Master Thesis
Software Engineering
Thesis no: MSE-2003: 24
August 2003

On the Scalability of Four Multi-Agent Architectures for Load Control Management in Intelligent Networks

Raheel Ahmad

Department of
Software Engineering and Computer Science
Blekinge Institute of Technology
Box 520
SE – 372 25 Ronneby
Sweden

This thesis is submitted to the Department of Software Engineering and Computer Science at Blekinge Institute of Technology in partial fulfillment of the requirements for the degree of Master of Science in Software Engineering. The thesis is equivalent to 20 weeks of full time study.

Contact Information

Author: Raheel Ahmad
Address: C/o Lindholm, Gränsvägen 13, 372 37 Ronneby, Sweden
E-mail: raheel_a@yahoo.com, raah02@student.bth.se

University Advisor: Professor Paul Davidsson
Department of Software Engineering and Computer Science
Blekinge Institute of Technology

Department of
Software Engineering and Computer Science
Blekinge Institute of Technology
Box 520
SE – 372 25 Ronneby
Sweden

Internet : www.bth.se/ipd
Phone : +46 457 38 50 00
Fax : +46 457 271 25

To my parents, Aijaz Ahmad & Nazra Parveen and my brother Adeel Ahmad. Your everlasting support and encouragement is the inspiration behind all my achievements.
Your Love added colors to my life.

ABSTRACT

Paralleling the rapid advancement in the network evolution is the need for advanced network traffic management surveillance. The increasing number and variety of services being offered by communication networks has fuelled the demand for optimized load management strategies. The problem of Load Control Management in Intelligent Networks has been studied previously and four Multi-Agent architectures have been proposed. The objective of this thesis is to investigate one of the quality attributes namely, s*calability* of the four Multi-Agent architectures. The focus of this research would be to resize the network and study the performance of the different architectures in terms of Load Control Management through different scalability attributes. The analysis has been based on experimentation through simulations. It has been revealed through the results that different architectures exhibit different performance behaviors for various scalability attributes at different network sizes. It has been observed that there exists a trade-off in different scalability attributes as the network grows. The factors affecting the network performance at different network settings have been observed. Based on the results from this study it would be easier to design similar networks for optimal performance by controlling the influencing factors and considering the trade-offs involved.

Keywords: Intelligent Network, Multi-Agent, Load Control, Scalability

ACKNOWLEDGEMENTS

First and foremost I would like to express my gratitude to my supervisor Paul Davidsson who has been a source of immense knowledge for me during this work. Stimulating discussions about this research with him always broadened my vision and brought maturity to my thoughts and work. Those intelligent questions he always posed, provoked me to think more and read between the lines. Without his constant support and guidance, this thesis would not have been possible.

I would also like to thank Stefan Johansson for his support and help during the entire duration of this work. Especially, for helping me out understand some essentials needed for this research. Some of the fruitful discussions that I had with him are reflected in parts of this work.

And especially, I would like to thank my parents for supporting me throughout this work

Table of Contents

CHAPTER 1

1 Introduction

In the ever-changing world we live in, nothing is permanent except *change*. The rapid pace of technological change has become a way of life. Certainly, the Telecommunications industry has been learning to survive in a highly dynamic environment. This rapid pace has been fuelled by customer demands for high-quality voice, data and multimedia communication services. Today, the technological progress in fibre optics, microprocessors, signal processing, photonics, software engineering and advanced network technologies is offering connectivity, *anywhere* and *anytime*. Our Telecommunication networks have already stepped into *Intelligent Optical Networks, 3G Wireless Networks* and *QoS-based Packet Networks*. As we develop more and more sophisticated technology, the study of these complex systems is a new challenge we face.

1.1 Resources - A Limitation in Communication Networks

Paralleling the rapid advancement in the network evolution is the need for *advanced* network traffic management surveillance. Rapid increase in the *network complexity* and *information volumes* has put forward the challenge of meeting *network availability* and *service quality*, for telecommunication Service Providers. *Congestion* in computer networks is becoming a significant problem due to the rapid growth in use of these networks, as well as due to increasing mismatch in link speeds caused by intermixing of old and new technology. Load Control mechanisms play a key role in performance management of these networks. Inefficient management of the available resources could turn the network susceptible to overloads, service denials and in the worst case system crash, all of which results in customer dissatisfaction. However, even highly advanced networks of today are faced with the dilemma of limited resources, similar to their ancestors.

The fundamental problem underlying the traditional load control mechanisms in today's networks is the lack of *efficient* management strategies of the processing units. When people and devices with good connectivity are clustered, they tend to do things at the same time and

hence get synchronized. For example when there is a natural disaster, an accident or a traffic jam on the road. The communication network receives service requests in synchronized bursts, overloading the network. For a telecommunication network lacking optimal strategies to manage network resources, this unexpected increase in the traffic load would result in network overload situations. On the other hand, a telephone network operator is confined to reserve resources and bandwidth for (i) communication within network elements and (ii) for emergency calls (e.g. 112, 911), which must always succeed, even when the network resources are overloaded. A need for *efficient* and *optimized* load control management strategies for communication networks is essential.

1.2 Intelligent Network Load Control Problem

The future advanced public communications networks will be built on three pillars i.e. *Bandwidth*, *high-speed Switching & Routing* and *Network Intelligence* [6]. The first of these pillars is provided by fiber-optics and a plethora of high-speed transmission schemes; the second increasingly provided by ATM and the new generation of routers. The third pillar will be the Intelligent Network (IN), a concept that is leading to new market opportunities as technological development and customer demands become more sophisticated. Technological advances have led to the increased usage of Telecommunication networks, which have been driven in part by the use of *Intelligent Networks* [11].

The increasing number and variety of services being offered by communication networks has fuelled the demand for greater network capacity. During peak periods of resource usage, an IN can become overloaded with service requests which leads to a degradation in the Quality-of-Service *(QoS)* provided by IN [11]. IN Load Control can be seen as a distributed resource allocation problem [1]. This problem deals with the allocation of resources between a number of *customers*, as well as the amount of the resources made available by the *providers,* that vary over time [4]. The needs and the available resources vary not only on an individual level but, on the overall *system* level. The load control mechanism has to deal with service requests that are unpredictable with respect to *time* and *volume*.

1.3 Scalability

The word *scalability* is used in a variety of ways by different simulation communities. Law [15] argues that, evaluation of a simulation system's scalability can be conducted at varying levels of abstraction throughout the systems life cycle to promote extensibility and support the software development process in areas ranging from requirements analysis to algorithm selection. The architectural capabilities such as algorithms, instruction mixes, execution times, message communication and performance requirements of a simulation, over the system development lifecycle provide parameters for scalability measures over the range of anticipated uses as well as system performance. As the simulation software is implemented, unit, component and system tests can be conducted to confirm the predictions of the scalability analysis efforts and detect unanticipated obstacles to scalability [15].

The use of Agent Technology for IN load control, using a market-based mechanism, has been studied by Arvidsson et al. [1], Carlsson et al. [2] and Johansson et al. [3]. With the help of a discrete event simulator proposed by Arvidsson et al. [1] and implemented by Ericsson AB, Johansson et al. [3] have started to evaluate and validate two of these four architectures. Kristell [13] has worked on implementing and comparing these four Multi-Agent architectures, especially finding and analysing the attributes that describe how these

architectures differ from each other. The main area of research for this thesis would be the study of Scalability of the four Multi-Agent architectures for IN load control.

1.4 The Research Question

It is possible to evaluate the Multi-Agent Systems with respect to more general quality attributes, such as *robustness, modifiability* and *scalability*. This Masters' thesis aims to investigate the Scalability of these four architectures. The objective would be to resize the overall network and study its performance in terms of Load Control Management through different scalability attributes.

The main investigations of this work would be,

- Impact of a resize in the Network, on an architectures' overall performance in terms of load management
- Comparison of the performance of four architectures over different scalability attributes.

The research carried out in this thesis would be a continuation of previous work by Arvidsson et al. [1], Johansson et al. [3], Carlsson et al. [2] and Johansson et al. [4] in the area of Load Control Management for Intelligent Networks. The investigations, in this thesis, would be based on results gathered through experimentation, using the Simulator. The results thus collected would then be analyzed.

1.5 Outline of the Thesis

The thesis is organized as follows,

Chapter 1, provides an Introduction to the problem of Resource Allocation in Telecommunication Networks, especially Intelligent Networks. It also provides an overview of the intended research for this thesis.
Chapter 2, gives a brief overview on Intelligent Networks, their physical components and architecture.
Chapter 3, contains a brief explanation of the four Multi-Agent architectures for IN load control, including a brief description of the Agent types involved.
Chapter 4, describes Scalability in general and *attributes* for scalability, which are investigated in this work to evaluate the four Multi-Agent architectures.
Chapter 5, enlists the general preconditions and configurations for the simulator, and input parameters for simulations.
Chapter 6, provides a detailed analysis of the results that have been recorded, based on different experiments, using the simulator.
Chapter 7, presents general conclusions based on the analysis, and recommendations for future work.

CHAPTER 2

2 Intelligent Networks

2.1 Evolution of Intelligent Network

Intelligent Network is a telephone network architecture originated by Bell Communications Research (*Bellcore*) in the United States. During the mid-1980s, Bell Operating Companies in the United States began requesting features to meet challenges as,

- Rapid deployment of services in the network
- Vendor Independence
- Standard Interfaces
- Offer services for Increased Network Usage

Telcordia Technologies Inc. responded to this request and developed the concept of Intelligent Network 1 (IN/1) in the mid-1980s, normally referred to as Intelligent Network or IN.

2.2 The Objectives

It seems that one of the major business imperatives driving the work on Intelligent Networks is an emerging competition among the operating companies as a result to responsiveness on the ever-changing customer demands. The Telecommunication Service Providers demand more control over the design and development of services, as well as a common technological base on which to deploy these services. This has resulted in a number of key objectives that IN technology has to satisfy. Some of them are discussed below.

2.2.1 Broadened Range of Services

To go beyond traditional voice and data bearer services and move into the dimensions of information services, broadband and multi-media.

2.2.2 Increased Service Velocity at Low Cost

To enable market driven, rapid introduction of new services, with a direct responsiveness to ever-changing customer needs. The new services should be 'optimized' for low costs.

2.2.3 Enable Vendor-Independent Deployment

Ensure services, which are independent of vendors' equipment and be able to work over multi-vendor equipment. This needs a high level of flexibility in the network in terms of hardware and software. The IN architecture should be designed such that integration within and among the software and hardware, possibly from different vendors, is possible.

2.2.4 Evolve from Existing Networks

The new network must inter-work and evolve from the existing network, since it would be very expensive to replace the existing networks.

2.3 What is an Intelligent Network (IN)

The Intelligent Network is more than just network architecture. An Intelligent Network is a *service-independent* telecommunications' network [5]. In an Intelligent Network, *intelligence* is taken out of the switch and placed in computer nodes that are distributed throughout the network. It is a complete framework for the creation, provisioning and management of advanced communications services [6], in which the service logic for a call is located separately from the switching facilities, allowing services to be added or changed without having to redesign switching equipment. This provides the network operator with the means to develop and control services more efficiently and new capabilities can be rapidly introduced into the network. Once introduced, services could easily be customized to meet individual customers' needs. According to Bell Atlantic, IN is a 'service-specific' architecture. That is, a certain portion of a dialled phone number, such as 800 or 900, triggers a request for a specific service.

IN has effectively put the destiny of incumbent carriers squarely in its own hands [6]. Distributing readily programmable *intelligence* across their networks and freed telecommunication service providers from their traditional dependence on switch manufacturers in the all-important provisioning of new AIN services. Software has had to be installed in far fewer locations, thanks to SS7s[1] support of centralized intelligent nodes.

A later version of IN called *Advanced Intelligent Network (AIN)* introduces the idea of a 'service-independent' architecture in which a given part of a telephone number can be interpreted differently by different services depending on factors such as time of day, caller identity, and type of call. AIN makes it easy to add new services without having to install new phone equipment. *For a more detailed understanding of IN and AIN, an interested reader is referred to [5], [6] and [7].*

[1] SS7 – refer to [14] for details on SS7 Network.

2.4 IN Architecture

Tsun-Chieh Chiang et al. [9] discusses that IN provides a framework to decouple service logic from switching nodes and make it easily accessible from other nodes within the network. In the physical plane of this framework, switching nodes are called S*ervice Xwitching Points* (SSP), while network nodes that host services are called S*ervice Control Points* (SCP). Call control and service switching functions are implemented within the SSP, while the service control functionalities are hosted by the SCP. Figure 2.4.1 (a) shows the physical architecture of an Intelligent Network.

An SSP implements connection management capabilities and supports a Signaling System 7 (SS7) signaling interface. IN defines standard call states and triggers[2], that can be enabled to cause the SSP to suspend call processing and query an SCP for further instructions on how to treat the call. An SCP hosts IN services that are accessed by SSP, performs real-time database query processing, and enables the SSP to access other network resources such as Intelligent Peripherals (IP) as required during call processing.

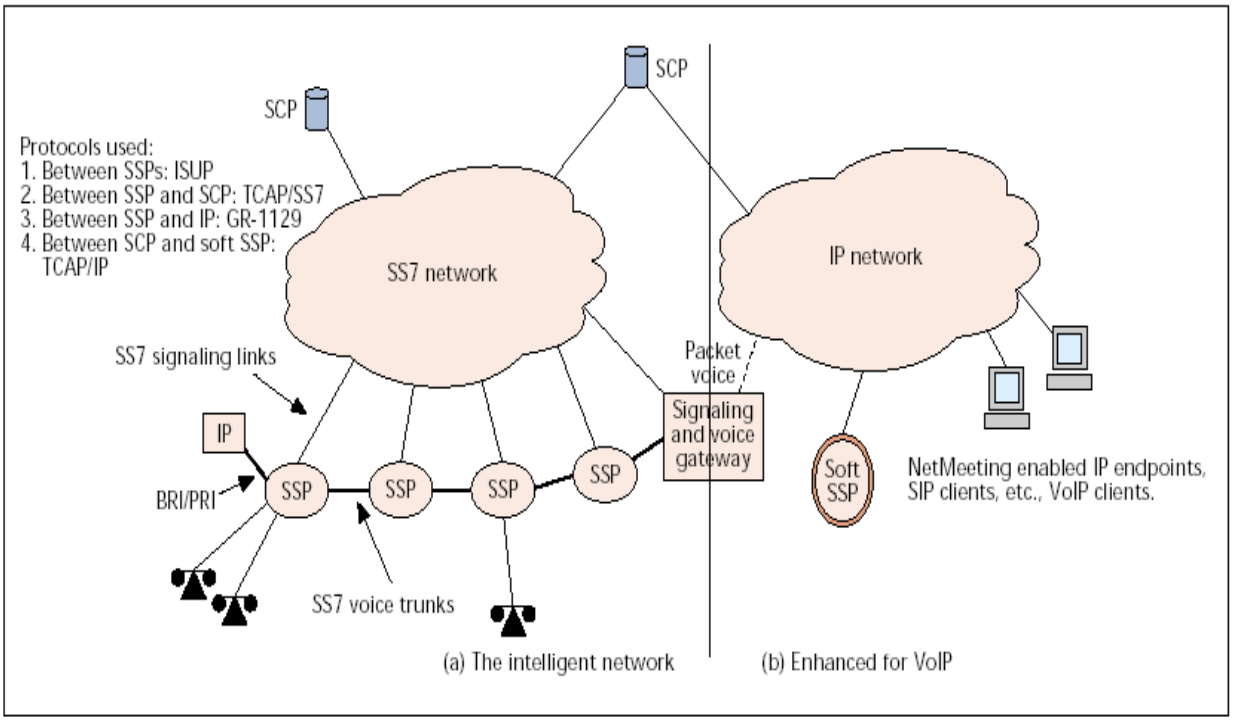


Fig. 2.4.1, The Intelligent Network Architecture [9]

The Intelligent Network architecture builds on a traditional network architecture while adding new elements. Amir-Ebrahimi et al. [8] discusses the six key elements of an Intelligent Network as discussed below. *For more details about the physical architecture of the IN, please refer to [8] and [9].*

[2] trigger – The process of detection of an IN call is known as 'triggering' [8].

2.4.1 Service Switching Points (SSP)

The SSP functionality is the key IN functionality of the switching system. An SSP switch is capable of detecting an IN service call, sending a signaling query to an SCP and responding to the responses/commands received from the SCP. The SCP, after analyzing the request, returns the appropriate control information to the SSP on how to process the call. This capability enables an IN end-office switch to interface with SCP databases using SS7 over Transaction Capabilities Application Part (TCAP) protocol.

2.4.2 Service Control Point (SCP)

The SCP has call control logic often with access to a co-resident database. It fields queries from the SSP, parses and processes them and provides the appropriate responses. It provides real time call processing logic for IN calls. It compares the information about the call provided in the query to the telco-defined data available for each service on the SCP to determine how the call should be treated (*It does not matter to an SCP-based service whether the call originated from the Public Switched Telephone Network (PSTN) or an Internet endpoint. As long as the SCP gets the required information through a standard protocol (TCAP) in a predefined format, it will execute its service logic and return the results to the SSP*). This is indicated to the SSP in the response message. Depending on this response, the SSP may either (i) route the call, (ii) forward to an announcement, or (iii) prompt the caller for collection of additional Dual-Tome Multi-Frequency (DTMF[3]) information.

2.4.3 Signal Transfer Point (STP)

Signal Transfer Points form the backbone of the Common Channel Signaling System 7 network. They are an integral part of the IN architecture and provide TCAP signaling between SSPs and SCPs.

2.4.4 Service Node (SN)

The Service Node provides an additional level of capabilities by providing a platform where IN calls can be forwarded to. These calls can access any of the existing services on the Service Node e.g. voice/fax mail or routing to another destination etc. Service Nodes can also be accessed by non-IN switches.

2.4.5 Service Creation Environment (SCE)

The SCE is a stand-alone product that provides a development environment for creating service applications and generating platform software for SCPs and SNs. It provides tools for writing and editing of Service Packages and compiling them for use on the network elements. The SCE enables the network provider to develop, test, deploy, and modify Intelligent Network services for the SCP and SN in a rapid and flexible manner.

[3] In this case the SCP will use additional information from the caller to decide how to route the call.

2.4.6 Service Management System (SMS)

The SMS provides an automated (remote) system for management of call processing data on the network element. Key data management functions are Insertion, Deletion and Update of data for a customer or service.

2.5 Intelligent Network Services

Intelligent Networks have been around long enough to give rise to several widely successful services, the most notable of which are *Toll-Free Numbers*[4]*, Carrier Calling Cards*[5] *and Virtual Private Networks*. A few examples of the current IN services are:

- *Routing by Day of Week* - A call is routed to a specific destination number based on the day of a week.
- *Routing by Time of Day* - A call is routed to a specific destination number based on the time of the day.
- *Portable Remote Number (Local Number Portability, LNP)* - The basic level of LNP is that of service portability i.e. the ability of the customer to choose providers while retaining the same telephone number [6].

2.6 The Future of Intelligent Networks

Finkelstein et al. [10] discuss how Intelligent Networks could play a role as carriers move to Next Generation Networks[6] (NGN) by inter-working with the Internet and Packet-based Networks to produce new *hybrid services*[7]. They have proposed a detailed migration path, discussing transition scenarios of IN to Next Generation Networks, for the future, *refer to [10] for further details.* Figure 2.4.1 (b) shows an SS7 based IN, enhanced for VoIP.

According to Finkelstein et al. [10], IN could play a role in four major areas, in communication networks of the future,

- Adding IN functionality to Internet capabilities to produce hybrid services.
- Serving as the model for the network controller in VoP networks (Voice-over-Packet Networks).
- Facilitating the interconnection between the Public Switched Telephone Network (PSTN) and NGN.
- Providing advanced voice services for VoP and NGN.

[4] Toll-Free Numbers - The party called, pays for the call.

[5] Carrier Calling Cards – Third party long-distance, calling cards.

[6] Next Generation Networks - An NGN is loosely defined as a packet-based telecommunications network that employs new distributed processing, control, management, and signaling techniques to provide all types of services, from basic narrowband voice telephony services to advanced broadband multimedia services. Hence, the concept of an NGN encompasses hybrid and VoP networks, but is broader in scope [10].

[7] Hybrid Services – IN inter-working with the Internet offers what the industry refers to as *hybrid* services [10].

The newer IN topologies would likely add Internet access, Tele-shopping, Telecommuting[8], and Video-on-Demand to commercial service mix.

[8] Telecommuting - Using IN technology, a Central Office switch can automatically forward an incoming business call to a remote workers' current location through a company communication server. These 'follow-me' routing schemes eliminate the need for a company to maintain distributed calling equipment [6].

CHAPTER 3

3 Agent-based Approaches to IN Load Control

The research community has recognized that Agent-based technology appears to offer a timely solution to the growing problem of designing efficient and flexible network management strategies [11]. Arvidsson et al. [1] have argued that the use of Agent technology for solving network load control problems is an effective methodology for building *flexible*, *adaptable* and *robust* solutions for this inherently distributed problem. They have further described the advantages offered by Agent technology over the traditional node-based mechanisms for IN load control. Patel et al. [11] have particularly discussed the distinct advantages of Multi-Agent technology over a single agent solution, especially for dealing with the problem of load control management in Intelligent Networks. In the following section is described, in brief, the four Multi-Agent Architectures for IN load control.

3.1 The Agent Types

Resource allocation in IN involves three main tasks [1],

1. Monitoring utilization levels of resources (SCP processors).
2. Control of allocation of these resources.
3. Grant/Denial of permission to individual service requests.

The three Agent types proposed by Arvidsson et al. [1], for IN load control, are discussed below,

3.1.1 Allocators

Allocators are associated with the Service Switching Points, which act as the point of entry for the service requests. They perform task (3), controlling the entry of service requests into the network. To control the incoming service requests, they form a *view* of the current load situation in the network, based on their 'predictive algorithm(s)' and information received by

other agents. Where possible, they also control the routing of traffic, with the objective of achieving optimal load balancing.

3.1.2 Quantifiers

Quantifiers are associated with the Service Control Points and perform task (1), monitoring and prediction of available capacity on SCP processors and reporting this information to other agents. They also implement node-based prediction mechanisms involving the selective discard of traffic in response to local overload onset detection.

3.1.3 Distributors

Distributors maintain an *overview* of the load and resource status throughout the network, communicating with other agents and observing through their own 'analysis algorithms'. They perform task (2), controlling and playing a supervisory role in allocating resources associated with Quantifiers.

3.2 Four Multi-Agent Architectures for IN Load Control

The four Multi-Agent architectures classified in terms of 'synchronization' and 'distributedness' are,

Centralized Auction and Centralized Leaky Bucket Architectures are *centralized*. The Hierarchical Distributed Auction and the Mobile Broker Architectures are *distributed.* In aspect of resource allocation, Centralized Auction and Hierarchical Distributed Auction Architecture are *synchronized*. Centralized Leaky Bucket Architecture and Mobile Broker Architecture are *asynchronous*

By *distributedness,* it is meant *to what degree the control over the system is distributed*. Two possible extremes could be, peer-to-peer autonomous agent systems and threaded systems (which all control is dependant on single agent) [4].

The degree of *synchronization* is a measure of how the execution of agents, interrelate with each other. There are highly 'sophisticated' agents, which interact only at special instances of time. Yet there are systems in which agents interact continuously, and are 'asynchronous' in nature [4].

Below is a short description of the four Multi-Agent architectures for IN load control.

3.2.1 The Centralized-Auction (CA) Architecture

The Centralized-Auction architecture is *centralized* and *synchronous*. Each SCP has a corresponding Quantifier, which supervises the SCP, keeping track of its processing capacity, its current load & status. On the other hand, each SSP has a corresponding Allocator which monitors the experienced load at its respective SSP and manages to buy the *correct* amount of SCP processing resources, an SSP node is expected to require. Figure 3.2.1 represents a diagrammatic representation of a typical Centralized-Auction Architecture.

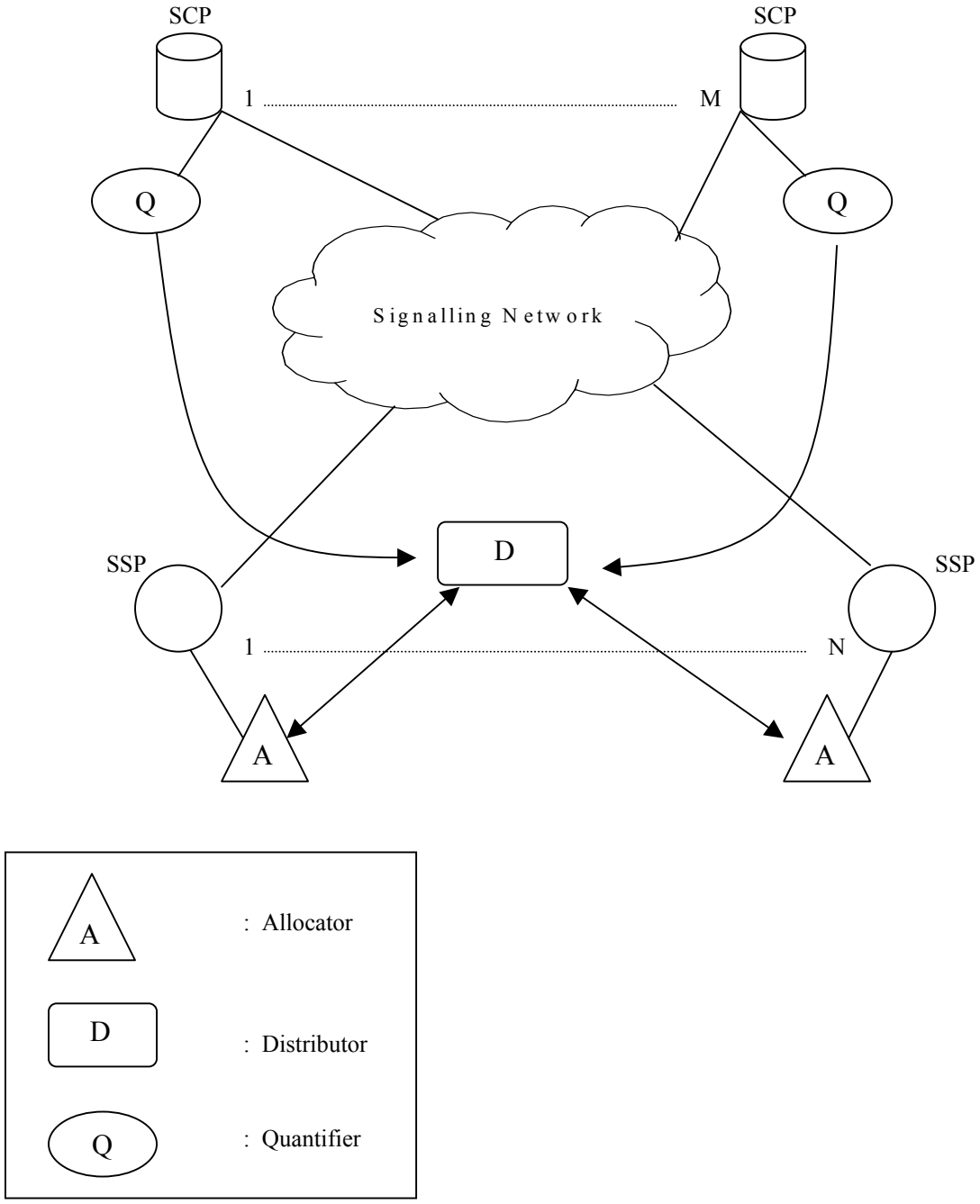


Fig. 3.2.1, The Centralized-Auction (CA) Architecture

In the CA architecture, all the Allocators maintain a pool of tokens (each token corresponds to an SCP's processing of a service request). These tokens are sold by the Quantifiers (on behalf of the respective SCPs) to the Allocators (acting on behalf of the respective SSPs) in every auction. These auctions are carried out by the Distributors, typically every 10 seconds, where the Quantifiers sell the SCP processing capacity to the Allocators, who send in their bids to buy the processing power offered by the Quantifiers. When an SSP accepts a service request from a subscriber, a token is removed from the pool of available tokens of the respective SSP. To serve a request, a direct connection between an SSP and an SCP is established. In the absence of any tokens, in a pool, an SSP would not accept any service requests.

In situations where the demand for the services goes beyond the available resources, there is a probability that SSPs would run out of the tokens, just after an auction, even when there is considerable time left in the next auction. Hence a 'Rejection Probability' is calculated so that the remaining tokens are equally distributed over time, between two auctions (see 'Percent Thinning' in the next section) in network overload situations.

As the load in the Intelligent Network is varying, it is unwise to utilise the IN resources at a maximum i.e. 100%. The reason being that in overload situations there would be no spare resources left over for emergency calls. The optimal solution should aim to utilise the system resources to a level below the system's total capacity, referred to as the 'Target Load'. Typically the Target load is set to 90%, which means that the load control algorithm should ensure that the total load on the system resources should not exceed the Target Load, i.e. 90% of the total available capacity. This is done by rejecting service requests, when the Offered Load exceeds the Target Load.

The Centralized Auction architecture originally proposed and implemented by Arvidsson et al. [1] was designed to maximize profits (also called Profit Optimization) for the Network Owner by favouring the services that result in higher profits. Which means that in network overload situations, the SSPs would reject the requests that would result in low profits, over high profit services. However in the absence of such a mechanism, the main task of an auction would be reduced only to distribution of SCP processing power to all SSPs, in accordance with the network load requirements. *Refer to Arvidsson et al. [1] for a detailed description on CA.*

3.2.1.1 Percent Thinning

Percent Thinning is used to distribute the accepted load evenly over an auction interval, to prevent selling all the available tokens immediately after an auction, during overload situations. This mechanism works by calculating the available tokens over the expected number of requests. This enables us to find out how many percent of the offered requests could be accepted [4].

In the absence of such a mechanism, all the available tokens would be sold to the bidding Allocators immediately after an auction has been held, in an overloaded network. This would result in rejection of all the service requests that would arrive later, before the next auction is held. There is a possibility that these rejected service requests might have maximized the overall profit for the network Service Provider. This may result in a non-optimal working of the whole system for the network owner.

In case of network overload situations where Percent Thinning is not used, a high number of incoming service requests to all SSPs would tend to push the Offered Load over Target Load, thus exceeding it beyond the permissible limits i.e. the total load would exceed 90%.

Another possibility that might occur during overload situations, in the absence of Percent Thinning, is that the SCP nodes would not manage to process all service requests that reach the SCPs concurrently. Immediately, after an auction has been held and the token pools have been refilled. A number of service requests would be accepted for processing by the 'Quantifiers' (on behalf of their corresponding SCPs). The SCPs would thereby process these service requests, one by one. In such an instance, a large number of these accepted service

requests would be queued for a long time before they get processed, maximizing 'response times' of the SCPs.

3.2.2 The Hierarchically Distributed Auction (HA) Architecture

The Hierarchically Distributed Auctions Architecture is *distributed* and *synchronous*. In an HA there exist Intermediate Distributors between a centralized Main Distributor and the Allocators. The Main Distributor would still be *central* to the system, possessing a global *view* of the whole system and making decisions of the same quality as in the CA architecture. However, HA adds the ability of making more and faster decisions at network subset levels. It therefore adds the possibility of a re-balance in load, in a subset of the network. An important fact still holds for the HA, that a re-balance cannot be performed at any other time then at the auctions [4].

The architecture also offers faster adoption of changes in load, since smaller auctions can be performed more often than the central auction. HA distributes the processing required to carry the auctions over several processing nodes, so we could say that smaller auctions are performed more often in an HA. The distributed auctions in an HA could therefore run simultaneously in different parts of the network. Figure 3.2.2 shows a representation of the HA architecture.

3.2.2.1 Percent Thinning

As the distributed auctions in HA are carried out more frequently, as compared to CA. The HA architecture will therefore be less dependent on a good Percent Thinning algorithm over the interval between two auctions during an overload situation, as compared to CA. One observation is noteworthy, as we divide the auction interval in more smaller units in HA, the effect of the Percent Thinning mechanism would be somewhat shifted towards the distribution of auctions, later discussed in the section on 'Analysis'. Which means the more distributed the auctions are in a system, the more they add to the percent thinning mechanism, automatically.

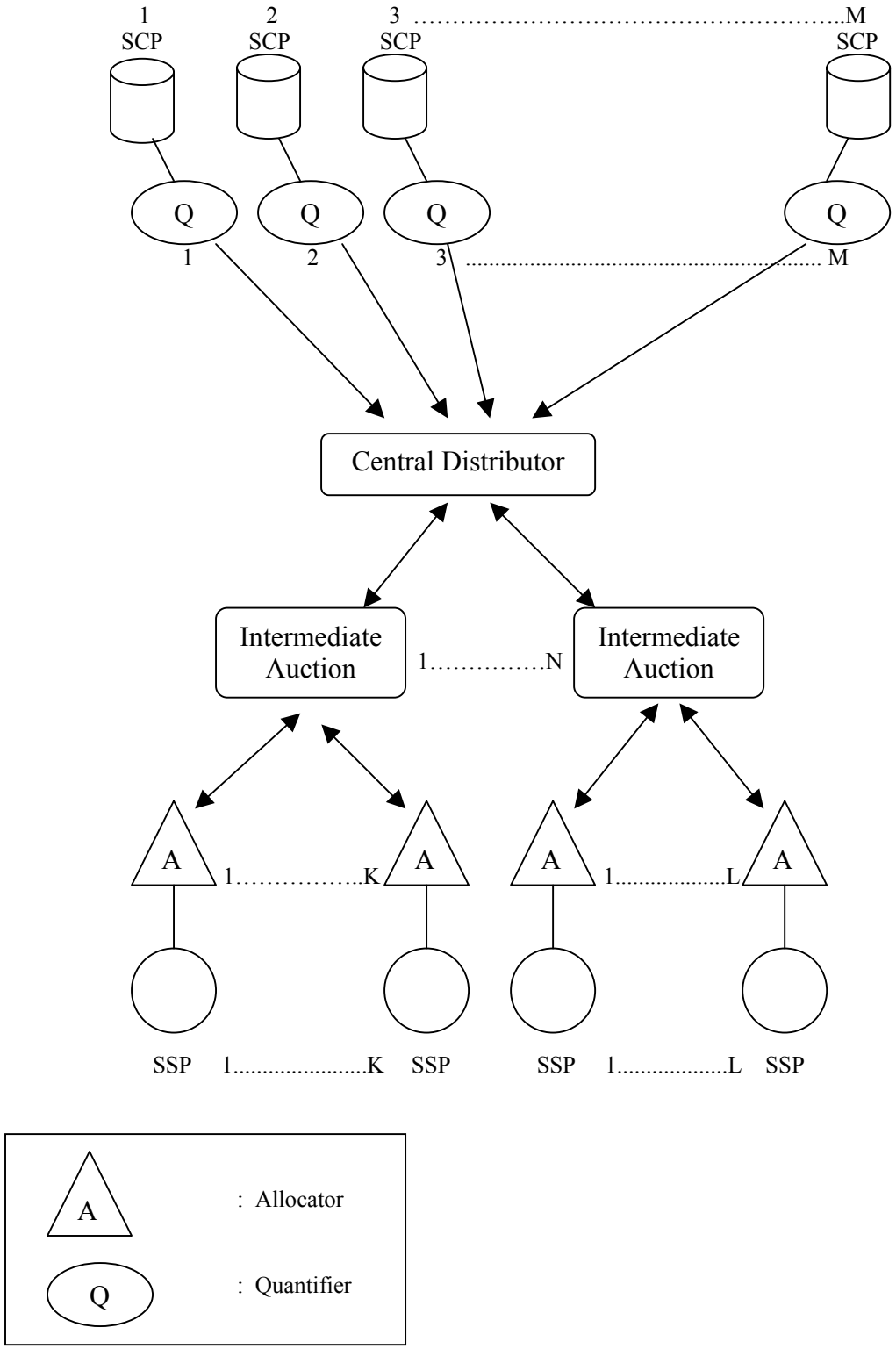


Fig. 3.2.2, The Hierarchically Distributed Auction (HA) Architecture

3.2.3 The Centralized Leaky Bucket (CLB) Architecture

The Centralized Leaky Bucket (or CLB) Architecture is *centralized* and *asynchronous*. The Leaky Bucket Architecture concept has been taken from the metaphor of a leaking bucket,

where the water leaks out from a bucket in a constant torrent. The size of the bucket is therefore a key factor in determining the upper bound on the number of requests that can be accepted. As the CLB architecture is centralized, it possesses the load situation information at all nodes in the entire network. This of course adds a strong basis to reach optimal decisions, after a consideration of the overall network load situation.

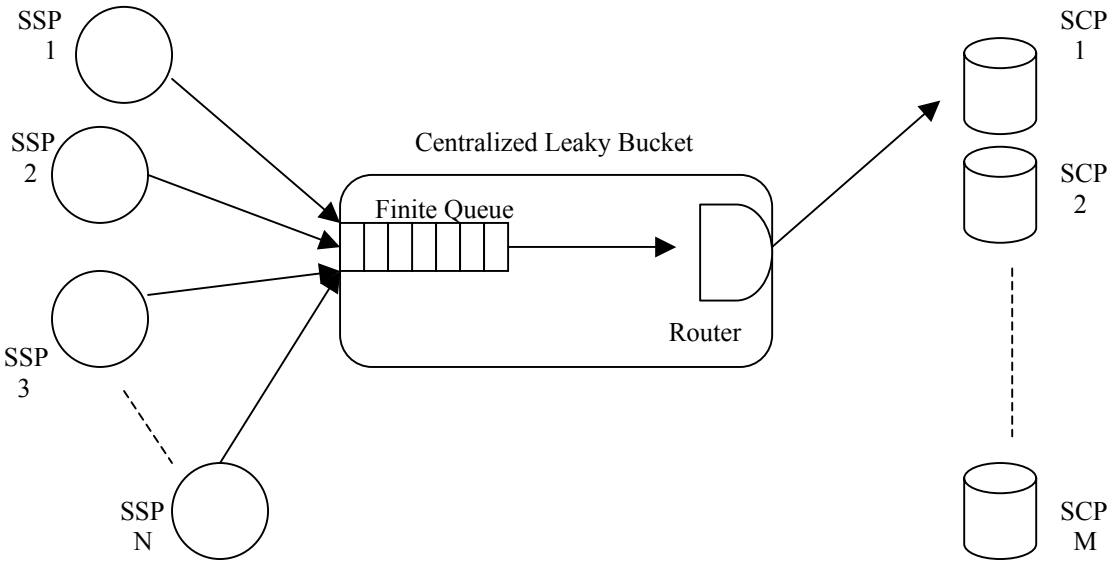


Fig. 3.2.3, The Centralized Leaky Bucket (CLB) Architecture

The Distributor in the CLB is implemented as a finite queue that holds service requests from the SSPs. The Distributor contains a Router, which de-queues these service requests and forwards them to the next available SCP at a rate that corresponds to the total capacity of the service providers. When this finite queue is full and could not accommodate any further service requests, an incoming service request is returned to the Allocator and rejected. Since the time required for accepting a service request during overload situations is proportional to the 'queue length' times 'the time it takes for the distributor to process one request'. Preferably, the size of the queue should be kept keeping in view so that the maximum 'acceptable connection time' for a service request is not exceeded. In the CLB architecture the Percent Thinning mechanism is not needed, since the Distributor in a CLB already handles the thinning operation.

3.2.4 The Mobile Broker (MB) Architecture

The Mobile Broker Architecture is *distributed* and *asynchronous*. In the MB architecture each SCP has a corresponding 'Broker', an agent, which continually travels around through a subset of Allocators, to sell resources on behalf of SCPs. The Broker routes are static and set up, so that each Allocator is visited by two different Brokers (in general). In addition it is designed so that the two Brokers do not arrive at an Allocator, at the same time. A different approach would be to devise *dynamic routes* for the Brokers. This would probably provide a

better means to distribute uneven load, over all available SCPs. However, practical implementation of dynamic routes has not yet been devised and simulated. Figure 3.2.4 represents the Mobile Broker Architecture.

In order to avoid selling all the resources possessed by a Broker to the first few Allocators in its route, during overload situations. A Broker keeps track of the total experienced demand at all Allocators in its route. Each Broker then calculates the percentage of load at current Allocator to determine the resources that could be sold to a particular Allocator, using the simple formula:

Percentage of Load at an Allocator = Demand at current Allocator /
Total Demand for all Allocators in the current route

Figure 3.2.4 represents the Mobile Broker Architecture. In addition to determining the amount of the resources to be sold to an Allocator, this procedure also makes sure that the Broker distributes *all* the processing power of its SCP if possible, to make sure that no processing power remains unused and in turn wasted. A drawback to this approach is that Broker will hand too much bandwidth during the first lap after a sudden increase in the Offered Load (as Brokers operate by handling out the abstract concept of bandwidth).

An additional balancing function that is used in the MB Architecture is that each Allocator tries to relieve pressure from its heaviest loaded Broker and move the requests to the other Broker, in case the Brokers are unevenly loaded. The Allocator calculates this load of a Broker from the quotient between the given request (i.e. the experienced demand) and the received allocation. *Refer to Johansson et al. [3] for details on MB Architecture.*

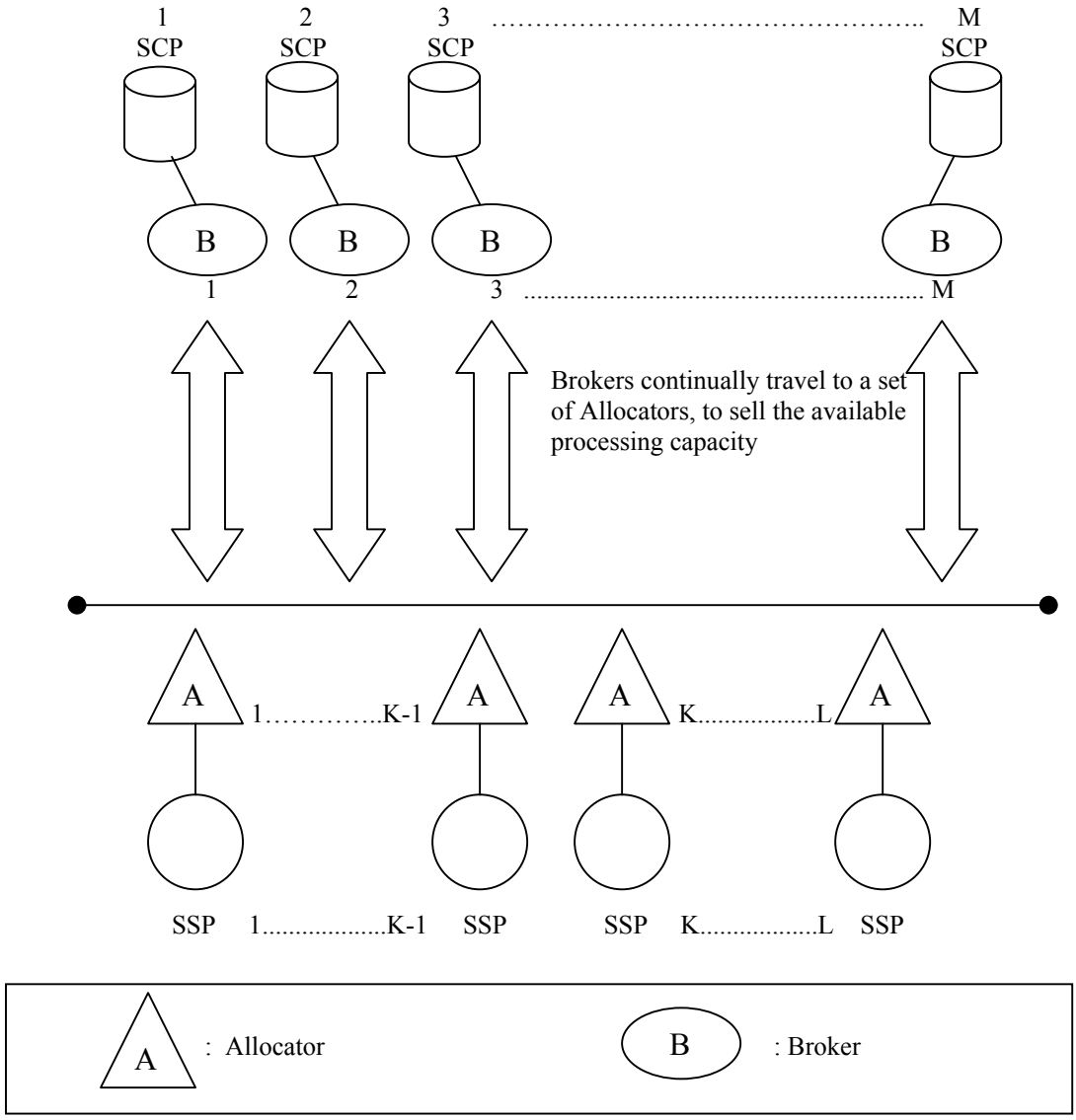


Fig. 3.2.4, The Mobile Broker (MB) Architecture

CHAPTER 4

4 Scalability, The Attributes

Scalability, for a communication network, is the aspect of analysing the *network utilization levels* with an increase in the network size. It determines, *how good the system is at handling an increased number of users (providers and customers)*. Davidsson et al. [12] have proposed that it is possible to evaluate Multi-Agent Systems with respect to several different quality attributes, both different performance related attributes and more general quality attributes, such as, robustness, modifiability and scalability. They have studied the general problem domain of *Dynamic Resource Allocation* with an evaluation of *four Multi Agent architectures* for IN Load Control management, based on the *network evaluation attributes* proposed by them and which are especially suited for this particular problem domain.

In case of an IN, scalability could be determined by analysing the impact on the network performance by the addition of SCPs, SSPs or any other processing hardware/software components. Some of the important parameters to analyse, for a load control architecture, in terms of scalability include utilization of resources, communication delays, Call Accept/ Reject rates etc. when the number of processing nodes and Offered Load is varied. The four Multi-Agent Architectures, which have been explained briefly in the previous chapter, will be examined and compared in terms of the different attributes of scalability. Below is a short explanation of some of the attributes, a few of them originally proposed by Davidsson et al. [12], would be of primary interest in the investigations that follow.

4.1 Utilization of Resources

This attribute examines, the utilization levels of the available resources by the different approaches (architectures) to load control management. Are the available resources utilized as much as possible [12]? The focus would be to analyse, how close to the Target Load the different architectures manage to carry the Offered Load as the network size increases.

4.2 Communication Delays

One of the important attributes of scalability to be determined for a communication network is the *communication delays*. Communication delays could be analysed by examining the following parameters.

4.2.1 Responsiveness

Responsiveness, in the network communication domain means, how long does it take for the customer to get a response from the network for a requested service. An important measure to determine the *Quality of Service (QoS)* from a users' perspective is to determine, how long it takes a service to get connected. It is important to analyse the change in communication delays, when the network size is supposed to increase/decrease.

4.2.2 Request Processing Delays

Another important attribute to study the delay in communication of the entire network is analysing the service request queues at the SCPs. It could be easily perceived that more messages received at the SCPs, the longer the queues of jobs waiting to be processed, and consequently the larger average delay in network response and performance.

4.2.3 Messaging Delays

Message delays in the network could be another important attribute to study for scalability. The more congested a network gets, the more time taken for messaging, the less responsive the network would be to service requests. The aspect to study is the determination of message delay times as the network size grows.

4.3 Call Accept/Reject Rates

The next attribute investigated in this study is the Call Accept/Reject rates at different Offered Loads and network sizes. Since we assume in our studies that the SSPs do not maintain any service request queues. From the perspective of the Network Operator it is important to investigate that does an increase in the network affects the number of calls accepted/rejected by the network. Does the call accept/reject rate remains the same, irrespective of the network size and Offered Load for an architecture. This in turn would help network designers' to resize the network, to gain maximum utilities.

4.4 Communication Overhead

This attribute would help determine the amount of the bandwidth needed for agent interaction during resource allocation. The aspect to investigate would be the bandwidth required during resource allocation as the network is resized, for different architectures.

4.5 Computational Overhead

The computational overhead refers to the computational complexity involved for the resource allocation mechanism alone, in the different architectures. In terms of scalability, it is important to determine the affect on the computational complexity of an architecture as the network grows in size.

4.6 Load Balancing

It is also important to investigate the load distribution at the SCPs by the different architectures, as we increase the network size and at different Offered Loads. It would be interesting to see, *how evenly is the load balanced between the resource providers* [12].

4.7 Reactivity

An important attribute to consider for a load control management architecture is, *how fast are the resources re-allocated when there are changes in demand* [12]. How fast can it adapt to changes in the Offered Loads as the network size grows.

CHAPTER 5

5 Simulation Preconditions & Settings

5.1 Simulation Preconditions

This following section discusses some general network settings and simulator configuration for the experiments, which have been performed in this study.

5.1.1 General Network Configuration

The basic network settings used in the simulations performed have an IN configuration of 8 SCPs, whereas the number of SSP is varied for different experiments i.e. 32, 64 and 128. The SCPs and SSPs are assumed to be statistically identical respectively. All SCPs are identical in respect of the processing capacity and software configuration i.e. they provide the same set of the services. Further, all SCPs are equally reachable by any SSP. The SSPs and SCPs communicate via a SS7² signalling network cloud. The network supports two service classes,

Service A, Virtual Private Network
Service B, Ring Back

The call holding times are negative exponentially distributed with a mean of 100 seconds. Service Requests arrive according to independent Poisson processes. The unit for load measurement used in the observations is *Erlang*. *Erlang* is defined as *'the load relative to the capacity to carry it'* [4]. In the simulations, the arrival rate is stated in terms of the Offered Load, between 0 and 2.0 Erlangs i.e. between no traffic and severe overload. It is assumed that a small part of the available bandwidth of this network is reserved for the load control mechanism, in this case the agent communication and transportation. It is assumed that all SCPs support the same set of the service classes and that all service requests can be directed by a SSP to any SCP. For the sake of simplicity, it is further assumed that no messages are lost, network resources do not overload and the resources cannot be buffered i.e. they have to be consumed immediately. The cost of communication (and transportation of resources)

between any customer-provider pair is supposed to be equal. *Target Load* corresponds to 0.9 Erlangs. It refers to the desired maximum allocated network bandwidth available for resource allocation.

5.1.2 Prediction of the Offered Load

Two methods of predicting the offered load for next interval of time, had been considered for implementation. A simpler approach is to assume that the Offered Load during the next interval will be the same as in the last interval. A more advanced approach to predict the offered load would be to apply some 'prediction algorithm' based on statistical methods. Once this algorithm performs well and could forecast acceptable trends for the anticipated load. An algorithm to analyse and learn from the predicted trends (e.g. a Machine Learning algorithm) might be integrated to the previously implemented prediction algorithm. However, this approach is more complex to implement and would require more computational resources. Therefore, the simpler approach is used in the simulations.

5.1.3 Architecture Specific Configurations

Below is a description of some general, architecture specific, settings for the simulations. However, any additional settings to these, for a specific experiment have been mentioned respectively. One thing noteworthy is that in the current working of the simulator, the auctions in the CA and HA Architectures currently takes zero simulated seconds to carry.

5.1.3.1 Centralized Auction Architecture

The main auction interval for the CA architecture is 10 seconds, regardless of the number of the SSPs.

5.1.3.2 Hierarchically Distributed Auction Architecture

Regardless of the number of the SSPs, the main auction interval for the HA architecture is 10 seconds and the intermediate auction interval is set to 3.0 seconds, which implies that there would be three intermediate auctions for every main auction.

When the number of SSP is set to 32, Allocator 1-8 is partitioned to intermediate auction one, Allocator 9-16 is partitioned to intermediate auction two, Allocator 17-24 is partitioned to intermediate auction three and Allocator 25-32 is partitioned to intermediate auction four.

When the number of SSP is set to 64, Allocator 1-16 is partitioned to intermediate auction one, Allocator 17-32 is partitioned to intermediate auction two, Allocator 33-48 is partitioned to intermediate auction three and Allocator 49-64 is partitioned to intermediate auction four.

When the number of SSP is set to 128, Allocator 1-32 is partitioned to intermediate auction one, Allocator 33-64 is partitioned to intermediate auction two, Allocator 65-96 is partitioned to intermediate auction three and Allocator 97-128 is partitioned to intermediate auction four.

5.1.3.3 Centralized Leaky Bucket Architecture

The size of the CLB finite queue is set to 80, for the simulations performed, regardless of the number of the SSPs.

5.1.3.4 Mobile Broker Architecture

When the number of SSPs is set to 32, the Broker routs are set up so that each Allocator is at least visited by two Brokers and each Broker route comprehend eight Allocators, where every Broker route crosses every other Broker route at least once in a lap.

When the number of SSPs is set to 64, each Allocator is at least visited by two Brokers and each Broker route comprehends sixteen Allocators, where every Broker route crosses every other Broker route at least twice in a lap.

When the number of SSP is set to 128, each Allocator is at least visited by two Brokers and each Broker route comprehends thirty-two Allocators, where every Broker route crosses every other Broker route at least four times in a lap.

The time that a Broker spends at each Allocator is 0.2 seconds.

5.2 Tabulation of Results

Results have been determined based on the simulations, which were 20 simulated minutes long. The measurements were taken for the last 10 minutes, since the system needs 5-7 minutes to stabilize (the call 'connection' and 'disconnection' phase gets stable) after a start-up. So a delay of a total of ten minutes would be appropriate for the system to stabilize. Results have been based on the values recorded in the last 600 seconds of a simulation and the Mathematical Mean was then calculated based on these values (at different Offered Loads) to determine the behaviours of various trends of data.

5.3 Simulation Runs

The different sets of simulation settings used for this analysis are described below,

Simulation 1 - Six different sets of simulations were run on all architectures (except the cases where some specific observations needed special executions, which are mentioned accordingly) with a constant arrival rate at all SSPs corresponding to an aggregated Offered SCP Load of 0.35, 0.70, 0.95, 1.05, 1.50 and 2.0 Erlangs respectively. The number of SCP was set to 8, whereas the number of SSPs was set to 32.

Simulation 2 - Six different sets of simulations were run on all architectures (except the cases where some specific observations needed special executions, which are mentioned accordingly) with a constant arrival rate at all SSPs corresponding to an aggregated Offered SCP Load of 0.35, 0.70, 0.95, 1.05, 1.50 and 2.0 Erlangs respectively. The number of SCP was set to 8, whereas the number of SSPs was set to 64.

Simulation 3 - Six different sets of simulations were run on all architectures (except the cases where some specific observations needed special executions, which are mentioned accordingly) with a constant arrival rate at all SSPs corresponding to an aggregated Offered SCP Load of 0.35, 0.70, 0.95, 1.05, 1.50 and 2.0 Erlangs respectively. The number of SCP was set to 8, whereas the number of SSPs was set to 128.

Simulation 4 – A constant arrival rate at all SSPs corresponding to an aggregated Offered SCP Load of 0.35 Erlangs was applied initially. At 400th second, the load was instantly increased from 0.35 to 2.0 Erlangs on all SSPs. At the 800th second the Offered Load was dropped back to 0.35 Erlangs in an instant, on all SSPs. The number of SCP was set to 8, whereas the number of SSPs was set to 32.

Simulation 5 – A constant arrival rate at all SSPs corresponding to an aggregated Offered SCP Load of 0.35 Erlangs was applied initially. At 400th second, the load was instantly increased from 0.35 to 2.0 Erlangs on all SSPs. At the 800th second the Offered Load was dropped back to 0.35 Erlangs in an instant, on all SSPs. The number of SCP was set to 8, whereas the number of SSPs was set to 64.

Simulation 6 – A constant arrival rate at all SSPs corresponding to an aggregated Offered SCP Load of 0.35 Erlangs was applied initially. At 400th second, the load was instantly increased from 0.35 to 2.0 Erlangs on all SSPs. At the 800th second the Offered Load was dropped back to 0.35 Erlangs in an instant, on all SSPs. The number of SCP was set to 8, whereas the number of SSPs was set to 128.

CHAPTER 6

6 *The Analysis*

6.1 Utilization of Resources

One of the important observations to start with, is to analyse how close to the Target Load the different architectures manage to carry the Offered Load as we resize the network (especially as the number of SSPs is increased). This would provide a basis to analyse the upcoming results in the following sections. It would help compare the performance of different architectures in carrying various Offered Loads and would allow us to determine the effects on Carried Load with an increase in the network size. This would however help develop a pre-notion for upcoming conclusions. Figures 6.1.1, 6.1.3 and 6.1.5 depict the results on the load carried by the four architectures at various Offered Loads and number of SSPs. The values for the Carried Load have been recorded at every second of a simulation. The mathematical mean is then taken to determine a mean value for the 'Carried Load' at a particular Offered Load.

Figure 6.1.1 present the results from Simulation-1. The results reveal that, in general, the CLB and CA architectures manage to carry the load, quite close to the Target Load as compared to the HA and the MB architectures, as we increase the Offered Load from 0.35 Erlangs to 2.0 Erlangs. The deviations are largest just when the Offered Load is equivalent to the Target Load. At this instant, the MB and HA architectures do not perform better than CA or CLB. However, as the Offered Load is further increased, the Carried Load by the MB architecture drifts toward the Target Load and performs no less than CA and CLB. HA on the other hand, could not carry the load better than 0.86 Erlangs, at Offered Loads beyond Target Load.

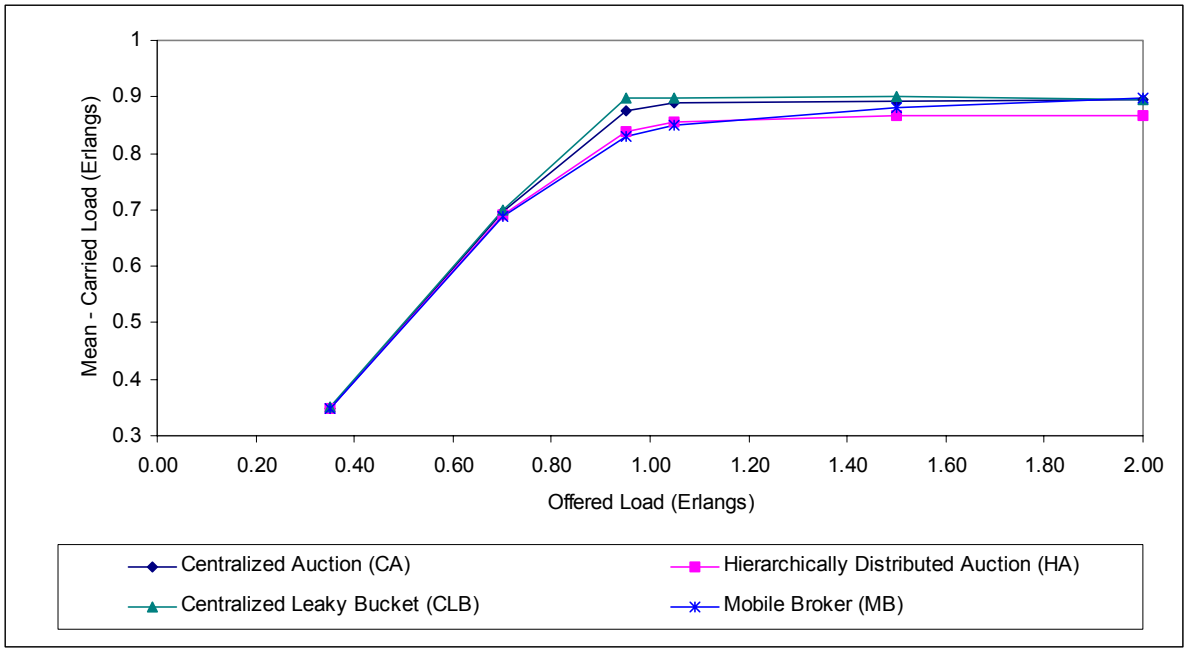


Fig. 6.1.1, Simulation-1 (32-SSPs), The 'Mean of Carried Load' at different 'Offered Loads'

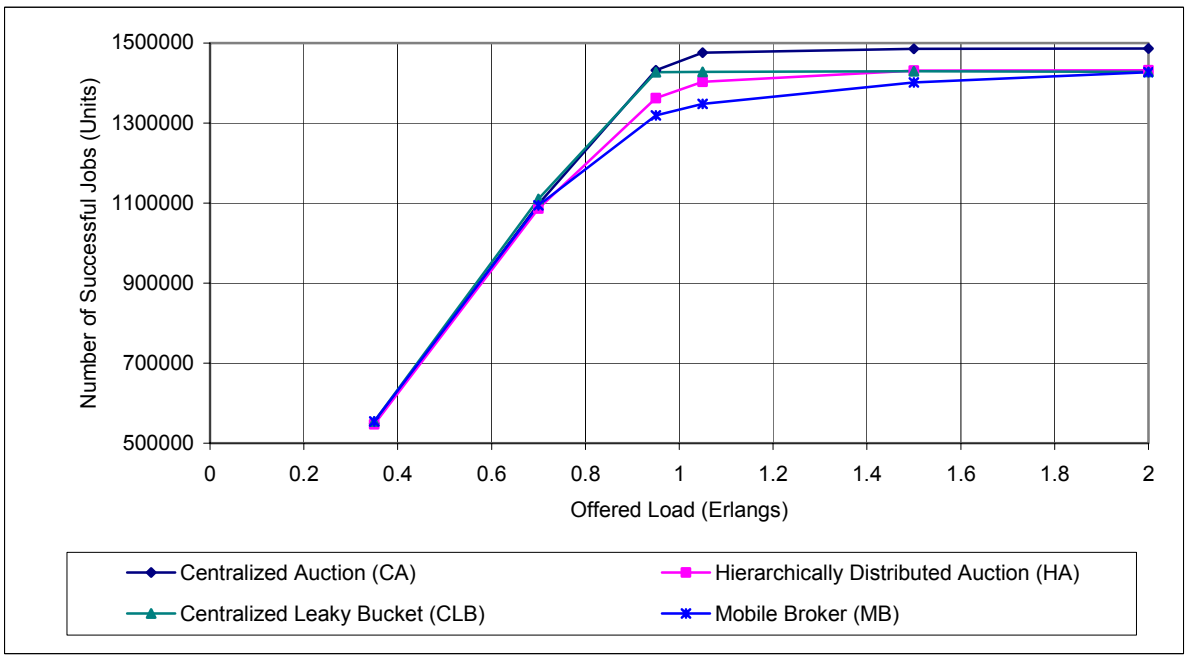


Fig. 6.1.2, Simulation-1 (32-SSPs), Analyzing the total 'Number of Successful Jobs' at different 'Offered Loads'

Though the CLB architecture manages to carry the Carried Load *closest* to the Target Load at all Offered Loads, yet the number of Successful Jobs[9] by this architecture aren't the largest (see figure 6.1.2), once the Offered Load goes beyond 0.9 Erlangs (Target Load). Also note that HA possesses as much Successful Jobs as CLB, though it fails to carry loads close enough to the Target Load.

Figure 6.1.3 shows results from Simulation-2. When the Offered Load is equal to the Target Load, both MB and HA carry load at approximately 0.79 Erlangs, which is significantly less than CA and CLB. On the other hand, when the Offered Load is 0.9 Erlangs, both CA and CLB carry the load closest to the Target Load at 0.88 and 0.89 Erlangs respectively. CLB however carries the Offered Load closest to the Target Load as compared to all the other architectures and at all Offered Loads.

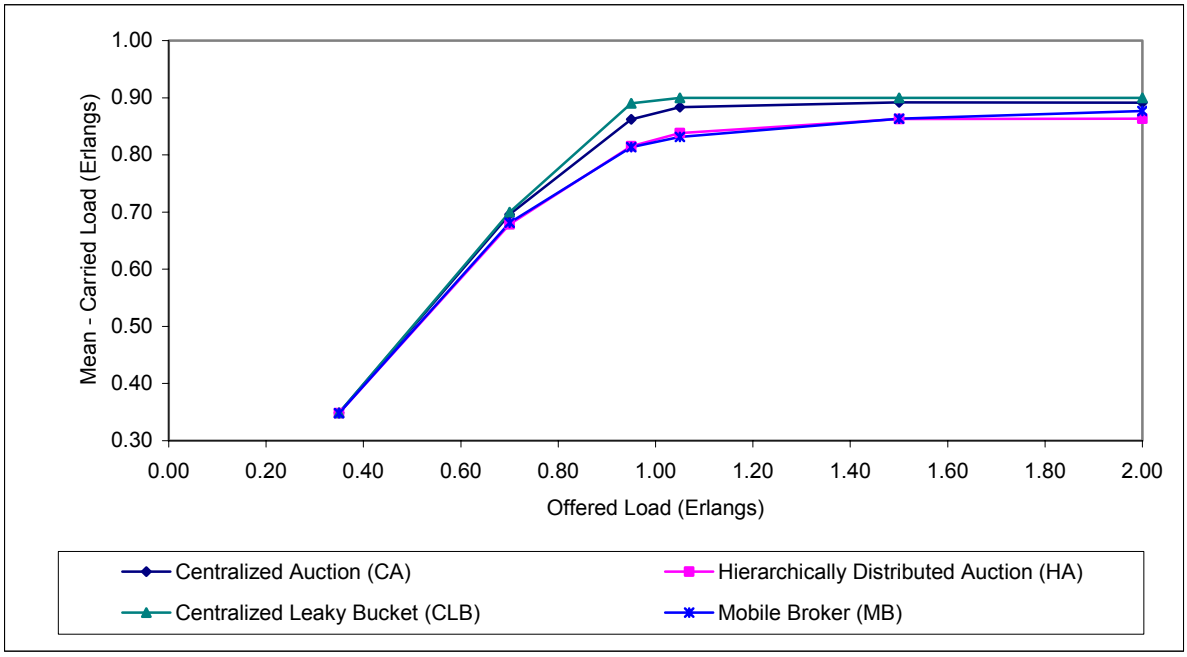


Fig. 6.1.3, Simulation-2 (64 SSPs), The 'Mean of Carried Load' at different 'Offered Loads'

It can be noted that, again CA and CLB carry the load closest to the Target Load at all Offered Loads. Though CLB has a greater rate of Successful Jobs than CA, before the Offered Load exceeds the Target Load, see figure 6.1.4. Yet as soon as the Offered Load goes beyond the Target Load, CA produces the highest number of Successful Jobs. Although HA does not perform better in carrying the Offered Load close to the Target Load as compared to CLB yet, generally, the number of Successful Jobs by the HA is somewhat the same as in CLB in this case (at higher Offered Loads), as can been seen from figure 6.1.4. The MB architecture has

[9] Successful Jobs, in general, refers to a Service Request Call, which is first generated, then accepted by an SSP so that the requested resources could be allocated to it. This service request *may* then be allocated the requested resources and connected to the SCP/s. After being served, it is successfully disconnected. On the other hand, some of the service request calls, though being accepted by an SSP would still not get a chance to get the requested resources, referred to as *Unsuccessful Jobs*.

the least number of Job Success Rate compared to other architectures after the Offered Load goes beyond 0.70 Erlangs.

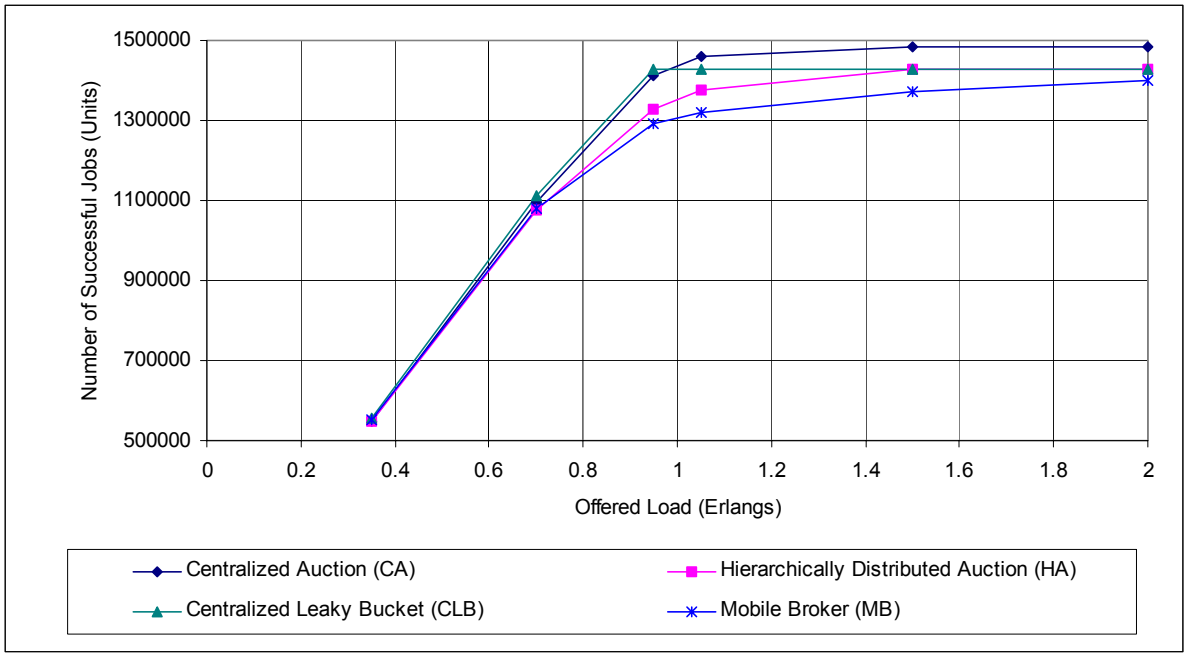


Fig. 6.1.4, Simulation-2 (64 SSPs), Analyzing the total 'Number of Successful Jobs' at different 'Offered Loads'

Figure 6.1.5 shows results from Simulation-3. Note the load carried by MB and HA when the Offered Load is 0.9 Erlangs, which is only around 0.76 Erlangs. HA manages to carry load close to the Target Load at higher Offered Loads (Offered Loads greater 0.95 Erlangs) as compared to MB, in contrast to the results from Simulation 1 and 2.

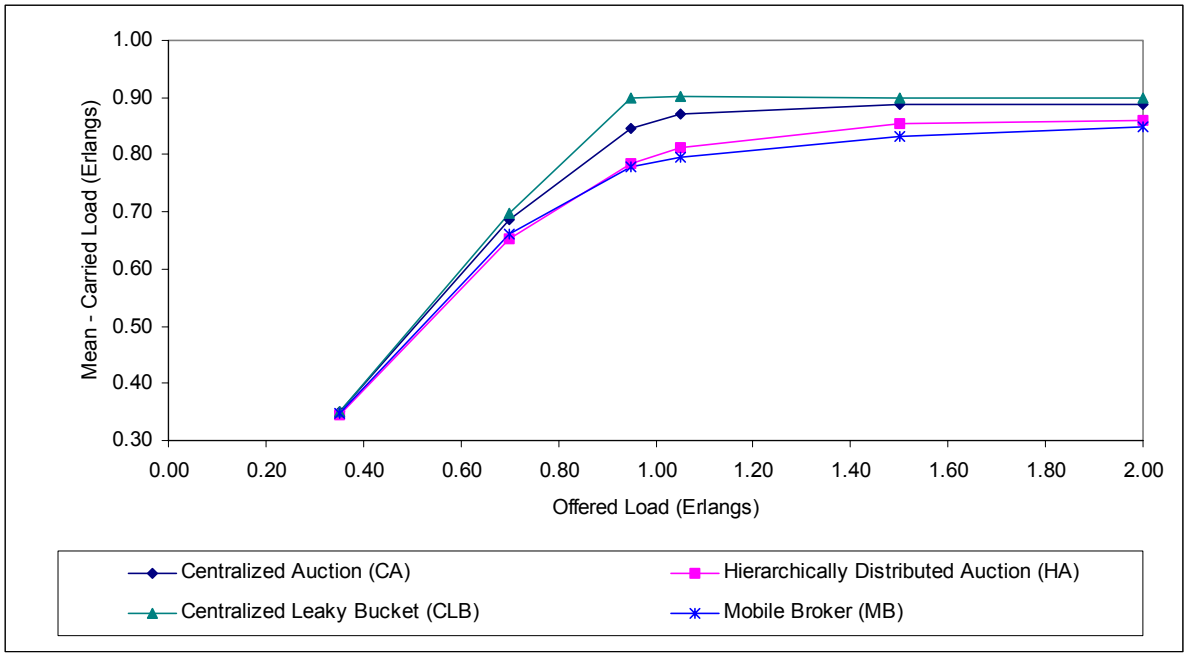


Fig. 6.1.5, Simulation-3 (128 SSPs), The 'Mean of Carried Load' at different 'Offered Loads'

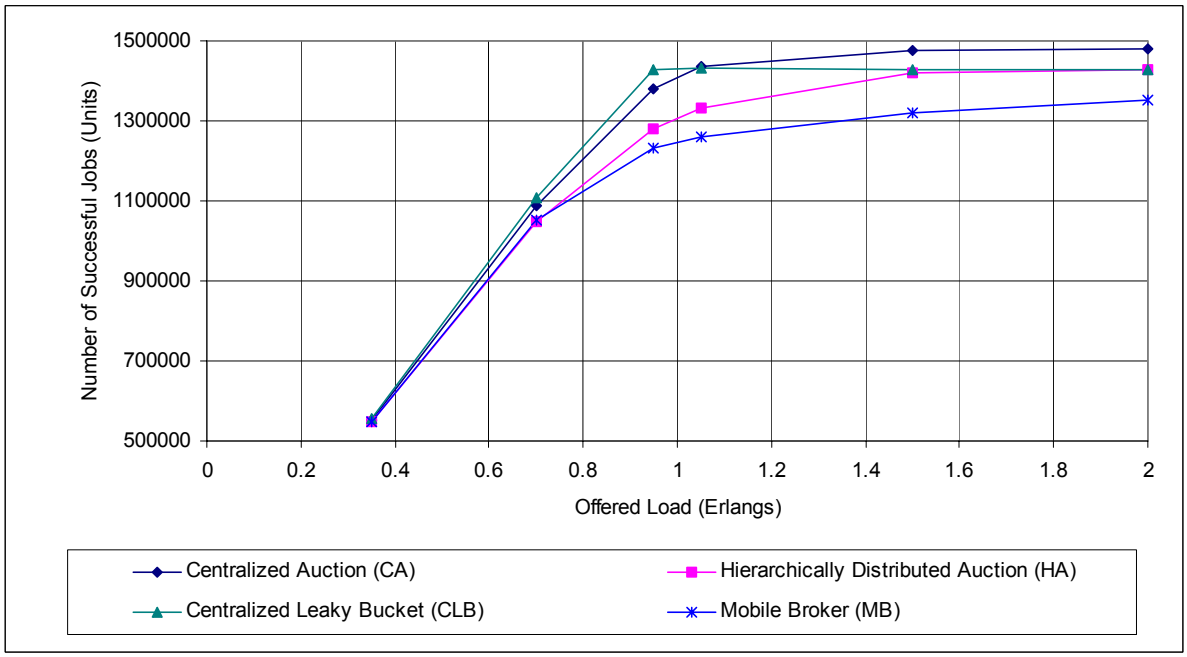


Fig. 6.1.6, Simulation-3 (128 SSPs), Analyzing the total 'Number of Successful Jobs' at different 'Offered Loads'

CLB, again manages to produce the highest number of Successful Jobs before the Offered Load exceeds 1.05 Erlangs. Yet as soon as the Offered Load goes beyond 1.05 Erlangs CA produces the highest number of Successful Jobs. MB again, is the worst as compared to other architectures with respect to the number of Successful Jobs.

Figures 6.1.7, 6.1.8, 6.1.9 and 6.1.10 present the Carried Loads at different Offered Loads and number of SSPs by individual architectures for simulations 1, 2 and 3.

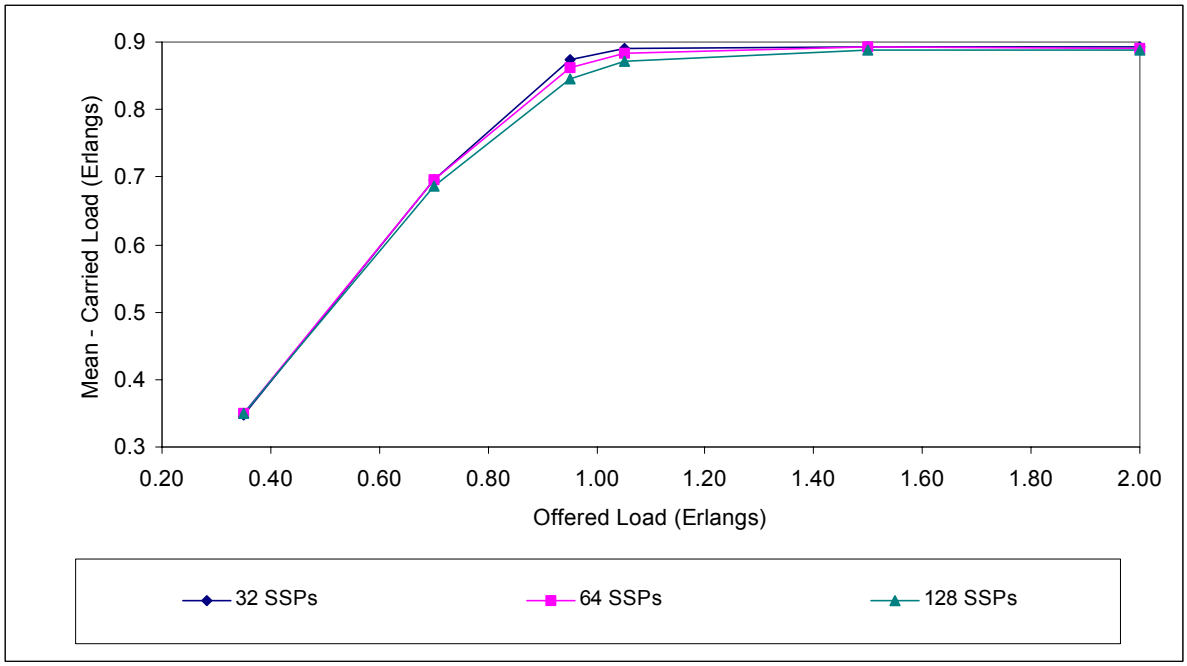


Fig. 6.1.7, Simulation 1-2-3, The 'Mean of Carried Load' at different 'Offered Loads' for Centralized Auction Architecture

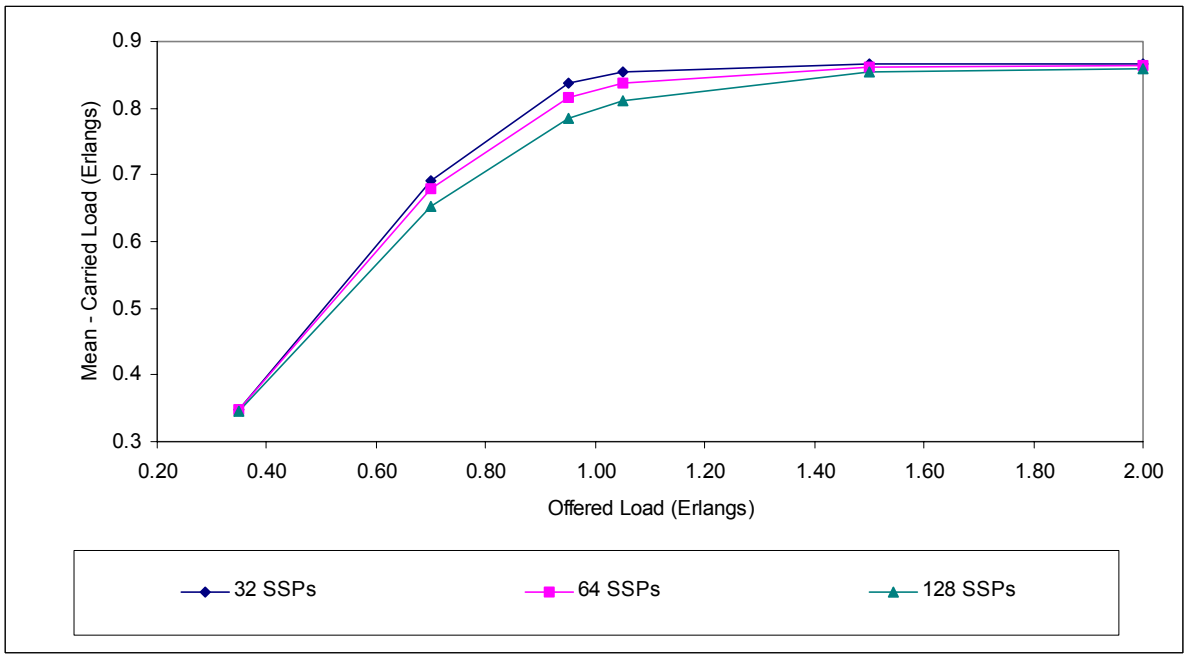


Fig. 6.1.8, Simulation 1-2-3, The 'Mean of Carried Load' at different 'Offered Loads' for Hierarchically Distributed Auction Architecture

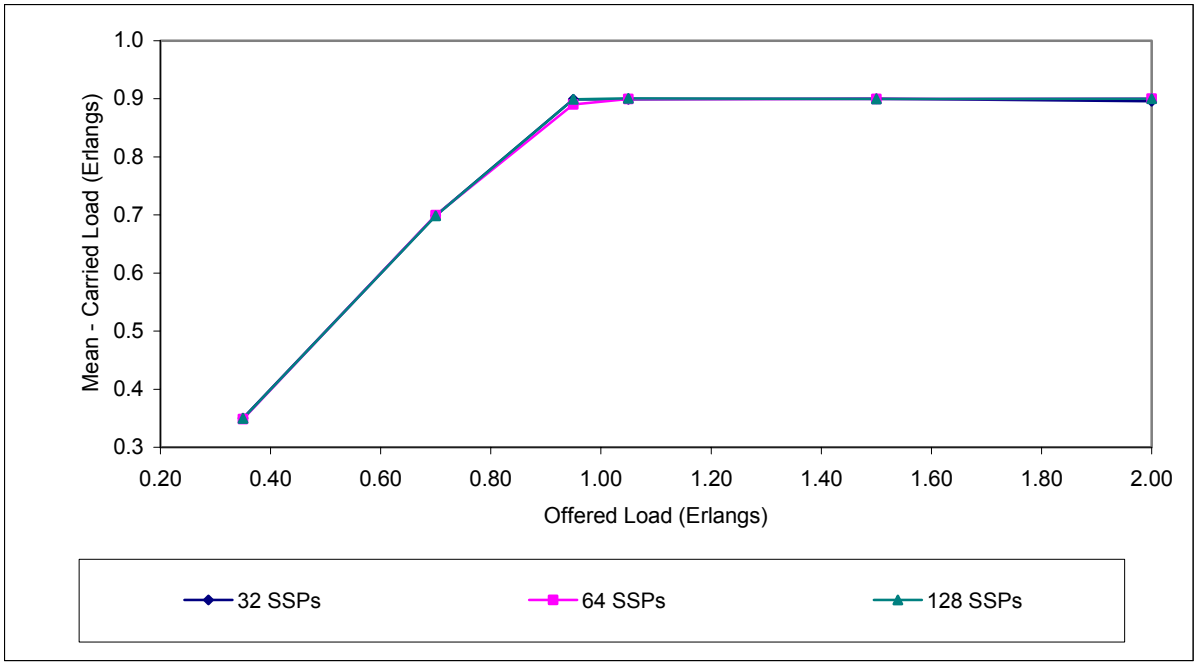


Fig. 6.1.9, Simulation 1-2-3, The 'Mean of Carried Load' at different 'Offered Loads' for Centralized Leaky Bucket Architecture

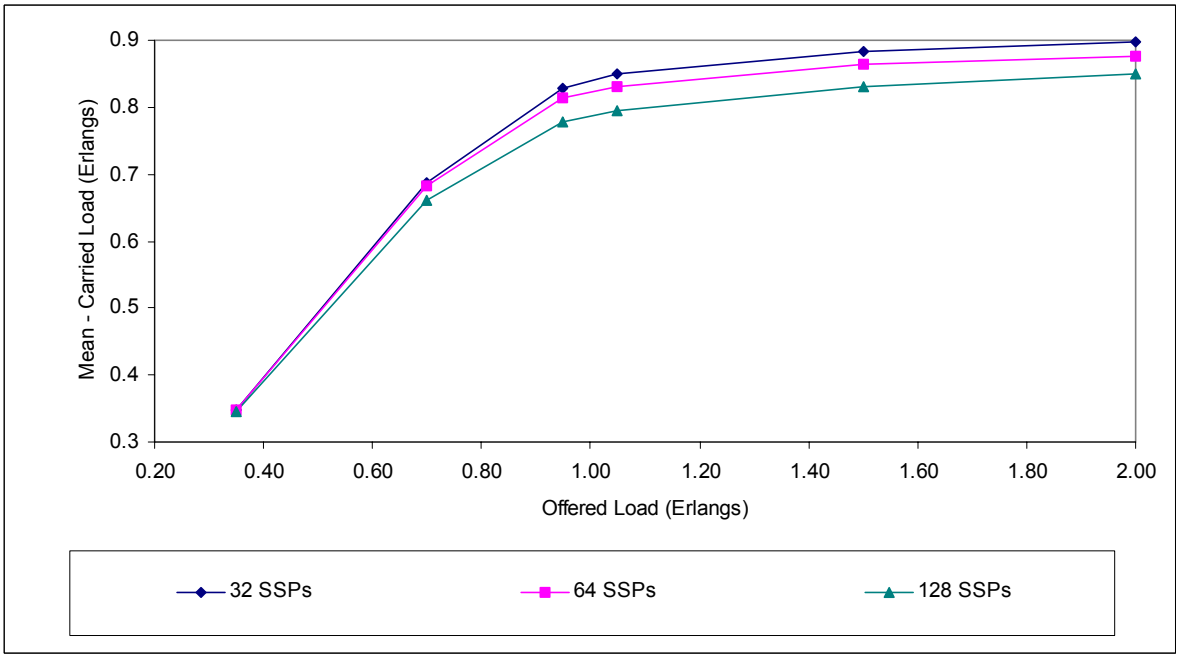


Fig. 6.1.10, Simulation 1-2-3, The 'Mean of Carried Load' at different 'Offered Loads' for Mobile Broker Architecture

In light of the results presented before, it can be noted that the Centralized Leaky Bucket architecture manages to keep the Carried Load closest to the Target Load, regardless of the increase in number of the network nodes (SSPs) and the Offered Load. The major factors

adding to its better performance in regard to carrying loads close to Target Load is its typical work mechanism, where requests are *continually* sent to the Central Distributor from all the Allocators. The Central Distributor possess a global overview of the entire network situation, which helps in making optimal decisions for resource allocation. The Distributor only adds a service request to the finite central queue if there is at least one slot available, with a penalty that service requests are simply rejected if the finite queue is full and in turn produces lesser Successful Jobs when the Offered Load exceeds the Target Load. Still, at Offered Loads less than Target Load, it manages to produce the highest number of Successful Jobs in comparison with other architectures.

CA architecture however performs less optimal than CLB, in carrying loads closer to the Target Load. It manages to produce most Successful Jobs than all the other architectures at different network sizes, especially when the Offered Load is greater than the Target Load. It is because of two reasons, first the CA architecture is 'synchronous' and performs allocations only when the auctions are held, unlike CLB, which results in a significantly less Carried Load as the network size grows due to this synchronized work mechanism. Secondly, it suffers from the phenomenon that shows up at the end of an auction interval when an arbitrary SSP is out of tokens of a service type (lets say, service one), just before the next auction [13]. Unfortunately, all the new incoming calls, request service type one. Though SSPs have enough service type two tokens, yet they could not accept the new incoming calls. So simply, these service type two tokens remain unused and ultimately have to be discarded. A considerable number of unused tokens in the CA are therefore discarded at the end of an auction interval, due to the occurrence of the above phenomenon, which certainly effects its performance to carry loads closer to the Target Load. More the auctions in the CA, more would be the number of tokens discarded. *The number of discarded tokens is therefore proportional to the number of auctions held.*

To better utilize the available tokens and enhance the overall performance of the CA architecture, a mechanism could be devised which should decide upon scheduling the next auction. A ratio between the available tokens of a particular service type and its incoming service requests should be calculated continuously during execution by the Main Distributor. The next auction, should not be scheduled until a considerable number of tokens for a significant number of incoming service requests are available, for that particular service type tokens. Once this ratio drops below a certain (optimal) threshold, the next auction could then be carried out. In order to avoid delays in cases where the Distributor needs to notify all Allocators about this, and wait for all of them to produce new bids. As we know that the above ratio would be decreasing, so before it reaches that 'optimal' threshold. Considerably before the current auction interval ends, the Allocators could be informed in advance to produce new bids. However, this idea needs to be implemented and validated for its performance.

The reason for HA of not carrying the Offered Load close to the Target Load is due to its distributed nature. In HA, auctions are performed more often and could occur simultaneously in different parts of the network. Due to this increased number of main and intermediate auctions, the percent thinning mechanism in the HA is partly handled by these distributed auctions. The HA architecture is therefore less dependant on an efficient percent-thinning mechanism. But at higher Offered Loads, performing auctions (especially Intermediate Auctions) too frequently results in lesser distribution of tokens over the entire auction interval. A large amount of tokens remain unused, because the auction interval might be small and all the tokens bought by the Allocators might not be utilized, in a small time interval. This

would result in wastage of the SCP processing capacity. HA therefore suffers from the above limitations, as the number of network nodes grow. Despite the fact that HA carries the least Carried Load, it is considerably better in the number of total Successful Jobs produced. In later sections on Responsiveness and Message Delays, it is observed that HA performs relatively better in offering better Response Times and produces lesser delays in communications (messages) at different Offered Loads.

The MB architecture experiences a decrease in Carried Load close to the Target Load, as the network size grows (as clear from figure 6.1.10). The MB architecture, being synchronous, is clearly *more* reactive to an increase in the number of SSPs, which means a longer Broker route and thus more communication overheads. As discussed in the simulation configuration settings (for MB) in the previous chapter. When the number of SSP is set to 64, each Allocator is at least visited by two Brokers and each Broker route comprehend sixteen Allocators, where every Broker route crosses every other Broker route twice. However when the number of SSP is set to 128, each Allocator is at least visited by two Brokers and each Broker route comprehend thirty-two Allocators, where every Broker route crosses every other Broker route four times in a lap. This increase in the route length for every Broker causes delays in request processing, this can be observed by the observations in section on Request Processing Delays (see figures 6.2.4, 6.2.6, 6.2.8 and 6.2.13), which clearly states that MB suffers from relatively large Queues of service requests at SCPs, as compared to other architectures, as the network size increase. Which results in its comparatively bad performance, in terms of carrying load close to the Target Load.

A general conclusion from the analysis of results presented in this section (especially from figures 6.1.7, 6.1.8, 6.1.9 and 6.1.10) is that *Centralized Architectures (CA and CLB), perform better than Distributed Architectures (HA and MB), with an increase in the network size and at differed Offered Loads*. The main reason for the centralized architectures to perform well, in carrying Offered Loads close to Target Load, as compared to the distributed architectures is their ‘centralized’ and ‘well-informed’ decision-making mechanism at the Distributors. So as the network size increases, the decision-making mechanism with a global overview (keeping in view of the overall increase in network nodes) results in better performance than the distributed architectures. The distributed architectures relatively perform decision-making based on their related subset of the network. This means that their results are less optimal globally, but comparatively more optimal locally.

6.2 Communication Delays

The next attribute to be analysed is the communication delays in the network, as the network is resized. While studying the scalability of a network, analysing the delays in communication is important. It is critical to determine the internal messaging within the network. Since on one hand, this would provide useful information for a network provider, to perceive the entire network bandwidth requirements with the growth in the network. And on the other hand, would help determine the delays in request processing with a resize in the network nodes. Three parameters have been discussed in subsequent sections on Responsiveness, request Processing Delays and Messaging Delays.

6.2.1 Responsiveness

Responsiveness of a network depends heavily on its load situation. In case of IN, a heavily loaded network would mean larger service request queues at the SCPs and an immense

number of incoming service requests at the SSPs, consequently overloading the network resulting in greater Response Times. A set of the simulations was performed to study the trend of Response Times, at varied Offered Loads and number of SSPs. Analysing the 'Connection Response Times' at different Offered Loads in all the architectures is a possible yardstick that could help observe the communication delays for our particular investigations on Communication Delays and in general, the scalability.

The 'Cumulative Response Time' had been recorded at every second of a simulation. The mathematical mean of these values is then taken to determine the 'Mean Cumulative Response Time' at a single Offered Load.

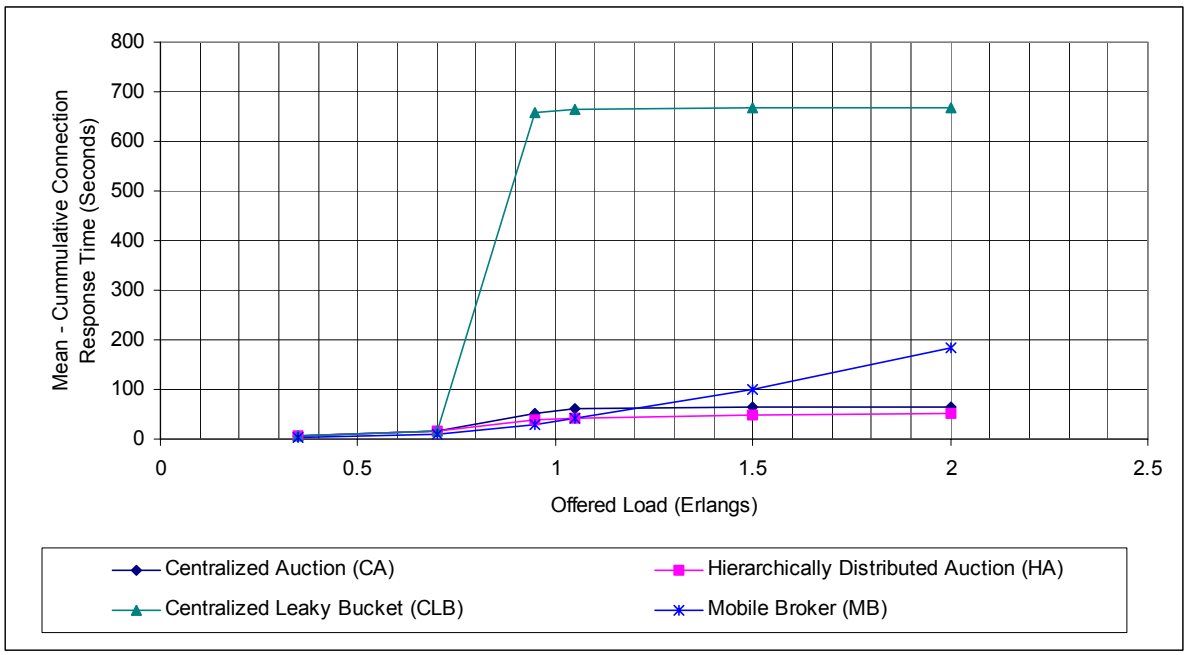


Fig. 6.2.1, Simulation-1 (32-SSPs), The 'Mean of Cumulated Response Time' at different 'Offered Loads'

Figure 6.2.1, shows a representation of Simulation-1 results, when the number of SSP was set to 32. As obvious from the figure, the Response Time for CA, HA and MB architectures is somewhat in close range at different Offered Loads, before the Offered Load exceeds the Target Load. Actually, the Response Times for the CA and HA is almost similar at all Offered Loads It can also be observed that when the Offered Load exceeds the Target Load, HA has the least Response Time.

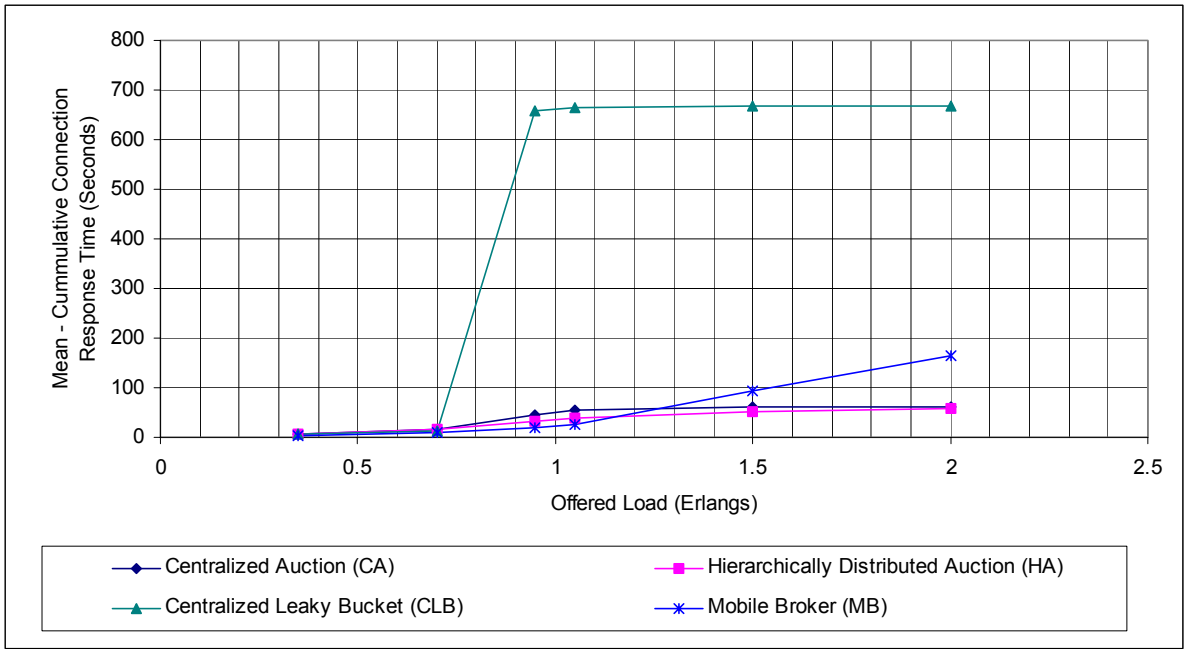


Fig. 6.2.2, Simulation-2 (64 SSPs), The 'Mean of Cumulated Response Time' at different 'Offered Loads'

Figure 6.2.2, shows a representation of results from Simulation-2. The maximum Mean Cumulated Response Time for the MB architecture is 163 seconds, when the Offered Load is 2.0 Erlangs (which is approx. 20 seconds less as compared to the case where SSP=32). In this case, the Cumulated Response Time for the CLB seems to be the worst, after the Offered Load exceeds 1.15 Erlangs, as compared to other architectures (exactly similar to the corresponding result from Simulation-1).

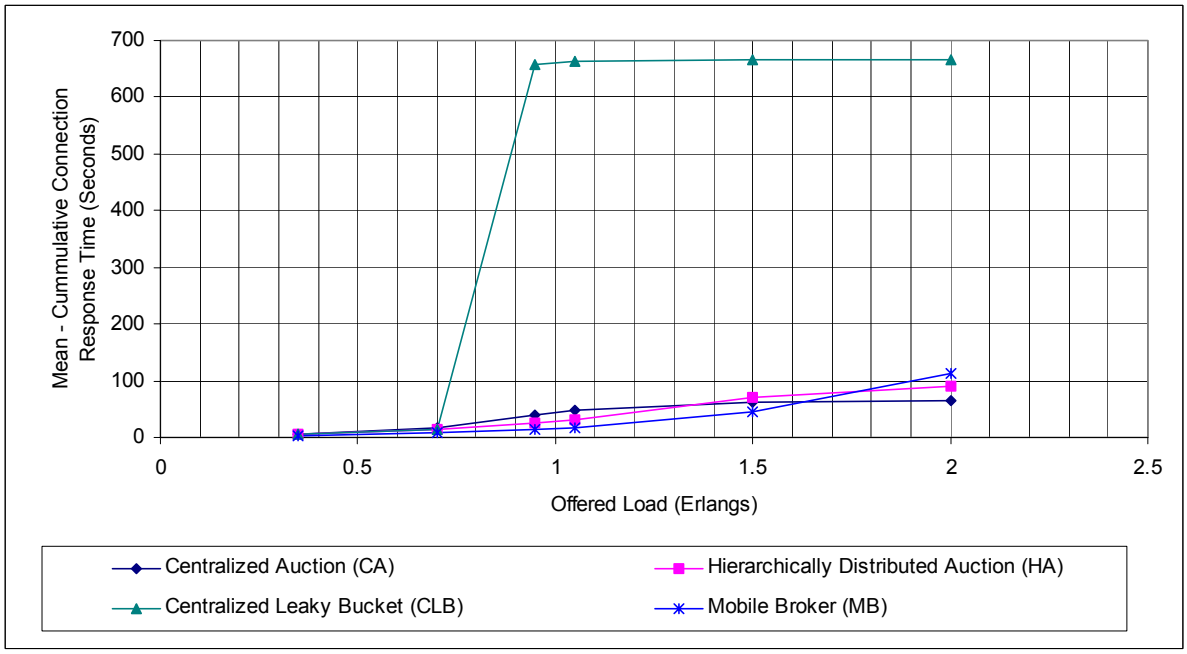


Fig. 6.2.3, Simulation-3 (128 SSPs), The Mean of 'Cumulated Response Time' at different 'Offered Loads'

Figure 6.2.3, shows a representation of the simulation results when the number of SSP was increased to 128 (Simulation 3). The Cumulated Response Time for the MB architecture is the least in this experiment, until the Offered Load exceeds 1.60 Erlangs. Refer to figure 6.2.13 and note the drop in the Queue Length at SCPs in MB for the same simulation, which is one reason why MB offers a better Response Time. Again, the Cumulated Response Time for the CLB seems to be the worst, after the Offered Load exceeds 0.7 Erlangs, compared to other architectures (exactly similar to the findings of corresponding result from Simulations 1 and 2).

The observations presented above based on results shown in figures 6.2.1, 6.2.2 and 6.2.3 generally state that the auction architectures perform well than CLB and MB, in respect of Response Times as the Offered Load and the number of SSPs are increased. CA and HA architectures which do not seem to be better than CLB in carrying Offered Loads close to the Target Loads for the same simulation settings (see figure 6.1.1, 6.1.3 and 6.1.5), has however considerably less Response Times than CLB. One of the reasons for their better performance is the 'synchronization' in these architectures. Since main auctions (in case of both CA and HA) and the intermediate auctions (in case of HA) are performed at specific time intervals. Therefore communication among network components occur at specific times (first, when the bids are sent in, just before an auction. Second, just after each auction, when the results are broadcasted to the Allocators). This results in less network traffic (messaging), avoiding network congestion and results in better Response Times. As we have increased the number of SSPs, this change in size hasn't affected the performance of these architectures by large. However one thing is noteworthy, when the SSPs were set to 128, HA does perform less optimal than CA, which means that as the distributed auctions in the HA are performed more often than the central auction in the CA. The communication delays and overheads grow with an increased number of main and intermediate auctions, as the number of SSPs to be served by the main and intermediate auction 'increase' in each subset of the network. It could be inferred that, by not overloading the network with too many auctions, CA and HA would result in a better Response Time than other architectures.

However, as can be seen from figure 6.2.3, MB offers better Response Times than all the other architectures, except Offered Loads greater than 1.6 Erlangs. Which implies that as the number of SSPs has grown to 128, synchronous communication in auction architectures have affected (increased) their Response Times. MB seems to offer better Response Time than CA and even HA, when the Offered Load is less than 0.9 Erlangs. Generally, as the number of SSPs has increased, the Response Time for MB has dropped accordingly. The maximum Response Time for MB in Simulation 1, 2 and 3 is 182, 163, 111 respectively.

It can be observed that the Response Time for MB grows 'linearly', when the Offered Load goes beyond the Target Load. One reason for this could be the uneven trend of the incoming service requests [13]. This causes a little more congestion at the SCPs in MB as compared to other architectures, increasing its Response Times. This argument is verified in the results presented and discussed in section on Request Processing, where it is discovered that MB accumulates relatively high rate of service request queues at SCPs (see figure 6.2.4, 6.2.6, 6.2.8) and results in a linear increase in the Queue Lengths as the Offered Load exceeds the Target Load. Hence, it results in high Response Times, due to internal congestion in MB architecture.

The CLB architecture seems to exhibit the *same* behaviour in terms of Response Time, irrespective of the number of SSPs at different Offered Loads. It always produces the

maximum Response Time as the Offered Load goes beyond 0.70 Erlangs. However, as can be seen from the figure 6.2.1, 6.2.2 and 6.2.3 the Response Time for the CLB increases as the Offered Load approaches the Target Load. It is due to the fact that as long as the Offered Load is kept below the Target Load, the finite queue either remains empty or is decreasing. As the Offered Load approaches the Target Load more closely (in the rage of 0.70 – 0.95 Erlangs), the finite queue starts filling up. Hence the Response Time for the CLB architecture grows exponentially in this phase. Once it passes the Target Load, the Response Time remains even, somewhat in the range from 660-680 seconds. This means that after the Offered Load crosses 0.9 Erlangs, the finite queue is filled to the maximum and therefore results in an evenly distributed Responses Time afterwards. In the CLB architecture, the size of the central queue imposes an upper bound on the burst size that can be accepted. Hence *the maximum Response Time of a request is proportional to the bucket size*.

6.2.2 Request Processing Delays

Another attribute to study the communication delays, in the different architectures is accessing the Processing Delays on the SCPs, at different Offered Loads. One important parameter that could help in determining the load situation on the SCPs is the Queue Length of the service requests that SCPs accumulate. This section provides an insight into these observations. The Queue Lengths of service requests at every SCP had been recorded at every second of a simulation. The mathematical mean of these values is then taken to determine the 'Mean Queue Length' at a single Offered Load.

The Mean Queue Lengths accumulated on all SCPs, in all the architectures, at different Offered Loads for Simulation-1 have been presented in figure 6.2.4. The Queue Lengths of all the architectures show larger deviations when the Offered Load is in the range of 0.7 to 1.05 Erlangs. As the Offered Load goes beyond the Target Load CLB, CA and HA maintain somewhat a constant queue size. However MB shows a linear increase in the Queue size as the Offered Load increases, beyond the Target Load.

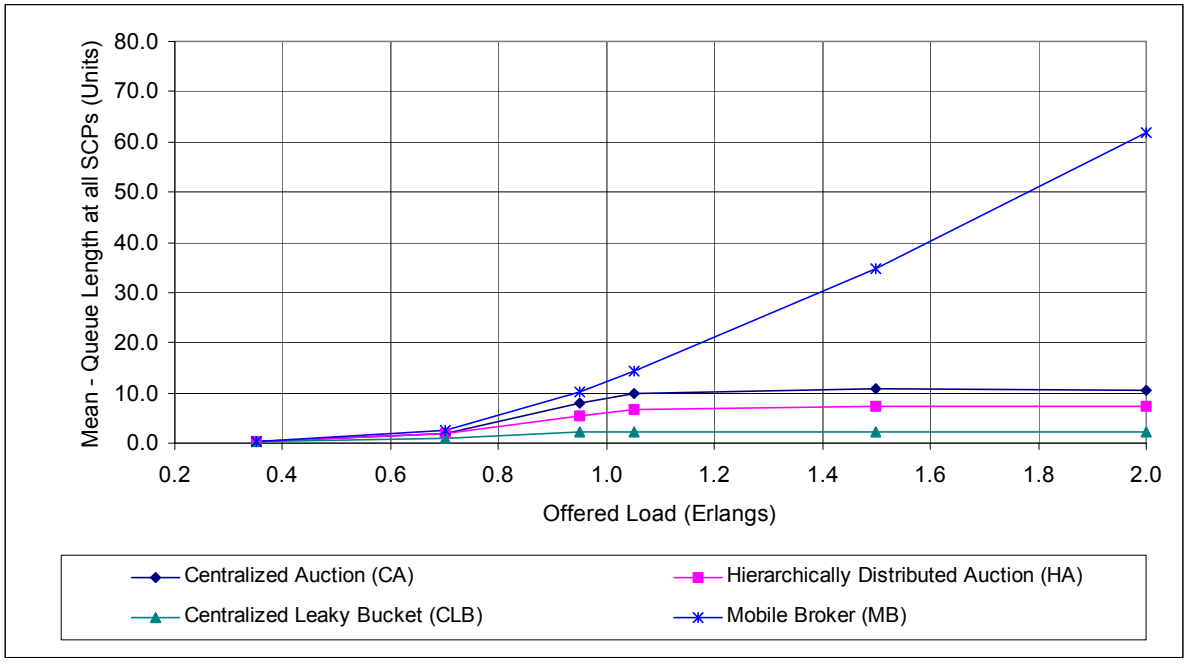
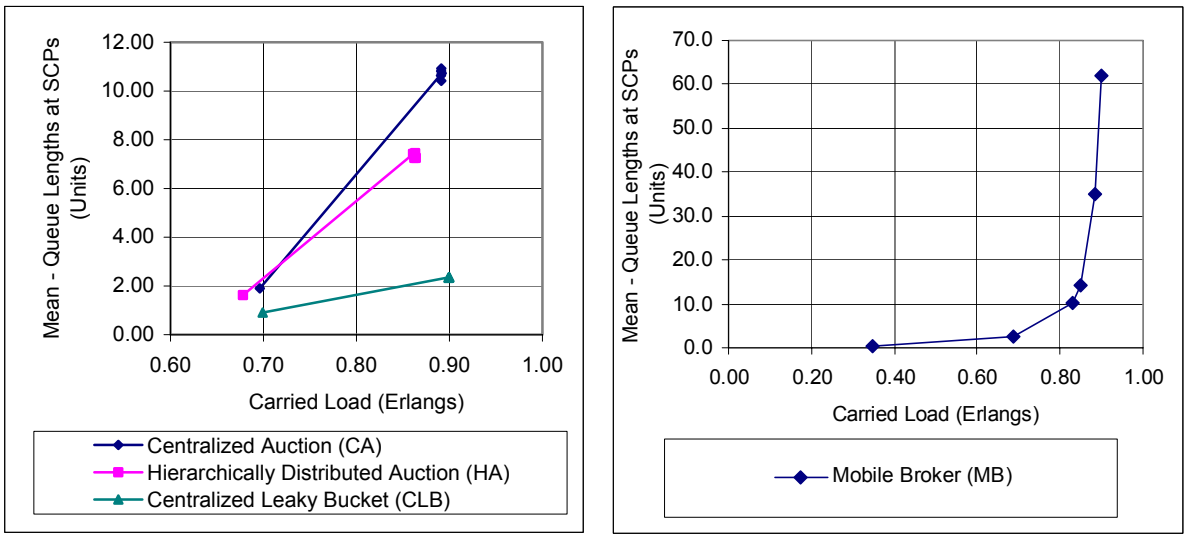


Fig. 6.2.4, Simulation-1 (32-SSPs), The 'Mean of Queue Lengths' at different 'Offered Loads'

Figure 6.2.5 (a), shows a graph where the Queue Length at the SCPs are plotted against the Carried Load for CLB, CA and HA architectures. Note the sharp increase in the Queue Length of the MB architecture, figure 6.2.5 (b), which reveals that as soon as the Carried Load approaches the Target Load, the Queue Length grow exponentially.



(a) *(b)*

Fig. 6.2.5, Simulation-1 (32-SSPs), The Mean of 'Queue Lengths' at different 'Carried Loads'

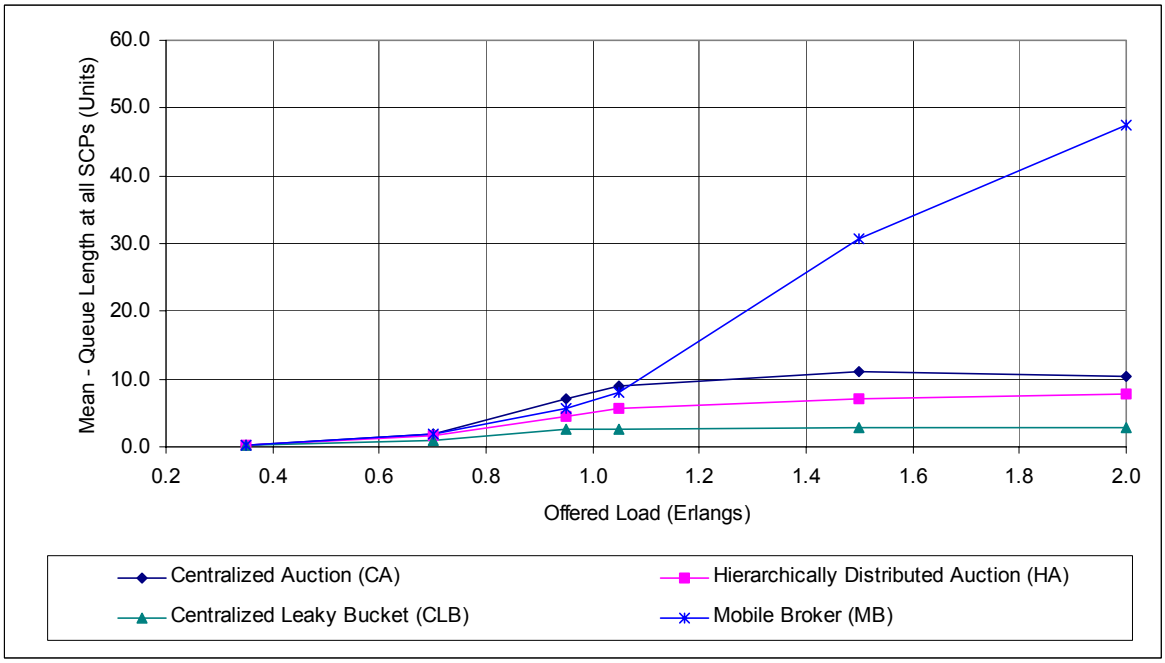
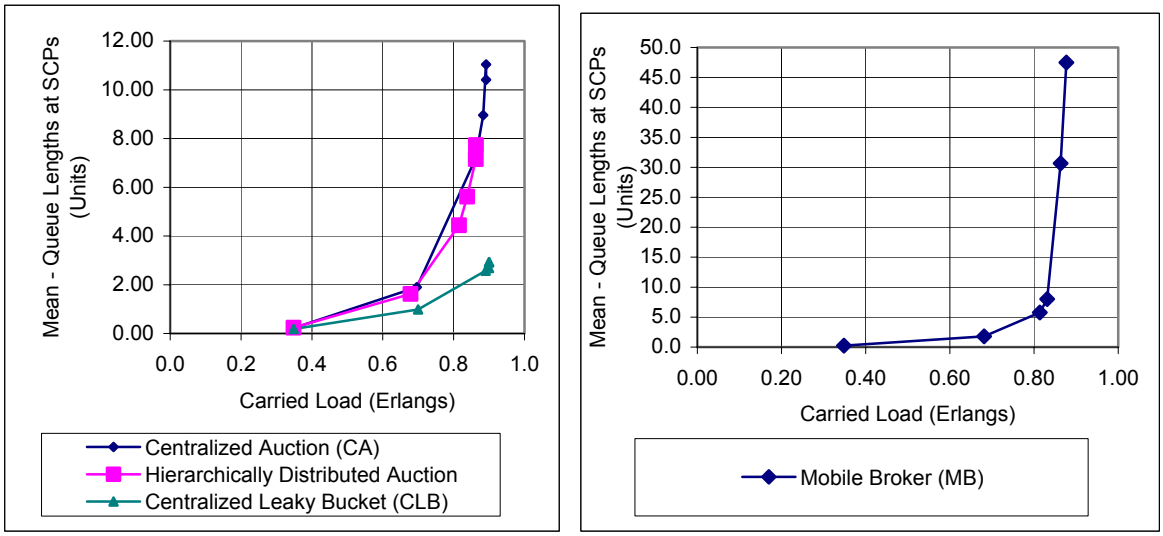


Fig. 6.2.6, Simulation-2 (64 SSPs), The Mean of 'Queue Lengths' at different 'Offered Loads'

Results from Simulation-2, are presented in figure 6.2.6. Again, the CLB architecture posses the minimum Queue Lengths of all the architectures at different Offered Loads. Note that before the Offered Load exceeds 1.1 Erlangs, the Queue size at MB is less that that of CA. Figure 6.2.7 (a), shows a graph where the Queue Lengths are plotted against the Carried Load for CLB, CA and HA architectures. Notice that after the Offered Load approaches the Target Load, the Queue Lengths grow exponentially for CA, HA and MB.



(a) *(b)*

Fig. 6.2.7, Simulation-2 (64 SSPs), The Mean of 'Queue Lengths' at different 'Carried Loads'

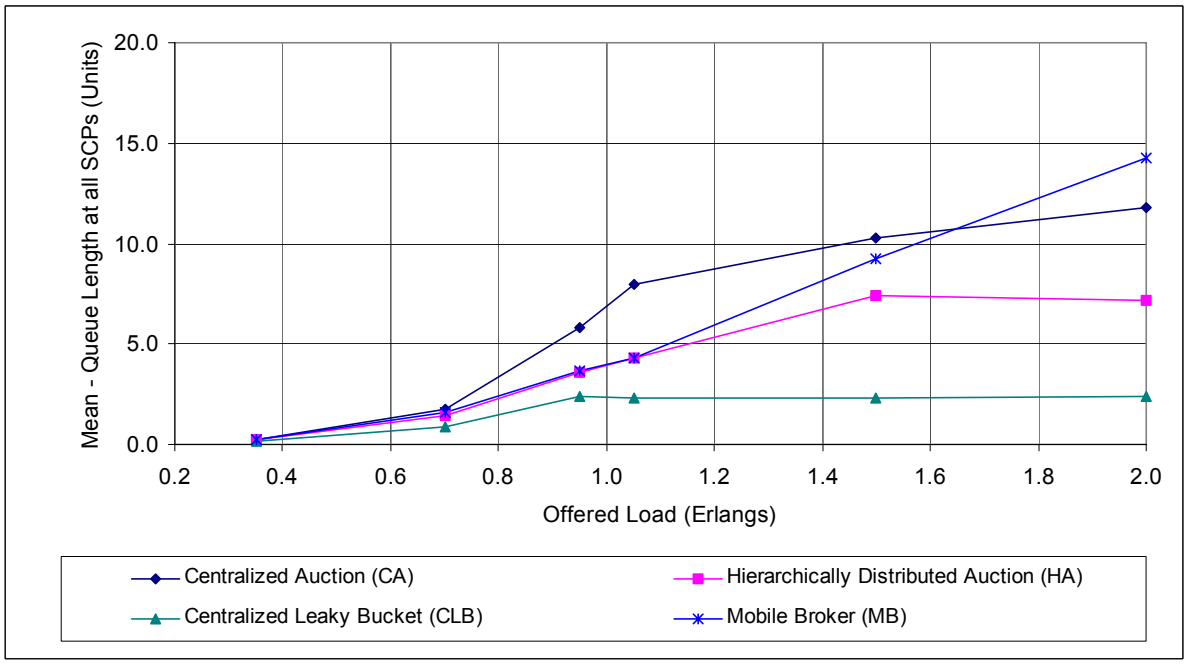
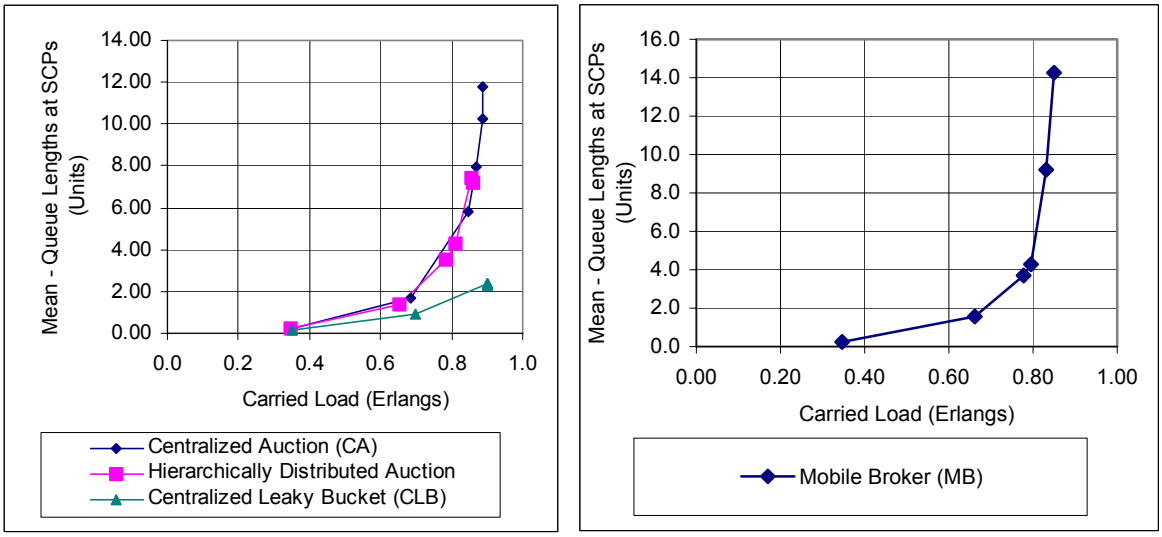


Fig. 6.2.8, Simulation-3 (128 SSPs), The Mean of 'Queue Lengths' at different 'Offered Loads'



(a) *(b)*

Fig. 6.2.9, Simulation-3 (128 SSPs), The Mean of 'Queue Lengths' at different 'Carried Loads'

The Mean Queue Lengths for Simulation-3 have been shown in Fig. 6.2.8. Further increase in the number of SSPs has affected the Queue Lengths of different architectures. CA architecture, in this case, accumulates most queues of service requests at SCPs before the Offered Load exceeds 1.65 Erlangs. For MB architecture, the increase in the SSP number from 64 to 128 reveals that the Maximum-Mean Queue Lengths has decreased as can be seen in result from Simulation-1 and Simulation-2, refer to figures 6.2.5 (b) and 6.2.7 (b).

Figures 6.2.10, 6.2.11, 6.2.12 and 6.2.13 present the Mean Queue Length of service requests at the SCPs, at different Offered Loads and number of SSPs by individual architectures for Simulations 1-2 and 3.

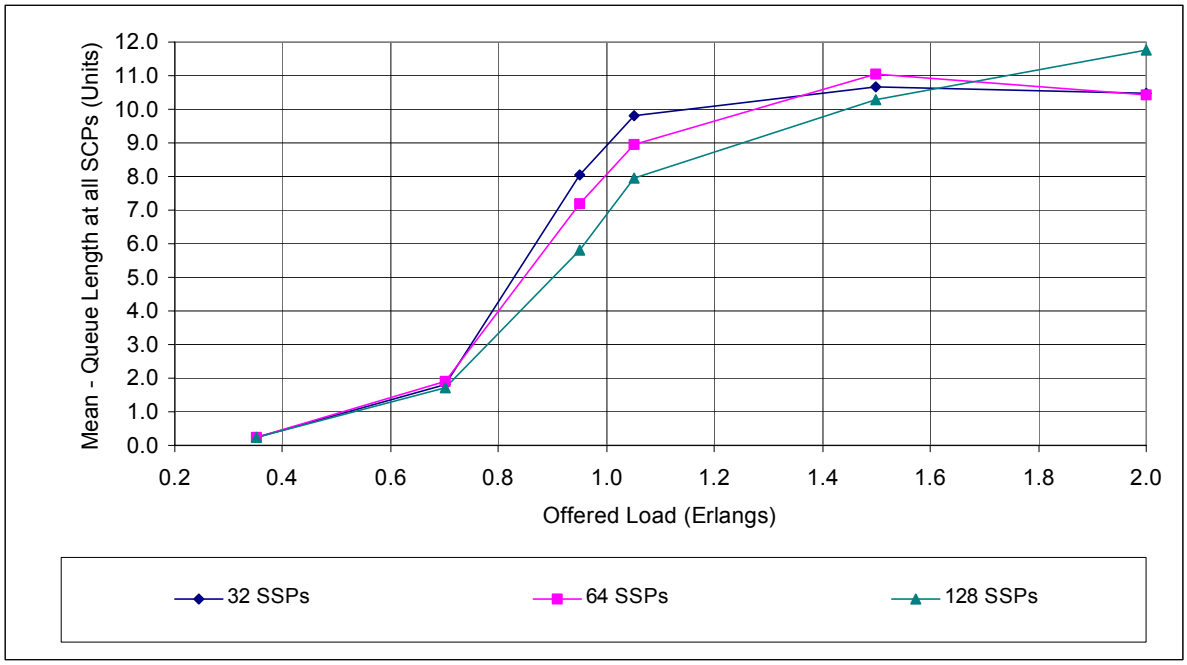


Fig. 6.2.10, Simulation 1-2-3, The 'Mean of Queue Lengths' at different 'Offered Loads' for the Centralized Auction Architecture

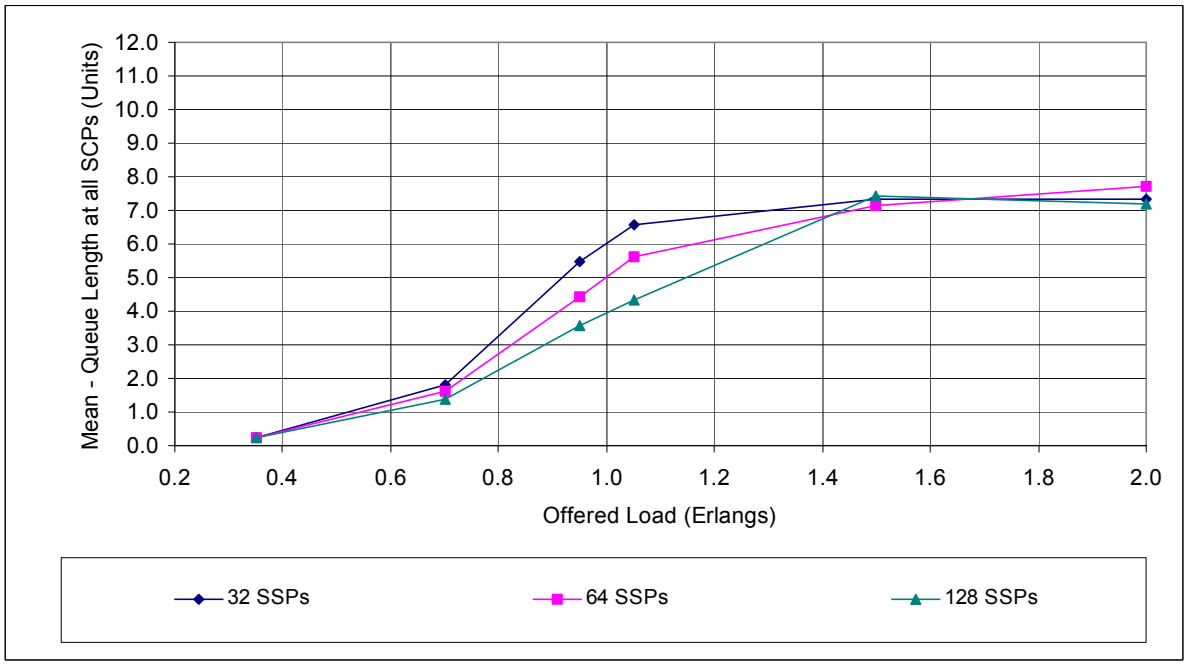


Fig. 6.2.11, Simulation 1-2-3, The 'Mean of Queue Lengths' at different 'Offered Loads' for the Hierarchically Distributed Auction Architecture

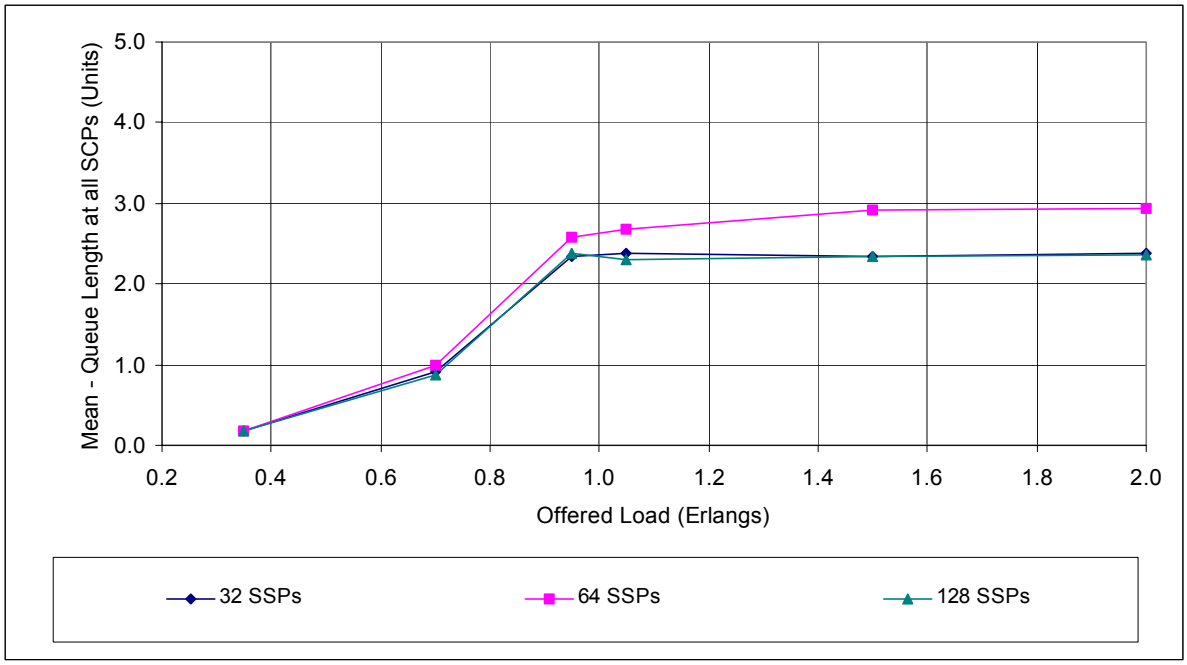


Fig. 6.2.12, Simulation 1-2-3, The 'Mean of Queue Lengths' at different 'Offered Loads' for the Centralized Leaky Bucket Architecture

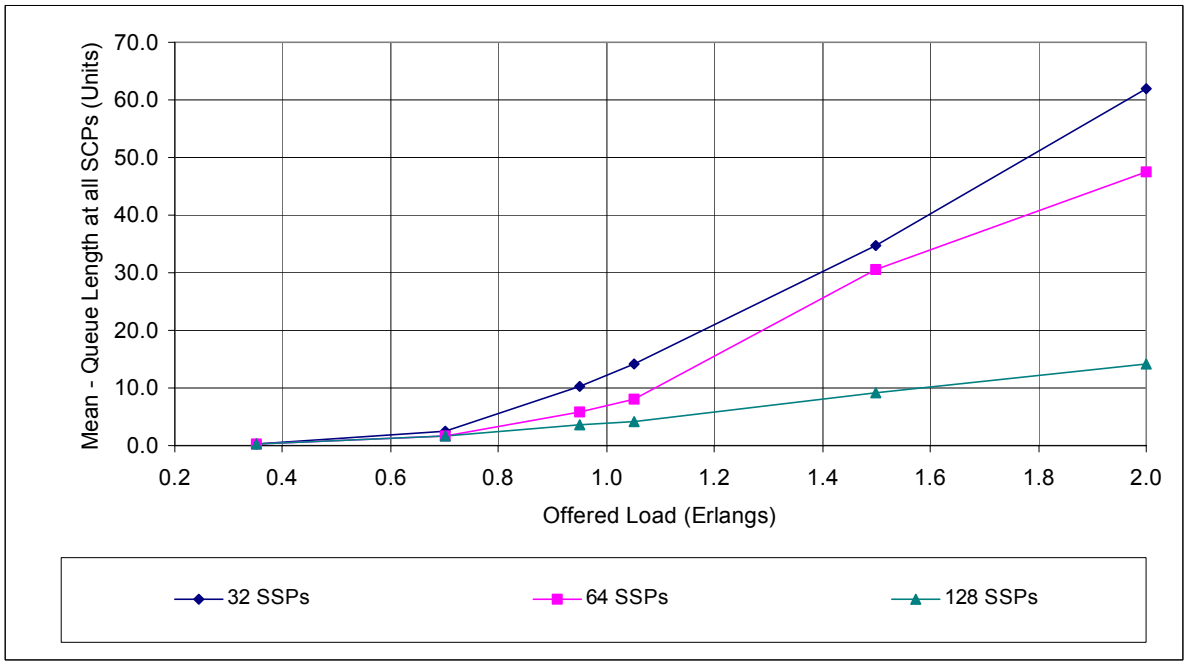


Fig. 6.2.13, Simulation 1-2-3, The 'Mean of Queue Lengths' at different 'Offered Loads' for the Mobile Broker Architecture

It is observed from the results presented in this section that MB stands out from the rest of the architectures in accumulating higher Queue Lengths at SCPs. Notice the significant decrease in the Queue Lengths by MB, in results from Simulation 3 as compared to results in

Simulation 1-2. The CA and HA architectures possess less request Queues at SCPs, waiting to be processed and therefore result in a better performance in respect to network Responsiveness and Messaging Delays comparatively.

CLB accumulates the least number of process request queues at SCPs. The main reason being, *in the CLB all the service requests are de-queued by the central router to an SCP, which has the largest idle time, at a rate equal to the Target Load*. This mechanism helps prevent SCPs accumulate large queues or service requests. This working mechanism results in avoiding large service request queues at SCPs. Though accumulating the least number of queues at SCPs, CLB suffers from higher message delays (as discussed in the next section on Message Delays). Also clear from the results of Simulation 1-2-3, shown in figures 6.2.1, 6.2.2 and 6.2.3. CLB also has comparatively high Response Times (as discussed in previous section on Responsiveness).

6.2.3 Messaging Delays

The next set of the analysis represents an insight into the network message delays. In order to determine the affect of network resize on the messaging, the start/stop times for the Call Connection Messages have been recorded and analysed. This difference in the call start and stop times would give us the precise information about the time delays encountered by network messages, when the network grows in size.

Note that the results used in this section were recorded in the last 10 minutes, every 100th second. The mathematical mean of these values was then calculated and is represented in the figures that follow.

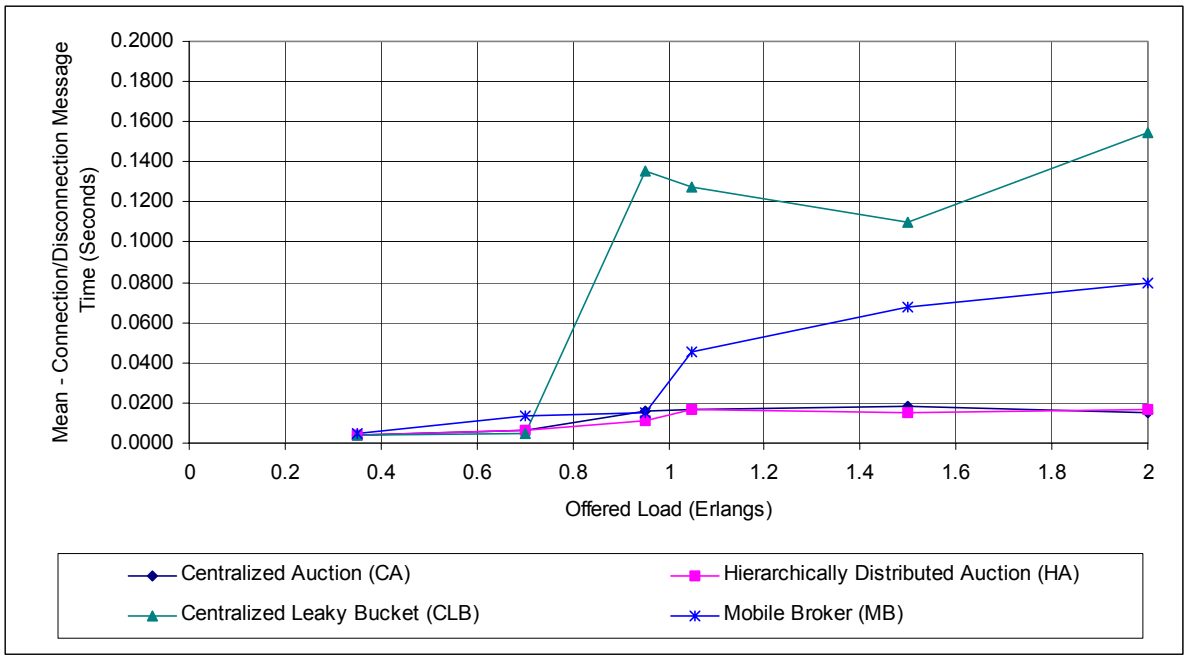


Fig. 6.2.14, Simulation-1 (32-SSPs), The 'Mean Connection Time' at different 'Offered Loads'

Figure 6.2.14 represent results from Simulation-1. CLB showing the largest increase and deviations in connection times. The connection times for the MB architecture are better than CLB and shows highest deviations when the Offered Load exceeds 0.9 Erlangs. The CA and HA offer comparatively least times for connection/disconnection messaging, not exceeding 0.02 seconds at higher Offered Loads.

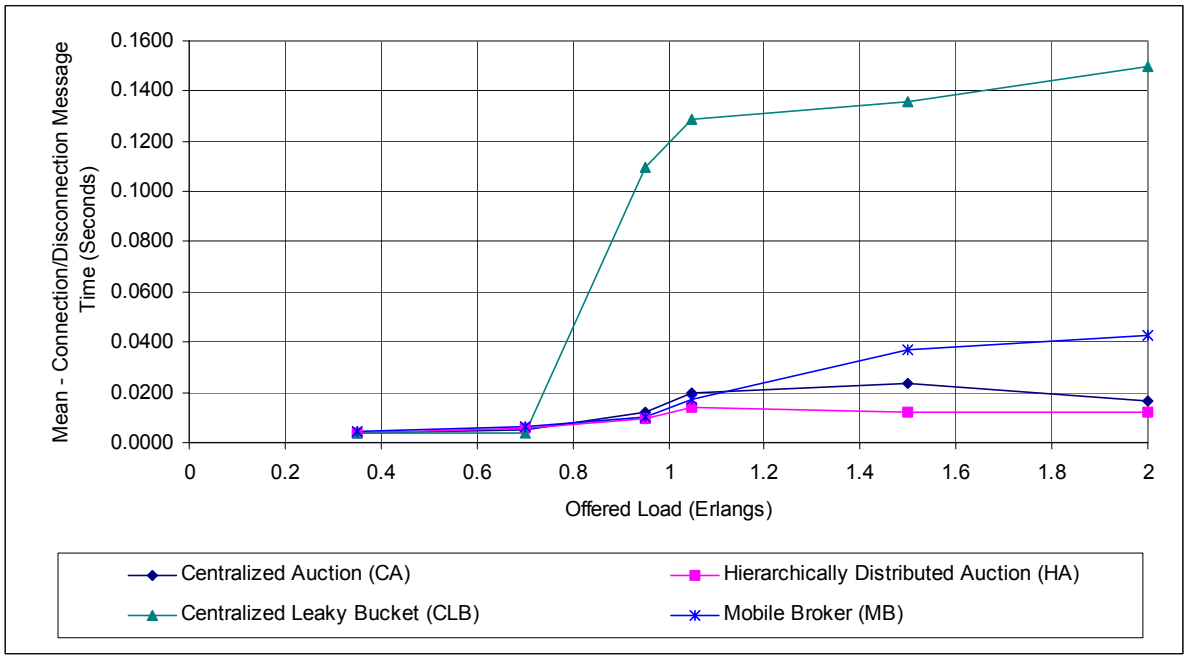


Fig. 6.2.15, Simulation-2 (64 SSPs), The 'Mean of Connection Time' at different 'Offered Loads'

Figure 6.2.15, shows a representation of the time delays, when the number of SSP was set to 64 (Simulation-2). In this case, note the exponential increase in the call connection message times when the Offered Load approaches the Target Load increasing from 0.004 to 0.13 seconds as the Offered Load moves from 0.70 to 1.05 Erlangs. HA has the least delays in connection, of all the architectures, i.e 0.012 seconds. Note that as the Offered Load approaches 2.0 Erlangs the call connection time for CA converges to HA.

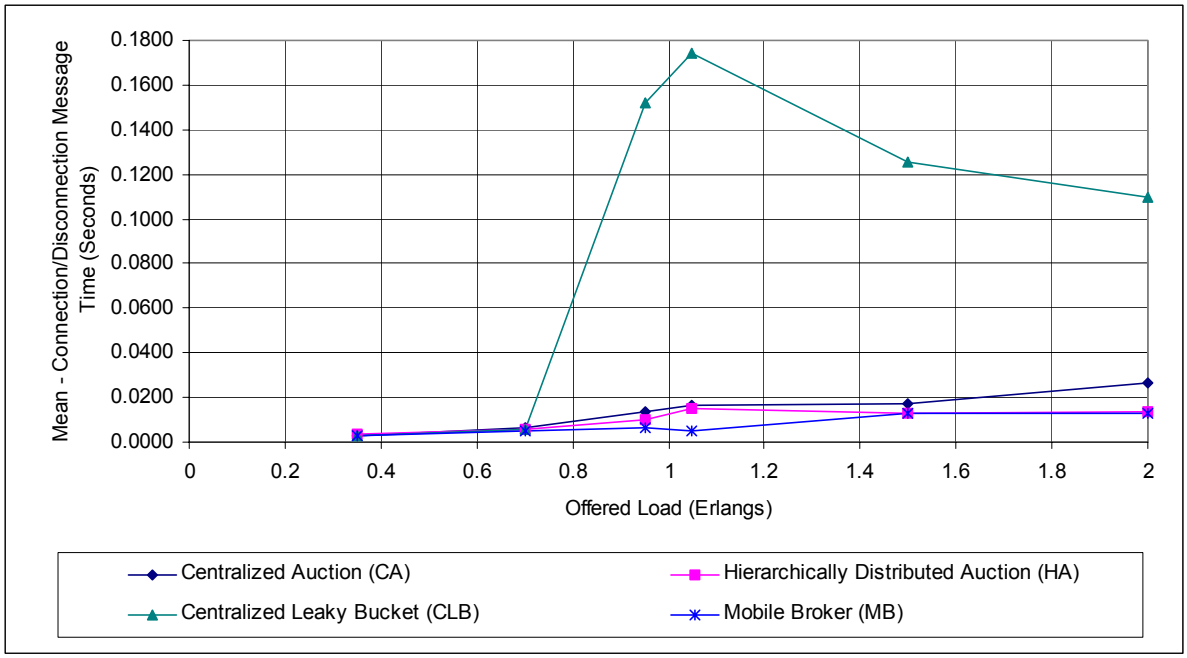


Fig. 6.2.16, Simulation-3 (128 SSPs), The 'Mean of Connection Time' at different 'Offered Loads'

Figure 6.2.16, shows a representation of the time delays, when the number of SSP was set to 128 (Simulation-3). In this case, notice the drop in the connection/disconnection times for CLB, after the Offered Load surpasses 1.05 Erlangs. It is worth mentioning that MB now reflects comparatively least messaging time than all the other architectures, when the Offered Load exceeds 1.50 Erlangs. However, HA and MB have the same call connection times at higher Offered Loads.

Figures 6.2.17, 6.2.18, 6.2.19 and 6.2.20 present the Mean Connection Time of service requests, at different Offered Loads and number of SSPs by individual architectures for Simulations 1-2 and 3.

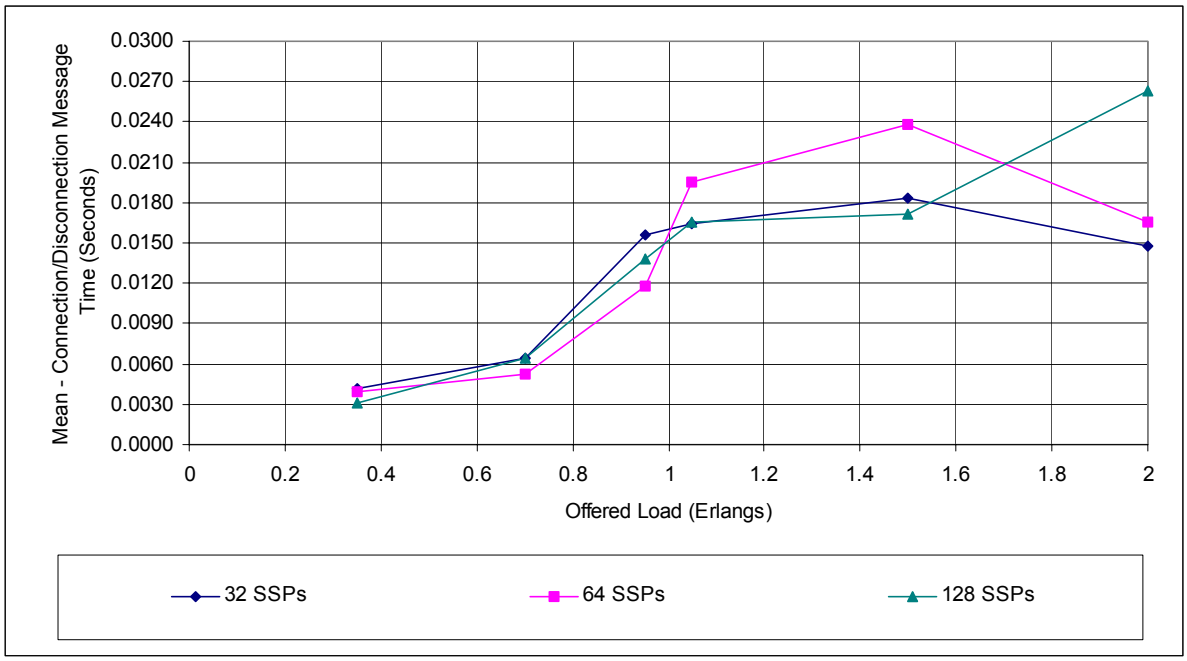


Fig. 6.2.17, Simulation 1-2-3, The 'Mean Connection Time' at different 'Offered Loads' for the Centralized Auction Architecture

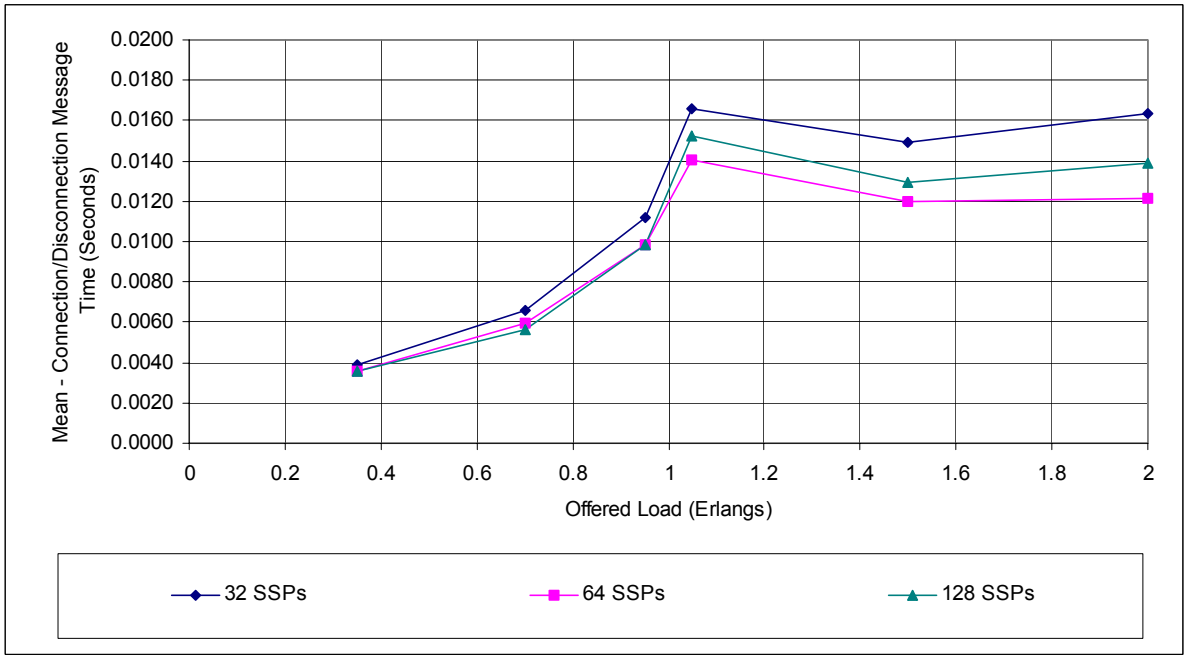


Fig. 6.2.18, Simulation 1-2-3, The 'Mean Connection Time' at different 'Offered Loads' for the Hierarchically Distributed Auction Architecture

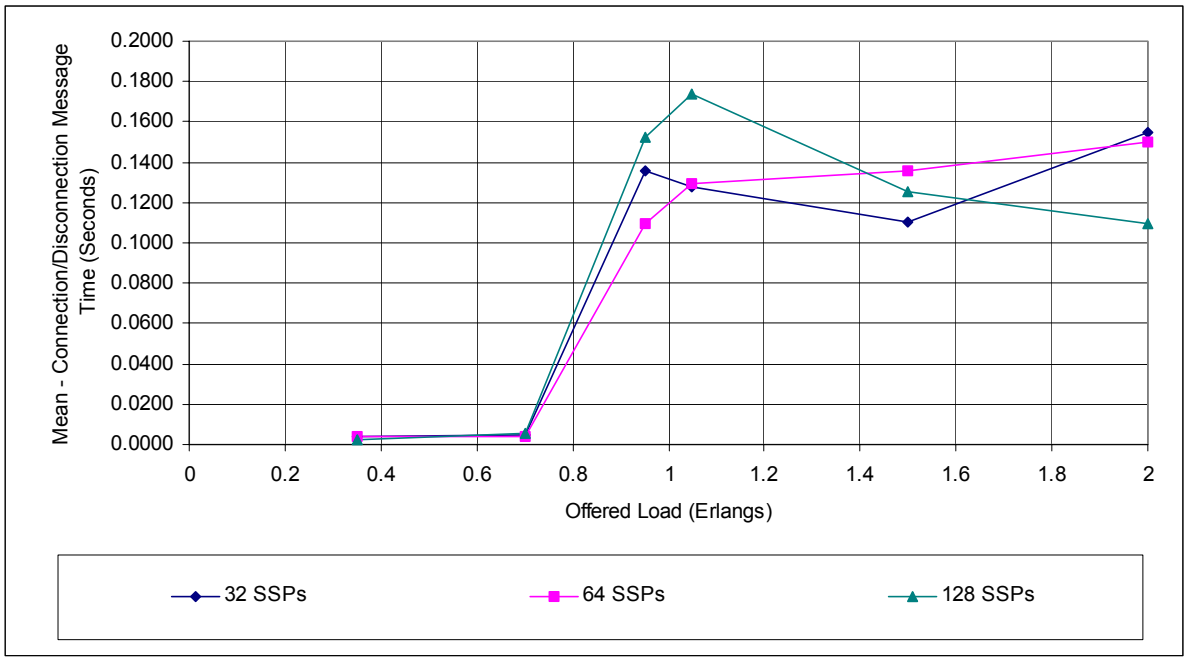


Fig. 6.2.19, Simulation 1-2-3, The 'Mean Connection Time' at different 'Offered Loads' for the Centralized Leaky Bucket Architecture

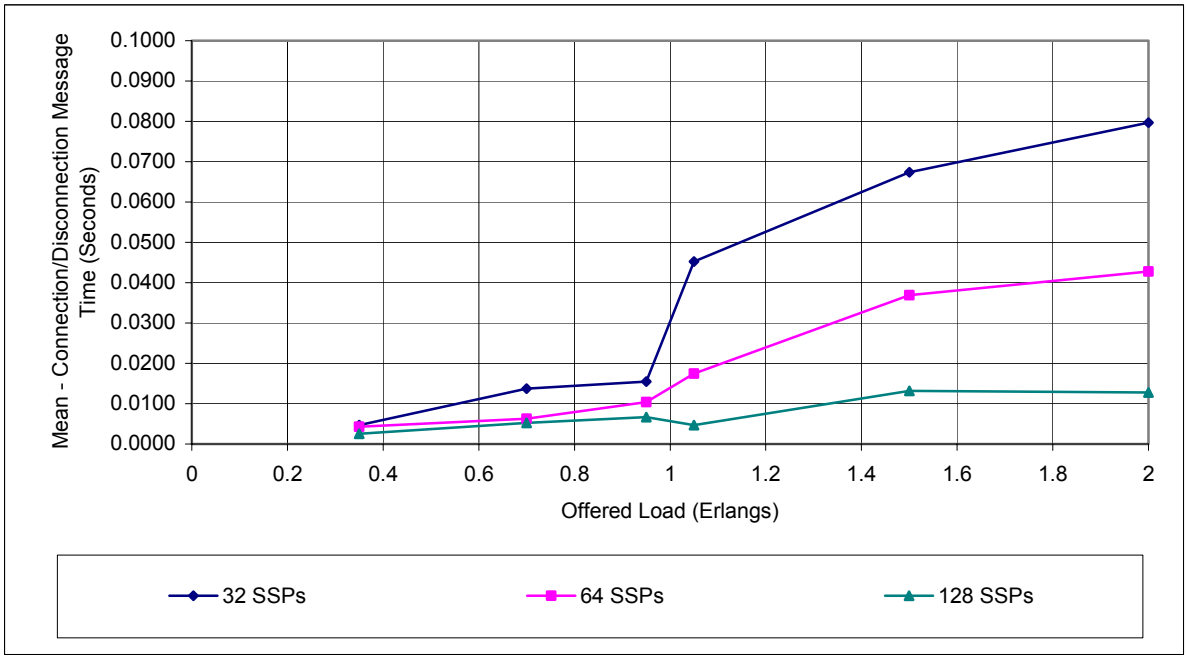


Fig. 6.2.20, Simulation 1-2-3, The 'Mean Connection Time' at different 'Offered Loads' for the Mobile Broker Architecture

A general observation is that *the Call Connection messages in asynchronous architectures (CLB and MB) are more affected by increase in Offered Loads and network resize as*

compared to synchronous architectures (CA and HA). This is due to the enhanced level of communication that is due to the inherent nature of (continuous) *communication* typical of asynchronous architectures.

6.3 Call Accept/Reject Rates

In this section an investigation is performed to analyse the affect of a resize in the network over number of calls accepted/rejected by the network. The total number of Accepted Calls and Rejected Calls by an architecture plotted against the different Offered Loads, for investigations.

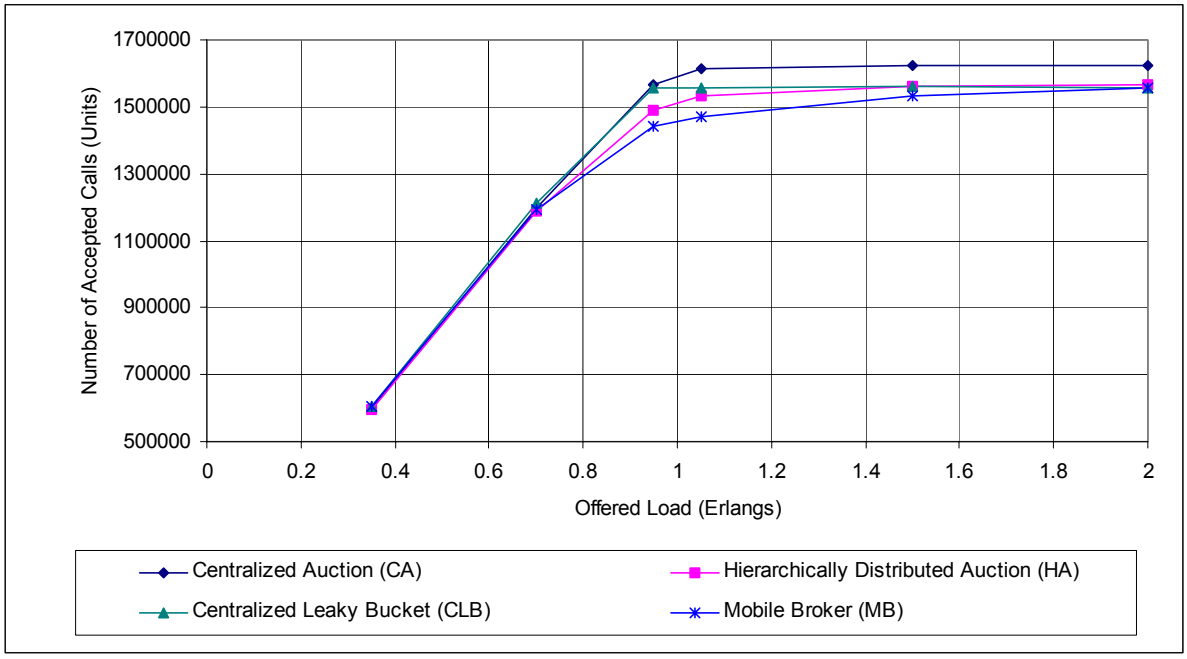


Fig. 6.3.1, Simulation-1 (32-SSPs), 'Call Accept Rate' at different 'Offered Loads'

Figure 6.3.1 above, represents results from Simulation-1. The figure shows that all the architectures manage to accept almost the same number of calls until the Offered Load equals 0.7 Erlangs. It can be easily observed that CA stands out, accepting the maximum number of calls as compared to the other architectures, at all Offered Loads. Figure 6.3.2 depicts results of Simulation-1, showing the Call Reject rates of the four architectures. The CLB, is though the architecture that carries the Carried Load closest to the Target Load, accepts 'lesser' calls than CA, at Offered Loads greater than the Target Load.

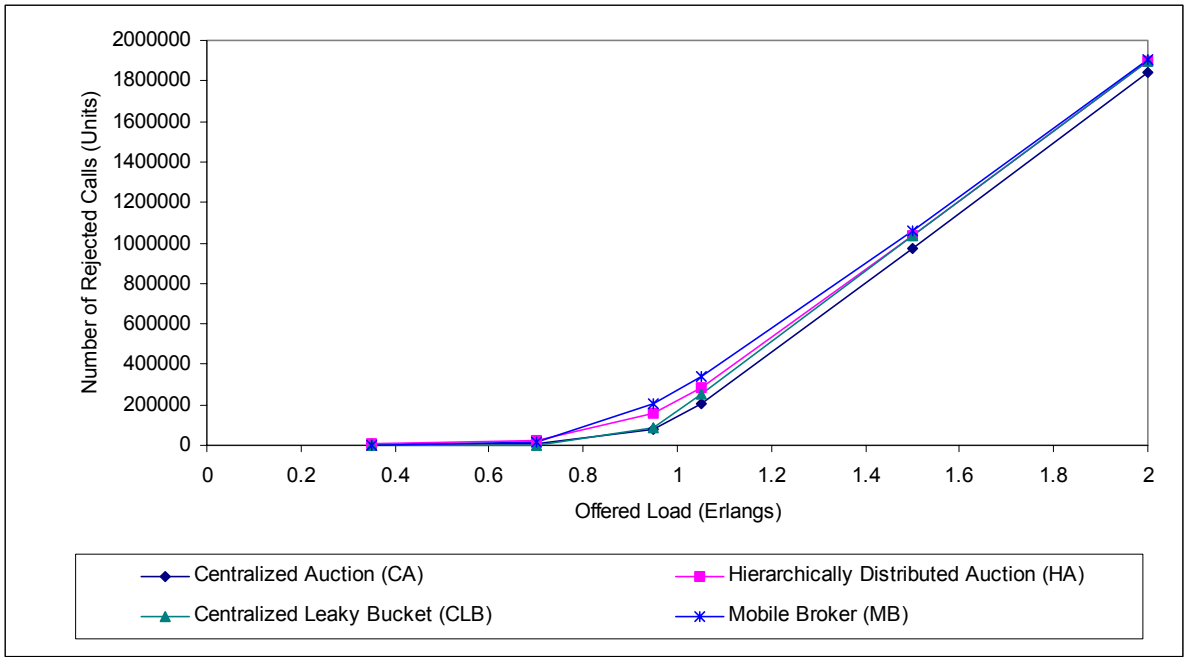


Fig. 6.3.2, Simulation-1 (32-SSPs), 'Call Reject Rate' at different 'Offered Loads'

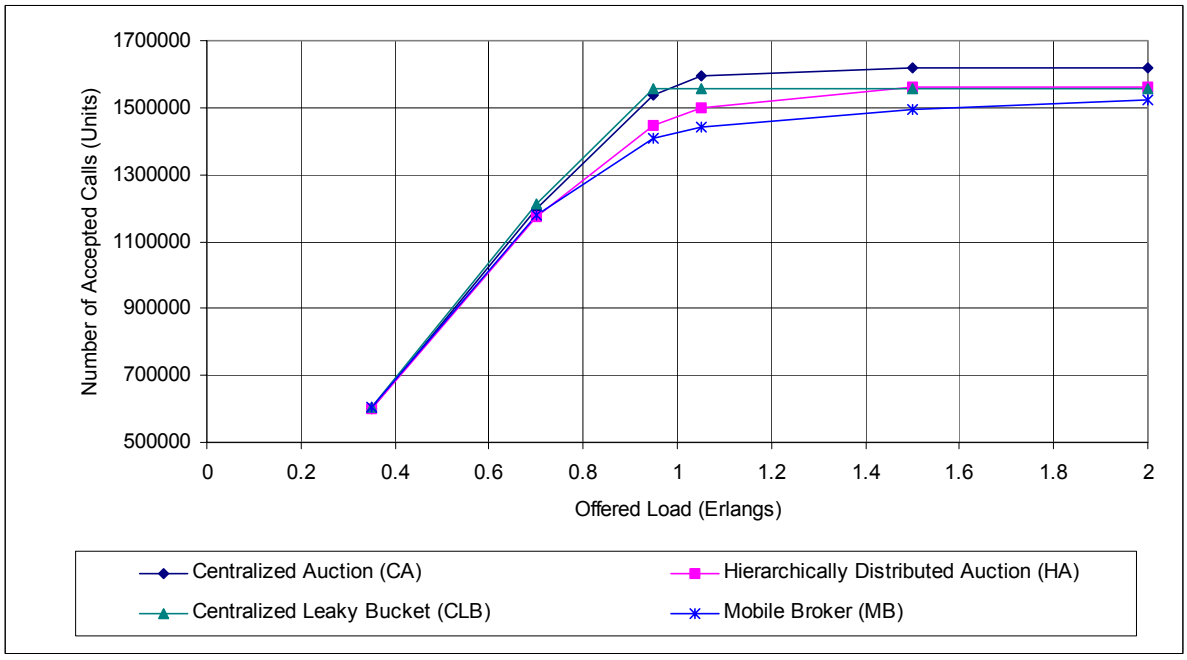


Fig. 6.3.3, Simulation-2 (64 SSPs), 'Call Accept Rate' at different 'Offered Loads'

Figure 6.3.3, represents similar results from Simulation-2. One difference is significant in this particular result as compared to the results from Simulation 1 i.e. before the Offered Load exceeds the Target Load, CLB accepts the most incoming calls. Yet when the Offered Load surpasses the Target Load, CA accepts the most incoming calls as compared to all the other architectures.

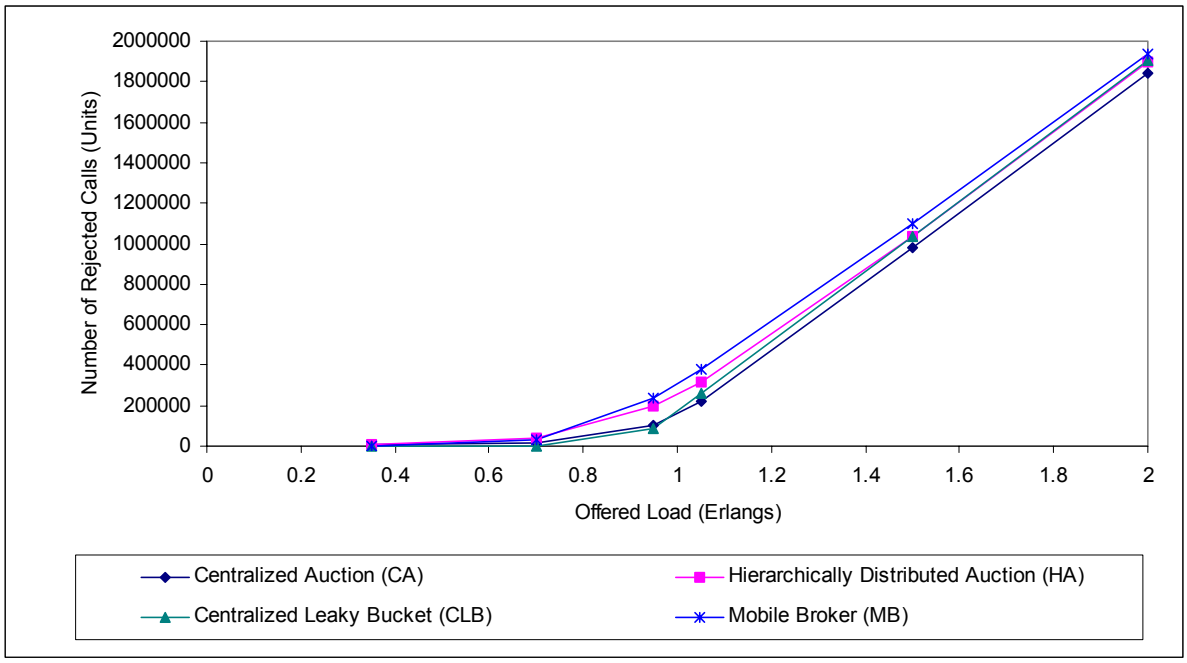


Fig. 6.3.4, Simulation-2 (64 SSPs), 'Call Reject Rate' at different 'Offered Loads'

Figure 6.3.4 depicts results of Simulation-2, showing the Call Reject rates of the four architectures

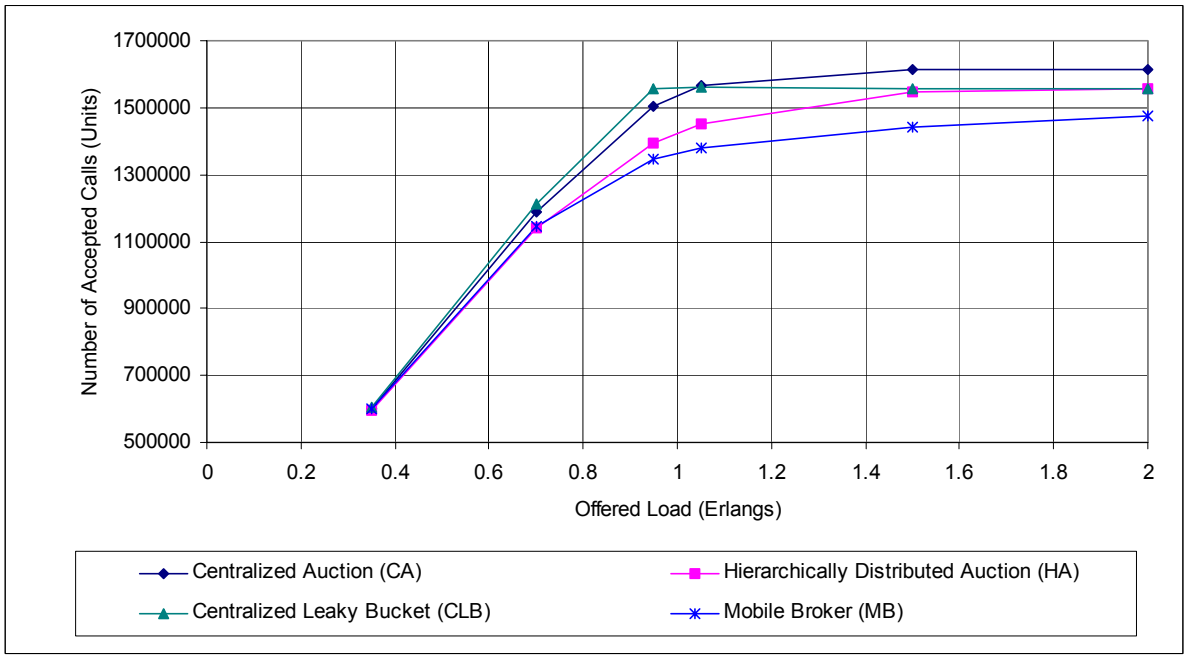


Fig. 6.3.5, Simulation-3 (128 SSPs), 'Call Accept' Rate at different 'Offered Loads'

Figure 6.3.5, depicts results from Simulation-3. The number of the Accepted Calls by all the architecture reflect almost the same behaviour as in Simulation 1 and 2. As can be seen in figure 6.3.6, the Call Reject rates for MB is highest, after the Offered Load exceeds 0.7 Erlangs. Again the CA accepts the maximum number of calls.

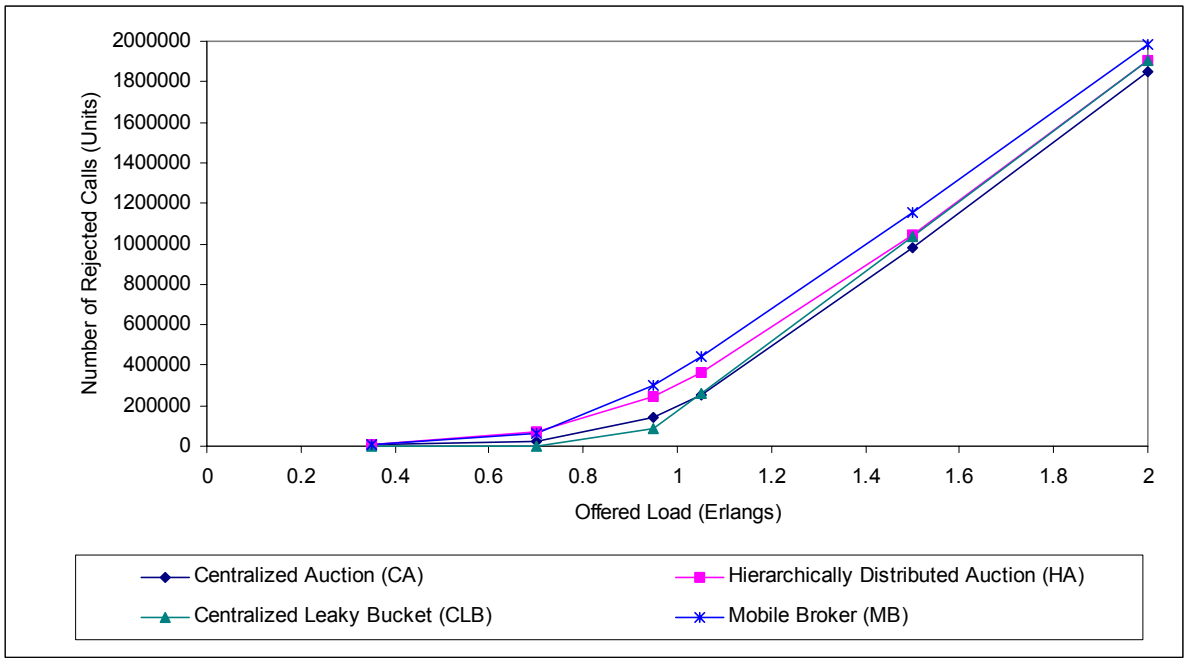


Fig. 6.3.6, Simulation-3 (128 SSPs), 'Call Reject Rate' at different 'Offered Loads'

One of the general observations which have been noted by the results of the analysis is that the number of Accepted Rejected Calls by CA, HA and CLB architectures do not show significant difference in behaviors as we increase the Offered Load and the number of SSPs. For the MB architecture, there is a slight decrease in the number of Accepted Calls as we increase the number of SSPs and especially at Offered Loads greater than Target Loads. *One thing is worthwhile to mention, that the Call Accept Rate for an architecture would be relative to the Carried Load by it. This number would drop if the overall Carried Load is dropped and vice versa.*

6.4 Overhead Communication

This section presents an analysis of the Communication Overhead produced by the *allocation mechanism*, only. There exist two different methods to determine the communication overheads. One is based on the number of 'sent messages' and the other on the 'required bandwidth' [13]. The *auction architectures (i.e. CA and HA)* place a more severe constraint on the underlying messaging infrastructure, when the required bandwidth is considered. Since bulk of the messaging occurs just before (when the bids are sent in) and after (when the results about allocations are broadcasted to the Allocators) the auctions. Besides this, no messages related to allocations are sent in the auction architectures, keeping the message channel idle most of the time.

In light of the above considerations, the main factor affecting the communication overhead in auction architectures is the size of the auction time intervals. The number of the agent interaction messages in the CA is proportional to the number of main auctions, number of Allocators and the number of Quantifiers. Whereas the number of the agent interaction messages in the HA is proportional to the number of main and intermediate auctions, number of Allocators and the number of Quantifiers.

For calculating the communication overheads for CA and HA, it is assumed that all bids from the SSPs and SCPs are sent evenly distributed within one second. The Quantifiers send in their tokens corresponding to the available capacity of their respective SCPs to be sold, to the Distributors. And on the other hand, the Allocators send in their bids to the Distributors to buy this available capacity, in an auction.

The amount of the communication overhead in the CA is around 40 messages per second irrespective of the Offered Load, in Simulation 1. The value has been calculated such that for Simulation 1, 32 Allocators would send in their bids to the Main Distributor (32 messages sent to the Distributor, one message by every Allocator since each Allocator acts on behalf of one SSP), and 8 Quantifiers send in their tokens for selling the available capacity (8 messages sent to the Distributor, one message by every Quantifier, where each Quantifier acts on behalf of one SCP). Using the same method as above, there would be around 72 and 136 messages per second, irrespective of the Offered Load for Simulations 2 and 3 respectively for CA.

Similarly for HA, there would be a maximum of 40 messages per second irrespective of the Offered Load, in Simulation 1. Accordingly, for HA when the number of SSPs is 64 (with 4 Intermediate Distributors), there would be a maximum of 72 messages per second, irrespective of the Offered Load. When the number of SSPs is 128 (with 4 Intermediate Distributors), there would be a maximum of 136 messages per second, irrespective of the Offered Load.

In the MB architecture, since every Broker spends 0.2 seconds at an Allocator, one Broker would be able to visit 5 Brokers in a second. In one second, 8 Brokers would visit 40 Allocators. So 40 messages per second would be the overhead communication, when the number of SSPs is 32. This number of messages would remain the same irrespective of the number of SSPs.

In CLB the communication overhead caused by the allocation is proportional to the number of connection requests i.e. the Offered Load. In fact when the Offered Load exceeds the Target Load, the overhead increases even faster [4]. The reason being that when the Offered Load exceeds the Target Load and the central finite queue is filled to the maximum. The Central Distributor must inform the requesting SSPs, in this case, that their service request has been denied. But in our computations for simplicity, we have not accounted for additional messages sent to SSPs in case a request is denied (at higher Offered Loads), to calculate the required bandwidth. It is accounted for, that only one message is sent informing the SSP, whether to connect to the SCP or if the request has been denied. Table 6.4.1 shows the required bandwidth for CLB, as the network size grows (number of SSPs increase).

Offered Load	Required Bandwidth		
	32 SSP	64 SSP	128 SSP
0.35	1009.843	2019.686	4039.373
0.70	2019.84	4039.68	8079.36
0.95	2740.48	5480.96	10961.92
1.05	3029.12	6058.24	12116.48
1.50	4327.68	8655.36	17310.72
2.00	5770.541	11541.08	23082.16

Table. 6.4.1, Required Bandwidth for CLB, projected values for different number of SSPs

Figure 6.4.4 represents the data from Table 6.4.1 in graphical form, against different Offered Loads.

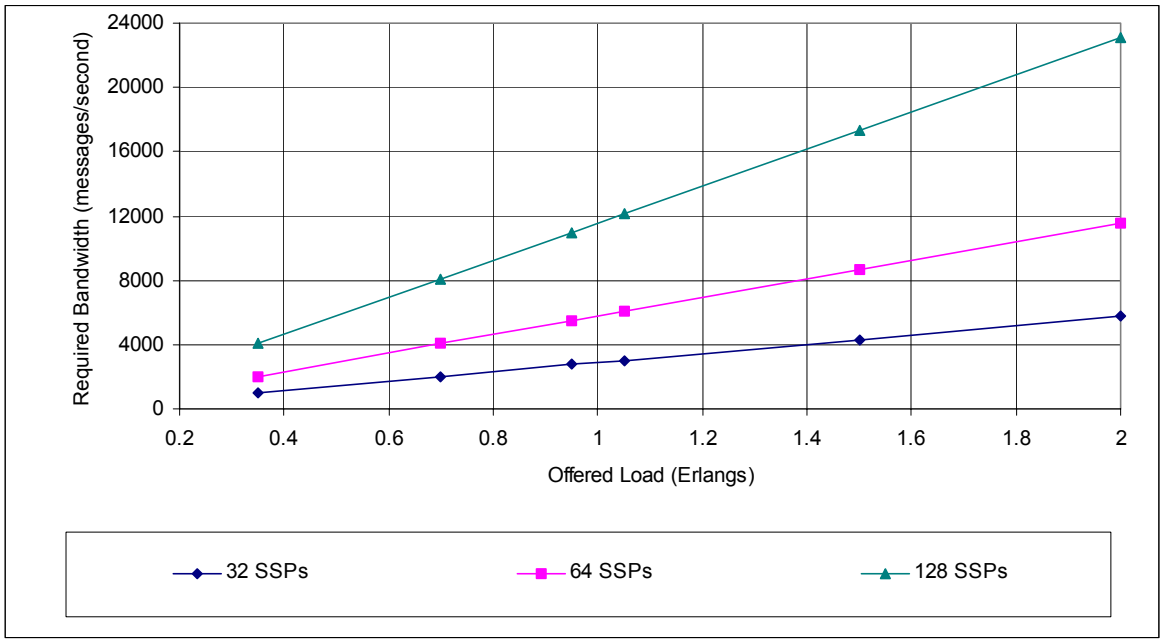


Fig. 6.4.4, Communication Overhead caused by allocations, for Simulation 1-2 and 3 by CLB

6.5 Overhead Computations

The *computational overheads* include the *computational complexity* involved for the resource allocation mechanism alone. For the CA and HA architecture, the major factor having a direct influence on computational efficiency are the computations performed in an auction by the Main Distributor. In case of CA it involves the computational complexity of the Main Distributor however for HA it involves the complexity of Intermediate Distributors in addition to the Main Distributor. Therefore determining the computational complexity of the Distributor/s would provide us with information on computational overheads, as the network size increases.

The computations performed in an auction are dependent on the number of tokens to be sold by Quantifiers, the buyers (Allocators) bidding for them and the number of service types. Lets say, both for HA and CA, that we have 'n' Quantifiers and 'm' Allocators, taking part in an auction. Say we have 'x' service types.

So the maximum number of tokens in the Main Auction would be 'nx'. The maximum number of computations could then be a *combination* 'nmx'. Now if we double the number of Allocators i.e. 2m, as we have done in Simulation 2. The maximum number of computations could be a *combination* '2nmx'. So for higher values of Allocators, we could neglect the constant (i.e. 2) and could infer that the computational overhead for CA is O[n*m*x].

For HA, The Main Distributor would be receiving tokens from the Quantifiers which would be a maximum number 'nx', and requests from Intermediate Distributors to acquire these tokens. Assuming that the number of Intermediate Distributors is kept small, the complexity of the Main Distributor would be O[nx]. Since in this case the number of Intermediate Distributors is kept small, so they would not significantly influence the complexity of the Main Distributor by large. But for Intermediate Distributor the complexity would be O[n*m*x]. Since in Intermediate Auctions there would be 'm' Allocators bidding for 'nx' tokens,

For the Centralized Leaky Bucket architecture an increase in the network size wouldn't affect the number of incoming requests to the Central Distributor. Since the incoming requests would just be distributed with an increase in the number of SSPs. The complexity of the Distributor in the CLB would be O(p), 'p' being the Offered Load.

In the MB architecture Brokers could be seen as Distributors i.e the Agents performing resource allocation. The complexity lies in the route length of the Brokers. The greater the route length, more Allocators visited in turn greater would be the computational complexity. Hence the overall complexity of the MB architecture would grow *linearly* in proportion to the number of Allocators.

Architecture	Computational Complexity
Centralized Auction Architecture	O(nmx)
Hierarchically Distributed Auction Architecture	O(nmx)
Centralized Leaky Bucket Architecture	O(p)
Mobile Broker Architecture	O(m)

Index:

n - Number of Quantifiers
m - Number of Allocators
x - Service Types
p - Offered Load

Table. 6.5.1, The Computational Complexity of the four Architectures

6.6 Load Balancing

The next attribute studied for scalability is measuring the balancing of load by the *providers* (SCPs). In the investigations that follow, the consumption of the resources by the SCPs at different Offered Loads is determined referred to as load balancing. The results have been calculated by taking the standard deviation of the Carried Load at every second. The Average

Standard Deviation is then determined for a particular Offered Load, which is presented in the following results. The results are presented in figures 6.6.1, 6.6.2 and 6.6.3.

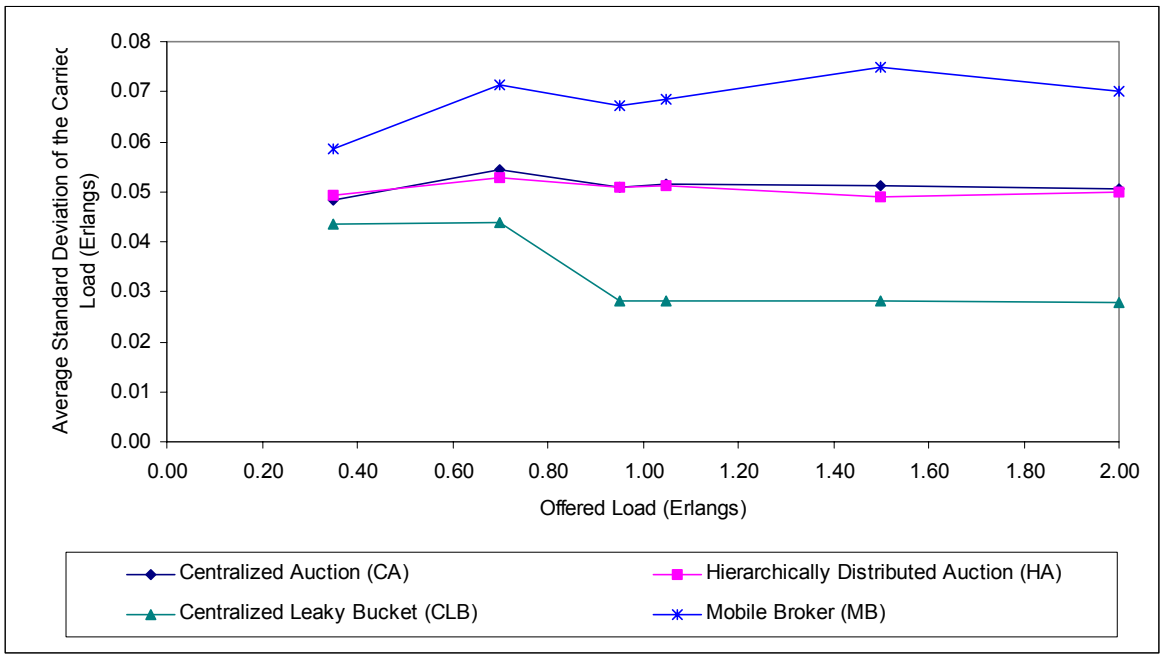


Fig. 6.6.1, Simulation-1 (32 SSPs), The Average Standard Deviation of the Carried Load at the SCPs, at different Offered Loads

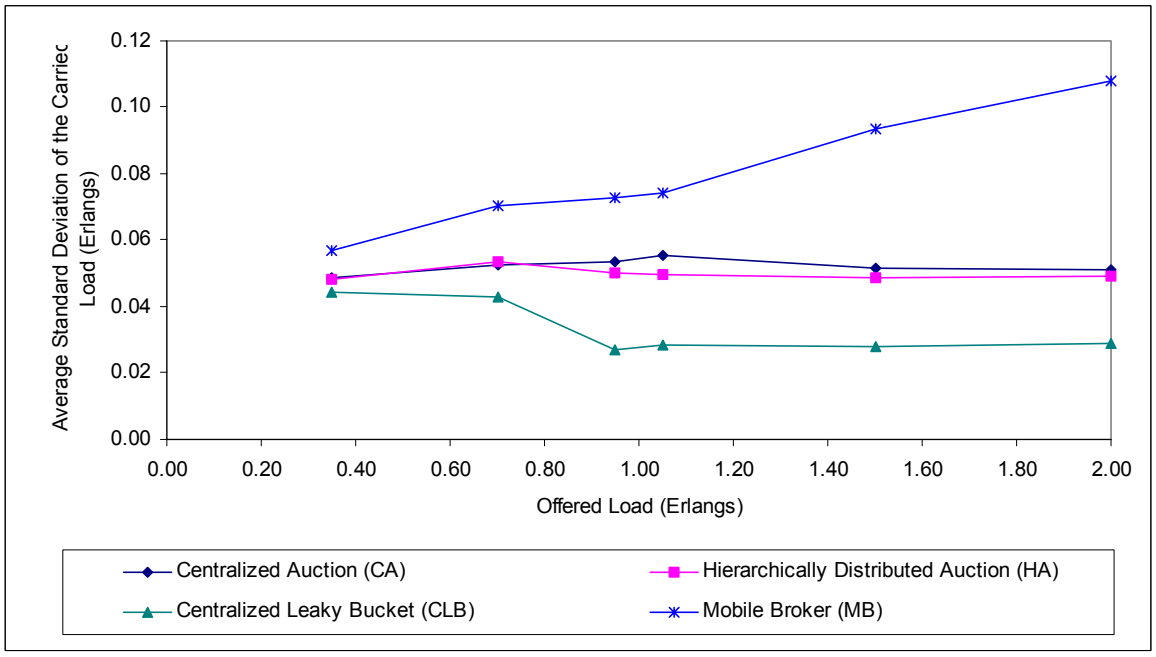


Fig. 6.6.2, Simulation-2 (64 SSPs), The Average Standard Deviation of the Carried Load at the SCPs, at different Offered Loads

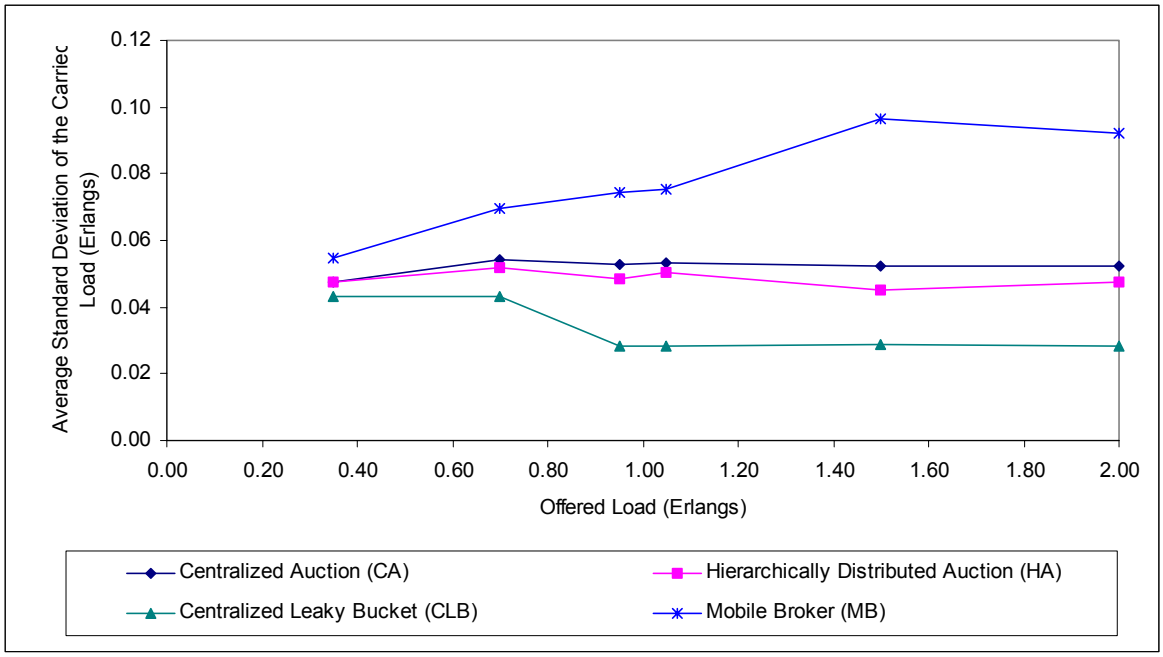


Fig. 6.6.3, Simulation-3 (128 SSPs), The Average Standard Deviation of the Carried Load at the SCPs, at different Offered Loads

Due to the stochastic nature of the statistical distribution used to determine the time to the next request, the time between two consecutive service requests from an SSP varies even though the value of the Offered Load parameter of the SSP is not changed. An SSP can be offered resources from more than one SCP and one SCP can provide resources to more than a single SSP. Which reveals that more SSPs an SCP is serving the less impact on its performance when there is a change in the Offered Load on any of these SSPs and vice versa.

The results presented in this section reveal that the CLB architecture is most stable in terms of load balancing at all Offered Loads and with an increase in the number of SSPs. It produces largest deviations, in load balancing, when the Offered Load reaches the Target Load. It distributes the Offered Load optimally by de-queuing service requests through the central router to an SCP, which has the largest idle time at a rate equal to the Target Load. This distributes the load well among the available SCPs irrespective of an increase in the number of SSPs. Secondly, irrespective of an increase in the network size the additional number of service requests resulting from an increased number of SSPs are simply rejected if the finite queue at the Central Distributor could not accommodate them. Hence the total load on the SCPs is not highly affected by the increase in the network size. Producing almost similar behaviour for Simulations 1, 2 and 3.

In the CA and HA architectures the standard deviation in load balancing is almost coinciding at all Offered Loads and in all the simulation results. However, HA shows relatively more deviations than HA. The reason being an added level of Intermediate Auctions where resource allocations are done in addition to the allocations made at the Main Auctions. So the load at the SCPs is varied due to these added allocations as a result of Intermediate Auctions producing more variations, than CA.

As has been pointed out in the section on Communication Overheads the number of tokens in every auction remains the same irrespective of the number of SSPs. So the allocation mechanism in Auction Architectures does not require more allocations on part of the SCPs with an increase in the number of SSPs. Hence in CA and HA architectures the overall deviations are somewhat same, as the network size in increased.

The MB Architecture produces comparatively higher variations in load balancing compared to all the other architectures as the network size increases. The reason is that the increase in the number of SSPs has proportionally increased the Broker Route length. As we know, that one Broker acts on behalf of a single SCP and most of the Allocators in the Broker route remain the same. As a result, an SCP is mostly serving a particular subset of SSPs. This limitation prevents distribution of the load and adversely affects the performance of the MB architecture in terms of Load Balancing as the network grows.

6.7 Reactivity

It is important to analyse the network *adaptability* to a sudden increase/decrease in the Offered Loads. As has been discussed earlier that in case of an emergency situation the communication network might receive service requests in synchronized bursts. This unexpected increase in the traffic load might overload the network and needs to be managed to avoid resource exhaustion. On the other hand, a telephone network operator is confined to reserve resources and bandwidth for (i) communication within network elements and (ii) for emergency calls (e.g. 112, 911), which must always succeed, even when the network resources are overloaded. So its also important to determine that does a sudden increase in the network Offered Loads results in pushing the Carried Load over the Target Load or not. All of this implies that Reactivity of a network is very important to determine.

6.7.1 Overload Control

To find out how adaptive the different architectures are to sudden increase/decrease in the Offered Loads. First an instant increase in the Offered Load from 0.35 to 2.0 Erlangs was applied following a sudden decrease from 2.0 to 0.35 Erlangs, to all the architectures (see details about the input parameters in Section 5.3 for Simulations 4, 5 and 6). The results have been presented by plotting the Carried Loads (recorded at every second) against the Time.

Before moving on to the actual simulation results, lets determine the shape of the expected result graphs that follow. According to the simulator specifications,

- The duration of the user interaction A1 (phone ringing) is to be drawn separately for each service session from a negative exponential distribution with a mean of 5 seconds.
- The duration of the user interaction A2 (Conversation) is to be drawn separately for each service session from a negative exponential distribution with a mean of 100 seconds.

There are two phases in general, to the resource allocation mechanism in the different architectures namely the ‘Call Connection Phase’ and the ‘Call Disconnection Phase’. The point to consider is that with a sudden increase in the Offered Load at a particular time instant would initiate the Call Connection Phase of a number of accepted incoming service requests. So during the Disconnection Phase of these already connected requests the architectures

would not be able to carry the load closer to the Target Load. Only when the Disconnection Rate would be equal to the Connection Rate, the architectures would then be able to make correct decisions about the expected loads (which takes place normally, in our case, at around 600th second). And in turn would carry the load closer to the Target Load beyond that point. With a sudden decrease in the Offered Load, the Carried Load will lag behind a time corresponding to the average time between the Connection and Disconnection of accepted service requests [13]. Figure 6.7 shows the graph representing the expected results.

In the current working of the simulator, the auctions in the CA and HA Architectures currently takes zero simulated seconds to carry. It seems that the results on Reactivity would be somewhat affected if an auction would take some time to be carried out proportional to the number of customers/providers taking part in an auction.

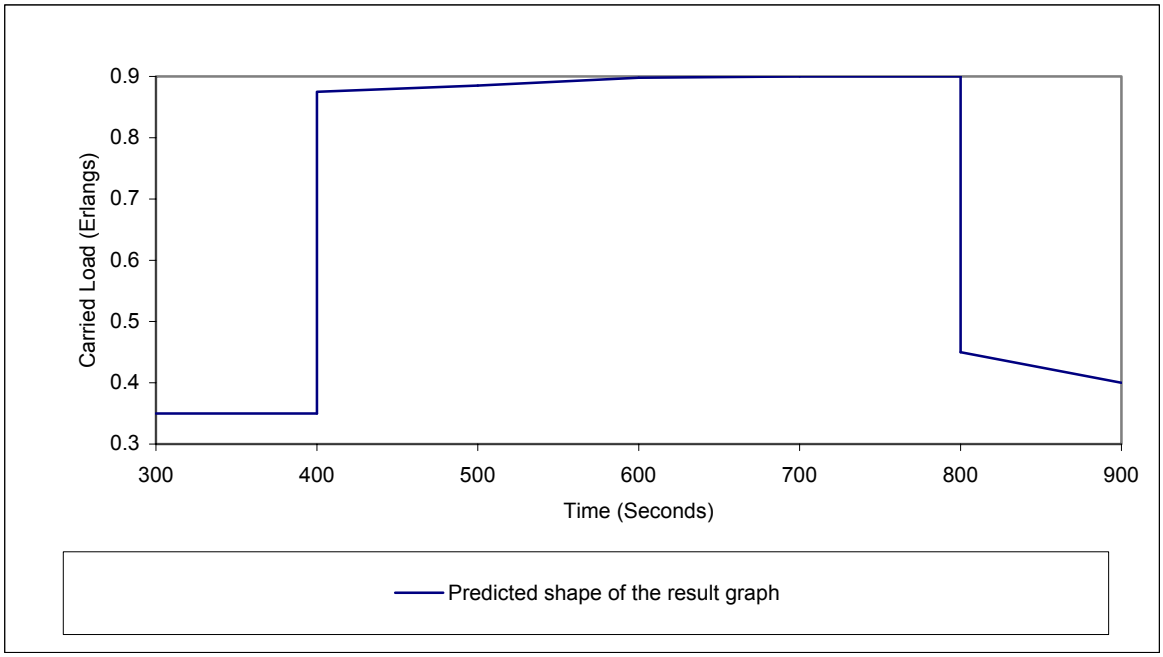


Fig. 6.7, The Expected Simulation Results for the synchronized peak simulations

Figures 6.7.1 – 6.7.28 represents the results from Simulations 4, 5 and 6 for all the architectures.

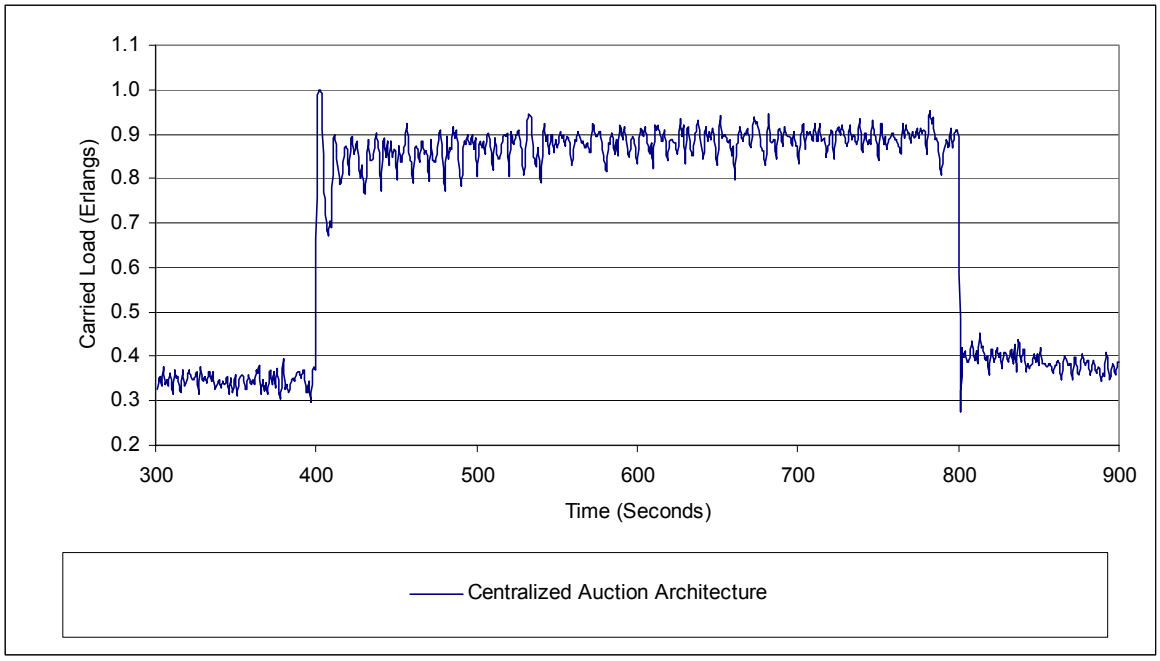


Fig. 6.7.1, Simulation-4 (32 SSPs), The Centralized Auction Architecture exposed to a synchronized peak

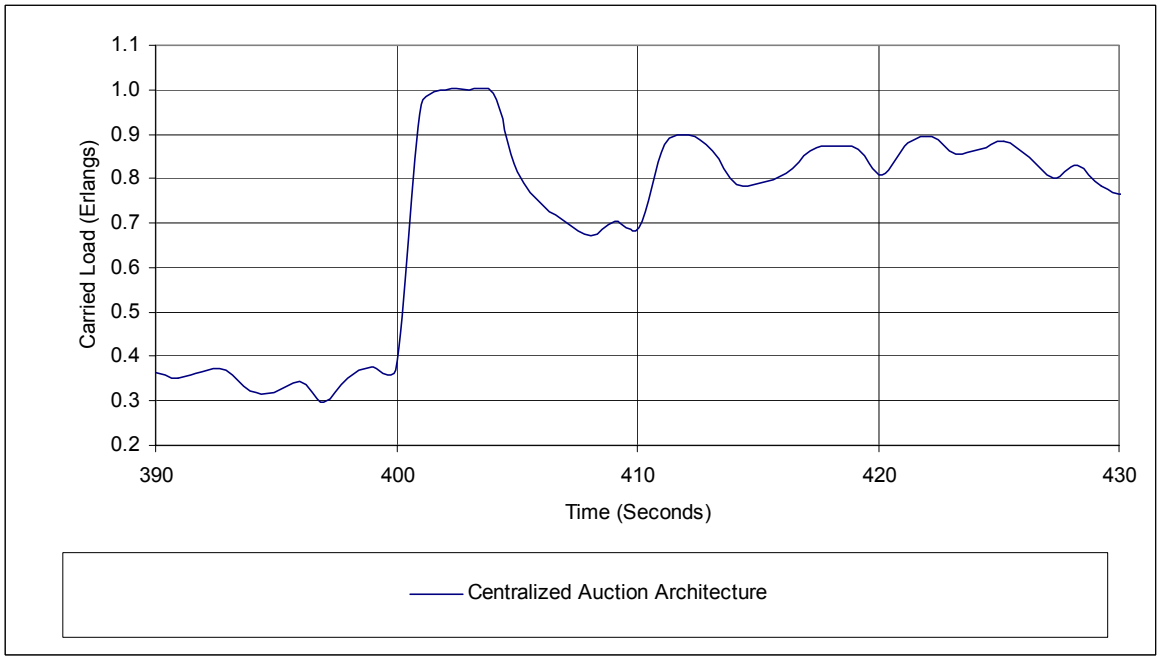


Fig. 6.7.2, Simulation-4 (32 SSPs), A magnified view of figure 6.7.1 between 390-430 seconds

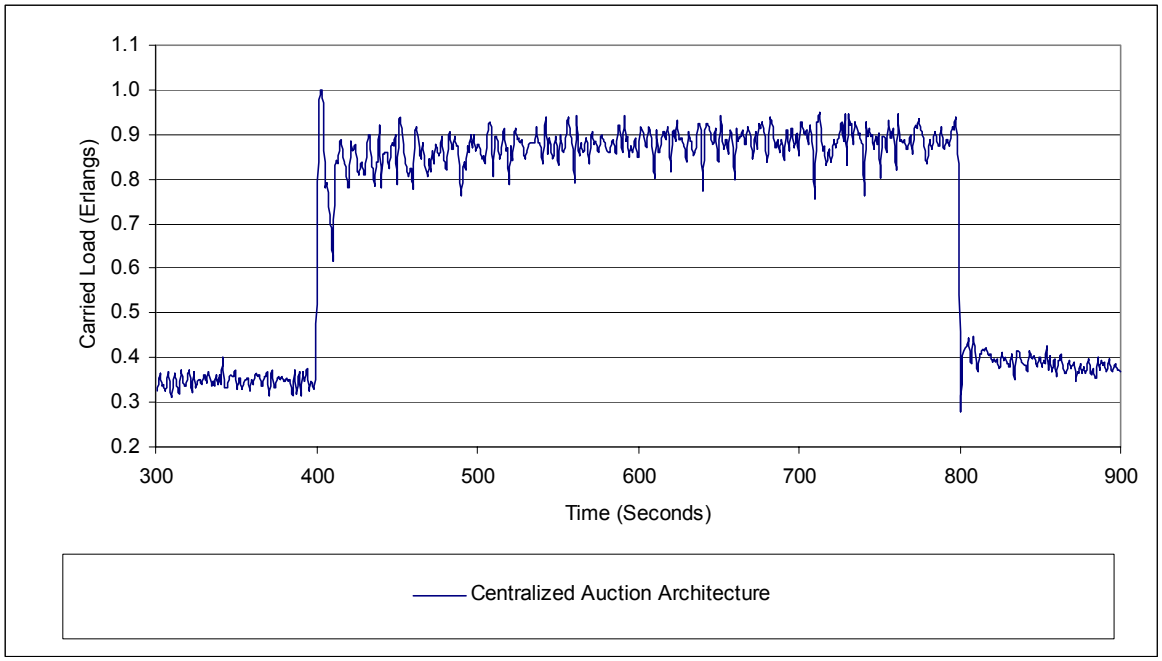


Fig. 6.7.3, Simulation-5 (64 SSPs), The Centralized Auction Architecture exposed to a synchronized peak

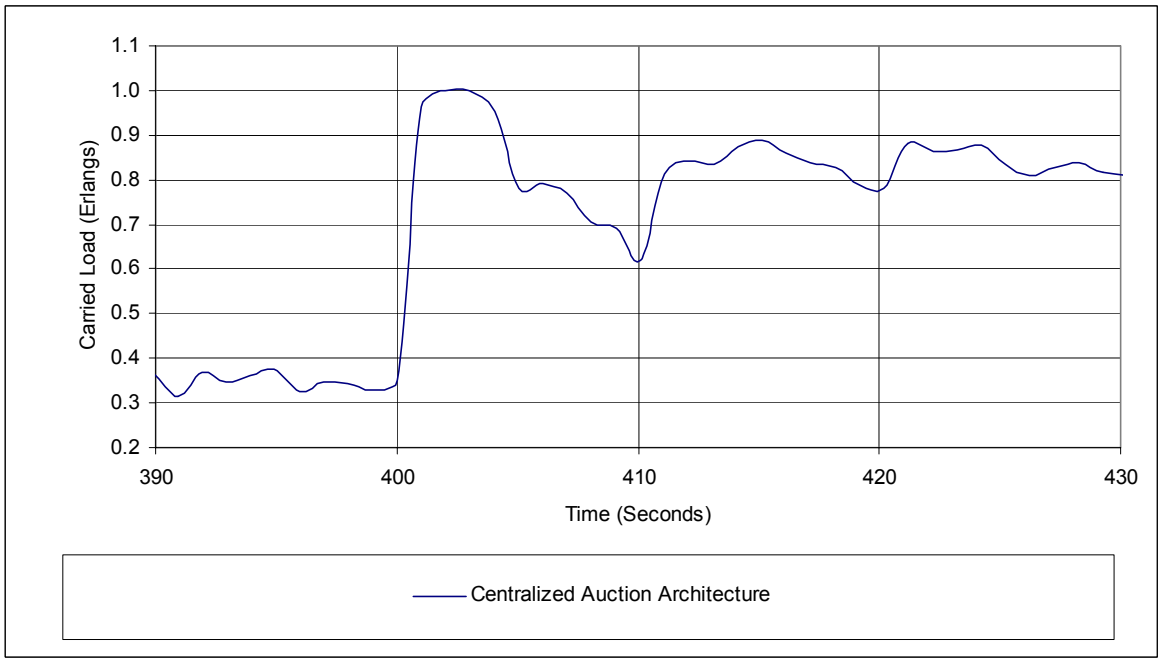


Fig. 6.7.4, Simulation-5 (64 SSPs), A magnified view of figure 6.7.3 between 390-430 seconds

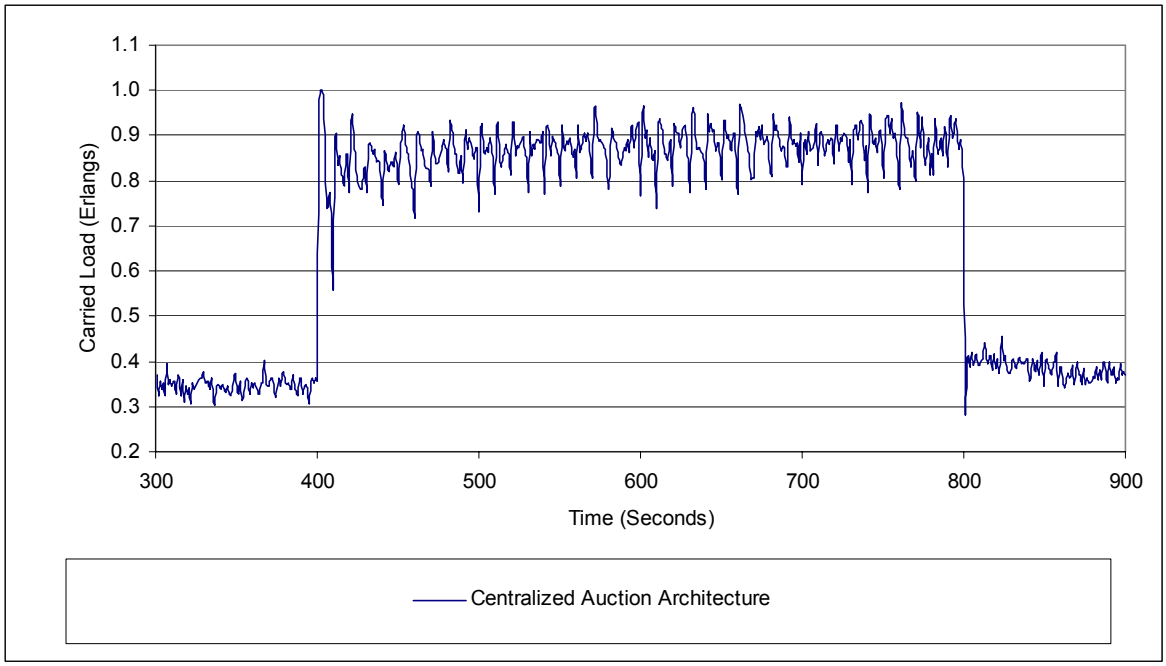


Fig. 6.7.5, Simulation-6 (128 SSPs), The Centralized Auction Architecture exposed to a synchronized peak

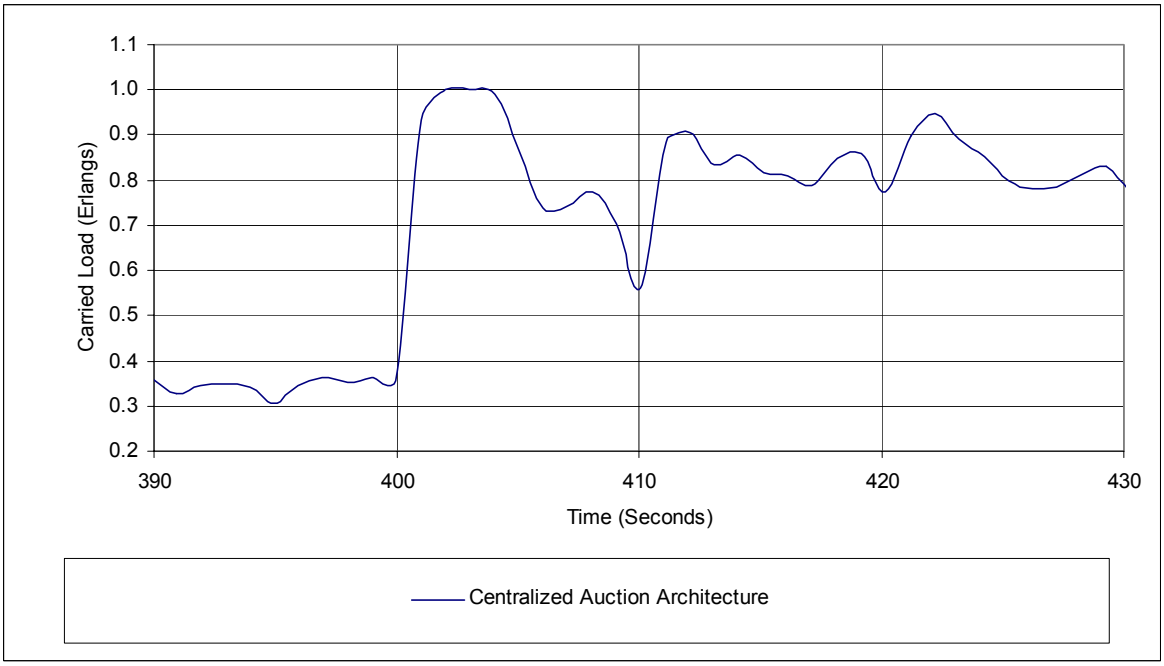


Fig. 6.7.6, Simulation-6 (128 SSPs), A magnified view of figure 6.7.5 between 390-430 seconds

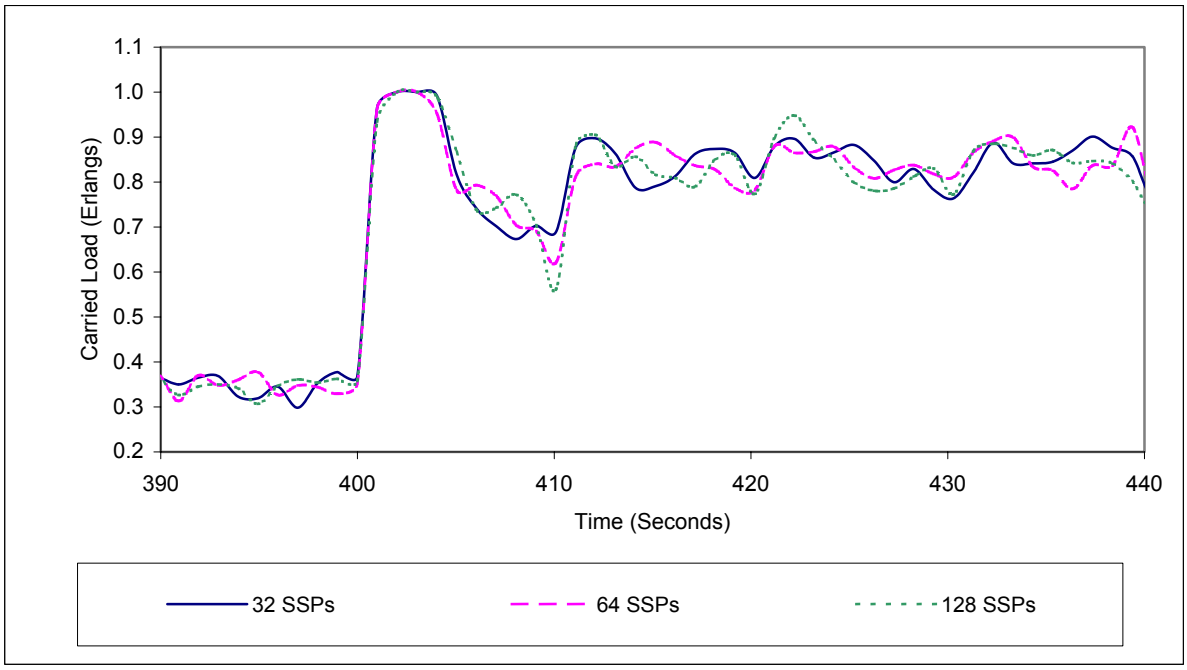


Fig. 6.7.7, Simulations 4-5-6, results on Reactivity for The Centralized Auction Architecture between 390-440 seconds

The Centralized Auction Architecture shows considerable alternations when the Offered Load is suddenly increased from 0.35 to 2.0 Erlangs (figures 6.7.1-6.7.6). The alternation is an affect caused by the combination of the auction interval time and the percent-thinning algorithm [13]. Just after the Offered Load is suddenly increased from 0.35 to 2.0 Erlangs, the percent-thinning algorithm basis its computations of the expected Offered Load on the last auction interval, where the Offered Load was 0.35 Erlangs. This results in overloading of the processing nodes in the initial part of the next auction interval (between 400-404 seconds), see figures 6.7.2, 6.7.4 and 6.7.6 where the Carried Load exceeds the Target Load showing the network is overloaded to the maximum. But in the middle of the auction interval the percent-thinning algorithm realizes that it is running short of the tokens too early, before the next auction. To avoid running out of the tokens to early in the current interval, the percent-thinning algorithm would increase the Call Rejection Rate during the last part of the current auction interval. As soon as the next auction is held the computations about the expected load are now based on the previous interval, where the load was 2.0 Erlangs. So the algorithm performs better since the Offered Load in the immediately preceding auction interval is the same as the current interval. So the CA architecture manages to carry the load closer to the Target Load better in the next interval.

It can be noted from figures 6.7.2, 6.7.4 and 6.7.6 that a drop in the Carried Load in the second half of the time interval between 400-410 seconds is relative to the Call Rejection Rate. Since in this part the architecture has to reject most of the service requests, it fails to carry the load closer to the Target Load. Since in the auction architectures number of tokens in the auctions would be same, irrespective of the number of SSPs. The Call Reject Rate would increase with an increase in the number of SSPs. And in turn would result in a drop in Carried Load relative to the number of SSPs, in this interval of time. Refer to figure 6.7.7

where the collective results from simulations 4-5-6 have been represented in a single graph, for the CA Architecture, between the interval 390-440 seconds.

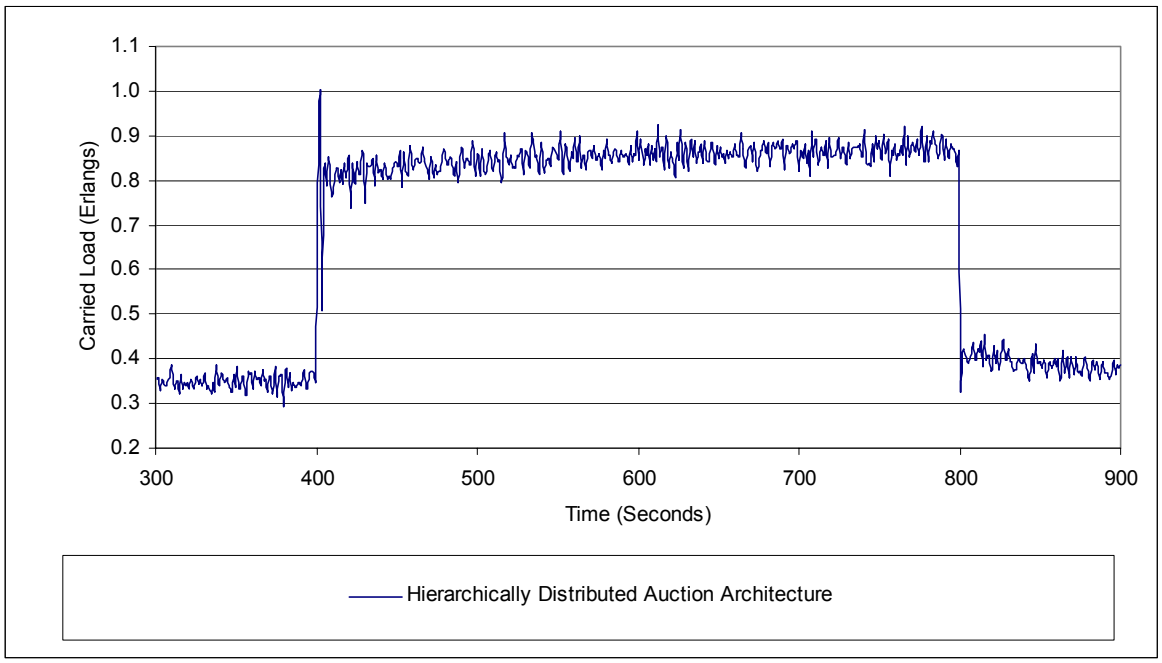


Fig. 6.7.8, Simulation-4 (32 SSPs), The Hierarchically Distributed Auction Architecture exposed to a synchronized peak

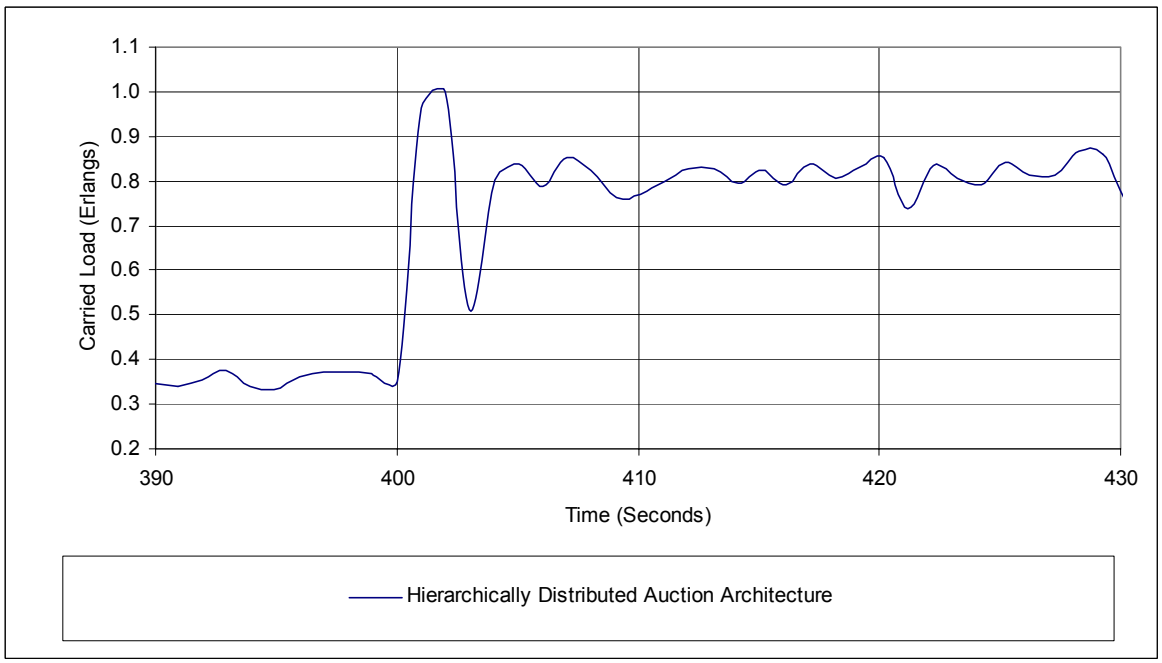


Fig. 6.7.9, Simulation-4 (32 SSPs), A magnified view of figure 6.7.8 between 390-430 seconds

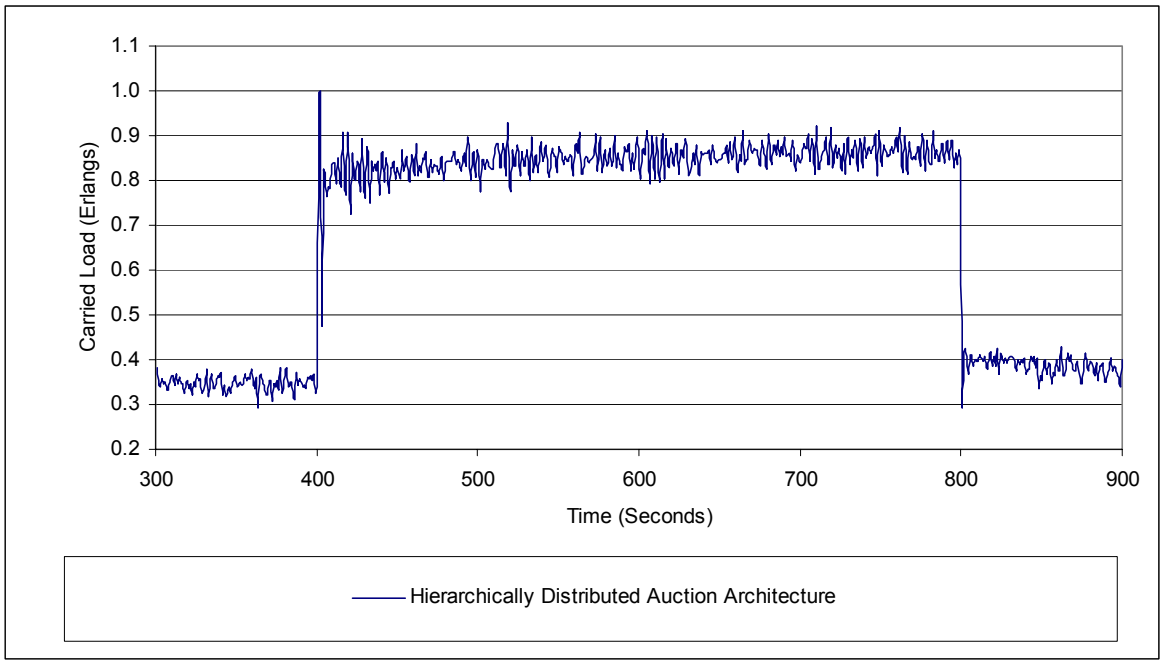


Fig. 6.7.10, Simulation-5 (64 SSPs), The Hierarchically Distributed Auction Architecture exposed to a synchronized peak

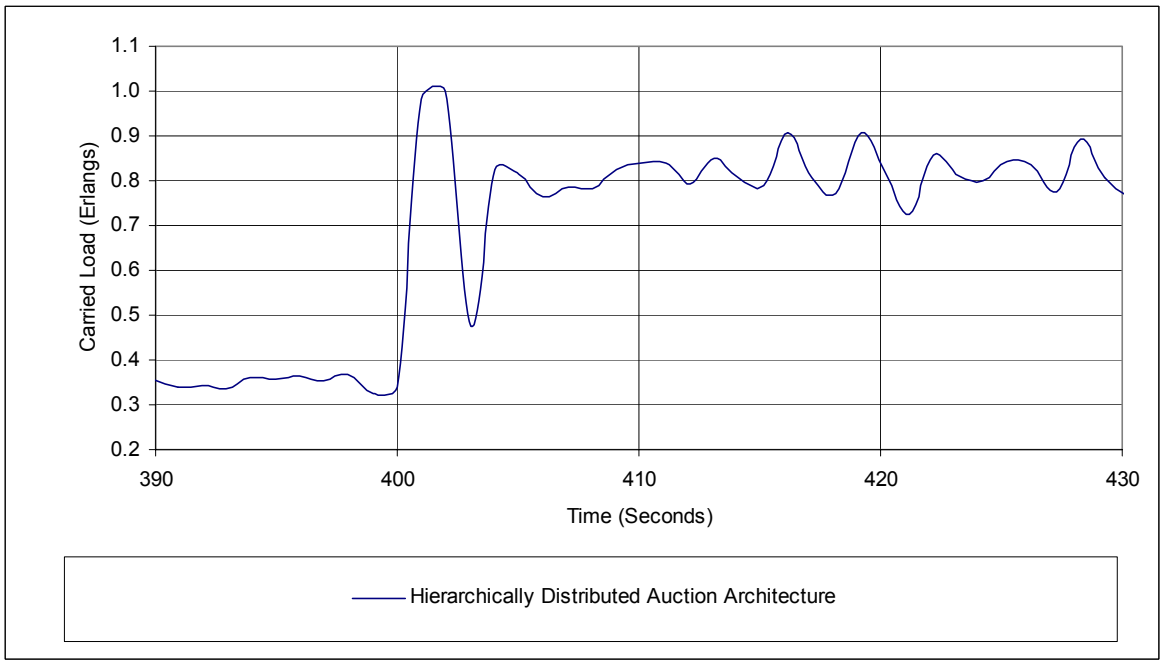


Fig. 6.7.11, Simulation-5 (64 SSPs), A magnified view of figure 6.7.10 between 390-430 seconds

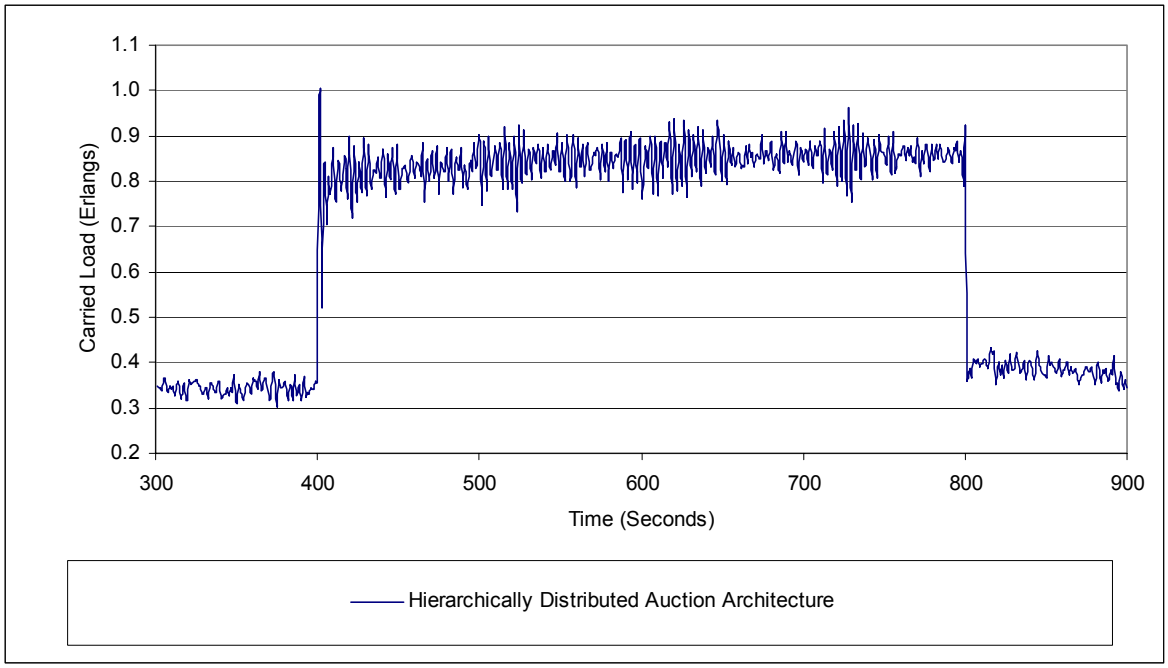


Fig. 6.7.12, Simulation-6 (128 SSPs), The Hierarchically Distributed Auction Architecture exposed to a synchronized peak

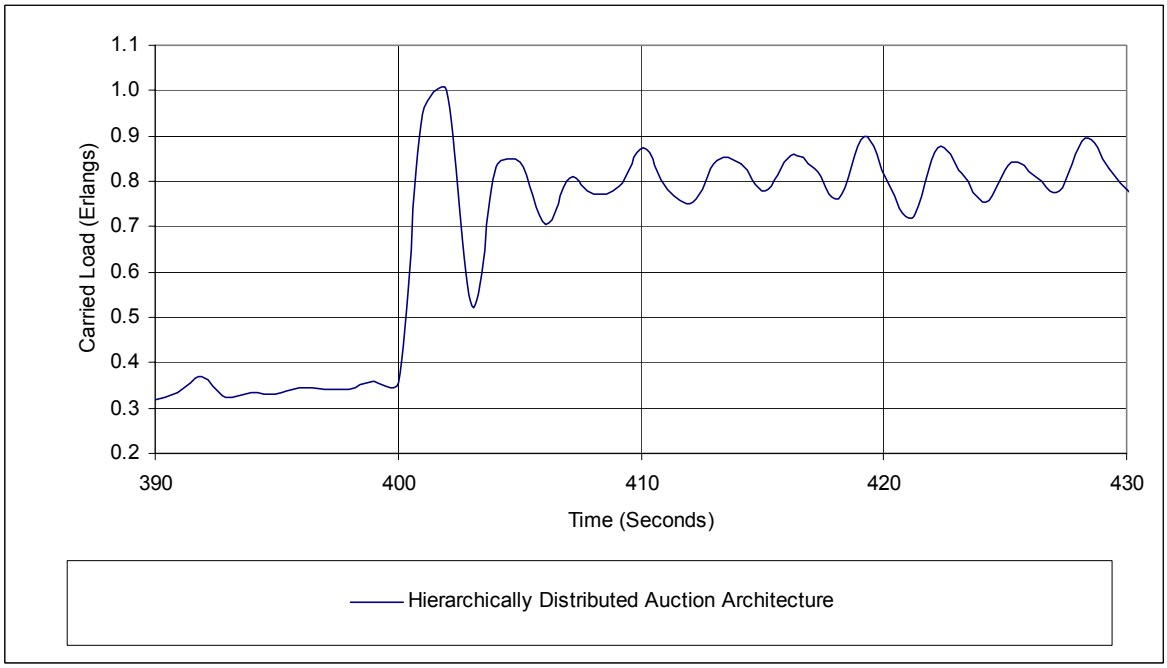


Fig. 6.7.13, Simulation-6 (128 SSPs), A magnified view of figure 6.7.12 between 390-430 seconds

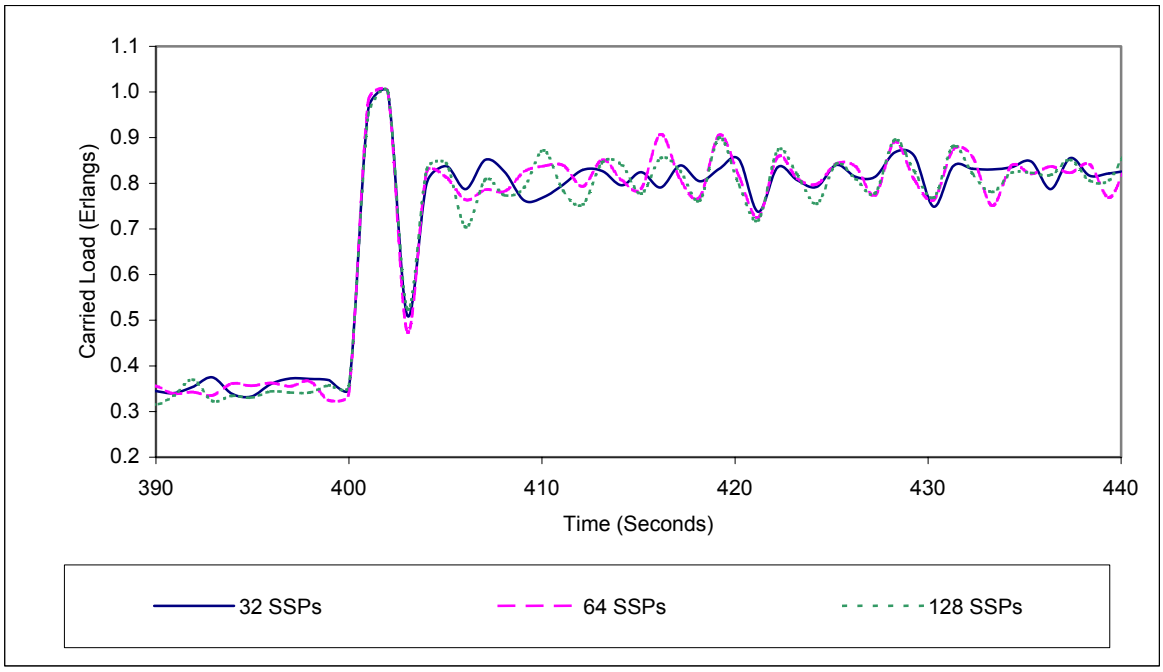


Fig. 6.7.14, Simulations 4-5-6, results on Reactivity for The Hierarchically Distributed Auction Architecture between 390-440 seconds

The simulation results for studying the Reactivity in the Hierarchically Distributed Auction Architecture has been depicted in figures 6.7.8-6.7.14. Though HA, like CA, carries the load at around 1.0 Erlangs, as a result of a sudden increase in the Offered Load to 2.0 Erlangs (in the initial part before the first Intermediate Auction). Yet the HA architecture adapts better to a sudden increase in the Offered Load compared to CA. The architecture stabilizes as soon as the first Intermediate Auction is held as can bee seen in the figures 6.7.9, 6.7.11 and 6.7.13. Irrespective of an increase in the network size, the Carried Load remains around 0.5 Erlangs in the worse case between the time interval 400-410 seconds. It is clear that the Intermediate auctions in the HA would help the architecture stabilize, depending on the time when the first Intermediate Auction takes place, earlier as compared to the CA Architecture. Refer to figure 6.7.14 where the collective results from simulations 4-5-6 have been represented in a single graph, for the HA Architecture, between the interval 390-440 seconds.

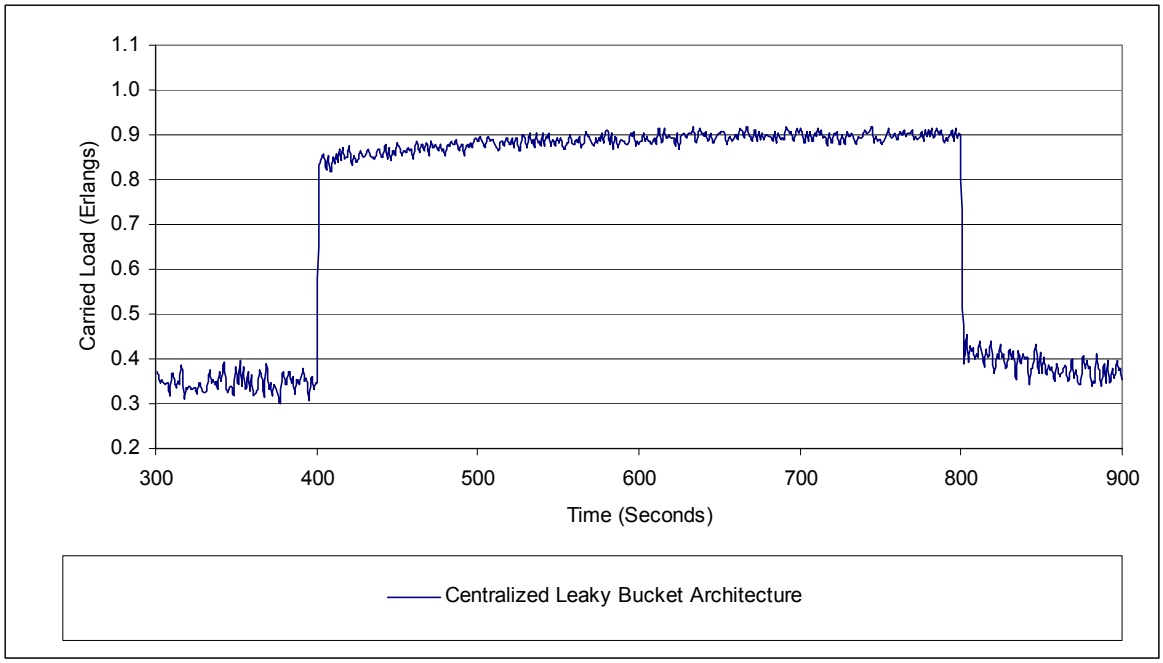


Fig. 6.7.15, Simulation-4 (32 SSPs), The Centralized Leaky Bucket Architecture exposed to a synchronized peak

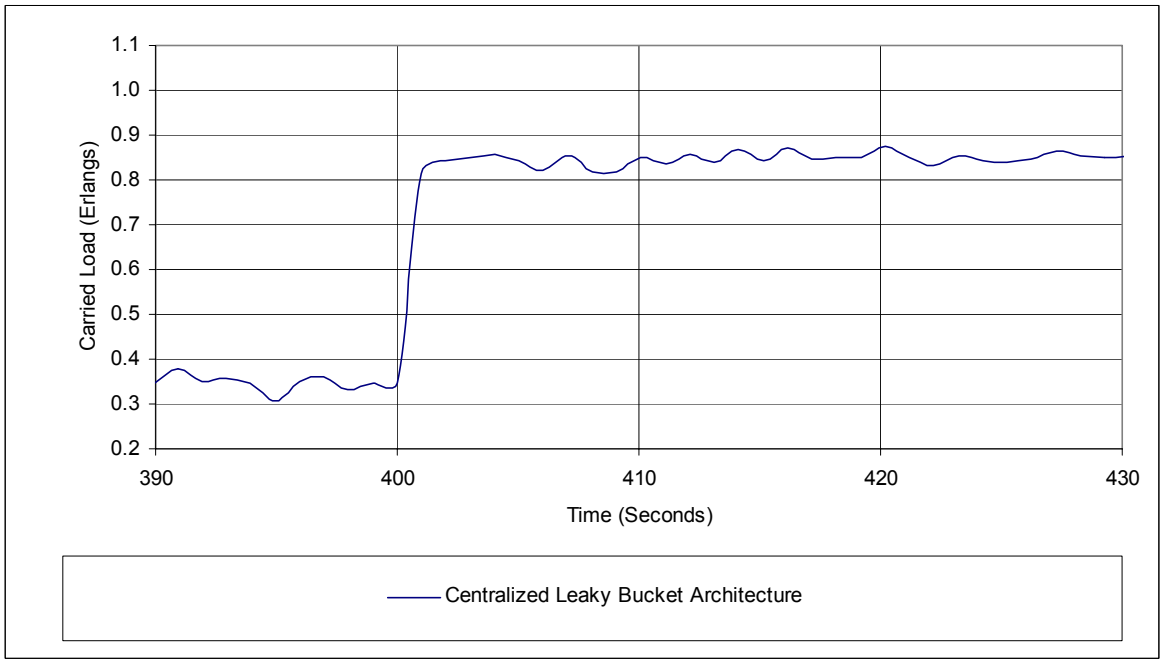


Fig. 6.7.16, Simulation-4 (32 SSPs), A magnified view of figure 6.7.15 between 390-430 seconds

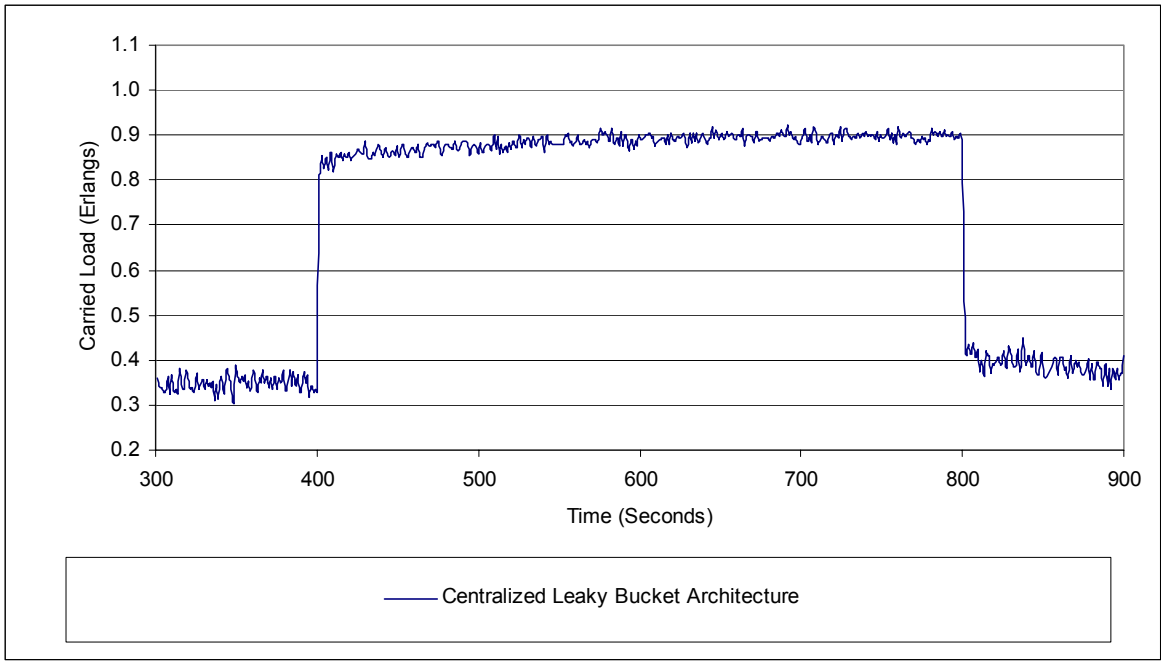


Fig. 6.7.17, Simulation-5 (64 SSPs), The Centralized Leaky Bucket Architecture exposed to a synchronized peak

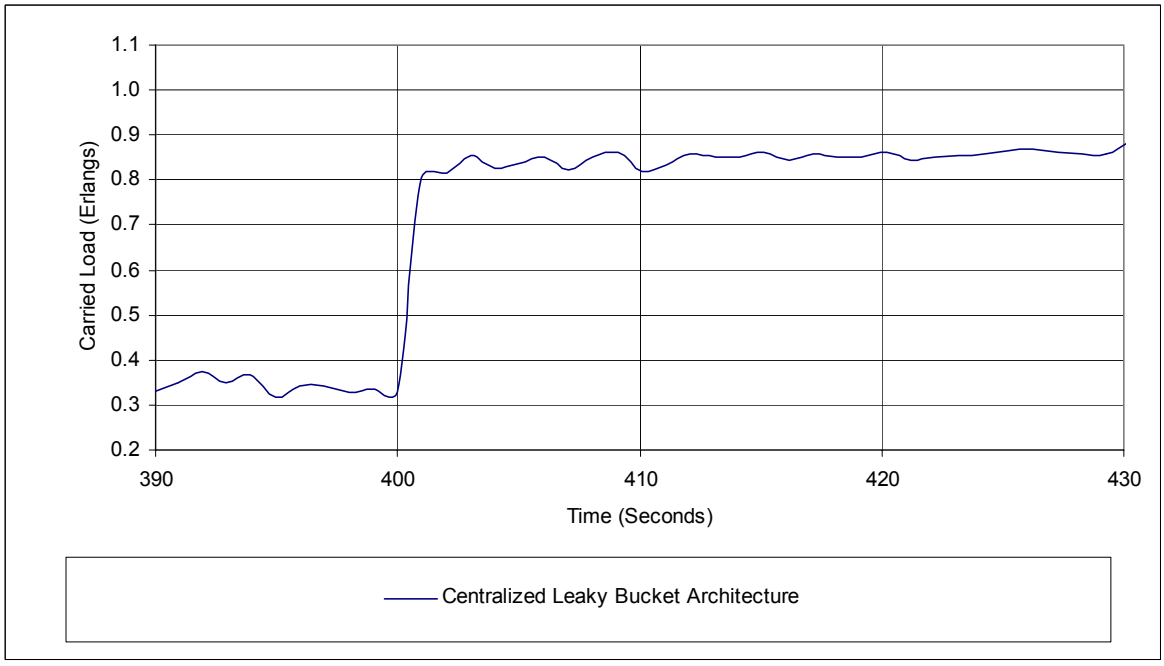


Fig. 6.7.18, Simulation-5 (64 SSPs), A magnified view of figure 6.7.17 between 390-430 seconds

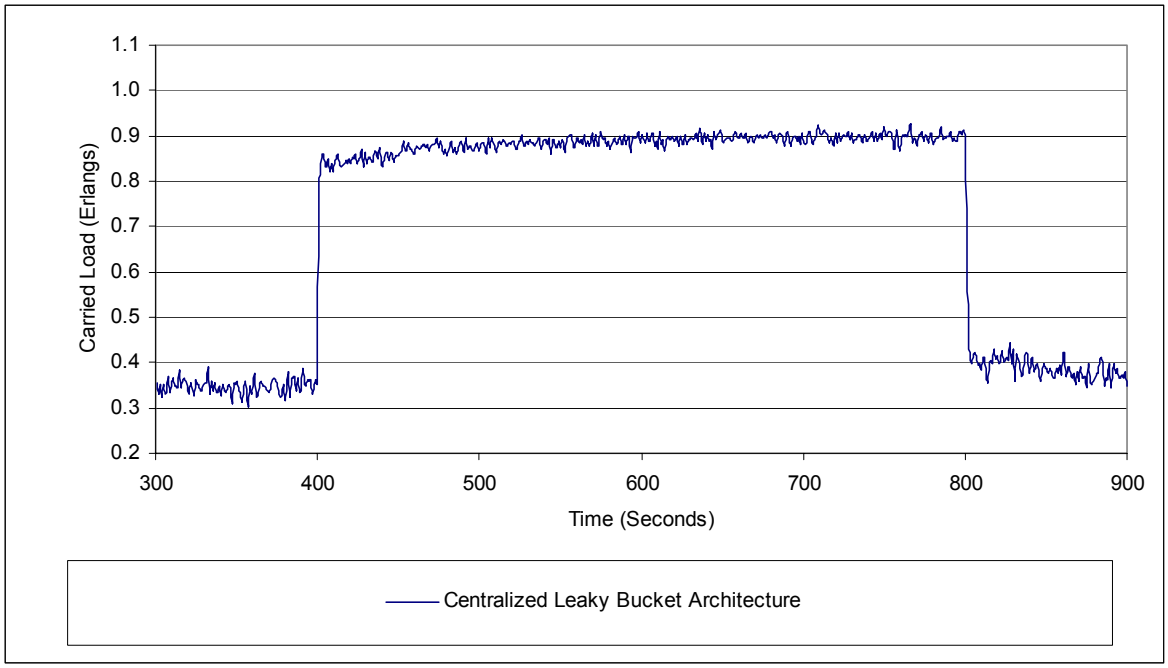


Fig. 6.7.19, Simulation-6 (128 SSPs), The Centralized Leaky Bucket Architecture exposed to a synchronized peak

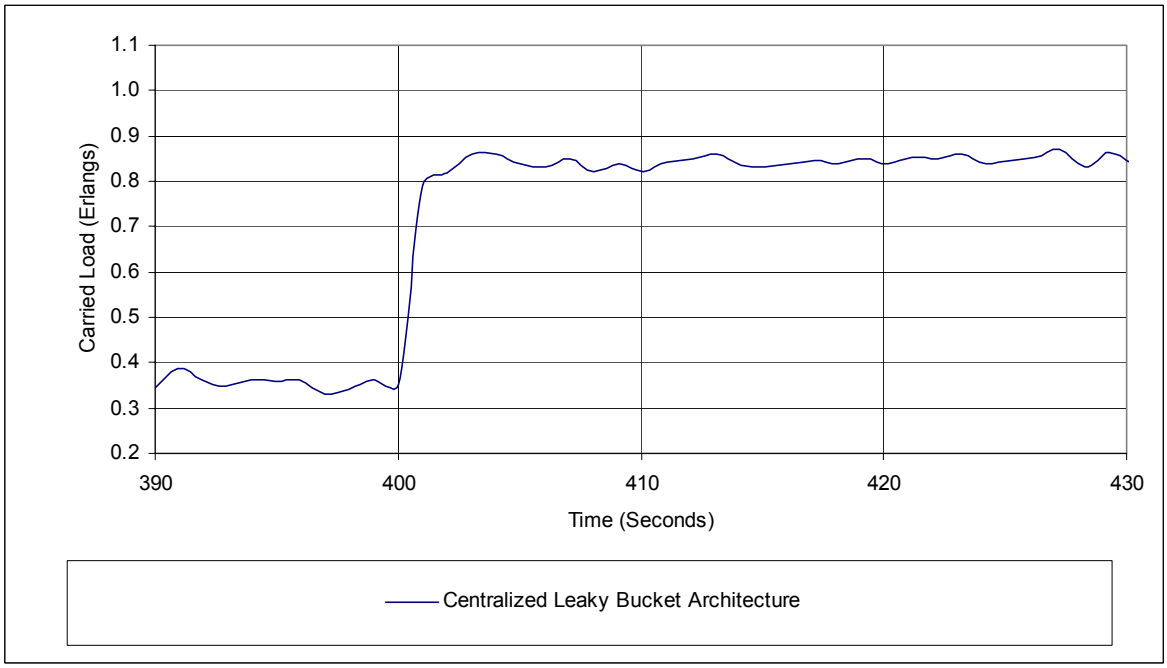


Fig. 6.7.20, Simulation-6 (128 SSPs), A magnified view of figure 6.7.19 between 390-430 seconds

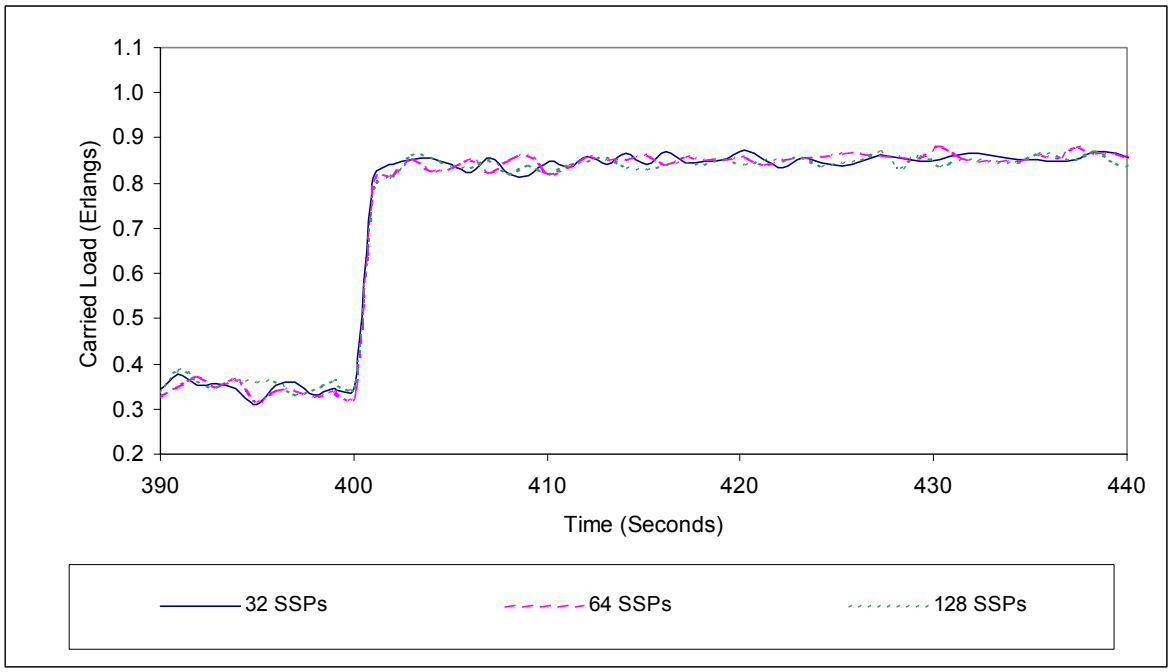


Fig. 6.7.21, Simulations 4-5-6, results on Reactivity for The Centralized Leaky Bucket Architecture between 390-440 seconds

Refer to the figures 6.7.15-6.7.21, representing the results on Reactivity for the CLB Architecture. It is obvious from the results that CLB produces very less variations and alternations, irrespective of the sudden increase/decrease in the Offered Load and the number of SSPs. The architecture best manages the Offered Load and avoids exceeding the Carried Load than the Target Load at any time instant. Since the CLB Architecture is synchronous, and receives the service requests *continuously*, not waiting for any event to occur for initiating the resource allocation mechanism. As soon as the Offered Load increases the Target load, the architecture manages to accommodate the Offered Load accordingly. The reason for the better performance of the CLB Architecture is that irrespective of the Offered Load and the number of SSPs, it simply rejects the incoming requests which it cannot accommodate as the Offered Load is increased. Which results in carrying the Offered Load closest to the Target Load. Refer to figure 6.7.21 where the collective results from simulations 4-5-6 have been represented in a single graph, for the CLB Architecture, between the interval 390-440 seconds.

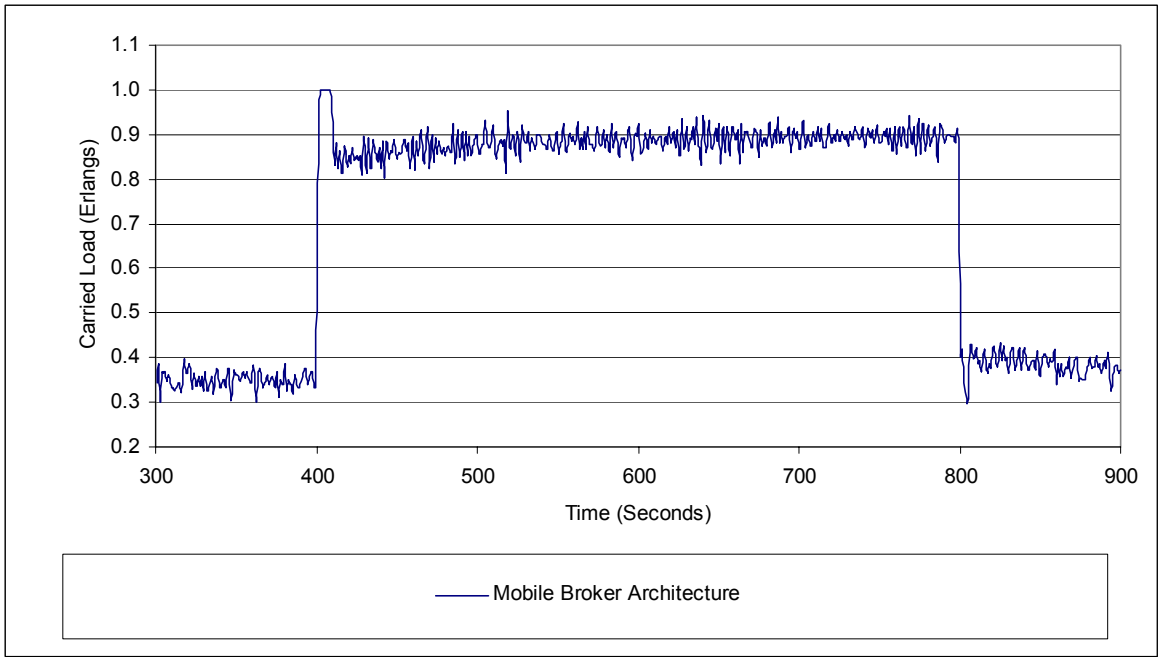


Fig. 6.7.22, Simulation-4 (32 SSPs), The Mobile Broker Architecture exposed to a synchronized peak

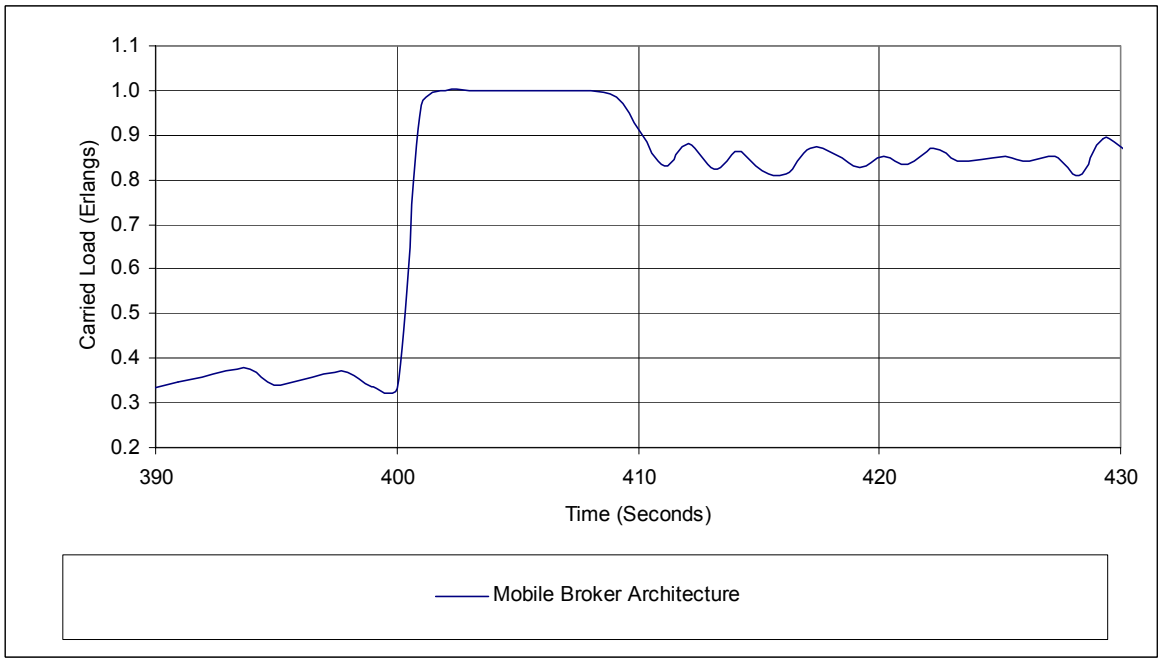


Fig. 6.7.23, Simulation-4 (32 SSPs), A magnified view of figure 6.7.22 between 390-430 seconds

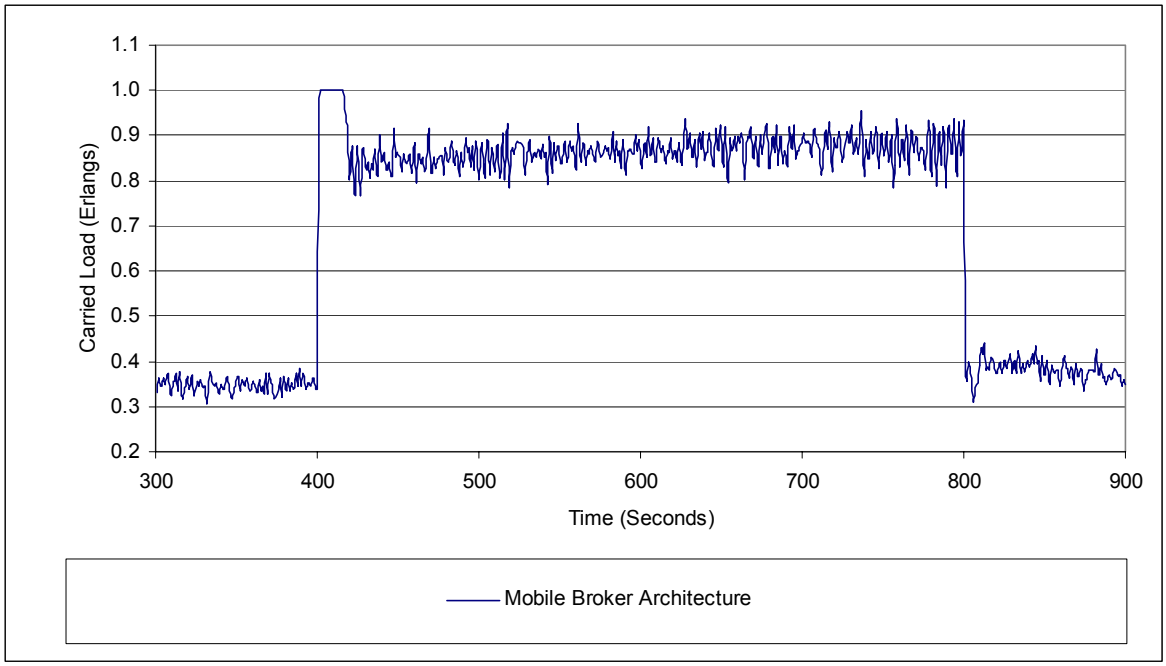


Fig. 6.7.24, Simulation-5 (64 SSPs), The Mobile Broker Architecture exposed to a synchronized peak

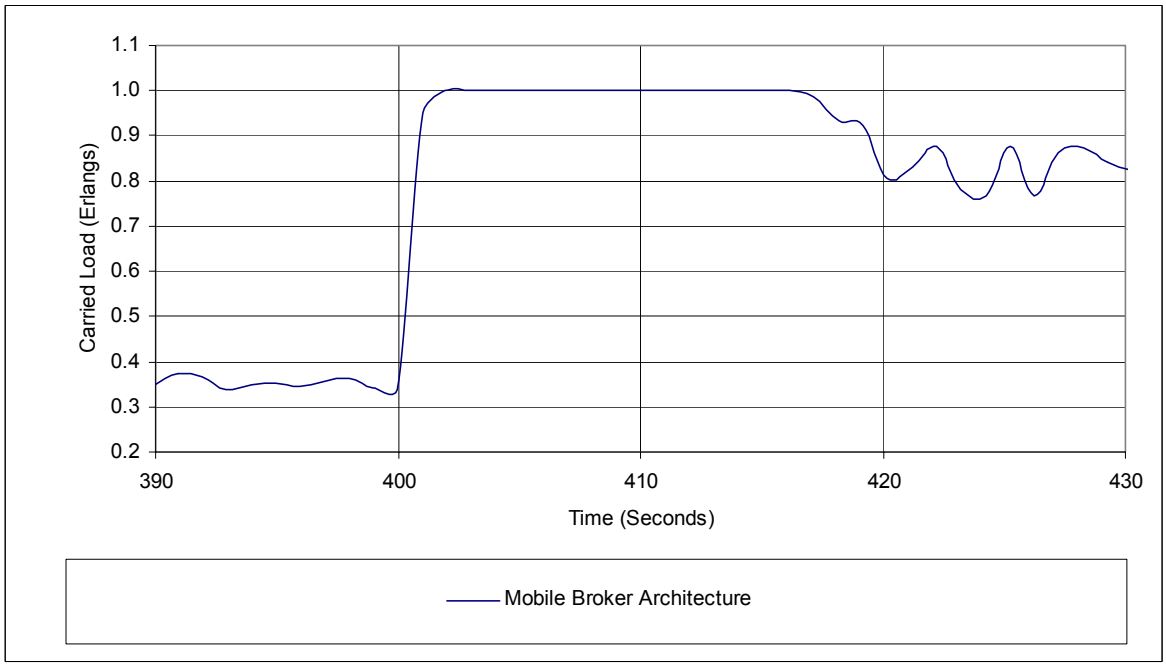


Fig. 6.7.25, Simulation-5 (64 SSPs), A magnified view of figure 6.7.24 between 390-430 seconds

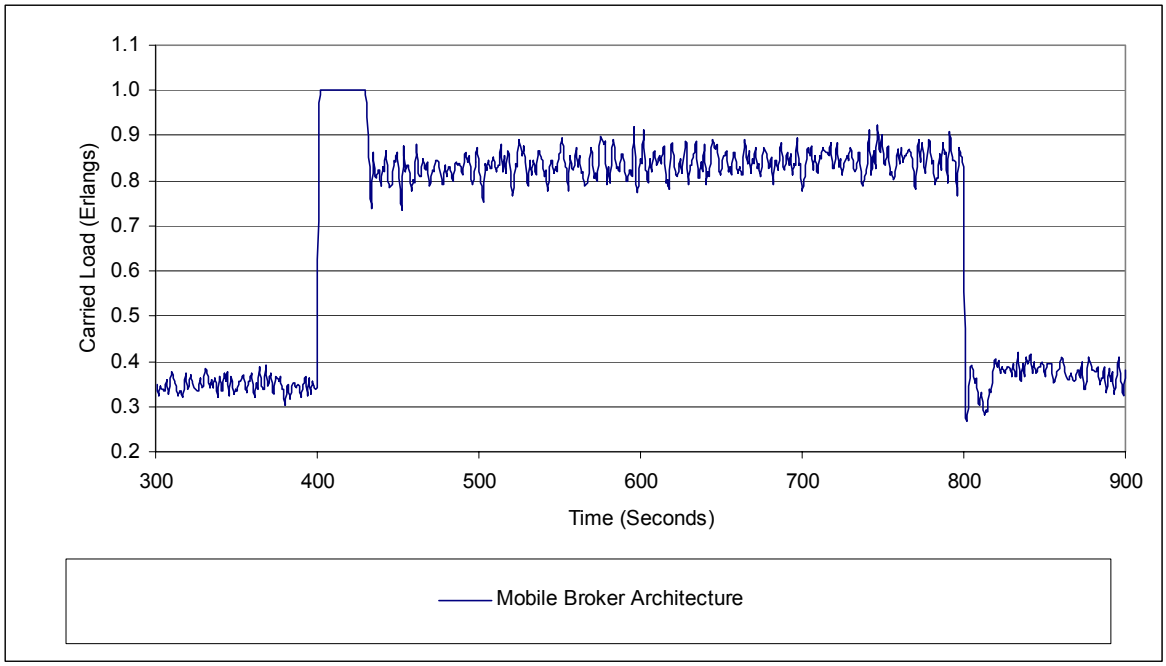


Fig. 6.7.26, Simulation-6 (128 SSPs), The Mobile Broker Architecture exposed to a synchronized peak

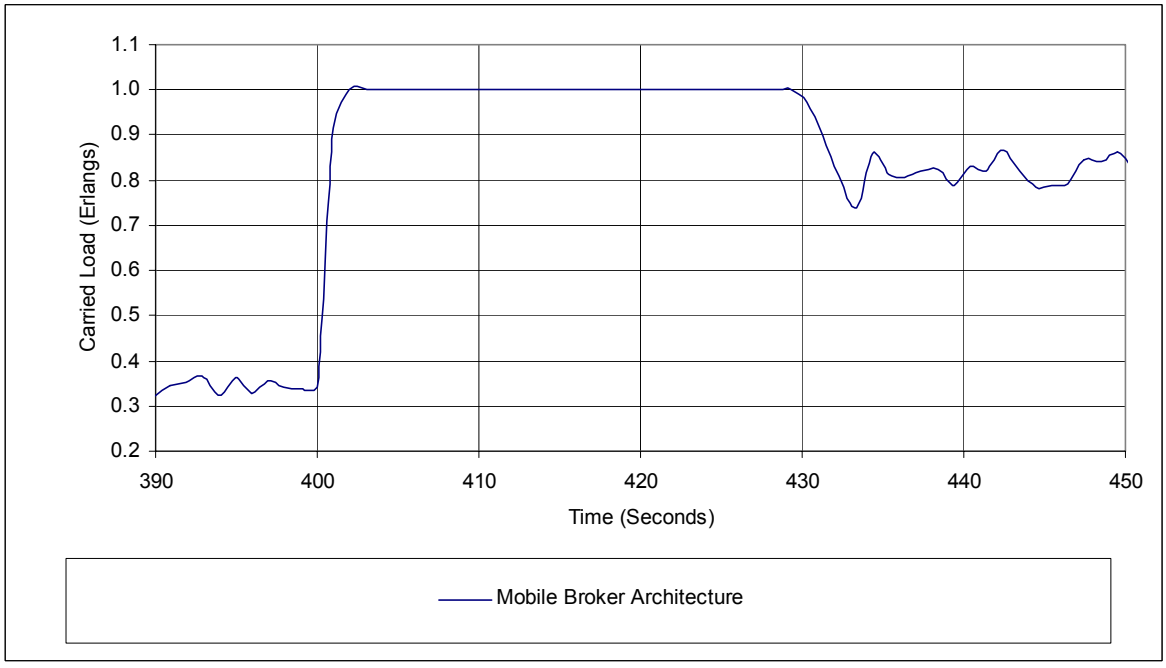
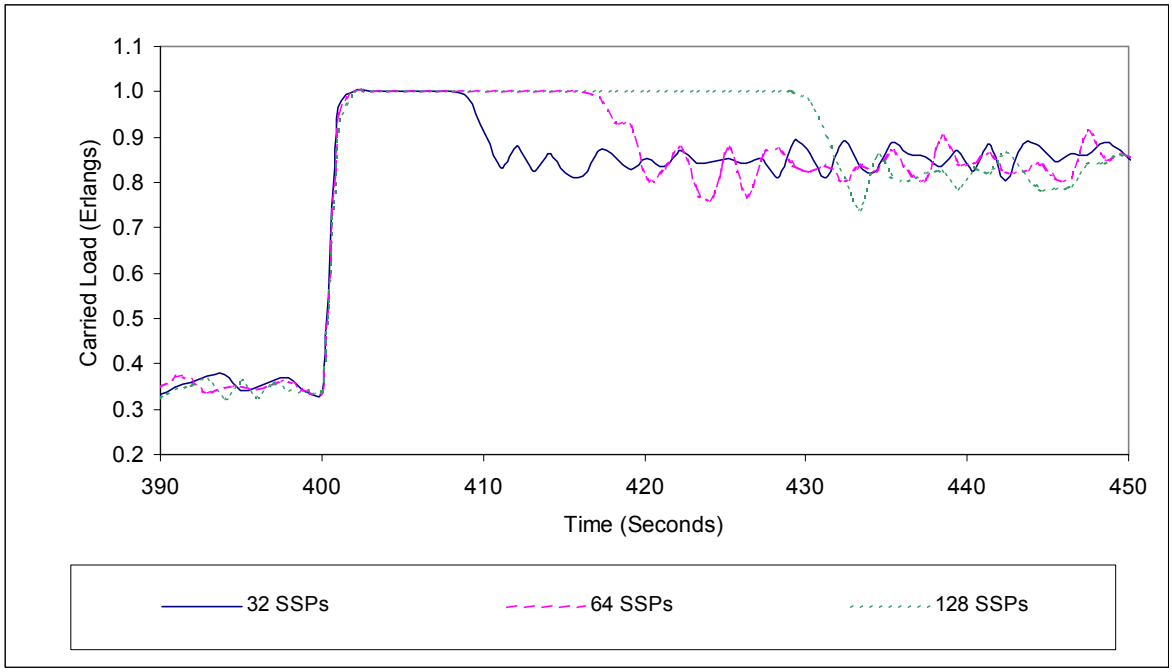


Fig. 6.7.27, Simulation-6 (128 SSPs), A magnified view of figure 6.7.26 between 390-450 seconds



,

Fig. 6.7.28, Simulations 4-5-6, results on Reactivity for The Mobile Broker Architecture between 390-450 seconds

The results on Reactivity for the Mobile Broker Architecture are depicted in the figures 6.7.22-6.7.28. Figures 6.7.23, 6.7.25 and 6.7.27 explicitly clarify the behaviour of the MB architecture at the synchronized peak. Which shows that when the synchronized peak is offered, the MB architecture gets heavily overloaded as compared to all the other architectures. According to the typical work mechanism of the MB Architecture, a sudden in the Offered Load would make Brokers sell too much capacity to the Allocators during the first route of the lap. Since we know that a Broker would spend 0.2 seconds at each Allocator (in all simulations). In case where the number of Allocators is 32, every Broker would complete one route in less than 2.0 seconds. It takes about 5.0 seconds for the SCP processors to finish the job processing as a result of this sudden increase in the Offered Load [13]. This argument has however been verified in the section on 'Processing Capacity'. Refer to figure 6.2.13, and note the Queue Lengths of jobs when the Offered Load is 2.0 Erlangs, for Simulation-1. Which shows that the Average Queue Length is around 60 units. So for a total of approximately 7.0 seconds the system is heavily overloaded, see figure 6.7.22 (for Simulation 4).

In the section on Computational Complexity, it is determined that the complexity in the MB Architecture lies in the route length of the Brokers. The greater the route length, more Allocators visited in turn greater would be the computational complexity. As the Broker routes have increased and more Allocators are visited (Simulations 5 and 6), the more time it takes for a Broker to finish a lap and more computations to be performed, more distribution of jobs among the same number of SCPs and more time delays for the architecture to stabilize and carry the load closer to the Target Load. *In fact the duration of the overload situation (immediately after the Offered Load is increased) increases proportionally with an increase in the number of Allocators.*

CHAPTER 7

7 Conclusions and Future Work

The four Multi-Agent architectures proposed for load control management in Intelligent Networks have been examined, evaluated and compared in terms of *scalability*. It has been observed that *no single architecture excels in all aspects of scalability*. Each having its own particular advantages and drawbacks. Though observations and conclusions have been discussed in respective parts of the work, some of the general conclusions deduced from the experiments performed are,

In terms of the *Utilization of Resources*, Centralized Architectures (CA and CLB), perform better than Distributed Architectures (HA and MB), with an increase in the network size and Offered Loads.

In terms of *Communication Delays* in the network, Centralized Leaky Bucket is *least* responsive and has the *highest* messaging delays of all the architectures, generally after the Offered Load exceeds the Target Load. However, *synchronous* architectures (i.e. Centralized Auction and Hierarchically Distributed Auction architecture) perform better than *asynchronous* architectures (i.e. Centralized Leaky Bucket and Mobile Broker architectures) in terms of response times and messaging delays, at all Offered Loads, as the network grows in size and at varied Offered Loads.

It has been noted that the increase in the network nodes (SSPs) and Offered Loads had no significant impact on the total number of Successful Jobs, Accepted and Rejected Calls by any of the architectures. In terms of the *Required Bandwidth*, CLB asks for high bandwidth requirements as compared to CA, HA and MB Architectures, as the network size grows.

The MB Architecture is not as good at *load balancing* as the other architectures with an increase in the network size.

The results from the experiments on *Reactivity* have revealed that the CLB Architecture is the most stable in terms of carrying Offered Loads close enough to the Target Load with an instant increase or decrease in the Offered Loads, at different network sizes. The MB Architecture is, however, highly susceptible to instant increases in Offered Loads and the duration of the overload situation in the MB Architecture (immediately after the Offered Load is increased) increases proportionally with an increase in the number of Allocators.

Since the current working of the simulator takes relatively no time in carrying auctions. One of the things that still needs to be evaluated is to modify the auction mechanism in the auction Architectures i.e. Centralized Auction and Hierarchically Distributed Auction architectures so that the auction mechanism takes some time, proportional to the number of bidders in an auction. Accordingly, messaging related to resource allocation alone, could be modified so that it becomes relative to the increase in the network components. The impact of these modifications could then be studied, in terms of scalability.

In the current investigations, the number of SCPs was kept constant regardless of the number of SSPs. It would be interesting to determine that which setting of the number of SCPs and SSPs results in optimal performance, for every architecture, if we intend to increase the network size. This would provide us a measure on the required number of SCPs for a particular number of SSPs for a particular network configuration.

References

[1] Arvidsson, A.; Jennings, B.; Angelin, L.; "On the Use of Agent Technology for IN Load Control, with an example Intelligent Network (IN) 'Market-based' mechanism"; *In Proceedings of the 16th International Teletraffic Congress; Elsevier Science, 1999.*

[2] Carlsson, B.; Davidsson, P.; Johansson, S. J.; Ohlin, M.; "Using Mobile Agents for IN Load Control"; *In Proceedings of Intelligent Networks 2000*, *IEEE 2000, Page(s) 161-169.*

[3] Johansson, S. J.; Davidsson, P.; Carlsson, B.; "Coordination Models for Dynamic Resource Allocation"; In A. Porto and G. –C.Roman, editors, Coordination Languages and Models, Volume 1906 of Lecture Notes in Computer Science. Page(s) 182-197, Springer Verlag, 2000. *Proceedings of the 4th International Conference on Coordination.*

[4] Johansson, S. J.; Davidsson, P. Kristell, M.; "Four Multi-Agent Architectures for Intelligent Network Load Management"; *Mobile Agents for Telecommunication Applications*, *LNCS Vol. 2521, Springer Verlag, 2002*.

[5] The International Engineering Consortium (IEC); *Intelligent Network*; http://www.iec.org/online/tutorials/in/index.html.

[6] "Opportunities in Network Control for the Coming Decade"; *Advanced Intelligent Networks*; May 1998; http://www.gii.co.jp/english/ci2749_itelligentnetworks_summary.html.

[7] http://searchnetworking.techtarget.com/sDefinition/0,,sid7_gci213769, 00.html.

[8] Amir-Ebrahimi, I.; Frech, D.A.; "Call processing data in Intelligent Network elements and traditional Switching Systems"; *Global Telecommunications Conference, 1994, GLOBECOM '94 Communications, The Global Bridge', IEEE, Volume 2, 28 Nov- 2 Dec 1994, Page(s): 1248 –1252.*

[9] Tsun-Chien Chiang; Douglas, J.; Gurbani, V.K.; Montgomery, W.A.; Opdyke, W.F.; Reddy, J.; Vemuri, K.; "IN Services for Converged (Internet) Telephony"; *IEEE Communications Magazine*, *Volume: 38, Issue: 6, Jun 2000, Page(s): 108 –115.*

[10] Finkelstein, M.; Garrahan, J.; Shrader, D.; Weber, G.; "The Future of the Intelligent Network"; *IEEE Communications Magazine*, *Volume: 38, Issue: 6, Jun 2000.*

[11] Patel, A.; Prouskas, K.; Barria, J.; Pitt, J.; "A Computational Economy for IN Load Control Using a Multi-Agent System"; *Journal of Network and Systems Management, Vol. 8, No. 3, 2000.*

[12] Davidsson, P.; Johansson, S.; "Evaluating Multi-Agent System Architectures: A Case study concerning Dynamic Resource Allocation"; *Engineering Societies in the Agents World III*, *LNAI Vol. 2577, Springer, 2003.*

[13] Kristell, M.; "Four Multi-Agent Architectures for Intelligent Network Load Control Management"; *Masters' Thesis; Blekinge Institute of Technology, Sweden, 2002*.

[14] Atoui, M.; “Performance measurements of the SS7/Intelligent Network”; *Military Communications Conference, 1991. MILCOM ’91, Conference Record, ‘Military Communications in a Changing World’, IEEE, 4-7 Nov. 1991; Page(s): 774-778, Vol. 2.*

[15] Law, D. R.; “Scalable means more than more: A Unifying Definition of Simulation Scalability”; *Proceedings of the 1998 Winter Simulation Conference.*