introduced, it was evident that all vendors had to implement it. From a network operator's point of view, an alternative was desirable. At the same time vendors started

to realize the risk of adding more complexity to the switches. The reasons were the following: (1) service management was becoming too much of a complex task because of the enormous amount of switches deployed in the PSTN, (2) the huge amount of software stored in the switches increased the number of potential problems, and (3) adaptive measures had to be done on old software when new software (i.e., for a new service) was introduced. To diverge from what could have been a possible catastrophe speculated by some people, a new reorganization of the PSTN had to be done. Thus, there were a number of problems requiring an immediate solution, but there was also a strong requirement on all possible solutions: the huge investment in terms of equipment and man-power had to be preserved.

During the late 80s early 90s, the need was felt to deploy services in a rapid and efficient manner in order to satisfy the customers' needs. The increasing demand for new services and the need to deploy them in short time frames were the basis for the multitude changes that the telecommunication industry experienced (and still is experiencing). The fundamental network architecture, the switching mechanisms, and its operations evolved (and are still evolving) in order to satisfy the rapid deployment, economical and social need for new services.

The telecommunication industry is still at this moment in full reorganization. The deployment of new telecommunication infrastructures will no longer exist without taking care of the imposed constraints for developing, implementing and provisioning new services. Thanks to the *Intelligent Network* (IN) concept, a potential solution to some of the discussed problems and requirements can be solved.

1.2 Statement of the Challenges and Problems

One of the most important challenge nowadays for Intelligent Networks [2,8,15,19,26,63,64,66] is to assess the performance of the *Service Control Point* (SCP) [2,21,22,35,36,56,57,58,59,60]. In the literature, the SCP is described as being the “heart” or “core” entity of the Intelligent Network [2]. Performance issues are important factors for service providers to consider when selecting an SCP platform. Performance is significant to them because INs are frequently used to operate the customers’ business. Along with customers’ satisfaction, guarantees have to be offered to the users.

General questions emanate when investigating the performance issues of the IN and its most important entity; the SCP. What should be the maximum SCP throughput? How much delay are IN calls experiencing? What is the throughput and response time when service complexity increases? Under what conditions can a bottleneck situation appear? How does management activities influence the performance of the SCP and the *Quality of Service* (QoS)? How will network interworking influence the requirements of the IN? All of these questions are involved in the challenges that today’s IN (especially the SCP) are facing. To answer these, we proceed in this thesis to examine and suggest solutions to some of the technical problems encountered in the modeling and performance evaluation of a Service Control Point architecture.

Without going into a detailed description of the SCP architecture and its modeling at this time, we make use of the following definition: a Service Control Point is typically a transaction processing system [49,55], which is primarily designed to execute IN calls/services in order to access resources, typically databases. IN services primarily rely

on SCP-resident databases in order to properly serve the IN calls. As new IN services are introduced, it is difficult to predict the customers' expectations, growth rates, complexity of services, their influence/impact on the IN, the SCP throughput and SCP response time requirements, etc¹. It is for these reasons and many more (as it will become evident in chapter 3) that performance evaluation is becoming critical and required. For example, the most popular IN service nowadays is the freephone service² [2,6,7,8,33,61]. It has a predicted annual growth close to 20 percent per year [9,10]. It is also known that the traffic rate of the freephone service to the SCP during busy business hours (i.e., during peak hours) can be in the order of one million IN calls per hour [9]. For some of today's IN services, such as freephone and calling card service, delays due to SCP queries (i.e., database queries) are in the order of tens and hundreds of milliseconds, while the SCP throughput is in the order of tens and hundreds of calls per second. But with the introduction of tomorrow's IN services and continuous growth of today's IN services, there will eventually be an increase in traffic rate, service complexity, and other non-IN service activities. The small delays and throughput mentioned above could turn out to be no longer respected. Furthermore, IN services will probably result in new expectations and requirements; the increasing amount of customers and services will eventually influence the network storage mechanisms used (i.e., main memory-based database versus disk-based database), the SCP response time, the SCP throughput, etc. For instance, we will see that the use of I/O devices, such as disk-based databases, are almost unacceptable in order to meet the SCP requirements. This is primarily due to slow disk read and write operations, unable to provide the necessary response time and throughput required for

¹ Throughout this thesis, IN calls, IN services and services are used interchangeably. Also, SCP throughput and SCP response time are interchanged with throughput and response time, respectively.

² Appendix D gives a complete detailed definition of two of the most popular IN services, the freephone service (800/888 like numbers) and account card calling service (calling card service).

the SCP. But with the advent of faster processing units, and a reduction in main memory price, the use of large main memory database systems can be adopted.

Another challenge for the evolution of the IN consists of activities (i.e., tasks) that involves the acquisition and compilation of statistics, information, and billing data. It turns out that the collection and retrieval of valuable information (e.g., call sampling data, system measurements, and billing data) are rather important for the customer's business and network maintenance as one can imagine. Access to such data by the customer, service provider and/or network operator will therefore be an ever increasing requirement in the near future. Some of these activities are actually known to be very CPU intensive, requiring lots of CPU cycles. Therefore, assessing the performance impacts on processing these activities will be evaluated in this thesis.

To answer the questions that relate to the challenges that the IN is facing, the SCP has to be modeled in such a way to respect guarantees for the customers, and a certain QoS for users. By definition, a *customer* (or service subscriber) is a person or entity who, or which, obtains a service from a service provider and is responsible for the payment of the charges due to that service provider. By definition, a *user* is a person who has access to and makes use of services. The two most important SCP performance metrics will consist on ensuring a certain level of throughput and response time from the SCP during normal operation. The performance of the SCP will be measured by its throughput in calls per second satisfying a response time constraint. For instance, the SCP must have an average response time of x seconds, with 95 percent of all calls completed in less than or equal to y seconds, where x is less than y . The x and y values

will be defined in chapter 3. They are usually obtained between a balance of the users' perception and opinion of IN services, and the actual limitation of IN capabilities.

To overcome some of the challenges and problems identified in this section, we will propose, model, simulate, and evaluate a highly modular SCP node architecture capable of answering the questions and satisfying the requirements identified in this section. This will be done throughout the following chapters of this thesis.

1.3 Research Contributions

With respect to performance and modeling issues of the Intelligent Network, this thesis adds new contributions and results to the existing literature. The contributions and results are mostly focused on the performance and modeling of the SCP entity. They are:

1. designing an overall multiprocessor SCP architecture;
2. planning and sizing the SCP;
3. considering the internal SCP architecture;
4. developing queuing models;
5. provide extensive information on the SCP performance and modeling issues;
6. investigate and analyze the SCP's capabilities;
7. investigate two critical activities of the SCP (IN call processing and IN call sampling);
8. propose a solution to SCP database system problem (main memory database)
9. propose a solution to SCP data management problem (specialized processor)

During the course of this research, three conference papers were published. They are:

M. Ouellette, J. Bedoy, L. O. Barbosa, N.D. Georganas, *Performance Evaluation of a Multiprocessor SCP*, IN'97 Workshop Record, Volume 3, Colorado Springs, CO, May 1997.

M. Ouellette, L. O. Barbosa, *Performance Evaluation of a Multiprocessor SCP for Intelligent Network Services*, Proceedings of IEEE CCECE'97, St-John's, Canada, pp. 804-807, May 1997.

M. Ouellette, L. O. Barbosa, N.D. Georganas, D. Kraychick, *Performance Evaluation of a SCP for Intelligent Network*, Proceedings of IEEE IASTED '97, Grand Bahamas Island, Bahamas, December 1997.

1.4 Outline of the Thesis

Chapter 2 presents an overview of the Intelligent Network. Important notions such as principles, characteristics, standardization and the ITU-T IN conceptual model are described. The freephone service (800/888 like numbers) is described by following the IN conceptual model.

Chapter 3 gives an introduction to the performance and modeling issues involved in the Service Control Point architecture. It consists on identifying and explaining the following: (1) a general high level model of the IN encompassing the physical entities under study, (2) the SCP potential problems, requirements and activities, (3) real-time network traffic measurements used to generate a traffic trace for our simulations, (4) main memory service database and disk-based service database characteristics and

parameterization, and (5) statistical distributions involved in the parameterization of physical entities.

Chapter 4 presents the queuing models associated with the high level model of chapter 3. The first configuration (configuration 1) is defined, and simulated in order to evaluate the performance of the SCP. It involves two sets of simulations, in which different performance aspects are investigated. Chapter 5 defines the second configuration (configuration 2), which is a solution to alleviate the problems encountered in configuration 1. It involves one set of simulations in which again different performance aspects are investigated. We also compare the results of configuration 1 and configuration 2. The performance metrics observed in both chapters include buffer size, SCP response time, SCP throughput, SCP call sampling rate, utilization, blocking probability, etc.

Chapter 6 gives a conclusion with respect to the topics investigated in this thesis.

Appendix A presents the OPNET Modeler. All simulations are conducted using the OPNET Modeler. Its primary characteristics, functions and operations are mentioned. Appendix B presents the IN basic call state model. Appendix C gives a general overview of the *Common Channel Signaling* Network (CCS) and the *Signaling System Number 7* (SS7) protocol. Appendix D gives a complete detailed definition of two of the most popular IN services, the freephone service (800/888 like numbers) and account card calling service (calling card service).

Chapter 2. Overview of the Intelligent Network

2.1 Introduction

In 1981, network intelligence reached a new plateau when AT&T introduced the use of centralized databases to support the calling card service and freephone service [2,62]. These databases were located in a family of AT&T's *Network Control Point* (NCP) systems and were accessed by SPC switches via the *Common Channel Interoffice Signaling* (CCIS) network. The centralized database approach allowed the introduction of services that would otherwise be impractical due to the large amount of SPC switches. Hence, such concept reduced the need for multiple copies of service software, and greatly simplified the management and maintenance process.

With the 1984 USA divestiture [14] of the Bell Operating Companies from AT&T, the newly formed *Regional Bell Operating Companies* (RBOC) embarked upon a program to deploy an infrastructure for new service offering. In 1985, a RBOC submitted a request for information for Feature Node concept with the following preliminary objectives [26]:

1. support the rapid introduction of new services in the network;
2. help establish equipment and interface standards to give the RBOCs the widest possible choice of vendor products;
3. create opportunities for non-RBOC service vendors to offer services that stimulate network usage.

This initiative allowed the RBOCs to be more competitive in a deregulated telecommunication environment. This Feature Node concept was later adopted by the research arm of the seven RBOCs, *Bell Communications Research* (Bellcore), and enhanced to become the first of a number of proposals. The proposals were later labeled as *Intelligent Network* (IN).

2.1.1 Evolution of the Intelligent Network

In 1985, Bellcore introduced the first proposal, IN/1 [15,18,26]. It was decided that such an infrastructure would be achieved in various steps, in a way to evolve towards a specific target. At first, the intention was to allow the network operators to introduce, control and administer services in a more efficient way. The IN/1 proposal provided a framework for the implementation of three new services: freephone, alternate billing service [26,33,39] and private numbering plan [26,33]. The most important characteristic of the proposal was the definition of both a logic and a physical architecture for service provisioning, which enabled a separation of the control logic of new services and the control logic of the basic normal telephony service (i.e., normal call processing of a voice communication). The physical architecture was based on a centralized database system, with the use of SCPs which stored the service logic. A number of nodes called *Service Switching Points* (SSP) were able to communicate with the SCP over a specific protocol, namely the SS7 protocol. Access to the SCPs, from these SSPs, was provided via the *Common Channel Signaling System Number 7³* (CCSS7).

³ Appendix C gives a general overview of the CCS network and SS7 protocol.

IN/1 aimed to support a few specific services. The idea to implement all kinds of supplementary services soon emerge. Recognizing that there was a potentially large number of services that might be offered by expanding the IN/1 functions, efforts were undertaken at Bellcore to define an architecture capable of scaling up. In 1986-1987, this resulted in a new proposal called IN/2. Its deployment was scheduled for 1991. The proposal included a set of capabilities called *Functional Components* (FC), and a system called the *Intelligent Peripheral* (IP). For example, the IP could deliver messages such as voice announcements, detect DTMF tones, etc. For different reasons (but mainly due to revenue and investment reasons) between the RBOCs and equipment vendors, IN/2 was never implemented. It was assumed the following:

1. it was not possible to update all the existing switches (which were large in number) mainly due to technical problems;
2. it would not be impossible to implement this architecture within the given time frame.

Therefore, a subset of IN/2 was defined. The proposal was labeled IN/1+, and was aimed for 1991. Its goal was to introduce service-independent capabilities to the network. Again due to performance reasons and other disadvantages, the effort to complete this proposal was stopped.

In 1989 Bellcore made a definitive attempt to define an Intelligent Network architecture. They decided to create a forum, called *Multi-Vendor Interactions* (MVI) [15,18]. This forum encompassed Bellcore, the RBOCs, and general telecommunication vendors. The purpose of the forum was to gather the best ideas from all these major organizations, and to define an architecture meeting the long-term objectives.

The results of the effort achieved by the MVI forum were labeled as *Advanced Intelligent Network*⁴ (AIN). A phased approach was proposed (as in the case of IN proposals). A first release, AIN Release 0, was planned for 1991. AIN Release 1 and AIN Release 2 were planned for the mid 90's. To avoid any incompatibilities between the different releases, a *Near Term Architecture* (NTA) was defined. However, NTA died before all reports were completed by the end of 1989. Due to business reasons and deregulation (increasing competition versus loss of revenues), network operators in joint ventures with equipment vendors implemented their own AIN Release 0, without waiting for the results of the MVI [63]. AIN Release 1 was also implemented in a joint venture by another network operator and some vendors. Therefore, Bellcore decided to base its future work on the AIN Release 1 and AIN Release 0 (i.e., on the ones already implemented). The AIN Release 0 was re-defined in a number of phases. In order to evolve from AIN Release 0 to AIN Release 1, AIN Release 0.0, AIN Release 0.1 and AIN Release 0.2 were respectively defined and expected for the year 1991, 1994, and 1995-1996. AIN Release 1 was said to be the target architecture, but no time frame was given.

Outside the US, there were no IN standardization activities prior to the 1989 creation of the MVI forum in the United States. However, during 1989 the *International Telecommunication Union* (ITU) decided to create a *Study Group* (SG XI) that would study the topic of Intelligent Networks. The study group was represented by the most important telecommunication network operators and equipment vendors of the world. Like the MVI, the ITU-T proposed to proceed in several phases, in order to evolve to the target architecture. Its mission was to define international IN standards based on specific

⁴ IN and AIN are used interchangeably in this thesis. Both technologies are based on the same concept, but differs in some capacity and technology enhancements of the network. AIN is the Bellcore definition,

Capability Sets (CS-*X*) [15,19,61,66]. The CS principle would enable the low risk introduction of a wide range of supplementary services together with rapid service delivery and customization capabilities. The ITU-T borrowed many concepts from previous Bellcore studies.

The standardization of ITU-T CS-1 [19,20,33,50] and CS-2 [51,52] are now completed. CS-1 was completed in 1993, whereas CS-2 (final version) was completed in 1996. At this time, a year after CS-2 was achieved, some network operators, equipment vendors and service providers are beginning to deploy both CS-1 and CS-2. The ITU-T is now working on capability set CS-3. No final time frame is given for the completion of CS-3.

2.2 ITU-T Intelligent Network Standards

The ITU-T term Intelligent Network is used to describe an architectural concept which is intended to be applicable to all telecommunication networks. The IN concept was defined as the idea to support rapidly increasing requests for new services and to make them globally available without altering the network conditions. The concept stated that a separation of the service control functions and the switching functions was to provide new functions.

The base ITU-T standard (i.e., recommendation) that serves as the development framework for other IN recommendations is Q.1200 - *Q-Series Intelligent Network*

whereas IN is the ITU-T International standard.

Recommendation Structure [64]. Table 2.1 lists the overall Intelligent Network Recommendation structural distribution. For example, Q.1215 defines the CS-1 physical plane, whereas Q.1232 defines the CS-3 service plane.

Table 2.1 ITU-T Recommendation Framework Structure

00 - General	01 - Principles
10 - CS-1	1 - Principles introduction
20 - CS-2	2 - Service plane (not included for CS-1)
30 - CS-3	3 - Global functional plane
40 - CS-4	4 - Distributed functional plane
50 - CS-5	5 - Physical Plane
60 - CS-6	6 - For future use
70 - CS-7	7 - For future use
80 - CS-8	8 - Interface Recommendations
90 - Glossary	9 - Intelligent Network user's guide

Throughout the following sections the IN model is described with respect to different aspects that encompasses it. A general view of the Intelligent Network capabilities is identified.

2.2.1 Objectives of the ITU-T Intelligent Network

The ITU-T IN standardization activity was motivated by the interests of network operators and service providers to rapidly and cost effectively satisfy their existing and potential market needs for services. Also, they seek to improve the QoS and reduce the cost of network service operations and management. In order to meet these requirements, some overall objectives were defined. They are:

1. increase service velocity to enable the market-driven, rapid introduction of new services, from conception to deployment;

2. 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;
3. enable a multi-vendor, competitive environment to ensure that the services will work correctly and consistently on any vendor's equipment, and across several vendors' equipment;
4. evolve from existing networks; the deployable technology must interwork with and evolve from existing networks since these cannot be replaced overnight.

In order to address the four points, an IN architectural concept was defined. It is explained in Q.1201 recommendation - *Principles of Intelligent Network Architecture* [65] and is characterized by:

1. an extensive use of information processing techniques;
2. efficient use of network resources;
3. modularization and reusability of network functions;
4. integrated service creations and implementation by means of the modularized reusable network functions;
5. flexible allocation of network functions to physical entities;
6. portability of network functions among physical entities;
7. standardized communication between network functions via service independent interfaces;
8. service subscriber (customer) control of some subscriber-specific service attributes;
9. service user control of some user-specific service attributes;
10. standardized management of service logic.

2.3 Intelligent Network Conceptual Model

In order to design the IN architectural concept, the *Intelligent Network Conceptual Model* (INCM) was defined [15,19,65]. According to Q.1201, the INCM provides a framework for the description of IN, its concepts and relations. The INCM consists of four planes, which represent different abstractions of the telecommunication network. The INCM planes are: *Service Plane* (SP), *Global Functional Plane* (GFP), *Distributed Functional Plane* (DFP) and the *Physical Plane* (PP). The INCM is shown in Figure 2.1. The *Service Plane* gives a service-oriented view, whereas the *Global Functional Plane* is concerned with the service creation and service logic aspects. The *Distributed Functional Plane* concerns the set of functional entities and functional entities actions for service creation and service logic, and the *Physical Plane* relates to the physical aspects of the entities present in the IN.

2.3.1 Service Plane

The SP gives a service oriented perspective of the Intelligent Network. The service implementation and all underlying details are completely masked within this plane. Only the perceptible service operations and mechanisms are made visible to the users. For example, the user can be aware of call queuing, dial tones, busy tones, voice announcements, audible ring, operator assistance, etc. Basically, these represent the high-level picture of the users' perception of the service. The main concepts of the service plane are *Services* and *Service Features* (SF). Each IN service is composed of one or many SFs, which constitutes the lowest level of an IN service.

2.3.2 Global Functional Plane

The GFP represents the IN as a single/whole entity, and deals with how the IN services are created. The most important concepts are the *Service Independent Building Blocks* (SIB) and *Service Logic* (SL). The SIBs are standard reusable software components (building blocks), used to create IN service. SL defines the composition of single/multiple SIBs that creates a service. One of the specific SIB is the *Basic Call Process* (BCP), which corresponds to the basic normal telephony service (i.e., normal call processing of a typical voice communication between two end-users). Inputs and outputs of the BCP are *Points of Initiation* (POI) and *Points of Return* (POR), respectively. They are used to access the IN capabilities for IN service related instructions.

2.3.3 Distributed Functional Plane

The DFP provides a distributed view of the functions scattered throughout the IN. *Functional Entities* (FE) contain a specific group of functions necessary to provide a service. A FE performs a number of *Functional Entity Actions* (FEA). The GFP SIBs are realized as a series of chained FEA. Functional entities do not include utility/house-keeping functions that are not directly involved in providing an IN service. Figure 2.2 identifies the IN DFP model.

The IN DFP model describes the functional architecture of the IN. The plane reveals how the service architecture functions are distributed throughout the network's physical entities (the physical entities are described in the next section). The DFP is the

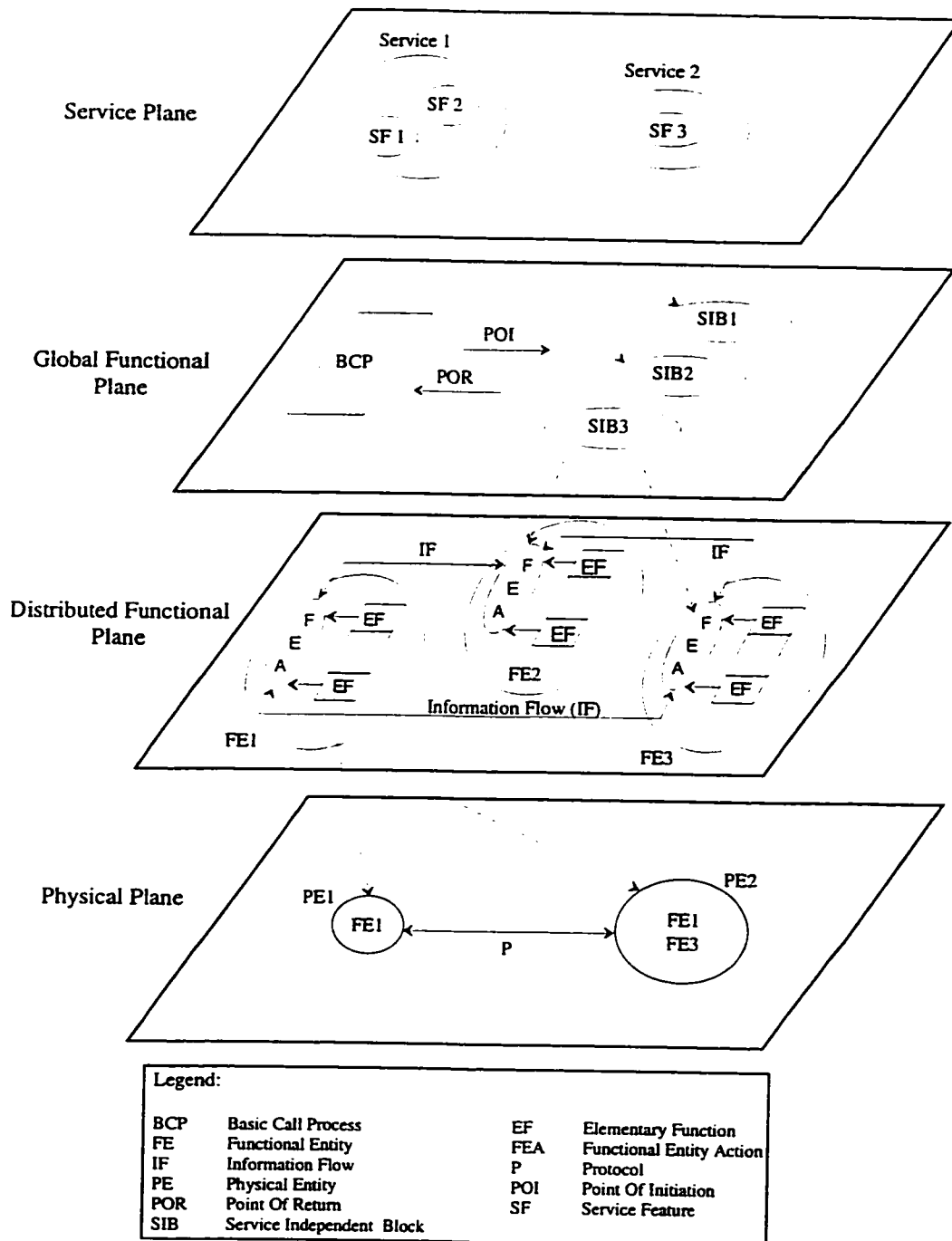


Figure 2.1 Intelligent Network Conceptual Model

level that distinguishes the IN, since it is at this level that the relationships between the service logic and call processing can be recognized. The functional entities shown in Figure 2.2 includes the following:

The *Call Control Function* (CCF) provides the normal call processing for establishing and controlling bearer services (i.e., non IN-services). It refers to the call and connection handling of the basic normal telephony service (i.e., voice communication). The *Call Control Agent Function* (CCAF) represents the interface between the user and the call control functions of the network. It allows the users to access IN capabilities. The *Service Switching Function* (SSF) is a function that recognize calls requiring “*special processing*”, that is to say, calls requiring IN service processing. The recognition is done through the use of a special control triggering mechanisms. Therefore, the SSF provides the interaction between the basic call processing and the IN service processing⁵.

The *Service Control Function* (SCF) contains the appropriate *Service Logic Programs* (SLP). It provides the proper logic and IN processing capabilities in order to execute IN calls/services. Such processing capabilities are: analysis, translation, screening, routing, authentication etc. The SCF interfaces and interacts with the SSF, SDF, and SRF.

The *Specialized Resource Function* (SRF) concerns the provision of special resources such as DTMF receivers, voice recognition, voice announcements, speech processing, etc. The *Service Data Function* (SDF) contains customer and network data

⁵ Appendix B presents the Basic Call State Model (BCSM) (i.e., the originating and terminating call state model that allow the interaction between basic call processing and IN service processing.

for which the SCF entity accesses in real time for the execution of an IN service. Furthermore, user and network data could be: authentication data, customer profile, translation tables, billing reports, etc.

The *Service Management Function* (SMF) provides service provisioning, service deployment and service management control in the IN. It allows for the retrieval and access of information related to service logic, service data, billing and statistic information that could be collected in the IN. The *Service Management Access Function* (SMAF) provides a man-machine interface for the operation, management and administration of the IN. The *Service Creation Environment Function* (SCEF) allows the definition, development, and testing of new IN services. The output of this function includes service logic, service management logic, service data templates and service trigger information. These are then provisioned and deployed throughout the IN.

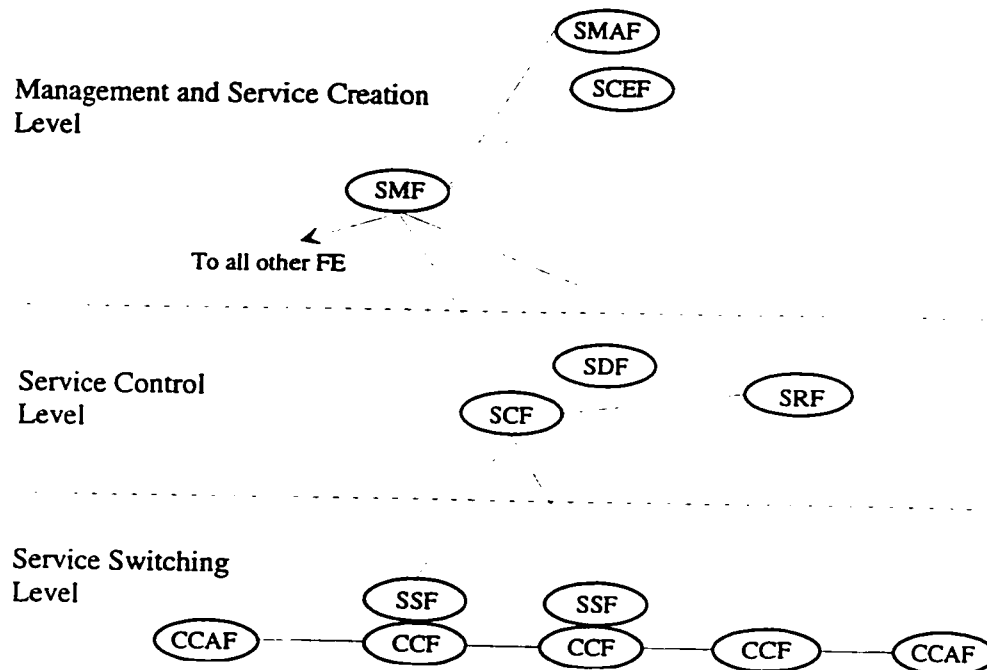


Figure 2.2 Distributed Functional Plane Model

2.3.4 Physical Plane

The PP represents the physical aspects of the IN-structured network. It defines the *Physical Entity* (PE) and the *Protocols* (P) by which all PEs communicate. It also identifies the mapping of the functional entities (discussed in the last section) on the PEs. Figure 2.3 depicts the IN physical architecture. A physical IN-structured network (e.g., Figure 2.3) will consist of some or all of the following physical entities:

The *Service Switching Point* (SSP) provides network access for the end-users. It provides switching functions and allows access to the set of IN capabilities. It contains the triggering mechanisms to detect IN services. It also contains other capabilities to communication with other PEs identified in the figure.

The *Service Control Point* (SCP) provides and contains the SCF. The SCP is often described as the heart of the IN. It contains the corresponding SLPs used to provide IN services, and may optionally contain customer data. It is also the key physical entity for the implementation and execution of service logic programs. It interacts with the SSP via the signaling network.

The *Service Management Point* (SMP) performs service management, service provisioning, and service deployment. Examples of functions it can perform are database administration, network surveillance and testing, network traffic management, and network data collection. The *Service Management Access Point* (SMAP) provides an access point for services managers and customers to access the SMP.

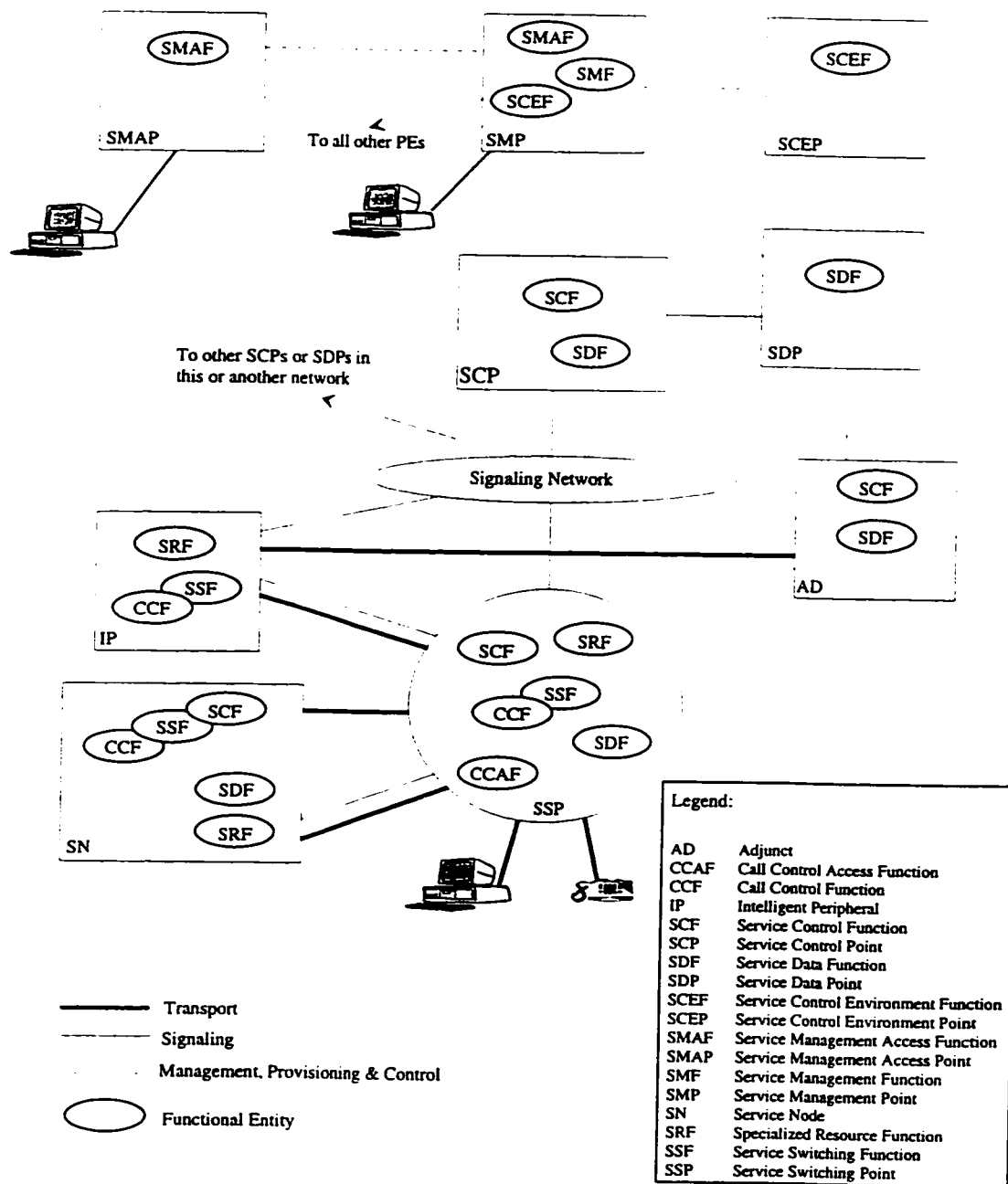


Figure 2.3 Intelligent Network Physical Architecture

The *Service Creation Environment Point* (SCEP) provides the capabilities to define, develop, and test an IN service.

The *Service Data Point* (SDP) contains specific data (e.g., customer data) used by the SLPs to provide individualized services.

The *Intelligent Peripheral* (IP) provides special resources for the customization of services. Such resources are: voice announcements, speech recognition devices, DTMF digit collection, tone generators, audio conference bridge, text to speech synthesis, protocol converters, etc.

The *Adjunct* (AD) is functionally equivalent to an SCP. The difference is that it is directly connected to the SSP by the use of a high speed interface. The signaling network is therefore irrelevant in this case.

The *Service Node* (SN) can control IN services and can engage in flexible information interaction with users. Similar to an adjunct and SCP, the SCF in a SN receives messages from the SSP, executes SLPs, and sends response back to the SSP. The SRF in a SN enables it to interact with users in a manner similar to an IP.

The possible mapping of the FEs to the PEs in an IN-structured network should follow the ITU-T recommendations. In the ITU-T recommendations, some mapping of FEs on PEs are mandatory, whereas some are optional. This was not shown in Figure 2.3.

2.4 Intelligent Network Services

Intelligent Network Capability Set-1 was the first standardized set of recommendations defining the IN as an architectural concept for the creation and provision of telecommunication services. It states that the IN architecture can be applied to various types of telecommunication networks, which include: PSTN, *Public Switched Packet Data Network* (PSPDN), mobile networks, and *Narrowband/Broadband Integrated Service Digital Network* (N- and B-ISDN). In the following sections a general description of the CS-1 service capabilities is given.

2.4.1 Intelligent Network Capability Set-1 Services

As stated in Q.1211 [33], a service is “*a stand-alone commercial offering, characterized by one or more core service features, and can be optionally enhanced by other service features*”, whereas a service feature is “*a specific aspect of a service that can also be used in conjunction with other services/service features as part of a commercial offering. It is either a core or part of a service or an optional part offered as an enhancement of a service*”.

Table 2.2 lists the services supported by, and defined in CS-1. It consists of twenty-five services, ranging from simple services (e.g., call forwarding) to complex services (e.g., universal personal telecommunications). The service features are not shown due to the large amount. Refer to [20,33] for a complete list of SFs.

Table 2.2 Intelligent Network Capability Set-1 Services

Abbreviated dialing (ABD)	Account card calling (ACC)
Automatic alternative billing (AAB)	Call distribution (CD)
Call forwarding (CF)	Call rerouting distribution (CRD)
Completion of call to busy subscriber (CCBS)	Conference calling (CON)
Credit card calling (CCC)	Destination call routing (DCR)
Follow-me diversion (FMD)	Freephone (FPH)
Malicious call identification (MCI)	Mass calling (MAS)
Originating call screening (OCS)	Premium rate (PRM)
Security screening (SEC)	Selective call forward on busy/don't answer (SCF)
Split charging (SPL)	Televoting (VOT)
Terminating call screening (TCS)	Universal access number (UAN)
Universal personal telecommunications (UPT)	User-defined routing (UDR)
Virtual private network (VPN)	

2.4.2 IN Service Decomposition

A hierarchical decomposition of IN services, with respect to the IN conceptual model is presented in Figure 2.4. Each IN service identified in Table 2.2 can be decomposed in a similar fashion. General comments are now given with respect to the freephone service, the IN conceptual model, and service decomposition. As this moment, the freephone service is the most popular of all IN services, generating huge revenues and profit for network operators, service providers, and customers. The freephone service will be used in the next chapters for modeling and evaluating the performance of the proposed SCP.

According to Q.1211 recommendation, here are the following detailed definitions of the freephone service.

- **Description No.1**

This service allows a reverse charging, the subscriber accepting to receive calls at his expenses and being charged for the whole cost of the call.

- Description No.2

Freephone allows the served user having one or several installations to be reached from all or part of the country, or internationally as appropriate, with a freephone number and to be charged for this kind of call.

The freephone service makes use of several service features, as identified in the service plane of Figure 2.4. With respect to what is mentioned in the recommendation, the freephone service uses some *core service features* and some *optional service features*. The core service features used for the freephone service are: ONE (One Number) and REVC (Reverse Charging).

The optional services features for the freephone service are: AUTZ (Authorization Code), CD (Call Distribution) CFC (Call forwarding on busy/don't answer), GAP (Call Gapping), LIM (Call Limiter), LOG (Call Logging), QUE (Call Queuing), CPM (Customer Profile Management), CRA (Customized Recorded Announcement), CRG (Customized Ringing), DUP (Destinating User Prompter), MAS (Mass Calling), ODR, (Origin Dependent Routing) OCS (Originating Call Screening), OUP (Originating User Prompter), TDR (Time Dependent Routing). According to the recommendation, the definition of the two core service features used in the freephone service is:

- ONE (One Number):

This feature allows a subscriber with two or more terminating lines in any number of locations to have a single telephone number. This allows businesses to advertise just one telephone number throughout their market

area and to maintain their operations in different locations to maximize efficiency. The subscriber can specify which calls are to be terminated on which terminating lines based on the area the calls originate.

- REVC (Reverse Charging)

This service feature allows for the separation of charges for a specific call, the calling and called party each being charged for one part of the call.

In order to implement the freephone service, the service features (core and optional) are mapped to some particular SIBs of the GFP. For instance, *service feature A* of the freephone service is mapped onto the *User Integration SIB* of the GFP (recall that a service/service feature is created by stitching a combination of one or more SIBs together). This is partly shown in Figure 2.4. As explained in section 2.3.3, the SIBs are realized by FEAs, which occur within one or more FE (e.g., SSF, SCF, SDF, etc.). For instance, the TRANSLATE SIB shown in Figure 2.4 requires more than one FE. In such case, when more than one FE is required, information flows (IF) results between FEAs.

CS-1 defines a total of fourteen SIBs used to create services and service features. The most important SIB for the freephone service is now given. It is the TRANSLATE SIB. Its definition is given as follows:

- TRANSLATE SIB:

It determines output information from input information. This SIB translates input information and provides output information, based on the various other input parameters. For example, in the case of a freephone

service this SIB could be used for modifying input information (for instance the dialed digits, time of day and calling party identification) into a destination directory number (physical number of the numbering plan).

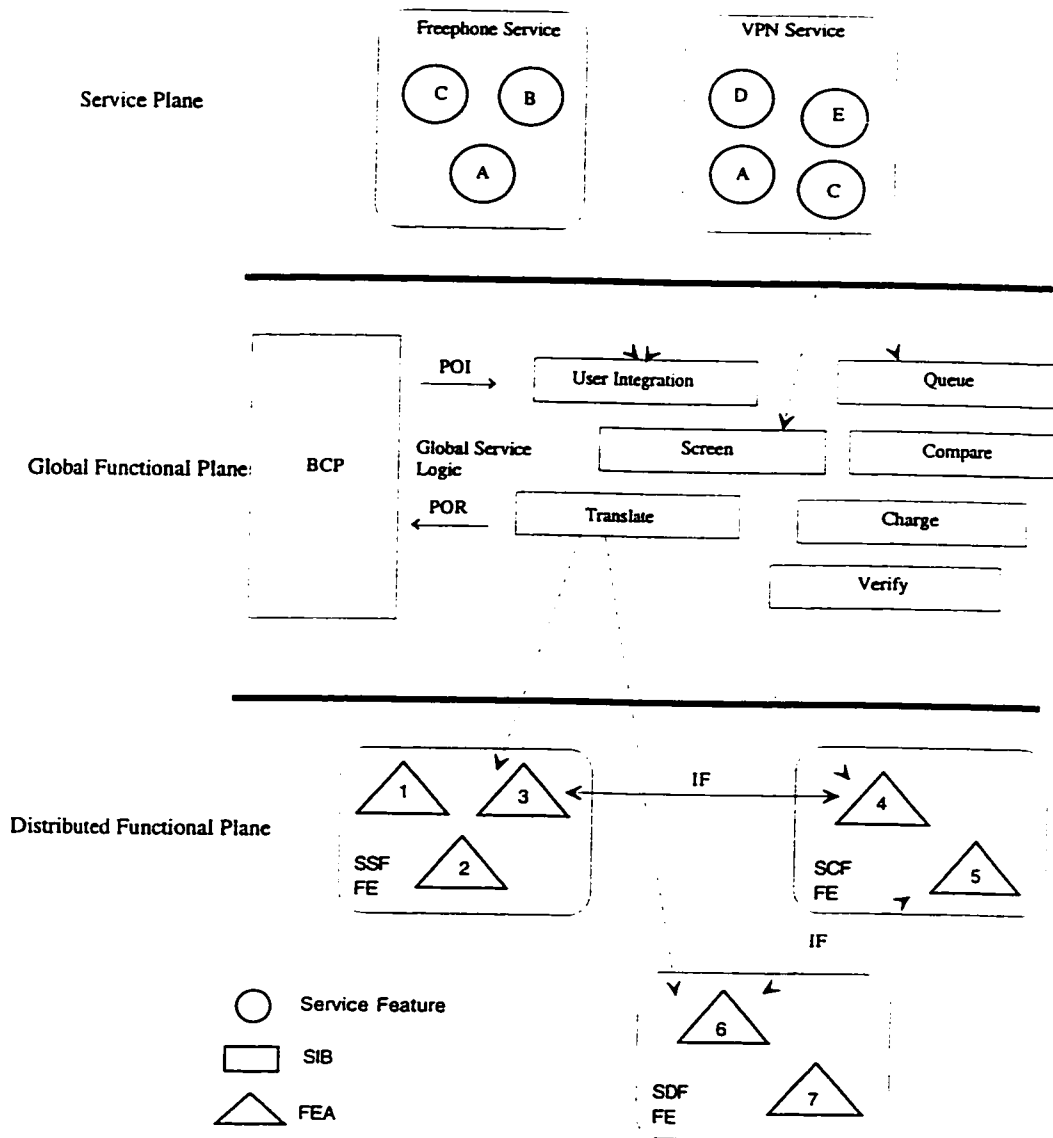


Figure 2.4 Intelligent Network Service Decomposition Model

Chapter 3. SCP Modeling and Performance Issues

3.1 Introduction

This chapter provides the details and insights needed for the performance and modeling issues of this research. A high level model of the Intelligent Network and its operations is given in the following section. The model presents a general view of the IN and the physical entities under study. The model extracts, from the mass of details, the aspects that are essential to the performance evaluation and modeling of the SCP. In chapter 4 and 5 the high level model will be decomposed into a series of low level models. The low level models (i.e., queuing models) will identify the internal architecture and operations of each physical entity, which encompasses the mass of details left out from the high level model. Figure 3.1 depicts the high level model of the proposed Intelligent Network. The model is primarily composed of the SCP, which is the core physical entity of the IN.

The most important entity under investigation is the SCP. The proposed SCP is modeled as a multiprocessor-based architecture [2,9,23,34,35,36,37,62]. It is composed of a set of processors linked together by a local area network [38]. Two different types of processors can be distinguished in Figure 3.1. The first type of processors are strictly specialized in the *IN call processing activity*⁶. These processors are known as *Call Processor stations* (CP). The second type is a group of specialized processors which takes care of management activities (i.e., call sampling, billing, statistics compilation, network measurements, etc.). They are not intended to participate in the IN call processing activity. These processors are known as the *Data Processor* (DP), *Data Management Processor* (DMP) and Billing Center.

⁶ The IN Call Processing Activity will be the primary activity under investigation in this thesis.

3.2 System Operation

This section explains the overall system operation with respect to the model depicted in Figure 3.1. In addition to this high level model, Figure 3.2 illustrates a simple IN service call flow. The IN service call flow will be the general call model used throughout our simulations and evaluation. It represents a timing diagram of the sequenced messages involved in the processing and execution of an IN service [39,48,68]. The overall system works as follows:

Regular phone calls that do not require any IN capabilities are solely handled by the SSPs. The SSPs [2,26,40,68], which are part of the PSTN, are in charge of performing "normal" call processing (i.e., regular phone calls) and to carry the traffic (i.e., voice communications). Normal calls are essentially usual day to day voice calls generated by PSTN users. Upon detecting an IN call that requires IN capabilities, an SSP issues a message *Service Request* (see Figure 3.2) to the SCP. Usually, the SCPs are physically located in a central office, or in a particular building where all the IN intelligence resides. The request is routed through the CCSS7 network packet switches, namely the STPs. Upon receiving the request, one of the call processor station within the SCP proceeds to execute the IN call. The CP station validates the request and determines if more user information is needed (e.g., PIN numbers and/or voice inputs). After the validation process is completed the CP station will rely on its database to retrieve and execute the *Service Logic Program*⁷ (SLP) associated to that service. This is shown by the DB

⁷ A SLP is defined as a program which is executable by an SCP in correspondance to each service specification.

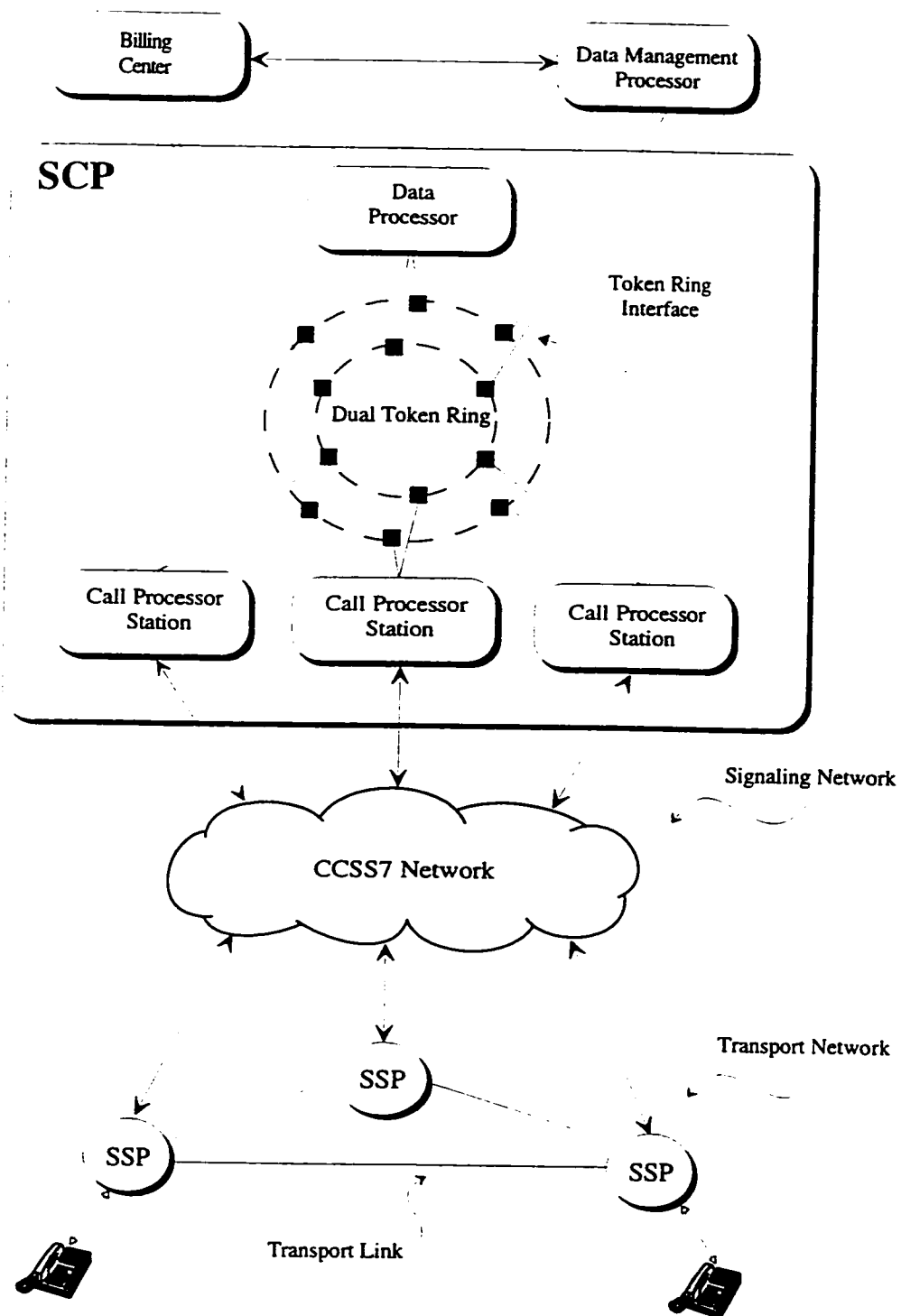


Figure 3.1 High Level Model of the Intelligent Network

Requirements message in Figure 3.2. For instance, an SLP and its data could contain the telephone number translation, the carrier to which the call is to be routed, and number to be charged, in the case of a freephone service. When the SCP (one of the call processor station) completes the call processing, it issues a *Response* message to the SSP with the proper information on servicing the IN service request. All these operations are part of the *IN call processing activity*.

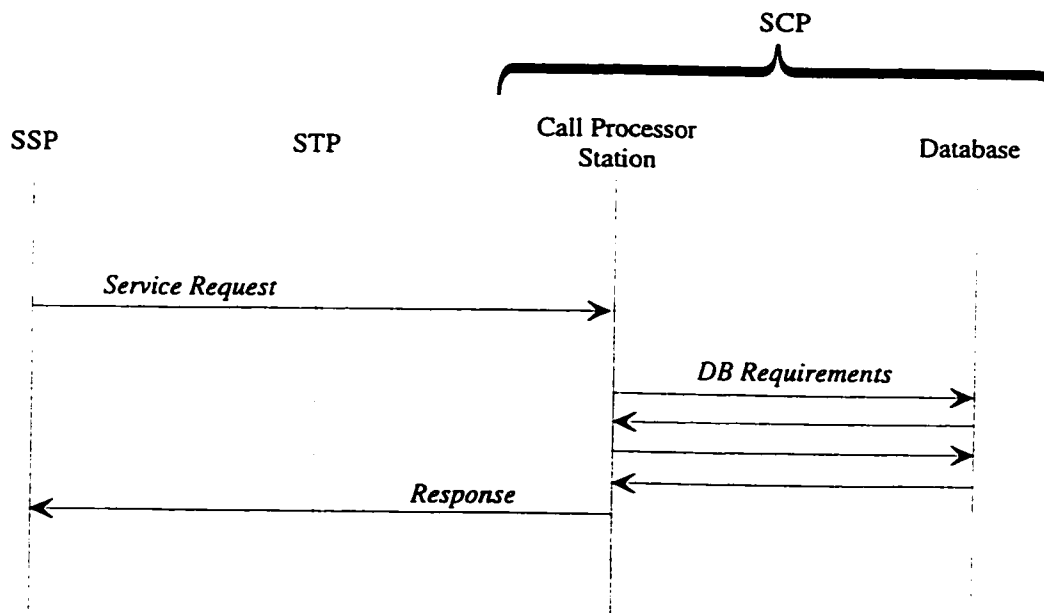


Figure 3.2 Intelligent Network Service Call Flow

For a certain number of IN calls, the CP stations will need to collect and store some sampling/statistical information on them. This activity will be known as the *SCP call sampling activity*. The action of collecting the sampling/statistical information is done during the IN call processing activity, and the name associated to the collected data is known as an *IN data record*. This will be described in a following section. Depending on the type of information collected and the retention period, the IN data records are later

transferred over the communication local area network (i.e., over the token ring topology of Figure 3.1) to an external administrative system. This activity will be known as the *Polling activity*. The external administrative system is one of the specialized processors identified earlier. More information will be given in a following section.

3.3 SCP Architecture, Performance and Activity Issues

The SCP is a real-time, highly available centralized database system hosting the service logic programs to process and execute IN services [2]. The SCP architecture depicted in the last section is modeled as a multiprocessor-based system. One of the significant issues surrounding this type of architecture relates to the ability of separating parallel processes in order to increase the overall SCP performance. This architecture provides an opportunity to utilize the capacity growth benefits of a distributed architecture. For example, it allows many redundant CP stations to be provisioned as needed, in order to add future additional processing capacity. That is, the SCP is designed so that it can be tailored to meet the needs for the current IN and to allow continuous upgrades along the IN evolutionary step for system expansion. Therefore, the SCP should be versatile and flexible enough to satisfy the requirements stemming from the IN technology.

An additional requirement is that the SCP must meet established availability and reliability requirements regardless of the architecture implementation [2,15,36,69]. For instance, in the case of our multiprocessor-based SCP model, a failure of one of the call processor stations should not cause the entire system to fail. Moreover, an arrangement of multiple processors will only degrade the performance instead of causing a complete

failure when one call processor station fails [34]. Even with multiple processors, the SCPs are usually deployed in a mated pair fashion that run in a load sharing configuration, in order to increase the availability in case of failure situations. With such architecture, performance is not only required during normal situations but also ensured during failure conditions and/or periods of overload (i.e., to protect the SCP in excessive calling scenarios).

Because of its role, the SCP must be designed in order to meet stringent telephony requirements. As a primary requirement the SCP must provide fast and highly reliable access to, and processing of IN services. *Sufficient SCP call processing*, *SCP response time constraints* and *SCP storage capacity* are considered as three main characteristics for the SCP. We briefly explain these three characteristics.

1. Because of the enormous number of PSTN users, sufficient SCP call processing is critical, in order to properly serve the IN calls, minimize the number of retries and reduce the blocking probability. Just imagine a case where a large number of users try accessing at the same time, an IN service such as the mass calling [33] or televoting service [33].
2. The SCP response time is a performance criterion critical to user's perception and opinion of telephony services. This cannot be ignored since it contributes to the generation of revenue for customers, service providers and network operators.
3. SCP storage capacity is also mandatory. It is mandatory because of the large number of customer's SLPs that must be stored in the SCP. The architecture must take into

account the service growth and service evolution for which future services could influence the storage mechanisms (in terms of storage capacity). For example, in 1990 there were around four million 800 numbers deployed throughout the U.S. IN [9]. This number still increases at a very fast pace, and it is projected that it will increase by more than 15% each year [9,10]. This could mean more than fifteen million 800 numbers by the end of the century.

Due to the emerging demand for service evolution and constant growth of businesses and residential communities, the SCP must therefore quickly respond to the market requirements. The SCP must also adhere to new QoS requirements, performance requirements, and future expectations. The remaining sections addresses these.

3.3.1 SCP Throughput and SCP Response Time Requirements

Performance demands will become more critical in the future as traffic rate to the SCP increases. Along the way, new and old services will place additional expectations and requirements on the performance of the SCP. In order to meet these, the SCP needs to be designed in a way to respect some pre-defined requirements/guarantees (i.e., a certain QoS). These requirements are the basis to provide customer and user satisfaction. They are explained next.

A certain level of throughput and response time requirements are required from the SCP during normal operation. The performance of the SCP will be measured by the throughput (denoted by the number of transactions/calls per second) it can deliver under a response time constraint. The response time will be defined as the time when the receipt

of an inbound request message for an IN service enters the SCP (at the CCS signaling network - SCP interface) to the time when a response message is delivered back to the signaling network. This latter quantity will be known as the SCP response time. SCP response time is a predominant factor in assessing service quality. In [39] they predict, with the use of a model, that for short duration calls, an increase in response time will result in a decrease of the percentage of “*Good or Excellent*” ratings of the service. They also state that premature abandonment by a user tends to increase as response time increases. [41] also presents a method so that administrators can be more responsive to customer requirements for new novel services.

The proposed SCP architecture must meet the following requirement. The SCP must have an average response time of 0.5 seconds, with 95% of all calls completed in less than or equal to 1.0 second, with respect to freephone service, alternate billing service, and credit/calling card service. Basically, this states that 95% of the time, the IN call will be processed under 1.0 second. The choice for these values was proposed in a joint research project between IBM, Siemens and Bell Atlantic [26] and in [35]. These values were calculated in such a way to reflect the balance equation between *users' satisfaction* (assessment of the IN service) and *IN capabilities* (allowed by the present technologies). In [39] they proposed a model to estimate response time. The model took into account new characteristics and new call patterns introduced by IN services. The model can be used to analyze the services from both the user's and the network's perspectives. To date, much effort has been concentrated on the second side of the balance equation (i.e., IN capabilities allowed by present technologies and network's perspectives). To achieve a good balance, objective studies will have to be conducted in

order to characterize the performance of current IN services, whereas subjective studies will be needed to gain insights into users' behavior with respect to IN services.

Figure 3.3 depicts the SCP response time of the objective study conducted by IBM, Siemens and Bell Atlantic. It presents the response for four IN services. The first three services actually have the same service complexity in terms of SLP size and processing time, hence the same response time values. The average response time and the maximum response time (i.e., 95% of all IN calls processed below 1.0 second) is given for the *freephone* (FPH), *Alternate Billing Service* (ABS) [62] and *Credit/Calling Card* (CCC) service. Another service, *Virtual Private Network* (VPN) [33], is also presented. The VPN service complexity is significantly higher than the other three services, resulting in an increase of the average and maximum response time. The response time values depicted in the figure are now being adopted by most of the telecommunication vendors. Recall that all four services mentioned are supported by the ITU-T CS-1.

The SCP's throughput, qualified by the associated SCP response time constraint identified earlier, is usually called the SCP maximum qualified throughput. The SCP throughput of our model shall be between 800 and 900 transactions/calls per second for services equivalent to freephone service and calling card service. The choice of this value was taken from a research study conducted in [9], and [21]. The study in [9] presented a methodology to estimate the potential impact of IN services on SCP transaction/call rates). From public network traffic statistics, they projected the SCP transaction/call rates for a ten year period. In order to project such rates, the methodology relied upon estimates for Average Busy Season, High Day, and Busy Hour variances since IN services are dependent upon actual user calling patterns. Figure 3.4 depicts the estimated

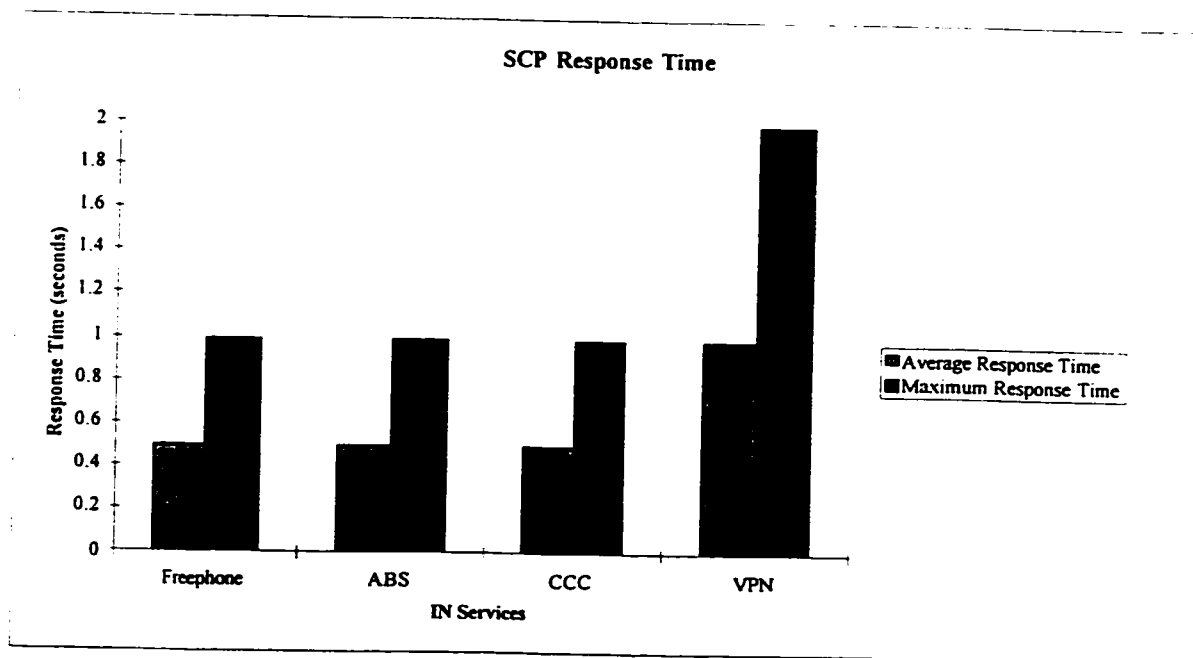


Figure 3.3 SCP Response Time Requirements

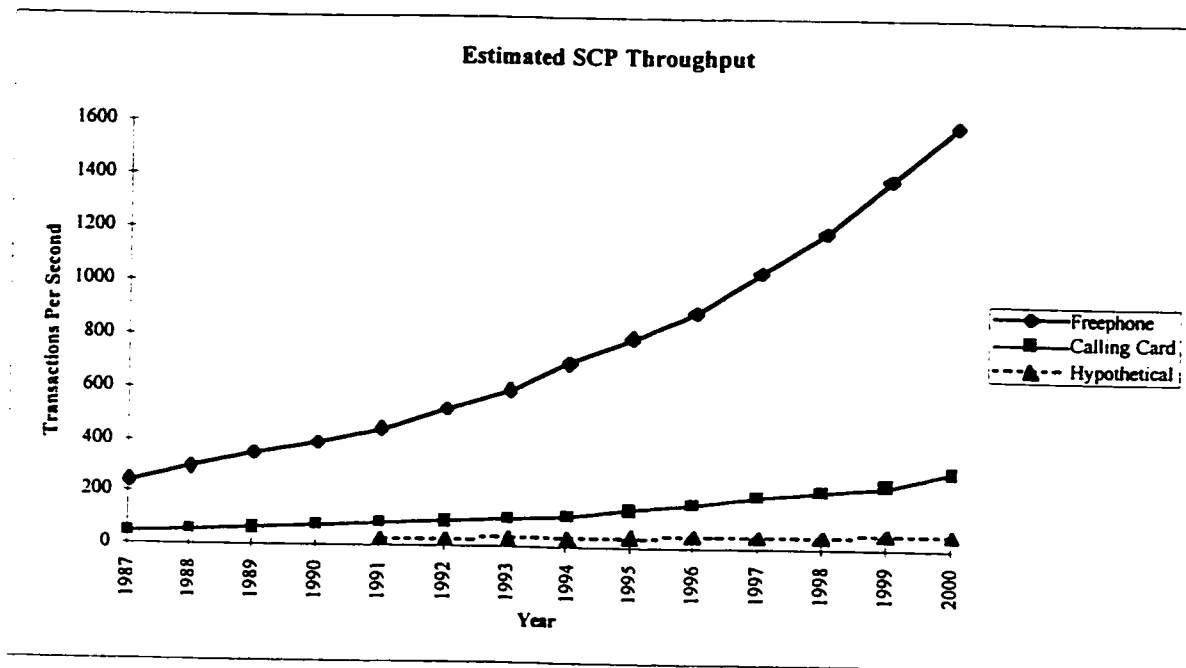


Figure 3.4 SCP Throughput Requirements

SCP throughput requirement of this study for the freephone service, calling card service and other hypothetical services. Hypothetical services represent new IN services such as wireless services, paging services, etc. The freephone service capacity started at about 250 calls per second in 1987, and assuming a 15% annual growth the rate will exceed 1600 calls per second by the end of the century. At this time, the freephone service is the most popular, hence generating most of the revenues for the network operators, service providers and customers. For example, the European market for the freephone service was worth more than US\$1 billion in 1996, and is expected to grow at compound annual growth. It is important to mention that most of today's SCPs deployed by major telecommunication equipment vendors achieve a throughput of 100 to 300 calls per second. As "*processing*" technology increases, this number will definitely increase.

3.3.2 SCP Call Sampling Activity

Another important challenge for the evolution of the IN is the acquisition and compilation of sampling/statistical information such as call sampling data (IN data records), billing data (IN billing records) and measurements [2,25,35,36,45]. From a management point of view, the SCP architecture must not only meet the range of throughput and response time requirements identified in the last section but must accommodate these administrative and maintenance activities. [42,43,44] gives a possible list of such activities. Real time access to such sampling/statistical information by the customer, service provider and network operator is an ever increasing requirement [25,47]. For example, operation and maintenance will require real-time information while forecasting and planning information could be provided after the event in non real-time. For all of these activities, the following major processing steps can be identified: (1) generation, collection and storage of data, (2)

analysis and processing of data, and (3) presentation and use of the analysis results. The generation, collection and output of raw data could be achieved by continuous as well as periodic and non-periodic measurements/sampling carried out in the SCP. The data analysis may be performed by the SCP itself or by another external administrative system depending on the following: (1) total amount of data, (2) need for analysis of data from multiple SCPs, and (3) processor load constraints.

It is known from real IN measurements that the SCP call sampling activity is responsible for major CPU resource consumption. The SCP call sampling activity can be up to five times the complexity of serving a freephone service [27,36]. When such activity is present in the SCP architecture, it will probably place a significant impact on the performance of the SCP. This could result in a degradation of the SCP throughput, SCP response time, blocking probability, etc. Impact on the performance due to the processing of this activity is therefore expected. This will be investigated in chapter 4 and 5. As explained in section 3.2, the SCP call sampling activity creates IN data records for a number of incoming IN calls. The IN data records collected by the SCP are then used by an external administrative system to gain information such as: (1) the network activities, (2) to provide user friendly reports to the customers such as detailed monthly statistics, (3) for billing and revenue settlements, (4) for administration of services, (5) for network planning and maintenance, (6) for traffic measurements, etc. For example, network operators would like to monitor the incoming traffic rate to the SCPs, or count the number of signaling messages, or keep some measurements when monitoring security issues, or to detect overload periods, or count the number of hardware and software failures, etc. As another example, a customer could request billing information and calling statistics, which could be generated on a daily, weekly or monthly report format. This

valuable information could help a customer optimize his business and/or monitor the operation of its business. In the case of a freephone service, a list of callers, their numbers and date, time of day when the call was made, how often the customer's number was busy, etc., could turn out to provide valuable information for the customer's business. In this thesis, we will not be focusing on the post-processing of the sampling/statistical information (e.g., creation of report format). We are only concerned on the performance impact of processing this activity.

3.3.3 SCP Polling Activity

Up to now the activities explained and performed by the SCP were: (1) processing of incoming IN calls, and (2) SCP call sampling. A third activity which can be deduced from the latter activity is, (3) polling activity. It is explained next.

We assume for now that the IN data record collected by each call processor stations of the SCP are stored in a disk-based system, for later retrieval. The DMP⁸ (see Figure 3.1), routinely polls the SCP (each CP stations) at pre-defined intervals, so that the IN data records collected on a time-basis can be transmitted to an external administrative system for further processing. After the IN data records are safely transferred over the token ring LAN (from SCP to DMP), the DMP's primary task is to process these IN data records (e.g., generate pre-defined and/or defined reports). Again the topic of creating/generating reports and post-processing the IN data records is not further studied in this thesis. We also assume that polling requests are usually generated outside busy hours (i.e., outside peak hours) in order not to degrade the performance of

⁸ Our DMP is somewhat equivalent to the SMP defined in the ITU-T IN standard.

the SCP. Thus, the transmission of IN data records is usually done overnight, when the traffic rate to the SCP is low. Hence, under these assumptions, the polling activity will not influence the stringent SCP call processing and SCP call sampling task.

3.3.4 SCP House-Keeping Activity

In addition to the activities identified in the last sections, the SCP must provide an array of house-keeping activities. Basically these activities include standard system management duties. Such duties are:

1. database management system software maintenance - storing and caching the SLPs and data, updating and deleting functions, memory management;
2. overload detection - allowing the SCP to obtain system overload state when abnormal situations occur;
3. event reporting and alarm management - performing logging of significant event during normal and abnormal conditions.

Many more house-keeping activities exist in an SCP [25,35,62]. These activities will be incorporated in our SCP model. More details will be given in section 3.6.

3.4 Service Control Point Memory Issues

In section 3.2 we presented a high level model of the IN. An SCP architecture was shown, in which call processor stations, a token ring LAN and specialized processors were depicted. Without going into the detailed internal SCP architecture at this time (we reserve this for chapter 4 and 5), we mention that the SCP holds a *service database*, which is distributed between *main memory* and a *disk-based system*. A disk-based system represents secondary storage. Throughout this thesis, the terms *main memory database* and *main memory service database* are used interchangeably, as well as *disk-based database* and *disk-based service database*. From now on, the former database will be known as MMDB, while the latter will be known as DBDB. The terms database and service database are also used interchangeably. The service database is used to store the SLPs required to execute IN services. In our SCP model, the service database is replicated in each call processor stations of the SCP (i.e., for availability purpose). Another DBDB system also exists in each call processor station. This database is known as the sampling database. It is used to store the IN data records collected during the SCP call sampling activity (refer to section 3.3.2.). In this section we focus on the service database modeling. Nevertheless the sampling database follows the same modeling issues.

The service database size of each call processor station depends on two factors: (1) the number of SLPs (customer records) to be stored, and (2) the SLP size. The SLP size basically depends on the data elements required to provide an IN service. For example, in the case of the freephone service, the SLP data elements could be the

following: the routing and number translation data, carrier to route the call to, indicators for validation, customer profile, and other optional data. Optional data could be: the time of day routing, restricted access number, call screening, call sampling, etc.

In order to assess the size of the service database, Table 3.1 lists the SCP service database characteristics that is used in our model. The table presents numbers for a service equivalent to the freephone service. In [35] the author also presents similar numbers for the ABS service and wireless services. With these values, an estimate of the storage capacity for the service database will be gained. This is analyzed in the following section.

Table 3.1 SCP Service Database Characteristics

Parameter	MIN Service (equivalent to freephone service)
1. Number of SLPs (to be stored) per SCP (in each call processor)	40K - 800K
2. Average SLP Size	600 bytes
3. Maximum SLP Size	10000 bytes
4. Average Service Database Size	24 Mbytes - 480 Mbytes
5. Maximum Service Database Size	0.4 Gbytes - 8 Gbytes

3.4.1 Main Memory-based Service Database

In order to meet the short SCP response time requirements and the high SCP throughput rate identified in section 3.3.1, the SCP should avoid all unnecessary overhead. One source of overhead that should be avoided is related to I/O operations (e.g., disk accesses to retrieve SLPs from the disk-based service database), since they are time consuming and degrade the performance of the SCP [36]. A solution to reduce the I/O overhead is to

store some SLPs into the *main memory service database* (MMDB), therefore reducing the number of I/O operations to the *disk-based service database* (DBDB). With the advent of semiconductor technologies it has become feasible to install larger and faster MMDB that can hold all SLPs required to execute and process IN services. But, to meet the response time and throughput requirements there is a tradeoff to such solution; a system where hundred thousands of SLPs (customer records) are to be stored, the price/performance ratio could impose a main memory constraint.

From the values given in Table 3.1, the *average service database size* should be 480 Mbytes per call processor station. Recall that this database is fully replicated in each CP station. Such database size is needed in order to store 800,000 SLPs having an average size of 600 bytes. This size does not take into account the database software and other applications software needed to ensure the operations of the SCP. It is clear that under certain conditions the service database size could turn out to be larger. For example, the database could increase when the average SLP size exceeds 600 bytes (which could mean that the service complexity increases), or when the number of SLPs to be stored exceeds the capacity of the SCP (which means that the customer demand for IN services increases beyond the forecasted values). A MMDB that could store all SLPs without any constraints would be the ideal situation in terms of SCP throughput and SCP response time guarantees, but would increase the system's price. Therefore, we introduce a parameter to cope with the case of large size SLPs and large MMDB. The parameter is called *IN call scenario*.

The IN call scenario parameter defines the distribution of SLPs that will be stored between the MMDB and the DBDB. In fact this parameter is used to capture the variety

of IN service types and their related complexity. As a basic assumption, the MMDB of the SCP will typically be used to store common SLPs (i.e., the SLPs that are the most utilized by users) which tend to be small in size (e.g., 600 bytes) but many in numbers (e.g., hundred thousands of SLPs). Hence the DBDB will be used to store SLPs (i.e., the ones that are not often utilized by users) which are large in size (e.g., 10000 bytes) but few in numbers (e.g., 40000 records). Table 3.2 presents an IN call scenario.

Table 3.2 IN Call Scenario Example

Number of DBDB visits	Distribution of SLPs
0 visit	80 %
1 visit	10 %
2 visits	10 %

The table states that 80% of all SLPs will be stored in the MMDB, since they require zero visit to the DBDB. That is to say, all service logic programs are stored into main memory, and no physical disk access is needed. The following two cases state that 20% of all SLPs will be stored in the DBDB. Ten percent will require one visit, while the other ten percent will require two visits to the DBDB. A visit is basically a fetch and store of a service logic program from the DBDB to the MMDB.

3.4.2 Disk-based Service Database

This section identifies the parameters involved in the modeling of the DBDB system (recall that this database constitutes one part of the whole service database). Several disk technologies exist, such as *Integrated Drive Electronics* (IDE) and *Small Computer System Interface* (SCSI) [16,17]. Although new technologies such as *Storage System Architecture*

(SSA) [70] are now being deployed in the SCP [10], our model will be based on the SCSI technology.

Disk average access time parameter is a measure of disk performance. It is the sum of the disk's *average seek time* and *average latency*. Disk average seek time reflects the average amount of time it takes to position the read/write head over the cylinder containing the data to fetch. The average latency is defined as the time it takes for a disk to rotate its disk platter, so that the proper sector can be read by the disk's head. Assuming SCSI technology, the parameters for the DBDB are given next. These parameters are used when parameterizing the queuing models in chapter 4 and 5.

The average seek time of a disk will be 8 milliseconds, and the latency will be 7200 revolutions per second. Therefore, the average access time is given as:

$$\text{average access time} = \text{average seek time} + \text{average latency} = 8.07 \text{ msec}$$

If we assume that the average SLP size stored on the disk is 600 bytes, and that the SCSI data rate is 10 Mbps, we have:

$$\text{average transfer time} = \frac{\text{average number of bytes per SLP}}{\text{data transfer rate}} = \frac{600 \text{ bytes}}{10 \text{ Mbytes per second}} = 0.06 \text{ msec}$$

Therefore, the *average service time* of the disk when a SLP is transferred between the DBDB and main memory will be:

$$\text{average service time} = \text{average access time} + \text{average transfer time} = 8.13 \text{ msec}$$

The *maximum service time* of the disk is twice the average service time, whereas the *minimum service time* is the average transfer time (i.e., assuming that the SLP to be read is already on the disk's read/write head). These two values will represent the upper and lower bounds of the disk service time.

$$\begin{aligned}\text{maximum service time} &= 2 \times \text{average service time} = 16.26 \text{ msec} \\ \text{minimum service time} &= \text{average transfer time} = 0.06 \text{ msec}\end{aligned}$$

3.5 Real-Time Network Traffic Measurements

This section presents the analysis of real-time traffic measurements observed on an active IN. Many SCPs in the IN elaborate various traffic measurements on an hourly basis (e.g., traffic rate offered to the SCP). In this traffic study, the history of the traffic rate data for the following time period was used: February 16 to 21, 1997, respectively starting on a Sunday and ending on a Friday. The history data was retrieved from the Stentor's AIN SCPs. Figure 3.5 depicts the SCPs geographical deployment in the Stentor's AIN. The AIN consisted of four SCPs, each of them dedicated to IN call processing. The SCPs were all interconnected by CCSS7 links emanating from the signaling network STPs (not shown in this figure). Notice in the figure that the SCPs operate in a mated pair fashion.

The IN services monitored were: freephone (800/888 like numbers), Service Finder and Switch Redirect. Service Finder and Switch Redirect services are not explained here, but significantly contribute to the traffic rate, as we will see.

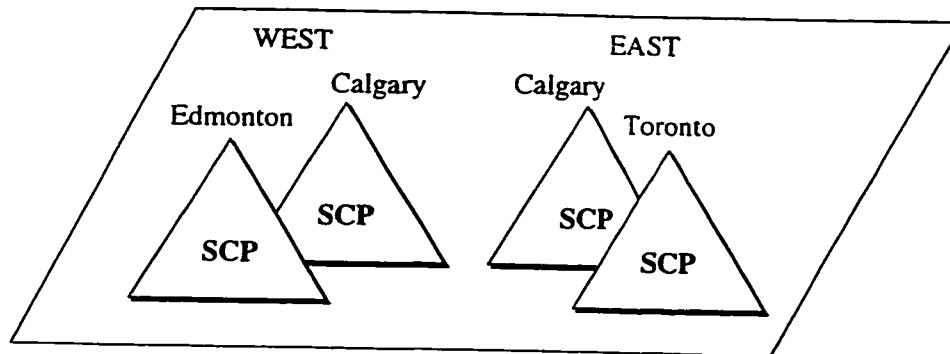


Figure 3.5 Geographical Deployment of SCPs

Figure 3.6 illustrates the hour offered-call curve over one week. Similar curves can also be observed in the French telephone network [46]. The figure shows two types of pattern: (1) a profile for weekdays, and (2) a profile for the weekend. Daily call curves have two peak periods and one off-peak period (except on Monday). During the weekend (Saturday is not shown) the traffic rate is fairly low compared to the rest of the weekdays (Monday through Friday). This is primarily due to the fact that users make more phone calls during weekdays, and that most calls, such as freephone, calling card service, Service Finder and Switch Redirect are related to businesses. Notice that the majority of incoming calls are generated during business hours, from 9:00 am to 5:00 pm.

From this figure, we are interested in the *traffic modeling* that characterizes the amount of calls per unit of time, namely the number of calls per second generated. Suppose we take Day 3 (Tuesday) as an example. The *peak traffic rate* is around 225000 calls per hour, which translates to 62.5 calls per second. But as we will see in the next paragraph, this traffic rate value does not represent the *real traffic rate* seen by the SCP. This is explained next.

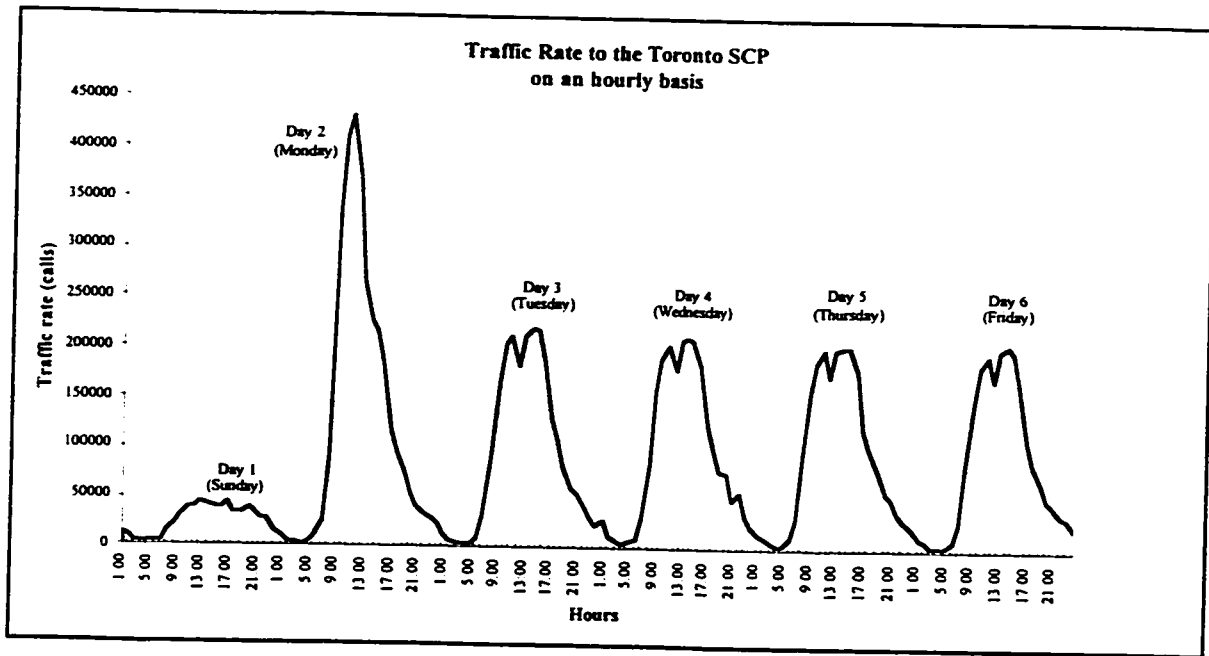


Figure 3.6 Traffic Rate to the Toronto's SCP

Suppose that a freephone SLP is normalized as having a complexity of one. It is known that the Service Finder and Switch Redirect services have complexities of approximately nine and five times the complexity of the freephone service, respectively [27]. Therefore, processing nine freephone calls is equivalent to process a Service Finder service, whereas processing five freephone calls is equivalent to process a Switch Redirect service. We further assume (based on data provided to us) that 70% of all calls generated during the peak hour period are freephone calls and that the remaining 30% are distributed in two equally half portions between the Service Finder and Switch Redirect service. From this assumption the *real traffic rate* (real arrival rate) with respect to the freephone service complexity is:

$$(0.7 \times 62.5) + (0.15 \times 9 \times 62.5) + (0.15 \times 5 \times 62.5) = 212 \text{ calls per second.}$$

This means that the Toronto's SCP processes the equivalent of 212 freephone calls per second during peak hours (not 62.5 calls per second as identified earlier). This parameter will turn out to be useful when modeling the traffic trace for our simulations.

We now explain how we model the traffic trace used in our simulations. The traffic trace is depicted in Figure 3.7. Notice that the shape is similar to the shape of weekdays presented in Figure 3.6. It represents a two day period of the six day traffic measurements presented in Figure 3.6. The first day is represented from 0 seconds to 240 seconds (simulation time), while the second day is from 240 seconds to 480 seconds. We assume that 10 seconds of simulation time represents one hour of real-time traffic measurements. It is also assumed that the whole traffic trace encompasses two four-hour traffic periods (i.e., two times 40 seconds of simulation time) at approximately the peak traffic rate. Note that the peak traffic rate in the trace is equal to the SCP throughput requirement imposed on our SCP model, which is around 800 to 900 calls per second.

To obtain the trace of Figure 3.7 (i.e., a traffic trace that achieves the peak rate of 800 to 900 calls per second) we simply multiplied the values of Figure 3.6 by a pre-determined constant. This had to be done since the incoming traffic rate to the Stentor's SCPs was well under the forecasted value of our SCP model.

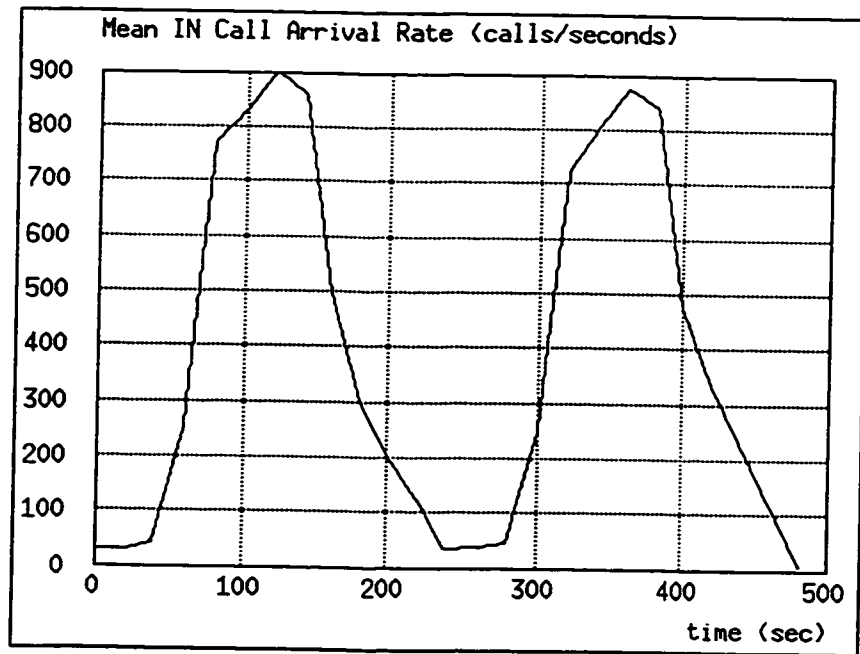


Figure 3.7 Traffic Rate Trace for the Simulations

3.6 Statistical Distributions

The low-level models (i.e., the queuing models) that will be develop in chapter 4 and 5 will encompass entities such as traffic generators, central processing units, disk systems, token ring topology, etc. In order to properly characterize some of their tasks, each of these entities will require their own set of attributes. This section considers two probability distributions that play an important role in characterizing and modeling some entities part of the SCP. For example, (1) the time between successive arrivals at a service facility (i.e., the interarrivals), can be modeled as a truncated exponential distribution, and (2) processing time at a service facility (e.g., of IN calls), denoted by the service time, can be modeled as a truncated normal distribution. These are two of the

most important probability distributions in the entire field of statistics. They are explained next.

A continuous *Random Variable* (RV) X having a bell-shaped distribution is called a normal RV [71]. The mathematical equation for the probability distribution (i.e., probability density function) of the normal RV depends on the parameters μ and σ , the mean and standard deviation, respectively. Hence we denote the values of the normal density function of X by $n(x; \mu; \sigma)$. The density function of the normal RV X shown in Figure 3.8, with mean μ and variance σ^2 , is:

$$n(x; \mu; \sigma) = f(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-\mu)^2}{2\sigma^2}} \text{ for } -\infty < x < \infty$$

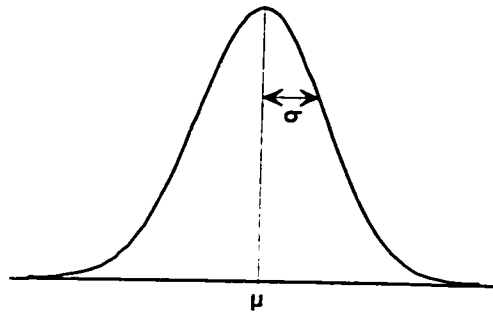


Figure 3.8 Normal Probability Density Function

A continuous RV X having the shape shown in Figure 3.9 is called an exponential RV [71]. The mathematical equation for the probability distribution of the exponential RV depends on the parameter $\frac{1}{\mu}$, its mean. Hence we denote the values of the exponential

density function of X by $\exp\left(\frac{1}{\mu}\right)$. The density function of the exponential RV X shown in Figure 3.9, with parameter $\frac{1}{\mu}$, is:

$$\exp\left(\frac{1}{\mu}\right) = f(x) = \mu e^{-\mu x} \text{ for } 0 \leq x < \infty$$

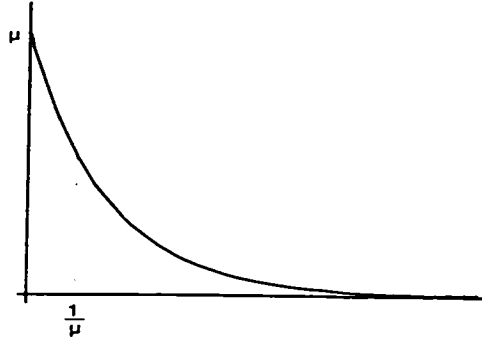


Figure 3.9 Exponential Probability Density Function

We now proceed to discuss the role that these two distributions play in the SCP modeling. We give two examples:

1. *Central Processing Units* (CPUs) used in the call processor stations of the SCP are dedicated to provide the IN call processing activity (e.g., dedicated to process freephone service or to validate calling card numbers), the SCP call sampling activity (i.e., to generate IN data records), and house-keeping activities. In order to model such activities, we assume that the CPU service time to process IN calls (with respect to the freephone service and calling card service) follows a *truncated normal distribution* with mean $\mu=3.33\text{msec}$ and standard deviation $\sigma=\pm 6\%$ [27]. With the use of standard normal

distribution equations and statistical tables [71], the bounds of the truncated normal distribution are calculated as follows:

$$P(x_1 < X < x_2) = \int_{x_1}^{x_2} n(x; \mu; \sigma) dx = \int_{z_1}^{z_2} n(z; 0, 1) dz = P(z_1 < Z < z_2), \text{ where } Z = \frac{X - \mu}{\sigma}$$

Therefore with $z_2 = \frac{3.75 - 3.33}{0.2}$ and $z_1 = \frac{3.0 - 3.33}{0.2}$ (and the use of the normal statistical table), the area under the standard normal distribution curve (i.e., the unit normal distribution) is:

$$P(z_1 < Z < z_2) = .9326$$

This probability states that 93.26% of the time, the CPU service time (i.e., RV X) will fall between 3.0msec and 3.75msec. These two CPU service time values represent the bounds of the truncated normal distribution. They represent the lower CPU service time bound and upper CPU service time bound. To fully model the truncated normal distribution the complement of the probability calculated above will be distributed as follows.

$$\text{If } \begin{cases} X < x_1 \text{ then } X = \cup [x_1, x_2] \\ X > x_2 \text{ then } X = \cup [x_1, x_2] \end{cases} \text{ else } X = x$$

where $x_1 = 3.0\text{msec}$ and $x_2 = 3.75\text{msec}$

Basically, when the CPU service time RV X falls outside the truncated normal distribution, it will be uniformly redistributed over the bound $[x_1, x_2]$. Else, the CPU service time is X . Such constraint will alleviate the potential situations where large or negative CPU service times could be generated from the distribution, which are not practical and conceivable cases.

2. Disk-based systems (i.e., the DBDB and sampling database) part of the call processor stations are used to store the SLPs and IN data records. Therefore an IN call requiring to visit the DBDB will experience a service time. Recall that in section 3.4.2 we defined the service time as being the addition of the access time plus the transfer time. We now assume that the disk service time follows a *truncated exponential distribution* with mean $\frac{1}{\mu} = 8.13\text{msec}$. This value is obtained from section 3.4.2. The bounds of the truncated exponential distribution were also given and calculated in section 3.4.2. The maximum service time was 16.26 msec, whereas the minimum service time was 0.06 msec. Therefore, with the use of the exponential distribution equations [71], we get:

$$P(x_1 < X < x_2) = \int_{x_1}^{x_2} f(x)dx = \int_{x_1}^{x_2} \mu e^{-\mu x} dx = 0.86$$

where $x_1 = 0.06\text{msec}$ and $x_2 = 16.26\text{msec}$

This probability specifies that 86% of the time, the disk-based service time will fall between 0.06msec and 16.26msec, the lower bound and upper bound, respectively. In order to fully model the truncated exponential distribution, the complement of the probability calculated above will be distributed as follows.

$$\text{If } \begin{cases} X < x_1 \text{ then } X = \cup [x_1, x_2] \\ X > x_2 \text{ then } X = \cup [x_1, x_2] \end{cases} \text{ else } X = x$$

where $x_1 = 0.06\text{msec}$ and $x_2 = 16.26\text{msec}$

Basically, when the disk-based service time RV X falls outside the truncated normal distribution, it will be uniformly redistributed over the bound $[x_1, x_2]$. Else, the CPU service time is X . Such constraint will alleviate the potential situations again where large and small values for disk-based service time could exist, which are not practical and conceivable cases.

Chapter 4. Simulation Results /1

4.1 Introduction

This chapter has three objectives: (1) to define and give the technical description of the first SCP configuration (configuration 1) under study, (2) to define a number of models and to parameterize these models, and (3) to investigate and assess by means of simulations the performance of the SCP.

By understanding the IN, and determining how it should perform, behave and operate, a series of low level models will be built with respect to the high level model and system operation presented in chapter 3. Some of these low level models will be described as queuing network models. The emphasis will be primarily placed on the internal architecture and the parameterization of the SCP, since it is the primary entity under study. The definition of these low level models will be made relatively straightforward by the close correspondence between the parameters of queuing network models and the parameters of real computer systems.

The models defined will then be used to simulate and assess the performance of the SCP in terms of the three primary performance metrics: (1) the SCP throughput, (2) the SCP response time, and (3) the SCP call sampling rate. Other performance metrics, ranging from buffer size, CPU utilization, blocking probability, etc., are also monitored. Tunable factors, parameter settings, routing policies, probability distributions, traffic rate, etc., are also covered along the way.

4.2 Configuration 1

We are now in a position to define the operation of configuration 1. In the following diagrams, we use round-corner boxes to represent subsystems that are composed of a set of basic building blocks. Rectangular boxes will be regarded as the basic building blocks of our model and their internal architectures will not be elaborated, although their functions will be mentioned.

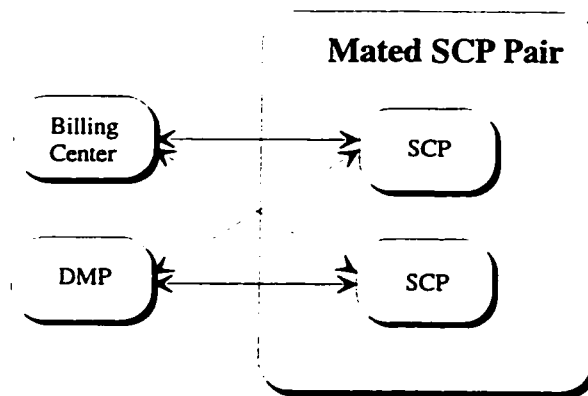


Figure 4.1 SCP Architecture for Configuration 1

At the highest level of our IN configuration, a mated pair of SCP node is abstracted as shown in Figure 4.1. SCPs are required to have seven days a week, 24 hours a day availability with very little down time (e.g., less than 3 minutes per year). The usual solution for achieving such high availability is to provide a mated pair of SCPs that run in a load sharing configuration where no faults occurs, and where the entire traffic rate can be handled by one of the pair during a failure. The DMP and (Billing Center) are external administrative systems that send polling requests and (billing requests) to each SCP node according to a deterministic traffic pattern. The SCPs respond by sending reply messages that contain sampling/statistical information and (billing data) back to the DMP and

(Billing Center). The DMP administrative system and the process of collecting/gathering IN data records was presented in chapter 3. Recall that configuration 1 is characterized by the absence of the *Data Processor* (DP), earlier shown in Figure 3.1 of chapter 3. The DP will be incorporated in configuration 2 of chapter 5.

Each SCP node of the mated pair configuration shown in Figure 4.1 shares the same architecture. The architecture is shown in Figure 4.2. It closely resembles the high level model that was depicted in chapter 3. The main structure in the SCP node consists of two unidirectional 16 Mbits per second token rings; one having clockwise traffic while the other has counter-clockwise traffic. A dual token ring LAN topology is used for availability and connectivity purposes. With this approach, connections between token rings, call processor stations and other specialized processors provides reliable communication. For instance, the use of interconnected call processor stations permits: (1) load-sharing between CP stations, (2) transfer of processing and information in a CP station failure situation, and (3) the exchange of information when the required SLP and/or data would be stored in another CP station (or in a remote database). Also, the use of all interconnected entities provides the reliable transfer of sampling/statistical information (i.e., IN data records) to a remote external administrative system (that could be interconnected through a TCP/IP network in a real system). In configuration 1, four stations are connected to the dual token ring. The first three stations are depicted as call processor stations, while the fourth station is depicted as a token ring interface (basically this interface would reside in a router). A token ring interface links the DMP with the token rings, since the DMP is an external physical entity (i.e., not physically part of the SCP). Three identical call processor stations are attached to the token rings. They equally share the IN call traffic rate coming from the SSPs. In reality, and as it was

identified in chapter 3, the SSPs should be connected to the SCP via CCSS7 links (i.e., over the CCS network). We do not consider in this thesis the modeling of the CCSS7 network. We assume that the CCSS7 network is lightly loaded, and that it does not influence the physical entities (e.g., queuing delay) part of the IN, nor does it create any bottleneck situations. Recall that the performance evaluation is mostly focused on the SCP architecture. For instance, we are not looking at the overall end-to-end delay where the end-to-end delay would include the following four components: SSP processing time, CCS network delay, SCP processing time and call setup time.

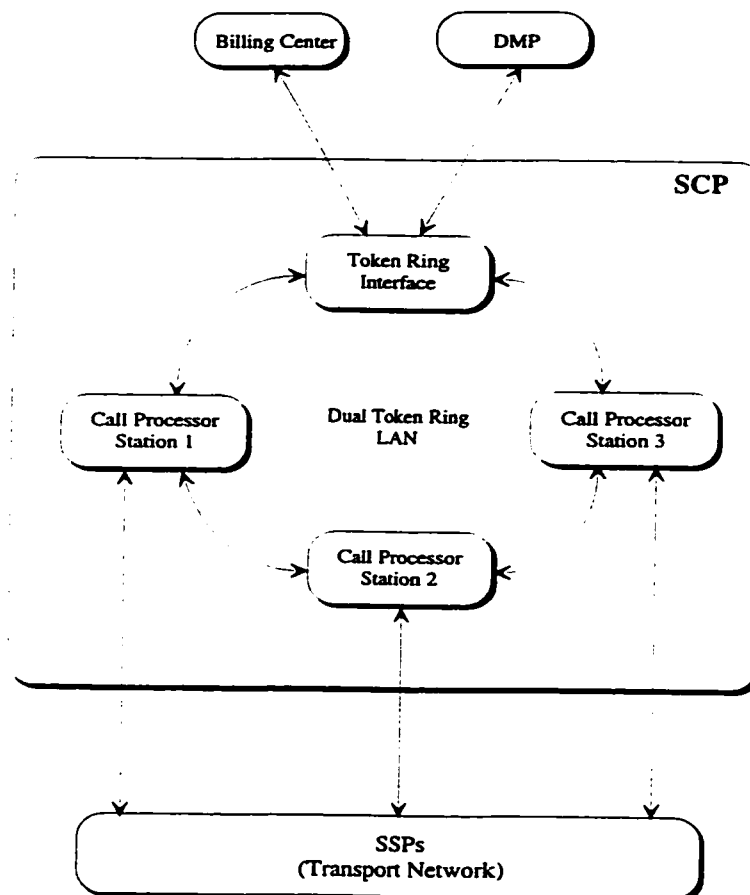


Figure 4.2 Service Control Point Model

We now examine the token ring interface model of the SCP. The token ring interface will also be found in each call processor station (since they are attached to the token ring). The use of the token ring interface is needed for two reasons: (1) in order to transfer the IN data records from the call processor stations to the DMP (when a polling request is received), and (2) used when a call processor station needs to communicate with one of its peer CP station. In the latter case the request would be sent over the token ring to the other call processor station via the token ring interface. The token ring interface model is shown in Figure 4.3. It is explained next.

Outgoing requests (either from the DMP, or from the call processor stations) choose one of the two rings at the *ring select* module. Currently, a probability of 0.5 is assigned to select either one of the two rings. A data packet (i.e., frame), which is the transmission unit over the token ring LAN, is then passed to the corresponding medium access control layer, *MAC module*, to wait for actual delivery. The MAC is a built-in module provided by the OPNET Modeler. It implements the token ring protocol. Incoming data packets (obviously coming from the two token rings) to the token ring interface are picked up by the two MACs and are pooled together at the *merger* module. They are then subsequently handed to the host (i.e., either the DMP or a call processor). There is no queuing nor any delay at the ring select and merger module. However, outgoing data packets have to queue up at the MAC module.

As shown in Figure 4.4, each call processor station of the SCP has a token ring interface and a call processor. Figure 4.4 depicts two inputs and two outputs to the call processor station. IN calls coming from the SSPs are directly routed to the call processor while polling requests (i.e., from the DMP) coming from the token ring are forwarded to

the call processor via the token ring interface of the CP station (e.g., polling requests circulate over the token ring up to the specified CP station). Outputs of the call processor station are used to send the response to the SSPs when the processing of IN calls is completed, or used to send the IN data records to the DMP.

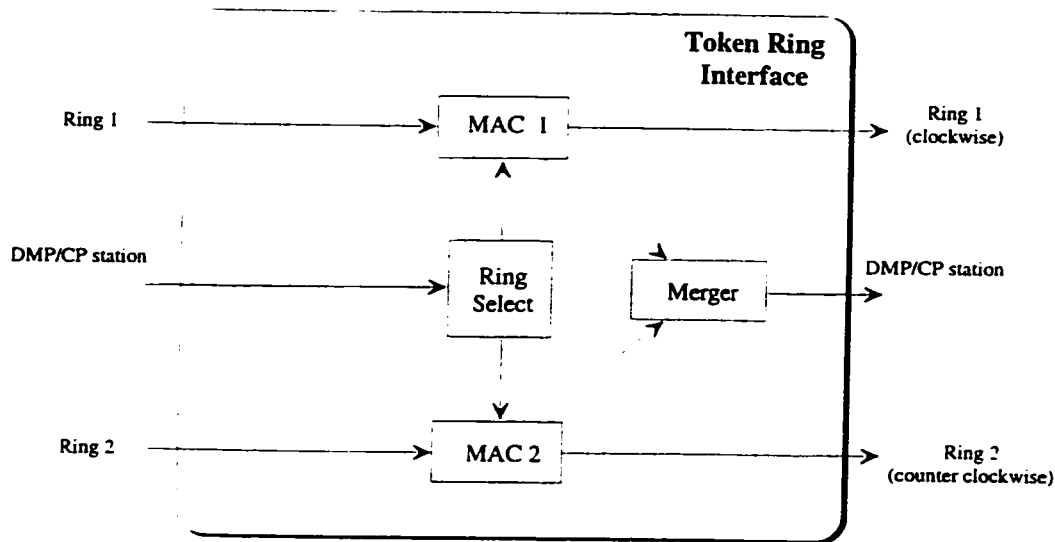


Figure 4.3 Token Ring Interface Model

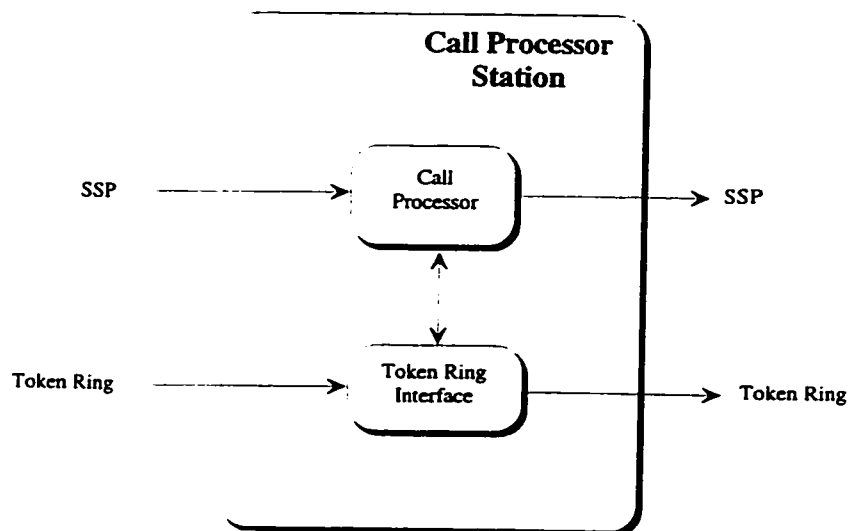


Figure 4.4 Call Processor Station Model

The core element of the SCP is the *call processor station*, whereas the core element of the call processor station is the *call processor* itself. The call processor queuing model is shown in Figure 4.5. It is composed of the following modules: *Network Interface Units (NIU)*, *CPU Subsystem (main memory and CPU)*, and a disk-based system which is decomposed into a *service database* (used to store the SLPs) and a *sampling database* (used to store the IN data records). The detailed explanation of the call processor is given next.

As identified earlier, polling requests generated by the DMP arrive at the call processor via the token ring interface. They are buffered at the network interface unit NIU-I2 upon their arrival to the CP (see Figure 4.5). Then they proceed to the CPU subsystem queue, where they are queued until being processed. IN calls coming from the SSPs arrive at the call processor where they are buffered at the network interface unit NIU-I1, upon their arrival (see Figure 4.5). Then they proceed to the CPU subsystem queue, where they are queued until being processed. The NIUs represent network interface processors (i.e., small frontend processors) [2,35,49]. Generally, network interface processors are used to perform pre-processing. For example, they are used to process portions of the SS7 protocol (i.e., measurements, generation of operation and maintenance data, and service recognition of IN calls). Most of all, the network interface processors are used to communicate with the CPU Subsystem. We assume that network interface processors do not involve any disk I/O operations and are engineered in order to avoid problems such as bottleneck situations. The network interface processor service time will follow an exponential distribution. It is usually how they are modeled in the literature [22,40,49]. Basically, these network interface processors are included in our model for clarity purposes, and to make the configuration act and look like a real system.

The CPU subsystem performs the SS7 TCAP application layer protocol processing of incoming IN calls. The CPU's main task is to serve IN calls by processing the appropriate SLPs. For some IN calls, the SLPs and data required to process the call may be stored in the CPU's MMDB, or in the DBDB (this was previously discussed in chapter 3). In the former case, the IN call will require zero visit to the service DBDB, whereas in the latter case, the IN calls will need to visit the DBDB (i.e., in order to copy the proper SLP and data from the disk into the CPU's main memory). IN calls requiring disk visits are routed to one of the two disks of the service database (see Figure 4.5). The visits are governed by a pre-defined distribution. The distribution was identified and explained in chapter 3 as being the *IN Call Scenario*. After the IN call CPU requirements and service database requirements are completed, a response exits from the CPU subsystem and returns to the SSP via NIU-O1.

A certain number of IN calls are to be sampled (refer to chapter 3 for a description of the SCP call sampling activity). After an IN call processing requirement is over (i.e., CPU processing and service database visit), the IN call enters the process of being sampled. This is done just before sending the response back to the SSP. The CPU is now in charge of sampling the call, and to create an IN data record. When the call sampling activity is over, a response is sent to the SSP in the same manner as when a call was not sampled, whereas the IN data record is routed to the sampling database. In reality, the sampling disk-based peripherals are connected on the same bus as the disk-based service database. Recall that with SCSI technology, eight peripherals can be supported on one controller. In our model, we assume that three of the eight disk peripherals are used to model the sampling database, while two of the eight disk peripherals are used to model the service database. The remaining three are storing the

operating system and other related application for the SCP. The latter three disks are not shown in Figure 4.5.

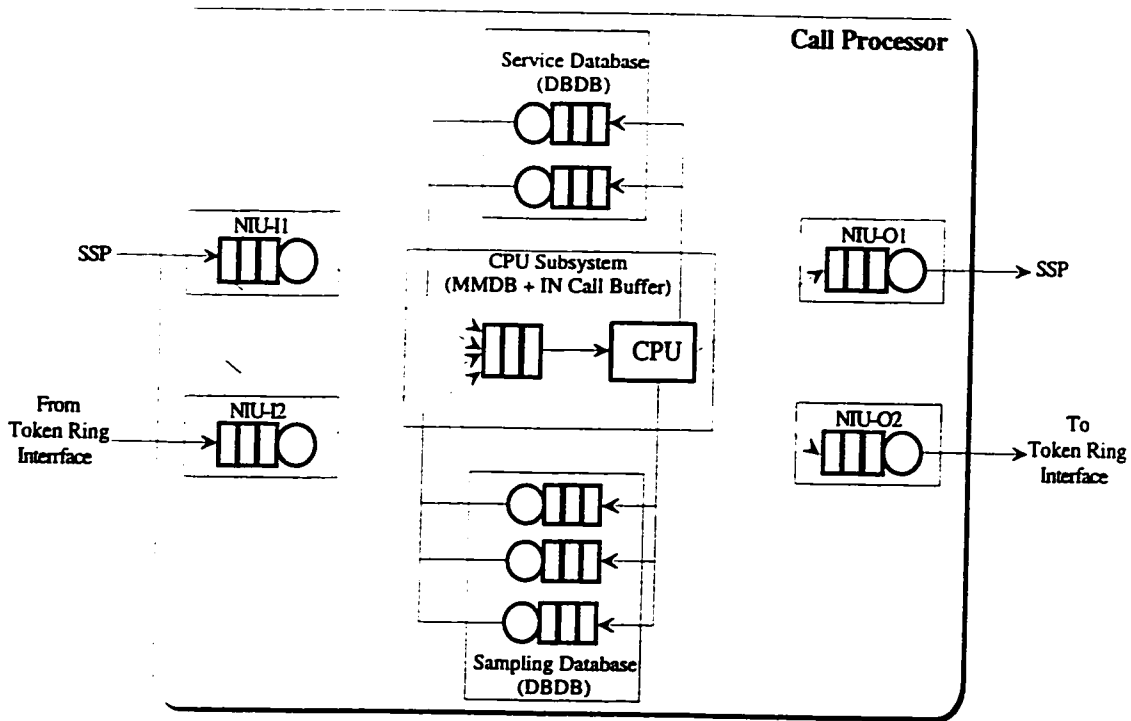


Figure 4.5 Call Processor Model

The main memory of the CPU subsystem is composed of two parts: the first part encompasses a MMDB which holds some SLPs and data (depending on the IN call scenario), while the second part encompasses a *buffer* (also in main memory) used to queue the incoming IN calls. They are depicted together in Figure 4.5. The buffer is modeled in such a way that a maximum number of x calls can be admitted in each call processor. If at a given point in time the number of calls (i.e., to be queued in the main memory buffer) exceeds the maximum limit of x calls, then the exceeding amount of IN

calls will be considered as *blocked* or *load-shared*⁹. This condition will mostly arise when the SCP is under heavy traffic rate and traffic workload, hence under a congestion situation (period of overload). In such case a call will be subject to two situations. The first situation assumes that the user receives a voice announcement asking him to retry again later. This is defined as a *blocked call*. For example, a voice announcements is usually in the following form: “*All lines are presently busy, please try again later...*”. The second situation assumes that a portion of blocked calls are to be routed to another SCP which is lightly loaded (i.e., load-sharing between SCPs). This is defined as a *load-shared call*. For example, the IN calls could be routed to its mated SCP. It is important to mention that a call handled by a load-sharing mechanism will definitively suffer a longer response time, because of the subsequent routing and processing time needed to serve the call. The issue of load-sharing mechanisms and the impact on the SCP response time is not studied in this thesis. It was shown in [23,24] that the number of blocked/load-shared calls could be reduced, at the cost of longer response time, when congestion control mechanisms (the same would apply to load-sharing mechanisms) were implemented in a similar SCP model to ours. Other studies [21,22,23] identify the QoS impact, when deploying SCP congestion control mechanisms.

The SSP model is shown in Figure 4.6. IN calls are generated at the *traffic generator* module. These calls are then routed by the *distributor* module to one of the three call processor stations of the mated pair of SCPs. Calls are routed in a round-robin fashion. The *SSP sink* module is used to collect the response received from the SCP (i.e.,

⁹ A *load-shared* call is defined as follows: it is an IN call that is migrated to an SCP (i.e., that has the proper resources to process the call) with the use of load-sharing mechanisms. Usually an IN call will be migrated from an SCP that is under periods of overload, to an SCP which is not under periods of overload.

the response used to complete the call setup). Recall that the traffic generator module used in the SSP is based on the real-time traffic measurements presented in section 3.5, and the traffic trace depicted in Figure 3.7. The DMP model in Figure 4.7 resembles the SSP model of Figure 4.6. A *polling generator* module produces polling requests. The *distributor* module decides which particular call processor station each polling request goes to, and which SCP of the mated pair is involved. In response to the polling requests, the *DMP sink* module collects the IN data records from the SCPs. When all IN data records are gathered, the DMP initiates the processing of the IN data records. The Billing Center would be based on the same model as the DMP, except that it would be in charge of polling and collecting the IN billing records gathered by the SCPs. Recall that we neglect and omit all issues concerning the Billing Center in this thesis.

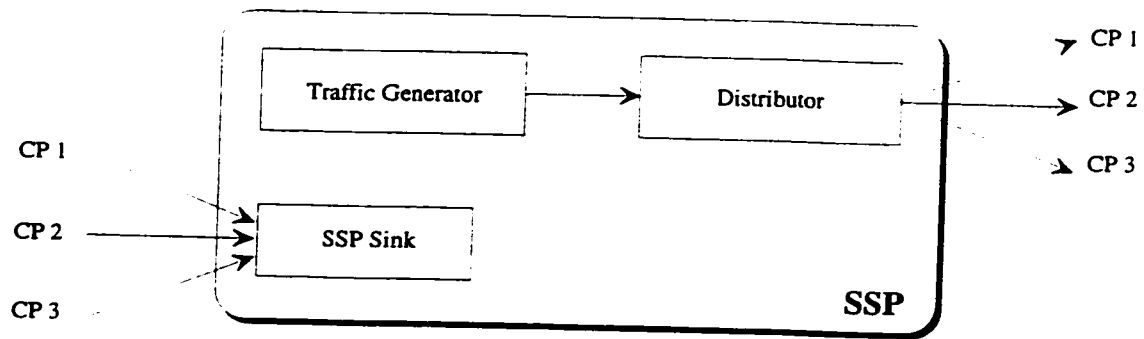


Figure 4.6 Service Switching Point Model

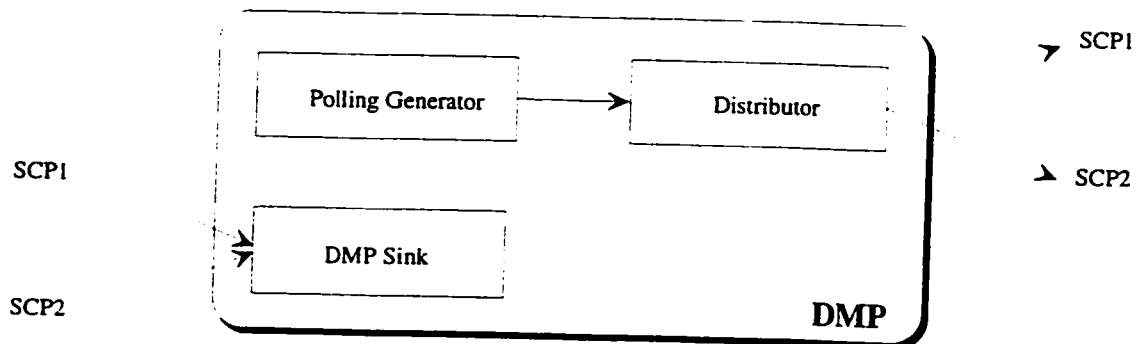


Figure 4.7 Data Management Processor Model

4.3 First Set of Simulations

The first set of simulations is based on configuration 1, with some minor modifications. All operations involving the SCP call sampling activity and polling activity are omitted. Therefore, all physical entities involved in these two activities (i.e., DMP entity and sampling disk-based system) do not have any influence on the performance of the SCP. Thus, the SCP model of Figure 4.2 is primarily composed of three interconnected call processor stations, which participate strictly to the IN call processing activity.

The main issue under investigation in this first set of simulations is related to the issue of *main memory-based service database* (MMDB) and *disk-based service database* (DBDB) deployment under the IN call processing activity. With respect to Figure 4.5, the MMDB is basically the main memory database residing in the *CPU Subsystem*, while the DBDB is basically the disk-based database residing in the *Service Database*. We are interested in evaluating and observing the performance of the SCP in terms of the SCP response time, SCP throughput and blocking probability. The goal will be to analyze and find a suitable *IN Call Scenario* able to fulfill the QoS and requirements imposed on our SCP model.

The distribution of SLPs between the MMDB and DBDB was explained in chapter 3. We identified a parameter known as the *IN Call Scenario*. An IN call scenario is basically composed of three values, and is denoted in the following form: $(X\%, Y\%, Z\%)$. $X\%$ represents the amount of IN calls that require zero visit (i.e., all SLPs reside in the call processor MMDB). $Y\%$ represents the amount of IN calls that require one visit to the DBDB, and $Z\%$ represents the amount of IN calls requiring two visits to

the DBDB. For example, the following IN Call Scenario (25%, 50%, 25%) means that 25% of all SLPs will be stored in the MMDB, whereas 50% will be stored in the DBDB and will require one visit. The remaining 25% will be stored in the DBDB and will require two visits. Table 4.1 presents the three IN Call Scenarios investigated in this first set of simulations.

Table 4.1 IN Call Scenarios Simulated

Scenario	0 access	1 access	2 accesses
1	75%	25%	0%
2	90%	5%	5%
3	95%	5%	0%

Before analyzing the simulation results of the three IN call scenarios listed in Table 4.1, we present and describe all the parameters involved in the parameterization of our SCP model. The parameters are listed in Table 4.2.

Table 4.2 SCP Parameters for Configuration 1

Parameter	Value
1. Mean CPU service time	3.33 msec
2. Mean NIU service time	25 μ sec
3. Mean DBDB service time	8.13 msec
4. Number of service database disks	2
5. Number of sampling database disks	0
6. Number of Call Processors per SCP	3
7. SCP call sampling rate	0%
8. Buffer Size	from 125 to 300

The *mean CPU service time* represents the CPU processing time for each IN call visit. The CPU service time follows a truncated normal distribution with a mean of 3.33 msec. The lower bound and upper bound are given as 3.0 msec and 3.75 msec respectively. These bounds were defined in chapter 3. The *NIU service time* is the processing time required to process an IN call at the CCSS7 network interface unit. The service time follows an exponential distribution with a mean of 25 μ sec. The latter value is a conservative estimate. Recall that in chapter 3, we identified that the NIUs (i.e., network interface processors) were engineered in order not to create bottleneck situations. The *mean DBDB service time* is the processing time required for each IN call visit to the disk system. The disk service time follows a truncated exponential distribution with a mean of 8.13 msec. The lower bound and upper bound were given in chapter 3 as 0.06 msec and 16.26 msec, respectively. The *number of disks* for the service database and sampling database in each call processor is 2 and 0, respectively. The *number of call processors per SCP* is given as 3 (recall that more than three CP stations could be added for additional processing capacity). The first set of simulations assume that the SCP call sampling activity and polling activity are not present, hence an *SCP call sampling rate* of 0%. The *buffer size* parameter represents the amount of IN calls that can be buffered in each call processor (i.e., in the CPU subsystem main memory).

4.3.1 Simulation Results for IN Call Scenario 1

This section presents the simulation results for IN Call Scenario 1 (75%, 25%, 0%). The results are extracted from the simulation performed under configuration 1, and the parameters identified in Table 4.2. The simulation results are also subject to all issues mentioned in chapter 3.

Table 4.3 is presented in a five column format. The first column identifies the buffer size of each call processor. Recall that this buffer resides in the CPU subsystem's main memory, and is used to queue incoming IN calls from the SSPs. The next three columns (column 2 - 4) identify the number of IN calls that were successfully served under three range (or category) of response time. The first range of response time is delimited from 0 to 0.5 seconds, whereas the second range is from 0.5 to 1.0 second. The third range of response time is from 1.0 second and beyond (1.0+ seconds). The fifth column identifies the number of IN calls that were not served by the SCP. We earlier assumed and explained in section 4.2 that an IN call could either be *blocked* (i.e., not admitted by the SCP) or *load-shared* (i.e., routed to another SCP). From now on, these two latter terms will be denoted as one single term called *blocked/load-shared*, unless stated otherwise.

Table 4.3 Simulation Results for IN Call Scenario 1

Buffer Size	0 to 0.5 seconds	0.5 to 1.0 seconds	1.0+ seconds	Blocked/Load-Shared
125	157717	24269	0	5277
150	148650	23176	5139	5204
175	83354	70146	22846	5248
200	83492	70502	22806	5083
250	82006	71373	23482	5147
300	81158	66873	29506	4970

Figure 4.8 depicts in a graphical format the IN call response time histogram versus the buffer size. The histogram cells are divided according to the last four columns identified in Table 4.3. The legend identifies the cell division. The observations that fall in each cell denotes the frequency of IN calls that were served under one of the three range of response time or the number of IN calls that were blocked/load-shared. Notice that a maximum buffer size limited to 125 and 150 presents the best results with respect to

response time. Hence, the number of calls that fall into the 0 to 0.5 seconds range for the two buffer size is considerably higher (i.e., approximately two times higher) than the remaining buffer size values. The histogram also shows that the number of blocked/load-shared calls remains constant throughout all buffer size values. Therefore, increasing the buffer size does not help in decreasing the number of blocked/load-shared calls, which is one issue that must be taken into account. Indeed, increasing the buffer size value as the effect of increasing the response time of IN calls.

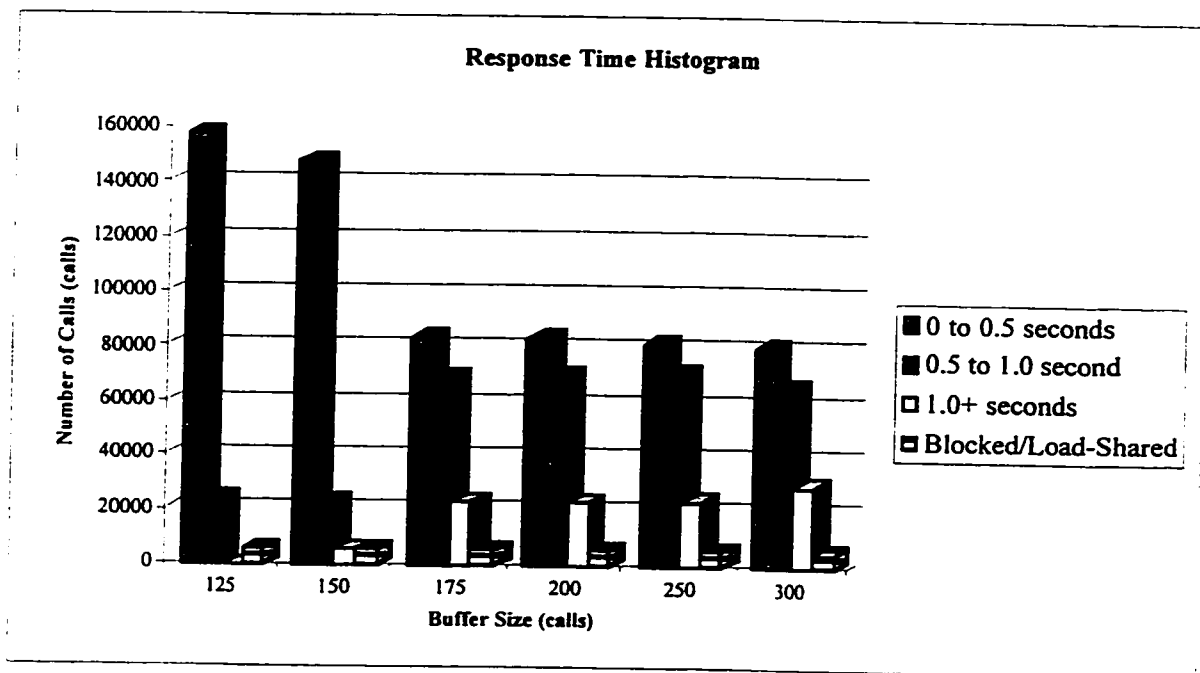


Figure 4.8 Response Time Histogram for IN Call Scenario 1

Table 4.4 presents the number of calls (in terms of percentage) that were processed below 1.0 second, and the number of calls (in terms of percentage) that were blocked/load-shared. We identified in chapter 3 that the SCP response time imposed on our model should guarantee that 95% of all IN calls be processed below 1.0 second. From now on,

this quantity will be known as the *95% quantile response time* value. Table 4.4 shows that as the buffer size increases the number of calls served below 1.0 second decreases, hence decreasing the 95% quantile value. This decrease is attributed to the fact that the number of blocked/load-shared calls remains constant throughout the buffer size values. Therefore, as the buffer size increases more IN calls are being processed beyond 1.0 second.

Table 4.4 95% Quantile Response Time for IN Call Scenario 1

Buffer Size	Number of Calls served below 1.0 second	Number of Blocked/Load-Shared Calls
125	97.1%	2.9%
150	94.3%	2.9%
175	84.5%	2.7%
200	84.6%	2.8%
250	84.2%	2.8%
300	81.1%	2.7%

Beyond a buffer size of 150 the 95% quantile response time is not guaranteed, even though the number of blocked/load-shared calls remains constant. For instance, when the buffer size is 300, the number of calls processed below one second falls to 81.1%, which is well under the requirement. The explanation for this decrease is as follows: as the buffer size increases, the total number of calls admitted in the SCP increases (i.e., buffered in main memory), hence increasing the response time of all calls (i.e., beyond 1.0+ second). The increase in response time will even be more critical when the SCP approaches saturation (i.e., when the SCP is pushed to its processing limit). Therefore, this scenario only guarantees the 95% quantile response time for a buffer size of 125 and 150. The latter value is considered even if it is not exactly 95%. It is worth mentioning that for a buffer size of 125, all calls are either processed below 1.0 second or blocked/load-shared (i.e., no calls are processed beyond 1.0 second). This is due to the

fact that a buffer size of 125 limits the maximum SCP response time to a value less than 1.0 second under this IN call scenario (even under periods of overload). When the buffer size limit is achieved, the calls entering the SCP will definitely fall into the blocked/load-shared category.

Figure 4.9 shows the CPU Queue Size and CPU Utilization of one CP, when the buffer size is 150. Notice that the CPU presents a bottleneck situation for a certain period of time. This is due to the fact that under heavy load conditions (i.e., around the vicinity of the peak traffic rate in Figure 3.7), the maximum capacity of the buffer is achieved. During such period of overload, the CPU utilization rises to 100%. This happens for a simulation time of approximately 92 to 158 seconds, and 350 to 408 seconds. Thus, it is during such overload periods that for instance (1) congestion control mechanisms could be used in order to regulate the incoming traffic rate, or (2) that load-sharing mechanisms could be employed in order to route the exceeding traffic rate to a mated SCP over the token ring, or to another distant SCP via the signaling network.

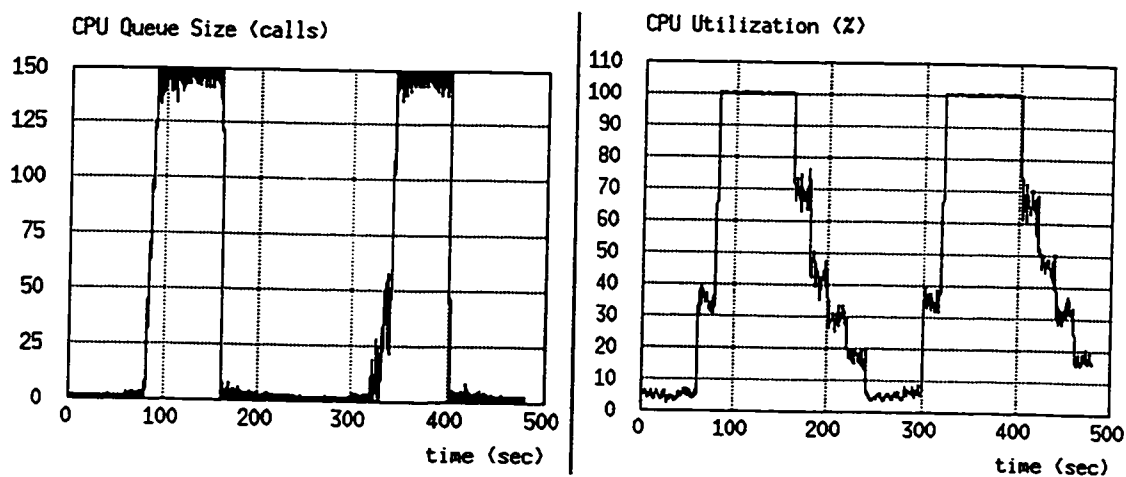


Figure 4.9 CPU Queue Size and CPU Utilization

Figure 4.10 presents the IN calls response time for a buffer size of 150. Notice that most calls are processed below 1.0 second, as mentioned earlier. In fact this figure depicts three levels of response time. The first level is around 50 msec (from simulation results). It is achieved when the SCP is able to handle the traffic rate generated by the SSPs. During the first level of response time, notice that the CPU is well under its maximum capacity queue size of 150, and utilization of 100% (see Figure 4.9). The second and third level of response time happens under the overload periods, hence when the CPU becomes the bottleneck entity. During the overload periods, the response time increases dramatically. This is attributed to the fact that the buffer is filling up and approaches its maximum limit of 150. The second level and third level of response time are around 0.5 seconds and 1.0 second, respectively. The reason for the latter two levels of response time is as follows: IN calls requiring zero visit (i.e., 75% of IN calls) fall in the 0.5 seconds level, whereas IN calls requiring one visit (i.e., 25% of IN calls) fall into the 1.0 second level. We also depict in Figure 4.10 a zoom-in view of the IN calls response time, due to the fact that the density of points in the lower figure of Figure 4.10 makes it hard to understand and see the details of the response time graph. From now on, we do not present anymore zoom-in of response time graphs.

Figure 4.11 presents the SCP throughput (i.e., aggregate of the three CP throughput). The maximum achievable SCP throughput is around 725 calls per second (from the simulation results). Notice how the SCP reacts to the incoming traffic rate from the SSPs.

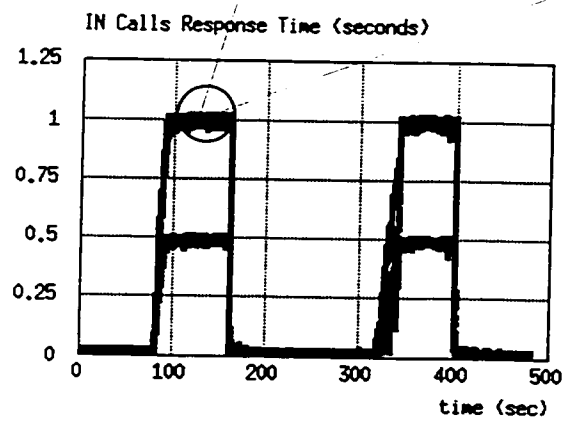
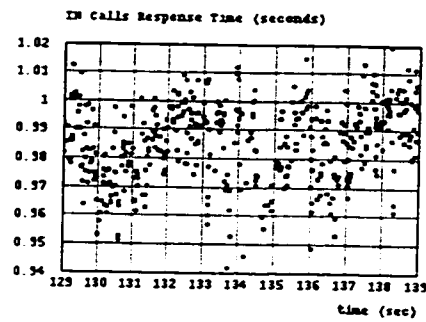


Figure 4.10 IN Calls Response Time

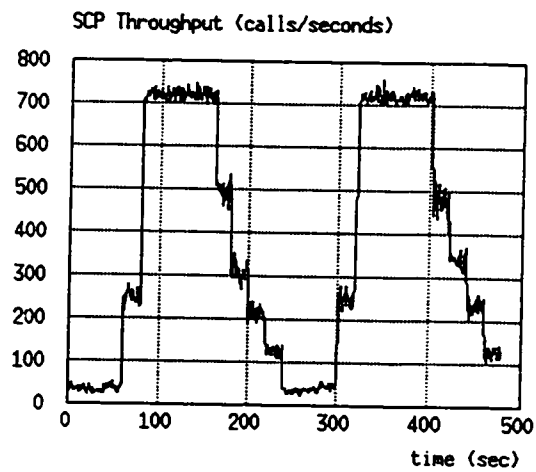


Figure 4.11 SCP Throughput for IN Call Scenario 1

It can be noted that there is a price to pay when using this IN call scenario. In such a scenario we identified that the number of blocked/load-shared calls was somewhat relatively large (~ 3% of all IN calls during heavy traffic rate). Even though a buffer size of 125 and 150 guarantees the response time QoS, it does not provide the SCP throughput of 800 to 900 calls per second imposed on our SCP model. In terms of memory size, such IN call scenario states that the SCP must store 75% of SLPs in the MMDB, while the complement is stored in the DBDB. Therefore, each CP must have a main memory size of 360 Mbytes (i.e., in order to store 75% of all SLPs requiring 600 bytes). In order to provide better performance in terms of blocked/load-shared calls, SCP response time and SCP throughput, the CP MMDB needs to be increased. Indeed, more SLPs need to be stored in the MMDB, as opposed to storing them in the DBDB (in order to decrease I/O overhead). This is also the drawback of increasing the system's price, since main memory is more expensive compared to disk storage. Under such circumstances we proceed to study another IN call scenario, where more main memory is added to each CP, and where we hope to increase the performance of the SCP.

4.3.2 Simulation Results for IN Call Scenario 2

This section presents the simulation results for IN Call Scenario 2 (90%, 5%, 5%). Compared to scenario 1, this scenario states that each CP must have a MMDB size of 432 Mbytes (in order to store 90% of all SLPs). This is an increase of 216 Mbytes per SCP compared to the scenario 1. This could translate in an increase of several thousands or ten thousands of dollars in the overall system's price at the cost of better performance. Table 4.5 presents the same categories as Table 4.3.

Table 4.5 Simulation Results for IN Call Scenario 2

Buffer Size	0 to 0.5 seconds	0.5 to 1.0 second	1.0+ seconds	Blocked/Load-Shared
125	176273	4447	3940	2572
150	172961	7176	4504	2477
175	113077	64052	7492	2309
200	111414	66472	7639	2320
250	114974	63442	7335	2247
300	107549	65668	12834	1970

Notice from the histogram presented in Figure 4.12 that a buffer size of 125 and 150 seems to present acceptable results, with respect to the number of calls processed below 0.5 seconds. Beyond a buffer size of 150 the number of calls falling in the 0.5 to 1.0 second range increases dramatically. This causes a decrease in the number of calls falling in the 0 to 0.5 seconds range. Still this should not affect the 95% quantile response time since we are interested in observing the number of calls processed below 1.0 second. There are now 3940 calls falling in the 1.0+ second range for a buffer size of 125, as opposed to 0 in scenario 1. As we will see, this increase is due to the fact that some calls require to visit the DBDB twice. Therefore, these calls experience a longer response time compared to the ones that require either zero or one visit.

Table 4.6 presents the number of calls (in terms of percentage) that were processed below 1.0 second, and the number of calls (in terms of percentage) that were blocked/load-shared.

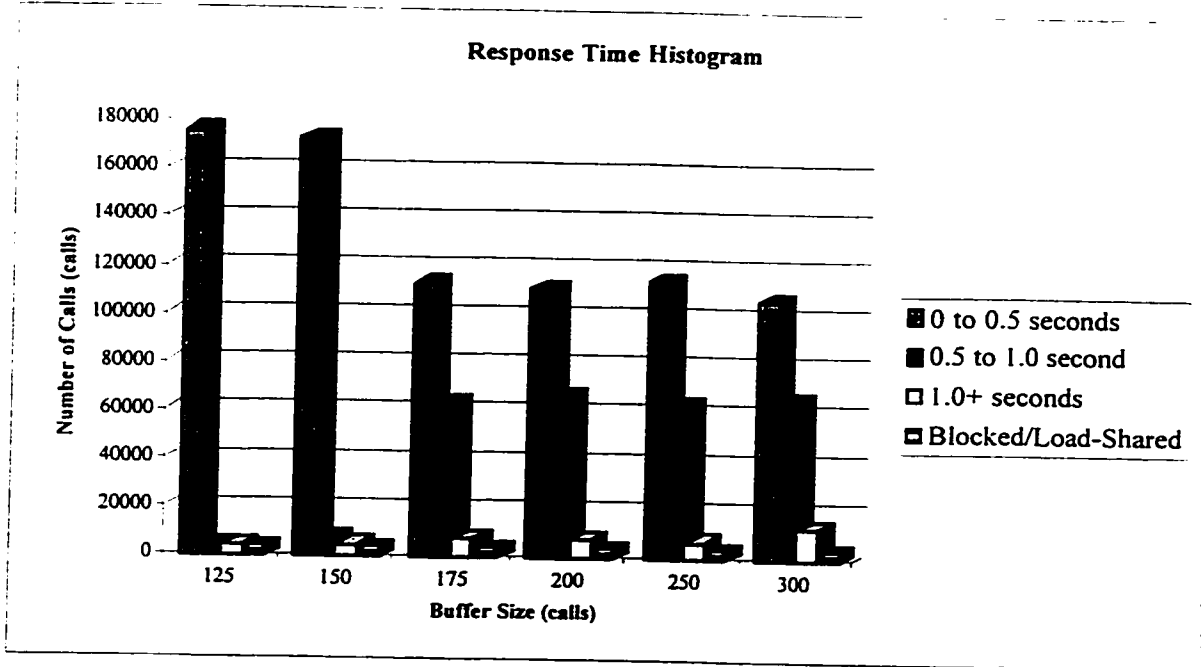


Figure 4.12 Response Time Histogram for IN Call Scenario 2

Table 4.6 95% Quantile Response Time for IN Call Scenario 2

Buffer Size	Number of Calls served below 0.5 second	Number of Blocked/Load-Shared Calls
125	96.5%	1.4%
150	96.3%	1.3%
175	94.8%	1.2%
200	94.7%	1.2%
250	94.9%	1.2%
300	92.1%	1.1%

The 95% quantile is guaranteed for a buffer size ranging from 125 to 250. Beyond 250, the 95% quantile value decreases since more calls fall into the 1.0+ second category (see Table 4.5). This is due to increasing the buffer size, hence increasing the response time of IN calls.

Figure 4.13 shows the CPU Queue Size and CPU Utilization when the buffer size is 150. Notice again that the CPU presents a bottleneck situation for a certain period of time. This is due to the fact that under heavy load condition, the maximum buffer capacity is achieved. This happens when the traffic rate reaches the 800 to 900 calls per second range. During such period, the CPU utilization reaches 100% utilization.

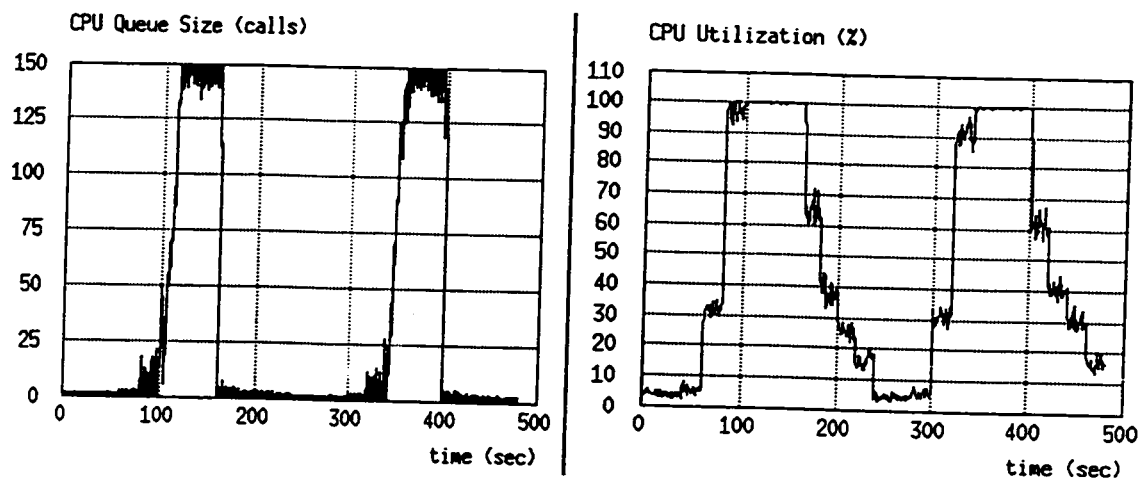


Figure 4.13 CPU Queue Size and CPU Utilization

Figure 4.14 presents the IN calls response time. Notice that there are four levels of response time in this figure, as opposed to three in scenario 1. When the traffic rate is fairly low (i.e., for which the CPU queue size is not saturated and utilization is fairly low) the response time is approximately 70 msec (from simulation results). The increase in response time happens for a simulation time period of approximately 100 to 160 seconds, and 356 to 400 seconds (see Figure 4.14). During such period, IN calls requiring zero visit are processed in about 0.5 seconds, whereas calls requiring one visit and two visits are processed in about 1.0 second and 1.5 seconds, respectively.

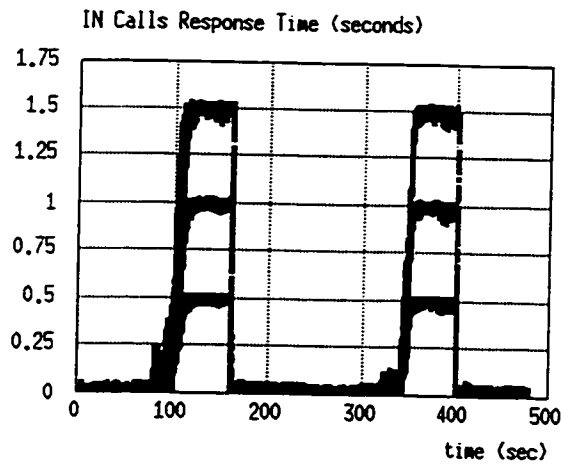


Figure 4.14 IN Calls Response Time

Figure 4.15 presents the SCP throughput. The maximum achievable SCP throughput for this scenario is approximately 800 calls per second, an increase of 75 calls per second compared to scenario 1. Recall that in chapter 3, we mentioned that our SCP model should guarantee a throughput ranging from 800 to 900 calls per second.

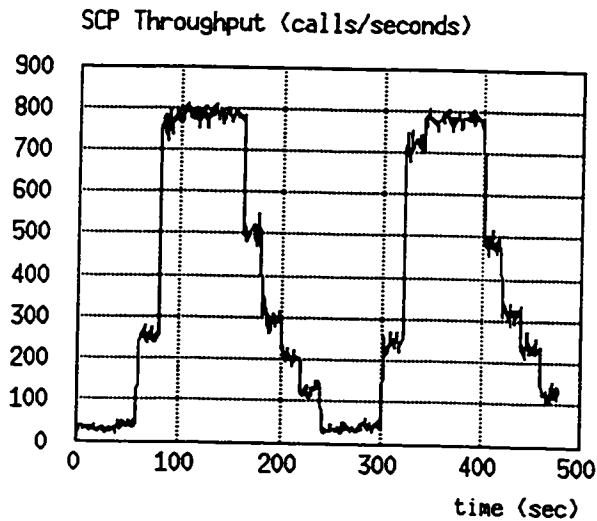


Figure 4.15 SCP Throughput for IN Call Scenario 2

This scenario provides better quality of service. By this we mean that less users are left retrying for the IN service, since the number of blocked/load-shared calls decreases. We also noticed that the number of blocked/load-shared calls decreases as the buffer size increases (from 2572 to 1970). This can be explained as follows: by letting some IN calls suffer longer response time (i.e., since the buffer size is increased), more calls can be admitted into the system, thus decreasing the number of blocked/load-shared calls. But this also causes a drawback in terms of response time QoS; an increase in buffer size surely reduces the number of blocked/load-shared calls but increases the response time of all IN calls. We now proceed to study the last IN call scenario, in which more main memory is added to each CP. For instance, evaluating an IN call scenario in which the number of blocked/load-shared calls approaches zero would be suitable, while respecting the SCP throughput and SCP response time requirements.

4.3.3 Simulation Results for IN Call Scenario 3

This section presents the simulation results for IN Call Scenario 3 (95%, 5%, 0%). We noticed in the last two scenarios that the number of blocked/load-shared calls (approximately 3% for scenario 1, and 1.5% for scenario 2) had to be reduced. We saw that by adding more main memory in each call processor, the performance in terms of SCP throughput and blocking probability significantly improved. We also noticed that the calls requiring two visits to the DBDB suffered longer response time. In this scenario, the CP stations require to have a MMDB that can hold 95% of all SLPs. Therefore, each CP must have a MMDB of approximately 456 Mbytes. Table 4.7 presents the same categories as in the last two scenarios.

Table 4.7 Simulation Results for IN Call Scenario 3

Buffer Size	0 to 0.5 seconds	0.5 to 1.0 seconds	1.0+ seconds	Blocked/Load-Shared
125	189631	1970	0	263
150	188755	2912	60	164
175	175063	15776	803	120
200	160382	29831	1539	95
250	160494	30210	1601	69
300	155879	34709	2086	0

Figure 4.16 depicts the response time histogram of Table 4.7. The histogram shares the same cell divisions as in the past histograms. Notice that the first range of response time (0 to 0.5 seconds) dominates for all buffer size values. The other range of response time are fairly negligible, especially the number of blocked/load-shared calls.

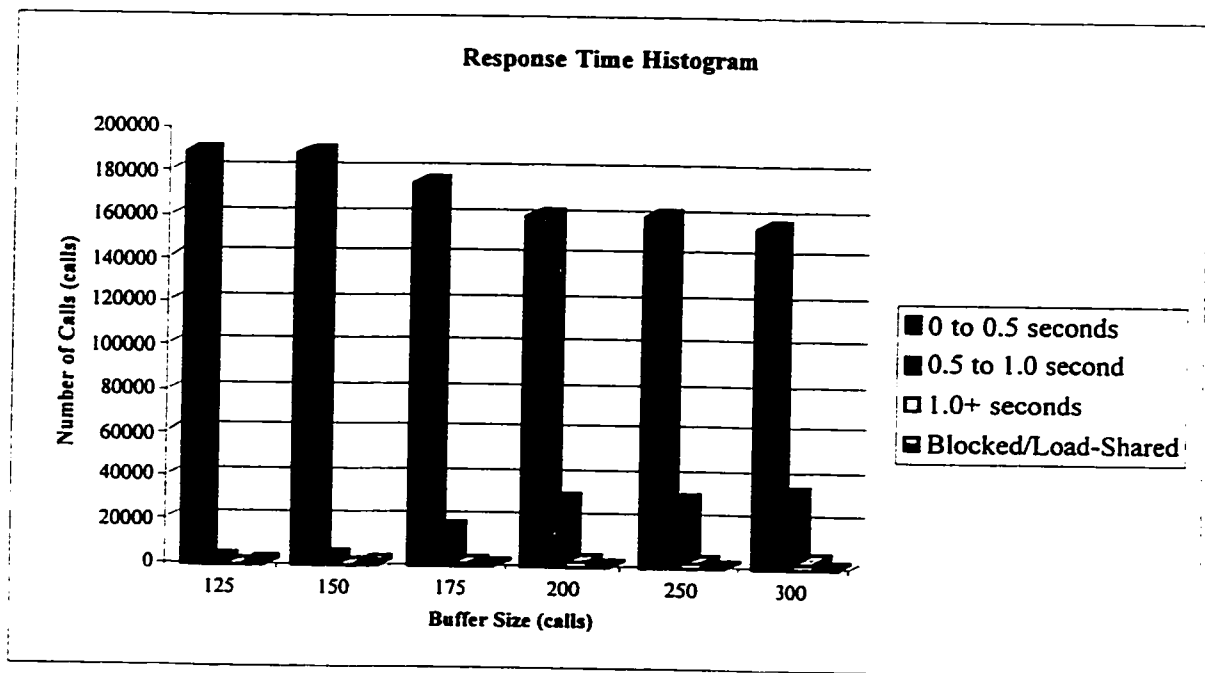


Figure 4.16 Response Time Histogram for IN Call Scenario 3

Table 4.8 presents the number of calls (in terms of percentage) that were processed below 1.0 second, and the number of calls (in terms of percentage) that were either blocked/load-shared.

Table 4.8 95% Quantile Response Time for IN Call Scenario 3

Buffer Size	Number of Calls served below 1.0 second	Number of Blocked/Load-Shared Calls
125	99.9%	0.1%
150	99.9%	0.09%
175	99.5%	0.06%
200	99.1%	0.05%
250	99.1%	0.04%
300	98.9%	0%

Notice from Table 4.8 that the 95% quantile response time value is guaranteed for a buffer size value ranging from 125 to 300. Notice that under this scenario the number of blocked/load-shared calls decreases from 263 to 0, for the buffer size values simulated. Hence, the number of blocked/load-shared calls represents a very small percentage of all IN calls generated.

Figure 4.17 and Figure 4.18 present the CPU Queue Size and CPU Utilization for a buffer size of 150 and 300, respectively. The small overload periods in Figure 4.17 arise under heavy traffic rate, which causes the CPU queue size to saturate. During the overload periods, the CPU utilization achieves 100%. Figure 4.18 presents similar characteristics as Figure 4.17. One difference can be mentioned. From Table 4.7 we can see that the number of blocked/load-shared calls is 0 for a buffer size of 300. Therefore, under a buffer size of 300 and peak traffic rate, there is no overload periods (i.e., no calls are blocked or need to be routed to another SCP), even if the CPU queue size increases at a very high value, and the CPU utilization reaches 100%.

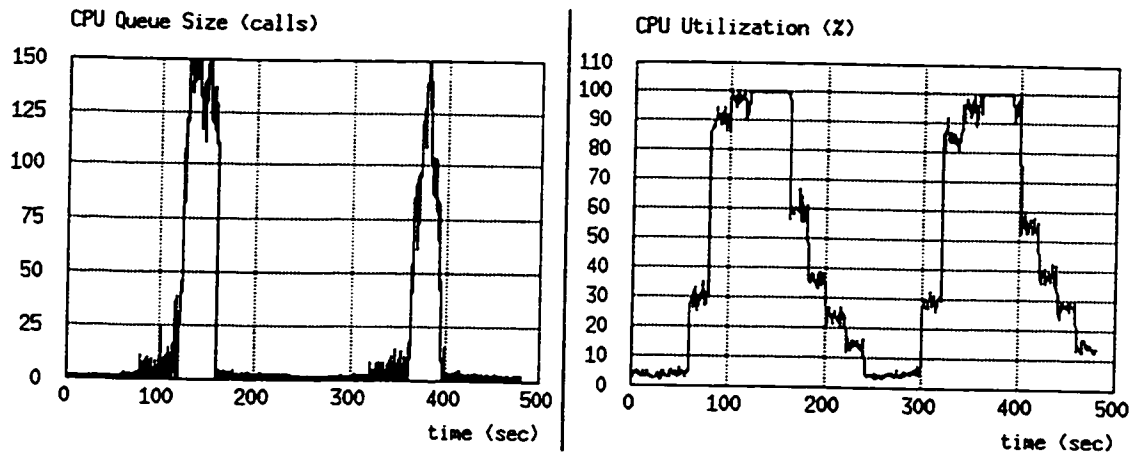


Figure 4.17 CPU Queue Size and CPU Utilization (Buffer Size = 150)

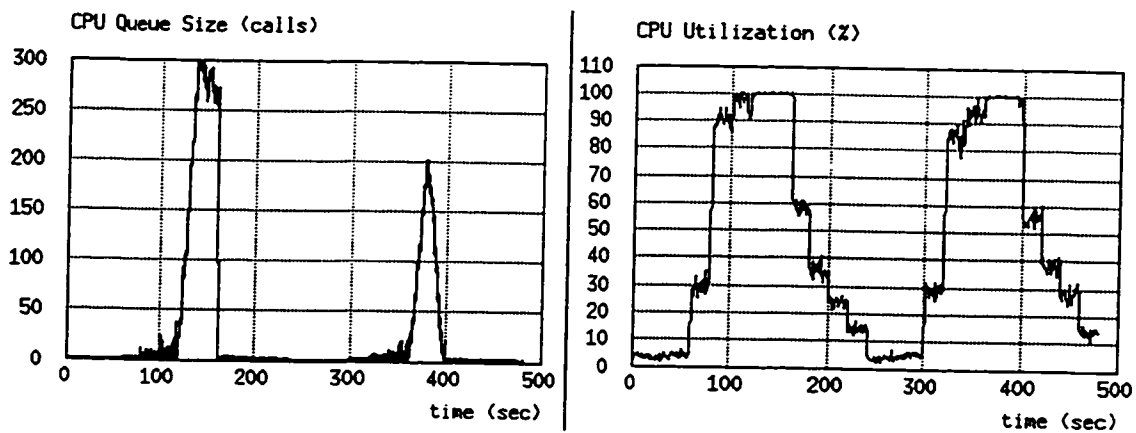


Figure 4.18 CPU Queue Size and CPU Utilization (Buffer Size = 300)

Figure 4.19 and Figure 4.20 presents the IN calls response time for a buffer size of 150 and 300, respectively. For both figures, we can identify three levels of response time. The first response time level is around 70 msec (from simulation results). During the overload periods of Figure 4.19 (from approximately 117 to 167 seconds) the calls requiring zero visit are processed with a response time of approximately 0.45 seconds to 0.5 seconds, whereas the calls requiring one visit are processed with a response time of

0.9 seconds to 1.0 second. Figure 4.20 follows the same shape of Figure 4.19. The only difference is the response time values under a buffer size of 300. The response time level for calls requiring zero visit are processed in approximately 0.75 to 1.0 second, whereas the calls requiring one visit are processed in approximately 1.4 to 1.8 seconds. Notice from these two figures that the response time values are the same for the simulation time ranging from approximately 362 seconds to 400 seconds. This is due to the fact that the

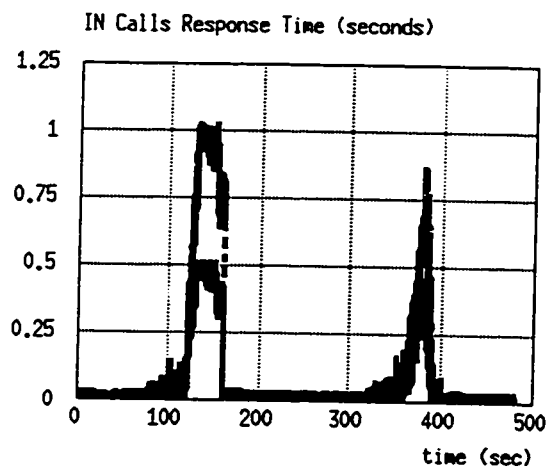


Figure 4.19 IN Calls Response Time (Buffer Size = 150)

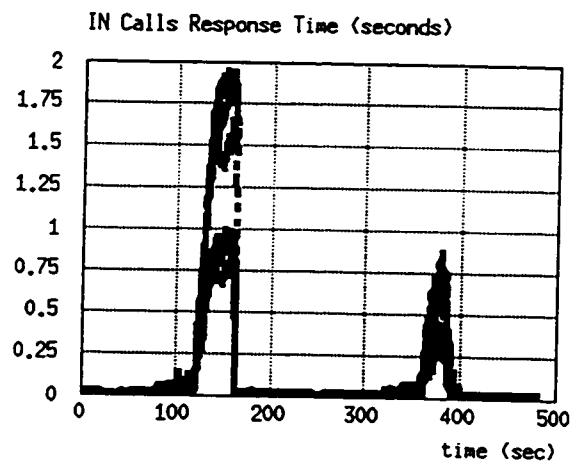


Figure 4.20 IN Calls Response Time (Buffer Size = 300)

CPU queue size is under its maximum capacity (see Figure 4.17 and Figure 4.18). Figure 4.18 clearly identified this, since the CPU queue size reached a value of approximately 200 calls, even if the maximum buffer size was 300 (for a simulation time period ranging from 362 to 400 seconds). It is also clear that during such period, calls requiring one visit suffer a longer response time than calls requiring zero visit.

Notice from Table 4.7 and Table 4.8 that an increase in buffer size (from 150 to 300) reduces the number of blocked/load-shared calls, up to a value of 0. This is somewhat good for users who access IN services, since they are left retrying less. But this decrease creates a performance drawback. There is still a price to pay for a small number of blocked/load-shared calls. The price to pay is that the SCP response time increases as the buffer size increases, even if it does not affect the 95% quantile response time requirement imposed on our model. For instance, if a buffer size of 150 is chosen, the number of blocked calls is 164, whereas the number of calls exceeding one second is approximately 0.1%, with a maximum response time of approximately 1.0 second. But for a buffer size of 300, the number of blocked calls is 0, whereas the number of calls exceeding one second is 1.0%, with a maximum response time of approximately 1.8 seconds. Therefore, there is a tradeoff between buffer size, blocking probability and response time. The buffer size value chosen will definitely depend on the users' satisfaction, since it is a performance metric critical to users perception and opinion of IN services. It is known that the longer the response time is, the more the users will become impatient. This will cause users to hang up and retry again, causing the performance of the SCP to degrade (and eventually the whole network). In chapter 3, we presented some issues concerning the impact of service response time versus users' perception and opinion of IN services.

Figure 4.21 presents the SCP throughput. The maximum achievable SCP throughput is around 870 calls per second (from the simulation results). This gives an increase of 70 calls per second compared to scenario 2, and an increase of 145 calls per second compared to scenario 1. This throughput value is within the required range imposed on our SCP model. The figure follows the same shape as the CPU queue size and CPU utilization. Thus, as the CPU utilization and queue size increases and decreases, the SCP throughput increases and decreases.

Figure 4.22 presents the DBDB utilization. We earlier mentioned that the CPU was a bottleneck entity, since it presented some overload periods where the CPU queue size was saturated. Figure 4.22 is shown in order to give an actual view of the disk-based database utilization, and to see if it is not a secondary source of bottleneck. Notice that the DBDB is not a bottleneck entity, since its utilization is well under 100%.

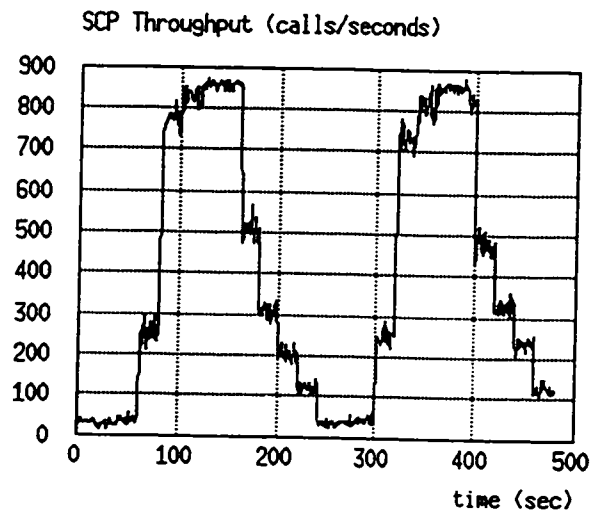


Figure 4.21 SCP Throughput for IN Call Scenario 3

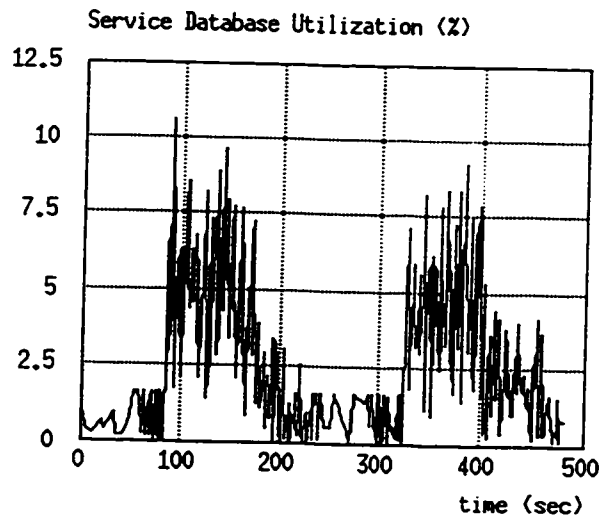


Figure 4.22 Disk-based Service Database Utilization

We conclude this first set of simulations by mentioning that acceptable QoS in terms of SCP response time, SCP throughput and blocking probability were obtained. All the performance metrics investigated in this scenario adhere to the requirements imposed on our SCP model. Recall that these requirements are usually set by the equipment vendors, network operators and users' perception of IN services. This last scenario gives performance as expected, even though some activities mentioned in chapter 3 were not included (i.e., present in the SCP). This first set of simulations was based under the IN call processing activity only. As another requirement for the evolution of the IN, the SCP architecture is required to not only encompass the IN call processing activity, but to also embody and accommodate activities such as SCP call sampling and polling (identified in chapter 3). These latter activities will definitely create another performance issue challenge, when introduced in the SCP model. The next section (i.e., second set of simulations) deals with the performance impact that results when the *SCP call sampling activity* is introduced in configuration 1. It is known that the SCP call sampling activity is

a very expensive operation. It requires enormous CPU processing power. Therefore, the SCP call sampling activity should not be neglected when evaluating the performance of the SCP.

4.4 Second Set of Simulations

This section analyses the results for the second set of simulations. The first set of simulations was primarily focused on assessing the performance of the SCP when the IN call processing and other house-keeping activities were present in the SCP. We found out that IN Call Scenario 3 provided acceptable performance in terms of SCP response time, SCP throughput, and blocking probability. The main difference between this second set of simulations and the first one is the introduction of the *SCP call sampling activity*. The reader is referred to section 3.3.2 and section 4.2 for a full description of the SCP call sampling activity. Its function, operation and how it affects the system's configuration was mentioned. With respect to the physical SCP architecture (identified in section 4.2, Figure 4.5), the introduction of the SCP call sampling activity will not affect configuration 1. The only difference lies in the fact that the sampling disk-based database will be used in this second set of simulations. This database will be used to store the IN data records generated by the SCP call sampling activity.

Figure 4.23 shows the truncated normal distributions that are used to model the IN call processing activity and SCP call sampling activity. The first truncated normal distribution, with a mean of 3.33 msec and a standard deviation of $\pm 6\%$, is used to model the IN call processing activity. It holds the same characteristics as in the first set of simulations (see section 3.6 and section 4.3). The second truncated normal distribution

has a mean of 16.65 msec and a standard deviation of $\pm 6\%$. This distribution is used to model the SCP call sampling activity. The truncation of each distribution (i.e., lower bound and upper bound values) is identified by the intersection of the *dashed* line to *full* line.

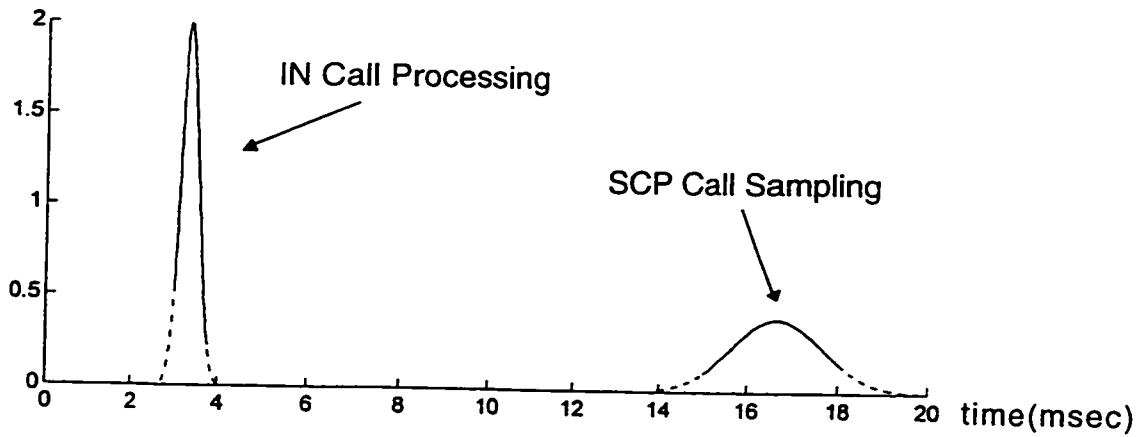


Figure 4.23 IN Call Processing and SCP Call Sampling Distribution

We now proceed to evaluate the simulation results of three different *SCP call sampling scenarios*. Each sampling scenario is associated to a *sampling rate* value. This rate identifies (in terms of percentage) the number of IN calls that the SCP will sample. For all sampling rate values (excluding 100%), the sampling activity is performed at random, based on a Bernoulli distribution with parameter *sampling rate*. The sampling scenarios investigated are identified in Table 4.9

Table 4.9 SCP Call Sampling Scenarios Simulated

Sampling Scenario	Sampling Rate
1	100 %
2	10%
3	5%

4.4.1 Simulation Results for Sampling Scenario 1

This section presents the simulation results when introducing a sampling rate of 100% in configuration 1. Therefore, this case assumes that all processed IN calls are sampled. Table 4.10 is presented in a five column format. The first column identifies the buffer size of each call processor. The next three columns (column 2 - 4) identify the number of IN calls that were successfully served under three range (or category) of response time. The first range of response time is delimited from 0 to 0.5 seconds. The second is from 0.5 to 1.0 second, and the third range of response time is from 1.0 second and beyond (1.0+ seconds). The fifth column identifies the number of IN calls that were not served by the SCP. We earlier assumed that an IN call was either *blocked* (i.e., not admitted by the SCP) or *load-shared* (i.e., routed to another SCP), unless stated otherwise.

Table 4.10 Simulation Results of Sampling Scenario 1

Buffer Size	0 to 0.5 seconds	0.5 to 1.0 seconds	1.0+ seconds	Blocked/Load-Shared
125	8555	1113	57351	41978
150	8175	1030	58015	41780
175	8028	1144	58048	41787
200	7309	1219	58918	41414
250	6931	1145	59678	41509
300	5779	1111	61385	41826

Table 4.11 presents the number of calls (in terms of percentage) that were processed below 1.0 second. Hence, this value represents our 95% quantile response time value. From the values listed in the table, we can see that the 95% quantile response time is not guaranteed. This is due to the fact that the number of calls falling in the 1.0+ seconds category and blocked/load-shared category is enormous.

Table 4.11 95% Quantile Response Time for Sampling Scenario 1

Buffer Size	Number of Calls served below 1.0 second
125	8.9%
150	8.4%
175	8.4%
200	7.8%
250	7.4%
300	5.6%

Figure 4.24 depicts the CPU Queue Size and the CPU Utilization for a buffer size of 150. The CPU queue size figure clearly identifies two periods where the CPU is overloaded. During these two overload periods, the queue size achieves its maximum capacity (i.e., the queue size saturates). Also, during these periods, the CPU utilization reaches 100%. This happens even when the traffic rate is considerably low (this is noticeable from the width of the overload periods). The first overload period lasts from approximately 75 seconds to 215 seconds. During that period, the traffic rate varies from approximately 200 calls per second, to a maximum of 900 calls per second, and back again to 150 calls per second (see Figure 3.7). The second overload period ranges from approximately 300 seconds up to 450 seconds. Because of these overload periods, the number of blocked/load-shared calls is extremely high, as one can see in Table 4.10. This has a direct effect to the fairly low percentage of calls being processed under 1.0 second.

Figure 4.25 shows the IN call response time for a buffer size of 150. Notice that there are three levels of response time. When the traffic rate is fairly low (i.e., for which the CPU queue size is not saturated and CPU utilization is fairly low) the response time

is approximately 50 msec (from simulation results). The second level and third level of response time happens under the overload periods (i.e., when the traffic rate is greater than what the SCP can handle). The second level of response time (for calls requiring zero visit) is approximately 2.5 seconds, whereas the third level of response time (for calls requiring one visit) is approximately 5.0 seconds. This huge increase in response time (compare to the first set of simulations) is due to the fact that all IN calls need to be sampled, and that the complexity of sampling an IN call demands a lot of processing time. In terms of users' satisfaction and users' perception of IN services, these response time values are inappropriate. It makes no sense to wait 2.5 seconds or even 5 seconds for a response of the SCP. Recall that the delay experienced by the user is not only the SCP response time, but the addition of the call setup establishment, signaling network delay, etc.

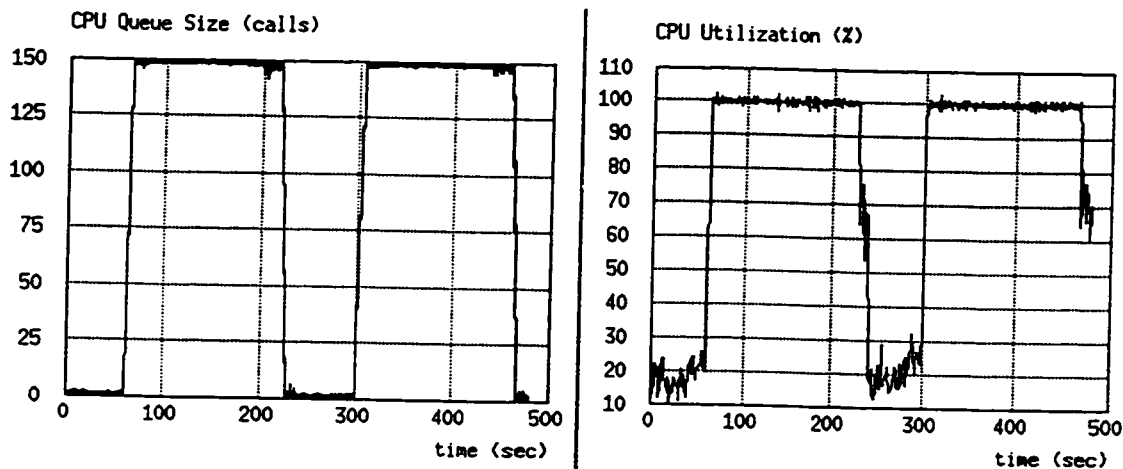


Figure 4.24 CPU Queue Size and CPU Utilization

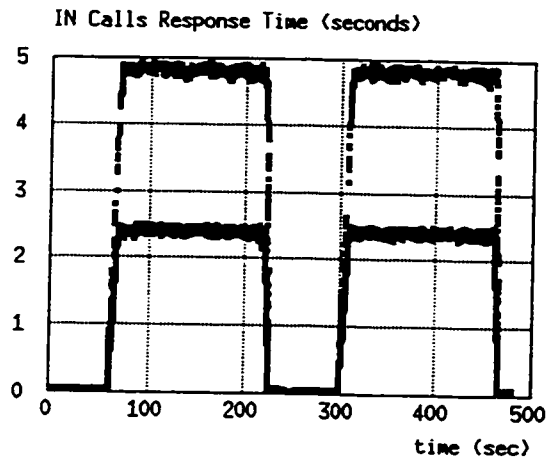


Figure 4.25 IN Calls Response Time

Figure 4.26 shows the SCP throughput. The figure shows that the maximum achievable throughput is approximately 175 calls per second. Therefore, during the overload periods depicted in Figure 4.24, 175 calls per second is an estimate of the achievable peak traffic rate into the SCP, when performing 100% sampling. This suggests that under a traffic rate generating less than 175 calls per second (i.e., if the traffic rate parameter values of Figure 3.9 could be decreased such that the peak traffic rate achieves 175 calls per second, instead of 900 calls per second), the SCP would operate under normal conditions. That would also mean that the number of blocked/load-shared calls would decrease, hence increasing the 95% quantile response time to a better value.

Therefore, the drawback to all of this is that even if one limits the traffic rate to the maximum achievable SCP throughput of 175 calls per second, and that 100% sampling is performed, the response time of IN calls are inappropriate values for the type of IN services under investigation.

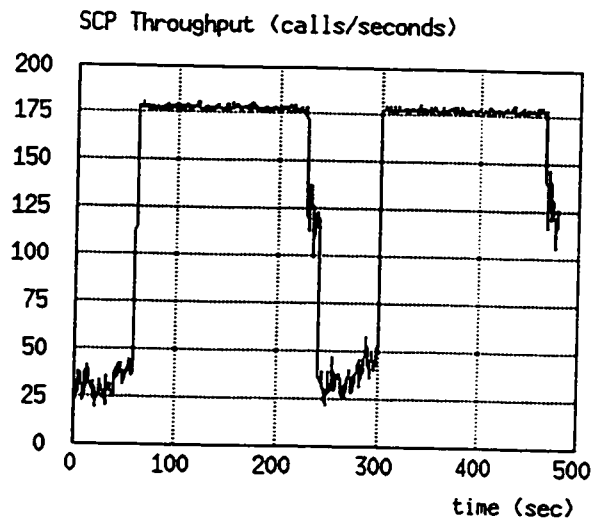


Figure 4.26 SCP Throughput for Sampling Scenario 1

Figure 4.27 depicts the sampling disk-based utilization. Notice that the disk system is not a bottleneck entity, since its utilization is under 100%. During heavy traffic rate, the utilization is around 10%. A low sampling disk utilization was expected, since the disk system was engineered not to create a potential bottleneck.

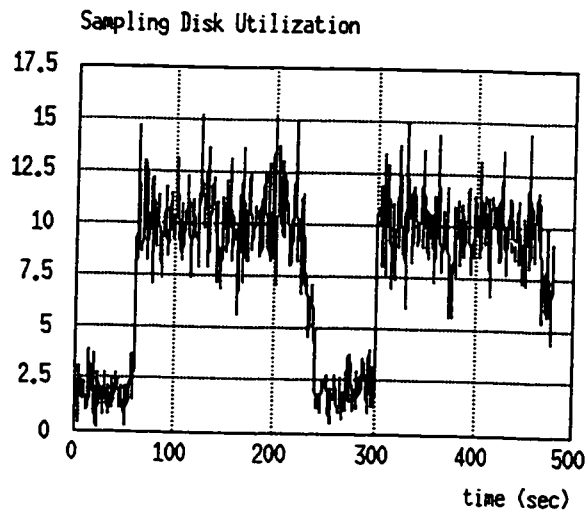


Figure 4.27 Sampling Disk Utilization

In this scenario we have shown that the SCP throughput and SCP response time requirements imposed on our SCP model were not guaranteed, or satisfactory. We saw that the CPU achieved 100% utilization for long simulation time. It is known that when the utilization approaches 100% for sustained periods of time, the integrity of the system is being compromised. In order to reduce the number of blocked/load-shared calls and the SCP response time, a change that can be made to this configuration consist of decreasing the SCP call sampling rate. Because of this, we now proceed to investigate the second sampling scenario, in which the sampling rate is decreased to 10%.

4.4.2 Simulation Results for Sampling Scenario 2

This section presents the simulation results obtained for a sampling rate of 10%. Table 4.12 and Table 4.13 presents the same categories as in the last sampling scenario.

Table 4.12 Simulation Results of Sampling Scenario 2

Buffer Size	0 to 0.5 seconds	0.5 to 1.0 second	1.0+ seconds	Blocked/Load-Shared
125	69366	86027	4700	10442
150	63458	91795	4916	10878
175	63548	92328	4912	10544
200	62381	88030	10522	10570
250	61646	6755	92914	10312
300	59955	5852	95624	10286

Table 4.13 95% Quantile Response Time for Sampling Scenario 2

Buffer Size	Number of Calls served below 1.0 second
125	91.1%
150	90.7%
175	91.0%
200	87.7%
250	39.9%
300	38.3%

Table 4.12 presents an improvement in terms of the number of blocked/load-shared calls, and Table 4.13 presents a significant improvement in the number of calls processed below 1.0 second, hence the 95% quantile response time. Still, with these buffer size it is not enough to guarantee the 95% quantile response time value imposed on our SCP model. In fact, the performance of the SCP are not better when the buffer size is increased, as one would think. There, as the buffer size increases, the number of calls to be sampled and admitted in the SCP increases, thus increasing the response time of all IN calls (recall that we now have activities with different complexities).

Figure 4.28 depicts the CPU Queue Size and CPU Utilization for a buffer size of 150. Notice that the overload periods are smaller in this scenario, compared to the last sampling scenario. This is attributed to the decrease of the sampling rate value. Still the CPU queue size saturates to its maximum capacity during overload periods.

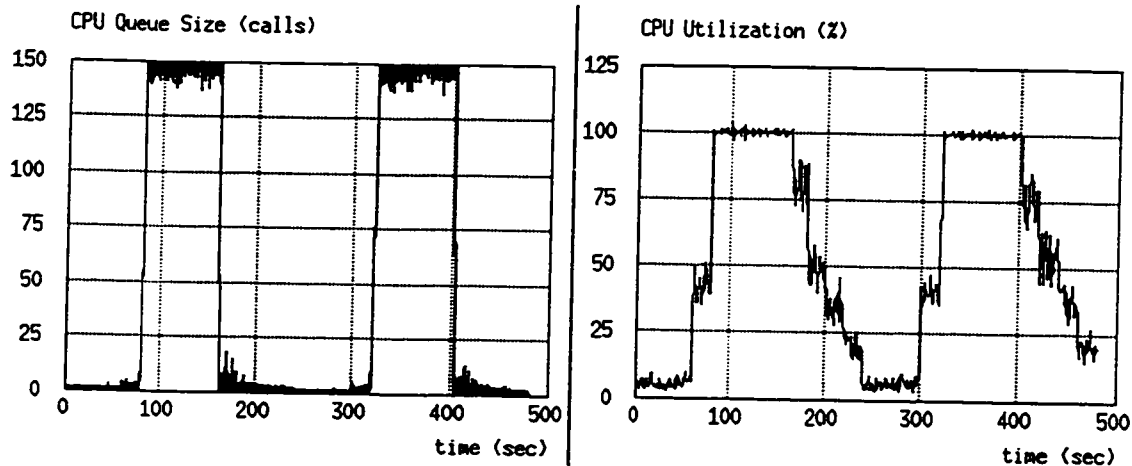


Figure 4.28 CPU Queue Size and CPU Utilization

Figure 4.29 depicts the IN calls response time for a buffer size of 150. Three levels of response time can be observed. The first level is around 70 msec (from simulation results). The other two levels are achieved under the overload periods (i.e., when the CPU queue size starts saturating). Calls requiring zero visit are processed in about 0.5 to 0.8 seconds, whereas calls requiring one visit are processed in about 1.2 to 1.5 seconds. In terms of users' satisfaction and opinion of IN services, these two latter levels of response time are acceptable, even if the 95% quantile response time is not guaranteed.

Figure 4.30 depicts the SCP throughput under a sampling rate of 10%. The maximum achievable SCP throughput under the peak traffic rate is approximately 630 calls per second (from simulation results). Hence, the SCP is not able to meet the required throughput range of 800 to 900 calls per second imposed on our SCP model. Still, the SCP would operate with satisfactory performance if the peak traffic rate generated by the SSP would be kept under 630 calls per second.

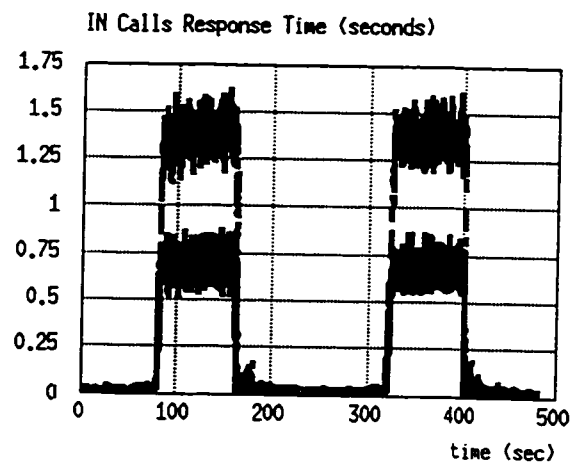


Figure 4.29 IN Calls Response Time

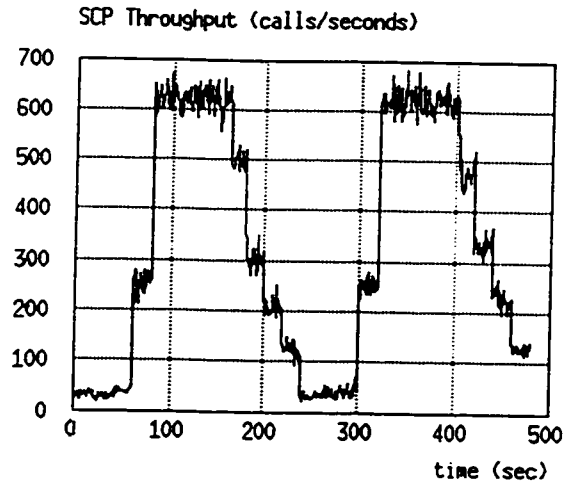


Figure 4.30 SCP Throughput for Sampling Scenario 2

This sampling scenario does not guarantee the two most important performance metrics; the SCP response time and SCP throughput, even though response time values of IN calls were acceptable. Therefore, we proceed to evaluate the last sampling scenario, in which the performance obtained should adhere to the requirements imposed on the SCP model.

4.4.3 Simulation Results for Sampling Scenario 3

This section presents the simulation results for a sampling rate value of 5%. Table 4.14 and Table 4.15 presents the same categories as in the last two sampling scenarios.

Table 4.14 Simulation Results of Sampling Scenario 3

Buffer Size	0 to 0.5 seconds	0.5 to 1.0 second	1.0+ seconds	Blocked/Load-Shared
125	173046	9307	97	2966
150	103974	75713	4132	3151
175	98630	80095	4223	2921
200	100073	79909	4187	2927
250	95191	82268	6191	2730
300	97013	17292	69745	2639

Table 4.15 95% Quantile Response Time for Sampling Scenario 3

Buffer Size	Number of Calls served below 1.0 second
125	98.3%
150	96.1%
175	96.1%
200	96.2%
250	95.2%
300	61.2%

As be can be seen from Table 4.15, this sampling scenario presents acceptable results with respect to the 95% quantile response time. The 95% quantile is guaranteed up to a buffer size value of 250.

Figure 4.31 and Figure 4.32 respectively depict the CPU Queue Size and CPU Utilization for a buffer size value of 150 and 250. Notice the overload periods where the CPU queue size saturates to its maximum value and the CPU utilization reaches 100%, even though the 95% quantile response time requirement is respected.

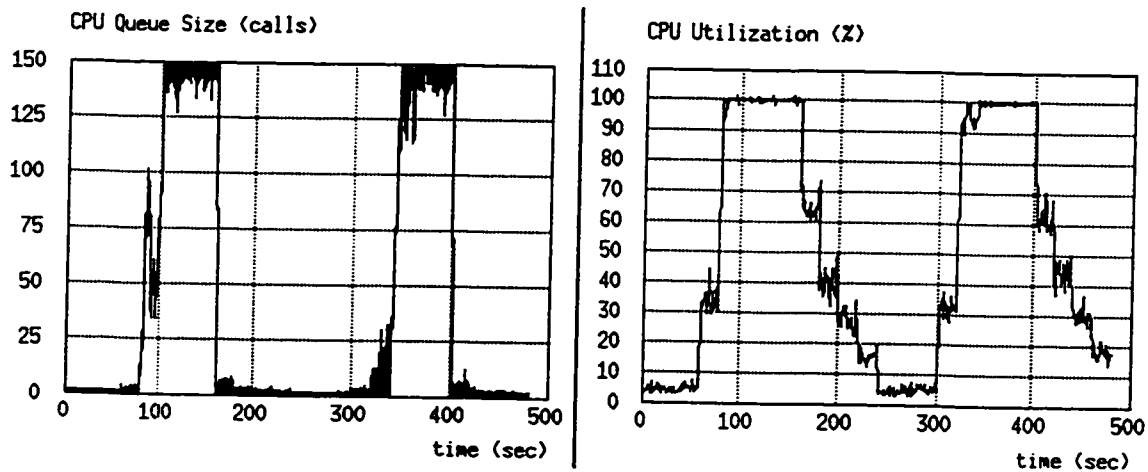


Figure 4.31 CPU Queue Size and CPU Utilization (Buffer Size = 150)

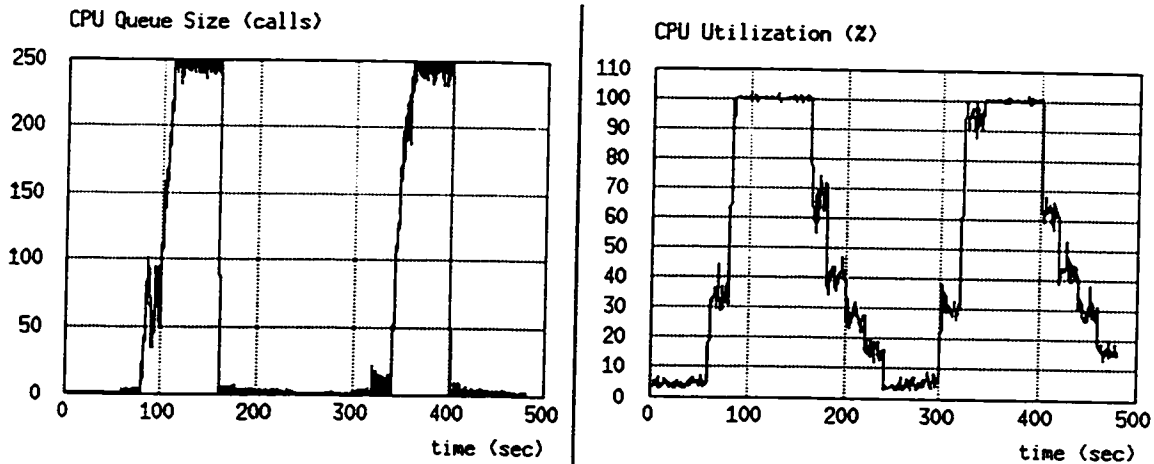


Figure 4.32 CPU Queue Size and CPU Utilization (Buffer Size = 250)

Figure 4.33 and Figure 4.34 respectively depict the IN calls response time for a buffer size of 150 and 250. Notice that there are again three levels of response time. The first level happens under normal operation of the SCP. During that period, the IN calls are processed in about 70 msec (from simulation results). During overload periods, the IN calls requiring zero visit are processed in about 0.5 to 0.6 seconds for a buffer size value of 150, and in about 0.8 to 1.0 second for a buffer size value of 250. For IN calls requiring one visit, calls are processed in about 1.0 to 1.2 seconds for a buffer size value of 150, and in about 1.75 to 2.0 seconds for a buffer size value of 250. In terms of users' satisfaction these response time values are acceptable for both buffer size. But, remember that in chapter 3 we mentioned that the maximum response time of a freephone service should be 1.0 second. Therefore, Figure 4.33 respects this condition, even if a small quantity of IN calls are processed beyond 1.0 second (approximately 2.2% of all IN calls). Figure 4.34 does not respect the maximum response time requirement. But, this will mostly depend on studies assessing the response time values that users can tolerate.

For instance, in some cases it would be profitable to increase a bit more the response time of some IN calls, in order to decrease the number of blocked/load-shared calls.

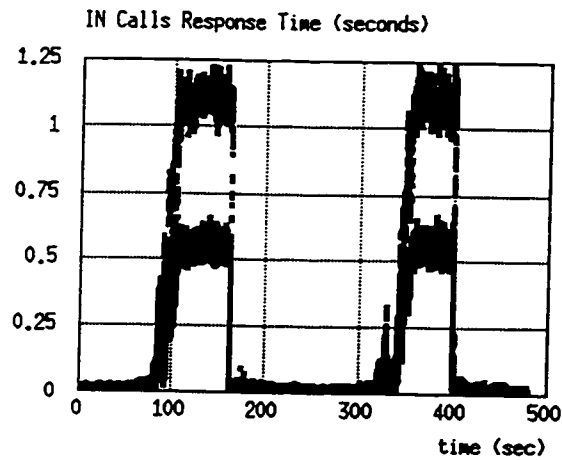


Figure 4.33 IN Calls Response Time (Buffer Size = 150)

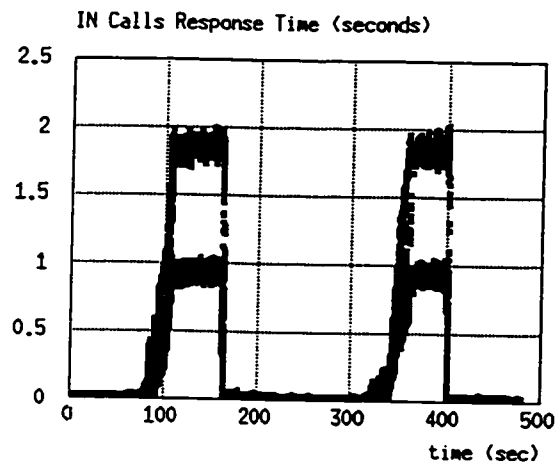


Figure 4.34 IN Calls Response Time (Buffer Size = 250)

Figure 4.35 respectively depicts the SCP throughput. The maximum achievable throughput is 800 calls per second. Therefore, 800 calls per second is an estimate of the achievable peak traffic rate into an SCP performing a sampling rate of 5%.

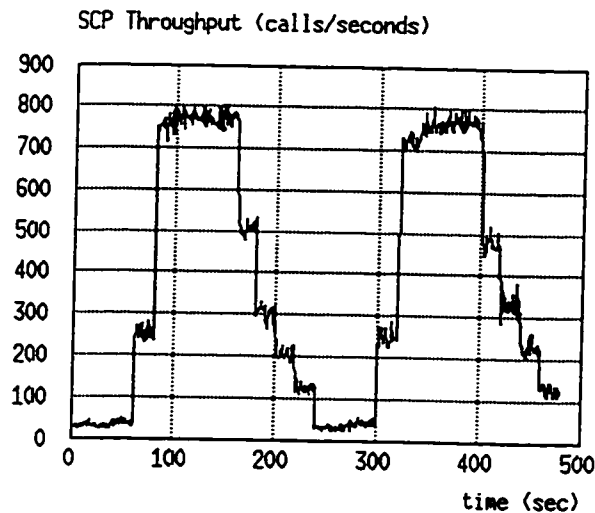


Figure 4.35 SCP Throughput for Sampling Scenario 3

4.5 Summary

Figure 4.36 depicts the SCP throughput versus the SCP call sampling rates investigated in the first and second set of simulations, under the IN call scenario (95%, 5%, 0%). We saw that the performance depended on the IN call scenario used. We found out that the IN call scenario (95%, 5%, 0%) provided the requirements imposed on our model. Notice that the SCP throughput decreases as the sampling rate increases. This is expected, since the SCP call sampling activity is complex and demand ample CPU processing power. In order to meet the QoS imposed on the SCP, we showed that our SCP model could only perform a maximum sampling rate of 5%, while processing IN calls. Beyond a sampling

rate value of 5%, we noticed that the SCP's performance in terms of SCP response time, SCP throughput, and blocking probability degrade dramatically. Figure 4.36 also gives us the maximum achievable throughput that will make the SCP operate under normal conditions, when the peak traffic rate does not increase beyond the specified throughput, and when the right buffer size value is chosen.

Figure 4.37 depicts the IN calls response time versus the SCP call sampling rate investigated in the first and second set of simulations, under the IN call scenario (95% 5%, 0%). The four curves depict the minimum and maximum response time for 0 and 1 visit, respectively (see legend). The response time values are for a buffer size of 150 calls. Notice that the response time increases in a non-linear fashion as the sampling rate increases. In order to meet the QoS requirements imposed on our SCP model, a maximum sampling rate of 5% can be achieved. In terms of users' satisfaction and users' perception of IN services, the response time values for a sampling rate of 5% or less are acceptable. But, as the sampling rate increases beyond 5%, the response time values become inappropriate. For example, it makes no sense for a user to wait 2.5 seconds or 5 seconds for a response of the SCP, under a sampling rate of 100%. It is known that the longer the response time is, the more users will become impatient. This will cause users to hang up and retry again, causing the performance of the SCP to degrade significantly.

Therefore, configuration 1 behaves as expected when (1) the IN call processing activity is assessed (first set of simulations), and (2) when the IN call processing activity and SCP call sampling activity are assessed (second set of simulations). Nevertheless, the first set of simulations provided fairly good results if only the IN call processing activity,

and other house-keeping activities are present in the SCP. Still, with a 5% SCP call sampling rate the SCP provided good performance.

We identified in chapter 3 that real-time access to statistics and information, by the customer, service provider and/or network operator is an ever increasing requirement for the evolution of SCP. In order to meet the market requirements, and to embrace that challenge, a solution is needed in order to increase the SCP call sampling rate (as we saw in this chapter) while respecting the QoS imposed on the SCP. This is due to the fact that a sampling rate of 5% is somewhat low, and limits the amount of potential customers, services providers and/or network operators in need of accessing sampling/statistical information. It is clear that the desired situation is when a SCP call

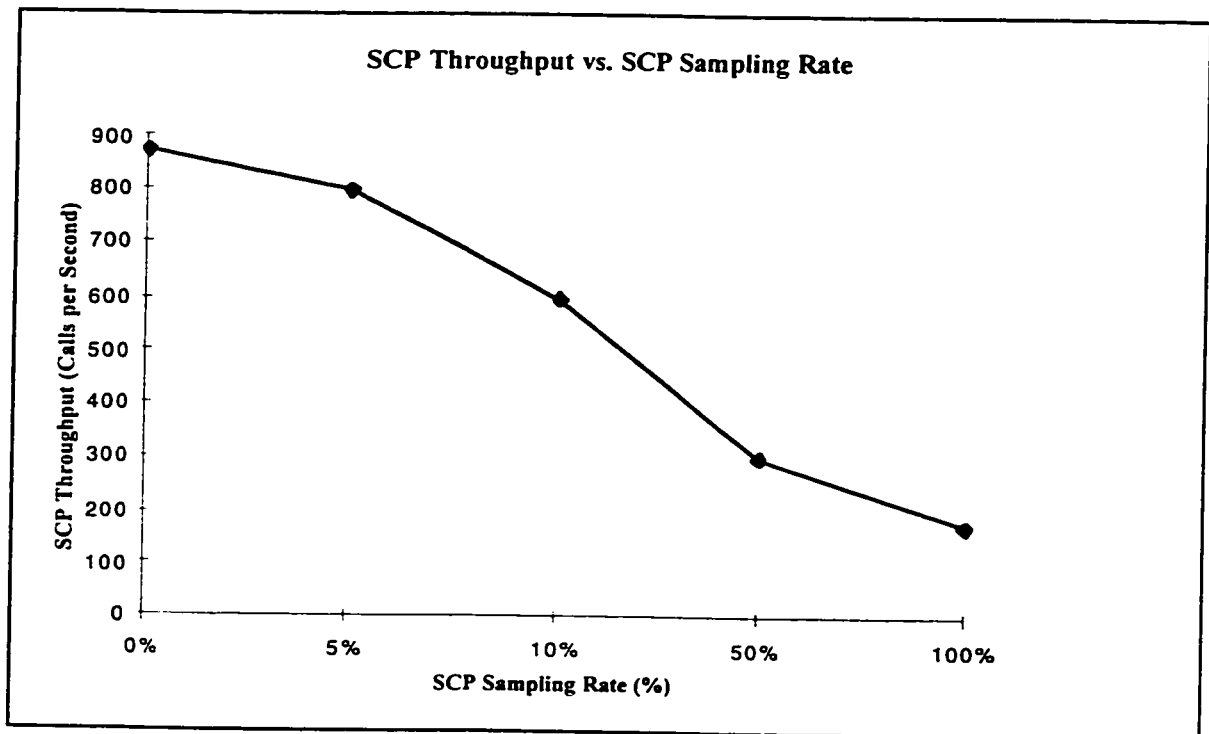


Figure 4.36 SCP Throughput versus SCP Call Sampling Rate

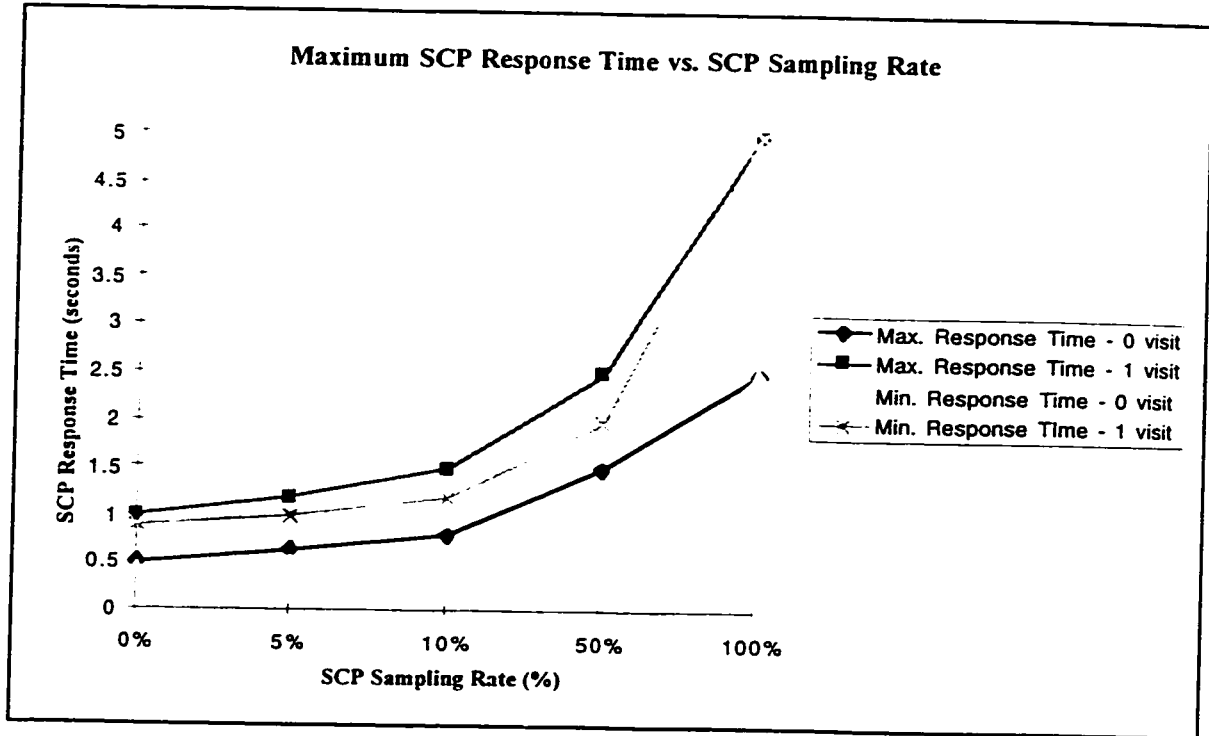


Figure 4.37 SCP Response Time versus SCP Call Sampling Rate

sampling rate of 100% can be achieved. We showed that it was not possible under configuration 1, because of low throughput and high response time values (unless limiting the SSP traffic rate). Therefore, in order to alleviate the situation where a maximum sampling rate of only 5% can be achieved, we propose a second configuration (i.e., configuration 2).

Chapter 5. Simulation Results /2

5.1 Introduction

This chapter provides a solution to the problems encountered in configuration 1. The proposed deployment option that we present in configuration 2 should in fact off-load the SCP call sampling activity performed at the SCP's CP stations. The resulting reduction in processing requirement at the CPs (due to off-loading the SCP call sampling activity) should therefore increase the SCP's sustainable IN call traffic workload, as well as improve the SCP response time, SCP throughput and blocking probability. Recall that in configuration 1 (i.e., second set of simulations) the SCP could only achieve the required QoS under a maximum sampling rate of 5%. This implies that, without off-loading the sampling activity (e.g., to an intermediate specialized processor), the SCP throughput requirement with 100% call sampling can not be achieved. From the simulations results of configuration 1, the performance measurements indicated that the SCP could sustain a peak traffic workload of only 175 calls per second when 100% sampling was performed. This traffic workload is well under the requirement of our SCP model, and is well below today's IN requirements. We have showed in chapter 3 that a real IN (i.e., Stentor's AIN) was submitted to more than 200 calls per second during busy hour periods. Recall that we are engineering the SCP so that it can handle a peak traffic workload ranging from 800 to 900 calls per second, in order to meet the forecasted market requirements.

5.2 Configuration 2

Configuration 2 is characterized by the addition of a specialized processor. This processor is introduced in each SCP node of the mated SCP pair. It is known as the *Data Processor* (DP), and was first introduced in chapter 3. Figure 5.1 shows a view of the mated SCP pair with their corresponding DP. The main purpose of the DP is to free the call processor stations (of the SCPs) from the sampling/statistical acquisition and compilation activities (i.e., creation of the IN data records), hence the SCP call sampling activity. We saw in chapter 4 that the SCP call sampling activity was a very expensive task in terms of CPU processing power. We showed that this activity, when performed in each call processor station, reduced the SCP's performance considerably in terms of QoS (i.e., response time, throughput and blocking probability).

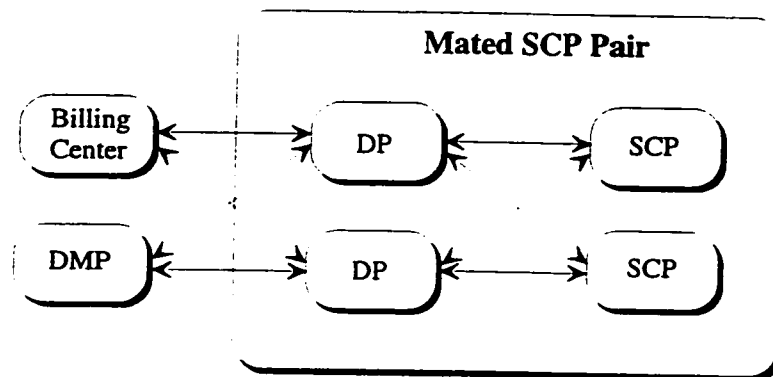


Figure 5.1 SCP Architecture for Configuration 2

In this configuration, a DP is inserted between each SCP and the external administrative systems (i.e., the DMP and the Billing Center), such that the only change to configuration 1 is the replacement of the DMP token ring interface entity, with the Data Processor entity. Figure 5.2 shows the change. The specialized processor (i.e., Data Processor) is

now interconnected to the dual token ring topology in the same manner as the three call processor stations are. Therefore, the Data Processor acts an intermediary processor residing between the call processor stations of the SCP and the external administrative systems.

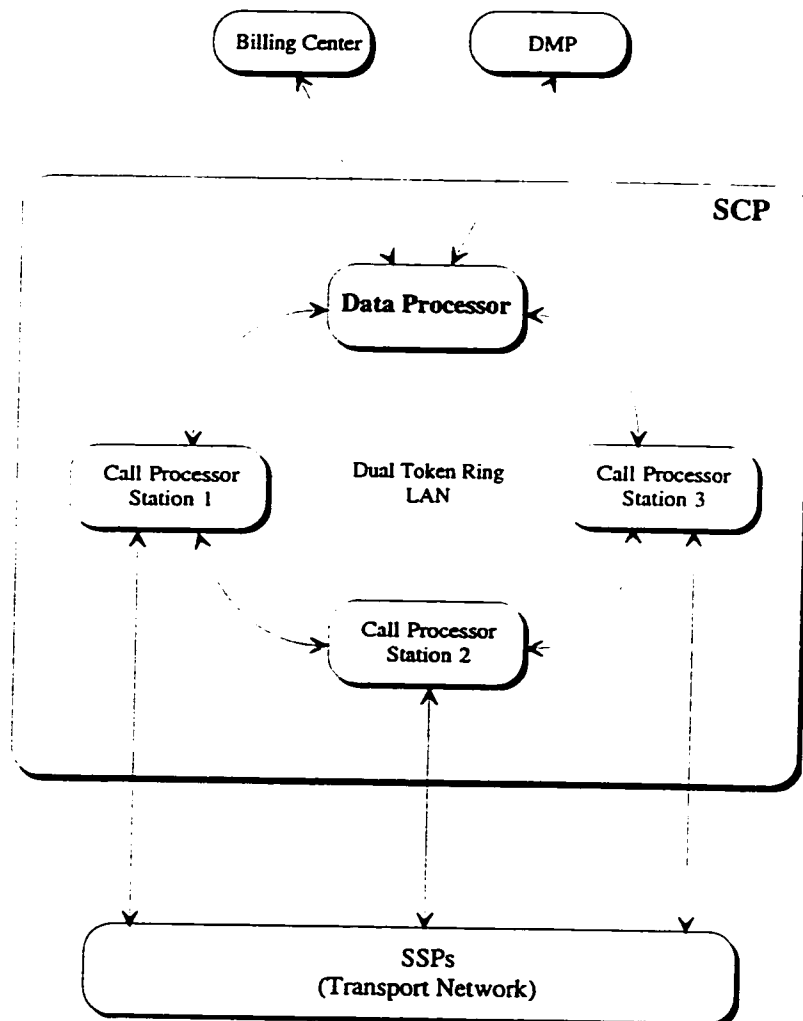


Figure 5.2 Service Control Point Model

We now show and explain the changes that are made to the queuing models of configuration 1, in order to obtain configuration 2. Figure 5.3 depicts the internal architecture of the Data Processor model. Recall that in configuration 1, the SCP call

sampling activity was performed immediately after the IN call was successfully served/processed. After the sampling was completed, the IN data record created was forwarded and stored in the sampling disk-based system of the CP station, for which it was later retrieved by the external DMP administrative system. The difference with configuration 2 is that after an IN call is served/processed by one of the CPs of the SCP (i.e., upon sending a response sent back to the SSP), a copy of the IN call is created and sent as a *data packet* to the Data Processor. *The data packet is sent over the token ring LAN*, from a call processor station to the Data Processor. Therefore, in configuration 2, no sampling is performed at the CPs. The creation of the IN data records is done exclusively at the DP. Each data packet sent could contain information on the IN calls such as freephone number dialed, time of day, billing data, number of retries, originating carrier, etc.

Upon their arrival to the DP, the data packets are picked up by the two *MAC modules* and are merged (buffered) in the *network interface unit module NIU-I2*. This network interface unit is similarly equivalent to the network interface processors of the CP stations. The NIU is used to perform pre-processing on the data packets (e.g., strip the header and trailer, decompression, etc.). The data packets then proceed to the *CPU Subsystem (main memory and CPU)* module where they are buffered before being sampled. They then wait for the CPU's attention. The buffering of data packets in main memory assumes that the IN data records will be processed on a first come first served basis. When the sampling requirement is over, every data packet will visit the sampling disk-based system. This represents the process of storing the IN data record into the disk system.

Polling requests from the DMP and Billing Center follow exactly the same logic flow as in Figure 4.5 of configuration 1. That is, polling requests are sent to the DP, where they are buffered at NIU-I1. Notice that in this configuration, the polling requests are not sent over the token ring to each CP of the SCP. This is shown in Figure 5.3. After the polling requests are buffered at NIU-I1, they proceed to the CPU subsystem. Again, we assume that the polling requests do not influence the DP performance. In response to the polling requests, the transfer of the IN data records from the sampling disk-based system would be buffered at NIU-O1 (i.e., for post-processing) before being sent to the external administrative DMP system.

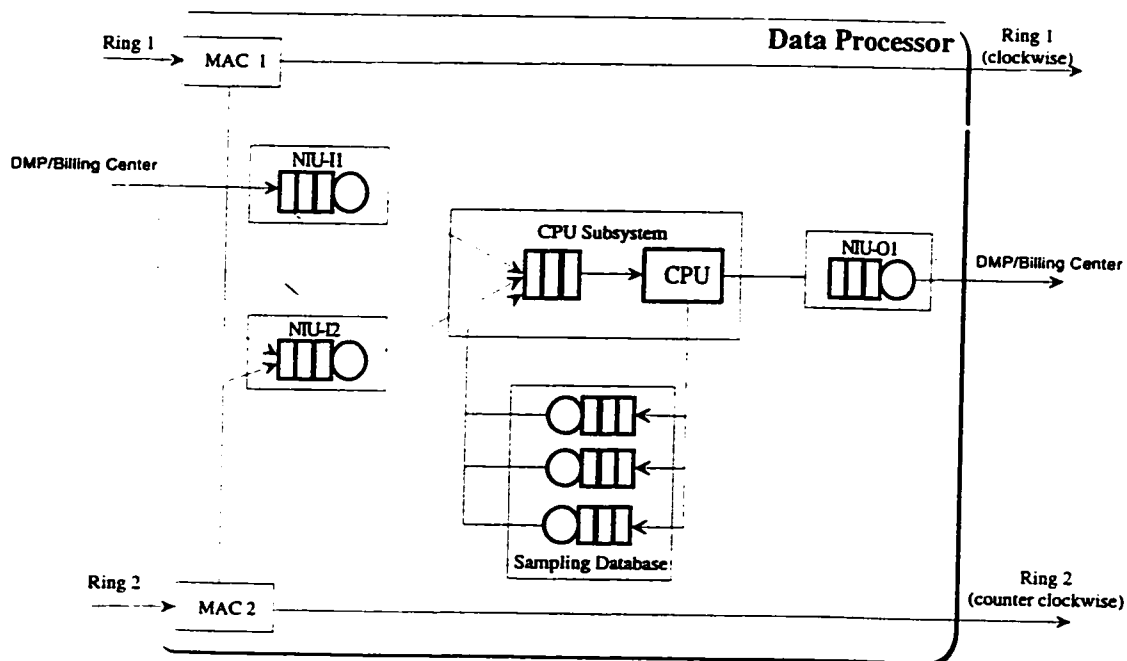


Figure 5.3 Data Processor Model

The DP effectively protects the call processor stations from the SCP call sampling and SCP polling activities. Since these activities are now performed at an intermediate

specialized processor, the queuing model of the call processor itself is simplified. It is shown in Figure 5.4. IN calls coming from the SSPs (via the CCSS7 network) follow the same path as illustrated and explained in Figure 4.5 of configuration 1. They are buffered at NIU-I1, where they are submitted to pre-processing. They then proceed to the CPU subsystem and the disk-based service database (if they require to visit the database). For every response sent back to the SSP (via NIU-O1), a copy of the call is made by the call processor (as mentioned previously). The copy is formatted into a data packet, and buffered at NIU-O2, before being sent to the DP (via the token ring).

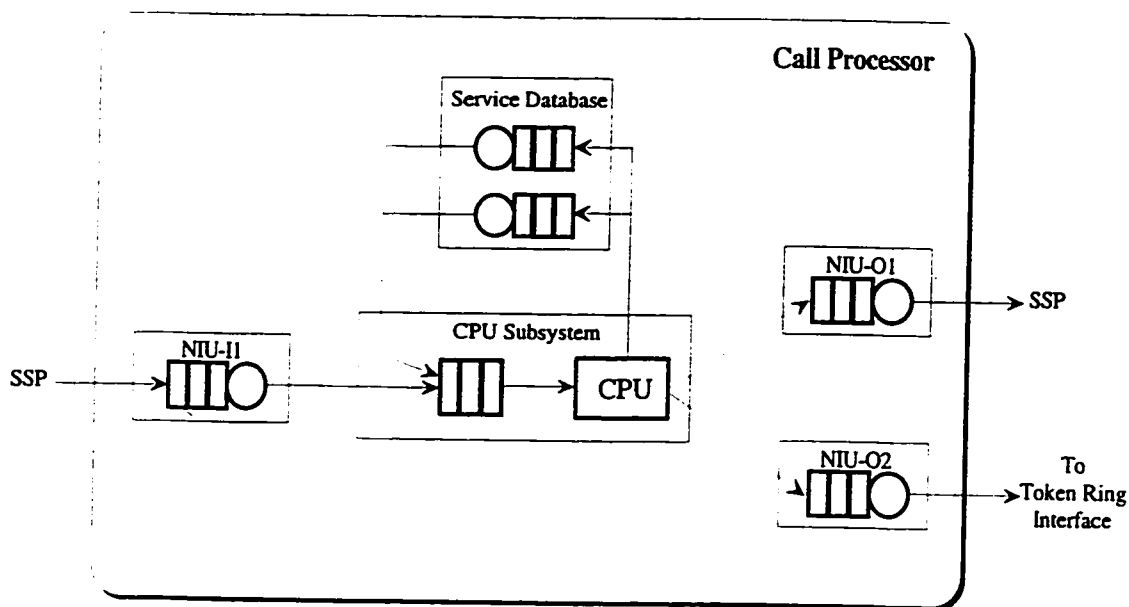


Figure 5.4 Call Processor Model

The addition of the DP entity introduces some additional simulation parameters. The additional parameters for configuration 2 are listed in Table 5.1. Also, all SCP parameters previously listed in Table 4.2 remains the same for configuration 2.

Table 5.1 Additional SCP Parameters for Configuration 2

Parameter	Value
1. Mean DP CPU service time	13.32 msec
2. Mean DP sampling disk service time	8.13 msec
3. Number of sampling disks	3
4. DP sampling rate	100%

The *mean DP CPU service time* parameter represents the processing time required to sample an IN call (i.e., to create an IN data record). The service time follows a truncated normal distribution with a mean of 13.32 msec. The lower bound and upper bound follow the shape of Figure 4.23. We also assume (unless stated otherwise) that the DP CPU is equivalent in terms of processing power to the one used in configuration 1. The *mean DP sampling disk service time* represents the time to transfer and store an IN data record from main memory to one of the sampling disk. The service time follows a truncated exponential distribution with a mean of 8.13 msec. The lower bound and upper bound are given as 0.06 msec and 16.26 msec, respectively. The *number of sampling disks* in each DP is 3. The *sampling rate* performed by the DP will be set to 100%, stating that all IN calls will be sampled.

5.3 First Set of Simulations

Table 5.2 is presented in a five column format. The first column identifies the buffer size of each call processor. The next three columns (column 2 - 4) identify the number of IN calls that were successfully served under three range (or category) of response time. The first range of response time is delimited from 0 to 0.5 seconds, whereas the second range

is from 0.5 to 1.0 second. The third range of response time is from 1.0 second and beyond. The fifth column identifies the number of IN calls that were not served by the SCP. We again assume that an IN call is either *blocked* (not admitted by the SCP) or *load-shared* (routed to another SCP). Again, we are only interested in obtaining the blocking probability, and to verify that the QoS requirements are guarantees (i.e., we are not concerned with the handling of blocked calls, or how they would be routed to another SCP). Table 5.3 presents the number of calls (in terms of percentage) that were processed below 1.0 second, and the number of calls (in terms of percentage) that were blocked/load- shared. The former quantity represents the 95% quantile response time and the latter quantity represents the blocking probability. Recall that we identified in chapter 3 that the response time values imposed on our model should guarantee that 95% of all IN calls be processed below 1.0.

Table 5.2 Simulation Results for Configuration 2

Buffer Size	0 to 0.5 seconds	0.5 to 1.0 second	1.0+ seconds	Blocked/Load-Shared
125	189534	22136	0	357
150	188370	28370	90	277
175	166493	23947	1218	127
200	163564	27465	1373	204
250	160080	29820	1540	197
300	155733	34200	2153	97

Table 5.3 95% Quantile Response Time for Configuration 2

Buffer Size	Number of Calls served below 1.0 second	Number of Blocked/Load-Shared Calls
125	99.8%	0.17%
150	99.8%	0.13%
175	99.3%	0.07%
200	99.2%	0.11%
250	99.1%	0.10%
300	98.8%	0.05%

The simulation results presented in Table 5.2 and Table 5.3 are very similar to the simulation results presented in section 4.3.3. Recall that the results of section 4.3.3 were obtained from the first set of simulations of configuration 1. The similarity between the results is due to the fact that the call processors in configuration 2 are subject to the same activities as in configuration 1 (i.e., IN call processing and house-keeping activities). In configuration 1 (first set of simulations) the SCP call sampling activity was omitted, while in configuration 2 it is present, but off-loaded to a specialized processor. The only minor difference to note is the slight increase in the number of blocked/load-shared calls. This increase is attributed to the fact that the CPs in this configuration have to perform some additional processing on the IN calls before sending a response back to the SSP. In fact, the CPs have to copy and create a data packet in order to send it to the DP. Hence, this extra processing is reflected in the CPU service time to process an IN call. It is evident from Table 5.3 that the 95% quantile response time is guaranteed for all buffer size values. Notice that the number of blocked/load-shared calls accounts for a very small percentage of all IN calls generated.

Figure 5.5 shows the Call Processor CPU Queue Size and the Call Processor CPU Utilization for a buffer size value of 150. The CPU queue size seems to identify two small overload periods. During these periods, the maximum capacity of the buffer is achieved and the CPU utilization rises to 100%. The first overload period happens during simulation time of approximately 127 to 138 seconds (from simulation results), while the second overload period can be neglected (from simulation results no calls were blocked/load-shared during that period, even though the queue size increased). It is during the first overload period that the number of blocked/load-shared calls increases to a value of 277 calls (which represents a blocking probability of 0.13%).

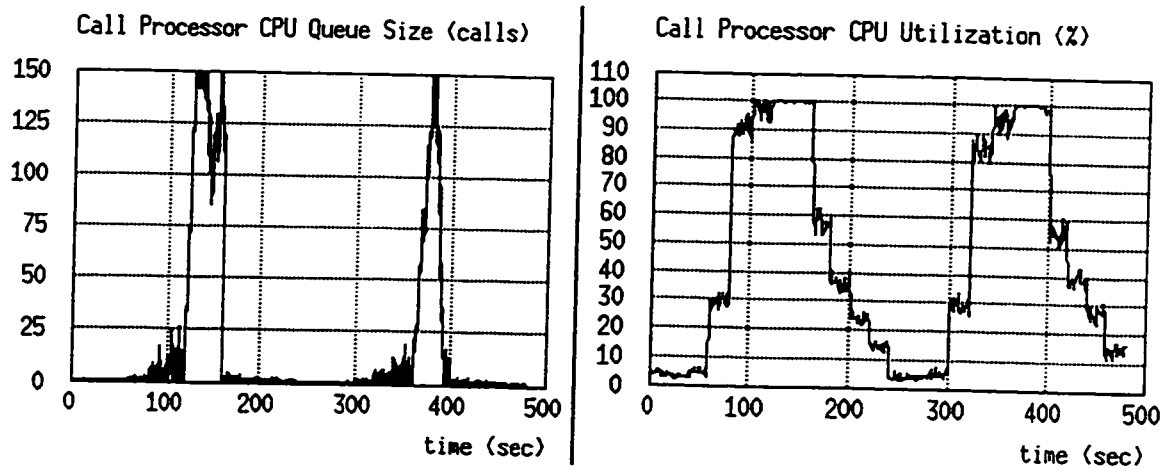


Figure 5.5 Call Processor CPU Queue Size and CPU Utilization

Figure 5.6 presents the IN calls response time for a buffer size of 150. Notice that almost all IN calls are processed below 1.0 second, hence respecting the maximum response time for the type of services investigated. The figure depicts three levels of response time. The first response time level is around 70 msec (from simulation results). This SCP response time is achieved when the SCP is able to handle the incoming traffic workload. During the first level of response time, we can also note that the CPU queue size is under its maximum capacity, and that the CPU utilization is well below 100%. The second and third level of response time happens under the overload periods, hence when the CPU queue size and utilization increases abruptly. The second and third level of response time are from 0.3 to 0.5 seconds and 0.6 to 1.0 second, respectively. Therefore, calls requiring zero visit (i.e., for SLPs are located in the MMDB) are processed within the 0.3 to 0.5 seconds range. The calls requiring one visit (i.e., for SLPs located in the DBDB) are processed within 0.6 to 1.0 second.

At this point it is good to compare these response time values to the results obtained in chapter 4 (for configuration 1). Recall that under a SCP call sampling rate of 100% introduced in configuration 1 (i.e., second set of simulations, Sampling Scenario 1), the response time of IN calls increased to 2.5 seconds for calls requiring zero visit, and 5.0 seconds for calls requiring one visit. Thus, we can clearly see that configuration 2 gives a considerable improvement in response time, while performing 100% sampling (as we will see next). This is attributed to the fact that the CPs in this configuration are freed from the SCP call sampling activity.

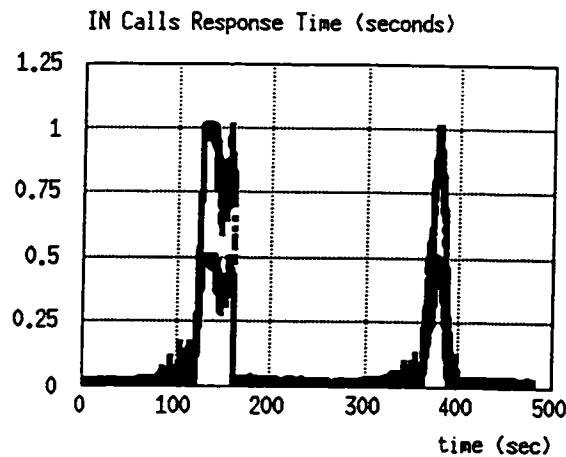


Figure 5.6 IN Calls Response Time

Figure 5.7 illustrates the SCP throughput. For configuration 2, the maximum achievable SCP throughput is approximately 865 calls per second (from simulation results). Therefore, 865 calls per second is an estimate of the achievable peak traffic workload into the SCP. Hence, this configuration respects the SCP throughput requirement imposed on our SCP model. If we compare this result to the results obtained in configuration 1, we

can notice the two following differences: (1) in configuration 1 (first set of simulations), the achievable SCP throughput under a sampling rate of 0% was 870 calls per second, and (2) in configuration 1 (second set of simulations), the achievable SCP throughput under a sampling rate of 100% was 175 calls per second. Therefore, we can clearly see that configuration 2 presents the best of both set of simulations analyzed in configuration 1. The reason for this is that configuration 2 achieves 865 calls per second and acceptable response time values, while at the same time off-loading a sampling rate value of 100% to an intermediate specialized processor (Data Processor).

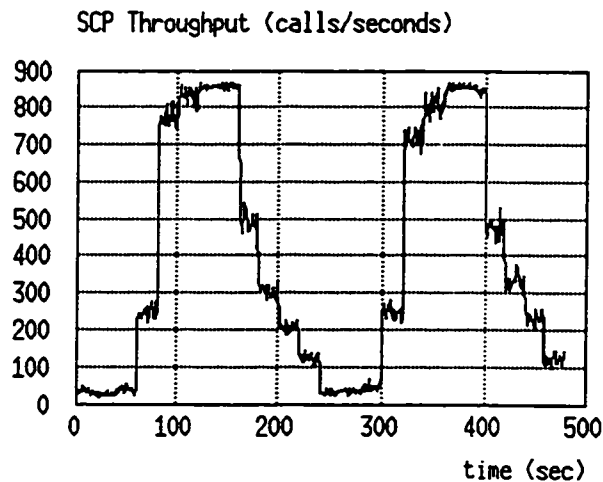


Figure 5.7 SCP Throughput

We now present some the results of the so-called specialized processor, namely the Data Processor. Figure 5.8 shows the Data Processor CPU Utilization. Notice that the DP CPU Utilization reaches 100% for most of the simulation time. During such period the Data Processor CPU queue size increases at a very fast rate (not shown herein). This is due to the fact that the Data Processor CPU is busy sampling the remaining IN calls from the SCP. Hence, it is totally overloaded by the enormous amount of incoming calls

requiring sampling. Because the DP is a one processor-based, and that the complexity of sampling a call is now four times the complexity of serving an IN call, the DP is not able to handle the incoming traffic workload from the token ring (i.e., the data packets sent from the SCP to the DP over the token ring). Hence, the Data Processor CPU is now the bottleneck entity in configuration 2. But, it is good to mention that this bottleneck situation does not affect the SCP throughput, SCP response time and blocking probability, which is somewhat good. These latter performance metrics are the key to provide good service to the users (since users are the primary source of revenue and profit).

It is known that when the utilization approaches 100% for sustained periods of time, the integrity of the DP is being compromised. To eliminate this behavior, a solution is needed (this will be analyzed later in this section).

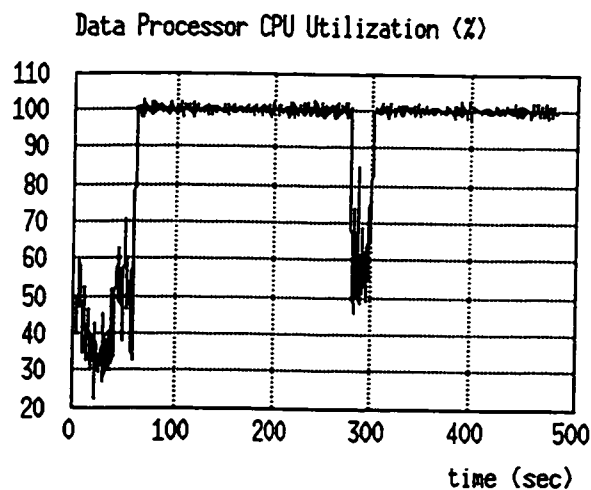


Figure 5.8 Data Processor CPU Utilization

Figure 5.9 shows the number of incoming *data packets* to the DP. Recall that a data packet is sent to the DP after the IN call is completely served at the SCP (i.e., the data packet is sent to the DP at the same time as the response is sent back to the SSP). Notice that the SCP throughput (see Figure 5.7) is approximately equivalent to the incoming data packet rate shown in this figure. This is due to the fact that for each IN call served at the SCP a data packet is sent to the DP. This figure also gives us an idea of the amount of traffic circulating over the token ring. In section 5.3, we will show that this rate (in Mbps) is well under the maximum token ring data rate (i.e., 16 Mbps).

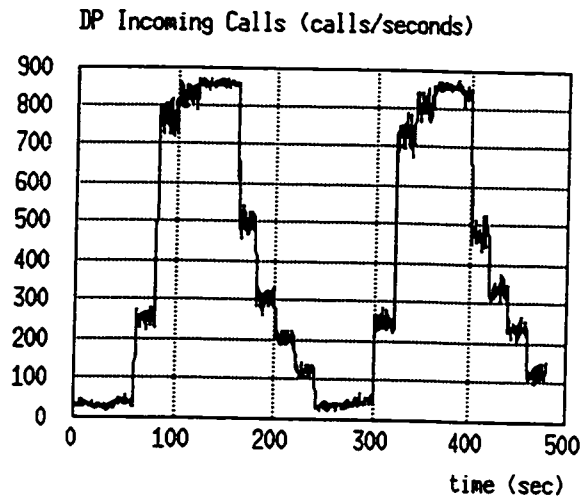


Figure 5.9 Data Processor Incoming Calls

Figure 5.10 depicts the DP throughput. The maximum achievable DP throughput is 75 calls per second. This throughput remains constant (at its maximum) for most of the simulation time. Notice that the incoming data packet rate presented in the last figure is well over the DP throughput presented in this figure. Therefore, we can state that our one processor-based DP is not able to handle the incoming load. But, if the number of incoming data packet would be less than 75 calls per second, the DP would be able to

process the incoming traffic workload. In such case, the DP would operate under normal conditions.

In order to increase the performance of the DP, one solution is to put a faster processor. That is, to increase the CPU processing power, hence decreasing the service time to sample an IN call.

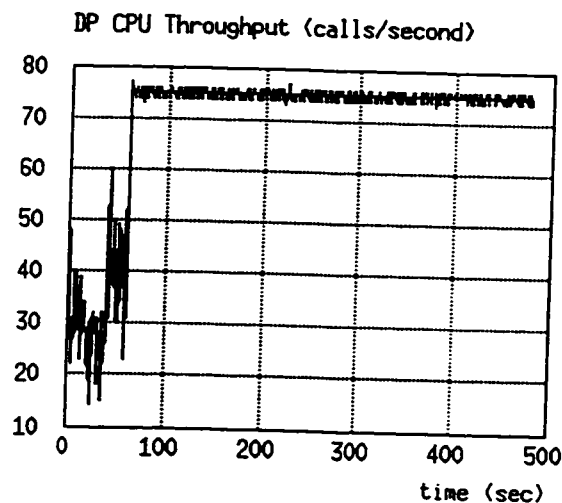


Figure 5.10 Data Processor Throughput

Figure 5.11 and Figure 5.12 show the DP throughput when the CPU processing power is increased by a factor of two and four, respectively. In such case, the mean service time to sample an IN call is reduced to 6.66 msec (when doubling the CPU processing power) and 3.33 msec (when quadrupling the CPU processing power), respectively. From these two figures, we can see that the DP is now able to process the incoming load (which was not the case in Figure 5.10). Notice that the throughput decreases at some point during simulation time (i.e., it does not remain to a maximum constant value throughout the whole simulation time). Indeed, during simulation time 260 to 300 seconds in Figure 5.11,

and 215 to 300 seconds in Figure 5.12, the throughput decreases, meaning that the Data Processor was able to process all the incoming data packets from the SCP (i.e., the data packets that were queued in the DP, and waiting to be sampled). This would actually be equivalent to say that during the morning of the second day (simulation time around 220 seconds), the DP is able to catch up on the calls that were queued for sampling. The same thing happens in Figure 5.12, except that the DP catches up on its sampling task faster. Notice that when doubling the CPU processing power, the maximum achievable DP throughput is 150 calls per second, whereas when the CPU processing power is quadrupled, the maximum achievable DP throughput is 300 calls per second.

All of this suggests that if the DP is engineered to have a throughput of more than 150 calls per second (i.e., by increasing the processing power), the DP will be able to perform 100% sampling, hence respecting the 100% sampling requirement. If the DP throughput is somewhat less than 150 calls per second (i.e., as in the first case where the DP achieved only 75 calls per second), it will not be able to guarantee the 100% sampling rate.

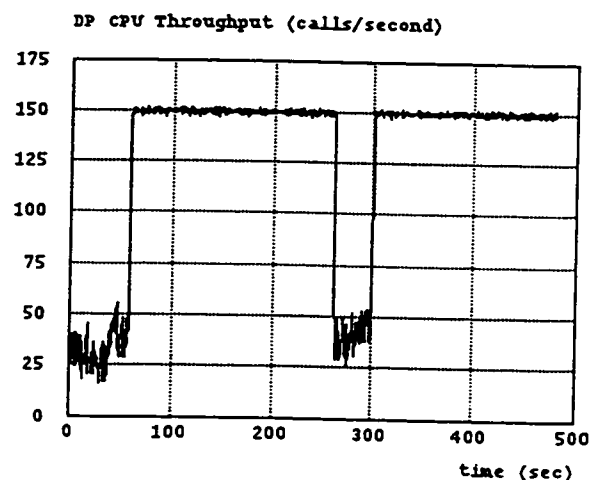


Figure 5.11 Data Processor Throughput (Doubling CPU processing power)

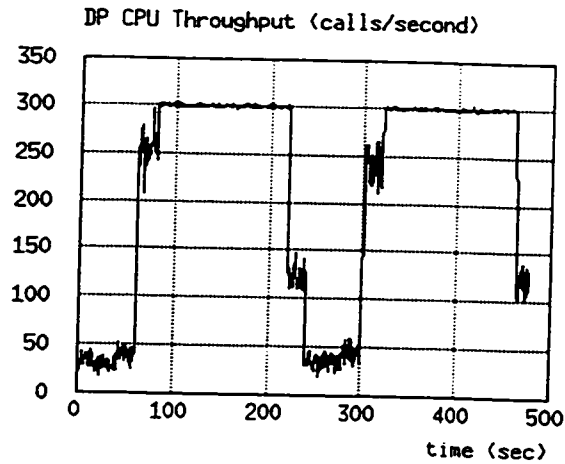


Figure 5.12 Data Processor Throughput (Quadrupling CPU processing power)

5.3 Data Processor Modeling and Performance Issues

In this section we present useful numbers that characterizes the modeling of the Data Processor (e.g., DP peak data packet input rate, estimated average volume of data stored per day, DP CPU processing power, etc.). We saw in section 5.1 that the DP encompasses a CPU subsystem and some sampling disks. The CPU subsystem has a main memory which is used to buffer the incoming data packets, while the sampling disks are used to store the IN data records. In our proposed configuration, we saw that the DP is receiving a data packet for each IN call processed by the SCP. This means that during heavy traffic load, the DP may receive up to 900 calls per second since we inject the SCP with the traffic workload trace presented in Figure 3.7.

We first calculate the average number of data (in bytes) that could be created by the sampling activity. The value represents the average amount of data stored on a daily

basis. This value permits us to estimate the number of sampling disks needed to store the IN data records, as well as their retaining period. The average IN data record size created is assumed to be $\frac{200 \text{ bytes}}{\text{IN Data Record}}$ [27]. The record size is usually proportional to the number of variables that needs to be stored in an IN data record (e.g., 800 number dialed, time of day, number of failures, etc.). Since the traffic trace values (and shape) are known to us (see Figure 3.9), we can see that there are approximately two busy peak hours per day (business hours). These two peak hours generate approximately 900 calls per second. The remaining 6 business hours, and 3 non-business hours generate approximately 300 calls per second. The remaining non-business hours generate a traffic workload of 100 calls per second or less. From these values, the *average number of IN data records per second* created is:

$$\frac{900 \times 2 + 300 \times 9 + 100 \times 13}{24} = 242 \text{ IN data records per second}$$

Therefore, the *average number of data (in bytes)* stored per day due to the sampling activity is:

$$\frac{200 \text{ bytes}}{\text{IN data record}} \times \frac{242 \text{ IN data records}}{\text{second}} \times \frac{86400 \text{ seconds}}{\text{day}} = 4.18 \text{ Gbytes per day}$$

The latter value gives the average amount of data that needs to be stored for a single weekday. Also, the amount of data does not take into account the overhead needed to store it. For example, a 3 day retention period would approximately hold 13 Gbytes of sampled data. In such conditions, the sampling disk-based system, which encompasses 3

disks, should be able to store 13 Gbytes of data and more. Hence, each sampling disk of our DP must approximately have a storage capacity of at least 4.2 Gbytes.

We now proceed to calculate the *peak data packet input rate* into the Data Processor. The DP peak input rate arises when the traffic workload to the SCP is at its maximum. We saw in configuration 2 that the maximum achievable SCP throughput was around 865 calls per second. Thus, it is for this SCP throughput rate (since the output of the SCP is sent to the Data Processor over the token ring) that we calculate the peak input rate. Again we assume that a data packet formatted at the CP, and transmitted over the token ring is approximately $\frac{200 \text{ bytes}}{\text{Data Packet}}$ (this size includes the header field and data field of the data packet). Therefore, the *peak input rate into the DP* is:

$$\frac{200 \text{ bytes}}{\text{Data Packet}} \times \frac{865 \text{ Data Packet}}{\text{second}} \times \frac{8 \text{ bits}}{\text{byte}} = 1.384 \text{ Mbits per second}$$

The value gives us an estimate of the total bit rate sent over the dual token ring to the DP, under the busy peak hours or heavy traffic load. We can justify that this value is well under the maximum capacity of the 16 Mbits per second token ring transmission. Therefore, we can state that the token ring is not a source of bottleneck, since it is able to handle such transfer rate (from the SCP to the DP).

We must also realize that the DP needs to be able to handle such bit rate (1.36 Mbits per second). We now proceed to determine, by a quick evaluation, if the DP can process such bit rate. We know that the mean service time to create an IN data record is given as 13.32 msec (i.e., if we use the “*standard*” processor used throughout the

configurations). The mean DP service rate is therefore given as the inverse of this value, and is equal to 75 calls per second (this is what we obtained from the simulation). If we know that there are approximately two hours at 900 calls per second, nine hours at 300 calls per second and thirteen hours at 100 calls per second, we can then calculate the mean number of calls generated per day. Therefore, the *number of calls generated per day* is 20.88 millions. If the DP is capable of sampling a maximum of 75 calls per second, then in a whole day it should be able to process approximately 6.5 million calls. Now, we can logically see that the number of calls generated per day is larger than the number of calls that the DP can process. Hence, this DP (i.e., with the specified processing power) will not be able to handle the traffic workload generated per day. Therefore, a quick solution was to increase the CPU processing power. We saw in section 5.2, by means of simulation, that increasing the CPU processing power (e.g., by a factor of two and four times) provided the DP to achieve a 100% sampling rate.

5.5 Summary

In this configuration the SCP throughput was estimated to be around 865 calls per second, meeting the requirements imposed on our SCP model. We also noticed that the SCP response time was below 1.0 second, hence meeting the 95% quantile response time. We also noticed that the blocking probability was fairly low. We also saw that the DP was the bottleneck source, since it could not process the incoming DP load (coming from the token ring). Even if the DP bottleneck situation did not influence the performance of the SCP in terms of response time, throughput and blocking probability, it did affect the performance of the DP in terms of the number of calls it could sample. The maximum

achievable DP throughput was around 75 calls per second. But the main concern to all of this was that the CPU had a utilization of 100% for a sustained period of simulation time (recall that 480 seconds of simulation time represents two weekdays). In fact, the DP CPU utilization gave us an indication of the percentage of time it was busy for the given workload. It is known that the resource with the highest utilization is called the bottleneck, and that performance optimization at this resource offers the highest payoff (e.g., in terms of throughput). Therefore, a solution to alleviate this high utilization and low throughput was needed. Perhaps one of the most common configuration change is the upgrade of a CPU within a family of processors. This is one easy change to evaluate using queuing network models. For instance, when doubling the CPU processing (halving the service time) we noticed that the DP was able to sample all incoming data packets. This change provided a DP throughput of 150 calls per second. When the CPU processing rate was quadrupled, a DP throughput of 300 calls per second was achieved.

Other solutions could have been investigated. One solution would have consisted on assessing the possibility of acquiring a second processor to form a tightly-coupled multiprocessor system (i.e., two processors cooperating to complete work from a single shared queue), or a loosely-coupled multiprocessor system (i.e., two processor having their own separate job queue). Even for such solutions, a lot of other concepts such as reliability (i.e., the probability measure of errors/failures), availability (i.e., the fraction of time the system is available to users), replication (i.e., the cost of replicating data throughout the system) and cost/performance ratio (i.e., to compare two or more systems) would need to be investigated.

Chapter 6. Conclusions

The “*business*” of IN services (e.g., freephone, calling card, credit card service, etc.) is pivotal in the sense that users represent the dominant part of revenue and profit for customers, service providers and network operators. In order to provide users with more than enough satisfaction, addressing their present needs will continue to be a major challenge for the network operators and service providers. Analogous to this challenge, new challenges will arise when services such as universal personal telecommunication, interactive multimedia services and information services increases as the globalization of IN services drives IN technological development. Thus, performance demands will become more critical, especially when the traffic workload to the SCP includes a larger volume of data, interworking, increasing service complexity, broad range of activities, etc. Also, emerging services will mostly result in new expectations and requirements on service performance. These are reasons why the design, modeling and evaluation are considered as important attributes in assessing the SCP architecture, also known as the “*heart*” of the Intelligent Network.

Chapter 1 presented how the Intelligent Network originated. In the late 80s, problems found in the PSTN required an immediate solution. But to all possible solutions, the huge investment in terms of equipment and man-power had to be preserved. A new reorganization of the PSTN was envisaged. Thanks to the Intelligent Network concept, a potential solution was on the verge of being defined. Chapter 1 also stated the objectives, problems, and challenges to which this thesis tries to answer. We identified that the SCP architecture was the primary entity under investigation. General questions were formulated, for which answers are provided throughout this thesis.

In chapter 2, we saw that the telecommunication industry took a bold step towards achieving high level standards for the IN technology, giving network operators, services providers, and customers flexibility to either create, provision and deploy IN services in the PSTN. The first initiative and request that lead to the IN technology consisted of the following three points:

1. support the rapid introduction of new services in the network;
2. help establish and interface standards to give the RBOCs the widest possible choice of vendor products;
3. create opportunities for non-RBOC service vendors to offer services that stimulate network usage.

We also presented the history and evolution of Intelligent Network, up to today's standards. Along the way we mentioned the pitfalls and results of the Intelligent Network evolution. We presented the framework of Bellcore and the ITU-T sector. Since the ITU-T defines the international recommendation, we explained it with more details.

In chapter 3, we identified all the requirements, guarantees and QoS that the SCP architecture has to meet. The three most important metrics identified were the SCP throughput, SCP response time and SCP call sampling rate. The proposed SCP architecture was modeled as a multiprocessor-based system. The SCP architecture is flexible enough to allow continuous modifications along the Intelligent Network evolutionary step, and system expansion. Such type of architecture also provides smooth, cost effective and capacity increments that will minimize growth difficulties. Chapter 3 gave a detailed understanding of the targeted network and requirements.

Emphasis was placed on issues having most of the significant IN services features for the evaluation of the Service Control Point. We presented the following:

1. high level model and system operation of the proposed SCP architecture;
2. the SCP response time and SCP throughput requirements;
3. the SCP call sampling activity and polling activity;
4. analysis of real-time traffic measurements;
5. main memory database and disk-based database characteristics;
6. statistical distributions used in the modeling of the SCP.

Chapter 4 described configuration 1. We presented the low level models of the SCP architecture (i.e., the queuing models associated to the high level model of chapter 3). Two sets of simulations were conducted. The first set of simulations analyzed the performance of the SCP with respect to different performance issues. It was primarily focused on the performance impact of processing IN calls. We investigated three IN call scenarios. In order to meet the desired quality of service imposed on the SCP model, we saw that the SCP had to store most of its SLPs in the MMDB. That is to say, 95% of all SLPs, which represents approximately 456 Mbytes, had to be stored in primary storage (main memory). We noticed that secondary storage, which represents disk-based systems, degraded the performance of the SCP in terms of response time and throughput. The following IN call scenario (95%, 5%, 0%) was therefore within specification, hence respecting the SCP throughput, SCP response time and blocking probability imposed on the SCP model. It is therefore mandatory that the SCP maintains large main memory databases in order to provide excellent quality of service.

The second set of simulations consisted on evaluating the performance of the SCP when the SCP call sampling activity was introduced. Due to the complexity of the

sampling activity, we noticed that the performance degraded considerably as the SCP call sampling rate increased. However, even though the performance of the SCP decreased, it was still within the requirements for a sampling rate of 5%. The maximum achievable SCP throughput was reduced from over 870 calls per second (when the sampling rate was 0% in the first set of simulations) to about 800 calls per second (when the sampling rate was 5% in the second set of simulations). Still, the latter value is within the requirements imposed on the SCP model, and is even beyond today's traffic workload seen by existing INs, such as the Stentor's AIN. For a sampling rate of 100%, we noticed that the maximum achievable SCP throughput was 175 calls per second. Of course this value is well under the requirement imposed on the SCP model, but suggests that in order to perform 100% sampling under configuration 1 and have satisfactory quality of service, the traffic workload generated by the SSPs must be at most 175 calls per second. But, a drawback of performing 100% sampling is that the blocking probability and SCP response time increases dramatically, hence reflecting the users' satisfaction and users' perception of IN services. For example, it makes no sense for a user to wait 2.5 seconds or 5 seconds for a response of the SCP. It is known that the longer the response time is, the more the users will become impatient. This will cause users to hang up and retry again, causing the performance of the SCP to degrade significantly.

Chapter 5 presented configuration 2. Since the SCP call sampling activity is an ever increasing requirement, a solution to alleviate the low SCP call sampling rate that can be achieved in configuration 1 was needed. The performance of the SCP in configuration 2 proved totally acceptable when we introduced the specialized processor, namely the Data Processor. The achieved SCP throughput was estimated to be around 865 calls per second. Also, 95% of all IN calls were successfully served under 1.0 second. Moreover, a SCP sampling rate of 100% could be performed. All of this was accomplished at the cost of an extra unit, the Data Processor. We noticed that the DP was the bottleneck

entity. We also noticed that the integrity of the DP was being compromised, due to the excessive workload offered to it (i.e., creation of IN data records). In order to alleviate such condition, we proposed to increase the CPU processing power. We observed that for the DP to operate under normal conditions, it had to achieve a throughput of at least 150 calls. We also mentioned that the DP's average number of bytes per day to be stored was in the order of Gbytes, more precisely 4.2 Gbytes per day. This value indicated the size of the disks that was needed to store the IN data records created by the SCP call sampling activity over the specified retention period. We also estimated the peak input bit rate to the DP, and concluded that this value was under the maximum capacity of the 16 Mbits per second token ring transmission rate.

References

- [1] R. L. Martin, *Future Telecommunications Services*, Proceedings of Global Telecommunications Conference, Globecom '88, Volume 2, pp. 721-725, Hollywood, FL, November 1988.
- [2] R. B. Robrock II, *The Intelligent Network - Changing the Face of Telecommunications*, Proceedings of the IEEE, Volume 79, Number 1, pp. 7-20, January 1991
- [3] A. E. Joel, et al., *A History of Engineering and Science in the Bell System*, Switching Technology (1925-1975), Bell Telephone Laboratory, 1982.
- [4] M. Scharwtz, *Chapter 11 - Call Processing in Digital Circuit-Switching System*, Telecommunication Networks - Protocols, Modeling and Analysis, Stephanie Kaylin, ed. Addison-Wesley Publishing Company, Reading, MA, 1987.
- [5] W. Keister, H. W. Ketchledge, H. E. Vaughn, *No. 1 ESS: System Organization and Objectives*, Bell System Technical Journal, Volume XLIII, Number 5, pp. 1831-1844, 1964.
- [6] D. M. Hursey, *800 Service in BellSouth*, Proceedings of Global Telecommunications Conference, Globecom'86, Volume 3, pp. 1330-1335, Houston, TX, November 1986.
- [7] T. Murphy, T. E. Newman, W. Woodbury, D. P. Worrall, *Intelligent Network Control of BOC 800 Service*, Proceedings of Global Telecommunications Conference, Globecom '86, Volume 3, pp. 1320-1324, Houston, TX, November 1986.
- [8] R. J. Hass, R. B. Robrock II, *The Intelligent Network of the Future*, Proceeding of Global Telecommunications Conference, Globecom '86, Volume 3, pp. 1311-1315, Houston, TX, November 1986.
- [9] M. Bray, *Impact of New Services on SCP Performance*, Proceedings of the IEEE International Conference on Communications, Volume 1, pp. 241-247, Atlanta, GA, 1990.
- [10] C. A. Lucenius, *INside Intelligent Networks - A Newsletter for the Intelligent Network Community*, pp. 1-11, October 1997.
- [11] MIL 3 Inc., *OPNET Modeler Release 2.5.B - Modeling Manual 2.0*, Washington, DC, 1995.

- [12] ITU Recommendation, *Human Factors Aspects of Public Terminals: Generic Operating Procedures*, Recommendation E.134, Geneva, Switzerland, 1992.
- [13] ITU Recommendation, *International Telephone Service*, Recommendation E.105, Geneva, Switzerland, 1992.
- [14] R. E. Allen, *A View of Divestiture - Ten Years Later*, IEEE Communications Magazine, Volume 31, Number 12, pp. 18-19, December 1993.
- [15] A. Hansson, *Evolution of Intelligent Network Concepts*, Computer Communications, Volume 18, Number 11, November 1995.
- [16] L. B. Glass, *The SCSI Bus - Part 1*, BYTE Magazine, Volume 15, Number 2, pp. 267-274, February 1990.
- [17] L. B. Glass, *The SCSI Bus - Part 2*, BYTE Magazine, Volume 15, Number 3, pp. 291-298, March 1990.
- [18] R. K. Berman, J. H. Brewster, *Perspectives on the AIN Architecture*, IEEE Communications Magazine, Volume 30, Number 2, pp. 27-32, February 1992.
- [19] J. M. Duran, J. Visser, *International Standards for Intelligent Networks*, IEEE Communications, Volume 30, Number 2, pp. 34-42, February 1992.
- [20] K. E. E. Raatikainen, *Information Aspects of Services and Service Features in Intelligent Network Capability Set-1*, Report C-1994-45, Department of Computer Science, University of Helsinki, Helsinki, Finland, 1994.
- [21] B. S. Northcote, *Analysing SCP Congestion Control Strategies: A Study of Control Dynamics*, IN '96 Workshop Record, Volume 1, Melbourne, Australia, 1996.
- [22] R. Kawahara, T. Asaka, *Overload Control for Intelligent Networks Based on an Estimation of Maximum Number of Calls in a Node*, IN'96 Workshop Record, Volume 1, Melbourne, Australia, 1996.
- [23] M. Ouellette, J. Bedoy, L. O. Barbosa, N. D. Georganas, *Performance Evaluation of a Multiprocessor SCP*, IN'97 Workshop Record, Colorado Springs, CO, 1997.

- [24] J. Bedoy, L. O. Barbosa, *Study of load control mechanism for the effective and robust delivery of IN services*, Internal Report, Department of Electrical and Computer Engineering, University of Ottawa, Ottawa, Canada, 1997.
- [25] G. S. Abdo, *Evolution Towards Advanced Intelligent Network Services*, Proceedings of the Intelligent Network - The Path to Global Network, P. W. Bayliss, ed. IOS Press, Burke, VA, pp. 279 - 287, 1992.
- [26] W.D. Ambrosch, A. Maher, B. Sasscer, *The Intelligent Network - A Joint Study by Bell Atlantic, IBM and Siemens*, ed. Springer-Verlag, New York, NY, 1989.
- [27] M. Severson, *ISCP V4.0 Performance Test Report*, Internal Report, Stentor Resource Center Incorporated, Ottawa, Canada, May 1996.
- [28] ITU Recommendation, *Distributed Functional Plane for Intelligent Networks CS-1*, Recommendation Q.1214, Geneva, Switzerland, 1995.
- [29] ITU Recommendation, *Distributed Functional Plane for Intelligent Networks CS-2*, Draft Recommendation Q.1224, Melbourne, Australia, May 1996.
- [30] ITU Recommendation, *International Network Management Guidance for Common Channel Signaling System No. 7*, Recommendation E.415, Geneva, Switzerland, 1992.
- [31] W. Stallings, *Chapter 11 - Signaling System Number 7*, ISDN and Broadband ISDN, John Griffan, ed. Macmillan Publishing Company, New York, NY, pp. 433-492, 1992.
- [32] ITU Recommendation, *Specification of Signaling System No. 7*, Blue Book Recommendation, Q.700-Q.716 and Q.721-Q.766, Geneva, Switzerland, 1988.
- [33] ITU Recommendation, *Introduction to Intelligent Network Capability Set-1*, Recommendation Q.1211, Geneva, Switzerland, 1993.
- [34] J. Kim, K. Yun, K. Lee, D. Im, *Design and Implementation of Service Database Management in the Combined SCP/SMS*, Proceedings of the Intelligent Network - The Path to Global Network, P. W. Bayliss, ed. IOS Press, Burke, VA, pp. 592-599, 1992.

- [35] A. Syed, *Intelligent Networks Transaction Processing Nodes (TPN)*, Proceedings of the Intelligent Network - The Path to Global Network, P. W. Bayliss, ed. IOS Press, Burke, VA, pp. 256-267, 1992.
- [36] S. Hollywood, *SCP Development in a Multi-processor UNIX Environment*, Proceedings of the Intelligent Network - The Path to Global Network, P. W. Bayliss, ed. IOS Press, Burke, VA, pp. 279-287, 1992.
- [37] S. C. Ahn, G. C. Choi, K. S. Lee, *Implementation of IN in Korea Telecom*, Proceedings of the Intelligent Network - The Path to Global Network, P. W. Bayliss, ed. IOS Press, Burke, VA, pp. 224-235, 1992.
- [38] W. Stallings, *Data and Computer Communications*, B. Jones, ed. Macmillan Publishing Company, New York, NY, 1994.
- [39] B. Hoang, *Service Completion Time for Advanced Intelligent Network Services*, Proceedings of IEEE International Conference on Communications, Volume 2, pp. 584-588, Atlanta, GA, 1990.
- [40] G. B. Choi, H. H. Lee, S. Kim, H. G. Bahk, *A Service Switching Point for Intelligent Networks*, Proceedings of the Intelligent Network - The Path to Global Network, P. W. Bayliss, ed. IOS Press, Burke, VA, pp. 268-274, 1992.
- [41] ITU Recommendation, *Forecasting New Telecommunication Services*, Recommendation E.508, Geneva, Switzerland, 1992.
- [42] ITU Recommendation, *Traffic Measurement Data Analysis*, Recommendation E.503, Geneva, Switzerland, 1992.
- [43] ITU Recommendation, *Traffic Measurement By Destination*, Recommendation E.491, Geneva, Switzerland, 1992.
- [44] ITU Recommendation, *Traffic Measurement and Evaluation - General Survey*, Recommendation E.508, Geneva, Switzerland, 1992.
- [45] ITU Recommendation, *Billing Integrity*, Recommendation E.433, Geneva, Switzerland, 1992.
- [46] E. Didelet, *Circuit-switched traffic forecasting using neural networks*, Proceedings of the 15th International Teletraffic Congress - ITC, V. Ramaswami and P. E. Wirth, ed. Elsevier, New York, NY, pp. 111-121, 1997.

- [47] H. Tanaka, H. Tokunaga, Y. Kirihaara, *Intelligent Network Service Operation Features*, Proceedings of the Intelligent Network - The Path to Global Network, P. W. Bayliss, ed. IOS Press, Burke, VA, pp. 63-73, 1992.
- [48] L. Mahmood, *Interworking Between Public Networks and Customer Premises Systems on Dynamic Call Routing*, Proceedings of the Intelligent Network - The Path to Global Network, P. W. Bayliss, ed. IOS Press, Burke, VA, pp. 202-206, 1992.
- [49] W. H. Kohler, Y-P. Hsu, T. K. Rogers, W. H. Bahaa-El-Din, *Performance Evaluation of Transaction Processing Systems*, Digital Technical Journal, Volume 3, Number 1, pp. 1-14, Winter 1991.
- [50] ITU Recommendation, *Q.1210 - Series Intelligent Network Recommendation Structure*, Recommendation Q.1210, Geneva, Switzerland, 1993.
- [51] ITU Recommendation, *Q.1220 - Series Intelligent Network Recommendation Structure*, Draft Recommendation Q.1220, Geneva, Switzerland, 1996.
- [52] ITU Recommendation, *Introduction to Intelligent Network Capability Set-2*, Draft Recommendation Q.1221, Melbourne, Australia, May 1996.
- [53] White Paper, *The AT&T Worldwide Intelligent Network*, Available at: <http://www.att.com/network>, AT&T, 1996.
- [54] R.L. Bennett, G.E. Policollo II, *Switching Systems in the 21st Century*, IEEE Communications Magazine, Volume 31, Number 3, pp. 24-28, March 1993.
- [55] P.A. Bernstein, W.T. Emberton, V. Treba, *DECdta - Digital's Distributed Transaction Processing Architecture*, Digital Technical Journal, Volume 3, Number 3, pp. 1-7, Winter 1991.
- [56] S. Jin, D. Lim, J. Hong, *Performance Analysis Service Control Point in the Intelligent Network*, Proceedings of IEEE TENCON '93, Beijing, China, pp. 673-676, 1993.
- [57] D. E. Smith, *Ensuring Robust Call Throughput and Fairness for SCP Overload Controls*, IEEE/ACM Transactions on Networking, Volume 3, Number 5, pp. 538-548, 1995.

- [58] X. H. Pham, R. Betts, *Congestion Control For Intelligent Networks*, Computer Networks and ISDN System, Volume 26, pp. 511-524, 1994.
- [59] A. M. Mishra, M. Sundararaman, A.M. Mazzario, *Performance Analysis of Intelligent Networks*, Proceedings of the IEEE International Conference on Communications, Volume 2, pp. 594-597, Atlanta, GA, 1990.
- [60] O. E. Hoharran, G. Duguay, *Intelligent Network Traffic Management*, Proceedings of ICCT '96, Beijing, China, 1996.
- [61] C. E. Labonte, X. Zhang, *Guidelines for CS-IR Based Service Design*, Proceedings of ICCT '96, Beijing, China, 1996.
- [62] L. R. Bowyer, R. B. Robrock, *Intelligent Network Service for Data Customers*, Proceedings of the IEEE International Conference on Communications, Volume 2, pp. 602-606, Atlanta, GA, 1990.
- [63] P. A. Russo, et. al., *IN Rollout in the United States*, IEEE Communications Magazine, Volume 31, Number 3, pp. 56-63, March 1993.
- [64] ITU Recommendation, *Q-Series Intelligent Network Recommendation Structure*, Recommendation Q.1200, Geneva, Switzerland, 1993.
- [65] ITU Recommendation, *Principles of Intelligent Network Architecture*, Recommendation Q.1201, Geneva, Switzerland, 1993.
- [66] J. J. Garrahan, P.A. Russo, K. Kitami, R. Kung, *Intelligent Network Overview*, IEEE Communications Magazine, Volume 31, Number 3, pp. 30-36, March 1993.
- [67] M. Fujioka, Y. Wakahara, *Consideration on Common Channel Signaling Evolution for Global Intelligent Networks*, IEEE Journal on Selected Areas in Communications, Volume 12, Number 3, pp. 510-518, April 1994.
- [68] J. Homa, S. Harris, *Intelligent Network Requirements for Personal Communications Services*, IEEE Communications Magazine, Volume 12, Number 2, pp. 70-76, February 1992.
- [69] P. E. Wirth, *Teletraffic Implications of Database Architecture in Mobile and Personal Communications*, IEEE Communications Magazine, Volume 12, Number 6, pp. 54-59, June 1995.

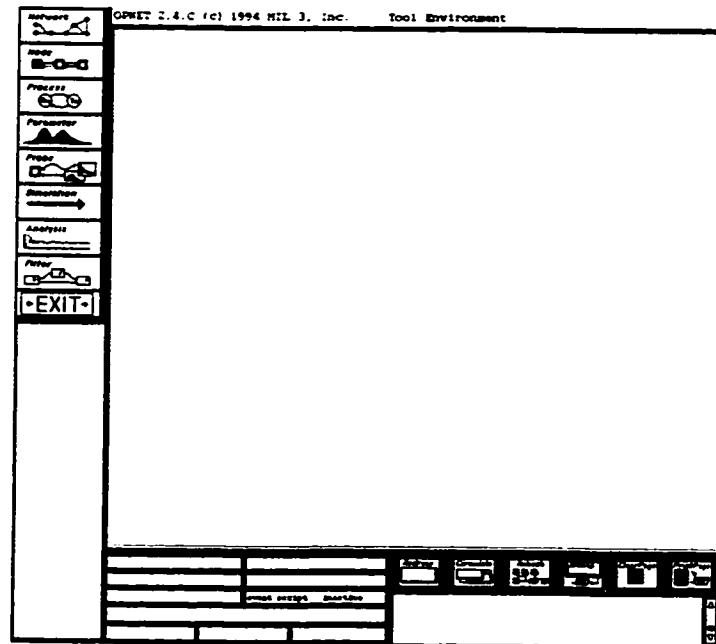
- [70] Serial Storage Architecture (SSA), Available at: <http://www.storage.ibm.com/storage/oem/ssa.htm>, Commercial Product, IBM, 1997.
- [71] R. Jain, *The Art of Computer Systems Performance Analysis: techniques for experimental design, measurement, simulation, and modeling*, ed. John Wiley & Sons Inc., New York, NY, 1991.
- [72] ITU Recommendation, *Intelligent Network Distributed Functional Plane Architecture*, Recommendation Q.1204, Geneva, Switzerland, 1993.

Appendix A. OPNET Modeler

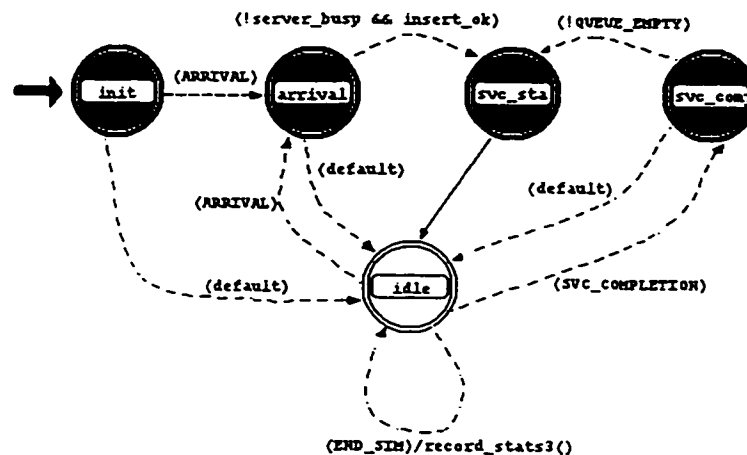
The *Optimized Network Engineering Tool* (OPNET) is a comprehensive software development environment for the modeling, simulation and performance analysis of communications networks, protocols, and distributed systems [11]. OPNET uses a *Graphical User Interface* (GUI) based on Xwindow systems. In general, OPNET provides multi-windowing, pop-up menus, dialog areas, and button-driven operations. These GUI features are identical across several computer platforms, and can be transported without any conversion. The most important tools/features provided by the OPNET Modeler are explained next.

OPNET provides the development of hierarchical model structures. In OPNET terms, this is represented at the *Network Level* and *Node Level*. It provides users a hierarchical view and a visual representation of the models, and at the same time, a parallel view to real systems. In order to achieve this, OPNET makes use of graphical object-based editors for building network topologies, node architectures, data flows, etc. OPNET uses objects as a definition principal. Similar to object-oriented techniques, OPNET provides objects to be defined by a set of attributes, which are conveniently edited via attribute menus. Appendix A.1 shows the OPNET Modeler GUI environment.

At a lower level, OPNET provides a powerful and flexible language to design protocols, algorithms, queuing policies, traffic generators, routing, etc. In OPNET terms this is represented at the *Process Level*. The Process Level makes use of *Finite State Machine* (FSM) representation, which represents a process model. Specification and full generality of the FSM is based on *State Transition Diagrams* (STD) with embedded C statements. Appendix A.2 shows the FSM of an M/M/1 queue. States and transitions are clearly identifiable.



Appendix A.1 OPNET Modeler GUI Environment



Appendix A.2 Finite State Machine of an M/M/1 Queue

An extensive library of communication and simulation-specific procedures, known as *Kernel Procedures* in OPNET terms, is provided to make programming straightforward. Appendix A.3 shows some partial C code and kernel procedures for the *arrival state* of the FSM M/M/1 queue.

```

arrival enter crrcs
1 /* Get packet & enqueue it, if possible */
2 /* As of 5th Feb only one queue exists at the CPU */
3 /* This is why each case statement as DATABASE as the name of the queue */
4
5 pktz = op_pkt_get (op_intrpt_stem ());
6 switch (op_intrpt_stem())
7 {
8     case NEW_STIM_DS
9         if (op_subq_pkt_insert (NEW_CALL, pktz, OPC_QPOS_TAIL) != OPC_QIMS_OK)
10         {
11             op_pkt_destroy(pktz);
12             insert_ok = NO;
13         }
14         else
15         {
16             insert_ok = YES;
17         }
18         break;
19     case DB
20         if (op_subq_pkt_insert (DATABASE, pktz, OPC_QPOS_TAIL) != OPC_QIMS_OK)
21         {
22             op_pkt_destroy(pktz);
23             insert_ok = NO;
24             printf("\nCall not entering CPU sub-queue (from DB)");
25         }
26         else
27

```

Appendix A.3 Partial C Code and Kernel Procedures

Finite state machines (which embeds C code) need to be compiled. Node, network and process models are archived and binded together in order to create an executable discrete-event simulation file. Simulation files are then executed via the *OPNET Simulation Tool*, or executed with user-defined script files. Appendix A.4 shows one of the script files used in this thesis. The simulation results obtained from the execution of the simulation file can then be plotted and manipulated as time series graphs, scatter plots, histograms, etc. Standard statistics, such as time average, mean queue length, confidence intervals, etc., are easily obtained.

OPNET provides many more advanced capabilities and features. Some of them are for visualizing dynamic simulation events or to provide interactive debugging. Others provide an open interface to generate PostScript or TIFF files for desktop publishing. Other tools that have a significant importance are the *Parameter Editor* and *Probe Editor*. They are used to define parametric constructs such as packet formats and probability distribution functions, and to specify statistics to be gathered during run-time simulation.

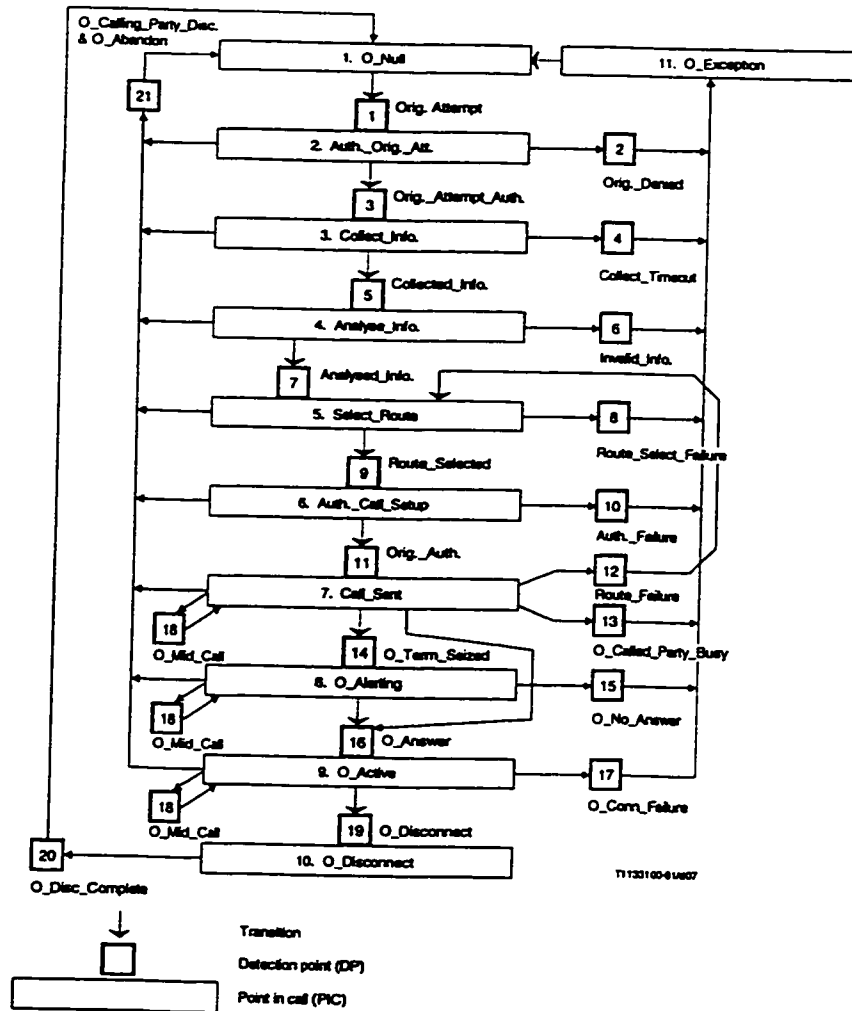
```
SCRIPT_FILE : enter execs
1  #!/bin/csh
2  #script to run simulations for *** configuration 1 ***
3  #this config includes Data Processor and no Data Management Processor
4  rm Scalar.os
5  rm Vector.ov
6
7  @ seedval = 79
8  while ($seedval < 80)
9      echo 'executing simulation with seed: 'seedval
10     scp_config_1.sim
11     "-*.gen.ssp.arrival_rate"          2.12 \
12     "-*.gen.ssp.sim_time"              480 \
13     "-*.cp.CPU[0].pk_capacity"          1000 \
14     "-*.cp.DB*[0].pk_capacity"         1000 \
15     "-*.cp.Sampling Disk 1*.pk_capacity" 1000 \
16     "-DP.polling.interarrival args"    10 \
17     "-*.cp.NIU-I1.sampling_pourcentage" 0.05 \
18     "-*.cp.CPU.sampling_pourcentage"   0.05 \
19     "-*.dummy_gen.interarrival args"   1 \
20     "-*.cp.DB*.service_time_normal"    8.6e-03 \
21     "-*.cp.CPU.service_time_normal"    3.33e-03 \
22     "-*.cp.CPU.variance_normal"        0.0543 \
23     "-*.cp.NIU*.service_time"          25.0e-06 \
24     "-*.cp.NIU-I1.number_database_acc" 1 \
25     -probe Probes \
26     -seed $seedval \
27     -duration 481 \
28     -upd_int 1 \
29     -os_file Scalar \
30     -ov_file Vector \
31     @ seedval = ($seedval + 17)
32 end
33
```

Appendix A.4 Simulation Script File

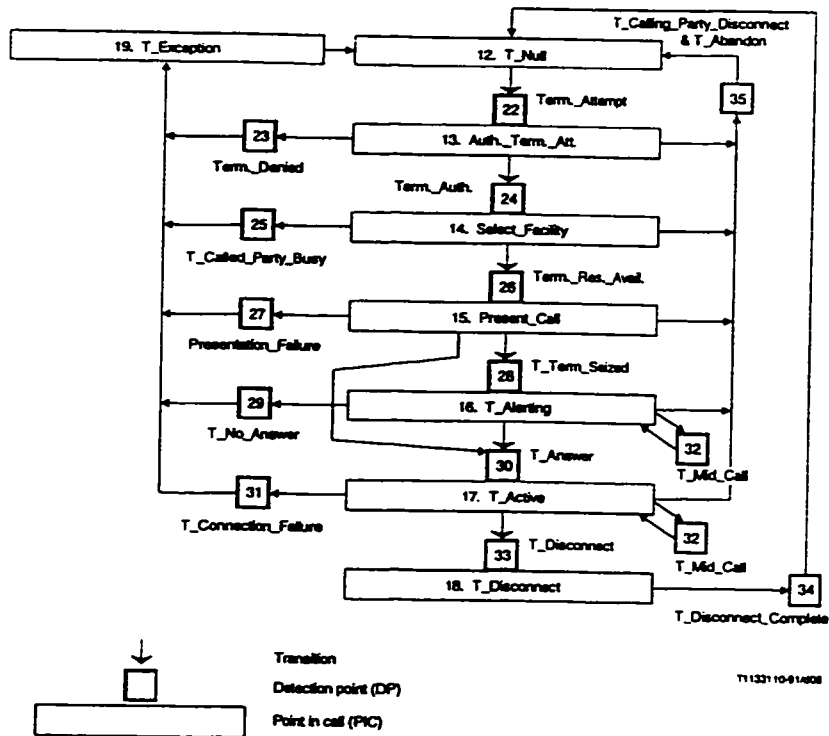
Appendix B. IN Basic Call State Model

The *Basic Call State Model* (BCSM) [15,18,19,28,29,72] is the set of states, transitions, and detection points, which are used to illustrate the different steps that a call can go through, from origination (i.e., when the user goes off-hook) to termination (i.e., when the user goes on-hook). The call model is separated into two parts: (1) an originating call model, and (2) a terminating call model. Both sides of the *Basic Call Model* (BCM) includes a number of *Points In Calls* (PICs), *Detection Points* (DPs) and transitions. PICs identify call control function activity required to complete one or more basic/normal call/connection states of interest to IN service logic instances. DPs refer to an event in basic call processing that is made visible to IN service logic. The detection of a DP event allows the IN service features to supplement call processing (i.e., IN capabilities). In the case where a DP event is not detected the call processing continues without involving any other functions (i.e., it continues with normal call processing). A transition indicates the normal flow of standard call processing from one PIC to another. Events cause transitions in and out of PICs.

The originating and terminating BCSM are respectively shown in Appendix B.1 and Appendix B.2. These figures are taken from recommendation *Q.1204 - Intelligent Network Distributed Functional Plane Architecture* [72].



Appendix B.1 Originating Basic Call State Model



Appendix B.2 Terminating Basic Call State Model

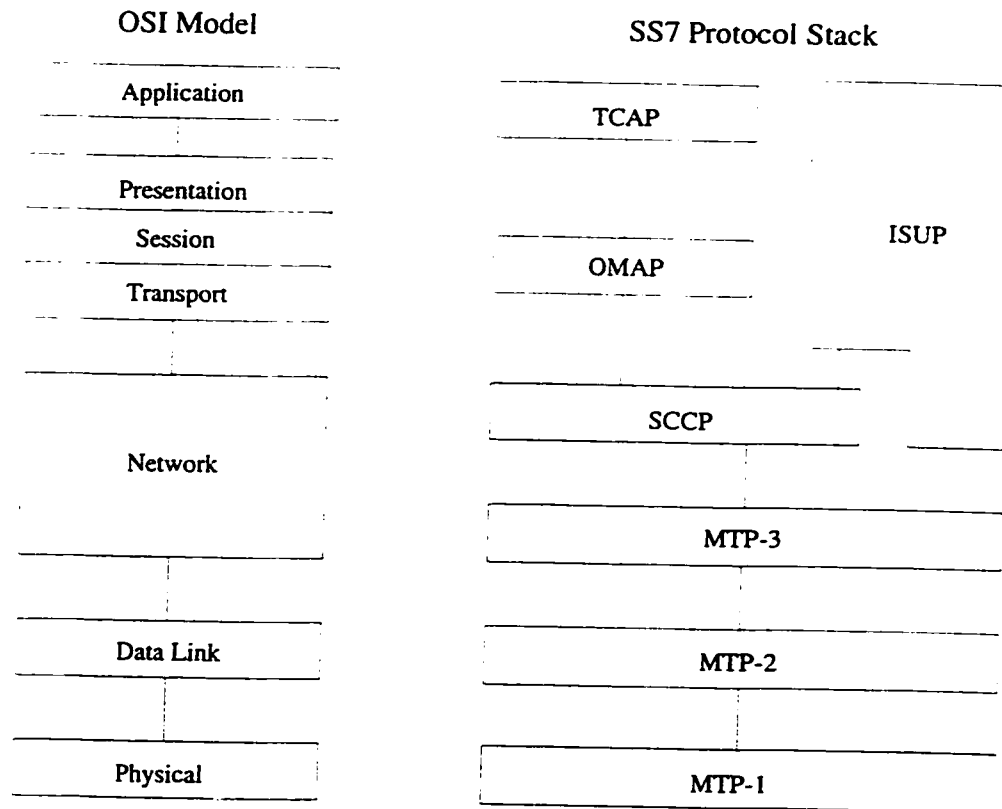
Appendix C. Signaling System Number 7 Protocol

The *Common Channel Signaling* (CCS) network [32,67], also known as Non-Associated Signaling, is a separate network which operates in conjunction with the PSTN. The PSTN provides voice and data circuits, while the CCS provides for the management of connections and transportation of signaling and database transactions (i.e., establishment of calls, routing of IN calls, etc.). The CCS network comprises all physical entities which interfaces the signaling network (i.e., SSP, SCP, IP, SN) plus the physical entities part of the signaling network itself. The most important physical entity of the signaling network is called *Signaling Transfer Point* (STP) [31]. STPs are always implemented in a mated pair fashion and are interconnected by a series of different links (A links, B links, C links, D links, E links and F links), which are usually 56 Kbytes per second packet trunks. The STP's task is to perform SS7 message routing and CCS network management.

The IN operates in the *Signaling System Number 7* (SS7) protocol environment [30,31,32]. It is also known as the PSTN interoffice (between SSPs) signaling protocol. The SS7 protocol is an international standard. It defines the rules for the operation of the CCS and its supported services, and provides highly reliable data transfer with low delay and without loss or duplication. The SS7 protocol comprises a stack of protocols in the same manner as the OSI model. The SS7 protocol layers are explained next and depicted in Appendix C.1.

- *Message Transfer Part* (MTP) layers provide reliable transfer of signaling messages between STPs. MTP functions include specifications for the signaling link interface and connection, error detection, recovery and retransmission, CCS network routing, load distribution and, management and flow control.

- *Signaling Connection Control Part (SCCP)* provides logical signaling connections within the CCS networks as well as a connectionless transfer mode for signaling units. These functions may require translating information part of the messages into routing addresses used by the MTP.
- *Integrated Services Digital Network User Part (ISUP)* is used to provide interexchange signaling to support circuit control and ISDN access services (i.e. ISDN D-channel signaling, trunk setup, etc.).
- *Operations, Maintenance, and Administration Part (OMAP)* provides the capabilities to support the operation of the CCS network.
- *Transaction Capabilities Application Part (TCAP)* [2] provides the ability to exchange operations and information using non-circuit control signaling. This protocol is used to convey call processing instructions in order to access resources, such as the SCP. For instance, in the case of the freephone service (800/888 like numbers), a TCAP message could contain information such as SSP ID, carrier, the originating number, the 800 number dialed, PIN number, etc.



Appendix C.1 Signaling System Number 7 Protocol Stack

Appendix D. Freephone and Calling Card Service

A description of the ITU-T *Freephone* (FPH) service and the *Account Card Calling* service (ACC) is given next¹⁰. The description is taken for recommendation *Q.1211 - Introduction to Intelligent Network Capability Set-1* [33]. As well, we present a detailed description of the freephone and calling card service.

1. ITU-T Freephone Service (FPH) Definition:

Description No. 1

This service allows a reverse charging, the subscriber accepting to receive calls at his expenses and being charged for the whole cost of the call.

Description No. 2

Freephone allows the served user having one or several installations to be reached from all or part of the country, or internationally as appropriate, with a freephone number and to be charged for this kind of call.

By using this service, a user can dial any 800/888 number at the expense of the customer (i.e., the user does not have to pay any fees). This service is widely used by government sectors and businesses for marketing and information services. When a user dials an 800 number, a connection is established to the closest SSP. A trigger is then activated, for which the SSP knows that the call requires special treatment (i.e., IN capabilities). The SSP therefore suspends normal call processing, encodes and launches a TCAP message via

¹⁰ The freephone service is also known as 800/888 numbers, and the account card calling service is also known as calling card numbers.

the CCSS7 network to the proper SCP freephone service database. A TCAP message contains information such as the dialed 800/888 number, the calling party's *Automatic Number Identification* (ANI), the originating SSP ID, time of day, etc.

The STPs, which are signaling packet switches of the signaling network perform *Global Title Translation* (GTT) where a *Destination Point Code* (DPC) identifies one of the SCPs containing the freephone service database. The STPs are used to route the signaling messages to the proper SCP over the common channel signaling network. The exchange of information between the SSP and SCP (via the CCS network) is transparent to the user. However, because of the SCP database query, the user may perceive the end-to end delay to be longer compared to a normal call establishment (i.e., when IN capabilities are not needed).

When the SCP receives the TCAP message, the service database reads the appropriate service logic program of the freephone service (the called 800/888 number). It then interprets call-handling instructions of the “*virtual*” number in order to determine the carrier, the destination number of the numbering plan (i.e., the real physical number) based upon criteria such as geographic origin of the call, time of day, day of week, etc.

When IN call processing is completed, the SCP encodes and transmits a TCAP response message to the originating SSP with the required information. Upon reception of the TCAP message, the originating SSP uses the information to establish the call setup, by routing the call through the transport network switches (i.e., local exchange, transit exchange, SSPs, etc.) to the appropriate destination.

2. ITU-T Account Card Calling (ACC) Definition:

Description No. 1

The account card calling service allows a user to make a call from any card reading telephone and have the charges for the call automatically debited to a domestic or business account number as defined by the card content. The user is given an access code and a *Personal Identification Number* (PIN). The user invokes the service by dialing the access code and on request, enters his PIN and “wipes” the card through the reader. The system validates the information and gives the user an indication of acceptance. The user can then proceed in the usual manner of making a call.

Description No. 2

The account card calling service allows subscribers to place calls from any normal access interface to any destination number and have the cost of those calls charge to the account specified by the ACC number.

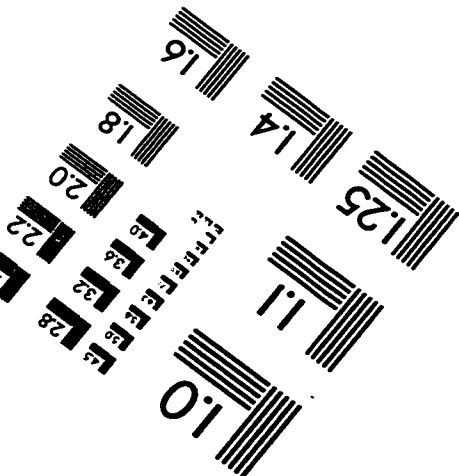
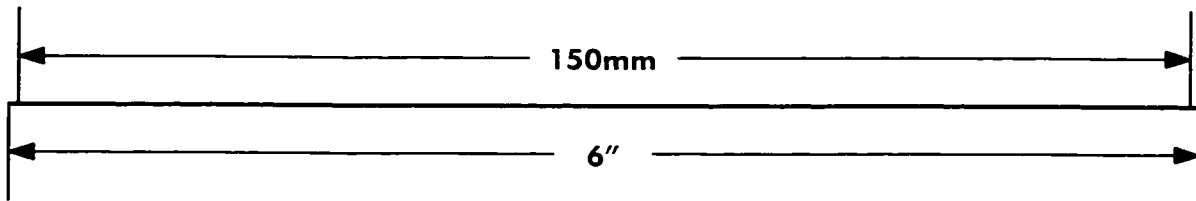
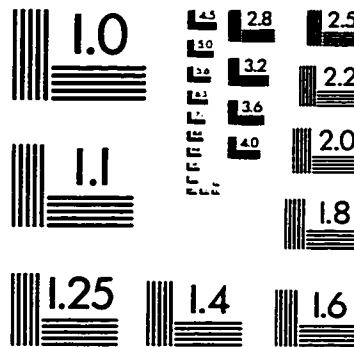
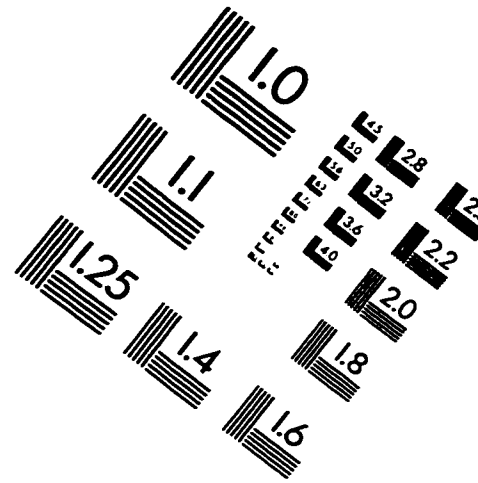
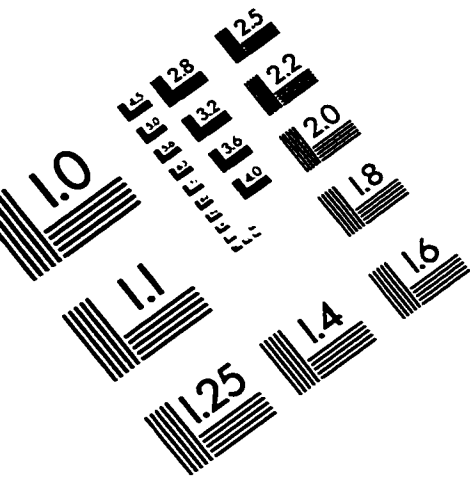
Description No. 3

This service allows the caller to be automatically charged on a telephone account subscribed with the network operator, for any type of outgoing call. The caller has to dial his card number and a PIN, then the called number.

By using this service, a user can place calls anywhere and anytime, from any normal access interface. The fees are billed to the calling card number, for which the user receives monthly statements. In order to access such service, the user usually dials the zero digit followed by the appropriate destination number. For example, the user dials the 0-NPA-NNX-XXXX number, where the zero digit precedes the area code (NPA) and destination number (NNX-XXXX). It is the zero digit (followed by any numbers) that triggers the SSP to suspend normal call processing, and begin special treatment (i.e., IN capabilities).

After a very short delay, the network (i.e., Intelligent Peripheral) issues a “*bong*” tonality, prompting the user with a voice announcement to enter his/her calling card number followed by the PIN number. Such voice announcements could be of the following form: “*Welcome to Bell Canada’s automated billing service... To charge this call to a calling card enter...* ”. Then, the SSP launches a TCAP message to the SCP in the same manner identified for the freephone service. Only the information placed in the TCAP message fields is different. The SCP then proceeds to validate the user’s input (i.e., calling card number + PIN) by accessing the service database. After validation, a TCAP response is sent back to the SSP, with the authorization or not to establish the call setup. If the authorization is valid, the call is then completed to the terminating party and the user hears ringing.

IMAGE EVALUATION TEST TARGET (QA-3)



APPLIED IMAGE, Inc.
1653 East Main Street
Rochester, NY 14609 USA
Phone: 716/482-0300
Fax: 716/288-5989

© 1993, Applied Image, Inc., All Rights Reserved

