NTNU
Innovation and Creativity

Simulation of Performance Scalability in Pervasive Systems

Hans Inge Berg

Master of Science in Communication Technology
Submission date: June 2006
Supervisor: Poul Einar Heegaard, ITEM
Co-supervisor: Babak Farshchian, Telenor R&D

Norwegian University of Science and Technology
Department of Telematics

Problem Description

Pervasive computing is one of the major improvements we will see in telecommunications. Large amounts of new services are being developed and even more are on the drawing board. But without proper use and knowledge of the network and its topology, we will see unnecessary transmission and usage of network resources and probably encounter congestion and even network failure.

We will define a modeling framework that considers the performance and scalability requirements. When the network topology is established we can define the applications resource needs from the network. Based on these needs we can optimize placement of the applications functional blocks within the network, simulate the behavior and propose new improvements.

The assignment includes the following tasks:

- Define a modeling framework
- Use the framework on at least one scenario
- Try to simulate the implementation of the application on the network with emphasis on the awarelets placement.

Assignment given: 18. January 2006
Supervisor: Poul Einar Heegaard, ITEM

ABSTRACT

As increasingly more services and devices become integrated into pervasive systems, future network topologies will be vastly more sophisticated with numerous heterogeneous devices interconnected. To integrate a new service into this already complex network topology and traffic can give unwanted results if the functional blocks (applets) of a service are not placed at the best suited locations (devices).

This thesis will look into the performance and scalability issues when confronted with options of multiple locations in which to run an applet. We will define a modelling framework taking into consideration system usage, network loads, device loads, overloads, timing requirements and propagation delays to mention some factors. In this framework we are able to set up our own scenarios with user patterns and the amount of users in the system. This framework will be written in Simula. From the output gained from this framework we can improve the system or the applets to improve overall traffic flow and resource usage. The framework will be run on a total of 8 different scenarios based on an airport usage model. We will have 6 static applets residing in their own devices and one dynamic applet which we will try to find the best location for within a predefined network topology. The amount of users can be set to a static amount or it can be a dynamic amount changing from hour to hour. The results produced give a better picture of the whole system working together. Based on these results it is possible to come to a conclusion of best suited applet location.

ACKNOWLEDGEMENTS

Several people have helped me in the period I wrote this thesis. First of all, I would like to thank my Supervisors Poul Heegaard and Babak Farshchian at Telenor R&D. They have both been understanding and patient with my work. Both Poul and Babak have provided me with both invaluable material for this project and reviewing aid. Poul has aided me in the right directions to get this project written as well as possible.
I would like to thank Telenor R&D at Tyholt for allowing me to use their facilities and equipment to do my work.
Torolf Holte at Gardermoen Airport (OSL) deserves a thank you for the information he dug up from OSL regarding passenger statistics.
I would also like to thank my girlfriend Gro for all the support she has given me.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

ABBREVIATIONS

Abbreviation	Description
MIPS	**M**illion **I**nstructions **P**er **S**econd
A4C	**A**uthentication, **A**uthorization, **A**ccounting, **A**uditing and **C**harging
P2P	**P**eer **to P**eer
UPnP	**U**niversal **P**lug a**n**d **P**lay
MSC	**M**essage **S**equencing **C**hart
LAN	**L**ocal **A**rea **N**etwork
WLAN	**W**ireless **LAN**
DAIDALOS	**D**esigning **A**dvanced network **I**nterfaces for the **D**elivery and **A**dministration of **L**ocation independent, **O**ptimised personal **S**ervices
MONET	**M**ultimedia **O**perating Systems and **Net**working
ISP	**I**nternet **S**ervice **P**rovider
PreSim	**Pre**diction **Sim**ulator
DEMOS	**D**iscrete **E**vent **M**odelling **o**n **S**imula
GUI	**G**raphics **U**ser **I**nterface

Table 1: Abbreviations

PART I: INTRODUCTION

I.1 Background and motivation

This section presents the background and motivation for service load simulation in future networks. We start by stating the problems we will look into and why. We will define the thesis topic, define thesis limitations and explain what has previously been done. Last we will explain the methodology of research and angle of approach. During the thesis, whenever possible, there will be references to other sources backing up our claims.

With the ongoing annual growth in network traffic of 70-150% as reported in [1] and [2] and the never ending flow of new heterogeneous devices [8], and more or less useful services emerging. The need for efficient network traffic is ever increasing. With devices becoming more inter compatible and pervasive, they are able to run the same applications [18]. This gives us the opportunity to place programs and applets in different devices[1]. The real challenge is where to place them in order to get the best performance. To find this best possible location, one has to take into account:

- Differences in device resources
- Differences in network capacities
- Potential bottlenecks in the system
- Overloaded devices

[1] This is further explained in PART II:

- Propagation delays
- Network traffic

One might ask why someone would like an applet to run elsewhere than on their device. The answer is as stated in [3], performance. There might be a better suited device to run the applet elsewhere on the network that can outweigh the extra overhead required. Provided the extra bandwidth is available.
Simulations carried out in another project [11] indicates considerable improvements in performance and energy savings, averaging 24% and 53% respectively when using offloading (I.5).

I.2 Why simulate this?

For effective use of resources we need to know something about the topology of the network. What lays where, capacities and potential bottlenecks that can cause problems, and possibly something about how many users will use the system.

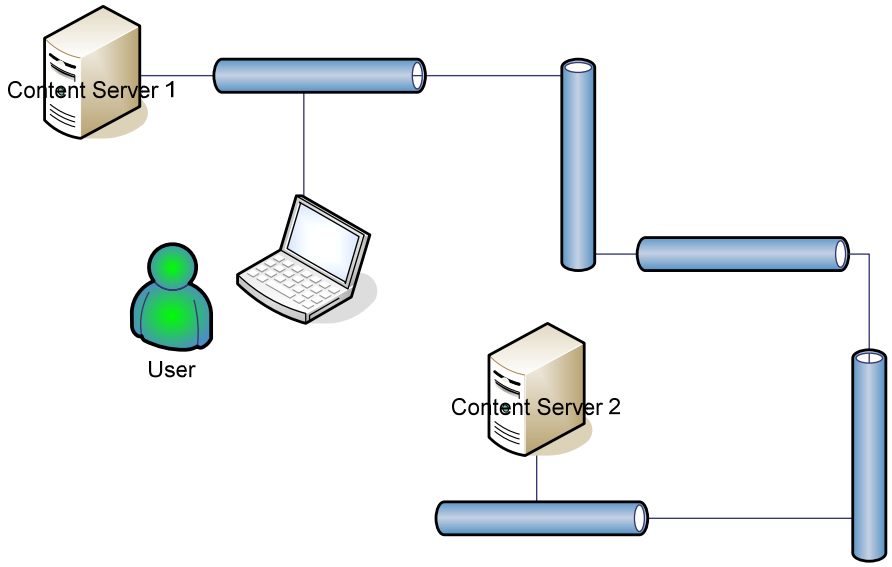


Figure 1 Misuse of network Resources due to bad applet placement

As an example, the user in Figure 1 uses Content server 2, and it would work, until the whole system looked like Figure 2.

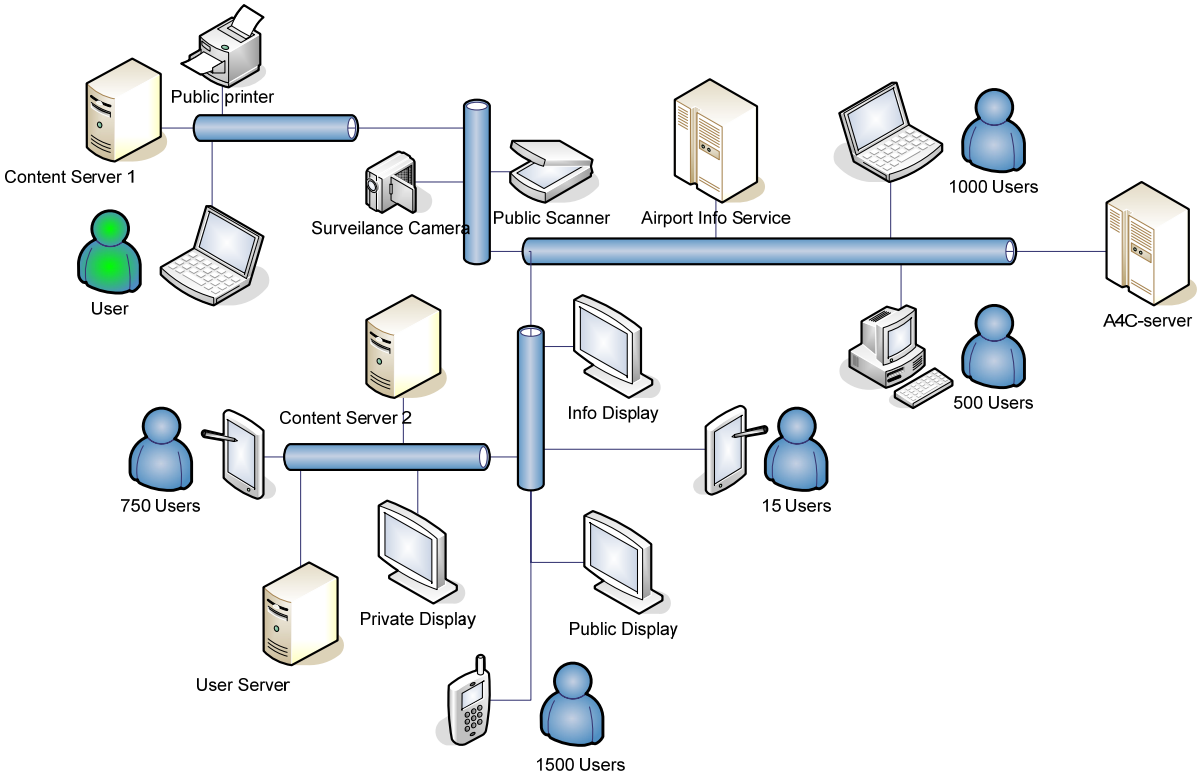


Figure 2 More representative image of a complex network

Here we see a better image of how networks are and will be in the near future, with countless different devices spread over a number of different interconnected networks.

In shared resources will be consumed since it will use bandwidth from all the networks shared by others. If one of these networks is slower and/or has more traffic than the others it can cause a bottleneck that will slow the whole service and system down. If on the other hand, placement of the content applet had been on Content server 1 (Figure 3), Fewer resources would have to be used to get the information to and from the user and Content Server 1. The users' perception of the system would then be like Figure 3:

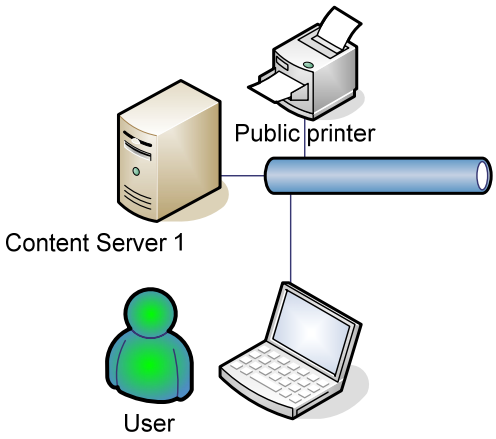


Figure 3 Better applet placement for better resource utilization than

Due to the involved dynamics of deployment, load balancing and traffic, a simulation approach will allow system designers to test, validate and possibly improve their deployment models before any investment in physical equipment is done. Our approach in this thesis has therefore been to investigate how simulation techniques can be used to assist this pre-validation process.

Further details of the simulator model are explained in PART II:

I.3 Thesis Topic

Pervasive computing is one of the major improvements we will see in telecommunications. Large amounts of new services are being developed and even more are on the drawing board. But without proper use and knowledge of the network and its topology, we will see unnecessary transmission and usage of network resources and probably encounter congestion and even network failure.

We will define a modelling framework that considers the performance and scalability requirements. When the network topology is established we can define the applications resource needs from the network. Based on these needs we can optimize placement of the applications functional blocks within the network, simulate the behaviour and propose new improvements.

The assignment includes the following tasks:

- Define a modelling framework
- Use the framework on at least one scenario
- Try to simulate the implementation of the application on the network with emphasis on the applets placement.

I.4 Scope

Time is an issue in this thesis, so we have decided to limit the simulator to an amount of devices and networks one might find at an airport.
We will not consider user roaming and we will assume that the network traffic we generate is the only traffic in the system. We will, based on this, create services as generic as possible. These services will be used in our user defined network topology communicating between devices. Our simulation framework will be able to run these services in a simulated network extracting runtime information. We will do one simulation pass for each location the dynamic applet can be placed in.

I.5 Related work

A number of other projects have looked into the world of offloading[2] [3] in pervasive systems. To my knowledge there are no projects that have measured, simulated, estimated and compared the performance of running the offloaded partitions *in different locations in a network with simulation*. Other projects use the option of asking if some other device is available using a service discovery protocol or UPnP. These other projects primarily look into the methods of committing offloading, but not so much at how the effects of diverse placements throughout a system. [17] Is the predecessor to this project thesis. [17] Took an easier approach, trying to estimate the different loads using spreadsheet calculation.

One way of distributing (offloading) workload is called adaptive offloading [3]. The idea behind adaptive offloading is that one can dynamically partition an application for parts or applets to run on a nearby more powerful device called a surrogate[3]. This will in turn give the initial device less congestion and the service will take less time to complete. The offloading is triggered in [3] when a resource requirement is higher than the available or a resource is running low. This way we avoid congestion or system failure this could have caused. The offloading itself can be carried out pre-emptive of resource deprivation as a precaution or as a way of solving a resource problem after it has occurred. One can look at WLAN as a good example for offloading. If an application requires numerous communications over WLAN, it might be better to offload the class or classes that require this communication pre-emptive of the user entering an area of lower bandwidth.

Adaptive offloading can be the solution to many resource related problems, but it requires that it is possible to partition applications. It would be of importance to know some of the resource requirements before an execution. In [3] classes are used to partition an application. Other projects such as the MONET [6] and Coign [7] projects mention other means of

[2] Shifting responsibilities and or workload from one unit to another

[3] A surrogate acts as a substitute for the parent device carrying out its same tasks. This surrogate usually is a device with idle resources.

offloading, but require adaptations to the applications for them to partition. The Puppeteer project [5] on the other hand does not.
It is impossible to predict how an application will perform or behave in a system without knowing anything about it beforehand. So the best way to find application behaviour and potential offloading surrogates is to simulate with the traffic this application has to coexist with in the whole system, and learn from these results.
A method of measuring and applications resource needs is mentioned by Sverre Hendseth at NTNU (Yet to be published) which is based on taking a small portion of the application and timing it. By doing so one can get a rough estimate of the time it will take to run the application based on the recorded data as well as finding resource requirements.
Once the requirements have been found and offloading procedures have been established. The search for the optimal offloading surrogate can start.

I.6 Research methodology

This chapter describes the research method used in this project.
Based on the project thesis [17], the simulation model was formed. We reused the scenario and its network topology, but changed way of obtaining output from calculation to simulation.
The choice of programming language fell on SIMULA with the DEMOS (Discrete Event Modelling on Simula [13]) SIMULA is a simpler and earlier version of object based languages. It was developed for discrete event simulation, but was later expanded and reimplemented as a full scale general purpose programming language. With the DEMOS add on package SIMULA becomes a good language for running discrete event models.
We had to establish a way of emulating the limited resources in an entire ubiquitous network with heterogeneous devices. To achieve this we decided to see all devices as inter compatible and having equal capabilities except from the amount of resources each device possesses.
We then set up a procedure that enables us to acquire and return resources from each device.
Now we decided to cycle trough the resource reservation once per message, for all messages in a service and for all services in a timeline.
To avoid getting parameters mixed up we decided to include all the processes handling reservation and timeline inside a user entity class.

With the users being sent on their way trough their timelines, we had to sample the amount of resources being used throughout the simulation. This is done with a separate entity thread sampling in parallel with the simulation.
We also needed to set up a routing table based on the network topology. This routing table has to be redrawn for every run because we wanted to keep the topology changeable. The routing method we chose is based on an algorithm created by Prof Dr Edsger Wybe Dijkstra [APPENDIX F: Sources on the internet, Link I].
All the applets to run on the system should be dynamic. Meaning they have to be read in before every run.
To be able to find variances due to randomicity we had to do several runs with different seed.
We also did stress tests to see if the simulator performed well. This is discussed in IV.2 .

GUIDE TO THE REPORT

We will start by defining and explaining the different terms used trough this report. Then we will define the hardware and other testing requirements for PreSim. We will define the model and scenarios on which PreSim is to work, look at how the simulator is built and how it works. We will then guide you trough the different test scenarios and why we chose them. We will present the results of all the scenarios, and discuss them in the discussion chapter. Here we will also look into how one can find the best possible location for a new applet in a network. In the end we will summarize everything leading to a conclusion and an evaluation of both the simulator and the work presented. A user manual, external and internal descriptions of PreSim, extra results tables and graphs are all included in the appendix. If one wants to use the simulator, the user manual is recommended reading for correct usage [APPENDIX A: Using the simulator]. For a better understanding of PreSim, one should read the external view first in Appendix B: PreSim external view before digging deeper into PreSim with the more code related chapter Appendix C: PreSim Internal View.

PART II: THE SIMULATOR

In this chapter we will first look at common terms used throughout this thesis. Then we will look at PreSim's system requirements, go trough the model of which the simulator works upon and then the scenario which is used in the simulation. We will explain why we went for this specific model and scenario. Last we will go trough the way in which the PreSim simulator works.

The simulator name, PreSim, is an abbreviation of "Prediction Simulator" which means to simulate beforehand to be able to predict errors and failures before implementing into a real life system.

II.1 DEFINITIONS

The following definitions are commonly used throughout this thesis.

Scenario

A scenario is the setting in which the simulation run will be performed. This scenario consists of multiple settings of user distribution, timelines, randomization seed, pass, duration and so on. One scenario will be explained in detail, while 7 other will be based on variations of the first.

Run

One run of a scenario in the PreSim simulator is what we will refer to when we say a run. One run will consist of a number passes equal to the number of devices.

Pass

A pass is a part of a simulation run. A pass is then one simulation of the experimental applet placed in one device or one applet location. After one pass has finished, the simulator will reset and another pass is started with the experimental applet in the next device.

Service

A service is a collection of messages going from one applet location to another. A service is often shown as a MSC (Message Sequencing Chart) with the messages going back and forth. The whole collection of messages make up a series of actions required to perform the paramount objective. Illustrated in Figure 4:

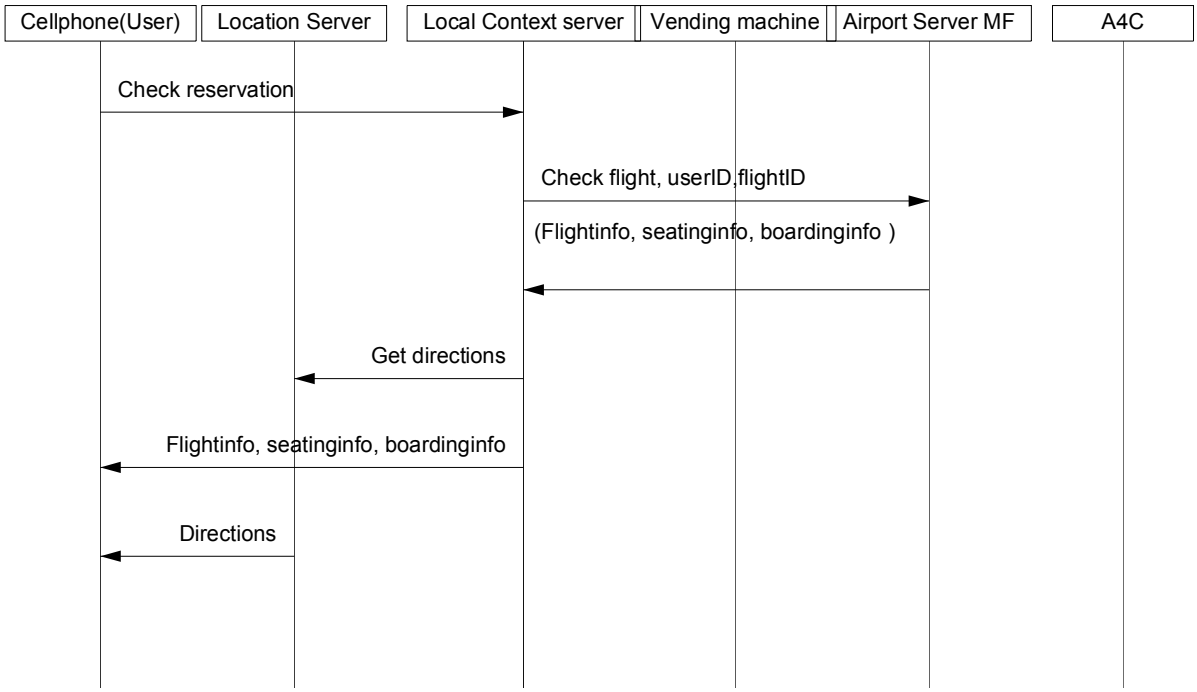


Figure 4 illustration of a service

The illustration (Figure 4) is of a service that checks your reservation. There can be more information going between the different devices, but this is the level of abstraction we have chosen to use in the illustrations and simulator.

Applet

An applet is the application making a device into something useful. In the location server we have a location applet running on a server, by doing so, making this device a location server. An applet can be placed on any device conforming to the applets requirements and do its job from there. An applet is in some cases referred to as a servlet or an awarelet [19].

Applet location

An applet location is an environment in which a applet can be run. This environment must inhibit the applets requirements. A device can be such an environment provided it has the required resources.

Placed applet

A placed applet is an applet that already is given its dedicated applet location. In the PreSim simulator these applets will be locked to their devices.

Unplaced applet

An unplaced applet is what the PreSim simulator will try to find the best locations for. Only one unplaced applet can be defined in PreSim. This test applet will be attempted placed in one device after the other.

Message

A message is the collective term used in this thesis of packets and signalling needed to convey information from a sending part to a recipient correctly. The lower levels of communications are omitted. Messages have additional information like the requirements of bandwidth, timing demands, processing, storage and propagation delay.

Timeline

The timeline defines which services a user will use during their visit to the airport. The timeline is selected using a User Type.

User types

A user type is a generalization of the different people in the airport. In our example there are 5 different user types. See Chapter VI.1.4.1 ”User types and timelines” in “APPENDIX A: Using the simulator” for more info.

Device

A device is a collective term of servers, cell phones, displays, coffee machines and so on. They all have in common that they have available resources for use by users or other devices. They are heterogeneous, pervasive and ubiquitous. Some have battery limitations as well. A

device can also be referred to as a service enabler. In Figure 4 the devices could be: Cell phone, Location server, local server, vending machine, airport server MF and A4C. A service enabler or device is an object that is capable of running an applet with its resources. In our example one can say the cell phone runs a cell phone applet and the Location Server runs a location applet. So in fact we have 7 service enablers and 7 applets. The test applet which is one of the 7 is what we want to place somewhere in the system. A device is sometimes referred to as a functional block.

Network

The network itself can also be seen as a device, but it has only one resource which is bandwidth. So we assume that the networks internal components routers, switches and so on are up to the job and the only limitation we see from the outside is the bandwidth limitation.

Resource

A resource is in this model defined as the limitations like disk storage, processing power and the amount of simultaneous users a service enabler can handle. We only consider bandwidth in each network as a resource.

Model

The model we will use to explain the usability and use of the simulator is of an airport. We use the visitors and their behaviour (use of the available services) as a guideline for network usage. The model is better explained in chapter II.4 "Model explanation"

.

II.2 Hardware requirements

The following list of hardware is needed for the system to work:
The PreSim simulator can be compiled to run in any environment supported by the CIM compiler. During the thesis it has been run successfully in Windows XP and Ubuntu Linux environments. When simulating large environments (100k users) over a long time (24hrs+), the simulations are long lasting and memory hungry. Sometimes lasting for several hours. Shorter runs on the other hand with ex. 1000 users for 2 hours take a couple of seconds to complete on a Laptop computer.

The used simulator environments:

1. Laptop Computer 2GHz Centrino with 1Gb memory, 100Gb ATA, Win XP SP2
2. HP server 2x Dual Core 3.4 GHz Xeon 4 GB memory 3x150 GB SCSI RAID 5 Ubuntu Breezy Linux

PreSim has in both environments been precompiled with cim to use a maximum of 100Mb of memory using the –m100 parameter.

II.3 Testing requirements

PreSim requires some additional files to run. These are: input.txt containing the amount of each resource, the number of users, the simulation runtime, and various other settings (See APPENDIX A: Using the simulator for setup information). Topology.txt is required, defining the topology of the system and the user distribution. If PreSim is to be run with a dynamic user load it requires in addition userload.txt to define the amount of users\hour. Other than this PreSim only requires a platform for which the cim compiler is compatible with.
Being: Windows, Linux and Mac OS, UNIX and GNU.

II.4 Model explanation

The model is based upon the idea that the required amount of each resource is reserved before a service can be committed. Each resource (storage, processing and bandwidth) is a dedicated object of type “res” in SIMULA. This “res” cannot become negative or exceed the predefined value of resource. The individual resources are what found in the output.txt file [VI.5.11]. When a request for a resource is given, a check for available resources is committed. If the required resources are available, we can reserve, then commit and move on. If not, we have to wait some time before trying to get the resource again. Figure 5:

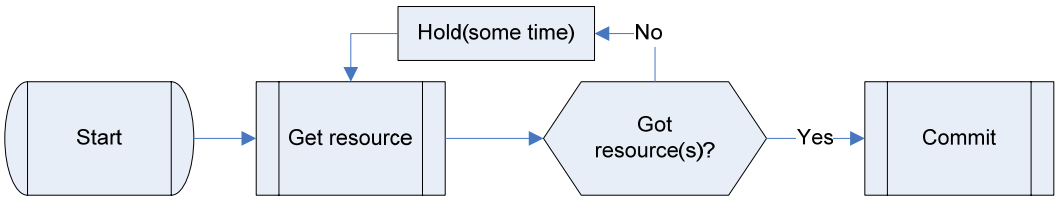


Figure 5 the idea of resource reservation

If the resource is reserved or in use it can not be used by others. If a user is either done using the resource or can for some reason not acquire all the necessary resources required for committing its service, All the previously reserved resources will be released (Figure 6).

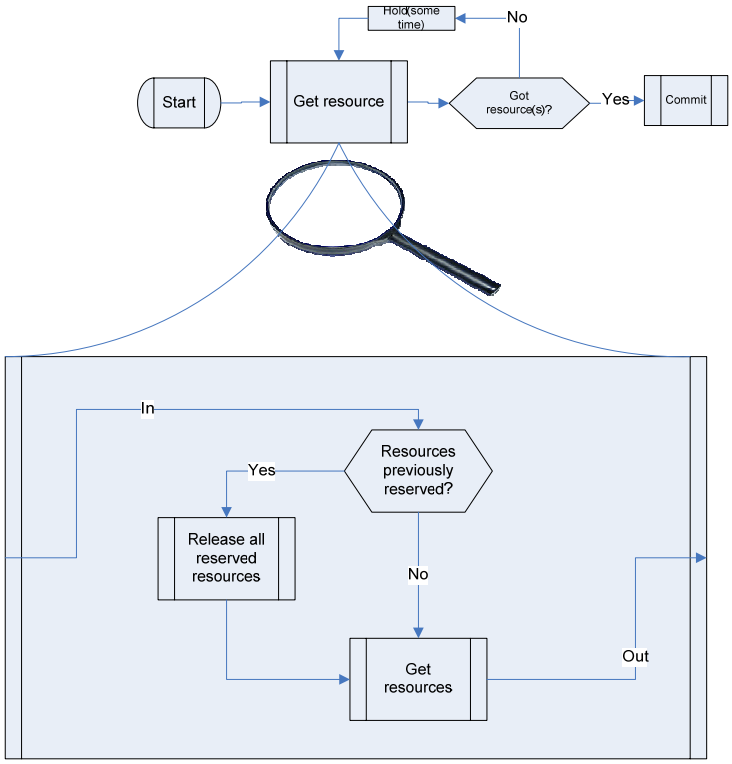


Figure 6 Internals of the "get resource-block" from Figure 5

Users of course have limited patience, so a timeout is implemented after 10 seconds of failed tries and the service is terminated and considered a failure (Figure 7).

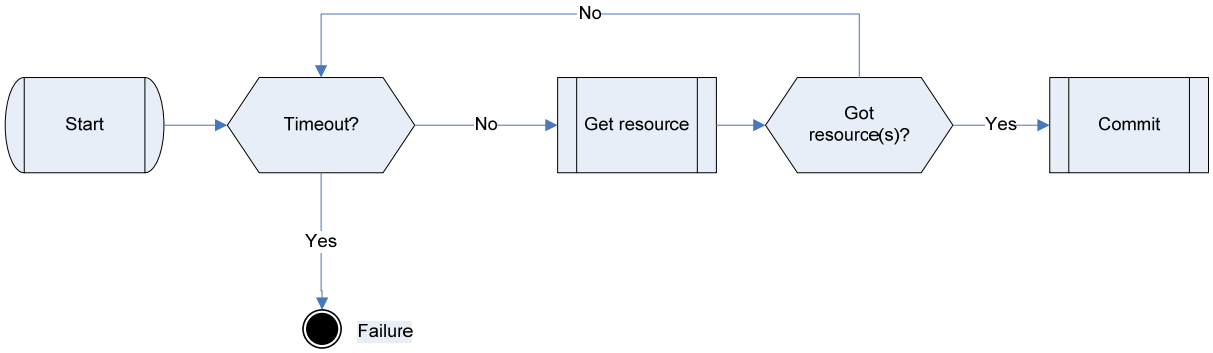


Figure 7 Resource reservation with timeout and failure

II.5 Network model

The Network model is set in an airport. This is because it is easier to limit the amount of different services being used and the amount of users since it is a fairly confined space. The topology of the network can also be scaled down a bit.

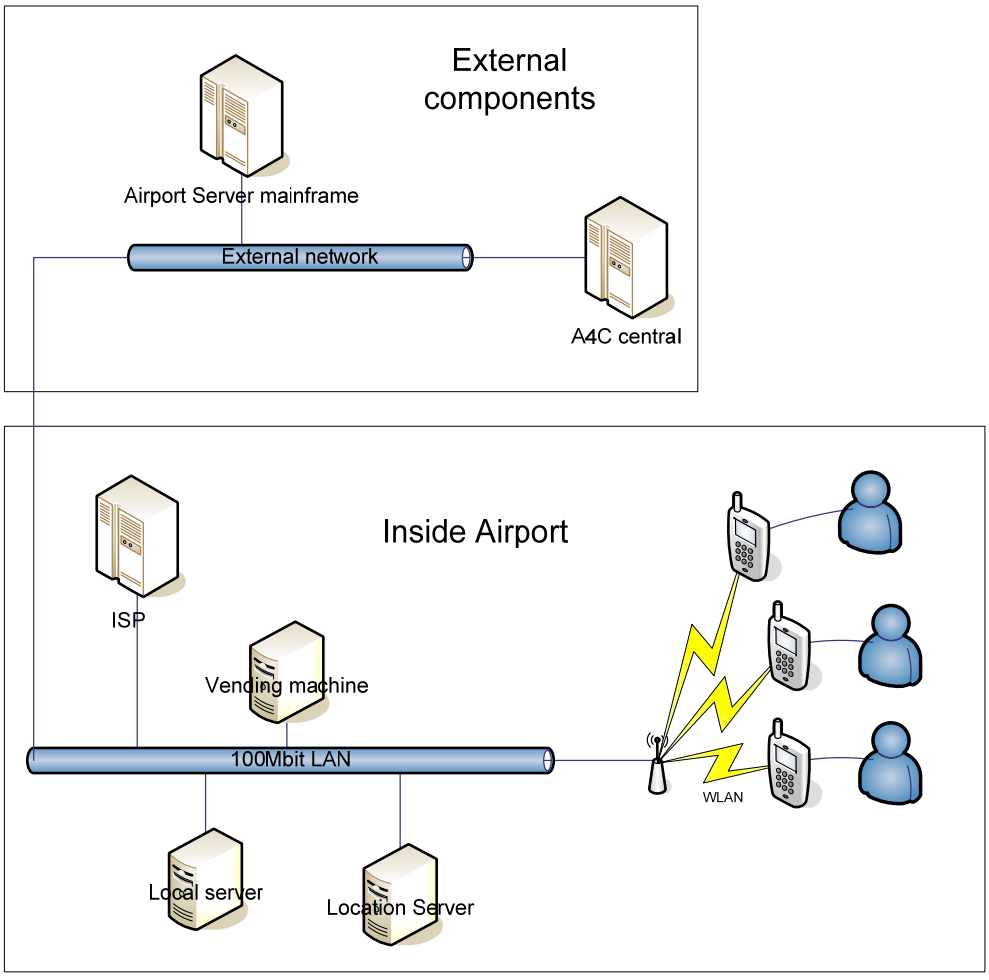


Figure 8 Scenario network topology

Figure 8 illustrates the network we will run our simulations on. The lower part of Figure 8 is the internal components at the airport. The upper part contains the external components located outside the airport. The ISP has a direct connection (faster connection than 100Mbps) to the internal network to minimize strain on the outgoing line.

The deployment model takes care of the placement of the applets into applet locations. The service model then takes care of messages going back and forth between the applets in locations placed onto the networking components. Figure 9 displays the connection between the deployment, service and network model:

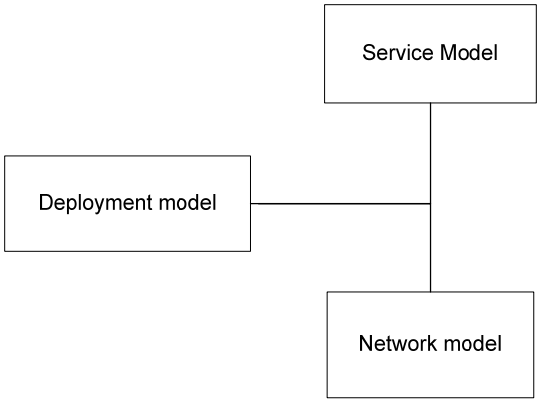


Figure 9 Placement of the functional blocks in each applet is placed by the deployment model into the network model

Services are deployed onto a pre-placed set of applets into devices. These are placed from the deployment model.

II.6 PreSim simulator

PreSim tries to emulate traffic on a user defined network with user defined devices and services. This is done by doing some assumptions:

- We have a predefined maximum amount of users. The amount of users in the system will not exceed this amount.
- Device 1 is always the user terminal and has a battery limitation.
- Networks and devices are error-free
- All devices have processing power, storage and the possibility to transmit
- There are 5 different user types

PreSim is made on the basis that all messages from one user are done serially and all users are asynchronous. Multiple users can send messages in parallel, but each user's messages must be completed serially as noted in the respective service-file.

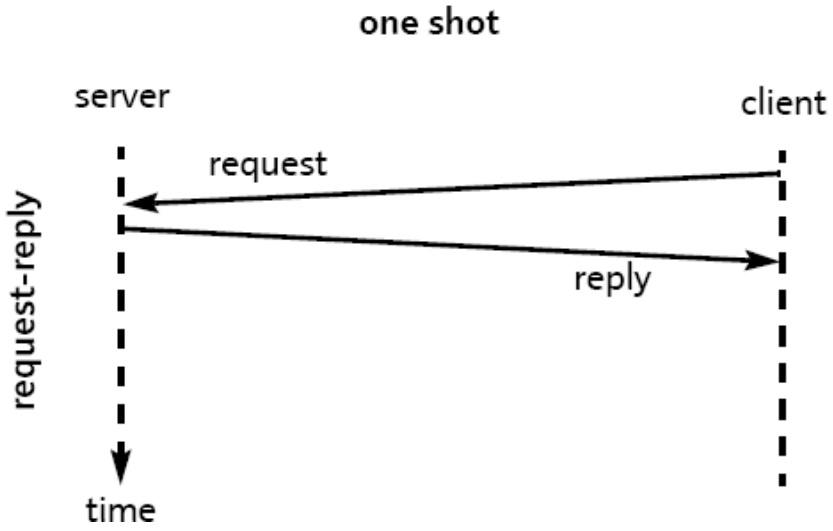


Figure 10 Asynchronous communication [8] between a server and client

For every message in Figure 10, the sender is the client and the receiver is the server.

II.6.1.1 Traffic generation

Instead of dedicated traffic generators for HTTP, FTP, WWW, Speech and so on we use the services themselves to generate traffic by sending messages and holding resources. In other words, the services *are* the traffic. We can do this because the timelines and activities the users are set up to use mimic the actual traffic that would be going on at an airport. There may be more network traffic, but this will then be omitted in the simulations. By doing this we make the test Applet a part of the overall traffic itself instead of an extra component added to the background traffic. Network traffic in real life looks a lot like the graph in Figure 16 which is of visitor arriving at an airport. When simulating with the dynamic user amount. Network load will follow the amount of users in the network, thereby generating realistic network traffic.

II.6.1.2 Message routing

Since the network can be user defined, the routing table has to be set up prior to sending messages back and forth. PreSim sets this table up based on the topology defined.
For message routing in PreSim a version of Dijkstra's algorithm [Link I in APPENDIX F: Sources on the internet] is used;
Dijkstra's algorithm is an algorithm for finding the shortest path between two graph vertices in a graph. It functions by constructing a shortest-path tree from the initial vertex to every other vertex in the graph. In the case of PreSim this means that we use the basic idea of Dijkstra's to set up a routing table.
If say we have a cluster of networks like the one in Figure 11, and would like to find the shortest route from a device connected to network 1 to a device connected to network 6. If we were to use Dijkstra's algorithm, we would start in network 1 and measure the distance to the neighbouring networks.

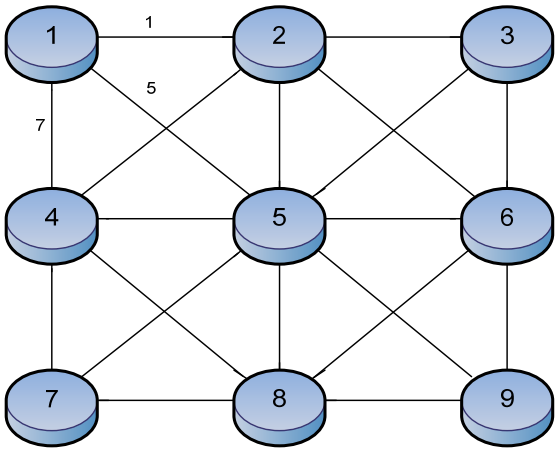


Figure 11 Step 1 of Dijkstra's algorithm

Second we would launch the same test from all the neighbouring networks to its neighbours until we have all routes covered. In Figure 12 we can see all the measured distances in between the networks. With the obtained information we can calculate the shortest paths leading to network 6. The amount of routes will in this case be very high. From every node you will have at least 2 new paths to go. I will not try to calculate how many, but with a similar topology the router in PreSim started 6904881 traces to figure out the shortest paths to

and from any node. With our illustration we get a lot of potential routes as well, but there are only a few which are interesting.
We will only look at the routes which have less than 4 jumps. A jump is defined by a jump from one network to another. From Figure 12 our options are:

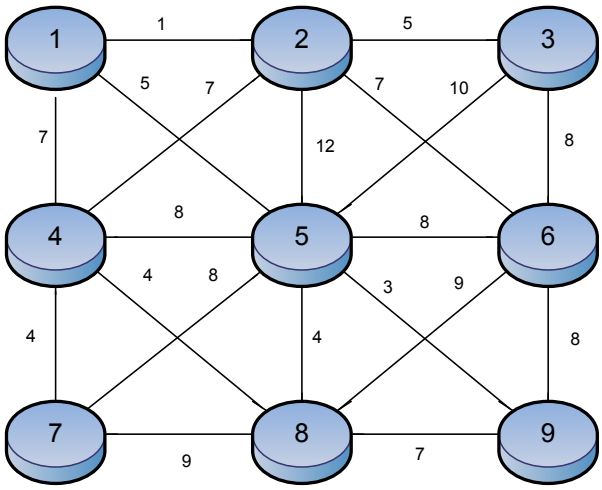


Figure 12 All distances are measured

- 1-2-3-6 gives the cost: 1+5+8=14
- 1-4-5-6 gives the cost:7+8+8=23
- 1-5-6 gives the cost: 5+8=13
- 1-5-3-6 gives the cost: 5+10+8=23
- 1-4-2-6 gives the cost: 7+7+7=21
- 1-2-6 gives the cost: 1+7=8

So the least costly route would have been 1-2-6 with the cost of 8 and 2 jumps between networks.
In PreSim we use a scaled down version of this routing procedure. Since we do not use distance vectors or weighting of networks or connections, we will only consider amount of jumps. The PreSim router will, at start-up, read in the specified network to network connections from the topology.txt file.

```
Network# connected to network#
1 2 5 4
2 3 1 4 5 6
3 2 5 6
4 1 2 5 8 7
5 1 2 3 4 6 7 8 9
6 2 3 5 8 9
7 4 5 8
8 7 4 5 6 9
9 8 5 6
```

Code 1 network connections in the topology.txt file

Place these in a table showing which network is connected to which:

		Network 1	Network 2	Network 3	Network 4	Network 5	Network 6	Network 7	Network 8	Network 9
		1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
Network	**1**	1	1	0	1	1	0	0	0	0
Network	**2**	1	1	1	1	1	1	0	0	0
Network	**3**	0	1	1	0	1	1	0	0	0
Network	**4**	1	1	0	1	1	0	1	1	0
Network	**5**	1	1	1	1	1	1	1	1	1
Network	**6**	0	1	1	0	1	1	0	1	1
Network	**7**	0	0	0	1	1	0	1	1	0
Network	**8**	0	0	0	1	1	1	1	1	1
Network	**9**	0	0	0	0	1	1	0	1	1

Table 2 Routing table created by PreSim's router

A “1” indicates a connection between the intersecting networks

And start routing from device to device until a complete routing table has been established. The PreSim router first starts in device 1 and tries to route to the other devices. Lets again say it tries to route to device 6.

The router first sends out scouts to its connected networks 2, 4 and 5. If these networks are connected to any new networks, the routing will repeat itself from this location. See Figure 13:

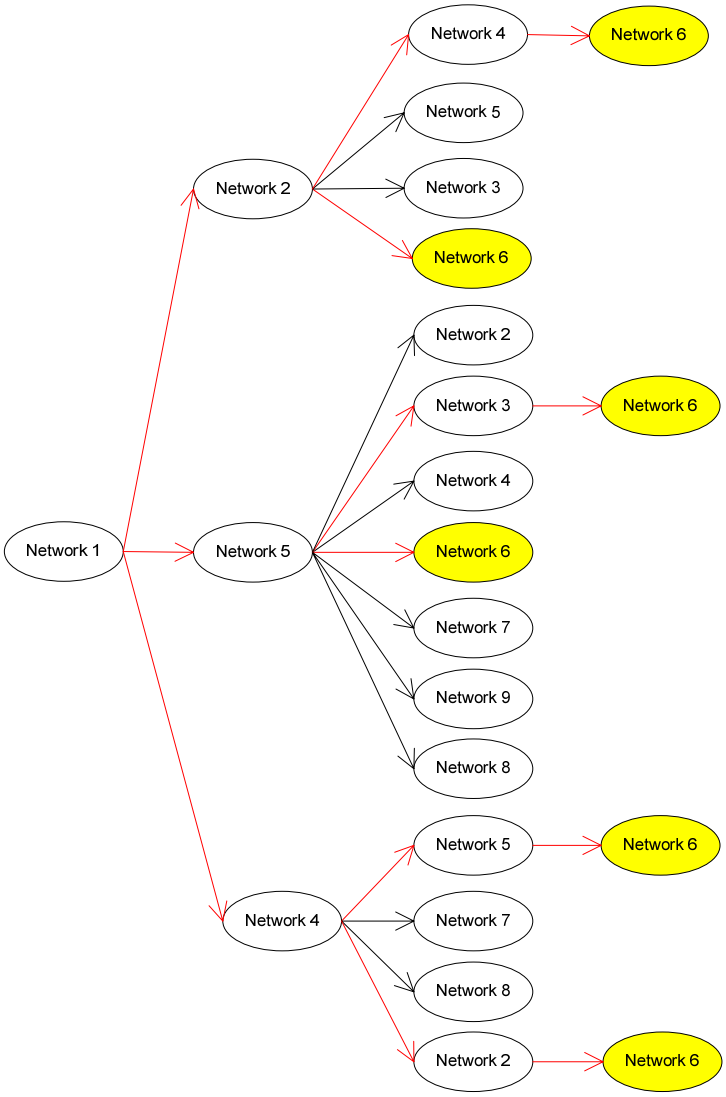


Figure 13 Illustration of the routing mechanism. Misses of the target and further routing options are omitted

Before each new jump the last jump is recorded as jump# where # is the number of the jump. The location the scout was sent from is called the sender, the location we are in now, is called the receiver and the next jumps are called scouts. With each consecutive jump the next jump becomes the receiver and the previous receiver becomes the sender. The receiver is stored as the variable jump# and passed on to the next routing jump. In this way it is possible to avoid sending scouts backwards to the previous location. But it is still possible to keep routing in a circle. So the searching stops after jump# equals the amount of networks. If the receiver is the targeted network, then the path is stored into the routing table at the position that equals the amount of jumps to the target network.

The routing is described more easily in Table 3:

Table 3 Step by step description of the routing procedure

Jump#	Sender	Receiver	Scout	J1	J2	J3	J3	J4	J5	J6	J7	J8	J9
0	0	1	2	1	0	0	0	0	0	0	0	0	0
1	1	2	3	1	2	0	0	0	0	0	0	0	0
2	2	3	6	1	2	3	0	0	0	0	0	0	0
3	3	6	X	1	2	3	6	0	0	0	0	0	0

In Table 3 the first column represents the number of routing jumps from sender. In the positions J# the last receiver is stored. Scout is the next receiver to try to look for a further path.

Now we have found a possible route from network 1 to network 6. So this route is stored in RoutingTable [Code 16] in PreSim as:

RoutingTable (1,6,3,1)=1
RoutingTable (1,6,3,2)=1
RoutingTable (1,6,3,3)=1
RoutingTable (1,6,3,6)=1

In this example the route is stored as:

RoutingTable (sender, recipient, number of jumps, network#)

Networks 1, 2, 3 and 6 are marked as used while the others remain unused for a path with 3 jumps.
If there are other routes that are the same length that are discovered after this one, then the latest discovery will overwrite the previous. In our case there is a route that is shorter, which is the 1-2-6 with 2 jumps or the 1-5-6 also with 2 jumps. So the last one to be discovered, which is 1-5-6 will be placed in position 1,6,2,X with the used networks in the "X":

RoutingTable (1,6,2,1)=1
RoutingTable (1,6,2,5)=1
RoutingTable (1,6,2,6)=1

When the resource manager wants to route something from device 1 to device 8 it has to look up the DeviceConnectionsTable [Code 15] for devices 1 and 8. Device 1 will be "fromDevice" and device 8 will be "toDevice". Let's say that DeviceConnectionsTable(1) is 1 and DeviceConnectionsTable(8) is 6 which mean that device 1 is connected to network 1 and device 8 is connected to network 6. We can the look up the routing table for directions from network 1 to 6 like this:

```
RoutingTable(DeviceConnectionsTable(fromDevice),DeviceConnectionsTable(toDevice),j,n)
```

Code 2 A routing table call

j and n will be used for finding the shortest route by the findWayAndInfo procedure [Code 10 tryToRoundUpResources Procedure. They are abbreviations for jumps and network. The findWayAndInfo procedure sweeps trough RoutingTable starting at RoutingTable(1,6,j,n) with j and n set to 1. Incrementing n until it finds the value "1" (connection to a network, indicating a valid route is present). If not, j is incremented by one and so on until a route is found. When a route is found we have the route giving us the networks we will reserve the required bandwidth from.

II.6.1.3 Users in the system

When we define an amount of users, we define the total *user mass* available for the simulation. This is the maximum population of the airport. This is not to be misconceived with the amount of users in the system. The amount of users in the system is roughly found by this formula:

$$\text{Users in the system} = T_{iu} * T - \frac{T}{\overline{T}_{tc}}$$

Equation 1

T_{iu} = Inter user time. This is the time between each new user in the system.

T = Time is the time since the start of the simulation

$\overline{T}_{tc}$ = average time to complete. This is the average time it takes for a user to complete a given timeline

This amount of users is the number of users using the system, taken from the user mass. The amount of users is depending on the capacity of the system as well as the user load and time elapsed. The user load will create the gradient of the graph in Figure 14.

In the beginning there will be no users in the system. After some time the system either has all the users in the system (Figure 14):

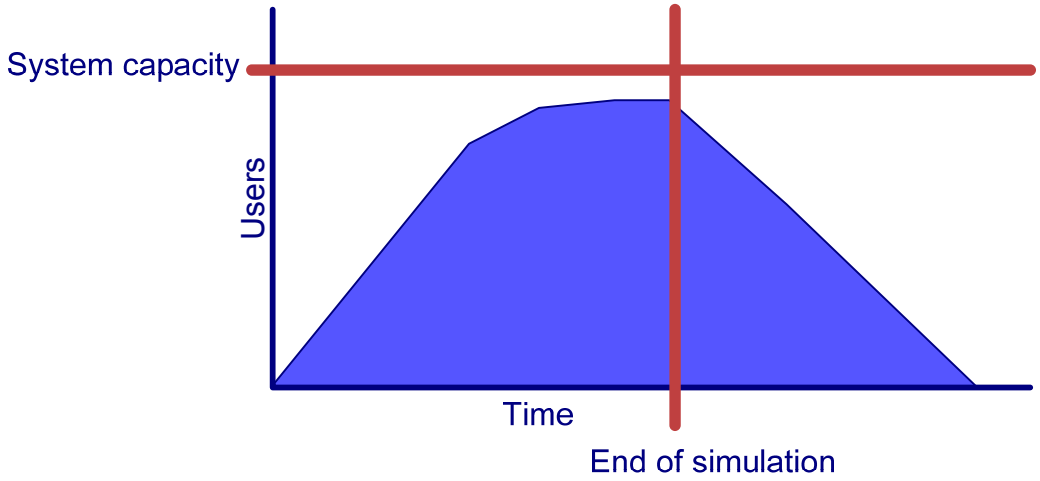


Figure 14 Illustration of a system within its limits of traffic

or some users are on queue to get resources to begin or continue their timeline due to capacity issues in the system (Figure 15).

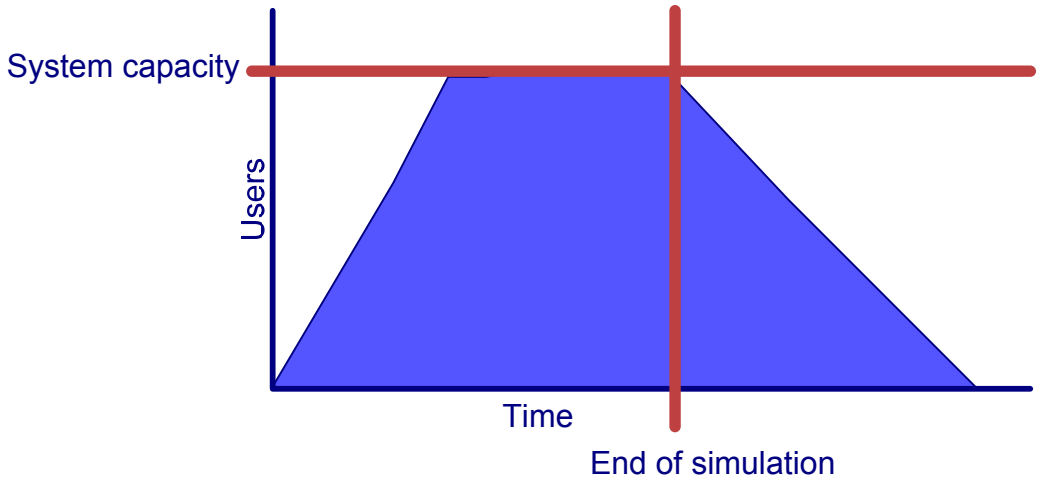


Figure 15 Illustration of a system in which capacity is exceeded and users are rejected.

In Figure 15 one can see what happens when there are too many users consuming resources or the resources are being used in an ineffective way. The result will be resource deprivation. This will trigger a whole chain of problems leading to excessive failures like timeouts and too many retries from users trying to access services. This will keep the overall throughput of the system low until network load has been reduced and users eventually finish their timelines.

This leads us back to the whole meaning of this simulation; to use the resources in the system as effective as possible.
The amount of users arriving at the airport can be static or adjusted to suit the actual amount of arriving visitors at any time. The file userload.txt [Table 31] and Figure 16 includes statistics from Oslo Airport Gardermoen showing the average amount of visitors throughout one day (24hrs).

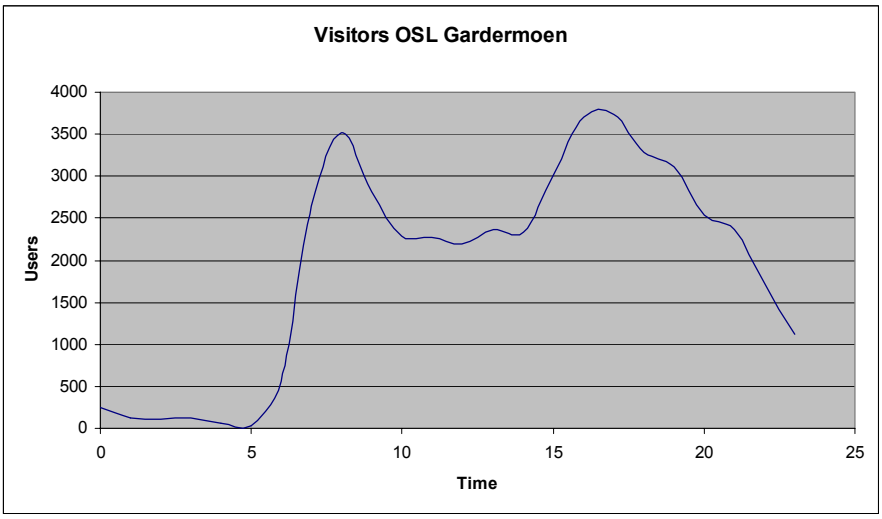


Figure 16 Average amount of visitors at OSL

See Table 31 in APPENDIX I: Misc. for source data.

This file will only be used if InterUser is set to 0 (user load is set to dynamic) in input.txt. Else the given value of users per hour will be used.

II.7 Details of simulation model

The simulator, at time 0 will start to read in the parameters (see Figure 17).

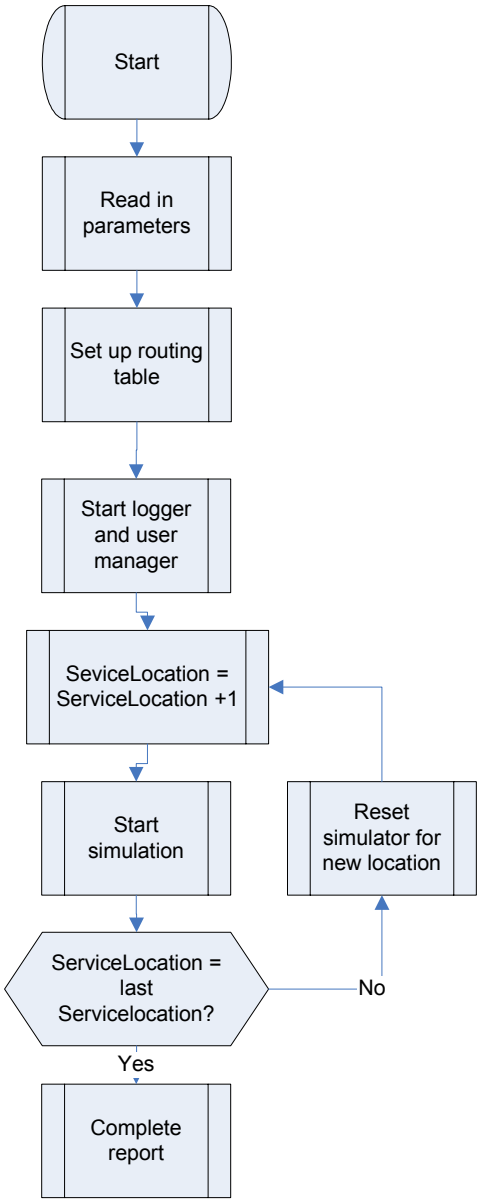


Figure 17 The simulation at a high abstraction level

Then it will set up the required routing-tables from the specified topology-file, the applet (from now on called the testApplet) we will try to get the best placement for is first placed in Location 1 and the simulation can start. A user is then created (Figure 18) with the following information:

- A username
- An amount of battery in the cell phone
- A user type is defined (which services it will use)

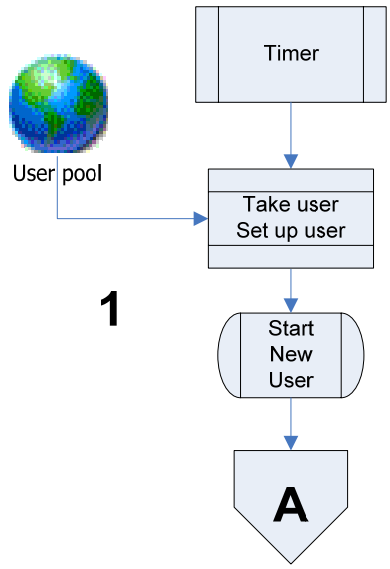


Figure 18 The User starter

User 1 is generated in Figure 18 and can go right ahead and try to reserve its resources (locations 3, 4, 5, 6, 7 in Figure 20). The letters in Figure 18, Figure 19 and Figure 20 link the figures together. When arriving at Letter A in Figure 18 you continue to the incoming block with the A in the other figures. As long as the user does not try to reserve more than the total amount of resources in the devices, this will go just fine.
The next user is started after an "interUserTime" has passed. This user will again try to reserve its requirements of resources, the next user is started and so on. When a whole pass is completed we move on to the next applet location: Figure 19:

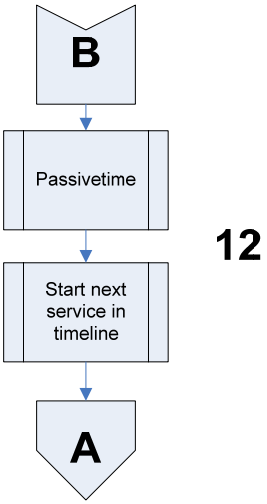


Figure 19 restarting PreSim for a new location

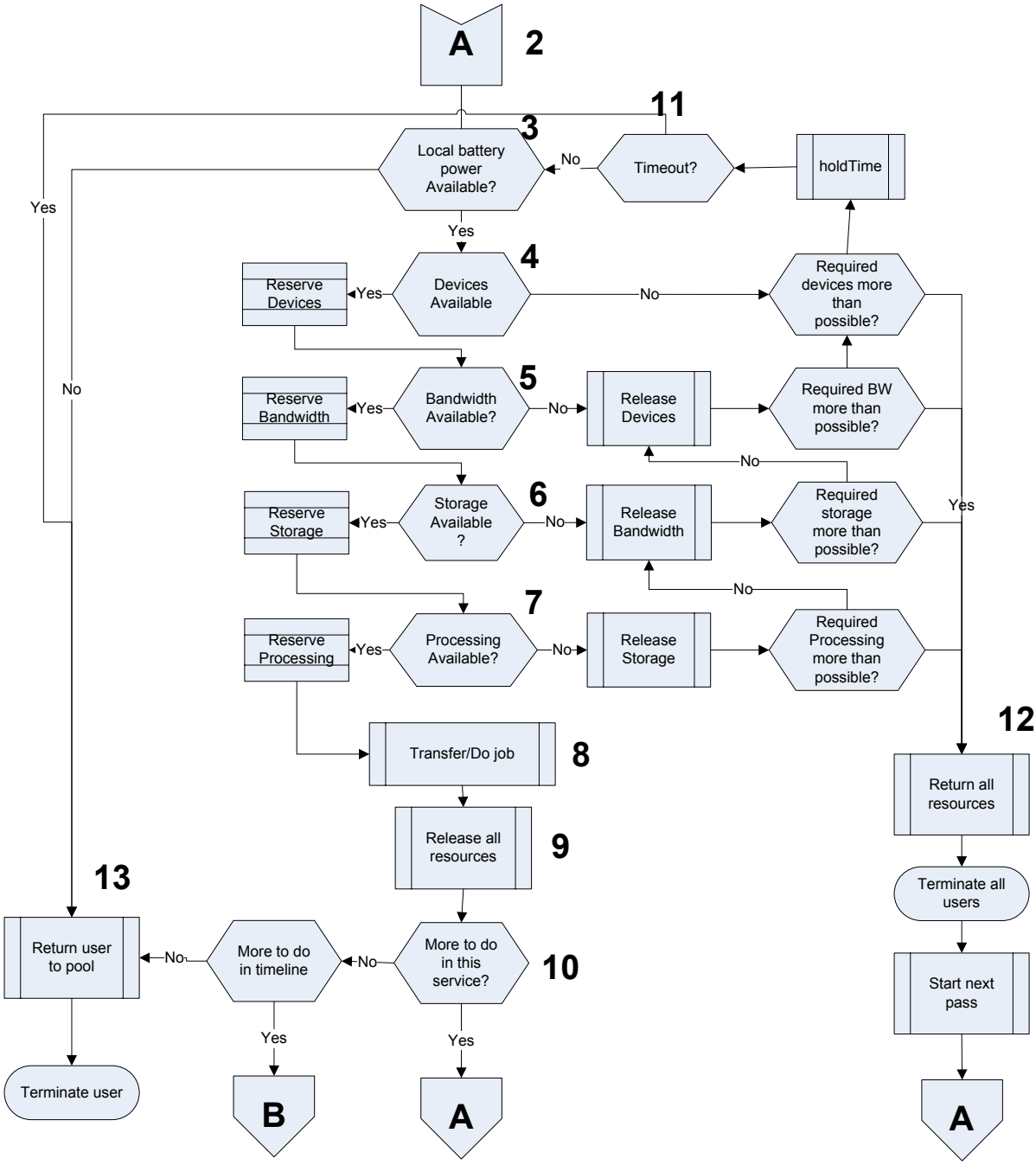


Figure 20 Illustration of the simulation

If a resource reservation fails, the user has to try to re-reserve the resource after a random "holdTime" has passed in location 11 in Figure 20. If the user does not get the resources after the amount of seconds indicated as Timeout in input.txt, the attempt is considered a failure and aborted (13 in Figure 20). If the user tries to reserve more than the total amount of any given resource from a device, the attempt will be classified as a Total failure and both the user's timeline and pass is aborted because it will be impossible to complete (12 in Figure 20).

After the defined time simulation time is reached, we reset the simulator to gather information from the system again (Figure 19), but this time the testApplet is in Device 2. The simulator runs on until the testApplet has been in all the devices in the system.
When the simulation ends and all users are returned to the user bin, the simulator creates a spreadsheet-XML file (output.xml) this can be opened in a spreadsheet editor for reviewing the results and resource usage. The best suited location will then be in the device in which we get the least amount of failures, retries, timeouts and complete failures. All values in this file except the time and the values in the running results are percent used of initial or maximum value. More about the simulation can be found in chapter VI.1 "APPENDIX A: Using the simulator".

II.7.1.1 Deployment model

The simulator tries to find the best suitable location for the local context applet. To achieve this it places this applet in one location after the other and measures the performance. In our system we have 7 devices, 6 of them running their specific applet. So any device dedicated to run an extra applet would have to perform a double role and take on a higher workload. Figure 21 displays the link between MSCs and the scenario.

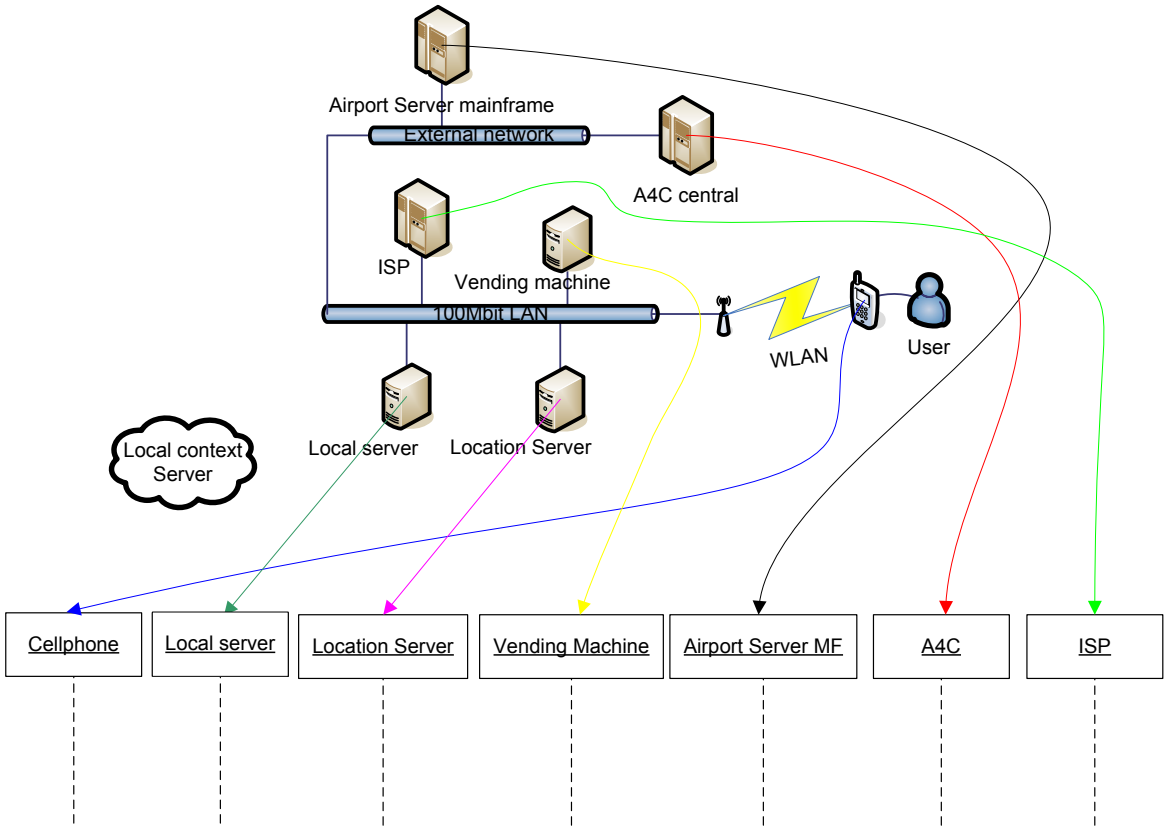


Figure 21 Visual mapping of scenario to MSC. Context applet is not placed.

In Figure 22 one can see how the cell phone has to take on the extra workload of the new applet. Figure 22 and Figure 23 illustrates the system performing a check-in of a user. These are both examples of deployment by the deployment model.

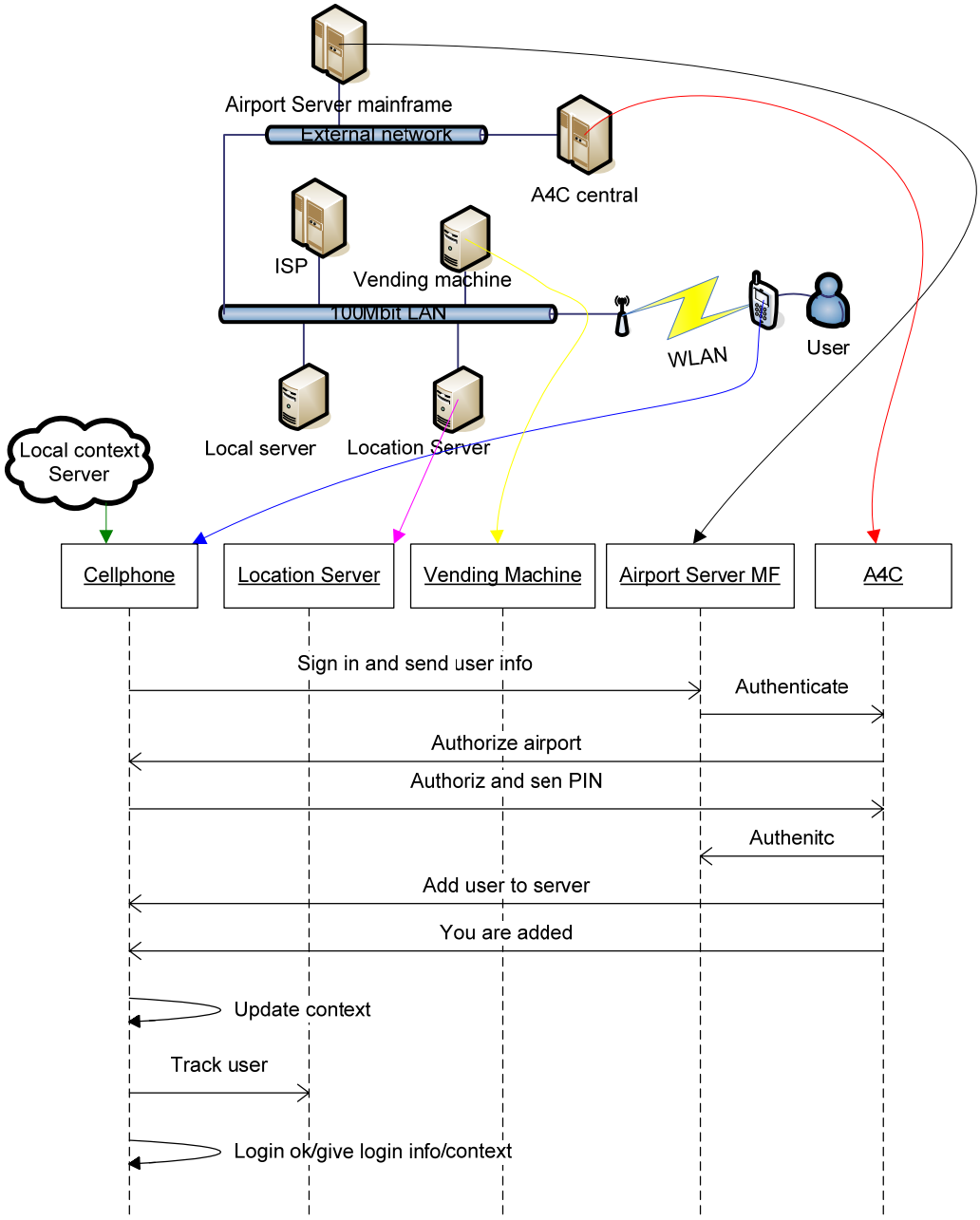


Figure 22 MSC of the context applet located in the cell phone (device 1)

In Figure 23 the context applet is located in the location server. The actual MSC of this service can be found in APPENDIX D: MSC-charts VI.4

.

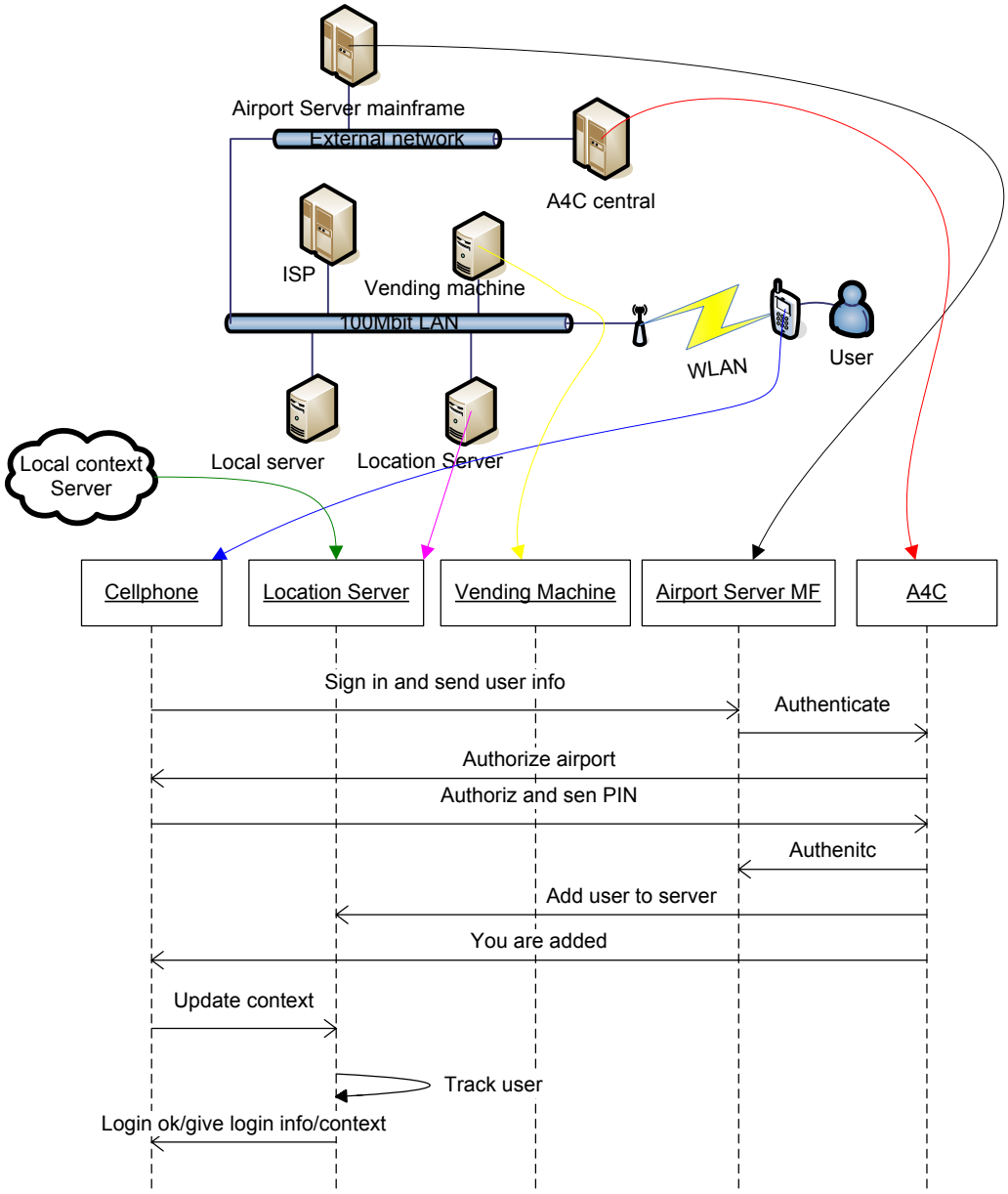


Figure 23 MSC of the context applet located in the Location server (device 2)

PART III: SCALABILITY SCENARIA

We will test some different scenarios to find potential errors in PreSim and to try to gather as much runtime information as possible before drawing a conclusion. First we will try a low stress simulation with 100 users and a runtime of 1 hour. We will admit the same amount of users per hour in all static user load tests which is 10800 users per hour or 3 users per second logging onto the system and starting their timelines, except from scenarios 6 and 7. The network topology described earlier in this thesis will be used in all runs. We will try some different user group distributions in scenarios 2-4, a dynamic user load in scenario 5 and different user loads in scenarios 6 and 7. In scenario 8 we will try a longer run of 96 hours. The scenario descriptions and their set up are better illustrated in Figure 24:

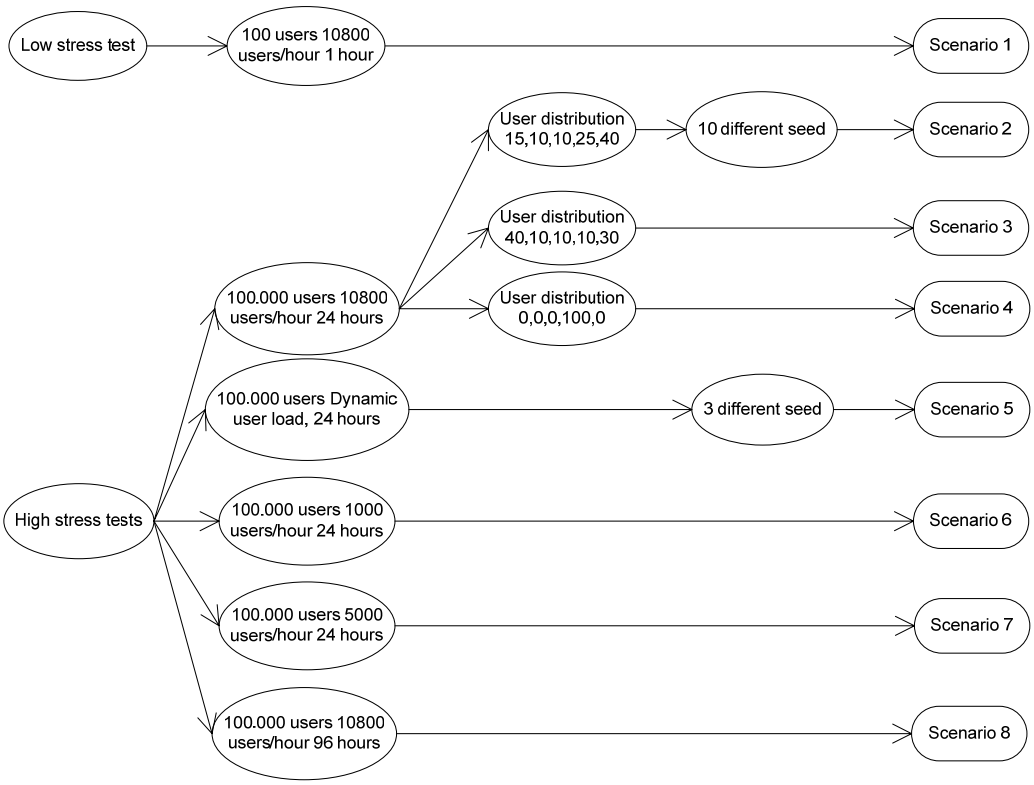


Figure 24 Scenaria descriptions

The scenarios are displayed in Table 4:

Table 4 scenaria overview

DISTRIBUTION in %

Scenario	# users	Dynamic user load	Businesspeople	Backpackers	Earlybirds	Directionless	Parents	User load	seed	Duration [hr]	Comment
1	100		15	10	10	25	40	10800	1	1	Short, low stress test
2-1	100000		15	10	10	25	40	10800	1	24	High stress, seed 1/10
2-2	100000		15	10	10	25	40	10800	41	24	High stress seed, 2/10
2-3	100000		15	10	10	25	40	10800	100	24	High stress, seed 3/10
2-4	100000		15	10	10	25	40	10800	357	24	High stress, seed 4/10
2-5	100000		15	10	10	25	40	10800	413	24	High stress, seed 5/10
2-6	100000		15	10	10	25	40	10800	456	24	High stress, seed 6/10
2-7	100000		15	10	10	25	40	10800	1000	24	High stress, seed 7/10
2-8	100000		15	10	10	25	40	10800	123465	24	High stress, seed 8/10
2-9	100000		15	10	10	25	40	10800	654321	24	High stress, seed 9/10
2-10	100000		15	10	10	25	40	10800	1000001	24	High stress, seed 10/10
3	100000		40	10	10	10	30	10800	1000	24	Different user distribution
4	100000		0	0	0	100	0	10800	1000	24	Different user distribution
5-1	100000	X	15	10	10	25	40	10800	1	24	Dyn. user load seed 1/3
5-2	100000	X	15	10	10	25	40	10800	1000	24	Dyn. user load seed 2/3
5-3	100000	X	15	10	10	25	40	10800	1000001	24	Dyn. user load seed 3/3
6	100000		15	10	10	25	40	1000	1000	24	Lower user load
7	100000		15	10	10	25	40	5000	1000	24	Medium user load
8	100000		15	10	10	25	40	10800	1000	96	Long run

PART IV: RESULTS

IV.1 Scenario 1

Here we have the results of the simulation of the scenarios we set up.
With the given settings there are some problems. You can see this by looking for timeouts, complete failures and a low successful rate.

Table 5 Runtime information from scenario 1

Applet Location	Time Used	Complete Failures	Users Started	Tries	Retries	Successful %	Timeouts	Empty UserBins
1	19390077	100	100	956	100	0	48	4250
2	21197349	0	108	1080	0	100	0	7146
3	22029544	0	103	1030	0	100	0	7164
4	17996672	100	100	963	100	0	39	2832
5	21352827	0	106	1060	0	100	0	7168
6	20382293	0	104	1040	0	100	0	7169
7	23455537	0	108	1080	0	100	0	7068

We can see from Table 5 that placing the applet in device 1 or 4 is not a great idea. This is due to attempting to get more than available resources from a device. In the case of applet location 1 we got the error message on screen:

```
Trying to get more than total Processing from device 1 in service:  1 line:   6
 trying to get:   1500
```

Output 1 Error message from scenario 1, applet location 1

And in case of applet location 4 we got:

```
Trying to get more than total Processing from device 4 in service:  1 line:   6
 trying to get:   1500
```

Output 2 Error message from scenario 1, applet location 4

When trying to get more than available resources, the simulator will end the pass and move on to the next applet location. This is because it will be impossible to place the applet on the specified devices.

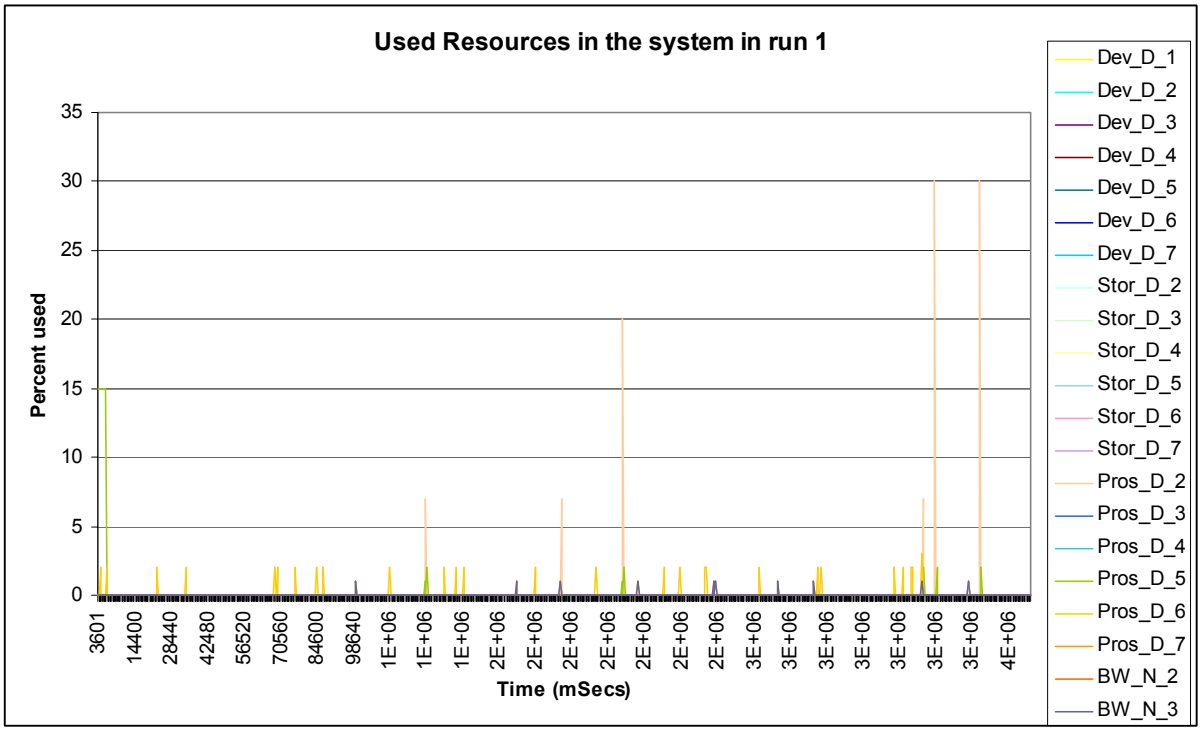


Figure 25 Runtime results from applet location 2. Used resources in scenario 1

Figure 25 displays the used resources in scenario number 1 in applet location 2.

Resource graphs are in APPENDIX H: Result graphs
Other tables are located in APPENDIX G: Extra tables of interest

IV.2 Scenario 2

We can now try to see how many users the system can handle. We set the user pool to 100.000 and see how it goes. We also set the simtime to 24 hrs and enable the short run option. This simulation takes a long time.

Table 6 runtime info from scenario 2

Applet Location	Time Used	Complete Failures	Users Started	Tries	Retries	Successful %	Timeouts	Empty UserBins
1	21541123	670	670	6444	670	0	275	0
2	109668644	0	259200	2671301	647226	76	567925	0
3	109349855	0	259201	2634794	372743	86	329959	0
4	24622855	2779	2779	26702	2847	0	1236	0
5	109232211	0	259200	2608280	38039	99	21759	0
6	109329308	0	259201	2608279	37669	99	21400	0
7	110076199	0	259201	2608014	37786	99	21782	0

Once again, not surprisingly, we get the same errors from scenario number 2. Device 1 and 4 cannot run the applet.
Here you can see the system is saturated since “empty userbins” is 0 in all locations.
If we plot the output of users in the system from all applet locations we get Figure 26:

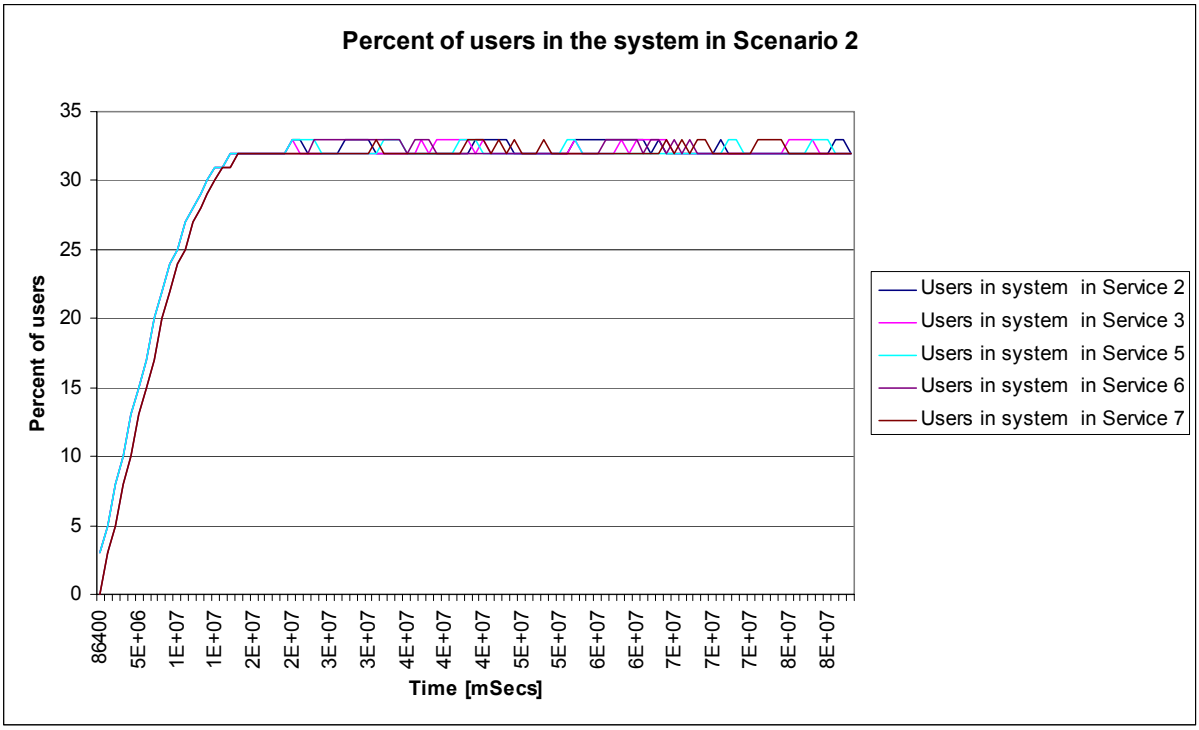


Figure 26 Users in the system in scenario number 2 from all applet locations

This shows us that the system cannot service more than 32-33% of the users we set up as the user pool (100.000). So the user limit for this system is about 33.000 at any given time with a constant traffic load. Applet location 1 and 4 are omitted.

From the resource graphs we get the following information from Applet location 2:

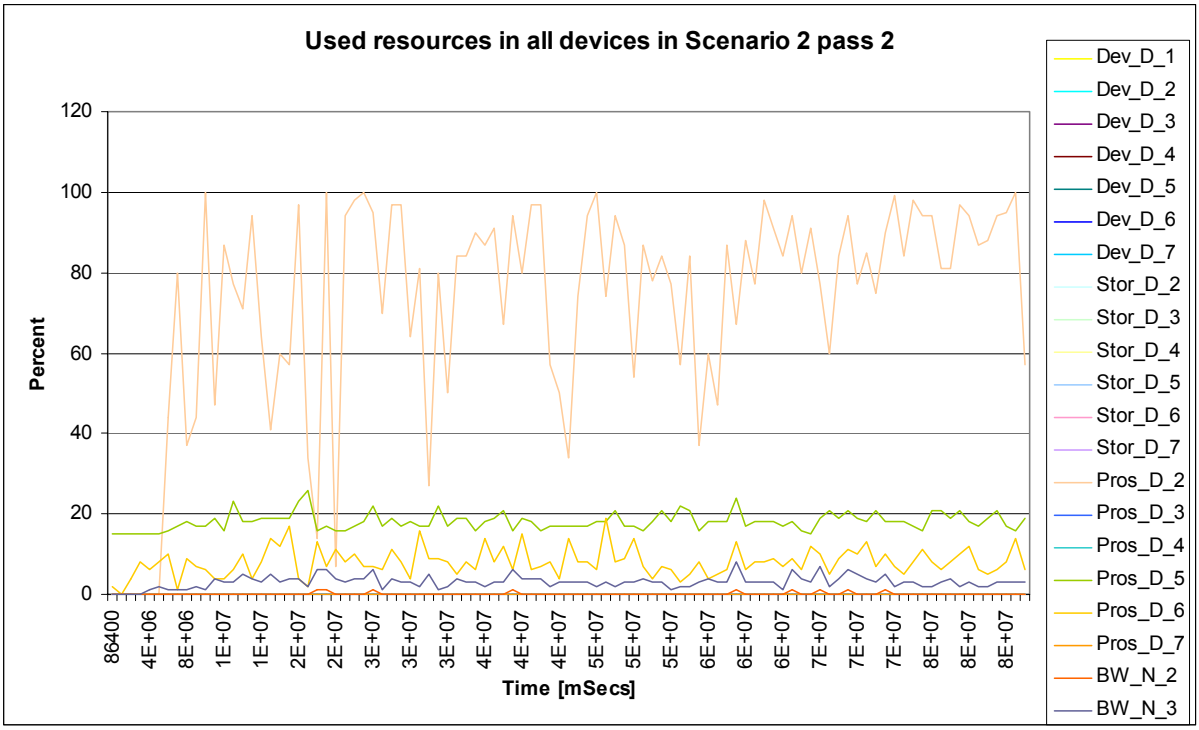


Figure 27 Runtime results of available resources with the applet placed in device 2 in scenario 2

One can see that the only resource that is running low is processing in Device 2. This is better displayed in Figure 28:

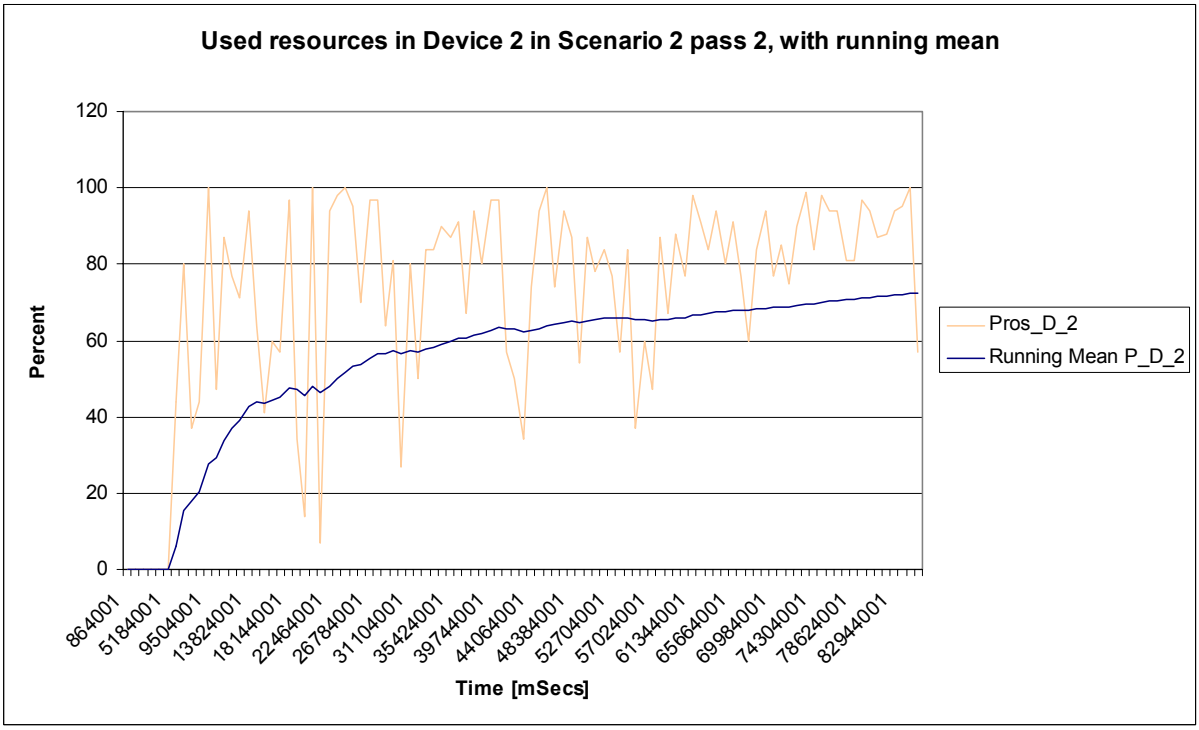


Figure 28 Runtime resource usage of Processing in device 2 scenario 2 pass 2 with running mean

In Figure 28 one can see the running mean of the resource is topping out at 72%. This is a mean value, from the resource graph in Figure 28 one can see the higher values that block incoming resource requests of becoming committed.

The following results are from applet location 3 in scenario 2:

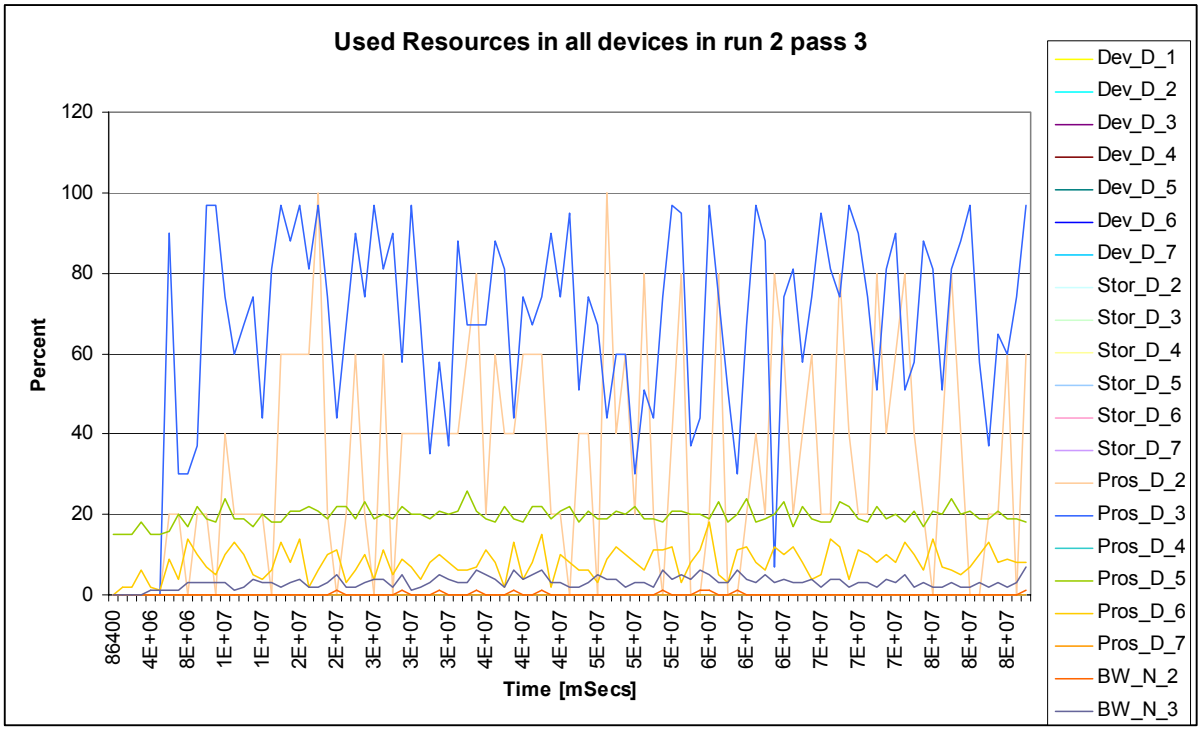


Figure 29 Runtime results of available resources with the applet placed in device 3 in scenario 2

Here you can see that the Processing in Device 3 is running low as well as processing in Device 2. In Figure 30 one can see that some of the load from pass 2 distributed between device 2 and 3.

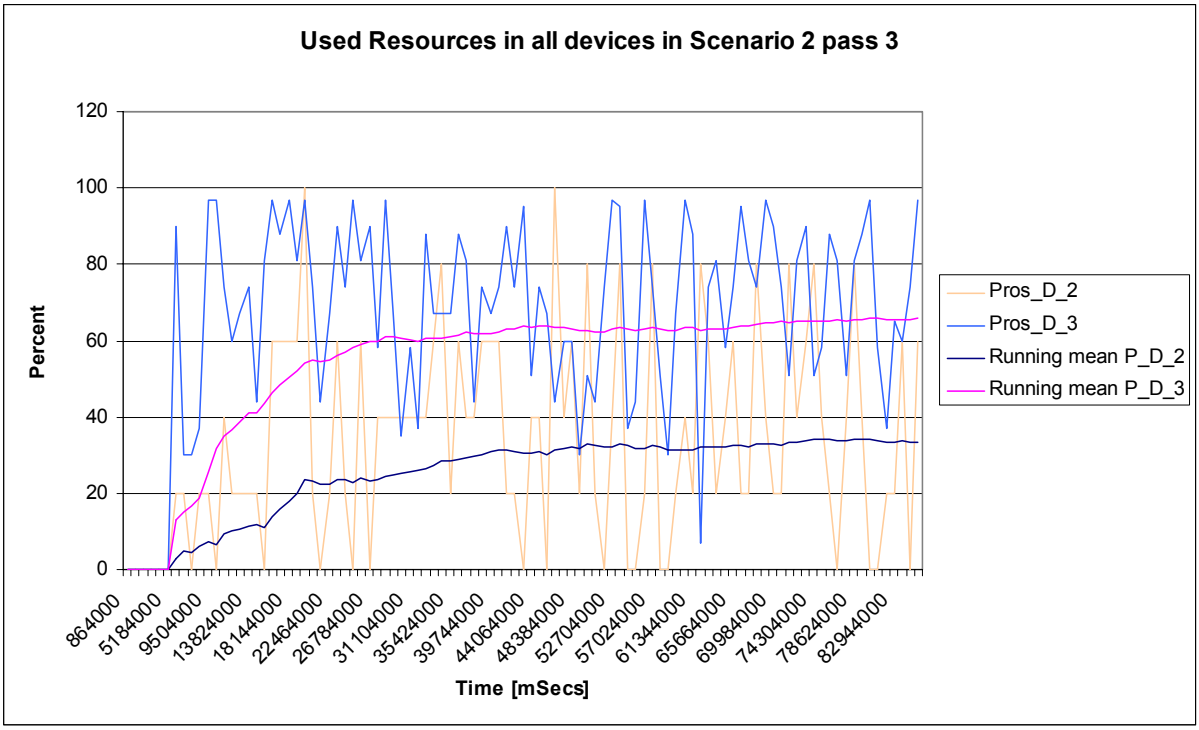


Figure 30 Runtime resource usage of processing in device 2 and 3 scenario 2, pass 3 with running mean

From Figure 28 and Figure 30 you can see the difference when moving the applet from one location to another. In Figure 28 one can see the mean usage of processing in device 2 at around 72, while in Figure 30 this is reduced to around 33% since device 3 is taking the load of the extra applet.

The first run had seed for randomization 123465. In Table 17 to Table 27 in Appendix D you can see the runtime results with 10 different seeds. Giving mean results with variance as shown in Figure 31:

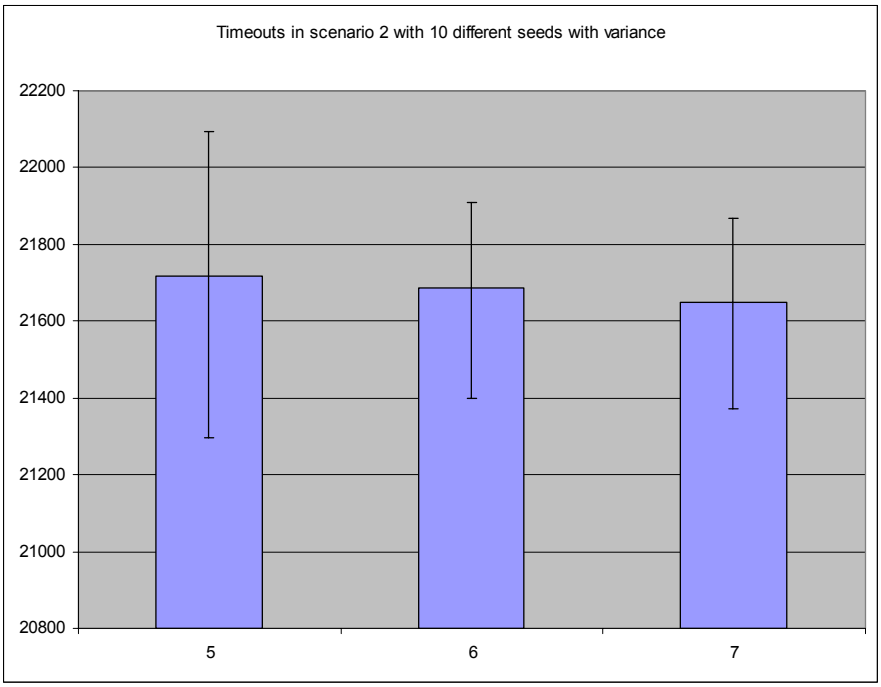


Figure 31 runtime information from 10 runs of Scenario 2. Timeout values are shown

Resource graphs are in APPENDIX H: Result graphs

Other tables are located in APPENDIX G: Extra tables of interest

IV.3 Scenario 3

Table 7 Runtime results from scenario 3,

Applet Location	Time Used	Complete Failures	Users Started	Tries	Retries	Successful %	Timeouts	Empty UserBins
1	23601596	3666	3666	35365	3666	0	1395	0
2	108462932	0	259201	2678270	642968	76	556708	0
3	110126502	0	259200	2648551	376289	86	319738	0
4	24087583	2342	2342	22488	2397	0	1068	0
5	109032992	0	259201	2605957	37464	99	23517	0
6	108824333	0	259200	2606159	37320	99	23161	0
7	108795662	0	259201	2606108	37775	99	23677	0

Resource graphs are in APPENDIX H: Result graphs

Other tables are located in APPENDIX G: Extra tables of interest

IV.4 Scenario 4

Table 8 Runtime results from scenario 4,

Applet Location	Time Used	Complete Failures	Users Started	Tries	Retries	Successful %	Timeouts	Empty UserBins
1	26229434	6591	6591	65916	6597	0	0	0
2	111058826	0	259200	3180618	981846	69	393228	0
3	109914608	0	259201	2935911	456355	84	112454	0
4	26449724	7583	7583	77569	9583	0	261	0
5	109201586	0	259201	2626012	44845	98	10843	0
6	110574891	0	259200	2626210	44829	98	10619	0
7	110074725	0	259201	2626085	44820	98	10745	0

Resource graphs are in APPENDIX H: Result graphs

IV.5 Scenario 5

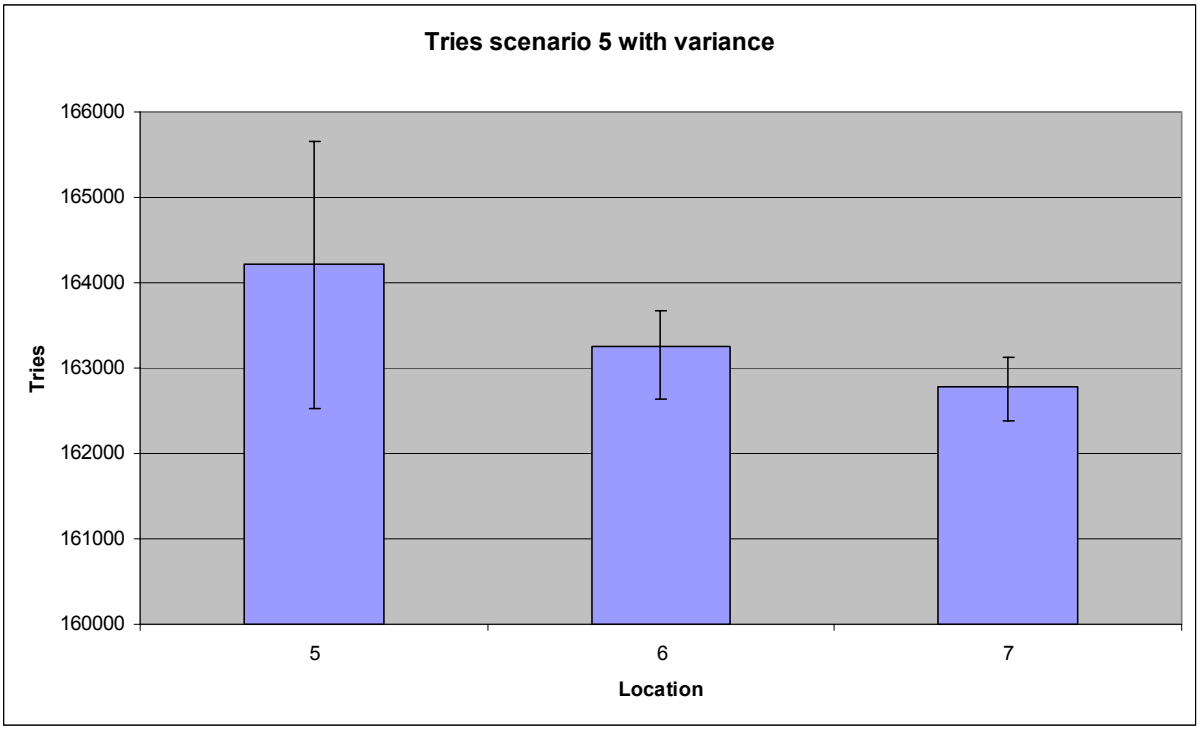


Figure 32 tries in scenario 5 with variance

We get virtually the same results from dynamic test as with the static:

Table 9 Runtime results of Dynamic run 1 seed 1 in scenario 5

Applet Location	Time Used	Complete Failures	Users Started	Tries	Retries	Successful %	Timeouts	Empty UserBins
1	19292699	55	55	531	55	0	20	0
2	103251964	0	16205	162171	340	100	219	0
3	102511163	0	16368	163717	94	100	57	0
4	24314575	99	99	935	99	0	63	0
5	102034479	0	16566	165660	0	100	0	0
6	103128578	0	16343	163430	0	100	0	0
7	100784497	0	16284	162840	0	100	0	0

Resource graphs are in APPENDIX H: Result graphs

Other tables are located in APPENDIX G: Extra tables of interest

IV.6 Scenario 6

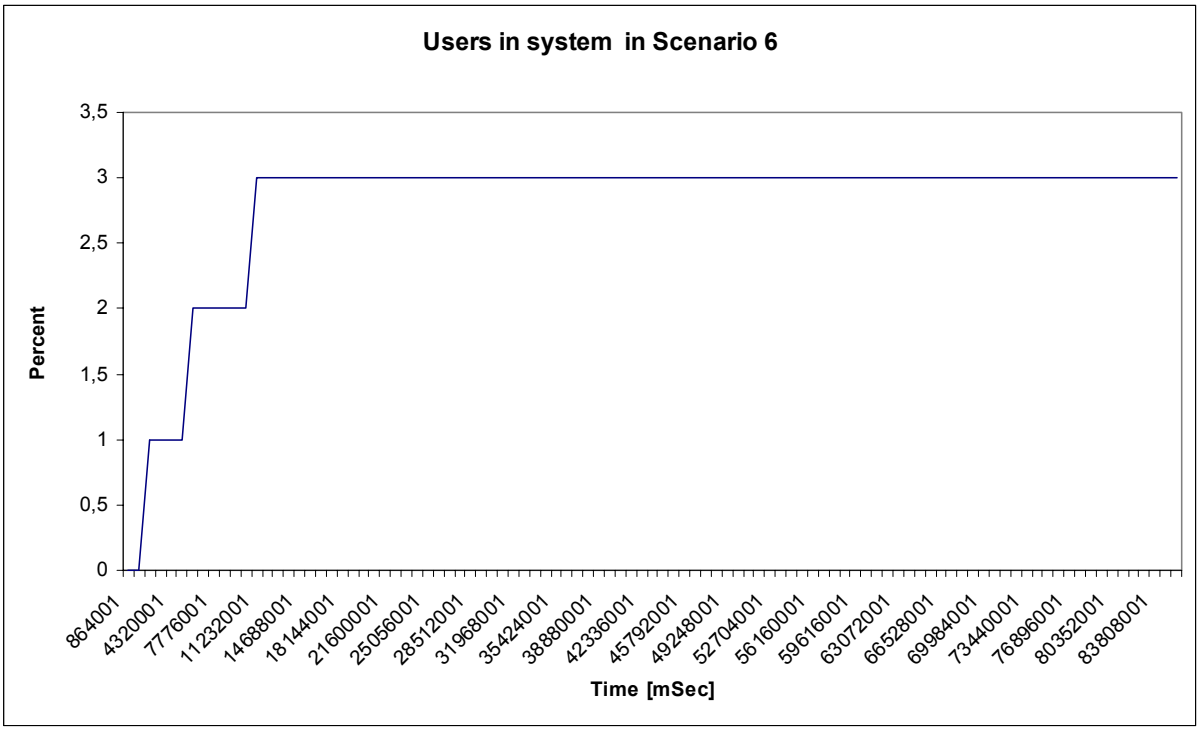


Figure 33 Users in system in scenario 6

Table 10 Runtime results from scenario 6,

Applet Location	Time Used	Complete Failures	Users Started	Tries	Retries	Successful %	Timeouts	Empty UserBins
1	22700482	399	399	3863	399	0	136	0
2	107201216	0	24000	240158	381	100	223	0
3	107729902	0	24000	240062	97	100	35	0
4	22729686	285	285	2756	285	0	100	0
5	107555825	0	24000	240000	0	100	0	0
6	107510923	0	24000	240000	1	100	1	0
7	108043301	0	24000	240000	0	100	0	0

Resource graphs are in APPENDIX H: Result graphs

IV.7 Scenario 7

Table 11 Runtime results from scenario 7,

Applet Location	Time Used	Complete Failures	Users Started	Tries	Retries	Successful %	Timeouts	Empty UserBins
1	23354731	1138	1138	10949	1138	0	456	0
2	109532072	0	120000	1228512	128340	90	99828	0
3	108704694	0	120000	1212187	44540	96	32353	0
4	22738866	1179	1179	11353	1187	0	477	0
5	109362480	0	120001	1200645	1228	100	593	0
6	108708118	0	120000	1200617	1211	100	594	0
7	108914010	0	120000	1200570	1185	100	615	0

Resource graphs are in APPENDIX H: Result graphs

IV.8 Scenario 8

Table 12 Runtime results from scenario 8

Applet Location	Time Used	Complete Failures	Users Started	Tries	Retries	Successful %	Timeouts	Empty UserBins
1	23172400	1024	1024	9778	1024	0	522	0
2	367892518	0	1036800	10776627	2703557	75	2294930	0
3	367904660	0	1036801	10556977	1510626	86	1321659	0
4	21718746	1463	1463	13913	1472	0	823	0
5	368005280	0	1036800	10431595	162392	98	98797	0
6	368560835	0	1036801	10432010	161666	98	97666	0
7	369706510	0	1036800	10431706	162101	98	98395	0

Resource graphs are in APPENDIX H: Result graphs

PART V: DISCUSSION

V.1 Scenario 1

One thing we can observe from Scenario 1 in Table 5 is that the column "Time used" gives times much higher than the actual simulation time we put in the input.txt file. The reason for this is that the simulator waits for all the users to end their sessions in the system before restarting or ending (Figure 34). The "Time used" will not be correct for aborted attempts like the ones in Table 5 applet location 1 and 4.

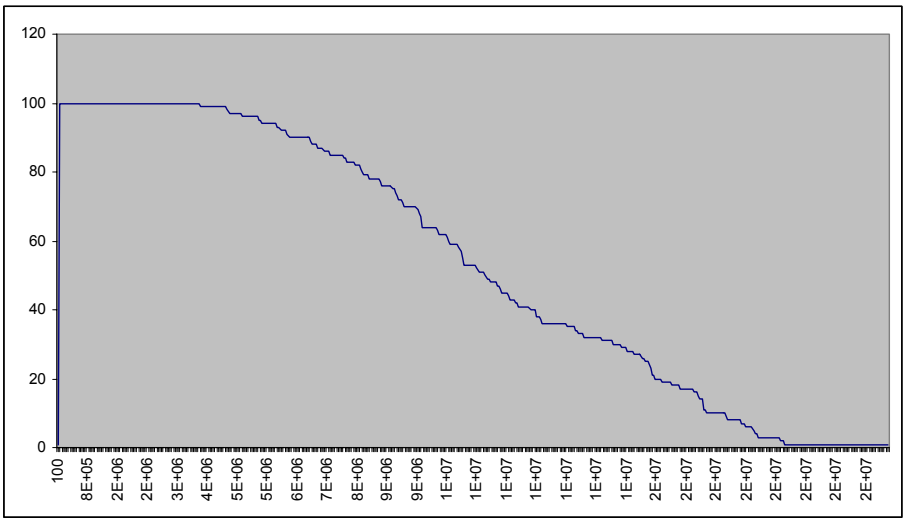


Figure 34 Short run disabled

To avoid this one can enable the short Run option. This will only give you the results from the simulation time. The result will look like Figure 35:

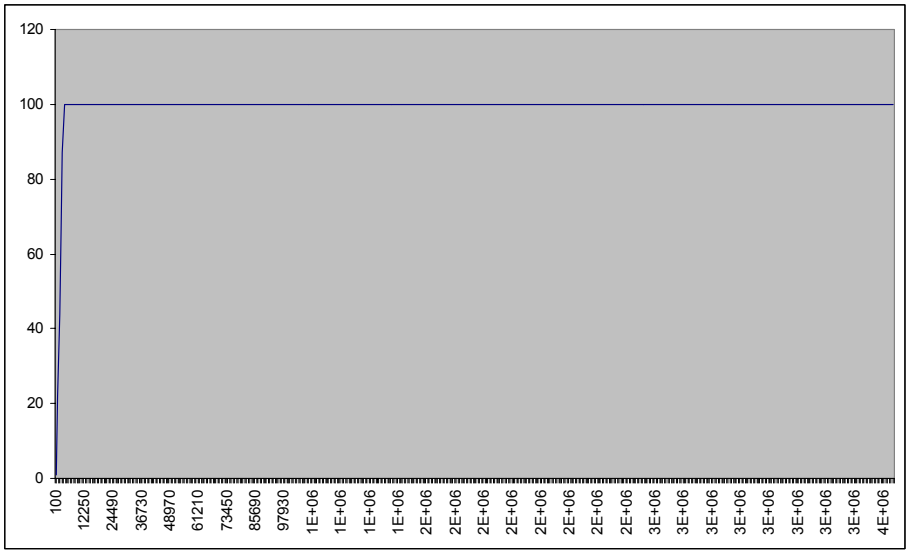


Figure 35 Short Run enabled, percent of users over time

Another point of interest from Table 6 is the amount of "empty userbins" which has high values. This is caused by the simulator trying to start more users than there are available. This could be due to a low amount of users or a too high user rate (users/hour).
In our case it is probably both since the user bin is 100 and we use 10800 users/hour as user rate. Giving each user about 30 seconds to complete their timelines.

$$\frac{100users}{10800users/hour} \approx 30s$$

Equation 2

Which in our case is not enough time and leads to the empty userbins.
The rest of the results from applet location 2,3,5,6 and 7 are satisfactory. Meaning that the resources in the system are well within their boundaries of what they can cope with. 1000 samples was selected here to better see the resource reservations. 100 samples is used in the rest of the scenarios.

V.2 Scenario 2

In scenario 2 we try a higher user load for the first time. We get some interesting results from PreSim. First we get that the highest amount of users in the system at any time is about 32-33% [Figure 47] of the given user pool of 100.000. This can indicate that

A. The system cannot handle more than 33.000 users at one time.
B. SIMULA cannot handle more than 33.000 threads at one time.

From the resource graphs [Figure 46 and Figure 50] one can see that the resources are drained multiple times throughout the scenario. This can be good indicators that it is the system that is maxing out and not SIMULA's thread handling which is the problem. This is further backed up by the results in Scenario 4.
From Figure 48 one can also see a clear difference in the resource usage of processing in device 2 compared with Figure 47. This shows that PreSim is doing what it is supposed to when it is supposed to move the resource form device to device. Based on the changes in resource loads from Figure 47 to Figure 48.

V.3 Scenario 3

Here we redistribute users within the user groups. This should effectively give changes in the resource loads. And it does, but not with any dramatic effects. Comparing Figure 48 and Figure 55 one can see changes in the pattern of resource usage, but the overall changes in for example timeouts in device 5 from scenario 2 to 3 are only 1943 or an increase of 8,9%.

V.4 Scenario 4

In scenario we realise the information desks nightmare. All users are now of the type "Directionless" [Directionless peoples' timeline in APPENDIX A: Using the simulator]. This will increase load on the location server and all other devices and services associated with it[4]. From Figure 57 one can't see any major differences in resource load usage. From Table 8 Runtime results from scenario 4, one can see a reduction in timeouts from scenarios 2 and 3 by over 50%. This is primarily due to the short duration of the get directions service. This also enables the system to handle as much as 43% of the given user pool. 43.000 users can then be handled at the same time, an increase of 10% from scenarios 2 and 3 eliminating option B from the question we asked in V.2 .

V.5 Scenario 5

Here we try the dynamic user load. The meaning of this option is to simulate the real amount of users in a system during a day. The amount of users per hour and the amount of users in the system is controlled by the numbers in the userload.txt file. (Table 31 in Appendix E)

At time 0 the maxusers variable in the simulator is 256. So the amount of users per hour arriving is 256 until the time is 01:00. At 02:00 maxusers and user/hr is 121 and so on. This means that the amount of users will not try to change before after the maxusers and users/hour variables have been changed. This is illustrated in Figure 36.

[4] see MSC II Check Reservation MSC for the associated devices and services

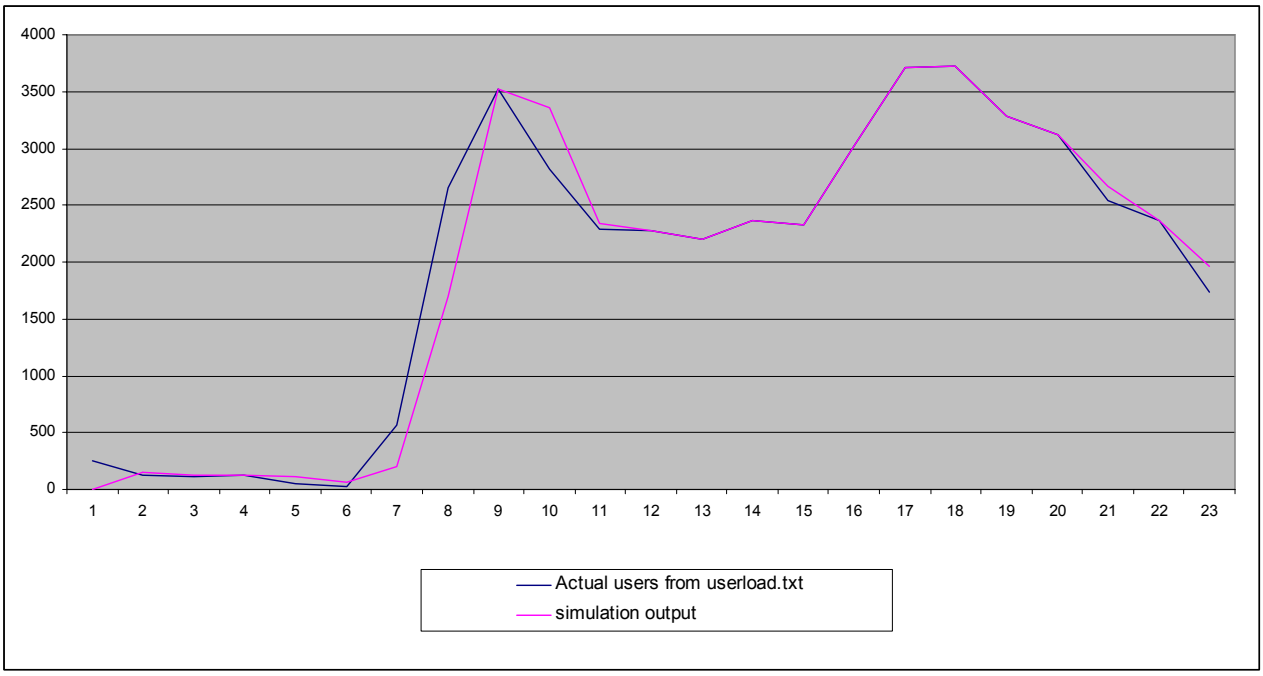


Figure 36 real values of visitors from userload.txt compared to users in the simulator with dynamic user load over time

In Figure 36 you can see how the simulator attempts to mimic the amount of users specified in the userload.txt file. But due to the fact that it takes time for a user to finish, the negative changes (where the previous amount of users is higher than the next amount) are harder to follow.

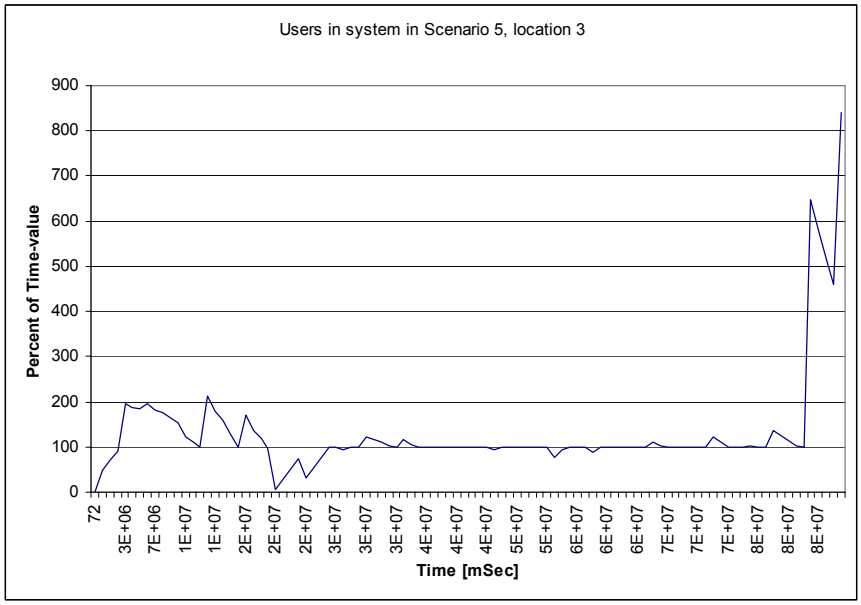


Figure 37 Deviation of given user load value when simulating with a dynamic user load

Figure 37 displays the deviation of the given user load value from userload.txt. When user load changes from lower to higher the value will be lower than 100 and vice versa. This stabilizes after some time.
The same problems that occurred in run 1 trough 4 will occur here as well. The results give the same conclusions as with the static test.
The system seems to cope with the given user loads just fine. This is because the user load never exceeds 3729 users per hour.

V.6 Scenario 6

In scenario 6 we only get about 3300 (3%) users into the system [Figure 60]. This is due to the low arrival rate of the users. Users pass trough the system faster than they arrive. Other than this, the system handles this scenario well.

V.7 Scenario 7

Here we have 5 times the arrival rate as in scenario 6. So the maximum amount of users in Figure 61 is 5 times higher than in scenario 6. From Table 11 Runtime results from scenario 7, one can see that there are virtually no Timeouts or retries. Making this arrival rate good for efficient throughput.

V.8 Scenario 8

This is a longer simulation of scenario 2 with only one seed. From the results produced in Table 12 and Figure 67 to Figure 72 it is not obvious that longer time has any effect on the simulation.

V.9 Finding the best location

When looking for the best suitable location for the applet, we have to look at the runtime results table with a high user load in scenario 2 (Table 13). This is because the lower user load simulation does not stress test [15] the system. We are looking for the worst case scenario this system can handle. Therefore we will analyze the simulation with the higher amount of users.

Points of interest in Table 13 are the low success rate and failures in location 1 and 4 and the higher amount of timeouts in location 2 and 3.

Table 13 runtime info from scenario number 2. Looking for potential bottlenecks

Applet Location	Time Used	Complete Failures	Users Started	Tries	Retries	Successful %	Timeouts	Empty UserBins
1	23388127	1795	1795	17265	1795	0	736	0
2	109800867	0	259200	2671530	648208	76	568678	0
3	110513102	0	259201	2633961	372980	86	331029	0
4	23445592	1743	1743	16728	1761	0	769	0
5	110330886	0	259201	2608356	38067	99	21721	0
6	108684049	0	259201	2607948	37511	99	21573	0
7	108752240	0	259201	2608564	38422	99	21868	0

This eliminates device 1 and 4 as a location for our test applet to run. The resources needed are not sufficient. Locations 2 and 3 are sufficient, but they are outperformed by devices 5, 6 and 7. This is probably due to the higher amounts of resources in devices 5, 6 and 7.

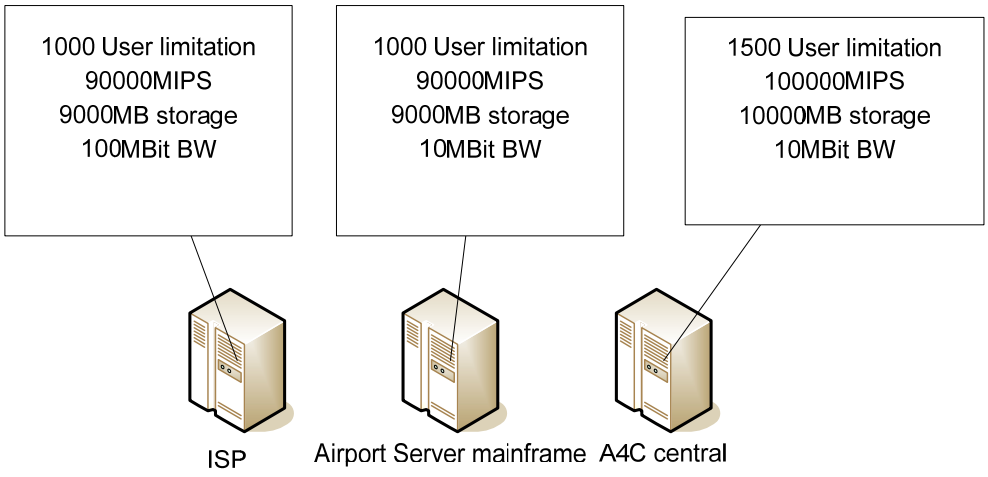


Figure 38 Devices 5, 6, 7 and their resources

From the 10 runs with different seed there is not one location that is the obviously best suited. So we can draw the conclusion that all 3 of device 5, 6 and 7 are well suited for the applet from the simulation point of view. This is easier to see in Figure 31 where the timeouts from all 10 runs are compared side by side and shown with variance.

We can not just look at the simulation for answers. In real life there are more factors that must be taken into consideration. Devices 5, 6 and 7 are

- Device 5: Airport server mainframe
- Device 6: A4C central
- Device 7: ISP

So which one of these would be better suited for the job? My guess is that a service provider as ISP would not take the task of handling user context as well as its primary job of being the ISP. But the A4C and Airport server mainframe could have an advantage in handling context. This is due to their need of user identification and authorization from context when performing transactions and information exchange.
So, when narrowing down to the airport server mainframe and the A4C service provider, we must take into consideration that these two providers have more users than the ones we implemented in our simulation. When taking this into the equation, my guess is that the A4C service provider has more transactions per day than the airport server mainframe. This leads us to the conclusion that the Airport server mainframe will be the location for the user context applet.

V.10 Considering offloading in the future

From the results discussion, a new type of business emerges. Since there in our example are external components better suited for handling the offloading of the test Applet. One can see the potential for setting up servers or server farms for handling customer offloading. This offloading could then be charged as a service for better performance if needed or requested. One can also see uses of pervasive and ubiquitous computing with heterogeneous inter

compatible devices in which one can give, sell or trade your IDLE processing time to nearby potential customers. This can be done much like the ongoing P2P programmes going on around the world today[5], with or without a profit. This has to be done based on preferences and considering if your device has battery constraints, or if you would like to participate at all.

[5] Programmes which use IDLE time on PCs to find answers to complex riddles such as a cure for cancer and the search for extraterrestrial life (SETI@Home)

PART VI: CLOSING COMMENTS

CONCLUSION

We set out to find a way of modelling and simulating a network to measure scalability and performance based on applet placement in a system. This should ultimately lead to a good placement of the tested applet. By creating the simulation framework, PreSim, we have been able to get good qualitative output enabling us to see how this applet would perform in different locations throughout the tested scenario and network. PreSim also gives us good indications concerning which devices the errors occur and due to what in the services. Based on this output we are able to propose improvements to either the network itself or improvements to the services being used on the system for a better flow of information. This tool can be used when one wants to try a new service on an existing or imaginary network to see the effects or to find optimal or better placement of applets. PreSim shows us that applet placement is not necessarily as straight forward as placing an applet as close to the user, instead one must take into account all surrounding factors as well. This can sometimes result in unexpected optimal placement.

EVALUATION

A limitation in this thesis is that we did not have access to a real life pervasive system to test or to compare with. So the results gathered will not yet be possible to verify. Current work in this field may well be in progress, but due to the economic values of such work, it is not distributed freely on the internet and can not be considered in this thesis.
The airport model served as a good model for limiting the amount of factors for consideration. It provides us with the option to generate all the used network traffic without having to make dedicated traffic generators for "background noise". In an more urban model, the treaffic scenario would have been more diverse and traffic generators would be more relevant. The scenarios used in this thesis provide a good basis to be able to draw conclusions on where to place an applet. But more scenarios with even more diverse settings could have been run. Unfortunately this would have made this thesis endless.

Some limitations were for time restriction reasons implemented in PreSim:

- There is a limited amount of different services which is 9.
- There is a limited amount of different applets which is 9. (8 static, 1 dynamic)
- The maximum amount of services in a timeline is 9
- The maximum amount of interconnected networks is 9
- The maximum amount of devices is 9
- The maximum amount of user groups is 5
- The maximum amount of messages in a service is 1000
- User Roaming is omitted

The work on PreSim took more time than expected. This was primarily due to the enormous amount of parallel threads going trough the system. This lead to unexpected problems such as confusion of the position of the reader\writer pointer in files and in some cases trouble with shared variables. In the end PreSim produced good output. From this output one can see very well that PreSim is working as it should, enabling it to be used in trial runs of future pervasive

and offloading systems. The results produced were as expected from a resource consumption point ov view. Given the lack of a real life pervasive systemfor comparison.
A java based GUI was created, but it was abandoned due to the fact that the method of inputting parameters did not become more effective. We found that editing a text file was actually easier and faster. I am sure there are better ways of making this GUI, but time did not allow us to make one. That project will therefore be moved to “future work”.

FUTURE WORK

PreSim is still in an early stage. At this point the way we describe the communication between services is protocol independent. Future work on this topic could be;

- a GUI controlling PreSim, generating services, timelines and user information based on less user input needed for the current version. This GUI could draw the output graphs in real time as PreSim generates them.
- Better descriptions of services with greater detail can be made as well. These services can for example be made from real communications logs between applications over a network.
- Roaming should be taken into consideration. This should be a random physical path based on a users' timeline.
- PreSim can be expanded to include more devices, services and networks.

Sometime in the near future when a real life pervasive system is created, one could test PreSim versus the real system to see if the results are correct.

APPENDIX

VI.1 APPENDIX A: Using the simulator

In this chapter we will describe how to set up and use the simulator.

To use the simulator one must:

1. Find a suitable scenario and its network topology
2. Define resources
3. Define services
4. Set up timelines
5. Set up the remaining parameters

VI.1.1 Find a Suitable scenario and its network topology

First one will have to say how many devices and networks there are. Let's use the example described below in Figure 39.

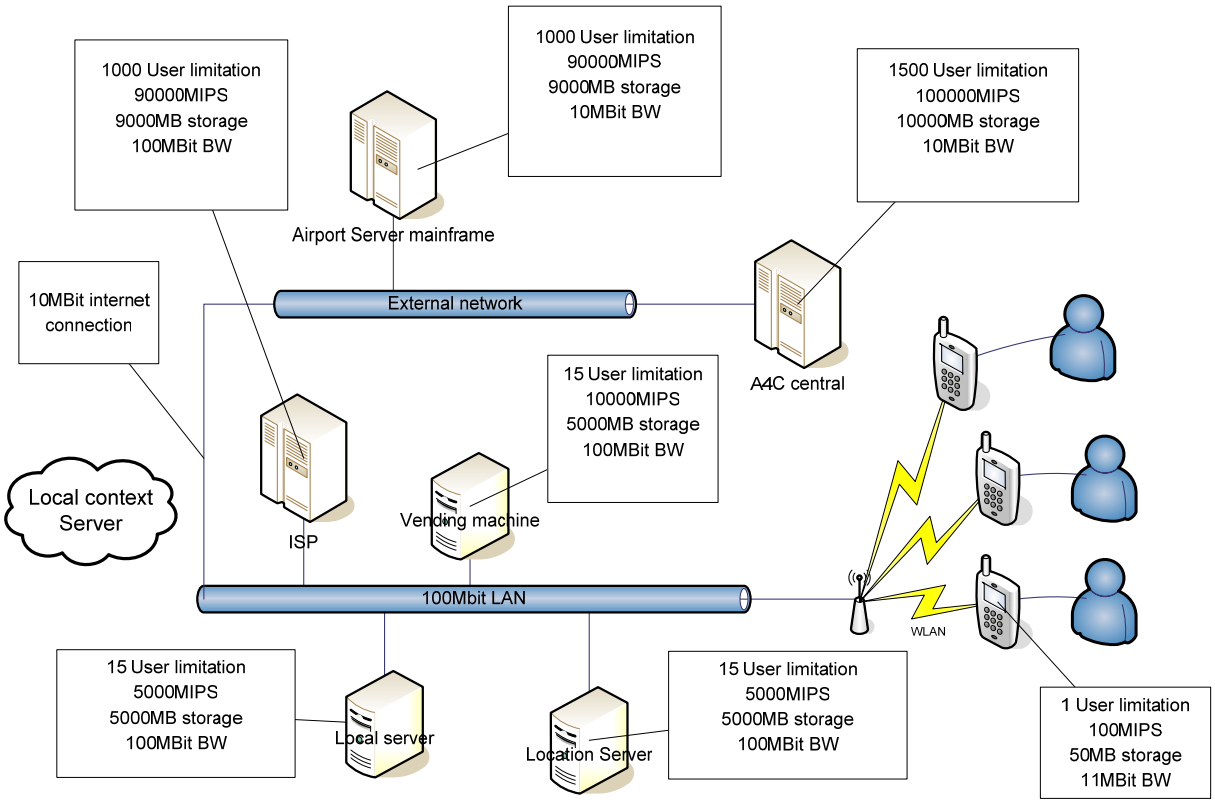


Figure 39 A Network Simulation example with device resources

This example has:

7 Devices:

- Cell phones, 1 per user
- Location Server
- Vending machine
- A4C Central
- Airport server mainframe
- ISP
- Local server

6 placed applets in all devices but the local server.

1 Unplaced applet waiting to be placed (defined as a device)

- Local context server

3 Networks:

- The internal LAN
- The internal WLAN
- The external Internet Connection

So, 7 devices or service enablers of which 6 have a dedicated applet, 1 unplaced applet and 3 networks.

Now it is wise to number the devices for future reference

Device 1: cell phone
Device 2: Location Server
Device 3: local server
Device 4: Vending machine
Device 5: Airport server mainframe
Device 6: A4C central
Device 7: ISP
Network 1: internal LAN
Network 2: internal WLAN
Network 3: external Internet Connection

The network topology (which device is connected to what network) needs to be defined

Device 1 (cell phone) is connected to network 1 (internal WLAN)
Device 2 (Location Server) is connected to network 2 (internal LAN)
Device 3: (local server) is connected to network 2 (internal LAN)
Device 4: (user server) is connected to network 2 (internal LAN)
Device 5: (Airport server mainframe) is connected to network 3 (external Internet)
Device 6: (A4C central) is connected to network 3 (external Internet)
Device 7: ISP is connected directly to network 2 (internal LAN) for less strain on network 3

Network 1(LAN) is connected to network 2 (WLAN)
Network 2 (LAN) is connected to network 3 (External network) and 1
Network 3 is connected to Network 2.

VI.1.1.1 Placing the information for the simulator to read

The two last lists enable us to route a packet from say device 1 to device 6. The packet would have to travel trough network 1, 2 and 3.

This has to be put into a topology-file (topology.txt) shown in Code 3:[6]:

```
Device# connected to network# Describes how each component are connected to which network.
1 1
2 2
3 2
4 2
5 3
6 3
7 2
Network# connected to network#
1 2
2 3 1
3 2
```

Code 3 the systems topology put into the topology.txt file

- Network topology, user timelines and the percentage of each user group goes into the topology.txt file
- Resource information and miscellaneous parameters go into the input.txt file
- The values for dynamic amount of users go in the userload.txt file

[6] For network to network connections it is not nessescary to say that network 2 is connected to network 3 and network 3 is connected to network 2. The router sets this up for you assuming all links are full duplex. But chosing to do so will not produce errors either.

VI.1.2 Define resources

Now we have the topology defined. Then we can move on defining the devices resources. In Figure 39, each Service enabler is displayed with its limited resources.

Device	Storage [Mb]	Processing power [MIPS]	#devices	#devices
1. Cell phone	50	100	1	1
2. Location Server	5.000	5.000	15	15
3. Local context server	5.000	5.000	15	15
4. User server	5.000	10.000	15	15
5. Airport server m.	9.000	90.000	1000	1000
6 A4C central	10.000	100.000	1500	1500

Table 14 Devices and their properties

Network#	Bandwidth [MBit/s]	Connected to network
1. Airport LAN	100	2,3
2 .Airport WLAN	54	1
3. External LAN	10	2

Table 15 Networks and their properties

The information from Table 14 and Table 15 are now put into the file input.txt like described in Code 4:

```
Devices
7
Networks
3
Users
100
Storage capacities in each device
1000 1000 1000 1000 1000
Processing capacities in each device
5000 5000 5000 5000 5000
Bandwidth capacities in each Network [MBit/s]
3 3 3 3 3
SIMTIME in hours
1 [hours]
Devices Available
1 1 1 1 1
```

Code 4 all the devices' resources put into the input.txt file

Now we have all the device and network properties defined. We can start defining the different users' timelines.

VI.1.3 Define services

A service is as mentioned earlier a collection of messages going to and forth. These services make up the meaning of the system. The services are illustrated as MSCs:

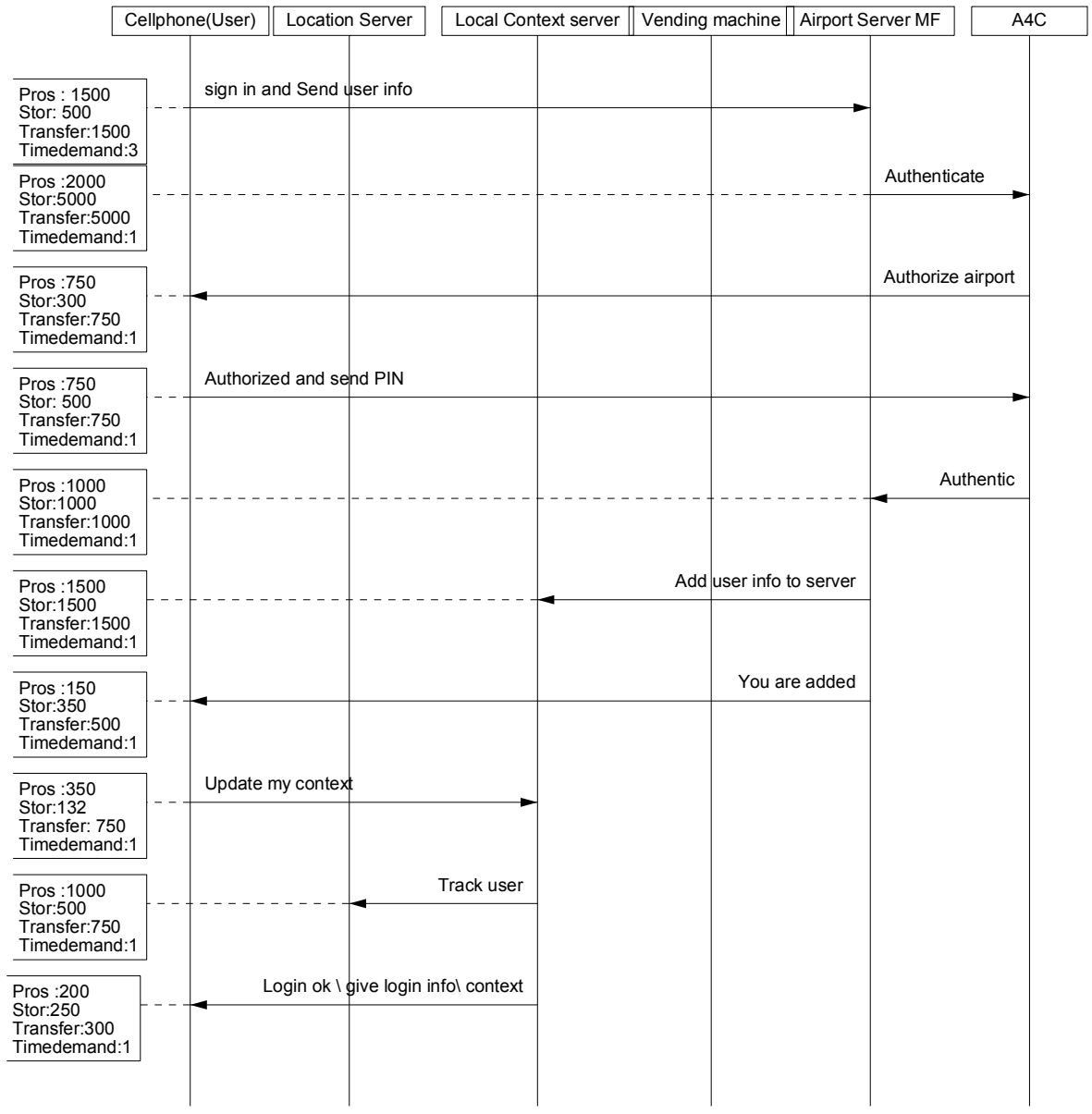


Figure 40 Check-in MSC

The MSC needs to be placed into a readable text file for our simulator to read. The check-in MSC will look like Code 5 when entered into the Service1.txt file. Each sequence has its separate line.

```
1 5 1500 0 1500 500 3 0
5 6 5000 0 2000 5000 1 0
6 1 750 0 750 300 1 0
1 6 7500 750 500 1 0
6 5 10000 1000 1000 1 0
5 3 1500 0 1500 1500 1 0
5 1 5000 150 350 1 0
1 3 750 0 350 132 1 0
3 2 750 0 1000 500 1 0
3 1 300 0 200 250 1 0
0 0 0 0 0 0 0 0
```

Code 5 The Check-in MSC in a simulator file

As you can see there are some extra parameters in the service file. The parameters are:

1. Sending device
2. Receiving device
3. #kB to transfer to recipient
4. number of required devices required at recipient
5. processing power required in recipient
6. Storage required in recipient
7. Timing demand. Used to set the time in which we need a reply. Not to be confused with Timeout, but used as an integer to find the Bandwidth need and processing need.
8. Propagation delay. This can be used if a device is far away or there for some other reason is delay due to external factors

A line of 0's indicates that the service is finished.

Messages are mapped from the MSC-X to the serviceX.txt file sequentially as shown in Figure 41:

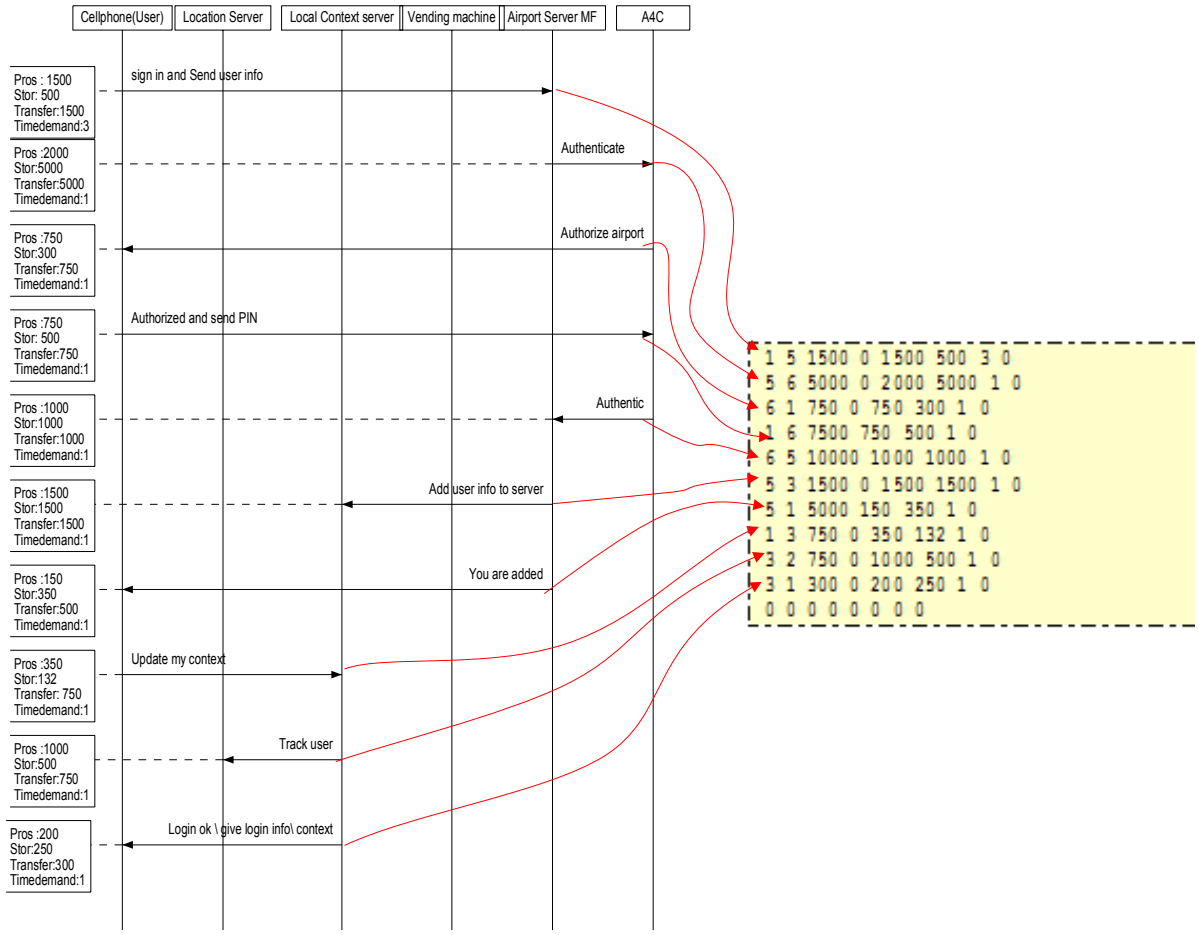


Figure 41 Sequential message mapping from MSC to service file

The MSCs of the remaining services can be found as appendices:

- MSC I Check-in MSC
- MSC II Check Reservation MSC
- MSC III Get directions MSC
- MSC IV Surfing internet MSC
- MSC V Conference MSC
- MSC VI Vending machine MSC
- MSC VII Upgrade reservation MSC
- MSC VIII Secure connection MSC

VI.1.4 Set up timelines

The timelines consist of series of services. Each of these services can be viewed as a separate MSC. You can see the MSC for the check-in in (Figure 40). The MSC also displays the individual resource needs required from the recipient. Take the "sign in and send user info" sequence at the top of Figure 40. This sequence needs 1500 MIPS, 500kB of storage and is 1500kB long when transferred over the network. The sequence also indicates a timing demand, which in this case is 3 seconds. The timing demand is an indicator of how fast we need the information transferred and is used in calculating the bandwidth need. The bandwidth need would in this case be:

$$\frac{1500\text{kB} * 8}{3s} = 4000kbit / s$$

Equation 3 Calculation of required bandwidth

VI.1.4.1 User types and timelines

The timelines describe what the users do with the system, what services they use and in what order. We have decided to generalize a bit and say that we have 5 different types of people. This is the user distribution we will use as the baseline scenario.

User type:	Percent of user mass
The business types, use a lot of power hungry services	15
Backpackers, Check in. wait for their plane and leave	10
Earlybirds, show up 2 hours early, need coffee, surf the web a bit	10
People with no sense of direction	25
Parents with children	40

The business types' timeline

These typically use resource hungry services like conferencing, Security apps using A4C, upgrading to a better reservation, checking their reservation and flight info, need directions check in and general surfing.

Timeline:

1. Check in
2. conference
3. surfing
4. upgrade reservation
5. use security app (A4C)
6. check reservation/flight

Backpackers' timeline

This group is at the airport to travel, not to enjoy the systems possibilities.

Timeline:

1. Check in
2. Check reservation/flight

Earlybirds' timeline

They show up early and have 2 hours to play around the airport

1. Check in
2. Check reservation/flight
3. Directions
4. Buy coffee
5. Surf the web
6. Surf the web
7. Surf the web
8. Check reservation/flight
9. Buy coffee

Directionless peoples' timeline

Where to go? Ask the system!

Timeline:

1. Check in
2. Directions
3. Check reservation/flight
4. Directions
5. Directions
6. Directions
7. Check reservation/flight
8. Directions
9. Directions

Parents with children's' timeline

These parents are of the up-to-date kind. They have been so smart as to buy locators for their 2 children. The children being children want candy.

Timeline:

1. Check in
2. Missing child, consult location context server, same service as Directions service
3. Check reservation/flight
4. Missing child, consult location context server
5. Buy candy (same as buy coffee)

Giving each service a number we can generate a timeline with just numbers to guide PreSim trough.

Service	Number
Check in	1
Check reservation	2
Surfing	3
Conference	4
Upgrade reservation	5
Use secure app. With A4C	6
Get directions	7
Buy coffee/soda	8

These timelines and user types go in the end of the topology.txt file with the percentage as the 10th integer. So it will look like this:

```
Device# connected to network# Describes how each component are connected to which network.
1 1
2 2
3 2
4 2
5 3
6 3
Network# connected to network#
1 2
2 3 1
3 2
UserGroups:
1 4 3 5 6 2 0 0 0 15
1 2 0 0 0 0 0 0 0 10
1 2 7 8 3 3 3 2 8 10
1 7 2 7 7 7 2 7 7 25
1 7 2 7 8 0 0 0 0 40
pointers:
1 userGroupNo
2-9 services used in the order written
A 0 indicates the end of the timeline
10 user percentage
```

Code 6 complete topology file with timelines

Now the topology.txt file is complete. The last 5 lines are optional. We have to define what each service actually does, and how much of each resource they need. To do this we make MSCs of each service. These are found in VI.4 .

VI.1.5 Remaining settings

Only a couple of settings remain. They go in the inpu.txt file.

```
Devices
7
Networks
3
Users
100
Storage capacities in each device
50 5000 5000 50 9000 10000 9000
Processing capacities in each device
1000 5000 5000 1000 90000 100000 90000
BandWidth capacities in each Network [Mbit/s]
11 100 10
SIMTIME in hours
1
Devices Available
1 15 15 15 1000 1500 1000
Seed (for randomization)
123465
RandomBattery Min/max
90 100
HoldTime random from .. to.. [mseconds] Time to wait (random) before tying to start new users if the userBin is empty
0 1000
WaitTime random between 0 and.. [mseconds] time to wait before restarting new resourcesearch when no resources have
been found. integer between
0 10000
PassiveTime random [mseconds] time between each new service is started, integer between
0 3600000
RunPass, Time to Start recording, Time to End Recording. Set to 0 to run trough all devices.
0 0 0
InterUser. The number of users per hour that arrive at the airport 3600 gives 1 user per second [users/hour]set 0 for
dynamic load (userload.txt)
10800
Display Successful tries
0
Trace
0
BatteryLoss, Set to 1 if batteries are to be discharged by 1% each hour
1
Debug
0
ErrorReport displays where in the timelines the errors have occurred
0
Report
0
Timeout[mSecs]
10000
Short run
1
Samples
500
applet (Applet to test)
3
```

Code 7 the complete input.txt file

We have already covered the first part of the input.txt concerning the devices parameters.

Now the simulator parameters remain.

Seed

This is used for randomization. One specific random integer will always be the same at one given time if the seed is not changed.

RandomBattery

Here you can input how well charged you want the visitors' batteries to be when they enter the airport. The battery value will still be random between the two integers you supply. The batteries will deplete themselves after time if the BatteryLoss option is enabled.

Holdtime

Time to wait before trying to start a new user, if all users are busy[7]. This is also a random integer between the two integers you supply.

WaitTime

The time to wait for the next try if a resource allocation has failed.
PassiveTime: the time a user waits after completing one service until it starts the next one. A random integer between the two values.

Passivetime

Tries to mimic the time in between the user ends one service and starts the next one. This is a random integer which will be drawn from the numeric difference between the integers given in milliseconds.

RunPass

This is a useful option if you just want to have a look at one specific pass. You can also indicate if you would like to have a closer look (zoom) at one specific period in time.

[7] if the userbin is empty

If the first integer is 0 PreSim will run all passes. If not, only the specified pass will be run. The interesting time domain can be specified with the two next integers (in milliseconds). With the two integers set to 0, PreSim will record the whole simulation period given by runShort. With values given to both time integers or just the last, all the specified sampled will be within this range of time.

InterUser

This is the value representing the amount of users logging onto the system per hour. If you enter 3600 you will get 1 user per second and so on. If you enter 0 here the simulator will use the input from userload.txt. This file has to be a representation of 24 hours of traffic. We have included statistical data from Oslo airport Gardermoen which the amount of users per hour over 24 hours.

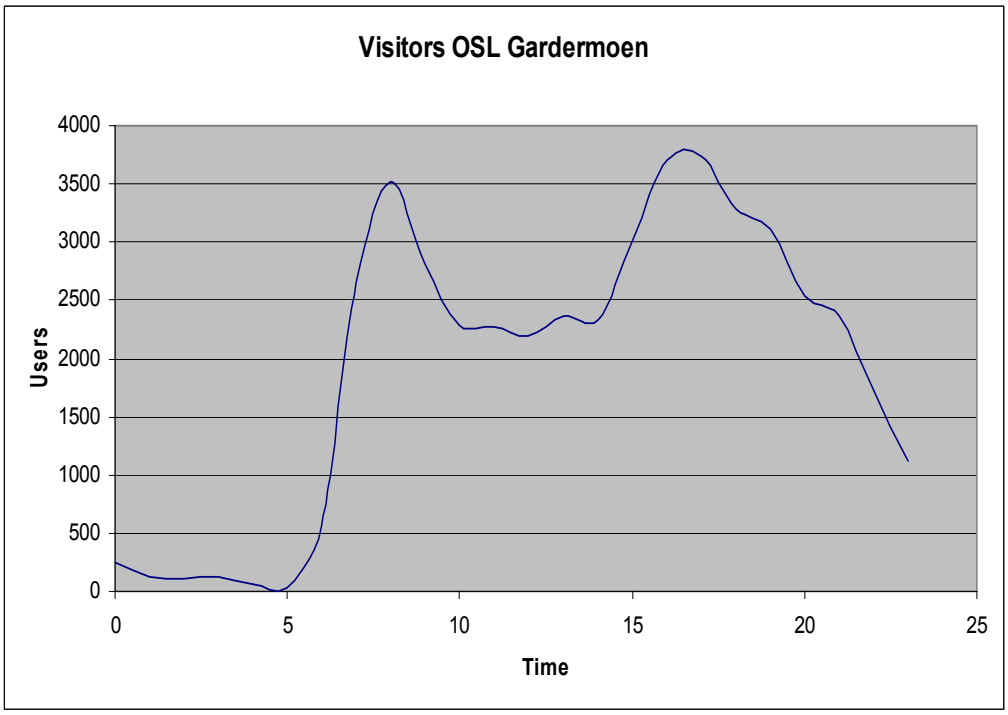


Figure 42 contents of Userload.txt as a graph[8]

[8] The file starts at time 00:00 and ends at 23:00. values are listed in Table 31 in the appendix

Display Successful tries

This is a nice option if you want to see the resources reserved. It will display all the successful reservations. With this option enabled you will get a considerable amount of output on screen. And the simulation will take almost forever to complete. Example output:

```
Sending From Device 1 To Device 5 From Network 1 To Device 3
Jumps   2 1 2 3 D: 0 P: 1500 S:    5 BW:      4
 D Taken:          0 From Device5:
 B Taken:          4 From NetWork1:
 B Taken:          4 From NetWork2:
 B Taken:          4 From NetWork3:
 S Taken:        500 From Device5:
 P Taken:       1500 From Device5:
```

Output 3 "Display successful tries" output

```
Sending From Device  1 To Device  5 From Network  1 To Device  3
```

Displays which devices and networks are involved as sender and recipient

```
Jumps   2 1 2 3 D: 0 P: 1500 S:    5 BW:      4
```

Displays which networks this sequence will travel trough. First digit is the number of inter-network jumps. The consecutive numbers is the networks the packets will have to travel trough. P: 1500 means 1500 MIPS of processing. S: 5 is 5kB storage BW: 4 is 4kB/s bandwidth.
The consecutive lines represent successful reservations of resources from the mentioned devices and networks.

Trace

With this enabled you get the trace command from Simula. This displays all the timing and resource holding output. Will give even more output than Display Successful tries. Not recommended if you want the simulation to end at all.
Values: 1 enabled 0 disabled.

BatteryLoss

Set to 1 if batteries are to be discharged by 1% each hour. Values: 1 enabled 0 disabled.

Debug

Used for displaying various info about the simulation. Values: 1 enabled, 0 disabled.

Displays the following:

Routing table:

```
fastest route between:  1 and  1 is: Jumps   1 1 0 0
fastest route between:  1 and  2 is: Jumps   1 1 2 0
fastest route between:  1 and  3 is: Jumps   2 1 2 3
fastest route between:  2 and  1 is: Jumps   1 2 1 0
fastest route between:  2 and  2 is: Jumps   1 2 0 0
fastest route between:  2 and  3 is: Jumps   1 2 3 0
fastest route between:  3 and  1 is: Jumps   2 3 2 1
fastest route between:  3 and  2 is: Jumps   1 3 2 0
fastest route between:  3 and  3 is: Jumps   1 3 0 0
```

Output 4 Routing table output

All the services

```
service 1
    1    5 1500    0 1500  500    3    0
    5    6 5000    0 2000 5000    1    0
    6    1  750    0  750  300    1    0
    1    6 7500    0  750  500    1    0
    6    510000    0 1000 1000    1    0
    5    3 1500    0 1500 1500    1    0
    5    1 5000    0  150  350    1    0
    1    3  750    0  350  132    1    0
    3    2  750    0 1000  500    1    0
    3    1  300    0  200  250    1    0
    0    0    0    0    0    0    0    0
```

Output 5 Confirmed service output

And each sequence as it is read from the service

```
info read: usr 1
UserGroup:             1
Servicecode:           1
UserTimelinePlace:     1
FromDev:               1
ToDev:                 5
size:               1500
devs:                  0
Pros:               1500
stor:                500
Timedemand:            3
Propdelay:             0
TIME:           25200000
line:                  2
```

Output 6 Debugging info

ErrorReport

This displays where in the timelines the errors have occurred (values: 1 enabled, 0 disabled). It appears at the end of each run of the simulation:

```
Storage problem, failures:                  124
|
|-->Device6 Storage problem, failures:          24
|-->Device5 Storage problem, failures:         100
```

Output 7 Error analysis

Report

If this is enabled, you will get the standard DEMOS report (values: 1 enabled, 0 disabled). If not you will get this report:

```
************************************************************************
*                                                                      *
*                           Results                                    *
*                                                                      *
************************************************************************

With the applet in device             7
Time:                          40326195 mSecs
User pool                         10000 Users
failures                              0
users started:                    20918 Users
Completed Services:               80838
SuccessRate:                         98 %
Tries:                           210294
Retires:                           3469
Timeouts:                          2355
Empty_Userbins:                   22035

************************************************************************
*                                                                      *
*               END            Results            END                *
*                                                                      *
************************************************************************
```

Output 8 Result from one applet position without error analysis

With Report set to 0 and ErrorReport set to 1 you will get a very descriptive end report:

```
**************************************************************************
*                                                                        *
*                            Results                                     *
*                                                                        *
**************************************************************************

With the applet in device             7
Time:                          40326195 mSecs
User pool                         10000 Users
failures                              0
users started:                    20918 Users
Completed Services:               80838
SuccessRate:                         98 %
Tries:                           210294
Retires:                           3469
Timeouts:                          2355
Empty_Userbins:                   22035




Processing problem, failures:                    3469
|
|-->Device3 Processing problem, failures:         3445
|-->Device2 Processing problem, failures:           24
**************************************************************************
*                                                                        *
*               END            Results            END                  *
*                                                                        *
**************************************************************************
```

Output 9 Result from one applet position with error analysis

Timeout

Here you can set the timeout variable. This is the time the system will try to get the required resources. If they are not acquired within the timeout period, the timeline and user will be aborted. Values given in milliseconds.

Short Run

With this option enabled you will only record within the simtime window. With it disabled you will record the simtime window and until all users are trough the system. This is approximately simtime*6.

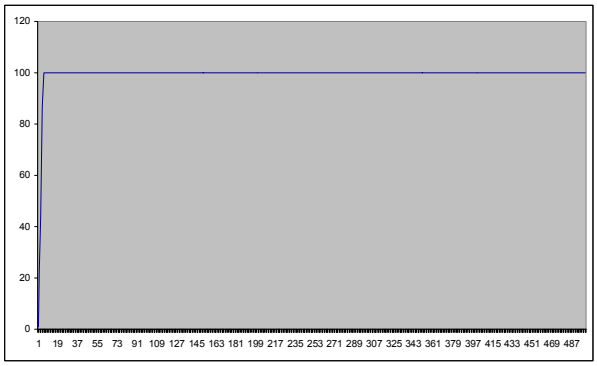


Figure 43 short run enabled

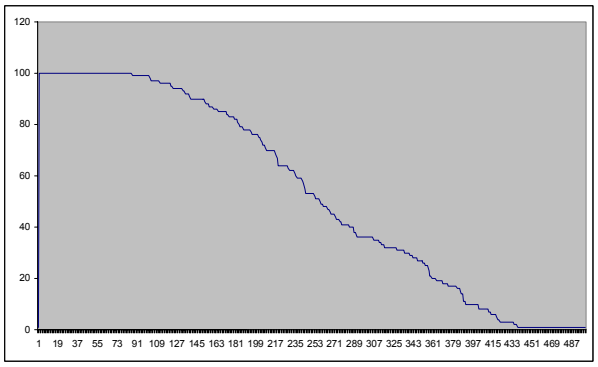


Figure 44 ShortRun disabled

Samples

Samples define how many samples per pass you want in the output.xml file. 100 samples will produce an output file of approximately 0,1MB per device in the system.

Applet

This is the applet we are about to test and find the best locations for. In our simulations we will test the Local context server applet. The integer written represents the number of the applet (fromdevice or todevice) in the service files. Our service files have been set up with the local context applet as number 10.

Now install cim and DEMOS [Link V]. Set the path to the demos.atr file at the top of PreSim.sim and compile[9].

With all these settings set the simulator should be ready to run.

[9] The path c:\cim\demos.atr is preset

VI.2 Appendix B: PreSim external view

In this chapter we will look at the simulator from an external point of view. We will not dig deep into the inner working as this is done in "Appendix C: PreSim Internal View".
The whole objective of the simulator is to test the impact of the location of the test applet on a network. This is done by attempting to place the applet in the available devices one after the other and logging the performance of the network and devices. This on the other hand, means that the runtime for a simulation will be the simulation time multiplied by the number of devices. Actually the runtime will be longer than simtime*numDevices due to the fall off time at the end of the simulation time (see Figure 14). The fall off time is due to the fact that all the users have to go trough their timelines, timeout or fail before a new simulation can begin. The fall off time can be 6 times or more of the total simulation time. Giving a runtime for one simulation pass of at least 7*Simulation time. With the short run option enabled we will only record the resource usage in the simtime period, but the overall time including falloff time will be written at the run time results page in the output.xml spreadsheet ant to screen.

VI.2.1 Start-up

Before running the simulation, the simulator will first read in the different services, timelines and all the parameters. The services and timelines will be put into separate tables, and all resources (bandwidth, storage, users and so on) will be stored as SIMULA resources (RES). Before any sending and receiving can start, the routing table must be set up. For this we use a mechanism similar to Dijkstra's shortest path algorithm. (Link I and Chapter II.6.1.2). This creates a routing table that shows the shortest routes from Device A to B. This routing table can be displayed if you enable the debug function. It will then pop up as the first output to screen (Output 4). Set simtime to 0 if you just want this table to be displayed. There will be an error message, but this does not matter since you are only looking for the routing table.

Before it starts sending people into the system it starts the "fileout class" [VI.5.11] and the "interuserclass" [VI.5.12]. The fileout class is responsible for printing out runtime results to a

spreadsheet compatible XML-file (output.xml). The InteruserClass entity is what takes care of the correct value of users per hour into the simulator. If InteruserTime is 0 then it will read from the userload.txt file to mimic the amount of users per hour given.

VI.2.2 Enter the test applet

At the first pass the simulator will place the testApplet in device number 1. The placement of the testApplet will be incremented by 1 by each consecutive pass.

VI.2.3 Users enter the equation

Now the simulator will move on, starting to send users into the network. This is done in “The user starter" (APPENDIX E: Code Snippets, VI.5.1). This is the part of the simulator which feeds the user-hungry simulator.

At zero time, user number 1 is started as a personal thread or entity (in SIMULA). It is assigned a user Group which is decided by the given percentage at the end of each timeline in the topology.txt file. This User Group indicates if this user is (in our case) a Businessman, backpacker, earlybird, directionless or a parent. And will, based on the user group number, decide which timeline to take in the simulation. This user is also assigned a random cell phone with a random remaining battery capacity, storage, bandwidth and processing capacity as well as a number indicating the amount of cell phones or devices this user has.
This user is now on his way in his timeline. We now end up in the users “execution manager” (APPENDIX E: Code Snippets, Code 9). The execution manager is what runs the user trough his timeline. The next services and messages to be executed are found here.
In the execution manager the information already stored from the service files is retrieved for use. We run trough all the messages in the service until its end. At the end of a service, ServiceCode is incremented by 1 and the procedure looks if there is a next service in the user’s timeline and runs it if that is the case.
If there are no more services the user is terminated or considered complete.

VI.2.4 Looking for resources

As user 1 goes trough the execution manager for the first time he will try to go trough his first service. The requirements are read in the execution manager. This procedure attempts to, with the reserveResources procedure, to reserve the required resources. If these resources cannot, for some reason, be reserved it will try on until the specified timeout is reached. Then the attempt is considered a failure, the timeline is aborted and the user is terminated.

When attempting to reserve the resources the reserveResources procedure is called. This is a procedure in which the simulator attempts to grab the required resources from the respective owners. APPENDIX E: Code Snippets, Code 11 shows a small part of the procedure. The procedure first looks for the recipient of the sequence (toDevice). When found it checks if there are enough available resources for the required reservation. If there are they will be reserved, and the procedure continues to reserve the rest of the required resources until all are reserved. If the simulator does not find the required resources it will try again until timeout. If the required resource is greater than the total amount of this resource in the device the whole pass is terminated. This is done because the timelines will be impossible to complete without enough resources.

VI.2.5 Resources reserved, transfer

When we get all the required resources for the sequence, it is time to transfer the information from sender to recipient. This is simulated by holding the resources for some time. The transfer procedure (APPENDIX E: Code Snippets, Code 12) is called. The procedure holds the resources, and completes.

Each resource is held for a specific time period found by this equation:

$$\frac{Message_{size}[kbytes]*8}{BWNeed[kBit/s]}+\frac{P_{Res}}{P_{Res}+P_{avail}}+t_{propagation}=holdingTime[s]$$

Equation 4 Resource holding time

In Equation 4 $Message_{size}$ is the size specified for each message being sent from A to B in Kilobytes, BWNeed is the required and reserved bandwidth for this service instance in kBit/s. P_{Res} is the reserved Processing capacity, P_{avail} is the remaining available processing power in the device. $t_{Propagation}$ is the propagation delay.

VI.2.6 Resources release

The reserved resources have to be released for others to use. The resources are released when

- A resource reservation has failed
- Transfer is complete
- After a timeout
- After a complete failure

This is done with the releaseResources procedure (APPENDIX E: Code Snippets, Code 13).

VI.2.7 Simulation roundup

After a set of passes with the testApplet placed in all the devices, PreSim begins to round up the information. Throughout the simulation runtime information has been logged to the output.tmp file. The results of each pass are now placed into this file as well and it is made readable as the ouput.xml file. The recorded information from each pass is:

- Time (Time)
- Users in the system (Users in system in Service X)
- Devices available from device 1 to the number of devices
- Storage available from device 1 to the number of devices
- Processing available from device 1 to the number of devices
- Bandwidth available from networks 1 to the number of networks

The information from each pass will be displayed as a separate sheet (in Microsoft Excel) named AppletLocationX. The running results; Passtime, Complete failures, Tries, Retries and Timeouts will be displayed in the Running_results sheet.
This file will contain a pre-specified amount of samples from each pass.
In addition to this you will be presented, on screen, with the results from each pass. If the Error report option is enabled you will be presented with the location (in which device) the errors have occurred and if there are any complete failures they will be presented on screen as well.

VI.3 Appendix C: PreSim Internal View

PreSim was written as one large code-file due to the large number og shared variables used by nearly all running classes, entities and procedures.

The simulator consists of some major functional blocks which are:

- The Service file reader APPENDIX E: Code Snippets VI.5.7
- The route sniffer APPENDIX E: Code Snippets VI.5.9 explained in II.6.1.2
- User Group Selector APPENDIX E: Code Snippets VI.5.10
- XML output file entity class APPENDIX B: VI.5.11
- The user starter APPENDIX E: Code Snippets VI.5.1
- Execution Manager APPENDIX E: Code Snippets VI.5.2
- tryToRoundUpResources procedure APPENDIX E: Code Snippets VI.5.3
- ReserveResources procedure APPENDIX E: Code Snippets VI.5.4
- Transfer procedure APPENDIX E: Code Snippets VI.5.5
- Release Resources procedure APPENDIX E: Code Snippets VI.5.6

VI.3.1 The Service file reader

This is a text file parser that reads in the services from the ServiceX.txt files. These services are stored in internal lists. This is done to prevent multiple position pointers in a single file that would have caused failures.

VI.3.2 The route sniffer

This is the class which finds the routes from and to each device. See Code 16 for the code. It sets up a routing table based on the given network topology using a lightweight version of Dijkstra's' algorithm [Chapter II.6.1.2].

VI.3.3 User Group Selector

This class [APPENDIX E: Code Snippets, Code 17] makes sure the specified amount of each user is created.

VI.3.4 XML output file entity class

This entity class samples the resources with a predefined sampling rate. Before starting the sampling it writes the header of the XML-file[10]. A new sheet is started for the first run with the name applet location 1.

The sampling rate is found by Equation 5:

$$\frac{t_{stop} - t_{start}}{\#samples} = samplingtime$$

Equation 5 sampling time equation

t_{Stop} is the specified time to stop sampling. If no time is specified, the parameter is decided by the shortRun parameter if it is supposed to record just the simtime of the simtime including falloff time.

t_{Start} is the specified time to start sampling.

#samples are the number of wanted samples in each pass indicated in the input.txt file.

For each finished pass[11] the xmlwriter writes the footer of the sheet and a new header is created. The changing applet locations are found by comparing the locally stored applet location with the global integer appletLocation. If it changes, then a new sheet is required. At the end of each run[12] the sheet footer and spreadsheet footer is written. SIMULA needs a specified number of byte-spaces specified each time an integer is written to screen of to a file. Unfortunately XML does not like to read spaces if the following string is specified as a number. And in SIMULA it is not possible to write the character (“) which is required by

[10] Actually the xml file is written to a temporary output.tmp file first. To be explained.

[11] Completed simulation of one servicelocation.

[12] All service locations (passes) have been simulated.

XML to function with Excel. Both these problems are solved with a byte parser run at the end of each run. This parser[13] reads the whole output.tmp file and writes it to the ouput.xml file. Fortunately the byte reader in SIMULA can output any characters except the character (‘). So we read in the output.tmp file looking for the character (#) which we have used to replace the characters (“) when creating output.tmp. we replace them with (“). We also look for spaces that are not supposed to be there and remove them. Before the parsing and after can be seen below:

Before: <Cell><Data ss:Type=#Number#> 10</Data></Cell>

After: <Cell><Data ss:Type="Number">10</Data></Cell>

VI.3.5 The user starter

The user starter is located in the main class of the PreSim application. This procedure is repeated once for every location the test applet will be placed in. Before starting any users the procedure will see if the user bin is empty. If it is the user starter will hold for some time. If it is not, the user starter will start a new user.
For each user that is started the NEW(“usr”)… command is executed.

```
NEW
usr("usr",u,1,TIME,Device1SNumber,ProsDevice1Number,Device1BW,Device1DResNumber,RandBattery.sample,userType).schedule(
NOW);
```

This starts a new thread of the entity user. This user will then start its timeline and start the execution manager.
The User Starter will then hold for some time before trying to start a new user.

[13] The parcer is the XMLClean procedure in PreSim.sim

VI.3.6 Execution Manager

The Execution Manager procedure is responsible for reading in the requirements for the relevant message to send. This message is found by using the users' user group and its place in the timeline. Once read it will try to find the resources by executing the tryToRoundUpResources procedure.

VI.3.7 tryToRoundUpResources procedure

This procedure tries to fetch the required resources found by the Execution Manager. It does this by executing the ReserveResources procedure. While trying, it will make sure the time tried to fetch the resources does not exceed the timeout. If a positive "commit" is returned from the ReserveResources procedure it can move on to transfer the message and last release the resources. If "commit" is not returned true before the timeout, the attempt is considered a failure and the users' timeline is terminated.

VI.3.8 ReserveResources procedure

This procedure looks for the available resources of bandwidth, processing, storage and battery. If they are available they will be reserved. If not, "commit" will be set to false and the "tryToRoundUpResources" procedure has to keep trying. If the requested resources are higher than the total resource in one of the devices, "cannotCommitAtAll" is set to true and the whole simulation pass is ended. This is because the services requesting the resource will never get the request granted and can thereby not continue.

VI.3.9 Transfer procedure

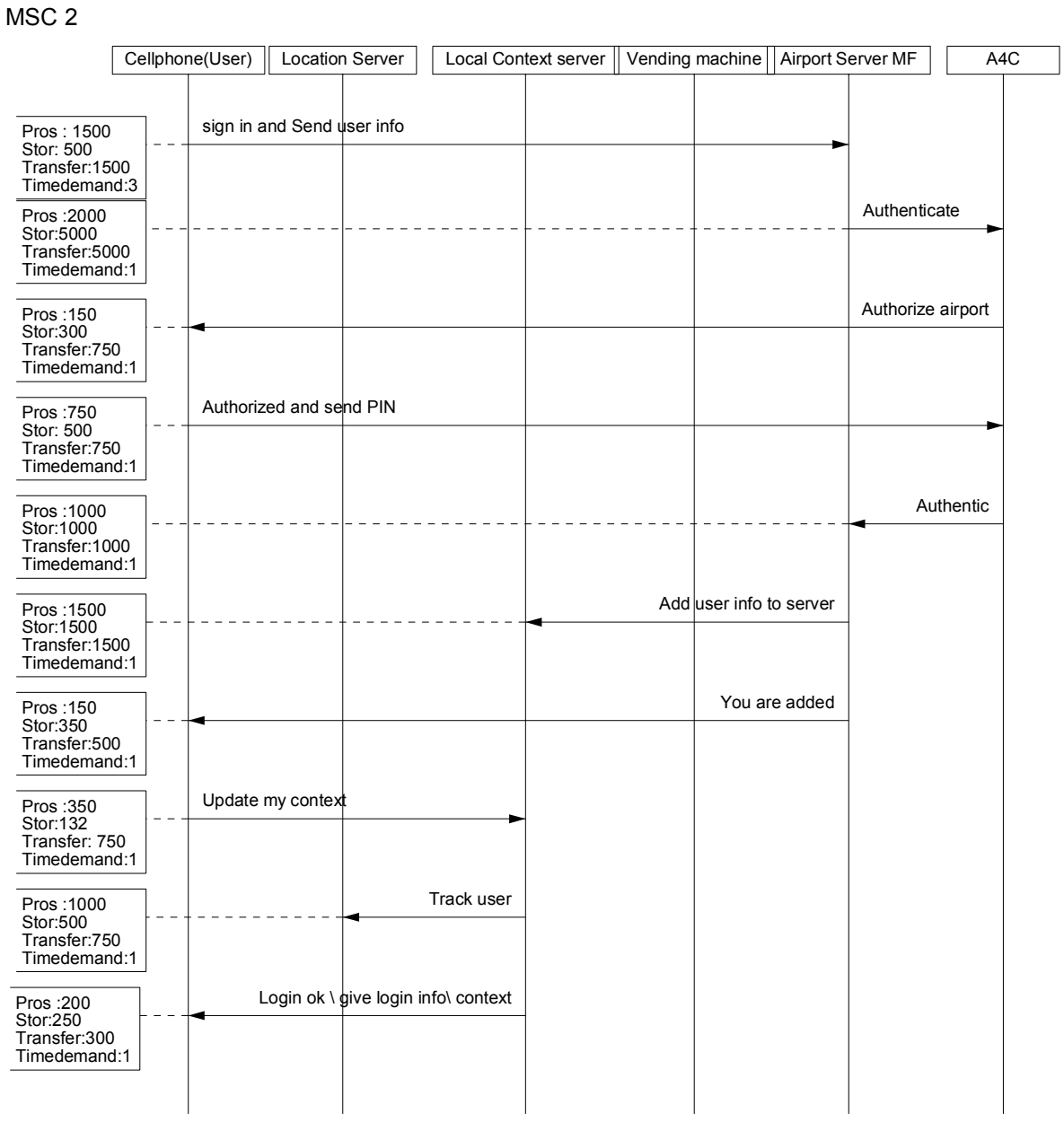
This procedure finds the time the resources will be held and holds them this long. Holding time is found by Equation 4.

VI.3.10 Release Resources procedure

This procedure releases the reserved resources and resets the Boolean parameters used in the ReserveResources procedure.

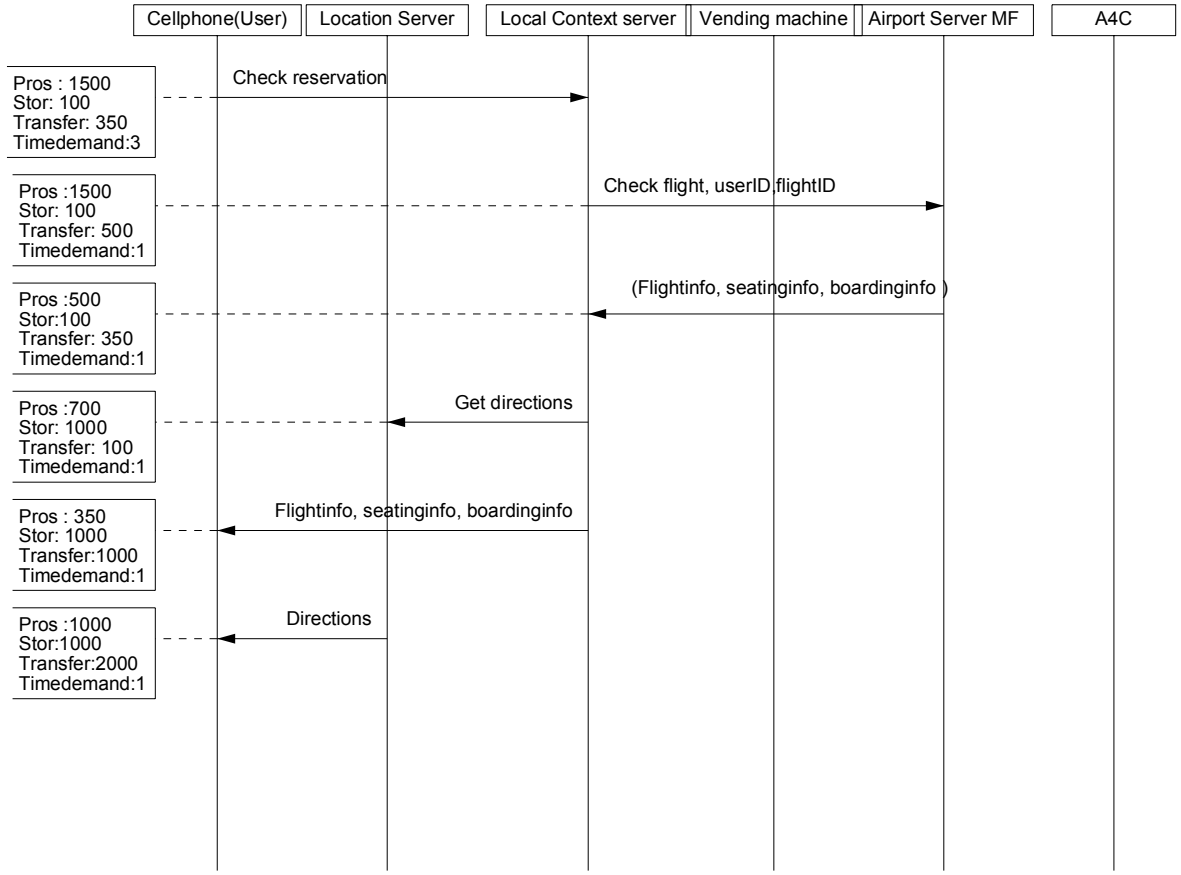
VI.4 APPENDIX D: MSC-charts

VI.4.1 Check In MSC



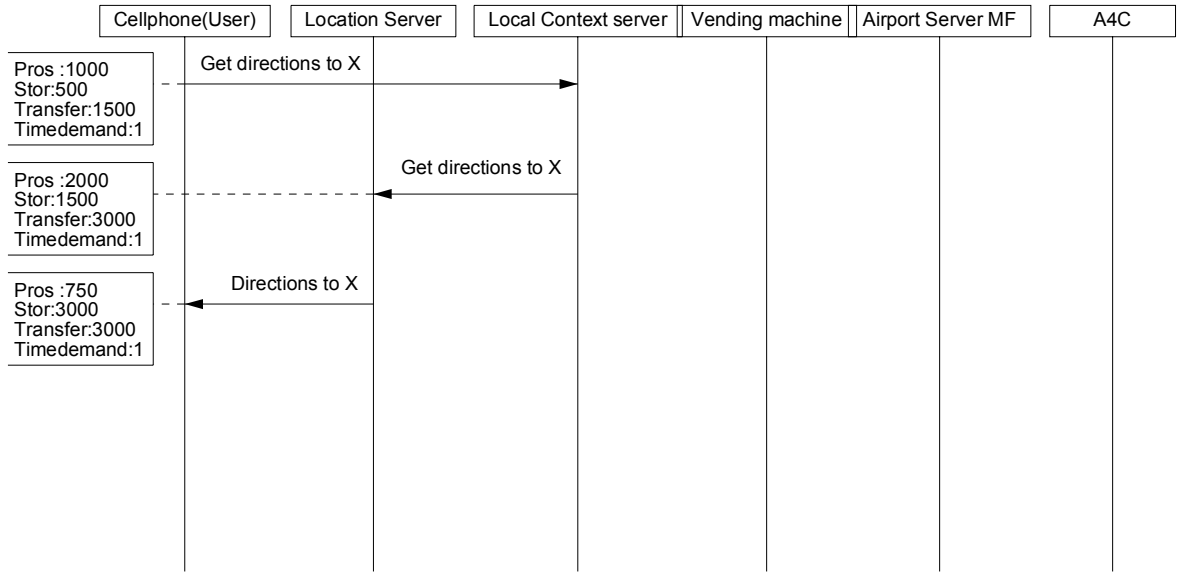
MSC I Check-in MSC

VI.4.2 Check Reservation MSC



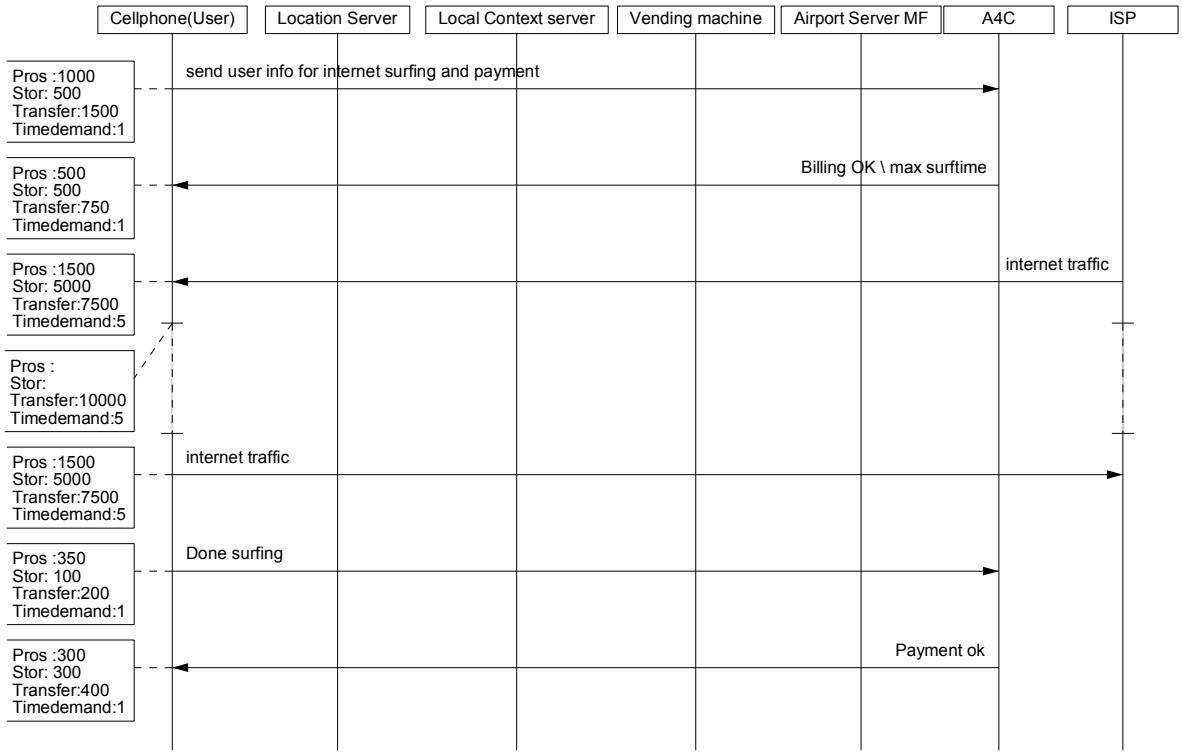
MSC II Check Reservation MSC

VI.4.3 Get Directions MSC



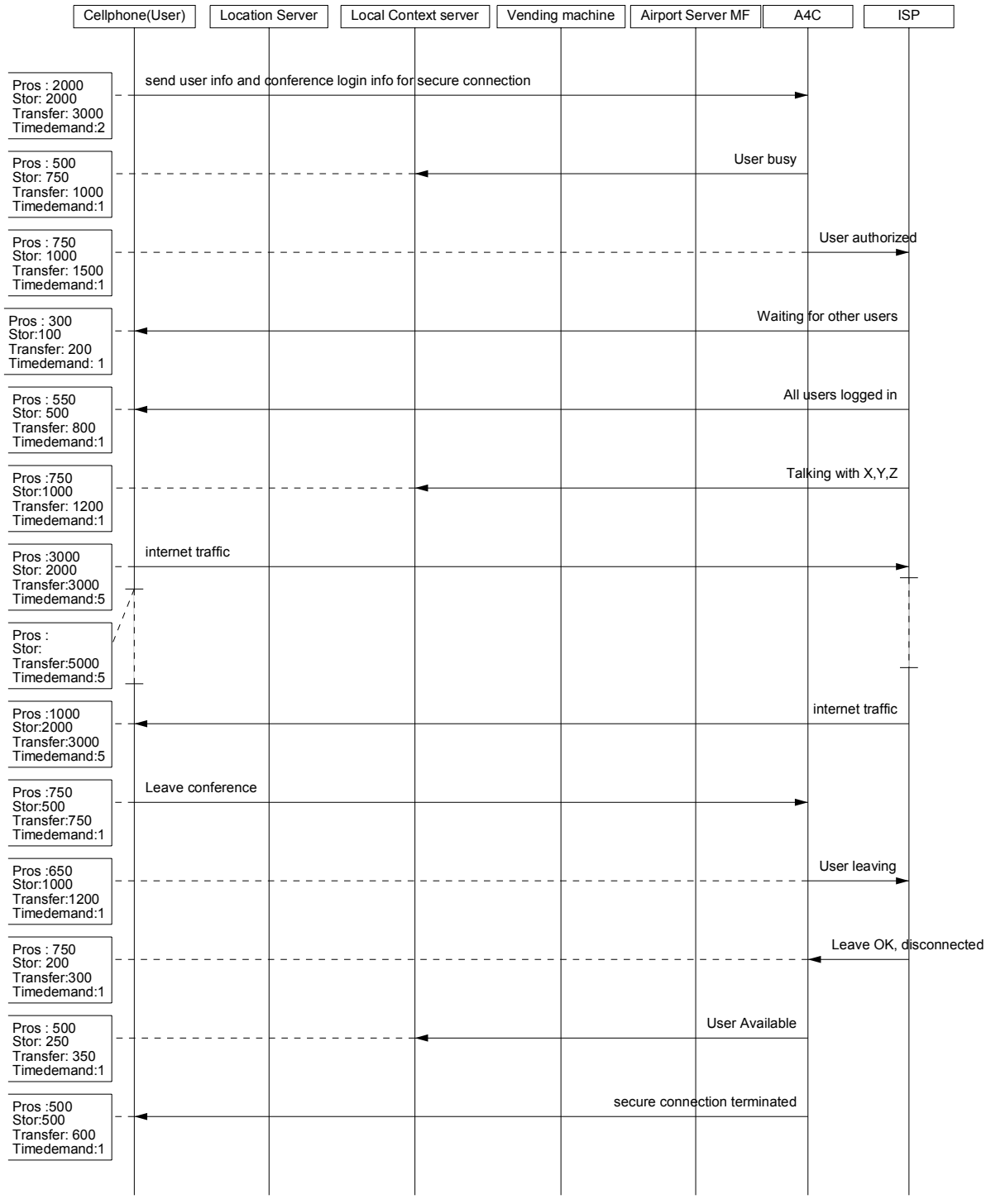
MSC III Get directions MSC

VI.4.4 Surfing Internet MSC



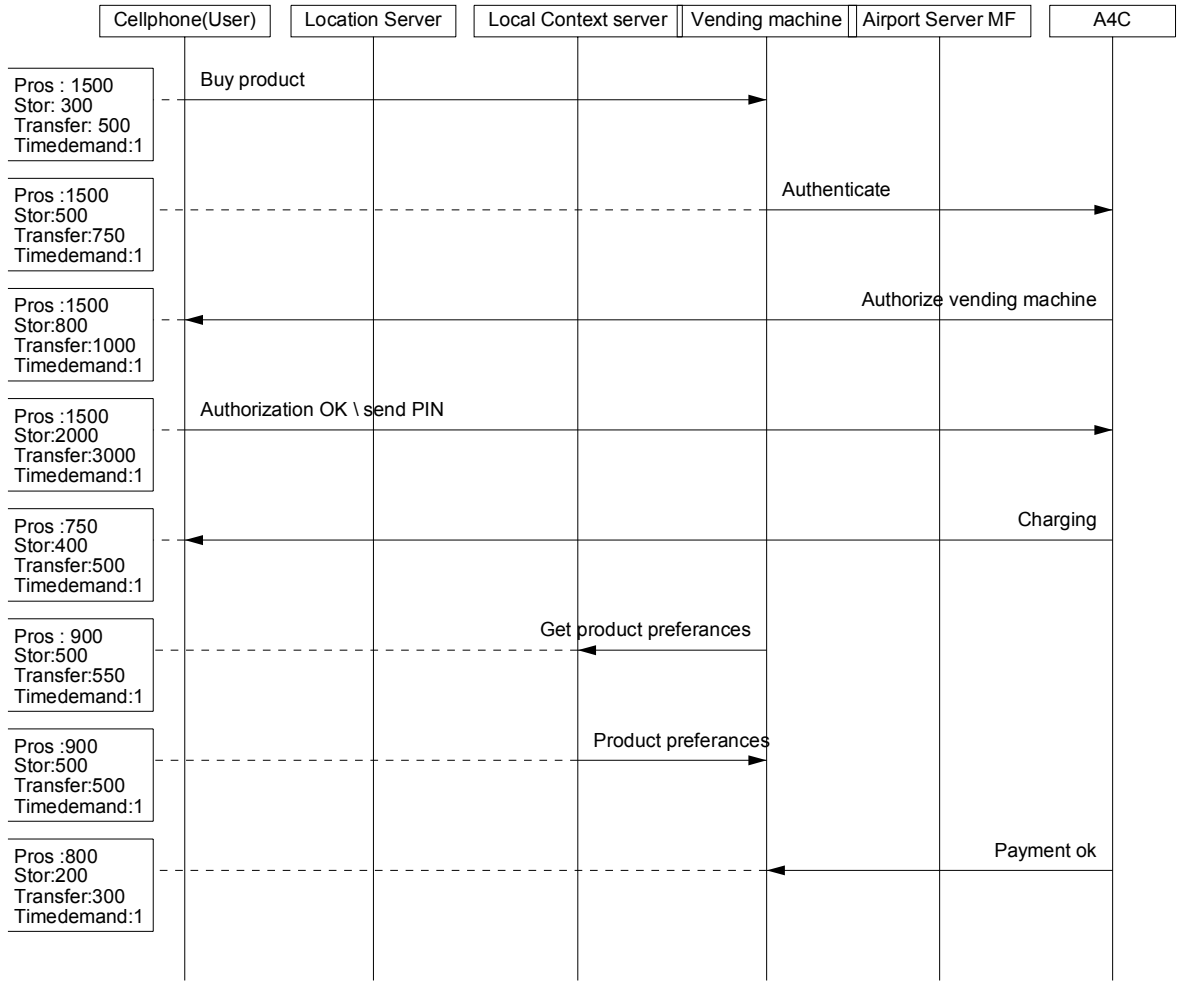
MSC IV Surfing internet MSC

VI.4.5 Conference MSC



MSC V Conference MSC

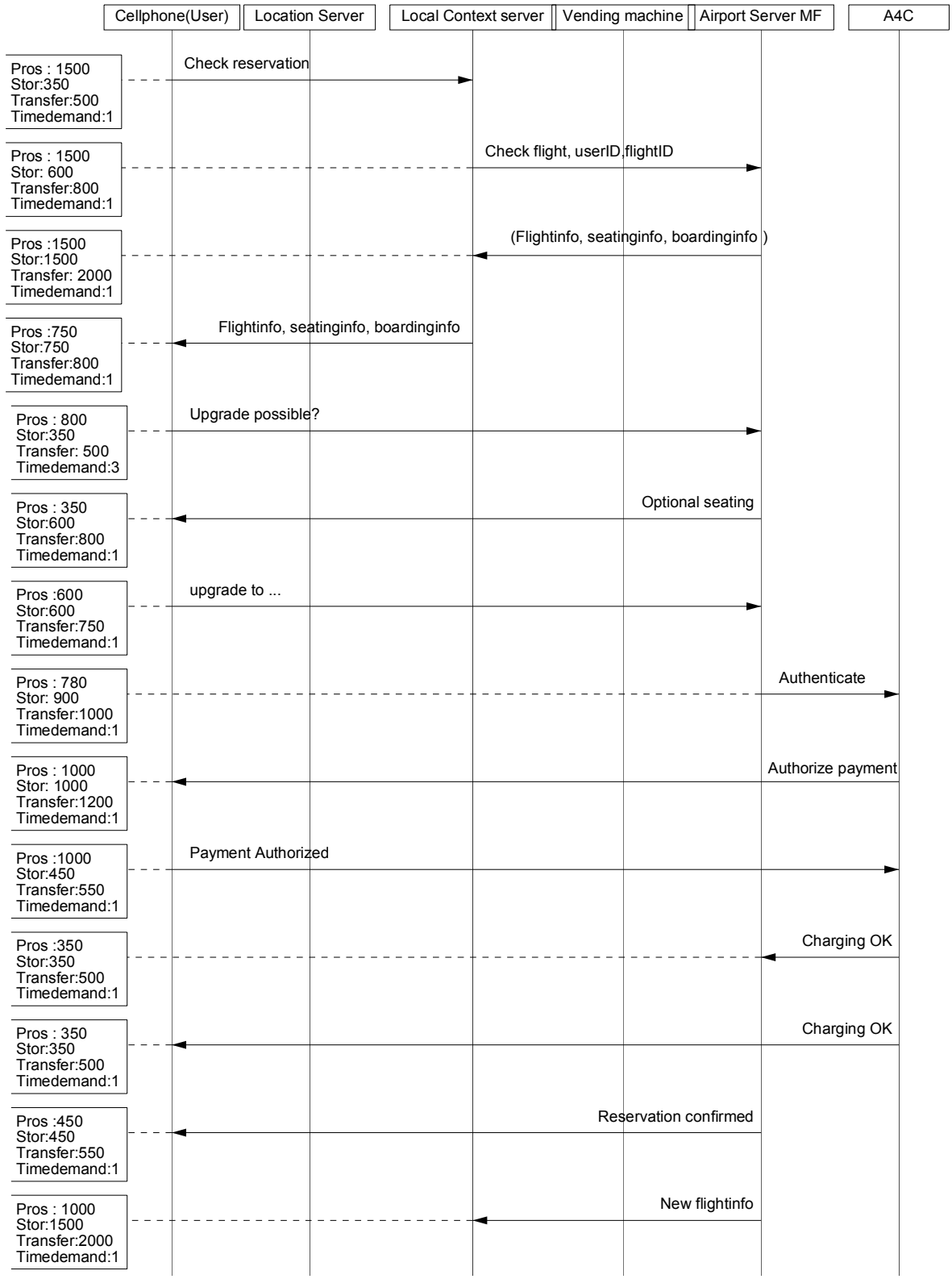
VI.4.6 Vending Machine MSC



MSC VI Vending machine MSC

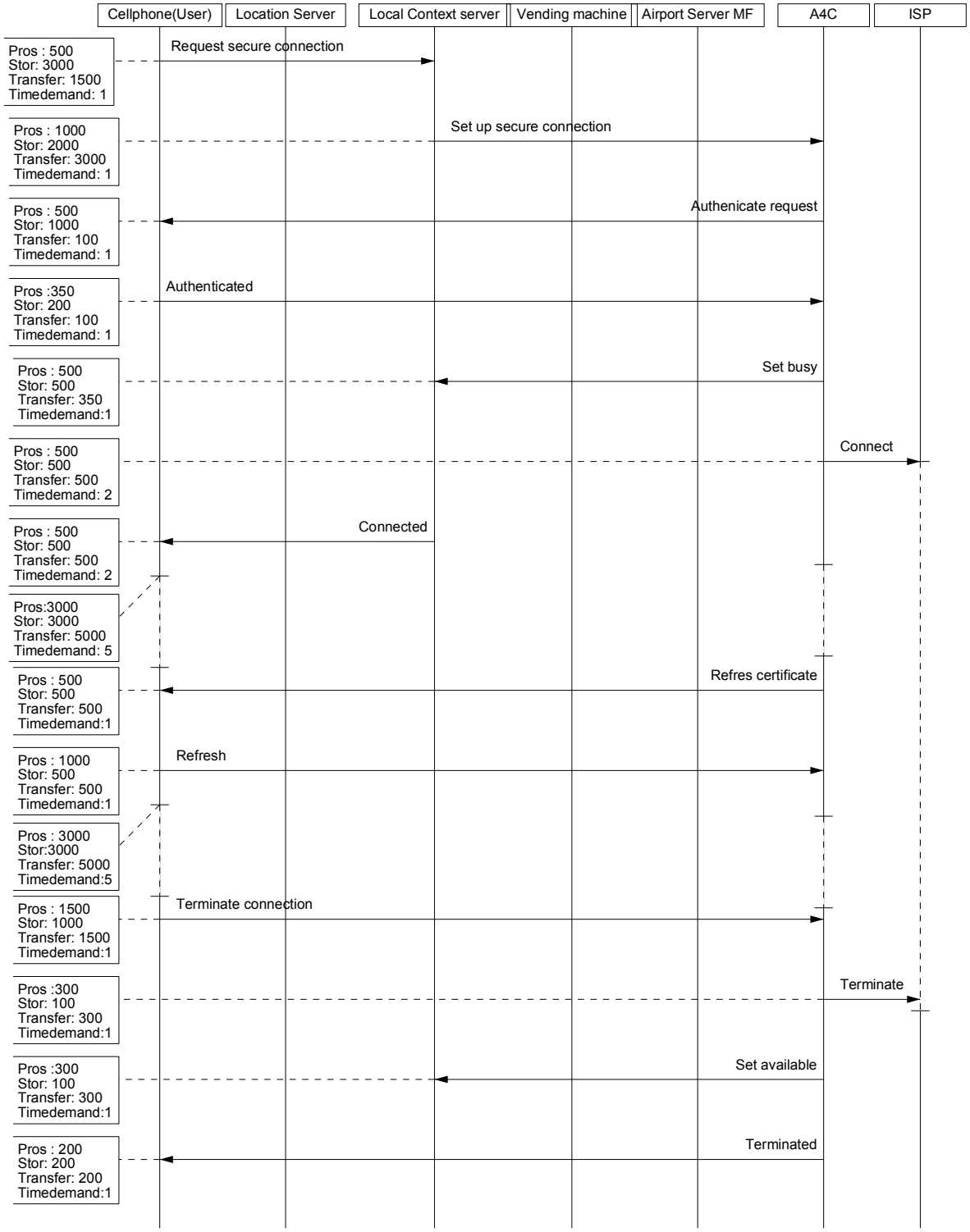
VI.4.7 Upgrade Reservation MSC

MSC upgrade



MSC VII Upgrade reservation MSC

VI.4.8 Secure Connection MSC



MSC VIII Secure connection MSC

VI.5 APPENDIX E: Code Snippets

VI.5.1 The user starter

```
for AppletLocation := startLocation step 1 until endLocation do

    begin !AppletLocationSelector;
        resetcounters;
        u:=0;
        keepgoing := true;
        RunTime := TIME;
        while (time-RunTime) < SIMTIME  and keepgoing do
        begin
            if (users.initial-users.avail) < maxusers then
            begin
                u:=u+1;
                users.acquire(1);
                number := (RandUserGroup.sample);
                FindUserGroup;

                NEW
usr("usr",u,1,TIME,Device1SNumber,ProsDevice1Number,Device1BW,Device1DResNumber,RandBattery.sample,userType).schedule(
NOW);

                hold(InterUser);
                StartedUsers.update(1);
            end
            else
            begin
                hold(holdtime.sample);
                if users.initial = maxusers then EmptyUserBin.update(1);
            end;
        end;
        if runshort then reset := true;

        users.acquire(totalusers);
        users.release(totalusers);

        statsReport;
        outimage;
        outimage;
        startreporting;
        if not runshort then reset := true;
        while not xmlwriterreset and not valuesreset do hold(100);
        reset := false;
        xmlwriterreset := false;
        valuesreset := false;
    end AppletLocationSelector;
    appletlocation := appletlocation + 1;
    XMLNewSheetEnd;
    xmlEndWriter;
    XMLClean;
    outputfile.close;
```

Code 8 The user starter

VI.5.2 Execution Manager

```
           Device1StorRes := Device1StorNumber;
           Device1ProsRes := ProsDevice1Number;
           Netw1BW := Device1BWNumber;
           cannotCommitAtAll := false;
           linenumber:=1;
           while UserTimelinePosition(UserGrp,serviceCode) ne 0 and UserTimelinePosition(UserGrp,serviceCode) < 10 and
not cannotCommitAtAll and (TIME - startTime) < timeoutTime do
           begin
               commit:=false;
               starttime := time;
               if UserTimelinePosition(UserGrp,serviceCode) = 1 then
               begin
                   while Service1Table(1,linenumber) ne 0 do
                   begin
                       fromDevice  := Service1Table(1,linenumber);
                       toDevice  := Service1Table(2,linenumber);
                       size := Service1Table(3,linenumber);
                       devs := Service1Table(4,linenumber);
                       pros := Service1Table(5,linenumber);
                       stor := Service1Table(6,linenumber);
                       timeDemand := Service1Table(7,linenumber);
                       propDelay := Service1Table(8,linenumber);
                       linenumber := linenumber + 1;
                       if fromDevice = service then fromdevice := appletlocation;
                       if toDevice = service then toDevice := appletlocation;
                       tryToRoudUpResources;
                   end servicereader;
                   serviceCode := serviceCode+1;
               end ifUG1;
.
.
.
.
           end readingTimeline;
       users.release(1);
```

Code 9 Execution Manager

Please note: this is only a part of the procedure. The dots indicate missing code. See the source file for the entire code.

VI.5.3 tryToRoundUpResources procedure

```
procedure tryToRoudUpResources;
begin
    commit:= false;
    cannotCommitAtAll := false;
    if toDevice ne 0 then
    begin
        while not commit and not cannotCommitAtAll and (TIME - startTime) < timeoutTime do
        begin
            findWayAndInfo;
            reserveResources; !Get resources, if not available, wait for them until timeout;
            if not commit then
            begin
                releaseresources;
                if BatteryLoss then Device1BatteryNumber := Device1BatteryNumber - 3600;
                hold(waitTime.sample);
                Retries.update(1);
            end;
        end whileNotCommitAndAnd;

        if cannotCommitAtAll then
        begin
            completeFailure.update(1);
            commit := false;
            toDevice := 0;
        end;
        if (TIME - startTime) > timeoutTime then
        begin
            timeout.update(1);
            commit:= false;
        end;
        if commit then
        begin
            transfer;
            releaseresources;
            if BatteryLoss then Device1BatteryNumber := Device1BatteryNumber - 3600;
            AvgCommitTime.update(TIME - startTime);
            Successful.update(1);
        end ifCommit;
    end ifToDevIs0;

    if UserTimelinePosition(UserGrp,serviceCode) ne 0 and serviceCode le 8 and toDevice ne 0 then
    begin
        ServiceCode := ServiceCode + 1;
        hold(passivetime.sample);
        starttime := time;
    end
    else
    begin
        if (TIME - startTime) < timeoutTime and not cannotCommitAtAll then Complete.update(1);
    end;

    If Not Batt then DepletedBatt.update(1);

end tryToRoudUpResources;
```

Code 10 tryToRoundUpResources Procedure

VI.5.4 ReserveResources procedure

```
if toDevice=6 then
begin
    if toDevice=AppletLocation then devs := devs+S10D;
    if Device6DevicesRes.avail - devs ge 0 then
    begin
        if devs > 0 then Device6DevicesRes.avail := Device6DevicesRes.avail - devs;
        Device6Devices := true; Devices := true; if DisplaySuccess then begin Outtext(" D
Taken:");OutInt(devs,10);Outtext(" From Device6:");OutImage end;
    end
    else
    begin
        Devices := false;
        DeviceCode := 6;
        DeviceFailureTable(Devicecode,ServiceCode):=DeviceFailureTable(Devicecode,ServiceCode)+1;
        Device6DevicefailureTries.update(1);
        if Debug then outimage;OutText("Device6 device Failure, total
Devices:");outimage;outint(Device6DevicesRes.initial,4);outtext("
Available:");outint(Device6DevicesRes.avail,4);outtext(" trying to get: ");outint(devs,4);
        if Debug then outimage;
        if Device6DevicesRes.initial < devs then
        begin

            cannotCommitAtAll:= true; if keepgoing then begin keepgoing := false; outtext("Trying to get more than
total devices"); outimage;end;

        end;
    end;
end;
```

Code 11 reserveResources snippet

Please note: this is only a part of the procedure. The whole procedure is 1302 lines long.

VI.5.5 Transfer procedure

```
Procedure transfer;
begin
    holdingTime := 0;
    if toDevice = 1 then holdingTime := ((size*8)/bwneed+(pros/(Device1ProsRes+pros))+propDelay);
    if toDevice = 2 then holdingTime := ((size*8)/bwneed+(pros/(Device2ProsRes.avail+pros))+propDelay);
    if toDevice = 3 then holdingTime := ((size*8)/bwneed+(pros/(Device3ProsRes.avail+pros))+propDelay);
    if toDevice = 4 then holdingTime := ((size*8)/bwneed+(pros/(Device4ProsRes.avail+pros))+propDelay);
    if toDevice = 5 then holdingTime := ((size*8)/bwneed+(pros/(Device5ProsRes.avail+pros))+propDelay);
    if toDevice = 6 then holdingTime := ((size*8)/bwneed+(pros/(Device6ProsRes.avail+pros))+propDelay);
    if toDevice = 7 then holdingTime := ((size*8)/bwneed+(pros/(Device7ProsRes.avail+pros))+propDelay);
    if toDevice = 8 then holdingTime := ((size*8)/bwneed+(pros/(Device8ProsRes.avail+pros))+propDelay);
    if toDevice = 9 then holdingTime := ((size*8)/bwneed+(pros/(Device9ProsRes.avail+pros))+propDelay);
    hold(holdingTime);
    if debug then begin outimage; outtext(current.title); outtext("held resources for"); outint(holdingTime,10);
outtext("mSecs");outimage; end;
end transfer;
```

Code 12 Transfer procedure

VI.5.6 Release Resources procedure

```
Procedure releaseresources;
begin
    if Device1Devices then if devs > 0 then Device1DevicesRes.avail := Device1DevicesRes.avail + devs;
    if Device2Devices then if devs > 0 then Device2DevicesRes.release(devs);
    if Device3Devices then if devs > 0 then Device3DevicesRes.release(devs);
    if Device4Devices then if devs > 0 then Device4DevicesRes.release(devs);
    if Device5Devices then if devs > 0 then Device5DevicesRes.release(devs);
    if Device6Devices then if devs > 0 then Device6DevicesRes.release(devs);
    if Device9Devices then if devs > 0 then Device9DevicesRes.release(devs);
    if Device8Devices then if devs > 0 then Device8DevicesRes.release(devs);
    if Device7Devices then if devs > 0 then Device7DevicesRes.release(devs);

    if Device1Bandwidth then if BWNeed > 0 then Netw1BW := Netw1BW + BWNeed;
    if Device2Bandwidth then if BWNeed > 0 then Netw2BWRes.release(BWNeed);
    if Device3Bandwidth then if BWNeed > 0 then Netw3BWRes.release(BWNeed);
    if Device4Bandwidth then if BWNeed > 0 then Netw4BWRes.release(BWNeed);
    if Device5Bandwidth then if BWNeed > 0 then Netw5BWRes.release(BWNeed);
    if DEVICE6BANDWIDTH then if BWNeed > 0 then Netw6BWRes.release(BWNeed);
    if Device9BANDWIDTH then if BWNeed > 0 then Netw9BWRes.release(BWNeed);
    if Device8BANDWIDTH then if BWNeed > 0 then Netw8BWRes.release(BWNeed);
    if Device7BANDWIDTH then if BWNeed > 0 then Netw7BWRes.release(BWNeed);

    if Device1storage then if stor > 0 then Device1StorRes := Device1StorRes + stor;
    if Device2storage then if stor > 0 then Device2StorRes.release(stor);
    if Device3storage then if stor > 0 then Device3StorRes.release(stor);
    if Device4storage then if stor > 0 then Device4StorRes.release(stor);
    if Device5storage then if stor > 0 then Device5StorRes.release(stor);
    if Device6Storage then if stor > 0 then Device6StorRes.release(stor);
    if Device9Storage then if stor > 0 then Device9StorRes.release(stor);
    if Device8Storage then if stor > 0 then Device8StorRes.release(stor);
    if Device7Storage then if stor > 0 then Device7StorRes.release(stor);

    if Device1processing then if pros > 0 then Device1ProsRes := Device1ProsRes + pros;
    if Device2processing then if pros > 0 then Device2ProsRes.release(pros);
    if Device3processing then if pros > 0 then Device3ProsRes.release(pros);
    if Device4processing then if pros > 0 then Device4ProsRes.release(pros);
    if Device5processing then if pros > 0 then Device5ProsRes.release(pros);
    if Device6Processing then if pros > 0 then Device6ProsRes.release(pros);
    if Device9Processing then if pros > 0 then Device9ProsRes.release(pros);
    if Device8Processing then if pros > 0 then Device8ProsRes.release(pros);
    if Device7Processing then if pros > 0 then Device7ProsRes.release(pros);

            DEVICE1DEVICES:= false;
            DEVICE2DEVICES:= false;
            DEVICE3DEVICES:= false;
            DEVICE4DEVICES:= false;
.
.
.
.
end releaseresources;
```

Code 13 releaseResources procedure

VI.5.7 The Service file reader

```
for i:= 1 step 1 until 9 do
begin
    if i=1 then transferfile:-NEW InFile("Service1.txt");
    if i=2 then transferfile:-NEW InFile("Service2.txt");
    if i=3 then transferfile:-NEW InFile("Service3.txt");
    if i=4 then transferfile:-NEW InFile("Service4.txt");
    if i=5 then transferfile:-NEW InFile("Service5.txt");
    if i=6 then transferfile:-NEW InFile("Service6.txt");
    if i=7 then transferfile:-NEW InFile("Service7.txt");
    if i=8 then transferfile:-NEW InFile("Service8.txt");
    if i=9 then transferfile:-NEW InFile("Service9.txt");
    transferFile.Open(Blanks(180));
    transferFile.InImage;
    number2:=1;
    j:=0;
    if debug then begin outimage; outtext("service");outint(i,2);outimage; end;
    while number2 ne 0 do
    begin
        j:=j+1;
        for k:=1 step 1 until 8 do
        begin
            number := transferFile.InInt;
            if debug then outint(number,5);
            if k = 1 then number2:= number;
            if i = 1 then Service1Table(k,j):=number;
            if i = 2 then Service2Table(k,j):=number;
            if i = 3 then Service3Table(k,j):=number;
            if i = 4 then Service4Table(k,j):=number;
            if i = 5 then Service5Table(k,j):=number;
            if i = 6 then Service6Table(k,j):=number;
            if i = 7 then Service7Table(k,j):=number;
            if i = 8 then Service8Table(k,j):=number;
            if i = 9 then Service9Table(k,j):=number;
        end;
        if debug then outimage;
    end;
    if debug then outimage;
end;
```

Code 14 The service file reader

VI.5.8 The routing table creator

```
class routingTableClass;
begin
    for sender := 1 step 1 until numNetworks do
    begin
        for reciever := 1 step 1 until numNetworks do
        begin
            new router(reciever,sender,sender,numNetworks,-1,sender,0,0,0,0,0,0,0,0,0);
            for jumps:=0 step 1 until numNetworks do
            begin
                for network:=1 step 1 until numNetworks do
                begin
                    if network ne numNetworks then
                    begin
                        if BuildRTable(sender,reciever,jumps,network) = reciever then
                        begin
                            for i := network+1 step 1 until numNetworks do
                            begin
                                BuildRTable(sender,reciever,jumps,i):=0;
                            end;
                        end;
                    end;
                end;
            end;
            found:=false;
            jumps:=0;
            while not found do
            begin
                if BuildRTable(sender,reciever,jumps,1) = 0 then
                begin
                    jumps:= jumps+1;
                end
                else
                begin
                    for network:=1 step 1 until numNetworks do
                    begin
                        found:=true;
                    end;
                end;
            end;
        end reciverFor;
    end senderFor;
end RoutingTable;
```

Code 15 Routing Table Creator

VI.5.9 The route sniffer

```
class router (reciever,sender,nextjump,numNetworks,jumps,routeNum,lastjump,jump1,...,jump9);
    integer reciever,sender,numNetworks,nextjump,jumps,routeNum,lastjump,jump1,...,jump9;
    begin
        integer route,jumpNo;
        integer thisjump;
        route:= routeNum;
        jumps := jumps+1;
        if jumps < numNetworks then
        begin
            if sender ne reciever then
                begin
                for thisjump:=1 step 1 until numNetworks do
                begin
                    if thisjump ne nextjump and thisjump ne lastjump and thisjump ne jump1 and thisjump ne jump2 and
thisjump ne jump3 and thisjump ne jump4 and thisjump ne jump5 and thisjump ne jump6 and thisjump ne jump7 and thisjump
ne jump8 and thisjump ne jump9 then
                    begin
                    lastjump := thisjump;
                    if NetwConnectionstable(nextjump,thisjump)=1 then
                        begin
                            if thisjump = reciever then
                            begin
                                jumpNo:=thisjump;
                                for i:=0 step 1 until numNetworks do
                                begin
                                    if i=0 then JumpNo:=sender;
                                    if i=1 then JumpNo:=jump1;
                                    if i=2 then JumpNo:=jump2;
.
.
                                    if i=9 then JumpNo:=jump9;
                                    BuildRTable(sender,reciever,jumps+1,i+1):=JumpNo;
                                end;
                                BuildRTable(sender,reciever,jumps+1,jumps+2):=reciever;
    route:=jump1+jump2*10+jump3*100+jump4*1000+jump5*10000+jump6*100000+jump7*1000000+jump8*10000000+jump9*10000000;
                                routes(jumps,0):=route;
                                routes(jumps,1):=routes(jumps,1)+1;
                            end
                            else
                            begin
                                if jumps=0 then Jump1:=thisjump;
.
.
                                if jumps=8 then Jump9:=thisjump;
                                jumpNo:=nextjump;
                                new
router(reciever,sender,thisjump,numNetworks,jumps,route,lastjump,jump1,jump2,jump3,jump4,jump5,jump6,jump7,jump8,jump9
);
                            end;
                        end;
                    end;
                end;
            end
            else
            begin
                routes(0,0):=sender;
                routes(0,1):=1;
                BuildRTable(sender,reciever,1,1):=reciever;
            end;
        end;
    end router;
```

Code 16 Route Sniffer

VI.5.10 User Group Selector

```
Procedure FindUserGroup;
begin
    if number = 1 then begin if uT1 > 0 then begin uT1 := uT1-1; UserType := 1;    end      else begin FindUG2;
    end; end;
    if number = 2 then begin if uT2 > 0 then begin uT2 := uT2-1; UserType := 2; endelse begin FindUG2;         end;
end;
    if number = 3 then begin if uT3 > 0 then begin uT3 := uT3-1; UserType := 3; endelse begin FindUG2;         end;
end;
    if number = 4 then begin if uT4 > 0 then begin uT4 := uT4-1; UserType := 4; endelse begin FindUG2;         end;
end;
    if number = 5 then begin if uT5 > 0 then begin uT5 := uT5-1; UserType := 5; endelse begin FindUG2;         end;
end;
end;


Procedure FindUG2;
begin
    if uT1 = 0 and uT2 = 0 and uT3 = 0 and uT4 = 0 and uT5 = 0 then
    begin
        uT1:=UserTimelinePosition(1,10)/10;
        uT2:=UserTimelinePosition(2,10)/10;
        uT3:=UserTimelinePosition(3,10)/10;
        uT4:=UserTimelinePosition(4,10)/10;
        uT5:=UserTimelinePosition(5,10)/10;
    end;

    if uT1 > 0 then
    begin
        uT1 := uT1-1;
        UserType := 1;
    end else if uT2 > 0 then
    begin
        uT2 := uT2-1;
        UserType := 2;
    end else if uT3 > 0 then
    begin
        uT3 := uT3-1;
        UserType := 3;
    end else if uT4 > 0 then
    begin
        uT4 := uT4-1;
        UserType := 4;
    end else if uT5 > 0 then
    begin
        uT5 := uT5-1;
        UserType := 5;
    end;
end FindUG2;
```

Code 17 User Group Selector

VI.5.11 XML output file entity class

```
entity class fileout;
begin
    integer lastAppletlocation,holdingtime;

    while appletlocation = 0 do begin hold(100); end;

        if endRec = 0  and runShort then endRec := simtime;
        if endRec = 0  and not runShort then endRec := simtime*6;
        holdingtime := ((endRec-startRec)/samples);
        xmlLines := 100000;
    while (time-runTime) < startRec do hold(1);
    xmlStartWriter;
    while maxusers = 0  do begin hold(10); end;
    while appletlocation le EndLocation do
    begin
        if appletlocation ne lastAppletLocation then
            begin
            xmlNewSheetstart;
            lastAppletlocation := appletlocation;
        end;
        while(appletlocation = lastAppletLocation) and (time-runTime) < endRec do
        begin
            xmlnewrow;
            number := (time-RunTime); xmlLineWriter;
            number := (((users.initial-users.avail))/(maxusers))*100; xmlLineWriter;

            if numdevices > 0 then begin number :=100-(Device1DevicesRes.avail/Device1DevicesRes.initial)*100;
xmlLineWriter; end;
            if numdevices > 1 then begin number :=100-(Device2DevicesRes.avail/Device2DevicesRes.initial)*100;
xmlLineWriter; end;
.
.

            if numdevices > 1 then begin number :=100-(Device2StorRes.avail/Device2StorRes.initial)*100; xmlLineWriter;
end;
            if numdevices > 2 then begin number :=100-(Device3StorRes.avail/Device3StorRes.initial)*100; xmlLineWriter;
end;
.
.
            if numdevices > 1 then begin number :=100-(Device2ProsRes.avail/Device2ProsRes.initial)*100; xmlLineWriter;
end;
            if numdevices > 2 then begin number :=100-(Device3ProsRes.avail/Device3ProsRes.initial)*100; xmlLineWriter;
end;
.
.           if numnetworks > 1 then begin number :=100-(Netw2BWRes.avail/Netw2BWRes.initial)*100; xmlLineWriter; end;
            if numnetworks > 2 then begin number :=100-(Netw3BWRes.avail/Netw3BWRes.initial)*100; xmlLineWriter; end;
.
.           number := reTries.obs; xmlLineWriter;
            number := timeout.obs; xmlLineWriter;
            xmlendrow;
            hold (holdingtime);
            while reset do
            begin
                xmlwriterreset:= true;
                hold(1);
            end;
            if (time-runTime) ge endRec and runPass ne 0 then appletlocation:=appletlocation+1;
        end appletlocationNotChanged;
        hold(holdingtime);
        while (time-runTime) > endRec and appletlocation = lastAppletLocation do begin hold(100); end;
        if runPass = 0 then xmlNewSheetend;
    end EachService;
end fileoutandtimer;
```

Code 18 XML output file writer entity class

VI.5.12 Interuser entity Class

```
entity class interuserclass;
begin
    integer temporaryInteger;
    if InterUserTime = 0 then
    begin
        while (Runtime/1000) < (SIMTIME*6/1000) do
        begin
            temporaryInteger := (time-RunTime)/(3600000);
            while temporaryInteger ge 23 do begin temporaryInteger := temporaryInteger - 23; end;
            InterUser := 1/(InteruserTable(temporaryInteger)/3600000);
            maxusers := InteruserTable(temporaryInteger);
            !outint(InteruserTable(temporaryInteger),10);
            !outint(temporaryInteger,10);
            !outint((users.initial-users.avail),10);
            !outimage;
            hold(1000000);
        end;
    end
    else
    begin
        Interuser := 1/(InterUserTime/3600000);
        maxusers := users.initial;
    end;
end;
```

Code 19 Interuser entity class

VI.6 APPENDIX F: Sources on the internet

http://www.dgp.toronto.edu/people/JamesStewart/270/9798s/Laffra/DijkstraApplet.html
Link I Explanation applet to Dijkstra's shortest path algorithm

http://portal.acm.org/ The ACM digital Library
Link II ACM digital library

http://ieeexplore.ieee.org/ IEEEXplore online digital library
Link III IEEE digital library

http://en.wikipedia.org/wiki/Dijkstra Information regarding Edsger Dijkstra
Link IV Edsger Dijkstra

http://www.item.ntnu.no/fag/SIE5015/cim/SimulaCimDemos.htm
Link V Simula, cim and DEMOS guide

VI.7 APPENDIX G: Extra tables of interest

VI.7.1 Scenario 1:

Table 16 Runtime information from Scenario 1

Applet Location	Time Used	Complete Failures	Users Started	Tries	Retries	Successful %	Timeouts	Empty UserBins
1	19390077	100	100	956	100	0	48	4250
2	21197349	0	108	1080	0	100	0	7146
3	22029544	0	103	1030	0	100	0	7164
4	17996672	100	100	963	100	0	39	2832
5	21352827	0	106	1060	0	100	0	7168
6	20382293	0	104	1040	0	100	0	7169
7	23455537	0	108	1080	0	100	0	7068

VI.7.2 Scenario 2:

Table 17 runtime results from Scenario 2 with seed 654321, userload 10800, scenario 2

Applet Location	Time Used	Complete Failures	Users Started	Tries	Retries	Successful %	Timeouts	Empty UserBins
1	22544775	1741	1741	16727	1741	0	724	0
2	108581963	0	259200	2672533	648739	76	568206	0
3	108966352	0	259200	2633280	373215	86	331935	0
4	21972287	616	616	5933	617	0	247	0
5	109061434	0	259201	2608327	37968	99	21651	0
6	109822809	0	259200	2608504	37964	99	21460	0
7	109702399	0	259200	2608236	37943	99	21707	0

Table 18 runtime results from Scenario 2 with seed 78946513, userload 10800, scenario 2

Applet Location	Time Used	Complete Failures	Users Started	Tries	Retries	Successful %	Timeouts	Empty UserBins
1	21541123	670	670	6444	670	0	275	0
2	109668644	0	259200	2671301	647226	76	567925	0
3	109349855	0	259201	2634794	372743	86	329959	0
4	24622855	2779	2779	26702	2847	0	1236	0
5	109232211	0	259200	2608280	38039	99	21759	0
6	109329308	0	259201	2608279	37669	99	21400	0
7	110076199	0	259201	2608014	37786	99	21782	0

Table 19 runtime results from Scenario 2 with seed 41, userload 10800, scenario 2

Applet Location	Time Used	Complete Failures	Users Started	Tries	Retries	Successful %	Timeouts	Empty UserBins
1	21693835	1532	1532	14696	1532	0	676	0
2	110344223	0	259201	2672152	649043	76	568901	0
3	108866767	0	259200	2633110	373139	86	332029	0
4	22372672	1045	1045	10050	1050	0	433	0
5	108754507	0	259200	2607984	37654	99	21670	0
6	108993964	0	259200	2608415	38070	99	21655	0
7	111156071	0	259200	2607948	37974	99	22026	0

Table 20 runtime results from Scenario 2 with seed 100, userload 10800, scenario 2

Applet Location	Time Used	Complete Failures	Users Started	Tries	Retries	Successful %	Timeouts	Empty UserBins
1	22775293	2249	2249	21668	2249	0	884	0
2	108930405	0	259201	2672486	648992	76	568516	0
3	108741268	0	259201	2632486	372926	86	332450	0
4	22370002	1365	1365	13161	1377	0	538	0
5	110604230	0	259201	2608312	38054	99	21752	0
6	108790220	0	259200	2608082	37935	99	21853	0
7	109589259	0	259201	2608213	37931	99	21728	0

Table 21 runtime results from Scenario 2 with seed 357, userload 10800, scenario 2

Applet Location	Time Used	Complete Failures	Users Started	Tries	Retries	Successful %	Timeouts	Empty UserBins
1	23093581	2049	2049	19671	2049	0	884	0
2	110030945	0	259200	2672777	648927	76	568150	0
3	108570078	0	259201	2635995	373570	86	329585	0
4	23083735	1876	1876	18117	1898	0	715	0
5	109313354	0	259201	2608106	37906	99	21810	0
6	110376789	0	259201	2608272	37979	99	21717	0
7	109651855	0	259200	2608084	37645	99	21561	0

Table 22 runtime results from Scenario 2 with seed 413, userload 10800, scenario 2

Applet Location	Time Used	Complete Failures	Users Started	Tries	Retries	Successful %	Timeouts	Empty UserBins
1	23571670	2129	2129	20511	2129	0	830	0
2	109513192	0	259200	2672364	648640	76	568276	0
3	109313516	0	259201	2633857	373781	86	331934	0
4	23550196	1890	1890	18113	1913	0	873	0
5	109422762	0	259201	2608444	38320	99	21886	0
6	109399718	0	259201	2607822	37352	99	21540	0
7	109788386	0	259200	2608303	37677	99	21374	0

Table 23 runtime results from Scenario 2 with seed 1000, userload 10800, scenario 2

Applet Location	Time Used	Complete Failures	Users Started	Tries	Retries	Successful %	Timeouts	Empty UserBins
1	22203425	1397	1397	13480	1397	0	519	0
2	110649962	0	259200	2674037	650135	76	568098	0
3	109407876	0	259200	2633967	373064	86	331097	0
4	22620891	3154	3154	30318	3272	0	1439	0
5	109758723	0	259201	2608028	37493	99	21475	0
6	109746574	0	259201	2608437	38154	99	21727	0
7	109315072	0	259201	2608475	38599	99	22134	0

Table 24 runtime results from Scenario 2 with seed 123465, userload 10800, scenario 2

Applet Location	Time Used	Complete Failures	Users Started	Tries	Retries	Successful %	Timeouts	Empty UserBins
1	23388127	1795	1795	17265	1795	0	736	0
2	109800867	0	259200	2671530	648208	76	568678	0
3	110513102	0	259201	2633961	372980	86	331029	0
4	23445592	1743	1743	16728	1761	0	769	0
5	110330886	0	259201	2608356	38067	99	21721	0
6	108684049	0	259201	2607948	37511	99	21573	0
7	108752240	0	259201	2608564	38422	99	21868	0

Table 25 runtime results from Scenario 2 with seed 1000001, userload 10800, scenario 2

Applet Location	Time Used	Complete Failures	Users Started	Tries	Retries	Successful %	Timeouts	Empty UserBins
1	22423584	2163	2163	20773	2163	0	912	0
2	108926102	0	259200	2673111	650049	76	568938	0
3	108870773	0	259201	2633734	373272	86	331548	0
4	22670807	444	444	4262	444	0	192	0
5	109430455	0	259201	2608754	38213	99	21469	0
6	109818843	0	259201	2608068	38106	99	22048	0
7	108963700	0	259200	2608505	38129	99	21624	0

Table 26 runtime results from Scenario 2 with seed 465, userload 10800, scenario 2

Applet Location	Time Used	Complete Failures	Users Started	Tries	Retries	Successful %	Timeouts	Empty UserBins
1	21900490	2236	2236	21520	2236	0	899	0
2	109362653	0	259201	2673186	649230	76	568054	0
3	109449704	0	259201	2633424	372802	86	331388	0
4	22927667	1397	1397	13433	1406	0	593	0
5	109486838	0	259200	2608313	37609	99	21296	0
6	110314584	0	259200	2608003	37820	99	21817	0
7	109889407	0	259200	2608041	37856	99	21815	0

Table 27 runtime results from Scenario 2 with seed 1, userload 10800, scenario 2

Applet Location	Time Used	Complete Failures	Users Started	Tries	Retries	Successful %	Timeouts	Empty UserBins
1	21541123	670	670	6444	670	0	275	0
2	109668644	0	259200	2671301	647226	76	567925	0
3	109349855	0	259201	2634794	372743	86	329959	0
4	24622855	2779	2779	26702	2847	0	1236	0
5	109232211	0	259200	2608280	38039	99	21759	0
6	109329308	0	259201	2608279	37669	99	21400	0
7	110076199	0	259201	2608014	37786	99	21782	0

VI.7.3 Scenario 3:

Table 28 Runtime information from Scenario3

Applet Location	Time Used	Complete Failures	Users Started	Tries	Retries	Successful %	Timeouts	Empty UserBins
1	22958300	2820	2820	27168	2820	0	1120	0
2	109674455	0	259201	2677283	641682	76	556409	0
3	109265066	0	259201	2647956	375906	86	319960	0
4	23956731	1169	1169	11271	1180	0	466	0
5	110057279	0	259201	2606301	37993	99	23702	0
6	108963432	0	259200	2606085	37656	99	23571	0
7	109225693	0	259200	2606261	37657	99	23396	0

VI.7.4 Scenario 5:

Dynamic Userload:

Table 29 Dynamic run 3 seed 1000001

Applet Location	Time Used	Complete Failures	Users Started	Tries	Retries	Successful %	Timeouts	Empty UserBins
1	22085679	60	60	573	60	0	30	0
2	102284826	0	16407	164197	304	100	177	0
3	105746786	0	16305	163083	88	100	55	0
4	21732494	152	152	1438	152	0	91	0
5	100516754	0	16253	162530	0	100	0	0
6	102294024	0	16263	162630	0	100	0	0
7	99698195	0	16238	162380	0	100	0	0

Table 30 Dynamic run 3 seed 1000

Applet Location	Time Used	Complete Failures	Users Started	Tries	Retries	Successful %	Timeouts	Empty UserBins
1	24416148	143	143	1367	143	0	71	0
2	103891728	0	16587	165991	290	100	169	0
3	102964225	0	16300	163022	73	100	51	0
4	22194262	79	79	752	79	0	42	0
5	101657856	0	16447	164470	0	100	0	0
6	104228164	0	16368	163680	0	100	0	0
7	103517023	0	16312	163120	0	100	0	0

VI.8 APPENDIX H: Result graphs

VI.8.1 Scenario 1

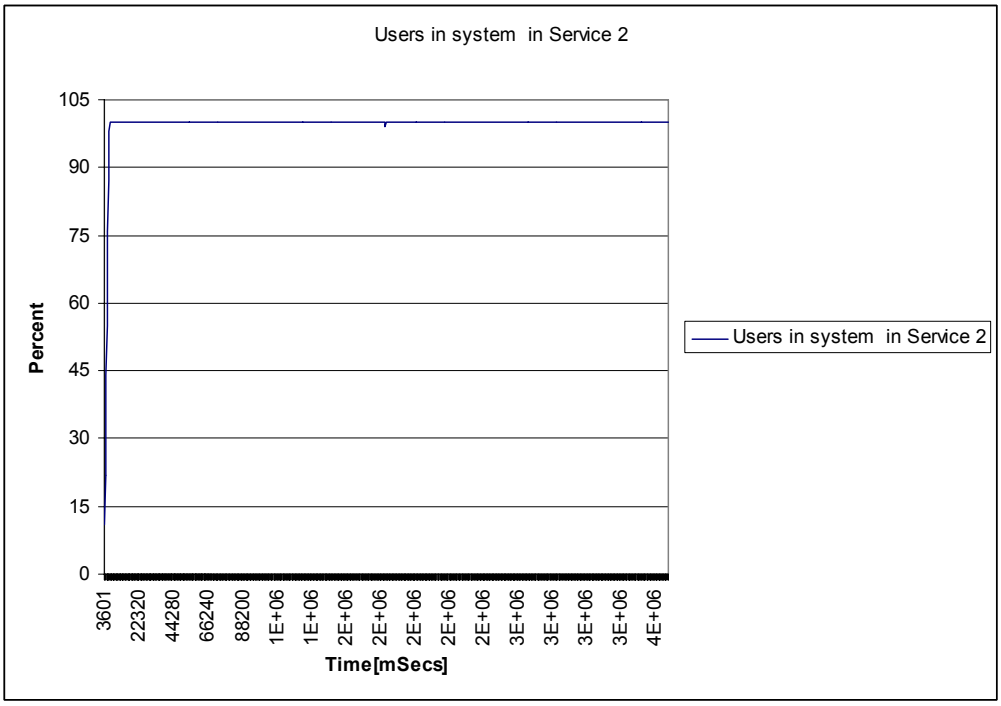


Figure 45 Users in system in scenario 1

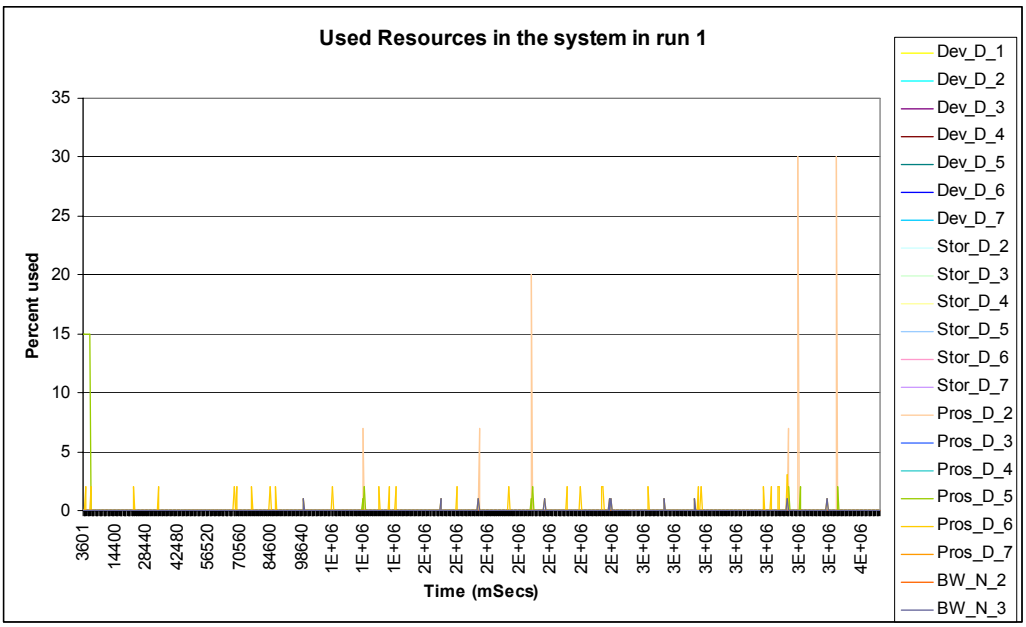


Figure 46 Runtime results from applet location 2. Used resources in scenario 1

VI.8.2 Scenario 2

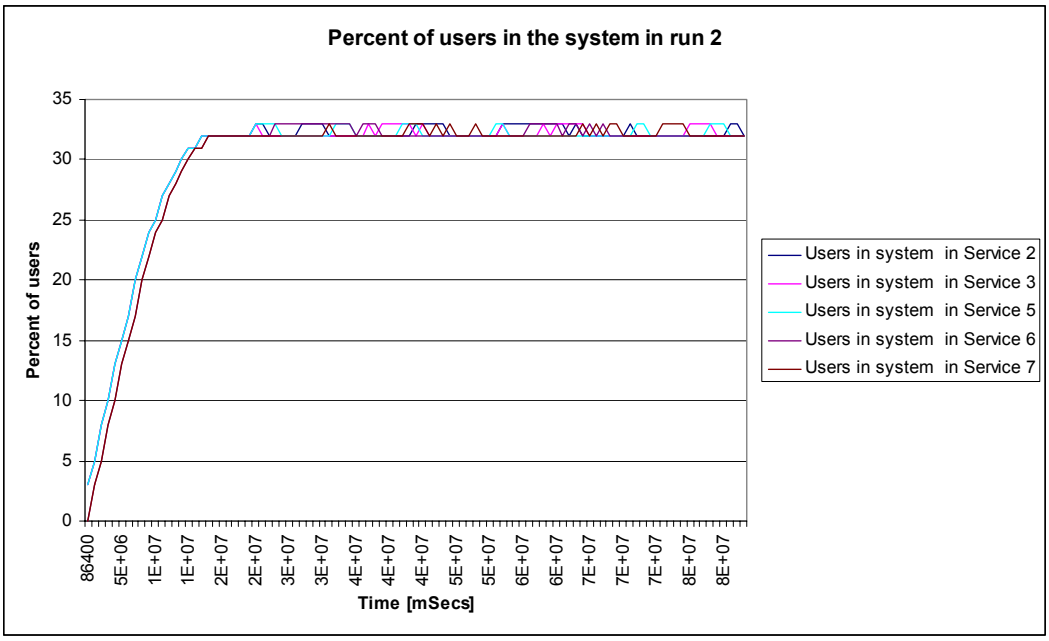


Figure 47 Users in the system in scenario 2 from all applet locations

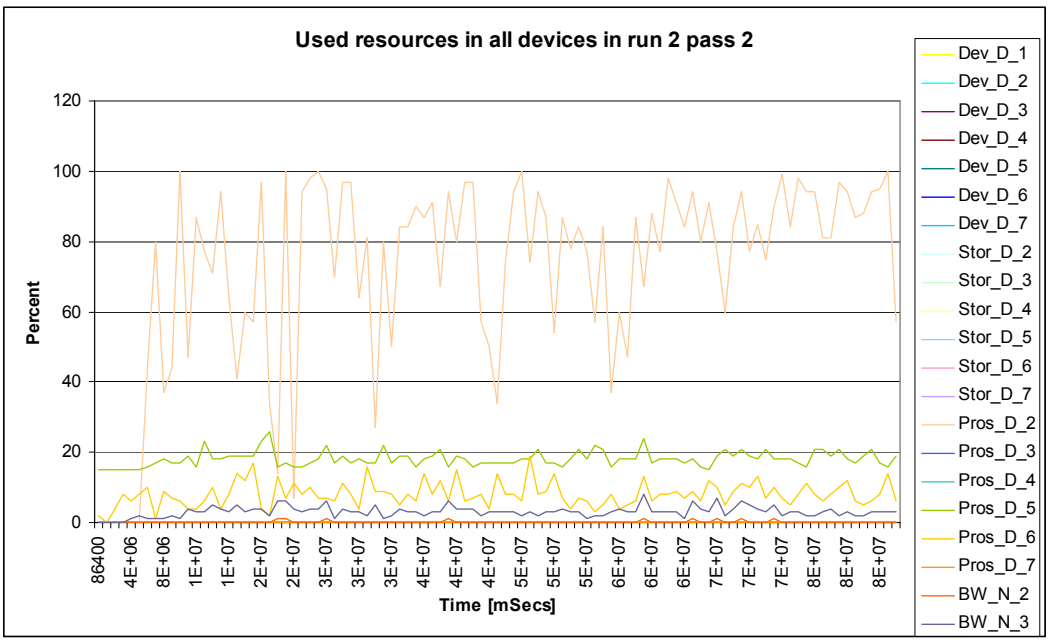


Figure 48 Runtime results of available resources with the applet placed in device 2 in scenario 2

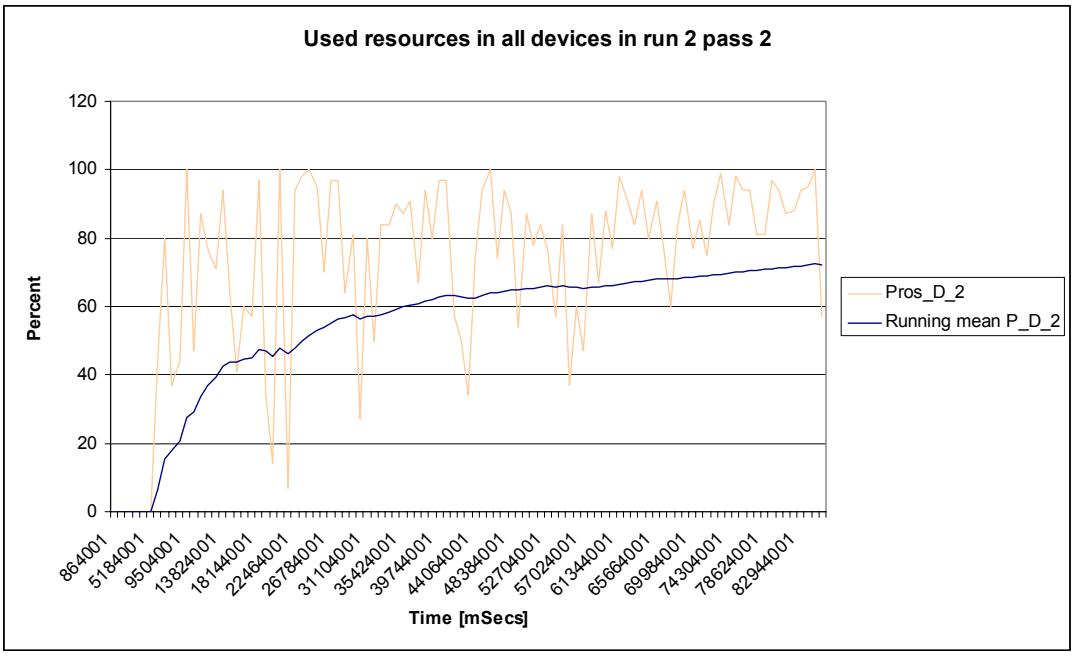


Figure 49 Runtime resource usage of Processing in device 2 scenario 2 pass 2 with running mean

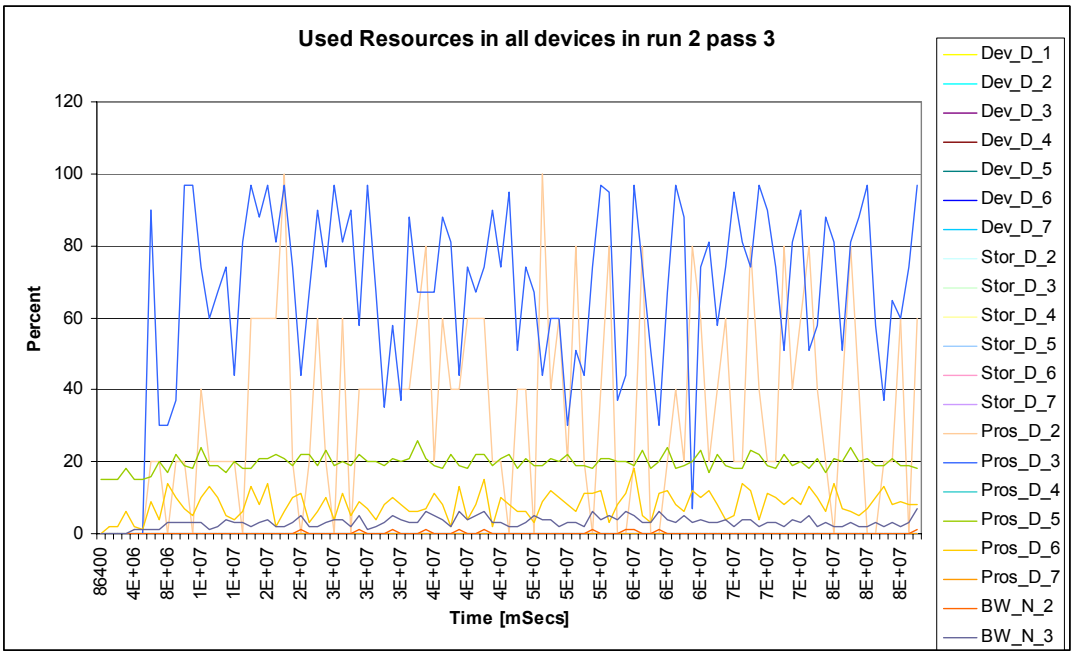


Figure 50 Runtime results of available resources with the applet placed in device 3 in scenario 2

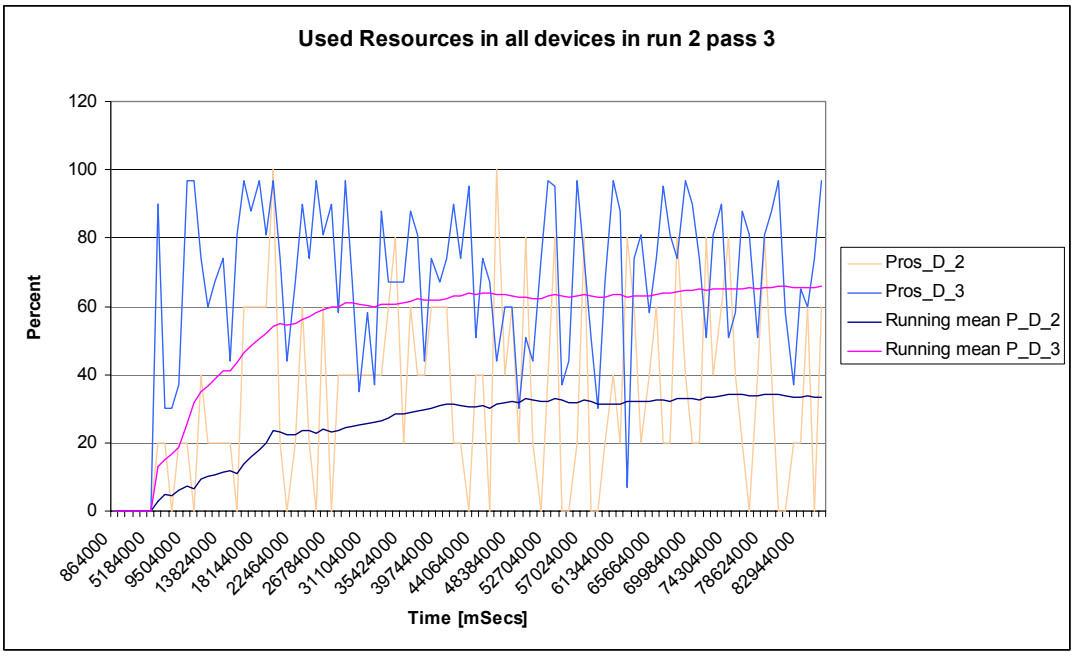


Figure 51 Runtime resource usage of Processing in device 2 and 3, scenario 2, pass 3 with running mean

Figure 52 runtime information from 10 runs of scenario 2. Timeout values are shown with variance

VI.8.3 Scenario 3

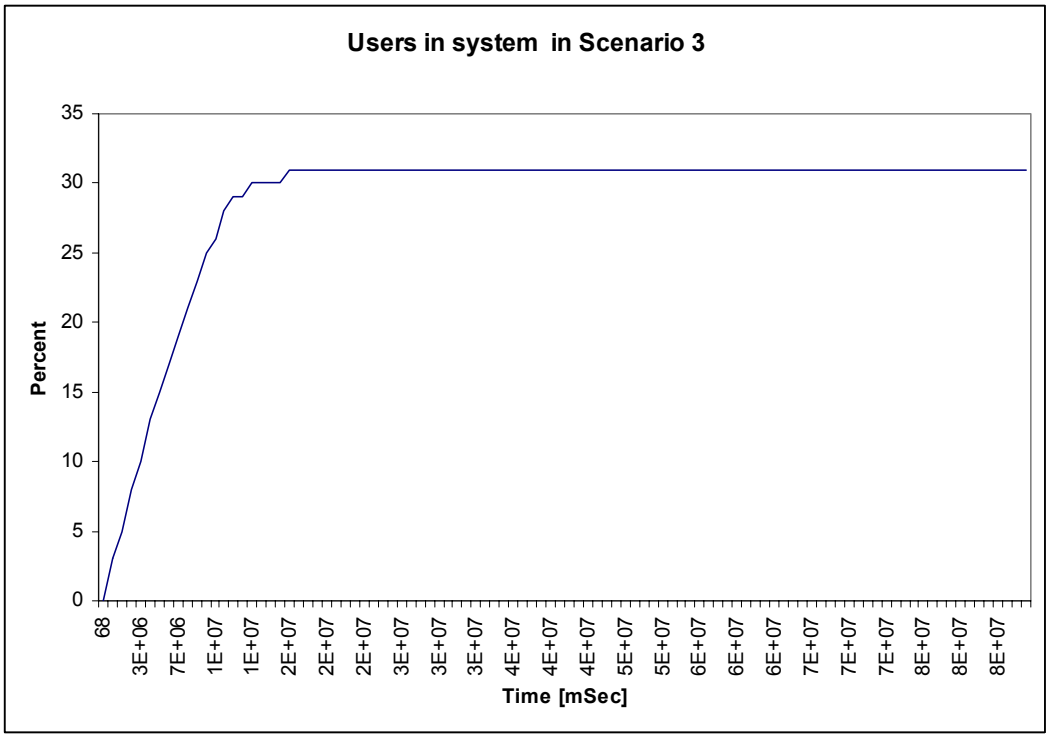


Figure 53 Users in system in scenario 3

Locations 1 and 4 are omitted due to the same failures as in scenarios 1 and 2.

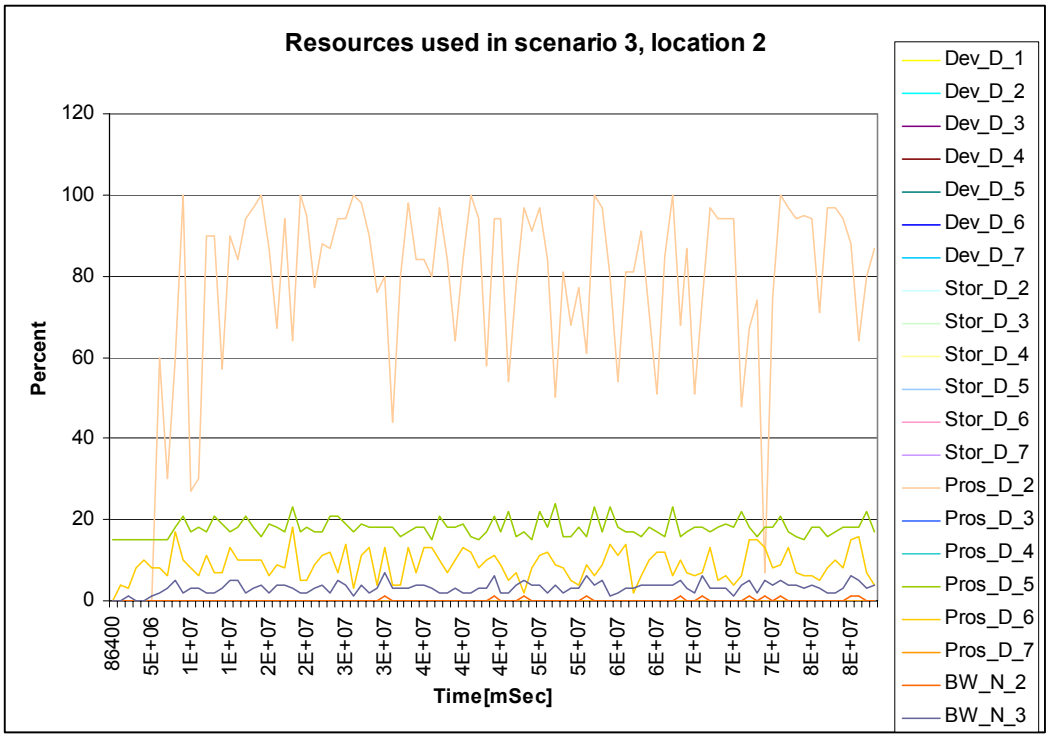


Figure 54 Resources used in scenario 3, location 2

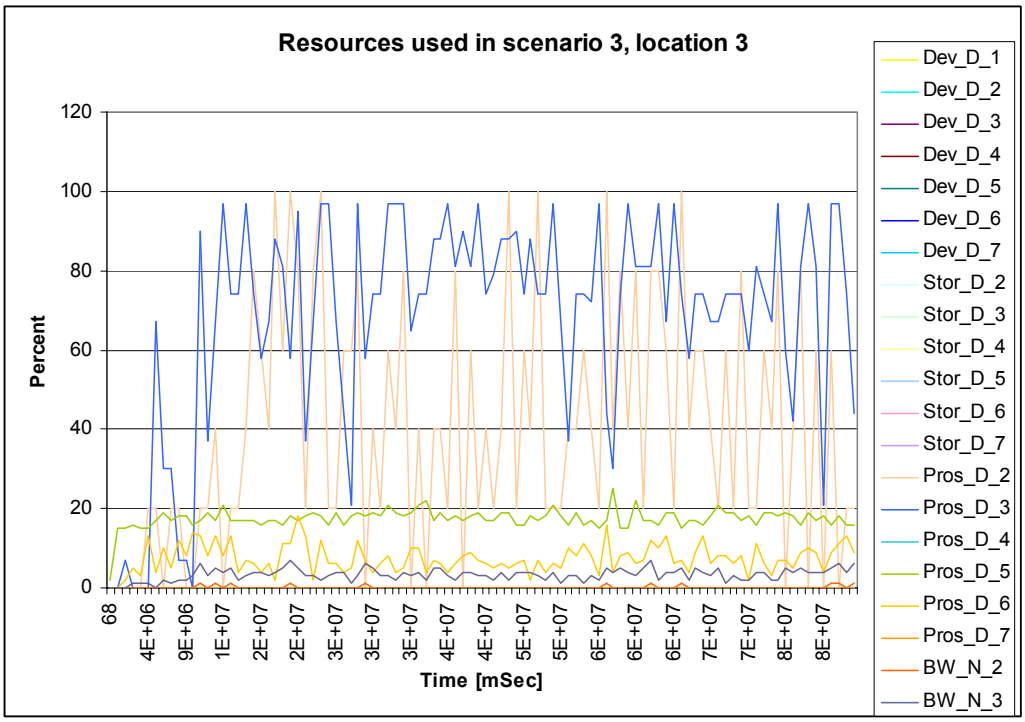


Figure 55 Resources used in scenario 3, location 3

VI.8.4 Scenario 4

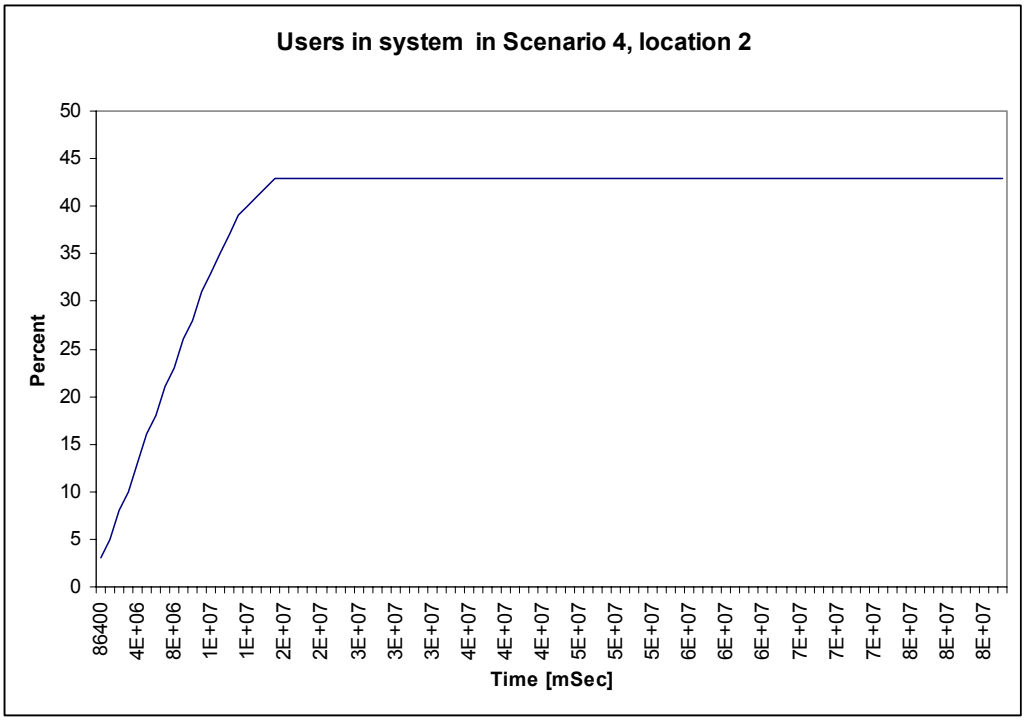


Figure 56 Users in system in scenario 4

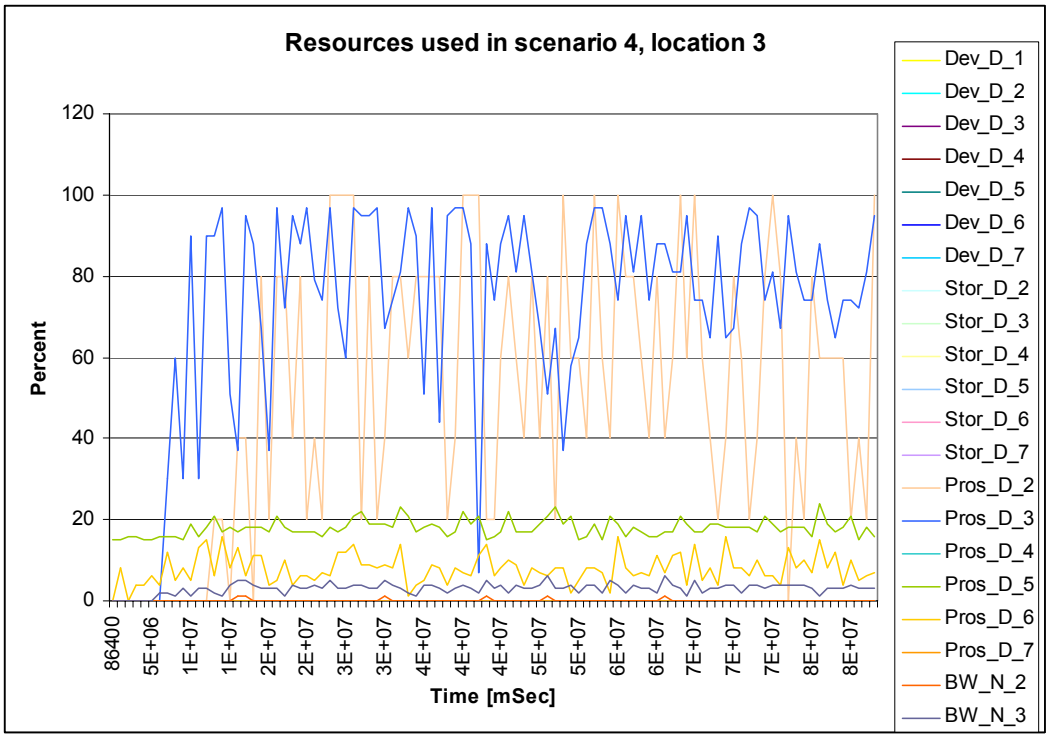


Figure 57 Resources used in scenario 4, location 3

VI.8.5 Scenario 5

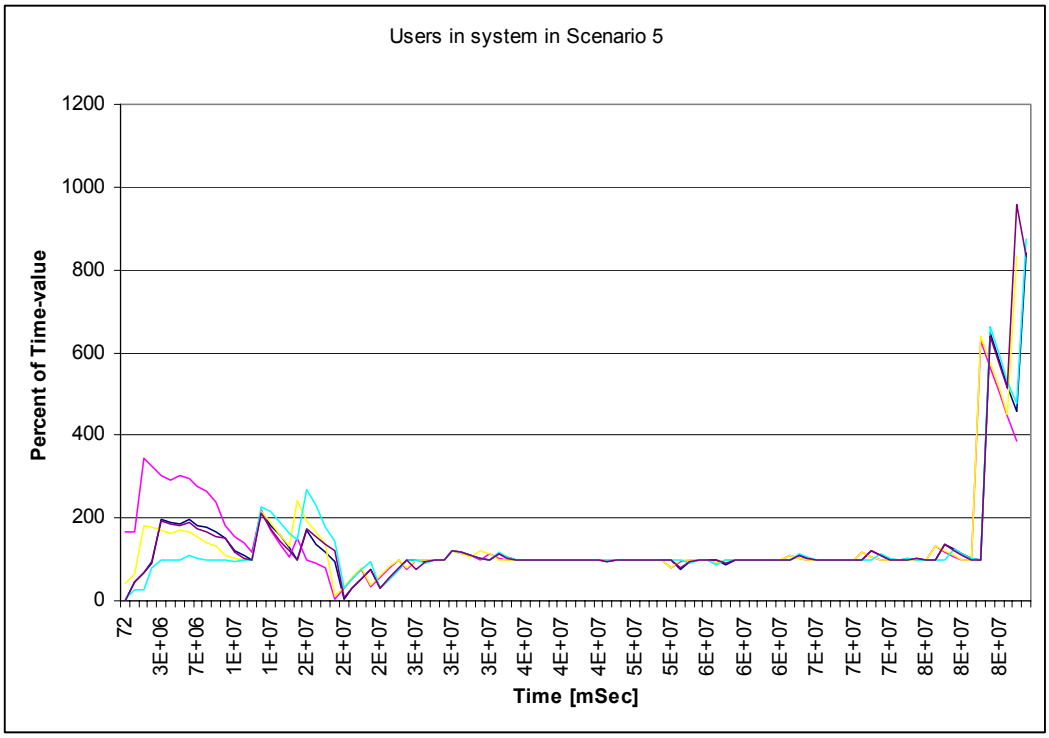


Figure 58 Users in system in scenario 5

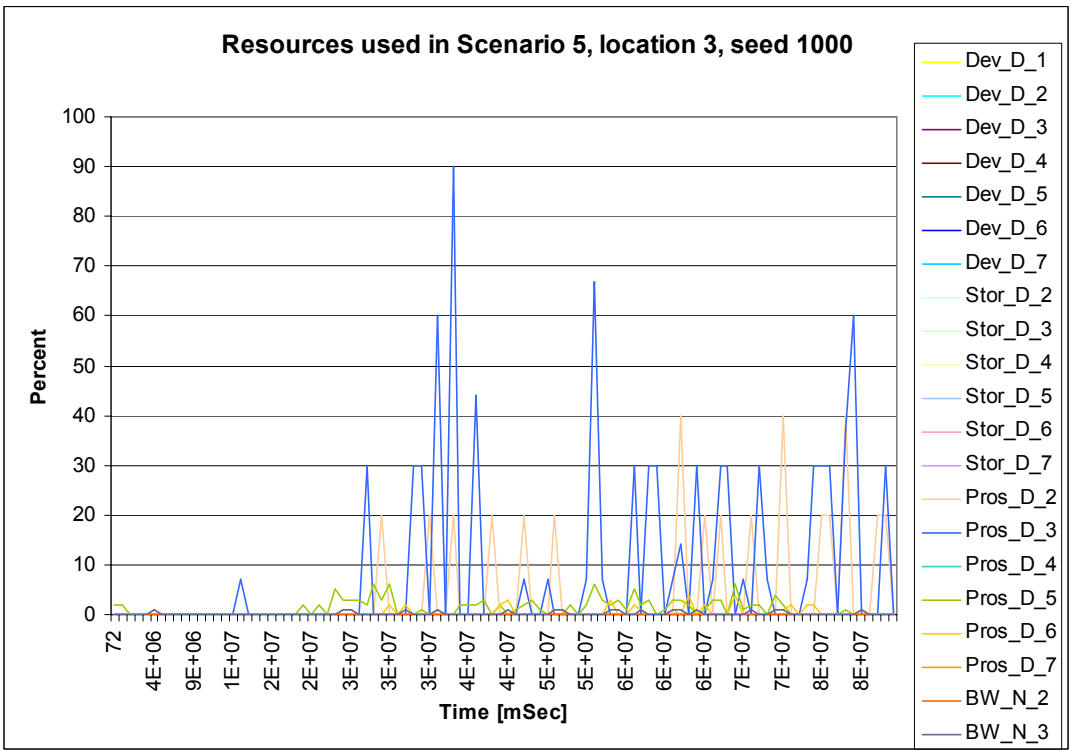


Figure 59 Resources used in scenario 5, location 3

VI.8.6 Scenario 6

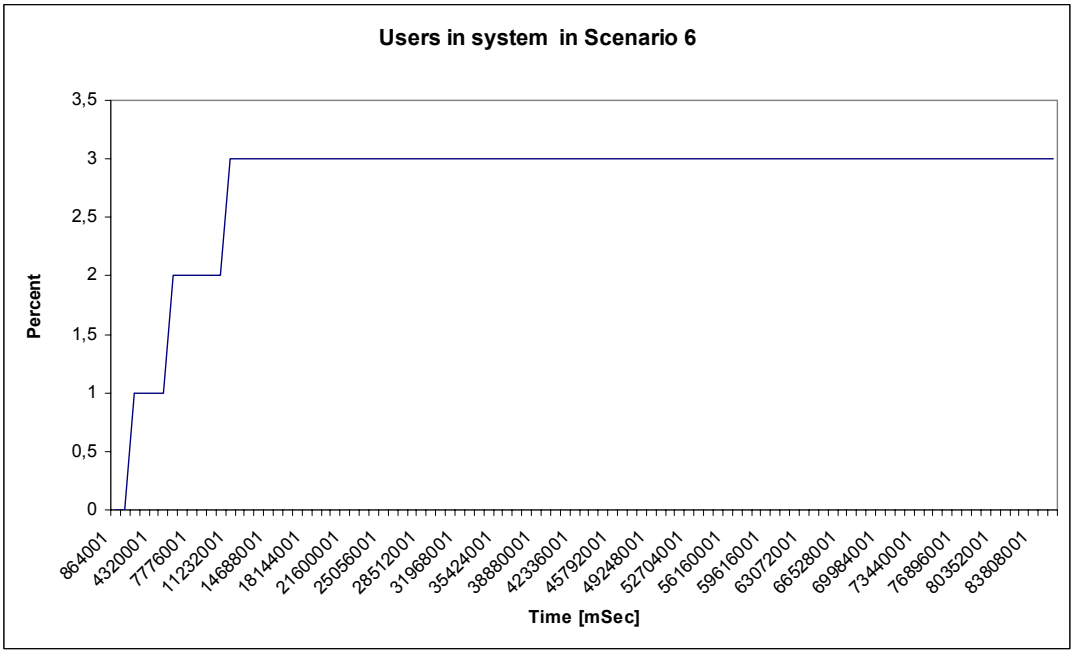


Figure 60 Users in system in scenario 6

VI.8.7 Scenario 7

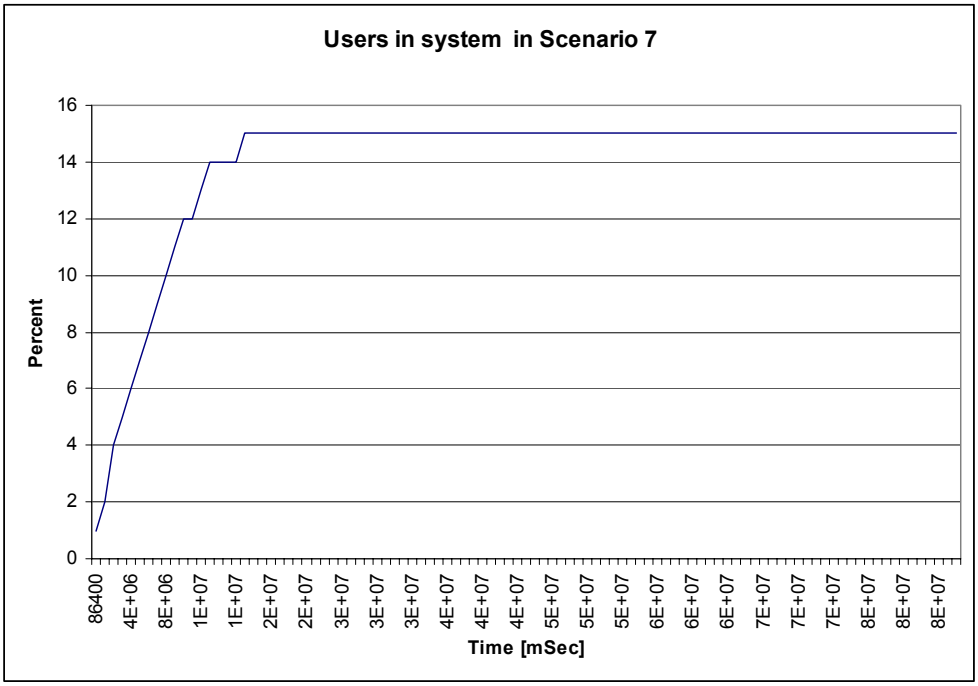


Figure 61 Users in the system in Scenario 7

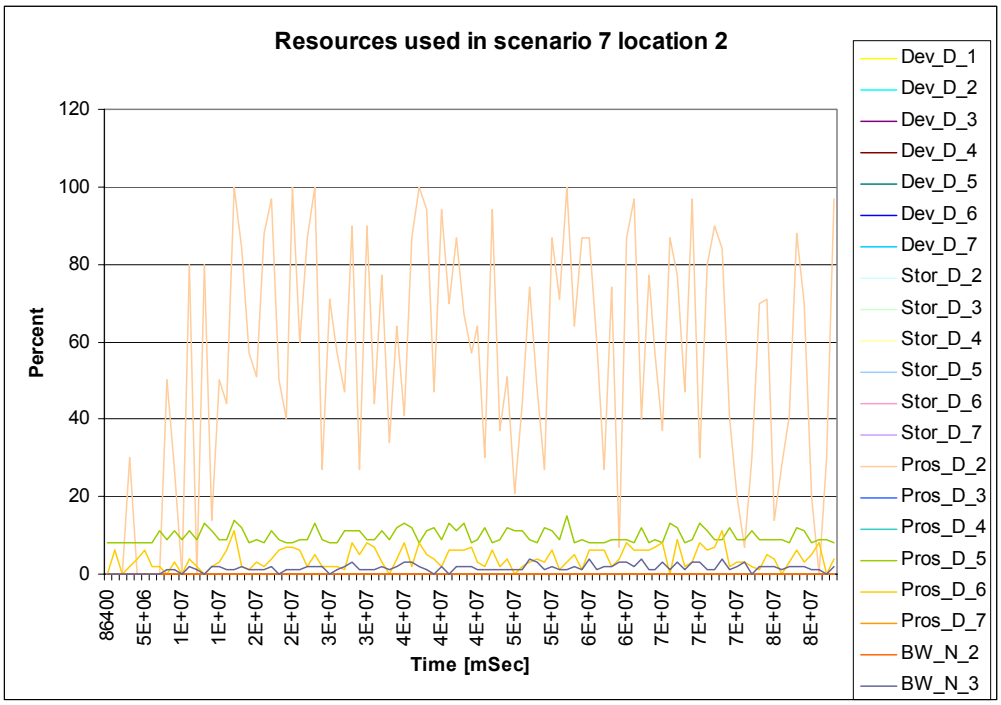


Figure 62 Resources used in scenario 7, location 2

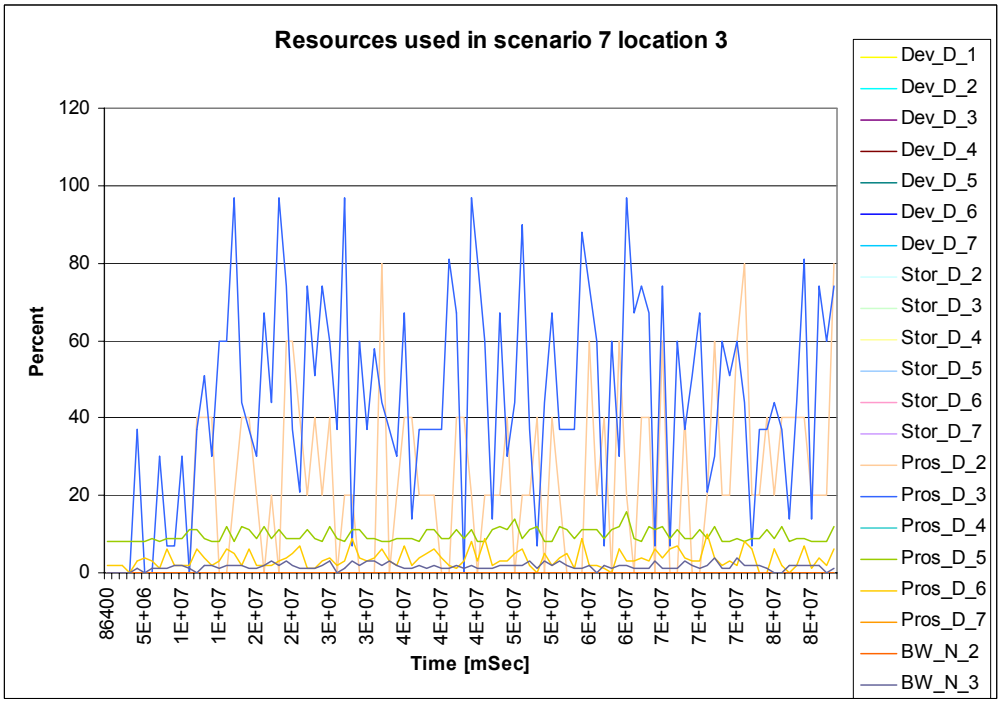


Figure 63 Resources used in scenario 7, location 3

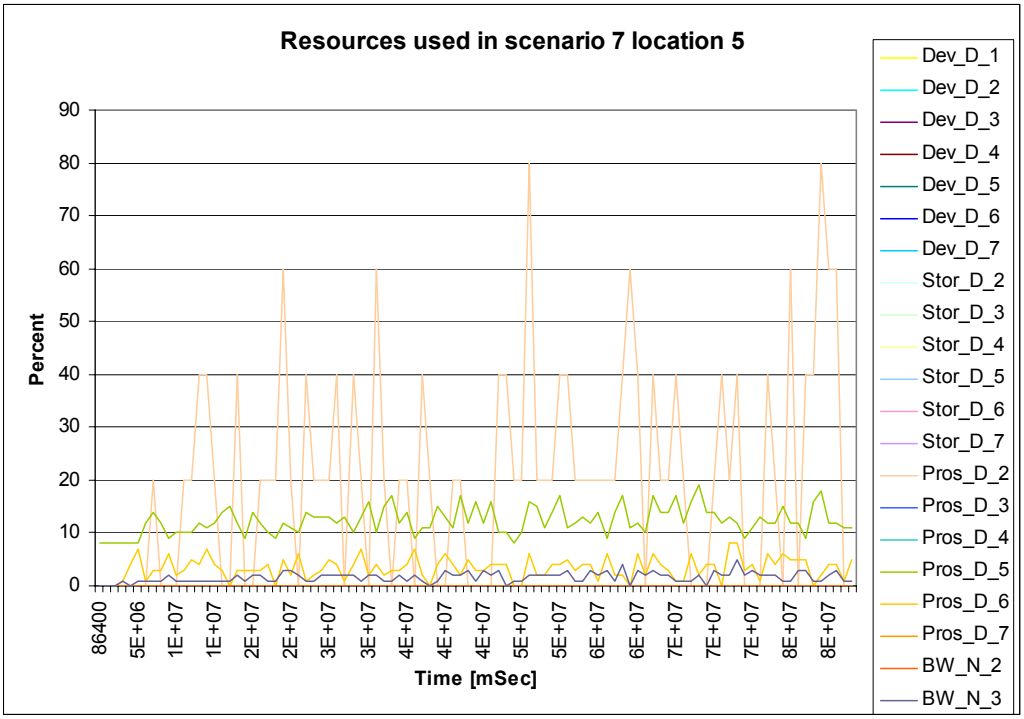


Figure 64 Resources used in scenario 7, location 5

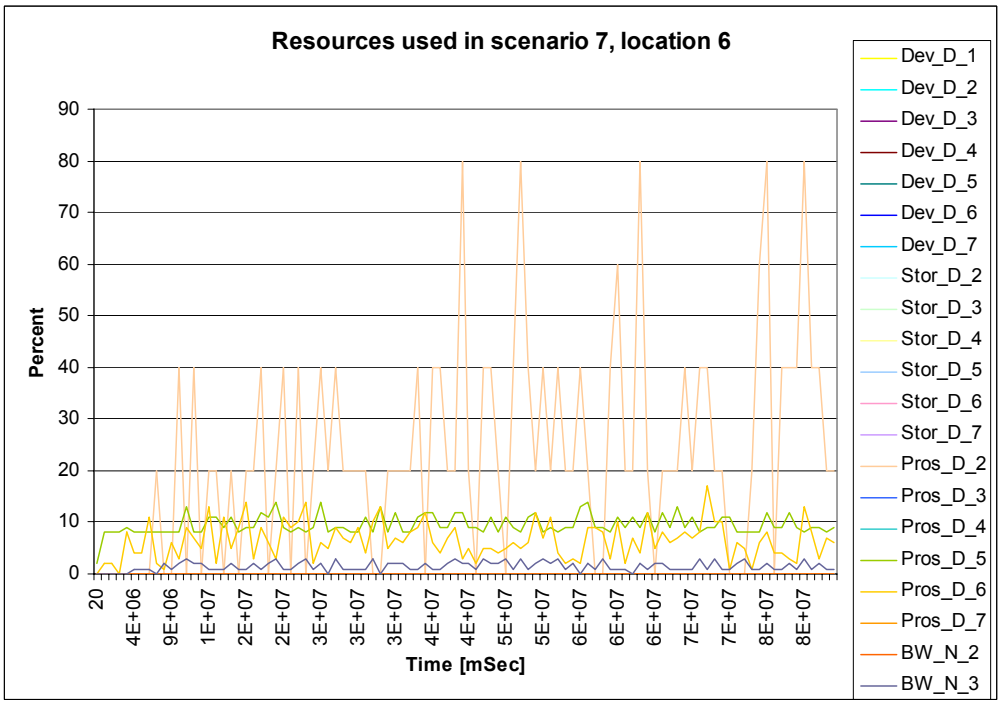


Figure 65 Resources used in scenario 7, location 6

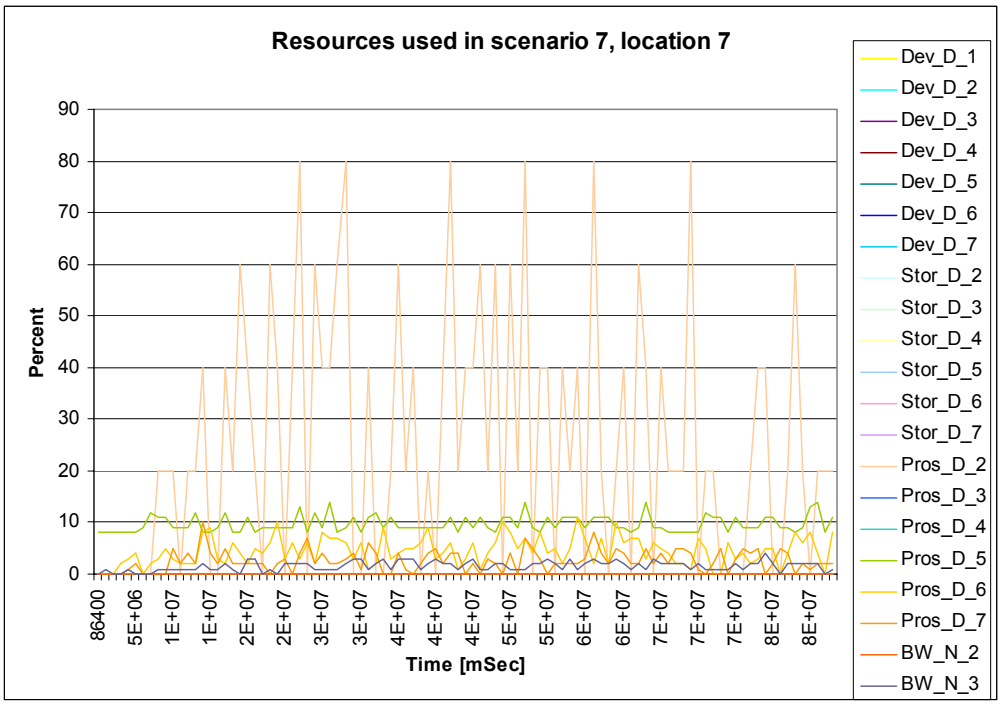


Figure 66 Resources used in scenario 7, location 7

VI.8.8 Scenario 8

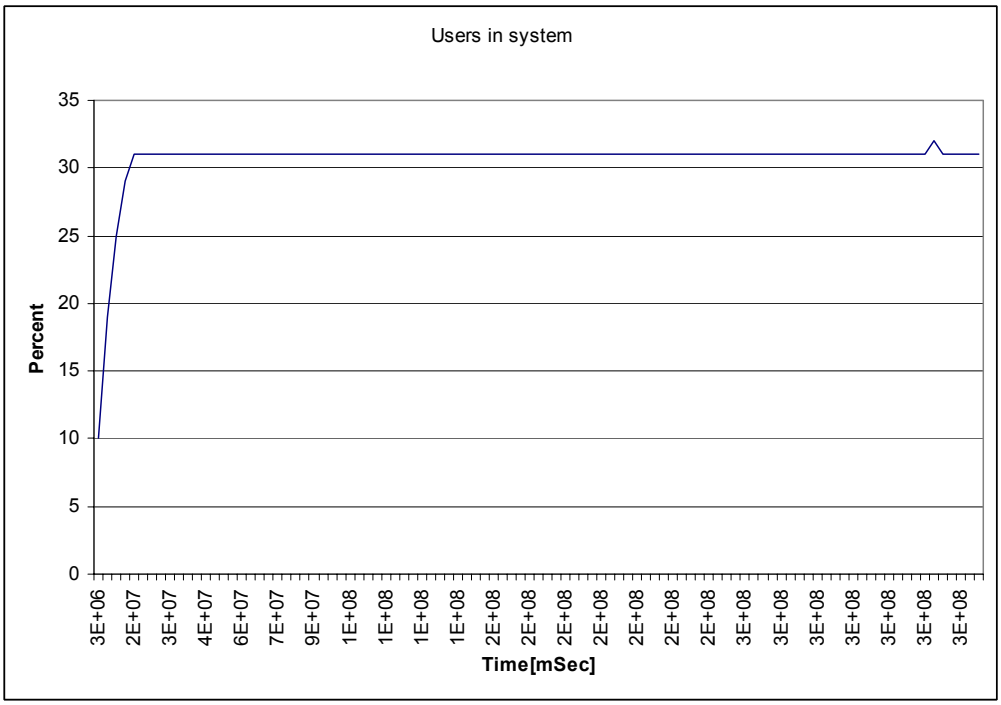


Figure 67 Users in system in scenario 8. applies to all locations except 1 and 4

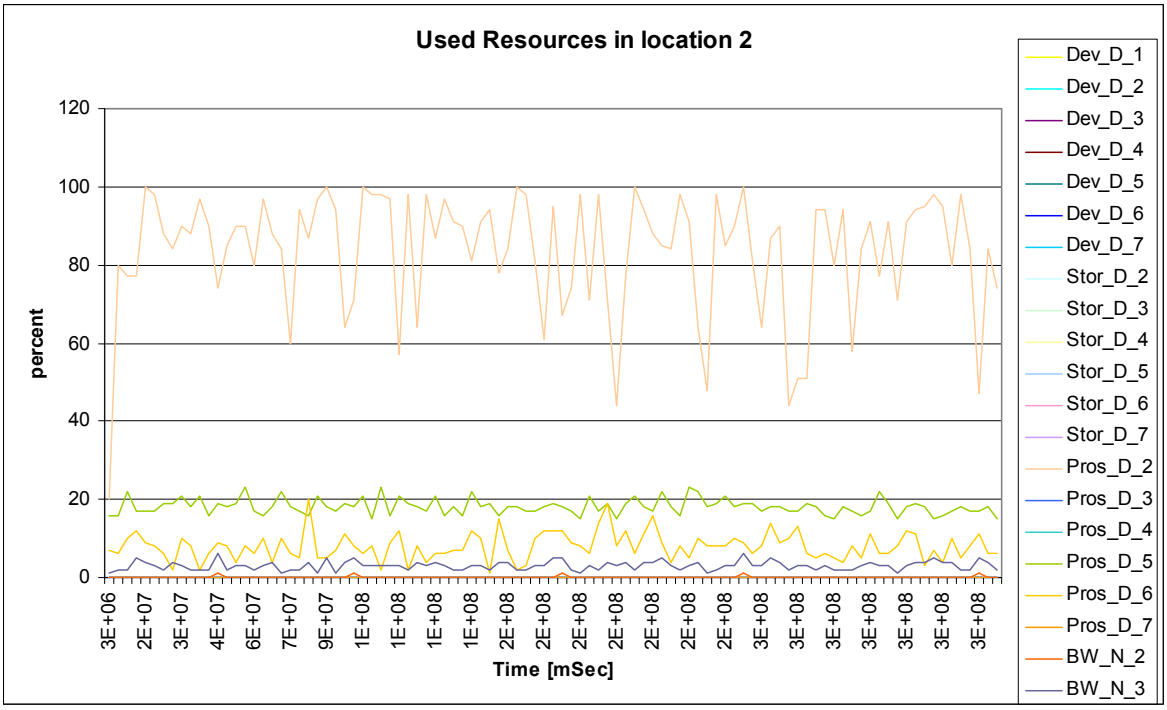


Figure 68 Used resources in scenario 8, location 2

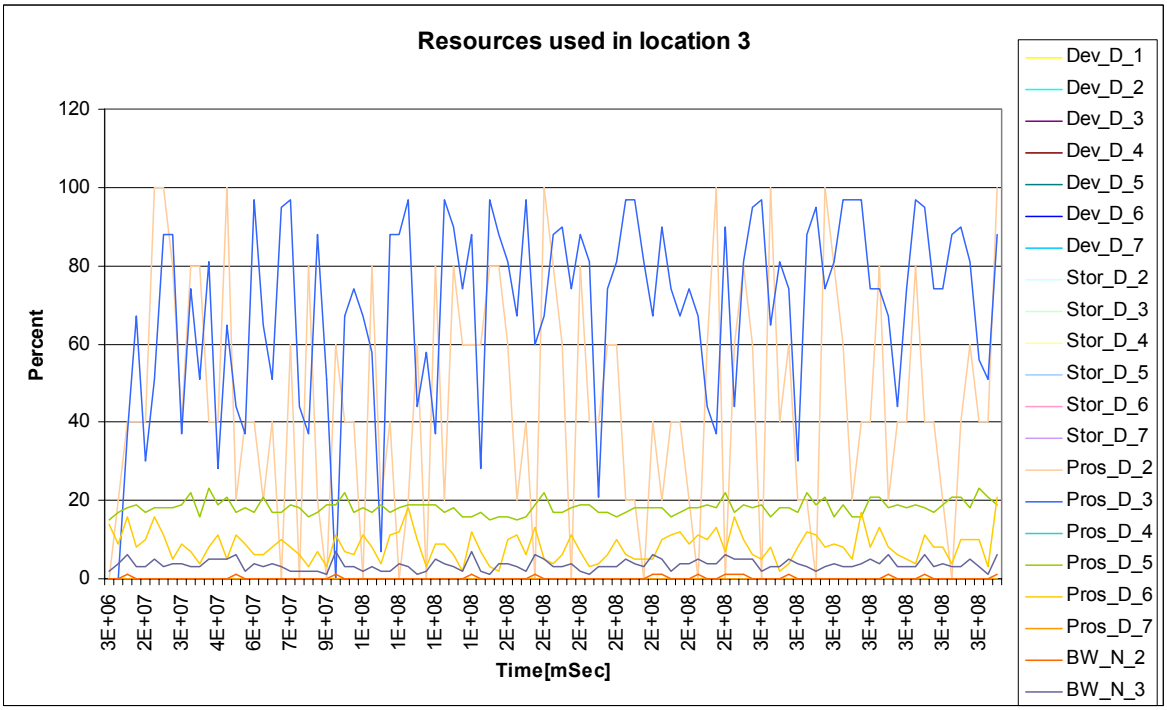


Figure 69 Used resources in scenario 8, location 3

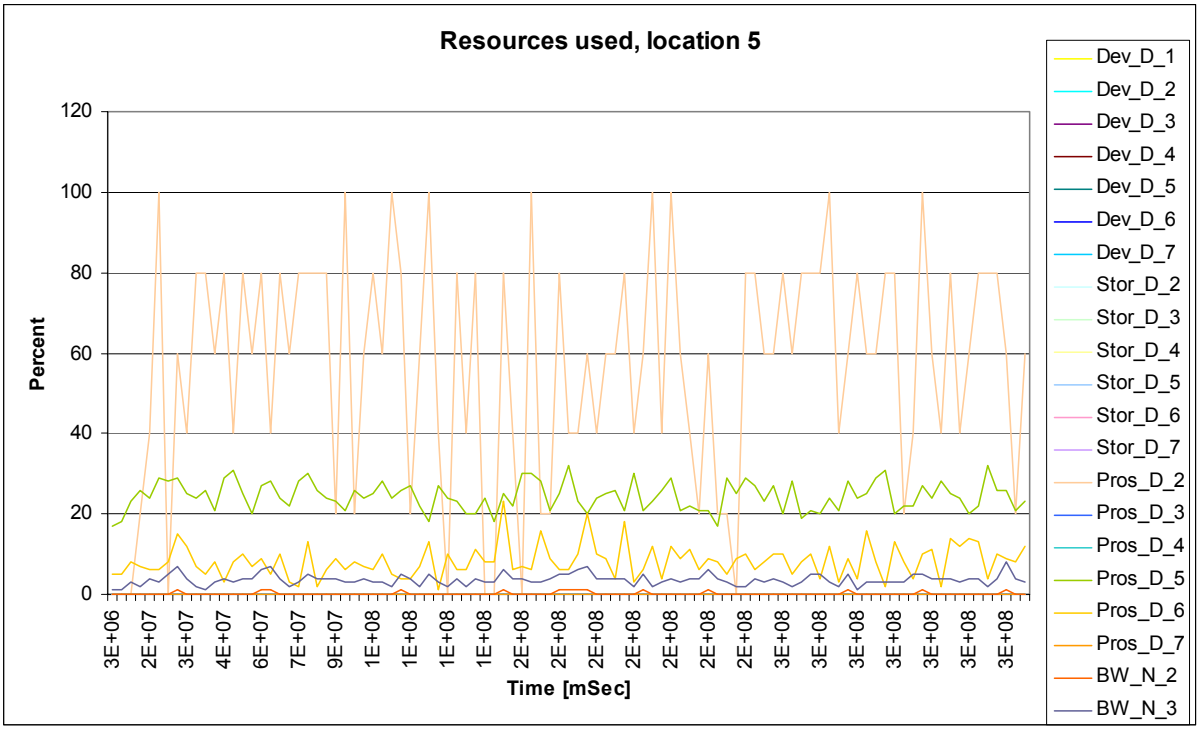


Figure 70 Used resources in scenario 8, location 5

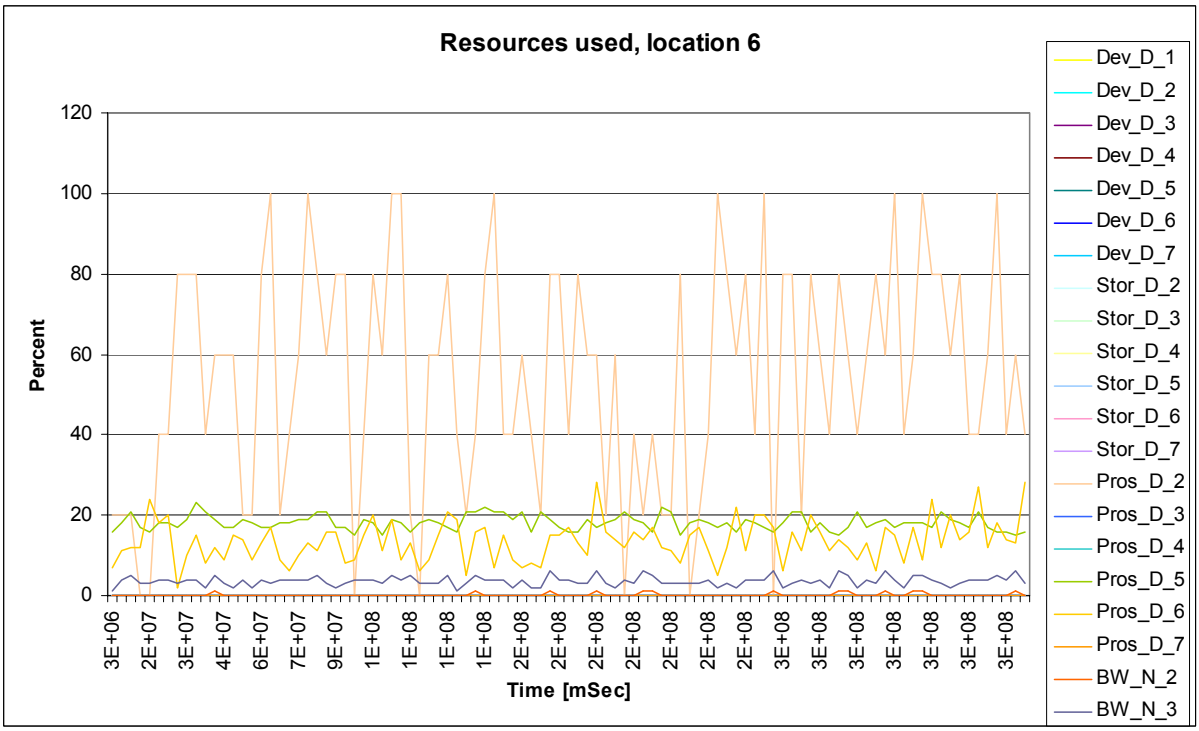


Figure 71 Used resources in scenario 8, location 6

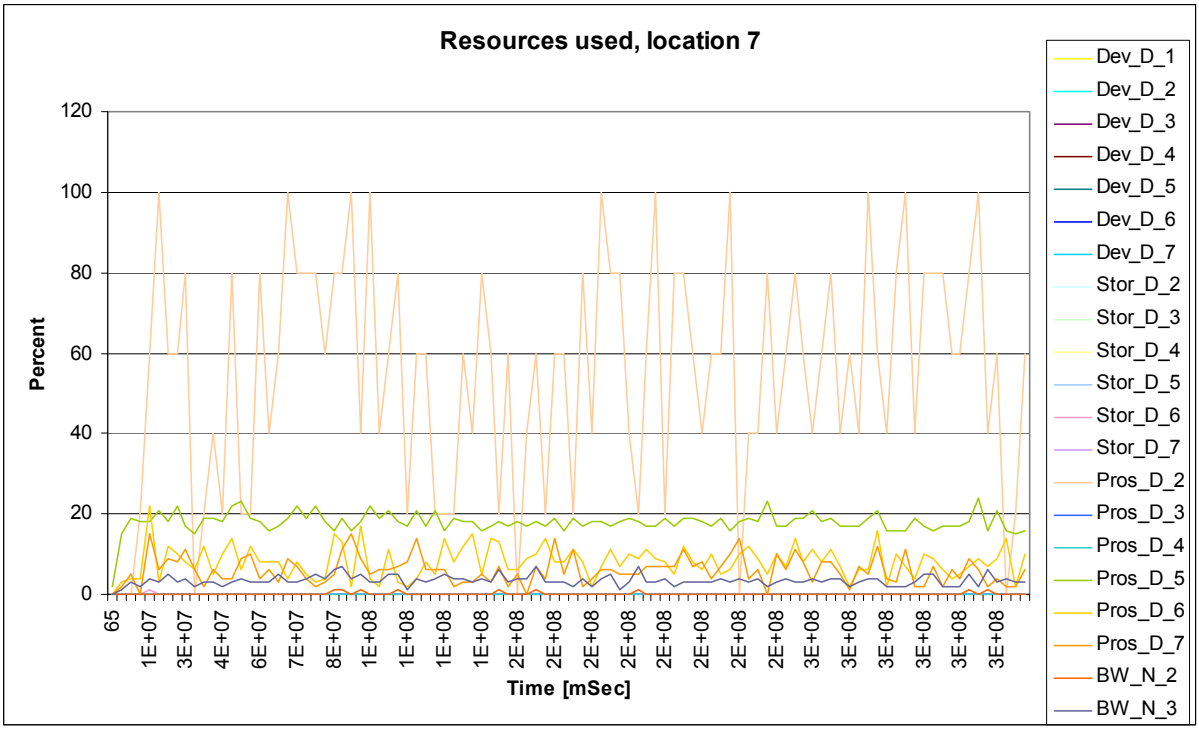


Figure 72 Used resources in scenario 8, location 7

VI.9 APPENDIX I: Misc.

VI.9.1 Average number of passengers per hour at Oslo Airport OSL

In 2005 courtesy of Torolf Holte, OSL.

Average number of passengers per hour at Oslo Airport OSL in 2005	
Hour	Passengers
0	256
1	121
2	111
3	128
4	56
5	27
6	560
7	2649
8	3516
9	2823
10	2292
11	2276
12	2201
13	2365
14	2332
15	3025
16	3706
17	3729
18	3289
19	3119
20	2536
21	2371
22	1733
23	1121
Sum	46342

Table 31 Average number of passengers per hour at OSL 2005

Bibliography/References

[1] **Beyond Moore's law: Internet growth trends**
Roberts, L.G.; "Computer" Volume 33, Issue 1, Jan. 2000 Page(s):117 – 119

[2] **Quality of Service Differentiation, Teletraffic Analysis and Network Layer Packet Redundancy in Optical Packet Switched Networks**
Harald Øverby, Doctoral thesis NTNU.
ISBN 82-471-7126-0

[3] **Adaptive offloading inference for delivering applications in pervasive computing environments**
Pervasive Computing and Communications, 2003. (PerCom 2003). Proceedings of the First IEEE International Conference, page(s): 107 - 114

[4] **The Spectra Project**, J. Flinn, S. Park, and M. Satyanarayanan. Balancing Performance, Energy, and Quality in Pervasive Computing. Proc. of 22nd IEEE International Conference on Distributed Computing Systems (ICDCS 2002), Vienna, Austria, July 2002.

[5] **The Puppeteer Project**, E. Lara, D. S. Wallach, and W. Zwaenepoel. Puppeteer: Component-based Adaptation for Mobile Computing. Proc. Of 3rd USENIX Symposium on Internet Technologies and Systems, March 2001.

[6] **Dynamic QoS-Aware Multimedia Service Configuration in Ubiquitous Computing Environments.** MONET Research Group, X. Gu and K. Nahrstedt. Proc. of 22nd IEEE International Conference on Distributed Computing Systems (ICDCS 2002), Vienna, Austria, July 2002.

[7] **The Coign Automatic Distributed Partitioning System.** The Coign Project, G. C. Hunt and M. L. Scott. Proc. of the 3rd USENIX Symposium on Operating System Design and Implementation (OSDI'99), February 1999.

[8] **The Implications of Pervasive Computing on Network Design**, R. Briscoe, BT Technology Journal archive
Volume 22 , Issue 3 (July 2004) Pages: 170 - 190
ISSN:1358-3948 R. Briscoe

[9] **Measured Performance of an Ethernet Local Network**, Communications of the ACM archive, Volume 23 , Issue 12 (December 1980), Pages: 711 – 721, ISSN:0001-0782, John F. Shoch and Jon A. Hupp, Xerox Palo Alto Research Center

[10] **Performance measures for a local network** ACM SIGMETRICS Performance Evaluation Review Volume 12 , Issue 2 Spring-Summer 1984, Pages: 34 – 37, ISSN:0163-5999, M. K. Rajaraman The University of Texas at Arlington, Arlington, TX

[11] **Garbage collector assisted memory offloading for memory-constrained devices** Chen, D.; Messer, A.; Milojicic, D.; Sandhya Dwarkadas;
Mobile Computing Systems and Applications, 2003. 9-10 Oct. 2003 Page(s):54 – 63

[12] **UML Distilled,** Second edition. Marin Fowler with Kendall Scott.
ISBN: 0-201-65783-X

[13] **DEMOS – a system of Discrete Event Modelling on Simula**.
Graham Birtwistle, DRAFT July 31, 1997

[14] **Kompendium TTM4110 Pålitelighet og ytelse med simulering** (Reliabilty and performance with simulation). 6. ed 2003. Peder J. Emstad, Poul E. Heegaard, Bjarne E. Helvik. Tapir akademisk forlag.

[15] **Accelerated stress testing to detect probabilistic software failures** Gullo, L.J.; Davis, R.J.; Reliability and Maintainability, 2004 Annual Symposium – RAMS 2004 Page(s):249 - 255 Digital Object Identifier 10.1109/RAMS.2004.1285456

[16] **Parametric analysis for adaptive computation offloading.** Conference on Programming Language Design and Implementation archive Proceedings of the ACM SIGPLAN 2004 conference on Programming language design and implementation, Pages: 119 – 130, ISBN:1-58113-807-5

[17] **Evaluation of performance scalability in pervasive systems**, Hans Inge Berg, NTNU, Department of Telematics Trondheim, Norway, Project Thesis 2005

[18] **Adaptive applications for mobile heterogenous devices** Inverardi, P.; Marinelli, G.; Mancinelli, F.; Distributed Computing Systems Workshops, 2002. Proceedings. 22nd International Conference on 2-5 July 2002 Page(s):410 - 413
Digital Object Identifier 10.1109/ICDCSW.2002.1030805

[19] **Systems Daidalos Deliverable D412**, Christoph Kuhmünch (Ed), Revised Architecture for Daidalos Pervasive, Siemens AG Version 0.08, March 2005.