DUBLIN CITY UNIVERSITY

School of Electronic Engineering

Network-oriented Load Control for SS.7/IN

Brendan Jennings BEng

Thesis submitted in partial fulfilment of the requirements for the award of PhD

Name of Supervisor: Prof. Thomas Curran

Submission Date: May 31st 2001

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

Signed: [illegible] ID No.: 93700938

Date: May 31st 2001

TABLE OF CONTENTS

Thesis Title: Network-oriented Load Control for SS.7/IN
Author: Brendan Jennings

Intelligent Network (IN) systems are today the prevalent means of providing services based on manipulation of voice call set-up in the public telecommunications infrastructure. As INs are evolving into more dynamic and open service delivery platforms, it is becoming increasingly difficult to dimension them to meet the performance requirements of a dynamically changing set of services. This is leaving networks more susceptible to overloads, which result in end-user dissatisfaction and revenue loss for the operator. In this context, load control strategies that protect against harmful overloads, are of critical importance.

Load control in IN systems and underlying Signalling System No.7 (SS.7) networks is characterised by node-oriented strategies centred on the protection of individual network resources, without regard to overall network performance. We argue that, given current and future increases in number and usage of IN services, node-oriented load controls cannot guarantee that desired performance levels are consistently achieved. Instead, we advocate adoption of network-oriented approaches that seek to optimise the usage of network resources from a global, rather then localised, perspective.

We first present a detailed analysis of standard SS.7 and IN node-oriented controls that demonstrates their inherent flaws and shows that they can interact poorly when invoked simultaneously. We then specify a load control that is based on explicit allocation of critical resources in a profit-optimal manner and that employs a token-based paradigm. This strategy is shown to perform at least as well as two node-oriented strategies in the context of controlling a single IN Service Control Point (SCP). More significantly, we introduce strategy enhancements that provide the following: co-ordinated, profit-optimal control over multiple resource types of varying characteristics; minimisation of the load of the SS.7 network; and reaction to SS.7 overloads or equipment failures. The enhanced strategy represents a highly efficient and flexible network-oriented control solution for SS.7/IN.

ACKNOWLEDGEMENTS

Firstly I want to thank Prof. Tommy Curran, my supervisor, for giving me the opportunity to pursue this work, for his guidance, ideas and advice on both technical and non-technical issues relating to the work and the preparation of the thesis, for helping with all aspects of the MARINER project and, above all, for his patience and good humour throughout.

A large part of this thesis is the result of a collaboration with Dr Åke Arvidsson at Ericsson Network Core Products; indeed it is he who was responsible for the original formulation of the token-based strategy and provided the inspiration for many of the strategy enhancements. Conversations with him were at all times fruitful, thought-provoking, and most importantly, very enjoyable. I thank him for all his help and advice.

Two former colleagues were also direct contributors to the work described in this thesis: Niall Lynch, who developed part of the SS.7 simulation model and Dr Fiona Lodge, whose IN simulation model I modified and who collaborated with me on the analysis of the interaction between SS.7 and IN load controls.

I would also like to thank all my other present and former DCU colleagues who have helped me in various ways over the years. Particular thanks are due to Dr Dmitri Botvich for his useful comments at various stages, Navin Wathan for his hard work and enthusiasm for MARINER, Conor McArdle for many thought-provoking discussions and Rob Brennan for numerous helpful ideas and suggestions. Conor and Rob also deserve great credit for taking such an interest in my progress over the past few months and for their detailed and insightful comments on chapter drafts.

I also thank all of the participants in the MARINER project, in the context of which much of the work described here was undertaken. Together with those mentioned above, MARINER-related discussions with the following helped me greatly in furthering my own work: Lars Angelin and Stefan Pettersson at Ericsson Software Technology; Dr Jeremy Pitt and Dr Javier Barria at Imperial College of Science, Technology and Medicine; Dr Christan Nyberg and Niklas Widell at Lund University; Dr Joachim Quantz at IKV++; and Frikkie Scholtz at Telkom South Africa.

I am very grateful to all my family and friends for not only tolerating my endless PhD-related worrying and moaning, but also for encouraging me to finish it. Special thanks are due to my parents, without whose support, encouragement and many sacrifices over the years I would never have had the opportunity to do this.

I am hugely indebted to my fiancé Edel, who has had to suffer my stress, tiredness and general ill-temper over the past months and years, but who has been understanding throughout and gave me the confidence to persevere when the work was progressing slowly. Thank you so much!

"I'm set free... to find a new illusion"

— Lou Reed (The Velvet Underground)

CHAPTER 1

INTRODUCTION

Recent years have seen a huge increase in the penetration of desktop computing in homes and businesses world-wide. This has been fuelled to a large part by the success of the Internet, which has, amongst other things, demonstrated the immense commercial potential of advanced multimedia services. Exposure to the Internet has raised expectations of the services offered by the public telecommunications infrastructure. End-users anticipate that video telephony, interactive multi-user games, e-commerce applications and a wide range of other complex services, will soon be available at low-cost and regardless of physical location or terminal equipment capabilities. Whilst the main technological components required to realise this vision are available, there remains significant technical, regulatory and economic challenges in deploying these components in a manner that meets the demands of a volatile and highly competitive market.

As the level of interconnection between fixed, mobile, Internet and enterprise networks grows, a key component in ensuring their ongoing success will be the availability of a common platform for the delivery of advanced communications services. Today the *Intelligent Network (IN)* is the prevalent means of realising services based on manipulation of voice call set-up and is widely viewed as a starting point for the integrated service delivery platform of the future. Some widely used IN-based services are Freephone, Prepaid, Televoting and Number Portability.

IN-based services are executed under the control of service logic residing in a dedicated network node called a *Service Control Point (SCP)*. End-user requests for IN services are detected at network switches, known as *Service Switching Points (SSPs)*, which pass them on to an SCP for service execution. In providing services, SCPs may be assisted by *Service Data Points (SDPs)*, which house databases containing service and end-user data, and *Intelligent Peripherals (IPs)*, which provide automated announcements and digit collection. Communication between the various IN network elements is facilitated by the highly reliable and robust *Signalling System No. 7 (SS.7)* network. The cornerstone of the IN concept is the separation of service logic from the network switches. This separation has the advantages of greatly reducing the time from service conception to deployment, decreasing the dependence of network operators on particular switch vendors and increasing the potential for the provision of services by third party providers.

As network operators are increasing the range of IN services they themselves offer and also opening their networks to other service providers, the task of managing network resources so that end-users receive acceptable quality-of-service is becoming problematic. In particular, it is extremely difficult to plan and dimension networks that can consistently meet the performance requirements of a dynamically changing set of complex services and their associated resource usage demands. For this reason INs are becoming more susceptible to *overload*, the situation where the traffic load offered to one or more network resources exceeds the resource's capacity to process that traffic in a timely manner. Overloads significantly degrade service session completion rates and result not only in lost revenue for the operator, but also in high levels of end-user dissatisfaction. The latter is very significant in a deregulated market, where end-user perceived quality-of-service can be an important differentiator between competing service providers.

In a rapidly evolving IN environment it is important to employ efficient and flexible *load control strategies*, which provide automatic detection of overload with some means of re-routing or reducing the traffic offered to affected resources, or take measures to actually prevent the overload onset. Currently deployed IN load control strategies are *node-oriented* in nature, being designed to protect individual network nodes during overload. Given current and future increases in the number and usage of IN services we argue that node-oriented strategies alone cannot guarantee that IN performance targets are consistently achieved. Because of their localised nature they are not able to control traffic in a manner that is optimal from the perspective of the whole network. In addition, individual node-oriented controls tend not to interact effectively with each other, so that often they become ineffectual, or even cause the propagation of overload to previously unaffected network resources.

The objectives of this thesis are to demonstrate the poor performance of standard SS.7/IN node-oriented load controls and to specify and evaluate a *network-oriented* IN load control strategy, which seeks to optimise the resource usage in the network as a whole, rather then on a localised basis. The strategy we propose provides for co-ordinated, profit-optimal control of multiple IN resources and protection against overloads in the SS.7 network. We believe it provides the foundation for a managed load control facility capable of optimising the performance of existing and future IN-based service platforms.

1.1 MAIN CONTRIBUTIONS

This thesis makes contributions in the research area of load control for distributed telecommunications systems. Specifically, it addresses load control in SS.7/IN environments, demonstrating both the drawbacks of current approaches and the potential benefits of adopting well-designed network-oriented strategies. The main contributions are:

- A comprehensive examination and appraisal of the performance of currently deployed SS.7 and IN load control strategies, incorporating:
  - A survey of the public literature relating to aspects of the operation and performance of load control strategies for SS.7 and IN, together with an indication of some issues not addressed therein;
  - An analysis of the operation of the standard SS.7 and IN load controls, using quasi-analytic and simulation models employing an SS.7/IN network topology that is more realistic then those employed in models described in the literature. The models are used to investigate in detail some aspects of control operation not addressed in previous studies;
  - An analysis of the interaction between SS.7 and IN load controls in the presence of various combinations of SS.7 and IN resource overloads;
  - An outline of a quasi-analytic modelling approach for the transient analysis of SS.7/IN load controls in the context of realistic network topologies. The approach is a significant enhancement of an approach originally described by [Zepf and Willmann, 1991];
- A new framework for network-oriented IN load control, founded on the use of a token-based paradigm for governing the allocation of network resources, incorporating:
  - A base strategy for SCP load control by means of explicit, profit-optimal allocation of processing capacity (in the form of tokens) to multiple SSPs. Also provided is a detailed comparative analysis of the performance of this strategy with two other SCP controls;
  - Enhancements to the base strategy that facilitate centralised, distributed or distributed hierarchical load control of multiple SCPs of varying processing characteristics and supported service mix (we refer to these as *heterogeneous* SCPs);
  - Enhancements to the base strategy that facilitate co-ordinated, profit-optimal load control of multiple SCPs and SDPs, together with an indication of how the strategy could be further enhanced to provide control over other resource types;
  - Enhancements to the base strategy that optimise the routing of service requests to minimise SS.7 network load and react to the onset of SS.7 resource overload;
  - An outline of how the framework could be further developed and applied to load control problems in other application domains.

1.2 MAIN CONCLUSIONS

The overriding conclusion of this thesis is that standard node-oriented SS.7/IN load control strategies are seriously flawed and that a network-oriented approach, such as the proposed token-based strategy, offers a more optimal, robust and flexible solution. In some more detail the main conclusions are:

- Standard SS.7 and IN load controls are badly designed, in that they fail to provide adequate protection during overload and are unfair in their treatment of different service types. This is due to their reactive nature, reliance on statically-defined control parameters and inability to make globally optimal load control decisions;
- The proposed token-based SCP load control strategy protects, in a profit-optimal manner, individual SCPs from overload. The strategy is predicated on the explicit allocation of SCP processing capacity, which ensures that pre-specified target utilisation is never exceeded. Allocation of processing capacity is based on service specific profit ratings, thus more valuable services (from the network operator's viewpoint) are protected during periods of high load. The token-based strategy performs significantly better then a widely-deployed node-oriented strategy and at least as well as a similar profit-optimising control;
- The token-based strategy can provide globally co-ordinated load control for INs containing multiple SCPs of varying processing characteristics. In doing so it provides a means to introduce the dynamic routing of service requests by SSPs to any SCP, whilst simultaneously ensuring that SCP overloads do not occur. Realisations of the strategy can be structured to mirror the topology of large-scale INs;
- The token-based strategy can take into account that different types of resource will become the system bottleneck in different traffic conditions. The strategy adapts its behaviour to protect different types of resource by explicitly allocating tokens for each type of resource required for the successful completion of a service session;
- The pheromone-based SCP selection and delay-based load throttling enhancements to the token-based strategy respectively provide an effective means to minimise the IN signalling load carried by the SS.7 network and the capability to react to the onset of resource failures or overloads in the SS.7 network;
- Tokens provide an understandable and useful non-technical abstraction of the operation of load control in the IN. For example, the allocation of tokens can be directly related to *Service Level Agreements (SLAs)* agreed between a network operator and the customers on the behalf of which particular IN services are offered;

1.3 THESIS OUTLINE

Chapter 2 places in context the work described in this thesis. It commences by introducing the area of load control in distributed telecommunications systems, outlining the reasons why load controls are required, standard methods of load control and the attributes of an ideal control. Brief introductions to the structure and operation of SS.7 and IN, with particular emphasis on deployed and proposed load controls, are then provided. This is followed by a comprehensive overview of the public literature in the area of SS.7/IN load control. The chapter concludes with an identification of open research issues, together with a justification for the research work undertaken.

Chapter 3 provides background information on the various methods and paradigms required for a full understanding of both the simulation/analytic models and the load control strategies described in subsequent chapters.

Chapter 4 presents the results of a detailed analysis of the operation of standard SS.7 and IN load control strategies. It commences by introducing the SS.7/IN network topology used in the study and proceeds to describe the quasi-analytic and simulation models of this network. Seven separate overload scenarios are investigated, involving the overload of various subsets of network resources. Through these scenarios both the independent and combined operation of SS.7 and IN plane controls is examined. The chapter concludes with a summary of the limitations of currently deployed controls and a discussion of what attributes a new control strategy must possess in order to avoid similar drawbacks.

Chapter 5 presents the base version of the token-based load control strategy. This version provides control over a single SCP, by periodically calculating and sending allocations of tokens to SSPs, which use them to control the number of accepted service requests. Performance of the strategy is compared with that of the *Automatic Call Gapping (ACG)* control and another proposed strategy that employs mathematical optimisation.

Chapter 6 presents and analyses three major enhancements to the token-based strategy. The first allows for profit-optimal control of an IN containing multiple, possibly heterogeneous, SCPs. This enhancement provides a co-ordinated means of introducing the dynamic routing of service requests by SSPs to more than one of the SCPs in the network. The second enhancement involves the profit-optimal control of multiple resource types. Finally a strategy enhancement that takes into account the potential for SS.7 overloads is presented. In this enhancement SSP round-trip delay measurements for messages sent to SCPs are used to select favourable SCPs in order to minimise SS.7 network load and to react to failures or overloads.

Chapter 7 summarises the main results of the work, discusses issues relating to implementation of the token-based strategy in a real IN, outlines other potential applications of the strategy and identifies directions for further research.

CHAPTER 2

BACKGROUND INFORMATION AND LITERATURE REVIEW

In this chapter we provide a brief overview of the state-of-the-art regarding load control in the signalling-plane of public telephony networks. §2.1 discusses the importance of load control and outlines the necessary characteristics of a network-oriented load control strategy. §2.2 and §2.3 provide an overview of SS.7 and IN respectively, focussing in particular on the load control strategies employed in each. §2.4 then provides a comprehensive review of the publicly available literature relating to the SS.7 and IN controls. Finally, §2.5 draws conclusions from the literature review and outlines the approach described in the remainder of the thesis.

2.1 LOAD CONTROL IN TELECOMMUNICATIONS NETWORKS

All communications systems and networks have a finite capacity that limits the volume of traffic (telephone calls, file transfers *etc.*) they can successfully process within a given time period. In situations where the usage of the network by end-users is higher then normal the traffic arrival rate may exceed the maximum service rate. This situation is referred to as an *overload*. In this thesis we deal with (public) telecommunications networks supporting services involving an end-user initiating a voice communication with one or more other end-users or automated announcement facilities. Typically overloads of these networks occur due to either equipment failures or mass call-in events (triggered by mass media stimulated competitions and telephone polls), during which a large segment of the population initiates calls to a single number within a short time-span.

In realising telephony services, control information must be passed between various nodes in the network. During an overload the messages carrying this information encounter overloaded network resources and are queued in input buffers. Queuing increases delays, which, in sustained overloads, often grow to the degree that end-users abandon their service attempts. Messages relating to abandoned requests remain in the network and, since they do not result in revenue for the network operator, waste processing resources that could otherwise be spent profitably. Furthermore, end-users tend to reattempt earlier abandoned service requests, which increases offered load and therefore delays, leading to an even higher abandonment rate. This results in positive feedback, resulting in all network resources being fully utilised but the amount of utilisation relating to useful work, termed *throughput*, rapidly approaching zero.

As described above, overloads degrade network throughput, which is highly undesirable from the point of view of both the end-user and the network operator. Therefore, it is essential that steps be taken to minimise the impact they have on network performance. *Load Control* refers to strategies and mechanisms used to manage network traffic so that network resources are efficiently used and service completion rates are maximised in all traffic conditions, in particular during overload. A broad spectrum of load controls have been proposed and implemented. In general they employ one or more of the following techniques:

- *Message Dropping:* Messages are dropped without notifying their originator. For example, messages arriving to a full queue are often dropped, as are messages that have been stored in a queue for an unacceptable delay. This technique is generally only efficient during short periods of overload, since during sustained overload, reattempts resulting from prematurely terminated sessions will tend to further exacerbate and prolong the overload;
- *Reactive Control*: An overloaded resource employs an *overload detection* algorithm that explicitly notifies the originators of message traffic when it detects overload onset. The intention is that the traffic originators will respond by reducing traffic towards the affected resource;
- *Message Throttling:* Messages are rejected (or *throttled*) if certain requisites are fulfilled. Throttles generally reside at the point of ingress of service requests into the network and are invoked in response to overload indications from affected resources and/or on the basis of locally-available measurements. Throttles are generally progressive in nature, rejecting an increasing proportion of requests as the overload situation grows in severity;
- *Prioritisation:* Messages may be assigned *priorities* on the basis of their relative importance, the intention being that throttles will initially react to overload by throttling the least important messages, with higher priority messages being affected only as a last resort. Typically messages relating to ongoing service sessions are given priority over initial service requests, in order to avoid termination of ongoing sessions;
- *Alternative Routing:* Networks are generally designed to ensure that more than one communication path exists between all pairs of nodes. During an overload it may then be possible to route messages over an alternative path, thereby bypassing the affected part of the network.

Traditionally, load control strategies have been *node-oriented* in nature; that is, they focus on protecting individual network nodes from overload. We argue that, given the current trend towards more open and dynamic networks, node-oriented strategies alone cannot guarantee that desired performance levels are consistently achieved. In particular, node-oriented strategies cannot control traffic in a manner such that overall network throughput is maximised, or ensure that network operator goals like profit maximisation and protection of important services are achieved. Moreover, node-oriented controls have in certain scenarios been shown to further worsen an overload situation by interacting badly with each other and causing overload propagation [Houck *et al.*, 1994; Atai and Northcote, 1996]. We focus instead on the development of *network-oriented* load controls, in which control decisions are co-ordinated throughout the network in order to achieve globally-optimal resource utilisation. More specifically, an ideal network-oriented load control strategy will satisfy the following generic requirements:

1. *Optimality*

   The optimality of a strategy refers to its ability to allocate network resources in a manner that maximises some global measure of network performance. We use *profit* as a global measure, since it encapsulates the relative importance of services to the network operator;

2. *Efficiency*

   The efficiency of a strategy refers to its ability to maintain resource throughput at the maximum allowed level during overload. In addition, an efficient strategy will place minimal overhead on critical network resources;

3. *Robustness*

   The robustness of a strategy refers to its ability to provide optimal and efficient control in the face of changes in network state and varying input traffic profiles;

4. *Scalability*

   The scalability of a strategy refers to its ability to provide optimal and efficient control in different networks, regardless of their size or topology;

5. *Responsiveness*

   The responsiveness of a strategy refers to its ability to react rapidly to changes in offered traffic levels;

6. *Stability*

   The stability of a strategy refers to its ability to behave in a controlled manner: its operation should not result in fluctuations or oscillation in resource utilisation;

7. *Fairness*

   The fairness of a strategy refers to its ability to distribute resources to service requests in 'a fair manner.' The interpretation of 'a fair manner' is contentious. For example, a fair strategy has been defined variously as one in which the probability of rejection is the same for all end-users, regardless of location [Lodge, 2000] or as one in which only those services causing an overload should be throttled [Tsolas, 1992];

8. *Simplicity*

   The simplicity of a strategy refers both to the ease which the strategy can be understood and to the ease with which it can be implemented and administered.

2.2 SIGNALLING SYSTEM NO.7 (SS.7)

Signalling in telecommunications networks consist of communications between customer premises equipment, switches, network-based servers and databases, to manage and complete end-user service sessions. The signalling protocols typically used by telecommunications network operators and equipment vendors for communications between the various core network elements are those standardised by the ITU: the *Common Channel Signalling System No. 7 (SS.7*[1]*)* protocol suite [ITU, 1993a].

The SS.7 signalling network can be viewed as a packet-switched data network overlaid on the switched voice-circuit network. As illustrated by Figure 2.1, an SS.7 network consists of *Signalling Points (SPs)* and *Signalling Transfer Points (STPs)*, connected together by *Signalling Links*. SPs correspond to the switching exchanges in the switched circuit network and to other elements such as network databases or management centres. STPs are packet switches which route signalling messages towards their destination SP.

SS.7 specifications mandate that an SS.7 network meets stringent availability targets, for example the maximum allowed downtime for signalling routes is 10 minutes per year. To meet these targets SS.7 networks must be very robust to failures, so they are designed to incorporate a high degree of functional redundancy. To achieve redundancy SS.7 network elements are usually deployed in mated, redundant pairs and are connected by means of a mesh topology of signalling links, thus ensuring that signalling messages can be routed on alternative paths when a failure occurs at a single point (link or node) in the network.

In order to improve network robustness the SS.7 protocols incorporate a comprehensive range of procedures for error/failure detection, alternative routing, load control and failure recovery, in addition to procedures for their core functionality of transferring signalling messages between SPs. The SS.7 protocol and its relationship to the *Open Systems Interconnection (OSI)* seven-layer model [ITU, 1989] is shown in Figure 2.2.

[1] Other widely used acronyms/abbreviations for SS.7 are SS7, C7 and "Number 7."

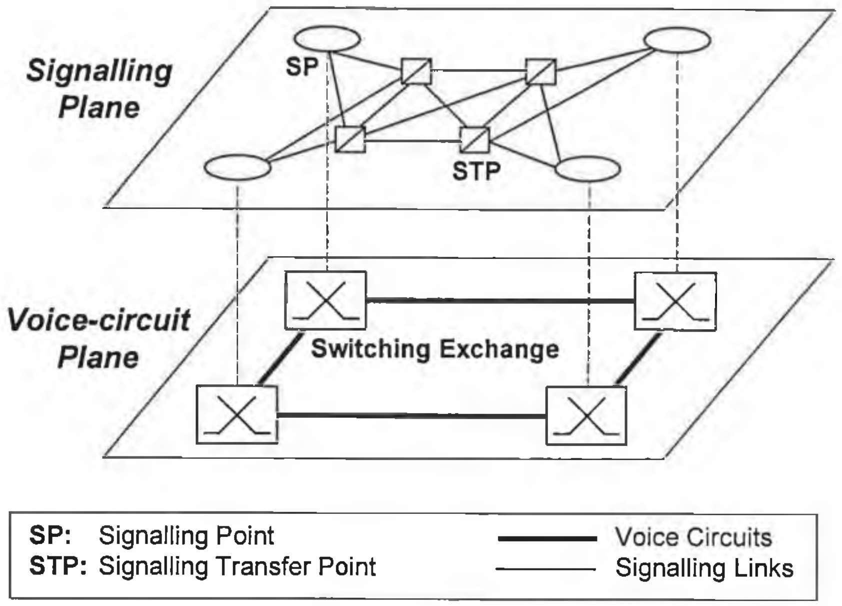


Figure 2.1: Components of an SS.7 network.

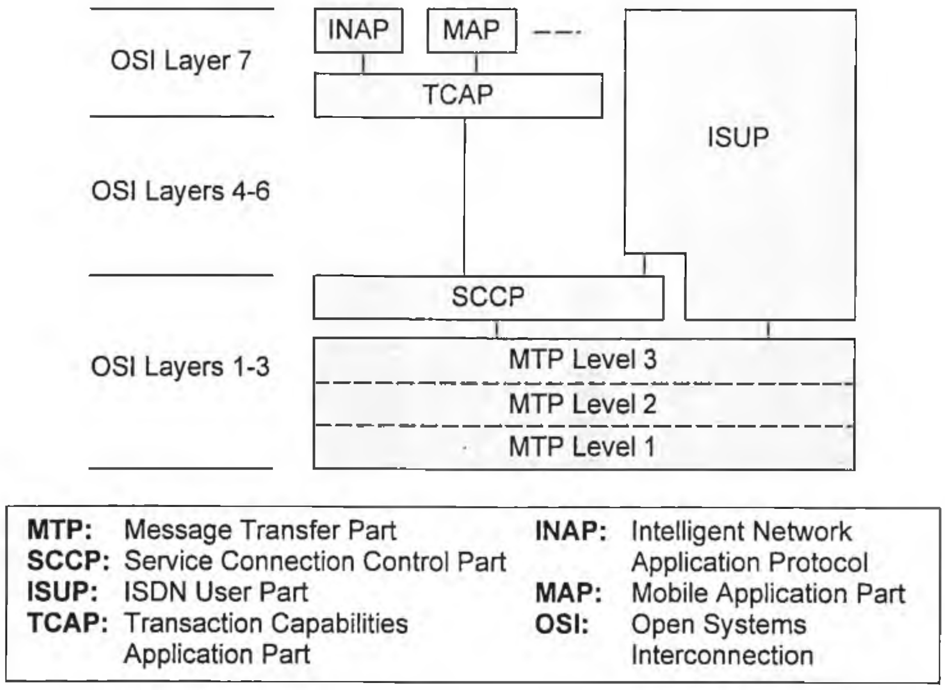

OSI Layer 7
OSI Layers 4-6
OSI Layers 1-3
INAP
MAP
TCAP
ISUP
SCCP
MTP Level 3
MTP Level 2
MTP Level 1
MTP: Message Transfer Part
SCCP: Service Connection Control Part
ISUP: ISDN User Part
TCAP: Transaction Capabilities Application Part
INAP: Intelligent Network Application Protocol
MAP: Mobile Application Part
OSI: Open Systems Interconnection

Figure 2.2: The SS.7 protocol suite and its relationship to the OSI seven-layer model.

We now provide a brief overview of the functionality incorporated in the various SS.7 protocols:[2]

[2] Note that the protocol suite is designed to be modular and extensible; in particular new protocols availing of the services of TCAP have been introduced as networks have evolved. For this reason INAP and MAP protocols shown in Figure 2.2 are not part of the set of SS.7 protocol specifications *per se*, rather they are part of the specifications for Intelligent Networks and second generation mobile telephony networks respectively.

Message Transfer Part – Level 1 (MTP1)

The MTP1 protocol defines the physical and electrical characteristics of the signalling links of an SS.7 network. Signalling links utilise 64 kbit/s digital transmission channels[3] which are usually multiplexed on the same transport infrastructure as voice circuits.

Message Transfer Part – Level 2 (MTP2)

MTP2 provides link-layer functionality, ensuring that the two end-points of a signalling link can reliably exchange signalling messages. It incorporates procedures for error checking, sequence checking, retransmission and basic flow control.

Message Transfer Part – Level 3 (MTP3)

MTP3 provides (partial) network-layer functionality. It ensures that signalling messages can be reliably delivered between two SPs in the SS.7 network, whether or not they are directly connected. All SPs within an operator's SS.7 network are assigned a unique *Point Code*, which is used for addressing purposes by MTP3: each signalling message carries an *Originating Point Code (OPC)* and a *Destination Point Code (DPC)*, as well as a service indicator, in its routing label. Static routing tables at each SP are used to specify outgoing links upon which signalling messages with particular DPCs are to be transferred.

MTP3 incorporates a range of procedures for dealing with failure conditions, for example procedures which inform remote parts of the network of failures, procedures which re-route traffic away from failed links and procedures for restoration of failed links. It also incorporates procedures for link overload detection and for informing traffic sources of link overload status (*cf.* §2.2.1 for a full description).

Integrated Services Digital Network (ISDN) User Part (ISUP)

The ISDN user part defines the messages used in the establishment and tear-down of voice and data calls over the *Public Switched Telephone Network (PSTN)*, and to manage the trunk network upon which they rely. In spite of its name, ISUP is nowadays used for both ISDN and non-ISDN calls (it is backwards-compatible with the Telephony and Data User Parts which predated it). The protocol also provides support for supplementary ISDN services such as User-to-User signalling.

Signalling Connection Control Part (SCCP)

SCCP provides two major functions that augment and enhance the addressing capabilities of MTP3. The first function is the capability to address applications within an SP (MTP3 can only send and receive messages to/from SPs as a whole). SCCP allows for explicit addressing of software subsystems within a signalling node through use of *Sub-System Numbers (SSNs)*.

[3] Except in the United States / Canada, where 56 kbit/s is the data rate of signalling links.

The second function is the ability to perform incremental routing using a capability named *Global Title Translation (GTT)*. Global titles, or *Global Title Addresses (GTAs)* are user part significant addresses, such as dialled digits or network terminal numbers, which do not explicitly contain the routing information (*ie.* the DPC and service indicator) required by MTP3 to successfully route a signalling message. GTT, normally residing at an STP, translates a global title into a DPC/service indicator pair (this is typically a many-to-one translation). Use of GTT frees user parts from the burden of maintaining lists of every potential SS.7 PC to which they may have to send a message. This greatly simplifies network management and makes it easier to add new SPs or modify routing plans.

An additional, but rarely used, capability of SCCP is the provision of a fully connection-oriented message transport service.

Transaction Capability Application Part (TCAP)

TCAP consists of two sub-layers: the *Transaction Sub-Layer (TSL)*, which provides a thin end-to-end connection for the transfer of remote operations using the *Component Sub-Layer (CSL)*, which is based on the OSI *Remote Operations Service (ROS)* [ITU, 1994a]. This is essentially a remote procedure call like capability. The specifics of the remote operations and their returns are described as TC-user protocols using the *Abstract Syntax Notation One (ASN.1)* [ITU, 1994b] and encoded using the *Basic Encoding Rules (BER)* [ITU, 1994b]. For example, the remote operations that define the interactions between switches and network servers (containing service-specific intelligence) are defined by a TC-user protocol called the *Intelligent Network Application Protocol (INAP)* [ITU, 1993b].

2.2.1 SS.7 Load Controls

SS.7 standards incorporate a range of complex network management functionality aimed at preserving connectivity between signalling points and acceptable performance levels in the event of failures of signalling links/nodes or network overloads. SS.7 load controls are complex and involve the interaction of a number of procedures residing in the MTP, SCCP and user parts. The central approach used in these controls is that of employing 'choke packets,' which are sent from the point at which overload occurs to the signalling points where the traffic causing the overload is originated. These *traffic sources* then reduce the level of traffic sent to the relevant destination(s), thereby alleviating the overload. In this section we describe the detailed operation of the various SS.7 load control mechanisms and identify those having the greatest impact on the operation of networks during overload.

2.2.1.1 MTP Controls

MTP load control mechanisms can be classified into two groups: those designed to protect signalling links from overload (*signalling network MTP controls*) and those designed to protect against nodal (SP/STP) overloads (*MTP nodal controls*). We outline the operation of both and comment on their relative importance in the context of actual SS.7 overloads.

Signalling Network MTP Controls

There are three separate versions of the signalling network MTP controls, known respectively as the *International Option (IO)*, the *National Option (NO)* and the *National Option with Congestion Priorities (NOCP)*. The two national options are used in the United States and Canada, whereas the IO is used elsewhere. In the IO control the MTP does not assign overload priorities to messages and any decision to throttle messages is taken only in the user parts.[4] In the NO control the MTP can assign multiple overload levels to affected destinations that it reports to user parts, however message throttling is again performed only in the user parts. In the NOCP control, multiple overload levels are employed and the MTP bases message throttling decisions on these priorities, separately and in addition to whatever discard may occur in the user parts.[5]

In this thesis we restrict ourselves to analysis of the operation of the IO control, which is employed in the majority of SS.7 networks world-wide. This decision is justified as previous studies conclude that either there is little difference between the operation of the national and international controls [Manfield, 1994], or else the IO control performs significantly better [Rumsewicz, 1994a; Rumsewicz and Smith, 1995; Northcote and Rumsewicz, 1995].[6] In particular, since the IO throttles at the user part level, it can operate on calls rather then messages (as NOCP does), so it can be less susceptible to user part recovery procedures (§2.4.1.2 describes the negative impact these can have on control performance). Given the above, we can conclude that study of the IO control gives us a best case for the operation of the SS.7 controls. The description of the signalling network MTP controls in the remainder of this section is based on the international specifications.

As alluded to previously, the signalling network MTP controls are designed to react to overload of signalling links. MTP3 entities continually monitor the overload status of all links emanating from a node. A link is deemed to be overloaded once the combined length of its transmission and retransmission queues exceeds an implementation-dependent *onset threshold* value (denoted L_O). Likewise, link overload is deemed to abate when the combined queue length falls below an implementation-dependent *abatement threshold* value (denoted L_A). The abatement

[4] Of course message discard will occur in the MTP when physical memory/buffer capacity exhausts.

[5] In most studies it is assumed that where NOCP is employed there is no user part load throttling. [Rumsewicz and Smith, 1995] compare versions of the control in which user part throttling is and is not present and conclude that the former is generally superior; we use NOCP to refer to this particular version of the control.

[6] We have more confidence in the conclusions of the latter three studies due to the fact that more detailed models, incorporating for example user part recovery procedures, were used and results were analysed in terms of call rather then message throughput rate performance.

threshold is set below the onset threshold in order to provide hysteresis during overload abatement.

Once a link becomes overloaded the associated node's MTP3 entity invokes the *Transfer Controlled (TFC)* procedure, which informs traffic sources, by means of *TFC messages*, that signalling routes using the affected link are overloaded. One in every m (default $m = 8$) arriving signalling messages to be sent over the overloaded link, trigger the sending of a TFC message to the message's originating SP.[7] Transmission of TFC messages continues until the status of the link returns to normal. Note that no discard occurs in the MTP, hence all signalling messages are routed to MTP level 2 for transmission over the (overloaded) link. It is important to note that TFC messages indicate that a destination SP (not a specific link/route) is overloaded, hence traffic sources will throttle traffic on all routes towards a destination, regardless of whether they are actually overloaded or not.

The operation of the TFC mechanism when a signalling message arrives for a link currently designated as overloaded is illustrated by Figure 2.3. The sequence of events is as follows: the arriving message is the $m = 8^{th}$ to have arrived since the last TFC message was generated ***(1)***; the message is sent to the link for transmission ***(2)***; a TFC message with parameter indicating the overloaded destination (SPa) is generated and sent to the SP that originated the message (SPx) ***(3)***.

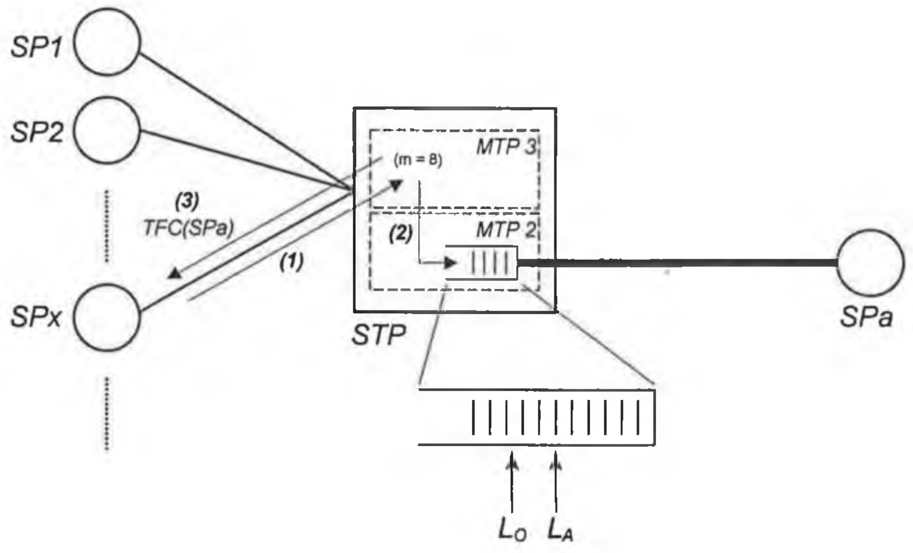


Figure 2.3: Illustration of the operation of the MTP level 3 overload detection and the TFC procedure.

Upon reception of a TFC the MTP3 entity in the receiving SP informs *all* local user parts that the specified destination SP is overloaded, by means of a *Congestion Indication (CI)* encapsulated in a MTP-STATUS primitive. This is done for every TFC arriving for the affected

[7] In the case where the overloaded link is local to SP which itself originates traffic sent over the overloaded link a TFC message is not sent, rather the MTP level 3 entity indicates to the local user parts that the destination is overloaded by means of the MTP-STATUS primitive.

destination. Note that the MTP3 entity does not itself retain information regarding overloaded destinations, its only response to the arrival of TFCs is to send CIs to all local user parts.

MTP Nodal Controls

MTP nodal controls aim to address overload caused as a result of overload or outage of (SS.7 related) processors at signalling nodes (both STPs and SPs). Nodal controls are present in both MTP2 and MTP3.

The MTP2 nodal *flow control* is initiated when overload is detected (in an implementation-dependent manner) at the receiving end of a signalling link. The onset of overload is indicated to the transmitting side by means of special link status signal units; the transmitting side can take the link out of service if the overload persists beyond a duration controlled by a set of timers. More importantly, the receiving side can withhold acknowledgements, resulting in the queues at the transmitting side growing to the level where the signalling network MTP controls are invoked (albeit at neighbouring nodes); this process is known as *backpressure*.

For the case of the MTP3 *nodal controls*, overload detection is implementation-dependent and it is assumed that nodal overload results in the build-up of message queue lengths, which again triggers the signalling network MTP controls. Specifically we can assume that upon nodal overload detection, backpressure is applied to all local links, resulting in queue build-up at neighbouring nodes, then resulting in invocation of the TFC procedure as described above.

From the above description we see that nodal overload will always result in the sending of TFC messages to traffic sources, so load throttling behaviour will be very similar to that seen in the link overload case. In general we would expect node processing capacity to be provisioned so that it is not the bottleneck and therefore links will generally overload before SP processors. For these reasons we choose to focus solely on link overload and ignore the possibility that node processors may overload.

2.2.1.2 ISUP Controls

User part controls respond to overload detected by the MTP by throttling traffic destined for overloaded destinations. ISUP throttles traffic to the destinations indicated in CIs by reducing traffic in a stepwise manner, under the control of two timers, denoted *T29* and *T30*. Both the number and size of the traffic reduction steps are implementation-dependent. On receipt of the first CI for a destination SP, the traffic load to that destination is reduced by one step and both timers are started. Until *T29* expires, further arriving CIs for the destination in question are discarded; this avoids reducing the traffic level too quickly. If a CI is received after the expiry of *T29*, but before the expiry of *T30*, the traffic is reduced one more step and both timers are re-started. This process is repeated while CIs continue to arrive, until the maximum traffic reduction level is reached. If *T30* expires the traffic load is increased by one step and *T30* is re-started. This process is repeated until full load has been restored. The operation of the control is illustrated in Figure 2.4.

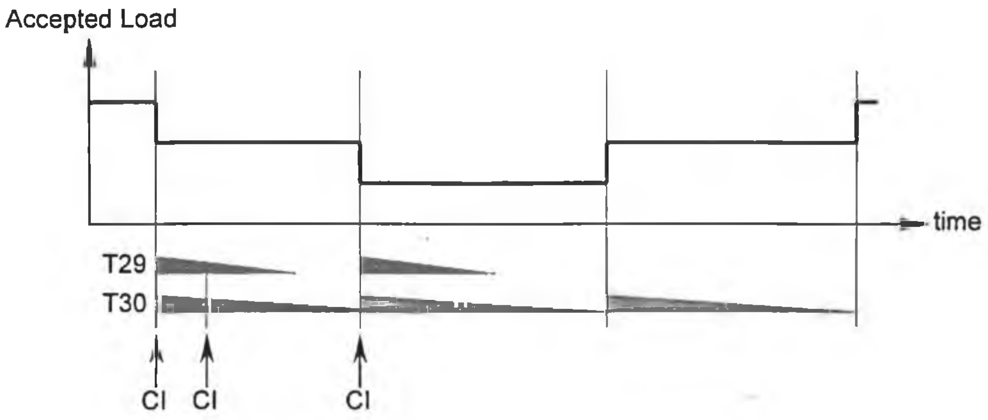


Figure 2.4: Illustration of ISUP load throttling.

While a number of models for the ISUP throttling process can be envisaged, it appears prudent to assign priorities to different ISUP message types in order to minimise the impact on higher-level call processing. An example assignment, in order of increasing priority, is as follows:

Priority 1. User-to-user signalling messages, which are generally associated with a general data service and can be regarded as non-essential to call processing;

Priority 2. *Initial Address Messages (IAMs)*,[8] these are the initial messages sent in an ISUP session, hence throttling them has the least impact on call processing;

Priority 3. Call progress messages (*Address Complete (ACM)*, *Answer (ANM)*, *Release (REL)*, *etc.*), these messages are associated with calls already in progress hence they should be given priority over IAMs;

Priority 4. Management messages, these messages should be throttled only as a last resort;

Using the assignment above load reduction steps could correspond to the throttling of different types of message. In addition, there could be a number of steps for each message type, for example IAMs could be throttled by 25%, 50%, 75% or 100% depending on the current level.

2.2.1.3 SCCP Controls

The only controls specified in current SCCP standards relate to SCCP class 3 service for connection-oriented data transfer with flow control. The flow control is realised by a window mechanism (*cf.* §2.3.1.1), where the size of the window is negotiated at connection establishment and is not changed during the data transfer phase. As noted previously SCCP class 3 is rarely used in practice, hence the operation of this control has negligible impact on actual network performance.

[8] A description of the roles played by IAM and other ISUP message types in ISUP telephony sessions is provided in §4.1.2.1, page 71.

In SS.7 networks employing the SCCP GTT service it is often not possible for MTP entities to send TFC messages to the true sources of overload traffic, instead they will be sent to the node at which GTT is performed, in many cases an STP. Even if TFCs can reach the source and CIs are passed to SCCP users, it will be impossible for these users to impose suitable throttles because they will have no knowledge of the point code addressing scheme used by the MTP. Given this, it is clear that SS.7 vendors must incorporate some form of load control for connectionless SCCP data transfer in their products, to provide control over the ever-increasing volume of SCCP traffic carried by networks. This has been recognised by ITU, which in January 1996 proposed a standard load control mechanism for connectionless SCCP classes [McMillan and Rumsewicz, 1996]. This control aims at ensuring equitable treatment of all MTP user traffic and is effectively a mirror of the ISUP control, incorporating stepwise traffic reduction controlled by timers.

2.3 INTELLIGENT NETWORKS

The Intelligent Network (IN) came into being in the mid-1980s as a way of de-coupling telecommunications service logic from the implementation of switching exchanges. This facilitated centralised service processing functionality, eased the deployment of new services and reduced the escalating complexity of exchanges. The IN provides the Public Switched Telephony Network (PSTN) with the infrastructure to provide advanced services such as Freephone and Number Portability. IN standardisation has taken place in ANSI [ANSI, 1999], ETSI [ETSI, 1994] and the ITU [ITU, 1997]. Unfortunately, this distribution of standardisation effort and the proprietary enhancements to IN by vendors have created a plethora of non-interworking IN solutions. However within geographical regions, for example the USA or Europe, there is sufficient agreement on standards that interworking is possible and, at least nominally, the ITU standards form the basis for international interworking.

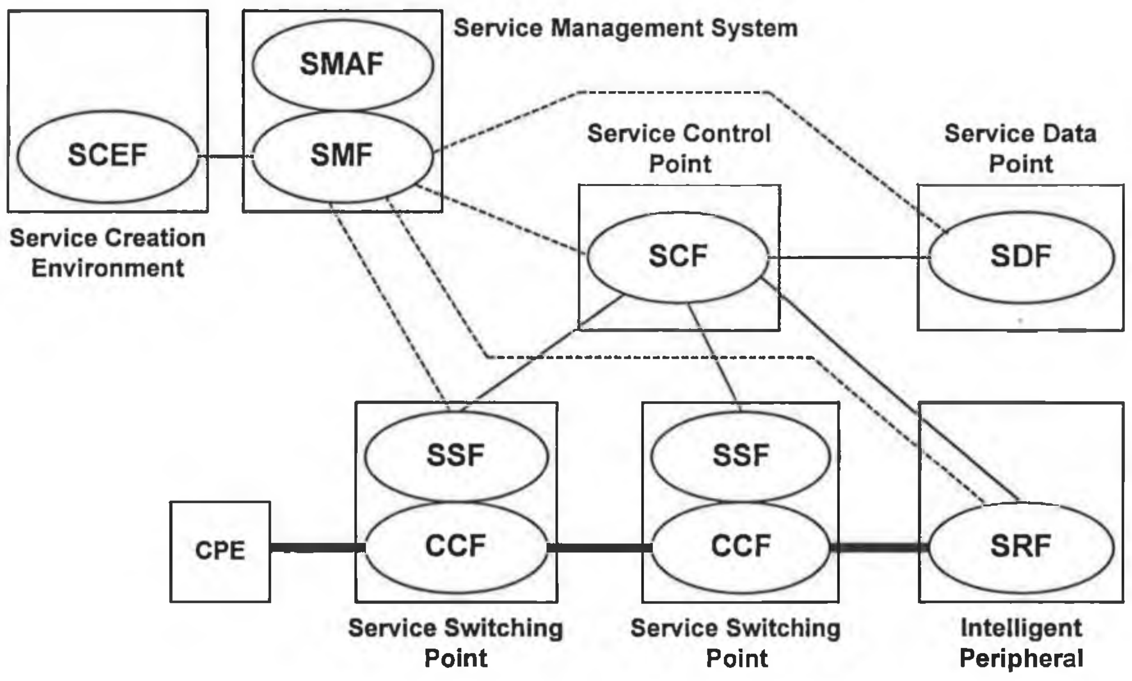
The basic IN model comprises a distributed functional architecture, containing functional entities that are collectively responsible for handling any telephony calls that involve service logic which is more sophisticated than that which can be provided by traditional call routing on the basis of dialled digits. Figure 2.5 below shows the functional entities, their relationships and illustrates how these entities are typically grouped into platforms in a network.[9] Control relationships between the functional entities are normally supported by SS.7 signalling infrastructure.

Each *Functional Entity (FE)* plays a role in the completion of calls requiring additional processing prior to routing through the network. The CCF is responsible for the routing of

[9] Other groupings are also employed, for example SCPs often contain both SCF and SDF functional entities.

normal calls through the PSTN and detects when an IN service session should be initiated, typically through examination of dialled digits. For an IN call the CCF cedes control to the co-located SSF, which initiates an INAP session with the SCF in order to request any special handling instructions for this call. The SCF will normally respond with a destination address (number) to which the call should be routed. Many variations on this basic scheme are possible, involving the SCF querying a database (the SDF) or instructing the SSF to connect the call to a recording playback device controlled by an SRF. The SCF may also instruct the SSF regarding specific billing arrangements to be used when dealing with this call. Supporting entities provide functionality such as service creation (by the SCEF) and service management (by the SMF and SMAF); they are not directly involved in the processing of IN service sessions.

IN has been structured into a number of modular extensions called *Capability Sets*. The first Capability Set (denoted CS-1) was standardised in 1993 and is widely deployed (although often with proprietary extensions). CS-2 has also been standardised and is available from many equipment manufacturers. At the time of writing CS-3 really exists only as a standard and work on CS-4 has recently commenced. Each Capability Set extends the IN model and INAP protocol to deal with more sophisticated services while remaining within the original IN architectural framework.



SMF:	Service Management Function	**SCF**:	Service Control Function
SMAF:	Service Management Agent Function	**SSF**:	Service Switching Function
SCEF:	Service Creation Environment Function	**SDF**:	Service Data Function
SRF:	Specialised Resource Function	**CCF**:	Call Control Function
———	Control Relationship	**CPE**:	Customer Premises Equipment
------------	Management Relationship		
▬▬▬	Call/Bearer Relationship		

Figure 2.5: Basic functional model of an Intelligent Network.

Apart from work on the specification of IN Capability Sets a number of groups are developing technologies that aim at evolving and enhancing the capabilities of IN. For example the IETF SPIRITS/PINT work [IETF, 2001] address how Internet applications can request and enrich telecommunications services. The Parlay consortium is specifying an object-oriented service control *Application Programming Interface (API)* that facilitates the access, control and configuration of IN services by enterprise information technology systems [Parlay, 1999]. The Object Management Group's IN/CORBA interworking specification [OMG, 1999] enables CORBA-based systems interwork with existing IN infrastructure, thereby promoting the adoption of CORBA for the realisation of IN functional entities. Further information on these initiatives can be found in [Brennan *et al.*, 2000].

2.3.1 IN Load Controls

SCPs maintain and execute IN service logic, therefore during periods of high traffic loading they receive a large volume of service requests from the SSPs in the network, making them very susceptible to overload. For this reason most of the large body of publicly documented research on IN load control has focussed on the strategies for protecting SCPs against overload. Broadly speaking, two approaches are taken: in the first SSPs detect SCP overload based on their own measurement and take the necessary prophylactic steps, whilst in the second the SCP detects overload itself and notifies the SSPs of its overload status. These two approaches have been classified by [Lodge, 2000] as *active* and *reactive* respectively, on the basis that the former require the control to be in place continually, while the latter only requires control invocation for the duration of the overload.

We start this section by giving a brief introduction to the most common active and reactive IN load control strategies investigated in the literature. We then provide more detailed descriptions of two reactive strategies: ACG, a widely deployed control supported in IN standards and an adaptive strategy based on mathematical optimisation; the operation of these two controls is investigated in chapter 5.

2.3.1.1 Active IN Load Control Strategies

Active strategies reside in SSPs and use only locally available measurements as the basis for deciding whether an SCP is overloaded. Typically these measurements are the response delays for messages the SSP sends to the SCP during an IN service session. If these delays grow too large it is reasonable for the SSP to assume that the SCP is overloaded (though of course the overload may be in the signalling network). The SSP can then throttle new IN sessions until it detects overload abatement (again through monitoring of response delays). The most widely studied active strategy is *Window*, though others have also been described, for example [Nyberg and Olin, 1994] outline an active strategy employing a *Percentage Thinning (PT)* throttle (a specified percentage of requests that arrive in a given time period are throttled).

The Window strategy involves the use of two parameters in the SSP to determine whether or not a service request can be accepted and forwarded to the SCP. The *window size* (W) indicates the maximum number of requests for which responses may be outstanding at a given time, and a *counter* (C) indicates the actual number of these outstanding requests. Note that the strategy measures the response delays for only the first pair of messages in a service session, applying the standard load control guideline that sessions should not (unless absolutely necessary) be throttled if the network has spent significant processing capacity on them. It operates as follows:

- If $C = W$, the transmission window is 'closed' and the next service is throttled, otherwise the SSP accepts the request and increments C;
- Timers control the value of W: every time a message is sent to the SCP, a timer is started at the SSP, then if a timeout on response occurs, the SSP interprets this as indicative of SCP congestion and W is decremented;
- When a positive response is received (before the associated timer expires), a counter $C0$ is incremented and the size W is increased by one when $C0$ reaches a threshold value $C_{\max}$. The minimum value of W is 1 and there is a maximum value $W_{\max}$;[10]

At the SCP side any new arrival is discarded in cases where the queue of incoming messages is full. The operation of the mechanism is illustrated in Figure 2.6.

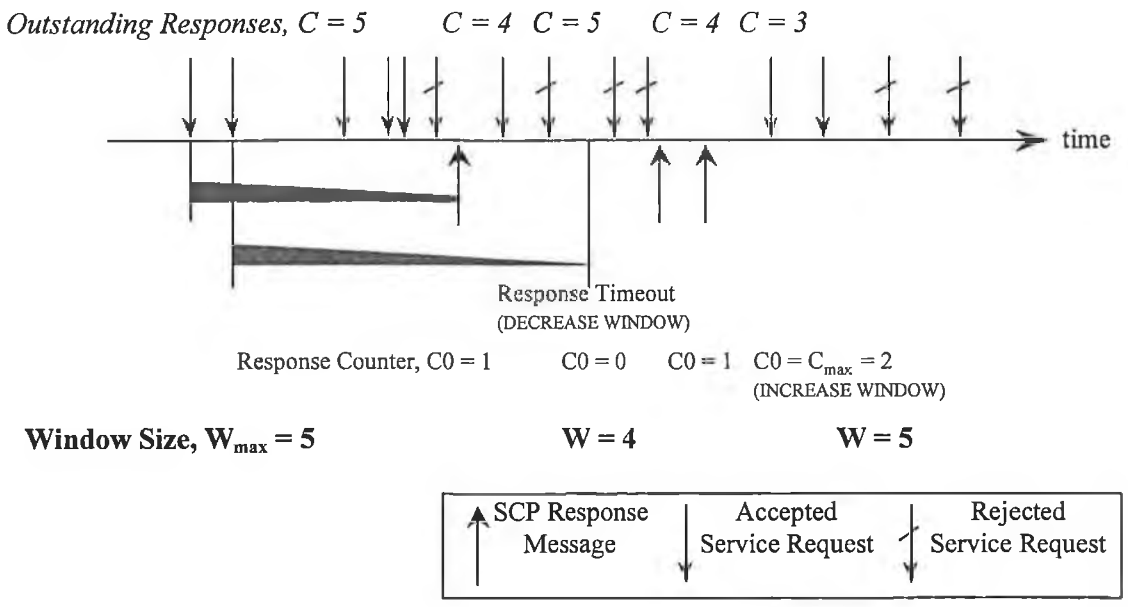


Figure 2.6: Illustration of the operation of the Window IN load control strategy.

[10] Clearly the values of $W_{\max}$ in the various SSPs collectively determine the maximum number of sessions that the SCP will process at any one time. Therefore careful setting of the $W_{\max}$ values is necessary to ensure that the SCP does not overload.

2.3.1.2 Reactive IN Load Control Strategies

Reactive IN load control strategies require interaction between an *overload detection* algorithm residing at the SCP and a *load throttling* algorithm residing at the SSPs. The detection algorithm can run continually, indicating overload when a pre-specified threshold for some measurement is passed, or at set intervals, comparing the mean value of some measurement over the last interval against a threshold. In either case, once overload is detected an overload indication is sent to all, or a subset of, the SSPs. Often indications are specific to a subset of the supported services, or even to specific sub-populations of dialled digits. Upon reception of an overload indication the SSP commences load throttling, the degree of which is usually also dictated by the indication. Throttling continues until the SCP detects overload abatement and indicates it to the SSPs.

SCP Overload Detection Algorithms

A range of SCP overload detection algorithms, making use of different performance metrics, have been proposed and investigated. These include:

- *Queue Length Control (QLC):* the length of the SCP central processor input queue is examined every time a new message is added to the queue; if a pre-specified threshold length is exceeded overload is deemed to have occurred. Overload indications are immediately sent to the SSPs. Similarly, when a pre-specified abatement threshold is passed from above overload is deemed to have abated. This control has the advantage of reacting very quickly to overload onset but can lead to unnecessary over-control caused by random fluctuations in arrival rates;

- *Incoming Message/Session Rate:* The number of messages or new sessions (initial messages of service sessions) arriving at the SCP over a set interval is compared to a threshold. The intention is that these counts will be directly proportional to the session's processing requirements, however this will clearly not be the case if the SCP supports heterogeneous services, each with different processing requirements at the SCP;

- *Average Response Delay (or Response Time Control (RTC)):* The average time spent by messages in the SCP, from placement in the input queue to the end of processing, is measured over a set interval and compared to a threshold. We note that it may be difficult, especially in the presence of heterogeneous service types, to specify thresholds between acceptable and unacceptable levels of delay;

- *Dropped Messages:* SCPs typically drop messages if they have been in the input queue for a time longer then a pre-specified threshold. If messages are delayed for an unacceptable time then there is a high probability that the end-user has abandoned the session and therefore processing capacity would be wasted unnecessarily. In overload conditions, input buffers will fill, resulting in large delays, therefore the presence of dropped messages is a clear indication of an overload. As noted by [Northcote and Smith, 1998] this approach has the advantage that avoiding dropped messages is a well defined performance objective, as opposed, for example, to keeping response delays within acceptable limits;

- *Processor Utilisation (or Load Measure Control (LMC)):* The proportion of time the SCP central processor spends on processing service-related messages is measured or estimated over a set interval and compared to threshold values. Since it does not depend on any assumptions regarding the homogeneity of message processing requirements this measurement, if directly available, gives an accurate indication of the loading experienced by the SCP. When employed, the aim is typically to keep utilisation in a pre-defined range, or close to but below some *target capacity*.

It is also possible to employ combinations of the above algorithms. For example the incoming session rate measurement count could be used to initially detect overload, triggering the lowering the threshold of acceptable queue length, which when surpassed causes the queue length control algorithm to send overload indications to the SSPs [Wallström and Nyberg, 1991]. The detection algorithm can indicate either an *overload level*, or an absolute value of some throttle parameter to the SSPs. In the former case a new level can be calculated every control interval, or the level can be changed in single steps every control interval.

SSP Load Throttling Algorithms

A range of load throttling algorithms that reduce the acceptance rate of IN service requests in response to SCP overload indications have been investigated. These include:

- *Call Gapping (CG):* Limits the number of requests accepted in a certain interval to the specified number. The throttle operates by enforcing a minimum time spacing between call acceptances, by means of a gap timer that is set upon request acceptance. No additional requests can be accepted while the gap timer is active (*cf.* Figure 2.7 page 25 for an illustration of call gapping operation). In the basic version of the control the gap timer duration is indexed, using the indicated overload level, from a table of pre-specified values. However, more adaptive versions, in which gap duration is calculated to accurately match expected arrival rates have been specified [Pham and Betts, 1992; 1994; Smith, 1995]. Modified call gapping throttles have also been proposed. For example [Turner and Key, 1991; Hac and Gao, 1998] describe a version of the algorithm in which fixed gap interval 'slots' are used, but in a manner having no association with request acceptances. A counter is incremented every time there are no arrivals during a slot; arriving requests are then accepted if the counter is non-zero and the counter is decremented;
- *The Leaky Bucket (LB):* The Leaky Bucket employs two parameters: a burst parameter, indicating the maximum number of requests that can be accepted in an interval and a

parameter for the maximum number of requests that can be accepted in multiple consecutive intervals. Similarly to CG, it imposes an absolute limit on the number of requests that can be specified, but it differs in that it allows, to a degree, acceptance of bursts of requests;

- *Percentage Thinning (PT):* With percentage thinning a specified proportion of requests arriving during a time interval are accepted. The decision as to whether or not a particular request is accepted can be based on Bernoulli trials (*cf.* §3.1.1.2), where the probability of success is the *PT coefficient* indicated by the SCP. Note that this throttle does not impose an absolute limit on the number of acceptances, so it will accept too many requests if arrival rates increase rapidly;
- *Token Throttle:* Token based systems generally employ a token bank that initially contains a fixed number of tokens, matched to system capacity (see for example [Seraj, 1985]). When a request is accepted it removes a token from the bank, then when the session completes the token is returned. In order to limit the number of accepted requests the number of tokens in the bank can be modified in response to overload indications [Berger, 1991a];

As with overload detection mechanisms it is also possible to combine the throttles described above; for example, gap timers could be used to space out acceptances in a token-based system.

2.3.1.3 Automatic Code Gapping

The IN standards used in the USA, [Bellcore, 1992a; 1993; 1994; 1995], provide a relatively detailed specification of a call gapping based load control entitled *Automatic Code Gapping (ACG)*. The ACG specification mandates to a large degree the low-level control behaviour of SSPs and SCPs and therefore gives a good indication of how call gapping is actually realised in IN systems. The detailed operation of ACG, as mandated by Bellcore, will now be described in terms of the actions that take place at the SCP and the SSPs respectively.

SCP Actions

At set *measurement intervals* the SCP assigns itself an *overload level* on the basis of one or more performance measurements made available by the SCP management system. A number of measurements may be made and these can be used either separately (with the highest overload level indicated by the individual measurements being used), or in some arbitrary combined manner, to assign the SCP's overload level. Four main measurement types are described, though none are mandatory and other, arbitrary measurements may be used. The four measurements are *Dropped Message Count*, *Average Response Delay*, *Incoming Message Rate* and *Processor Utilisation*. All four measurements are mean values of some quantity over the previous measurement interval.

Rules for translating overload measurements into the SCP overload level are not mandated in the Bellcore specifications. An example of a rule for use with the dropped message count measurement is provided in [Smith, 1995]: if during the just passed interval *any* message has been dropped (due to being delayed by more than t seconds) then the SCP overload level increases by one, otherwise it drops to zero. Since the overload measurements affect only the increments in the overload level this is in effect a search algorithm in which it typically takes a number of intervals before the most appropriate overload level is ascertained. [Northcote and Smith, 1998] note that assigning an overload level is often complicated by the fact that many SCP implementations incorporate two or more distinct subsystems, all of which must be protected from overload.

The specifications mandate the use of 15 overload levels. Once assigned, the overload level is mapped to *gap interval* and *gap duration* values, which are used by the SSP throttling procedure. Both these values are taken from tables of standard values prescribed in the specifications, however no mapping rules are mandated. It seems natural to use a simple mapping rule, whereby the overload level is used to directly index the table values.

The SCP can inform SSPs of its overload level in one of two ways: it can fill a TCAP component of the next IN response messages it sends to an SSP in the coming interval, or it can create and send *autonomous* ACG messages. ACG requests apply only to a sub-population of the population of calls originating at the SSP. The sub-population is selected by indicating the first four or six digits of the ten digit GTAs used by SSPs to address IN service requests (*cf.* §2.2.1.3). The SCP therefore controls traffic to the granularity of all IN requests having partially matching GTAs; such groupings are termed *sources*.

If ACG requests are incorporated in response messages then these requests will refer to the source associated with that message. When using autonomous ACG messages it is not so clear how sources are chosen; one obvious approach would be to send ACG requests for all sources from which messages were received during the previous interval. It is noted in [Smith, 1995] that use of autonomous ACG messages appears more attractive then use of response message ACG requests since, in the latter case, all incoming messages may be dropped with the result that no response messages are sent to the sources.

SSP Actions

An ACG request sent to an SSP contains three parameters: the source to gap (indicated by the first four/six GTA digits), a gap interval g_i and a duration interval τ. Upon reception of an ACG with a non-zero value of g_i the SSP activates the standard call gapping throttling procedure: only one service request per g_i seconds is accepted over the next τ seconds. The ACG throttling procedure is illustrated in Figure 2.7.

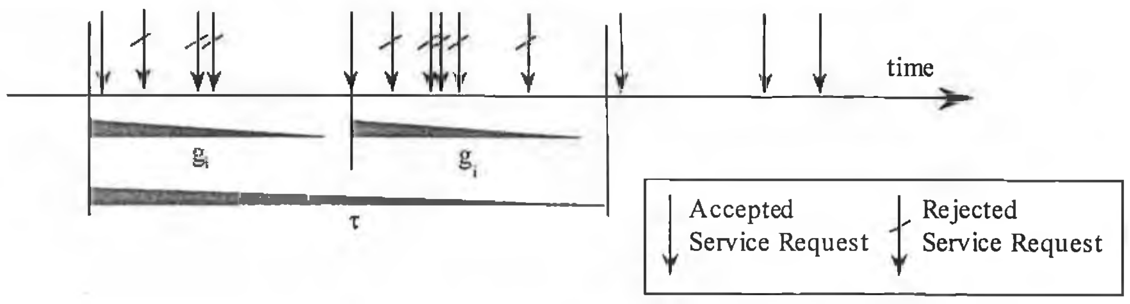


Figure 2.7: Operation of Automatic Code Gapping at the SSP.

The purpose of the gap duration parameter is to place a limit on the activation period of the control in the case that the SSP does not receive further updates of the ACG parameters in response messages (assuming ACG requests are carried in this manner). When autonomous ACG messages are used the SSP is, ignoring message losses, guaranteed to receive updates every SCP measurement interval, so the use of a gap duration appears gratuitous.

When the SSP receives an ACG message it will randomise the gap interval and duration (if used) uniformly between 90% and 110% of the indicated value. This is an effort to avoid synchronisation of the throttling behaviour of the SSPs in the network. When the SSP receives new gapping parameters it can optionally immediately reset active gap interval and gap duration (if used) timers with the new. This process is called *control refreshing* and has the effect of increasing the length of time gap timers are active in a random fashion.

2.3.1.4 Optimisation Strategy

In this section we describe the adaptive IN load control strategy proposed in [Lodge *et al.*, 1999; Lodge, 2000]. The strategy involves the formulation of mathematical optimisation problems (*cf.* §3.1.3 for an overview of this area), whose solution defines the best possible coefficient values to be used by a percentage thinning load throttle residing at SSPs. The optimisation involves the maximisation of generated revenue subject to load constraints on SCPs *and* SSPs, as well as constraints to ensure that pre-defined weightings (similar to priorities) between service types are reflected in the PT coefficients.

The strategy, as specified in [Lodge *et al.*, 1999; Lodge, 2000], consists of two independent optimisation processes, one operating at the SCP and the other at SSPs. At set intervals the SCP optimisation process is invoked and takes as input estimations of IN service request arrival rates at SSPs for the coming interval, SCP target utilisation, service type weights and other service-related information. It outputs PT coefficients for each IN service type / SSP pairing in the network. These values are transferred to the SSPs, which use them as inputs into their optimisation processes. SSP optimisation may serve to modify the PT coefficients in light of local loading constraints and estimated arrival rates and processing requirements of non-IN service types.

Key to the operation of the optimisation strategy is the concept of service type *weights*, which are representative of the relative importance of successfully setting up a session of a given service type in comparison to a session of other service types. Weights can be thought of as a generalisation of priorities. In the strategy specification weights are calculated using information regarding service session set-up revenue, processing requirements and SLAs agreed between the operator and customers. Specifically the weight of service type j at resource x (SSP or SCP), denoted $\omega_{x,j}$, is given by:

$$\omega_{x,j} = \frac{R_j q_j e_{x,j} \mu_{x,j}}{\sum_{j=1}^{J} R_j q_j e_{x,j} \mu_{x,j}}$$

where R_j is the set-up revenue associated with service type j, q_j is the numerical Quality-of-Service level of service type j, $e_{x,j}$ is the number of messages in a type j service session that are processed by resource x, $\mu_{x,j}$ is the mean service rate of service type j messages at resource x and J is the number of services supported by resource x. Quality-of-Service levels are arbitrarily set by the network operator, on the basis of factors such as acceptable delays, customer importance or financial penalties associated with non-adherence to SLAs.

We now describe in detail the optimisation process at the SCP. The first step is to calculate estimates of arrival rates of IN service requests at SSPs for the coming interval. This calculation is based on its knowledge of previously set PT coefficients and measurements of the arrival rate of initial service session messages form each SSP in the past interval. The latter are assumed to be a sufficiently accurate estimate of the arrival rates for the coming interval. Thus we have:

$$\lambda^{*}_{k,j}(t) \approx \frac{\lambda^{initial}_{SCP,k,j}(t)}{p^{a}_{SCP,k,j}(t-T)}$$

where $\lambda^{*}_{k,j}(t)$ denotes the estimate at time t of type j requests that will arrive at SSP k in the coming interval, $\lambda^{initial}_{SCP,k,j}(t)$ denotes the arrival rate of initial messages of type j from SSP k and $p^{a}_{SCP,k,j}(t-T)$ denotes the values of PT coefficients set by the SCP for service j for SSP k at the start of the current interval, which is of duration T time units.

We are now in a position to state the SCP optimisation problem:

$$\underset{p^{a}_{SCP,1,1},\ldots,p^{a}_{SCP,K,J}}{Maximise} \sum_{k=1}^{K} \sum_{j=1}^{J} R_j p^{a}_{SCP,k,j}(t) \lambda^{*}_{k,j}(t)$$

subject to the constraints:

I. SCP Load: $$\sum_{k=1}^{K}\sum_{j=1}^{J}\frac{p^{a}_{SCP,k,j}(t)e_{SCP,j}\lambda^{*}_{k,j}(t)}{\mu_{SCP,j}} \leq \rho^{\max}_{SCP}$$

II. Bounds on $p^{a}_{SCP,k,j}(t)$: $0 \leq p^{a}_{SCP,k,j}(t) \leq 1 \quad \forall \quad k \in \{1,...,K\},\ j \in \{1,...,J\}$

III. Weighting: $$1 \leq \frac{p^{a}_{SCP,k,j'}(t)}{p^{a}_{SCP,k,j}(t)} \leq \frac{\omega_{SCP,j'}}{\omega_{SCP,j}} \quad \forall \quad k \in \{1,...,K\},\ j,j' \in \{1,...,J\}\ with\ j \neq j'$$ $$where\ \omega_{SCP,j'} = \max(\omega_{SCP,1},...,\omega_{SCP,J})$$

where $\rho^{\max}_{SCP}$ is the target SCP load and $\omega_{SCP,j'}$ is the highest of the service type weights – which relates to service type j'. Note that the objective function and constraints are linear functions in $p^{a}_{SCP,k,j}(t)$, hence the optimisation functions falls into the category of Linear Programming Problems and can be readily solved.

PT coefficients $p^{a}_{SCP,k,j}(t)$, output by the SCP optimisation process, are transmitted to the relevant SSP and are used as inputs to the SSP optimisation process, if present. If the SSP does not run its own optimisation process then the SCP PT coefficients are used directly for load throttling over the coming interval. Where SSP optimisation takes place, the SCP PT coefficients are used as upper bounds on the values that can be taken by the SSP PT coefficients for IN service types. SSP optimisation takes into account expected arrival rates of both non-IN and IN service types and the processing demands they collectively place on the SSP. Analogously to SCP optimisation, the problem is formulated to ensure that revenue is maximised, whilst satisfying SSP load constraints, service type weights and bounds on PT coefficients (here the upper bound for IN service PT coefficients are the SCP-indicated coefficients rather than one). Full details of this process can be found in [Lodge *et al.*, 1999; Lodge, 2000].

2.4 LITERATURE REVIEW

In this section we provide an review of the publicly documented research studies on SS.7/IN load control. We focus on both the models employed in the studies and the insights gained into the operation and performance of the various load controls examined. §2.4.1 reviews work on load control at the SS.7 level and §2.4.2 reviews work on IN load control.

2.4.1 SS.7 LOAD CONTROL RESEARCH

In the early 1990s a number of SS.7-related failures in the United States telecommunications infrastructure caused sustained outages of the public telephony network; one of these is described in detail in [Houck *et al.*, 1994]. Widespread negative publicity, resulting from these outages, focussed the attention of network operators and the research community on all issues relating to the robustness and performance of SS.7 networks, in particular on the operation of the SS.7 load controls. A concerted research effort resulted in greater understanding of the how the controls should be configured and identification of their inherent shortcomings. In this section we first review the network modelling approaches adopted and then give an overview of the main results and conclusions of the various studies.

2.4.1.1 Approaches to SS.7 Network Modelling

During the design of an SS.7 network, or subsequently, when new resources or services are introduced, it is necessary to undertake performance modelling and analysis studies to ascertain whether the resulting Quality-of-Service parameters stay within the targets mandated in SS.7 specifications. Consequently, numerous research studies have been concerned with modelling the performance characteristics of individual SS.7 components, end-to-end performance of entire SS.7 networks and analysis of SS.7 protocol functions such as the signalling network MTP controls.

There are two main approaches used for estimation of the performance of communications networks, namely analytic modelling and discrete event simulation (overviews of these approaches are provided in chapter 3). Analytic modelling has the advantage of providing a direct relationship between model parameters and performance metrics and in general requires comparatively small development and computational effort. Analytic methods have been applied to the estimation of the performance of SS.7 links and linksets; see for example [Akinpelu and Skoog, 1985; Skoog, 1988; Wang, 1991].

A detailed and all-encompassing approach to analytic modelling of SS.7 is described in [Willmann, 1988; Willmann and Kühn, 1990; Bafutto *et al.*, 1994]. In this approach the network is decomposed into sub-models that correspond to the separate functional entities described in the SS.7 recommendations. These sub-models are analysed in isolation to obtain message transfer delays, then network-wide delays are calculated by simply adding these individual delays. Another approach to end-to-end SS.7 performance evaluation, based on multiple-chain, product-form queuing networks is described in [Conway, 1990].

Discrete event simulation facilitates modelling of the functional rules of a system with great exactness and allows arrival processes, service time distributions and other model parametersto be chosen freely. For these reasons it is used as the main tool for load control studies [Körner *et al.*, 1994]. This is almost invariably the case in studies of the operation of the SS.7 load controls. One exception is [Zepf and Willmann, 1991], where a hybrid approach, involving both analytic modelling and random number generation, is used to analyse the operation of the TFC procedure.

Simulation studies analysing the SS.7 controls generally model one of the two network topologies illustrated in Figure 2.8. The first model contains multiple source SPs, a single STP and a single destination SP. It is used to analyse overload of the linkset between the STP and the destination SP [Zepf and Willmann, 1991; Smith, 1994a; 1994b; Manfield *et al.*, 1994; Northcote and Rumsewicz, 1995; Rumsewicz and Smith, 1995]. In general this linkset contains a single link, the exception being [Zepf and Willmann 1991]. The model has also been used to analyse overload of the STP [Rumsewicz, 1993; 1994; Northcote and Rumsewicz, 1995]. The second model contains two STPs and multiple source and destination SPs. It is used in [Zepf and Rufa, 1994; Houck *et al.*, 1994; McMillan and Rumsewicz, 1996] to analyse overload of the linkset connecting the two STPs.

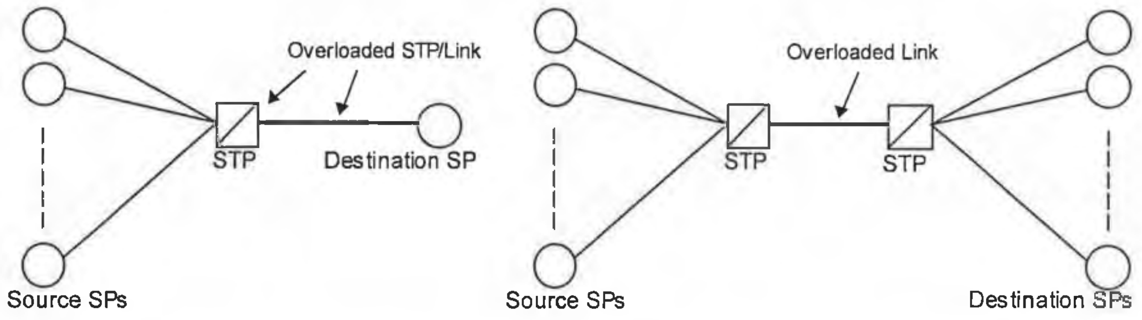


Figure 2.8: Models for simulation-based analysis of SS.7 load controls. Left: Single-STP model. Right: Two-STP model.

These models incorporate all the MTP level 2 and MTP level 3 functionality relating to the operation of the MTP load controls. For MTP level 2 this comprises the acknowledgement procedures, which influence the growth of the link transmission and retransmission queues. At MTP level 3, link queue size monitoring and TFC generation functionality are modelled in detail. Models of user part entities incorporate the throttling of messages in response to CIs received from the local MTP entity, although the manner in which these throttles operate varies from study to study. In studies where STP overload is analysed [Rumsewicz, 1993; 1994; Northcote and Rumsewicz, 1995] the STP is assumed to have a central processor and numerous peripheral processors to handle signalling links; it is assumed that it is the central processor which overloads.

All studies, with the exception of [Manfield *et al.*, 1994], model the message flows associated with ISUP signalling traffic on a session-by-session basis. Delays relating to call answer and conversation phases are generally assigned constant values or are uniformly distributed within a set range; exceptions are [Smith, 1994a; 1994b], where conversation delays are modelled according to the distributions described in [Bolotin, 1994]. In general it is assumed that all accepted session are completed by end-users, unless a signalling message is throttled/discarded. An exception here is [Rumsewicz and Smith, 1995], in which the situation where a media-stimulated mass call-in results in all calls remaining unanswered is examined. A detailed model of the ISUP automatic recovery procedures is included in the models used by [Smith, 1994a; 1994b; Rumsewicz, 1994; Northcote and Rumsewicz, 1995; Rumsewicz and

Smith, 1995]. Signalling traffic relating to both IN and ISUP sessions is modelled in [Zepf and Rufa, 1994; McMillan and Rumsewicz, 1996].

2.4.1.2 Configuration and Operation of SS.7 Load Controls

Research studies on the SS.7 load controls have uncovered a large number of insights into the basic operation of the controls and how best to configure them to give best performance. In our discussion of these studies we classify the various studies into those addressing respectively the setting of queue size thresholds, the impact of network latency on control performance, the impact of user-part recovery procedures, STP processor overload, comparison of control options, the impact of IN services and overload propagation through network management controls.

Setting of overload detection queue size thresholds

A critical component of any load control strategy is the mechanism used to detect overload. With the signalling networks MTP controls, overload is detected by means of overload and abatement thresholds for link queue sizes. The values of these thresholds have a direct impact on the timing and volume of TFC messages sent to traffic sources and hence on the throttling behaviour of these sources. For this reason much of the early work on the analysis of the SS.7 controls concentrated on finding optimal threshold values.

In [Akinpelu and Skoog, 1985] the authors consider the setting of onset and abatement thresholds in the presence of link changeovers[11] occurring when the traffic offered to the network is at normal levels. During the changeover process the input queue of the out-of-service link grows (because messages are not being acknowledged) and the messages in this queue are transferred onto the input queue of an alternative link. The aim of the work is to establish suitable thresholds such that the queue size during the transient period of the changeover is insufficient to trigger detection of overload onset (the queue build-up is transitory, so throttling at traffic sources is not necessary to alleviate it). An iterative algorithm for the calculation of thresholds for the national option controls, based on a mathematical transient analysis of the changeover procedure, is proposed and analysed.

Setting of thresholds to avoid overload detection during changeover can be seen as a minimal requirement, of more interest is how to set the thresholds to maximise the performance of the load controls during sustained link overload. In [Lee and Lim, 1989] the authors address threshold setting during sustained overload by calculating the threshold values so as to ensure that a certain number of signalling messages are transmitted by the link in the time from the crossing of the abatement threshold until the crossing of the onset threshold.

[11] The MTP level 3 protocol incorporates procedures for transferring the messages in the input queues of a link that is being taken out of service (due, for example, to unacceptably high bit error rates) to another link in the same linkset.

[Manfield *et al.*, 1994] also consider threshold setting during sustained overload. The authors agree that the first[12] onset threshold should be set to avoid unnecessary control invocation during link changeover. In the study a simulation model of an overloaded link at an STP, a single destination SP and multiple source SPs is employed. Taking an onset threshold of 1,400 Bytes, the simulation model is used to identify suitable values for other threshold values and to investigate sensitivity of the controls to other parameters, such as the duration of the timers used in the throttling procedures at sources.

Simulation results suggest that it is the difference between the onset and abatement thresholds (which, to a large extent, dictates how long the control remains 'on'), rather then their absolute values, that is critical. Clearly this difference should not be too large, which would result in sources over-controlling traffic, or too small, which would result in under-control and/or oscillatory on/off behaviour. The authors conclude that the abatement threshold should be set roughly 600 bytes less than the onset threshold (regardless of the latter's value), but larger than a minimum value of 600 bytes. The values of the other thresholds (in the NOCP control) is not seen as critical. In addition, the results show that the operation of the control is relatively insensitive to the duration of source timers. This is a somewhat surprising result since the timers control the duration of message throttling, thus if they are active for too long over-control should result.

Whilst the studies discussed above provide valuable insights into the operation of the signalling network MTP controls they have the major drawback that they depend on measures of message throughput to evaluate control performance. Message throughput can be a very misleading measure of network performance as perceived by end-users. For example, it has been shown by [Rumsewicz, 1993] that, in certain circumstances, call completion rates can be very low even when message throughput levels are very high.

Impact of Network Latencies on Control Performance

One of the main design flaws with the signalling network MTP controls is that they do not take into account *network latency*, which is the interval between the time at which the change in link overload status is detected and the time at which traffic sources react by throttling traffic. Database update delays in STPs and SPs account for much of this latency, which can be as much as 1*s* [Smith, 1994a]. During overload TFC messages are continually sent to sources until overload abates, however network latency means that sources cannot respond quickly enough to reduce offered load, so too many TFCs are sent and sources tend to throttle too much traffic.

[12] For the IO this is of course the only onset threshold. The authors analyse both the NOCP and the IO in this study, but conclude that there is little difference in performance between them (although they note that the IO may be superior because it minimises processing associated with messages that are throttled). The authors propose that first onset and abatement threshold values for the NOCP be used for the IO onset/abatement thresholds.

In [Smith, 1994a], the author studies the effect of network latency on the performance of the NOCP control. The scenario addressed involves a focussed overload of ISUP calls towards a single SP, resulting in the overload of a single link. Simulation results show that traffic sources over-control traffic, both spatially and temporally. When the link overloads, network latency causes suppression of traffic from too many (often all) sources. In addition, there is temporal over-control, due both to the delay in recognising that link overload has abated and the operation of the *T15* and *T16* timers used to control MTP message discarding.[13] Because sources are throttled for longer then they need be, the link queue empties before sources re-commence accepting messages; therefore the link's capacity is severely under-utilised. As would be expected, the degree of over-control is found to be greater the larger the network latency.

Smith also shows that network latency leads to synchronisation of sources and oscillation of the length of the overloaded link's queue size. While all sources are still sending traffic via the link its buffer fills; once they stop, the queue empties and the link becomes idle until the sources (in synchronisation) resume accepting messages. This leads to overload onset again and the cycle repeats. This oscillatory behaviour results from the speed at which TFCs are generated upon overload onset, with TFCs being sent towards all sources in a short time interval (were there no networks latencies this would not be a problem as sources would be able to react immediately).

To combat the behaviour described above, Smith investigates modifying the control both to use shorter timer lengths and have larger onset/abatement thresholds (with larger spacing between them). For the former, the sources can react faster to the changes in queue length, whilst for the latter, the rate of change of overload status is slowed down to a level which sources can better track. Both approaches are found to be of benefit and have the major advantage that they do not necessitate changes to SS.7 specifications or deployed systems. These simple changes can result in a big improvement in call completion rates during an overload. We note that although Smith's work focussed solely on the NOCP control, we can expect similar behaviour from the IO control, where traffic throttling is also controlled by timers (*T29* and *T30*) and no account is taken of network latencies.

Impact of User Part Behaviour on Control Performance

Another important issue highlighted in [Smith, 1994a], and also in [Rumsewicz, 1993; Rumsewicz, 1994b; Rumsewicz and Smith, 1995], is the impact of user part recovery procedures, in particular the reattempt behaviour of ISUP REL messages. ISUP sessions involve the reservation of a voice circuit in the trunk network, hence it is important that signalling messages associated with the session reach their destination so that the circuit can be released

[13] These timers control the throttling of messages in the MTP for the national option with congestion priorities. They play a role similar to the ISUP T29 and T30 timers described in §2.2.1.2: TFCs arriving while T15 is active are ignored and expiry of T16 is used to trigger a test of whether the overload has alleviated.

when appropriate. To this end the ISUP protocol includes procedures for responding to the failure of an ISUP message to reach its destination within an acceptable timeframe. Specifically, if an IAM message is not acknowledged by an ACM or ANM message before the expiry of a timer, a REL message is sent to terminate the session. If this message is not acknowledged by an RLC message, the REL reattempts continue under control of a timer T_{REL} (length 4-6s) for an interval of length 60s. Subsequent to this, reattempts continue at a rate of one per 60s, but *Reset Circuit (RSC)* messages are sent instead of RELs. The same procedure occurs if the original message to fail is itself a REL or RLC message.

It is clear that in overload conditions a large proportion of ISUP messages may be throttled and will trigger one or more REL/RSC reattempts, which inflate the load offered to the network, even when call origination rates return to their normal, non-overload levels. This exacerbates the overload, increasing the volume or REL/RSC reattempts. We therefore observe an 'avalanche' of REL/RSCs which takes up a sizeable portion of available network capacity. Worse still, REL/RSCs will generally be assigned a higher priority than IAMs, therefore fewer new sessions are accepted and call completion rates decrease rapidly. The number of new IAMs is also limited by the availability of voice circuits: if very few RLC messages reach their destination then voice circuits are held and cannot be assigned to new sessions.

In a subsequent study [Smith, 1994b], the author attempts to find a solution to the REL avalanche problem. He shows that two obvious approaches, namely limiting the number of REL/RSC reattempts and lengthening the timers controlling these reattempts, do not significantly improve call completion rates. Instead, a dynamic algorithm that limits the rate at which REL/RSC messages reattempt to the rate at which the network bottleneck (the overloaded link, or indeed overloaded STP) can process them, is proposed. The approach is distributed in the sense that it involves an algorithm running at every ISUP entity, which uses only locally available information including the current rate of arrival of RLC messages and the current backlog of REL messages for which an RLC has not yet arrived. It is shown that the proposed algorithm performs considerably better in terms of call completion rates then the standard reattempt procedure.

The performance degradation due to ISUP REL reattempts discussed above indicates that application level recovery procedures can have a strong impact on SS.7 network performance during overload. Interplay between the SS.7 protocols and the applications utilising their services therefore points towards the desirability of taking load throttling decisions at the application level, rather then at the SS.7 protocol level.

Overload of SS.7 Node Processors

Sustained overload of STP processors (as opposed to signalling links) is comprehensively addressed in [Rumsewicz, 1993; 1994b; Northcote and Rumsewicz, 1995]. In the first of these papers a mathematical analysis (both equilibrium and transient) shows that, in the presence of end-user reattempts, call completion rates can approach zero if the only control used is discarding of messages at the overloaded STP. In the second and third paper an alternative approach that employs the TFC procedure is analysed. This approach is somewhat different from the MTP nodal controls described in §2.2.1.1, in that backpressure is not used to trigger the invocation of the TFC procedure in another node, rather the TFC mechanism is invoked directly at the affected STP. The approach is similar to the NOCP control, in that message throttling is assumed to take place, on a priority basis, within the MTP of the traffic sources. Note that although this control is not part of international SS.7 standards it is specified as an option in Bellcore SS.7 recommendations [Bellcore, 1992b], hence it is likely to be implemented in many networks, particularly in North America.

Simulation-based analysis of the TFC-based control as applied to STP processor overload shows similar control behaviour (over-control and oscillation, degradation in call completion rates due to application recovery procedures *etc.*) as observed by [Smith, 1994a] for link overload. However one important difference is highlighted: for the processor overload case, requests for generation of a TFC must join the same processor job queue used for jobs triggered by arriving signalling messages, thus the queue size is artificially sustained at a larger value, leading to the generation of more TFCs and therefore a larger degree of over-control then might otherwise be expected.

A central conclusion of the analysis is that the rate at which TFC messages are generated plays a very important role in the determining call completion rates; this is somewhat at odds with [Smith, 1994a] where the emphasis is placed on the degree of network latency and of the duration of the *T15*/*T16* timers. Rumsewicz shows that varying the rate at which TFCs are generated can have a beneficial impact on call completion rates. However this rate must be chosen carefully, as it depends heavily on factors such as the number of sources sending traffic to the STP and the STP processing capacity. This implies that adequate call completion rates during overload would only be achieved if the rates are set properly for all STPs in the network, a tedious and inexact process.

Comparison of National and International Options

A detailed comparative analysis of the performance of the IO and NOCP controls, as well as a combined control in which throttling takes place in both the MTP and ISUP, is provided in [Rumsewicz and Smith, 1995]. A mass call-in situation, where there is a large increase in the volume of call requests routed towards a single destination number, is studied. Two separate scenarios are modelled: in the first all calls are answered (there is a large pool of operators answering calls) and in the second no calls are answered (approximating the case where there may be only a single operator answering the calls). The link connecting an STP to the destination SP of the overload calls is assumed to overload.

The authors show that, for a range of overload severity and values of control parameters, the IO provides significantly better call completion rates than either NOCP or the combined strategy. In general the combined strategy also performs better than NOCP. The only measure by which the IO performs worse than the other two control is fairness,[14] it tends to favour SPs that originate larger numbers of call attempts during the overload.

It should be noted that the most important reason for the superiority of the IO in this study is the form of the ISUP load throttle. It operates by throttling a variable percentage of IAMs, where the percentage increases/decreases in a stepwise manner according to the arrivals of CIs from MTP. At no stage are other ISUP messages, relating to sessions in progress throttled. This minimises the volume of reattempting REL messages, which greatly improves control performance.

[Northcote and Rumsewicz, 1995] also compare the IO, NOCP and combined controls, but this time in the context of an STP processor overload. Similar parameters to those described above were used. They find that the difference in the performance of the controls is not as great as in the link overload case, although NOCP performs worse in more severe overloads. The IO control is shown to lead to a degree of oscillation in call completion rates, however this is not a significant drawback, since the mean completion rate is comparable to the best of the NOCP and combined controls.

Impact of Intelligent Networks and New Services

All of the studies discussed above address the operation of the SS.7 load controls in situations where all user traffic is associated with the basic ISUP telephony service. However, the proliferation of IN services and mobile networks has meant that a large proportion of the traffic now carried by SS.7 networks is originated by TC applications entities, in particular INAP and MAP. Signalling traffic relating to IN or mobile service sessions will clearly have different characteristics to ISUP traffic, hence it is important to ascertain what impact this has on load control performance.

TCAP uses the SCCP connectionless service to transfer data across the SS.7 network, with the SCCP GTT capability typically being employed to alleviate the need for application entities to have knowledge of the SS.7 addressing scheme. As discussed in §2.2.1.3, a consequence of this is that it is then impossible for application entities to react to congestion indications arriving from the MTP[15] and any throttling must take place in the SCCP. We note that since SCCP can only throttle traffic on a message basis, it may be inferior to a well-designed ISUP control,

[14] The authors consider a control fair *"if each subscriber in the network has, for a randomly selected call attempt, an equal probability of completing that call."*

[15] By default the SCCP indicates the arrival of congestion indications from the MTP to all its users by means of the *N*-PCSTATE primitive.

which can throttle on a call basis by rejecting IAMs first and therefore avoiding disruption to ongoing end-user sessions.

The impact of TC application entity traffic on the operation of the SS.7 load controls is studied in [Zepf and Willmann, 1991; Zepf and Rufa, 1994; McMillan and Rumsewicz, 1996]. In the latter two studies overload scenarios in which the SS.7 network carried both ISUP and INAP traffic are addressed. The main conclusion of both studies is that the SS.7 load controls (both IO and NOCP) are unfair in the sense that those users generating signalling messages of shorter mean length are forced to throttle proportionally more traffic than users transmitting larger messages.

SS.7 load controls would ideally have been designed so that the rate at which CIs are received by a user part (ISUP or SCCP) reflects the volume of traffic that user part transmits via the overloaded resource (be it signalling link or STP). However, to simplify the control procedures, they were designed so that TFC generation rates (and hence CI rates) are derived not from the actual bit rate of messages arriving from a particular user part entity, but from the total message rate transmitted by *all* user part entities residing at a source SP.[16] In addition, for every TFC arriving at an MTP entity, a CI is sent to each of the local user parts, regardless of which of these originated the message that triggered the sending of this TFC.

INAP messages tend to be four to five times longer than ISUP messages. Given that ISUP and INAP entities at a source will typically not send signalling traffic to the same destinations the difference in their mean message sizes suggests that ISUP user part entities will be forced to throttle proportionally more traffic than INAP entities. Simulation results provided by [Zepf and Rufa, 1994] and [McMillan and Rumsewicz, 1996] confirm this to be the case.[17] [McMillan and Rumsewicz, 1996] compare completion rates for INAP and ISUP sessions and show that, for the situation modelled (in which ISUP and INAP contribute equally to the overload), approximately 80% of ISUP messages are throttled, as compared to only 20% for INAP.

In [Zepf and Willmann, 1991] the authors note that TC application entities often send all messages belonging to a single session along the same route through the SS.7 network.[18] This may lead to asymmetric distribution of load amongst the constituent links of an SS.7 linkset, with some links becoming more prone to overload. However the operation of the SS.7 load controls means that traffic routed towards all the links will be throttled, even if some of the links

[16] This simplification was acceptable at the time the MTP specifications were originally developed, since SS.7 networks at the time predominantly carried signalling messages relating to standard telephony calls, all of which are of similar length.

[17] To provide a fair comparison both ISUP and SCCP are assumed to have essentially the same load throttling procedure: a stepwise throttling of messages, with no message priorities (in ISUP).

[18] This is achieved through the setting of a three-bit field in the SCCP primitive used to transfer data. Messages with same value for this field are transmitted over the same signalling links, thus ensuring that they arrive at their destination in sequence.

are loaded at normal levels. The authors propose a modification to the TFC procedure in which TFCs indicate overload of individual links, however it is difficult to see how this could be realised in practice without major changes to deployed systems.

[Kihl and Rumsewicz, 1995] note that end-user IN service sessions often involve both INAP and ISUP signalling. For example with the Freephone service an INAP query is sent to an SCP, which provides a destination number that is then used to initiate a standard ISUP telephony session. During an overload the session is then subjected to load throttles in both SCCP and ISUP, which raises the question as to whether IAMs relating to Freephone sessions should be given priority in ISUP.

One factor not studied is fairness between TC application entities, whose messages are treated as a single stream by SCCP. If there is a difference of the size of messages produced by two application entities then an SCCP throttle based on message counts will result in unfair treatment of the entities sending smaller messages.

Overload Propagation through Trunk Network Actions

Also of importance when considering SS.7 network overload is the action of various switch network management controls that are designed to find alternative routes for call requests when an intermediate switch, on what would be a request's default route, becomes overloaded or unavailable. These controls typically involve the sending of messages by an overloading switch to its neighbours, which subsequently attempt to re-route requests. An actual overload incident, illustrating the manner in which these controls can actually result in the propagation of SS.7 overload, is reported in [Atai and Northcote, 1996]. The scenario is depicted in Figure 2.9:

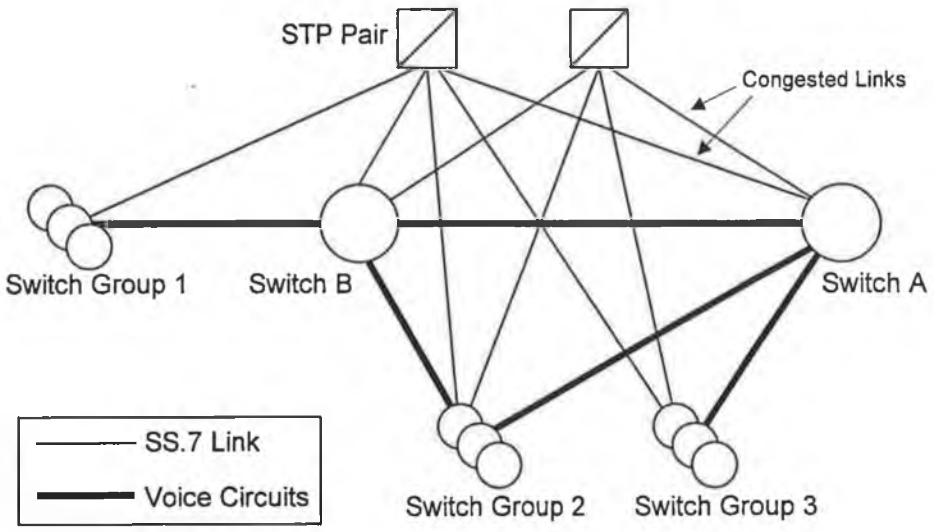


Figure 2.9: Overload propagation through operation of switch network management controls.

In this scenario a focussed overload results in the overload of the signalling links towards SP *A* and the TFC procedure is invoked at the two STPs. TFC messages are sent to switch groups 1, 2 and 3. Switch group 1 does not however throttle all call requests towards the target SP, rather it attempts to re-route these requests via switch B. As a result the signalling links connecting the STPs to switch B overload. Therefore the switch network management controls have not succeeded in their task of finding an non-overloaded route to the target SP, rather they have propagated the overload and caused further degradation of overall call completion rates.

[Houck *et al.*, 1994] describe a similar scenario, in which network management controls lead to overload propagation. However the situation they describe was brought about by a software defect that resulted in many switches in the network becoming periodically unavailable then available. This led to a significant volume of network management messages that caused severe overloading of the signalling network. The authors demonstrated, by means of simulation, that significantly better performance would have been achieved were the network management controls turned off. Of course this course of action may not be appropriate in all situations, so it is critical that potential interactions between network management and SS.7 controls are fully considered during the network design process.

2.4.1.3 Alternatives to Existing SS.7 Controls

We see from the previous section that the majority of the published literature on the subject of SS.7 load control is concerned with analysis of the performance of the control procedures mandated in international SS.7 standards. These controls are widely deployed, so research has naturally focussed on their optimisation, rather then on the proposal of alternative controls. However, some researchers have recently advocated the adoption of controls that prevent, rather than react to the onset of link/node overload and which take a network-oriented rather than a node-oriented approach. Such controls would operate at the application-level and therefore would not require changes to SS.7 equipment or protocol stacks.

In [Arvidsson *et al.*, 1997] the authors outline a network-oriented load control strategy consisting of a load predictor described as a state machine and a decision procedure formulated in Bayesian terms. The approach is based on the recognition that single signals have little or no value on their own since customers neither accept, nor pay, for anything but actually delivered services and that signalling sessions may differ in value or urgency depending on the service they support. They argue that the overall goal should then be to ensure that all accepted service requests can be successfully completed and that important services are given priority during overload.

In the proposed strategy, a maximum signalling session completion time is assigned to each service. The time is set with respect to protocol standards and users' expectations or, for pure load control purposes, simply engineered to match completion times for nearly-overloaded networks. Sessions are assumed to consist of an arbitrary number of signals. Three distinct outcomes of a signalling session, namely *successful*, *unsuccessful*, and *rejected,* are distinguished. A request for a service is successful only if the completion time of the associated session is less than or equal to the maximum completion time, and unsuccessful otherwise. A rejected service is terminated before any signals are dispatched, thus no network resources, except those required to make the rejection decision, are consumed. Each outcome of a session is assigned, on a per-service basis, values representing the associated revenue and cost to the operator.

The strategy is based on the maximisation of *network profit*, defined as revenue minus costs per time unit; this clearly results in the maximisation of the number of successful sessions and in particular the number of successful sessions of those services deemed most valuable by the operator. During overload the strategy rejects less profitable sessions, thereby releasing network capacity for more profitable sessions. It is noted that, even if all sessions are considered equally important, the overall throughput in terms of successful signalling sessions may be increased by a load control that rejects a few requests (the ones which are least likely to become successful) at an early stage. This is also a key facet of the strategy: it predicts the outcome of a session and prematurely terminates those sessions that eventually will become unsuccessful, so the available network resources are spent on useful work.

Measurements of signal round trip times between SP pairs are used to estimate two performance metrics used in the strategy: the 'route' load between the two SPs and the overall load in the network. For the latter an estimate of network load is generated by each originating SP through summing all the individual estimated route loads between that SP and all the destination SPs it communicates with. Information relating to session round trip times is incorporated into a state machine to produce predictions of signal session completion times. The idea is to deliver a prediction before the first signal of the session has left the originating node. There is one such state machine per origination-destination pair, consisting of three states and three transitions (details can be found in [Angelin, 1996]).

Using this estimate, load control is applied by accepting a signalling session if the estimated session time is less than the maximum allowed time. To incorporate profit maximisation into the throttle, Bayesian decision theory is used to combine satisfaction/profits with uncertain predictions in order to determine whether it is profitable to accept or reject a service request (details can be found in [Pettersson, 1996]). The most profitable action is that which maximises the expected value of the gain function, for example it is better to reject a session prematurely if an unsuccessful session costs more than a rejected one and there is an uncertainty in the prediction.

Simulation results presented in [Arvidsson *et al.*, 1997] show that use of the delay-based throttle protects against resource overloads and results in a considerable improvement in throughput over standard SS.7 controls. In addition, incorporation of profit maximisation based acceptance/rejection results in a further improvement in throughput as well as a significant increase in network profit. These results indicate that signalling round trip delays, if used appropriately, provide a valuable metric for applications to gauge the load status of the network and thereby invoke appropriate throttling policies.

One possible criticism of the strategy is that it does not explicitly protect resources against overload: no absolute limits are placed on the number of sessions that are offered to particular resources. In addition, it assumes that service requests are statically routed to specific destinations. Later in this thesis we argue that strategies incorporating dynamic routing of service requests, whilst protecting against overload, can provide more optimal control solutions.

2.4.2 IN Load Control Research

International IN specifications mandate only a minimal degree of load control functionality; they simply support SCPs indication of their current overload status to SSPs. Equipment vendors and network operators are therefore afforded a great deal of freedom when implementing IN load controls. Numerous approaches have been proposed and compared in the literature; in this section we review the main conclusions of these studies. §2.4.2.1 describes the IN models employed, then §2.4.2.2 discusses the results of analyses of various controls.

2.4.2.1 Models and Analysis Methods Employed

The majority of publicly documented IN load control studies focus on control of a single SCP and use a model of the form depicted in Figure 2.10; examples include [Pham and Betts, 1992; 1994; Nyberg and Olin, 1994; Smith, 1995; Kihl and Nyberg, 1997; Hac and Gao, 1998; Patel *et al.*, 2000a; 2000b]. The model contains a single SCP, generally modelled as a single finite queue with a First-In-First-Out (FIFO) queuing discipline and a single server with constant or exponentially distributed service times. The model also includes multiple SSPs, generally modelled as servers with an infinite service rate, so they do not overload. Occasionally the model includes the SS.7 network, but only as a constant delay experienced by messages passed between the SSPs and the SCP (see for example [Kihl and Nyberg, 1997]). First-offered service requests are invariably assumed to arrive at SSPs according to Poisson processes.

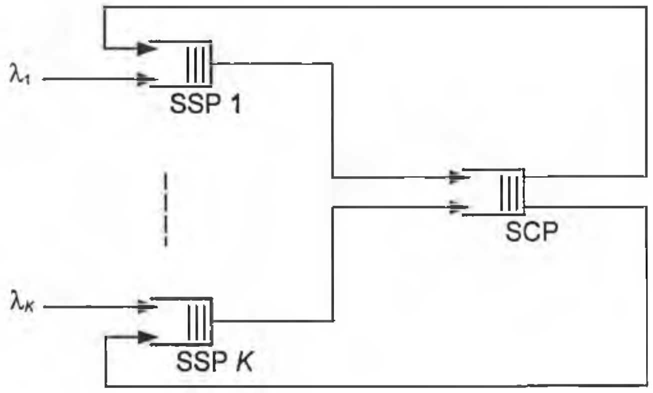


Figure 2.10: Multi-queue IN model.

In many studies, for example [Pham and Betts 1992; 1994; Nyberg and Olin, 1994; Kihl and Nyberg, 1997; Hac and Gao, 1998], it is assumed that the modelled IN service consists of a single query/response pair of messages exchanged between an SSP and the SCP. In these studies the service represented is either a simple service, like Freephone, requiring only a single query/response pair, or alternatively a more complex service for which only the first query/response pair is modelled. The latter approach is commonly justified by the fact that the studied load controls only throttle at the service level (they throttle initial query messages), thus when a request is accepted all further query/response pairs associated with that session are granted access to the required system resources. This assumption is somewhat crude, in that it ignores that at any given time ongoing sessions place 'residual' demands on the SCP and that, during overload, sessions can be prematurely terminated due to the discard of (non-initial) query messages at the SCP queue. To overcome these inaccuracies other studies, for example [Smith, 1995, Northcote and Smith, 1998; Lodge *et al.*, 1999; Patel *et al.*, 2000a; 2000b] do model the complete flows of information associated with sessions of the service(s) modelled.

Many studies model only a single service type; some exceptions are [Northcote and Smith, 1998; Lodge *et al.*, 1999; Patel *et al.*, 2000a; 2000b]. In all studies except [Lodge *et al.*, 1999; Patel *et al.*, 2000a; 2000b], all messages, regardless of service type, are processed at the same rate by the SCP. In general end-user re-attempt behaviour is omitted from the model, exceptions here are [Smith, 1995; Northcote and Smith, 1998; Lodge *et al.*, 1999]. If included, users are assumed to reattempt their service requests following throttling of the original request and, in some studies, following premature session termination. Reattempts are assumed to occur with a set probability, after a delay uniformly distributed within a finite range.

Some studies employ variations of the model described above. For example [Northcote 1996; Northcote and Smith, 1998] model the SCP as containing two sub-systems (servers), one used for Freephone service processing, the other for other 'complex' services. [Bedoy *et al.*, 1998] model the SCP as a multi-processor system (with each processor modelled as a single queue) that also incorporates a local database modelled as a finite-capacity queue. [Kihl and Rumsewicz, 1995] model a combined SSP/SCP, referred to as a *Service Switching Control Point (SSCP)*, as a finite delay queue and a central processor with an associated infinite job queue.

A small number of studies focus on IN resources other than SCPs. [Lodge *et al.*, 1999; Lodge, 2000] address overload of the SSP and employ a relatively complex SSP model incorporating three queues relating to various phases in service processing, as well as an additional queue representing interaction with a local IP. [Kihl and Rumsewicz, 1996] investigate overload of IPs and employ a model consisting of a single IP, which is connected via a finite number of voice circuits to a single SSP; SSPs seek to acquire a voice circuit for the use of a service session at the behest of one of a number of SCPs. There is no queuing at the IP, hence sessions are blocked if the SSPs attempts to acquire a circuit when all circuits are in use.

Due to the relative complexity of IN models described above, the majority of studies use simulation techniques to analyse the behaviour and performance of various IN load controls. Nevertheless, some studies have employed analytic methods; for example [Newcombe *et al.*, 1994; Lodge *et al.*, 1999] solve mean queue sizes and delays using the decomposition approximation method (*cf.* §3.1.1.3) and [Kihl and Rumsewicz, 1996] employ fluid flow approximation to model IP circuit occupancy.

2.4.2.2 Analyses of Deployed/Proposed Controls

In this section we review the results of research studies that analyse the performance of various IN load controls in order to ascertain which perform best over a range of operating conditions. We group these studies into four main categories. First are those comparing various detection and throttling algorithms for reactive SCP control strategies. Second are those comparing the performance of reactive strategies with Window, the most popular active strategy. Third are studies focussing on controls for protecting SSPs, SDPs and IPs from overload. Finally, are some more recent studies, which address network-oriented IN load control.

Overload Detection and Load Throttling Algorithms for Reactive Strategies

As outlined in §2.3.1.1 reactive load control strategies require an overload detection algorithm resident at the SCP and a load throttling algorithm resident at SSPs. Numerous detection and throttling algorithms have been proposed and many studies have compared their performance in order to identify the best combination.

[Lodge *et al.*, 1994; Lodge, 2000] investigate the performance of four overload detection algorithms operating in co-operation with a call gapping throttle. The algorithms are QLC, RTC, LMC and incoming session rate, where the latter is referred to as *Call Count Control (CCC)*. These algorithms are compared under three types of overload, resulting from: steady mean traffic arrival rate, linearly increasing arrival rates and bursty arrival rates. QLC and RTC are seen to react slowly to overload onset and LMC is seen to perform badly in response to bursty traffic. Although not ideal, CCC exhibits the best performance over the three types of traffic and the conclusion is that it should be chosen above the others. We note that the version of LMC examined involves use of an indirect measure of processor utilisation (calculated using message arrival rates and a single mean message processing delay) and that a direct utilisation measurement will clearly result in more accurate overload detection.

Comparisons between CG and PT have been provided by [Berger, 1991a; Kihl and Nyberg, 1997; Lodge, 2000]. In all cases it is assumed that CG uses static gap intervals indexed from a table, although in general these intervals are approximately optimised for the particular network under study. PT is more dynamic, in that the PT coefficients are dynamically computed to provide the necessary reduction in expected traffic during the coming control interval. In general it is found that both throttles are approximately equal in terms of protecting the SCP, albeit under the assumption that CG gap intervals are appropriate to the network. PT is seen to exhibit subscriber fairness, because all SSPs throttle the same proportion of traffic, regardless of their size. For the same reason PT is a scalable throttle: not only is it independent of SSP size, it is also independent of the number of SSPs in the network. On the other hand, [Berger, 1991a; Lodge, 2000] note that a significant advantage of CG is that it places a strict upper limit on the number of accepted sessions and therefore is not susceptible to sudden increases in arrival rates, as is PT. A particular concern with PT in this regard is its vulnerability to reattempts, which lead to increases in offered load, resulting in higher then intended acceptance rates [Smith, 1995].

In light of the above discussion we can conclude that a CG-based throttle that uses dynamically computed gap intervals would combine the fairness and scalability advantages of PT with CG's strict control over acceptance rates. [Smith, 1995] proposes an adaptive version of the standard ACG control that takes this approach. The control identifies the aggregate message arrival rate that can be handled by the SCP, then calculates gap intervals so that all SSPs (regardless of size) reduce their traffic by the same percentage. He finds that the adaptive version provides near optimal throughput performance, regardless of the volume of overload traffic, number of SCPs, or the overload detection algorithm employed (incoming message rate and dropped message measurements are modelled).

[Pham and Betts, 1992; 1994] compare adaptive versions of CG and LB throttles, in which control parameters are dynamically computed by the SCP using its estimates of actual arrival rates at SSPs. In addition, controls are put in place only at those SSPs for which significant increases in arrival rates have been observed. Results show that the two throttles give broadly similar performance, though LB provides marginally higher throughputs. This is due to LB's capability to accept requests in bursts, in comparison to CG, which imposes minimal intervals between acceptances.

Comparison of Active and Reactive Strategies

The relative merits of reactive strategies and the active Window strategy is a somewhat contentious issue, with studies reaching conflicting conclusions. [Pham and Betts, 1992; 1994] compare Window with their adaptive CG and LB throttles and conclude that it provides significantly superior throughput performance for a range of operating conditions. They argue that this superiority is due to continual feedback to SSPs (at every response message arrival) of SCP overload information. [Tsolas, 1992] compares Window with CG, but reaches the opposite conclusion, arguing that CG responds more quickly to overload onset. [Lodge, 2000] compares Window, static and dynamic CG controls and an optimisation-based strategy, concluding that Window can react quicker to overload onset, but considerably overprotects the SCP during relatively mild overloads and under-protects it during relatively severe ones. She shows that the optimisation strategy gives the best overall control performance of the strategies addressed. We address the performance of this strategy in detail in chapter 5.

Window and call gapping are also compared on the basis of ease of configuration. [Tsolas, 1992] argues that if the SCP can indicate an exact gap interval to the SSPs CG's centralised nature makes it easier to re-configure as new services and resources are introduced. In contrast, [Pham and Betts 1992; 1994; Kihl, 1999] argue that placement of the control at the SCP is actually a disadvantage, since it adds significant system complexity. [Lodge, 2000] points out that updating Window is non-trivial, since it involves configuration of timer duration and window sizes (on a per-service basis) at all SSPs. Indeed, she shows that it is very difficult to configure Window in a multi-service environment, making it unsuitable for deployment in current and future IN systems.

Control of SSPs/SDPs/IPs

Load control of generic network databases (not specifically IN SDPs) has been addressed by [Farel and Gawande, 1991]. The authors investigate a network in which a database receives queries from multiple access nodes and propose a control strategy incorporating an internal database overload protection mechanism together with an external CG throttle, residing at the access nodes. The internal mechanism simply discards arriving queries when the input queue size exceeds a specified threshold; the rate of discards is used to periodically specify one of a set of static gap intervals, which is put in place at all access nodes. Clearly this strategy is directly applicable to an IN where multiple SCPs access an SDP.

[Nyberg and Olin, 1995] also address this multiple-SCP, single-SDP scenario, but model multiple services, sessions of which required a different number of queries to the SDP. They propose a Window-based mechanism and investigate its performance in terms of throughput maximisation and fairness (in terms of the amount of SDP capacity allocated to different SCPs and different services). Whilst both of these studies produced encouraging results it is clear that both strategies will suffer from the drawbacks of CG/Window, as described above. They also have the additional disadvantages that placing throttles at the SCP will be a significant drain on its resources and that they may terminate ongoing sessions on which significant network resources have already been expended.

In [Kihl and Rumsewicz, 1996] the authors address load control for an IP, where the limiting resources is the number of available voice circuits rather than processor capacity. The main objective for an IP control is thus to maximise circuit utilisation during overload. The authors argue that requests for circuits should be throttled at the SCP rather then blocked at the IP; this allows differentiation between different classes of request. The following fairness criterion is used as the basis for throttling decisions: only requests relating to an overloading service should be throttled and other services should be protected. They propose an active strategy, in which SCPs apply PT throttles to 'responsible' services on the basis of measurements of the per-service offered load, mean IP circuit holding times and blocking probability at the IP. Simulation and fluid flow analyses show that the strategy performs well in terms of maximising circuit utilisation in a fair manner. We note that strategy again suffers from being resident at the SCP, therefore it uses SCP capacity and terminates ongoing service sessions.

Overload of SSPs is partially addressed in [Kihl and Rumsewicz, 1995], where the performance of IN and SS.7 controls during an SSCP overload is examined. The work concentrates mainly on fairness issues in the presence of independent IN and SS.7 controls and does not address overload of SSPs *per se*. The latter issue is addressed in [Lodge, 2000], where the two-phase optimisation strategy described in §2.3.1.4 is investigated. In the strategy the SCP optimisation process provides PT coefficients for all IN services, which are used as inputs into the SSP optimisation process, along with the estimated arrival rates, processing requirements and relative priority weights of non-IN services. The process outputs PT coefficients, which serve to protect both the SSP and SCP from overload whilst enforcing service priorities such that generated profit is maximised. The strategy is shown to protect the SSP (and the SCP) under a range of overload conditions, to which it dynamically adapts its parameters to achieve profit-optimal control.

Network-oriented Approaches

In the discussion above nearly all IN load control studies have focussed on node-oriented controls that protect a single IN resource from overload; the exception being Lodge's combined SSP/SCP optimisation strategy. The network-oriented control proposed by Angelin, Arvidsson and Pettersson (*cf.* §2.4.1.3) was originally conceived as a generic SS.7 control but has also been investigated in the context of IN [Arvidsson *et al.*, 1997]. This strategy uses delay measurements as its basis, so it detects overload of all IN (and SS.7) resources, providing protection for the whole network. We note that the same argument also applies to Window, hence in this regard it too can be considered as network-oriented. However, as noted in §2.1, an ideal network-oriented will provide scalable, globally-optimal control, which is not possible using Window.

In an IN containing multiple SCPs, the scheme used by SSPs to route service requests plays an important role in determining how susceptible individual SCPs are to overload. Clearly a dynamic routing scheme, integrated with an appropriate load control, provides the best opportunity to optimise SCP resource usage during all traffic conditions. Nevertheless, the majority of IN load control studies have assumed that the network is logically partitioned so that every SSP communicates with only one SCP. One exception is [Milito *et al.*, 91], where the generic problem of dynamically distributing a stream, consisting of several classes of job, to multiple remote nodes with finite input queues, is addressed. The proposed algorithm employs Markov decision theory to distribute jobs in order to minimise the probability that they are blocked upon arrival at the remote node, whilst differentiating between classes of job on a profit basis. The authors suggest that this approach can be applied to IN SSPs, however it is not clear if it would give globally optimal performance in the presence of SCPs of varying processing characteristics.

The use of agent technology for IN load control has been investigated by [Patel *et al.*, 2000a; 2000b; Davidsson *et al.*, 2000].[19] Patel *et al.* describe a multi-agent system realising an artificial computational market in which the processing capacity of SCPs is 'sold' to SSPs in a manner that maximises global utility, which in this case is generated profit.[20] This strategy is very similar to the token-based strategy we describe in this thesis, but suffers from the auction process requiring iterative submission of bids by SSPs and SCPs, which adds significantly to the volume of signalling traffic. Davidsson *et al.* describe an approach based on (mobile) broker agents, which sell SCP processing capacity to SSPs on an autonomous basis, that is, not in the context of an auction.

2.5 SUMMARY AND CONCLUSIONS

This chapter commenced with an introduction to the aims of and standard approaches to load control in telecommunications networks. Following this, a brief description of Signalling System No.7 and Intelligent Networks, with particular emphasis on implemented and proposed load controls, was provided. These descriptions show that load control in SS.7/IN networks is achieved by means of a number of essentially independent controls, operating at different protocol levels. Each of these controls attempt to protect a single resource by detecting overload and employing simple rate-based throttles, but in an manner not co-ordinated with the operation of other controls.

A typical network supporting IN services incorporates standard SS.7 controls for protecting signalling links, controls for protecting STP processors, load throttles in various SS.7 user part entities, trunk network controls that automatically re-route traffic during overload, reactive or active IN-level controls for protection of individual SCPs and possibly other IN resources, as well as internal, vendor-specific protection mechanisms in each network node. Clearly, it is difficult, if not impossible, to ensure that these controls interact in a manner that provides effective and efficient traffic management during overloads.

Considerable research effort has been spent on the analysis of the operation of standard SS.7 and IN load control strategies. Operation of the SS.7 controls is to a large degree mandated by international standards, so research has concentrated on configuring them for optimal behaviour. Unfortunately research studies have identified numerous inherent shortcomings of the SS.7 controls; the principle shortcomings are summarised as follows:

[19] These works, as well as the work described in chapters 5 and 6 of this thesis were undertaken in the context of the ACTS-MARINER collaborative research project [ACTS-MARINER, 2001].

[20] More information on market-based distributed resource allocation is provided in §3.1.4.

- They are dependent on static parameters such as onset/abatement thresholds, TFC generation rates and implementation-dependent throttling policies. These parameters must be carefully selected to match a given network if acceptable control performance is to be achieved;
- They do not take account of network latencies and are therefore susceptible to over-control and oscillatory behaviour;
- They do not explicitly take account of application-level recovery procedures, which can cause severe degradation in network throughput if throttling policies are not carefully designed;
- They are unfair in their treatment of applications producing signalling messages having different mean lengths;
- They operate at lower protocol levels, where it is difficult to make control decisions that take into account factors such as service-level priorities and profit.

Whilst careful configuration can help alleviate many of these problems, it is clear that there would be significant advantages in realising load throttling at the application-level rather then at the SS.7 protocol stack level. In addition, we believe that the best means of detecting overloads in an SS.7 network is not by means of explicit overload notification by TFC messages, but by means of end-to-end delay measurements taken at the application level. The superiority of this approach has been convincingly demonstrated by [Arvidsson *et al.*, 1997].

Research on IN load controls has focussed mainly on strategies for the protection of individual IN nodes, in particular the SCP. The performance of a number of standard active and reactive controls has been compared in a number of studies, sometimes with conflicting conclusions. In relation to reactive controls, a number of SCP overload detection algorithms have been investigated, however in practice it is likely that the optimal approach will vary according to a vendor's specific implementation. Regarding load throttles, it is clear that the two most common algorithms, call gapping and percentage thinning each have disadvantages, relating to fairness and protection from sudden traffic increases respectively. Adaptive versions of reactive strategies, in particular those incorporating profit maximisation have been shown to significantly out-perform standard, static versions. The issue of reactive versus active strategies is more contentious, but most studies favour the former because they are seen to be more responsive and also simpler to configure. We note however that active strategies such as Window are more network-oriented in nature, since they are capable of detecting overload relating to any network resource that processes service sessions.

The research studies discussed in this chapter have produced considerable insights into the operation of the various SS.7 and IN load controls. However, we believe that a major failing of practically all these studies is that they do not employ sufficiently realistic network models.

Studies of the SS.7 controls have mainly analysed control performance in a simple network topology, containing a single overloaded link/STP, a single destination SP and multiple source SPs. Clearly any issues relating, for example, to the overload of multiple links or the placement of SCCP GTT facilities, are not evident from these studies. Similarly, studies of IN controls have completely ignored the presence of an underlying SS.7 network and associated controls, despite the fact that experience in the field shows that SS.7 controls can pre-empt and interfere with IN-level controls [Atai and Northcote, 1996].

In chapter 4 we employ comprehensive SS.7/IN models to analyse in detail the operation of the SS.7/IN controls in a realistic scenario, involving multiple SSPs, a fully connected STP mesh and an SCP, with both IN and ISUP telephony services being fully modelled. A particular emphasis of this study is to ascertain whether the SS.7 and IN controls can interact in a manner that degrades overall network performance.

Besides not taking account of the potential for SS.7 overloads, existing IN controls do not completely fulfil the requirements for a fully-network oriented load control strategy, as outlined in §2.1. Even those strategies incorporating adaptive parameter setting and/or profit maximisation do not provide for co-ordinated control of multiple SCPs, or control over other resources like SDPs or IPs. We contend that the best way to achieve globally-optimal resource usage is to explicitly allocate the capacity of the available resources in a co-ordinated, profit-optimal manner. To achieve full flexibility it is also necessary to integrate the dynamic routing of service requests by SSPs with load control. An SSP could then direct service requests to the SCP that gives that SSP the best performance because, for example, it has been optimised for provision of the service in question, or is in close physical proximity to the SSP. A strategy of this kind would provide a fully network-oriented load control solution with the flexibility to adapt to the demands of a dynamically changing IN environment.

In chapter 5 we take the first step toward the development of a fully network-oriented IN load control by specifying an SCP load control founded on the profit-optimal allocation of processing capacity. The ability of the strategy to satisfy the basic IN load control requirements is ascertained by comparing its performance with two existing strategies. Chapter 6 then specifies a number of enhancements to the basic strategy, providing for control of multiple instances of different IN resource types, minimisation of SS.7 network load and detection of SS.7 overload. Taken together these offer a comprehensive and flexible load control solution to the IN network operator.

CHAPTER 3

METHODS AND TOOLS FOR PERFORMANCE ANALYSIS AND STRATEGY DEVELOPMENT

In this thesis we are primarily concerned with the development and performance analysis of load control strategies for SS.7/IN networks. Our approach is based on three steps: firstly, identification of the shortcomings of existing strategies; secondly, specification of new strategies; and thirdly, analysis of the likely performance of these strategies. Completion of these steps requires the development of detailed models of SS.7/IN networks for use in performance analysis and the application of suitable network control paradigms to develop load controls specifically tailored to the requirements of SS.7/IN. In this chapter we provide background information on the methods, tools and paradigms we employ. The chapter is split into two parts: §3.1 introduces analytic and simulation-based network-modelling, then §3.2 introduces the methods and paradigms applied in the development of the load control strategies we propose.

3.1 NETWORK MODELLING

Modelling is the process of developing a representation of a physical system for the purpose of analysing aspects of the system's behaviour. Telecommunications networks are highly complex systems, hence when developing a model it is necessary to consider only those aspects of the network of interest. The model is thus a simplification of the real system, however it should be sufficiently detailed to permit valid conclusions to be drawn about network operation.

Telecommunications network models are mathematical in nature; they use symbolic notation and mathematical equations to represent the system. Models are classified as being *analytic* or *simulation-based*, the distinction relating to the type of method used to acquire the results of interest from the model. When using analytic models, the deductive reasoning of mathematics is employed to explicitly 'solve' the model to provide the necessary data. Although analytic models provide exact solutions it is often impossible or impractical to develop a solvable model that encapsulates all of the system characteristics of interest. Simulation, on the other hand, allows development of more detailed models, which can be 'solved' using computational procedures. Simulations produce 'solutions' specific to particular inputs: models are 'run' in order to provide sample outputs from which the true system performance measures can be inferred.

3.1.1 Analytic Network Modelling

Analytic network modelling is based on the application of queuing theory, a branch of mathematics which applies the theory of stochastic processes to analysis of the behaviour of queuing systems. This section provides a brief introduction to queuing theory, with the aim of providing the information required to understand the analytic models we describe later in the thesis. §3.1.1.1 introduces basic probability theory, §3.1.1.2 presents an overview of stochastic processes and §3.1.1.3 briefly describes the application of stochastic processes to mathematically describe queues and networks of queues.

3.1.1.1 Basic Probability Theory

Probability theory is based on the paradigm of a random experiment, that is, an experiment whose outcome cannot be predicted with certainty before the experiment is run. It is usually assumed that the experiment can be repeated indefinitely under essentially the same conditions. Probability theory is concerned with the long-term behaviour as the experiment is replicated. For example, if an (unbiased) coin is tossed many times, one intuitively expects that the outcome will be heads in approximately half of the cases. Therefore one can state that the *probability* of a heads outcome is ½. We will now introduce some of the basic terminology used in probability theory.

Consider an experiment having n possible outcomes, denoted $o_1,...,o_n$, where the exact outcome of the experiment cannot be predicted in advance. An experiment of this kind is called a *random experiment* and the set of all its outcomes is called the *space of elementary outcomes*, or the *sample space*, denoted $O = \{o_1,...,o_n\}$. An event is the result of a single random experiment and comprises a subset A of the sample space. A *probability measure* of event A, denoted $P[A]$, is a non-negative number indicating the likelihood of the occurrence of that event as the result of a single experiment, or alternatively, the expected frequency of occurrence of the event over multiple experiments. Probabilities are defined so that the sum of the probabilities of all possible outcomes of an experiment sum to one, $P[O]=1$.

It is often the case that the actual outcome of a random experiment does not matter, instead some other amount, for example the size of a resulting loss or gain, relating to that outcome is of interest. This leads to the concept of a *random variable*. The random variable X is a function, defined on the space of elementary events O, which takes a value $X(o)$ for each $o \in O$. A random variable X is random in the sense that its value depends on the outcome of the experiment, which cannot be predicted with certainty in advance. Random variables can be classed as *continuous* or *discrete*, depending on whether its *range* (the set of values it can take on) is discrete or continuous. For any event A, there is a simple random variable I called the *indicator variable* of A, that takes on the value 1 if A occurs and the value 0 if A does not occur.

We are mainly interested in the probability that a random variable takes a certain value x, this is denoted as $P[X = x]$. For discrete random variables this leads to the description of *a probability mass function* (pmf), denoted $p(x)$, as follows:

$$p(x) := P[X = x]$$

Another convenient form for expressing the probabilities associated with a random variable is the *cumulative distribution function* (cdf). The cdf of a random variable X is defined as:

$$F_X(x) := P[X \leq x]$$

and expresses the probability that X takes on a value less than or equal to x.

Where $F_X(x)$ has a continuous derivative everywhere, a related function, the *probability density function* (pdf), can be defined as follows:

$$f_X(x) := \frac{dF_X(x)}{dx}$$

Note that:

$$\int_{-\infty}^{\infty} f_X(x)dx = 1$$

thus the pdf is a function which, when integrated over an interval, gives the probability that the random variable X takes on a value in that interval.

The k th moment of a random variable X, denoted $E[X^k]$, is defined by:

$$E[X^k] := \int_{-\infty}^{\infty} x^k f_X(x)dx$$

The first moment of a random variable X, denoted $E[X]$ or $\overline{X}$, and known as the *expectation*, or *mean*, or *average value* of X, is given by:

$$E[X] := \int_{-\infty}^{\infty} x f_X(x)dx$$

Another important value, the *variance* of a random variable X, denoted $V[X]$ or σ_X^2, is given by:

$$V[X] = \sigma_X^2 = \int_{-\infty}^{\infty} (x - \overline{X}) f_X(x)dx$$

Where the mean and variance of a random variable X are known the *square of the variation coefficients* (svc), denoted Kx, is given by:

$$Kx := \frac{\sigma_X^2}{(\overline{X})^2}$$

If we consider two random variables, X and Y, defined for some sample space, then the extension of the cdf for the two variables is defined as:

$$F_{XY}(x,y) := P[X \leq x, Y \leq y]$$

Associated with this function is a joint pdf, defined as:

$$f_{XY}(x,y) := \frac{d^2 F_{XY}(x,y)}{dx\,dy}$$

Finally, X and Y are said to be *independent* if and only if:

$$f_{XY}(x,y) = f_X(x) f_Y(y)$$

3.1.1.2 Stochastic Processes

A *stochastic process* (or *random process*) is a function $X(t,o)$ (commonly denoted $X(t)$ for simplicity) of both time and probability space. For a fixed value of t it becomes a function of probability space, *ie.* a random variable, whereas for a fixed value of o it is a function of time and is referred to as a *sample function* of the process. Stochastic processes are widely used to model the behaviour of telecommunications systems, for example, the number of IN service requests arriving at an SSP in a time interval can be modelled as a stochastic process.

The cdf of a stochastic process, denoted $F_X(x,t)$ is defined as follows:

$$F_X(x,t) := P[X(t) \leq x]$$

Furthermore, for n allowable values of t, $\{t_1, t_2, ..., t_n\}$, a *joint cdf* may be defined for the process as follows:

$$F_{X_1,X_2,...,X_n}(x_1,x_2,...,x_n;t_1,t_2,...,t_n) := P[X(t_1) \leq x_1, X(t_2) \leq x_2, ..., X(t_n) \leq x_n]$$

The joint cdf is commonly denoted using the vector notation $\boldsymbol{F_X(x;t)}$. In order to completely specify a stochastic process the values of $\boldsymbol{F_X(x;t)}$ must be specified for all possible subsets of $\{x_i\}$, $\{t_i\}$ and all n. However for many of the interesting and useful stochastic processes it is possible to provide this specification in very simple terms. We now list some classifications of stochastic processes based on properties they possess.

Independent Processes

The simplest and most trivial stochastic processes are those for which $\{X_n\}$ forms a set of independent random variables , so that:

$$F_X(x_1, x_2, ..., x_n; t_1, t_2, ..., t_n) = F_X(x_1; t_1) F_X(x_2; t_2) ... F_X(x_n; t_n)$$

Stationary Processes

A stochastic process $X(t)$ is said to be *stationary* if $F_X(x;t)$ is invariant to shifts in time for all values of its arguments:

$$F_X(x; t+\tau) = F_X(x; t)$$

where $t+\tau$ is defined as the vector $(t_1+\tau, t_2+\tau, ..., t_n+\tau)$.

An associated property, that of *wide-sense stationarity*, is held by $X(t)$ if merely its mean and variance are invariant to shifts in time. Clearly all stationary stochastic process are wide-sense stationary, but not conversely.

Markov Processes and Markov Chains

A stochastic process is classified as a Markov process if and only if its next state is dependent only on its current state and not on any previous values. This can be expressed analytically as follows:

$$P[X(t_{n+1}) = x_{n+1} | X(t_1) = x_1; X(t_2) = x_2; ...; X(t_n) = x_n] = P[X(t_{n+1}) = x_{n+1} | X(t_n) = x_n]$$

A Markov process with a discrete state space is referred to as a *Markov chain*. Markov chains can be either discrete-time or continuous-time. For a discrete-time Markov chain the instants at which the state changes are preordained (a state transition takes place at each instant even if the state does not change as a result of the transition). For a continuous-time Markov chain the state transitions can take place at any instant in time. Of particular interest is the random variable describing how long a Markov chain remains in its current state before a transition to another state occurs. For a discrete-time Markov chain this time can be shown to be geometrically distributed, whilst for a continuous-time Markov chain it is exponentially distributed.

A widely-used graphical representation of a Markov chain is that of a *state-transition diagram*. In a state-transition diagram the discrete states of the chain are represented by labelled circles, which are connected by directed arrows representing the probability that when the current state is that at the source of the arrow the state after a state transition will be that at the arrow destination. A simple state transition diagram is illustrated in Figure 3.1.

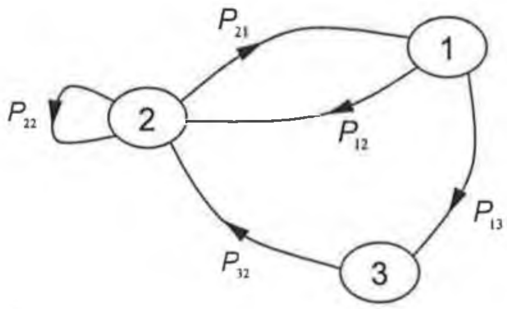


Figure 3.1 Example state-transition diagram representation of a Markov chain.

Birth-Death Processes

A *birth-death process* is a (discrete- or continuous-time) Markov chain in which state transitions only take place between neighbouring states. If, with no loss of generality, the set of integers is chosen as the discrete state space then the birth-death property requires that that if $X_n = i$, then $X_{n+1} = i-1$, i,or $i+1$ and no other value. Birth-death processes play an important role in queuing theory, since they provide a means of modelling a queuing system where the time intervals approach zero (a continuous-time process), so that only a single event, an arrival or a departure, can occur during an interval.

The probability of a birth-death process being in a particular state k at time t is denoted by $P_k(t)$, where:

$$\begin{aligned} &P_k(t+\Delta t) = P_k(t) - (\lambda_k + \mu_k)\Delta t P_k(t) + \lambda_{k-1}\Delta t P_{k-1}(t) + \mu_k \Delta t P_{k+1}(t) + o(t) && k \geq 1 \\ &P_0(t+\Delta t) = P_0(t) - \lambda_0 \Delta t P_0(t) + \mu_1 \Delta t P_1(t) + o(t) && k = 0 \end{aligned}$$

where λ_k is the *birth rate* (or *arrival rate*), representing the rate at which births (arrivals) occur when the population (number in the system) is k; and μ_k is the *death rate* (or *departure rate*), representing the rate at which deaths (departures) occur when the population (number in the system) is k. The above equations can also be written in the form:

$$\begin{aligned} &\frac{dP_k(t)}{dt} = \lambda_{k-1}P_{k-1}(t) - (\lambda_k + \mu_k)P_k(t) + \mu_k P_{k+1}(t), && k \geq 1 \\ &\frac{dP_0(t)}{dt} = -\lambda_0 P_0(t) + \mu_1 P_1(t), && k = 0 \end{aligned}$$

Semi-Markov Processes

We noted above that for discrete- and continuous-time Markov chains the time spent in a state respectively obeys a geometrical or exponential distribution. In a *semi-Markov process* the time the process remains in a state obeys and arbitrary probability distribution. However, at the instants of state transitions, the process behaves just like a Markov chain, at those instants the process is referred to as an *embedded Markov chain*.

Random Walks

A *random walk* is a Markov chain in which the next state the process occupies is equal to its current state plus a random variable, whose value is drawn independently from an arbitrary distribution; this distribution, however, does not change with the state of the process. Expressed analytically, a sequence of random variables $\{X_n\}$ is referred to as a random walk if

$$X_n = \sum_{i=1}^{n} U_i$$

where $U_1, U_2, \ldots$ is a sequence of independent random variables with a common, arbitrary distribution.

Renewal Processes

The *renewal process* is a specific application of the random walk; it counts the transitions made by the process as a function of time – $U(t)$ equals the number of transitions that have taken place by time t (it is also assumed that the process begins in state 0, *ie.* $U(0) = 0$).

The Poisson Process

The *Poisson process* is widely used in queuing theory for the modelling of arrival processes such as the sequence of times at which calls are originated by users of a telephony network. It is a special case of the birth-death process in which the arrival rate is constant and the departure rate is zero in all states ($\lambda_k = \lambda,\ \mu_k = 0\ \ \forall\, k$). Therefore we have:

$$P_k(t) = \frac{(\lambda t)^k}{k!} e^{-\lambda t} \qquad k \geq 0,\ t \geq 0$$

which is known as the *Poisson distribution.*

For a Poisson process, the average number of arrivals in $(0,t)$ is λt and the variance of the number of arrivals in the same time interval is also equal to λt. The interarrival times of a Poisson arrival process are exponentially distributed, *ie.* the pdf of the interarrival times is given by:

$$f(t) = \lambda e^{-\lambda t} \qquad t \geq 0$$

The mean of the exponential interarrival time distribution is $1/\lambda$, while its variance is $1/\lambda^2$. The exponential distribution also exhibits the *memoryless* property, whereby the distribution of the time until a future arrival is independent of the time since the last arrival. Therefore if, at some random time t, an estimate of the time that will elapse until the next arrival is evaluated then the result will be independent of the time that has elapsed since the last arrival.

Bernoulli Trials

Bernoulli trials are a stochastic process widely used to model a sequence of independent generic trials that can result in two outcomes, success or failure, where the probability of success is p and the probability of failure is $(1-p)$. Analytically we can describe the Bernoulli trials process with a sequence of indicator random variables $I_1, I_2, \ldots, I_n$, where the j^{th} indicator variable is used to describe the outcome of trial j. Therefore we have:

$$P[I_j = 1] = p$$
$$P[I_j = 0] = (1-p)$$

3.1.1.3 Queuing Theory

Queuing theory involves the study and analysis of the behaviour of queuing systems, where a queuing system is any system in which arrivals place demands upon a finite-capacity resource [Kleinrock, 1975]. Queuing theory is concerned with estimating values such as the mean queue size, mean waiting time or length of idle period, which are key metrics used for the evaluation of the performance of many systems.

In general the length of a queue depends on the mean arrival rate (of *customers*), the mean rate at which arrival demands are serviced (the *service rate*) and on the statistical fluctuations of these rates. Clearly, when the mean arrival rate exceeds the system capacity the queue will grow in an unbounded manner. However, even where the mean arrival rate is less than system capacity queues will sometimes grow, due to clustered arrivals and/or variations in demands; the effect of these variations will be greater when the arrival rate approaches the maximum capacity of the system. We now introduce some of the basic terminology used in queuing theory.

In order to completely specify a queuing system the stochastic process(es) that describe the arrivals to the system and the structure and discipline of the server(s) must be described. The arrival process to a queue is typically described in terms of the probability distribution of the interarrival times of usage requests, denoted $A(t)$, where:

$$A(t) = P[\text{time between arrivals} \leq t]$$

The mean arrival rate to the queue is denoted λ, giving the mean interarrival time as $1/\lambda$. The svc of $A(t)$ is given by $Ka = \sigma_A^2 \lambda^2$. An arrival stream may be comprised of more than one *class* of arrivals, which may be described by different interarrival distributions.

The server process of a queue is typically described in terms of the probability distribution of the service times of request processed by the queue, denoted $B(x)$, where:

$$B(x) = P[\text{service time} \leq x]$$

The mean service rate of the queue is denoted μ, giving the mean service time $1/_{\mu}$. The svc of $B(x)$ given by $Ks = \sigma_B^2 \mu^2$. It is possible for a queue to contain more than one server and it is possible that distribution of service times will differ for each server.

The load of a queue, denoted ρ, is a measure of the proportion of time the queue server is busy; it is calculated as $\rho = \lambda/_{\mu}$. An important structural description of a queue is that of the *queuing discipline*, which describes the order in which requests are taken from the queue and allowed into service. Some common queuing disciplines are *First-In-First-Out (FIFO)* and *Last-In-First-Out (LIFO)*. Some queuing disciplines distinguish between classes of request arrivals on the basis of *priority*, with higher priority requests being granted preferential access to the server. The extent of storage capacity available in the queue to hold waiting requests may also be limited, the number of requests that can be stored is often denoted by K.

A fundamental result in queuing theory is *Little's Law*, which states that the mean number of requests in a queuing system (denoted $\overline{N}$) is equal to the mean arrival rate of requests to the system (λ), times the mean time spent by requests in the system (denoted T):

$$\overline{N} = \lambda T$$

Queues can be classified according to the widely-used shorthand notation $A/B/n$, where A describes the queue's interarrival time distribution, B describes its service time distribution and n is the number of servers in the queue. Values which A and B can take on include exponential (M), deterministic (D), Erlangian (E) and general (G). We will concentrate here on the $M/M/1$ queue, which is widely used in the modelling of telecommunications networks.

The M/M/1 Queue

The $M/M/1$ queue is a single server queue with a Poisson arrival process and an exponential service time distribution. Therefore both the arrival rate, λ and the service rate, μ are independent of both time and the state of the arrival and service processes.

The mean number of requests in the queue is given by:

$$\overline{N} = \frac{\rho}{1-\rho}$$

and the variance of the mean number of requests is given by:

$$\sigma_N^2 = \frac{\rho}{(1-\rho)^2}$$

The average time spent in the system is given by

$$T = \frac{1/\mu}{1-\rho}$$

The svc of both the interarrivals to the queue and the service times is given by

$Ka = 1,\ Ks = 1$

A significant result relating to the $M/M/1$ queue is *Burke's Theorem*, which shows that, in steady state, the output of a stable $M/M/1$ with interarrival rate λ is a Poisson process with the same rate λ. An important consequence of this is that the formulae listed above can also be applied to a network of $M/M/1$ queues, but only if the network is feedforward (there is no feedback between any of the queues in the system) and all queues have only a single service rate.

Networks of Queues

The study of computer systems and communications networks often requires analysis of queuing systems characterised by complex service disciplines and networks of interconnected queues. Unfortunately the range of queuing networks for which solutions are known in an explicit form is limited to so-called *Jackson* and *BCMP* (Baskett, Chandy, Muntz and Palacios) classes of network. Jackson networks are systems of queues of unbounded length having independent service times, distributed according to exponential laws[21] with no distinction being made between different classes of customer. Open systems, which receive customers from the exterior according to a Poisson process, or closed systems, containing a constant number of customers, are permitted. BCMP are somewhat more general in nature, in that they allow for the existence of different classes of customers and service disciplines other than First-In-First-Out. Therefore, in order to analyse networks involving different classes of customers, with service time dependent on class (as would be the case for a reasonably realistic IN queuing model) it is necessary to employ an approximate method.

A number of approximate methods for queuing network analysis are available. The most widely used among these are the *decomposition method*, which consists of studying each queue of the network along with characterisation of intermediate streams; *mean value analysis*, which involves the study of the first moments of the mean numbers in each queue; the *aggregation method*, which involves study of the system group by group, where groups are characterised by weak interactions with the exterior; and the *isolation method*, which consists of isolating each queue and studying it formally taking into account the effects of interaction with the exterior.

[21] These laws can have parameters that depend on the length of the respective queue.

Other approximation methods involve iterative methods or the lead to numerical solutions and are generally more difficult to employ than those listed above.

For our fully analytic IN model (presented in §5.3.2) we require an analysis method that is applicable to queuing networks exhibiting the following characteristics:

- Openness: in an IN service requests arrive from a source external to the networks and depart to some external sink;
- Poisson arrival process: it is reasonable to approximate the arrivals of IN service requests as a Poisson process;
- Support for multiple classes of customer: INs typically support multiple service classes, with individual service sessions comprising different message types, each placing different processing demands on IN resources;
- Exponentially distributed service times: it is reasonable to assume exponentially distributed service times for messages processed by IN resources (the mean service time for different messages types may of course vary).

The results we require from our analytic model in order to evaluate the performance of various load control strategies are the mean load, queue length and (if possible) delays for each service type. Mean arrival rates and hence mean loads can be calculated directly by inspection of the queuing network, but queue lengths and delays must be provided by the analysis method.

Since we require the analysis method to support multiple classes of customer approaches based on Jackson and BCMP networks are not usable and an appropriate approximation method must be chosen. Of the available approximation methods the simplest one to apply that provides the necessary outputs is the decomposition method. This was therefore chosen for the development of our analytic model.

The Decomposition Method for Queuing Network Approximation

The decomposition method provides an approximation of the mean queue lengths and delays by means of a queue-by-queue decomposition of the network. The method involves the formulation of a set of linear equations that approximately express the svc of the intervals between two departures from station i *(interdeparture times)*, denoted C_i, in terms of the svc of the interdeparture times for all other stations in the network. These set of equations are solved and their solutions allow calculation of the svc of the interarrival times at each station, denoted Ka_i. Knowing λ_i and Ka_i, the mean queue length of station i can be calculated by means of Kingman's formula and mean delays using Little's Law.

Central to the decomposition is the assumption that the departure process from every station in the network is a renewal process (the time interval between successive departures does not depend on the length of previous intervals). Clearly this assumption is exact in the case where all stations have a Poisson arrival process and exponentially distributed service times, which is the case for the analytic IN model we develop in chapter 5. Generally, the method is not applicable where some stations contain more than a single server or when the queuing discipline at any stations is other than FIFO. We will now provide the formulation of the decomposition equations and outline how their solution can be used to calculate mean queue lengths and mean delays. A detailed derivation of the formulation of the equations can be found in [Harrison and Patel, 1993].

Consider a queuing network consisting of J stations. Let the stations in the network each support R classes of customer and let r denote an arbitrary customer class. Let λ_{jr} denote the mean rate of arrival of class r customers at station j. For each customer class there is a stream of arrivals from the exterior, with arrival rate denoted λ_{0r} and svc denoted Ka_{0r} ($j = 0$ is used to denote the exterior). The total arrival rate to the system, denoted λ, is given by:

$$\lambda = \sum_{r=1}^{R} \lambda_{0r}$$

Each station j has a mean service for class r customers μ_{jr} and svc Ks_{jr}. Upon departure from station i a customer of class r will arrive at station j as a customer of class r' with a probability denoted $p_{ir,jr'}$. The relative frequency of the number of visits by a customer of class r to station i is denoted by e_{ir}. The values for e_{ir} are the solutions of the following system of linear equations:

$$e_{ir} = \sum_{r=1}^{R} \left(\sum_{j=1}^{J} e_{jr'} p_{jr',ir} + p_{0r',ir} \right)$$

The load at station j due to customers of class r, denoted ρ_{jr}, is then given by:

$$\rho_{jr} = \frac{\lambda e_{jr}}{\mu_{jr}}$$

The mean total load at station j, denoted ρ_j is given by:

$$\rho_j = \sum_{r=1}^{R} \rho_{jr}$$

Similarly the mean arrival rate at station j , denoted λ_j is given by:

$$\lambda_j = \sum_{r=1}^{R} \lambda_{jr}$$

Using the terminology defined above the decomposition equations for station j, $1 \le j \le J$, can be written as follows:

$$C_j = -1 + \sum_{r=1}^{R} \rho_{jr} \mu_{jr}^{-1} (Ks_{jr} + 1) + (1 - \rho_j)(Ka_j + 1 + 2\rho_j)$$

where:

$$Ka_j = \lambda_j^{-1} \sum_{k=0}^{J} [(C_k - 1) P_{kj} + 1] \lambda_k p_{kj}$$

with:

$$P_{kj} := \sum_{r=1}^{R} \sum_{r'=1}^{R} \frac{\lambda_{kr}}{\lambda_k} p_{kr,jr'}$$

The set of linear equations in C_j, $1 \le j \le J$ can be solved simultaneously and the results used to calculate Ka_j, $1 \le j \le J$. Knowing λ_j and Ka_j, the mean length of the queue at station j, denoted $\overline{L_j}$ can then be calculated using Kingman's formula:

$$\overline{L_j} = \rho_j \left[1 + \frac{\rho_j (Ka_j + Ks_j)}{2(1 - \rho_j)} \right]$$

The mean delay at the jth station can then be obtained by applying Little's Law:

$$T_j = \overline{L_j} / \lambda_j$$

The mean delay for each customer in passing through the network can be obtained by simply summing the mean delays at each queue for every queue visited (or re-visited) by the customer

3.1.2 Simulation-Based Network Modelling

Simulation is the imitation of the operation of a real-world process or system over time [Banks, 1998]. It involves the generation of an artificial history of a system, given specific inputs, and the observation of that artificial history to draw inferences concerning the actual performance characteristics of the system. Because simulation models can be built to closely mirror the behaviour of aspects of a real system without significant simplifying assumptions they are particularly useful for the modelling of highly dynamic systems, whose behaviour cannot be easily modelled analytically.

The most common form of simulation is know as *discrete event simulation*; it relates to the modelling of systems whose state variables change only at those discrete points in time, points at which *events* occur. This is the case with most telecommunications networks, where the behaviour of interest is almost always of a discrete nature (for example the arrival of a data packet at a node, or the end of a telephone call).

Events in a discrete event simulation model occur as a consequence of the completion of system activities, where activities typically involve the processing of an object passing through the system by a system resource. Objects often compete for resources, possibly joining queues to wait for the resource in question to be released by another object. The model is executed over time, by a mechanism that moves simulated time forward. System state is updated at each event, along with the capturing and freeing of resources that may occur at that time. Models are invariably implemented as software programs with inputs provided by random number generators that emulate stochastic behaviour typical of the real world.

The simulation models described in this thesis were implemented using the OPNET™ simulation model development environment [OPNET, 2001]. OPNET is a hierarchical, object-oriented development environment that is designed specifically for modelling and analysis of communication networks. It provides a hierarchical graphical interface for model specification in which network, node and process models are combined to realise a complete network model, as well as a range of tools for the specification of simulation inputs and filtering and analysis of outputs.

OPNET network models define the position and interconnection of communicating entities, or nodes. Each node is described by a block structured data flow diagram, or OPNET node model, which typically depicts the interrelation of processes, protocols and subsystems. Each programmable block in a node model has its functionality defined by a process model, which is defined by means of C programming code encapsulated within a graphically laid out state-transition diagram. Specification of processes in C is facilitated by an extensive library of support functions providing a range of simulation services.

3.2 LOAD CONTROL STRATEGY DEVELOPMENT

In this section we briefly introduce three approaches used in the specification of the load control strategies whose performance we analyse in chapters 5 and 6. §3.2.1 gives an overview of mathematical optimisation and linear programming, whilst §3.2.2 and §3.2.3 introduce *Market-based Control* and *Ant Colony Optimisation*, two paradigms for network control that have been developed by the research community in recent years.

3.2.1 MATHEMATICAL OPTIMISATION

Mathematical optimisation is the process of ascertaining the existence of and evaluating either the maximum or minimum value of a mathematical function and the values of function parameters which result in this optimal value, subject to a number of (equality or inequality) constraints. The function to be optimised is referred to as the *objective function*. The possible values of the objective function, subject to the constraints, form a *feasible region*; the optimisation process returns the point giving a maximum/minimum value of the function.

Optimisation problems are classified as *linear* or *non-linear*, depending on whether or not the objective function and all constraints are linear in nature. The simplest and most widely used class of optimisation problems are known as linear programming problems, in which the objective function and all constraints are linear. Linear programming problems can be expressed analytically in the form:[22]

$$\begin{array}{ll} \text{Minimise} & cx \\ \text{s.t.} & Ax = b \\ & x \geq 0 \end{array}$$

Where x is the vector of variables to be solved for; A is a matrix of known coefficients; and c and b are vectors of known coefficients. Note that usually A has more columns than rows, so $Ax = b$ is therefore quite likely to be under-determined, leaving great latitude in the choice of x with which to minimise cx.

The most commonly employed method for solving linear programming problems is the*simplex method*, which involves visiting a progressively improving series of trial solutions, until a solution is reached that satisfies the conditions for an optimum. The simplex method exists in two forms, the single-phase form which requires that all constraints are upper-bounded and the two-phase form which can handle constraints for which lower-bounds are defined.

In chapter 5 we compare the performance of an IN load control strategy we develop with that of a similar strategy in which load control parameters are calculated as the solution of a linear programming problem. The linear programming problem is formulated in a manner such that the objective function and the constraints are linear and all parameters are upper-bounded, hence the single-phase simplex method can be used to provide a solution. The LP_SOLVE library [Berkelaar and Dirks, 1995] written using the C programming language was used to implement the single-phase simplex method in our performance analysis studies.

3.2.2 MARKET-BASED CONTROL

Market-based control is a paradigm for the design and implementation of distributed resource allocation and optimisation problems involving the maximisation/minimisation of some global measure. With this paradigm system entities emulate a real market by trading commodities with each other. The producers and consumers (referred to as *agents*) of system resources are

[22] This is the *standard form* representation of a linear programming problem. General linear programming problems, containing inequality constraints and upper/lower bounds on x can always be expressed in this form.

therefore modelled as the self interested decision makers described in standard microeconomic theory. Resource allocation decisions are governed by a market price system in which agents interact by offering to buy or sell commodities at given prices. Market-based control systems typically are centred around algorithms for finding the *general equilibrium* of the market, that is, a set of prices such that supply meets demand.

In order to model the bidding behaviour of agents *utility functions*, which encapsulate the preferences of a consumer are employed. A high value of the utility function for some consumption bundle means that such a bundle is preferred over a bundle with a lower utility function value. An example utility function for a single commodity is illustrated inFigure 3.2. Apart from the utility function each agent has associated with it an *endowment* (initial allocation) of each commodity.

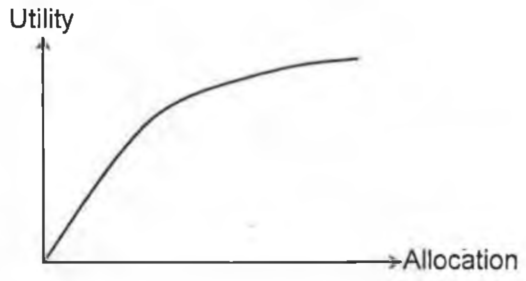


Figure 3.2: Example single commodity utility function.

If a number of agents can trade commodities in such a manner that all agents have higher utility after the trade, then the agents are motivated to trade commodities with each other. Clearly, if trading is performed in the context of a price-based market (every commodity is evaluated in terms of another commodity or using a monetary unit) each agent will face the optimisation problem of how to maximise its utility given the prevailing market prices and its utility function.

General equilibrium is found when a set of prices is found such that the amounts that all agents wish to buy and sell (given their utility functions) sums to zero for each commodity. The change in allocation that an agent desires of a particular commodity at certain price levels, certain endowments and a certain utility function is referred to as the *net demand* for that commodity. General equilibrium can then also be defined as the situation in which the sum of all net demands (*the aggregate excess demand*) is zero for all commodities. The equilibrium for a single commodity (*partial equilibrium* in a multi-commodity market) is illustrate in Figure 3.3.

Given the above discussion, we see that the central problem in developing a market-based control system is designing an algorithm that finds the equilibrium allocation of commodities. The search for equilibrium is commonly implemented so that agents submit bids to an *auctioneer* during the search. From these bids the auctioneer updates its information and requests new bids in a iterative fashion; once equilibrium is found the commodities are allocated using the equilibrium prices. Computing the equilibrium and doing (re-)allocation only when the equilibrium has been found is referred to a *tâtonnement process*.

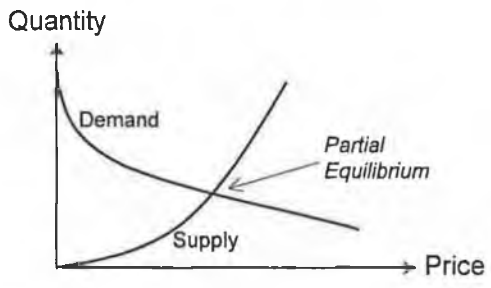


Figure 3.3: Illustration of partial equilibrium.

There are two approaches, *price-oriented* and *resource-oriented*, used to realise tâtonnement processes in market-based control implementations. Price-oriented approaches search for a set of prices such that the supply meets demand using price as the free search parameter; if at an auction iteration, the demand for a commodity exceeds supply, the price of the commodity is increased for the next iteration. A well-known realisation of this approach is the WALRAS system [Wellman, 1993] which has been applied to applications such as ensuring quality-of-service for multimedia applications [Yamaki *et al.*, 1996] and IN load control [Patel *et al.*, 2000a; 2000b].

Resource-oriented approaches differ from price-oriented approaches in that the quantity traded, not the price is used as the free search parameter. The auctioneer sets the allocation of commodities to agents at each iteration and agents report how much they are prepared to pay for an additional small amount of each commodity. The auctioneer then takes these declarations into account when changing the allocation in the next iteration; agents with high willingness to pay get more, the others less. The algorithm terminates when, for all commodities, all agents are willing to pay the same price for a small amount of the commodity.

In chapter 5 we develop a IN load control strategy employing a resource-oriented market algorithm to allocate the resources of an SCP. In the approach utility functions are formulated in terms of the profit generated by successful sessions of IN services and resources are allocated so that overall profit is maximised.

3.2.3 ANT COLONY OPTIMISATION

Ant Colony Optimisation is the application of approaches based on the behaviour of real ant colonies to static and dynamic optimisation problems. The approach has been applied to problems such as routing in circuit- [Di Caro and Dorigo, 1998] and packet-switched [Schoonderwoerd *et al.*, 1997] networks as well as to combinatorial optimisation problems such as the travelling salesman problem [Dorigo and Gambardella, 1997]. These studies have demonstrated that ant-based distributed control can have many advantages in telecommunications applications, for example: high performance in terms of maximising call throughput; scalability; and adaptability to different network topologies, traffic patterns and transient overloads.

Individual ants are very unsophisticated insects from a behavioural point of view, however colonies of ants, acting as a collective, are capable of performing relatively complex tasks, such as building and protecting their nest, forming bridges, colony emigration and finding the shortest routes from the nest to a food source. Such behaviour arises through indirect communication between the ants, affected through modifications induced in the environment, a process named *stigmergy*. An example of stigmergy is the means foraging ants use to quickly find the shortest path to a food source: individual ants following a particular path will deposit a type of highly volatile hormone, called a *pheromone*, giving rise to a *pheromone trail* for that path. In general, ants faced with a decision between alternative paths will follow the path for which the pheromone trail is strongest. However, the shortest path will tend to acquire stronger trails more rapidly (since the ant's round trip time will be shorter and thus more pheromone will be deposited), thereby causing more ants to follow it. This results in positive feedback, ensuring that the shortest path is quickly identified and utilised by the majority of the foraging ants.

The trail laying behaviour described above can be readily applied to routing in telecommunications networks by replacing the routing tables in network nodes by tables of probabilities representing the 'pheromone strength' for onward routes from that node. Every node has a pheromone table for every possible destination in the network, with each table having an entry for each outgoing link. These entries represent the probability that influences whether an ant (or call, or data packet) selects that particular trail (*ie.* outgoing link) in order to reach its destination.

Assuming that the routing tables in network nodes are structured as described above, there are a number of ways in which they can be used to realise an ant-based routing strategy. For example, as described in [Schoonderwoerd *et al.*, 1997], a number of ants can be continually launched from every network node destined for a randomly chosen node. At every node an ant passes through it chooses an outgoing link based on the probabilities in the local pheromone table for its destination. In addition ants update the pheromone tables associated with their source node, by increasing the probability for the link they just arrived on and proportionally decreasing the probabilities for all other links. Updating of pheromone probabilities in this manner will directly affect future ants/calls/packets travelling towards the ant's source node: calls/packets travel from their source to destination by choosing, at every node they pass through, the outgoing link for which the pheromone probability relating to their destination node is the highest.

The process of encouraging ants to avoid routes through the network that are overloaded is achieved by progressively reducing the value by which the ant increments a probability in the pheromone tables as the ant grows older. Ants passing through an overloaded area will be delayed significantly thus their influence on pheromone tables will be lesser than that of ants that have taken more favourable routes. Ongoing manipulation of routing tables by ants should have the effect that optimal routes (given the network topology and current traffic patterns) through the network are established and should be able to adapt to overloads, or resource failures.

In chapter 6 we present a modified version of the pheromone-based routing scheme outlined above which is used by SSPs to decide which SCP an IN service request should be routed towards. SSPs are thereby encouraged to favour SCPs which are in close physical proximity (because round-trip delays are shorter for these SCPs) and/or can be reached via non-overloaded routes.

CHAPTER 4

PERFORMANCE ANALYSIS OF STANDARD SS.7/IN LOAD CONTROLS

A large body of publicly available literature has discussed the results of numerous studies regarding the performance of standard SS.7 and IN load controls. However, these studies typically employ overly simple network topologies and do not investigate the effects of simultaneous invocation of SS.7 and IN controls. The goal of this chapter is to further enhance the understanding of SS.7/IN control operation by addressing both these aspects. Our results also help us identify the inherent limitations of existing SS.7/IN controls and the basic characteristics of an alternative approach that would overcome them.

We commence in §4.1 by providing an overview of the two SS.7/IN models we employ in our study. The first of these is quasi-analytical in nature and is described in §4.2, while the second is simulation-based and is described in §4.3. §4.4 presents and discusses the results of seven different overload scenarios, for which the performance of the load controls is analysed. Finally, §4.5 summarises the results, draws overall conclusions regarding the operation of the SS.7 and IN load controls and identifies desirable characteristics of an alternative approach.

4.1 OVERVIEW OF SS.7/IN MODELS

This section introduces the quasi-analytic and simulation models used to analyse the operation of the SS.7 and IN load controls. It provides an overview of the network topology, the load profiles studied and the various SS.7/IN network element models. Only those characteristics common to both the quasi-analytic and simulation models are discussed here; details specific to each model type can be found in §4.2 and §4.3.

4.1.1 NETWORK TOPOLOGY

A key goal in the design of the quasi-analytic and simulation models is to facilitate the modelling of realistic overload scenarios; therefore it is important to model a SS.7/IN network topology similar to those implemented by network operators world-wide. SS.7 networks are typically deployed using a hierarchical mesh-like arrangement of STPs that forms the backbone of the network, with all SPs being directly linked to a pair of the mesh STPs.[23] We model a network containing a fully connected mesh of four STPs, a single SCP and two 'groupings' of six switches (each having the same characteristics); the topology is illustrated by Figure 4.1.

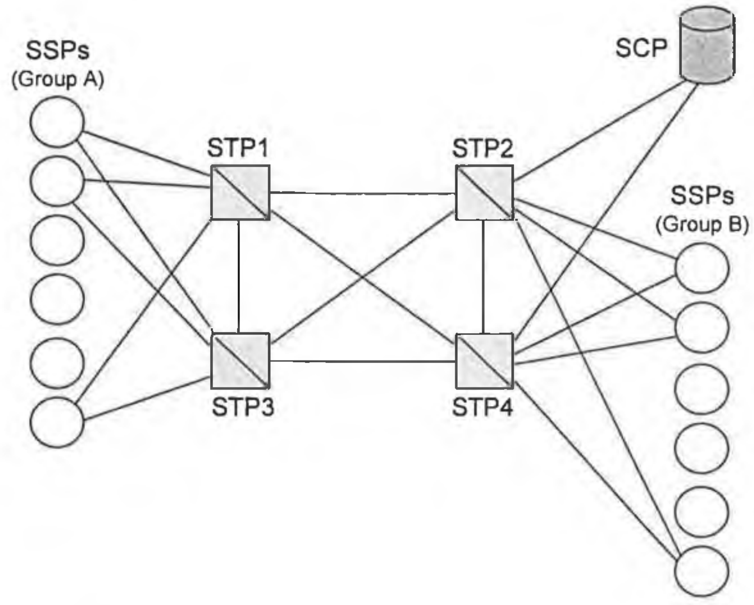


Figure 4.1: SS.7/IN network topology for quasi-analytic and simulation models.

In many existing networks not all switches incorporate SSF functionality and must therefore route all IN service requests to an SSF-enabled switch. However, as network operators increasingly use IN to deploy high-volume services such as Number Portability and Pre-paid there is a trend towards incorporating SSF functionality in as many switches as possible. In our study we assume all twelve switches are SSF-enabled, so the potential effects of switches forwarding IN requests to each other during overload conditions are ignored.

Another significant simplification is to completely omit the trunk network from the models. In particular, no limits are placed on the number of voice circuits that may be concurrently set-up between two SSPs. In real networks the unavailability of trunk network resources during overload would cause the rejection of service requests, thereby limiting the load offered to the signalling network. In addition, the operation of switch network management controls and their impact on load offered to the signalling networks is not considered. Ignoring the impact of switch network management controls is justifiable as [Houck *et al.*, 1994] have shown that their activation can lead to the propagation of overload and that networks perform better in their absence (*cf.* §2.4.1.2). Their omission also allows us to focus on the particular interactions between the SS.7 and IN controls.

[23] To improve fault-tolerance, nodes in an SS.7 network are typically deployed as mated pairs, only one of which is active at a time. As the pair share a single SS.7 point code we can model mated pairs as a single entity. We do not model scenarios in which a mated pair is taken out of service, thus the SS.7 procedures that deal with recovery from this type of failure have been omitted from our models.

A major focus of the study is to analyse the interaction between the SS.7 and IN load controls in scenarios where both SS.7 and IN resources are susceptible to overload. As outlined in [Atai and Northcote, 1996], overloads of this nature have been observed in real networks. The SS.7 load controls were designed to cope with the overload of SS.7 links, but as described in §2.4.1.2, their application to STP processor overload has also been investigated. We assume here that only SS.7 links and not STPs overload; as shown by [Rumsewicz, 1994b] the oscillatory throttling behaviour of traffic sources (SSPs) is broadly similar in both cases. On the IN plane we focus on overloads of the SCP central processor, since the SCP is the physical entity whose behaviour is most critical to IN performance. We ignore the possibility that overloads of SDPs or IPs may occur. In addition, we assume that each SSP incorporates an SRF and that the SCP incorporates an SDF, thus IPs and SDPs do not appear in Figure 4.1.

For simplicity we assume that the linksets shown in Figure 4.1 each contain only a single link. The routing scheme is such that SSPs and the SCP send half of their outgoing traffic over each of their two outgoing links. STP1 and STP3 send half of their traffic for Group B SSPs or the SCP over each if the two links connecting them to STP2 and STP4; similarly STP2 and STP4 send half of their traffic destined for Group A SSPs over each of the two links connecting them to STP1 and STP3. This routing scheme means that the links connecting STP1 with STP3 and STP2 with STP4 do not carry any traffic, so they have been omitted from the models. Links between SSPs and STPs clearly carry a significantly smaller volume of signalling traffic than links interconnecting STPs or connecting the SCP to its STP pair, therefore the latter have also been omitted from the models. Link/node failures and the associated SS.7 recovery procedures are not modelled. Figure 4.2 below illustrates those links that have been fully modelled. In this study we investigate scenarios in which links may become overloaded in either the 'forward' or 'reverse' direction, thus it is convenient to think of each link as actually consisting of two unidirectional links. We use the link numbers assigned in Figure 4.2 throughout the chapter.

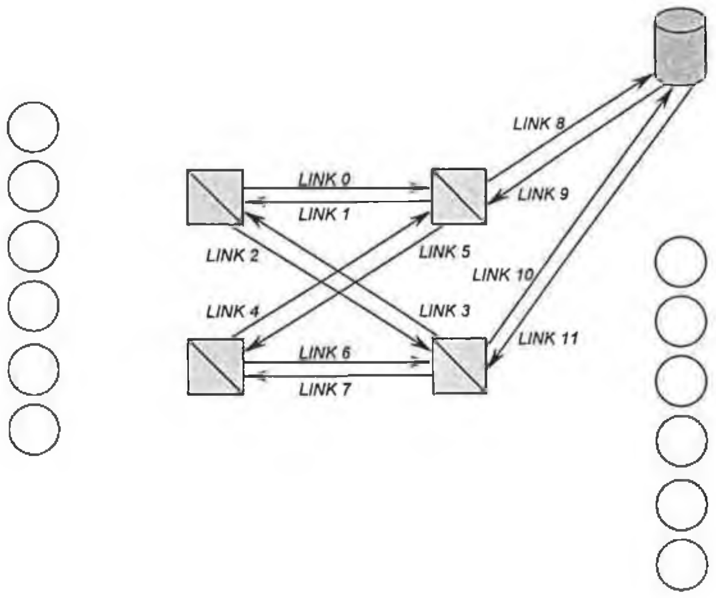


Figure 4.2: Links included in the quasi-analytic and simulation models.

Both the quasi-analytic and simulation models support the three service types described in §4.1.2; one is ISUP telephony, whilst the other two are IN services. ISUP telephony involves the exchange of signalling messages between two SSPs in the network. We model only ISUP telephony service sessions where the two SSPs are in different groups.[24] Overloads may occur for any of the three service types and originate in one or both of the SSP groups.

4.1.2 Supported Services

This section describes the actions and signalling messages / *Information Flows (IFs)*[25] involved in a successful service session of each of the three modelled service types. In keeping with the goal of developing realistic SS.7/IN models, an effort has been made to reproduce as accurately as possible the characteristics of the signalling sequences and processor requirements that would apply in a real network. It should be noted that detailed data on services and processing requirements is not made publicly available by operators and equipment vendors, hence the values used here are estimates only; they are based to a large extent on the values used by other researchers and informal discussions with network operator and equipment vendor representatives. The three supported services are ISUP Telephony, IN Freephone and IN Televoting; each will now be described.

4.1.2.1 ISUP Telephony

The ISUP Telephony service involves the set-up and subsequent tear-down of a voice path between a calling and called party (assumed in our models to be attached to two different SSPs in different SSP groups). Only the SS.7 signalling associated with the service session is modelled; no account is taken of the trunk network. The signalling sequence is based on that described in the SS.7 ISUP recommendations and is illustrated by Figure 4.3. In this figure the horizontal arrowhead lines depict the signalling messages and the vertical lines represent both the functional entities and the passage of time (from top to bottom). The illustrated sequence pertains to a session where the callee answers his/her telephone and the caller terminates the call by putting his/her telephone on-hook.

The sequence of actions involved in the session is as follows:

- Caller goes off-hook and dials the number of the callee;

[24] ISUP telephony sessions between switches in the same group load only the STPs and the links connecting the STPs to SSPs. Since overload of both these resource types is not addressed it is not necessary to model intra-group sessions.

[25] Information Flows are used in IN recommendations to specify the information that can be passed between IN FEs. IFs are mapped to operations invoked by the INAP entity associated with the originating IN FE; operations are then encapsulated within TCAP messages transferred over the SS.7 network.

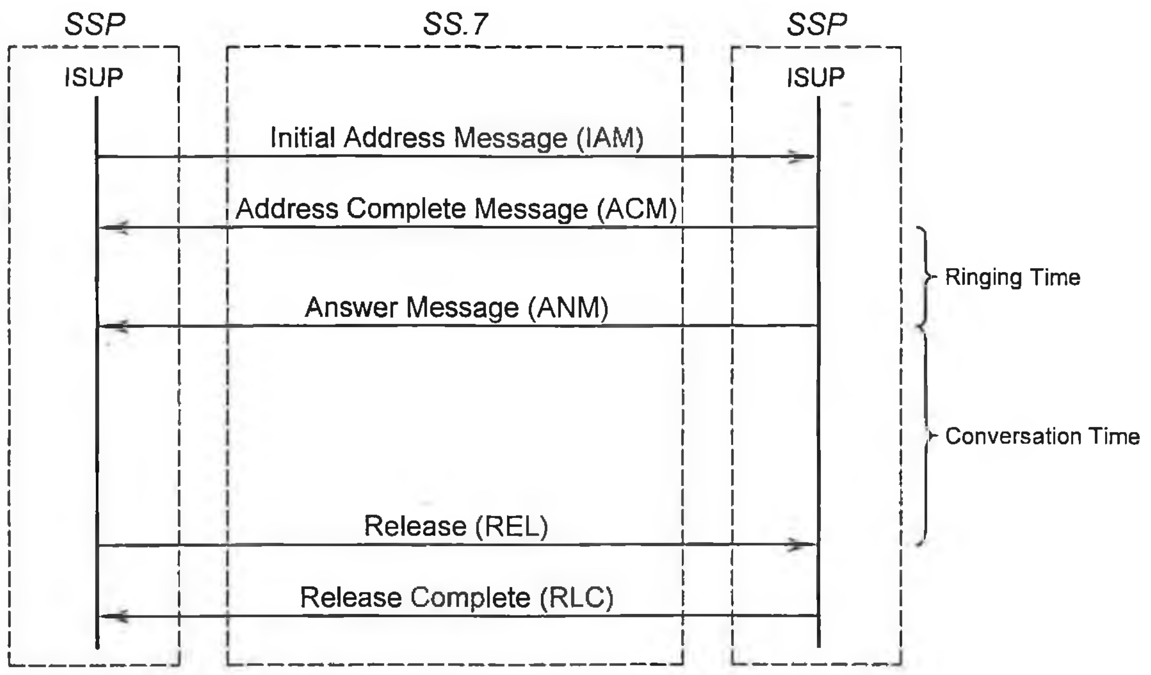


Figure 4.3: Signalling sequence for successful ISUP Telephony service session.

- The dialled digits are collected by the call processing application, which recognises that the callee is attached to a remote switch. The call processing application requests the local ISUP entity to initiate contact with the remote ISUP entity;
- The ISUP entity sends an *Initial Address Message (IAM)* to the remote ISUP entity via the SS.7 network;
- The remote ISUP entity receives the IAM and under instruction from the local call processing application, returns an *Address Complete Message (ACM)* indicating that the called number is valid;
- The call processing application at the callee's SSP rings the callee's telephone;
- The callee goes off-hook, triggering the call processing application to request ISUP to send a *Answer Message (ANM)* to the caller's ISUP entity;
- The ANM is received and the call processing applications set up a voice path between the caller and callee;
- At the end of the conversation the caller (in this case) goes on-hook, triggering the call processing application to request ISUP to send a *Release Message (REL)*;
- The REL is received by the callee's ISUP entity and the call processing application requests that a *Release Complete (RLC)* is sent to the caller's ISUP entity;
- The RLC is received and the call processing applications tear-down the voice path.

Table 4.1 below shows the message lengths for the ISUP Telephony service used in the models. These values are the same as those used by [Smith, 1994a].

Table 4.1: Message lengths for ISUP Telephony service.

Message Type	Message Length in Bytes
IAM	48.0
ACM	20.0
ANM	18.0
REL	22.0
RLC	17.0

4.1.2.2 IN Freephone

When an end-user accesses the Freephone service he/she dials a number with a special prefix (often 1800 or 0800) and is connected to a customer service agent or automatic announcement, with the cost of the service session being charged to the Freephone customer. The network operator allocates the number dialled by the end-user; when it is dialled the call processing application (the CCF) recognises it as a request for the Freephone service. The request is forwarded to the SCF, which accesses its SDF to translate the dialled number to the actual destination number of the Freephone customer. This number is passed to the SSF; control is ceded to the CCF, which proceeds to set-up the call to the destination number. In addition, the billing system is updated to ensure that the Freephone customer, not the end-user, is charged for the service session (we have not modelled the charging aspect in our study).

The sequence of IFs for the IN portion of the Freephone session is illustrated in Figure 4.4. This IF sequence, along with the analogous sequence for IN Televoting (presented in §4.1.2.3) are based on the sequences described by [Lodge, 2000] and use the IN CS-1 IF types, as defined in [ITU, 1993b].

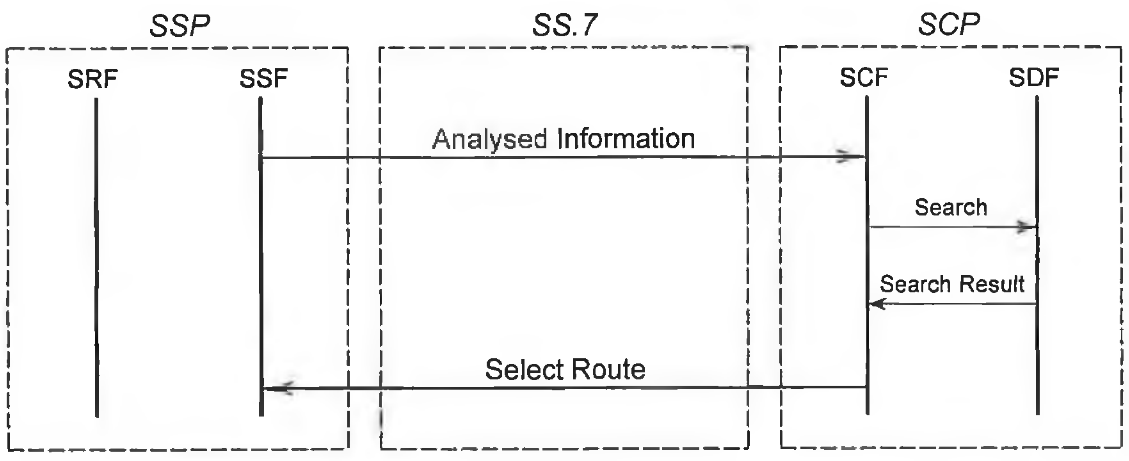


Figure 4.4: Information Flow sequence for IN portion of Freephone service session.

The sequence of actions involved in the session is as follows:

- The end-user goes off-hook and dials the customer's Freephone number;
- The CCF recognises the end-user has requested an IN service and suspends normal call processing, passing control to the SSF;
- The SSF creates an *Analysed Information* IF containing the dialled digits and forwards it to the SCF;
- The SCF receives the IF and creates a new instance of the Freephone service logic program;
- The *Service Logic Program Instance (SLPI)* creates a *Search* IF and sends it to the SDF, which translates the number via a database access and returns it to the SCF within a *Search Result* IF;
- The SCF forwards the result of the number translation to the SSF by means of a *Select Route* IF;
- The SSF instructs the CCF to continue normal call processing using the destination number supplied by the SCF.

Successful completions of the IN portion of the session are immediately followed by a request for an ISUP Telephony session. We assume that the ISUP entity treats requests for these 'follow-on' ISUP calls in the same manner as other ISUP requests (they are subject to ISUP load controls as normal).

For both the quasi-analytic/simulation models and for both IN Freephone and IN Televoting we assume that all signalling messages containing IFs sent from the SSF to the SCF are of length 100 Bytes, whilst those containing IFs sent from the SCF to the SSF are of length 120 Bytes.

4.1.2.3 IN Televoting

Televoting services allow a customer to collect data from a large population of end-users, typically by means of automated announcements and digit collection. They are regularly used by local or national media to run competitions or perform opinion polls. When an end-user dials the designated number, he/she is connected to an announcement system and instructed to select options by dialling sequences of digits. The data entered by the end-user is used to update a database and the end-user's selections are then acknowledged. The sequence of (announcement, digit collection, database update, acknowledgement) may be iterated a number of times depending on the customer's requirements. The IF sequence for the version of Televoting modelled in this study is illustrated byFigure 4.5; it involves two announcement phases and one database update.

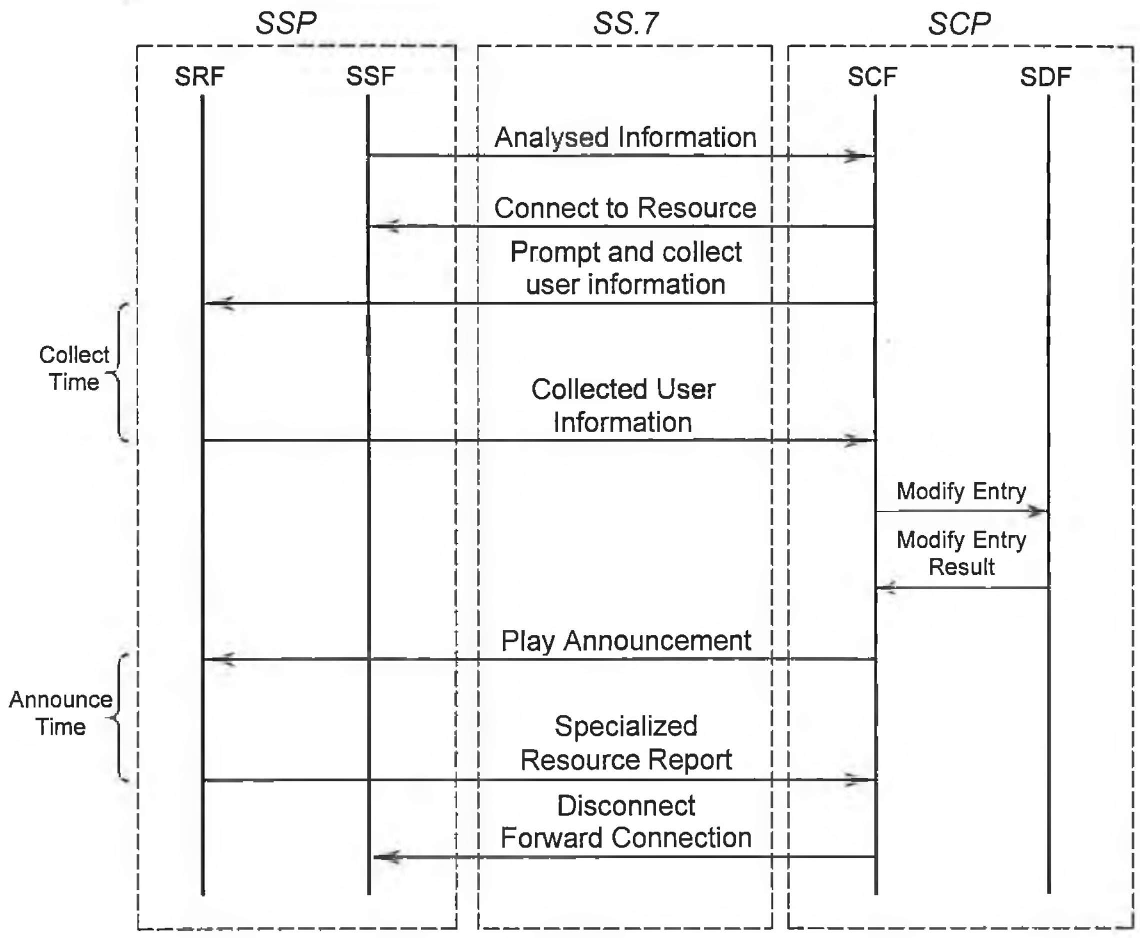


Figure 4.5: Information Flow sequence for IN Televoting service session.

The sequence of actions involved in the session is as follows:

- The end-user goes off-hook and dials the customer's Televoting number;
- The CCF recognises the end-user has requested an IN service and suspends normal call processing, ceding control to the SSF;
- The SSF creates an *Analysed Information* IF containing the dialled digits and forwards it to the SCF;
- The SCF receives the IF and creates a new instance of the Televoting service logic program;
- The SLPI creates and sends a *Connect to Resource* IF to the SSF that requests a voice path be opened between the end-user and the SRF;
- The SLPI creates and sends a *Prompt and collect user information* IF to the SRF requesting that the indicated announcement be played and the digits dialled in response by the end-user be collected;
- The SRF returns the end-user dialled digits within a *Collected User Information* IF;

- The SLPI creates an *Modify Entry* IF and sends it to the SDF, which updates the relevant database entry and confirms the update to the SLPI by means of an *Modify Entry Result* IF;
- The SLPI creates and sends a *Play Announcement* IF to the SRF, requesting that the indicated announcement be played;
- The SRF plays the announcement and returns a *Specialized Resource Report* IF as confirmation;
- The SLPI creates and sends a *Disconnect Forward Connection* IF to the SSF requesting it to tear-down the voice path between the end-user and the SRF;
- The SSF cedes control to CCF for further call processing.

4.1.3 SS.7/IN Network Element Modelled Functionality

The section provides an overview of the functionality of the SS.7/IN network elements included in the quasi-analytic and simulation models. Particular emphasis is placed on the choice of parameter values for the SS.7 and IN overload detection and load throttling procedures.

4.1.3.1 Modelled STP Functionality

The models can contain two types of STP: 'standard' STPs, incorporating the functionality of the MTP and 'enhanced' STPs, which additionally incorporate an SCCP entity that performs GTT. We assume that STPs have a single central processor that carries out the MTP3 and (if present) SCCP functionality, together with separate processors for handling incoming and outgoing signalling links. The central processor is assumed to be of infinite capacity, so packets experience no queuing or processing delays when passing through it. We describe the modelled STP functionality by first describing the modelled MTP functionality and then describing the modelled SCCP functionality. It should be noted that these descriptions also pertain to the SSP and SCP, which also contain MTP and SCCP entities.

MTP Entity

As described in §2.2 the main purpose of the MTP is to reliably transport signalling messages (MSUs) between signalling nodes in the SS.7 network. Both the quasi-analytic and simulation models incorporate the MTP routing functionality, using the routing scheme described in §4.1.1. We do not model STP or link failures and therefore omit the comprehensive and complex MTP3 failure recovery procedures from the model.

The MTP3 TFC procedure, using the IO, is modelled in detail in both models. This procedure ascertains the overload level of all outgoing links through measurements of the current size of the link queue. For both models, the overload onset threshold was set to 1400 Bytes and the overload abatement threshold was set to 800 Bytes (these are the values recommended by [Manfield *et al.*, 1994]). During the period in which an outgoing link is overloaded a TFC message is generated for every $n = 8^{th}$ MSU processed by the link. The TFC indicates overload of the SP indicated in the MSU's DPC field and is sent to the SP indicated in the MSU's OPC field. Note that SS.7 recommendations mandate that all management messages including TFCs are assigned a higher priority level than other signalling messages, hence they will not be affected by any link overloads on their route through the network. Specific details of the manner in which the TFC procedure, including queue size estimation and TFC generation, is realised for the quasi-analytic and simulation models can be found in §4.2 and §4.3.

SCCP Entity

As described in §2.2, SCCP adds capabilities to those of the MTP in order to provide a full Network Layer service to its users. These capabilities include GTT, which translates arbitrary network addresses (typically dialled digits) into SS.7 point codes. This functionality is often deployed in STPs within an SS.7 network; in our models SCCP/GTT can be optionally deployed in STP2 and STP4. In such a scenario, SSPs forward INAP messages to one of these STPs. Upon arrival they are passed to the local SCCP entity, which changes the value in their DPC field to that of the SCP and passes them back to the MTP for transfer to the SCP.

We also model a SCCP load throttle closely resembling the standardised ISUP load throttle. As discussed in §2.2.1.3, load throttles implemented for the SCCP connectionless service are generally similar in nature to the ISUP load throttle, so the latter has been used as the basis for standardisation. Our SCCP load throttle updates its overload status for destination signalling points by responding, under the control of two timers (which we label *T1* and *T2*), to the arrival of CIs from the MTP. Throughout this study we set the duration of *T1* to 0.3s and the duration of *T2* to 5.1s.[26] Our version of the throttle employs percentage thinning, rejecting at random a percentage of the messages arriving at the SCCP entity that are addressed to the overloaded destination. In this study we have set the number of overload levels to four. Specifically, 0%, 33%, 67% and 100% of messages are rejected at overload levels 0, 1, 2 and 3 respectively, where level 0 indicates no overload and level 3 indicates severe overload.

4.1.3.2 Modelled SSP Functionality

The focus of this study is on the analysis of the performance of the SS.7 and IN load controls in the presence of SS.7 link and/or SCP processor overloads. In view of this, SSPs are modelled in a relatively simple manner. We assume a similar processor architecture to that used for STPs: processors for handling signalling links and a single central processor of infinite capacity, which carries out all MTP3 and user part functionality. Therefore, we do not address the manner in which overloads of SSP resources would impact on the operation of the SS.7/IN load

[26] These values are chosen arbitrarily, but are within the allowed range of values for the ISUP *T*29 and *T*30 specified in SS.7 recommendations.

controls. We describe the SSP model by first describing the modelled ISUP functionality and then describing the modelled functionality of the two IN entities, the SSF and the SRF.

ISUP Entity

In our models the ISUP entity maintains the originating and terminating call state process instances for each invoked ISUP Telephony service.[27] This involves modelling of end-user behaviour and the realisation of the signalling sequence shown in Figure 4.3. A description of the signalling sequence can be found in §4.1.2.1 and the details of how it is modelled can be found in §4.2 and §4.3.

Also included is the ISUP load throttle, which is modelled in accordance with the standardised IO ISUP throttle (described in §2.2.1.1). We set the duration of timer *T29* to 0.3s and the duration of timer *T30* to 5.1s. We employ three overload levels, as follows: level 0 indicates no overload so no messages are throttled, level 1 indicates mild overload so all IAM messages are throttled, finally level 3 indicates severe overload with all message types being throttled. IAMs are given a lower priority than the other 'call progress' messages since the latter are associated with service sessions already admitted to the network and on which network resources have already been spent.

SSF Functional Entity

In our models the SSF FE maintains the *Basic Call State Model (BCSM)* instances for the IN Freephone and IN Televoting services,[28] the IF sequences for which are described in §4.1.2.2 and §4.1.2.3. It manages the complete SSP-side processing of a IN service session from trigger detection; through instantiation of a BCSM instance for the session; handling of communications with the other IN entities; and finally, session termination.

The SSF FE also includes the ACG load throttle. This throttle accepts/rejects requests for IN services using call gapping intervals, the lengths of which are dictated by ACG messages, periodically sent by the SCF to the SSF. The version of ACG we model is based on that described in [Northcote and Smith, 1998], which bears close resemblance to the controls deployed in existing IN systems. A gapping throttle can be put in place for each traffic source at an SSF; we consider a source to be constituted by all the requests for a particular service type that originate at a particular SSP.

We employ thirteen separate overload levels, which index the table of interval lengths listed in Table 4.2 below (these values are those standardised in [Bellcore, 1994; 1995]). Upon ACG

[27] These call state process models incorporate functionality associated with both the ISUP entity and the CCF.

[28] The BCSM is used in IN recommendations to provide a model of a telephony session containing *trigger points*, which, if *armed*, can result in the invocation of an IN service sessions. BCSM instances should reside in the CCF, however for simplicity we place them in the SSF functional entity model.

arrival, the actual gap interval length that is put in place is calculated by randomising between 90% and 110% the table value indexed by the overload level contained in the newly arrived ACG message. Level 0 has a gap interval length of 0s and is used to indicate that gapping is currently inactive. Control refreshing is not used, that is, gap interval timers are not reset upon arrival of a new ACG message. We do not model the use of a gap duration, since the SCFs periodically send ACG messages and so its use is unnecessary.

Table 4.2: Gap interval lengths for the ACG throttle.

Overload Level	Gap Interval (s)
0	0
1	0.1
2	0.25
3	0.5
4	1.0
5	2.0
6	3.0
7	4.0
8	6.0
9	8.0
10	11.0
11	16.0
12	22.0

SRF Functional Entity

We do not address overload of SRF resources in this study, hence we assume that there are an infinite number of voice circuits and associated resources available for servicing requests for announcements and data collection. We assume that the time required to set up these circuits is negligible. Therefore the only delay incurred at the SRF is the delay between the arrival of the INAP message requesting an action and the generation of the appropriate response message. This is the delay relating to the playing of announcements and data collection. The values for these delays differ slightly between the quasi-analytic and simulation models; details can be found in §4.2 and §4.3 respectively.

4.1.3.3 Modelled SCP Functionality

For the SCP we again employ a processor architecture in which a single central processor carries out the high-level functionality and peripheral processors handle the signalling links. The central processor has a service rate that we vary for the different scenarios addressed. As a simplification, we assume that all messages arriving at the SCP experience the same processing delay. We assume that the time the central processor spends assigning its overload level generating ACG messages is negligible.[29] The central processor has a single input queue whose capacity is 15,000 Bytes. Messages arriving when the queue is full are discarded; we assume that discarding requires no processing resources. We will describe the remainder of the SCP model by first describing the modelled SCF functionality and then describing the modelled functionality of the SDF.

SCF Functional Entity

The SCF functional entity incorporates the SLPs for the IN Freephone and IN Televoting services, the IF sequences for which are described in §4.1.2.2 and §4.1.2.3. The complete SCP-side processing of the IN service sessions from SLPI instantiation, handling of communications with other IN entities and SLPI termination.

In our models the SCF functional entity also includes the ACG overload detection procedure, which is again based on that described in [Northcote and Smith, 1998]. At measurement intervals of duration 10s an overload level is assigned to the SCP using a processor utilisation based measurement. The overload level is assigned using the following algorithm, which is designed with the aim of to ensuring that the utilisation is kept between a lower threshold value (we use 0.8 Erlangs) and an upper threshold value (we use 0.9 Erlangs):

```
If utilisation > upper_threshold and current_level < max_level then:
  new_level = current_level + 1
else
  If utilisation < lower_threshold and current_level > 0 then:
    new_level = current_level - 1
```

The number of overload levels is thirteen, corresponding to the thirteen gap intervals that can be used by the SSF ACG throttle. We assume that the SCF employs autonomous ACG messages to inform the SSFs of the SCP's overload status. Each time a new overload level is assigned, ACG messages are sent to all SSPs in the network. The ACG messages indicate the overload level, which is then used to index the gap interval for each of the sources corresponding to that SSP. Therefore, once the SCP overloads the same gap interval will be put in place for all sources (unless the ACG is lost or excessively delayed in the SS.7 network).

SDF Functional Entity

The SDF functional entity models the reading and updating of a database in response to requests arriving from the SCF. We assume that the delays for databases reads and updates are incorporated into the processing delay for the INAP message that arrived at the SCF and triggered the SCF to contact the SDF. We do not address the potential for the SDF/database to be the bottleneck in the network.

[29] In real systems the assignment of ACG overload levels is often the responsibility of a management system running on a separate processor.

4.2 SS.7/IN QUASI-ANALYTIC MODEL

The SS.7/IN quasi-analytic model combines analytic techniques with random number generation to evaluate the transient behaviour of the SS.7 and IN load controls. It employs and significantly enhances a hybrid modelling approach originally proposed in [Zepf and Willmann, 1991]. The structure of the model is illustrated in Figure 4.6. It is comprised of three sub-model types:[30] an ISUP user part sub-model, an IN user part sub-model[31] and a resource sub-model for the modelling of resource utilisation, queue sizes and overload detection. The IN user part sub-model supports Freephone and Televoting, whilst the ISUP user part sub-model supports ISUP telephony. All services are modelled in accordance with the specifications in §4.1.2.[32]

The user part sub-models take as inputs current service request arrival rates, the characteristics of the supported services and the current parameters of the SS.7 UP and IN load throttles; they output the per message type volumes of traffic offered to network resources for a given iteration interval. The resource sub-model uses user part sub-model outputs, along with information on the traffic held in link/SCP queues at the start of the iteration interval, to calculate mean utilisation and mean queue lengths for the iteration interval. It also models generation of TFC/ACG messages by overload detection procedures. This section provides a brief overview of the three sub-models; a fuller specification of each is provided in Appendix A.

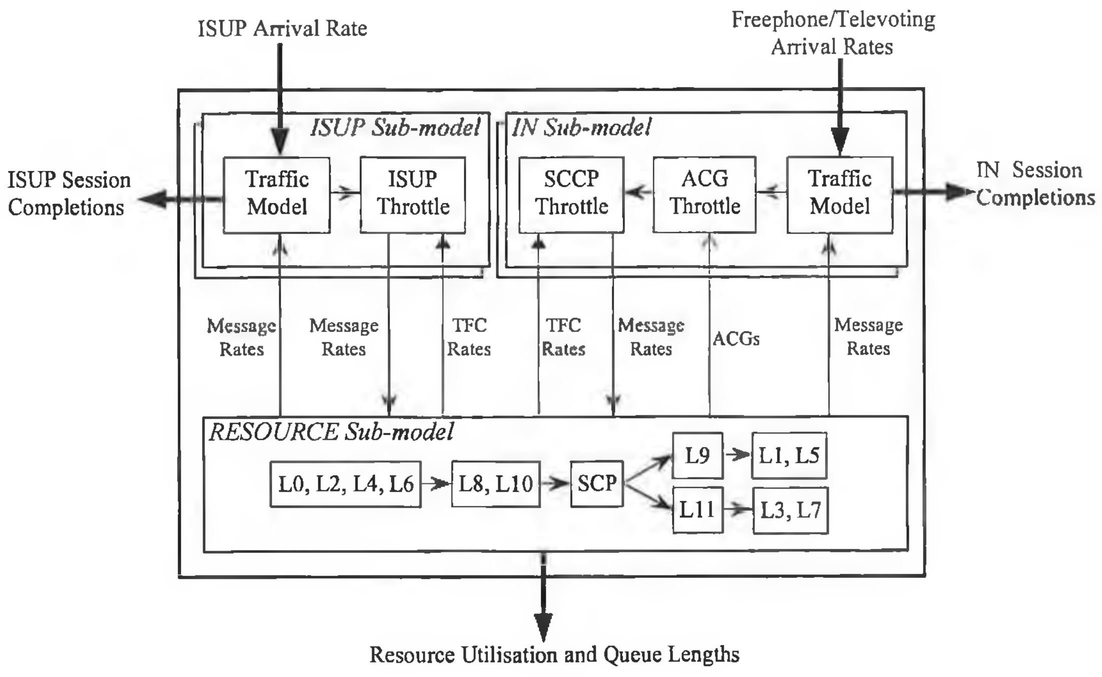


Figure 4.6: Structure of SS.7/IN Quasi-Analytic Model.

[30] There are multiple instances of the user part sub-models: each SSP has one ISUP and one IN sub-model instance, whilst the SCP has one IN sub-model instance. There is only one resource sub-model instance.

[31] The IN user part model incorporates aspects of the functionality of SCCP, INAP, the IN SSF/SCF and the service logic for the supported services.

[32] It would be straightforward to modify the traffic models to support a different mix of services with different characteristics.

4.2.1 Overview of ISUP Sub-Model

The ISUP sub-model provides a model of the ISUP messages generated by ISUP telephony service sessions and the operation of the ISUP load throttle. First-offered service requests are assumed to arrive according to Poisson arrival processes, where the mean arrival rates for particular SSPs can vary with time over the course of the iterative analysis. ISUP service session message sequences are modelled in accordance with §4.1.2.1, with the assumption that the callee always answers and that the caller initiates session termination. The duration of the ringing time is a constant value of $3s$ (10 iteration intervals) and the duration of the conversation time is exponentially distributed, with a mean value of $160.8s$ (536 iteration intervals).[33] The sub-model also supports service request reattempts as follows: all service requests which are throttled by the ISUP UP controls, or whose sessions are prematurely ended due to discard of IAM, ACM or ANM messages (not RELs/RSCs) reattempt with probability P_R. Reattempting requests re-enter the network after a delay uniformly distributed in the range (T_1, T_2) seconds. The maximum number of times a request may reattempt is limited to R_{MAX}.

We model the operation of a ISUP load throttle (relating to a single destination ISUP entity) as an embedded Markov chain.[34] The state transition diagram for the embedded Markov chain is illustrated in Figure 4.7. The state k is defined as the throttle's current load level for the destination user part, where k ranges from value 0 (no load throttling) to K (maximum load throttling).

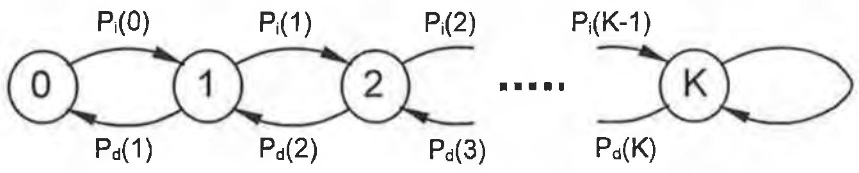


Figure 4.7: Embedded Markov chain for modelling UP load throttling.

The embedded points in time are the instants at which the load level changes, due either to the arrival of a CI (triggered by a TFC arrival) whilst $T29$ is inactive (load level increases) or to the expiry of $T30$ (load level decreases). $P_d(k)$ denotes the probability of decreasing the load level whilst in state k and $P_i(k)$ denotes the probability of increasing it whilst in state k. The values

[33] This value was chosen to be in broad agreement with the simulation model, which employs a detailed model based on the distributions described in [Bolotin, 1994].

[34] Although the length of time the load throttle spends in a particular state (load level) follows an arbitrary probability distribution (it is dependent on TFC/CI arrival times) at the instants it does change state the new load level depends only on the current load level. We therefore have an embedded Markov Chain (*cf.* §3.1.1.2).

of these probabilities depend on a number of factors, such as t_{T29} (the duration of $T29$) and t_{T30} (the duration of $T30$), the state k and link overload status.

In order to simplify the analysis of the transient behaviour of the ISUP load throttling procedure we assume that that changes of load level k and other parameters only occur at discrete time instants t_n, $n = 0, 1, 2, \ldots$, at the end of each iteration interval. We can then use mean values over these intervals for the analysis. As a further simplification we choose the iteration interval length $t_{n+1} - t_n = t_{T29}$ and $t_{T30} = A \cdot t_{T29}$, where A is an integer. The former is a natural choice, since all CIs arriving while $T29$ is active are ignored; this means that the load level can only change a maximum of once per iteration interval.

We assume that all messages (including TFCs) are generated according to Poisson processes. TFC messages are generated in proportion to arriving message rates, therefore, since we also assume that TFCs always arrive at their destination during the same interval they are generated,[35] they also will arrive at SSP according to Poisson processes. Given this assumption, it is straightforward to calculate the probability of the arrival of a CI at an ISUP user part whilst the relevant $T29$ timer is inactive. The probability of CI arrival is also the probability, $P_i(k)$, that the load level k is increased. If we let t_n denote iteration interval number n, $\lambda_{TFC}(t_n)$ denote the mean arrival rate of TFCs during iteration interval t_n, $\tau_{T29}(t_n)$ denote the portion of the iteration interval for which $T29$ is active, then $P_i(k)$ is given by:

$$P_i(k) = 1 - e^{-\lambda_{TFC}(t_n)[t_{T29} - \tau_{T29}]}$$

Random number generation is used at each iteration to decide whether a CI does actually arrive and, if one does, the exact time during the iteration interval at which it does (the latter is used to calculate the time for which $T29$ will be active in the next iteration interval).

Knowledge of CI arrivals during the current iteration interval is used to update the load level for the next interval. If a CI arrived for a destination user part the corresponding load level is incremented. If no CI arrived and the relevant timer $T30$ is active a check is made to ascertain if $T30$ expires; if it does the load level is decremented ($P_d(k) = 1$).

4.2.2 Overview of INAP Sub-Model

The IN sub-model provides a model of the traffic generated by sessions of the Freephone and Televoting services and of the operation of the ACG and SCCP load throttles. First-offered service requests are again assumed to arrive according to Poisson processes and service sessions

[35] TFCs are management messages, so will be given the highest priority with the MTP. It is therefore reasonable to assume that they always arrive promptly at their destination.

are modelled in accordance with the descriptions in §4.1.2.2 and §4.1.2.3. End-user reattempts are modelled in a manner analogous to that outlined for the ISUP sub-model.

The model of the operation of the ACG load throttle is predicated on the assumption that all IN service requests arrive according to a Poisson arrival process;[36] given this, it is straightforward to estimate how the presence of a gap interval modulates the level of accepted service requests. Let $T'(t_n)$ denote the mean interarrival time of service requests, $\lambda'(t_n)$ denote the mean arrival rate of service requests, $g(t_n)$ denote the gap interval length, $T(t_n)$ denote the mean interdeparture rate of accepted requests from the ACG throttle and $\lambda(t_n)$ denote the mean departure rate of accepted requests. Because of the Poisson arrival process interarrival times will be exponentially distributed. The memoryless property of the exponential distribution means that the mean interdeparture time in the presence of gapping is given by:

$$T(t_n) = g(t_n) + T'(t_n)$$

Therefore:

$$\frac{1}{\lambda(t_n)} = g(t_n) + \frac{1}{\lambda'(t_n)} \quad \Rightarrow \quad \lambda(t_n) = \frac{\lambda'(t_n)}{1 + g(t_n)\lambda'(t_n)}$$

The SCCP load control is modelled in accordance with the description given in §4.1.3.1. Load levels for destination SCCP entities are updated in response to CI arrivals and timer expiry in a analogous manner to the ISUP control, described previously. Load throttling is achieved on a stepwise basis with an increasing percentage of the traffic being throttled as the load level increases, as the load level is incremented. If we denote by $\lambda'(t_n)$ the arrival rate of messages to the SCCP throttle, by $\lambda(t_n)$ the departure rate of accepted messages from the SCCP throttle, by k_{SCCP} the current SCCP load level and by K_{SCCP} the maximum SCCP load level, we then have:

$$\lambda(t_n) = \left[1 - \frac{k_{SCCP}}{K_{SCCP}}\right] \lambda'(t_n)$$

4.2.3 Overview of SS.7/IN Resource Sub-Model

This section describes the iterative procedure to calculate the utilisation and queue lengths for links/SCPs, TFC generation rates and SCP overload levels. At each iteration the volume of message traffic offered to all resources in the network for the iteration interval is computed. This includes messages held in the resources queue at the start of the iteration interval, messages arriving directly from user parts and messages arriving from other resources.

[36] Note that in the presence of end-user reattempts this is an approximation.

The topology and routing scheme of the network being modelled will dictate the order in which the calculations for the various resources can be made.[37] For example in the network topology used for this study (Figure 4.2, page 70) L0, L2, L4 and L6 direct traffic towards L8 and L10, therefore their utilisation must be calculated before the load offered to L8 and L10 can be computed. The full order in which resource utilisation and queue lengths are calculated for this network topology is illustrated in Figure 4.6, page 81.

Because the load offered to a resource queue differs from iteration to iteration (it consists of a different mix of message types) it is necessary to keep track of both the number and position of messages of various types held in a queue, so that resource utilisation can be accurately calculated. To achieve this, the concept of 'message groupings' is introduced: a message grouping is the set of messages offered to a resource queue during one iteration interval. Queues are modelled as ordered lists of message groupings and resource utilisation is calculated on the basis that groupings at the head of the list are processed before those at the tail.

Once the queue lengths and utilisation of all the links and SCPs in the network have been calculated, the next step is to assign an overload status to each link and an overload level to the SCP. Link overload status is assigned according to the IO procedures described in §4.1.3.1: if a link is overloaded one TFC is generated for every N messages sent by the source user part to a the destination user part and processed by the link in question during the current iteration interval. The SCP overload level is assigned on the basis of a processor utilisation measurement, as described in §4.1.3.3.

4.3 SS.7/IN SIMULATION MODEL

The SS.7/IN simulation model uses discrete event simulation (*cf.* §3.1.2) to provide a detailed model of the operation of the transient behaviour of the SS.7 and IN load controls. The flow of traffic through the network is modelled on a packet-by-packet basis, facilitating the modelling of complex behaviour that can not be easily modelled using rate-based approximation (as used in the quasi-analytic model). This added flexibility is gained at the expense of significant added complexity in model development and greatly longer execution times to produce results for overload scenarios. This section provides a description of the simulation model characteristics not addressed in the overview of the models provided in §4.1.

[37] There may be network topologies with routing schemes for which it would be impossible to define an ordering that would allow all resource's utilisation and queue lengths to be calculated in this manner. In such circumstances the model as described here could not be employed.

4.3.1 STP Simulation Model

The structure of the enhanced STP simulation model is illustrated in Figure 4.8 (the standard STP model is similar, the only difference being the absence of the SCCP functional block). The figure shows the functional blocks that constitute the model and the flow of MSUs, CIs and other indications between them. Also shown is the mapping of the blocks to the processors upon which their functionality is executed.

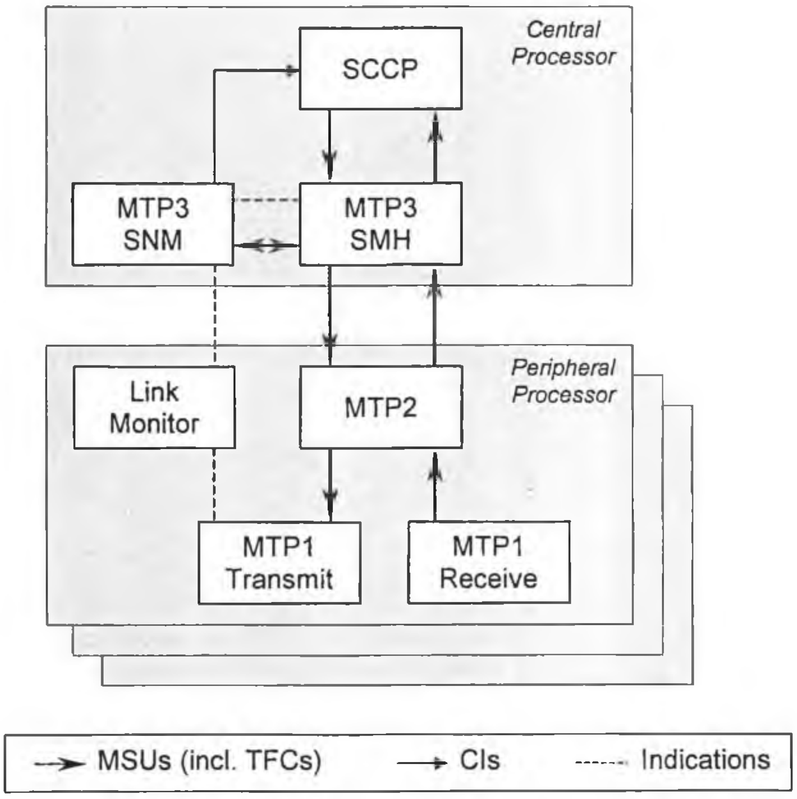


Figure 4.8: Structure of enhanced STP simulation model.

The SCCP functional block implements GTT and the SCCP load throttle; the MTP3 *Signalling Message Handling (SMH)* block implements the MTP 3 routing functionality and the MTP3 *Signalling Network Management (SNM)*[38] block implements the TFC procedure, all as described in §4.1.3.1. The remaining functional blocks provide a detailed model of the signalling links; these were not modelled in detail in the quasi-analytic model. Since an STP can be attached to a number of signalling links there will be multiple instances of the link-related functional blocks.

Real SS.7 links are bi-directional, utilising a retransmission procedure in which signal units transmitted in one direction are acknowledged by signal units later transmitted in the other direction. As noted in §4.1.1 we adopt the convention of treating links as two unidirectional links, thus the MTP1 Transmit and MTP1 Receive functional blocks refer to end-points of two of the numbered links in our network topology. These two links are 'paired' in the simulation

[38] Both the SMH and SNM are defined as MTP functional blocks in SS.7 standards.

model in order to facilitate operation of the MTP2 acknowledgement procedure. The MTP1 functional blocks transmit/receive signal units at 64Kbps (exceptions are those relating to links between STPs and SSPs, which are modelled as being of infinite capacity); neither link disturbances nor link failures are modelled.

The MTP2 functional block partially models the MTP2 retransmission procedure, maintaining both a *Transmission Buffer (TB)* and a *Retransmission Buffer (RTB)* and implementing a *Go-back-N* acknowledgement procedure. Since link disturbances are not modelled the MTP2 error monitoring functions and signal unit retransmission are not included. Similarly, link failures do not occur, hence the MTP2 link recovery and associated functions have not been modelled.

The task of Link Monitor functional block is to report the overload status of its associated outgoing signalling link to MTP3. To do so it continually monitors the size of the link queue. It assigns the overload status based on traversals of the link overload onset and abatement thresholds. This is done in accordance with the IO (so the overload status is either TRUE or FALSE).

4.3.2 SSP Simulation Model

The structure of the SSP simulation model is illustrated in Figure 4.9. The MTP3 and SCCP functional blocks operate in the same manner as described in the previous section, however the blocks relating to the signalling links do not. As outlined in §4.1.1 the links connecting SSPs to STPs are to be excluded from our study. To achieve this the MTP1 functional blocks in the SSP (and the corresponding pair in the STP) are modelled as having an infinite service rate. As there will be no link queues a Link Monitor functional block is not included. The SSF and SRF blocks implement their functionality as described in §4.1.3.2, so are not discussed further here.

The SSP simulation model includes traffic generator functional blocks for each of the three supported services. These blocks generate first-offered requests for their associated service at random time intervals, exponentially distributed around a specified mean. This mean value can vary over the course of a simulation run, allowing for the modelling of a range of overload scenarios. Other functional blocks in the simulation model trigger the traffic generators to generate service reattempts after an original service request has been throttled or abandoned due to message discard.

The ISUP functional block maintains originating and terminating call state processes for ISUP telephony service sessions, models user behaviour during those sessions and implements the ISUP throttle. The simulation model includes a user behaviour model that is more realistic than that employed for the quasi-analytic model. It is based on the findings of [Bolotin, 1994], in which an analysis of real traffic data is performed and the results used to develop a model of end-user behaviour for ISUP telephony. In our model 3% of calls are 'short calls' (relating for example to credit card validations), the 'conversation' time for which are uniformly distributed in the range (1s, 3s). The remainder are 'standard calls,' of which 11% terminate due to a busy called line, 11% terminate due to no answer and 78% proceed as normal. If the callee's line is busy the caller is assumed to go on-hook immediately. If the callee's line is ringing the caller goes on-hook after a random delay uniformly distributed in the range (10s, 15s). If the callee answers he/she will do so after a random delay uniformly distributed in the range (0s, 15s). For answered calls the conversation time is randomly selected using a call holding time distribution of the kind defined by [Bolotin, 1994]. This distribution characterises mainly residential traffic and is a mixture of two lognormal distributions with an overall mean call holding time of 161s and a coefficient of variation of 2.0.[39]

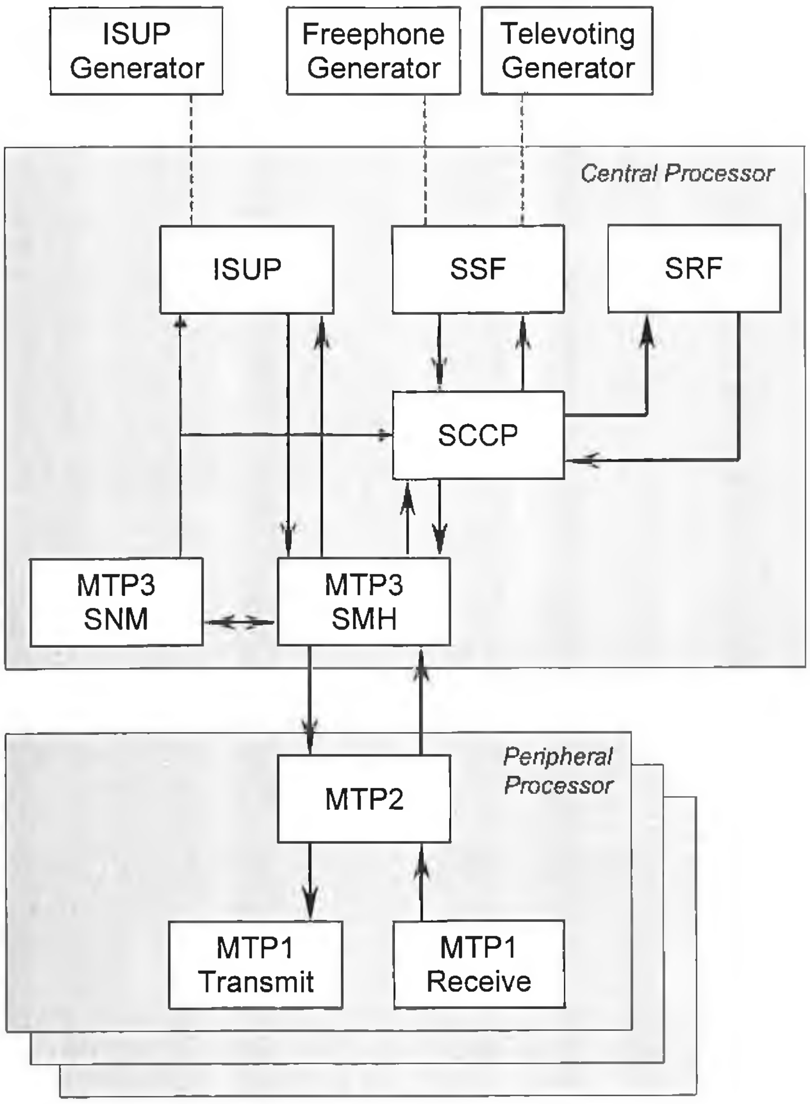


Figure 4.9: Structure of SSP simulation model.

[39] More precisely the distribution is characterised by the following parameters (*cf.* [Bolotin, 1994]): $\alpha = 0.35, \mu_1 = 1.37, \sigma_1 = 0.36, \mu_2 = 2.32, \sigma_2 = 0.58$.

4.3.3 SCP Simulation Model

The structure of the SCP simulation model is illustrated in Figure 4.10. The signalling link related, MTP3 and SCCP functional blocks all operate in the same manner as described above for the STP model. The SCF and SDF blocks realise the functionality described in §4.1.3.3, namely the instantiation and termination of SLPs, handling of communications with other IN entities and the reading/updating of service-related information in the database.

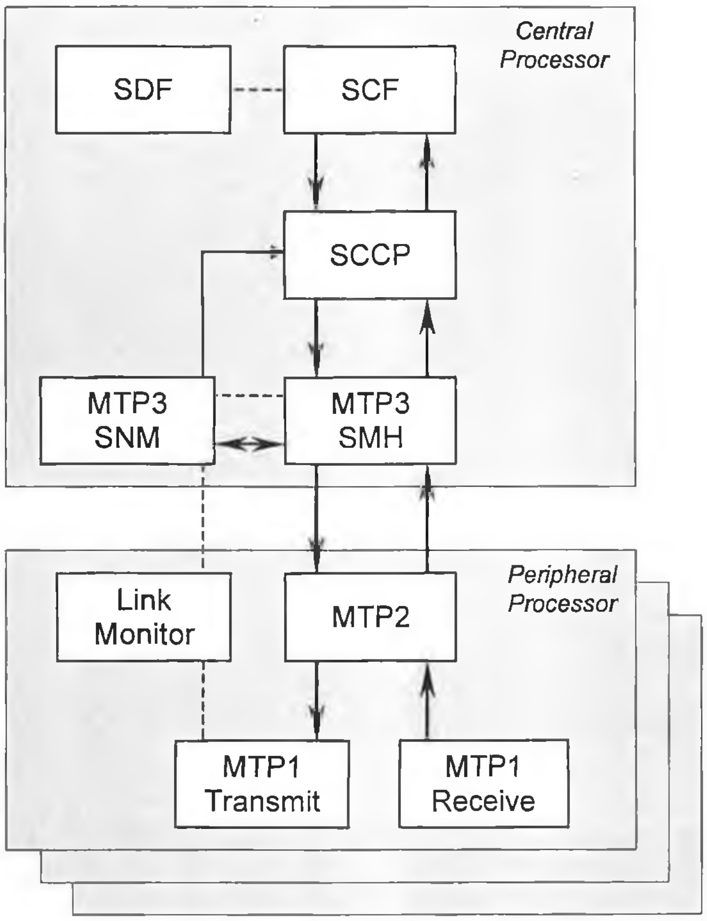


Figure 4.10: Structure of SCP simulation model.

4.4 ANALYSIS OF PERFORMANCE OF SS.7/IN LOAD CONTROLS

In this section we present results obtained using the quasi-analytic and simulation models. Analysis of these results demonstrate the performance characteristics of the SS.7 and IN load controls when operating independently of each other and the interactions that may occur when they are invoked simultaneously. The results are presented as seven scenarios, in which network parameters and load profiles are varied such that different subsets of network resources overload. The scenarios were developed with the aim of achieving the following goals:

1. Partial validation of the efficacy of the quasi-analytic and simulation models through reproduction of previously demonstrated behaviour of the SS.7 and IN load controls;

2. Exploitation of the models' use of a realistic network topology to add to the understanding of the behaviour of the SS.7 and IN load controls;
3. Analysis of the performance of the independent operation of the SS.7 and IN load controls in overload conditions not addressed in the literature;
4. Analysis of overload conditions in which the SS.7 and IN load controls are invoked simultaneously and the identification of any harmful or beneficial interactions.

The first three scenarios relate to the operation of the SS.7 and IN load controls in isolation and address goals 1 and 2; the remaining four scenarios address goals 3 and 4. For the first three scenarios results obtained using both the quasi-analytic and simulation models are presented. Discounting effects relating to differences in features modelled (for example, ISUP REL/RSC reattempts are included in the simulation model only) both model types are seen to produce similar results. For brevity we use only results obtained using the quasi-analytic model for the final four scenarios.

We employ three performance metrics in our analyses, namely: resource utilisation, resource input queue size, and service session offer/completion rates. Resource (SCP or link) utilisation gives an indication of how efficiently load controls utilise resources during high load conditions. SS.7 standards recommend signalling links never experience sustained utilisation levels greater than 0.8 Erlangs. For the SCP we have previously stated that ACG aims to keep its processor utilisation between the levels of (0.8, 0.9) Erlangs.

Resource queue sizes also give an indication of the how efficiently load controls utilise resources. In addition, they give an indication of both the queuing delays experienced by signalling messages and whether or not messages are discarded due to full queue occupancy. Large queuing delays clearly have an adverse impact on the quality-of-service received by the end-user, whilst message discards often lead to premature termination of service sessions and therefore have a very detrimental effect on overall network performance.

Although useful, resource utilisation and queue size are somewhat crude metrics of load control performance. In particular, they do not take into account whether resource processing time or queue space respectively is used for a useful purpose. For example, during an overload a link may process signalling messages relating to a service session which has been prematurely terminated due to an abandonment by the end-user. We therefore require a metric that gives a more end-user centric performance measure. This is provided by service session offer/completion rates, which give an overall indication of the grade-of-service received by end-users and an indication of the level of revenue earned by the network operator.

4.4.1 Scenario 1: Overload of SCP Only

Our first scenario addresses the operation of ACG under the assumption that when the SCP overloads the SS.7 links do not. This mirrors the analyses of previous researchers who have addressed the operation of the ACG control. Our goal in repeating this work is twofold: firstly to provide a degree of validation of the quasi-analytic and simulation models, secondly to serve as a reference for later scenarios in which interactions between SS.7 and IN controls are investigated.

We set the SCP processing capacity so that the SCP will overload at an offered load level that is insufficient to cause overload of any of the mesh or SCP links. In normal load conditions the traffic profile is as follows: 60% of service requests offered to the network are for ISUP Telephony, 25% are for IN Freephone and 15% are for IN Televoting. An equal mean number of requests for each service type arrives at each SSP in the network. The volume of Freephone and Televoting requests is such that the mean SCP load is 0.4 Erlangs during normal load conditions. During overload[40] the level of Televoting requests arriving at group A SSPs is increased so that the first-offered load to the SCP is 1.4 Erlangs. This increase occurs in a single step at time 600s and the subsequent decrease occurs in a single step at time 1200s.[41] End-user reattempt behaviour is included for all three service types.

Figure 4.11 shows the SCP processor utilisation and queue size obtained using the quasi-analytic and simulation models. Both processor utilisation graphs show the mean value of utilisation since the previous sample instant, with the sampling period ($10s$) being the same as the control interval used by ACG. The quasi-analytic model queue size graph shows mean values of queue size for each iteration interval, with the iteration interval length being the value of *T29* ($0.3s$). The simulation model queue size graphs shows the actual queue size at the sampling instant, a sampling period of $0.3s$ is again used.

In Figure 4.11 we see a large degree of oscillation of the SCP processor utilisation beyond the target range of (0.8, 0.9) Erlangs. This is an artefact of the use of a small number of tabulated values (thirteen) for gap interval calculation. Clearly, for this overload scenario, thirteen gap interval values cannot provide sufficiently fine-grained control of input traffic to keep the utilisation within the specified thresholds. We also observe a significant difference between the size of the SCP queue predicted by the quasi-analytic and simulation models: the queue periodically becomes saturated in the quasi-analytic model, but this only occurs at the start of the overload in the simulation model. As we discuss below, this is related to the difference in acceptance behaviour for the Televoting service in the two models.

[40] Throughout the remainder of the text we use the term 'overload' as a shorthand for the period of time that the offered load is increased. It is not meant to imply that any resources actually become overloaded

[41] A step input occurring at these times is the input load profile used for all simulation runs described in this chapter.

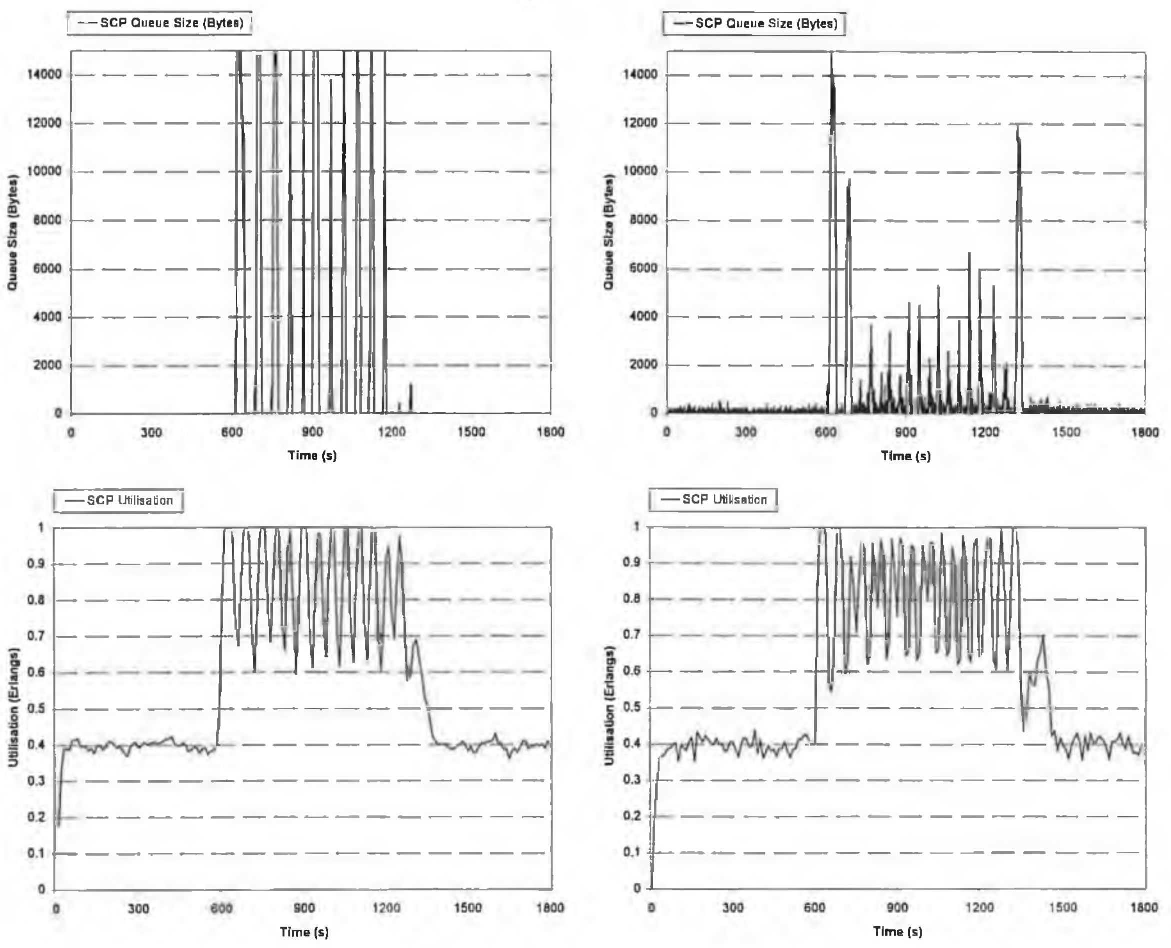


Figure 4.11: SCP utilisation and queue size. Left: quasi-analytic model. Right: simulation model.

Figure 4.12 shows offer and completion rates per second (calculated as moving averages with window $10s$ for the quasi-analytic model and mean values every $10s$ for the simulation model) for all service types. Since the SS.7 mesh links are not overloaded, there is no discernible impact on the completion rates of ISUP sessions during the period of SCP overload. The difference in the time for the ISUP offer and completion rates to converge is due to variations in the modelling of conversation delays between the two model types.

We see that, during the overload period, the rate of offered Televoting (and Freephone) requests increases significantly beyond the first-offered rate; this is due to end-user reattempt behaviour. Reattempts not only have the effect of intensifying the overload, they also result in it persisting for some time after the abatement of the increase in first-offered Televoting load. The volume of reattempts is larger in the simulation model than in that quasi-analytic model; this relates to the manner in which reattempt volumes are calculated (the quasi-analytic model is less accurate as an approximation method is used). The volume of reattempts impacts on acceptance rate behaviour for the two IN services (acceptance rates are not presented here but for Televoting and Freephone mirror closely the completion rates shown in Figure 4.12). For Televoting in particular the degree of oscillation in Televoting acceptance rates is higher in the quasi-analytic model, with a higher number of acceptances at the oscillation peaks. The latter increases the burstiness of the volume of messages being sent to the SCP, which is the primary cause of the periodic queue build-ups evident in Figure 4.11. During overload there is also a discernible decrease in the completion rates for the Freephone service, together with a corresponding slight increase in the offered rate due to reattempts. There are two factors that contribute to this: firstly ACG controls are put in place for all IN services, at all SSPs, secondly some Freephone messages will get discarded at the SCP queue.

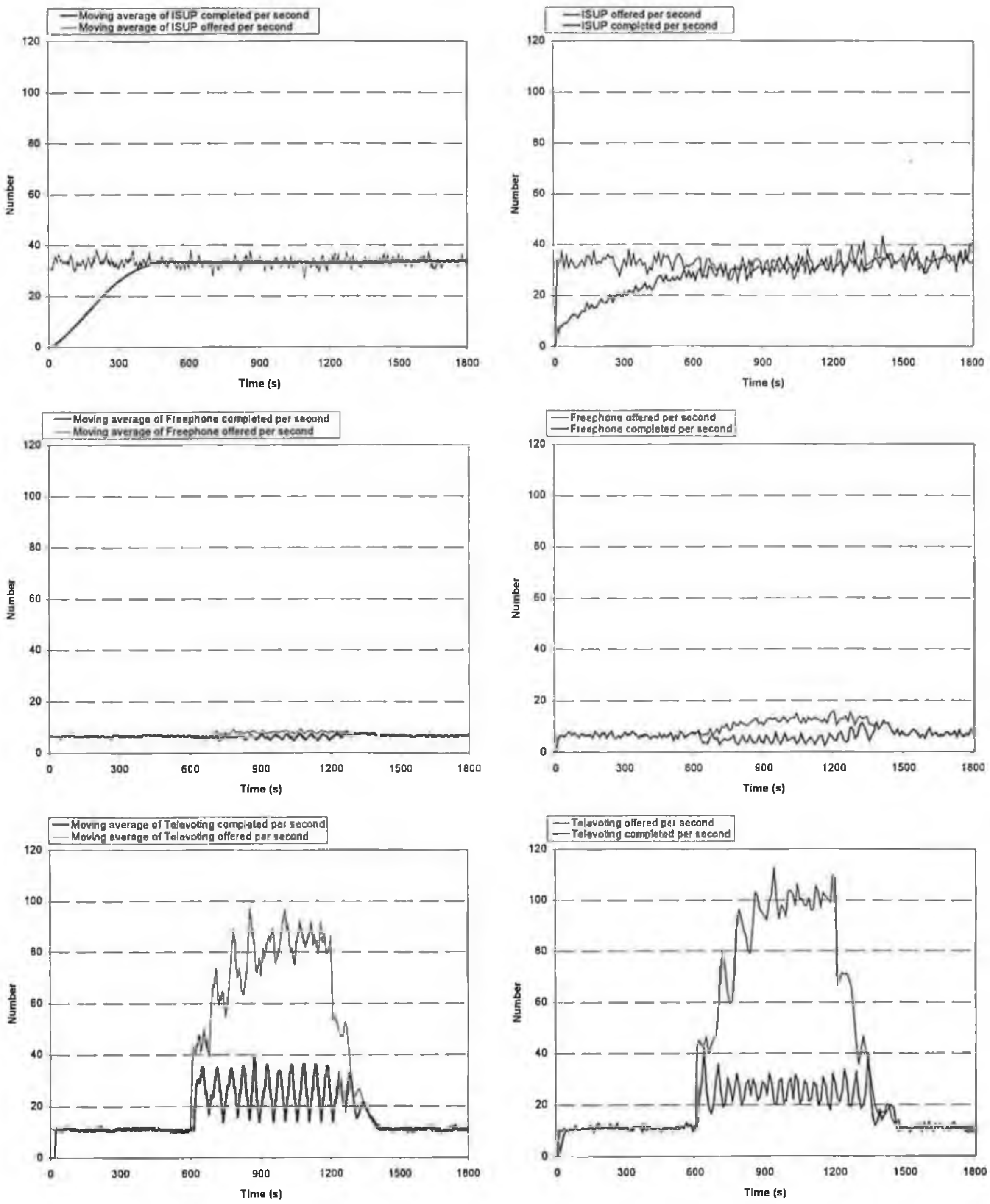


Figure 4.12: Service session offer/completion rates. Left: quasi-analytic model. Right: simulation model. Top: ISUP Telephony. Middle: IN Freephone. Bottom: IN Televoting.

The results presented for this scenario are in broad agreement with those presented in [Smith, 1995; Northcote and Smith, 1998], where the operation of the ACG control is investigated in great detail. Our results point towards many of the same conclusions of these works, for example: static table-driven controls can not provide suitable fine-grained control in all circumstances, service types not contributing directly to the overload suffer, as do SSPs not contributing directly to the overload. In chapter 5 we present a more comprehensive analysis, comparing the operation of ACG with two more complex, adaptive and revenue-maximising IN load control strategies.

4.4.2 Scenario 2: ISUP Overload of Mesh Links

The second scenario we describe addresses the operation of the SS.7 load controls in isolation. As with scenario 1 we demonstrate behaviour previously analysed by other researchers, although in the context of a more realistic network model. Firstly, we illustrate how the ISUP load throttles lead to oscillations of link utilisation and queue size; secondly, the negative impact of REL message reattempts on call completion rates is demonstrated. In addition, we show that poor performance of the SS.7 load control can lead to the propagation of overload and consequent undesired load throttling.

In scenario 2 the normal load offered to the network is supplied equally by the group A and group B SSPs and is equivalent to a loading of 0.4 Erlangs on each of the mesh signalling links. This load is generated by requests for the ISUP Telephony service only (there are no requests for Televoting or Freephone). Overload occurs due to an increase in the volume of requests arriving at the group A SSPs. Once increased, the volume of signalling load is equivalent to a first-offered load of 1.4 Erlangs on those mesh signalling links that transmit signalling messages from group A to group B (links 0, 2, 4 and 6).

Figure 4.13 shows the utilisation and queue size for link 0 for the full duration of the simulation run, obtained using both model types. Figure 4.14 (page 96) shows 'close-ups' of the link 0 utilisation and queue size. From Figure 4.13 we see that once the load offered to the link increases, the queue size quickly exceeds the overload onset threshold (1400 bytes), causing TFC procedure invocation at STP1. This causes STP1 to send TFCs for group B SSPs to the group A SSPs, which then commence load throttling. Because of the speed of overload onset all group A SSPs receive TFC messages at approximately the same time, leading to a high degree of synchronicity in their throttling behaviour. The links thus become starved of messages, TFC generation stops causing the SSPs to subsequently increase their accepted load, again with a high degree of synchronicity. This synchronicity causes oscillations in link utilisation and queue size, as evident from the figure.

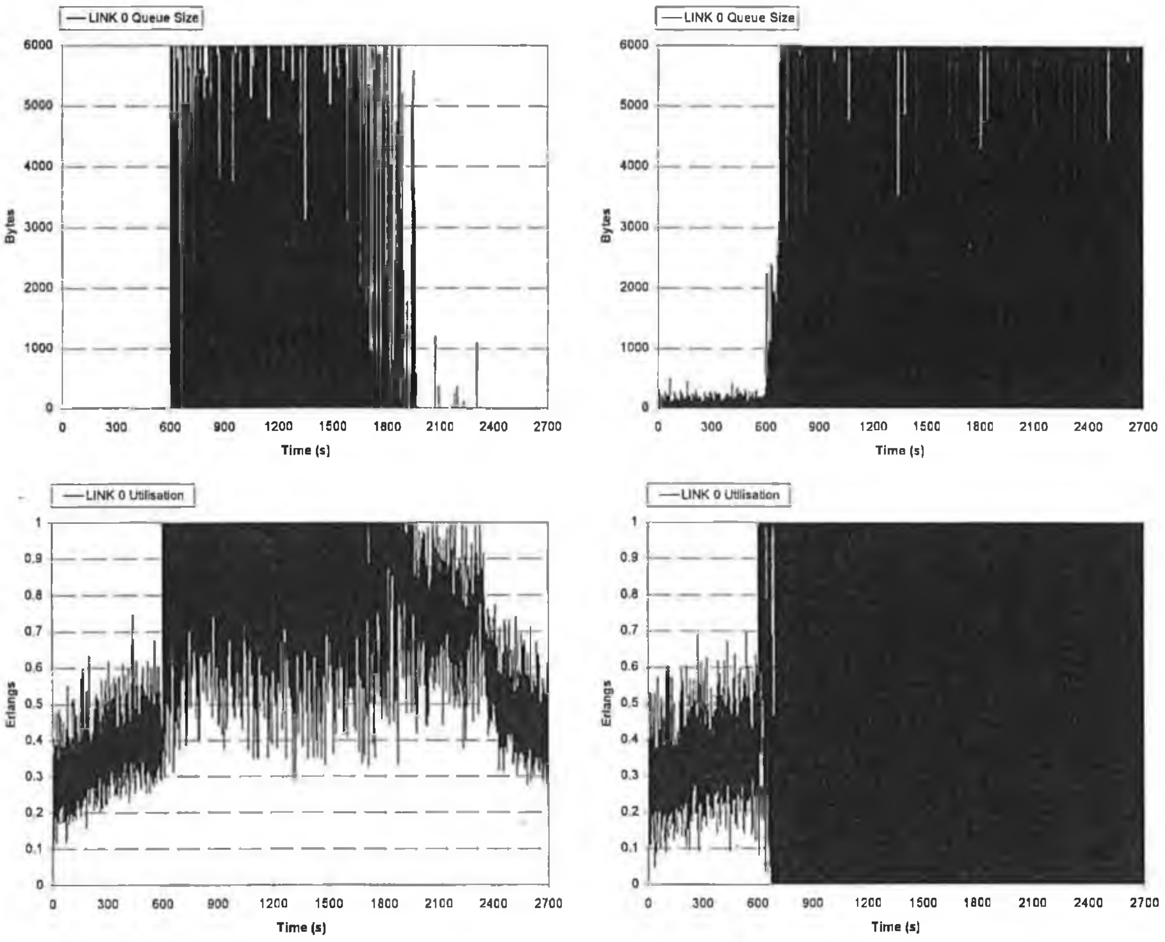


Figure 4.13: Link 0 utilisation and queue size. Left: quasi-analytic model duration. Right: simulation model.

The quasi analytic results presented in Figure 4.13 show the persistence of link overload beyond the time at which the volume of first-offered load decreases, due to the end-user reattempt behaviour. For the simulation model end-user reattempt behaviour is omitted for this scenario, so that the effect of ISUP REL reattempts on overload persistence and completion rate degradation can be quantified in isolation. Results for the simulation model are similar to those obtained using the quasi-analytic model to the degree that they demonstrate the oscillatory behaviour of the controls. However with the simulation model, the observed degree of synchronisation of the SSPs in Figure 4.14 is much greater then observed for the quasi-analytic model, with the link utilisation actually dropping to zero for the much of the time during overload.

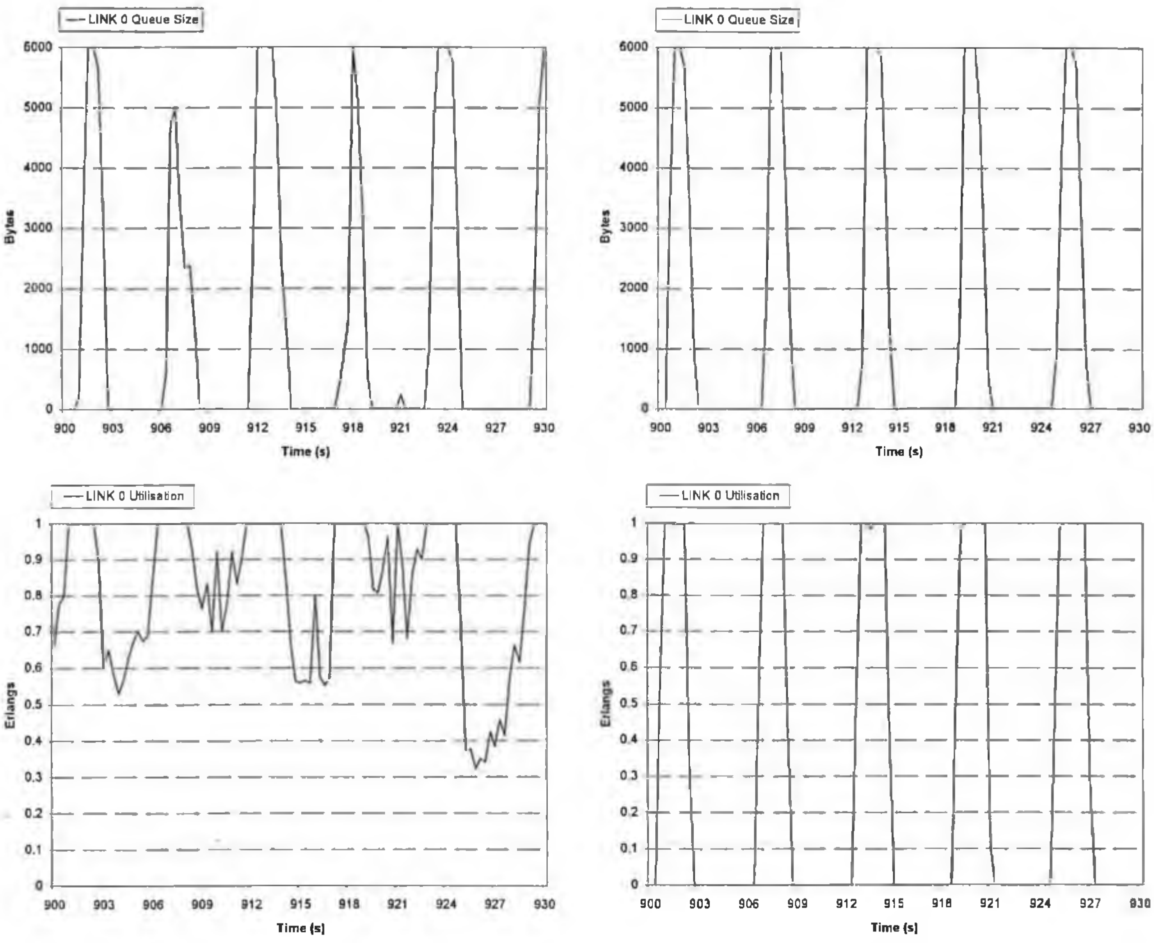


Figure 4.14: Link 0 utilisation and queue size. Partial simulation run duration. Left: quasi-analytic model. Right: simulation model.

The most significant difference between the results obtained using the quasi-analytic and simulation models is that the simulation model shows that the overload condition persists long after the level of offered load decreases. This is due to the REL reattempt behaviour. When ISUP messages are throttled or discarded at link queues, the originating ISUP entity sends REL/RSC messages under the control of timers (*cf.* §2.4.1.2 for details of this procedure). As shown by [Smith, 1994b], this leads to an avalanche of REL messages offered to the links. Reattempting RELs and RSCs displace other messages types, so link overload persists and the majority of processed messages are REL/RSCs. This is confirmed by Figure 4.15, which shows that approximately 90% of messages processed by the link are REL/RSCs, increasing from a value of approximately 20% during normal conditions.

In our model there are no limits on the number of REL/RSC reattempts that can be generated by ISUP entities. In real networks, operators are likely to either statically restrict the number of reattempts that can be made or employ a dynamic restriction algorithm such as that proposed by [Smith, 1994b]. Therefore our model is overly pessimistic, as indefinite persistence of overload would not happen in a real network. However the results do demonstrate the importance of ensuring that the behaviour of signalling user parts or higher level applications do not exacerbate overload situations.

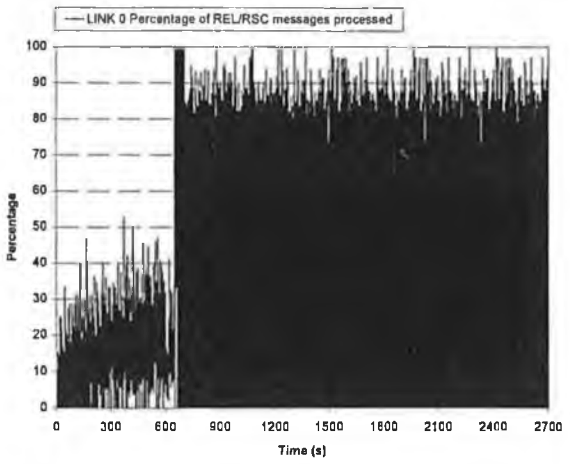


Figure 4.15: Simulation model. Link 0 percentage of link utilisation relating to REL messages.

We now address the potential for overload of mesh links transmitting traffic in one direction to cause propagation of overload to those mesh links transmitting in the opposite direction. In this scenario the signalling messages carried by a link relate to ISUP Telephony sessions and generally trigger the destination SSP to send one or more messages in response. If there is a sufficient volume of response messages generated during an overload then links 1, 3, 5 and 7 may also become overloaded. This leads to the invocation of the TFC procedure at STPs 2/4 and the throttling of load by the group B SSPs. This is despite the fact that the group B SSPs are not the source of the overload traffic.

Figure 4.16 shows the utilisation and queue sizes for link 1 obtained using the quasi-analytic and simulation models. These graphs show that link 1 does indeed become overloaded and the utilisation and queue sizes oscillate subsequent to the group B ISUP load throttles being invoked. It can be seen that the onset of overload happens some time after the onset of overload of link 0. The delay is related to the cumulative effect of ringing and conversation times, which delay the generation of response messages. Again the overload persistence effect of REL reattempts is evident from the results obtained using the simulation model.

The results discussed above show that the oscillatory behaviour of the load controls, together with the REL reattempting behaviour of ISUP, means that mesh links are under-utilised during overload and that those messages transmitted may be predominantly reattempting REL/RSCs. We now address the impact this has on the session completion rates for end-users. Figure 4.17 shows the rates per second of offered, started and completed ISUP sessions as obtained using both model types.

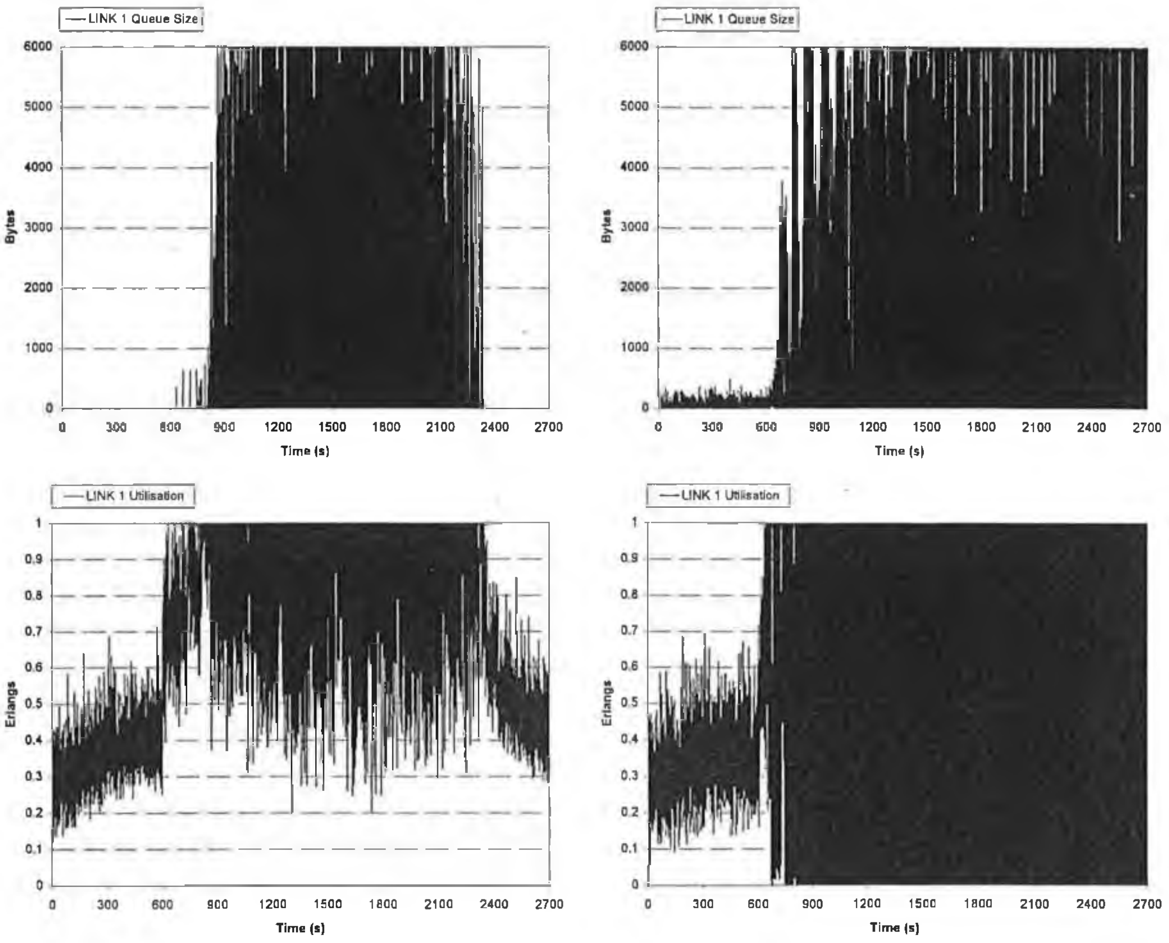


Figure 4.16: Link 1 utilisation and queue size. Left: quasi-analytic model. Right: simulation model.

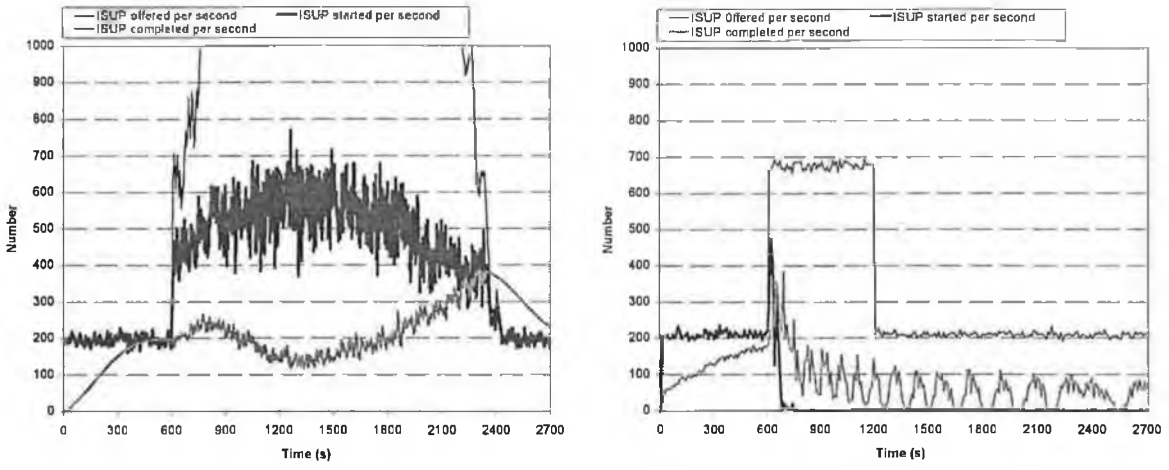


Figure 4.17: Moving average of session offer/start/completion rates. Left: quasi-analytic model. Right: simulation model.

For the quasi-analytic model we observe that when overload commences end-user reattempts cause the offered rate to increase significantly before eventually abating near the end of the run. Invocation of the ISUP throttle means that the rate of started sessions is less than the offered rate. More significantly the completion rate gradually reduces to a low of only approximately 25% of the started rate, before gradually recovering when offered load abates.

In the simulation model end-user reattempts are omitted, however the rate at which sessions are started rapidly goes to and remains at zero. This is due to the ISUP load throttles all staying at either level 1 or 2 (in both cases IAMs are throttled) but never reaching level 0 (where all message types are accepted). When one of the ISUP throttles changes from level 2 to level 1 a large volume of (predominantly REL/RSC) messages is accepted. Upon reaching the links, these messages quickly trigger the sending of TFC messages and CIs are received by the ISUP entities before the throttle can change to level 0 and accept IAMs. Therefore the increase in the volume of ISUP messages (caused by reattempting RELs/RSCs) that is offered to the network has a hugely adverse effect on the rate of session acceptances.

This scenario has clearly demonstrated the detrimental effects of ISUP REL/RSC reattempts, however as alluded to above, operators are likely to take measures to avoid or minimise the impact of REL/RSC avalanches [Smith, 1994b]. Even if automatic REL avalanche prevention procedures are not deployed, network management systems are likely to raise alarms that would ensure that overloads do not persist indefinitely in the manner depicted here. For this reason and in order to facilitate quantitative analysis of the operation of the core SS.7/IN load control operation in various situations, we omit the REL reattempting behaviour in the simulation runs used for the remaining scenarios in this chapter. Therefore, it is important to note that the performance of the controls as described below is likely to be worse in a real network in which REL reattempt behaviour is not properly controlled.

4.4.3 Scenario 3: ISUP/IN Overload of Mesh Links

In scenario 3 we address the operation of the SS.7 load controls in a network supporting both ISUP and IN services. Scenario 2 illustrated the poor operation of the controls in terms of oscillatory behaviour and completion rates; here we focus here on the unfairness of the controls in their treatment of different user parts and service types. We first demonstrate the desirability of including a load throttle in SCCP and then investigate the operation of the SCCP load throttle specified in §4.1.3.1.

For this scenario we assume that GTT is not in operation at STP2 and STP4, hence SSP SCCP entities perform GTT and messages are addressed directly to the SCP. We assume that SCP links never overload, therefore the presence or absence of GTT at the STPs has no effect on the load throttling behaviour.[42] In normal load conditions the mesh links are loaded at 0.4 Erlangs, with an equal load contribution from sessions of ISUP Telephony and Televoting (there are no Freephone sessions). During overload the first-offered load to the mesh links is 1.4 Erlangs, with the increase relating to IN Televoting requests arriving at the group A SSPs only. To

[42] If GTT were present at the STPs and both mesh/SCP links overload, IN message streams from SSPs to the SCP would be subject to SCCP throttling at both the SSP and the STP.

enable a quantitative comparison of the fairness with which user parts and services are throttled we omit from the models end-user reattempt behaviour and ISUP REL/RSC reattempts.

We first address the case where there is no load throttle incorporated into SCCP, which means that CIs arriving at SSP SCCP entities are ignored. Figure 4.18 shows the utilisation and queue size for link 0, obtained using both the quasi-analytic and simulation models. In the absence of reattempts, the utilisation of the link returns to normal promptly after the abatement of high offered load. An obvious difference between the two models is that queue is shown to be frequently fully occupied in the quasi-analytic model whereas it never is in the simulation model. This is primarily related to the different ISUP Telephony user behaviour models employed: with the simulation model there are busy/unanswered calls and longer mean conversation times, so there are not so many messages processed during the overload period.

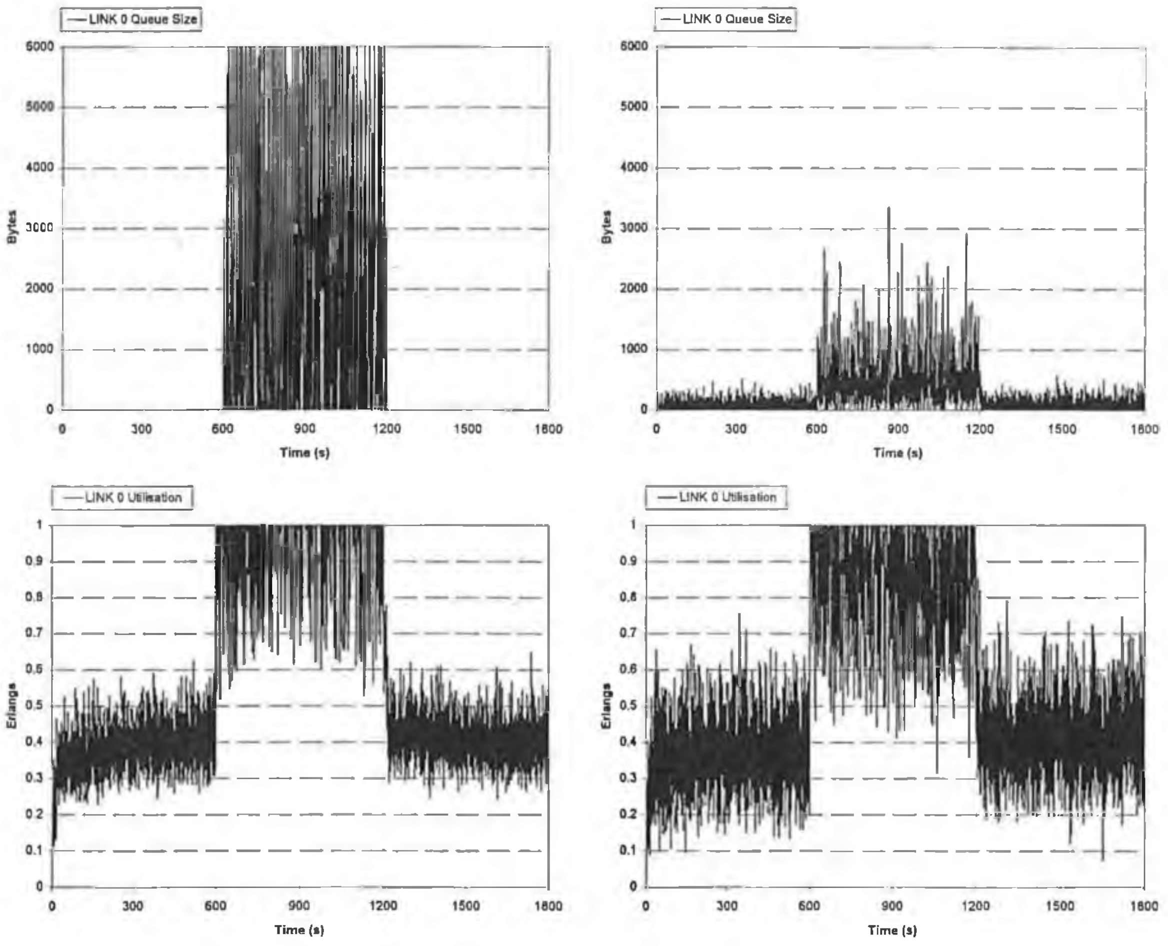


Figure 4.18: Link 0 utilisation and queue size. No SCCP controls. Left: quasi-analytic model. Right: simulation model.

Figure 4.19 shows the service session offer and completion rates obtained using the quasi-analytic and simulation models. Due to the absence of SCCP load throttles only ISUP sessions are throttled during overload, so ISUP completion rates are significantly less than the offered rates. In contrast, Televoting sessions remain largely unaffected. These results show that, without SCCP controls, ISUP entities are forced to throttle an unfairly large proportion of their offered load. This would also be the case if, for example, an increase in the volume of Televoting requests were solely responsible for the overload. Indeed, in such circumstances Televoting would be contributing the majority of the traffic carried by the links so the queues would remain continually full and a large number of messages would be discarded and sessions prematurely terminated. Therefore, some form of SCCP load throttling is not only desirable from a fairness viewpoint but is necessary in order to protect against very severe degradation in completion rates.

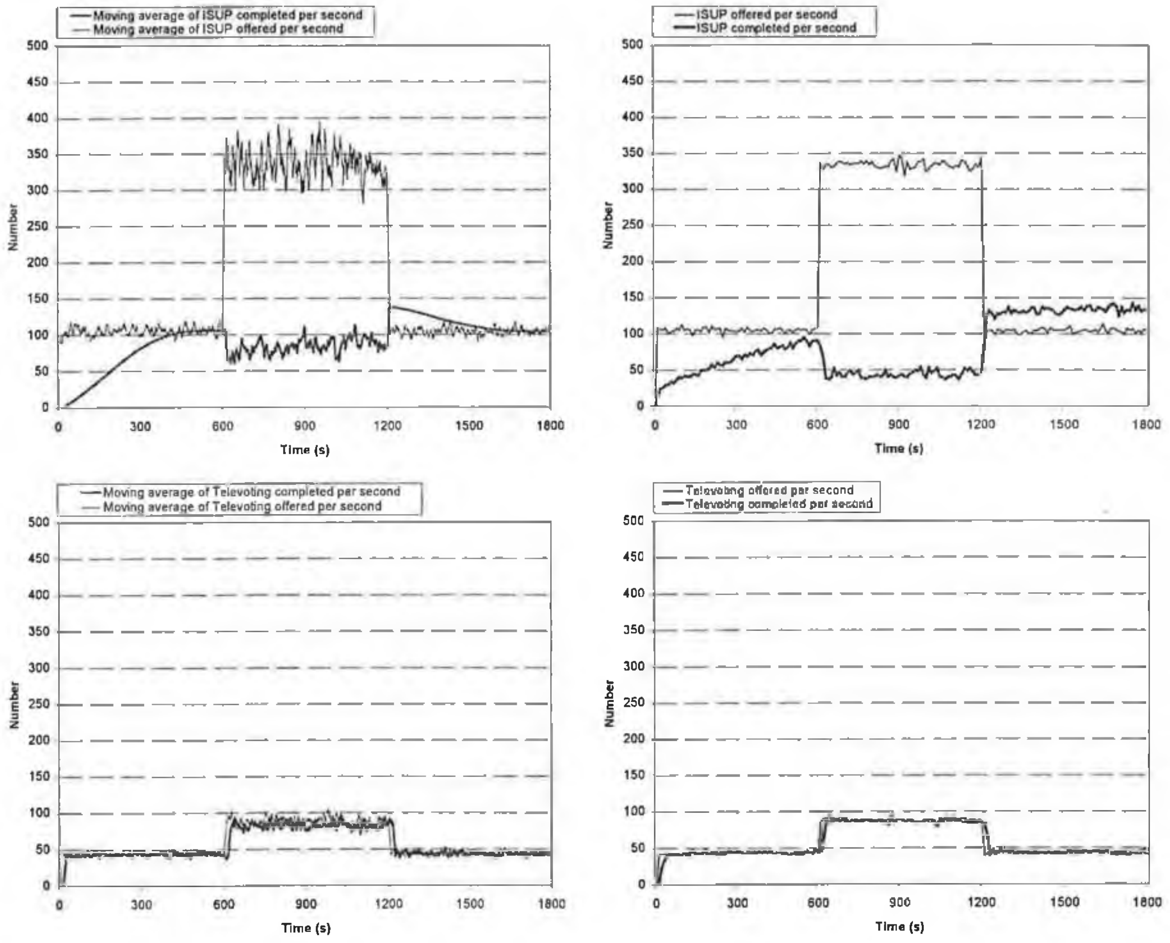


Figure 4.19: Moving average of session offer/completion rates. No SCCP controls. Left: quasi-analytic model. Right: simulation model.

Figure 4.20 shows the offer and completion rates as obtained from the quasi-analytic and simulation models for the case where the load throttle is incorporated into the SCCP. In this instance we see that sessions of both ISUP and Televoting are throttled, but ISUP not as severely as previously. Approximately 62% of the ISUP offered requests result in completed sessions during overload whilst the corresponding value for Televoting is approximately 55%.[43] Thus the SCCP and ISUP throttles working in tandem do not result in completely equitable treatment of the two service types.

[43] These values were obtained from the quasi-analytic model results.

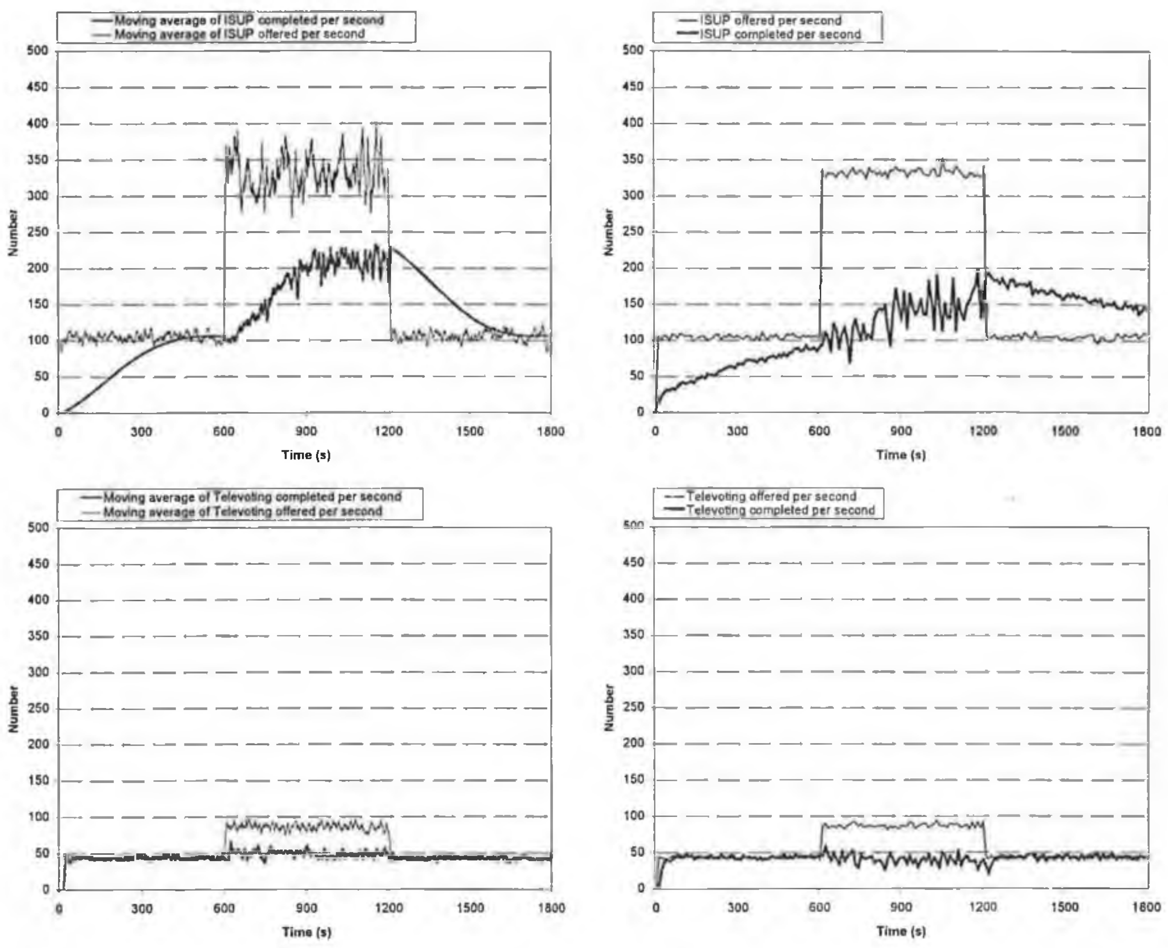


Figure 4.20: Quasi-analytic model. Moving average of session offer/completion rates. SCCP controls present. Left: quasi-analytic model. Right: simulation model.

A major contributing factor to the smaller completion rate percentage for Televoting is that sessions of this service type involve the transfer of seven signalling messages between the SSP and the SCP, whereas for ISUP only five messages are passed between the two SSPs. There is thus a greater probability that Televoting sessions are prematurely terminated due to a message being throttled. Therefore, even though ISUP and SCCP throttles operate fairly in terms of the proportions of messages they throttle, they fail to take into account that sessions of different service types involve varying numbers of messages. In addition, they do not take into account message sizes and therefore do not treat services fairly in terms of their contribution to the load they place on network resources [Zepf and Rufa, 1994; McMillan and Rumsewicz, 1996].[44]

An additional complication in regard to the fairness of the combined operation of the ISUP and SCCP load throttles is that successful IN sessions often lead to the set-up of a 'follow-on' ISUP Telephony session. A good example is the IN Freephone service, which involves a number

[44] In this discussion we ignore the important consideration that during overloads a network operator may not want to have services treated in the manner we describe as 'fairly.' On the contrary, services may generate different levels of revenue or may be seen as more important from the end-user point-of-view and the operator may want to afford these higher priority during overload. We return to this point in chapters 5 and 6.

translation and the set-up of a call to the translated number. If, as we assume here, the request for the ISUP Telephony call is subject to the ISUP load throttle as normal then, during overload, the end-user's service request must, in effect, pass both the SCCP and the ISUP throttles. Thus there is a more complex relationship between the relative rates then for the ISUP/Telephony.

We now investigate the case where the network supports only ISUP Telephony and IN Freephone services, with requests of both types resulting in equal contributions to link load during normal conditions. Overload of mesh links results from an increase in the rate of arrivals of requests for IN Freephone only at the group A SSPs. Figure 4.21 shows the offer and completion rates for both service types, obtained using the quasi-analytic and simulation models.[45] In these graphs the increase in the number of arriving ISUP requests corresponds to the number of successfully completed Freephone sessions. We see that both Freephone and ISUP requests are throttled, so the actual completion rate for Freephone end-users is less than that indicated by the Freephone graph.

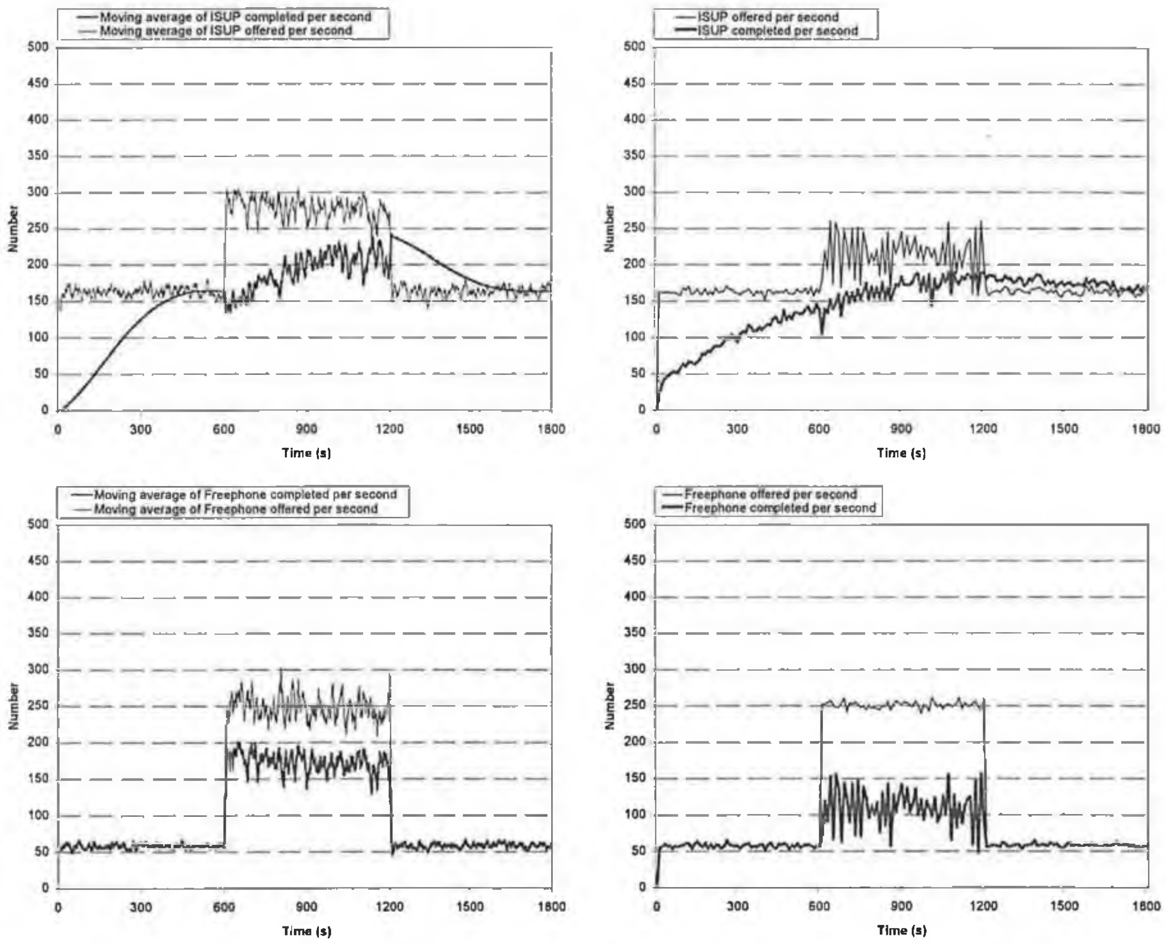


Figure 4.21: Moving average of session offer/completion rates. SCCP controls present. Left: quasi-analytic model. Right: simulation model.

[45] The noticeable difference in Freephone completion rates is due to the different ISUP user behaviour models employed in the two model types. The simulation model includes busy/unanswered calls and longer conversation delays, so there are less ISUP messages transmitted during the overload, with the results that proportionally more Freephone messages are throttled then in the quasi-analytic model.

4.4.4 Scenario 4: Overload of Forward SCP Links Only

In this scenario we address overload situations in which forward SCP links, but neither mesh links nor the SCP, initially overload. Forward SCP links carry only IN traffic, so the importance of deploying SCCP load throttles becomes clearly evident in this context. We first demonstrate that poor performance ensues if an SCCP load throttle is not deployed and then show that deployment of the SCCP load throttle does not actually result in performance improvement.

As indicated at the start of §4.4 for this and the remaining three scenarios we present results obtained using the quasi-analytic model only. In normal conditions the ratios of service arrival volumes are set as follows: ISUP 60% : Freephone 25% : Televoting 15%. Overload is caused by an increase in the arrival rate of Televoting requests at group A SSPs, to a level corresponding to a first-offered load of 1.4 Erlangs for each of the two forward SCP links. For this scenario SCP capacity is set to a value sufficient to ensure the SCP does not overload. In addition we also assume the reverse SCP links will not overload as a result of response traffic generated by the SCP.[46]

The first results we present relate to the case where there is no GTT in place at STP2 and STP4, the SCCP load throttle is absent and end-user reattempts are not modelled. Figure 4.22 shows the link utilisation and queue size for link 0 and link 8, obtained using the quasi-analytic model. These graphs show that mesh links do not overload but the SCP links do; this is because the group B SSPs, which provide half of the overload traffic, do not utilise the mesh links for communication with the SCP. The absence of SCCP load throttling at STP2 means that the load on link 8 is not decreased, so the link remains overloaded and a significant number of messages are discarded.

Figure 4.23 shows the service session offer and completion rates for all three service types. No IN service requests are throttled, nevertheless many accepted sessions will be prematurely terminated due to discards at the SCP links. This can be clearly seen for the Televoting service, for which only approximately 60% of the offered sessions complete successfully. Freephone sessions are also affected in the same manner, though the smaller number of messages in a Freephone session means that the completion percentage is significantly higher than that for Televoting. The drop in Freephone completion rates during overload also manifests itself as a reduction in ISUP offer/completion rates due to the consequent decrease in the number of generated follow-on ISUP sessions.

[46] The scenario where reverse SCP links do overload is addressed in §4.4.6.

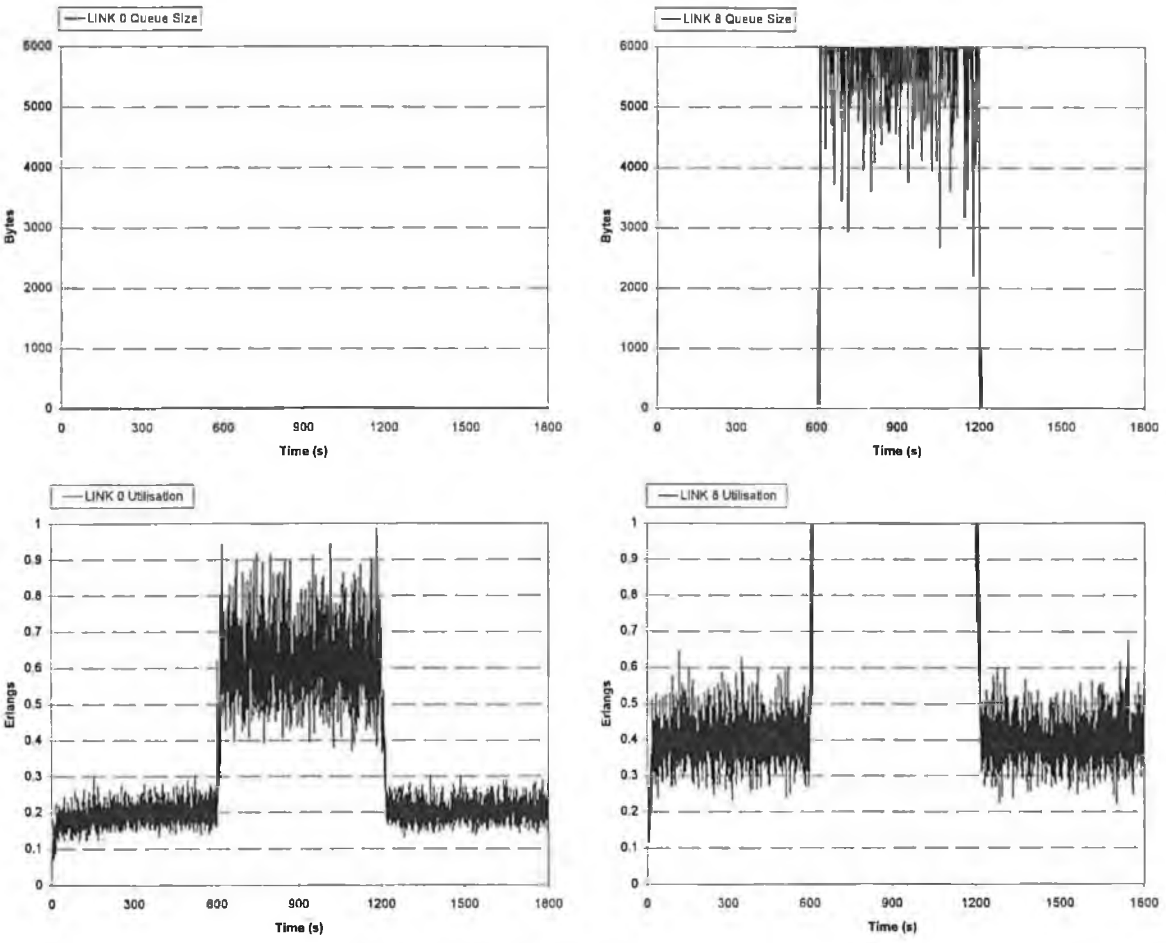


Figure 4.22: **Quasi-analytic model. Link utilisation and queue size. No reattempts, no SCCP load throttling. Left: Link 0. Right: Link 8.**

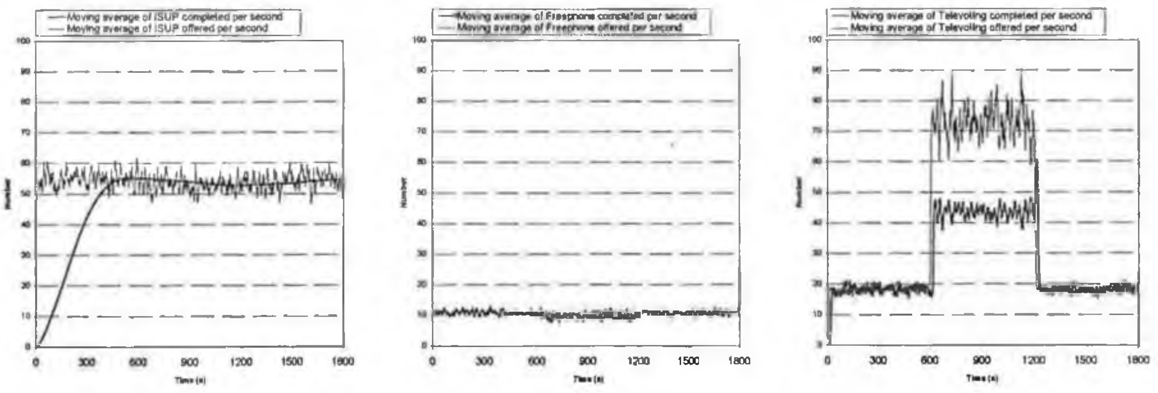


Figure 4.23: **Quasi-analytic model. Moving average of service offer/completion rates. No reattempts, no SCCP load throttling. Left: ISUP. Centre: IN Freephone. Right: IN Televoting.**

We now investigate the impact of end-user reattempts. The high level of Televoting sessions terminated due to discards at the SCP link queues will result in a large volume of reattempts, increasing greatly the load offered to the network. Therefore, there will be potential for overload propagation to the mesh links and completion rate degradation for both non-IN and IN service types. Figure 4.24 shows the utilisation and queue size for link 0 and link 8 for this case; it shows that reattempts do indeed lead to propagation of overload to the mesh links. We see that

Televoting reattempts quickly cause the forward mesh links (link 0) to overload and, because of the lack of an SCCP throttle, keep the link queues saturated for the duration of the increase in first-offered load. Once the first-offered load abates, the proportion of Televoting traffic decreases and the oscillatory effects of the operation of the ISUP load throttles become evident. Overload and queue saturation of mesh links mean that significant numbers of both Televoting and ISUP messages continue to be discarded.

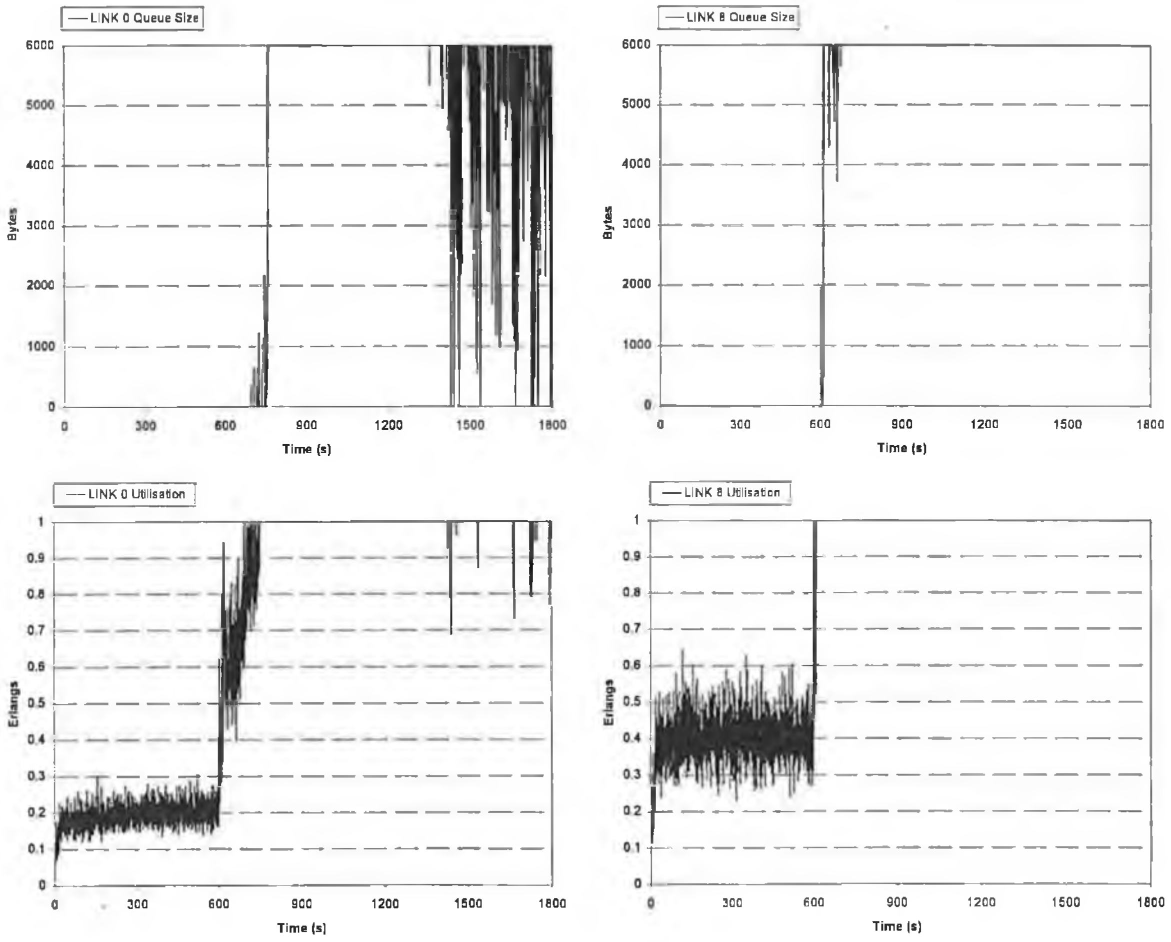


Figure 4.24: Quasi-analytic model. Link utilisation and queue size. With reattempts, no SCCP load throttling. Left: Link 0. Right: Link 8.

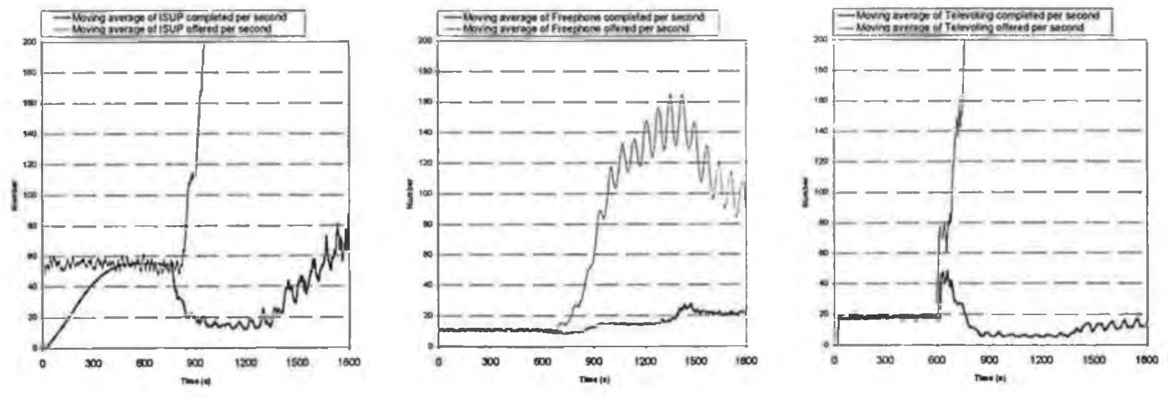


Figure 4.25: Quasi-analytic model. Moving average of service offer/completion rates. With reattempts, no SCCP load control. Left: ISUP Telephony. Right: IN Televoting.

Figure 4.25 shows the session offer and completion rates for the three services; it shows that all services are severely affected by the reattempt-triggered overloads. These results provide further evidence of the necessity of deploying some form of load throttling in SCCP entities.

We now investigate whether the inclusion of the SCCP load avoids the severe performance degradation observed in its absence. We assume that GTT is in operation at STP2 and STP4, so that link 8 overload will trigger SCCP throttling at these nodes. We employ the same load profile as previously and model end-user reattempt behaviour. Figure 4.26 shows the link utilisation and queue size for link 0 and link 8. We see that, upon overload onset, the operation of the SCCP load throttle leads to oscillation of link 8 utilisation and queue size, caused by the on-off throttling of INAP messages at STP2. This oscillatory behaviour is expected as the SCCP load throttle is based on the ISUP throttle. Under-utilisation of link 8 (and link 10) means that a greater number of INAP messages are discarded then in the previous case. The volume of end-user reattempts increases accordingly, further exacerbating the overload and degrading IN session completion rates. In addition, SCCP load throttling at the STPs does not protect the mesh links from overload, so ISUP completion rates are again effected (though not as severely as previously). The impact on completion rates is shown in Figure 4.27; clearly the SCCP load throttle has (in this scenario at least) little positive impact on network performance.

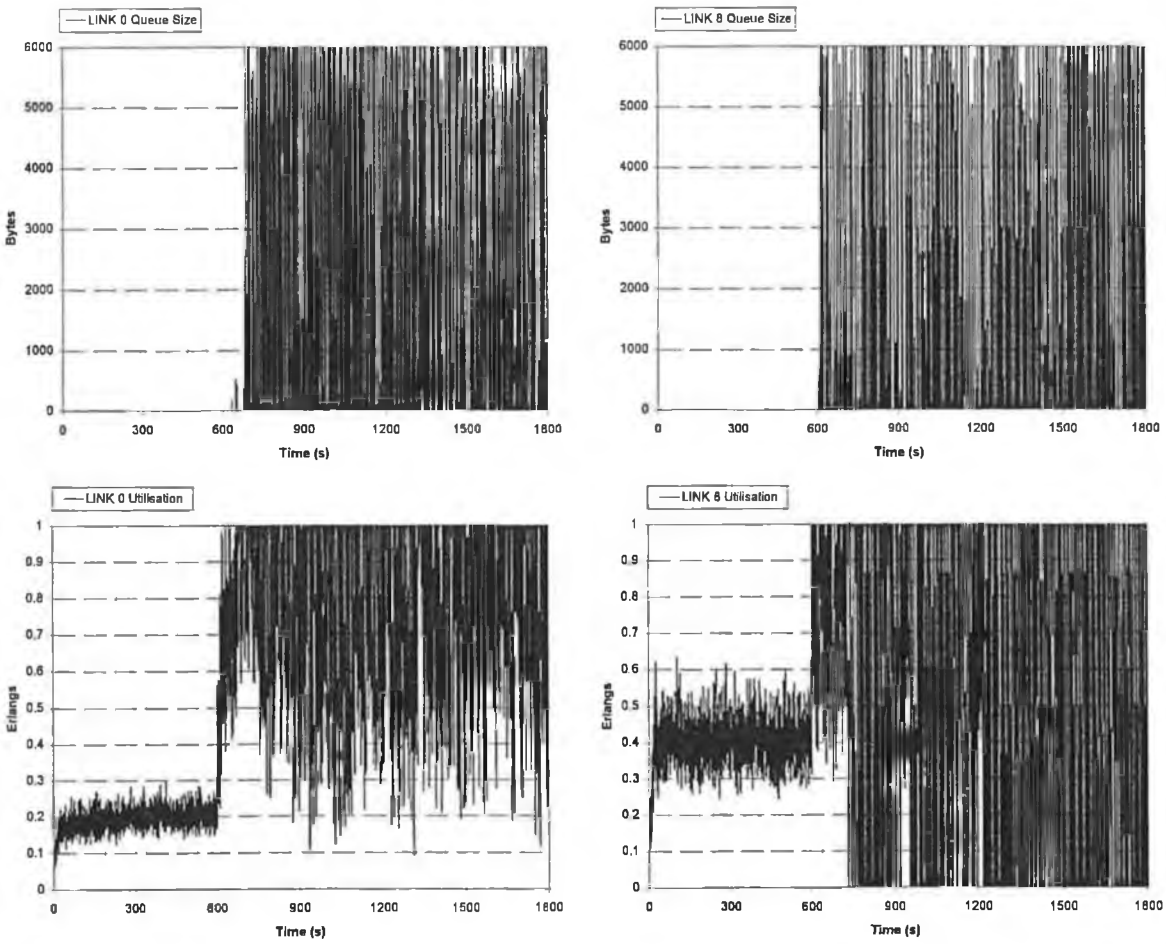


Figure 4.26: Quasi-analytic model. Link utilisation and queue size. With reattempts, SCCP load throttling and GTT at STPs 2/4. Left: Link 0. Right: Link 8.

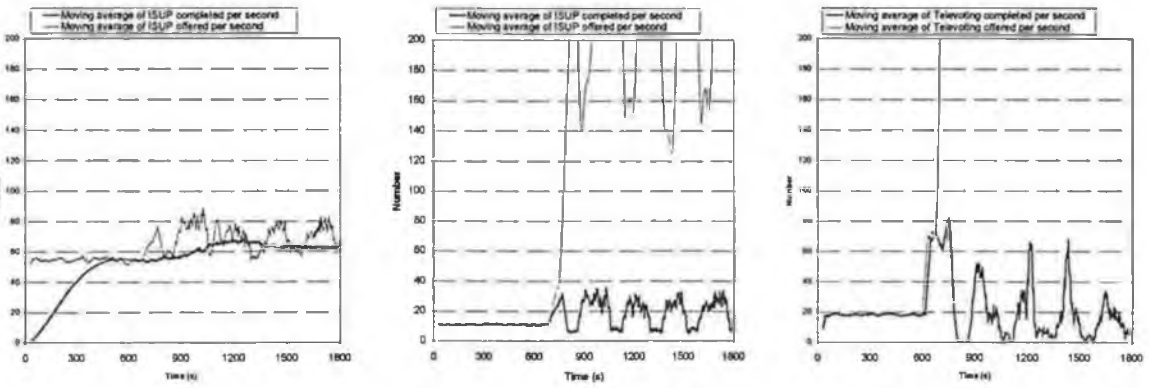


Figure 4.27: Quasi-analytic model. Moving average of service offer/completion rates. With reattempts, SCCP load throttling and GTT at STPs 2/4. Left: ISUP Telephony. Right: IN Televoting.

4.4.5 Scenario 5: Overload of Forward SCP Links and SCP

For scenario 5 we examine the potential for simultaneous overload of SCP forward links and the SCP itself. This would be a possibility in the case where the cumulative rate at which SCP links can process IN messages exceeds the rate at which they can be processed by the SCP. We employ the same load profile used previously, but set the SCP capacity so that the increase in the volume of Televoting requests is sufficient to overload both it and the forward SCP links.

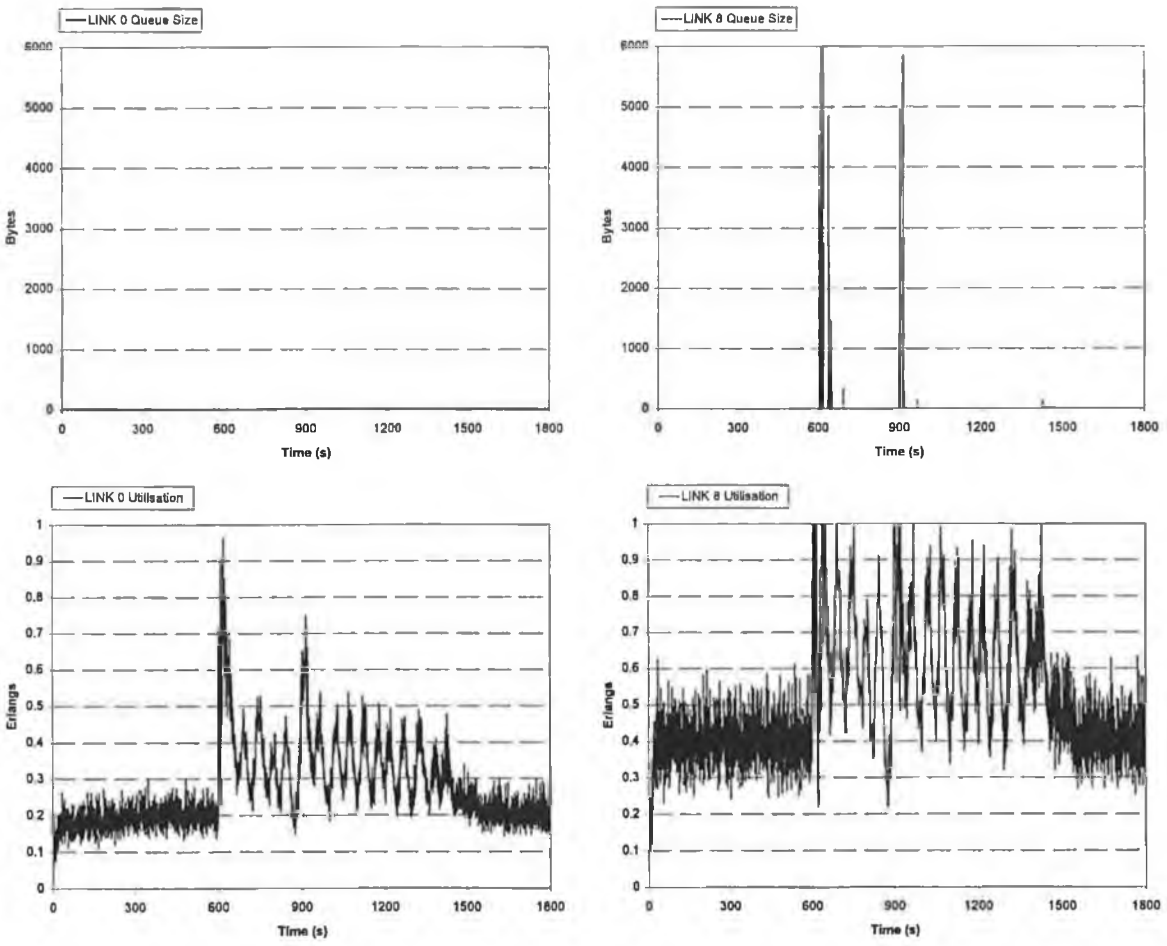


Figure 4.28: Quasi-analytic model. Link utilisation and queue size. Left: Link 0. Right: Link 8.

Again we assume that the reverse SCP links do not overload. GTT is in-place at STP2 and STP4, the SCCP load throttle is operational and end-user reattempt behaviour is modelled.

Figure 4.28 above shows the utilisation and queue size, obtained using the quasi-analytic model, for link 0 and link 8. We see that while mesh link 0 does not overload, the link 8 queue size quickly surpasses the overload onset threshold once the volume of Televoting requests increases. However, link 8 recovers after approximately 50s and only briefly overloads again during the remainder of the overload situation. This is due to the onset of overload at the SCP (illustrated by Figure 4.29), which causes the activation of the ACG throttle at the SSPs. In reducing the load offered to the SCP, ACG also reduces the load offered to the forward SCP links. Indeed ACG protects the links more effectively than did the SCCP load throttle investigated previously: the link queues do not saturate, queuing delays are reduced and relatively few messages are discarded. Figure 4.30 shows the offer and completion for the three services; these graphs are broadly similar in form to those ofFigure 4.12, page 93, which relate to the scenario where only the SCP overloads.

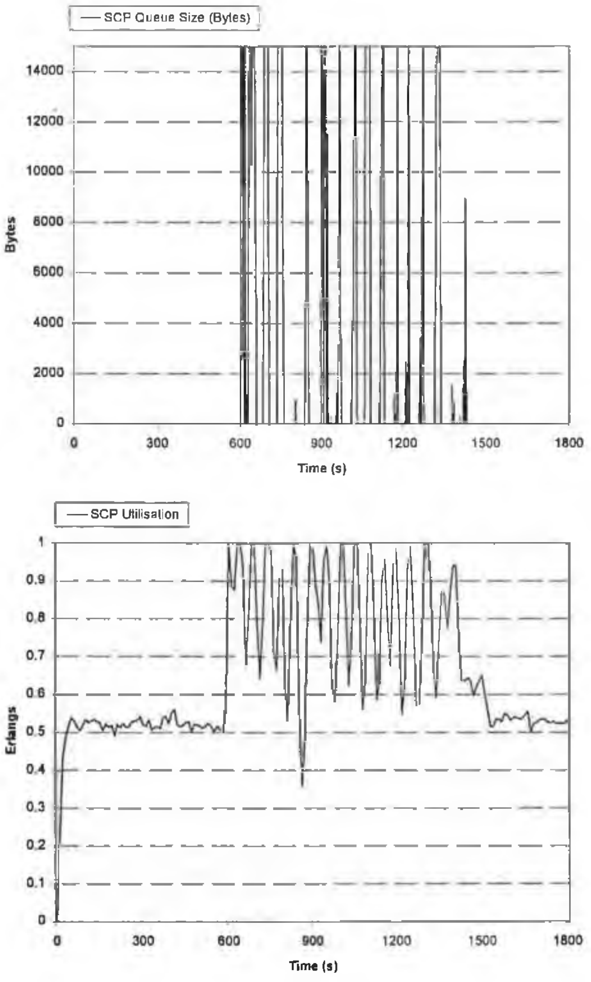


Figure 4.29: Quasi-analytic model. SCP utilisation and queue size.

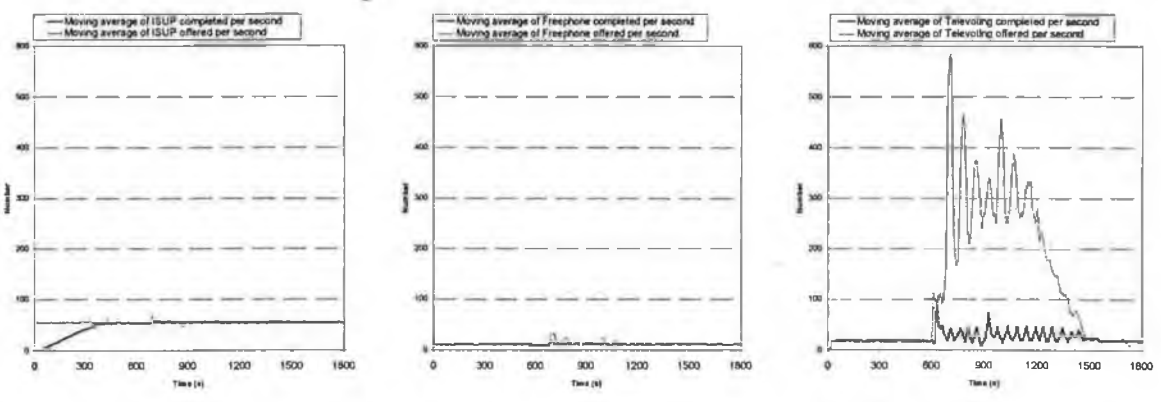


Figure 4.30: Quasi-analytic model. Moving average of service offer/completion rates. Left: ISUP Telephony. Centre: IN Freephone. Right: IN Televoting.

4.4.6 Scenario 6: Overload of Reverse SCP Links

Reverse SCP links are those links that the SCP uses to send response messages to its STP pair for forwarding to SSPs. Like other links, reverse SCP links are susceptible to overload, but there are two additional factors that indicate that they may overload either before or instead of forward SCP or mesh links. The first is that SCP response messages are often longer than the corresponding query messages; therefore, an equal number of messages can represent a greater load offered to reverse links then to the forward links. The second factor is that IN service sessions may be unbalanced in terms of the number of messages sent from the SSP to the SCP compared to those sent in the opposite direction. This is the case for our Televoting service, which involves three messages sent from SSP to SCP, but four in the opposite direction. We note that from the point of view of overall performance, overload of a reverse link (either SCP or mesh) is particularly undesirable, since it leads to the throttling or discarding of messages generated as a result of significant processing at the SCP.

In this scenario we employ a load profile similar to that used for the previous two scenarios but with the volume of first-offered load during overload being set such that reverse SCP links, but neither forward SCP links, the SCP, nor mesh links, initially overload. We assume that all response messages generated by the SCP are of length 120 Bytes (as opposed to 100 Bytes for query messages arriving at the SCP). GTT is operational at STP2 and STP4 and load throttles are activated at all SCCP entities. The latter is assumed to include the SCP SCCP entity, whose throttle will be invoked by CIs triggered by the arrival of SCP-generated response messages arriving at an overloaded reverse link.

Figure 4.31 shows the utilisation and queue size for a reverse SCP link (link 9) and a reverse mesh link (link 1). Figure 4.32 shows the SCP utilisation obtained using the quasi-analytic model; we see that the SCP does not overload at any time. Although the SCP itself does not overload the reverse SCP link does, causing the SCP SCCP entity to throttle SCP response messages. We again observe the oscillatory nature of the SCCP load throttle, which results in throttled and discarded response messages and overload persistence due to end-user reattempts. The load offered to link 1 is not sufficient to overload it for a sustained period; this is due both to a portion of the reverse SCP link load being destined for the group B SSPs and the remaining portion, destined for the group A SSPs, being dispersed over four (rather then two) links.

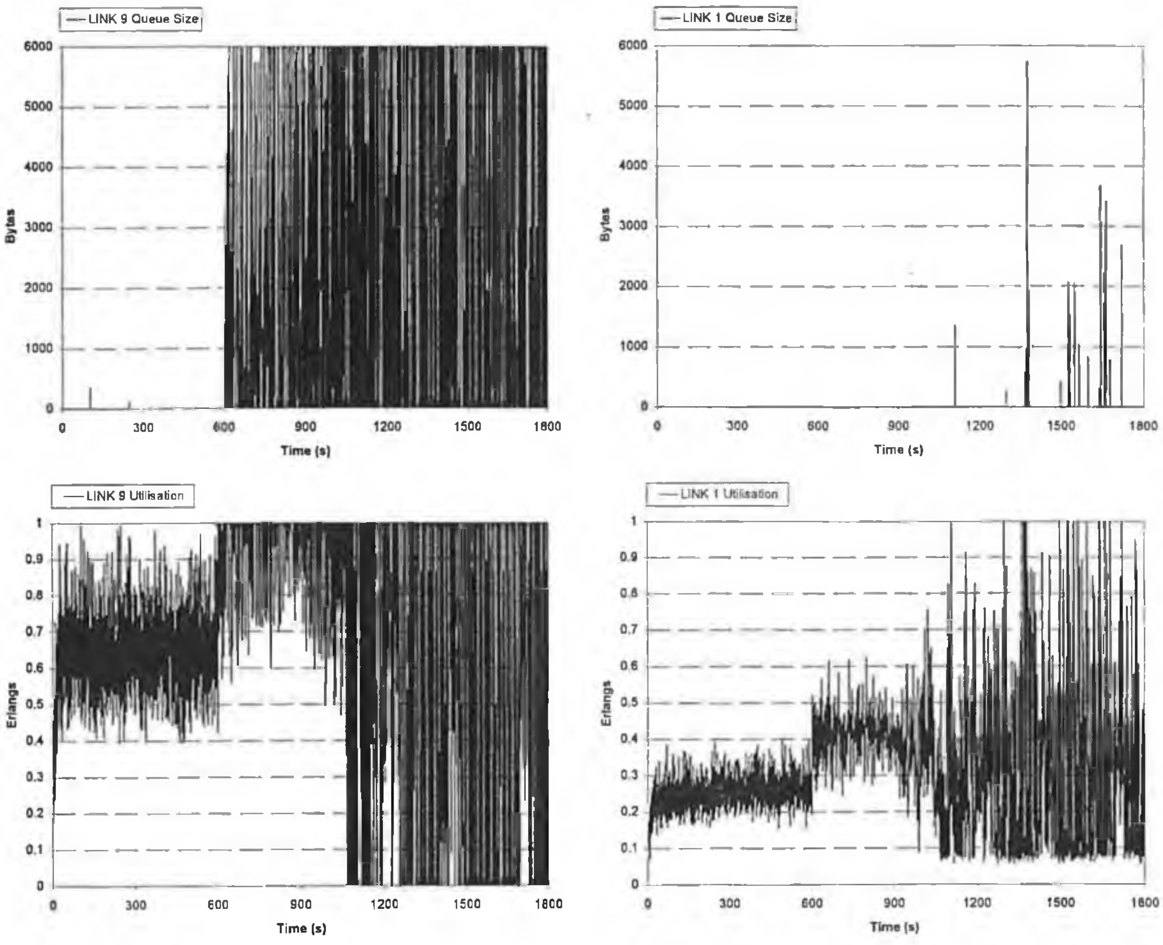


Figure 4.31: Quasi-analytic model. Link utilisation and queue size. Left: Link 9. Right: Link 1.

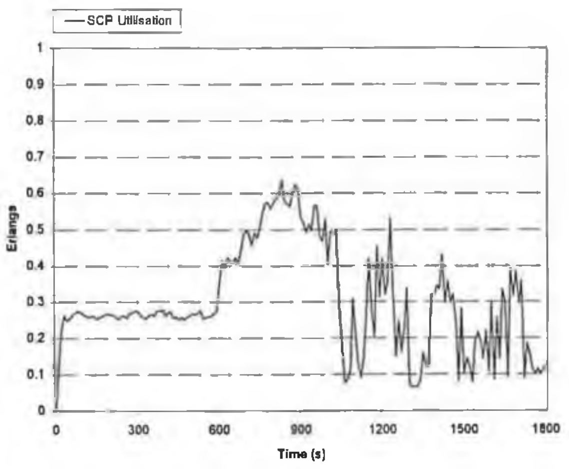


Figure 4.32: Quasi-analytic model. SCP utilisation.

End-user reattempts triggered by premature session terminations due to messages discarded at the reverse SCP links serve to increase the volume of IN messages sent by the SSPs to the SCP, thereby increasing the load offered to both the forward SCP links and the SCP itself. In this scenario the resultant increase in load offered to the SCP does not drive it into overload, however this is not the case for the forward SCP links. Figure 4.33 shows the utilisation and queue size for link 0 and link 8. We see that the load offered to link 8 increases until the overload onset threshold is surpassed, SCCP load throttling then commences at STP2 and STP4 and the link queue size starts to oscillate. We note that onset of forward SCP link overload will serve to somewhat decrease the level of load offered to the reverse SCP links, however this decrease is of insufficient magnitude to impact upon their overloaded state.

Overload of the forward SCP links further increases the volume of end-user reattempts, which, as shown in Figure 4.33, lead to overload of the mesh links. This further exacerbates the overload, which grows in severity and persists well beyond the abatement of the increase in first-offered load. Overload of mesh links also means that completion rates for non-IN services are degraded to some degree, despite the fact that the overload originates in a part of the network not used by those services.

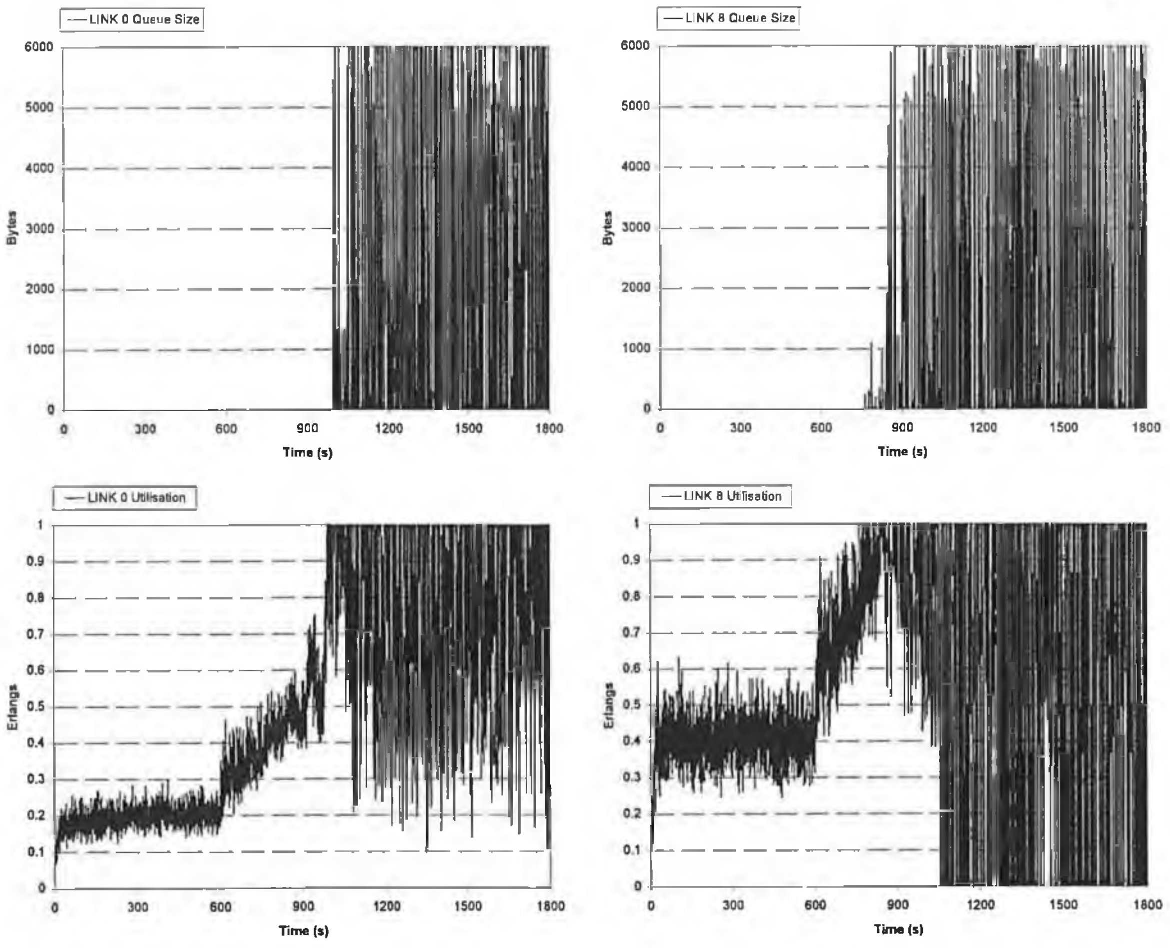


Figure 4.33: Quasi-analytic model. Link utilisation and queue size. Left: Link 0. Right: Link 8.

In this scenario forward, SCP links, reverse SCP links and mesh links overload, thus a large proportion of the IN messages are throttled or discarded. We therefore expect to see a severe degradation in completion rates for IN service sessions. Figure 4.34 shows the service session offer and completion rates for the three services. As in previous scenarios end-user reattempts increase the number of offered IN service requests and only a small number of these result in successful sessions. Completion rates for both IN services are very poor, falling below their non-overload levels. The slight degradation in completion rates for ISUP towards the end of the run is a result of the onset of overload of the mesh links.

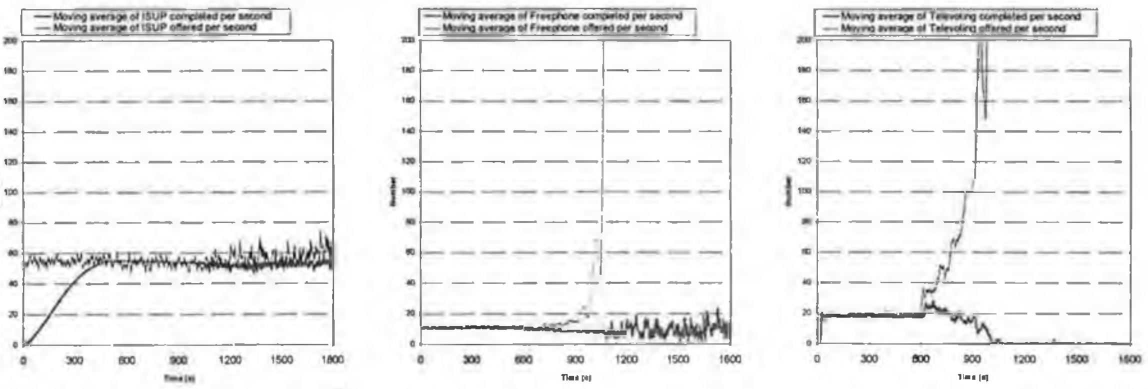


Figure 4.34: Quasi-analytic model. Moving average of service offer/completion rates. Left: ISUP Telephony. Centre: IN Freephone. Right: IN Televoting.

4.4.7 Scenario 7: Overload of SCP and Reverse SCP Links

For our final scenario we address the potential for simultaneous initial overload of the reverse SCP links and the SCP. The same load profile and network parameters as used previously are employed again, except that the SCP capacity is lowered so that it becomes overloaded once the level of Televoting requests is increased. We present results for two separate cases (relating to two values of SCP capacity), for which different behaviour with regard to the invocation of the ACG load throttle is observed.

For the first case, the SCP capacity is such that the SCP first-offered load is 0.95 Erlangs. Figure 4.35 shows the SCP utilisation, obtained using the quasi-analytic model. Instead of the oscillatory pattern normally associated with the invocation of the ACG mechanism, the utilisation is seen to stay, for the most part, within the target utilisation levels. This behaviour is related to the overload of the reverse SCP links (illustrated in Figure 4.36). Overload of the reverse SCP links results in message throttling by the SCP SCCP entity and some message discarding when the link queues saturate. Many of these messages will be the response messages at the start of a Televoting session, so the sessions are prematurely terminated and the SCP is freed from processing further messages it would normally receive to complete the sessions. Overload of the reverse links thus serves to lower the load offered to the SCP; the reduction ensures that, in this case, SCP utilisation remains mostly within acceptable levels.

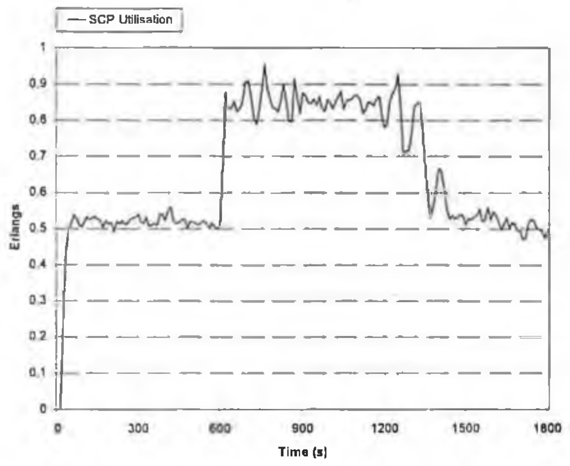


Figure 4.35: Quasi-analytic model. SCP utilisation.

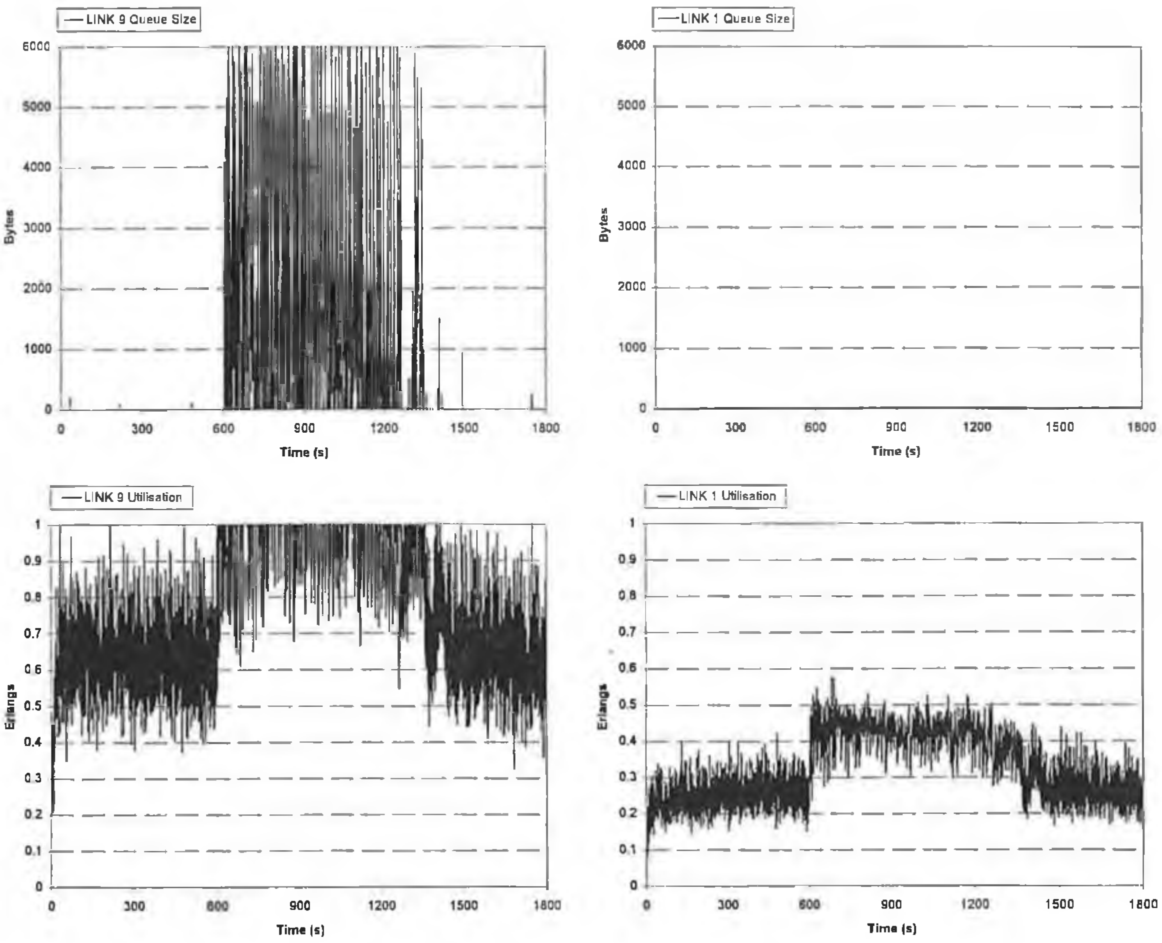


Figure 4.36: Quasi-analytic model. Link utilisation and queue size. Left: Link 9. Right: Link 1.

The effect of 'better' control of SCP utilisation can be seen from the session offer and completion rates, shown in Figure 4.37. Comparing against the scenario 1 rates (Figure 4.12, page 93), in which only the SCP overloads, we see that in this scenario a greater percentage of the offered sessions complete successfully. However, this apparent improvement is at the expense of the throttling/discarding of SCP response messages relating to ongoing service sessions. In contrast, for scenario 1 ACG controls SCP utilisation by throttling service *requests*, to a large degree avoiding the problem of premature session termination. Proper invocation of the ACG throttle would therefore be a more desirable (though of course flawed) means of SCP load control from the point of view of maximising overall network performance. These results provide an example of how the operation of the SS.7 controls can detrimentally interfere with the operation of the higher-level IN control.

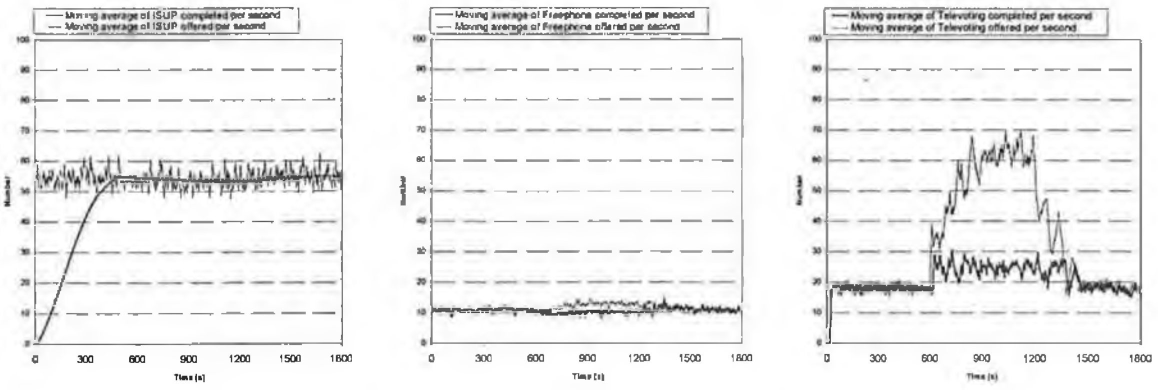


Figure 4.37: Quasi-analytic model. Moving average of service offer/completion rates. Left: ISUP Telephony. Centre: IN Freephone. Right: IN Televoting.

If the capacity of the SCP is lower than that used for the previous case, the reduction in SCP offered load resulting from reverse SCP link overload may not be sufficient to keep SCP utilisation within acceptable limits and ACG will be properly activated. To investigate this possibility, we adjust the capacity of the SCP so that its first-offered load becomes 1.15 Erlangs. Figure 4.38 shows the utilisation and queue size for the SCP and reverse SCP link 9. We see that the SCP is initially heavily overloaded, to the extent that its queue becomes saturated. This leads to activation of ACG and subsequent oscillatory behaviour, as previously discussed. Gapping remains in place until the overload situation abates.

We observe a reduction in offered load due to reverse SCP link overload: the SCP queue stops filling completely, but the reduction is not sufficient to interfere with the operation of ACG. From the link 9 graphs we see that the reverse SCP links still overload, but not as severely as in the previous case. However these overloads still trigger message throttling by the SCP SCCP entity and consequent premature session termination. To quantify the impact of message loss through SCP SCCP load throttling, we compare the offer and completion rates for the cases with and without reverse link overload (the latter is scenario 1, but we use a different SCP capacity and offered load volume). Figure 4.39 shows moving averages (window 100s) of the offer and completion rates for the two IN service types. We see that completion rates are broadly similar for both cases and that the major difference relates to the Televoting offer rates. Reverse SCP link overload results in sessions being prematurely terminated; these sessions will reattempt as normal and therefore the level of offered load increases. The presence of these prematurely terminated sessions leads to a greater degree of end-user dissatisfaction, which is highly undesirable from the network operator's viewpoint.

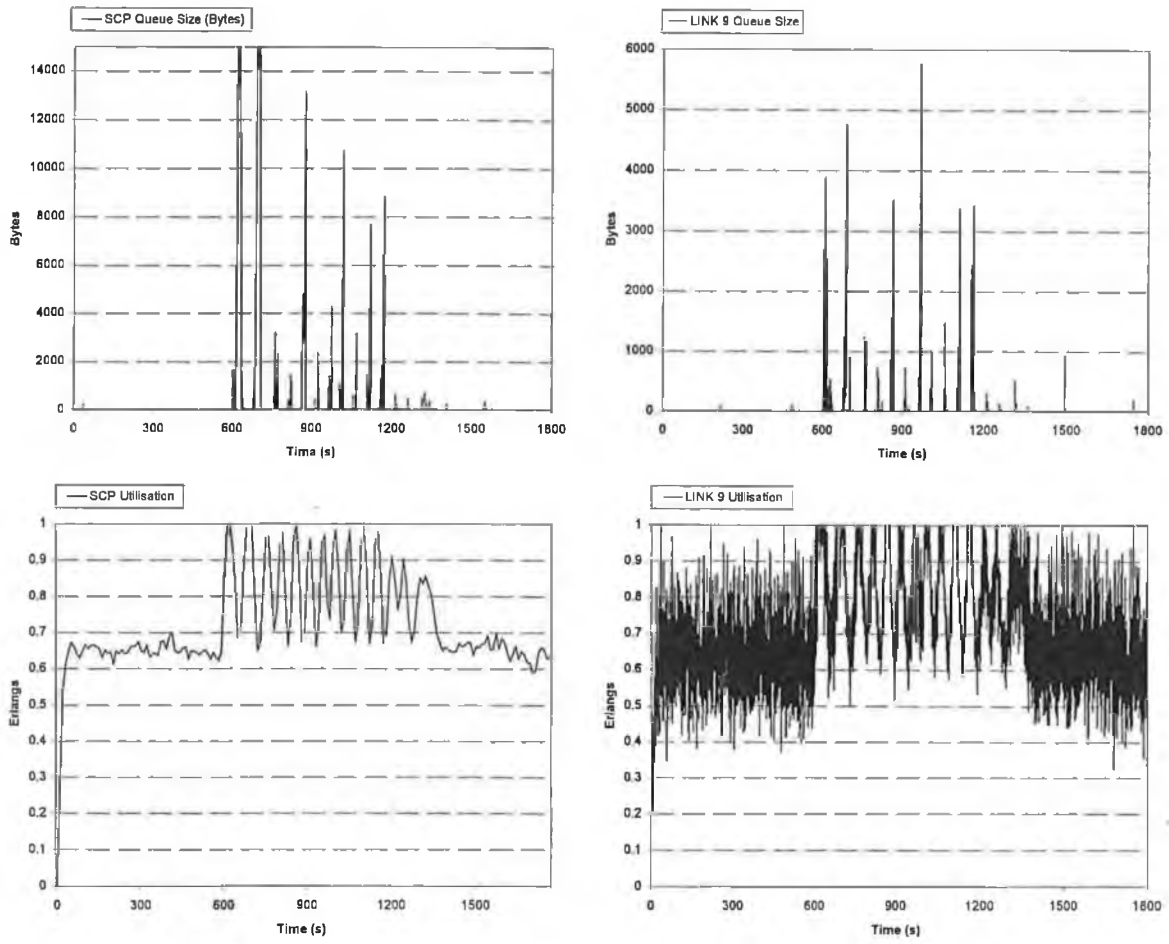


Figure 4.38: Quasi-analytic model. Utilisation and queue size. Left: SCP. Right: Link 9.

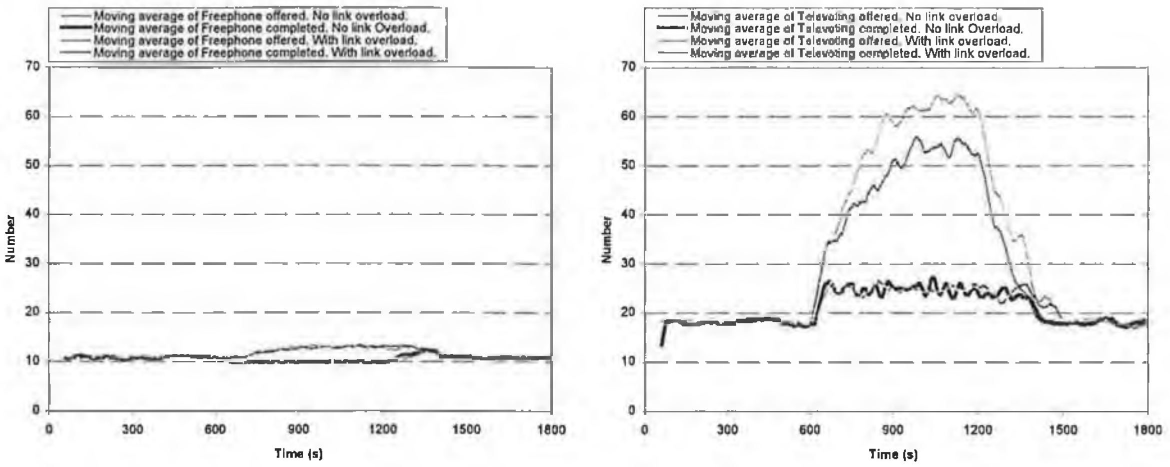


Figure 4.39: Quasi-analytic model. Comparison of offer/complete rates between cases where links are and are not overloaded. Left: IN Freephone. Right: IN Televoting.

4.5 SUMMARY AND CONCLUSIONS

We now summarise the main findings of the analysis of the seven overload scenarios. Scenarios 1 and 2 were designed to replicate, using our models, the behaviour of IN ACG and the SS.7 TFC-based load control that has been previously demonstrated in the literature. Scenario 1 showed that static, table-driven IN controls like ACG are unlikely to act with sufficient granularity of control to meet SCP utilisation targets and prevent message discard at saturated SCP input queues. They therefore lead to poor session completion rates for the services supported by the SCP. Scenario 2 demonstrated the inherently oscillatory behaviour of the SS.7 load controls, which results from temporal over-control and source synchronisation. The impact of end-user reattempts and (more dramatically) ISUP REL/RSC message reattempts on prolonging overload beyond the time at which first-offered traffic levels abate, was also demonstrated. Both control oscillation and reattempt behaviour were shown to result in very severe degradation in session completion rates.

Scenario 2 exploited the use of a full mesh topology in our network models to demonstrate overload propagation caused by the inadequate performance of the SS.7 load controls. For this scenario, mesh links initially overloaded in one direction (due to an overload at the group A SSPs), but this overload was not adequately controlled and subsequently led to overload onset of the mesh links in the opposite direction. This overload propagation resulted in load throttling at the group B SSPs, despite the fact that they were not the source of the original overload.

Scenarios 3 and 4 addressed the requirement for SCCP load throttles in SS.7 networks carrying IN-related signalling traffic and analysed the operation of a typical SCCP load throttle. Scenario 3 argued the case for SCCP load throttles on the basis that they are necessary to ensure some degree of equitable treatment of SS.7 user parts during overload. In particular, the issue of fairness between the level of load throttling applied to ISUP and SCCP user parts was addressed. Factors such as the varying number of signalling messages in sessions of different session types, the presence of hybrid ISUP/IN services and differences in throttling algorithms were shown to adversely affect fairness when measured in terms of comparative completion rate percentages.

In scenario 4 only forward SCP links overload, so SCCP load throttles are clearly required to protect these links from sustained overload. The performance of the SCCP throttle was investigated and shown to suffer from similar drawbacks to those of the ISUP throttle. Moreover, for this scenario, the operation of the SCCP throttle was seen to have no positive effect during the overload situation. This was due to message throttling by SCCP entities at those STPs where GTT takes place; IN messages are independently throttled here and at the originating SSPs. Over-throttling leads to an increase in the volume of reattempts and hence increased prolongation of the overload situation.

Overload of both the forward SCP links and the SCP was investigated in scenario 5. It was shown that although SCCP load throttles were initially invoked to control link loads, it was the invocation of the ACG throttles that had the most significant effect, as they provided effective control of the link. For this case, the higher-level ACG control was demonstrated to be much superior to the lower level SS.7 control.

Scenario 6 addressed the potential for reverse SCP link overload, due to SCP response messages being larger than query messages and/or service sessions comprising more response than query messages. The scenario results demonstrated that reverse SCP link overload can indeed occur and, when it does, it leads to undesired throttling and discarding of SCP-generated response messages. This results in premature session termination and consequent degradation of completion rate performance. The potential for end-user reattempts caused by prematurely terminated sessions to result in overload of forward SCP and mesh links was also demonstrated; this phenomenon exacerbates further the degradation of completion rates.

The final scenario investigated overload of reverse SCP links simultaneously with overload of the SCP. Two separate cases were addressed. In the first, reverse SCP link overload caused a reduction in the load offered to the SCP, which kept its utilisation within the desired range. However, this came at the price of highly undesirable premature session terminations. In the second case, the reduction in load offered to the SCP was insufficient to prevent it from overloading and ACG was invoked. It was observed that although the operation of ACG meant that the load offered to the reverse SCP links was reduced, there was still a significant amount of message throttling/discarding and consequent premature session terminations. This scenario thus represented a clear example of invocation of the lower level SS.7 controls having a negative impact on the performance of ACG.

Given the analysis results summarised above, we can make a number of general remarks regarding the performance of the standard SS.7/IN load controls:

- Standard SS.7 and IN load controls detect overload after its onset and depend for their success on the values of a small number of statically-defined parameters. They react to overload, rather then preventing it. Moreover, the success of the reaction is highly dependent on the suitability of the statically-defined control parameters to factors such as network size and topology, service characteristics and offered load volume. Results presented here show that the static nature of the controls leads to oscillatory behaviour, causing premature session termination, overload propagation and poor completion rate performance;
- SS.7 load controls that throttle traffic on a message basis do not provide adequate load control performance. Results presented here show that throttling on a message basis, even where a priority-based system is employed, is inadequate and can lead to premature session

termination and poor completion rate performance. Message-based throttling may also lead to the invocation of user part recovery procedures, such as REL/RSC reattempts, which have a severely detrimental impact on network performance in overload conditions;

- SS.7 SCCP load throttles perform badly, particularly in networks where GTT is in place at STPs. Results presented here show that not only will SCCP load throttles suffer from the same problems as does the standard ISUP load throttle, but their deployment has no favourable impact in situations in which they would involve SCCP throttling at both originating SSPs and other GTT-enabled nodes. The need for some form of control of SCCP signalling traffic is clear, however our results indicate that the SCCP itself may not be the ideal place for such a control to reside;
- The presence of independent, uncoordinated load controls at both IN and SS.7 planes can lead to unintended and potentially harmful control interactions. For example, we have shown that in the presence of reverse SCP link overload, message throttling due to the operation of SS.7 controls degrades the performance of the higher-level IN ACG control. This supports the view that SS.7/IN resources should be controlled in a more integrated manner;
- Low-level SS.7 controls do not have available sufficient service-related information to allow them automatically ensure that high-level goals, such as fairness in the treatment of different service types, are attained. For example, our results show that the completion rates for different service types are highly dependent on factors such as the average number of messages in the sessions of particular service types. Furthermore, the uncoordinated operation of SS.7 and IN controls means that hybrid services like Freephone will be over-throttled in many overload situations. These observations point towards the desirability of application-level controls that can be easily configured to mirror network operator preferences regarding service priorities and fairness aspects.

Given these remarks we can identify some basic characteristics of an SS.7/IN load control strategy that would the drawbacks of currently-deployed approaches. These characteristics are as follows:

1. *Preventative control:* the strategy should primarily be focussed on the prevention of overloads, rather then reacting to them. One means of achieving this is through explicit resource allocation together with the enforcement of absolute limits on the number of sessions that can be accepted in a give timeframe;
2. *Reactive control using delay measurements:* Where it is not possible or practical to explicitly allocate all critical network resources, the strategy should employ reactive control. However, this aspect of the control should rely solely on measurements that are locally available in SSPs in order to detect overload onset. In particular, the use of response delays

for overload detection has been shown to be an attractive means for detecting overloads from the edge of the network [Arvidsson *et al.*, 1997];[47]

3. *Dynamic computation of control parameters:* Controls depending on static parameters cannot adapt to changes in traffic patterns, service mix and resource availability. An alternative is the use of dynamic approaches, which adjust control parameters in response to changes in network/traffic conditions;
4. *Co-ordinated control:* the strategy should co-ordinate the control of multiple instances of the different types resource types so that overall network performance is optimised.
5. *Application-level control:* placement of load controls at the application-level has the advantages of controlling traffic on a per-session basis (thus avoiding the problems with message-based controls discussed above); allowing the network operator greater control over aspects like fairness and service priorities; and avoiding major changes to already-deployed equipment and software systems.

In chapters 5 and 6 we develop a network-oriented SS.7/IN load control strategy that exhibits the characteristics listed above. Chapter 5 first specifies an SCP load control strategy founded on the explicit, profit-optimal allocation of SCP processing capacity and using a token-based paradigm. Chapter 6 then expands this approach to facilitate fully network-oriented control over both SS.7 and IN resources.

[47] We note that use of delay measurements offer an alternative means of detecting overload of signalling links due to IN and/or ISUP traffic (or indeed other traffic types). Such measurements will trigger (application-level) load throttling which would alleviate the link overload. This offers an alternative to the use of SCCP-based load throttles, which as we have shown, give extremely poor performance, particularly in scenarios where GTT is deployed in SS.7 network STPs.

CHAPTER 5

A TOKEN-BASED STRATEGY FOR IN SCP LOAD CONTROL

Our first step towards development of a network-oriented approach to load control for Intelligent Networks is to specify a strategy that provides effective control of the processing resources of a single SCP. The strategy is based on the use of *tokens*, allocations of which are dynamically computed on the basis of the current volume and make-up of offered load, in a manner that seeks to optimise generated profit. This dynamic behaviour contrasts sharply with the static nature of table-driven controls deployed in existing INs.

We commence in §5.1 by describing the concepts upon which the token-based approach is founded and providing a specification of the token generation algorithm. §5.2 outlines two IN load control strategies we use for a comparative analysis of the performance of the token-based strategy, then §5.3 provides an overview of the analytic and simulation models employed for this analysis. Results of the analysis are presented and discussed in §5.4. Finally §5.5 draws overall conclusions regarding the strategy's performance.

5.1 SPECIFICATION OF TOKEN-BASED STRATEGY

The token-based strategy (TOKEN for short) is based on two fundamental contentions. The first is that, where the processing charactersitics of a resource (here an SCP) is well known,[48] a load throttle that enforces absolute limits on the number of admitted usage requests can guarantee that overloads do not occur. Of course there will typically be usage requests of different classes vying for access to a resource, so during overload the problem arises as to which should be admitted. Our second contention is that the best solution to this problem is to endeavour to

[48] Equipment manufacturers provide network operators with specifications of the performance of SCPs under varying operational conditions. These may contain, for example, details of the relationship between the rate at which query messages arrive at the SCP and the mean processing delay for these queries. We assume that processing delays are independent of message arrival rates, although in reality this is unlikely to be the case. It would be possible to modify the token generation process to take into account a non-linear arrival rate / processing delay relationship in order to maximise SCP performance. Formation of 'processing requirement' bids could then be done on a proprietary basis by each equipment manufacturer.

maximise the total *profit* [49] generated by successful requests. This differs from traditional approaches to load control, in that we focus more on the network operator viewpoint: maximise profit whilst maintaining high quality of service, rather than on the equipment manufacturer viewpoint: maximise resource throughput in all load conditions.

In the strategy we enforce absolute limits on the number of admitted service requests through use of *tokens*. A token is representative of the amount of SCP processing capacity required by a session of a particular service type. Service requests are admitted only if an appropriate token is available, in which case the token is 'consumed' by the request. At the start of *control intervals* of fixed length tokens, collectively corresponding to a pre-specified SCP *target capacity*, are generated and distributed to the SSPs, where they replace the previous allocations.

The *token generation process* aims to maximise generated profit. Profit values are assigned to each service type (and possibly also on a per-SSP basis) by the network operator; they take into account aspects such as revenue generated by successful sessions of that service, financial penalties associated with rejecting a session of that service type and customer-perceived service importance. Besides profit values, other inputs to the process are the available SCP processing capacity, the total processing requirements of sessions of the various service types and per-SSP, per service type estimates of the future service request arrival rates. These are referred to as *bids*; the former two are supplied by the SCP itself and the latter are sent to the SCP by SSPs. The outputs of the process are per-service, per-SSP, per-SCP *token allocations*, which are sent to the SSPs, where they dictate the number of requests for each service that the SSP can accept over the coming control interval.

The SCP available processing capacity bid is formulated by simply multiplying the target capacity by the length of time for which tokens will be valid. However, more complex approaches that, for example, take into account the future processing requirements of currently ongoing service sessions, would be possible. For SSP bids we assume that the intervals between token generations are small enough that the number of requests that arrived in the previous interval will be a sufficiently accurate estimate of the number that will arrive in the coming interval.[50] Again more complex formulations, for example taking into account traffic trend analysis, would be possible.

[49] We use profit as a generic term to represent any form of prioritisation.

[50] For 'short' measurement intervals, of the order of tens of seconds, arrivals of requests for telephony calls have be shown to be sufficiently well approximated by the Poisson arrival process. Since IN service requests originate from telephony calls it is invariably assumed in the literature that they also exhibit similar behaviour. Therefore, assuming that the control interval is suitably small, the number of arrivals in one period should be a sufficiently accurate estimate of the number arriving in the following interval (although the estimate will clearly be inaccurate if there is a sudden change in the overall mean arrival rate).

The token generation algorithm is a resource-oriented market algorithm that maximises expected profit by allocating tokens one-by-one, such that the expected increase in profit is maximised for every token allocated. To do so, it associates a *price* with every token allocated, with price being defined as the profit value of the associated service times the probability that the token will be consumed. Clearly the probability of token usage will be higher the higher the magnitude of the bid sent by the SSP and will decrease as tokens are allocated in response to that bid. For an arbitrary SSP bid this relationship can be expressed as a *utility function*, which relates expected profit to the number of allocated tokens. This utility function will be convex, because expected profit will increase as does the number of allocated tokens and the magnitude of these increases will tend to zero as the number of allocated tokens goes to infinity. Clearly the *global utility function*, which is simply the sum of all the utility functions for all the submitted bids, will then also be convex. Overall expected profit can then be maximised through use of a greedy algorithm, which successively allocates tokens such that each allocation is the one that produces the largest increase in utility at the smallest cost in terms of price per unit of SCP capacity. The algorithm proceeds by iterative examination of all possible candidate allocations, terminating when SCP target capacity has been allocated.

The operation of the token generation process is illustrated in Figure 5.1. SSP bids are represented as white circles and tokens allocated in response to these bids as coloured circles. As the generation algorithm iterations proceed tokens are generated and the remaining *supply* of SCP processing capacity diminishes. When the algorithm terminates the token allocations are sent to the SSPs,[51] where they are consumed under the control of a *token spending algorithm*. In non-overload conditions more tokens than the number requested will be allocated in response to all bids, whereas during high load conditions at least some SSPs will be allocated less tokens than the number requested. Tokens are not additive between control intervals, hence when a new token allocation is received by an SSP it replaces the previous allocation.

5.1.1 Specification of Token Generation Algorithm

We now specify the token-generation algorithm in detail. Assume an IN consists of a single SCP, K SSPs and supports J service types; let k and j denote an arbitrary SSP and service type respectively. Let $r_k(j)$ denote the profit generated by a successful session of service type j originating at SSP k. Let the total available capacity bid by the SCP be denoted by c and let the total processing costs associated with a type j service session be denoted by $p(j)$.

[51] In our analyses we assume that the total time taken for SSPs to send bids, for the execution of the token generation process and for SSPs to receive token allocations is negligible. In addition, we assume error-free transmission of both bids and token allocations.

Let $q_k(j)$ denote the bid of SSP k for tokens of service type j and let $n_k(j)$ denote the number of tokens allocated in response to this bid.

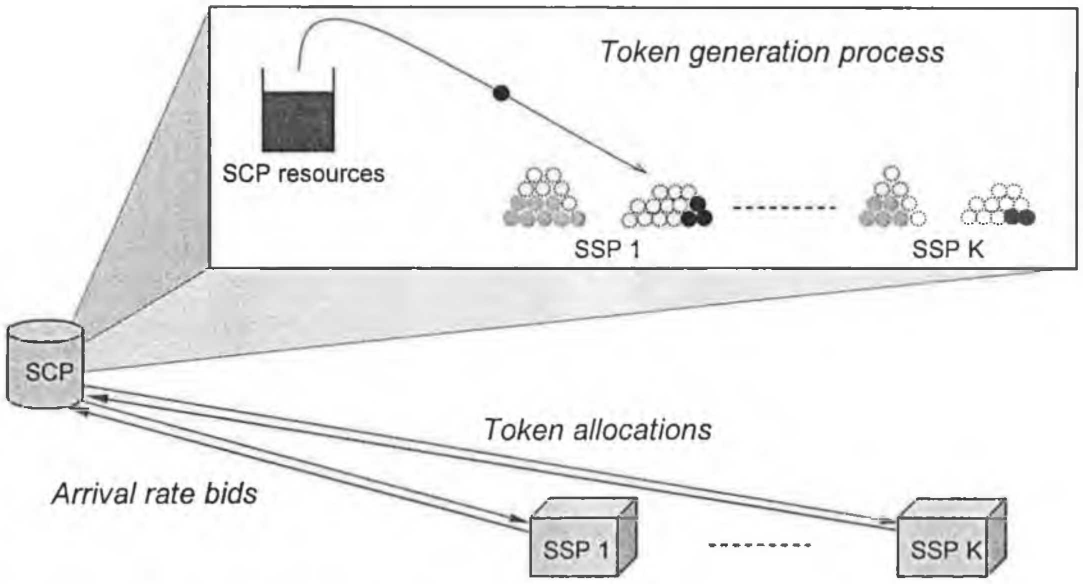


Figure 5.1: Illustration of token-generation process.

All SSPs maintain J pools of tokens, one for each service type. When the SSP admits a type j service request into the network one token is removed from token pool j. If pool j is empty then, until the token pools are refilled, all further requests for the service type j will be rejected. Token pools are refilled as a result of the token generation process, which is executed at the start of control intervals of duration T time units.

Tokens are generated such that expected overall utility, measured as total profit generated over the next T time units, is maximised. The method is to allocate tokens one by one such that the expectation of marginal utility to marginal cost is maximised in each allocation. During the process records are kept of the total remaining supply of processing capacity and the price per allocation of tokens.[52] These records are denoted by s and $c_k(j)$ respectively.

The *marginal utility* of an additional token is the profit expected from that token; it is computed as the profit associated with a successful session times the probability that the token will actually be consumed. Let $u_k(j)$ denote the marginal utility associated with allocating a type j token to SSP k and let $n_k(j)$ denote the total number of type j tokens allocated to SSP k during the preceding iterations. By interpreting the expected number of service requests $q_k(j)$ as the average of a Poisson distribution, we obtain $u_k(j)$ as:

$$u_k(j) = r_k(j) \sum_{w=n_k(j)+1}^{\infty} \frac{q_k(j)^w}{w!} e^{-q_k(j)}$$

[52] Token allocation prices are not used in the single SCP case studied in this chapter, however they play an important role in the distributed token generation case described in Chapter 6.

The marginal cost of an additional token can be simply interpreted as the total processing capacity expended by the SCP on the session associated with consumption of the token. Let $v(j)$ denote the marginal cost associated with issuing a type j token. We have:

$$v(j) = p(j)$$

Let a (j,k)-allocation refer to assigning a type j token to SSP k. The marginal utility per marginal cost of such an action, which we denote by $\delta_k(j)$, is:

$$\delta_k(j) = u_k(j)/v(j)$$

$\delta_k(j)$ expresses the derivative of the utility function with respect to the processing required for a (j,k)-allocation. The algorithm seeks to maximise total overall utility by distributing the resources in a series of allocations such that each allocation results in a maximal increase in overall utility. The optimal allocation in each step is thus the one with the highest derivative.

It is possible that more then one candidate allocation will provide the same maximum increase in utility.[53] From the viewpoint of achieving global utility any candidate allocation can be selected at random. We recognise however that this situation is an opportunity to apply secondary criteria when making such allocations. A multi-step 'culling' procedure, where at each step a number of candidate allocations are eliminated can be employed, after which one of the remaining candidates can be selected at random. In the algorithm statement below we provide some example elimination criteria, though other types of criteria would also be possible.

We can now state the algorithm formally. For ease of computation a recursion-based approach is employed for updating $u_k(j)$ at each iteration of the algorithm. For this approach $\pi_k(j)$ is used to denote the probability that there will be exactly $n_k(j)$ arriving requests for service type j at SSP k during the coming control interval and $\Pi_k(j)$ is used denote the probability that there will be more then $n_k(j)$ arriving requests for service type j at SSP k during the coming control interval.

Step 1: Initialisation.

```
For all SSPs $k = 1,...,K$ do:
    For all service types $j = 1,...,J$ do:
        Set token allocations $n_k(j) = 0$ and prices $c_k(j) = 0$.
```

[53] In general, the only time that multiple allocations provide the same maximum increase in utility is where the SSP bids are the same. If these bids are based on counts of previously arrived requests then this will be a relatively infrequent occurrence. However, it is possible that a network operator may choose to fix bids for certain services, in which case the culling procedure will be frequently invoked.

```
        Set $\pi_k(j) = e^{-q_k(j)}$.
        Set $\Pi_k(j) = 1 - \pi_k(j)$.
        Set marginal utilities $u_k(j) = r_k(j)\Pi_k(j)$.[54]
    For all service types $j = 1,\ldots,J$ do:
        Set marginal costs $v(j) = p(j)$.
    Set remaining SCP processing capacity $s = c$.
```

Step 2: Identify optimal allocations.

```
    For all SSPs $k = 1,\ldots,K$ do:
        For all service types $j = 1,\ldots,J$ do:
            List all candidate allocations that maximise $\delta_k(j)$.
```

Step 3: Arbitrate between optimal allocations.

```
    For all SSPs $k = 1,\ldots,K$ do:
        For all service types $j = 1,\ldots,J$ do:
            If allocation $(j,k)$ is in the candidate list then:
                Criterion A: Step size.[55]
                    If $p(j) < p_{\max}(j)$, where $p_{\max}(j) = \max(p(1),\ldots,p(J))$, then:
                        Eliminate candidate allocation $(j,k)$.
                Criterion B: Provider concentration.[56]
                    If $\sum_{k=1}^{K} n_k(j) > n_{\min}(j)$, where $n_{\min}(j)$ denotes the minimum number of tokens
```

[54] Initially we have $n_k(j) = 0$. Hence:

$$u_k(j)\Big|_{n_k(j)=0} = r_k(j)\sum_{w=1}^{\infty}\frac{q_k(j)^w}{w!}e^{-q_k(j)} = r_k(j)\left(\sum_{w=0}^{\infty}\frac{q_k(j)^w}{w!}e^{-q_k(j)} - e^{-q_k(j)}\right) = r_k(j)\left(1 - e^{-q_k(j)}\right)$$
$$= r_k(j)\Pi_k(j)$$

[55] To maximise expected overall utility, larger steps in terms of resources spent must be preferred to smaller ones (larger values of $p(j)$ means we have larger values of probability of token usage and/or $r_k(j)$). This is achieved by identifying the maximal step size among the allocation candidates and eliminating the ones with smaller step sizes.

[56] To ensure stability against sudden load changes with respect to specific services, the SCP is encouraged not to focus on particular services. This is achieved for each service by identifying the allocation candidates that refer to the lowest number of issued tokens and eliminating the ones where more tokens have been issued.

allocation for any of the service types $j = 1,\ldots,J$, then:

Eliminate candidate allocation (j,k).

Criterion C: Fairness.[57]

If $n_k(j)/q_k(j) > m_k(j)$, where $m_k(j)$ denotes the minimum of the proportions of demands satisfied for any of the SSPs $k = 1,\ldots,K$ and service types $j = 1,\ldots,J$, then:

Eliminate candidate allocation (j,k).

Selection.

Randomly select allocation (j',k') from those remaining in the candidate list.

Step 4: Perform optimal allocations.

```
Set $n_{k'}(j') = n_{k'}(j') + 1$.
Set $c_{k'}(j') = c_{k'}(j') + u_{k'}(j')$.
Set $s = s - p(j')$.
```

Step 5: Update internal variables.

```
Set $\pi_{k'}(j') = \pi_{k'}(j') q_{k'}(j') / n_{k'}(j')$.
Set $\Pi_{k'}(j') = \Pi_{k'}(j') - \pi_{k'}(j')$.
Set $u_{k'}(j') = r_{k'}(j') \Pi_{k'}(j')$.
```

Step 6: Loop statement.

```
If $s > 0$ then:
    GOTO Step 2.
Else:
    STOP.
```

When the token generation process terminates the supply of SCP processing capacity will have been distributed in an profit-optimal manner between the SSP demands. Token allocations are

[57] To ensure fairness, even under conditions where the demand is much higher or much lower than the supply (in which case marginal utility for candidate allocations may be all 0 or $r_k(j)$ respectively, independent of current allocation $n_k(j)$), fairness in relative allocation is encouraged. This is achieved for each SSP and service type by identifying the allocation candidates for which the lowest allocation of tokens relative to demand have been issued and eliminating those having higher relative allocations.

transmitted to the SSPs, which use them to replenish the token pools. Token allocations are not additive between intervals, thus new allocations invalidate old ones.

5.1.2 Specification of Token Spending Process

If service requests arriving at an SSP are simply allowed to consume tokens if any remain, then, in high load conditions, token pools will quickly deplete at the start of the control interval. This can lead to large increases in the SCP queue size at the start of the interval, resulting in a significant increases in response delays for end-users. To avoid this, we specify a token-spending process that ensures tokens are consumed at an approximately steady rate for the duration of the control interval. The process employs percentage thinning to regulate the acceptance of service requests. At sub-intervals of the control interval, PT coefficients for each service type are updated, using estimates of the number of requests that will arrive before the end of the control interval and the number of remaining tokens. Arriving service requests are subjected to a percentage thinning throttle using the relevant PT coefficient as parameter.[58] We now specify the PT coefficient updating and token spending algorithms.

5.1.2.1 PT Coefficient Updating Algorithm

Let τ denote the length of the sub-intervals; τ is chosen such that $T = A.\tau$, where A is some integer. Let $a \in (1, A)$ denote the current sub-interval number. Let $\gamma^*(j)$ denote the estimated number of arrivals of requests for service type j until the end of the control interval and let $n'(j)$ denote the number of tokens remaining for service type j. Let $m_j(a)$ denote the number of requests for service type j which arrived during sub-interval a and let $m'_j(a)$ denote the number of these that were accepted. Finally let $p^a(j)$ denote the probability of acceptance of a request for service type j (this is the PT coefficient). The algorithm contains two steps: the first (Initialisation) is executed at the start of the control interval and the second (Update PT coefficients) is executed at the start of each sub-interval.

Step 1: Initialisation.

Set $a = 1$.

For all service types $j = 1,...,J$ do:

Set $\gamma^*(j) = q_k(j)$, where $q_k(j)$ is the number of requests for service type j that arrived over the duration of the previous control interval.

Set $n'(j) = n_k(j)$.

[58] A undesirable side-effect of employing this algorithm is that it is possible for a small number tokens to remain unused at the end of the control interval even though requests were rejected. However, we believe that this can be tolerated given that we achieve a high degree of control over the SCP queue size.

```
        Set $p^a(j) = \min(1, n'(j)/\gamma^*(j))$.

Step 2: Update PT coefficients.

    For all service types $j = 1,...,J$ do:

        Set $n'(j) = n'(j) - m'_j(a)$.

        Set $\gamma^*(j) = \frac{(A-a)}{a}\sum_{a'=1}^{a} m_j(a')$.

        Set $p^a(j) = \min(1, n'(j)/\gamma^*(j))$.

        Set $a = a + 1$.
```

5.1.2.2 Token-spending Algorithm

With the token-spending algorithm arriving service requests are simply subjected to a percentage thinning throttle, which dictates whether they are to be rejected or accepted. The algorithm is as follows:

```
Select a random number $X$ uniformly distributed in the range $(0.0, 1.0)$.

If $X < p_a(j)$ then:

    Service request is accepted.

Else:

    Service request is throttled.
```

5.1.3 TOKEN Strategy Enhancement for Control of 'Legacy' SSPs

In this section we address the deployment of the TOKEN strategy in a network containing 'legacy' SSPs, that is SSPs not having the capability to control IN traffic using tokens, but which do employ some standard load throttle like call gapping.[59] In this scenario the SCP forms bids on behalf of the legacy SSPs using its knowledge of the number of initial service session messages received from the each legacy SSP during the last control interval and the gap interval length it set at the beginning of that interval. The token generation process proceeds as normal and, once completed, token allocations for the legacy SSPs are converted into appropriate gap intervals. We assume that the SCP is able to indicate to the legacy SSPs an actual gap interval duration, as opposed to an overload level. Were this not possible the SCP would require knowledge of the gap interval tables at the SSP, to allow it indicate an overload level that gives the most appropriate gap interval.

[59] It would also be possible to convert token allocations into parameters of other throttle types. For brevity we only consider call gapping here.

5.1.3.1 Estimation of SSP Service Request Arrivals

To produce an estimate of service request arrivals at the legacy SSPs during the previous interval, denoted $q_k^*(j)$, the SCP uses the calculation described below. For the case where the gap interval set at the start of the previous interval, denoted $g_k(j)$, is such that $g_k(j) < T$, the calculation makes use of the number of initial service session message arrivals measured at the SCP, denoted $d_k(j)$. The basis of the calculation is that the estimated length of time the gap timer was inactive in the previous interval, denoted $\tau_{inactive}^*$, divided by the interval length T, is equivalent to the proportion of accepted service requests, that is:

$$\frac{\tau_{inactive}^*}{T} = \frac{d_k(j)}{q_k^*(j)}$$

The gap timer is activated for each accepted request, hence for the case where $g_k(j) < T$ we have:

$$\tau_{inactive}^* = T - d_k(j).g_k(j) \quad if \quad g_k(j) < T$$

Note that the above is an estimate as it makes the simplifying assumption that the gap timer is not active at the end of the control interval. Substituting for $\tau_{inactive}^*$ in the first expression and solving for $q_k^*(j)$ yields:

$$q_k^*(j) = \frac{d_k(j).T}{T - d_k(j).g_k(j)} \quad if \quad g_k(j) < T$$

For the case where $g_k(j) \geq T$ the gap timer may be active over the full duration of the control interval, in which case it is not possible for the SCP to estimate the SSP arrival rate. For this reason we let the SCP simply use its last non-zero measurement of arrivals of initial messages for service type j coming from SSP k, which we denote $d'_k(j)$. That is:

$$q_k^*(j) = d'_k(j) \quad if \quad g_k(j) \geq T$$

5.1.3.2 Conversion of Token Allocations into Gap Intervals

The calculation for conversion of a token allocations, $n_k(j)$, into a corresponding gap interval, $g_k(j)$, is based on the intention that in the coming interval SSPs will accept*at most* the same number service requests as the number of tokens allocated. This number will thus be the maximum number of times the gap timer is permitted to be activated during the coming control interval. Furthermore, the maximum total amount of time for which the gap timer will be active should ideally equal the proportion of requests to be rejected times the control interval length.

Thus, for $q_k^*(j) \neq 0$, we have:

$$n_k(j).g_k(j) = \left[1 - {n_k(j)} \middle/ {q_k^*(j)}\right].T$$

When $q_k^*(j) = 0$ we set the gap interval to the control interval duration. Therefore, we have:

$$g_k(j) = \begin{cases} \left[{1} \middle/ {n_k(j)} - {1} \middle/ {q_k^*(j)} \right].T & if \quad q_k^*(j) \neq 0 \\ T & if \quad q_k^*(j) = 0 \end{cases}$$

5.2 STRATEGIES USED FOR COMPARISON

In order to analyse the likely performance of the TOKEN strategy both quantitatively and qualitatively it is necessary to compare it with one or more other IN load control strategies. Ideally it should be compared with both a 'traditional' strategy, of the sort currently deployed in IN systems, as well as with a more advanced strategy that is also adaptive[60] in nature and seeks to maximise generated revenue/profit. We do both, comparing TOKEN with the ACG strategy (described in §2.3.1.3) and Lodge's optimisation strategy (described in §2.3.1.4).

The version of the ACG strategy implemented for this study is the same as that used in Chapter 4 (*cf.* §4.1.3.2 and §4.1.3.3), which is in turn based on the ACG specifications, [Bellcore, 1994; 1995]. To summarise: overload detection takes place at the SCP using measurements of processor utilisation and a mapping rule that produces one of 13 overload levels, in a manner intended to keep processor utilisation in the range (0.8, 0.9) Erlangs during overload. The overload level is calculated every $T = 10s$ and indicated to all SSPs in the network by means of autonomous ACG messages. SSPs activate gapping on all sources; we assume all requests for a service type at an SSP constitute a single source. Gap intervals lengths are indexed by the overload level from the table of Bellcore-standard values, but they are randomised in the range (90%, 100%) for each source before being put in place. Neither control refreshing, nor a gap duration parameter are used.

The version of the optimisation strategy (which we refer to as OPT) studied here is closely based on the general strategy description provided in §2.3.1.4, the major difference being that, since SSP overload is not addressed, we do not have an optimisation process in operation at the SSPs. The service type weights used in OPT play a similar role to the profit values used in TOKEN. To ensure direct comparison we set the revenue values, R_j, used in OPT equal to the TOKEN profit values, $r_k(j)$, additionally assuming that for the TOKEN strategy profit values

[60] In describing a strategy as adaptive we simply mean that it computes load throttle parameters in a dynamic manner and so 'adapts' to the prevailing load conditions.

are the same at all SSPs ($r_k(j) = r(j) \;\; \forall \;\; k \in \{1,...,K\}$). We also set the values of all the OPT Quality-of-Service levels, q_j, equal to 1. Thus the weight for service type j is given by:

$$\omega_{SCP,j} = \frac{R_j q_j e_{SCP,j} \mu_{SCP,j}}{\sum_{j=1}^{J} R_j q_j e_{SCP,j} \mu_{SCP,j}} = \frac{r(j) e_{SCP,j} \mu_{SCP,j}}{\sum_{j=1}^{J} r(j) e_{SCP,j} \mu_{SCP,j}}$$

In the absence of SSP optimisation, the PT coefficients calculated by the SCP will be those put in place by the SSPs, *ie.* $p^a_{k,j}(t) = p^a_{SCP,k,j}(t)$. Assuming that available SCP processing value and service processing costs used in TOKEN are expressed in terms of fractions of the total SCP processing capacity then the OPT maximum SCP load equals the TOKEN SCP available processing capacity, that is $\rho^{\max}_{SCP} = c$. Given the above, the SCP optimisation problem can be stated as follows:

$$\underset{p^a_{1,1},...,p^a_{K,J}}{Maximise} \sum_{k=1}^{K} \sum_{j=1}^{J} r(j) p^a_{k,j}(t) q_k(j)$$

subject to the constraints:

$$I.\;\; SCP\ Load: \qquad \sum_{k=1}^{K} \sum_{j=1}^{J} \frac{p^a_{k,j}(t) e_{SCP,j} q_k(j)}{\mu_{SCP,j}} \le c$$

$$II\;\; Bounds\ on\ p^a_{k,j}(t): \qquad 0 \le p^a_{k,j}(t) \le 1 \quad \forall \quad k \in \{1,...,K\},\; j \in \{1,...,J\}$$

$$III.\;\; Weighting: \qquad 1 \le \frac{p^a_{k,j'}(t)}{p^a_{k,j}(t)} \le \frac{\omega_{SCP,j'}}{\omega_{SCP,j}} \quad \forall \quad k \in \{1,...,K\},\; j, j' \in \{1,...,J\}\ with\ j \ne j'$$
$$where\ \omega_{SCP,j'} = \max(\omega_{SCP,1},...,\omega_{SCP,J})$$

5.3 DESCRIPTION OF NEW IN MODELS

For this chapter we address IN SCP load control in isolation and ignore the potential for SS.7 overloads or failures that may adversely effect overall performance.[61] It is therefore possible to completely omit SS.7 resources from our models, which consequently are much simpler than those used for the chapter 4 analyses. An additional advantage of omitting SS.7 is that it is feasible to employ a more sophisticated approximation technique (Decomposition) for development of a (fully) analytic model. This section provides an overview of the 'new' simulation/analytic models and describes the service types modelled.

[61] This issue is re-visited in Chapter 6 where the Token-based load control strategy is enhanced in order to provide protection against SS.7 overload and equipment failures.

5.3.1 Simulation Model

The simulation model uses discrete event simulation to model the flow of traffic through the IN network on a packet-by-packet basis. The network topology employed consists of a single SCP and K SSPs; it is illustrated in Figure 5.2. The signalling network used to transfer messages between SSPs and the SCP is not modelled, with messages sent from one node arriving instantaneously at the input queue of the destination node with no messages being lost, having errors or arriving out-of-sequence. No functionality associated with the underlying voice-circuit network is modelled. SSPs are of three sizes, where size relates to the rate of IN service requests arrivals at that SSP during normal load conditions. The sizes are small, medium and large with a respective arrival rate ratio of 1 : 3 : 7; there are an equal number of SSPs of each size in the network.

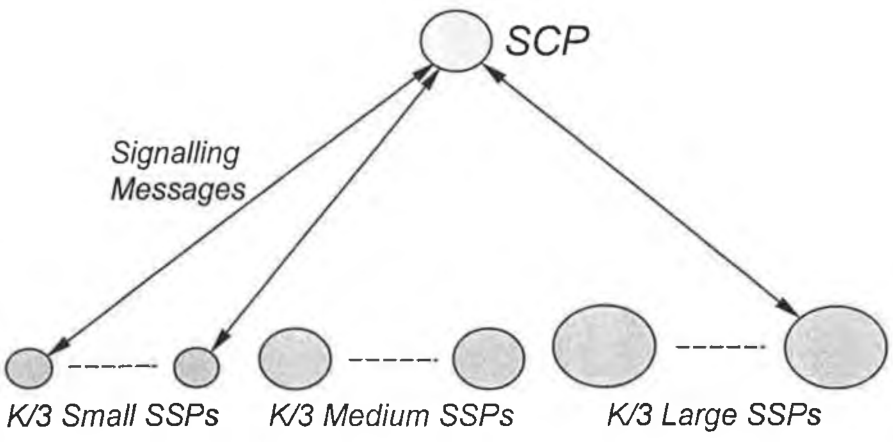


Figure 5.2: Network Topology for IN Simulation Model.

5.3.1.1 SSP Model

The SSP model is similar to that employed for the simulation model used in Chapter 4. It contains a single central processor of infinite capacity, hence signalling messages experience zero queuing and processing delays at the SSP and SSPs never overload. SSPs contain the following functional blocks: Service Request Generators, the SSF and the SRF.

The Service Request Generator functional blocks generate IN service requests according to independent Poisson processes with rate $\lambda_{k,j}$ requests per second (this rate may vary over the course of a simulation run) for each service class j and SSP k respectively. All requests of a particular service type arriving at a particular SSP are assumed to use the same GTA for communication with the SCP. End-user reattempt behaviour is modelled as follows: when a service request is throttled by the SSF or when a message associated with a service session is dropped or discarded at the SCP the end-user will reattempt the session with a probability of 0.7. Reattempts are scheduled after a random delay uniformly distributed in the range (60s, 90s). The end-user will re-attempt a service request a maximum of 2 times. In addition we assume that all calls are answered; in other words, the number of humans available to answer calls during overload conditions is not a limiting resource.

The SSF functional block maintains the BCSM instances for the supported services. It models the complete SSF-side processing of an IN service session, through BCSM instance instantiation, communication with other IN entities and session termination. It also realises the load throttles associated with the various load control strategies studied. The SRF functional block is assumed to contain an infinite number of voice circuits and associated resources, hence it never overloads. The only delay experienced at the SRF relates to the user delays involved in the playing of announcements and digit collection (these delays are specified in §5.3.3).

5.3.1.2 SCP Model

The SCP model is also somewhat similar to that used in chapter 4. It contains a single central processor that has the capacity to execute $c = 28{,}000{,}000$ instructions per second. Signalling messages arriving at the SCP require the execution of a different number of instructions, depending on their type; details of signalling message processing requirements are provided in §5.3.3. The SCP has a single input queue that can hold a maximum of 500 messages; we assume that all messages are of the same size, 100 Bytes. If the queue is full then all arriving messages are discarded, with no associated processing cost. If messages have been waiting in the queue for longer than $t = 2s$ by the time they commence processing they are dropped, a process that involves expending 24,000 instructions.

SCPs contain two functional blocks: the SCF and the SDF. The SCF incorporates the SLPs for the supported services. It models the complete SCF-side processing of service sessions, through SLP instantiation, handling of communication with other IN entities and SLP instance termination. It also incorporates functionality associated with the load control strategies studied. The SDF functional block models the reading and updating of database entries; we assume that delays associated with these actions are incorporated into the processing delay of the arriving message that triggered them.

5.3.2 Analytic Model

The analytic model was developed with the aim of modelling as closely as possible the behaviour of the IN simulation model, described above. Its main purpose is to provide a degree of validation of the efficacy of the results generated using the simulation model. In this section we specify the analytic model, note those aspects in which it differs from the simulation model and discuss the impact of these differences.

Considering the central processors of the SSPs and SCP as queues with arbitrary arrival/service rate distributions and number of servers results in the queuing network illustrated inFigure 5.3;

this mirrors the simulation model network topology illustrated in Figure 5.2. Throttles, operating on the input streams λ_k, are in place at the output of each SSP queue.

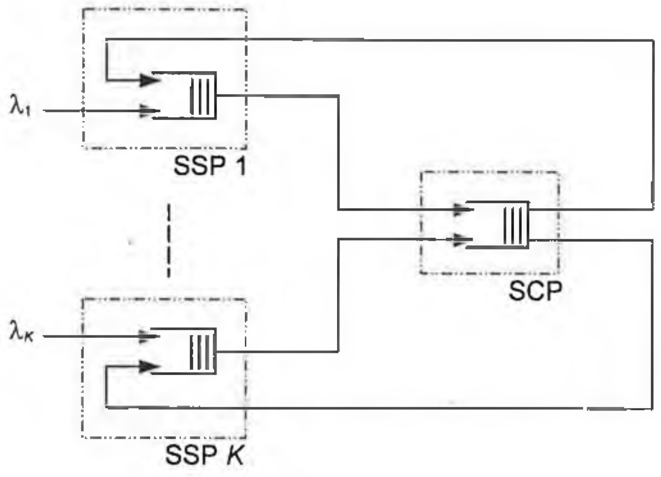


Figure 5.3: IN Queuing Network, 1 SCP and *K* SSPs.

In the simulation model SSP central processors are in effect treated as infinite service rate queues, hence in the analytic model SSP queues (but not throttles) can be ignored. We are then left with a single queue that models the SCP central processor. Modelling in strict accordance with the simulation model would necessitate use of a multi-server queue, with one server for each type of message that arrives at the SCP and a service rate corresponding to the number of instructions required for processing of that message type. Our first simplification is to assume that all messages associated with a particular service involve the same number of instructions (the mean value for that service), consequently we need model only a *J*-server queue. For ease of analysis we additionally assume that that actual service times for arriving messages are exponentially distributed.

In the simulation model the SCP central processor has a single queue with a finite capacity of 500 messages. For the analytic model we model a queue with infinite capacity, hence no message discarding occurs. We do not model the dropping of messages that have been excessively delayed in the queue. These differences may lead to some discrepancies between the results obtained using the simulation and analytic models.

Delays representing user interactions are omitted from the analytic model since, in steady state, they have no impact on the performance of the system. The main impact of this simplification is that measurements obtained from the analytic model will be seen to settle almost immediately after a change in input traffic; this will not be the case for the simulation model. We do not model end-user reattempt behaviour in the analytic model, but overcome this limitation by comparing results with results from simulation runs where end-user reattempts are also omitted.

The final simplification relates to the models of the load throttles. We model all throttles types by means of PT coefficients. OPT provides these coefficients directly, but the outputs of ACG and TOKEN (gap interval lengths and token allocations respectively) must be converted into equivalent PT coefficients; formulae for these conversions are provided below. This conversion will lead to minor discrepancies in load throttling behaviour between the simulation and analytic models. We note that this simplification means that in effect we do not model any wastage of tokens that may arise from the token spending algorithm.

5.3.2.1 Specification of Analytic Model

Given the simplifications and assumptions described above we proceed to specify the analytic model. The model outputs the mean SCP load and mean SCP queue size for a interval, given the arrival rates of requests for all J services at all K SSPs and PT coefficients for the same interval. Let T denote the interval duration, which we set equal to the control interval of the load control strategies we study. Let $\lambda_{k,j}(t)$ denote the mean arrival rate of *requests* for service type j at SSP k at time t. Let $p^a_{k,j}(t)$ denote the PT coefficient for service j at SSP k as set at time t; these values are in effect the probabilities of acceptance of service requests during the interval $[t,t+T]$. The mean arrival rate of *sessions* relating to service type j at SSP k at time t, denoted $\lambda_{SCP,k,j}(t)$, is then given by:

$$\lambda_{SCP,k,j}(t) = p^a_{k,j}(t)\lambda_{k,j}(t)$$

The SCP queue has J servers; let $\mu_{SCP,j}$ denote the service rate of server j. Let $\gamma_{SCP,j}(t)$ denote the arrival rate of *messages* of service type j at the SCP at time t. Let $e_{SCP,j}$ denote the number of messages relating to a session of service type j that arrive at the SCP for processing. Finally, let $\rho_{SCP}(t)$ denote the load of the SCP at time t.

Estimation of SCP arrival rates and SCP load

The arrival rates of service type j messages at the SCP can be estimated by:

$$\gamma_{SCP,j}(t) = e_{SCP,j}\sum_{k=1}^{K}\lambda_{SCP,k,j}(t) = e_{SCP,j}\sum_{k=1}^{K}p^a_{k,j}(t-T)\lambda_{k,j}(t)$$

These values can be used to estimate the load of the SCP, as follows:

$$\rho_{SCP}(t) = \sum_{j=1}^{J}\frac{\gamma_{SCP,j}(t)}{\mu_{SCP,j}}$$

Estimation of SCP mean queue size and mean queuing delay

We employ the decomposition approximation method, described in §3.1.1.3, to estimate SCP mean queue size. The following expression for the square of the variation coefficient (svc) of the interarrival times of messages at the SCP at time t, denoted by $ka_{SCP}(t)$, was generated by substituting into the general form of the decomposition equations given in §3.1.1.3.

$$ka_{SCP}(t) = \frac{\sum_{k=1}^{K}\sum_{j=1}^{J} p^{a}_{k,j}(t-T)\lambda_{k,j}(t)}{\sum_{j=1}^{J} e_{SCP,j} \sum_{k=1}^{K} p^{a}_{k,j}(t-T)\lambda_{k,j}(t)}$$

The mean SCP queue size at time t, denoted $\overline{L_{SCP}(t)}$, can now be estimated using Kingman's formula. Note that since the service times associated with the J servers are all exponentially distributed we set the svc of the service time of messages at the SCP, $Ks_{SCP}(t)$ equal to 1, giving:

$$\overline{L_{SCP}(t)} = \rho_{SCP}(t)\left[1 + \frac{\rho_{SCP}(t)[ka_{SCP}(t)+1]}{2[1-\rho_{SCP}(t)]}\right]$$

The mean queuing delay at the SCP queue at time t, denoted $\overline{T_{SCP}(t)}$, can then be estimated by applying Little's Law:

$$\overline{T_{SCP}(t)} = \frac{\overline{L_{SCP}(t)}}{\sum_{j=1}^{J} \gamma_{SCP,j}(t)}$$

Calculation of Equivalent PT Coefficients for ACG Strategy

Let $g_{k,j}(t)$ be the gap interval duration set by the SCP for service type j at SSP k at time t. In the case where $g_{k,j}(t) \geq T$ we can simply set the equivalent PT coefficient, $p^{a}_{k,j}(t)$, equal to zero:

$$p^{a}_{k,j}(t) = 0 \qquad if \qquad g_{k,j}(t) \geq T$$

Where $g_{k,j}(t) < T$ we make use of the fact that all requests arriving while the gap timer is active are accepted, whilst all those arriving when it is inactive are rejected. Therefore, assuming that the gap timer is inactive at the start and end of the control interval, the value of $p^{a}_{k,j}(t)$ can be calculated as the fraction of the control interval for which the gap timer is inactive. If we denote the aggregate time for which the gap timer is inactive by $\tau_{inactive}$, we have:

$$p^{a}_{k,j}(t) = \frac{\tau_{inactive}}{T} \qquad if \qquad g_{k,j}(t) < T$$

Now, the number of service j requests that will be accepted by SSP k during the control interval is given by $\lambda_{SCP,k,j}(t+T).T$, so the aggregate length of time for which the gap timer will be active, denoted τ_{active} is given by:

$$\tau_{active} = \lambda_{SCP,k,j}(t+T).T.g_{k,j}(t) = p^{a}_{k,j}(t)\lambda_{k,j}(t+T).T.g_{k,j}(t)$$

Given the above we can calculate $\tau_{inactive}$ as:

$$\tau_{inactive} = T - \tau_{active} = T - p^{a}_{k,j}(t)\lambda_{k,j}(t+T).T.g_{k,j}(t)$$

Substituting for $\tau_{inactive}$ in the expression for $p^{a}_{k,j}(t)$ and solving yields:

$$p^{a}_{k,j}(t) = \frac{1}{1+\lambda_{k,j}(t+T).g_{k,j}(t)} \qquad if \qquad g_{k,j}(t) < T$$

In summary:

$$p^{a}_{k,j}(t) = \begin{cases} \dfrac{1}{1+\lambda_{k,j}(t+T).g_{k,j}(t)} & if \quad g_{k,j}(t) < T \\ 0 & if \quad g_{k,j}(t) \geq T \end{cases}$$

Calculation of Equivalent PT Coefficients for TOKEN Strategy

If the number of service request arrivals during an interval is equal to or greater than the number of allocated tokens then the required PT coefficient is 1. Otherwise, the PT coefficient is calculated by dividing the number of allocated tokens by the number of arrivals during the interval. This leads to the following conversion formula:

$$p^{a}_{k,j}(t) = \begin{cases} 1 & if \quad \lambda_{k,j}(t+T).T < n_{k,j}(t) \\ \dfrac{n_{k,j}(t)}{\lambda_{k,j}(t+T).T} & if \quad \lambda_{k,j}(t+T).T \geq n_{k,j}(t) \end{cases}$$

5.3.3 Supported Services

The SS.7/IN models used in Chapter 4 supported two basic IN services (Freephone and Televoting), as well as a single non-IN service (ISUP Telephony). Here and in chapter 6 we wish to focus on services that are more complex (in terms of number of signalling messages exchanged per session) and therefore more representative of the type of service currently being deployed on IN platforms. We model three 'new' service types: *Virtual Private Network*, *Ringback* and *Restricted Access Call Forwarding*. The models of these services are loosely based on the prototype implementation descriptions in [Karagiannis *et al.*, 1998]. One of the primary reasons for using these three services is that these descriptions are sufficiently detailed to allow reasonably accurate estimation of processing requirements of individual signalling messages arriving at the SCP, thus the accuracy of our models is improved.

5.3.3.1 Service A: Virtual Private Network

Virtual Private Network (VPN) services create a logical sub-network, spanning single or multiple IN network domains, which appears to a specific group of end-users as a private network, providing a range of features normally associated with private telephony exchanges. All calls are controlled by an SCP, which provides facilities such as number translation and call monitoring. Figure 5.4 illustrates the sequence of information flows involved in a successful session of the service; the labels in brackets will be used to identify the signalling messages corresponding to the IFs. The profit associated with a successful session of the VPN service is 5 profit units.

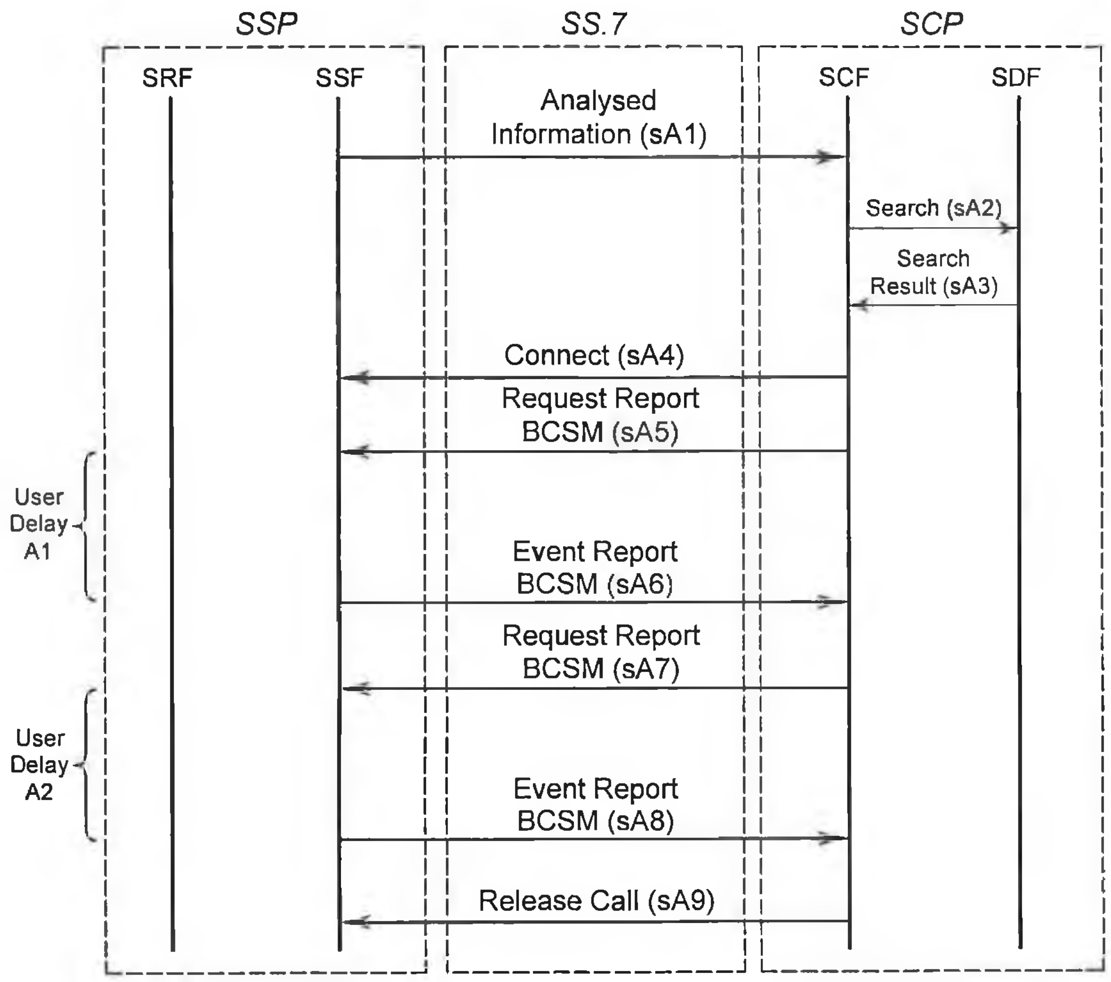


Figure 5.4: Information Flow sequence for Virtual Private Network service session.

The processing requirements, in terms of executed instructions, for messages arriving at the SCP are shown in Table 5.1. These values were used directly in the simulation model, whilst for the analytic model the mean message execution time was used for the calculation of the SCP service rate for Service A.[62]

Table 5.1: Service A SCP processing requirements.

Arriving Signalling Message	Departing Signalling Message(s)	SCP Instructions
sA1	sA4, sA5	170,000
sA6	sA7	35,000
sA8	sA9	35,000
		240,000

In the simulation model the duration of the end-user delay phases of the service are drawn at random from a negative exponential distribution on a session-by-session basis. The mean of the distribution is 5 seconds for user delay A1 (telephone ringing) and 100 seconds for user delay A2 (conversation). In the analytic model these end-user delays are not modelled.

5.3.3.2 Service B: Ringback

Ringback services allow a calling party, upon receipt of an engaged tone for a called party, to request that a call be automatically initiated to that callee once his/her current call has terminated. To realise this service the SCP signals the SSP to report when the callee's current call terminates, after which it signals the SSP to initiate a call between the caller and callee. Figure 5.5 illustrates the sequence of information flows involved in a successful session of the service. A successful session is assumed to generate 10 profit units.

The processing requirements, in terms of executed instructions, for messages arriving at the SCP are shown in Table 5.2. These values were used directly in the simulation model, whilst for the analytic model the mean message execution time was used for the calculation of the SCP service rate for Service B.[63]

In the simulation model the duration of the end-user delay phases are chosen at random using the distributions described in Table 5.3. In the analytic model these end-user delays are not modelled.

[62] Mean number of instructions = 240,000 / 3 = 80,000. $\mu_{SCP,SERVICE\_A}$ = 28,000,000 / 80,000 = 350.

[63] Mean number of instructions = 300,000 / 6 = 50,000. $\mu_{SCP,SERVICE\_B}$ = 28,000,000 / 50,000 = 560.

Table 5.2: Service B SCP processing requirements.

Arriving Signalling Message	Departing Signalling Message(s)	SCP Instructions
sB1	sB2	90,000
sB3	sB4	35,000
sB5	sB6	35,000
sB7	sB8, sB9	45,000
sB10	sB11, sB12	45,000
sB13	sB14, sB15	50,000
		300,000

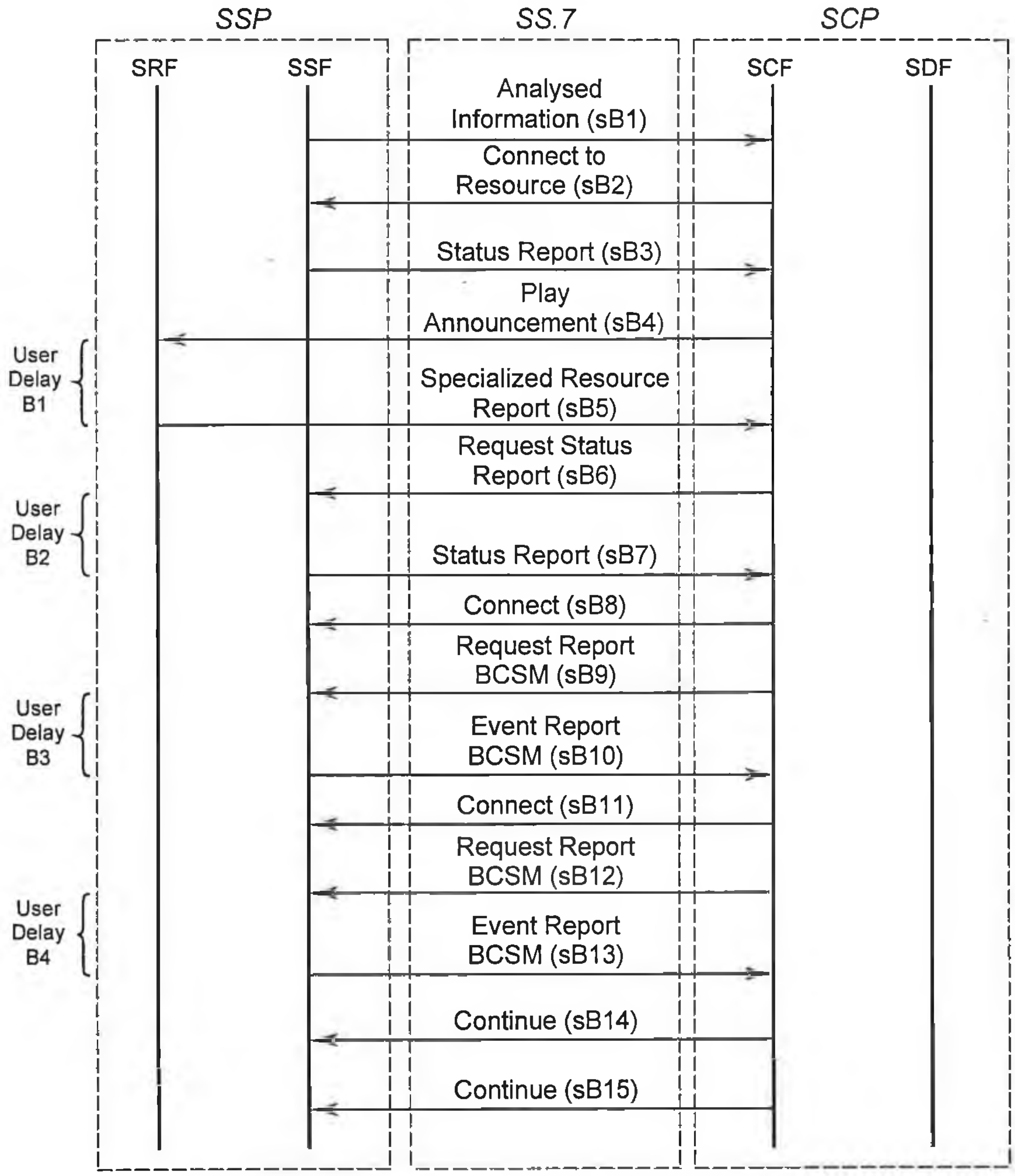


Figure 5.5: Information Flow sequence for Ringback service session.

Table 5.3: Service B end-user delay phase duration distributions.

End-User Delay	Description	Distribution
B1	Announcement	None (constant duration of 5s)
B2	Time to end of conversation	Negative exponential, mean 50s
B3	Callee telephone ringing	Negative exponential, mean 5s
B4	Caller telephone ringing	Negative exponential, mean 5s

5.3.3.3 Service C: Restricted Access Call Forwarding

Basic call forwarding allows an end-user to redirect incoming calls to a different number in a manner transparent to the calling party. A variation of this, in which the calling party must enter a specified PIN number before the call is forwarded to the other number, is modelled here. Figure 5.6 illustrates the sequence of information flows involved in a successful session of the service. A successful session is assumed to generate 1 profit unit.

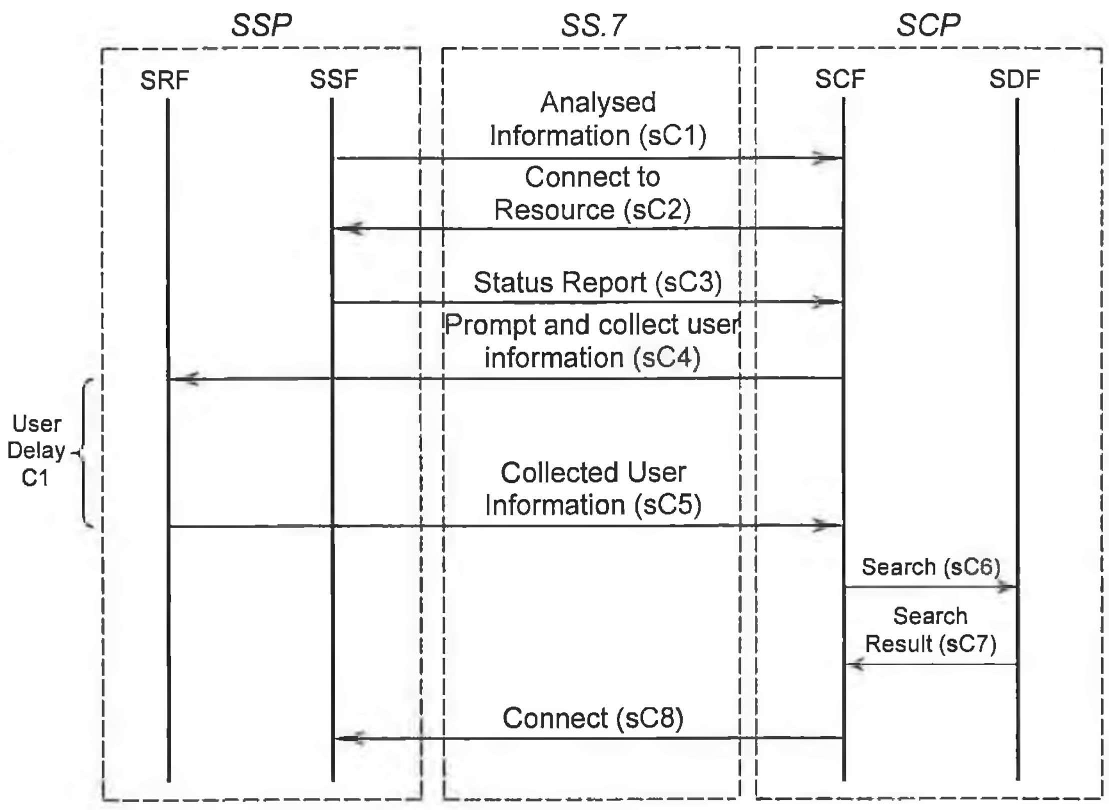

SSP
SS.7
SCP
SRF
SSF
SCF
SDF
Analysed Information (sC1)
Connect to Resource (sC2)
Status Report (sC3)
Prompt and collect user information (sC4)
User Delay C1
Collected User Information (sC5)
Search (sC6)
Search Result (sC7)
Connect (sC8)

Figure 5.6: Information Flow sequence for Restricted Access Call Forwarding service session.

The processing requirements, in terms of executed instructions, for messages arriving at the SCP are shown in Table 5.4. These values were used directly in the simulation model, whilst for the analytic model the mean message execution time was used for the calculation of the SCP service rate for Service C.[64]

[64] Mean number of instructions = 160,000 / 3 = 53,333. $\mu_{SCP\ SERVICE\ C}$ = 28,000,000 / 53,333 = 525.

Table 5.4: Service C SCP processing requirements.

Arriving Signalling Message	Departing Signalling Message(s)	SCP Instructions
sC1	sC2	70,000
sC3	sC4	25,000
sC5	sC8	65,000
		160,000

In the simulation model the duration of the end-user delay phase of the service, which relates to digit collection, is drawn at random from a negative exponential distribution with a mean of 5s. In the analytic model this end-user delay is not modelled.

5.4 ANALYSIS AND COMPARISON OF STRATEGY PERFORMANCE

This section presents and discusses the results of a comparative study of the ACG, OPT and TOKEN strategies, performed using analytic and simulation models. The primary aims of the study were to confirm that TOKEN is likely to perform significantly better than static strategies (such as ACG) and to explore its merits and disadvantages as compared to similar adaptive, revenue-maximising strategies (such as OPT). The analysis focuses solely on the effectiveness of TOKEN in controlling the load of a single SCP; strategy enhancements for realisation of a more network-oriented control scheme are dealt with in chapter 6.

The experiments carried out for the study were designed to allow investigation of both strategies' basic load control behaviour and their robustness to varying network/load conditions. The central goal of an IN SCP load control strategy is to maximise the service session completion rate during an overload situation whilst ensuring that the utilisation on the SCP stays close to but does not exceed a target level. In addition, SCP throughput should be maximised, that is the SCP should ideally not have to devote any processing time to the dropping of excessively delayed messages. It must also be robust to the volume of overload traffic, the sources of overload traffic, the number of traffic sources in the network and the supported service mix.

We employ five performance metrics in our analyses, namely: SCP utilisation, SCP throughput, SCP input queue size, service session completion rates and generated profit. Since all three strategies are founded on the principle of keeping SCP utilisation within defined limits and maximising throughput, the use of utilisation and throughput as performance metrics is natural. SCP input queue size gives an indication of the degree of overload and the response delays experiences by service end-users. Utilisation, throughput and queue size are equipment manufacturer-centric performance metrics; the remaining two metrics, completion rates and profit, give a more network operator-centric viewpoint. Session completion rates give an indirect measure of the number of users receiving adequate quality-of-service from the network during overload, whilst generated profit gives a measure of strategy performance regarding use of SCP resources in an optimal manner.

This section is structured as follows: §5.4.1 addresses basic strategy behaviour; §5.4.2 addresses selective throttling of service types based on profit; §5.4.3, §5.4.4 and §5.4.5 address robustness with regard to volume of overload traffic, source of overload traffic and the number of traffic sources respectively; finally, §5.4.6 address the operation of the TOKEN enhancement for control of legacy SSPs.

5.4.1 Basic Load Control Behaviour

In order to facilitate a direct comparison of basic load control behaviour the results presented here relate to the scenario where the arrival rates for service B and service C are set to zero; in effect only service A is supported by the network. This choice avoids issues relating to profit-based service differentiation and fairness, which are dealt with separately below. The arrival rates for service A are set so that during normal load conditions the SCP offered load is 0.35 Erlangs, whilst during overload the SCP first-offered load is 1.4 Erlangs. The volume of service requests is increased in one step at $t = 600s$ and decreased in one step at $t = 1200s$. In both normal and overload conditions service requests arrive at all SSPs in volumes proportional to SSP size and with exponentially distributed interarrival times. The control interval is $10s$ for all three strategies. End-user reattempt behaviour is not modelled initially. For ACG the target SCP utilisation range is set to $(0.8, 0.9)$ Erlangs, whilst for OPT and TOKEN the target SCP utilisation is $c = 0.9$ Erlangs.

Figure 5.7 and Figure 5.8 show the offered load and SCP throughput[65] for ACG, OPT and TOKEN, as obtained using the analytic and simulation models respectively. We see that ACG performs very poorly: throughput exceeds and falls short of the target value in an oscillatory fashion. As discussed in §4.4.1 this occurs because, when using the standard gap intervals, ACG initially greatly over-controls the load offered to the SCP. Sources then reduce traffic by too large an amount, and once SCPs have serviced the backlog of messages in their buffer the processor becomes starved, utilisation falls below the lower threshold, SSPs remove gapping, the SCP becomes overloaded and the cycle repeats.

OPT and TOKEN perform significantly better than ACG in controlling utilisation/throughput, since they dynamically compute load throttle parameters based on arrival rate estimations and these parameters will almost always be more accurate than those chosen from the small set of static gap intervals used by ACG. Both OPT and TOKEN keep the mean utilisation/throughput

[65] In this scenario no messages were dropped by the SCP for any of the three scenarios, hence for the simulation results SCP throughput is equivalent to SCP utilisation. For the analytic model message dropping is not modelled, hence throughput is always equivalent to utilisation.

very close to the target of 0.9 Erlangs during overload, however we see that with OPT there is oscillation around this value. TOKEN, on the other hand, controls the utilisation without ever exceeding the target of 0.9 Erlangs. This would be a significant advantage of TOKEN in the case where the target utilisation of the SCP must not be exceeded if system integrity is to be guaranteed. The drawback of TOKEN in this regard is that the utilisation falls a little short of the target, due to the operation of the percentage thinning based token spending process. The main reason for this difference in behaviour is that OPT employs relative load throttle parameters (PT coefficients), whereas TOKEN employs absolute parameters (token allocations). The difference in time taken for throughput to approach the target in the analytic and simulation models is due to the modelling of end-user delays in the simulation model.

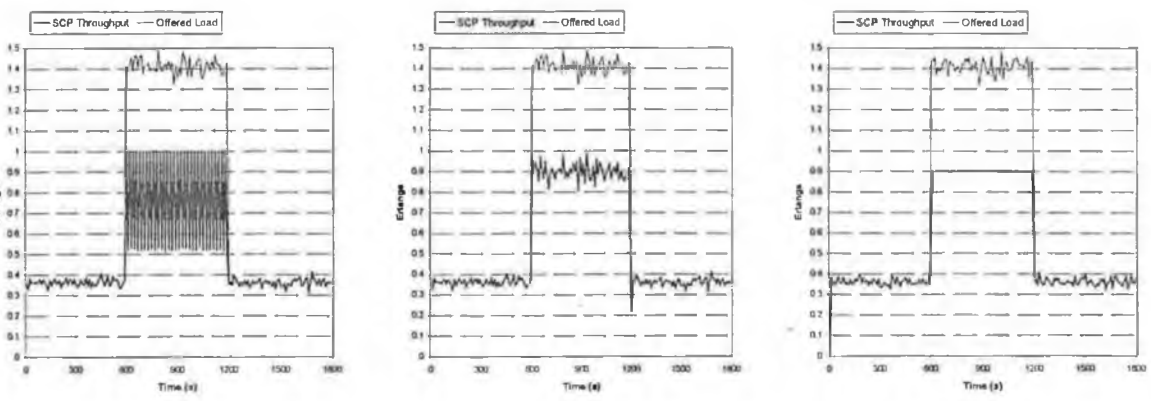


Figure 5.7: Analytic model. SCP offered load and utilisation. Left: ACG. Centre: OPT. Right: TOKEN.

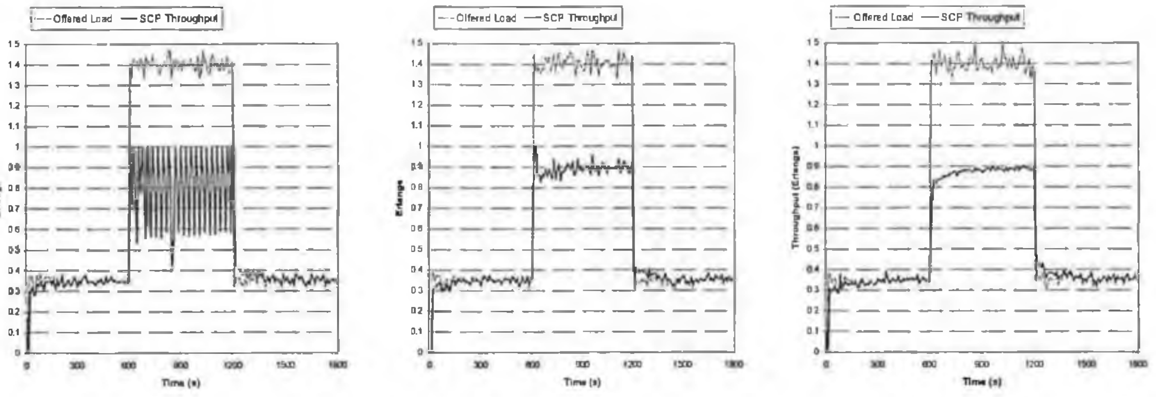


Figure 5.8: Simulation model. SCP offered load and utilisation. Left: ACG. Centre: OPT. Right: TOKEN.

The impact of processor utilisation oscillation on SCP input queue sizes can be seen in Figure 5.9[66] and Figure 5.10. The closer utilisation is to 1.0 the larger the mean queue size, which increases both the dropping probability for arriving messages and the mean queuing delay for messages not dropped. Clearly, message dropping and large delays degrade the quality of service received by service end-users. We see that with ACG the queue size oscillates in

[66] The decomposition method cannot be used to estimate queue size when a queue is saturated (utilisation equals one). With ACG the SCP is often fully utilised, hence we can not show queue size for ACG.

tandem with the oscillations of SCP utilisation. In this scenario the queue size does not grow to the stage where messages are dropped, however (as will be shown later) message dropping will occur when offered load is higher.

At the start of the overload OPT has not yet reacted to the increase in the volume of arriving requests, all of which are accepted. This means that the SCP utilisation quickly goes to and stays at 1 for the duration of the first control interval after overload onset. Therefore the analytic model is unable to calculate a mean queue size for this interval. In Figure 5.9 we have set the queue size for this interval to zero, this accounts for the obvious difference between the two OPT queue size graphs. As shown by the simulation model graph the queue size does not, in this scenario, grow to the stage where messages are dropped, however in more severe overload conditions this is a likely occurrence. For TOKEN we see that the queue size is well controlled, even at the start of the overload.

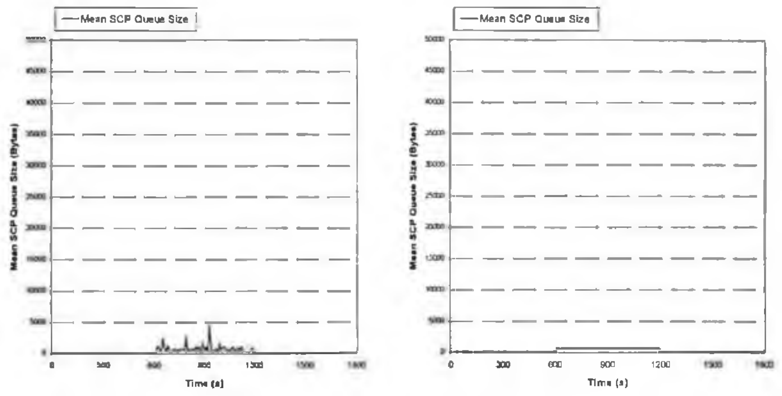


Figure 5.9: Analytic model. Mean SCP Queue Size. Centre: OPT. Right: TOKEN.

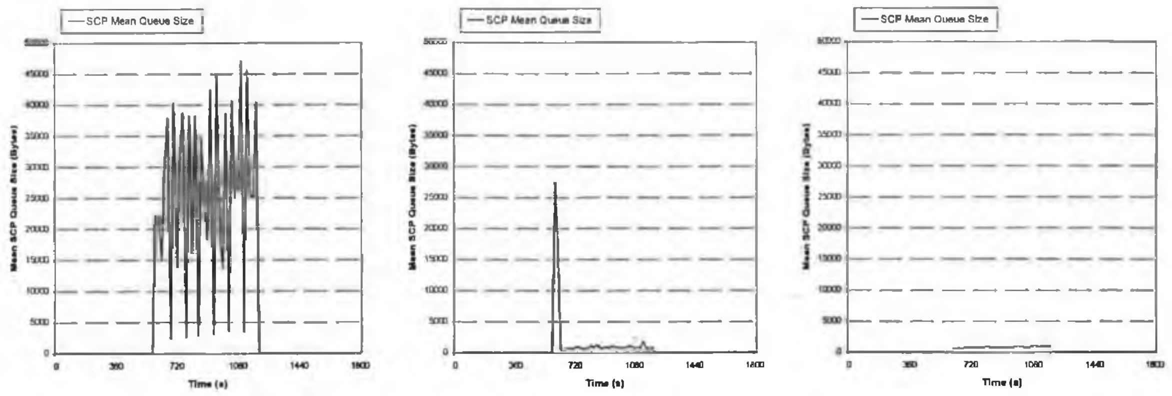


Figure 5.10: Simulation model. Mean SCP Queue Size. Left: ACG. Centre: OPT. Right: TOKEN.

Moving averages (window 60*s*) of mean service session completion rates for each of the three SSP types, as obtained using the simulation model, are shown in Figure 5.11. It is clear that both OPT and TOKEN demonstrate a significant improvement in completion rates over ACG. There are only small differences between the total completion rate performance of OPT and TOKEN, reflecting their similar performance with regard to controlling SCP throughput.

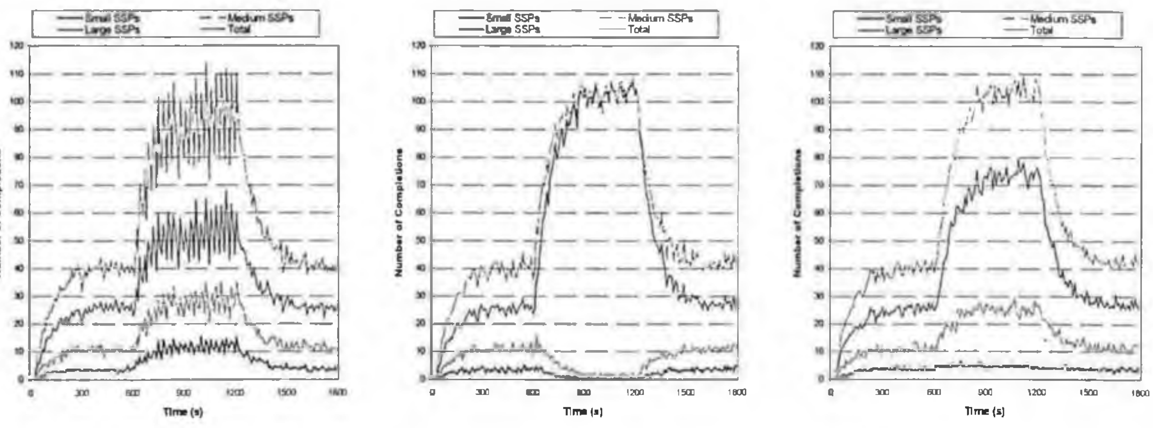


Figure 5.11: Simulation model. Per SSP size completion rates. Left: ACG. Centre: OPT. Right: TOKEN.

Regarding fairness between the treatment of SSP sizes it is evident from Figure 5.11 that ACG treats sources more equitably than either OPT or TOKEN. Table 5.5 shows the ratios of completion rates for small/medium/large sized SSPs during the overload. For complete fairness, in terms of throttling SSPs of different sizes in proportion to their contributions to overload traffic, these ratios should equal those of the SSP size ratio (*1 : 3 : 7*). For OPT we see almost all of the accepted requests originate at large SSPs, a result of the lack of an explicit constraint relating to fairness in the formulation of the optimisation problem. TOKEN also tends to favour larger SSPs, because there is a greater probability that tokens allocated to these will be used.[67] We note that a bias towards larger SSPs may be desirable from the network operator's viewpoint, since large SSPs tend to be associated with urban areas where there is a larger concentration of business customers; who are often regarded as being of higher importance than residential customers.

Table 5.5: Ratios of completion rates for small/medium/larger SSPs.

Strategy	Ratio (small : medium : large)
ACG	1 : 2.40 : 4.63
OPT	1 : 4.45 : 113.03
TOKEN	1 : 5.00 : 14.12

Our final performance metric is the rate at which profit is generated; Figure 5.12 shows these rates for the three strategies. As expected OPT and TOKEN perform very similarly, both generating a larger amount of profit than ACG.

[67] If fairness between SSPs sizes is a priority, it would be possible to modify SSP service profit values associated with requests arriving from different SSPs so that smaller SSPs receive more equitable treatment.

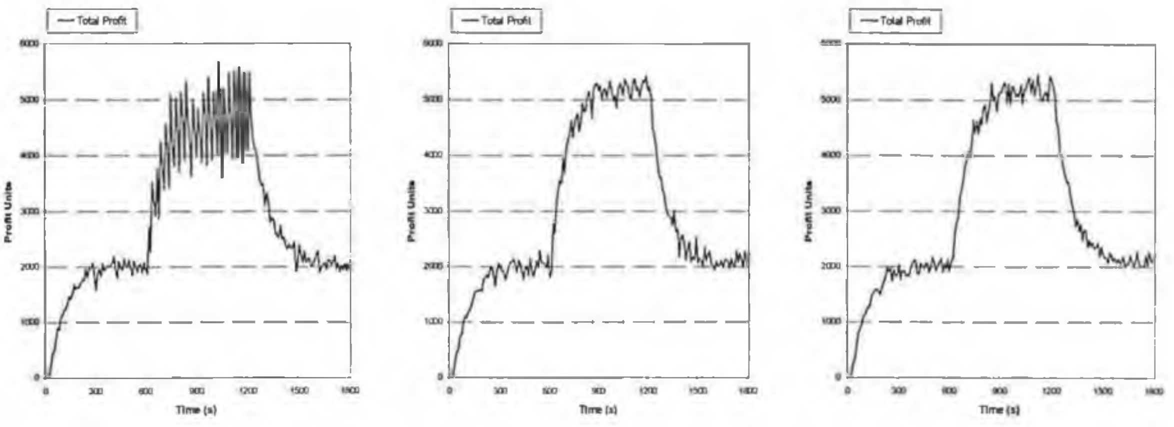


Figure 5.12: Simulation model. Profit generated per second. Left: ACG. Centre: OPT. Right: TOKEN.

For the results presented thus far end-user reattempt behaviour was omitted from the models. To show the impact of reattempts on the operation of the control strategies, reattempt behaviour was included in the simulation model in the manner described in §5.3.1.1. Reattempts cause the volume of load offered to the SCP to persist at a high level for some time after the abatement of first-offered overload traffic. Figure 5.13 shows that the volume of reattempt traffic is greater for ACG than OPT or TOKEN, the explanation being that ACG throttles a greater number of requests (it over-controls during an overload). The higher volume of traffic offered to the SCP, combined with ACG's oscillatory nature, means that a significant number of messages are dropped, leading to the decrease in SCP throughput seen in Figure 5.13. This in turn causes a decrease in the rate of completions for ACG, as evident from Figure 5.14. OPT and TOKEN continue to control SCP utilisation and queue size in a manner that maximises throughput and therefore both completion rates and generated profit.

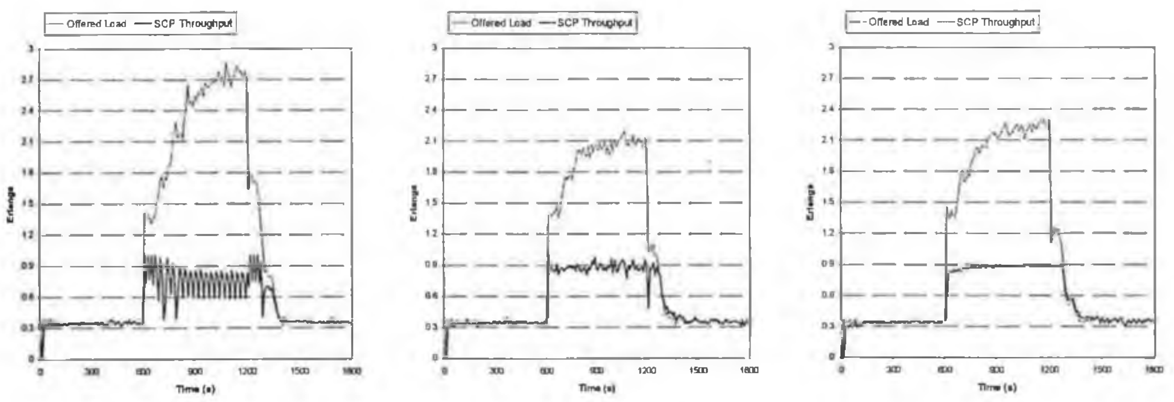


Figure 5.13: Simulation model, reattempts included. SCP offered load and throughput. Left: ACG. Centre: OPT. Right: TOKEN.

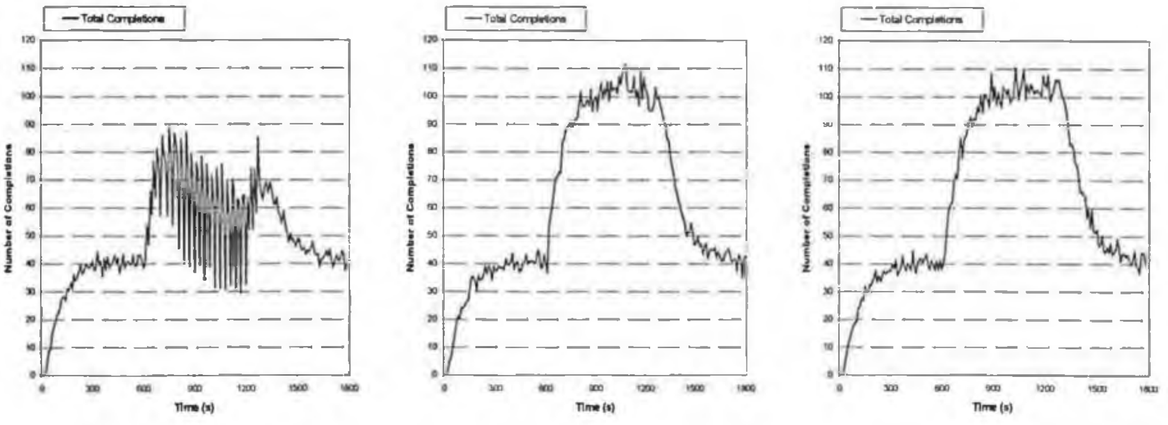


Figure 5.14: Simulation model, reattempts included. Overall completion rates. Left: ACG. Centre: OPT. Right: TOKEN.

5.4.2 Profit Maximisation

An ideal IN load control strategy will prevent SCP overload regardless of the number of supported services, by selectively throttling service requests at SSPs in a manner that maximises some global measure of network performance. We choose to maximise profit generated by successful service sessions, hence during overload requests for lower profit service types should be throttled in favour of requests for higher profit service types.

To analyse the performance of the three strategies in providing SCP protection in a heterogeneous service environment and simultaneously differentiating between service types for profit maximisation, we investigate a scenario in which the arrival rates for all three services are non-zero. In normal conditions service request arrival rates for all service types are set equal to each other and to values such that the mean SCP load is 0.35 Erlangs. Overload is caused by an increase in the number of arrivals of service A requests and results in a total mean SCP first-offered load of 1.4 Erlangs. In this situation the load control should accept as many service A sessions as possible, in preference to lower profit services (service C), but not in preference to higher profit services (service B), whose acceptance rate should remain unaffected.

Figure 5.15 and Figure 5.16 show the per service type SCP utilisation for ACG, OPT and TOKEN, as obtained using the analytic and simulation models respectively.[68] Note that with the simulation model end-user reattempt behaviour has been modelled. ACG applies the same gap intervals to all service types during an overload, with all service types being throttled regardless of their profit ratings. On the other hand both OPT and TOKEN exhibit the desired behaviour: service B acceptance rates remain unchanged during the overload whereas all service C requests are throttled.

[68] In these graphs each colour represents the contribution to SCP utilisation relating to processing of message types of the corresponding service. The contributions of the three services are represented as being 'on top of each other,' so the upper boundary of the shaded area is the actual SCP utilisation.

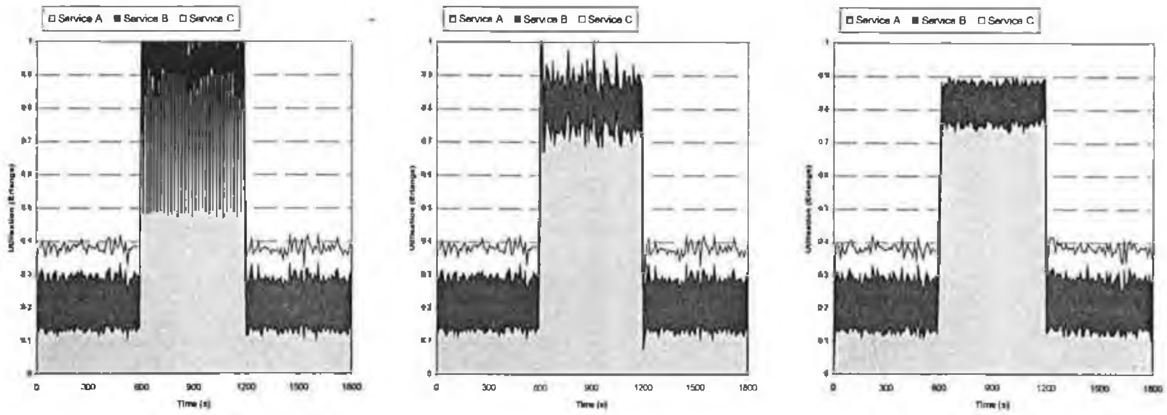


Figure 5.15: Analytic model. Per service type SCP utilisation. Left: ACG. Centre: OPT. Right: TOKEN.

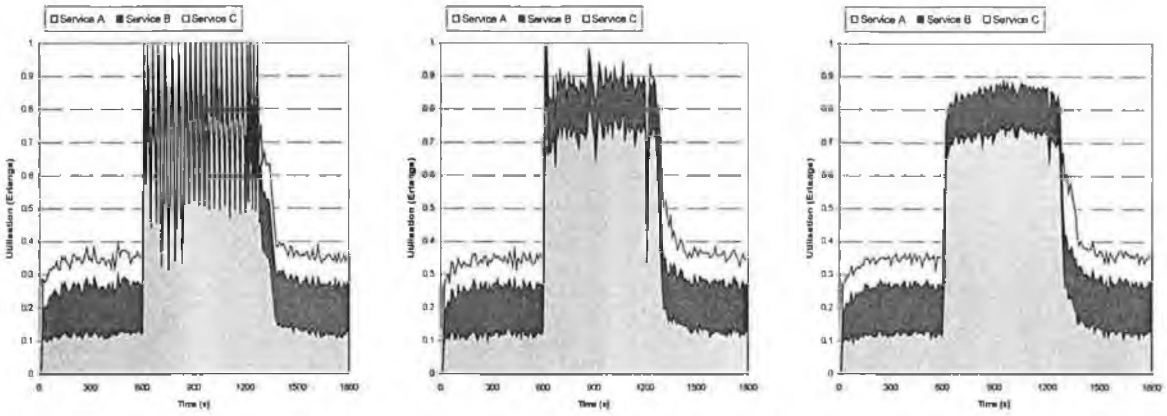


Figure 5.16: Simulation model. Per service type SCP utilisation. Left: ACG. Centre: OPT. Right: TOKEN.

Figure 5.17 and Figure 5.18 show the total completion rates and generated profit rates for the three strategies, as obtained using the simulation model. The completion graphs for OPT and TOKEN show a spike in the number of completions before the abatement of overload; this is due to reattempts for service C related to previous requests that were made just before the abatement of service A overload. The impact of service differentiation in control behaviour on generated profit is clearly illustrated by Figure 5.18, which shows that OPT and TOKEN are considerably better than ACG at maximising profit for the network operator.

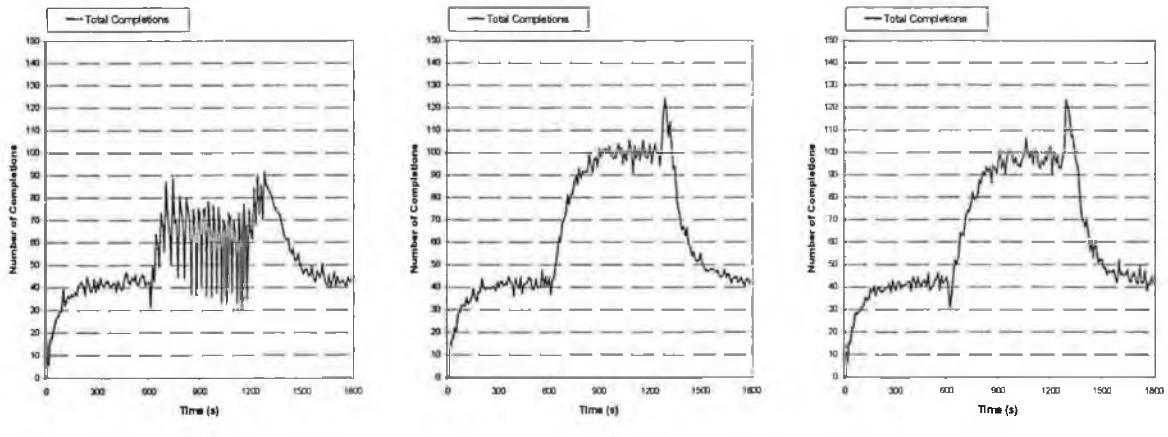


Figure 5.17: Simulation model. Total completion rates. Left: ACG. Centre: OPT. Right: TOKEN.

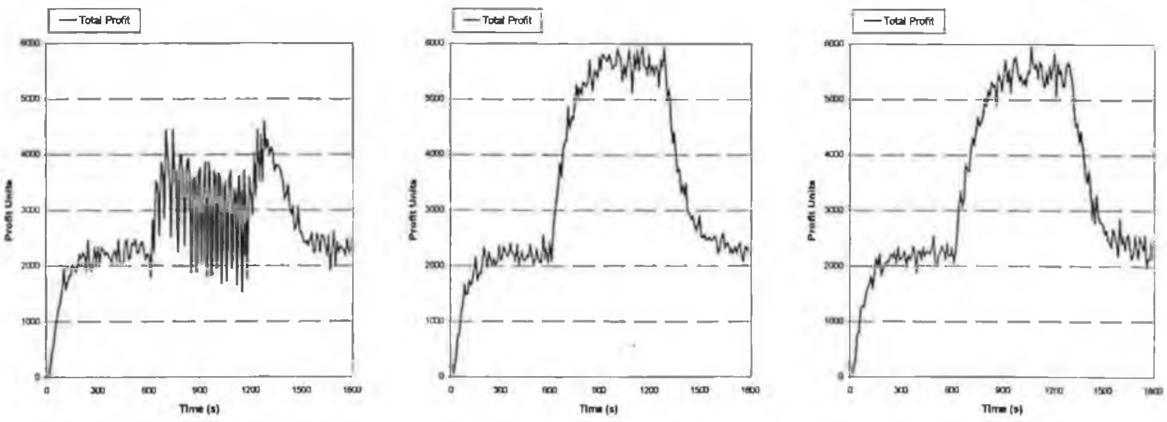


Figure 5.18: Simulation model. Profit generated per second. Left: ACG. Centre: OPT. Right: TOKEN.

5.4.3 Volume of Overload Traffic

Ideally an IN load control should protect the SCP from overload regardless of the volume of offered load. To investigate the robustness of ACG, OPT and TOKEN to the level of offered load fourteen scenarios, over which offered load during overload range from 0.35 to 4.0 Erlangs, are analysed. As with the preceding section the arrival rates are set equal for all three services during normal load and the overload is caused by a step increase in the volume of service A requests arriving at all SSPs. Reattempt behaviour is not included. Figure 5.19 shows the mean SCP throughput during overload for each strategy plotted against mean offered load, for the fourteen scenarios. The figure also shows the ideal characteristic: if offered load is less than the target utilisation, mean SCP throughput should equal the offered load; if it is greater mean SCP throughput should equal the target utilisation. The values used to generate these plots are listed in Table 5.6, page 153.[69]

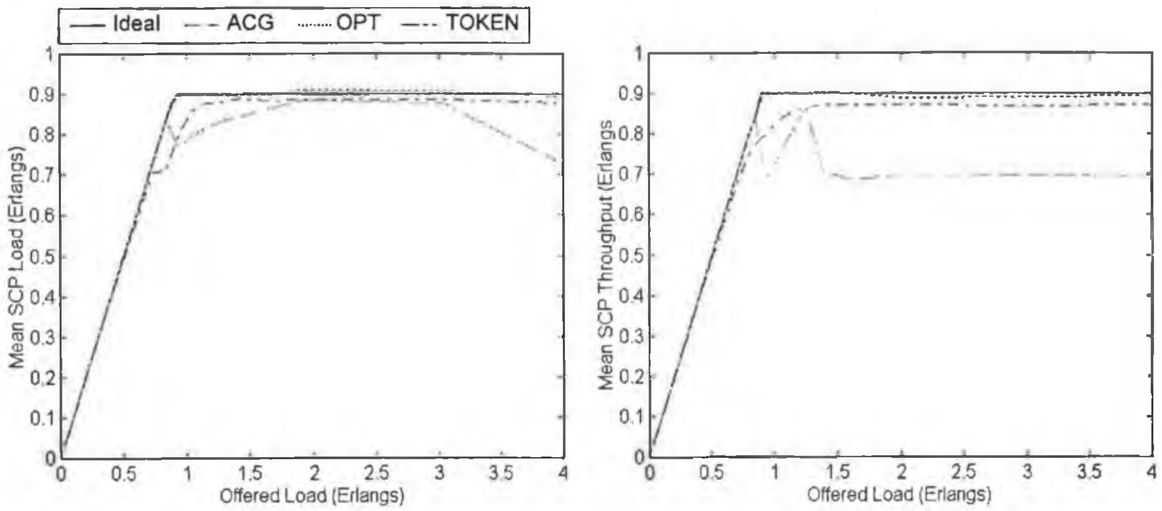


Figure 5.19: Comparison with ideal load control. Left: Analytic model. Right: Simulation model.

[69] These values are the mean of five corresponding values resulting from five simulation runs using different random number generator seeds.

Figure 5.19 shows that ACG is very sensitive to the volume of offered load, with the mean SCP throughput only approaching the target for a limited range of offered loads.[70] This is due to the use of a small number of static control parameters, which are unable to provide adequate control over a wide range of operating conditions. The difference in mean throughput for ACG between the analytic and simulation models is in part due to the simulation model including message dropping, which, as discussed in §5.4.1, causes a reduction in SCP throughput. The figure shows that OPT gives the best overall performance, with mean throughput closely tracking the ideal characteristic for all the offered load values (though, as noted previously, this is achieved at the expense of oscillation about the mean value).

TOKEN performs well at offered loads that are either much lower or much higher than the target utilisation. For offered loads less than approximately 0.7 Erlangs mean SCP utilisation matches offered load, whilst for offered loads greater than approximately 1.1 Erlangs mean SCP utilisation settles at a value that is a small percentage less than the target. For the latter the shortfall is related to token wastage related to the percentage thinning based token spending process and inaccuracies inherent in the arrival rate estimates. With respect to the latter we note that during overload SSPs receive slightly more tokens for service B than they bid for, however the number actually required may exceed or fall short of this number. Some SSPs will have a small surplus of service B tokens at the same time as others have an excess demand for them. Therefore a number of tokens are 'wasted,' leading to the mean SCP utilisation falling short of the target value. This drawback may not be very significant given that TOKEN does protect the SCP and, doing so without permitting the target SCP utilisation to be exceeded.

A more obvious drawback of TOKEN is its behaviour when the offered load is close to the target load. We see that in the region where offered load approaches 0.9 Erlangs SCP utilisation falls significantly below the offered load. Clearly it is undesirable for a load control strategy to throttle traffic when the SCP is not overloaded. The explanation for this behaviour is that where offered load is close to the target load larger bids, for higher profit services, receive more tokens than requested, at the expense of bids relating to lower profit services and/or smaller SSPs. Therefore, even though there are excess tokens in the network, a significant number of them are wasted and service requests are unnecessarily throttled elsewhere. However network operators are unlikely to dimension SCPs to operate this close to maximum utilisation during normal load conditions and so poor performance in this region may not be a critical drawback of the strategy.

[70] The strategies are also sensitive to other factors such as the number of traffic sources (*cf.* §5.4.5), so the plots shown here is valid only for the particular network and service mix studied.

Table 5.6: Data for ideal load comparison graphs.

Analytic Model				Simulation Model			
Offered	*ACG*	*OPT*	*TOKEN*	*Offered*	*ACG*	*OPT*	*TOKEN*
0.3823	0.3823	0.3823	0.3823	0.3479	0.3479	0.3479	0.3456
0.7309	0.7309	0.7309	0.7309	0.6903	0.6903	0.6902	0.6694
0.8329	0.8329	0.8329	0.7104	0.7960	0.7960	0.7957	0.7386
0.8829	0.8024	0.8778	0.7569	0.8497	0.8418	0.8458	0.7647
0.9325	0.7653	0.8970	0.7963	0.8973	0.7379	0.8863	0.7713
0.9842	0.7823	0.8993	0.8303	0.9388	0.6940	0.8959	0.7973
1.0343	0.7984	0.9002	0.8555	0.9966	0.7192	0.8979	0.8220
1.1320	0.8116	0.9006	0.8765	1.0961	0.7816	0.9018	0.8488
1.2844	0.8316	0.8988	0.8810	1.2508	0.8641	0.8997	0.8657
1.4355	0.8439	0.8971	0.8897	1.3889	0.7059	0.9034	0.8704
1.6335	0.8630	0.8970	0.8833	1.5899	0.6852	0.9000	0.8702
1.9166	0.8841	0.9088	0.8853	1.9976	0.6950	0.8896	0.8710
3.0341	0.8763	0.9060	0.8868	3.0008	0.6982	0.8921	0.8692
4.035	0.7345	0.9013	0.8842	4.0014	0.6938	0.8953	0.8702

Figure 5.20 shows the moving average (window 100s) of mean completion rates for the three strategies for three scenarios in which the mean first-offered SCP load is 1.1, 1.4 and 2.5 Erlangs respectively. The results were obtained using the simulation model with reattempt behaviour included. These graphs again demonstrate the insensitivity of ACG to the volume of offered load. For OPT and TOKEN the differences in completion rate abatement times result from the higher volumes of reattempts that occur at higher first-offered load levels.

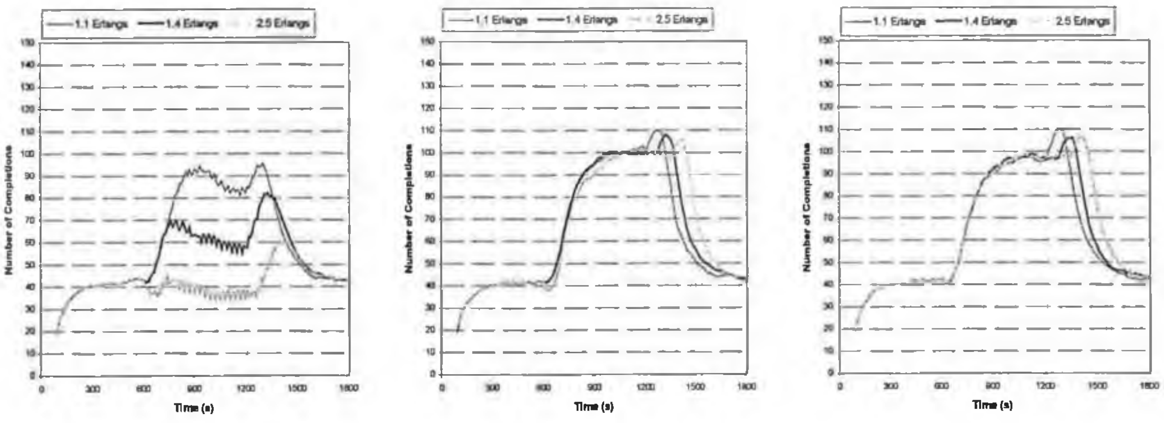


Figure 5.20: Simulation model. Overall completion rates for three offered load volumes. Left: ACG. Centre: OPT. Right: TOKEN.

5.4.4 Source of Overload Traffic

In the simulation scenarios discussed above the overload of service A traffic occurred at all SSPs in the network. As well as handling these *global overloads* an ideal IN load control should be able to handle overloads resulting from increases in arrival rates at a subset of the SSPs; termed a *focussed overload.* In such circumstances it seems reasonable that only those SSPs at which the overload occurs are forced to throttle requests for the overloading service. Note that the more important goal of maximising profit means that lower profit, non-overloading services should be throttled at all SSPs and no higher profit services should be throttled anywhere. To investigate strategy performance during a focussed overload we employ the same load profile as the last section with one main difference: service A overloading traffic originates from only half the SSPs (specifically half of the number of each of the three groups of differently sized SSP).

Figure 5.21 and Figure 5.22 show the SCP offered load and utilisation, as obtained using the analytic and simulation models respectively. As expected ACG is seen to perform as poorly in terms of controlling SCP utilisation as it did in the global overload case. OPT and TOKEN perform well, keeping SCP throughput at roughly the same levels as they did for the global overload case. Figure 5.23 compares the overall completion rates for the three strategies for the focussed and global overload scenarios; we see that the strategies achieve approximately the same performance in both cases.

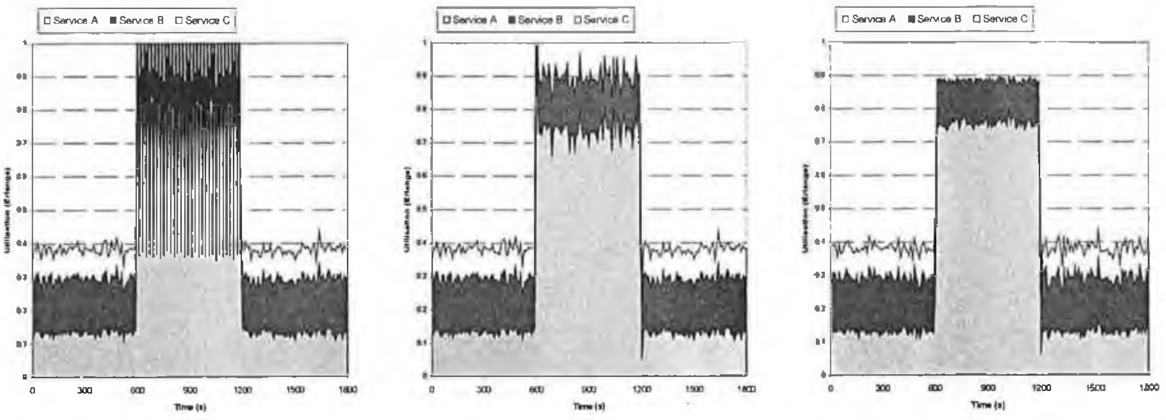


Figure 5.21: Analytic model. SCP offered load and utilisation. Left: ACG. Centre: OPT. Right: TOKEN.

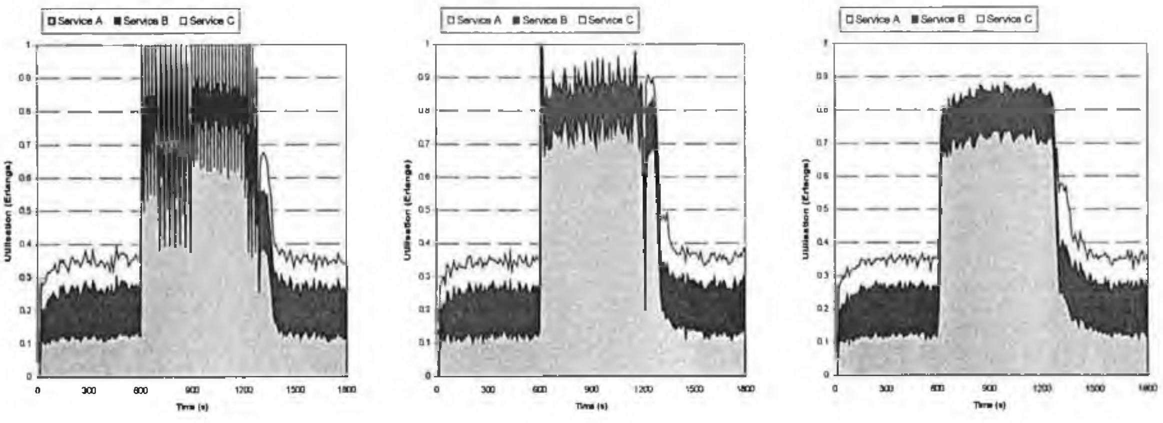


Figure 5.22: Simulation model, reattempts included. SCP offered load and utilisation. Left: ACG. Centre: OPT. Right: TOKEN.

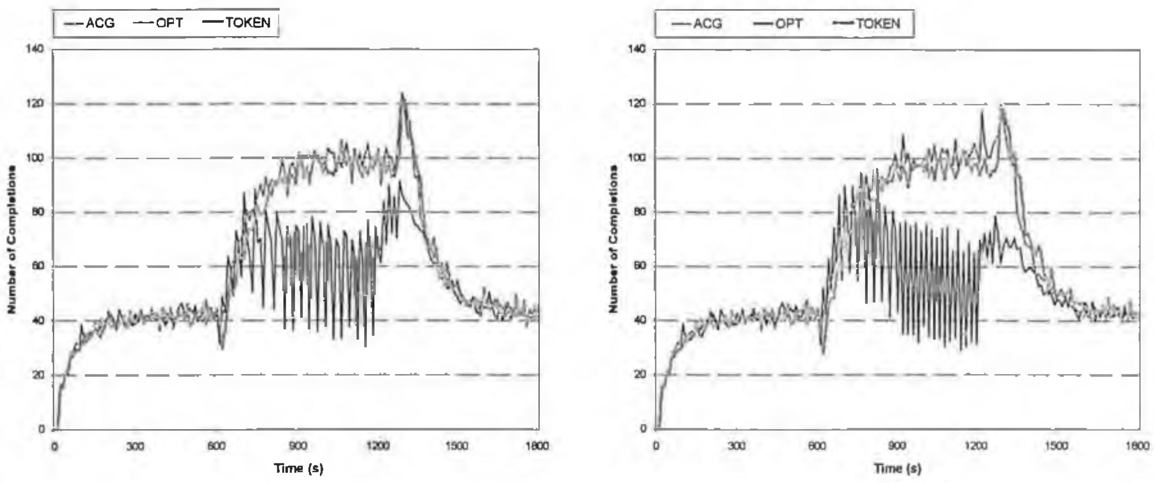


Figure 5.23: Simulation model. Overall completion rates. Left: Global overload. Right: Focussed overload.

To compare the fairness of the strategies in terms of throttling only requests originating at 'culpable' SSPs, Figure 5.24 shows the moving average (window 200*s*) of service A completion rates for a small, a medium and a large SSP which do not contribute to the overload. For these SSPs the rate of acceptance of service A requests should not be affected during the overload (at least with this definition of fairness). We see that ACG and TOKEN go further towards attaining this goal than does OPT. For OPT the reduction in completion rates is very significant and is a result of the linear programming problem not explicitly incorporating any fairness related constraints. With ACG the reduction results from the same gaps being put in place at all SSPs regardless of their contribution to the overload. Finally, for TOKEN the degradation is less pronounced and, as discussed previously, is a result of the token generation algorithm's tendency to allocate tokens to bids with higher arrival rate estimates.

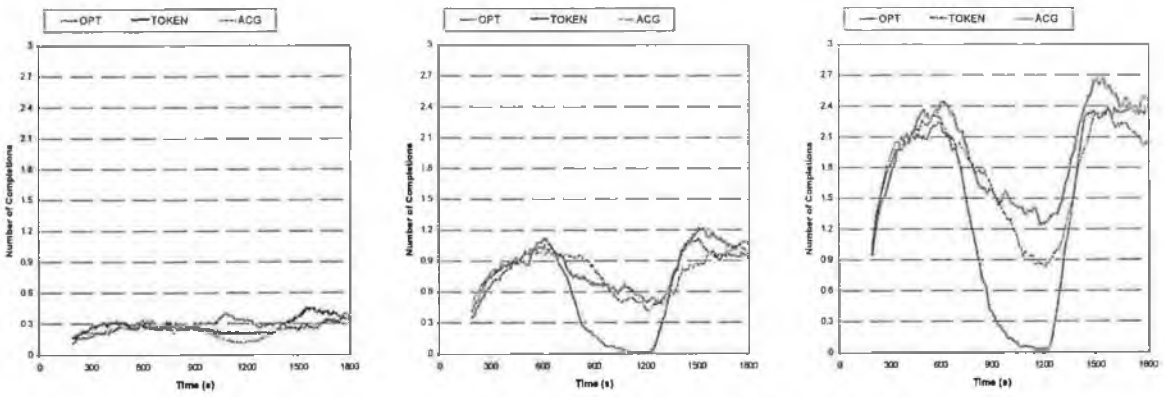


Figure 5.24: Simulation model. Focussed overload. Completion rates for non-overloading SSPs. Left: Small SSP. Centre: Medium SSP. Right: Large SSP.

5.4.5 Number of Traffic Sources

We now address the operation of ACG, OPT and TOKEN in networks consisting of a larger number of SSPs. In scenarios investigated above the network contained 12 SSPs, which meant that the load control strategy had to control 36 sources of service requests. Intuitively, we can expect that OPT and TOKEN will have greater difficulty in computing optimal values for control parameters in a network containing a larger number of sources. To investigate this, we analyse strategy operation in networks containing 24 and 96 SSPs, both with equal numbers of small, medium and large SSPs. A global overload of service A is analysed, with arrival rates being set to give an SCP first-offered load of 0.35 Erlangs in normal conditions and 1.4 Erlangs during high load conditions.

Figure 5.25 and Figure 5.26 show the SCP offered load and utilisation for a network with 24 SSPs, whilst Figure 5.27 and Figure 5.28 show the same for a network with 96 SSPs We see that for the network with 24 SSPs ACG actually performs better than in previous scenarios, specifically throughput is kept closer to the target and the frequency of oscillation is reduced. However, in the network with 96 SSPs we see worse behaviour than in the 12 SSP scenario: although there are few oscillations load is, for the most part, kept well below the target.

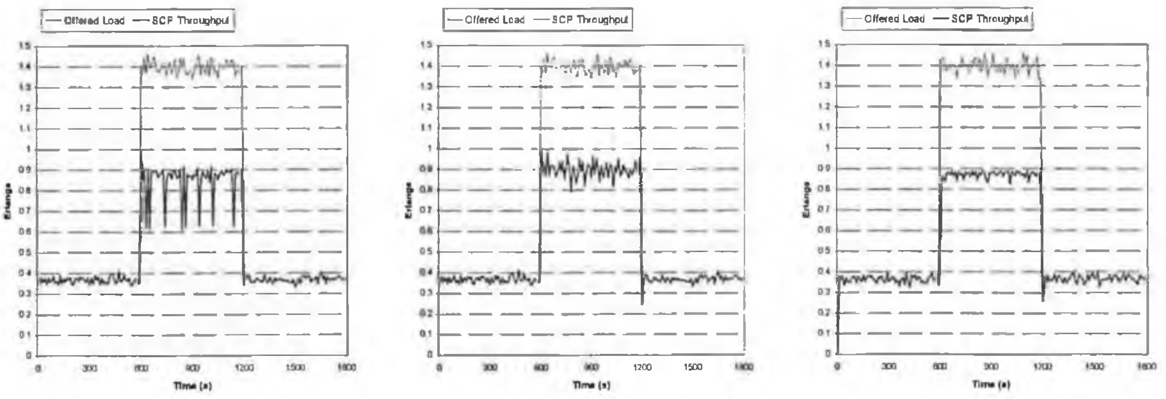


Figure 5.25: Analytic model. Network with $K = 24$ SSPs. SCP offered load and throughput. Left: ACG. Centre: OPT. Right: TOKEN.

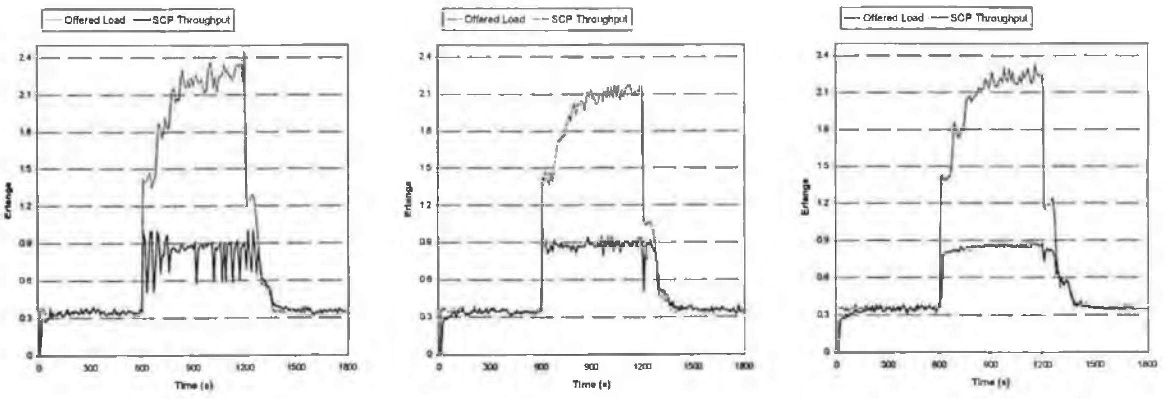


Figure 5.26: Simulation model. Network with $K = 24$ SSPs. SCP offered load and throughput. Left: ACG. Centre: OPT. Right: TOKEN.

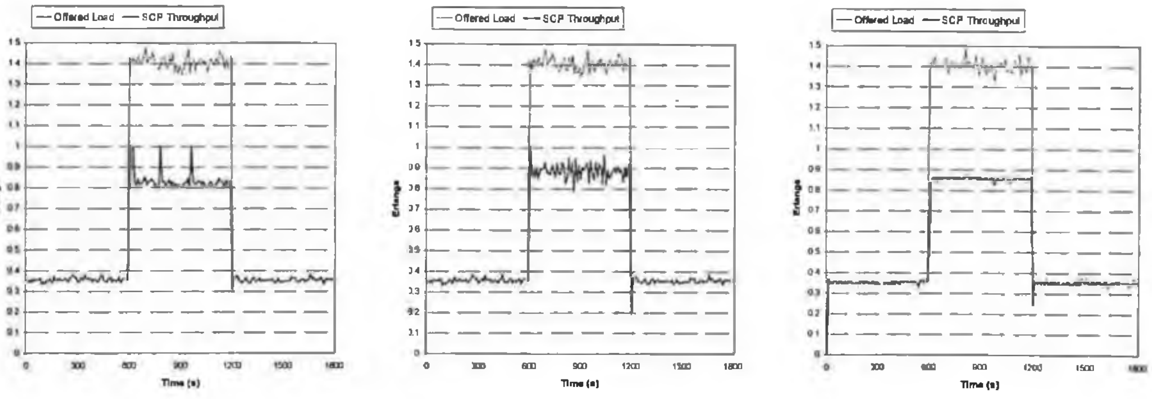


Figure 5.27: Analytic model. Network with $K = 96$ SSPs. SCP offered load and throughput. Left: ACG. Centre: OPT. Right: TOKEN.

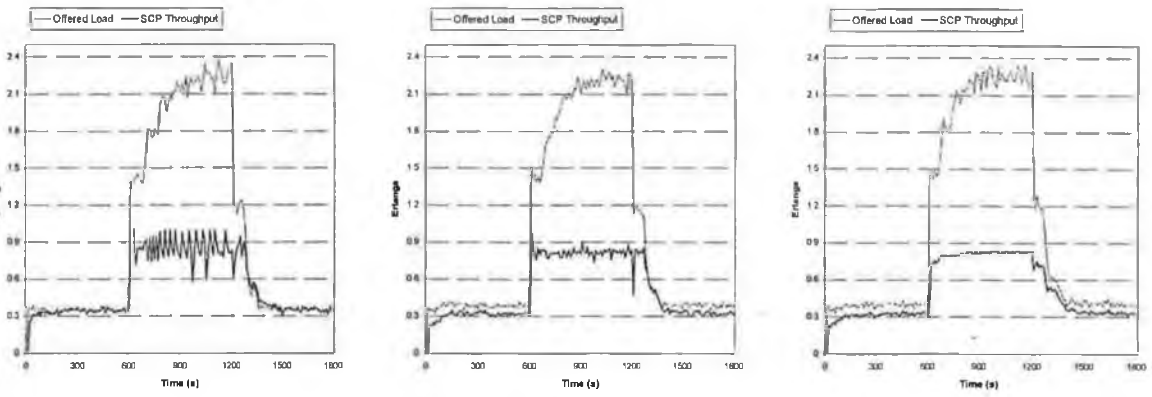


Figure 5.28: Simulation model. Network with $K = 96$ SSPs. SCP offered load and throughput. Left: ACG. Centre: OPT. Right: TOKEN.

Both OPT and TOKEN show some degradation in throughput during overload as compared to the 12 SSP scenario. The degree of degradation is quantified by Table 5.7, which shows the mean SCP throughputs during overload. However both strategies do still provide effective protection against SCP overload. Of greater concern is the fact that, for the 96 SSP case, the simulation results show SCP throughput falls below offered load in normal load conditions, which indicates that requests are being rejected unnecessarily. This is due to the smaller absolute numbers of service requests arriving at the small/medium SSPs in the 96 SSP case as compared to the 12 SSP case; often during overload no requests for one or more service types arrive at many of the SSPs during a control interval. This means that many of the arrival estimates inputted into OPT/TOKEN are zero, which hampers the algorithms ability to calculate optimal solutions. This limitation should be taken into account when applying either strategy.

Table 5.7: Mean SCP throughput during overload for varying network sizes.

Number of SSPs	Analytic Model			Simulation Model		
	ACG	*OPT*	*TOKEN*	*ACG*	*OPT*	*TOKEN*
12	0.8439	0.8971	0.8842	0.7418	0.8826	0.8561
24	0.8470	0.8990	0.8716	0.8253	0.8774	0.8445
96	0.8347	0.8864	0.8565	0.8550	0.8146	0.8119

5.4.6 Operation of Token-based Strategy Enhancement for Control of Legacy SSPs

We now analyse the performance of the TOKEN strategy enhancement described in §5.1.3, which allows conversion of token allocations into gap intervals that can be used for load throttling by legacy SSPs. We simulate a network containing a single SCP and 12 legacy SSPs, where, as before, the offered SCP load is 0.35 Erlangs in normal conditions and 1.4 Erlangs during overload. Figure 5.29 shows the SCP offered load, throughput, mean queue size and overall completion rates. We see that, for the most part, TOKEN succeeds in providing near-ideal load control. The exception is at the onset of the overload where the SCP utilisation goes to 1.0 because no gapping is active at the SSPs. As expected these results are very similar to the results acquired for OPT for the same load conditions (see §5.4.2).

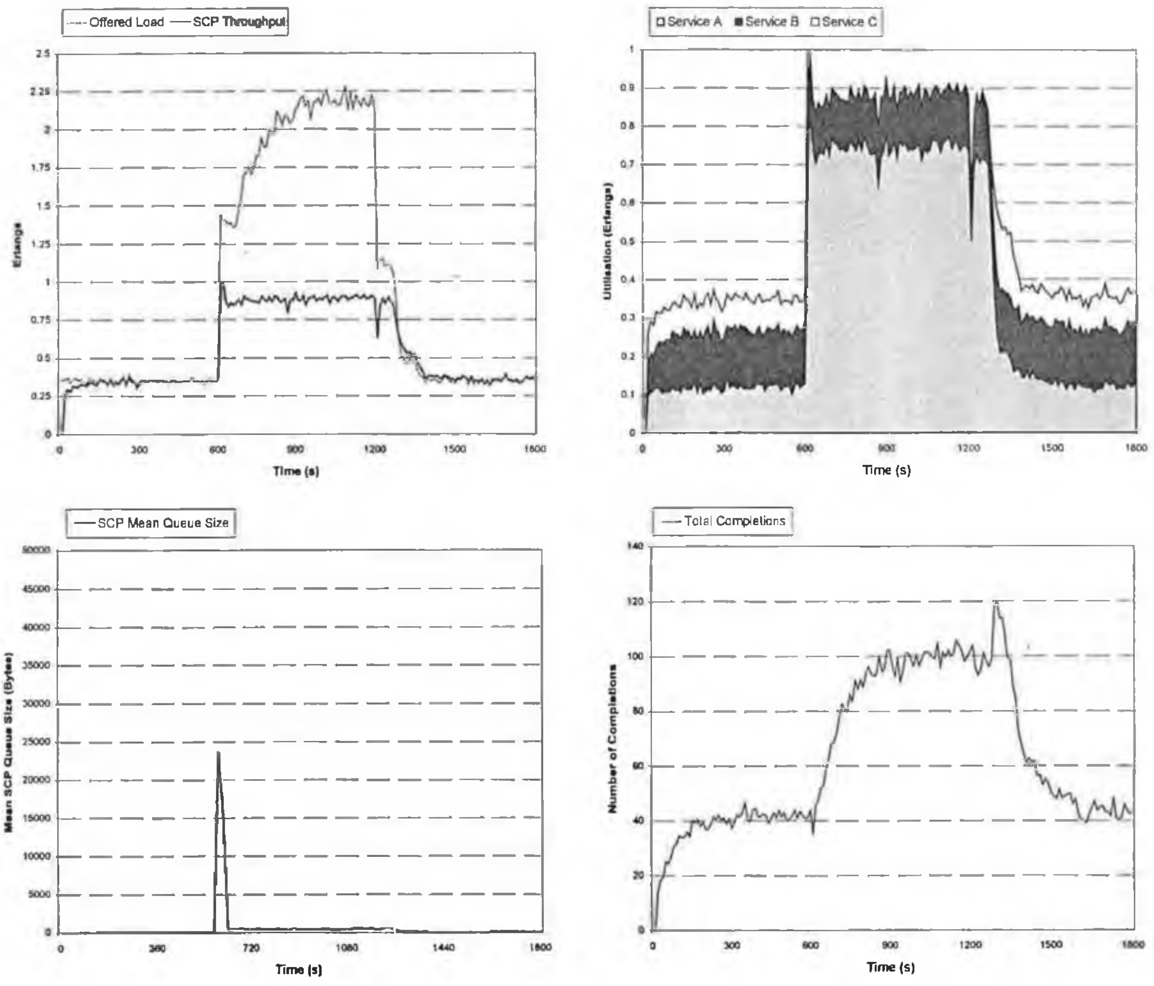


Figure 5.29: Simulation model, 12 legacy SSPs. Top left: SCP offered load and throughput. Top right: Per service type SCP utilisation. Bottom left: Mean SCP queue size. Bottom right: Overall Completion rates.

5.5 SUMMARY AND CONCLUSIONS

In this chapter we have presented an IN SCP load control strategy in which access of IN service requests to the SCP is controlled by means of tokens. When an arriving request is accepted it consumes a single token (of the appropriate type), granting it use of all the SCP resources required over the duration of the service session. Through matching the number of tokens generated to the number of service sessions that can be handled by the SCP, the strategy ensures that SCP overload does not occur. Furthermore, the strategy controls the manner in which tokens are allocated between different service types and traffic sources so as to maximise the profit generated during overload.

The likely performance of the TOKEN strategy was analysed by comparing it with both ACG and Lodge's optimisation strategy, by means of analytic/simulation models. Performance of the three strategies was compared in terms of basic load control behaviour, profit maximisation and robustness to factors such as volume of overload traffic and number of traffic sources. The results of this comparison are summarised in Table 5.9 (page 160).

There are two main conclusions to be drawn from the comparative analysis of the three strategies. Firstly, ACG has major failings as an IN load control: it is very prone to oscillations and not robust to changes in operating conditions due to its reliance on the use of a relatively small number of standard gap intervals. More optimal sets of gap intervals could be applied for a particular IN, however calculation of suitable gap intervals is non-trivial and would have to be repeated every time there is a change in SCP resources or supported service mix. Secondly, OPT and TOKEN operate significantly better than ACG and deliver very similar performance under all the scenarios studied. This is due mainly to the fact that both strategies dynamically compute control parameters with the goal of maximising profit whilst protecting the SCP from overload.

The conclusion that OPT and TOKEN give similar performance is not unexpected, since the market-based token generation algorithm is in effect a form of optimisation process.[71] The relatively minor differences in their performance stems from the fact that OPT employs relative control parameters (PT coefficients) whilst TOKEN employs absolute control parameters (token allocations). From the load control performance viewpoint the main advantage of TOKEN over OPT is that it ensures SCP utilisation never exceeds the target utilisation, whereas with OPT there will be oscillation around the target value. The main advantage of OPT over TOKEN is that it keeps mean utilisation closer to target utilisation for all offered loads, especially where

[71] [Ygge, 1998] proves that a very general class of optimisation problems can be transformed into analogous market-based control problems, the general equilibrium solutions to which are the solutions to the corresponding optimisation problems.

offered load is close to the target, a region where TOKEN performs badly. We note that it would be straightforward to convert the PT coefficients generated by OPT into equivalent token allocations and vice versa. Therefore we believe that there are no firm arguments (from the load control viewpoint) for recommending the use of one of these strategies over the other.

Table 5.8: Summary of results of comparative analysis of ACG, OPT and TOKEN.

Attribute	ACG	OPT	TOKEN
Staying close to but under SCP processor target utilisation during overload.	*NO* Utilisation oscillates due to use of standard gap intervals.	*PARTIAL* Some oscillation around target utilisation due to use of PT as load throttle.	*YES* Utilisation slightly below target. Use of tokens ensures target not exceeded.
Maximisation of SCP Throughput	*NO* Oscillations may cause message dropping due to queue size build-up.	*YES* Message dropping or discarding due to queue size build-up unlikely.	*YES* Message dropping or discarding due to queue size build-up unlikely.
Profit-maximisation.	*NO* No concept of profit-based service type differentiation.	*YES* Objective function seeks to maximise profit.	*YES* Tokens generated so as to maximise profit.
Maximisation of service session completion rates.	*NO* SCP under-utilisation means completion rates not maximised.	*YES* Within bounds set by profit maximisation constraint.	*YES* Within bounds set by profit maximisation constraint.
Minimisation of SCP queue size and connection delays.	*NO* SCP queue size oscillates leading to large delays for some end-users.	*PARTIAL* SCP queue size may grow at the start of a severe overload.	*YES* Significant oscillation in SCP queue size unlikely.
Immediate reaction to overload onset.	*NO* Up to one control interval before gapping put in place.	*NO* Up to one control interval before PT throttle in place.	*YES* Volume of accepted requests limited by token allocation size.
Robustness to offered load level.	*NO* Standard gap intervals unsuited to a range of offered load.	*YES* Mean SCP utilisation close to target regardless of offered load.	*PARTIAL* Utilisation close to target except where offered load is close to target value.
Scalability to number of traffic sources in the network.	*NO* Standards gap intervals unsuited to a range of network sizes.	*PARTIAL* Poor performance in large networks where arrival rates low for some SSPs.	*PARTIAL* Poor performance in large networks where arrival rates low for some SSPs.
Fairness in relation to traffic source sizes.	*NO* Same gap intervals put in place at all SSPs.	*NO* Large SSPs strongly favoured.	*NO* Token generation process favours larger SSPs.
Selective source throttling during focussed overloads.	*NO* Same gap intervals put in place at all SSPs.	*NO* Non-overloading SSPs experience severe throttling.	*PARTIAL* Non-overloading SSPs experience some throttling.
Compatibility with legacy systems.	—	*YES* Only requires support for PT load throttle	*YES* Token allocations can be converted to gap intervals.
Simple control implementation.	*YES* ACG is a table-driven control.	*NO* Requires solution of linear programming problems.	*NO* Requires relatively complex token generation algorithm and throttle.

It may be argued that the use of tokens for load control is a more intuitively understandable approach than the use of 'abstract' mathematical optimisation. Hence there may be non-technical reasons to adopt TOKEN over OPT. For example, use of tokens would provide a more direct linkage between load control and service level agreements (SLAs) agreed between the network operator and a customer. In such agreements the operator may guarantee, at the risk of sever financial penalties, that a minimum number of offered service requests will be accepted per unit time regardless of overall traffic levels in the network. This information could be inputted into a modified token generation process, which would impose minimum allocations for that service type, regardless of demands for other services.

This chapter has focussed solely on strategies for the control of load offered to a single SCP, the OPT and TOKEN strategies having been shown to provide excellent load control performance in this context. However, the overriding focus of this thesis is to explore the area of network-oriented, rather than node-oriented, IN load control. In the next chapter we attempt to realise this aim, by enhancing the TOKEN strategy to provide profit-maximising load control in networks containing multiple, possibly heterogeneous resource types.

CHAPTER 6

EXTENSION OF TOKEN-BASED STRATEGY TO ACHIEVE NETWORK-ORIENTED SS.7/IN LOAD CONTROL

At the start of this thesis we argued that the increasingly dynamic nature of IN networks means that over-dimensioning, static routing and traditional, node-based load control can no longer provide adequate performance management in today's INs. In this chapter we enhance the token-based load control strategy, introduced in chapter 5, to provide network-oriented dynamic routing and load control. The enhanced strategy allows for the dynamic routing of service requests to multiple SCPs whilst ensuring that neither SCPs, other IN resources, nor the SS.7 network overload. Its dynamic nature facilitates automatic adaptation to changes in traffic patterns, changes in the supported service mix, the introduction/withdrawal of network resources and network failures.

The chapter is structured in three parts. §6.1 shows how the token-based strategy can be enhanced to provide dynamic routing and load control in a multiple-SCP IN. §6.2 shows how the token-based strategy can be extended to provide load control for multiple resource types, using as an example the situation where the network contains multiple SDPs that must be protected from overload. Finally, §6.3 addresses how the token-based strategy can be enhanced to provide protection against SS.7 resource overloads: strategy enhancements that encourage SSPs to use those SCPs in closest proximity and which use delay measurements to detect SS.7 network overload are proposed and analysed.

6.1 TOKEN-BASED LOAD CONTROL OF A MULTIPLE-SCP IN

Original deployments of IN systems provided a relatively small number of services, whose execution and management had previously been directly supported by network switches. Because these services were small in number and had well-understood traffic and resource usage characteristics, it was possible for network operators to realise them using a small number of SCPs and configure each SSP to statically route all its initial service session messages to a single SCP thus resulting in a logical partitioning of the IN. This approach has the advantage that focussed overloads (within a single 'partition') are contained – they do not lead to overload of other SCPs. However, static partitioning also meant that SCPs could overload in circumstances where there was sufficient residual processing capacity available in other SCPs.

Logical partitioning of IN networks through static routing of initial service session messages remains standard practice for many network operators. However, rapid introduction of a wider range of complex services is necessitating deployment of ever-larger numbers of SCPs, which often only support a subset of the network's IN service portfolio. In this type of environment dynamic routing, combined with effective load control, can offer a means of ensuring that resources are utilised optimally. In this section we show how the token-based load control strategy can be extended to facilitate dynamic routing of initial service sessions messages between multiple SCPs in a manner providing protection against overload.

§6.1.1 presents simulation results[72] demonstrating how static routing can lead to sub-optimal resource usage in certain overload conditions. Subsequently, §6.1.2 specifies how the token-based strategy can be extended to provide centralised control of multiple SCP resources and §6.1.3 shows how a similar degree of control can be provided in a more distributed fashion. Finally, §6.1.4 presents a hierarchical distributed version of the strategy, suitable for application in large-scale INs.

6.1.1 Demonstration of the Limitations of Static Routing

The network scenario illustrated in Figure 6.1 was used to investigate the impact of SSPs statically routing IN initial service session messages to a single SCP. The network contains two identical SCPs, each serving six SSPs. We simulate a focussed overload in which the arrival rate of service A requests increases only at SSPs 1-6, which initiate sessions with SCP 1. During overload the first-offered load to SCP 1 is 1.4 Erlangs; end-user reattempt behaviour is modelled. Load control is achieved using the token-based load control strategy, as described in chapter 5.

The per service type SCP throughputs for SCP 1 and SCP 2 are shown in Figure 6.2. During the overload SSPs 1-6 are forced to reject service requests even though SCP 2 is running at normal load and has available processing capacity. Unnecessary throttling of service requests clearly reduces the amount of profit generated, so there is a strong case for the introduction of a more dynamic routing scheme. Of course SCPs must still be protected from overload, so we argue that a dynamic routing scheme should only be realised as an integrated component of a load control strategy.

[72] All simulation results presented in §6.1 were obtained using a simulation model employing the same node and service models used for chapter 5.

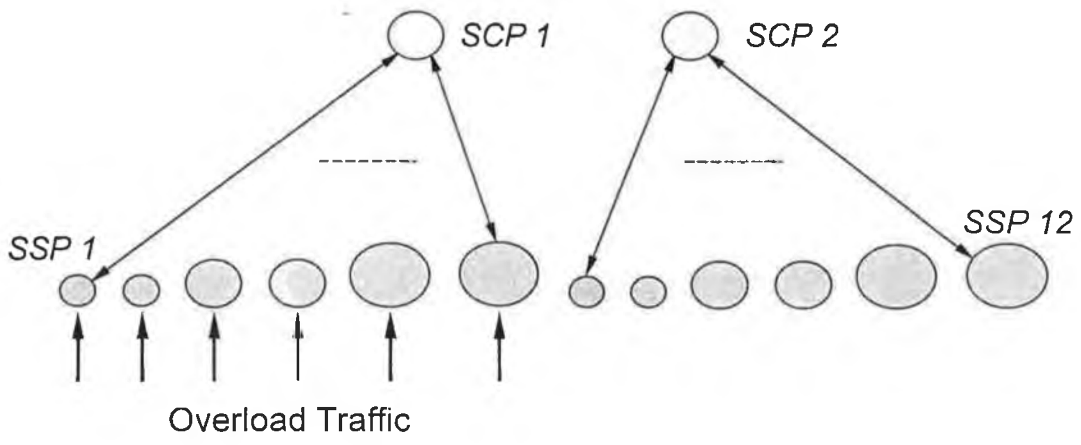


Figure 6.1: Dual-SCP network with static routing, focussed overload.

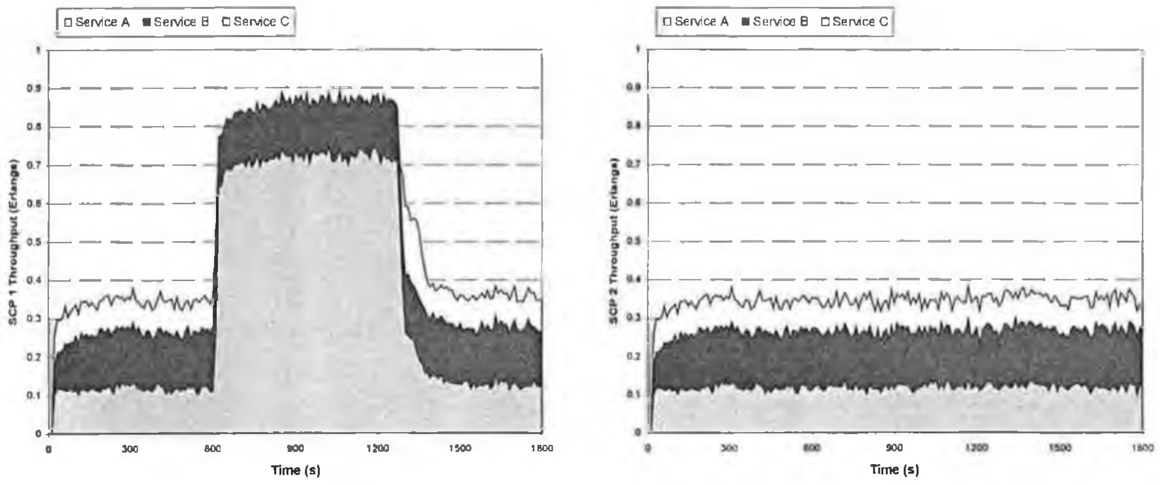


Figure 6.2: Dual SCP network with static routing. Per service type SCP throughput. Left: SCP 1. Right: SCP 2.

6.1.2 Centralised Control of Multiple SCPs

The token generation algorithm described in chapter 5 operates by allocating tokens on a one-by-one basis until the supply of SCP processing capacity exhausts. Because of this iterative nature it is straightforward to enhance this algorithm to allocate resources for more than one SCP. Tokens are still allocated on the basis of maximum marginal utility per marginal cost and secondary criteria, but potential allocations are now examined on a per-SCP, as well as on a per-SSP, per-service basis. The algorithm terminates when the supplies of processing capacity of all SCPs are exhausted. Application of this algorithm clearly requires that information regarding the capacity and per-service processing costs of each SCP be made available at the network node where the algorithm executes, hence we have 'centralised' control of the SCP resources.

Upon completion of the token generation process per-service, per-SCP token allocations are sent to SSPs, which use them for accepting service requests over the coming interval. New service requests are accepted if any tokens (for any SCP) remain and on the basis of a percentage thinning based token-spending algorithm. A number of strategies to decide which SCP an accepted request should be routed towards can be envisaged; the one we investigate is to select the SCP for which the remaining supply of tokens (for the service in question) is largest. Employing token allocations as inputs to the routing decision effectively integrates dynamic routing with the token-based load control strategy. In §6.1.2.1 we specify a first version of the enhanced algorithm, whilst in §6.1.2.2 the enhanced token spending process is specified. The performance of the enhanced strategy is analysed and improvements to the token generation algorithm described in §6.1.2.3.

6.1.2.1 Specification of Enhanced Token Generation Algorithm

To specify the enhanced token generation algorithm we employ much of the notation used in §5.1, where the original, single-SCP token-based strategy was specified. Before providing the algorithm specification we introduce the complete notation:

Assuming the network consists of I SCPs, K SSPs and J service types, let i, k and j denote an arbitrary SCP, SSP and service type respectively. Let $r_k(j)$ denote the profit generated by a successful session of service type j originating at SSP k. Let the total available capacity bid by the SCP i be denoted by c_i and let the total processing costs associated with a type j service session at SCP i be denoted by $p_i(j)$. Let $q_k(j)$ denote the bid of SSP k for tokens of service type j and let $n_{i,k}(j)$ denote the number of SCP i tokens allocated in response to this bid.

All SSPs maintain IJ pools of tokens, one for each SCP and service type pairing. When the SSP admits a type j service request one token is removed from token pool ij. Token pools are refilled as a result of the token generation process, which is executed at the start of control intervals of length T time units. During the token generation process records are kept, for each SCP, of the total remaining processing capacity and the price per allocation of tokens associated with the transactions; these records are denoted by s_i and $c_{i,k}(j)$ respectively.

Let $u_k(j)$ denote the marginal utility associated with allocating a type j token to SSP k, and $n_k(j)$ denote the total number of type j tokens allocated to SSP k as a result of previous algorithm iterations. By interpreting the expected number of service requests $q_k(j)$ as the average of a Poisson distribution, we obtain $u_k(j)$ as

$$u_k(j) = r_k(j) \sum_{w=n_k(j)+1}^{\infty} \frac{q_k(j)^w}{w!} e^{-q_k(j)}$$

The marginal cost of an additional token can be simply interpreted as the total processing cost incurred by the SCP arising directly from consumption of that token. Let $v_i(j)$ denote the marginal cost associated with issuing a type j token at SCP i. We have:

$$v_i(j) = p_i(j)$$

Let a (i,j,k)-allocation refer to assigning a type j token from SCP i to SSP k. The marginal utility per marginal cost, denoted by $\delta_{i,k}(j)$ of such an action is:

$$\delta_{i,k}(j) = u_k(j) / v_i(j)$$

We can now specify the algorithm formally. As in the single SCP version of the strategy the variables $\pi_k(j)$ and $\Pi_k(j)$ are used to denote the probability of respectively exactly $n_k(j)$ and more than $n_k(j)$ arrivals of service requests of type j at SSP k during the coming control interval.

Step 1: Initialisation.

```
For all SCPs $i = 1,...,I$ do:
    For all SSPs $k = 1,...,K$ do:
        For all service types $j = 1,...,J$ do:
            Set token allocations $n_{i,k}(j) = 0$ and costs $c_{i,k}(j) = 0$.
For all SSPs $k = 1,...,K$ do:
    For all service types $j = 1,...,J$ do:
        Set $\pi_k(j) = e^{-q_k(j)}$.
        Set $\Pi_k(j) = 1 - \pi_k(j)$.
        Set marginal utilities $u_k(j) = r_k(j)\Pi_k(j)$.
For all SCPs $i = 1,...,I$ do:
    For all service types $j = 1,...,J$ do:
        Set marginal costs $v_i(j) = p_i(j)$.
For all SCPs $i = 1,...,I$ do:
    Set remaining SCP processing capacity $s_i = c_i$.
```

Step 2: Identify optimal allocations.

```
For all SCPs $i = 1,...,I$, SSPs $k = 1,...,K$ and service types $j = 1,...,J$ do:
    List all candidate allocations that maximise $\delta_{i,k}(j)$.
```

Step 3: Arbitrate between optimal allocations.[73]

```
For all SCPs $i = 1,...,I$, SSPs $k = 1,...,K$ and service types $j = 1,...,J$ do:
    If allocation $(i, j, k)$ is in the candidate list then:
        Criterion A: Supplies.[74]
            If $s_i < s_{i,\max}$, where $s_{i,\max} = \max(s_1,...,s_I)$, then:
                Eliminate candidate allocation $(i, j, k)$.
        Criterion B: Step size.
            If $p_i(j) < p_{i,\max}(j)$, where $p_{i,\max}(j) = \max(p_1(1),..., p_1(J),..., p_I(1),..., p_I(J))$, then:
                Eliminate candidate allocation $(i, j, k)$.
        Criterion C: Provider concentration.
            If $\sum_{k=1}^{K} n_{i,k}(j) > n_{i,\min}(j)$, where $n_{i,\min}(j)$ is the minimum number of tokens allocation
            for any of the service types $j = 1,...,J$ and SCPs $i = 1,...,I$, then:
                Eliminate candidate allocation $(i, j, k)$.
        Criterion D: Fairness.
            If $n_k(j)/q_k(j) > m_k(j)$, where $m_k(j)$ is the minimum of the proportions of
            demands satisfied for any of the SSPs $k = 1,...,K$ and service types $j = 1,...,J$,
            then:
                Eliminate candidate allocation $(i, j, k)$.
        Criterion E: Supplier Concentration.[75]
            If $n_{i,k}(j) > n_{i,\min,k}(j)$, where $n_{i,\min\,k}(j) = \min(n_{1,k}(j),..., n_{I,k}(j))$, then:
                Eliminate candidate allocation $(i, j, k)$.
        Selection.
            Randomly select allocation $(i', j', k')$ from those remaining in the candidate list.
```

[73] Two new criteria, relating specifically to the multiple-SCP case, are included along with those used for the single-SCP case. Other criteria may also be employed.

[74] To encourage load balancing between SCPs, those SCPs with a larger amount of unallocated capacity are preferred over other SCPs.

[75] To encourage stability against sudden load changes with respect to specific services/SSPs, those SCPs which have allocated a smaller number of tokens for service type *j* to SSP *k* are preferred over other SCPs.

Step 4: Perform optimal allocations.

Set $n_{i',k'}(j') = n_{i',k'}(j') + 1$.

Set $c_{i',k'}(j') = c_{i',k'}(j') + u_{k'}(j')$.

Set $s_{i'} = s_{i'} - p_{i'}(j')$.

Step 5: Update internal variables.

Set $\pi_{k'}(j') = \pi_{k'}(j') q_{k'}(j') / n_{k'}(j')$.

Set $\Pi_{k'}(j') = \Pi_{k'}(j') - \pi_{k'}(j')$.

Set $u_{k'}(j') = r_{k'}(j') \Pi_{k'}(j')$.

Step 6: Loop statement.

```
If there exists at least one SCP $i$ for which $s_i > 0$ then:
    GOTO Step 2.
Else:
    STOP.
```

6.1.2.2 Specification of Enhanced Token Spending Process

The enhanced token spending process involves two steps: firstly a percentage thinning throttle is applied to ascertain whether an arriving service request should be accepted; secondly an SCP, to which the initial service session message will be routed, is selected. The PT coefficients for each service are updated at sub-intervals of the control interval using the algorithm described in §5.1.2, page 128.[76] The two-step process for service request acceptance/throttling is specified as follows:

Step 1: Apply PT throttle.

```
Select a random number $X$ uniformly distributed in the range $(0.0, 1.0)$.
If $X < p_a(j)$ then:
    Service request is accepted.
Else:
    Service request is throttled.
    STOP.
```

[76] Here the updating algorithm is initialised with the *total* number of available tokens for the service type.

Step 2: Select SCP.

For all SCPs $i = 1,...,I$ do:

List all SCPs which maximise $n'_i(j)$, where $n'_i(j)$ is the number of tokens remaining in token pool ij.

Randomly select SCP i' from the list.

6.1.2.3 Performance Analysis of Centralised Control Approach

The network scenario illustrated in Figure 6.3 was used to analyse the performance of the centralised token-based load control strategy described above. The network contains four identical SCPs, all of which can serve all of the twelve SSPs. We simulate a global overload in which the arrival rate of service A requests increases, at each SSP, to a level corresponding to an first-offered load of 1.4 Erlangs to each SCP. End-user reattempt behaviour is modelled.

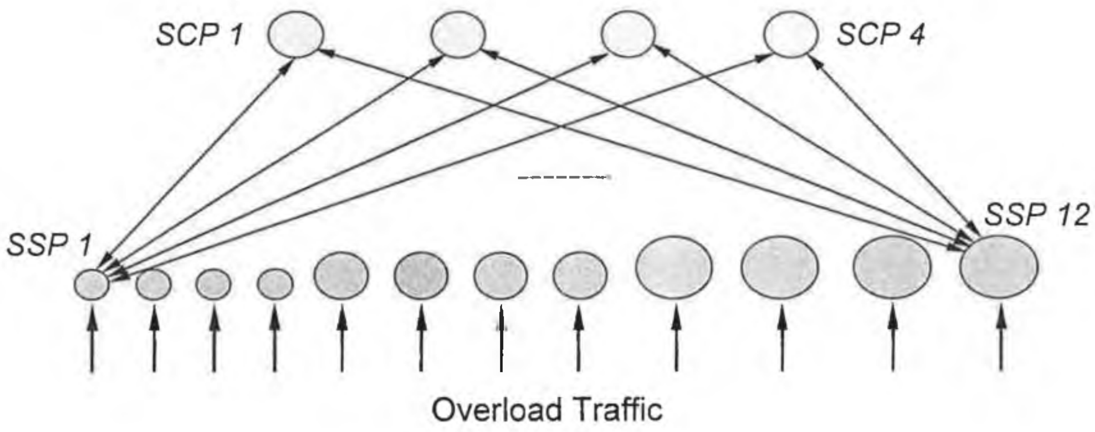


Figure 6.3: Network with $I = 4$ identical SCPs, $K = 12$ SSPs, global overload.

Figure 6.4 shows the offered load per SCP and the mean of the SCP throughputs for this scenario. The token-based strategy is seen to control the throughput of the four SCPs as a whole, with the mean SCP throughput closely approaching the target value during overload. However, to ascertain the actual success of the strategy in protecting against overload, it is necessary to examine the throughput of individual SCPs.[77] Figure 6.5 shows the individual throughputs of the four SCPs and the percentage spread in throughputs between the SCPs with the smallest and largest values. We see that throughputs of all the SCPs approach but do not exceed the target value and are well balanced, deviating by only a small percentage in all conditions.

[77] For all the simulation runs presented in this chapter SCP throughput values are the same as SCP utilisation values, since the token strategy ensures that SCP input queue sizes do not grow to the degree where messages are dropped or discarded. Therefore we show plots of throughput values only.

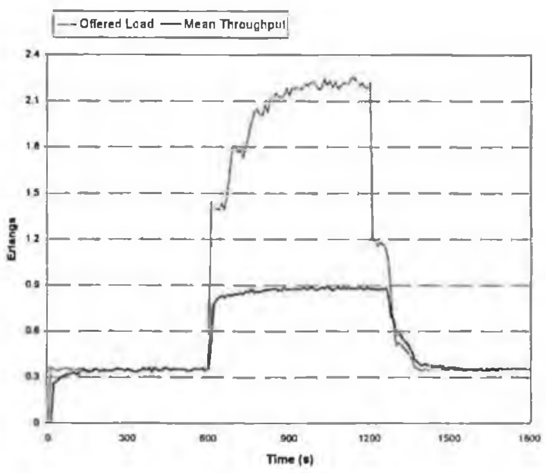
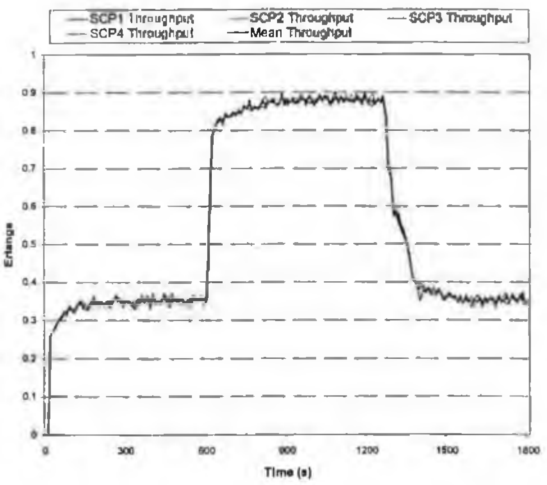


Figure 6.4: Identical SCPs, centralised control. Offered load and Mean SCP throughput.



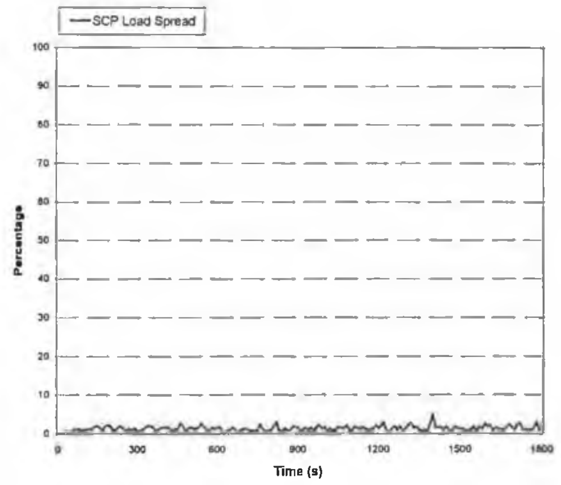


Figure 6.5: Identical SCPs, centralised control. Left: SCP throughputs. Right: Percentage spread of SCP throughputs.

Figure 6.6 shows the per service type SCP throughputs for two of the SCPs: SCP 1 and SCP 3. We see that the strategy succeeds in selectively throttling service requests to maximise profit. During the overload requests for service A are accepted in preference to those for lower profit service C, but not in preference to higher profit service B. Figure 6.7 shows the total completion rates and profit generated per second. The spike in completion rates as the overload abates is caused by reattempts resulting from service C requests throttled just before overload abatement; because of the low profit generated by successful service C sessions this spike is not mirrored in the profit plot.

Figure 6.8 shows the per SSP size completion rates for service A; we see that the strategy treats SSPs fairly: the completion rate ratio appears to change little between normal and overload conditions. Specifically, the completion rate ratio during overload is 1 : 3.17 : 7.43, which closely approximates the actual SSP size ratio of 1 : 3 : 7.

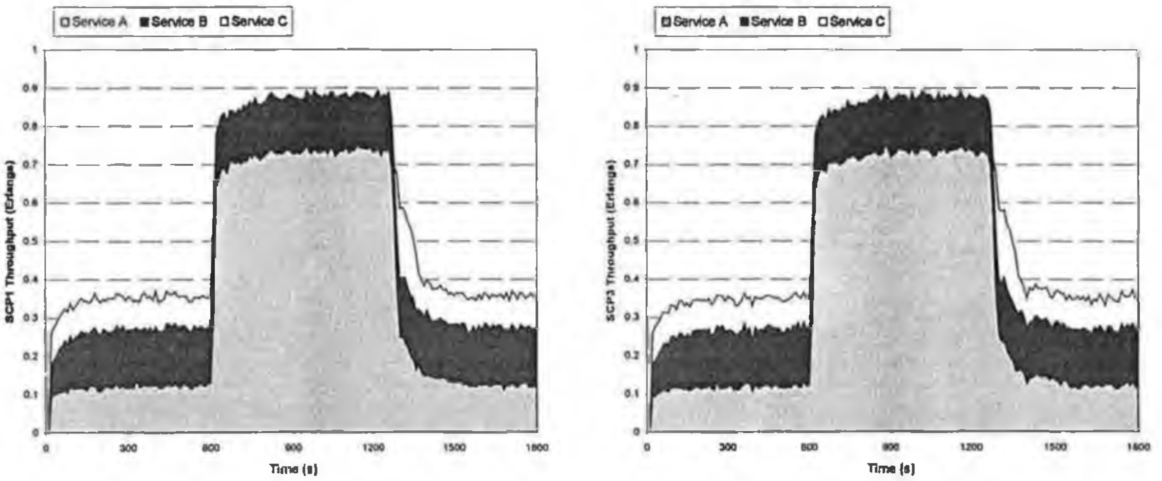


Figure 6.6: Identical SCPs, centralised control. Per service type SCP throughput. Left: SCP 1. Right: SCP 3.

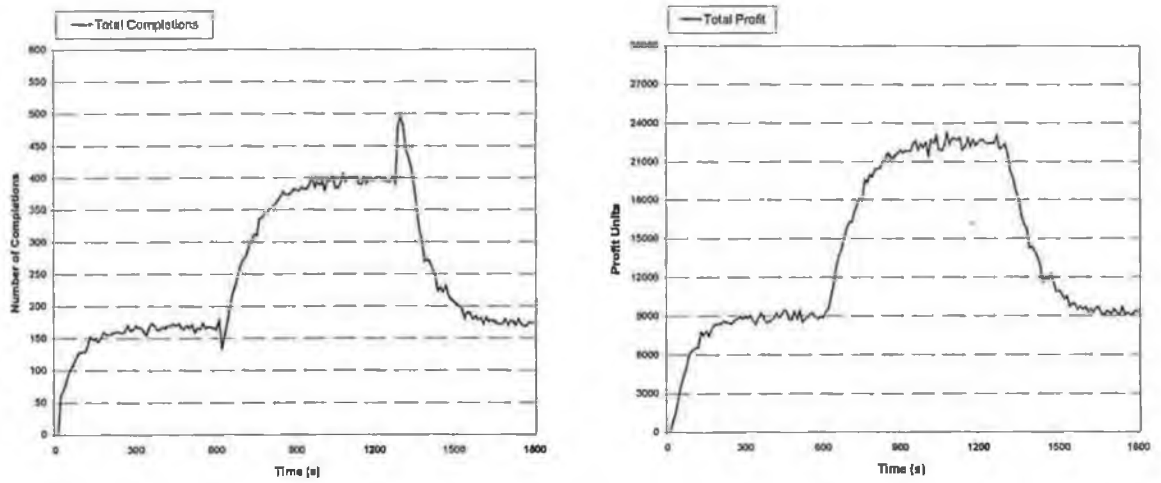


Figure 6.7: Identical SCPs, centralised control. Left: Total completion rates. Right: Profit generated per second.

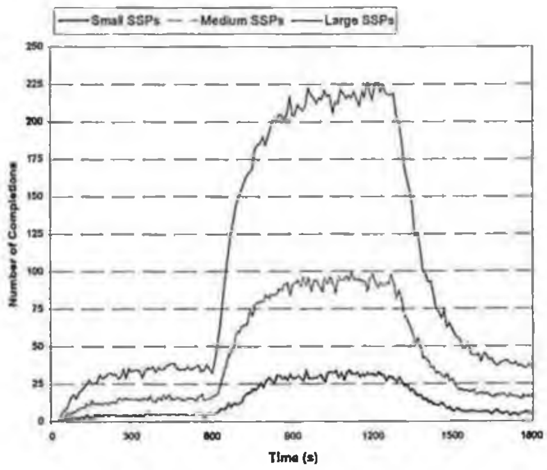


Figure 6.8: Identical SCPs, centralised control. Per SSP size completion rates.

The results presented above show that the strategy provides near-optimal load control when all SCPs have identical performance characteristics. The network scenario depicted in Figure 6.9 was used to analyse its performance in networks containing heterogeneous SCPs, that is, SCPs have differing resource requirements to process sessions of the same service type(s). In this scenario SCP 1 and SCP 2 are 25% faster at processing service A sessions;[78] all SCPs process service B and service C sessions at the same speed. Note that SCPs 1 and 2 will continue to allocate service B tokens in preference to service A tokens at the start of the token generation process, since $r_k(B)/p_i(B) > r_k(A)/p_i(A)$.[79] However, to maximise overall profit during a service A overload, it would be preferable for SCPs 1 and 2 to process only service A sessions and have SCPs 3 and 4 process all service B sessions.

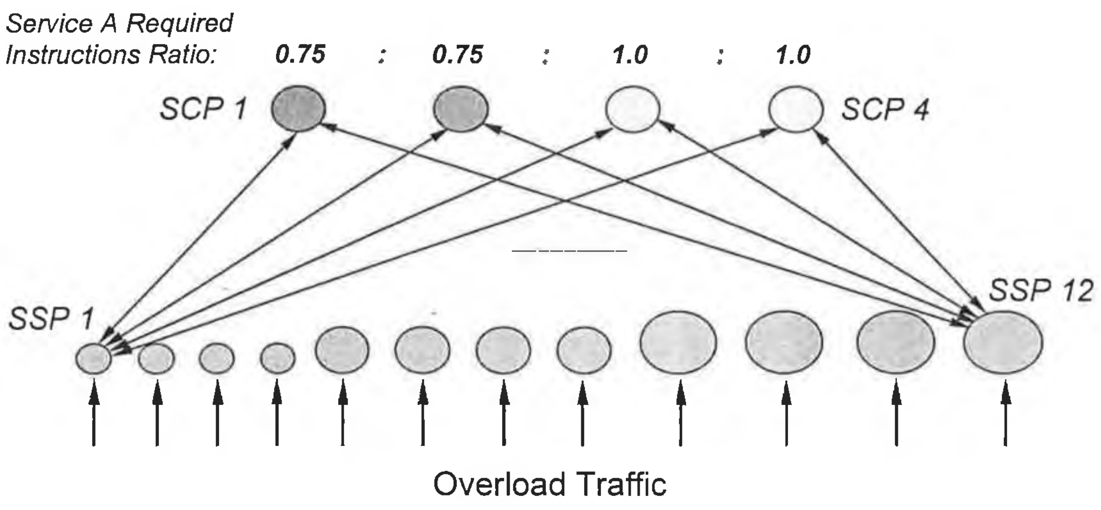


Figure 6.9: Network with $I = 4$ heterogeneous SCPs, $K = 12$ SSPs, global overload.

Figure 6.10 shows the throughputs of the four SCPs and the percentage spread in throughputs between those SCPs with the smallest and largest values. During normal load conditions we see that the throughputs of SCPs 3/4 is approximately 0.15 Erlangs greater than those of SCPs 1/2. As discussed below this is related to the manner in which the capacity of SCPs is allocated between the service types. The figure also shows a dip in the mean SCP load at the abatement of the first-offered overload, at which point the actual load offered to the SCPs is less then their target utilisation.[80] The just-completed token generation process will have allocated large token allocations to larger SSPs, however the arrival rates at these SSPs reduces, leading to a significant degree of token wastage and the dip in mean throughput. However, the next token

[78] We model this by reducing by 25% the required number of SCP processor instructions associated with each service A session message.

[79] Assuming there are reasonably large bids from each SSP, at the start of the token generation process the probability of token use is very close to 1.0.

[80] The increased SCP 1/2 processing rate for service A sessions means that the SCPs can handle this level of offered load; in the previous scenario, where all SCPs had the same characteristics, this was not the case.

generation process adapts to the changed SSP arrival rates and token wastage becomes negligible.

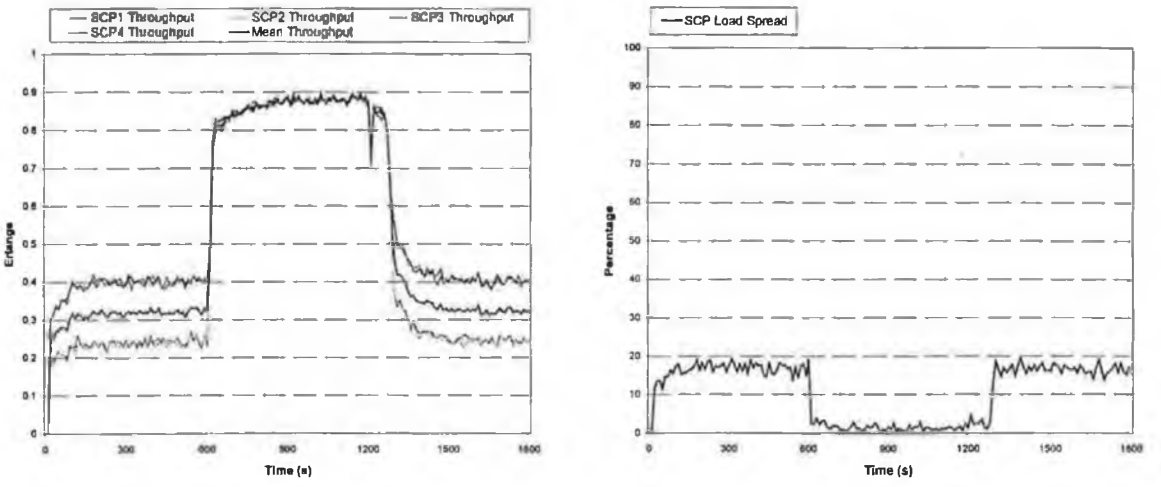


Figure 6.10: Heterogeneous SCPs, centralised control. Left: SCP throughputs. Right: Percentage Spread of SCP throughputs.

Figure 6.11 shows the per service type SCP throughputs for two of the SCPs: SCP 1 and SCP 3. During normal conditions tokens are initially allocated for service B from all four SCPs. It then becomes more optimal to allocate service A tokens, all of which are allocated from SCPs 1/2 since these SCPs provide faster processing rates for this service. Further service B tokens are allocated at the same time as the service A tokens, with most of these being allocated from SCPs 3/4, since SCPs 1/2 capacity has been mostly used for service A. This accounts for SCPs 3/4 having a larger service B throughput then SCPs 1/2. Once all service A/B tokens have been allocated the service C tokens are allocated from SCPs 3/4 (SCPs 1/2 capacity has been all but exhausted).

From Figure 6.11 we see that during overload all four SCPs have a broadly similar service B throughput. However, as discussed above, the SCP 1/2 processing capacity used on service B sessions would have been more profitably spent on service A, since the service B sessions could have been handled by SCPs 3/4. This points towards a flaw in the token generation algorithm: it does not properly ascertain the most profitable means, from the network-wide viewpoint, of utilising SCP processing capacity.

The flaw in the token generation algorithm is related to the expression used to calculate marginal cost: $v_i(j) = p_i(j)$. Use of the *absolute* value of session processing requirements to express marginal cost means that, when allocating a token, the algorithm finds the most profitable service to which blocks of processing capacity can be allocated from the viewpoint of the SCP in question. The algorithm does not take into account that a by-product of allocating the capacity to one service type is the denial of the capacity to the other service types, which from the global perspective may be better served by this SCP. Therefore, to ensure optimal allocation, it is necessary to employ a *relative* measure of cost that makes it possible to ascertain the most profitable means (from a global perspective) of allocating a block of processing capacity.

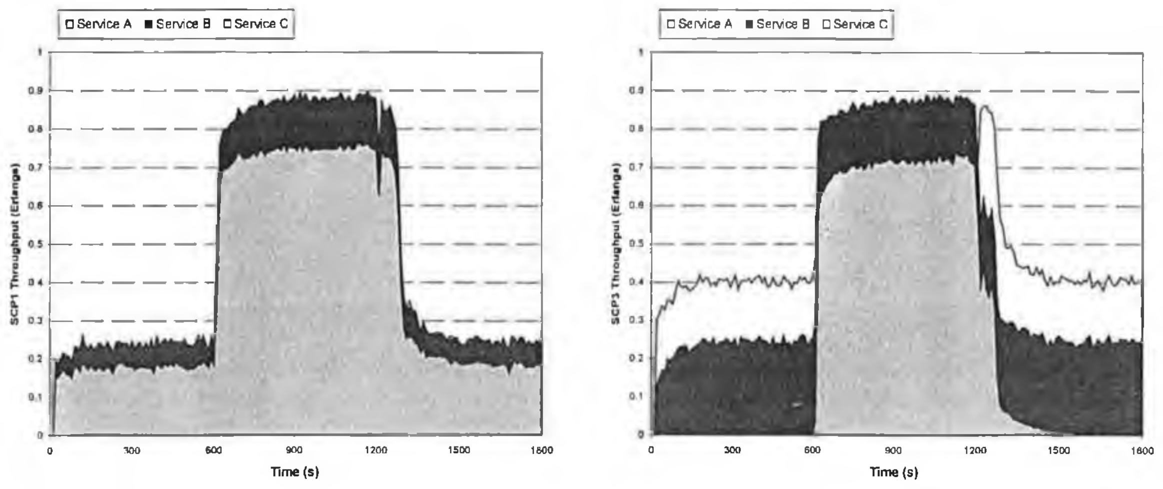


Figure 6.11: Heterogeneous SCPs, centralised control. Per service type SCP throughput. Left: SCP 1. Right: SCP 3.

To provide a relative measurement of the marginal cost of allocating processing capacity we can express it in terms of the alternative marginal gains that could have resulted from other allocations of the same amount of capacity, as follows:

$$v_i(j) = \sum_{j'=1}^{J} \sum_{k'=1}^{K} \frac{p_i(j)}{p_i(j')} u_{k'}(j')$$

In this expression the summations cover all possible allocations of a processing capacity block, division by $p_i(j')$ normalises session processing requirements relative to those for service type j at SCP i, and $u_{k'}(j')$ is the potential marginal gain from a (k', j') allocation.[81]

To illustrate how the updated expression for $v_i(j)$ ensures the optimality of the token allocation consider the example of SCP 1 (or SCP 2) allocating tokens during overload conditions. The summations used to calculate the $v_i(j)$ values implicitly take into account two factors: firstly that service B sessions can be processed at the same absolute cost at other SCPs as they can at SCP 1 and secondly, that less processing capacity is required to process service A sessions at SCP 1 then is required at SCPs 3/4. Therefore, the summation for service B for SCP 1 will yield

[81] Because $v_i(j)$ is now dependant on the $u_k(j)$ values it must be updated at each iteration of the token generation algorithm. To do so a recursion approach can again be employed. If we set:

$$w_i = \sum_{j=1}^{J} \sum_{k=1}^{K} \frac{u_k(j)}{p_i(j)},$$

where w_i gives a measure of the price per unit processing capacity at SCP i, then $v_i(j) = p_i(j) w_i$ and in Step 5 of the token generation algorithm (Update internal variables) the following lines can be added to update $v_i(j)$:

For all SCPs $i = 1,\ldots,I$ do: Set $w_i = w_i - r_{k'}(j')\pi_{k'}(j') / p_i(j')$.

For all SCPs $i = 1,\ldots,I$ and all service types $j = 1,\ldots,J$ do: Set $v_i(j) = p_i(j) w_i$.

a higher value then the summation for service A at SCP 1 and tokens for SCP 1 will be awarded to service A in preference to service B (or indeed service C).

We now analyse the performance of the token-based strategy in the presence of heterogeneous SCPs when the updated expression for the marginal cost of token allocations is employed. Figure 6.12 shows the throughputs of the four SCPs and the percentage spread in throughputs between the SCPs with the smallest and largest values. During normal load conditions we again observe a significant percentage spread between the throughputs of SCPs 3/4 and SCPs 1/2. We also observe a discernible spread between throughputs during the overload, with SCPs 1/2 throughputs approaching the target value more closely then those of SCPs 3/4.

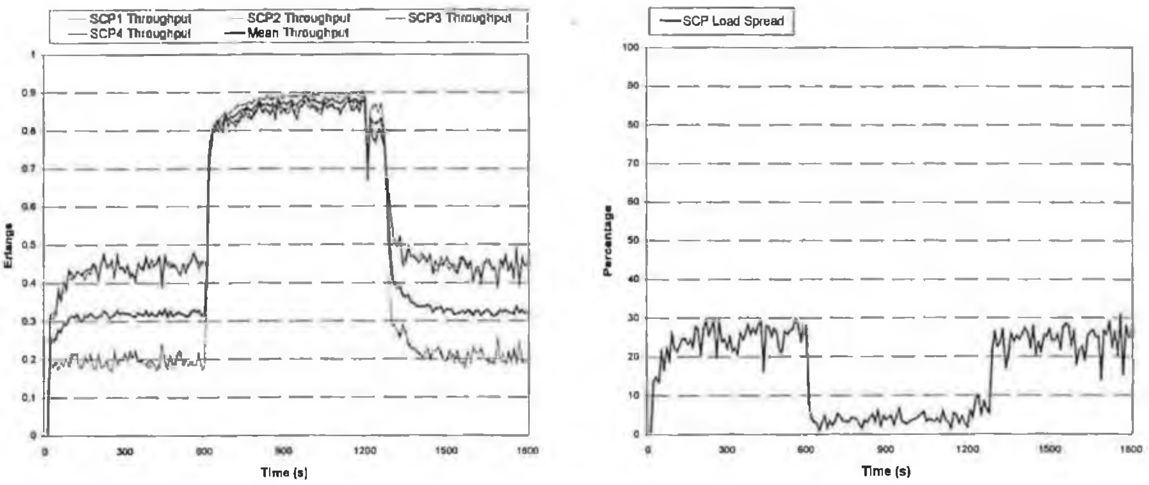
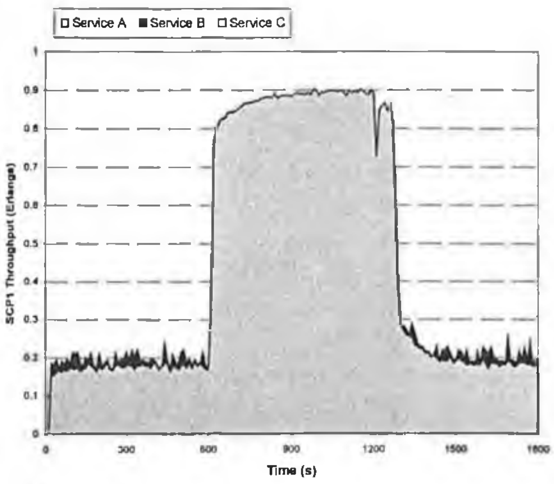


Figure 6.12: Heterogeneous SCPs, enhanced centralised control. Left: SCP throughputs. Right: Percentage Spread of SCP throughputs.

Figure 6.13 shows the per service type SCP throughputs for two of the SCPs: SCP 1 and SCP 3. During normal conditions we see that there is a smaller throughput for service B at SCPs 1/2 then in the previous case. This is because SCPs 1/2 now have a strong bias towards service A tokens; they only have service B allocated for them at the end of the token generation process, when the number of service A allocated tokens is so large that the expected marginal utility per marginal cost for service A reduces significantly.

During overload, we see that SCPs 1/2 no longer have service B tokens allocated from them, they only process service A sessions. All service B sessions are now processed by SCPs 3/4. This allocation results in the imbalance between throughputs during overload noted above. The high arrival rate of service A requests means that no service A tokens remain unused and the throughputs of SCPs 1/2 approach the target value very closely. This is not the case for service B: bids are implicitly inaccurate, so a small number of tokens will remain unused, with the result that the throughputs of SCPs 3/4 fall somewhat short of the target.



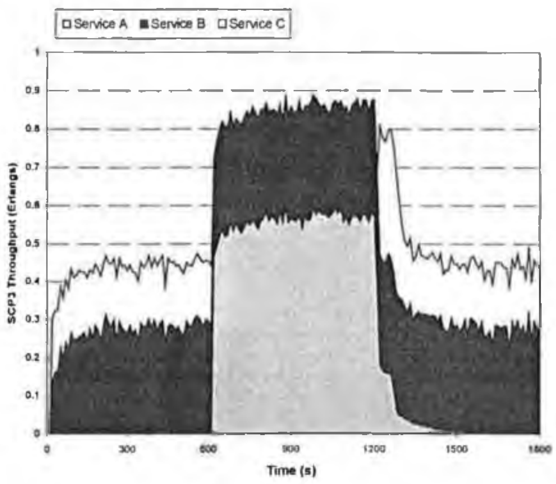
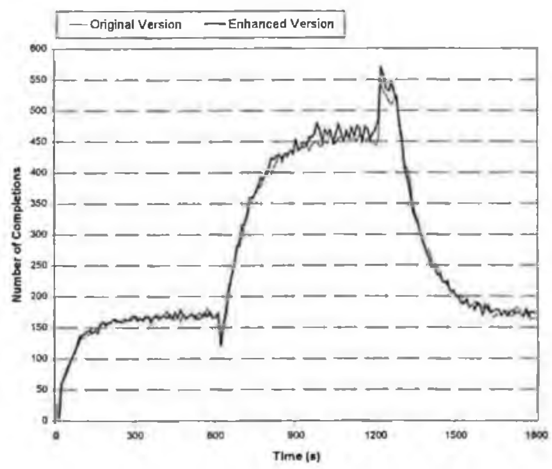


Figure 6.13: Heterogeneous SCPs, enhanced centralised control. Per service type SCP throughput. Left: SCP 1. Right: SCP 3.

Figure 6.14 compares the total completion rates and profit generated per second achieved when employing the original and updated versions of the marginal cost expression respectively. We see that, during overload, for the updated version a larger number of (service A) sessions complete successfully and that this is reflected in an increase in the level of generated profit.[82] In this scenario the increase in profit is marginal, however in scenarios where the differences in profit per unit of processing capacity for different SCPs is more pronounced the increase would become more significant.



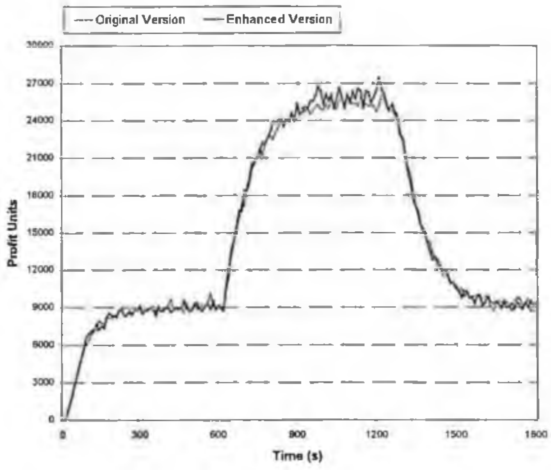


Figure 6.14: Comparison between original and updated versions. Left: Total completion rates. Right: Profit generated per second.

[82] We have verified that the differences between the completion rate and profit plots observed here relate to the different token allocations and not to statistical variations between simulation runs.

6.1.3 Distributed Control of Multiple SCPs

In the preceding section we investigated a centralised version of the token-based strategy that allocates SCP processing capacity by means of a single token generation process executing at a single network node. In a large-scale network it may be neither practical nor desirable to execute a central token-generation process requiring information from all SSPs and SCPs in the network. In this section we present a distributed version of the strategy, where each SCP executes its own token generation process and SSPs split their arrival rate bids between a number of SCPs.[83]

Assuming that an SSP splits its bids into I' sub-bids (where $I' \leq I$), then the total bid from SSP k for service type j can be written in terms of the sub-bids as follows:

$$q_k(j) = \sum_{i=1}^{I'} q_{i,k}(j) = \sum_{i=1}^{I'} a_{i,k}(j) q_k(j), \text{ with } \sum_{i=1}^{I'} a_{i,k}(j) = 1.0$$

In the above expression the $a_{i,k}(j)$ values are weights determining how much of the expected demand at SSP k for service type j sessions is bid to the token generation process at SCP i. Note that since we assume that streams of service requests behave as Poisson processes it is possible to split bids in this manner without affecting the accuracy of the cumulative bids received by the SCPs.

The most obvious approach an SSP can take in calculating values for $a_{i,k}(j)$ is to split bids evenly between all the SCPs it communicates with. In §6.1.3.1 we specify and analyse the performance of this approach and show that it leads to sub-optimal resource allocations in networks containing heterogeneous SCPs. In §6.1.3.2 we specify and analyse a more complex approach in which closing prices from the token generation process are used to calculate optimal bid splits.

6.1.3.1 Even Bid-Splitting Approach: Specification and Performance Analysis

For even bid splitting SSPs simply divide their total bid for a service type by the number of SCPs to which they will bid, as follows:

$$q_{i,k}(j) = a_{i,k}(j) q_k(j) = \frac{1}{I'} q_k(j)$$

[83] In the distributed version the token generation processes allocate tokens for a single SCP only, so the original token generation algorithm, as described in chapter 5, can be employed and there is no necessity to adopt the updated expression for marginal cost specified in §6.1.2.

To investigate the even bid splitting approach we first use the network scenario depicted in Figure 6.3 (consisting of four identical SCPs and 12 SSPs). We assume that all SSPs split their bids between all SCPs. Overload is again caused by an increase in the arrival rate of service A requests at all SSPs, corresponding to a first-offered per-SCP load of 1.4 Erlangs. Figure 6.15 shows the per-SCP offered load and the mean of the SCP throughputs for this scenario and Figure 6.16 shows the individual SCP throughputs and the percentage spread. These figures show the performance of the distributed strategy in protecting SCPs from overload is very similar to that of the centralised strategy in the same scenario; this confirms that splitting of bids does not affect ability of the strategy to protect against overload.

Figure 6.17 shows the per service throughputs of SCP 1 and SCP3, whilst Figure 6.18 shows the total completion rates and profit generated per second. These graphs confirm that, when all SCPs are identical, the distributed strategy exhibits similar profit maximising characteristics as the centralised strategy. However, Figure 6.19 shows that there is some difference between the two versions: the ratio of completion rates between differently sized SSPs during overload is not as fair for the distributed version (1 : 6.42 : 16.94) as it was for the centralised version (1 : 3.17 : 7.34). This is due to the bids inputted into the individual token generation processes being smaller in absolute terms in the distributed version: the smaller the absolute values of the bids the more the token generation algorithm is biased in favour of those SSPs with larger bids.

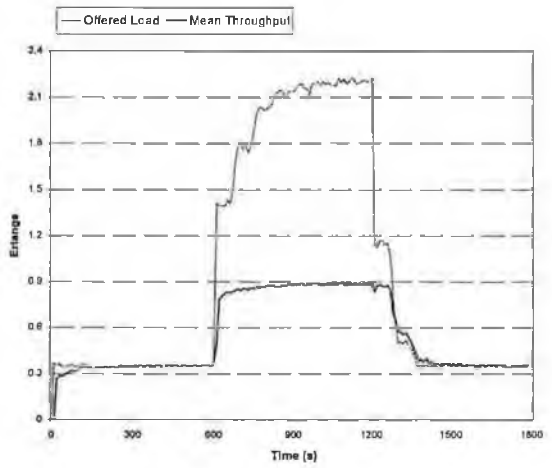
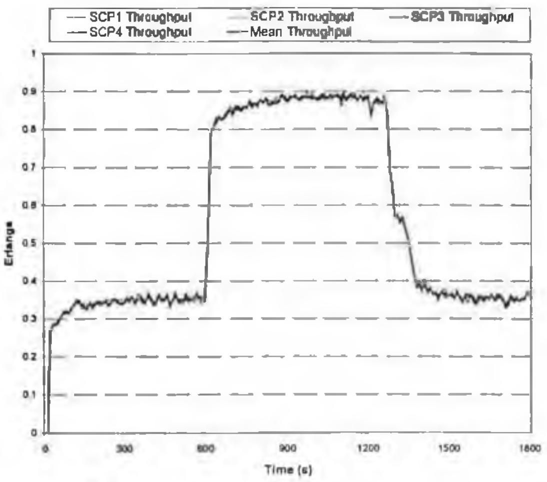


Figure 6.15: Homogeneous SCPs, distributed control with even bid splitting. Offered load and Mean SCP throughput.



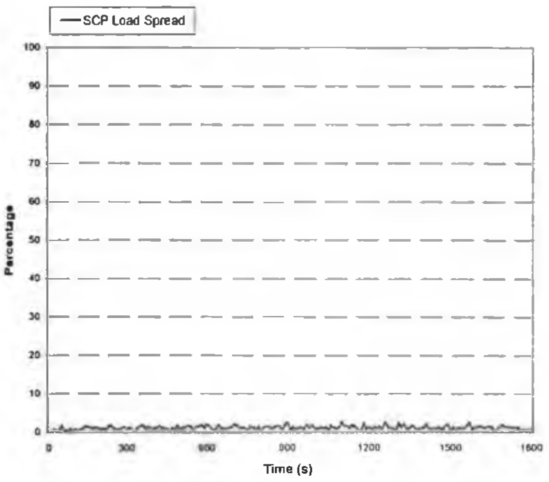
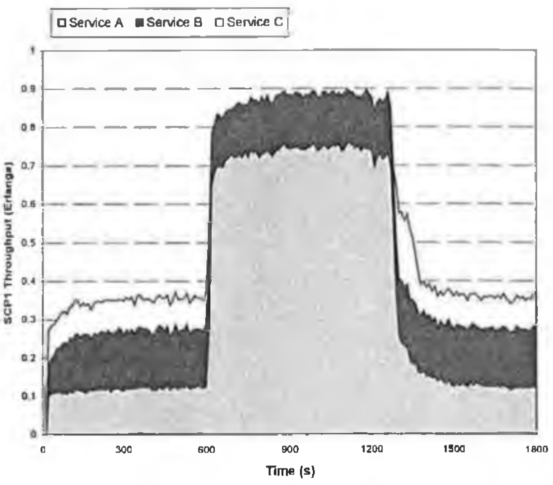


Figure 6.16: **Homogeneous SCPs, distributed control with even bid splitting. Left: SCP throughputs. Right: Percentage Spread of SCP throughputs.**



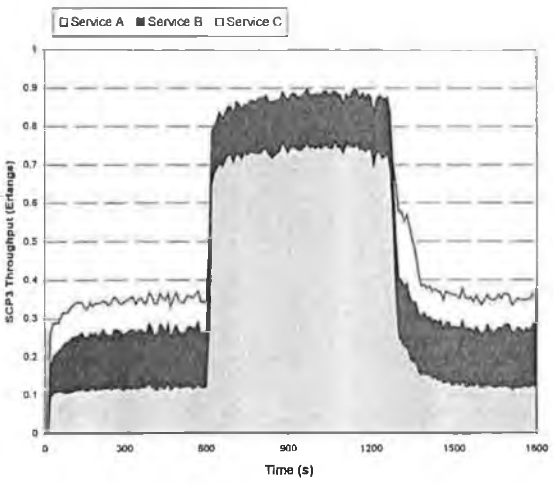
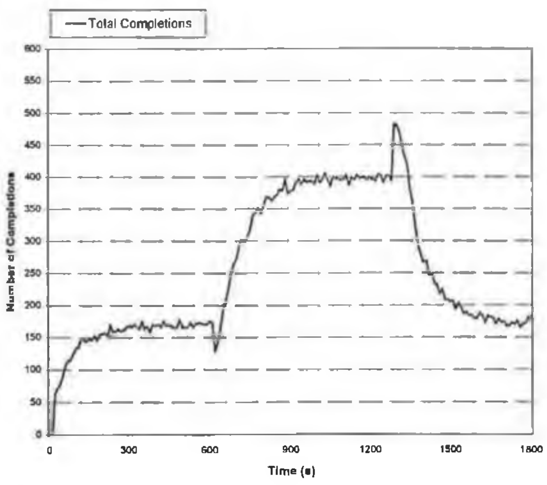


Figure 6.17: **Homogeneous SCPs, distributed control with even bid splitting. Per service type SCP throughput. Left: SCP 1. Right: SCP 3.**



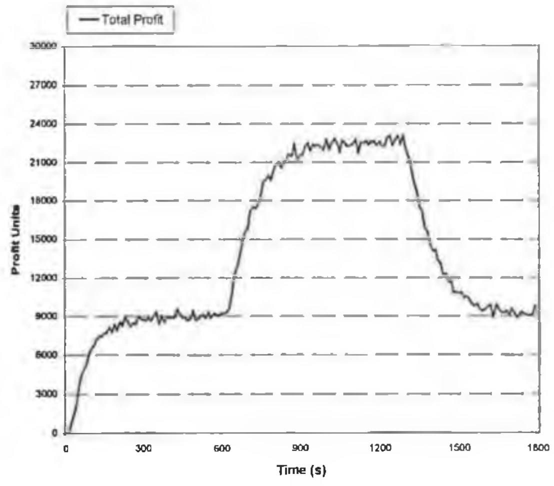


Figure 6.18: **Homogeneous SCPs, distributed control with even bid splitting. Left: Total completion rates. Right: Per SSP size completion rates.**

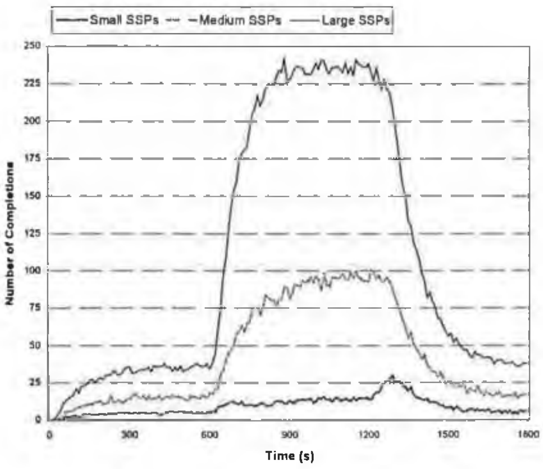


Figure 6.19: Homogeneous SCPs, distributed control with even bid splitting. Profit generated per second.

We now analyse the performance of the even bid splitting approach in the presence of heterogeneous SCPs, again employing the network scenario depicted in Figure 6.9. Figure 6.20 shows the throughputs of the four SCPs and the percentage spread in throughputs between those SCPs with the smallest and largest values. We see that in normal conditions there is an appreciable spread between the throughputs of SCPs 1/2 and those of SCPs 3/4, due to SCPs 1/2 allocating a greater number of tokens. However this spread is not as severe as in the centralised case, because SCP 1/2 service A tokens can only be allocated in response to bids sent to SCPs 1/2, whereas in the centralised case they could be allocated to the totality of SCP service A bids. This means that in normal conditions service A sessions will be processed by all SCPs, not just SCPs 1/2 (as evident from Figure 6.21).

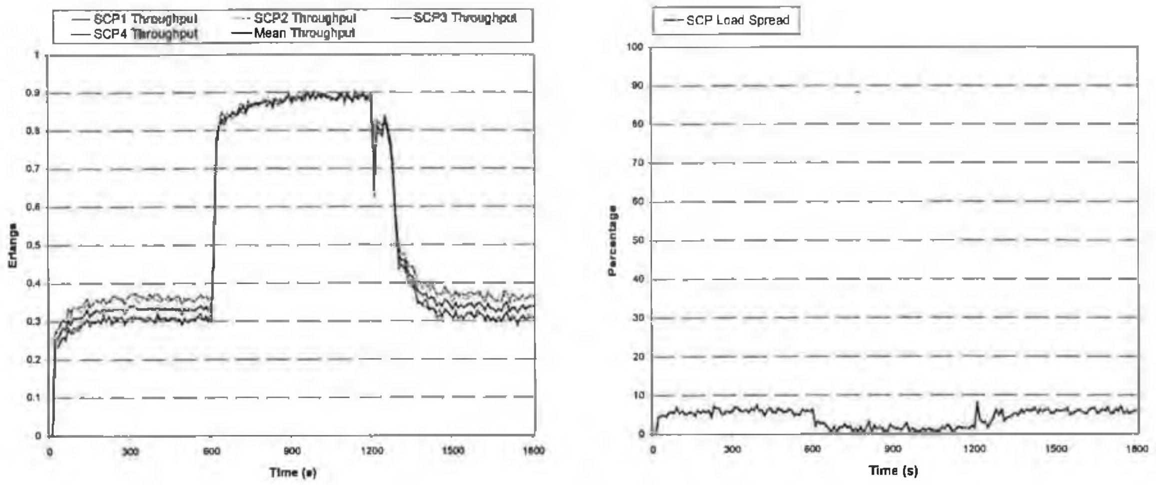
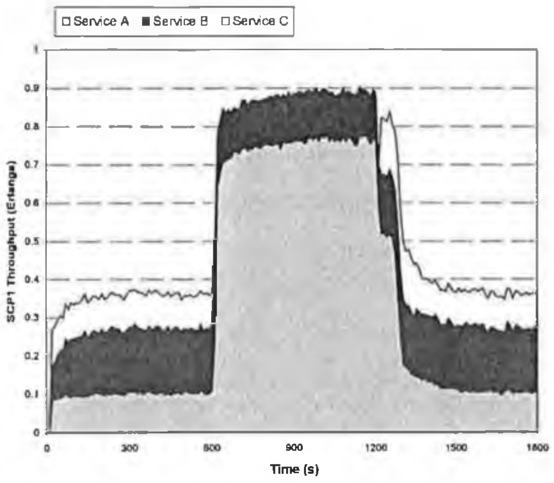


Figure 6.20: Heterogeneous SCPs, distributed control with even bid splitting. Left: SCP throughputs. Right: Percentage Spread of SCP throughputs.

Figure 6.21 shows the per service type throughputs for SCP 1 and SCP 3, whilst Figure 6.22 shows the total completion rates and profit generated per second. We see that during overload SCP 1 processes service B sessions, due to the token generation algorithm basing its marginal cost expression on an absolute measurement of profit per processing capacity unit at the SCP in question. As discussed in §6.1.2.3 this is sub-optimal behaviour, ideally SCP 1/2 should not process service B sessions during overload. To overcome this drawback, it is possible to employ a more advanced bid splitting approach, based on use of closing prices from token generation processes. This approach is investigated in the next section.



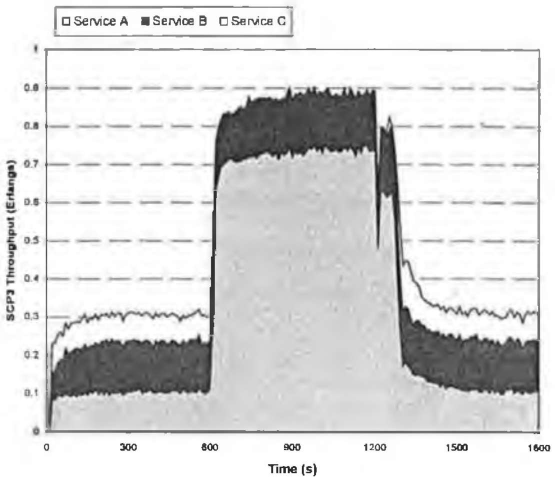
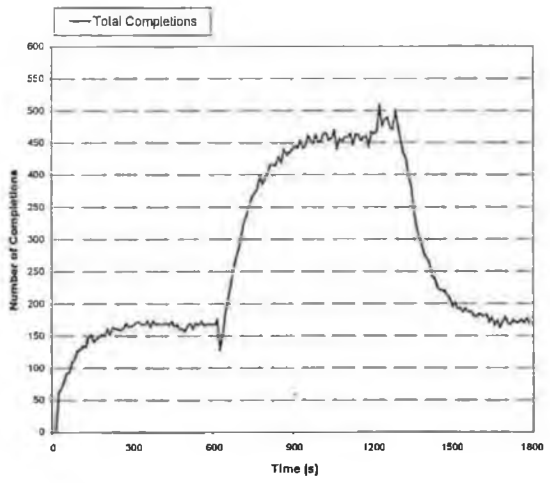


Figure 6.21: Heterogeneous SCPs, distributed control with even bid splitting. Per service type SCP throughput. Left: SCP 1. Right: SCP 3.



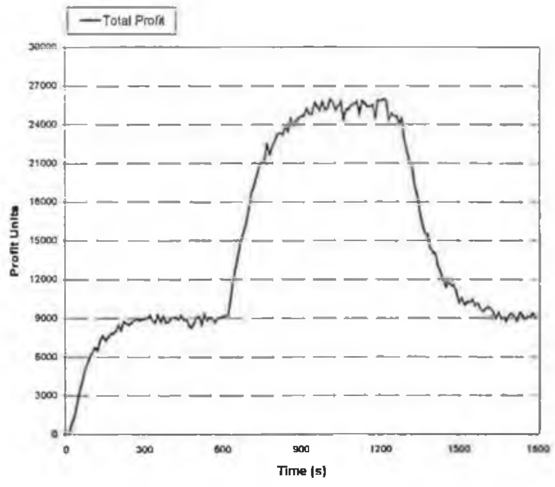


Figure 6.22: Heterogeneous SCPs, distributed control with even bid splitting. Left: Total completion rates. Right: Profit generated per second.

6.1.3.2 Price-based Bid Splitting Approach: Specification and Performance Analysis

For the price-based bid splitting approach prices per service type per token at each SCP, denoted $c_i(j)$, are computed at the end of the token generation process. These prices are calculated by dividing the sum of the per SSP token allocation prices for that service by the sum of the token allocations for the service type, as follows:

$$c_i(j) = \sum_{k=1}^{K} c_{i,k}(j) \Big/ \sum_{k=1}^{K} n_{i,k}(j)$$

These prices are indicated to the SSPs along with the new token allocations from that SCP. They are used by the SSP during the formulation of the bids sent to the token generation process at the start of the next control interval.

In formulating its bids an SSP aims to minimise the total cost of the tokens it will be allocated. In its calculation of the optimal bid splits $q_{i,k}(j)$ it assumes a linear relationship between bids and prices: if its bids increase/decrease from those of the previous interval the prices are assumed to increase/decrease proportionally. This implicitly assumes that all other SSPs will make bids that are identical to those they made in the previous interval. Of course this will not be the case since other SSPs will be adopting the same approach and changing their bids, thereby affecting closing prices. However, as the bids sent to SCPs with currently lower prices increase the prices will increase and *vice versa*, so prices will tend to stabilise (as long as offered load is stable) after a number of control intervals.

The details of the calculation are as follows: assume that the previous bid by SSP k to SCP i for service type j resulted in an allocation of $n_{i,k}(j)$ tokens at a price $c_i(j)$. The forthcoming bid, for $q_k(j)$ tokens should then be split into bids $q_{1,k}(j),\ldots,q_{I',k}(j)$[84] such that the resulting total cost:

$$\sum_{i=1}^{I'} q_{i,k}(j)\left[\frac{q_{i,k}(j)}{n_{i,k}(j)}.c_i(j)\right]$$

is minimised under the constraint

$$\sum_{i=1}^{I'} q_{i,k}(j) = q_k(j)$$

The above is a non-linear optimisation problem, however it can be solved in a straightforward manner. First we separate $q_{I',k}(j)$ from the summation in the total cost expression, giving:

$$\sum_{i=1}^{I'-1} \frac{q_{i,k}^2(j)}{n_{i,k}(j)}.c_i(j) + \frac{q_{I',k}^2(j)}{n_{I',k}(j)}.c_{I'}(j)$$

Then, we use the constraint to substitute for $q_{I',k}(j)$:

$$\sum_{i=1}^{I'-1} \frac{q_{i,k}^2(j)}{n_{i,k}(j)}.c_i(j) + \frac{c_{I'}(j)}{n_{I',k}(j)}\left[q_k(j) - \sum_{i=1}^{I'-1} q_{i,k}(j)\right]^2$$

[84] For simplicity we will again assume in our analysis that SSPs always split bids between all SCPs, that is, $I' = I$.

Differentiating with respect to $q_{i,k}(j)$ for all i and setting the resulting expressions equal to zero yields $I'-1$ linear equations in $q_{1,k}(j),...,q_{I'-1,k}(j)$, which can be solved simultaneously. Finally, $q_{I',k}(j)$ can be found through use of the constraint.

To analyse this approach we again simulate the network scenario depicted in Figure 6.9. Figure 6.23 shows the throughputs of the four SCPs and the percentage spread in throughputs between those SCPs with the smallest and largest values. This figure is similar to that for the even bid splitting approach (Figure 6.20), but with a slightly larger spread in the throughputs between SCPs 1/2 and SCPs 3/4 during overload, indicating that they are processing different service types. This is confirmed by Figure 6.24, which shows the per service type throughputs for SCP 1 and SCP 3. We see that after a settling period at the start of the overload SCPs 1/2 process service A requests only, which is the desired behaviour. A proportion of the throughput at the start of the overload relates to service B; this is due to both the residual processing requirements of service B sessions that were ongoing at overload onset and the fact that it takes a number of iterations for closing prices to adjust to the changed arrival rate bids.

Figure 6.25 compares the total completion rates and profit generated per second achieved when employing the even bid splitting and price-based bid splitting approaches. We see that, during overload, the price-based approach leads to a larger number of (service A) completions and a corresponding increase in generated profit. As with the centralised control strategy the increase is marginal in this scenario, but would increase in scenarios where differences in profit per unit of processing capacity for different SCPs are more pronounced.

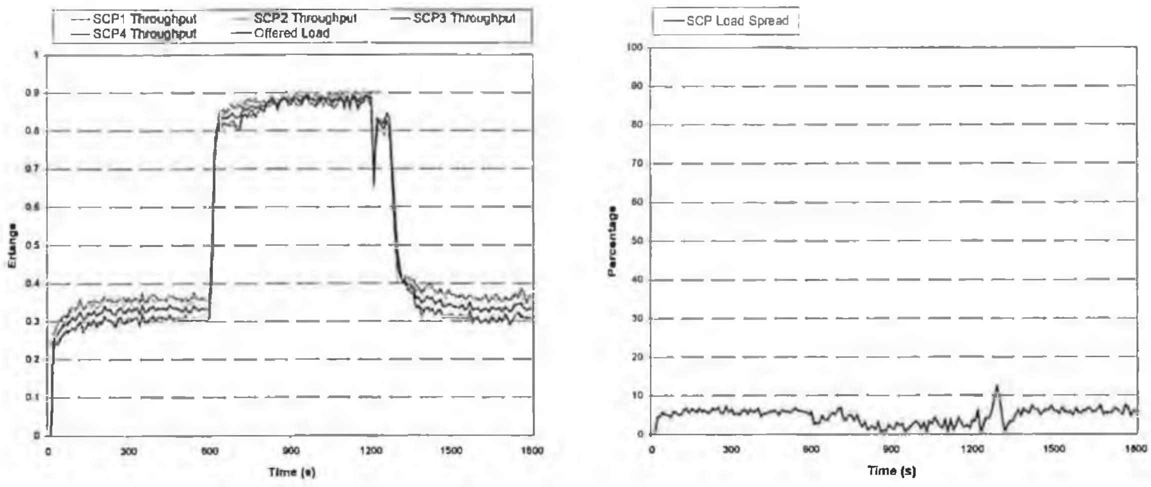
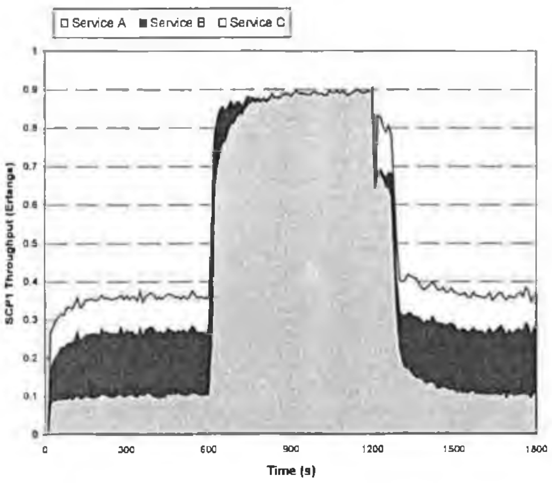


Figure 6.23: Heterogeneous SCPs, distributed control with price-based bid splitting. Left: SCP throughputs. Right: Percentage Spread of SCP throughputs.



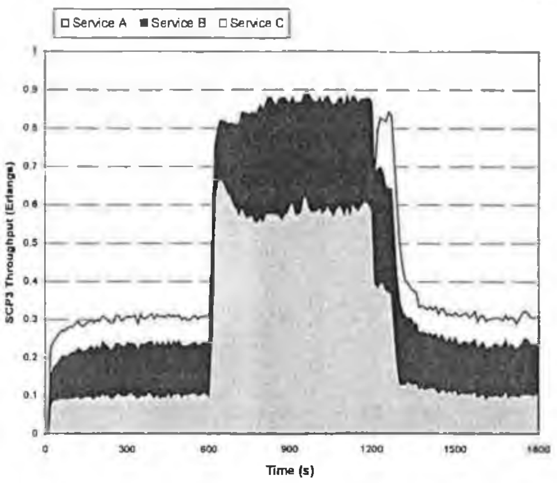
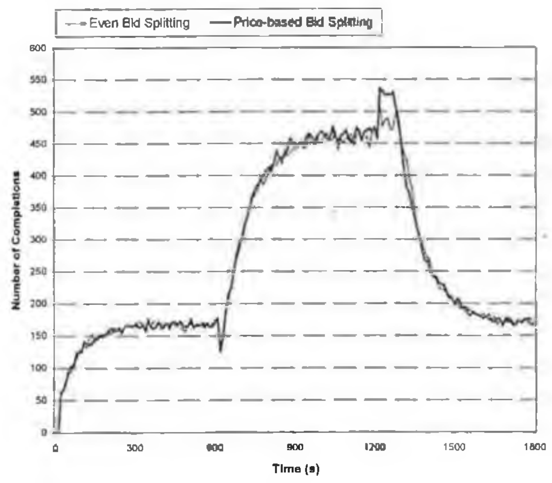


Figure 6.24: Heterogeneous SCPs, distributed control with price-based bid splitting. Per service type SCP throughput. Left: SCP 1. Right: SCP 3.



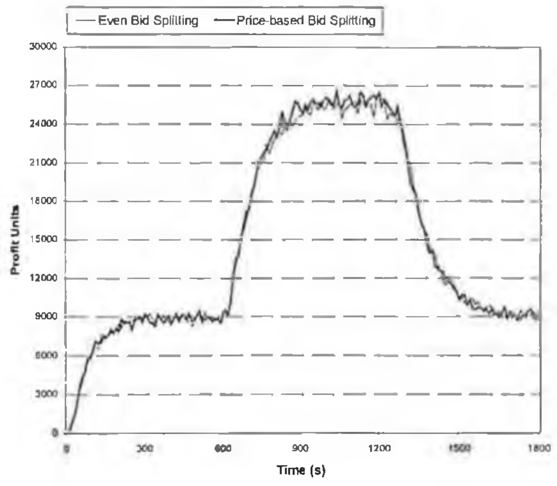


Figure 6.25: Heterogeneous SCPs, distributed control with price-based bid splitting. Left: Total completion rates. Right: Profit generated per second.

6.1.4 Hierarchical, Distributed Control of Multiple SCPs

The distributed strategy presented above involved all SSPs sending bids to potentially all SCPs in the network. A disadvantage of this scheme is that, in a network with a large number of SSPs/SCPs, the amount of bid-related traffic generated every control interval may be unacceptably large. A solution is to adopt a scheme in which certain SSPs bid on behalf of other SSPs (or other groups of SSPs), receive the subsequent token allocations and redistribute them to these SSPs. In effect this creates a 'bidding hierarchy,' which can be structured to mirror actual network topology. In this section we follow this approach, specifying a hierarchical, distributed version of the token-based strategy and analysing its likely performance.

Assume that in the bidding hierarchy all K SSPs are arranged into Y (possibly overlapping) SSP groups. Let y denote an arbitrary grouping and let X_y denote the number of SSPs in group y. Of these, one 'principal' SSP will collate bids received from the other $X_y - 1$

'secondary' SSPs with its own, forward the collated bid(s) to SCPs and/or other SSPs and redistribute resulting allocations between itself and the secondary SSPs. Let $x \in \{1,\dots,X_y\}$ identify an arbitrary SSP from group y. Furthermore, let every x map to $k \in \{1,\dots,K\}$, where each k uniquely identifies an SSP in the entire network. Finally, let k_y denote the unique identifier of the principal SSP of group y.

The principal SSP collates its own bid and those of the secondary SSPs on a per service basis, as follows:

$$q_{k_y}(j) = \sum_{x=1}^{X_y} q_x(j)$$

This bid may then be forwarded as-is to a centralised token generation process, or split (using either the even or price-based approaches) with sub-bids being forwarded to a number of remote token generation processes. In response to these bids the principal SSP will receive (cumulative) per-service token allocations, $n_{k_y}(j)$. These allocations must then be split into sub-allocations for each SSP in the principal's group (including itself).

A simple approach to calculating sub-allocations is to, on a per-service basis, split the overall allocations in a weighted fashion, where the weight for service type j for SSP x is given by $q_x(j)/q_{k_y}(j)$. However, this approach will not guarantee that the resulting sub-allocations are profit-optimal.[85] A better option is to reallocate tokens by running a SSP token distribution process that is analogous to the token generation process, but with SCP processing capacity supplies being replaced as inputs by overall allocations received by the principal SSP. In addition, since one token of service type j is equivalent to $p_i(j)/p_i(j')$ tokens of service type j', the distribution algorithm can be designed to allow 'token conversion' for the purpose of profit maximisation. We now proceed to specify such an algorithm.[86]

[85] Profit-optimal token allocation requires that the probability of token use be taken into account. A key input into the calculation of this probability is the absolute value of the estimate of arrivals in the forthcoming control interval, with higher estimates increasing the probability of token use. In the weight-based approach relative estimate sizes rather than token use probabilities based on absolute estimate values are used, hence the resulting allocations will be sub-optimal (from the profit viewpoint).

[86] The algorithm is based on the centralised token generation algorithm, but with SCP capacity supply replaced by per-SCP supplies of tokens for the lowest cost service at that SCP; service processing costs replaced by ratios of service processing costs to those of the lowest cost service at that SCP; and the number of SSPs in the network replaced by the number of SSPs in the principal's group. Unless explicitly specified all terminology used here has the same meaning as defined in §6.1.2.

Step 1: Initialisation.

```
For all SCPs $i = 1,...,I$ do:
    Set $m_i(j_{i,\min}) = \sum_{j'}^{J} \frac{p_i(j')}{p_i(j_{i,\min})} n_{k_y}(j')$
    where $j_{i,\min}$ identifies the service type with the lowest value of processing costs at SCP $i$
    and $m_i(j_{i,\min})$ is the pool of available $j_{i,\min}$ tokens for SCP $i$ that may be converted and
    distributed by the algorithm.
For all SCPs $i = 1,...,I$ do:
    For all SSPs $x = 1,...,X_y$ do:
        For all service types $j = 1,...,J$ do:
            Set token allocations $n_{i,x}(j) = 0$.
For all SSPs $x = 1,...,X_y$ do:
    For all service types $j = 1,...,J$ do:
        Set $\pi_x(j) = e^{-q_x(j)}$.
        Set $\Pi_x(j) = 1 - \pi_x(j)$.
        Set marginal utilities $u_x(j) = r_x(j)\Pi_x(j)$.
For all SCPs $i = 1,...,I$ do:
    Set $w_i = \sum_{j=1}^{J} \sum_{x=1}^{X_y} \frac{p_i(j_{i,\min})}{p_i(j)} u_x(j)$.
For all SCPs $i = 1,...,I$ do:
    For all service types $j = 1,...,J$ do:
        Set marginal costs $v_i(j) = \frac{p_i(j)}{p_i(j_{i,\min})} w_i$.
```

Step 2: Identify optimal allocations.

```
For all SCPs $i = 1,...,I$, SSPs $x = 1,...,X_y$ and service types $j = 1,...,J$ do:
    List all candidate allocations that maximise $\delta_{i,x}(j) = \frac{u_x(j)}{v_i(j)}$.
```

Step 3: Arbitrate between optimal allocations.

Apply secondary criteria[87] to eliminate less optimal token allocations, select an allocation at random from the remaining candidates.

Step 4: Perform optimal allocations.

Set $n_{i',x'}(j') = n_{i',x'}(j') + 1$.

Set $m_{i'}(j_{i',\min}) = m_{i'}(j_{i',\min}) - \dfrac{p_{i'}(j')}{p_{i'}(j_{i',\min})}$.

Step 5: Update internal variables.

```
    Set $\pi_{x'}(j') = \pi_{x'}(j') q_{x'}(j') / n_{x'}(j')$.
    Set $\Pi_{x'}(j') = \Pi_{x'}(j') - \pi_{x'}(j')$.
    Set $u_{x'}(j') = r_{x'}(j') \Pi_{x'}(j')$.
    For all SCPs $i = 1,...,I$ do:
        Set $w_i = w_i - r_{x'}(j') \dfrac{\pi_{x'}(j')}{p_i(j')} p_i(j'_{i,\min})$.
    For all SCPs $i = 1,...,I$ do:
    For all service types $j = 1,...,J$ do:
        Set marginal costs $v_i(j) = \dfrac{p_i(j)}{p_i(j_{i,\min})} w_i$.
```

Step 6: Loop statement.

```
    If there exists at least one SCP $i$ for which $m_i(j_{i,\min}) > 0$ then:
        GOTO Step 2.
    Else:
        STOP.
```

To analyse the performance of the hierarchical, distributed token-based load control strategy we simulate the network scenario depicted in Figure 6.26. This scenario consists of three SSP groups, with SSP 1, SSP 2 and SSP 3 acting as principals and each receiving bids from three secondary SSPs. The principal SSPs forward bids to token generation processes held at each of the four SCPs and employ price-based bid splitting.

[87] These criteria will be similar in nature to those defined for the token generation algorithms of the centralised and distributed strategies.

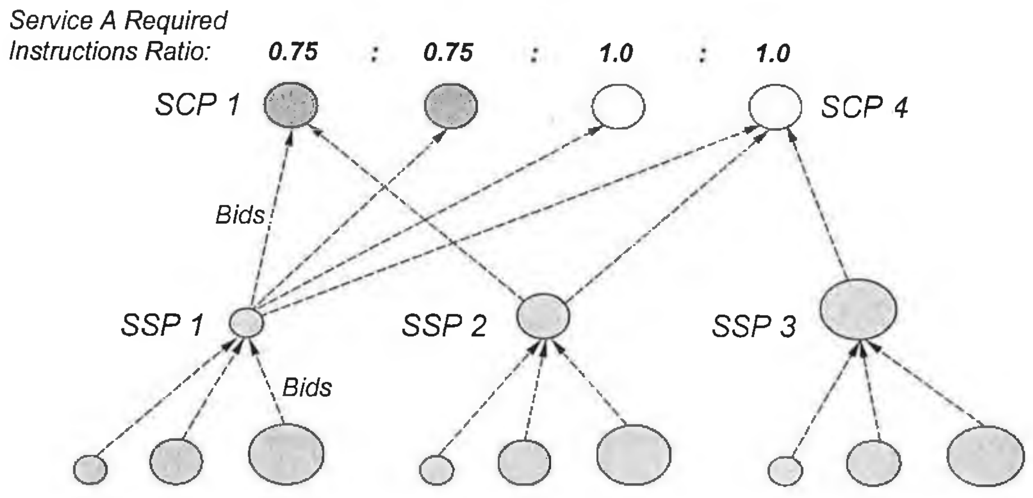

Service A Required Instructions Ratio: 0.75 : 0.75 : 1.0 : 1.0
SCP 1
SCP 4
Bids
SSP 1
SSP 2
SSP 3
Bids

Figure 6.26: Network with $I = 4$ heterogeneous SCPs, $K = 12$ SSPs, hierarchical, distributed control with 3 SSP groups.

Figure 6.27 shows the throughputs of the four SCPs and the percentage spread in throughputs between those SCPs with the smallest and largest values. We see that the strategy provides good protection from overload and note that the spread in overall throughput between SCPs 1/2 and SCPs 3/4 indicates that these SCP pairs are processing different service types. This is confirmed by Figure 6.28, which shows the per service type throughputs for SCP 1 and SCP 3. The throughputs of these two SCPs is more similar to those observed for the centralised version of the token-based strategy (see Figure 6.13, page 176) then to those observed for the distributed version (see Figure 6.24, page 184). The explanation for this is that the principal SSPs are provided with information regarding the service type processing requirements at all SCPs and use this information in their token distribution. Specifically, the use of the relative marginal cost expression results in all service A tokens being allocated from SCPs 1/2, as was the case for the centralised strategy, but not for the distributed strategy. The total completion rates and profit generated per second are shown in Figure 6.29, we see that the strategy performs as well as the centralised and distributed versions in maximising completion rates and profit.

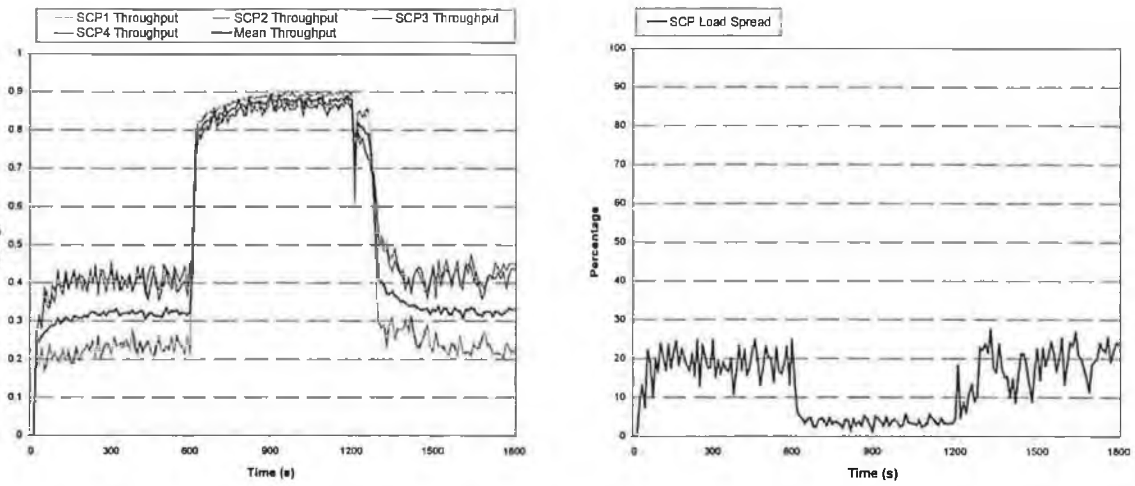
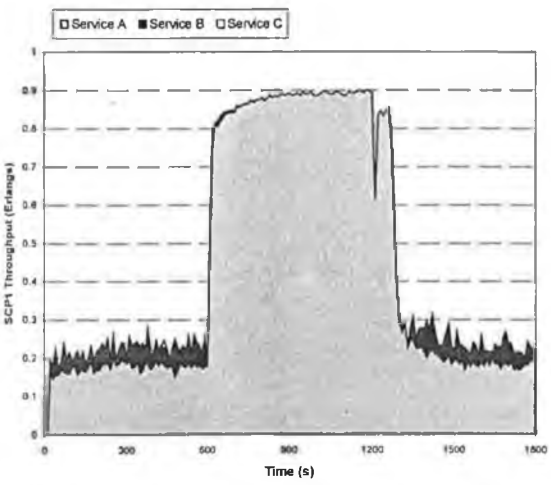


Figure 6.27: Heterogeneous SCPs, hierarchical distributed control with price-based bid splitting. Left: SCP throughputs. Right: Percentage Spread of SCP throughputs.



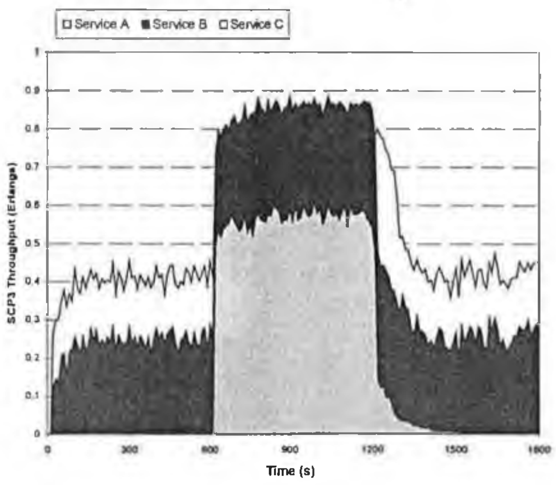
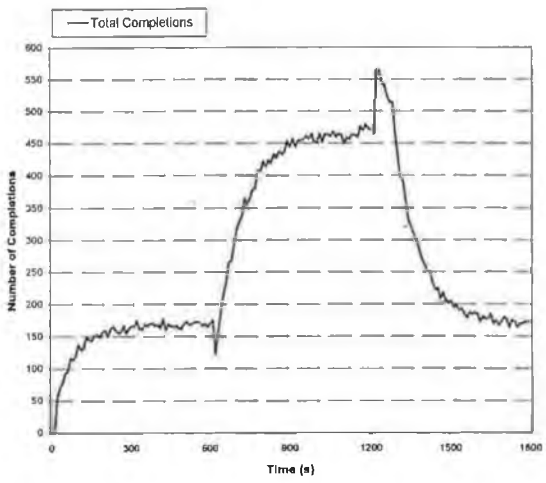


Figure 6.28: Heterogeneous SCPs, hierarchical distributed control with price-based bid splitting. Per service type SCP throughput. Left: SCP 1. Right: SCP 3.



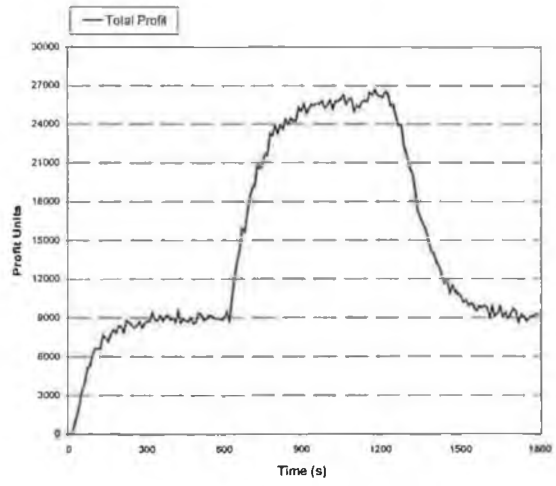


Figure 6.29: Heterogeneous SCPs, hierarchical distributed control with price-based bid splitting. Left: Total completion rates. Right: Profit generated per second.

6.2 TOKEN-BASED CONTROL OF SDP RESOURCES

Heretofore we have focussed solely on the development of a token-based strategy providing effective control of SCP processor resources. Although SCPs are in many ways the critical IN resource it is also possible that in some traffic conditions other resources, for example network databases or automated announcement facilities, become equally or more susceptible to overload. Depending on the characteristics of the overloading service(s) one of these resource types may become a network bottleneck: overloading at the same time as, or instead of, the SCPs. Resultant queue size build-up and message discarding/dropping will greatly affect end-user quality of service, despite the fact that the SCPs have controls in place to protect themselves from overload. Given the above, it is clear that a truly network-oriented load control strategy should protect multiple resource types from overload regardless of the offered traffic characteristics.

In this section we enhance the token-based strategy to incorporate profit-optimal allocation of both SCP and SDP processing resources. In the enhanced strategy SSPs are allocated two sets of tokens: one for SCPs, the other for SDPs. To accept a request for a service session requiring use of an SDP an SSP must have available one SCP token and one SDP token.[88] Allocation of SCP and SDP tokens is co-ordinated, with SCP and SDP tokens being allocated as a pair for the relevant service types. As previously, tokens are allocated on the basis of protecting resources from overload and maximisation of generated profit. The enhanced strategy is introduced and outlined in §6.2.1 and its likely performance is analysed in §6.2.2.

6.2.1 Specification of Token-based Strategy for SCP/SDP Control

A number of approaches to the generation of tokens for SDPs can be envisaged. The simplest is to allocate SDP tokens independently from SCP tokens: SSPs send bids to SDP token generation processes, which allocate tokens based on available SDP capacity and profit maximisation. These token generation processes may be centralised, distributed, or distributed hierarchical in the same manner as previously outlined for SCP token generation. This approach ensures that neither SCPs nor SDPs overload, however allocations may not be profit-optimal. For example, an SSP may receive more SCP tokens than SDP tokens for an overloading service, so when all SDP tokens are consumed the remaining SCP tokens cannot be used. In this case it would be more optimal to allocate the excess SCP tokens to a service type that did not require access to an SDP. Therefore, it is desirable to co-ordinate allocation of SCP and SDP tokens to ensure that, for the relevant service types, tokens are always allocated as a pair.

One approach to co-ordinated allocation of SCP and SDP tokens would be to employ a centralised token generation process that is supplied with information regarding expected arrival rates, profits, available SCP/SDP capacities and service-specific costs for both SCPs/SDPs. The token generation algorithm would be an extension of that specified in §6.1.2, with service related costs at SDPs being taken into account when identifying optimal allocations. As discussed previously a centralised approach is not ideal for large INs, so we instead focus on a distributed approach.

In our distributed approach SCPs each run their own token generation process and acquire SDP tokens by bidding to independent SDP token generation processes. The sequence of events is as follows: SSPs send bids to SCP token generation processes as outlined previously;[89] SCPs receive SSP bids and collate them on a per-service basis; these collated bids are submitted to the

[88] An SDP token grants access to the SDP for the duration of the service session, regardless of how many times the SDP is queried. We assume that the number of SDP queries per session is fixed for all sessions of a particular service type.

[89] SSPs may or may not be arranged in a hierarchy for bidding purposes and may use either even- or price-based splitting when bidding to SCPs.

SDP token generation process(es); SCPs receive SDP token allocations and use them as inputs into their own token generation process; SCPs generate SCP/SDP token allocations which they then send to the SSPs. SDP token generation processes can be arranged in a centralised or distributed manner; we specify and investigate a centralised scheme here.

SCPs will receive SDP tokens on a per service, but not on a per SSP basis, since collated bids were submitted to the SDP token generation process(es). The SCP can simply distribute the SDP tokens between SSPs or employ token conversion (of SDP tokens between different service types) to achieve a more profit-optimal allocation. We adopt the latter approach but note that it has the potential disadvantage that SCPs are required to have knowledge of the processing requirement ratios of the various service types at the various SDPs. During the SCP token generation process candidate allocations are identified by examining the marginal utility per marginal cost of all possible SSP/service type allocations. Where the service type requires access to an SDP candidate allocations additionally specify an SDP, with the choice between SDPs being based on minimising marginal cost (which is expressed using a relative measurement similar to that used for SCPs in §6.1.2.3).

We now proceed to specify the enhanced token-based strategy. §6.2.1.1 describes the centralised SDP token generation algorithm; §6.2.1.2 specifies the distributed SCP token generation / SDP token redistribution algorithm and §6.2.1.3 specifies the SCP/SDP token spending algorithm.

6.2.1.1 Centralised SDP Token Generation Algorithm

The centralised SDP token generation algorithm is analogous to the centralised SCP token generation algorithm specified in §6.1.2.1 (except that the marginal cost expression introduced in §6.1.2.3 is used), but with SCPs taking the place of SSPs and SDPs replacing SCPs. The notation changes as follows:

$I \to L$, where L denotes the number of SDPs in the network.

$i \to l$, where l denotes an arbitrary SDP.

$K \to I$, $i \to k$.

$J \to J_{SDP}$, where J_{SDP} denotes the number of service types requiring access to the SDP.

$j \to j \in \{J_{SDP}\}$, where $\{J_{SDP}\}$ denotes the set of service types requiring access to the SDP.

$r_k(j) \to r(j)$[90], where $r(j)$ denotes the profit generated by a successful session of type j.

$c_i \to c_l$, where c_l denotes the total processing capacity available at SDP l.

[90] For the SDP token generation algorithm we assume that profit values for each service type are the same at all SSPs in the network.

$s_i \rightarrow s_l$, where s_l denotes the capacity sold on behalf of SDP l.

$p_i(j) \rightarrow p_l(j)$, where $p_l(j)$ denotes the processing costs of service type j at SDP l.

$q_k(j) \rightarrow q_i(j)$, where $q_i(j)$ denotes the bid of SCP i for tokens of service type j.

$n_{i,k}(j) \rightarrow n_{l,i}(j)$, where $n_{l,i}(j)$ denotes the number of service type j allocated to SCP i on behalf of SDP l.

$n_k(j) \rightarrow n_i(j)$, where $n_i(j)$ denotes the total number of service type j tokens allocated to SCP i.

$c_{i,k}(j) \rightarrow c_{l,i}(j)$, where $c_{l,i}(j)$ denotes the price of an (l,i,j) allocation.

$u_k(j) \rightarrow u_i(j) = r(j) \sum_{w=n_i(j)}^{\infty} \frac{q_i(j)^w}{w!} e^{-q_i(j)}$, where $u_i(j)$ denotes the marginal utility associated with allocating a type j token to SCP i.

$v_i(j) \rightarrow v_l(j) = \sum_{j'=1}^{J_{SDP}} \sum_{i'=1}^{I} \frac{p_l(j)}{p_l(j')} u_{i'}(j')$, where $v_l(j)$ denotes the marginal cost of allocating a type j token on behalf of SDP l.

$\delta_{i,k}(j) \rightarrow \delta_{l,i}(j) = u_i(j)/v_l(j)$, where $\delta_{l,i}(j)$ denotes the marginal utility per marginal costs of a (l,i,j) allocation.

6.2.1.2 Distributed SCP Token Generation / SDP Token Redistribution Algorithm

We now specify the SCP token generation algorithm, using the notation defined previously and additional notation defined as the algorithm specification proceeds. The variable i is used to denote the SCP at which the algorithm executes. We assume that SCPs send bids to a centralised SDP token generation process (these bids need not be split).

Step 1: SDP bid dispatch.

```
For all service types $j \in \{J_{SDP}\}$ do:
    Set $q_i(j) = \sum_{k=1}^{K} q_k(j)$.
    Submit bids $q_i(j)$ to SDP token generation process.
...
Receive token allocations $n_{l,i}(j)$ from SDP token generation process.
...
```

Step 2: Initialisation.

```
For all service types $j = 1,\ldots,J$ do:
    For all SSPs $k = 1,\ldots,K$ do:
        Set SCP token allocations $n_{i,k}(j) = 0$ and costs $c_{i,k}(j) = 0$.
        Set $\pi_k(j) = e^{-q_k(j)}$.
        Set $\Pi_k(j) = 1 - \pi_k(j)$.
        Set marginal utilities $u_k(j) = r_k(j)\Pi_k(j)$.
For all service types $j = 1,\ldots,J$ do:
    Set SCP marginal costs $v_i(j) = p_i(j)$.[91]
Set remaining SCP processing capacity $s_i = c_i$.
For all SSPs $k = 1,\ldots,K$ do:
    For all service types $j \in \{J_{SDP}\}$ do:
        For all SDPs $l = 1,\ldots,L$ do:
            Set $n_{l,k}(j) = 0$, where $n_{l,k}(j)$ denote the number of type $j$ tokens allocated on behalf
            of SDP $l$ to SSP $k$.
For all service types $j \in \{J_{SDP}\}$ do:
    For all SDPs $l = 1,\ldots,L$ do:
        Set $m_l(j_{l,\min}) = \sum_{j=1}^{J_{SDP}} \frac{p_l(j)}{p_l(j_{l,\min})} n_{l,i}(j)$, where $j_{l,\min}$ identifies the service type with the lowest
        value of processing costs at SDP $l$ and $m_l(j_{l,\min})$ is the pool of available $j_{l,\min}$ tokens for
        SDP $l$ that may be converted and distributed by the algorithm.
For all SDPs $l = 1,\ldots,L$ do:
    For all service types $j \in \{J_{SDP}\}$ do:
        Set $w_l = \sum_{j=1}^{J_{SDP}} \sum_{k=1}^{K} \frac{u_k(j)}{p_l(j)}$.
For all SDPs $l = 1,\ldots,L$ do:
```

[91] As noted previously in the situation where tokens for a single resource are being allocated there is no necessity to employ a relative measure for marginal costs.

```
 For all service types $j \in \{J_{SDP}\}$ do:
     Set $v_l(j) = p_l(j)w_l$.
Set $\alpha = 0$, where $\alpha$ is the maximum value of $\delta_{i,k}(j)$ over all $(j,k)$ allocations previously
examined.
```

Step 3: Identify optimal allocations.

```
For all SSPs $k = 1,...,K$ and service types $j = 1,...,J$ do:
    If $j \notin \{J_{SDP}\}$ then:
        If $\delta_{i,k}(j) > \alpha$ then:
            Set $\alpha = \delta_{i,k}(j)$.
    Else:
        If there exists at least one SDP $l$ for which $m_l(j_{l,\min}) > 0$ do:
            If $\delta_{i,k}(j) > \alpha$ then:
                Set $\alpha = \delta_{i,k}(j)$.
                Set $\beta = \infty$, where $\beta$ is the minimum value of $v_l(j)$ over all $l$ allocations
                previously examined.
                For all SDPs $l = 1,...,L$ do:
                    If $m_l(j_{l,\min}) > 0$ then:
                        If $v_l(j) < \beta$ then:
                            Set $\beta = v_l(j)$.
For all SSPs $k = 1,...,K$ and service types $j = 1,...,J$ do:
    If $j \notin \{J_{SDP}\}$ then:
        If $\delta_{i,k}(j) = \alpha$ then:
            List $(j,k)$ as a candidate allocation.
    Else:
        If there exists at least one SDP $l$ for which $m_l(j_{l,\min}) > 0$ do:
            If $\delta_{i,k}(j) = \alpha$ then:
                For all SDPs $l = 1,...,L$ do:
                    If $m_l(j_{l,\min}) > 0$ then:
```

```
            If $v_l(j) = \beta$ then:
                List $(j,k,l)$ as a candidate allocation.
```

Step 4: Arbitrate between optimal allocations.

```
    For SSPs $k = 1,\ldots,K$, service types $j = 1,\ldots,J$ and SDPs $l = 1,\ldots,L$ do:
        If allocation $(j,k)/(j,k,l)$ is in the candidate list then:
            Apply secondary criteria to eliminate less optimal candidates.[92]
            Randomly select allocation $(j',k')/(j',k',l')$ from those remaining in the candidate list.
```

Step 5: Perform optimal allocations.

```
    Set $n_{i,k'}(j') = n_{i,k'}(j') + 1$.
    Set $c_{i,k'}(j') = c_{i,k'}(j') + u_{k'}(j')$.
    Set $s_i = s_i - p_i(j')$.
    If $j \in \{J_{SDP}\}$ then:
        Set $n_{l',k'}(j') = n_{l',k'}(j') + 1$.
        Set $m_l(j_{l,\min}) = m_l(j_{l,\min}) - \frac{p_{l'}(j')}{p_{l'}(j_{l',\min})}$
```

Step 6: Update internal variables.

```
    Set $\pi_{k'}(j') = \pi_{k'}(j') q_{k'}(j') / n_{k'}(j')$.
    Set $\Pi_{k'}(j') = \Pi_{k'}(j') - \pi_{k'}(j')$.
    Set $u_{k'}(j') = r_{k'}(j') \Pi_{k'}(j')$.
    If $j' \in \{J_{SDP}\}$ then:
        For all SDPs $l = 1,\ldots,L$ do:
            Set $w_l = w_l - r_{k'}(j') \frac{\pi_{k'}(j')}{p_l(j')}$.
        For all SDPs $l = 1,\ldots,L$ do:
            For all service types $j \in \{J_{SDP}\}$ do:
```

[92] Criteria such as those specified in §6.1.2.1 can be applied, as well as similar criteria governing the optimal manner in which to allocated SDP tokens.

```
        Set $v_l(j) = p_l(j)w_l$.
```

Step 7: Loop statement.

```
    If there exists at least one SCP $i$ for which $s_i > 0$ then:
        GOTO Step 3.
    Else:
        STOP.
```

6.2.1.3 SCP/SDP Token Spending Process

The SCP/SDP token spending process is analogous to that specified in §6.1.2.2, except that, when a request is accepted, an SDP as well as an SCP must be chosen. SDPs for which more tokens are currently available are again chosen in preference to other SDPs.

Step 1: Apply PT throttle.

```
Select a random number $X$ uniformly distributed in the range $(0.0, 1.0)$.
If $X < p_a(j)$ then:
    Service request is accepted.
Else:
    Service request is throttled.
    STOP.
```

Step 2: Select SCP.

```
    For all SCPs $i = 1,...,I$ do:
        List all SCPs which maximise $n'_i(j)$, where $n'_i(j)$ is the number of tokens remaining in
        token pool $ij$.
    Randomly select SCP $i'$ from the list.
```

Step 3: Select SDP.

```
    For all SDPs $l = 1,...,L$ do:
        List all SDPs which maximise $n'_l(j)$, where $n'_l(j)$ is the number of tokens remaining in
        token pool $lj$.
    Randomly select SCP $l'$ from the list.
```

6.2.2 Performance Analysis of Enhanced Strategy

The network scenario illustrated in Figure 6.30 was used to analyse the performance of the enhanced token-based strategy described in the previous section. The network consists of four identical SCPs and two SDPs having different processing requirements for service A (but not for service C). All SCPs and all SDPs can be used by all of the twelve SSPs in the network. We simulate two global overloads, in which the arrival rates of service A and service C respectively increase to a level corresponding to a first-offered load of 1.4 Erlangs to each SCP; end-user reattempt behaviour is modelled.

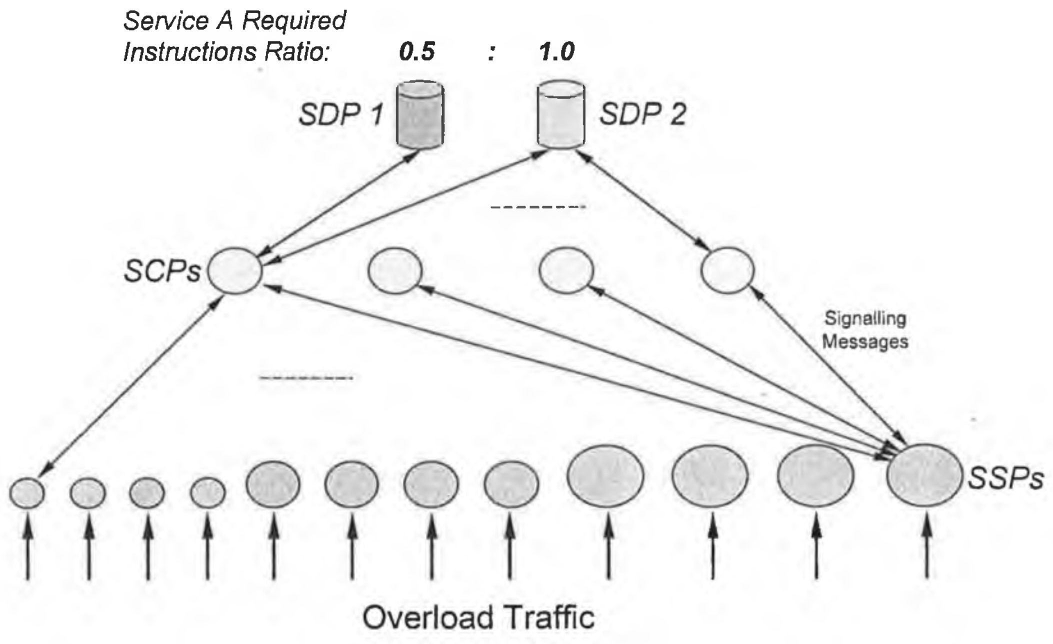


Figure 6.30: Network with $I = 4$ identical SCPs, $L = 2$ heterogeneous SDPs, $K = 12$ SSPs, global overload.

In the simulation model used previously the functionality associated with the SDF was incorporated into the SCP model (although SDF communication/processing was assumed not to require any SCP central processor instructions). Here we employ a separate SDP model that is similar in nature to the SCP one. It contains a single central processor that has the capacity to execute $c = 6{,}200{,}000$ instructions per second. Signalling messages arriving at the SDP require the execution of a different number of instructions, depending on their type; details of these requirements are provided in Table 6.1. The SDP has a single input queue that can hold a maximum of 500 messages (we assume that all messages are of the same size). If the queue is full then all arriving messages are discarded, with no associated processing cost. If messages have been waiting in the queue for longer than $t = 2s$ by the time they commence processing they are dropped, a process that involves expending 24,000 instructions. The SDP contains a single functional block: the SDF, which models the reading and updating of database entries. The SCP model is the same as that used previously except that messages requiring SDF processing are forwarded to the SDP model; this process is assumed not to have any effect on SCP processing requirements, which have the same values as those used previously.

Table 6.1: SDP processing requirements.

Arr. Signalling Message	Dep. Signalling Message	SDP 1 Instns.	SDP 2 Instns.
sA2	sA3	30,000	50,000
sC6	sC7	60,000	50,000

We first investigate the operation of the enhanced strategy in the presence of a service A overload. The relative capacities of the four SCPs and two SDPs are such that the SCPs are capable of processing more service A sessions per unit time then are the SDPs. Therefore, if the SDP utilisation is to be maintained below the target of 0.9 Erlangs during a service A overload the SCPs will have to have their utilisation maintained at a value significantly below their target. In effect the SDPs become the network bottleneck: it is their maximum capacity which dictates the maximum completion rate that can be achieved.

Figure 6.31 shows the offered SCP load, mean SCP throughput and mean SDP throughput for the duration of the simulation run.[93] We see that the enhanced token-based strategy succeeds in ensuring that the throughput of the SDPs is kept close to the target value of 0.9 Erlangs. Moreover, this is achieved by keeping the throughput of the SCPs at approximately 0.75 Erlangs. SCP throughputs are limited because the SCP token generation / SDP token redistribution process can only allocate as many service A token pairs as correspond to the token allocations received from the SDP token generation process. When SDP token allocations are exhausted the SCPs allocated their remaining capacity to services not requiring SDP access, in this case service B.

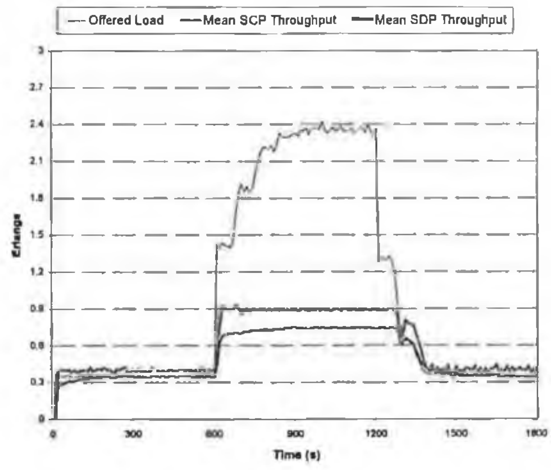


Figure 6.31: Identical SCPs, heterogeneous SDPs. Service A overload. Offered SCP load, mean SCP throughput and mean SDP throughput.

[93] As with previous cases the token-based strategy ensures that resource queues do not grow to the level where message dropping/discarding occurs, hence SCP/SDP throughput is equivalent to SCP/SDP carried load.

Figure 6.32 shows the per-service throughputs for SCP 1 and SCP 3. We see that the throughput of the SCPs is as expected: all service requests are accepted in normal conditions and SCP throughputs are well balanced, whereas during overload all service C sessions are throttled, all service B requests are accepted and the number of service A requests accepted is limited to the number that can be handled by the SDPs.

Figure 6.33 shows the per-service throughput for SDP 1 and SDP 2. In normal conditions we see that SDP throughputs are unbalanced, due to their differing processing requirements for service A. This means that more of the service A tokens generated are for SDP 1 and the majority of the service C tokens generated are for SDP 2. The token spending process is such that the SDP for which an SSP has the most tokens is used in preference to the other SDP. Hence we see that the service A throughput of SDP 1 is greater than that for SDP 2 and *vice versa* for service C. During overload we see that both SDPs allocate tokens for service A in preference to the lower profit service C and that end-user reattempt behaviour has the effect of prolonging the overload beyond the time of abatement of first-offered overload traffic.

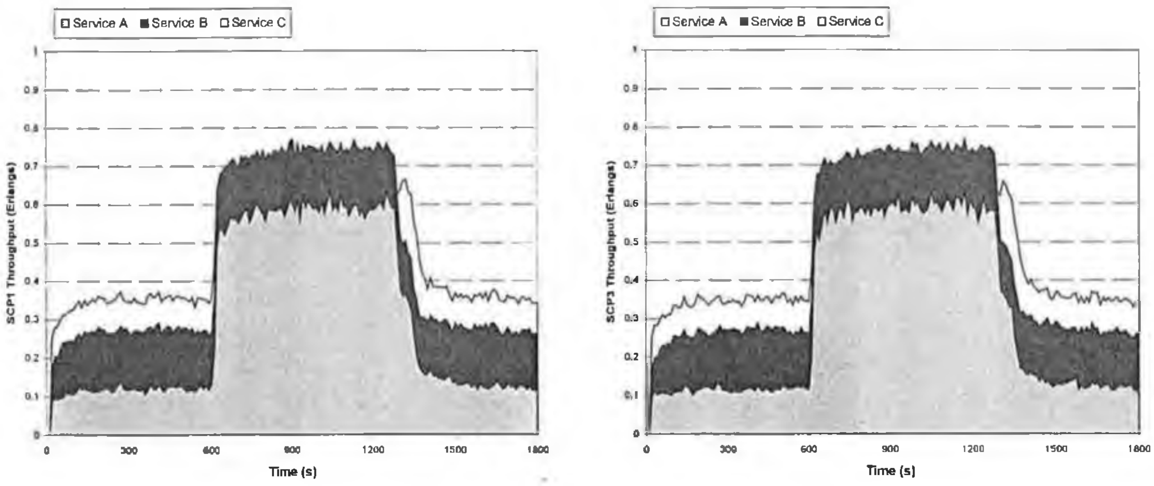


Figure 6.32: Identical SCPs, heterogeneous SDPs. Service A overload. Per service type SCP throughput. Left: SCP 1. Right: SCP 3.

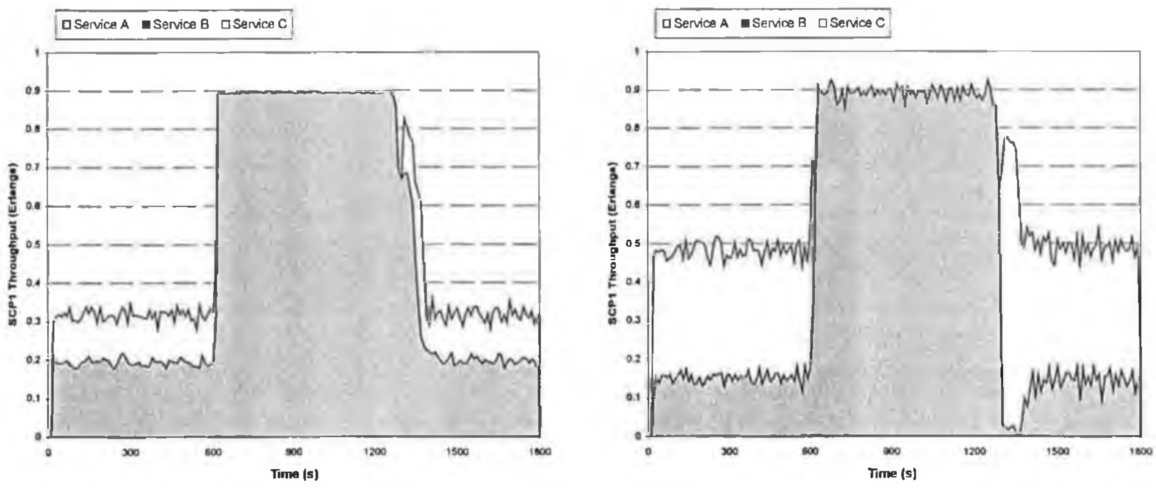


Figure 6.33: Identical SCPs, heterogeneous SDPs. Service A overload. Per service type SDP throughput. Left: SDP 1. Right: SDP 2.

We now investigate the operation of the strategy in the presence of a service C overload. As before, the relative capacities of the SCPs and SDPs mean that SCPs are capable of processing more service C sessions per unit time than the SDPs. Figure 6.34 shows the offered SCP load, mean SCP throughput and mean SDP throughput for the duration of the simulation run. Once again the SDP throughput is kept below the target value (a little further below than was that case for service A overload) while the SCP throughputs are kept at the corresponding level (approximately 0.53 Erlangs).

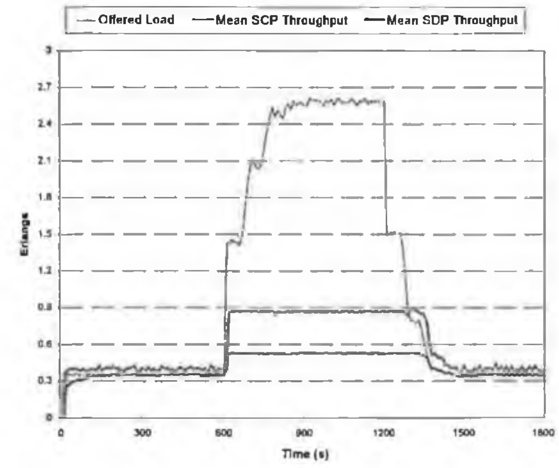


Figure 6.34: Identical SCPs, heterogeneous SDPs. Service C overload. Offered SCP load, mean SCP throughput and mean SDP throughput.

Figure 6.35 shows the per-service throughputs for SCP 1 and SCP 3. As expected all service A and service B requests are accepted during overload, while the number of accepted service C requests is limited in accordance with the available SDP capacity. Figure 6.36 shows the per-service throughput for SDP 1 and SDP 2. In normal conditions we again observe an imbalance in SDP throughput caused by unequal token allocations resulting from differing service A processing capacities. During overload we see that all service A tokens are generated for SDP 1, which has the better service A performance. Most of the remaining SDP 1 capacity and all of the SDP 2 capacity is used for service C sessions. The shortfall below the target of SDP 1 throughput is due to a larger number of tokens for service A being generated than originally bid for by the SSPs (the large difference between service A and service C profit means that the token generation algorithm deems it more profit-optimal to allocate capacity to service A rather than service C.)

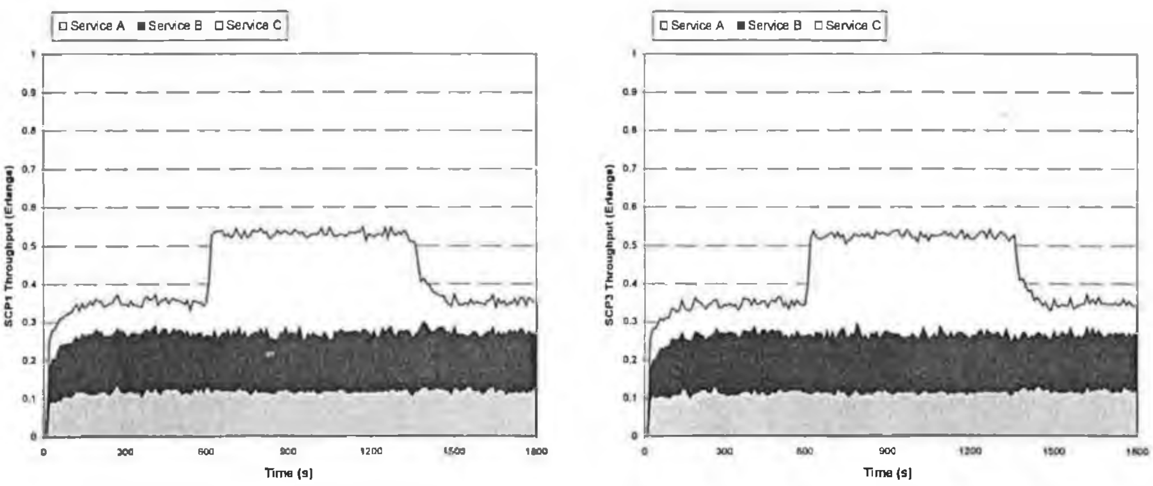


Figure 6.35: Identical SCPs, heterogeneous SDPs. Service C overload. Per service type SCP throughput. Left: SCP 1. Right: SCP 3.

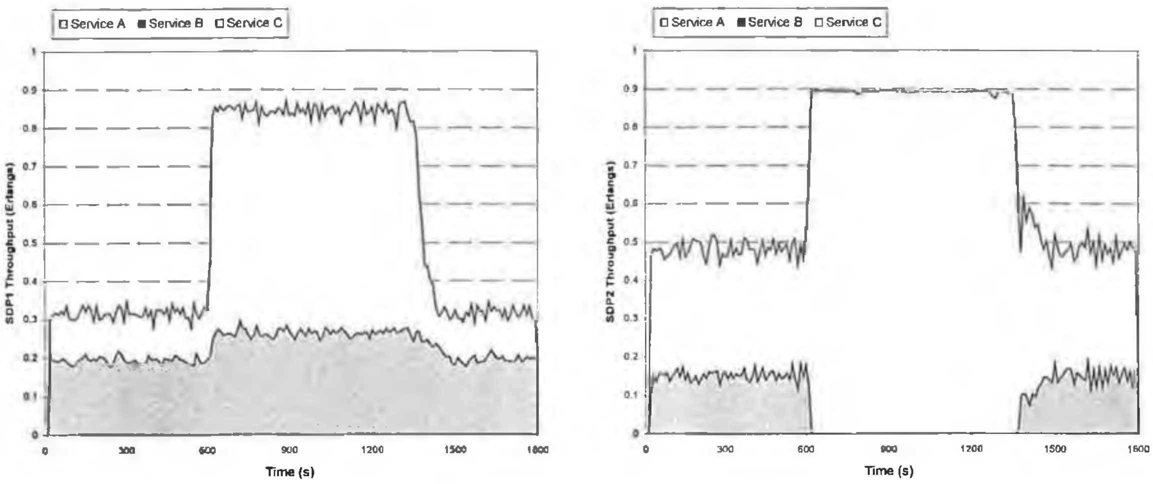


Figure 6.36: Identical SCPs, heterogeneous SDPs. Service C overload. Per service type SDP throughput. Left: SDP 1. Right: SDP 2.

6.3 TOKEN-BASED CONTROL OF SS.7 RESOURCES

In the analyses of the token-based strategy presented thus far we have focussed on overload of IN resources only and ignored the potential for SS.7 resource overload. As shown in chapter 4, overload of SS.7 links (or indeed STPs) can have a severely detrimental impact on network performance. The current token-based strategy does not incorporate any means for detecting or alleviating SS.7 overload, hence it can not prevent the unacceptable response delays and premature session terminations resulting from message discarding at overloaded links. To achieve fully network-oriented load control it is therefore necessary to enhance the token-based strategy to help prevent and/or respond to SS.7 overloads.

In theory it would be possible to use an approach similar to that outlined in §6.2 to enhance the token-based strategy to allocate tokens for individual links and STPs. Unfortunately, this would be highly impractical, since the SS.7 static routing scheme and capabilities for re-routing during failure conditions would necessitate very complex token generation algorithms and bidding hierarchies for even moderately sized networks. For this reason we must develop a simpler approach that does not require detailed knowledge of the SS.7 network topology or routing scheme.

One of the conclusions of chapter 4 was that use of message delay measurements appears to be the most practical means for SSPs to detect overload in the SS.7 network. In the strategy enhancement presented here round-trip delay measurements are employed to select 'favourable' SCPs from individual SSP's viewpoint and to detect if the routes through the SS.7 network towards an SCP (or the SCP itself) are overloaded. This information is used in enhanced bid splitting, load throttling and token spending processes; collectively these help maximise network performance in situations where the potential for SS.7 overloads is high.

§6.3.1 describes how round-trip delays experienced by special *ant* messages are used to select favourable SCPs; and how this information can be used in the bid splitting and token spending processes. §6.3.2 describes an enhanced load throttling processes that reduces traffic to destination SCPs where response delays exceed pre-specified thresholds. Finally, §6.3.3 presents an analysis of the performance of the enhanced strategy based on results obtained using a modified simulation model.

6.3.1 Pheromone-based SCP Selection

In a multiple-SCP network SSPs will ideally use those SCPs which, because of their physical proximity or prevailing network load conditions, provide better quality of service then other SCPs. For example, in normal load conditions an SSP should use those SCPs to which it is connected by the smallest number of signalling links; this both minimises response delays for end-users and reduces the probability of message loss. Similarly, if one or more signalling links are overloaded, SSPs should use those SCPs the routes towards which avoid these links.

SSP measurements of the delay between the time at which an IN query message is sent to an SCP and the time at which the corresponding response message is received, provide relative estimates of how 'favourable' it is for that SSP to use particular SCPs for future service sessions. During normal conditions the delay measured for SCPs that are in close proximity and/or process messages more promptly will be less then those measured for other SCPs. During overload (of signalling links, STPs, or of SCPs themselves) messages towards affected SCPs will encounter saturated queues and be delayed significantly. The approach we use to exploit the information provided by delay measurements employs an approach based on Ant Colony Optimisation (for a description of this resource allocation paradigm *cf.* §3.2.3.)

6.3.1.1 Specification of Pheromone-based SCP Selection Process

We use the concept of pheromone tables to select favourable SCPs based on SSP delay measurements. The formulae used for updating pheromone probabilities are based on those outlined in [Schoonderwoerd *et al.*, 1997]. The details of the approach are as follows:

- At intervals of length T_{ant}, for every service type j an ant message is generated at every one of the SSPs in the network and sent to a selected SCP;
- At each SSP k a pheromone table for each service type j is maintained. The pheromone tables contain entries for all the SCPs in the network; the entries are the probabilities P_{kij}, of SSP k choosing SCP i as the destination for a service type j ant message;
- The destination SCP of an ant message is selected using the information in the pheromone table, following one of two schemes: either the 'normal scheme' or the 'exploration scheme.' The scheme used is selected at random, but with the probability of using the exploration scheme, $P_{explore}$, being much less than that for the normal scheme, P_{normal};
- In the normal scheme the SCP is selected randomly, the probability of picking SCP i being the probability P_{kij} indicated in the pheromone table; this is analogous to the trail selection behaviour of real ants;
- In the exploration scheme the SCP is also selected randomly, however the probabilities of selecting all the SCPs are equal. The purpose of the exploration scheme is to introduce an element of noise into the system so that more performant SCPs can be found as network conditions change; this is analogous to a small number of real ants which do not follow the 'recommended' trail, but which, as a result, may find even better trails;
- Each ant message is routed to the designated SCP, which returns it directly to the originating SSP. The round-trip delay, as measured by the SSP, is used to update the pheromone table entries for the relevant service type. The calculations are as follows:

  $$P_{kij} = \frac{P_{kij} + \Delta P}{1 + \Delta P}$$

  $$P_{ki'j} = \frac{P_{ki'j}}{1 + \Delta P} \quad \forall\ i' \in [1, I], i' \neq i$$

  where i indicates the visited SCP and

  $$\Delta P = \frac{A}{t_{ant}} + \frac{B}{c_i(j)} + C$$

  where A, B and C are constants; t_{ant} is the measured round trip delay for the newly returned ant message; and $c_i(j)$ is the closing price per service type j token at SCP i as indicated to the SSP in the last received token allocation.[94]

- For each service type at each SSP, the SCP selected as being most favourable is the one for which the pheromone table entry is the largest.

We note that a pheromone probability can only be reduced following an increase in another pheromone probability and that, since this reduction is achieved by multiplying by a non-zero factor of value less than one the value will never reach zero. In addition, when a pheromone probability is increased the absolute and relative increase will be larger the smaller the initial probability value. This means that information relating to SCPs which are not currently favoured is amplified, so that the process of finding more favourable routes when network conditions changes is achieved quickly.

6.3.1.2 Use of Pheromone Probabilities for Token Spending and Bid Splitting

Pheromone probabilities can be used to control the order in which SSPs use the tokens allocated to them. It is desirable to use those tokens associated with SCPs with larger pheromone values first. These SCPs are the ones currently most favoured for that service type, because of their performance, proximity to the SSP, or the state of the routes that messages traverse to reach them. This approach is particularly beneficial in periods of normal traffic loading (when the majority of allocated tokens will not be used) as it ensures that favoured SCPs are given priority. §6.3.2 provides a specification of an enhanced token spending process incorporating pheromone-based SCP selection.

In a distributed token-based strategy pheromone probabilities can be used directly as the weight values $a_{i,k}(j)$ that dictate the amount of expected demand is divided between bids destined for different SCP token generation processes. Basing bid splitting on pheromone probabilities means that the SSP is more likely to receive tokens for those SCPs it currently favours most. The approach also has the advantage that bid splitting will automatically adjust to changes in network conditions, for example if an SCP becomes isolated due to SS.7 resource failure or overloaded links/STPs it will not receive significant bids until conditions improve.

6.3.2 DELAY-BASED LOAD THROTTLING

The SCP selection approach described above provides SSPs with information allowing them to avoid sending messages through overloaded regions of the network. However, it does not explicitly incorporate throttling of service requests destined for overloaded or unavailable

[94] When pheromone probabilities are being used to split bids (*cf.* §6.3.1.2) the use of closing prices in the pheromone updating calculation encourages the SSPs, to send approximately even bids to all SCPs in the network. Were this not used, SCPs would receive uneven bids, resulting in sub-optimal token allocations. For example, in our scenario SCPs 2/3 would receive smaller bids for all service types then would SCPs 1/4 and as a result during overload would allocate tokens to lower profit services whilst SCPs 1/4 would not be able to satisfy demand for higher profit tokens.

destinations. In severe failure or overload conditions messages may not be able to reach one or more destination SCPs, so these SCPs should not be sent any further requests until the overload abates.

To achieve explicit throttling of requests during network overloads, SSPs can monitor the response delays for departed (IN and ant) messages. Once response delays for a pre-specified number of departed messages, M, dispatched to SCP i, exceed a pre-specified threshold value no further requests are accepted for that SCP. The threshold delay[95] for detection of overload, denoted T_{onset}, can be chosen on the basis of the customer expectations of response delays for IN services; example values are outlined in [MacDonald and Archambault, 1994]. Alternatively, a purely load control oriented approach can be taken: by measuring mean response delays when the network is running at target capacity and setting the thresholds somewhat higher than the highest of these mean values.

The pheromone-based SCP selection process continues to send ant messages to all SCPs, including those for which no requests are currently being accepted (these SCPs will have very small pheromone probabilities but will have ant messages sent to them through the exploration scheme). These ant messages provide a means of detecting abatement of overload for affected SCPs: when an ant message returns within a pre-specified threshold delay,[96] denoted T_{abate}, service requests can again be accepted for the visited SCP.

To incorporate delay-based load throttling into the token-based strategy we can adopt a three-step load throttle. The first step involves the delay-based throttle described above being used to create a list of SCPs for which requests can currently be accepted for the service type in question. If there are no such SCPs the request is throttled. In the second step a PT throttle is applied as previously in order to regulate token usage. In the final step, the token pools for the created list of SCPs are examined and the request is accepted if tokens are available. The SCP having the highest pheromone probability of those SCPs for which tokens remain is selected. This approach leads to the following modified token spending process:

Step 1: Identify list of potential SCPs.

Create a list $\{I_{accept,j}\}$ of all SCPs for which requests of type j can be accepted.

If $\{I_{accept,j}\}$ is empty then:

[95] Clearly different service types involve different processing demands for message pairs, so ideally the delay threshold would be service and message specific. We assume that the threshold value will be significantly larger than the typical response delay for all message pair types, hence it is not necessary to support the additional complexity of maintaining a large number of threshold values.

[96] This threshold can be set at a lower level than the detection threshold in order to introduce hysteresis to prevent rapid oscillation in the rejection/acceptance of requests.

```
    Service request is throttled.
    STOP.
```

Step 2: Apply PT throttle.

```
Select a random number $X$ uniformly distributed in the range $(0.0, 1.0)$.
If $X > p_a(j)$ then:[97]
    Service request is throttled.
    STOP.
```

Step 3: Examine Token Pools.

```
Set $\upsilon = 0$, where $\upsilon$ is the maximum pheromone probability over those previously examined.
For all SCPs $i \in \{I_{accept,j}\}$ do:
    If $n'_i(j) > 0$, where $n'_i(j)$ is the number of tokens remaining in token pool $ij$, then:
        If $P_{kij} > \upsilon$ then:
            Set $\upsilon = P_{kij}$.
For all SCPs $i \in \{I_{accept,j}\}$ do:
    If $n'_i(j) > 0$ then:
        If $P_{kij} = \upsilon$ then:
            Service request is accepted and routed towards SCP $i$.
            Set $n'_i(j) = n'_i(j) - 1$.
            STOP.
If $\upsilon = 0$ then:
    Request is throttled.
    STOP.
```

[97] When $p_a(j)$ is updated the calculation should not take into account tokens for SCPs not currently members of the $\{I_{accept,j}\}$ list.

6.3.3 Performance Analysis of Enhanced Strategy

In this section we present the results of a simulation study of the performance of the token-based strategy, enhanced to incorporate delay-based throttling, pheromone-based bid splitting and pheromone-based token spending. The study necessitated updating the simulation model used previously to incorporate SS.7 signalling link and STP models. We analyse the performance of the strategy both during overload and in the presence of link/STP failures.

6.3.3.1 Description of Updated IN Simulation Model

To enable analysis of the performance of the enhanced strategy, SS.7-specific components of the simulation model used in chapter 4 were added to the simulation model described in §5.3. The resulting model incorporates SSPs, SCPs, STPs and signalling links, which can be arranged into arbitrary network topologies. The particular topology used for this study is shown in Figure 6.37.

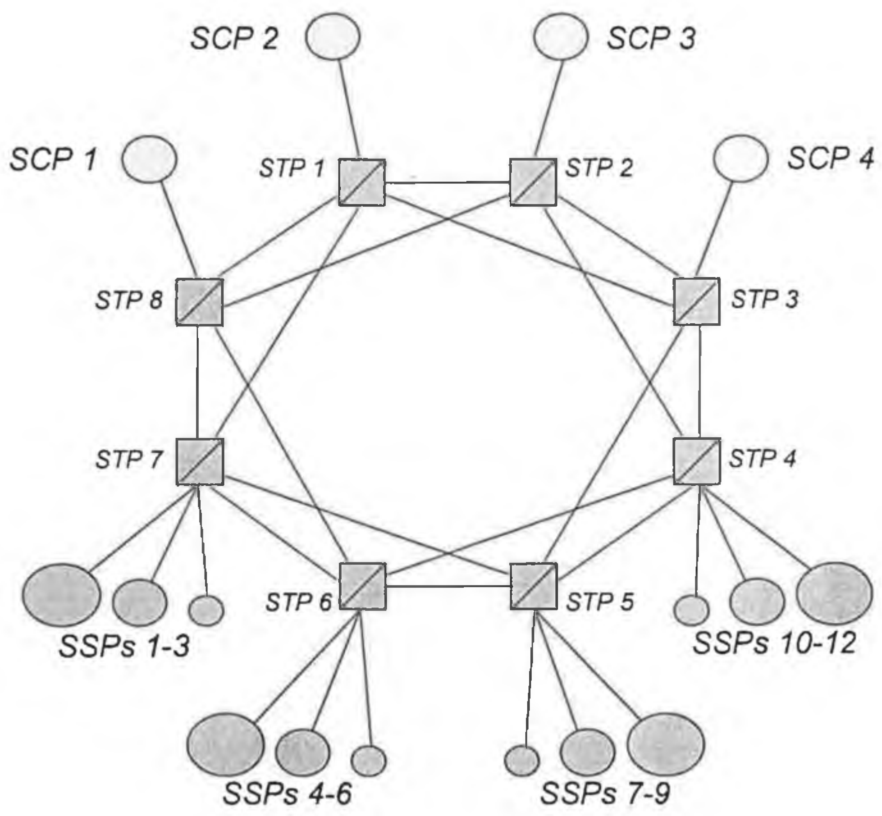


Figure 6.37: Network used for updated simulation model.

As can be seen from Figure 6.37 in the updated model there is, for simplicity, only a single signalling link connecting each SSP/SCP to an STP (as noted in chapter 4 SSPs/SCPs are usually connected to at least two STPs to provide redundancy). All messages between two signalling points follow the same route and no alternative routes are in place in case of link/STP failure. All functionality associated with the SS.7 load controls is omitted from the updated model. STPs are arranged in a mesh topology, with each STP being connected to its four nearest neighbour STPs. STP routing tables are configured to provide the shortest possible routes between SSPs and SCPs. In addition, the routing plan ensures that the load on signalling links is approximately balanced across the network in conditions where the same volume of traffic arrives at each of the four SSP groups.

The proximity of SSPs to SCPs in terms of the number of links that must be traversed by messages travelling in either direction is listed in Table 6.2. From this table we can conclude that, in conditions where no signalling links or STPs are overloaded, SSPs 1-3 should (if possible) use either SCP 1 or SCP 2, SSPs 4-6 should use SCP 1, SSPs 7-9 should used SCP 4 and SSPs 10-12 should use SCP 3 or SCP 4.

Table 6.2: Proximity, in numbers of links to be traversed, of SCPs to SSPs.

SSP Group	SCP 1	SCP 2	SCP 3	SCP 4
SSPs 1-3	3	3	4	4
SSPs 4-6	3	4	4	4
SSPs 7-9	4	4	4	3
SSPs 10-12	4	4	3	3

For the simulation runs described here ant messages are generated at intervals of $T_{ant} = 0.25s$ and the values of the constants used for pheromone probability updating are $A = 0.00025$, $B = 0.00025$ and $C = 0.0$. At the start of the runs the pheromone values are set equal for all SCPs ($P_{kij} = 0.25 \ \forall \ k,i,j$). The exploration probability was set to $P_{explore} = 0.05$. For delay-based throttling the number of messages that must be delayed beyond the threshold delay for an SCP to be deemed overloaded was set to $M = 3$. The onset and abatement threshold delays were set to $T_{onset} = 2s$, $T_{abate} = 1s$ respectively. The network was dimensioned so that signalling links do not overload (for convenience we simulate STP/link failures as opposed to overloads). We assume that all IN signalling messages are of length 100 Bytes and that ant messages are of length 25 Bytes.

6.3.3.2 Operation During Overload Conditions

To compare performance of the original and enhanced versions of the strategy during normal and overload conditions we simulate a global overload of service A to a level corresponding to a first-offered load of 1.4 Erlangs to each SCP. End-user reattempt behaviour is modelled and all SCPs have the same performance characteristics for all service types.

Figure 6.38 shows the mean and individual throughputs of the four SCPs for the original and enhanced strategy. For the original strategy we see that the throughputs of all four SCPs are well balanced during both normal and overload conditions. This is not the case for the enhanced strategy. In normal conditions the throughputs of the SCPs are not balanced, however they reflect the proximity of the SCPs to the various SSP groups. SSPs 1 and 4 are both three links away from two separate SSP groups, whereas SSPs 2 and 3 are both three links away from only one SSP group. For this reason the operation of the pheromone-based SCP selection process means that more SSPs favour SCPs 1 and 4 over SSPs 2 and 3, resulting in the former pair having higher throughputs during normal load. During overload the throughputs of the SCPs are much more evenly balanced, because SSPs are forced to use tokens for less favoured SCPs once all tokens for their favoured SCP are consumed. The sharp dip in throughputs at the end of the first-offered overload is due to sudden drop in the number of offered requests, which causes under-bidding to some SCPs at the same time as over-bidding to others.

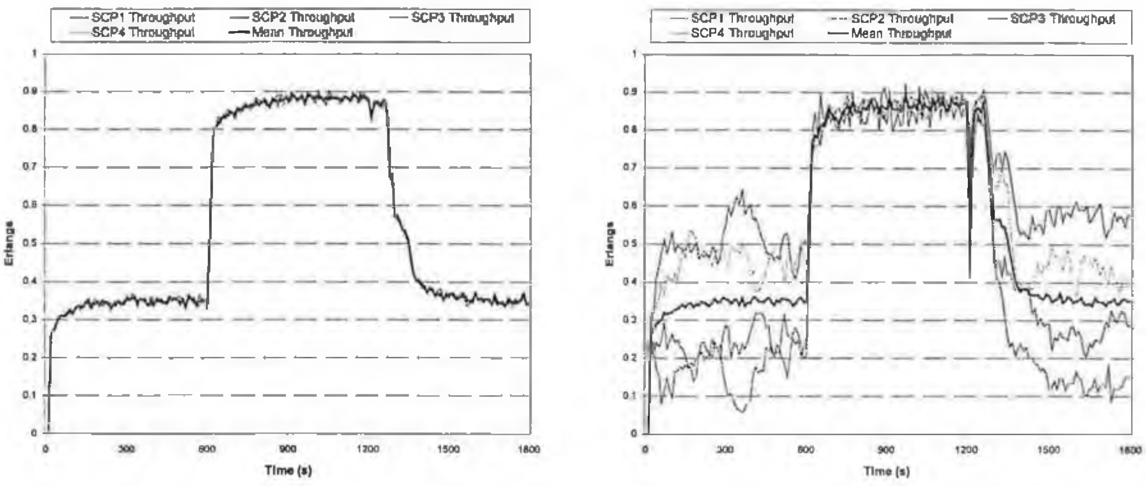


Figure 6.38: SCP mean and individual throughputs. Left: Original strategy. Right: Enhanced strategy.

Figure 6.39 and Figure 6.40 demonstrate which SCPs are favoured by the various SSP groups; they show the per source SSP group throughputs of the SCPs for the original and enhanced versions of the strategy respectively. For the original version we see that SCPs receive approximately equal numbers of service requests from the four SSP groups during both normal and overload conditions.

For the enhanced strategy we see the preferred SCPs of the various SSP groups. SSPs 4-6 and SSPs 7-9 send the majority of their requests to SCP 1 and SCP 4 respectively as these are the only SCPs that are three links away. In normal conditions SSPs 1-3 and SSPs 10-12 spread their requests between SCPs 1/2 and SCPs 3/4 respectively; the spread is somewhat uneven since the pheromone selection process arbitrarily selects only one of the two favoured SCPs for each service type at each SSP. In overload conditions we see that SSPs 1-3 and SSPs 10-12 for the most part use SCP 2 and SCP 3 respectively, because SCP 1 and SCP 4 are strongly favoured by SSPs 4-6 and SSPs 7-9. In all conditions we see that SSP groups occasionally use less favourable SCPs. This is an artefact of ant messages generated using the exploration scheme, which may briefly increase the pheromone probability of one of the SCPs that are four links away.

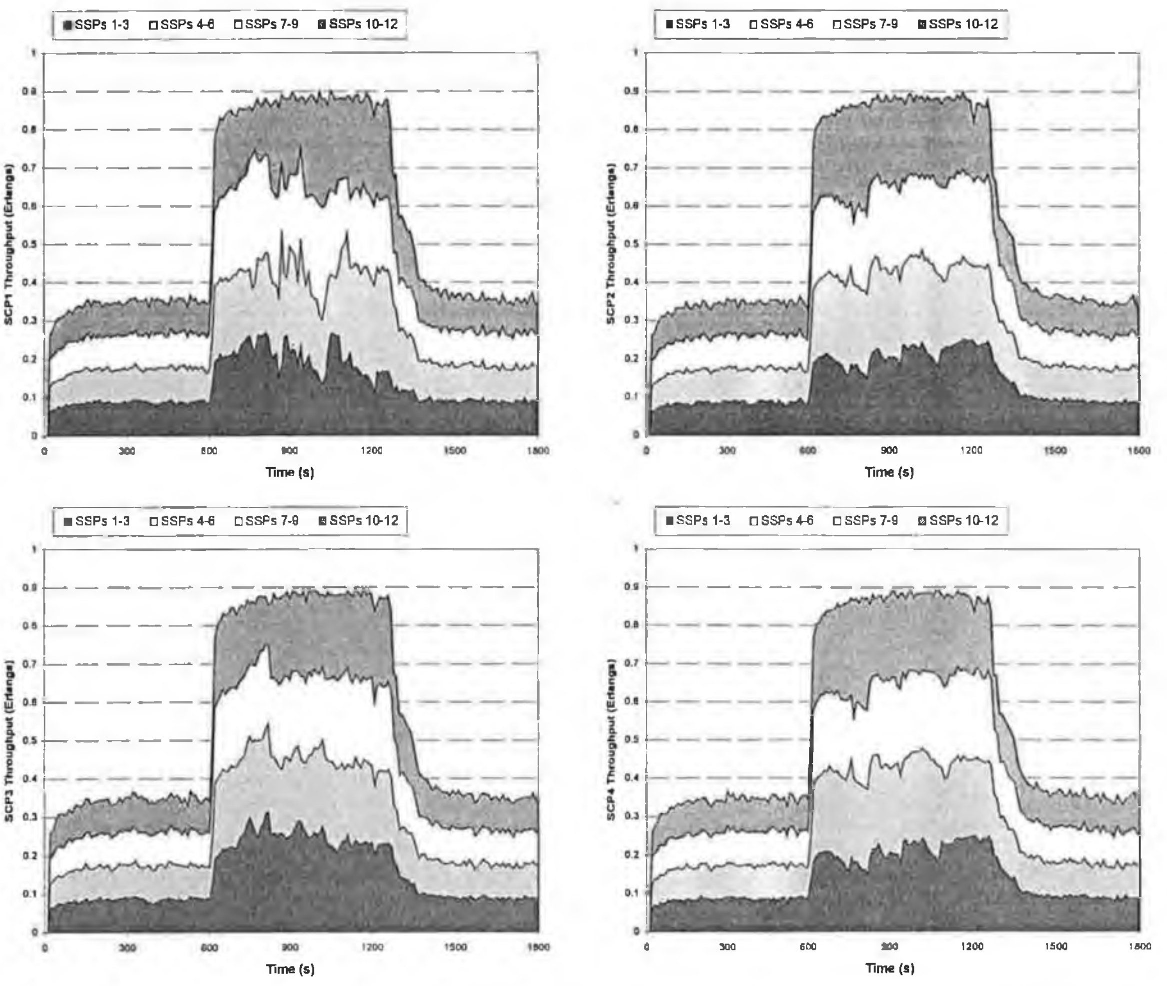


Figure 6.39: Original strategy. Per source SSP group SCP throughputs. Top left: SCP 1. Top right: SCP 2. Bottom left: SCP 3. Bottom left: SCP 4.

From the discussion above we see the major outcome of employing the enhanced strategy is that, all other factors being equal, SSP predominantly use those SCPs in closest proximity. As all SSPs use nearby SCPs, the total traffic carried by the SS.7 network will be minimised (within the constraints set by the SS.7 routing plan). Figure 6.41 quantifies this effect for the current scenario by comparing the total volume of signalling messages in Bytes transmitted by SS.7 links for the original and enhanced strategies.[98] We see that use of the enhanced strategy results in approximately 10% less overall traffic during normal conditions and 15% less during overload, though of course the size of this reduction is highly dependent on factors such as network size, routing plan, service session characteristics and the parameters of the pheromone-based SCP selection process.

[98] For the enhanced strategy both IN and ant messages are included in the total.

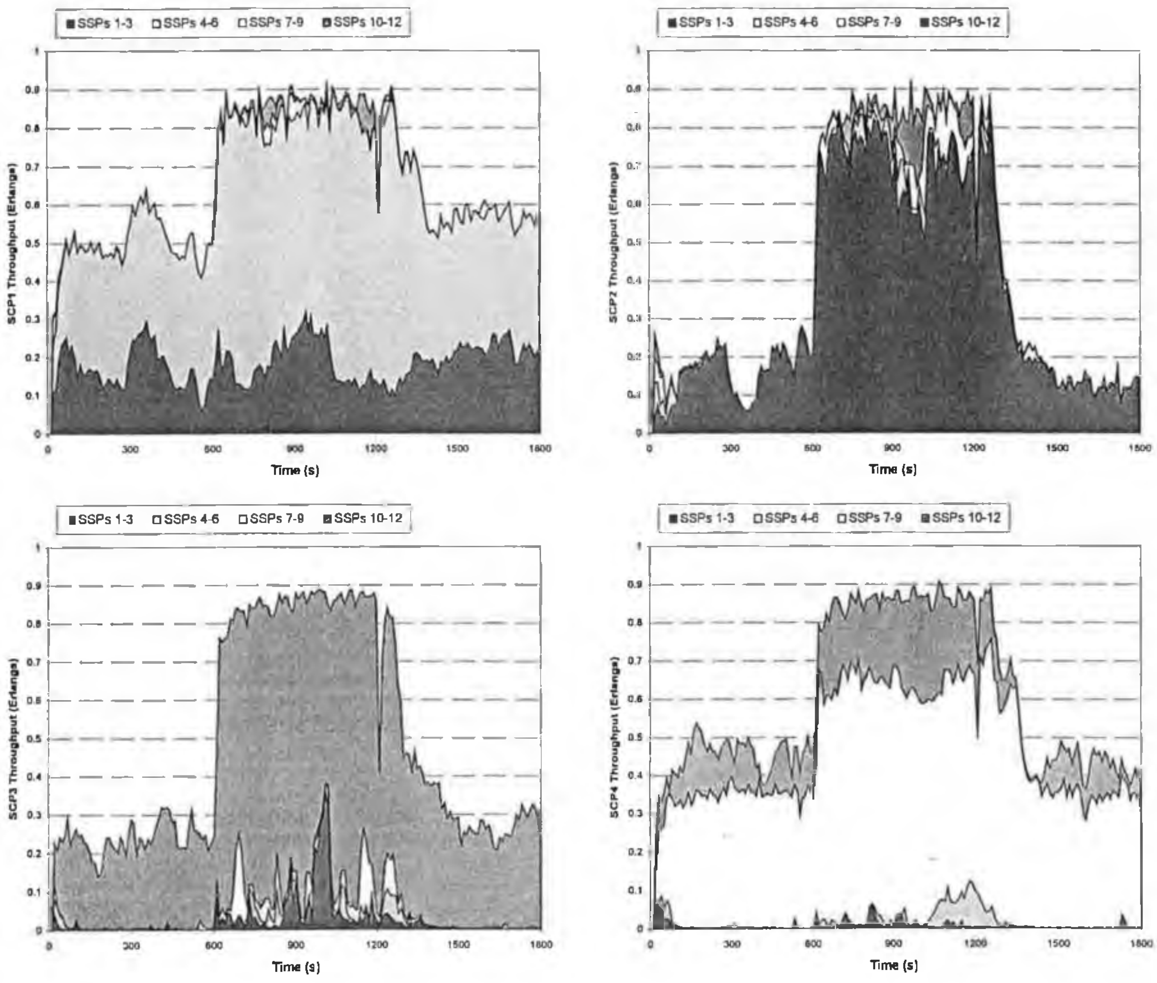


Figure 6.40: Enhanced strategy. Per source SSP group SCP throughputs. Top left: SCP 1. Top right: SCP 2. Bottom left: SCP 3. Bottom left: SCP 4.

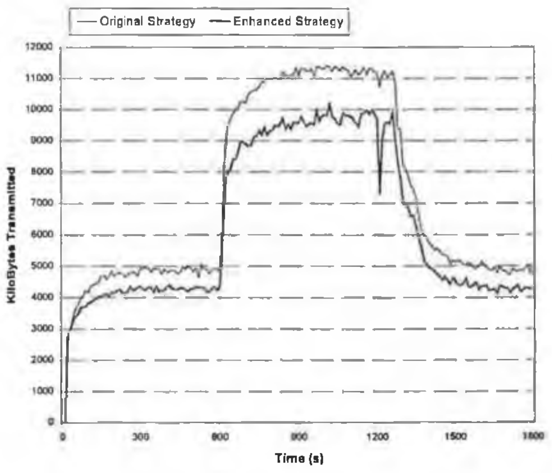


Figure 6.41: Number of SS.7 link packet transmissions for original/enhanced strategies.

6.3.3.3 Operation During Failure Conditions

We now address the operation of the enhanced strategy during two network failure conditions. Firstly we simulate a failure of the signalling link connecting SCP 1 to STP 8, which isolates SCP 1 from all the SSPs in the network. The link in question fails at $t = 600s$ and is brought back into service at $t = 1200s$, with the volume of traffic offered to the network remaining at normal levels throughout. The ideal behaviour in this case is for the SSPs to forward service requests which would previously have been sent to SCP 1 to the other SCPs, in effect their spare capacity is used to compensate for the unavailability of SCP 1.[99]

Figure 6.42 shows the mean and individual throughputs of the four SCPs. Soon after the link failure we see that the mean SCP throughput returns to normal, with all service requests now being routed to the three non-isolated SCPs for the failure duration. The explanation for this behaviour is as follows: in normal circumstances SSPs favouring SCP 1 send it a large number of ant messages and a much smaller number to the other SCPs. Once SCP 1 is made unavailable ant messages do not reach it and hence are not returned to their originating SSP. This is not the case for ant messages sent to other SCPs, so pheromone probabilities are quickly increased for these SCPs and reduced to almost zero for SCP 1. Ant messages sent in the exploration scheme are still sent to SCP 1, so once the link is brought back into service, these messages are returned to the SSPs, where they quickly cause the pheromone probabilities for SCP 1 to return to their previous levels.

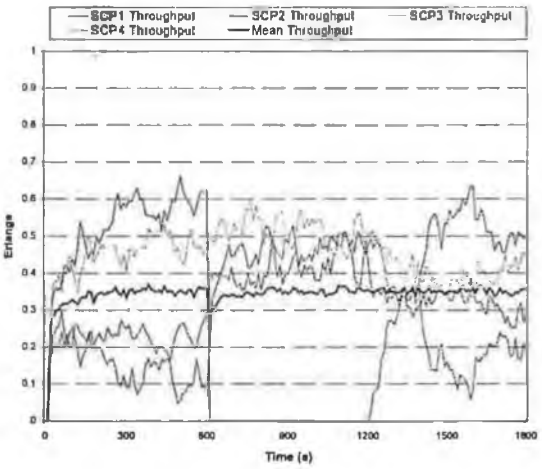


Figure 6.42: SCP 1 failure. SCP mean and individual throughputs.

Figure 6.43 shows the per source SSP group throughputs of the SCPs. We see that SCP 1 normally predominantly services requests originated by SSPs 1-3 and SSPs 4-6. Upon failure onset the requests originating from SSPs 1-3 are transferred to SCP 2 (the only remaining SCP that is three links away), whilst the requests originating from SSPs 4-6 are transferred to the three remaining SCPs (all of which are four links away from SSPs 4-6). We see that the requests are transferred to the other SCPs in an optimal manner, minimising the load carried by the SS.7 network.

[99] We assume here that bids and token allocations are not transferred to SCPs over the SS.7 network, rather over some network management infrastructure. If the former were the case SSPs would not receive token allocations from SCP 1 and would therefore use the other SCPs by default. However, it may take the SSPs as much as one token generation interval to realise that the SCP is unavailable, which will be unacceptable in situations where a relatively long token generation interval is in place.

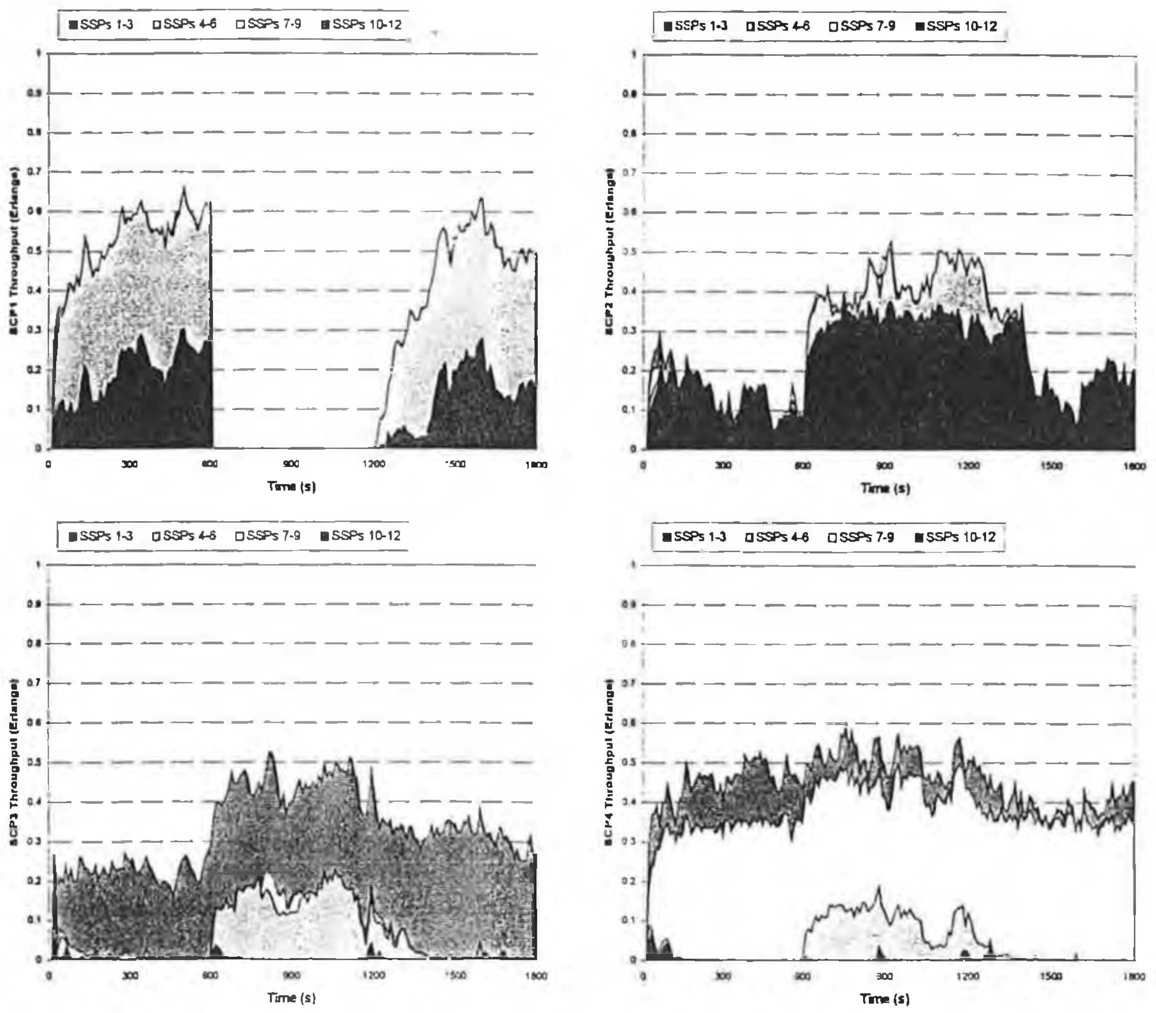


Figure 6.43: SCP 1 failure. Per source SSP group SCP throughputs. Top left: SCP 1. Top right: SCP 2. Bottom left: SCP 3. Bottom left: SCP 4.

We now simulate a failure of STP 6, which results in the failure of all links connecting it to other nodes in the network. The major effect of this is that SSPs 4-6 become isolated from the rest of the network. Additionally SCP 1 becomes unavailable to SSPs 7-12 (STP 6 is on the route from SCP 1 to these SSPs). Since SSPs 4-6 can no longer communicate with any of the SCPs it is clear that they must throttle all arriving IN service requests. As these SSPs cannot receive ant messages the only means they have of detecting that network conditions have changed is through the delay-based load throttle.

Figure 6.44 shows the mean and individual throughputs of the four SCPs. Once STP 6 fails we see that the mean SCP load drops by approximately 25%, reflecting the fact that SSPs 4-6 have been isolated. Figure 6.45 shows the per source SSP group throughputs of the SCPs. After the failure we see a severe drop in the throughput of SCP 1, a large portion of which had previously related to SSPs 4-6, as well as a smaller drop in SCP 4 throughput. When STP 6 is brought back into service ant messages start to reach their destinations and return to SSPs 4-6 within the abatement threshold delay, causing the SSPs to re-commence the acceptance of service requests. A spike in mean SCP load can be observed immediately following failure rectification, this is caused by end-user reattempts for requests previously rejected at SSPs 4-6.

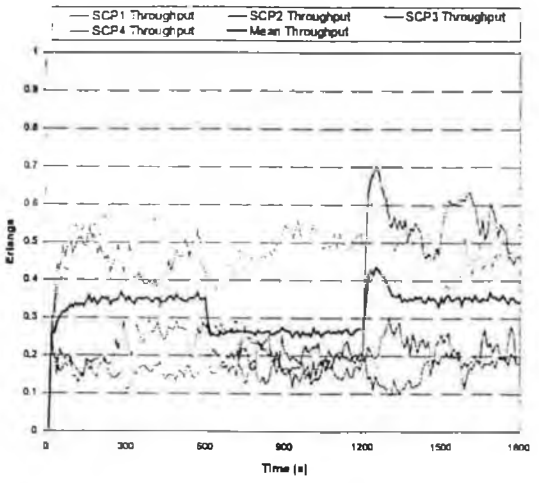


Figure 6.44: STP 1 failure. SCP mean and individual throughputs.

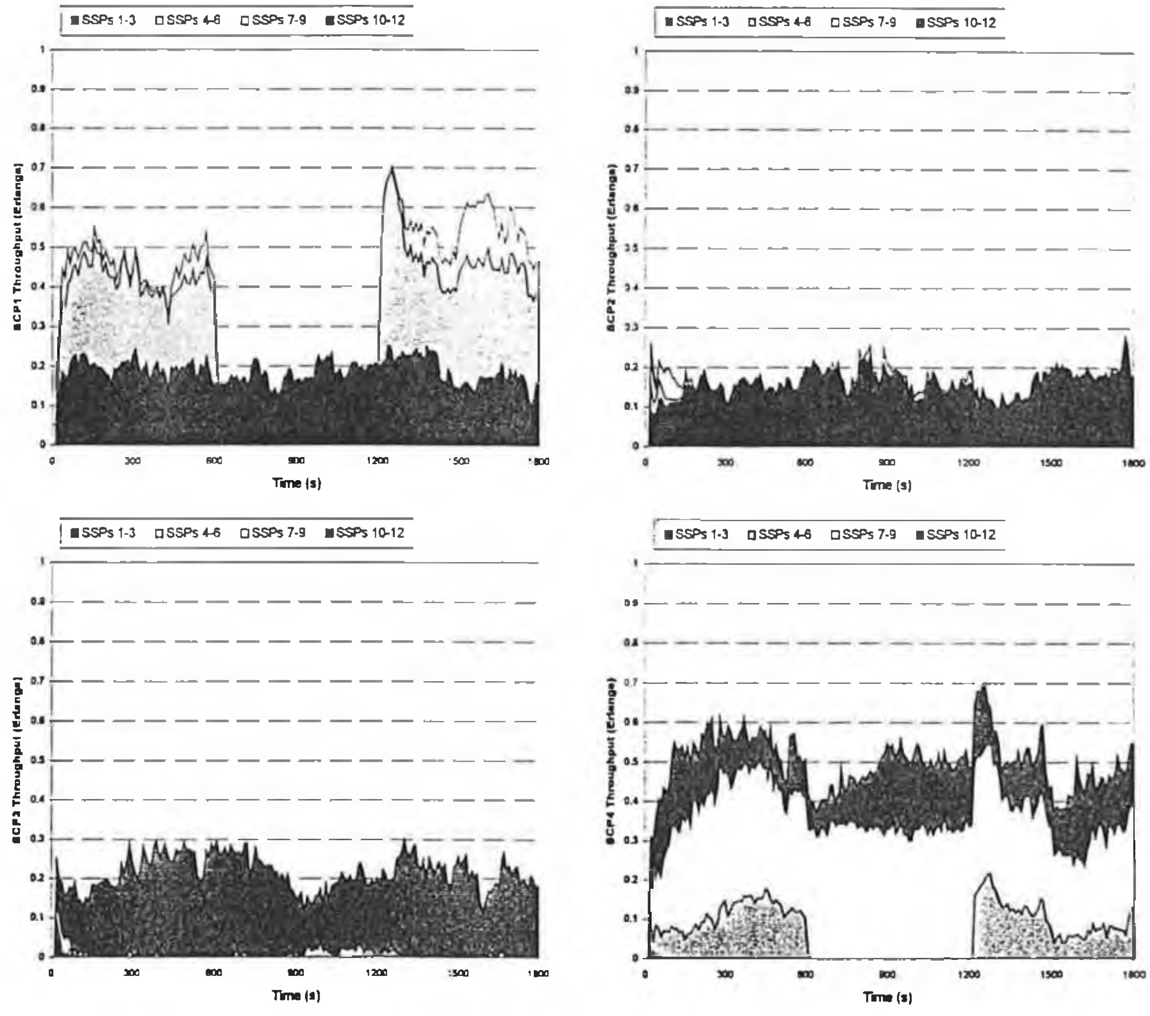


Figure 6.45: STP 1 failure. Per source SSP group SCP throughputs. Top left: SCP 1. Top right: SCP 2. Bottom left: SCP 3. Bottom left: SCP 4.

6.4 SUMMARY AND CONCLUSIONS

Chapter 5 introduced an SCP load control strategy based on profit-optimal resource allocation by means of tokens that are used to grant an IN service session access to the SCP over its full duration. In this chapter we have specified a number of enhancements to the original token-based strategy, which:

- Allocate tokens for the different types of resource required by a session. Depending upon network conditions and offered load, different resource types become susceptible to overload. By generating tokens for the different types of IN resource required by a service session the strategy can protect against overload of multiple resource types;
- Provide co-ordinated allocation of tokens for multiple resource instances to maximise overall network profit. In an IN individual resource instances (in particular SCPs) may provide differing performance for the various supported services. To maximise overall profit the strategy takes into account the service-specific relative performance of the available resources when allocating tokens;
- Introduce dynamic routing of IN service requests. The number of generated tokens is limited to protect against resource overloads, so it is possible to safely employ dynamic routing algorithms that select those resources currently providing the best performance for a particular service type and that adapt to failures and overloads in the SS.7 network.

These enhancements collectively transform the token-based strategy into a fully network-oriented IN load control, where all resources are protected from overload and where the allocation of resources to service requests is done in a co-ordinated manner and with the aim of maximising overall network profit.

The first part of this chapter enhanced the original token-based strategy to allocate the resources of multiple SCPs in a co-ordinated manner. The enhanced strategy incorporates dynamic routing of IN service requests by SSPs to all network SCPs (or a subset thereof). The strategy was shown to maximise the throughputs of all SCPs during overload and maximise profit through selective throttling of service requests. In addition, for networks containing SCPs exhibiting varying performance for different service types, the strategy was shown to match service types with SCPs in a profit maximising manner.

Three forms of the enhanced strategy were specified: a centralised version, in which tokens for all SCPs are generated by a single process; a distributed version, where tokens are separately generated at multiple SCP-specific processes; and a hierarchical distributed version, suitable for application in large-scale INs. The hierarchical distributed version involves designated, principal SSPs bidding on behalf of a group of local SSPs, receiving bundled token allocations and re-distributing these token allocations between themselves and their local SSPs.

We note that it would be straightforward to extend the SSP token re-distribution algorithm to provide protection against SSP overloads. The extended algorithm would take into account the demands placed on the SSP by both IN and non-IN services and their profit ratings. In an overload triggered by non-IN services, SSP tokens allocated to non-IN services would be limited to protect the SSP, whilst higher-profit IN services would remain unaffected. Similarly, in an IN overload, the tokens allocated to IN services would be limited in order to ensure protection of both the SSP and the SCP from overload. If an algorithm of this form were in place at SSPs then the token-based strategy would bear close resemblance to the two-stage optimisation strategy outlined by [Lodge *et al.*, 1999; Lodge, 2000].

The second part of this chapter outlined how the token-based strategy can be enhanced to provide control over multiple types of resource. A strategy enhancement that allocates tokens for both SCPs and SDPs in a co-ordinated manner was specified. SCPs receive bids from SSPs and, for the relevant service types, collate these bids, forward them to SDP token generation process(es), receive SDP token allocations, and use these as inputs into their own token generation process. SCP and SDP tokens are allocated as pairs, thereby ensuring that SSPs do not waste tokens of one type due to the absence of tokens of the other type.

It would be relatively straightforward to enhance the token-strategy to provide control over resource types other than SCP and SDP processing capacity. These resource types could be internal or external to the SCPs/SDPs, for example internal TCAP dialogue handlers, or external IP-based voice circuits used for end-user interaction. As noted by [Kihl and Rumsewicz, 1996] the latter often become the bottleneck in media-stimulated mass call-in focussed overloads, so access to them should be strictly controlled. To incorporate allocation of IP circuits into the token-based strategy we can take a similar approach to that used for SDPs: dedicated token generation processes to which SCPs send bids. One important difference is that the holding times of these circuits is likely to be of the same order as the token generation interval duration, therefore the token generation process must estimate the likelihood of each currently occupied circuit being released during the coming interval before allocating tokens for it.

The final part of the chapter showed how the strategy can be enhanced to provide protection against SS.7 overloads, which can seriously degrade the quality-of-service experienced by IN service end-users. The approach taken is based on the use of periodically-generated ant messages which travel from SSPs to selected SCPs and, upon their return, trigger the updating of a table of pheromone probabilities. Pheromone probabilities reflect the relative performance of SCPs from the point of view of SSPs. SSPs can use them as inputs into their token spending and/or bid splitting processes. Pheromone-based SCP selection is augmented by a delay-based load throttle that uses message round-trip delay measurements to detect SS.7/SCP overloads and explicitly prevents further requests being sent to the SCP in question whilst overload persists.

Analysis of the operation of the token-based strategy enhanced with pheromone-based token spending / bid splitting and delay-based load throttling showed how SSPs can be encouraged to use those SCPs in closest proximity, thereby minimising the load carried by the SS.7 network. In addition the strategy can rapidly respond to failure conditions (and indeed link/STP overloads) in the SS.7 network, by automatically adapting the routing of service requests (if possible) to minimise message loss and unnecessary throttling.

The strategy enhancements outlined in this chapter can, for the most part, be employed together to provide an effective load control system, tailored to a particular IN. For example in a large network a mix of centralised, distributed and distributed hierarchical SCP token generation processes, mirroring network topology, can be present, together with centralised and distributed processes for SDP and IP token generation. SSPs can also employ token re-distribution algorithms to protect themselves against overload, use pheromone-based token spending to identify and use close SCPs and detect SS.7 overloads/failures by means of delay-based load throttling.

One potential area of difficulty relates to the presence of heterogeneous SCPs in a large network: should SSPs use SCPs in close proximity or those giving the best performance for a particular service type? Either price- or pheromone-based bid splitting can be employed depending on the network operator`s answer to this question. We note that one possible solution would be to deploy pheromone-based token-spending with price-based splitting: SSPs would receive tokens primarily from those SCPs giving the best performance, but would prioritise between these on the basis of their proximity.

CHAPTER 7

CLOSING REMARKS AND TOPICS FOR FURTHER RESEARCH

Rapid advances in technology, together with market deregulation, is fuelling demand for communications, entertainment and financial services delivered over the telecommunications infrastructure. The consequent trend towards increased network complexity suggests that approaches to performance management of systems, services and networks will soon need to be more responsive, adaptive, proactive, and less centralised than those in use today. This is especially true in the context of INs, which will continue to act as the primary telecommunications service delivery platforms for years to come, but in which relatively simple, node-oriented load control strategies are currently employed.

In this thesis we have argued that node-oriented load control strategies are incapable of managing SS.7/IN traffic load in a manner that provides optimal usage of network resources. This viewpoint is supported by chapter 2, where the results of a detailed analysis of standard node-oriented strategies for SS.7 and IN were presented and discussed. These results demonstrate that, as well as being inherently flawed, these controls can interact poorly in the presence of simultaneous SS.7 and IN resource overloads.

As an alternative to node-oriented strategies we advocate the adoption of network-oriented approaches. An example network-oriented strategy, employing a token-based paradigm, was proposed and studied. Chapter 5 showed that, in the context of controlling the load of a single SCP, the token-based strategy compares favourably with two SCP control strategies of contrasting sophistication. More significantly, chapter 6 showed how the strategy can be extended to provide a fully network-oriented IN load control solution. The enhanced strategy was seen to have numerous advantages over traditional approaches; including:

- *Dynamic control:* the strategy dynamically calculates control parameters, hence it performs well in the face of variations in the volume of offered load, the service mix and resource availability levels. Its dynamic nature also means it is inherently scalable to networks of arbitrary size;

- *Preventative control:* the strategy is predicated on explicit allocation of critical network resources, hence it prevents overloads rather then simply reacting to them;
- *Profit-optimal control:* the strategy maximises generated profit by allocating resources to more valuable services during overload;
- *Control of multiple heterogeneous resources:* the strategy provides control over multiple instances of heterogeneous resource types. During an overload it protects those resources that are currently the system bottleneck from overload;
- *Dynamic routing of service requests:* the strategy provides a means of introducing co-ordinated dynamic routing of service requests by SSPs to multiple SCPs;
- *Minimisation of SS.7 network load:* the strategy can be enhanced to provide automatic selection of those IN resources in close proximity to an SSP, thus minimising the load carried by the SS.7 network;
- *Integration with legacy systems:* token allocations can be straightforwardly converted into control parameters such as gap intervals or PT coefficients, so legacy SSPs (employing traditional load throttles) can be integrated into the overall network-oriented control framework;
- *Understandable paradigm:* allocation of tokens is an easily understood concept, hence the strategy provides a useful paradigm for explanation of the operation of the IN performance management system to a non-technical audience.

In light of these advantages we believe that there would be significant benefits for the IN network operator in implementing the token-based load control strategy. These benefits arise from the strategy's direct focus on the load control needs of the operators in the context of a more open, dynamic IN environment. This focus contrasts sharply with that of traditional node-oriented approaches, where equipment vendors design controls to maximise throughput of individual switches without regard to overall network performance.

In the remainder of this chapter we outline and discuss ways in which the token-based strategy could be further enhanced. §7.1 discusses potential enhancements that could provide improved load control performance and facilitate increased implementation flexibility. §7.2 outlines two alternative application areas in which the token-based strategy could provide a solution to performance management problems. Finally, §7.3 discusses some broader research topics relating to the full integration of network-oriented load control strategies into a fuller network management solution incorporating long-term planning of network resource deployment.

7.1 FURTHER ENHANCEMENT OF THE TOKEN-BASED STRATEGY

The token-based strategy specified in chapters 5 and 6 was shown to provide near-optimal load control performance in a wide range of operating conditions. Nevertheless, we envisage a number of further enhancements to the strategy that may provide more optimal control and greater implementation flexibility in the context of a real IN environment.

Alternative Token Spending Algorithms

Simulation results for the token-based strategy have shown that the use of percentage thinning in the token spending process leads to wastage of tokens during high load, resulting in somewhat sub-optimal resource usage. Other means of controlling the rate at which tokens are spent can be employed; for example, chapter 5 shows that token allocations can be converted into CG gap intervals. It should be possible to convert token allocations into leaky bucket or Window parameters. The relative merits of these alternative approaches on the degree of token wastage should be explored.

The other component of the token spending process is the algorithm used to select between one of a number of resources for which tokens remain. The approach taken in chapter 6 was simply to select that resource for which the largest number currently remain. However, this was seen to lead to unbalanced loads when the network contains heterogeneous resources. Approaches such as selecting resources proportionally to the number of tokens remaining for them, should be investigated.

Control of SSP, IP and Other Resources Types

As outlined in chapter 6, it appears relatively straightforward to enhance the token-based strategy to provide explicit allocation of capacity of resource types other then SCPs and SDPs. For load control of SSPs we envisage enhancement of the SSP token redistribution algorithm to take into account the local processing demands of both IN and non-IN services. For control of IPs the approach would be similar to that used for SDPs, but would take into account that the critical resource is likely to be voice circuits, not processing capacity.

Integrated control of multiple IN resource types appears very attractive, however it should be noted that in many overload cases the critical resource is actually the limited pool of human agents available to answer end-user calls. Therefore, it may be very beneficial to allow manual configuration of the token generation processes with the number of operators available to answer calls to a specific number (or set of numbers), so that tokens are allocated accordingly. This would minimise the number of end-users receiving engaged tones and free up network resources for other, profitable sessions.

Co-ordinated allocation of tokens for more then a small number of different resource types is likely to result in very complex token generation processes. It is an open issue as to whether this complexity is justifiable, or whether alternative, simpler solutions would suffice. One such solution would be to run completely independent token generation processes for the various types of resource required to process service sessions, although this runs the risk of more significant levels of token wastage.

Enhanced Resource Capacity Estimation

The token generation algorithms make two significant simplifying approximations regarding the characteristics of resources for which they generate tokens. In real systems these approximations may be unacceptable and lead to sub-optimal control. The first approximation is that the processing capacity required at each resource for a service session is constant for all sessions of the same service. In reality the amount of capacity required to process individual messages may increase the higher the resource's current level of utilisation. Therefore, an appropriate non-linear relationship between processing capacity requirements and the amount of capacity already allocated should be used during token generation.

The second simplifying assumption is that once a service request is accepted all processing capacity required by that session is consumed within the current control interval. Clearly service sessions may involve lengthy delays relating to end-user interactions. Hence, at the time of a given token generation process invocation, resources will have associated 'residual' demands that may be 'exercised' during the coming control interval. The presence of these demands effectively reduces available capacity and should be taken into account if the target utilisation is not to be exceeded.[100] To do so, it would be necessary to maintain approximate estimates of the mean delays for the various end-user interaction phases of a session. These values could be used to calculate probabilities that sessions currently in a interaction phase will require processing capacity in the coming control interval; these probabilities could then be used to calculate the actual amount of capacity that can be safely allocated.

Other factors relating to resource characteristics may also be of relevance to the accuracy of the token generation process. For example, many SCPs employ multi-processor hardware architectures, so processing overheads relating to inter-processor communication and management tasks can increase significantly during periods of high utilisation. All such factors would need to be taken into account during the design of an optimal token generation algorithm. Clearly it would be in the interests of equipment vendor to design customised token generation

[100] In our studies the impact of residual demands was minimal, due both to the wastage of a number of tokens during overload and the fact that, for the services modelled, processing requirements were approximately balanced over the session duration. The latter means that the impact of sessions claiming residual demands is in effect offset by new sessions that do not use all the allocated capacity immediately.

algorithms that ensure that the resource in question meets performance targets agreed with the network operator.

Enhanced SSP Bidding

The bidding mechanism used by SSPs in the versions of token-based strategy specified in this thesis is very simple: it doesn't take into account long term trends in offered traffic. We can envisage a more advanced approach, in which SSPs would have access to a database containing records relating to previous demands for services and the results of previous bids (token prices, percentage of received tokens actually used *etc.*). This database could facilitate SSPs automatically detecting daily traffic patterns and changing bids accordingly. Also, by comparing current arrival rates with normal patterns, it should be possible to detect higher-than-average loads indicating onset of overload conditions.

Adaptive Invocation of Token Generation Process

One of the fundamental limitations of the token-based strategy is that token generation takes place at set intervals. Tokens are only valid for the duration of the control interval for which they are generated, therefore the strategy is vulnerable to delays and errors in the transmission of SSP bids and token allocations. In addition, frequent invocation of the token generation process may represent a significant drain on SCP resources and add significantly to the load carried by the SS.7 network.[101]

Problems associated with limited token lifetime can be solved by translating token allocations into parameters of load throttles such as Call Gapping or Window, as alluded to previously. Once put in place the throttles can use these parameters until the arrival, at an arbitrary time in the future, of the next token allocation. This also suggests an alternative to periodic invocation of the token generation process(es): SSPs could send updated bids at arbitrary times, for example when they detect abnormal increases in the level of demand for one or more services.[102] The receiver of updated bids could then invoke the token generation process, using both the new bids and old bids (the last submitted bids for those SSPs that have not submitted updated bids).

To further extend the alternative approach the resources themselves can be allowed to invoke the token generation process when they detect a change in available capacity. With this enhancement the token-based strategy could be used as an tool for automatically re-configuring the network when new resources are added, resource capacity changes, or the services supported by the resources change.

[101] The latter is relevant for the scenario where bids and token allocations are transferred over the SS.7 network.

[102] This approach could work particularly well if, as described above, SSPs maintain databases that can be used to detect longer-term traffic patterns. Then SSPs could, for example, submit updated bids in response to expected hourly and daily changes in arrival rates.

Clearly the design of algorithms to achieve the 'intelligent' bidding behaviour described above is a non-trivial task and it is difficult to quantity the resultant benefits. However, this represents an interesting topic for further research, since it deals with the application of longer-term measurements in load control. If a success such an approach would be a significant advancement over current approaches, which invariably use only short term measurements to enable reaction to overload onset.

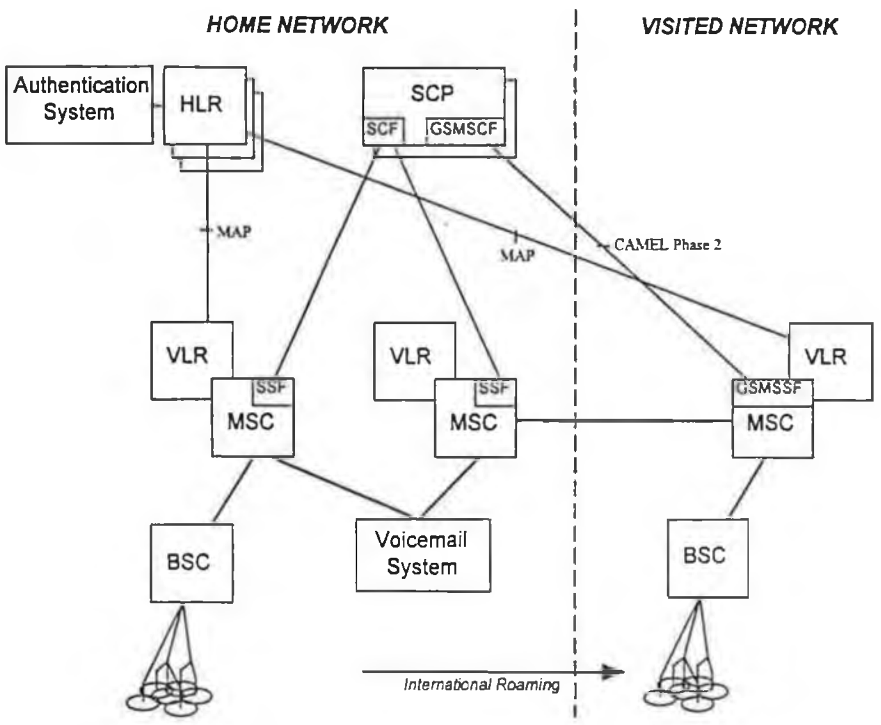
7.2 APPLICATION OF THE TOKEN-BASED STRATEGY TO OTHER CONTROL PROBLEMS

The token-based load control strategy is generic in nature: it does not depend on particular characteristics of IN resource or services in order to provide load control. There is no reason why the strategy can not be applied to control problems other then the protection of IN resources. In this section we outline how the strategy could be applied in two alternative application areas where load control and performance management is becoming a major issue. The first area is *Global System for Mobile communication (GSM)* networks enabled with *Customised Access for Mobile network Enhanced Logic (CAMEL)* capabilities, which bear close resemblance to the IN architecture. The second is large-scale *Voice-over-IP (VoIP)* networks, which also adopt distributed architectures for the delivery of telecommunications services.

7.2.1 LOAD CONTROL FOR GSM CAMEL

International roaming in GSM networks has been a reality for a number of years, however in general the roaming end-user does not have the ability to seamlessly access the services available in his/her home network. For example, when accessing voicemail the roaming end-user has to prefix his/her abbreviated dialling number with a full international code. The CAMEL initiative was set up in order to develop standards that will facilitate network operators offering end-users all the services to which they subscribe in a manner familiar to them. Existing CAMEL standards also allow for roaming of Prepaid customers between networks, a factor that is expected to greatly accelerate introduction of CAMEL capabilities between network operators. CAMEL standards specify the interface between the *GSM Service Control Function (GSMSCF)* and the *Mobile Switching Centre (MSC)*, thereby allowing the MSC remotely invoke service sessions when requested. An example scenario is illustrated in Figure 7.1 below.

When an end-user roams to a new network the local MSC/VLR detect that they do not have a profile for that user and contact the user's HLR, which forwards the required data, updates its own records and removes the users profile from the VLR with which the end-user was last associated. Based on the data received from the HLR the visited VLR will update the trigger table in the visited MSC so that when the end-user requests a service (for example by entering the abbreviated code for voicemail access) the MSC can detect the request and forward it, over the CAMEL interface, to the home SCP. For the duration of the service session the home SCP directs the GSMSSF in the visited MSC and, upon session completion, updates billing records as appropriate. For roaming Prepaid users there will be frequent communication between the visited MSC and the home SCP in order to track the customer's remaining credit. This will lead to a significant amount of signalling traffic crossing the network boundary.



HLR:	Home Location Register	**MSC:**	Mobile Switching Centre
VLR:	Visited Location Register	**BSC:**	Base Station Controller
GSMSCF:	GSM Service Control Function	**MAP:**	Mobile Application Part
GSMSSF:	GSM Service Switching Function		

Figure 7.1: International Roaming with CAMEL.

Signalling traffic crossing the network boundary represents a potential source of overload of the SCP(s) in the home network. Of course the situation is further complicated by the sources of the overload traffic being outside the control of the home network operator. Clearly it is necessary to implement some form of load control at the network boundary. The ability of the token-based strategy to protect high-revenue services whilst enforcing whatever SLAs may be in place between the two operators make it an attractive solution to this problem.

7.2.2 Load Control for Large-Scale Voice-over-IP Networks

A technology currently growing in importance is *Internet Telephony*, which involves the delivery of a full range of telephony services using the *Internet Protocol (IP)* and is therefore often referred to as Voice-over-IP. Currently there is explosive growth in the number of standards related to IP-telephony convergence. Here we briefly discuss two of the main approaches. The first, and most mature, is based on the H.323 series of standards from the ITU [ITU, 1996] and complimentary work from the ETSI TIPHON project [ETSI, 1999], while the second is being developed by the IETF and is based on their *Session Initiation Protocol (SIP)* and *Stream Control Transmission Protocol (SCTP)* specifications.

7.2.2.1 TIPHON H.323-based Voice-over-IP

The *Telecommunications and Internet Protocol Harmonisations Over Networks (TIPHON)* specifications are aimed at facilitating a scaleable, carrier-class H.323-based architecture. Central to the H.323 standard is the *Gatekeeper* functional entity, which is broadly analogous to a switch in a traditional telephony network. The H.323 Gatekeeper has been generalised into a 'cloud' of Gatekeepers by ETSI TIPHON and is likely to be further decomposed into smaller functional blocks such as *Name Servers* (performing address mapping) and *Authentication Servers* (maintaining authentication and authorisation information) by equipment vendors. These decomposed functional blocks will no longer be replicated in every Gatekeeper node, instead Gatekeepers will access them through an INAP-like protocol.

Large-scale H.323 implementations will contain multiple instances of the various resource types accessed by multiple Gatekeepers to provide end-user services. An example network scenario is depicted in Figure 7.2. It shows multiple Gatekeepers, which compete not only for Name Server and Authentication Server resources but also for:

- *Gateways:* required by Gatekeepers for calls that are routed to the destination network(s) accessible through a particular gateway;
- *Multi-point Control Units:* required by Gatekeepers to support centralised multi-party calls;
- *Service Control Points*: required by Gatekeepers to support advanced IN-based services.

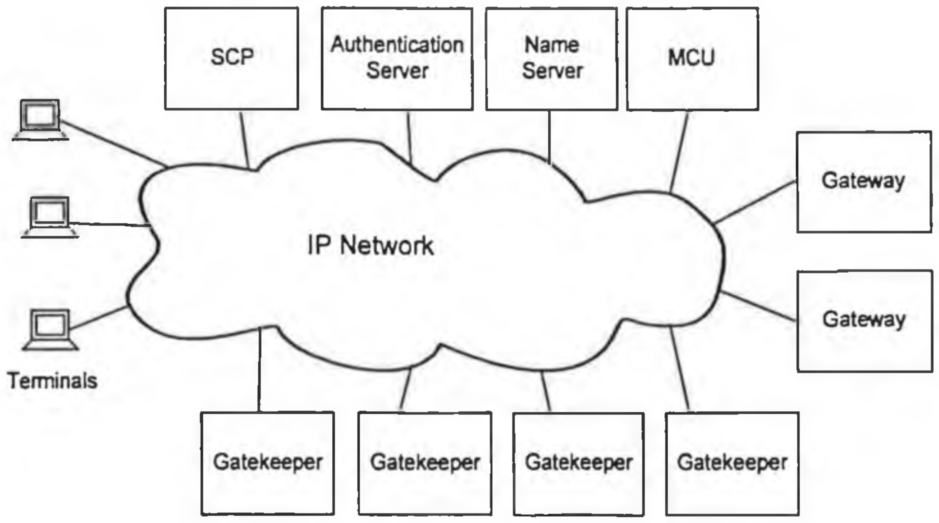


Figure 7.2: TIPHON Network Scenario.

The token-based load control strategy could easily be applied in the scenario described above, with tokens being allocated to the Gatekeepers for the various resource types. As previously the strategy would facilitate effective load control, dynamic routing and profit-optimal selective throttling of service requests during overload. There may also be potential to enhance the strategy to provide control over bandwidth allocated in the IP network. We note that such an approach is likely to require integration with traffic control strategies (such as *Multi-Protocol Label Switching (MPLS)* or *Diffserv*) operating at the IP layer.

7.2.2.2 SIP/SCTP-based Voice-over-IP

An alternative framework for the development of Voice-over-IP networks is being developed within the IETF and, is to a large degree, centred around the SIP and SCTP specifications. SIP is a generic protocol enabling the set-up of communication sessions between end systems and proxy servers. It incorporates mechanisms for terminal capability negotiation, caller and callee authentication, personal mobility, and simple telephony-related features like call forwarding and call transfer. SCTP enables the reliable exchange of (SS.7) signalling messages over and IP-based network infrastructure, thus providing a means of transparently supporting existing telephony-related services in an IP-based environment.

The use of SCTP means that it will be possible to introduce network elements like SCPs with SCCP entities that communicate directly with other IP-based entities through use of the SCTP message transfer service; thus SCTP replaces the MTP in this context. SCTP contains window-based flow control procedures, however research work on the analyses of their performance has commenced only recently. For example, it is not yet clear how these controls would operate in the type of overload scenario addressed in this thesis and what impact they may have on the operation of application-level controls such as ACG or the token-based strategy. Clearly issues such as these will need to be carefully considered before large-scale SIP/SCTP-based systems are deployed, thus analysis of these systems from the load control viewpoint is likely to become an important research topic.

7.3 BROADER RESEARCH TOPICS RELATING TO NETWORK-ORIENTED LOAD CONTROL

To conclude we will briefly discuss two longer-term directions for research relating to network-oriented load control. Throughout the thesis we have considered that the main goal of a network-oriented load control is to ensure that network resources are used in a globally optimal fashion in the face of relatively short-term variations in end-user demand for services. Furthermore, the major means of achieving this goal during high load conditions is selective throttling of service requests. It has been implicitly assumed that the load control strategy can directly influence neither the behaviour of end-users, nor the 'structure' of the network (in terms of which resources support which services, where resources are located in the network *etc.*). However, this need not be the case if load control is developed into a longer-term activity that is integrated with both network design/provisioning (thereby influencing network structure) and pricing policies (thereby exerting some influence over end-user behaviour).

Load control strategies attempt to achieve optimal resource utilisation within the constraints set by the physical and logical structure of the network. Networks are initially designed to optimise resource usage for an expected traffic mix, however as new services are introduced and end-user demands change the network invariably becomes more susceptible to overload. Data gathered by a network-oriented load control could be used to automatically detect long term traffic trends and form recommendations regarding redistribution of resources in the network. For example, with the token-based strategy increases in the mean closing price for tokens relating to a specific service could indicate that the service should be supported by more SCPs. If successful the approach could be taken a step further in future object-based systems, by having the load control strategy itself initiate the automatic redistribution of resources in response to changing traffic conditions.

In future networks supporting multimedia-based services end-users are likely to be charged, at least in part, on the basis of bandwidth allocated to, or used during, a service session. Moreover, it is likely that prices per unit bandwidth can be varied dynamically by the service provider, depending on prevailing conditions in the network. Network-oriented load control strategies may provide a valuable input into the bandwidth price-setting process. For example, the strategy could detect changes in end-user demands and recommend that bandwidth prices are increased so as to discourage end-users and thereby the onset of overload conditions.

PUBLICATIONS AND REPORTS RELATED TO THIS THESIS

This thesis contains material from a number of publications and reports. These are listed in chronological order as follows:

1. Jennings B., F. Lodge and T. Curran, An IN Load Control Strategy Incorporating Measures for Handling of SS7 Network Overloads, in *Proceedings of the 15th United Kingdom Teletraffic Symposium (UKTS-15)*, session 7, paper no. 1, Durham, England, March 1998.
2. Jennings B., F. Lodge and T. Curran, A Strategy for the Resolution of Intelligent Network (IN) and Signalling System No. 7 (SS7) Congestion Control Conflicts, in *Proceedings of the 1998 IEEE International Conference on Communications (ICC'98)*, session 44B, paper no. 4, Atlanta, USA, June 1998.
3. Arvidsson Å., B. Jennings, L. Angelin and M. Svensson, On the use of Agent Technology for IN Load Control, in *Proceedings of the 16th International Teletraffic Congress (ITC-16)*, P. Key and G. Smith (eds.), pub. Elsevier Science B. V., vol. 3b, pp. 1093-1105, Edinburgh, Scotland, June 1999.
4. Jennings B. and Å. Arvidsson, Co-operating Market/Ant based Multi-Agent Systems for Intelligent Network Load Control, in *Proceedings of the 1999 Workshop on Intelligent Agents in Telecommunications Applications (IATA'99)*, S. Albayrak (ed.), pp. 66-79, Stockholm, Sweden, August 1999.
5. Jennings B., R. Brennan, R. Gustavsson, R. Feldt, J. V. Pitt, K. Prouskas and J. Quantz, FIPA-compliant agents for real-time control of Intelligent Network traffic, *Computer Networks (The International Journal of Computer and Telecommunications Networking)*, vol. 31., no. 19, pp. 2017-2036, August 1999.
6. Wathan N., B. Jennings and T. Curran, Application of MARINER agents to Load Control in IP Telephony Networks, in *Proceedings of the 1st CAMELEON ACTS Workshop*, S. Bretzke (ed.), Singapore, September 1999.
7. Jennings B., Agent Technology for Load Control in Intelligent Networks, in *Agents Technology in Europe*, pub. ACTS-InfoWin project, pp. 65-74, 1999.
8. Jennings B., Å. Arvidsson and T. Curran, Network-oriented Load Control for Intelligent Networks, in *Proceedings of the 2000 IEEE Intelligent Network Workshop (IN'2000)*, F. J. Scholtz (ed.), Cape Town, South Africa, May 2000.
9. Jennings B., N. Wathan, R. Brennan and C. Smith, *Network-oriented Load Control for IN: Benefits and Requirements*, Submission to ETSI SPAN on behalf of the ACTS-MARINER project, May 2000, available (31/05/2001): http://www.teltec.dcu.ie/mariner.
10. Jennings B. (ed.), *Final Report of the ACTS-MARINER project*, June 2000, available (31/05/2001): http://www.teltec.dcu.ie/mariner.

11. Brennan R., B. Jennings, C. McArdle and T. Curran, Evolutionary Trends in Intelligent Networks, *IEEE Communications Magazine*, vol. 38, no. 6, pp. 86-93, June 2000.

At the time of writing the following paper has been accepted for publication:

12. Jennings B., Å. Arvidsson and T. Curran, A Token-based Strategy for Co-ordinated, Profit-optimal Control of Multiple IN Resources, to appear in the *Proceedings of the 17th International Teletraffic Congress (ITC-17)*, September 2001.

REFERENCES

[ACTS-MARINER, 2001] ACTS Project MARINER (AC333): Multi-Agent Architecture for Distributed IN Load Control and Overload Protection, available (31/05/2001): http://www.teltec.dcu.ie/mariner.

[Akinpelu and Skoog, 1985] Akinpelu J. and R. A. Skoog, Controlling transients and overloads in Common Channel Signalling networks, in *Proceedings of the 11th International Teletraffic Congress*, M. Akiyama (ed.), pub. Elsevier Science B. V., Kyoto, Japan, September 1985.

[Angelin, 1996] Angelin L., On the properties of a congestion control mechanism for signaling networks based on a state machine, *University of Karlskrona/Ronneby Technical Report*, ISRN HKR-RES-96/10-SE, Karlskrona, Sweden, 1996, available (31/05/2001): http://www.hk-r.se/fou.

[ANSI, 1999] American National Standards Institute (ANSI), *Intelligent Networks*, ANS T1.667-1999, April 1999.

[Arvidsson *et al.*, 1997] Arvidsson Å., S. Pettersson and L. Angelin, Profit Optimal Congestion Control in Intelligent Networks, in *Proceedings of the 15th International Teletraffic Congress*, V. Ramaswami and P. E. Wirth (eds.), pp. 911-920, pub. Elsevier Science B. V., Washington D. C., USA, 1997.

[Atai and Northcote, 1996] Atai A. H. and B. S. Northcote, AIN Focused Overloads – A review of USA CCS network failures and lessons learned, in *Proceedings of the ITC Mini-Seminar on Engineering and Congestion Control in Intelligent Networks*, Melbourne, Australia, April 1996.

[Bafutto *et al.*, 1994] Bafutto M., P. J. Kühn and G. Willmann, Capacity and Performance Analysis of Signaling Networks in Multivendor Environments, *IEEE Journal on Selected Areas in Communications*, vol. 12, no. 3, pp. 490-500, April 1994.

[Banks, 1998] Banks J. (ed.), *Handbook of Simulation: Principles, Methodology, Advances, Applications and Practice*, pub. John Wiley & Sons Inc., 1998.

[Bedoy *et al.*, 1998] Bedoy J., L. Orozco Barbosa and E. Quiroz, Study of Load Control Mechanisms for the Effective Delivery of IN Services, in *Proceedings of the 1998 IEEE International Conference on Communications (ICC '98)*, pp. 1738-1742, Atlanta, USA, June 1998.

[Bellcore, 1992a] Bellcore, *Advanced Intelligent Network (AIN) 0.1 switching system generic requirements*, Bellcore Technical Report TR-NWT-001284, iss. 1, August 1992.

[Bellcore, 1992b] Bellcore, *Signalling Transfer Point (STP) Generic Requirements*, TR-NWT-000082, iss. 4, December 1992.

[Bellcore, 1993] Bellcore, *AIN SCP generic requirements,* Bellcore Technical Report GR-1280-CORE, iss. 1, August 1993.

[Bellcore, 1994] Bellcore, *Advanced Intelligent Network (AIN) 0.2 switching system generic requirements*, Bellcore Technical Report GR-1298-CORE, iss. 2, December 1994.

[Bellcore, 1995] Bellcore, *Advanced Intelligent Network (AIN) 0.1 switching system issues list report*, Bellcore Technical Report GR-1298-ILR, iss. 2A, April 1995.

[Berger, 1991a] Berger A., Comparison of Call Gapping and Percent Blocking for Overload Control in Distributed Switching Systems and Telecommunications Networks, *IEEE Transactions on Communications*, vol. 39, no. 4, pp. 574- 580, April 1991.

[Berger, 1991b] Berger A., Determination of Load-Service Curves for Distributed Switching Systems: Probabilistic Analysis of Overload-Control Schemes, *in Proceedings of the 13th International Teletraffic Congress*, A. Jensen and V. B. Iversen (eds.), pp. 435-440, pub. Elsevier Science B. V., Copenhagen, Denmark, June 1991.

[Berkelaar and Dirks, 1995] Berkelaar M. and Dirks J, *The Linear Programming Toolkit LP_SOLVE version 2.0*, 1995, available (31/05/2001): ftp://ftp.es.ele.tue.nl/pub/lp_solve.

[Bolotin, 1994] Bolotin V. A., Modeling call holding time distribution for CCS network design and performance, *IEEE Journal on Selected Areas in Communications*, vol. 12, no. 3, pp. 433-438, April 1994.

[Brennan *et al.*, 2000] Brennan R., B. Jennings, C. McArdle and T. Curran, Evolutionary Trends in Intelligent Networks, *IEEE Communications Magazine*, vol. 38, no. 6, pp. 86-93, June 2000.

[Conway, 1990] Conway A. E., Queuing Network Modeling of Signaling System No.7, in *Proceedings of the 1990 IEEE Global Telecommunications Conference (Globecom'90)*, vol. 1, pp. 552-558, San Diego, USA, December 1990.

[Davidsson *et al.*, 2000] Davidsson P., B. Carlsson, S. Johansson and M. Ohlin, Using Mobile Agents for IN Load Control, in *Proceedings of the 2000 IEEE Intelligent Network Workshop (IN'2000)*, F. J. Scholtz (ed.), Cape Town, South Africa, May 2000.

[Di Caro and Dorigo, 1998] Di Caro G. and M. Dorigo, Mobile Agents for Adaptive Routing, in *Proceedings of the 31st Hawaii International Conference on Systems*, Hawaii, January 1998.

[Dorigo and Gambardella, 1997] Dorigo M. and L. M. Gambardella, Ant colony system: A cooperative learning approach to the traveling salesman problem, *IEEE Transactions on Evolutionary Computation*, vol. 1., no. 1, pp. 53-66, 1997.

[ETSI, 1994] European Telecommunications Standards Institute (ETSI), *Intelligent Network (CS1) Core INAP: Protocol Specification*, ETS 300 374-1, ed. 1, September 1994.

[ETSI, 1999] European Telecommunications Standards Institute (ETSI), *TIPHON, Network Architecture and Reference Configurations, Phase 2: Scenario 1 + Scenario 2*, ETSI TIPHON Doc. TS101313, February 1999.

[Farel and Gawande, 1991] Farel R. A. and Gawande M., Design and Analysis of Overload Control Strategies for Transaction Network Databases, in *Proceedings of the 13th International Teletraffic Congress*, A. Jensen and V. B. Iversen (eds.), pp. 115-120, pub. Elsevier Science B. V., Copenhagen, Denmark, June 1991.

[Hac and Gao, 1998] Hac A. and L. Gao, Analysis of Congestion Control Mechanisms in an Intelligent Network, *International Journal of Network Management*, vol. 8, pp. 18-41, 1998.

[Harrison and Patel, 1993] Harrison P. G. and N. M. Patel, *Performance Modelling of Communication Networks and Computer Architectures*, International Computer Science Series, pub. Addison-Wesley Publishing Company, 1993.

[Houck *et al.*, 1994] Houck D. J., K. S. Meier-Hellstern, F. Saheban and R. A. Skoog, Failure and Congestion Propagation Through Signaling Controls, in *Proceedings of the 14th International Teletraffic Congress*, J. Labetoulle and J. W. Roberts (eds.), pp. 367-376, pub. Elsevier Science B.V, Antibes Juan-les-pins, France, 1994.

[IETF, 2001] Internet Engineering Taskforce (IETF), *Service in the PSTN/IN requesting Internet Service (SPIRITS) charter*, available (31/05/2001): http://www.ietf.org/html.charters/spirits-charter.html.

[ITU, 1989] International Telecommunications Union (ITU), *Data Communication Networks: Open Systems Interconnection (OSI)*, Recommendations X.200-290, Blue Book, vol. VIII, Fascicles VIII.4-VIII.5, Geneva, Switzerland, 1989.

[ITU, 1993a] International Telecommunications Union (ITU), *Specifications of Signalling System No. 7*, Recommendations Q.700-795, White Book, vol. VI, Fascicles VI.7-VI.9, Geneva, Switzerland, 1993.

[ITU, 1993b] International Telecommunications Union (ITU), *Intelligent Networks*, Recommendations Q.1200-1290, White Book, Geneva, Switzerland, 1993.

[ITU, 1994a] International Telecommunications Union (ITU), *Information technology - Remote Operations: Concepts, model and notation,* Recommendation X.880, ISO/IEC 13712-1:1995, Geneva, Switzerland, 1994.

[ITU, 1994b] International Telecommunications Union (ITU), *Information technology - Open Systems Interconnection - Abstract Syntax Notation One (ASN.1),* Recommendations X.680-683, ISO/IEC 8824-1/2/3/4:1995, Geneva, Switzerland, 1994.

[ITU, 1996] International Telecommunications Union (ITU), *Visual Telephone Systems and Equipment for Local Area Networks which provide a non-guaranteed Quality of Service,* Recommendation H.323, Geneva, Switzerland, May 1996.

[Kihl and Nyberg, 97] Kihl M. and C. Nyberg, Investigation of Overload Control Algorithms for SCPs in the Intelligent Network, *IEE Proceedings*, vol. 144, no. 6, pp. 419-424, December 1997.

[Kihl and Rumsewicz, 1995] Kihl M. and M. Rumsewicz, Analysis of overload control strategies in combined SSP-SCPs in the Intelligent Network, in *Proceedings of the 15th International Teletraffic Congress*, V. Ramaswami and P. E. Wirth (eds.), pub. Elsevier Science B. V., Washington D. C., USA, 1997.

[Kihl and Rumsewicz, 1996] Kihl M. and M. Rumsewicz, On overload control of Intelligent Peripherals in Intelligent Networks, in *Proceedings of the 1996 IEEE Conference on Global Telecommunications (Globecom'96)*, London, England, 1996.

[Kihl, 1999] Kihl M., *Overload Control Strategies for Distributed Communication Net*works, PhD Thesis, Lund University, Sweden, ISSN 1101-3931, 1999.

[Kleinrock, 1975] Kleinrock L., *Queuing Systems*, vol. I (Theory), pub. John Wiley & Sons Inc., 1975.

[Körner *et al.*, 1994] Körner U., C. Nyberg and B. Wallström, The Impact of New Services and New Control Architectures on Overload Control, in *Proceedings of the 14th International Teletraffic Congress*, J. Labetoulle and J. W. Roberts (eds.), pp. 275-283, pub. Elsevier Science B.V, Antibes Juan-les-Pins, France, 1994.

[Lee and Lim, 1989] Lee K. J. and Y. Lim, Performance analysis of the congestion control scheme, in *Proceedings of 8th IEEE INFOCOM*, vol. 2, pp. 691-700, Ottawa, Canada, April 1989.

[Lodge *et al.*, 1994] Lodge F., T. Curran, M. Gulyani and A. Newcombe, Intelligent Network Congestion Control Strategies and their Impact on User-Level Quality of Service, in *Proceedings of the Australian Telecommunication Networks and Applications Conference (ATNAC'94)*, pp. 67-632, Melbourne, Australia, December 1994.

[Lodge *et al.*, 1999] Lodge F., D. Botvich and T. Curran, Using Revenue Optimisation for the Maximisation of Intelligent Network Performance, in *Proceedings of the 16th International Teletraffic Congress*, P. Key and G. Smith (eds.), pp. 953-965, pub. Elsevier Science B. V., Edinburgh, Scotland, June 1999.

[Lodge, 2000] Lodge F., *An Investigation into Intelligent Network Congestion Control Techniques*, PhD Thesis, Dublin City University, Ireland, 2000.

[MacDonald and Archambault, 1994] MacDonald D. and S. Archambault, Using Customer Expectation in Planning the Intelligent Network, in *Proceedings of the 14th International Teletraffic Congress*, J. Labetoulle and J. W. Roberts (eds.), pp. 95-104, pub. Elsevier Science B.V, Antibes Juan-les-Pins, France, 1994.

[Manfield and Zukerman, 1992] Manfield D. R. and M. Zukerman, Analysis of congestion onset thresholds for CCITT SS7 networks, in *Proceedings of the 1992 IEEE Conference on Global Telecommunications(Globecom'92)*, December 1992.

[Manfield *et al.*, 1994] Manfield D. R., G. K. Millsteed and M. Zukerman, Performance Analysis of SS7 Congestion Controls under Sustained Overload, *IEEE Journal on Selected Areas in Communications*, vol. 12, no. 3, pp. 405-414, April 1994.

[McMillan and Rumsewicz, 1996] McMillan D. and M. Rumsewicz, Analysis of Congestion Control for SCCP Traffic & the Impact on Intelligent Network Services, in *Proceedings of IEEE 1996 Intelligent Network Workshop (IN'96)*, Melbourne, Australia, 1996.

[Milito *et al.*, 1991] Milito R. A., Y. Levy and Y. Arian, Dynamic algorithms for Distributed Queues with Abandonments, in *Proceedings of the 13th International Teletraffic Congress*, A. Jensen and V. B. Iversen (eds.), pp. 329-334, pub. Elsevier Science B. V., Copenhagen, Denmark, June 1991.

[Newcombe *et al.*, 1994] Newcombe A., D. D. Botvich, F. Lodge and T. Curran, A decision support system for assurance of quality of service in intelligent network service provisioning, in *LNCS 851 – Proceedings of the 2nd International Conference on Intelligence in Services and Networks (IS&N'94)*, H. Kugler, A. Mullery and N. Niebert (eds.), pp. 419-432, pub. Springer-Verlag, Aachen, Germany, September 1994.

[Northcote and Rumsewicz, 1995] Northcote B. and M. P. Rumsewicz, An investigation of congestion dynamics in CCS networks during STP processor overload, *Proceedings of 1995 IEEE International Conference on Communications (ICC'95)*, Seattle, USA, 1995.

[Northcote and Smith, 1998] Northcote B. S. and D. E. Smith, Service Control Point Overload Rules to Protect Intelligent Network Services, *IEEE/ACM Transactions on Networking*, vol. 6, no. 1, February 1998.

[Northcote, 1996] Northcote B. S., Analysing SCP congestion control strategies: A study of control dynamics, in *Proceedings of 1996 IEEE Intelligent Network Workshop (IN'96)*, Melbourne, Australia, April 1996.

[Nyberg and Olin, 1994] Nyberg H. and B. Olin, On load control of an SCP in the Intelligent Network, *in Proceedings of the Australian Telecommunication Network and Applications Conference*, Melbourne, Australia, December 1994.

[Nyberg and Olin, 1995] Nyberg H. and B. Olin, *Load Control of Data Bases in the Intelligent Network*, in Proceedings of the 12th Nordic Teletraffic Seminar, Espoo, Finland, 1995.

[OMG, 1999] Object Management Group (OMG), *IN/CORBA Interworking*, OMG document /dtc/99-12-02, December 1999, available (31/05/2001): http://www.omg.org.

[OPNET, 2001] OPNET Technologies, Inc., *OPNET Modeler™ Product Overview*, available (31/05/2001): http://www.opnet.com/products/modeler/home.html.

[Parlay, 1999] Parlay Industry Group, *Parlay API Business Benefits White Paper v1.0*, June 1999, available (31/05/2001): http://www.parlay.org.

[Patel *et al.*, 2000a] Patel A., Prouskas K., Barria J. and Pitt J., IN Load Control using a Competitive Market-based Multi-agent System, in *LNCS 1774 – Proceedings of the 7th International Conference on Intelligence in Services and Networks (IS&N 2000)*, J. Delgado, G. D. Stamoulis, A. Mullery, D. Prevedourou and K. Start (eds.), pp. 240-254, pub. Springer-Verlag, Athens, Greece, February 2000.

[Patel *et al.*, 2000b] Patel A., Prouskas K., Barria J. and Pitt J., A Computational Economy for IN Load Control using a Multi-Agent System, *Journal of Network and Systems Management*, vol. 8, no. 3, September 2000.

[Pettersson, 1996] Pettersson S., Some Results on Optimal Decisions in Network Oriented Load Control in Signaling Networks, *University of Karlskrona/Ronneby Technical Report*, ISRN HKR-RES-96/11-SE, Karlskrona, Sweden, 1996, available (31/05/2001): http://www.hk-r.se/fou.

[Pham and Betts, 1992] Pham X. H. and R. Betts, Congestion Control for Intelligent Networks, in *Proceedings of the 1992 International Zurich Seminar on Digital Communications, Intelligent Networks and their Applications*, pp. 375-378, Zurich, Switzerland, 1992.

[Pham and Betts, 1994] Pham X. H. and R. Betts, Congestion Control for Intelligent Networks, *Computer Networks and ISDN Systems*, vol. 26, pp. 511-524, January 1994.

[Rumsewicz and Smith, 1995] Rumsewicz M. P. and D. E. Smith, A Comparison of SS7 Congestion Control Options During Mass Call-In Situations, *IEEE/ACM Transactions on Networking*, vol. 3, no. 1, February 1995.

[Rumsewicz, 1993] Rumsewicz M. P., Analysis of the effects of SS7 message discard on call completion rates during overload, *IEEE/ACM Transactions on Networking*, vol. 1, no.4, August 1993.

[Rumsewicz, 1994a] Rumsewicz, M. P., Critical Congestion Control Issues in the Evolution of Common Channel Signaling Networks, in *Proceedings of the 14th International Teletraffic Congress*, J. Labetoulle and J. W. Roberts (eds.), pp. 115-124,pub. Elsevier Science B.V, Antibes Juan-les-Pins, France, 1994.

[Rumsewicz, 1994b] Rumsewicz, M. P., On the Efficacy of Using the Transfer-Controlled Procedure During Periods of STP Processor Overload in SS7 Networks, *IEEE Journal on Selected Areas in Communications*, vol. 12, no. 3, pp. 415-423, April 1994.

[Schoonderwoerd *et al.*, 1997] Schoonderwoerd R., O. E. Holland, J. L. Bruten, Ant-like agents for load balancing in telecommunications networks, in *Proceedings of the 1st International Conference on Autonomous Agents (AA'97)*, 1997.

[Seraj, 1985] Seraj J., An analysis for processor load control in SPC systems, in *Proceedings of the 11th International Teletraffic Congress*, M. Akiyama (ed.), pp. 767-773, pub. Elsevier Science B.V., Kyoto, Japan, September 1985.

[Skoog, 1988] Skoog R. A., Engineering Common Channel Signalling Networks for ISDN, in *Proceedings of the 12th International Teletraffic Congress*, M. Bonatti (ed.), pp. 915-921, pub. Elsevier Science B. V., Torino, Italy, June 1988.

[Smith, 1994a] Smith D. E., Effects of Feedback Delay on the Performance of the Transfer-Controlled Procedure in Controlling CCS Network Overloads, *IEEE Journal on Selected Areas in Communications*, vol. 12, no. 3, pp. 424-432, April 1994.

[Smith, 1994b] Smith D. E., Preventing Release Message Avalanches in Common Channel Signaling Networks, in *Proceedings of the 14th International Teletraffic Congress*, J. Labetoulle and J. W. Roberts (eds.), pp. 105-114, pub. Elsevier Science B.V, Antibes Juan-les-Pins, France, 1994.

[Smith, 1995] Smith D. E., Ensuring Robust Call Throughput and Fairness for SCP Overload Controls, *IEEE/ACM Transactions on Networking*, vol. 3, no. 5, pp. 538-548, October 1995.

[Turner and Key, 1991] Turner P. M. D. and P. B. Key, A New Call Gapping Algorithm for Network Traffic Management, in *Proceedings of the 13th International Teletraffic Congress*, A. Jensen and V. B. Iversen (eds.), pp. 121-126, pub. Elsevier Science B. V., Copenhagen, Denmark, June 1991.

[Wallström and Nyberg, 1991] Wallström B. and C. Nyberg, Transient Model of Overload Control and Priority Service in SPC-systems, in *Proceedings of the 13th International Teletraffic Congress*, A. Jensen and V. B. Iversen (eds.), pp. 429-434, pub. Elsevier Science B. V., Copenhagen, Denmark, June 1991.

[Wang, 1991] Wang J. L., Traffic Routing and Performance Analysis of the Common Channel Signaling System No.7 Network, in *Proceedings of the 1991 IEEE Global Telecommunications Conference (Globecom'91)*, pp. 301-305, 1991.

[Wellman, 1993] Wellman, M. P., A market oriented programming environment and its application to distributed multicommodity flow problems, *Journal of Artificial Intelligence Research*, vol. 1, no. 1, pp. 1-23, 1993.

[Willmann and Kühn, 1990] Willmann G. and P. J. Kühn, Performance modeling of Signaling System No. 7, *IEEE Communications Magazine*, vol. 28, pp. 44-56, July 1990.

[Willmann, 1988] Willmann G., Modelling and performance evaluation of multi-layered signalling networks based on the CCITT No. 7 specification, in *Proceedings of the 12th International Teletraffic Congress*, M. Bonatti (ed.), pp. 930-940, pub. Elsevier Science B. V., Torino, Italy, June 1988.

[Yamaki *et al.*, 1996] Yamaki H., M. P. Wellman and T. Ishida, A Market-Based Approach to Allocating QoS for Multimedia Applications, in *Proceedings of The Second International Conference on Multi-Agent Systems (ICMAS-96)*, pp. 385-392, 1996.

[Ygge, 1998] Ygge F., *Market-Oriented Programming and its Application to Power Load Management*, PhD Thesis, Lund University, Sweden, chapter 3, pp. 43-47, 1998.

[Zepf and Rufa, 1994] Zepf J. and G. Rufa, Congestion and Flow Control in Signaling System No. 7 – Impacts of Intelligent Networks and New Services, *IEEE Journal on Selected Areas in Communications*, vol. 12, no. 3, pp. 501-509, April 1994.

[Zepf and Willmann, 1991] Zepf J. and G. Willmann, Transient Analysis of Congestion and Flow Control Mechanisms in Common Channel Signalling Networks, in *Proceedings of the 13th International Teletraffic Congress*, A. Jensen and V. B. Iversen (eds.), pp. 413-419, pub. Elsevier Science B. V., Copenhagen, Denmark, June 1991.

LIST OF ACRONYMS

ACG:	Automatic Code Gapping
ACM:	Address Complete Message
ACM:	Association for Computing Machinery
AIN:	Advanced Intelligent Network
ANM:	Answer Message
ANSI:	American National Standards Institute
ASN.1:	Abstract Syntax Notation One
BCMP:	Baskett, Chandy, Muntz and Palacois
BCSM:	Basic Call State Model
BER:	Basic Encoding Rules
CAMEL:	Customised Access for Mobile network Enhanced Logic
CCC:	Call Count Control
CCF:	Call Control Function
CCS:	Common Channel Signalling
cdf:	cumulative distribution function
CG:	Call Gapping
CI:	Congestion Indication
CORBA:	Common Object Request Broker Architecture
CPE:	Customer Premises Equipment
CS-1:	Capability Set 1
CSL:	Component Sub-Layer
DPC:	Destination Point Code
ETSI:	European Telecommunications Standardisation Institute
FE:	Functional Entity
FIFO:	First-In-First-OUT
GSM:	Global System for Mobile communication
GSMSCF:	GSM Service Control Function
GTA:	Global Title Address
GTT:	Global Title Translation
IAM:	Initial Address Message
IEEE:	Institute of Electrical and Electronic Engineers
IETF:	Internet Engineering Task Force

IF:	Information Flow
IN:	Intelligent Network
INAP:	Intelligent Network Application Protocol
IO:	International Option
IP:	Intelligent Peripheral *or* Internet Protocol
ISDN:	Integrated Services Digital Network
ISUP:	Integrated Services Digital Network User Part
ITU:	International Telecommunications Union
LB:	The Leaky Bucket
LMC:	Load Measure Control
MAP:	Mobile Application Part
MPLS:	Multi-Protocol Label Switching
MSC:	Mobile Switching Centre
MSU:	Message Signal Unit
MTP:	Message Transfer Part
MTP1:	Message Transfer Part – Level 1
MTP2:	Message Transfer Part – Level 2
MTP3:	Message Transfer Part – Level 3
NO:	National Option
NOCP:	National Option with Congestion Priorities
OMG:	Object Management Group
OPC:	Origination Point Code
OSI:	Open Systems Interconnection
PC:	Point code
pdf:	probability distribution function
PINT:	PSTN/IN Interworking
pmf:	probability mass function
PSTN:	Public Switched Telephone Network
PT:	Percentage Thinning
QLC:	Queue Length Control
REL:	Release Message
RLC:	Release Complete Message
ROS:	Remote Operations Service
RSC:	Reset Circuit
RTB:	Retransmission Buffer
RTC:	Response Time Control
SCCP:	Signalling Connection Control Part
SCEF:	Service Creation Environment Function
SCF:	Service Control Function
SCP:	Service Control Point
SCTP:	Stream Transmission Control Protocol
SDF:	Service Data Function
SDP:	Service Data Point
SLA:	Service Level Agreement
SLP:	Service Logic Program
SLPI:	Service Logic Program Instance

SMAF:	Service Management Agent Function
SMF:	Service Management Function
SMH:	Signalling Message Handling
SNM:	Signalling Network Management
SP:	Signalling Point
SPIRITS:	Service in the PSTN/IN requesting Internet Service
SRF:	Specialised Resource Function
SS.7:	Signalling System No.7
SSCP:	Service Switching Control Point
SSF:	Service Switching Function
SSN:	Sub-System Number
SSP:	Service Switching Point
STP:	Signalling Transfer Point
svc:	square of the variation coefficients
TB:	Transmission Buffer
TC:	Transaction Capabilities
TCAP:	Transaction Capabilities Application Part
TFC:	Transfer Controlled
TIPHON:	Telecommunications and Internet Protocol Harmonisations Over Networks
TSL:	Transaction Sub-Layer
VoIP:	Voice-over-IP
VPN:	Virtual Private Network

APPENDIX A

SPECIFICATION OF SS.7/IN QUASI-ANALYTIC MODEL

This appendix provides a detailed specification of the SS.7/IN quasi-analytic model used in chapter 4 for the analysis of the performance of standard SS.7/IN load controls. An overview of the model structure and the techniques used is provided in §4.2. Here we present the model in terms of the specifications for the three sub-model types: §A.1 specifies the ISUP user part sub-model, §A.2 specifies the IN user part sub-model and §A.3 specifies the resource sub-model. In these specifications, the points at which information is passed between the various sub-models is indicated.

A.1 ISUP USER PART SUB-MODEL SPECIFICATION

This section specifies the iterative procedure to calculate the traffic load offered to the SS.7/IN network resources by the ISUP user part entities. The procedure takes into account the relevant characteristics of the service supported (ISUP telephony) and the operation of the ISUP load throttle described in §4.1.3.2.

Step 1: Calculate offered traffic load.

Set $n = n + 1$,
where n denotes the number of the current iteration interval ($n = 0$ initially).

For all UPs $u \in \{U_{ISUP}\}$, where u denotes an arbitrary UP and $\{U_{ISUP}\}$ denotes the set of all ISUP UPs, do:

For all UPs $u' \in \{U_{ISUP}\}$, where $u' \neq u$, do:

// Ascertain number of ISUP service requests for this iteration.

Generate a random number, $\phi_{u,IAM,u'}(t_n)$, using an Poisson distribution with the pre-specified mean for UP u for the current iteration interval (denoted by t_n). $\phi_{u,IAM,u'}(t_n)$ is the number of first-offered ISUP service requests (equivalent to the number of generated IAM messages) arriving at UP u for destination UP u' during iteration interval t_n.

Set $\lambda_{u,IAM,u'}(t_n) = \phi_{u,IAM,u'}(t_n) + \psi_{u,IAM,u'}(t_n)$,

where $\psi_{u,IAM,u'}(t_n)$ and $\lambda_{u,IAM,u'}(t_n)$ are respectively the reattempted and total number of ISUP service requests arriving at UP u with destination UP u' for iteration interval t_n.[103]

// Generate other ISUP service session message types.

Set $\lambda_{u,ACM,u'}(t_n) = \delta_{u,IAM,u'}(t_{n-1})$,

where $\delta_{u,IAM,u'}(t_{n-1})$ denotes the number of IAM messages that arrived at UP u from UP u' during the iteration interval t_{n-1}.[104]

If $n - T_{ISUP\_ANSWER} > 0$, where n denotes the number of the current iteration and T_{ISUP_ANSWER} denotes the number of iteration intervals used to model the answer delay for the ISUP service, then:

  Set $\lambda_{u,ANM,u'}(t_n) = \delta_{u,IAM,u'}(t_{n-T_{ISUP\_ANSWER}})$.

Else:

  Set $\lambda_{u,ANM,u'}(t_n) = 0$.

If $n - T_{ISUP\_TALK} > 0$, where T_{ISUP_TALK} is the number of iteration intervals used to model the conversation delay for the ISUP service, then:

  Set $\lambda_{u,REL,u'}(t_n) = \delta_{u,ANM,u'}(t_{n-T_{ISUP\_TALK}})$.[105]

Else:

  Set $\lambda_{u,REL,u'}(t_n) = 0$.

Set $\lambda_{u,RLC,u'}(t_n) = \delta_{u,REL,u'}(t_{n-1})$.

[103] For brevity we omit the specification of the procedure for calculation of the volume of reattempt traffic. This procedure estimates reattempt volumes as follows. Whenever service requests are throttled or service session messages discarded (with the exception of ISUP REL/RSC messages) the number of relevant messages is calculated. Then, in accordance with each service's reattempt probability, the volume of resulting reattempts is calculated and these are scheduled to occur evenly across a range of future iteration intervals. Scheduled reattempt volumes, along with estimates of proportions of these that are first, second, third *etc.* reattempts, are stored until the relevant iteration interval. At each iteration interval, reattempts that have already attempted the maximum number of times are discounted and the remainder added to the first-offered attempt volume.

[104] We define numbers of arrived and generated messages for the other ISUP (and later IN) messages types in a similar manner.

[105] Note that we assume that the initiator of the service request (the caller) always initiates the end of the session by going on-hook first, thereby triggering the sending of a REL message to the callee's ISUP UP. For simplicity we assume a constant conversation delay here, however in the results presented in chapter 4 the model has been modified to support variable length conversation delays for ISUP sessions.

Step 2: Perform ISUP load throttling procedure.

```
For all UPs $u \in \{U_{ISUP}\}$ do:
    For all UPs $u' \in \{U_{ISUP}\}$, where $u' \neq u$, do:
        // Apply ISUP load throttling procedure. If load level equals 1 throttle IAMs only, if load
        // level equals 2 throttle all ISUP message types.
        If $k_{ISUP,u,u'} = 1$, where $k_{ISUP,u,u'}$ is the ISUP load level at UP $u$ for destination UP $u'$, then:
            Set $\lambda_{u,IAM,u'}(t_n) = 0$.
        If $k_{ISUP,u,u'} = 2$ then:
            For all message types $j \in \{J_{ISUP}\}$, where $j$ denotes an arbitrary message type and
            $\{J_{ISUP}\}$ denotes the set of all ISUP message types, do:
                Set $\lambda_{u,j,u'}(t_n) = 0$.
// Schedule reattempts for the rejected service requests.

// The values of message arrival rates from the ISUP UPs (along with those for the IN UPs) are
// now inputted into the MTP model (cf. §A.3) The MTP model then provides the number of
// messages arriving at the ISUP UPs during the current iteration interval.
```

Step 3: Calculate incoming traffic.

```
For all UPs $u \in \{U_{ISUP}\}$, do:
    For all UPs $u' \in \{U_{ISUP}\}$, where $u' \neq u$, do:
        For all message types $j \in \{J_{ISUP}\}$ do:
            Set $\delta_{u,j,u'}(t_n) = 0$.
For all UPs $u \in \{U_{ISUP}\}$ do:
    For all UPs $u' \in \{U_{ISUP}\}$, where $u' \neq u$, do:
        For all links $l \in \{L_{INPUT,u}\}$, where $l$ denotes an arbitrary link and $\{L_{INPUT,u}\}$ is the set of
        links that directly input load into UP $u$,[106] do:
            For all message types $j \in \{J_{ISUP}\}$ do:
```

[106] For our network topology these will be the pair of links connecting the STP pair to the SSP in question.

```
            Set $\delta_{u,j,u'}(t_n) = \delta_{u,j,u'}(t_n) + \frac{\rho_{u',j,l,u}(t_n) \cdot C}{B_j}$,
            where $\rho_{u',j,l,u}(t_n)$ is the load on link $l$ during iteration interval $t_n$ due to messages
            of type $j$ originated by UP $u'$ and destined for UP $u$; $C$ is the capacity of link $l$
            (we assume all links have the same capacity) in bits per iteration interval; and $B_j$
            is the length in bits of messages of type $j$.
```

Step 4: Calculate probability of arrival of CIs whilst T29 timers are inactive.

```
For all UPs $u \in \{U_{ISUP}\}$ do:
   For all UPs $u' \in \{U_{ISUP}\}$, where $u' \neq u$, do:
      // Calculate probability of TFC arrival under assumption of Poisson arrival process.
      // This is also the probability of an increase in the UP's load level for the destination UPs
      // indicated by the TFCs.
      Set $P^i_{u,u'}(t_n) = P_{CI,u,u'}(t_n) = 1 - e^{-\delta_{u,TFC,u'}(t_n)[t_{T29} - \tau_{T29,u,u'}(t_n)]}$,
      where $P_{CI,u,u'}(t_n)$ is the probability of arrival of a CI for destination UP $u'$ at UP $u$ during
      the portion of iteration $t_n$ in which timer the relevant $T29$ timer is inactive; $P^i_{u,u'}(t_n)$ is
      the probability that the ISUP throttle load level of UP $u$ for destination UP $u'$ (denoted
      $k_{ISUP,u,u'}$) is increased (if possible) during iteration interval $t_n$; $\delta_{u,TFC,u'}(t_n)$ is the number
      of TFCs relating to destination UP $u'$ arriving at UP $u$ during interval $t_n$; $t_{T29}$ is the
      length of timer $T29$; and $\tau_{T29,u,u'}(t_n)$ is the length of time the $T29$ timer for UP $u'$ at UP
      $u$ is active during the iteration interval $t_n$.
```

Step 5: Calculate new load levels for the next iteration.

```
For all UPs $u \in \{U_{ISUP}\}$ do:
   For all UPs $u' \in \{U_{ISUP}\}$, where $u' \neq u$, do:
      // Use random number generation to decide if CIs are actually received.
      Generate a random number $X$ uniformly distributed in the range $(0,1)$.
      If $X < P^i_{u,u'}(t_n)$ then:
         // Increment load level if possible, set/reset T30 and decide CI arrival time.
         If $k_{ISUP,u,u'} < K_{ISUP}$, where $K_{ISUP}$ is the maximum load level for ISUP UPs, then:
            Set $k_{ISUP,u,u'} = k_{ISUP,u,u'} + 1$.
```

```
        Generate a random number $Y$ uniformly distributed in the range
        $(t_{T29} - \tau_{T29,u,u'}(t_n), t_{T29})$.
        Set $\tau_{T29,u,u'}(t_{n+1}) = t_{T29} - Y$.
        Set $\tau_{T30,u,u'} = t_{T29}$,
        where $\tau_{T30,u,u'}$ is the length of time for which the $T30$ timer for UP $u'$ at UP $u$ has been active.
    Else:
        // If T30 expiry decrement load level, otherwise update T30 active duration.
        If $\tau_{T30,u,u'} > 0$ and $\tau_{T30,u,u'} = A \cdot t_{T29}$, where $A$ is a pre-specified integer, then:
            If $k_{ISUP,u,u'} > 0$ then:
                $k_{ISUP,u,u'} = k_{ISUP,u,u'} - 1$.
            If $k_{ISUP,u,u'} > 0$ then:
                Set $\tau_{T30,u,u'} = t_{T29}$.        // Restart $T30$.
            Else:
                Set $\tau_{T30,u,u'} = 0$.        // $T30$ inactive.
        Else:
            $\tau_{T30,u,u'} = \tau_{T30,u,u'} + t_{T29}$.        // Increment $T30$.
```

A.2 IN USER PART SUB-MODEL SPECIFICATION

This section specifies the iterative procedure to calculate the traffic load offered to the SS.7/IN network resources by IN user parts. The model takes into account the relevant characteristics of the Freephone and Televoting services, as well as the operation of the ACG load throttling procedure and the SCCP load throttle.

Step 1: Calculate offered traffic load.

// Offered traffic load is calculated in the same manner as used for ISUP (cf. §A.1), so we // omit a specification of this procedure here.

Step 2: Perform ACG load throttling procedure.

```
For all UPs $u \in \{U_{IN}\}$, where $\{U_{IN}\}$ is the set of IN UPs, do:
    For all UPs $u' \in \{U_{IN}\}$, where $u' \neq u$, do:
```

// Apply ACG load throttling procedure. Freephone first, then Televoting.

Set $\lambda'_{u,F1,u'}(t_n) = \dfrac{\lambda''_{u,F1,u'}(t_n)}{1 + g_{u,FREEPHONE,u'}(t_n) \cdot \lambda''_{u,F1,u'}(t_n)}$.

Set $\lambda'_{u,T1,u'}(t_n) = \dfrac{\lambda''_{u,T1,u'}(t_n)}{1 + g_{u,TELEVOTING,u'}(t_n) \cdot \lambda''_{u,T1,u'}(t_n)}$,

where $F1$ and $T1$ are used to denote the first messages in Freephone and Televoting sessions respectively; $\lambda''_{u,F1/T1,u'}(t_n)$ is the number of service requests for Freephone/Televoting arriving at UP u for destination UP u' during iteration interval t_n; $g_{u,F.../T...,u'}(t_n)$ is the gap interval in place for Freephone/Televoting at UP u for UP u' during iteration interval t_n; and $\lambda'_{u,F1/T1,u'}(t_n)$ is the number of Freephone/Televoting requests accepted by the ACG throttle at UP u for UP u' during iteration interval t_n.

// Schedule reattempts for the rejected service requests.

Step 3: Perform SCCP load throttling procedure.

For all UPs $u \in \{U_{IN}\}$ do:

For all UPs $u' \in \{U_{IN}\}$, where $u' \neq u$, do:

For all message types $j \in \{J_{IN}\}$, where $\{J_{IN}\}$ is the set of all IN message types (including the ACG message type), do:

Set $\lambda_{u,j,u'}(t_n) = \left[1 - \dfrac{k_{SCCP,u,u'}(t_n)}{K_{SCCP}}\right] \lambda'_{u,j,u'}(t_n)$,

where $k_{SCCP,u,u'}(t_n)$ is the SCCP load level of UP u for UP u' during iteration interval t_n; K_{SCCP} is the maximum SCCP load level; and $\lambda_{u,j,u'}(t_n)$ is the number of type j messages accepted by the SCCP throttle at UP u for UP u' during iteration interval t_n.

// Schedule reattempts for the rejected service requests.

// The values of message arrival rates from the IN UPs (along with those for the ISUP UPs) are
// now inputted into the MTP model (cf. §A.3) The MTP model then provides the number of
// messages (both service-related messages and ACGs) arriving at the IN UPs.

Step 4: Calculate incoming traffic.

// Incoming traffic rates are calculated in the same manner as used for ISUP (cf. §A.1), so we
// omit a specification of this procedure here.

Step 5: Update ACG Gap interval.

```
For all UPs $u \in \{U_{IN}\}$ do:
    For all UPs $u' \in \{U_{IN}\}$, where $u' \neq u$, do:
        If $\delta_{u,FREEPHONE\_ACG,u'}(t_n) > 0$, where $\delta_{u,FREEPHONE\_ACG,u'}(t_n)$ is the number of Freephone-related ACG messages for UP $u'$ arriving at UP $u$ during iteration interval $t_n$, then:
            Set $g_{u,FREEPHONE,u'}(t_{n+1})$ = the highest of the values indicated in ACGs received this iteration interval.
        If $\delta_{u,TELEVOTING\_ACG,u'}(t_n) > 0$ where $\delta_{u,TELEVOTING\_ACG,u'}(t_n)$ is the number of Televoting-related ACG messages for UP $u'$ arriving at UP $u$ during iteration interval $t_n$, then:
            Set $g_{u,TELEVOTING,u'}(t_{n+1})$ = the highest of the values indicated in ACGs received this iteration interval.
```

Step 6: Calculate probability of arrival of CIs whilst T29 timers are inactive.

// The probability of CI arrival is calculated in the same manner as used for ISUP (cf. §A.1), so
// we omit a specification of this procedure here.

Step 7: Update SCCP load control's load levels.

// The SCCP load control updates its load levels in response to CI arrivals in a manner
// analogous to ISUP, (cf. §A.1), so we omit a specification of this procedure here.

A.3 RESOURCE SUB-MODEL SPECIFICATION

We now specify the procedure for calculation of the utilisation and queue lengths for links/SCPs, TFC generation rates and SCP overload levels. At each iteration interval the load offered to all resources in the network for the iteration interval is computed. This load includes load held in the resources queue at the start of the iteration interval, load arriving directly from UPs and load arriving from other resources in the network.

Step 1: Initialisation.[107]

```
Set $n = 0$.

For all UPs $u$ do:
    For all message types $j$ do:
        For all links $l$ do:
            Set $\rho_{u,j,l}(t_0) = 0$,
            where $\rho_{u,j,l}(t_0)$ is the load, in Erlangs, of messages of type $j$ from UP $u$ carried by
            link $l$ for iteration interval $t_0$.

For all links $l$ do:
    Set $m_l = 0$,
    where $m_l$ is the number of distinct grouping of messages held in the link $l$ queue. These
    groupings consist of messages offered to the link queue in previous iterations that have not
    yet been processed by the link.
    Set $o_l$ = FALSE,
    where $o_l$ is the overload status of link $l$.
    Set $q_l(t_0) = 0$,
    where $q_l(t_0)$ is the link $l$ input queue size, in bits, at the end of iteration interval $t_0$.
```

Step 2: Calculate message rates offered to network resources by UPs.

```
For all UPs $u$ do:
    For all message types $j$ do:
        For all links $l \in \{L_{OUTPUT,u}\}$, where $\{L_{OUTPUT,u}\}$ is the set of links to which UP $u$ directly
        outputs load, do:
            // Calculate the mean volumes of messages offered, denoted $\varphi_{u,j,l}(t_n)$, individually for
            // each UP $u$, message type $j$ and link $l$.[108]
```

[107] This step is executed only once: at the first iteration of the model. The other steps are executed at each iteration.

[108] These values are calculated by taking into account the messages generated by the UP models described in §A.1 and §A.2, as well as the routes in the network over which traffic must pass to reach its destination. We omit a detailed specification of these calculations here (they are specific to the network topology under study).

Step 3: Calculate resource utilisation and queue lengths.[109]

```
For all links $l$ do:[110]
    // Calculate the number of messages of the various types arriving at the link queue.
    For every link $l' \in \{L_{INPUT,l}\}$ do:
        For all UPs $u$ do:
            For all message types $j$ do:
                Set $\gamma_{u,j,l',l}(t_n) = \frac{\rho_{u,j,l'}(t_{n-1}) \cdot C}{B_j}$,
                where $\gamma_{u,j,l',l}(t_n)$ is the number of UP $u$ originated type $j$ messages from link $l'$
                arriving to link $l$ during iteration interval $t_n$.
    // Add the new message grouping to the queue.
    Set $m_l = m_l + 1$.
    For all UPs $u$ do:
        For all message types $j$ do:
            Set $x_{u,j,l,m_l}(t_n) = \varphi_{u,j,l}(t_n) + \sum_{\substack{l'=1 \\ l' \in \{L_{INPUT,l}\}}}^{L} \gamma_{u,j,l',l}(t_n)$,
            where $x_{u,j,l,m_l}(t_n)$ is the number of messages of type $j$ from UP $u$ contained in
            grouping number $m_l$ at link $l$.
            Set $\rho_l'(t_n) = \frac{\sum_{u=1}^{U}\sum_{j=1}^{J}\sum_{m=1}^{m_l} x_{u,j,l,m} \cdot B_j}{C}$,
            where $U$ is the number of UPs, $J$ is the number of message types, and $\rho_l'(t_n)$ is the
            total load in Erlangs offered to link $l$ during iteration $t_n$.
            // If calculated queue size is larger than queue capacity truncate the new message
            // grouping.
            If $\rho_l'(t_n) \cdot C > Q_{MAX}$, where $Q_{MAX}$ is the maximum size of the link queue in bits, then:
                For all UPs $u$ do:
```

[109] For convenience we do not, in the remaining specifications in this section, index message counts and utilisation values by their destination UP.

[110] Taking into account the restrictions on the ordering of these calculations imposed by the topology and routing scheme of the network under study.

```
        For all message types $j$ do:

            Set $x_{u,j,l,m_l}(t_n) = \frac{Q_{MAX} - q_l(t_{n-1})}{\rho_l'(t_n) \cdot C - q_l(t_{n-1})} x_{u,j,l,m_l}(t_n)$.

    Set $\rho_l'(t_n) = Q_{MAX} / C$.

// Schedule reattempts for the messages discarded as a result of truncating the
// message grouping.

If $\rho_l'(t_n) \le 1.0$ then:

    // All message groupings in the queue are processed by the link.

    For all UPs $u$:

        For all message types $j$ do:

            Set $\rho_{u,j,l}(t_n) = \frac{1}{C} \sum_{m=1}^{m_l} x_{u,j,l,m}(t_n) \cdot B_j$.

    Set $m_l = 0$.

    Set $q_l(t_n) = 0$.

    Set $\rho_l(t_n) = \sum_{u=1}^{U} \sum_{j=1}^{J} \rho_{u,j,l}(t_n)$,

    where $\rho_l(t_n)$ is the utilisation of link $l$.

Else:

    // Not all message groupings are processed by the link.

    Set $m = 0$.        // First initialise variables.

    Set $\rho_l(t_n) = 0$.

    For all UPs $u$ do:

        For all message types $j$ do:

            Set $\rho_{u,j,l}(t_n) = 0$.

    // Loop through message groupings in the queue until the calculated link utilisation
    // exceeds 1 (full link utilisation).

    While $\rho_l(t_n) < 1.0$ and $m < m_l$ do:

        Set $m = m + 1$.

        For all UPs $u$ do:

            For all message types $j$ do:
               Set $\rho_{u,j,l}(t_n) = \rho_{u,j,l}(t_n) + \frac{x_{u,j,l,m}(t_n) \cdot B_j}{C}$.
           Set $\rho_l(t_n) = \sum_{u=1}^{U} \sum_{j=1}^{J} \rho_{u,j,l}(t_n)$.
       // Calculated link utilisation will exceed 1, so adjust utilisation values and
       // numbers of messages, relating to the partially-processed message grouping, that
       // remain in the queue.
       If $\rho_l(t_n) > 1.0$ then:
           For all UPs $u$ do:
               For all message types $j$ do:
                   Set $\rho_{u,j,l}(t_n) = \frac{1}{\rho_l(t_n)} \rho_{u,j,l}(t_n)$.
                   Set $x_{u,j,l,m}(t_n) = \frac{\rho_l(t_n) - 1}{\rho_l(t_n)} x_{u,j,l,m}(t_n)$.
           // Remove processed message groupings from the queue (by changing indices
           // of the remaining message groupings)
           For all UPs $u$ do:
               For all message types $j$ do:
                   Set $m'' = m$.
                   While $m'' < m_l + 1$ do:
                       Set $x_{u,j,l,(m''-m+1)}(t_n) = x_{u,j,l,m''}(t_n)$.
                       Set $m'' = m'' + 1$.
           Set $m_l = m_l - m + 1$.
           Set $q_l(t_n) = \sum_{u=1}^{U} \sum_{j=1}^{J} \sum_{m=1}^{m_l} x_{u,j,l,m}(t_n) \cdot B_j$.
           Set $\rho_l(t_n) = 1.0$.                    // Link fully utilised this iteration interval.
// Calculate the input queue length and utilisation for SCPs in the same manner as used for the
// links. For brevity we omit a detailed specification of these calculations here.[111]
```

[111] Calculation of SCP utilisation and queue size will generally take place at some mid point during the calculation of link utilisation and queue lengths. In our case it takes place after the calculations for L8/L10 and before those for L9/L11.

Step 4: Calculate overload level for SCPs.

```
For all SCPs $i$, where $i$ denotes an arbitrary SCP, do:
    // Calculate overload level using the processor utilisation measurement.
    If $\rho_{SCP,i} > \rho_{UPPER}$, where $\rho_{SCP,i}$ is the utilisation of SCP $i$ and $\rho_{UPPER}$ is the upper threshold for SCP utilisation during overload, then:
        If $o_{SCP,i} < o_{MAX}$, where $o_{SCP,i}$ is the overload level of SCP $i$ for iteration interval $t_{n-1}$ and $o_{MAX}$ is the maximum SCP overload level, then:
            Set $o_{SCP,i} = o_{SCP,i} + 1$.
    If $\rho_{SCP,i}(t_n) < \rho_{LOWER}$, where $\rho_{LOWER}$ is the lower threshold for SCP utilisation during overload, then:
        If $o_{SCP,i} > 0$ then:
            Set $o_{SCP,i} = o_{SCP,i} - 1$.
```

Step 5: Calculate overload status of links.

```
For all links $l$:
    // Calculate link overload status according to the International Option MTP procedures.
    If $q_l(t_n) > D_{ONSET}$ and $o_l$ = FALSE, where $D_{ONSET}$ is the link overload onset threshold in bits, then:
        Set $o_l$ = TRUE.
    If $q_l(t_n) < D_{ABATE}$ and $o_l$ = TRUE, where $D_{ABATE}$ is the link overload abatement threshold in bits, then:
        Set $o_l$ = FALSE.
```

Step 6: Calculate TFC generation rates.

```
For all UPs $u$ do:
    For all UPs $u'$, where $u' \neq u$, do:
        Set $\lambda_{TFC,u,u'}(t_n) = 0.0$.

For all UPs $u$ do:
    For all UPs $u'$, where $u' \neq u$, do:
        For all links $l$ do:
```

// Generate TFCs in proportion to processed messages for overloaded links.

If o_l = TRUE then:

Set $\lambda_{TFC,\mu,\mu'}(t_n) = \lambda_{TFC,\mu,\mu'}(t_n) + \dfrac{C}{N}\sum_{j=1}^{J}\dfrac{\rho_{\mu,jJ,\mu'}(t_n)}{B_j}$,

where one TFC message is generated for every N messages destined for a UP at the overloaded link.

// The UP models calculate the number of service-related messages of various types they have // received during the current iteration interval and generate the appropriate response // messages. They also calculate the number of CIs and ACG messages they receive and use // these to update their load levels and gapping parameters as appropriate. These procedures // are detailed in §A.1 and §A.2.

Go to Step 2.