Managing network loads with agent technology

Martin Kristell

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

This thesis is submitted to the Department of Software Engineering and Computer Science at Blekinge Institute of Technology in partial fulfilment of the requirements for the degree of Master of Science in Software Engineering. The thesis is equivalent to 20 weeks of full time studies. All material in this thesis which is not my own work has been identified and no material is included for which a degree has previously been conferred.

Contact Information
Author:
Martin Kristell
Address: Sjuhallavägen 66, 370 24 Nättraby, Sweden

University advisors:
Prof. Paul Davidsson
Ph.D. Stefan Johansson
Department of Software Engineering and Computer Science

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

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

ABSTRACT

The objective for this thesis is to implement and compare four multi-agent architectures proposed to manage utilisation levels in distributed computing. The main part of this is to find and analyse the attributes that describe how the architectures differ from each other and make them excel in different contexts. The intelligent network concept from the telecommunication industry is used as sample application for the empirical examinations.

Keywords: dynamic resource allocation, load balancing, congestion control

Contents

List of Figures

Chapter 1

Introduction

The main objective for this thesis is to implement and compare four multi-agent architectures proposed to manage utilisation levels in distributed computing. One part of this is to find and analyse the attributes that describe how the architectures differ from each other and make them excel in different contexts. The Intelligent Network concept[1] from the telecommunication industry is used as sample application for the empirical examinations.

The fundamental problem these architectures are proposed to cope with is that; When utilisation of the central processing unit in an unmanaged computer system (no matter if it is a stand-alone personal computer or an Intelligent Network) reaches a certain level, there will be problems in form of e.g. delays, service denials or in worst case a system crash. To prevent such unacceptable behaviour, the level of utilisation must be managed.

The needs for computational resources often come in bursts. That gives that it is unwise to use such a powerful system that it will handle the load peaks without problem, most likely a system that will manage to handle the average load level is a better choice. The latter means that you must be able to manage the utilisation level to prevent your system from uncontrolled behaviour during peaks in load. A telephone network operator may for example want to reserve resources to ensure that it will always be possible to make emergency calls to 112 even when the network is overloaded.

1.1 The Intelligent Network problem

Intelligent Network is a term used by the telephone industry to describe services such as e.g. "Call forwarding on no reply", "Call-back" and "Wake up call". As these services rely on centralised computing resources, these services are vulnerable to traffic peaks. And because the use of these services are often synchronised, initiated by television programs for voting or lottery purposes, the requests often come in bursts. The algorithm

---

[1]Please refer to [8] for a definition of Intelligent Network

that manages the network load must be able to cut off the peaks without annoying the telephone users, and it must be able to utilise the resources as close to the system's upper limit as possible without exceeding it. In a commercial telecommunication network, it is obvious that the operator wants no spare bandwidth, he wants to be paid for every piece of resource he owns.

The Intelligent Network load control problem has been studied by Arvidsson, Davidsson, Johansson and Carlsson, their work are presented in the articles [1, 2, 3]. They have in detail described and proposed four different multi-agent architectures to address the Intelligent Network load control problem. With help from a discrete event simulator, constructed by Arvidsson et al. and implemented by Ericsson AB, Davidsson, Johansson and Carlsson have so far started to evaluate and validate two of these four architectures. The continuum of these studies will be the subject for this thesis.

1.2   Research questions

The research questions this thesis will try to answer are

   – How do the proposed architectures manage compared to each other in terms of:

- ability and speed in adapting to changes in the offered load

- ability to keep the carried load close to the target load

- ability to balance the load equally between the providers

- how they manage to handle imbalanced and moving load

- how the connection response time differ

- other important aspects that will be discussed more briefly are *fairness*, *vulnerability*, *communication overhead* and *openness*.

1.3   Research method

The research method will be experimental validation through simulations.

Chapter 2

The application

This chapter describes the Intelligent Network concept that has been chosen as the application to be used for evaluation of the multi-agent architectures.

The Intelligent Network concept is chosen as the sample application for empirical examinations of the multi-agent architectures because of it's commercial nature that makes clear why available resources must be fully utilised, and because it visualise problems that the telecommunication industry is currently facing in the networks of today.

A high-level description of how a service ordering in an Intelligent Network is carried out can be outlined like this; "When a telephone user order an Intelligent Network service, the service requests are initiated into the network at a Service Switching Point (SSP). But an SSP node cannot provide any services on its own, for this purpose it connects to a Service Control Point (SCP), which are where all the service software resides." Intelligent Network services can be used from any telephone that is direct connected to the public telephone network (in e.g. Sweden), and is equipped with the [*] and [#] buttons.

The main task for an Intelligent Network control algorithm is to perform allocation of the resources available at the service providing SCPs to the consuming SSPs. In an article [1] on this subject, Arvidsson et al. argue that the traditional node-based control of Intelligent Networks are vulnerable and not able to allocate the resources in a way that is optimal for the whole network, therefore the idea of multi-agent load balancing and overload control is introduced. The traditional node-based Intelligent Networks typically use a load control algorithm such as *Automatic Call Gapping* [6]. This algorithm forces a minimum time separation between call requests.

As the load in an Intelligent Network is varying, you never want to utilise an Intelligent Network to 100%. Because if you do that, there will be no spare resources to handle the peaks that lies within the normal variations. When the offered load equals the system's capacity the peaks will be queued (i.e. delayed) and executed later when there is less than 100% load. To prohibit such behaviour, you better aim to utilise the system at a

3

level a bit below the system's capacity, thus the term *target load* is introduced. Typically the target load is set to 90%, which means that the goal of the load control algorithm is to keep the load of the system at a maximum of 90% of the system's capacity by rejecting requests when the offered load grows above the target load.

Chapter 3

The architectures

This chapter contains a walk-through of the four different architectures proposed for load distribution and congestion control

This thesis will concentrate on the four architectures:

1. CA – Centralised Auction

2. HA – Hierarchically Distributed Auctions

3. CLB – Centralised Leaky Bucket

4. MB – Mobile Broker

The architectures are chosen to complement each other with respect to synchronism and distribution.

- CA and CLB is centralised, while

- HA and MB is distributed.

In the aspect of resource allocation

- CA and HA is synchronised, while

- CLB and MB is asynchronous.

Each SCP have a corresponding *Quantifier* which is supervising the SCP, keeping track of its capacity, its current load and status, and try to sell available capacity to the *Distributor*. There also exist one *Allocator* for each SSP node. The Allocators are responsible for monitoring the experienced load at each SSP, and for reserving/buying the correct amount of SCP processing resources an SSP node will require.
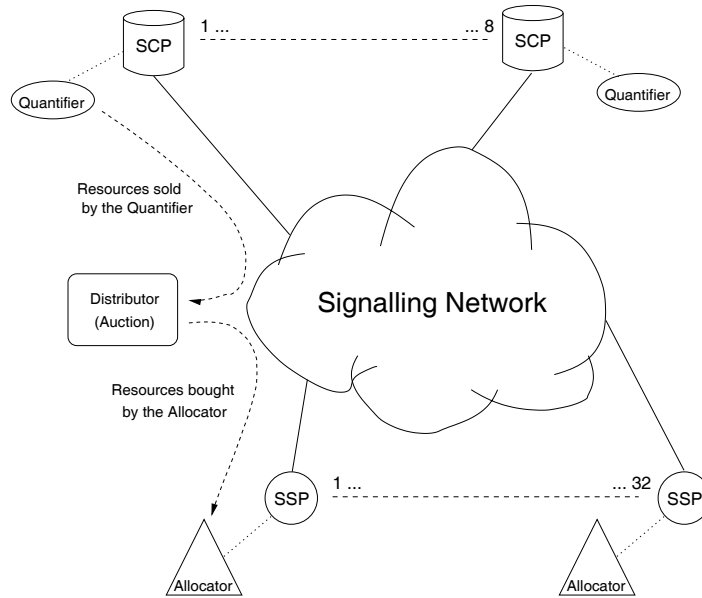
Figure 3.1: Agent interaction in the centralised auction architecture.

In the CA and the CLB architecture there exists one global Distributor, while there in the HA architecture exists additional intermediate Distributors between the central Distributor and the Allocators. In the MB architecture, the Quantifiers and Distributors are replaced by the so-called *Brokers* that continually travel around and visit each of the Allocator agents (or a subset), to offer them resources of the corresponding SCP node. In MB, there exists one Broker for each SCP node.

3.1   Centralised auction

The centralised auction architecture is *centralised* and *synchronous*. In the auction archi-tectures, all SSP nodes maintain a pool of tokens. Each token correspond to an SCP's processing of a service request. Tokens are sold to Allocators at the Distributor auctions on behalf of Quantifiers. The Distributor is located at no specific place but anywhere it can be reached from the signalling cloud. Quantifiers report to the Distributor how much processing power its SCP want to sell, and the Distributor perform auctions typically ev-ery 10 seconds where the SCP processing power are sold to the Allocators which have been sending in their bids of what they need.

When an SSP accept a service request, a token is removed from the pool of that SSP. When there are no more tokens in the pool, the SSP cannot accept any further requests. To avoid running out of tokens too early when the demand is larger than the supply, a rejection probability is calculated so that the remaining tokens are equally distributed

over the time from now until the next auction (see *percent thinning* below). To serve a request, a direct connection is established between an SSP and an SCP.

The centralised auction architecture originally proposed and implemented by Arvidsson et. al in [1] maximises profit for the network owner by favouring those services that gives the higher profit. That means that during overload situations, SSPs have the ability to reject requests for low profit services while accepting requests for services that gives a higher profit. If we remove this task of profit optimisation from the central auction, the main task for the auction is reduced to distribution of SCP processing power according to the percentage of load each SSP experience.

This claim have been validated by comparing simulations where the existing code was used with other simulations where the auction algorithm was replaced by another algorithm that just divides available SCP processing power among the SSPs after their percentage of experienced load. This second algorithm was introduced to minimise effects of the phenomenon that shows up at the end of an auction interval when an arbitrary SSP are out of tokens for service one, but have a few more tokens left for service two. However unluckily, there only arrives requests for service one, so when the auction interval is ending, the SSP have to throw away unused service two tokens (that could have been used up if there was only one type of token). This gives that when there is only one type of tokens, the architecture will carry slightly more load than when there is two. The phenomenon is most obvious and easiest to study when offered load equals the target load near the system's maximum capacity.

Percent thinning

Percent thinning is used to distribute the accepted load evenly over an auction interval, to prevent that all tokens are spent in the beginning of the interval during an overload situation. That would mean that all requests would have to be rejected for large coherent chunks of time, which is unacceptable.

Another problem that occur during overload if we do not use percent thinning is that SCP nodes will not manage to process all requests that arrive almost simultaneously just after an auction has been held and the token pools have been refilled. The request will then be queued and the connection response time will be suffering from extensive delays.

The percent thinning mechanism works by calculating available tokens over expected number of requests to find out how many percent of the offered requests it can afford to accept.

There exist two ways of predicting the offered load for the next interval of time. The more advanced is to apply a machinelearning algorithm on some statistics, for example the history of previous intervals, to try to recognise patterns or trends. This approach is more complex to implement and requires more computational resources than the other method. That is why the simpler approach is used in the simulations. The simpler approach is to assume that the offered load during next interval will be the same as it was
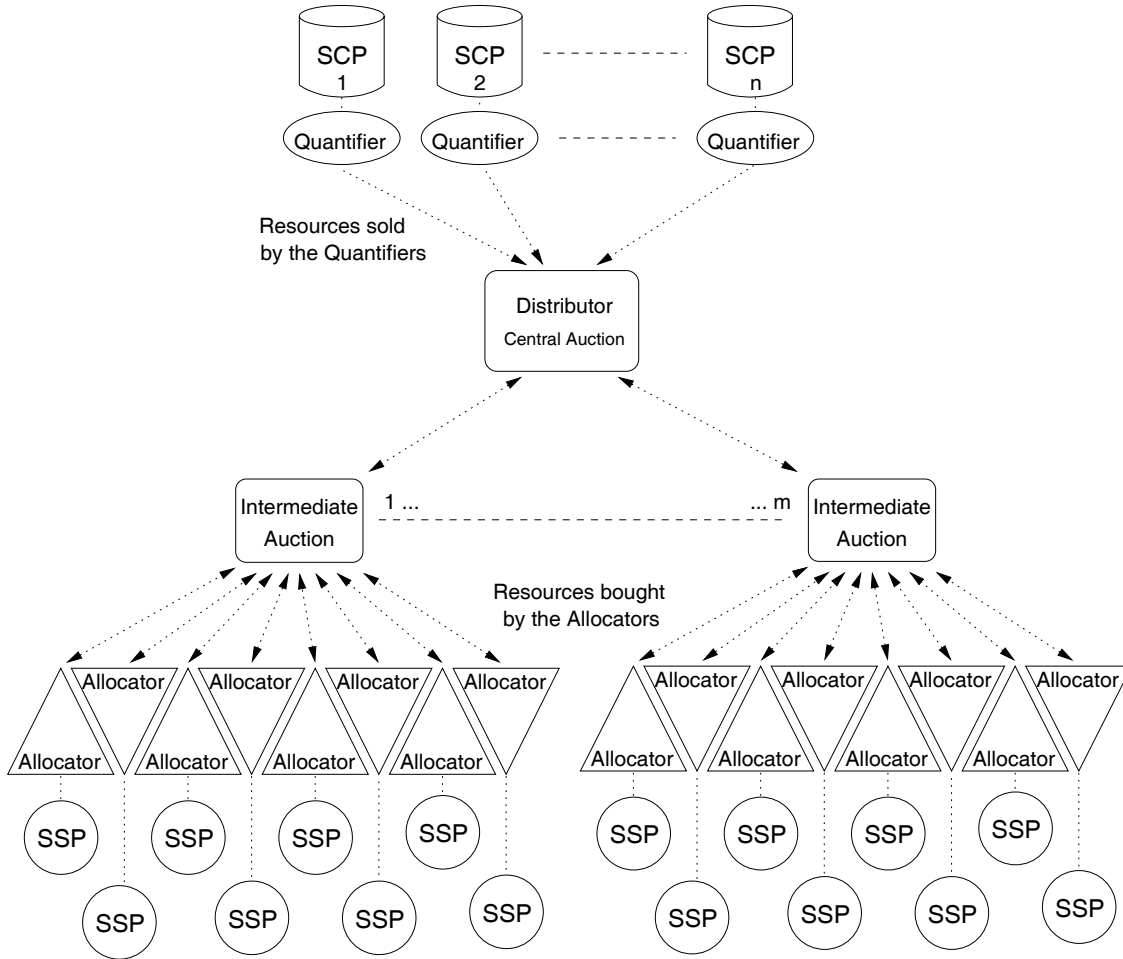
Figure 3.2: The hierarchically distributed auctions (HA) architecture

during the last interval. There exist a trade-off (and a parameter to tune) between the shorter time interval – the closer to current load situation, and the longer interval – the smaller becomes the errors inferred by variations in load.

3.2   Hierarchically distributed auctions

The hierarchically distributed auctions architecture is *distributed* and *synchronous*.

The main difference between CA and HA is that in the HA there exist intermediate Distributors between the central main Distributor and the Allocators. We still have the central Distributor with a global view of the whole systems that makes decisions at equal quality as in the CA. However, in addition HA adds the ability of making more and faster decisions at a lower level. HA thereby adds the possibility of a re-balance in a subset

of the network and faster adoption to changes in load as the smaller auctions can be performed more often than the central one. That the smaller auctions can be performed more often is true because HA distributes the processing required to carry out the auctions over several processing nodes, and as the distributed auctions can be located closer to the Allocators the messages only have to bother the closest subset of the network. Altogether, this means that the distributed auctions can be held simultaneously in different parts of the network. However, there is a penalty to run the auctions too often. The more often the auctions are performed the less load will be carried when the offered load is close to the target load due to tokens that get discarded (HA uses the same kind of percent thinning as CA and MB).

Another rather obvious fact is that is true for both of the auction architectures is that a re-balance cannot be performed at any other time than at the auctions. This is the main disadvantage of the synchronised architectures compared to the asynchronous.

Percent thinning

An algorithm like percent thinning is required to prevent overload of the processing nodes right after each auction, when the token pools have been refilled. However, as you divide the auction interval in even smaller shares, each share the algorithm will be responsible for will be smaller and some of the responsibility for normalising the load will be transferred to the distribution of auctions. As the distributed auctions of HA are performed more often than the central auction in CA, the HA architecture will be less dependent than CA of a good percent-thinning algorithm to normalise the rate at which each SSP node will accept requests over the interval between two auctions during an overload situation.

3.3   Centralised leaky bucket

The leaky bucket architecture is *centralised* and *asynchronous*.

The leaky bucket architecture[1] is named after the metaphor of a leaking bucket, where the water leaks from the bucket in a constant torrent. The basic idea is that each incoming request adds to the bucket if there is at lest one slot available. The size of the bucket imposes an upper bound on the burst size that can be accepted, those requests that do not find available slots are rejected. The maximum response time of a request will be proportional to the bucket size.

As the centralised leaky bucket (CLB) architecture is fully centralised, it is informed about the current load situations in all nodes, therefore this architecture have the best preconditions to make well-informed decisions.

The *Distributor* in the CLB architecture is implemented as a finite queue that holds the SSP service requests and a router that de-queues the requests and forward them to

---

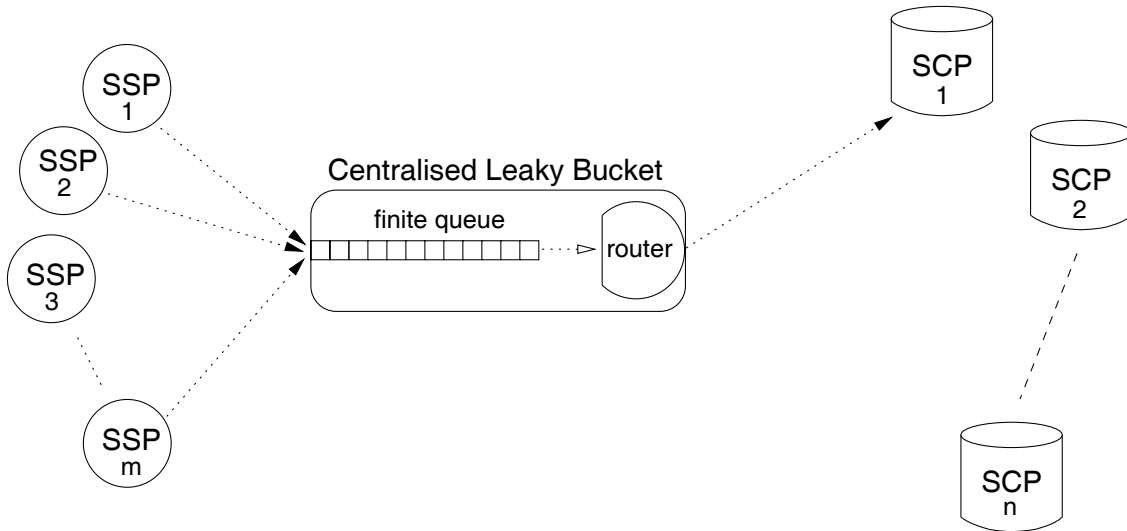[1]*Leaky Bucket* was first introduced for Asynchronous Transfer Mode (ATM) networks in [4].

Figure 3.3: In the CLB resource allocation is not carried out until the need is spawn from an incoming service request

the next available SCP at a rate that corresponds to the target load. After the initial connection, further interaction is accomplished using a direct connection between the SSP and SCP.

When the finite queue has no more slots available, further service requests are rejected. In our specific case, the size of the finite queue in the leaky bucket distributor is of subordinate importance beside the fact that the queue is finite and fills up during overload. Instead, the parameter best suited for deciding the size of the queue is the maximum acceptable response time. Preferably, the size of the queue should be kept so that the maximum acceptable connection time for a service request is not exceeded. The time required for accepting a service request during overload is proportional to the queue length times the time it takes for the distributor to process one request.

An alternative or complementary way to regulate the rejection rate (that sometimes is used in transaction processing systems such as GSM SMS text messaging systems) would be to instead of using a finite queue in the Distributor, define a maximum rate at which service request can be issued from the SSPs. This would work fine for securing the system against SCP overload, but it would be less flexible as no SSP nodes can have a higher load even if the overall network load is low.

In the CLB architecture, there is no use for a percent thinning-algorithm like the one used in the other architectures as the CLB Distributor already makes an optimal *thinning*.

3.4   Mobile broker

The MB architecture is *distributed* and *asynchronous*.

The MB architecture is the most complex one of the four. In MB each SCP have a corresponding Broker that continually travels around to a subset of the Allocators to sell resources. The Broker routes are static and have been set up so that each Allocator is visited by two different brokers, and so that those two broker-visits are spread in time (i.e. two brokers do not arrive at the same time). Each Broker route comprehend eight Allocators, and each Broker route cross every other Broker route at least once every lap.

A different approach would be to use dynamic routes, which probably would cause more deviations in response time and carried load, but a better ability to distribute uneven load evenly over all available SCPs. However, this is only theory since dynamic routes have not yet been examined in simulations.

In order to avoid selling all resources to the first Allocator in the route during an overload situation, each Broker keeps track of the total experienced demand at all Allocators in its route. The broker then calculates the percentage of load at current Allocator from the quote: the demand at current Allocator over the total demand for all Allocators in the Broker route. This is done by the Broker in order to find out how much of the resources that shall be given to the current Allocator. This procedure also makes sure that the Broker distributes all the processing power of its SCP, i.e. that it do not have spare capacity left over when the route is completed. A drawback of this approach is that the Broker will hand out too much bandwidth during the first lap after a sudden increase in the offered load (as brokers operate by handing out the abstract concept of bandwidth and hence can not run out of resources).

An additional balancing function that is used in the MB architecture is that each Allocator try to relieve pressure from its heaviest loaded Broker and move requests to it's other Broker in case they are unevenly loaded. The allocator calculates the load of its Broker from the quotient between the given request (i.e. the experienced demand) and the received allocation.
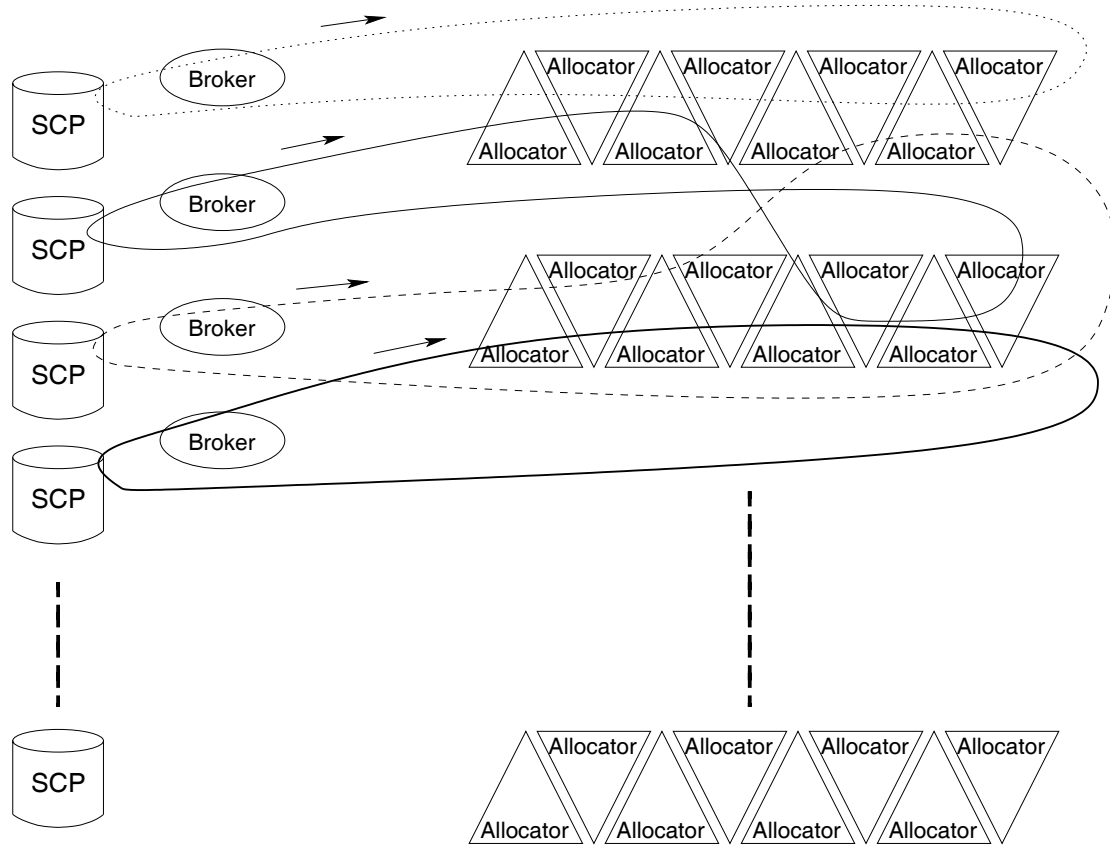
Figure 3.4: The Brokers continually travel and visit a subset of the Allocators in purpose
to sell the resources of its corresponding SCP

Position	$broker_1$	$broker_2$	$broker_3$	$broker_4$	$broker_5$	$broker_6$	$broker_7$	$broker_8$
1	A - 1	A-13	A-17	A-21	A - 4	A-30	A-32	A-25
2	A - 2	A-14	A-18	A-22	A-10	A-31	A - 6	A-27
3	A - 3	A-15	A-19	A-23	A-26	A-11	A-28	A - 7
4	A - 4	A - 8	A-20	A-24	A-16	A-29	A-12	A-30
5	A - 5	A - 9	A-21	A-25	A-22	A - 1	A-31	A-32
6	A - 6	A-10	A - 2	A-26	A-27	A-17	A-14	A-13
7	A - 7	A-11	A-15	A - 3	A-28	A-23	A-19	A-18
8	A - 8	A-12	A-16	A - 9	A-29	A - 5	A-24	A-20

Table 3.1: The Broker routes used in the simulations. Each Broker route comprehend
eight Allocators and cross every other Broker route at least once every lap. Each Allocator
is visited by two different brokers.

Chapter 4

Simulation – preconditions

This chapter is a description of the specific network configuration that is used in the simulations. It also holds a description of the attributes the simulations are supposed to investigate.

4.1 Simulator configuration

4.1.1 General setup issues

The basic network configuration that is used in the simulations are taken from the MARINER Simulation Scenario Handbook[1] and consists of 8 SCPs and 32 SSPs connected by a SS7[2] network cloud. It is assumed that no messages are lost and that a portion of the bandwidth have been set aside for the agents' interaction, so this will not be suffocated during network overloads. Each message sent through the network will experience a constant delay of five milliseconds. When the offered load is higher than the system manages to process and the load control algorithm is turned off, the SCPs will queue requests and process them later. The call holding times are negative exponentially distributed with a mean of 100 seconds. Service requests arrive according to independent Poisson processes. The services used in the simulator are set-up to resemble *real* services. The services are described in the MARINER Simulation Scenario Handbook, the services are:

- Service A, Virtual Private Network

- Service B, Ring back

A simplified figure of the simulated network is shown in figure 4.1 on page 14.

---

[1]An extract of the MARINER Simulation Scenario Handbook is attached as appendix A.
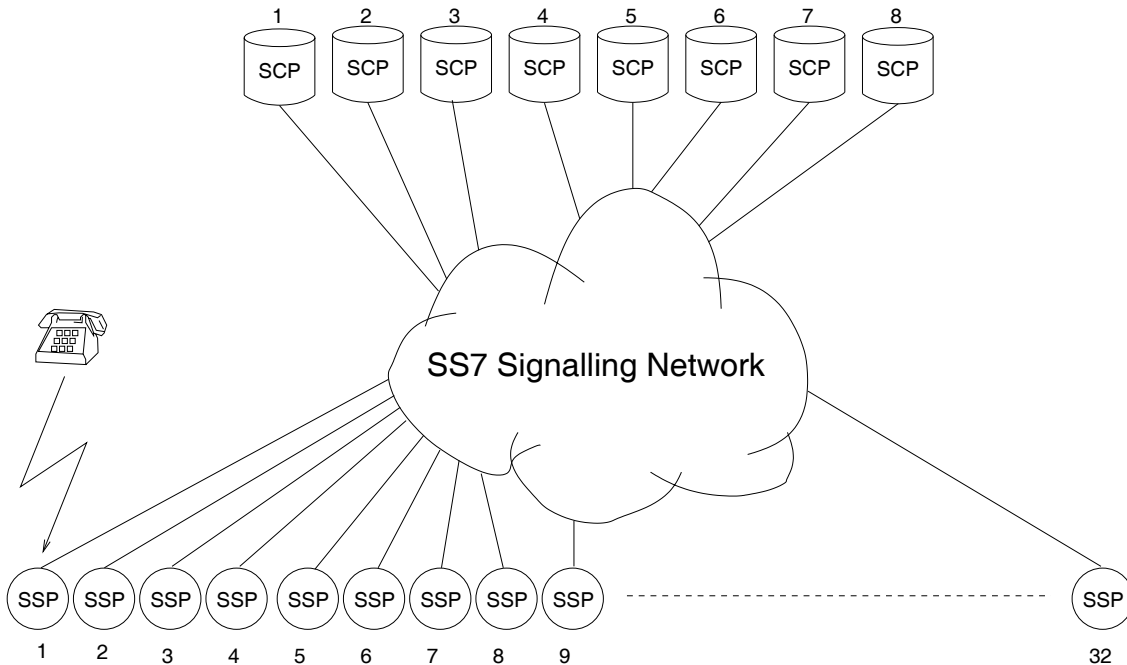[2]Please refer to [7] for a definition of the SS7 network architecture.

Figure 4.1: The basic network configuration used in the simulations

4.1.2   Architecture specific issues

All SCPs are identical in respect of processing capacity and software configuration i.e. they provide the same set of services, they are also equally reachable from all SSPs (i.e. a message will experience the same delay regardless of which SCP it is sent to). That all SCP nodes provide the same set of services is a prerequisite for the Mobile Broker architecture to work. Please refer to table 3.1 on page 12 for the specific broker routes used in the simulation.

In the HA architecture, Allocator 1-8 is partitioned to intermediate auction 1, Allocator 9-16 into intermediate auction 2, Allocator 17-24 into intermediate auction 3 and Allocator 25-32 into intermediate auction 4, meaning that a redistribution of resources between Allocator 1 and Allocator 5, e.g. can be performed at an intermediate auction while a redistribution between Allocator 1 and Allocator 9 cannot.

The auction interval of the CA architecture is set to 10 seconds, which is also the time interval between the main auctions in the HA architecture. The intermediate auction interval in HA is set to 2 seconds, which implies that there are 5 intermediate auctions for every main auction. The size of the CLB queue is set to 80. The Broker routes in the MB architecture takes approximately 1.6 seconds to complete, as a Broker spend 0.2 seconds at each Allocator.

4.2 Description of the attributes to be investigated

The architectures will be examined and compared according to several important attributes. Those attributes are:

- **Utilisation of resources**
  In this section, the aspect to investigate is how close to the target load the different architectures manage to carry the load at different conditions in terms of offered load.

- **Load balancing**
  When all SSPs are offered the same continuous load, how do the architectures differ in terms of load distribution? We now compare how the resources are consumed at the individual SCP level by measuring the standard deviation of the carried load between the SCPs.

- **Reactivity**
  An important attribute of a load control architecture is how fast it manages to adapt to new load situations, how fast it can re-allocate the resources when there have been a change in the demand. Two different aspects will be studied:

  - **Overload control**
    How will the architectures manage a synchronised instant increase or decrease in the demand? How fast will the carried load follow the offered load? In case of a severe increase in the offered load, will the carried load exceed the target load even for a short while? In case of a drastic drop in the offered load, will the carried load sink below the offered load even for a short while?

  - **Balancing of an unequally applied load**
    In this section the offered load is kept below the target load, so the system should be able to carry all load offered. The main part of the offered load is instantly moved from one side of the network to the other. The issue being investigated is how fast the architectures manage to redistribute the resources from one part of the network to another.

- **Response time**
  An important measure on the *quality of service* from the user's point of view is how long it takes a service to be connected. E.g, how long before the user gets a response from the system informing if the request has been accepted or rejected.

- **Communication overhead**
  What constraints do the architectures place on the underlying infrastructure for communication? How does the amount of bandwidth that has to be reserved for the agent interaction differ between the architectures?

- **Robustness**
  Is there any architecture that is more vulnerable than another, or that makes the Intelligent Network as a whole more vulnerable? In what ways can a certain architecture affect the Intelligent Network as a whole in case of a hardware failure in some part of the network?

- **Fairness**
  Do the architectures treat all involved agents equal, or does any agent become favoured or disregarded under some circumstance?

- **Scalability**
  Scalability is the aspect of how easy it is to resize the network, especially to add extra SSP or SCP nodes when the system is in production and cannot be shut down and reconfigured off-line.

Chapter 5

Simulation – results

This chapter will examine the set of architectures described in chapter 3 in the domain of intelligent network resource allocation, congestion control and load distribution presented in chapter 2.

5.1 Utilisation of resources

To figure out how close to the target load the architectures manage to keep the carried load, simulations that are 20 simulated minutes long have been used. The measurement was carried out during the last 10 minutes. Because the system needs only about five minutes to stabilise after a start-up, a delay of ten minutes should be enough.

Seven different simulations were run on each architecture. In the different simulations, the offered load was set to 35%, 70%, 85%, 95%, 105%, 150% and 200% of the system's maximum capacity respectively. The simulations were repeated with five different sets of random seeds. It was found that the standard deviation between the same runs but with different random seeds was about 0.1% on the average for all SCPs on a per second basis, and never more than 0.3% for an individual SCP.

The unit of measurement "load expressed in percent of the system's maximum capacity" is also called *Erlang*, which is the name that will be used throughout the rest of this chapter. The system's maximum capacity corresponds to one Erlang.

The result data from these simulations are presented in figure 5.1 on page 18. It shows that all the architectures manage to carry the load close to the target load (the CLB is in fact *on* the minimum of the target and the offered load, which is optimal). The deviations are largest just around an offered load at 1 Erlang. At this point CA carries the load a little better than HA and MB. But at severe overload, 2 Erlang in offered load, CA and MB comes almost as close to the target load as CLB while HA is left alone staying well below and is not approaching the target, suffering from shorter auction-intervals (which means more discarded tokens) than CA.
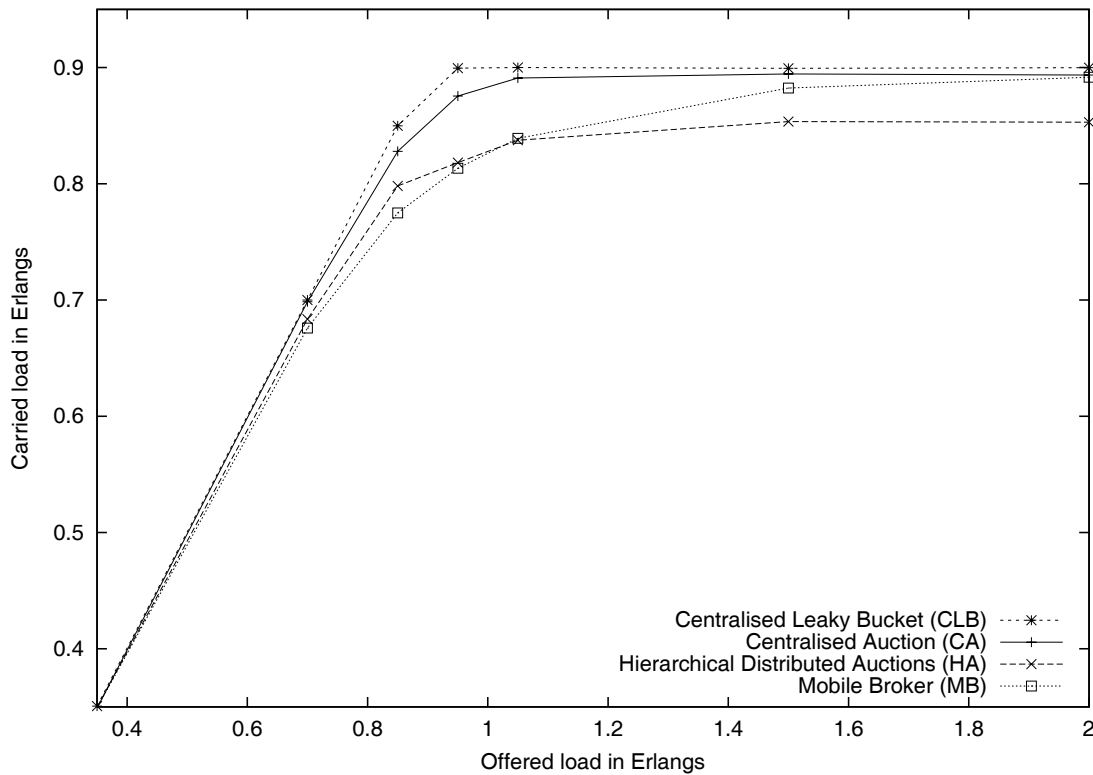
Figure 5.1: Shows how close to the target load the different architectures manage to keep the carried load during a constant offered load. Target load is set to 0.9 Erlang

There exist several possible ways to trim the HA architecture to carry the load closer to target. One could for example see to that extra tokens are handed out to compensate for those that are being discarded. Another way is making the auction intervals adaptive so they become longer when the offered load is stable and vice versa. However, these proposals remain to be validated in simulations.

5.2   Load balancing

Using the same simulation runs, we now compare how the resources are consumed at the individual SCP level by measuring the standard deviation of the carried load between the SCPs. An average of the load carried during the last simulated second are saved every simulated second for each SCP throughout the last 10 simulated minutes. Figure 5.2 on page 19 shows the average standard deviation between the SCPs during 600 simulated seconds.

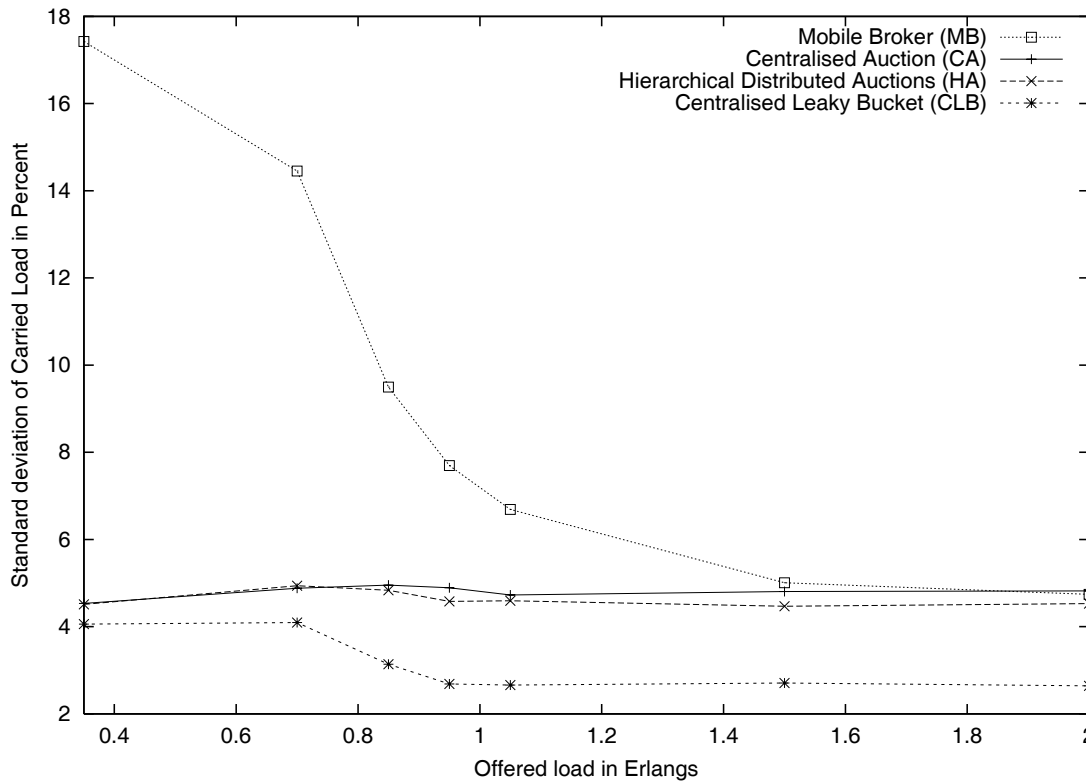Even when the offered load is continuous and stable at a certain level, there are vari-

Figure 5.2: The standard deviation of the carried load between the individual SCPs measured in percent.

ations in the arrival rate, i.e. the time between two consecutive service requests is not constant. In fact, the time is derived from Poisson processes. The variations in arrival rate can be smoothed out by having a set of SSPs to select next request from. If an SCP is only serving one SSP, and that SSP run out of resources, the SCP will get no requests. The more SSPs an SCP is serving, the smaller becomes the effect when one of the SSPs runs out of resources.

As the SCPs in the MB architecture only are serving 8 SSPs each, the carried load deviates more in MB than in CA and HA where each SCP serve all 32 SSPs. The CLB architecture always allocate each new request to the SCP that have had the longest idle time, and thus is optimal.

5.3   Reactivity

5.3.1   Overload control - preconditions

The time it takes before the system can act on changes in demand depends mainly on if there exists any synchronisation event that must be awaited before a change can take place or a message be propagated. Hence, asynchronous architectures perform better than synchronous in the aspect of reactivity.

To find out the difference in how fast the architectures can adapt to changes in the offered load, an instant increase from 0.35 to 2.0 Erlang was used. The peak occurs synchronised on all SSPs at time 400 seconds from start and last for another 400 seconds before it drops back to 0.35 Erlang. The simulation results are visualised in the figures 5.4 to 5.11 on page 21 to 24, and are commented in section 5.3.2.

The expected result

Before taking a look at the actual simulation results, we take a look at what the shape of the expected simulation results would be like. Because of the architectures unawareness of history, they make the faulty assumption that the processing required for disconnection of a service takes place at connection time. This assumption implies that the carried load will lag behind a time corresponding to the average time between connection and disconnection of accepted service requests. Only when the disconnection rate equals the connection rate, the architectures manage to make correct decisions about connection versus rejection and thereby reach the target load (if the offered load is higher than the target load).
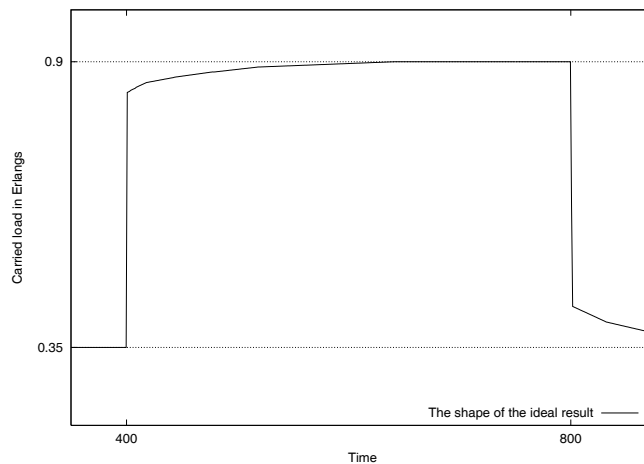

Figure 5.3: The expected simulation result of the synchronised peak simulation given the architectures prerequisites and the simulation parameters.
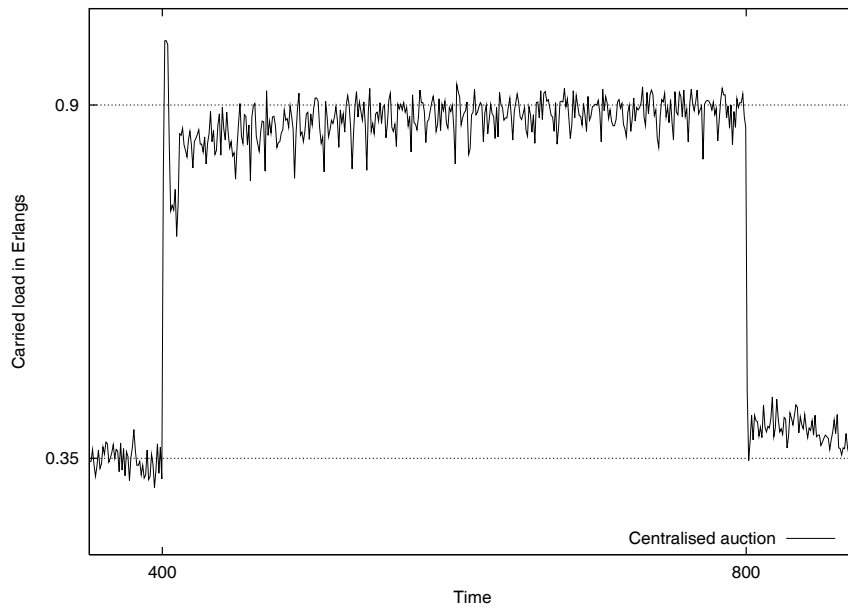
Figure 5.4: Centralised auction exposed to a synchronised peak. At time 400, the offered load is instantly raised from 0.35 Erlang to 2.0 Erlang. At time 800, the offered load drop back to 0.35 Erlang in an instant. Target load is set to 0.9 Erlang
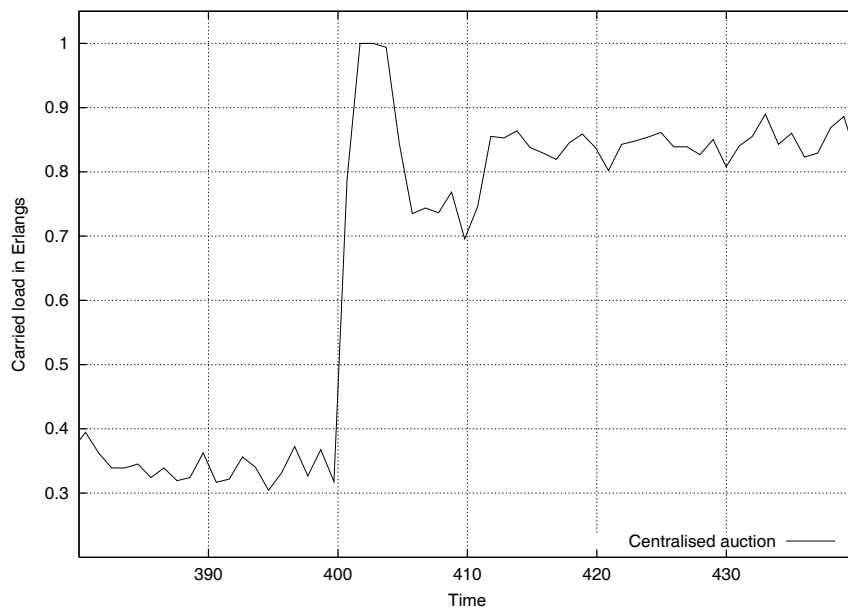

Figure 5.5: A zoom-in of the same simulation as in figure 5.4. The auction interval is 10 seconds. The points of time when the auctions are held are marked in the figure by the x-axis grid.
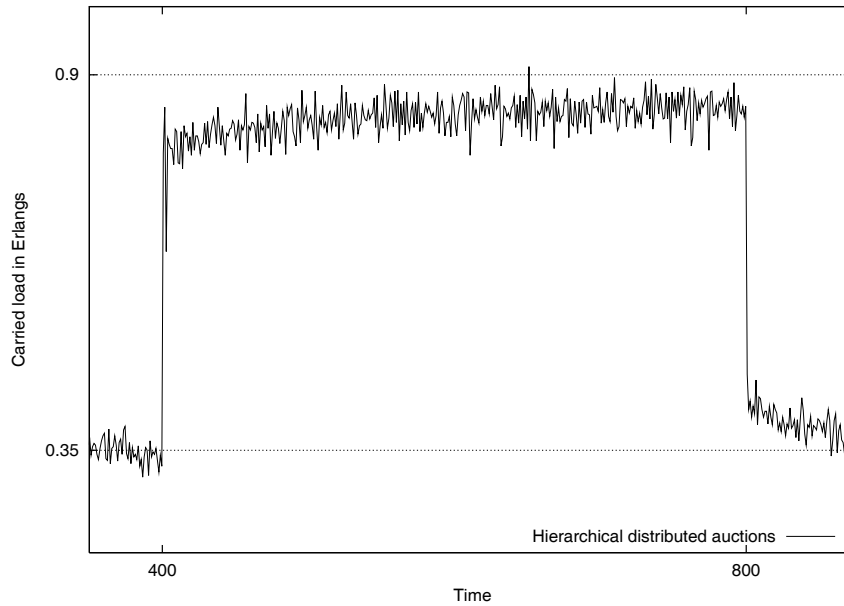
Figure 5.6: The hierarchically distributed auctions exposed to a synchronised peak. At time 400, the offered load is instantly raised from 0.35 Erlang to 2.0 Erlang. At time 800, the offered load drop back to 0.35 Erlang in an instant. Target load is set to 0.9 Erlang.
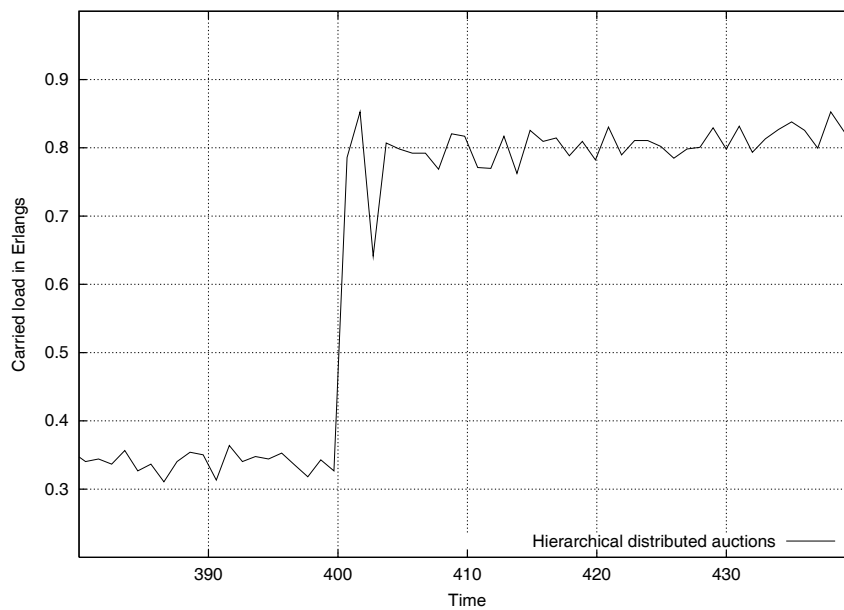

Figure 5.7: A zoom-in of the same simulation as in figure 5.6. The main auction interval is 10 seconds and the sub-auction interval is 2 seconds. The points of time when the main auctions are are held are marked in the figure by the x-axis grid.
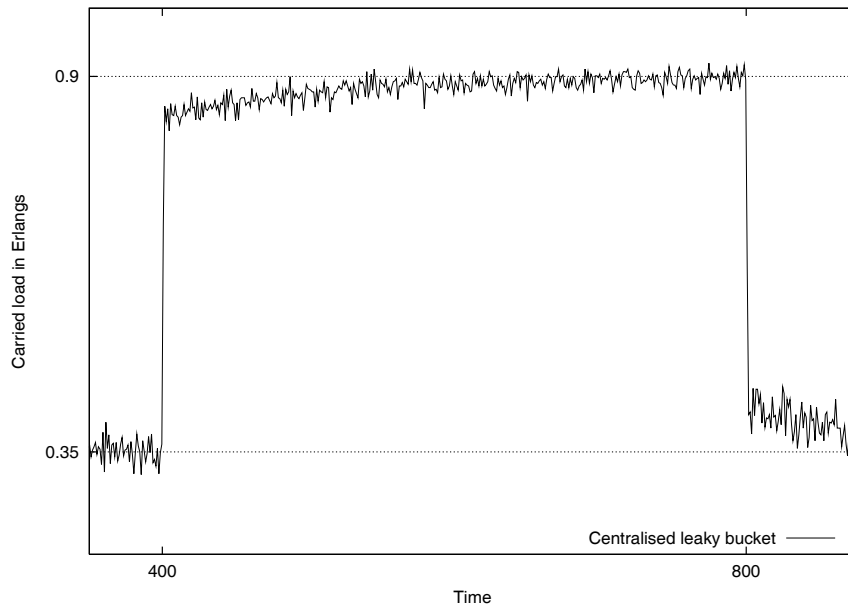
Figure 5.8: Centralised leaky bucket exposed to a synchronised peak. At time 400, the offered load is instantly raised from 0.35 Erlang to 2.0 Erlang. At time 800, the offered load drop back to 0.35 Erlang in an instant. Target load is set to 0.9 Erlang.

Figure 5.9: A zoom-in of the same simulation as in figure 5.8. CLB follows the load almost perfectly.

Figure 5.10: Mobile broker exposed to a synchronised peak. At time 400, the offered load is instantly raised from 0.35 Erlang to 2.0 Erlang. At time 800, the offered load drop back to 0.35 Erlang in an instant. Target load is set to 0.9 Erlang.
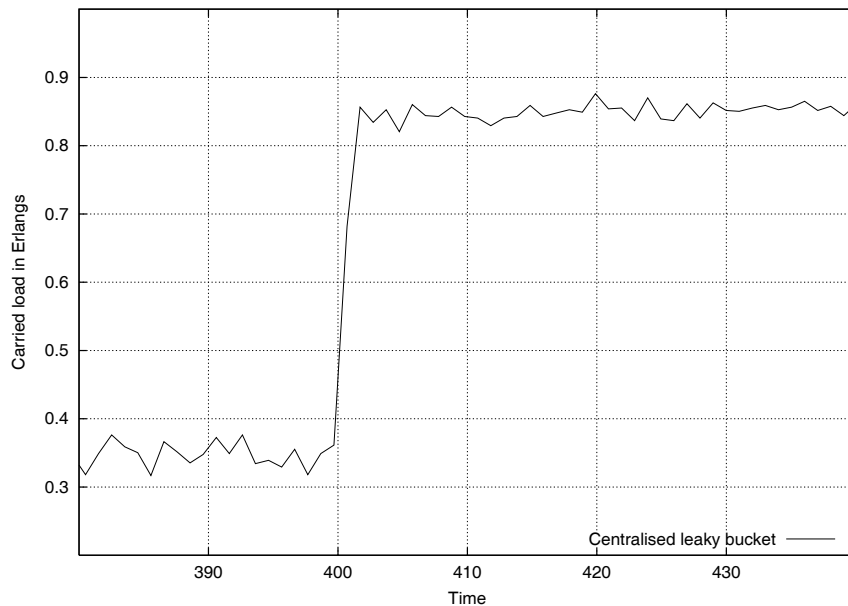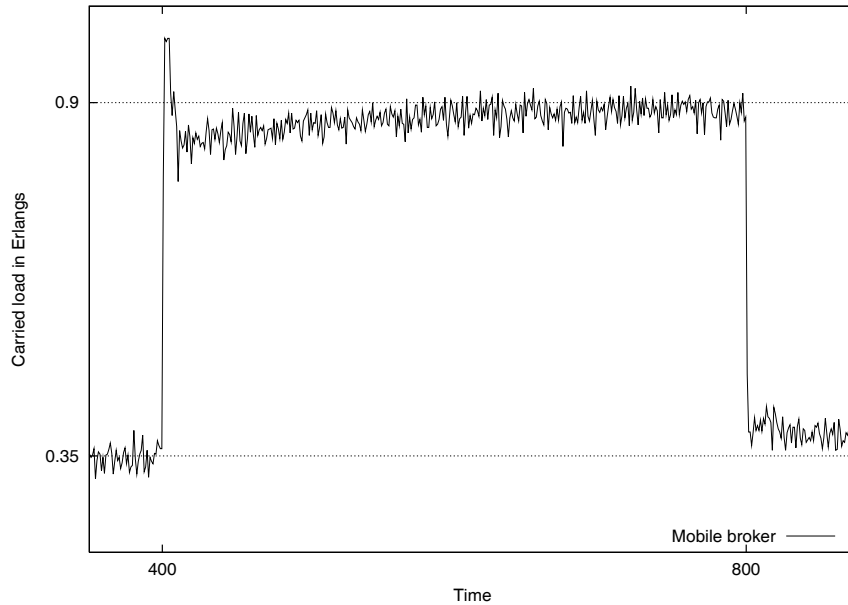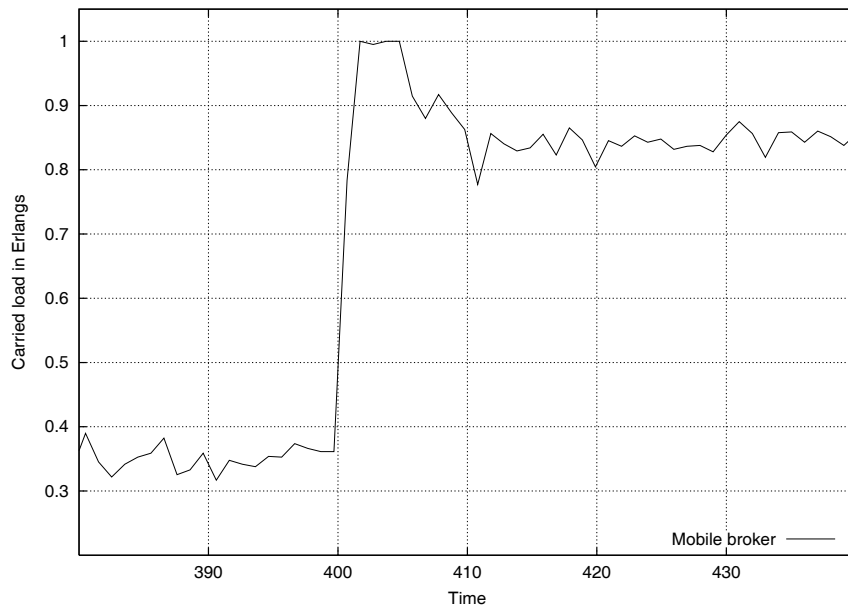

Figure 5.11: A zoom-in of the same simulation as in figure 5.10. The broker routes take approximately 1.6 seconds to complete.

5.3.2 Overload control - comments upon the simulation results

Centralised auctions

This simulation is presented in figure 5.4 and 5.5 on page 21.

CA is the architecture that causes the largest alternation from overload to under-load (1.0 to 0.7 Erlang) before the carried load stabilise in the second auction interval after the peak has occurred. The alternation is an effect caused by the combination of the auction interval time and the percent-thinning algorithm. Just after the offered load has been raised from 0.35 to 2.0 Erlang, the percent-thinning algorithm bases its computations of expected offered load on the 0.35 Erlang load conditions from the last time slot of the preceding auction interval. Therefore, the algorithm will overload the processing nodes during the first part of current auction interval before it realises it is running short of tokens using this now too low rejection rate. To avoid running out of tokens the algorithm will increase the rejection rate during the second part of current interval and in this way try to get the tokens to last until the next auction is held. *Since the algorithm already has spent too many tokens in the beginning of the interval, we must wait for the token pools to be refilled at the next auction before the algorithm can have a second chance and approach the target load.*

During the second interval of 2.0 Erlang offered load, the percent thinning algorithm manages to hold the accepted load close to the target load as the algorithm now has accurate input data (i.e. the offered load during the closest preceding auction interval is similar to current interval). Without the percent-thinning algorithm the alternation would occur repeatedly in every auction interval and be even more severe altering from 1 to 0 Erlang carried load.

Hierarchically distributed auctions

This simulation is presented in figure 5.6 and 5.7 on page 22.

As can be seen in the simulation, the HA architecture adopt faster to a change in the offered load than the CA architecture. Already in the middle of the second sub-auction interval the algorithm has stabilised, which is an improvement compared to the centralised auction architecture. However, some tokens will eventually remain at the end of each interval and they will be discarded. This discard event is directly proportional to the number of sub-auctions per main-auction. The HA architecture therefore carries less than CA during overload situations. In the simulation presented in figure 5.6, 5 sub-auctions per main-auction were held.

As discussed earlier in section 5.1, it is possible to trim HA to carry the load closer to the target load.

Centralised leaky bucket

This simulation is presented in figure 5.8 and 5.9 on page 23.

Compared to all the other architectures, the CLB carries the load closer to the target and smoother with less variation and without alternations caused by the architecture itself.

Mobile broker

This simulation is presented in figure 5.10 and 5.11 on page 24.

MB is the architecture that causes the most severe overload when a synchronised peak occurs. Even though the time needed for a Broker to complete a lap is less than 2 seconds, and that the brokers give away too much capacity only during their first lap of peak load, it takes about 5 seconds before the processing SCP nodes manage to finish the queues that were built up then. The time required for the MB architecture to stabilise is similar to that of CA. But in MB, when the SCP's queues are done, the carried load only sinks to the target load, and MB does not like CA dip deep below the target.

The timing for the Broker routes were chosen and tuned so that the MB architecture would use a similar amount of bandwidth for agent interaction (i.e. communication overhead) as the CA architecture.

5.3.3   Balancing of an unequally applied load

To study the investigated architectures' ability to distribute an unequally offered load evenly over the available processing nodes (SCPs), two different scenarios have been used. They are described below together with the simulation results.

Scenario 1 – synchronised swap

This simulation is set up as a worst-case scenario for the synchronised (the auction) architectures.

From start of simulation half of the SSPs (1-16) offer a load that cause a total system load at 0.1 Erlang. The other half (SSP 17-32) offer a load that cause a total system load at 0.7 Erlang. All together, the system is offered a load at 0.8 Erlang, which is below the target (set to 0.9 Erlang) and therefore should be possible to carry. The system is unequally offered the load as half of the SSPs are offered a small portion while the other half of SSPs are offered a rather large portion.

At time 400, the load offered by SSP 1-16 is swapped with the load offered by SSP 17-32. At time 410, the load is swapped back to the original situation again. What is focused upon in this scenario is how fast the different architectures manage to relocate the resources from one side of the network to the other. The way the load is offered is visualised in figure 5.12 on page 27.
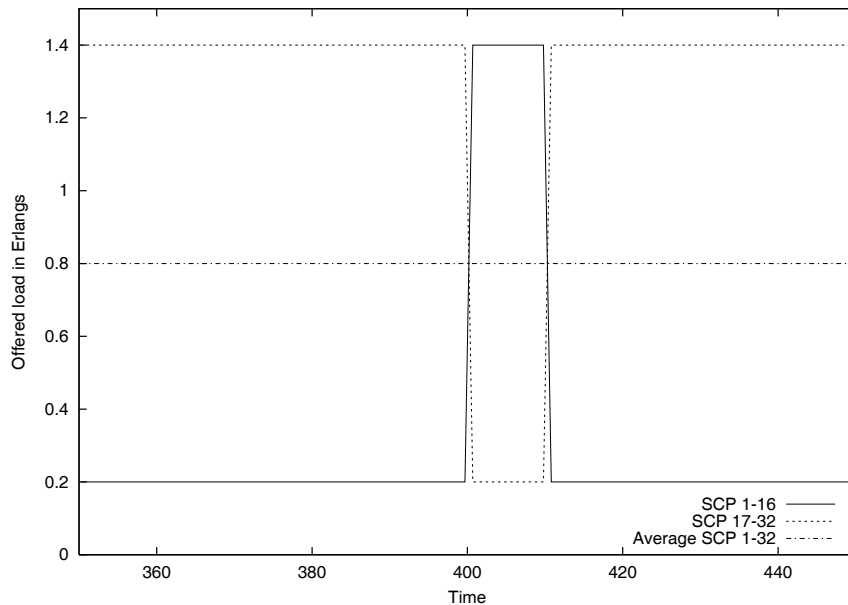
Figure 5.12: The way the load is offered in scenario 1 and 2 – synchronised swap.

Why scenario 1 is a worst-case for CA and HA

The SSPs are chosen so that (for the HA architecture) SSP 1-16 are partitioned to inter-mediate auction 1 and 2, and SSP 17-32 are partitioned to intermediate auction 3 and 4. That means that all SSPs that belong to the same intermediate auction are always swapped together. Therefore, there is no possibility for the intermediate auctions to do anything to regain equilibrium after a swap, as all of the SSPs in each of the intermediate auctions are facing the same situation. Only the main auction can take any action to balance the new load situation, which gives that CA and HA should have the same characteristics in this scenario.

The time for the first swap (400) is chosen to be simultaneous with a main auction to ensure that there will be no premonition about what is going to happen that the algorithm can benefit from. The second swap is set to be simultaneous with the following main auction (at time 410), which means that all statistics that have been collected about the new load situation are now made useless, and the auction algorithm will fail to distribute the resources in a way that would work during the next interval.

Why scenario 1 is a difficult case for MB

The broker routes are set up so that Broker 1 and 2 take care of SSP 1-16, which starts with the smaller load. Each one of SSP 1-16 also have a secondary Broker, all of Broker 3-8 share this task of being the second broker by taking on responsibility for approxi-

mately 2 SSPs each.

The fact that scenario 1 is a difficult case for MB is not caused by the swaps, it is caused by how the broker routes are set up in combination with how the load is offered. Scenario 1 is difficult for MB not during the swap (time 400-410), but before time 400 and after time 410 when SSP 1-16 is offered the smaller load. As none of Broker 1 and 2 visit any SSP with a higher load, they cannot carry any other load than the one offered to SSP 1-16. Because of this, SCP 1 and 2 will carry less load than SCP 3-8, which also visit SSPs with the higher load.

This scenario demonstrates that when using static broker routes, it will always be possible to find a load scenario that cannot be balanced. This could be dealt with, for example by giving each broker route a random extra SSP in each lap, but that would impact on the performance of the dual balancing mechanisms. Another way could be to introduce some kind of route permutation, similar to how the concept of permutation is used in genetic algorithms, but that would make it difficult to prevent that the broker visits for each SSP do not occur too close to each other in time. In this area, further empirical studies are probably necessary.
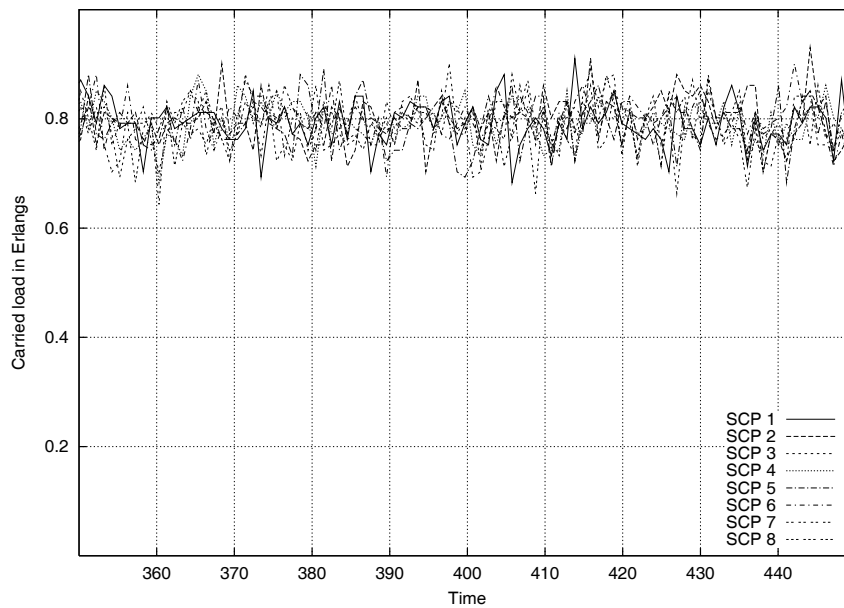

Figure 5.13: The load carried by CLB when scenario 1 is applied. The CLB always manages to distribute the load optimal.
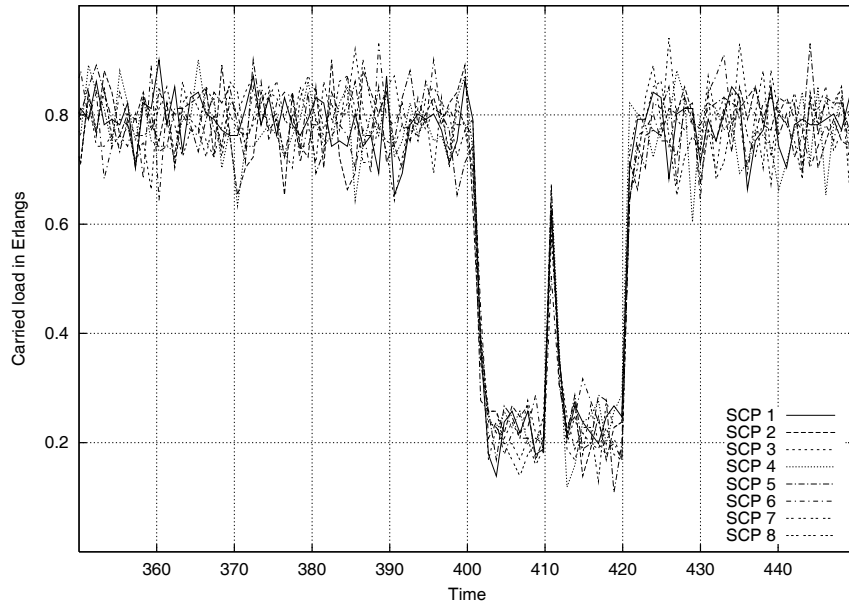
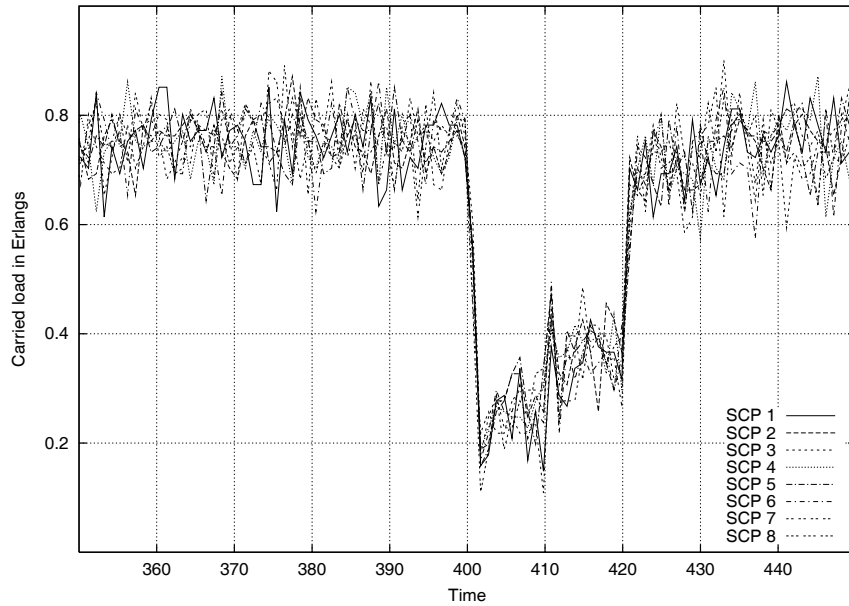Figure 5.14: The load carried by CA when scenario 1 is applied. The auctions are marked by the x-axis grid.

Figure 5.15: The load carried by HA when scenario 1 is applied. Despite the reasoning that the behaviour would be the same for HA as for CA, it turns out that this is not the case. The carried load after the second swap is improved compared to CA.

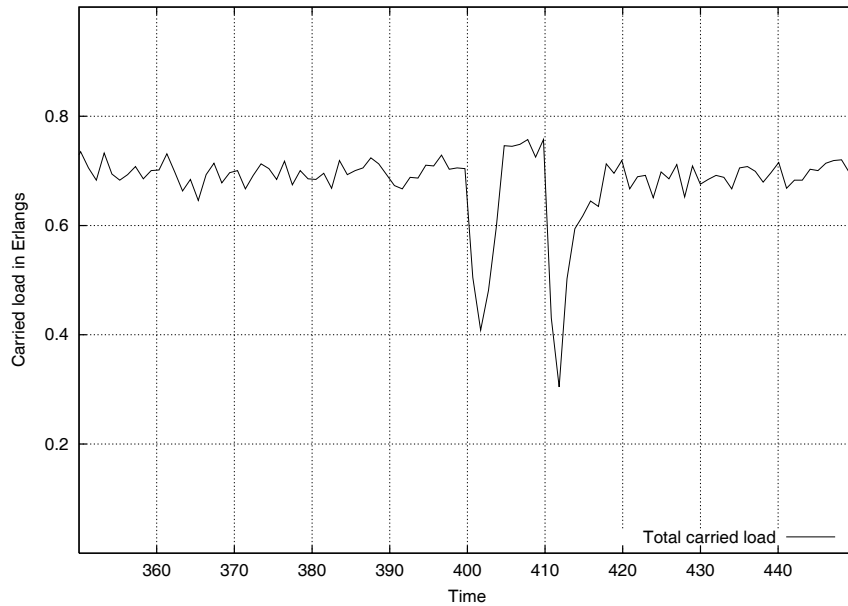Figure 5.16: The total carried load for the MB when scenario 1 is applied. MB manages to regain balance in the load distribution much faster than CA and HA. Also, note that the carried load is below what could be carried by MB because two of the Brokers cannot find the load (before time 400 and after time 410).
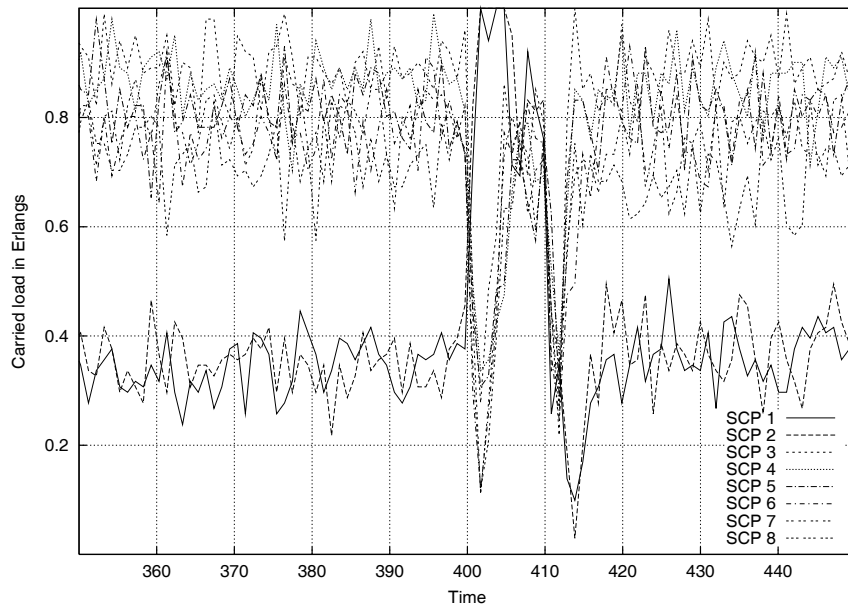

Figure 5.17: This is the same simulation as in figure 5.16, but the carried loads are showed for each SCP individually.

Scenario 2 – synchronised swap re-partitioned

The goal of scenario 2 is to visualise the difference between centralised and distributed architectures. To achieve this (scenario 1 is used as a base) the partitioning of SSPs have been modified while the other conditions stay the same, i.e. the way the load is offered to the system is described in figure 5.12 on page 27. In scenario 2, SSPs with uneven numbers start out by offering the lower load, while evenly numbered SSPs start with the higher load. In scenario 2, the centralised architectures CLB and CA show the same characteristics as in scenario 1 while HA is able to reallocate the resources already at the intermediate auctions. As MB in scenario 2 do not face any difficult case with regard to the broker routes it is now able to balance the load better than in scenario 1.
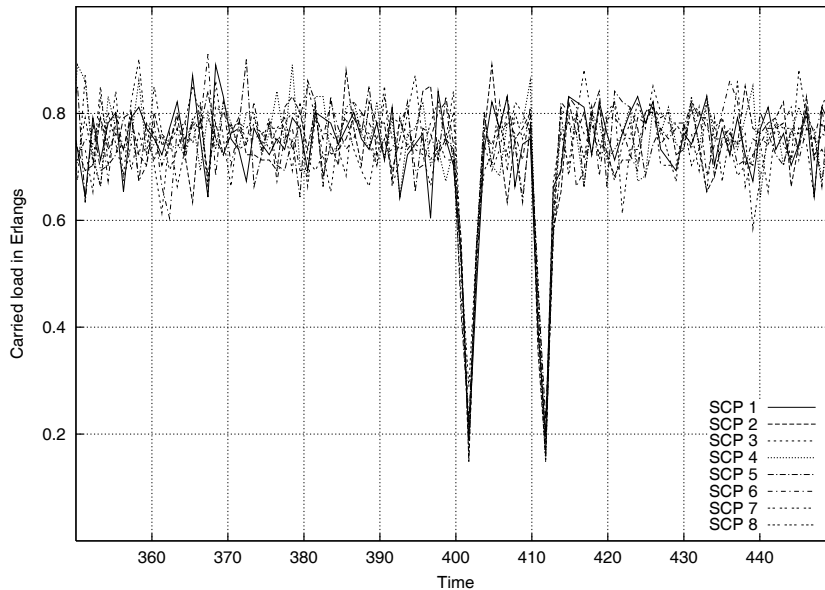

Figure 5.18: The load carried by HA when scenario 2 is applied. In contrast to CA which shows no improvement at all compared to scenario 1, HA performs much better when the load is offered in a way that makes it possible to re-balance the resources already in the distributed auctions. HA is now almost as good as MB. (Compare scenario 1 in figure 5.15 on page 29.)
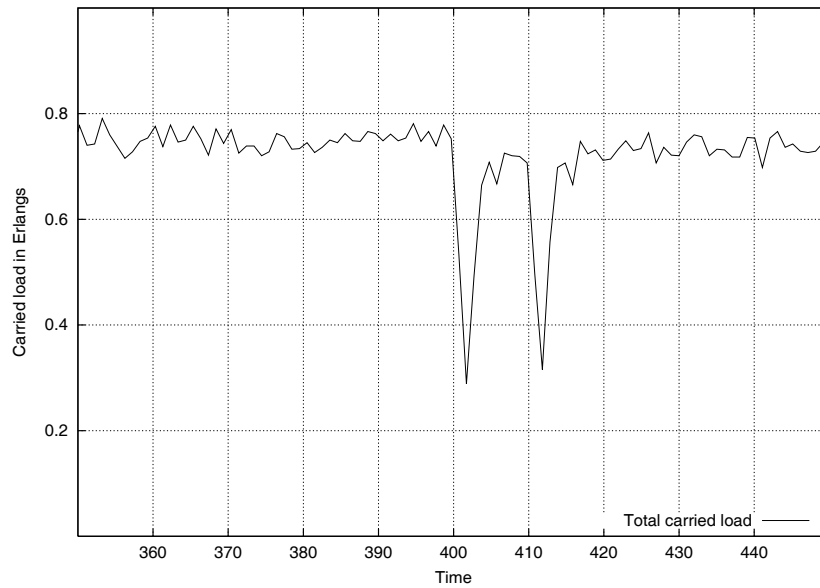
Figure 5.19: The total carried load for the MB when scenario 2 is applied. The time required for MB to regain balance in the load distribution after the load swap is the same as in scenario 1. Although, the carried load is now on track, and therefore the drop right after the first swap is deeper than in scenario 1, as there are no spare resources now ready to be consumed at SCP 1 and SCP 2. (Compare scenario 1 in figure 5.16 on page 30.)
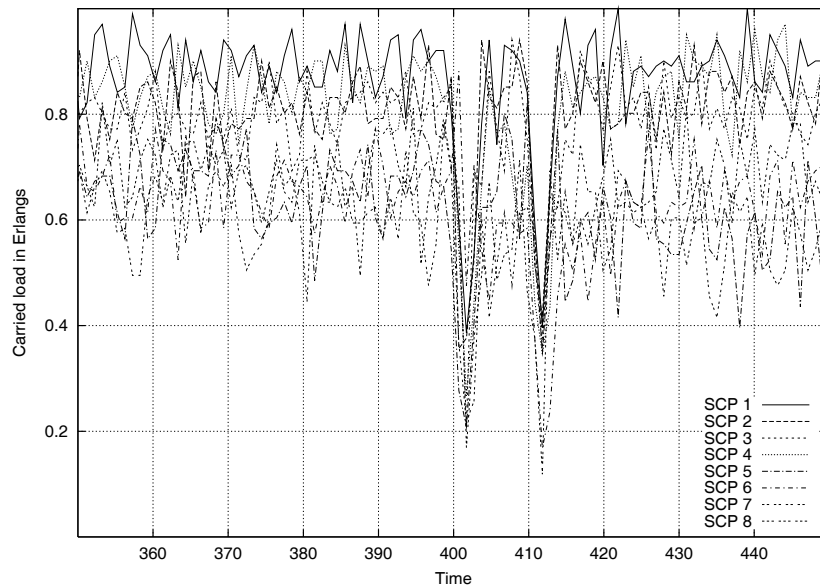
Figure 5.20: This is the same simulation as in figure 5.19, but the carried loads are showed for each SCP individually. The simulation shows that when the loads are offered in a way that is not a difficult case with regard to the broker routes, MB manage to distribute the load in a better way. The SCPs that carry less load than the other are serving seven SSPs from the group that started with the smaller load and only one SSP from the group that started with the higher load (and vice versa).

5.3.4 Conclusions regarding reactivity

Two conclusions can be derived from the simulations in section 5.3:

- Asynchronous architectures excel in balancing the load faster than synchronous architectures.

- Fully distributed architectures (such as MB) can never balance the load better than centralised architectures. (This statement does not concern the HA architecture because it is not *fully* distributed as the central auction is kept.)

5.4 Response time

The response time depends heavily on the load situation at the SCPs. If there is a queue of requests at the SCPs an incoming request will be delayed, and that means a longer response time.

As the CLB manages to distribute and balance the load optimally, this architecture causes no congestion at the SCPs as long as the setting of target load does not exceed 1 Erlang. In addition, as long as the offered load is kept below the target load, the CLB queue is empty (or decreasing). Therefore, when the offered load is kept below the target load, the response time of CLB is better than for the other architectures. But as soon as the offered load grows above target load, the CLB finite queue will fill up quickly as the size of the CLB queue used in the simulations is rather small compare to SCP processing capacity (in order to keep down the response time during overload).

The response time for the auction architectures is almost as good as for the CLB when the offered load is kept below the target load. However, when the offered load grows above target load, the auction architectures perform better than CLB as they do not suffer from any central queueing. The only delay that exists in the CA and HA architectures is (as direct connections are established between SSPs and SCPs) caused when more than one SSP places a request to the same SCP simultaneously, then one of those requests must be queued by the SCP. *Note that even if an architecture manages to allocate available resources in an optimal way, congestion can be caused by how the resources are consumed – and these are two different problems.*

The MB architecture have a slightly slower response time than the auction architectures. This is because MB causes a little more congestion at the SCPs than the other architectures, sometimes the requests come in groups and sometimes there comes no requests for a while. This is also the reason why the MB has a larger standard deviation in the carried load in figure 5.2 on page 19.

Despite what it looks like in figure 5.21, HA does not have a better response time than CA – they do in fact have the same response time when they carry the same load. This is demonstrated in figure 5.22 where the response time is plotted against the carried load
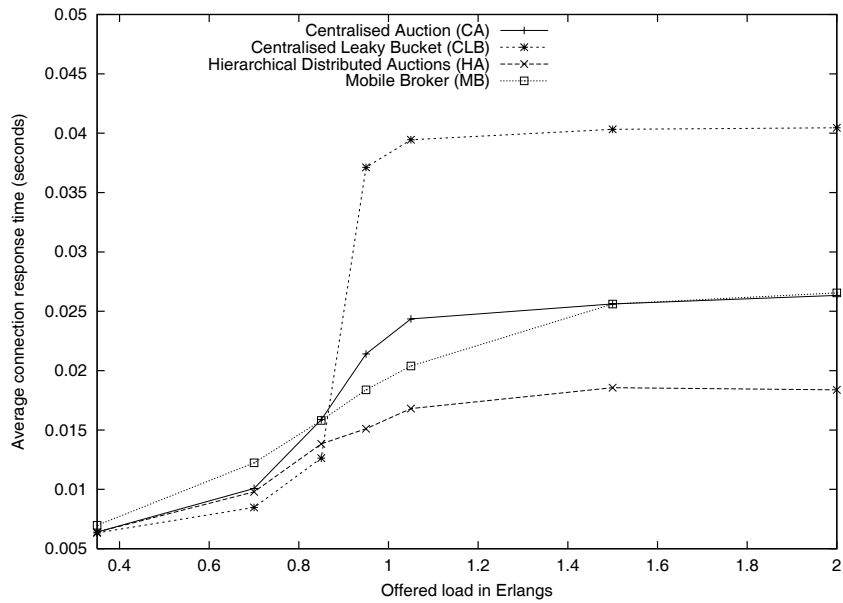
Figure 5.21: Average connection response time at different offered load situations. Note the increase in response time for CLB when it passes target load at 0.9 Erlang.
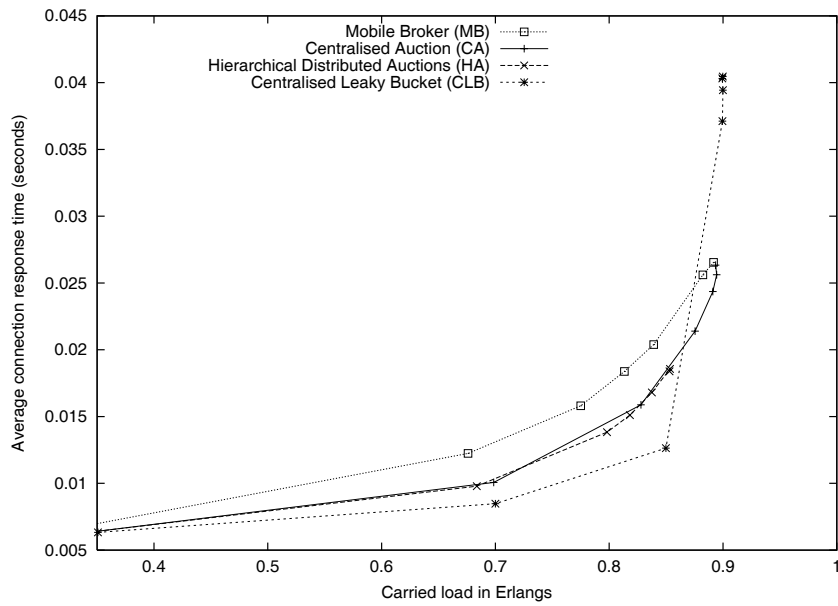

Figure 5.22: When the response time is plotted against the carried load instead of the offered load, it becomes clear that CA and HA have the same response time when they carry the same load.

instead of the offered load. The reason why HA carries less load than CA is explained in section 5.1 on page 18.

5.5   Communication overhead

This section is focused on the communication overhead produced by the allocation mechanisms only, no regard taken to the overhead that occur when the resources are consumed (in fact, this is the same for all four architectures).

Besides partitioning of intermediate auctions, overlap of broker routes and number of Allocators in a broker route, the main factor that has impact on the communication overhead is the size of time intervals. For CA and HA we have the main auction interval, and for HA also the sub auction interval. The number of agent interaction messages in CA and HA differ by a constant, proportional to the number of sub-intervals per main-interval and the number of Allocators in each sub auction. But measured as required bandwidth instead of as number of messages, both HA and CA cause a very similar communication overhead. The two different ways to consider communication overhead will be explained below.

In all the three architectures (CA, HA and MB) the communication overhead is proportional to the described configuration issues only. Then we have the CLB architecture, where the allocation is carried out in-line with each connection request rather than on a separate allocation level. That means that for CLB the communication overhead caused by the allocation is proportional to the number of connection requests, the offered load. We could argue that the communication overhead of CLB is also depending on whether the offered load is below or above the target load, because when the offered load raises over the target, the central distributor must tell the requesting SSP that the request has been denied. But in our implementation it turned out that the same communication overhead is required to tell an SSP that a request has been denied as is required to tell the SSP which SCP should be used to perform the request.

There exists two different ways to consider the communication overhead. One is based on the number of sent messages and the other on required bandwidth. The auction architectures place a more severe constraint on the underlying messaging infrastructure when the required bandwidth is considered, because all messaging takes place just before each auction when the bids are sent in, and just after each auction when the allocations are sent out. Besides that, no message with regard to allocation is sent by the auction architectures, the message channel is mostly idle.

The CLB and MB architectures distribute their messaging over time, so from their point of view it is better to consider the bandwidth. They place a lesser constraint on the bandwidth than on the number of messages.
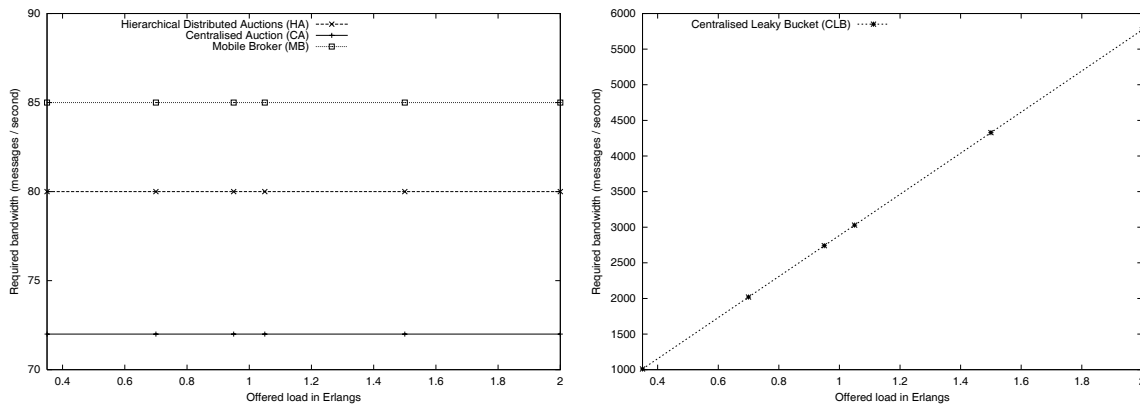
Figure 5.23: The communication overhead caused by allocation for the different architectures respectively.

5.6   Robustness

In this aspect one architecture stands out as the most vulnerable one, the CLB. The other architectures can be configured to assume the same allocation as in last auction in case of a failure in an auction node, or from last broker in case of a failure in a broker. In that way, a stable load that does not require the network to re-allocate the resources can be carried until the problem is repaired. But in the CLB architecture, not even one request can be connected in case of a failure in the central node. Some examples:

In case of a failure in the central resource allocation agent:

- CA - can perform no reallocation of resources, whereas a stable load can be carried.

- HA - partial re-allocation can be performed within intermediate auctions. Hence HA may carry a changing load a little better than CA as it can adapt to some changes in the offered load.

- MB - this architecture has no central agent.

- CLB - immediate shutdown. No allocation can be carried out, and no new service request can be accepted.

In case of a failure in an intermediate distributor (ID):

- CA - this architecture has no intermediate distributor

- HA - no re-allocation can be performed between, to or from the participants of the auction held by the failing ID. Participants to the failing ID can carry the same load

as they did before the ID failed. The other parts of the network continue as usual but with the difference that the central agent cannot re-allocate the resources that is stuck under the failing ID.

- MB - No re-allocation can be performed by the failing broker, whereas the other brokers continue to operate as usual.

- CLB - this architecture has no ID.

As a rule you can say that the more centralised an architecture is, the more vulnerable it becomes, while a more decentralised architecture will only suffer from a partial impact in case of a node failure.

5.7 Fairness

The auction architectures are fair in the sense that they do not disregard any individual Allocator/SSP pair, as long as the load does not change too much between two successive intervals.[1] Under normal circumstances, if an SSP node experiences 10 percent of the total demand, it will get a share corresponding to 10 percent of the network resources. The unfairness that can occur is caused by the delay imposed by the synchronisation events, the auctions. For example, when an SSP node experienced 30 percent of the total load in the current interval but only had 10 percent in the preceding interval, this SSP will have only 10 percent of the resources even though it experiences 30 percent of the demand in the current interval.

The CLB finite queue that operates in a FCFS [2] manner, does not guarantee any fairness. It will always reject bursts that arrive faster than they can be processed if the queue is full, or if the burst holds more requests than there will be vacant slots in the queue. But under normal operation SSPs experience CLB as fair, as it is quite random which requests that will be rejected (or during overload, which requests that will happen to arrive when there is a free slot).

The mobile broker architecture is fair if the brokers use dynamic routes, but if the routes are fixed, a node placed after another node with high variations in load will run a higher risk to run out of resources.

5.8 Scalability

All of the architectures can be said to be open in the sense that it is possible to add or remove SSP/SCP nodes to/from the network. If the architecture is synchronised or

---

[1]See in figure 5.14 on page 29 how CA fails to deliver resources to those SSPs that experience the load during time 400-420

[2]FCFS; first come – first served

asynchronous has no impact on this attribute. However, if the architecture is centralised or distributed does have some significance. At a first glance you may think that in a distributed architecture, addition of another network node will be an easier task because only the local part of the network needs to be concerned. But from a global view, addition of new participants in subparts of the network could cause unbalance in the network. Therefore, if the quality of the network allocations shall be preserved, addition of new network nodes is an easier task in centralised architectures.

Chapter 6

Conclusion

Four multi-agent architectures proposed for managing utilisation levels in distributed computing have been examined, evaluated and compared to each other. It has been shown that all architectures are feasible and manage to comply with the requirements imposed by the Intelligent Network domain. No architecture excels in all aspects, each has its own pros and cons. Some highlights of found differences are presented here:

In section 5.1 Utilisation of resources, it was found that all architectures perform very similar. Only HA stands out by carrying less load than the others, especially during overload. However, that could be compensated for, e.g. by handing out extra tokens corresponding to the amount that gets discarded (i.e, by increasing the target load).

Two important findings were reported in section 5.3.3 Balancing of an unequally applied load. One (that is not possible to find a work-around for) is that asynchronous architectures excel in balancing the load faster than synchronous architectures. The other that distributed architectures cannot balance the load better than centralised architectures. This points at a centralised and asynchronous architecture being the best solution. However, in section 5.6 Robustness, it is found that the centralised and asynchronous architecture CLB is the only architecture that features a single point of failure.[1] In addition it is settled in section 5.5 Communication overhead, that the CLB impose severe constraints on the underlying communication infrastructure as all requests must pass through a single central node and the communication overhead grows linear with the offered load. In a network of a realistic production scale, it will be hard to find the hardware fast enough to process all requests, the central node of the CLB architecture will clearly be a bottleneck of the system.

---

[1] *"This is a single element of hardware or software which, if it fails, brings down the entire computer system. When dealing with high availability, single points of failure are obviously highly undesirable."* Pfister [5], page 393

Future work

One of the things that remain to be done is to find out if it is possible to make the synchronous auction architectures better balance and carry the load by making them semi-synchronous/adaptive. This may be done by decreasing the synchronisation time interval during acceleration or retardation in the offered load.

For the MB architecture the concept of dynamic broker routes need further investigation as the load balancing of MB is not perfect and needs some improvement, and as static routing tables are hard to maintain.

References

[1] Å. Arvidsson, B. Jennings, L. Angelin, and M. Svensson. On the use of agent technology for IN load control. In *16th International Teletraffic Congress*. Elsevier Science, 1999.

[2] B. Carlsson, P. Davidsson, S.J. Johansson, and M. Ohlin. Using mobile agents for IN load control. In *Proceedings of Intelligent Networks '2000*. IEEE, 2000.

[3] S.J. Johansson, P. Davidsson, and B. Carlsson. Coordination models for dynamic resource allocation. In A. Porto and G.-C. Roman, editors, *Coordination Languages and Models*, number 1906 in Lecture notes in computer science, pages 182–197. Springer Verlag, 2000. Proceedings of the 4th International Conference on Coordination.

[4] Turner J.S. New directions in communications (or which way to the information age ? *IEEE Communications Magazin*, October 1986.

[5] Gregory F. Pfister. *In search of clusters*. Prentice Hall, second edition, 1998.

[6] D. E. Smith. Ensuring robust call throughput and fairness for scp overload controls. *IEEE/ACM Transactions on Networking*, 3:538–548, 1995.

[7] The International Engineering Consortium (IEC). Definition of signalling system 7. http://www.iec.org/online/tutorials/ss7/, 2002.

[8] The International Engineering Consortium (IEC). Intelligent network definition. http://www.iec.org/online/tutorials/in/index.html, 2002.

Appendix A

Extract of MARINER Simulation Handbook, 1999

Editors: Åke Arvidsson, akear@itm.bth.se
Brendan Jennings, brendan.jennings@teltec.dcu.ie
Javier

A.1   Service Specifications

Some assumptions:

- Service users never abandon ongoing service sessions, thus it is not necessary to implement the signalling required for premature session termination.

- Processing requirements remain constant for a particular signal over all sessions of that service type.

A.1.1   Service A: Virtual Private Network

Virtual Private Network services create a logical sub-network spanning a single or multiple IN network domains which appears to a specific group of users as a private network, providing the types of services normally associated with private exchanges. In this scenario all calls are controlled by an SCP, which provides facilities such as number translation and call monitoring. In the VPN service to be modelled on the MARINER trial platform, both the calling and the called parties are associated with the same SSP. Figure A.1 presents an MSC for a successful session of this service.
The total number of instructions carried out at each network element are as follows:
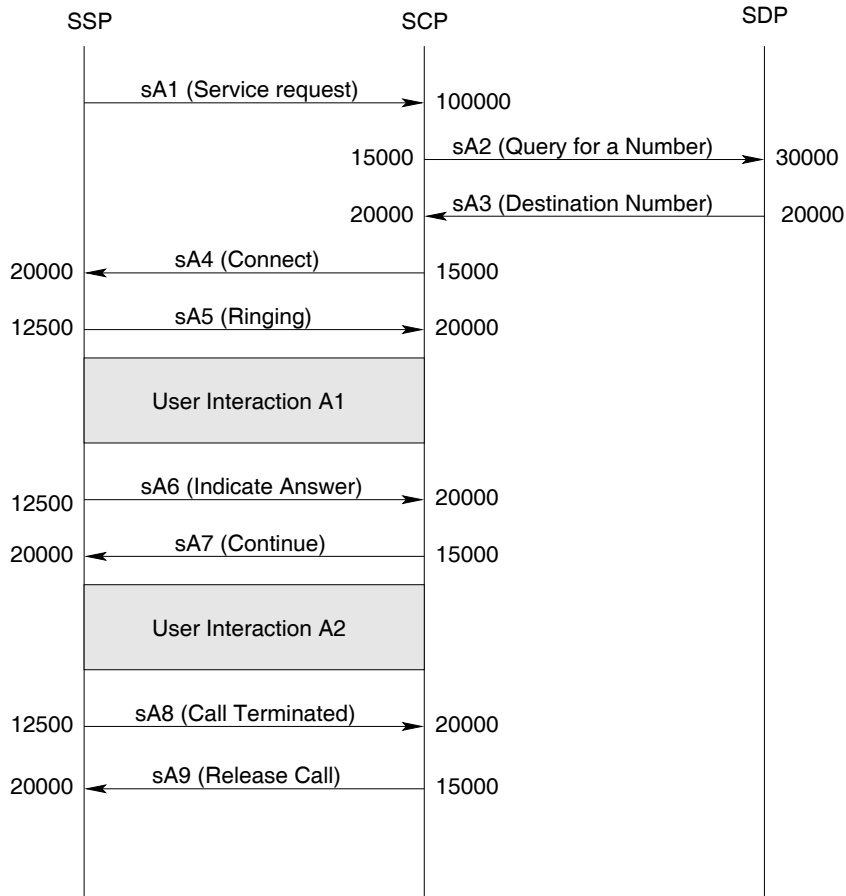
Figure A.1: Service A (VPN) MSC. The numbers are the number of instructions carried out when the signals passes through the processors.

SSP: 97,500 instructions
SCP: 240,000 instructions
IP: 0 instructions
SDP: 50,000 instructions

The duration of 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 user interaction **A2** (Conversation) is to be drawn separately for each service session from a negative exponential distribution with a mean of 100 seconds.
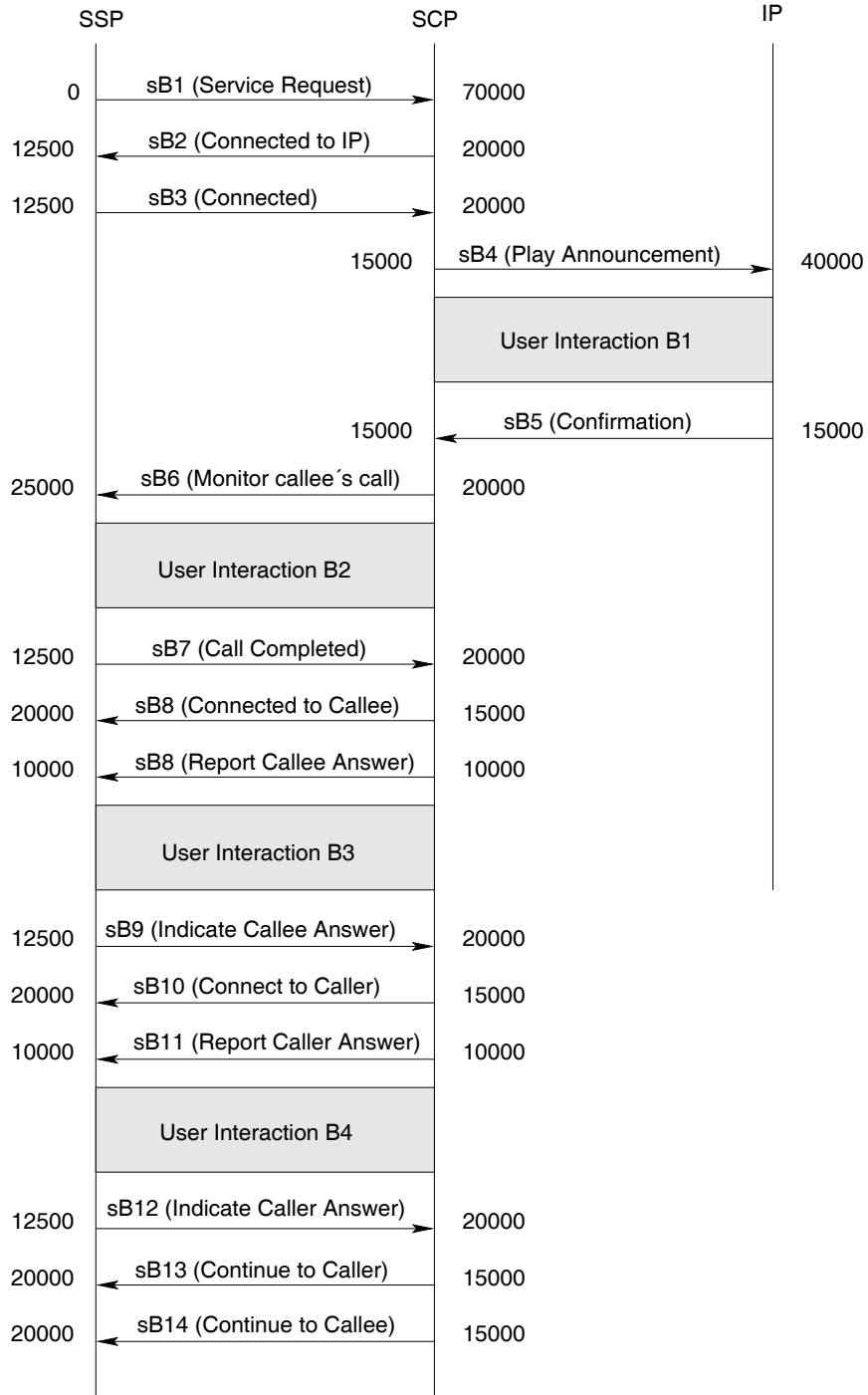
Figure A.2: Service B (Ringback) MSC

A.1.2   Service B: Ringback

Ringback service allow a calling party, upon receipt of an engaged tone for a specific called party, to request that a call be automatically initiated to that callee once his/her current call has terminated. To realise this service the SCP signals the SSP to report when the callee's current call terminates, after which it signals the SSP to initiate a call between the caller and callee. This behaviour is illustrated by the MSC shown in figure A.2.
The total number of instructions carried out at each network element are as follows:

SSP:	187,500 instructions
SCP:	300,000 instructions
IP:	55,000 instructions
SDP:	0 instructions

The duration of user interaction **B1** (Announcement) is a constant value of 5 seconds.

The duration of user interaction **B2** (Ongoing conversation) is to be drawn separately for each service session from a negative exponential distribution with a mean of 50 seconds.

The duration of user interaction **B3** (Callee 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 user interaction **B4** (Caller phone ringing) is to be drawn separately for each service session from a negative exponential distribution with a mean of 5 seconds.

A.2   Network and node specification

The basic network topology to be supported by the simulations consists of 8 SCPs and 32 SSPs, connected by a SS7 network cloud. No packets are lost in the SS7 network and messages traversing it are subjected to a delay of 5 ms.

All SCPs have identical hardware and software configurations and support all services. The processing rate is $56*10^6$ instructions per second and the nodes are supposed to operate at a load of about 0.35 Erlang under normal conditions. The targeted maximum load for an SCP is 0.90 Erlang.

Any delay at SSPs is related to processing only but no waiting take place. The rate of the SSP processor is the same as for the SCP. Note that from a model point of view, there are infinitely many processors at the SSP.