Institutionen för kommunikation och information
Examensarbete i datalogi med inriktning mot nätverks- och systemadministration 15hp
C-nivå
Vårterminen 2011

Thin-Clients

An open-source product comparison

Daniel Hedegren
2011-06-05

Thin-clients

This report is submitted in partial fulfillment of the requirements for the Bachelor's degree in Computer Science at the Institution for Communication and Information. This project has been supervised by: Thomas Fischer

2011-06-05

All material in this report which is not my own work has been identified and no material is included for which a degree has previously been conferred.

Signed: _____

Thin-Clients

Daniel Hedegren

Abstract

This thesis presents a comparison among two different open-source company-solutions, taking into account the cost, performance and functionality. The compared open-source solutions are: LTSP and Openthinclient. The thesis also explains the normal client/server environment, fat- and thin-client environments and the concept of cloud computing.

Giving a final answer to which product is the best is a challenging task since they both have strong and weak sides. Depending on what the customer is after when looking to invest into an open-source alternative of thin-client solutions, the result will be different.

Keywords: open-source, thin-client, fat-client, cost, performance, monitoring

INDEX

1 Introduction

According to Tyson (2010), the name "thin-client" was originally a description of a computer terminal published by Oracle in 1993.

Now it is the clear definition of a system which has a client that accepts input from local peripherals only to delegate the data to a server. The server will then process that data and send it back to the client where the information will be displayed. One of the big problems with this type of system (henceforth known as a thin-client environment) is that the potential lack of high performance. Nowadays there are many different companies that contribute to the thin-client environment, but the problem is to find out which of these companies supplies you with an adequate solution.

One of the main reasons for a company to invest into thin-client environments is to achieve centralized administration. This type of administration is preferable since it is easier to manage than a decentralized environment. By having this kind of solution, and because the thin-clients are more robust than the fat-clients, the administrator does not have to visit the end-clients that frequently. Another reason for a company to invest into thin-client environments is to decrease the cost of their IT-environment so it is important that the actual cost (including initial cost and support) is as low as possible while still maintaining quality.

1.1 Motivation

When designing the infrastructure of a network it is important to take all the options into consideration. It is also important to make sure that the design has an adequate performance level while taking the cost into consideration so that it does not exceed the budget. One of the ways to find the correct balance is to use a thin-client environment, but how do we know which product solution to choose?

Distributors of any product have a tendency to always see themselves as having the best product on the market. It is important to review these products from an objective point of view. A technical comparison between several different thin-client solutions would be preferable.

1.2 Purpose

The purpose of this report is to analyze specific attributes between several thin-client environments distributed by different companies to make a detailed comparison. The attributes that will be compared are:

- Performance
    - Bandwidth utilization
    - Response-time
- Requirements of the server
    - Memory
    - Processor
    - Secondary Storage
- Costs
    - Initial costs

- Running costs
- Support costs

The reason using those parameters is due to the fact that if all of them are taken into account, it provides a holistic perspective when comparing different solutions.

The purpose of the comparison using the attributes listed above is to find out which of the selected products is best in the respective area. The product seen as the "best" here will be the product which has the overall better result from the measurement in combination with the cost.

1.3 Limitations

The main scope of this report is to compare several selected thin-client environments to give an objective view of the different products. The structure and cost of the solution will be taken into consideration whereas the companies providing these will not. The limitations are set on a technical level to the specific products and/or solutions. There is a limitation in the number of solutions that will be tested. Only two different solutions will be tested in this case[1]. There is also a limitation to the scale of the tests. The tests are conducted with a few computers in a laboratory setting without any real-life environment test which means that the performed tests will be estimations and approximations. A few scenarios are constructed to carry out the comparisons.

[1] The solutions that will be tested are LTSP- and Openthinclient-environments (see 2.4.2).

2 Background

This chapter explains the normal client/server environment, the different types of clients, which product solutions of thin-client environments that will be compared, the concept of cloud computing and the monitoring tool that is used.

2.1 Related work

There is quite a lot of related work related to thin-clients. There are, however, only a few that has almost the same purpose as this thesis. There is however a few that should be mentioned:

Rizwan (2011) presents a study carried out to upgrade and evaluating the performance of thin clients in scientific Server Based Computing (SBC).

Tykesson et.al. (1999) presents a study to provide guidelines for the choice of architecture in future client/server-applications.

Both of these have parts in them that cover many topics handled in this thesis, but it does not seem like their respective thesis performs the same type of comparison performed in this thesis. Rizwan (2011) checks the performance of thin-clients in general and Tykesson and Ericsson (1999) compare thin-clients to fat-client while this thesis compares different open-source thin-client solutions.

2.2 Client/Server Environment

The client/server environment is a well known, distributed computing model that commonly used worldwide. According to Reese (2000), in the client/server model, the clients are used to initiate requests of specific services that are placed on a server while servers provide these services to clients. The thin-client environment is a type of client/server configuration where the main difference compared to the standard solution of the client/server model is the type of clients. According to McKenna (2002), the different types of clients are fat-clients and thin-clients. The thin-clients utilize the servers' computation capacity instead of local computation, which means that the thin-clients basically only sends requests to the server and handles local resources such as display, keyboard and mouse. The fat-client also sends a request to the server like the thin-client does, but the requests sent from the fat-clients are requests to authenticate users and to access storage. The main difference between the thin-clients and the fat-clients are that the fat-client uses the local computation capacity.

2.3 Fat-Clients

According to McKenna (2000), a fat-client is a type of client where the local hardware is used for different types of user applications. Updates to this type of client will be managed locally by the client itself or pushed out through policies or other equal methods from directory-services. This means that a fat-client needs local administration in addition to administration through a server. Most of these problems can, however, be solved by using network bootable images, scripts and group-policies. How these solutions work will not be described in this thesis. When architecture is designed for a fat-client environment, it utilizes a centralized control that mainly consists of authentication and storage, while attributes such as updates, audio and video are handled by the local client by decentralized processing. This implies that the hardware requirements on fat-clients are much higher than the ones

on thin-clients, which also means an increase in the initial cost when buying fat-clients.

One of the big advantages compared to thin-client environments is that the fat-client would still work even if the network is not reachable. There would be a decrease in functionality if that would happen, such as: problems to access storage or home-folders, unable to authenticate to the directory-service or unable to reach the company database. Even with these problems, the fat-clients still have the option to work offline with their local user, the local storage and local configuration.

2.4 Thin-Clients

According to Cantiora (2010), a thin-client is a client that has a direct connection to the centralized server. The thin-client acts as a user interface between the remote user and the server. This means that the thin-client is a tool for user-interaction that forwards the users request to the centralized server, where the request is processed and then sent back. The thin-client is more or less a gateway between the user and the server to request information from a server and to display the result. The client itself does not do any operations that require hardware but instead delegates it to the server. This does put a larger constraint on the server but it also means that the client can work on minimal hardware. Since the thin-client does not need any particular hardware it is cheaper, simpler and more reliable than the fat-client. This means that the thin-client can be deployed in harsher environments.

2.4.1 History of Thin-Clients

According to LLC Books (2010), the first computers were used as an automated calculation-device. There was no real-time interaction as we know of it today, only a string of numbers as input and output. These computers slowly developed different input-methods like hole-punch cards, paper tape, magnetic tapes and eventually a kind of terminal that acted like a typewriter. The research continued which led to simple I/O (Input/Output)-devices which had a keyboard as the input-device and a standard CRT (Cathode Ray Tube)-display as the output-device. These devices only had one input-device and one output-device which when several users tried to interact with, led to the development of a centralized node, a mainframe, with "dumb" terminals acting as the input-devices.

According to Greenberg (2005), the development of computers were driven by the idea of personal computers (PC), a dedicated computers with local processing-power. With this idea in mind, the development of a Graphical User Interface (GUI) was developed. The concept of a GUI was originally developed by XEROX to get a better way for the user to interact with the computer. The GUI was however not suited for the dumb-clients since the requirements were too high. It was said to be impractical to distribute a GUI through the dumb-clients. The PC increased the productivity of the users drastically, but it lacked the reliability and security gained from the centralized mainframe, which led to the development of the client/server architecture.

According to Greenberg (2005), the thin-client is as follows:

"-Thin Client/Server Based Computing is defined by the fact that the application is executed on the server and displayed on the client system. Therefore, a "thin client" terminal need only have sufficient power to render the display of the user session.

This is the same type of processing as described earlier called a “Mainframe”. This means that one of the first computers were a version of thin-clients, only much simpler.

According to Tyson (2010), the name “thin-client” originates from a computer terminal published by Oracle in 1993, the development on the clients called “Sun-ray clients”. This is a computer which does not do any local computation, but instead sends the information to a centralized server where the operation will be performed.

2.4.2 Product solutions

There are many different products/solutions available regarding thin-client environments. Choosing which one of these to pick might be hard because the companies distributing these might not give a fair image as to how good or bad their respective solution is.

In order to narrow the selection down, this thesis will only compare two different product solutions. In this case, the solutions that will be compared are Openthinclient and LTSP (Linux Terminal Server Project). While there are many available product-solutions for thin-client environments, both commercial and open-source, the chosen products are both open-source products. The reason why two open-source products were chosen instead of more commercial products like Sun-ray and Citrix is due to the fact that their solutions have a high cost, and without proper funds, open-source is the only alternative.

2.4.3 Compared products

The explanation of product-solutions is fetched directly through the given information of the products from the respective company distributing them. There are many available options when choosing among the thin-client products. However, most of them do not have available support other than forums and mail. Due to this, the chosen products are LTSP and Openthinclient.

LTSP: Linux Terminal Server Project

The Linux Terminal Server Project is an architecture that provides a thin-client solution for Linux. LTSP themselves provides support through forums but there is available support from http://disklessworkstations.com. The same place also has available hardware for sale. LTSP sends all the information between the client and server through an SSH-tunnel which makes the transactions secure. LTSP is licensed under GNU GPL v2 which makes the solution completely free to download.

Openthinclient

Openthinclient is a free thin-client solution based on Linux and Java that is intended for medium to large environments. The solution is available for both Windows and Linux. Both versions require a directory service such as LDAP (Lightweight Directory Access Protocol) or Active Directory (AD) to function and since they are fully supported, this solution can easily integrate with an existing solution. Openthinclient is licensed under GNU GPL v2 which makes the solution completely free to download. There are options for hardware-appliances and support of different levels.

2.4.4 Cloud Computing

According to Sobotta (2009), the definition of cloud computing is: "*The outsourcing of IT hardware and software to centralized external third parties capable of providing IT capabilities as a service, leveraging the internet [the cloud] and accessed via (potentially thin) clients that know not, nor care not, how the services are implemented, maintained or what infrastructure is used to support it.*"

According to Sobotta (2009), cloud computing is a concept that has existed for decades. In recent years, the concept of outsourcing has gotten much attention due to an increase of distributed services which also has put a large focus on cloud computing. The concept can be broken down into five smaller segments.

1. Applications

   In recent years, the adaptation of cloud computing has increased drastically, distributing many kinds of services through "the cloud". Since they are distributing services, meaning software, this is commonly known as SaaS (Software as a Service). Examples of these are: Dropbox, Google Mail, Google docs.

2. Clients

   According to Sobotta (2009), a cloud client is a computer that relies on "the cloud" for delivery of applications and/or services. Like the thin-client environment, if the network goes down – so does the cloud client.

3. Platforms

   Commonly known as PaaS (Platform as a Service). When referring to a platform in cloud computing, you are referring to a platform placed in "the cloud" that is distributing platforms and application to clients. Since the platform is distributed as well as it distributes applications, you could say that the PaaS is also distributing Software as a Service.

4. Storage

   When discussing storage in cloud computing, you are discussing normal storage placed in "the cloud". An example of this is Dropbox.

5. Infrastructure

   In cloud computing, it is also possible to distribute entire infrastructures. According to Sobotta (2009), this is commonly done by virtualization. This is commonly known as IaaS (Infrastructure as a Service).

There are many similarities between cloud computing and thin-client environments. Both of them use a form of distributed infrastructure building on centralized management, both have a form of clients that is dependent on a functioning network and both of these distribute services from the central node.

2.5 Host-based monitoring

In order to get a value on all of the wanted attributes that might be needed for the performance-tests, a monitoring tool must be used. The monitoring tools are generally

used to measure the load of a network or to monitor the health and integrity of different services to ensure that the network/service does not get overloaded in any way, or to let the administrator know if a service has gone down. This is especially useful in enterprise-environments since the server-parks tend to get quite large. Monitoring tools can also be used to see if a company lives up to the eventual Service Level Agreements (SLA). The SLA is an agreement between the company and customer where values are set on attributes, like availability on a service per year, which the company must uphold.

The monitoring tool that will be used in this case is called Zabbix, which is an open-source distributed enterprise solution that can be used for monitoring just about anything on a specific host or network. Most of the measurements are completed by using that which is called an "Agent[2]". This is a program that has a direct connection with the Zabbix-server. These agents collect the information locally by running different commands and send the results back to the server.

The Zabbix-server automatically creates graphs for every monitored object which will make it easy to monitor all the attributes needed to conduct the tests (see 4.3). For information on the installation and configuration of Zabbix, see 5.4.

[2] For more information about the components in Zabbix, please see the following link: http://www.zabbix.com/documentation/1.8/manual/installation/components

3 Problem

This chapter describes an introduction to the problem, purpose, motivation and goals of the report.

The main question of this thesis is as follows:

- **When comparing different product solutions of thin-client environments it is important to choose a solution based on performance, functionality and cost. In this case, two open-source products are chosen for the comparison but which one is considered the "best" based on the given attributes?**

3.1 Purpose

The purpose of this thesis is to analyze the thin-client environments called LTSP and Openthinclient to compare their respective solutions. This comparison will be performed by considering the different attributes shown in section 4.3.3. The final outcome is a detailed list of the solutions considered and the metrics chosen. This list supports technicians while selecting thin-client solutions.

3.2 Motivation

The main problem when designing the architecture for a new network is to sort out different product-solutions from each other. This is a problem because companies that distribute the different solutions do not provide an objective point of view; they will always put their product in front of the others. Thus, the main contribution of the thesis is to provide an objective comparison between specific product-solutions, to make it easier to know which thin-clients solution to choose.

3.3 Partial-goals

To ensure that this comparison will be as detailed as possible and as focused as it needs to be for a designer to be able to make important decisions, this report will be structured with four focused partial-goals. The following goals will be guidelines to the entire thesis.

1. Performance[3] of the product-solutions
2. Configuration options for the solution
3. Cost of the entire solution including support
4. Create scenarios to calculate the following attributes:
    - Performance
    - Cost
    - Background
    - Requirements

By following these goals, the comparison focuses on technical aspects but will also provide a comprehensive overview of the whole solution.

[3] For more information about the tests or the measured attributes, see section 4.3.

4 Method

This thesis is conducted by one person alone, so to ensure that the comparison is done in an objective manner, it is important to describe what knowledge he or she has.

My computer knowledge consists of daily configuration and use of the standard PC (both Windows and Linux) and home-router for about 15 years. I have also conducted the three-year education program on Högskolan i Skövde, called "Nätverks- och Systemadministration 180HP". This program focuses on administering and configuring network-devices and servers, but did not involve thin-client environments. Since the person conducting this research has not been involved in thin-client environments before it is highly likely that the comparison will be objective.

4.1 Implementation or Experimentation

The purpose of this thesis is to compare different solutions and present an objective comparison. With that in mind, the only methods that seemed relevant were either implementation or experiment. Due to that reason, the mentioned methods were chosen as potential candidates for this thesis.

According to Berndtsson et al. (2008), implementation is a method used for implementing a certain solution to demonstrate that the solution has the proposed properties. This method is often used to compare several implementations of existing solutions. When this method is used as a research method it is important to make sure the report is valid and reliable. To ensure the validity of the implementation, you work according to a proposed solution which in this case is the solutions given by the thin-client distributors. If the implementation will be handled incorrectly – the validity of the project will be compromised. To ensure reliability, one has to implement the solution in a way that it works well enough to be used in the research Berndtsson et al., 2008).

According to Berndtsson et al. (2008), experiment is a method used to verify or falsify a stated hypothesis. This method is often used to try and prove that a hypothesis is correct even though it cannot be said that because the hypothesis is true, the statement is proven. An experiment can be conducted from many different ways so even if a person proves a hypothesis, it does not mean that the experiment is fair.

Seeing as the content of this thesis is about comparing several different solutions, the chosen method is Implementation. This method seems the most systematically suited with regard to the purpose of this thesis.

4.2 Gather intelligence and constructing the environment

The first step in this project is to gather knowledge about the respective solutions, which in this case are Openthinclient and LTSP. This is completed by reading official guides from their respective homepages and/or reading forums if complications occur. This information is crucial when designing the structure for the tested solutions. With the given information, a topology[4] was created to be used as a template when constructing the environment and is also used to get the same measurement for the

[4] A Network-infrastructure map showing all the devices in the solution

price-list (see 4.4). Each step regarding the construction of the environment is documented in chapter 5 and the topology is shown in appendix A.

In order to ensure the validity and reliability (see 4.1), the configuration is based on the distributors own guidelines for best-practice, if such exists, and their own documented solutions.

4.3 Conducting Performance tests

After the product-solutions have been setup, a series of performance tests are conducted on the environment. The performance tests are conducted from the thin-clients in the environment and are based on "response-time" and "bandwidth utilization". Performance tests on the server regarding "memory-usage", "processor-load", "network-load" and "hard-drive space" are also carried out.

To conduct these tests, several different methods are used (these methods are explained below). In order to measure the results of the tests, the monitoring tool called Zabbix is used. In order to get a baseline to be used as a reference-point, this tool monitors the server with the mentioned attributes even before there actually is any traffic on the network. The monitoring will continue while the tests occur to get a reading as to how these tests affect the different systems. Zabbix will then display the results including history so it will result in a graph showing how the systems react to each of the tests. Since Zabbix contributes with a complete view of both history and current information displayed in graphs, this will be the main source where the performance are compared from.

The following references are taken from appendix A. The tests are conducted from Thin-client1 to Thin-client Server while monitoring from the Zabbix-server. While the tests are active, a ping-request is set up from Thin-client2 to Thin-client Server to measure the latency and eventual packet-loss of the packages.

4.3.1 Response-time

Since response-time is usually measured by sending a package from a source to a destination to measure the time it took, this test is performed by monitoring hosts with Zabbix while creating a bash script that uses the tool called "packgen". This tool can be used to simulate traffic based on what the configuration options inputted. The tool can also be used to increase or decrease the MTU, change the packet type, perform UDP-flood attacks and so on. The scripts will be separated based on UDP and TCP configurations to simulate different types of traffic and will be used from one of the clients. The reason why packgen was chosen over the other available alternatives[5] is due to discussions and suggestions from different forums where other users have been in contact with the tool. Packgen has been used in other similar experiments, i.e., see Chiorean (2009). Packgen is a simple tool that allows the users to conduct tests on the network bandwidth and Quality of Service (QoS).

While packgen is used in the script to simulate multiple users, it is important to note that it does not give a fair view of a real user. A user does not act statically during a certain amount of time but acts randomly. If the script would use a random-timer, however, the comparison between the products would not give a fair view.

[5] Other available tools are, for example, GASP, Packet Excalibur, PacketGenerator, pktgen.

While this script runs in the background, all the available client are used for normal desktop tests like trying to edit files and saving large files on the hard-drive. While these tests are performed, a standard ping-request is conducted from one of the clients to measure if there are any packet-losses. Since response-time is usually measured by sending a package from a source to a destination to measure the time it took, this test is performed by monitoring hosts with Zabbix while creating several bash scripts that uses the tool called "packgen". This tool can be used to simulate traffic based on what the configuration options set. The tool can also be used to increase or decrease the MTU, change the packet type, perform UDP-flood attacks and so on. The scripts will be separated based on UDP and TCP configurations to simulate different types of traffic and will be used from one of the clients. While this script runs in the background, all the available client, will be used for normal desktop tests like trying to edit files and saving large files on the hard-drive. While these tests are performed, a standard ping-request is conducted from one of the clients to measure if there are any packet-losses.

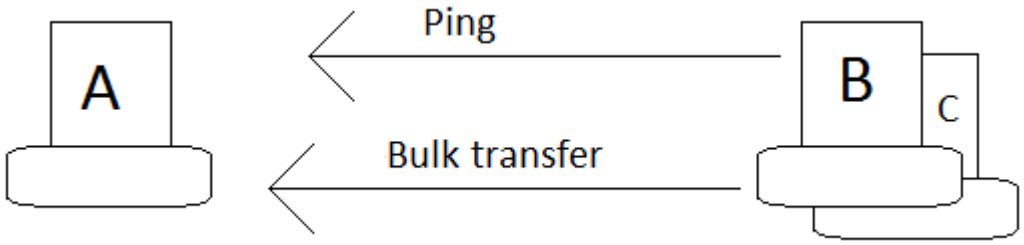

Figure 1: Response-time measurements

The ping-request is sent from computer B to computer A while the bulk transfer is conducted from computer C.

4.3.2 Bandwidth utilization:

This test is conducted by configuring the scripts (see 5.5) with a different number of simulated users so that it will send a lot more packets than before. This produces a higher load on the network which should result in latency or in worst case; packet-loss. The measurements are monitored with Zabbix while trying to connect to the server with another client to see if a connection can be made while the load on the network is high. It is important to note that it might not be the server that acts as the bottleneck here; it might be the network device. For that reason, Zabbix will be used to monitor all the devices in the network, including the network device (with the help of SNMP) to make sure which device it is that fails first.

4.3.3 Memory-usage, processor-load, network-load and hard-drive space

To ensure that the clients and servers will perform as they should, monitoring will be used on all of the nodes to measure their performance. This will be handled by measuring standard values that might affect the overall performance of a computer, such as memory-usage, processor-load, network-load and hard-drive space. The items that will be measured from these attributes are:

- Memory usage
    - Cached memory

        - To store information into a cached memory is to store the information in a place that has faster access time than the normal physical memory
    - Shared memory
        - Typically, it refers to a large block of memory that can be accessed by several CPU:s simultaneously in a multi-CPU computer
    - Free memory
- Processor load
    - Average processor load, one minute
    - Average processor load, five minutes
    - Average processor load, fifteen minutes
- Network load
    - Incoming traffic on the network interface
- Hard-drive space
    - Size of remaining space of the partition
        - The remaining size of the partition containing the operating system
    - Free disk space on /home
        - This purpose of this measurement is to see if the users home-folders are affected by the thin-client solutions
    - Free swap space
        - The swap space is a place where information can be stored temporarily to allow a computer to read and write the data into the physical memory faster than normal on a hard-drive

All of the compared solutions are monitored this way with Zabbix. The graphs extracted from these attributes are shown in appendix B-E.

4.4 Comparing prices

In order to give a correct comparison between different product-solutions it is important to take the costs into consideration. The different types of costs that are compared are: “initial cost”, “running cost” and “support cost”. For more information, see *Table 1: Thin-client Costs*. To ensure that the prices are correct, contact is made with the respective customer-contact support where the price-list will be requested. The price-lists will vary depending on the actual size of the network so all of these price-requests are based on the topology design created for the construction of the network (see 4.2).

Table 1: Thin-client costs

Type:	Definition:
Initial cost	After the design of the network topology has been completed, the required hardware needs to be purchased. It can also involve

	purchasing software and support. This is what is called the *initial cost*.
Running cost	*Running cost* is the cost of the electricity when the solution has been implemented and everything is up and running. This also includes replacement of hardware and maintenance.
Support cost	The available support for a solution may vary but there is usually an option for support where you pay a monthly or yearly fee to the distributors. Some of the open-source alternatives to thin-client environments do not require a purchase of support in order for the environment to work.

4.5 Constructing scenarios

To put the tests into practice, a few imaginary scenarios are created. These scenarios are based on the results from Zabbix and from own experiences gained from the testing. The graphs can be used to scale up the tests to calculate how well these tests would be suited in real-life situations. This is done to ensure the validity of the thesis, and to provide a larger view of the solutions based on the measured results.

The created scenarios is divided into three different levels; small, medium and enterprise. The purpose of dividing the scenarios into three levels is to create scenarios that might appeal a company disregarding of their size.

5 Implementation

This chapter explains the specifications of the environment and a step-by-step guide how the installation and configuration of the different product-solutions is completed.

5.1 Server specification

The server that is used in this configuration has the following specifications:

Table 2: Server hardware specification

Hardware	
CPU	INTEL Core i5 3.3GHZ
Motherboard	MSI P67A-GD55
Memory	2*Corsair 4GB DDR3 XMS3
Networking	Ethernet 10/100 Mbit/s
Audio	Realtek ALC892 7.1

5.2 Network configuration

To ensure that the solutions are constructed in the same way, the network-configuration is constructed as shown in the topology (appendix A). The switch from the topology is a Netgear RP614 100Mbps-switch.

5.3 Operating system

Due to the fact that the installation guides might not have a recommended operating system, a chosen operating system will act as the backup to be used if that would be the case. The chosen operating system is in this case Ubuntu Server 10.04.2 LTS due to the fact that the LTS version was developed for commercial and professional use and due to positive personal experience with Ubuntu earlier.

The installation of Ubuntu is in this case performed by downloading the image-file (.iso) from the webpage. The image-file is then mounted by an image-reading tool like Daemon-tools. This tool is designed for reading any kind of image-file to render the images to a simulated CD or DVD-drive. When browsing the simulated CD or DVD-drive, the contents are those of a normal installation disk from Ubuntu. The contents of this image can be copied to a bootable USB-drive or burned to a CD or DVD.

The installation of Ubuntu itself is conducted with standard parameters on partitions, language and extra packages to install. The hardware used with this installation is shown in Table 2.

5.4 Installing Zabbix

To make sure the installation of Zabbix is handled the correct way, a guide from the official homepage is followed. All the installation-options are made from this guide.

The first step from this guide is to make sure all the prerequisites are fulfilled so that the installation can proceed. The requirements are “mysql”, “snmp”, “php5”, “apache2” and several other packaged that can be by the “apt-get” command. The

installation and configuration of these requirements are not covered by this thesis. For more information about the requirements and installation-instructions, please visit http://www.zabbix.com/wiki/howto/install/ubuntu/ubuntuinstall.

The installation of Zabbix starts from scratch, creating users and compiling the program. Below follows information as to how the installation was conducted. For more details regarding the installation, please follow the link above.

The installation of Zabbix can be broken down into three parts:

- Creating the table and migrating the database
- Compilation of the program
- Creating an alias in Apache

The mentioned guide provides a link for this but that link point to an old version of Zabbix. To make sure that the tests are conducted properly, the latest stable version of Zabbix is to be used instead.

When creating the table in the MySQL database, it is important to set the correct rights on the table so Zabbix can access it. To create the database and set the correct users on the database, the following two commands were used.

create database zabbix;

grant all privileges on zabbix. to zabbix@localhost identified by 'password';*

The existing Zabbix database is then migrated to this table in the MySQL database.

After the database had been migrated, the next step was to compile the package and install it. A compilation of a program is usually done with scripts that come with the source-code that support additional arguments to adapt the installation. The following commands were used:

```
./configure --prefix=/usr --with-mysql --with-snmp --enable-server --enable-agent && make

sudo make install
```

In order to Zabbix get a connection to the MySQL database created earlier, the username and password to the database must be configured in the Zabbix-server configuration file. This file is placed at */etc/zabbix/zabbix_server.conf*. In that file, the following rows were changed:

```
DBUser=zabbix
DBPassword=password
```

To be able to access Zabbix from a browser an alias needs to be created in Apache. The alias is used to point a certain request through a browser, to a certain folder. The following was added to the Apache configuration file.

```
Alias /zabbix /home/zabbix/public_html/
<Directory /home/zabbix/public_html>
  AllowOverride FileInfo AuthConfig Limit Indexes
  Options MultiViews Indexes SymLinksIfOwnerMatch IncludesNoExec
  <Limit GET POST OPTIONS PROPFIND>
    Order allow,deny
    Allow from all
  </Limit>
  <LimitExcept GET POST OPTIONS PROPFIND>
    Order deny,allow
    Deny from all
```

```
  </LimitExcept>
</Directory>
```

There are a few adjustments that need to be configured to Apache, but these configurations are dependent on the current configurations of Apache and are therefore not mentioned in this thesis.

5.4.1 Zabbix agents

As mentioned earlier, Zabbix requires agents to be installed on each client that should be monitored. Installing these agents vary depending on the operating system. In this case, the agents need to be installed on Linux and on Windows.

Installing an agent on Linux is an easy task if you have a version of Linux that supports the program "apt-get". By using this command, you install the agent with all the needed dependencies. The full command is as follows:

```
apt-get install zabbix-agent
```

The agent also needs to be configured to point at the Zabbix-server so the system knows where to send the information. This configuration is done by editing the server IP-address in the following file: /etc/zabbix/zabbix_agentd.conf.

Installing an agent on Windows can be done by using a pre-compiled installation package available on the Internet, but also by downloading the source-file and compiling the package manually. In this case, the first option was chosen. During the installation, the user is prompted by the IP of the Zabbix-server.

5.4.2 Configuring Zabbix

Zabbix is constructed with three major components for configuration: "host-groups", "templates" and "hosts". A host is a normal device on the network that can be monitored by Zabbix using its agents while a host-group is a collection of hosts. The templates are used to have certain types of monitoring on different types of systems. These components should, according to the user manual at the official Zabbix homepage, be used as follows:

The hosts should be named aptly and should preferably be named in Zabbix with their DNS-name. They should then be placed into an appropriate host-group that is suited for other hosts similar to that one. For example: a Linux host should be placed in a Linux host-group and a Windows host should be in a separate Windows group. After the hosts have been placed into appropriate groups, the templates should be used. There are many preconfigured templates arriving as part of the standard installation of Zabbix that can be used for different systems. These templates can be applied to separate hosts, or to host-groups. Best-practice is to always use templates on host-groups due to the fact that if a change needs to be configured, you only have to configure the change once on the template, and the change will occur to all the hosts of the host-group.

In this case, the templates have all the objects that need to be monitored in order to measure the performance of servers and clients of the thin-client solutions. The item that needs to be configured is the agents, the hosts and the host-groups.

5.5 Script configuration

As mentioned before, to measure the performance of the devices the tool called "packgen" is used. This tool needs to be downloaded and installed manually since the

standard Ubuntu installation does not have it installed. Since packgen is a tool based on the programming language called “Ruby”, this also needs to be installed. The following commands were used to download packgen and Ruby.

```
wget http://rubyforge.org/frs/download.php/4556/packgen-0.2.tar.bz2
apt-get install ruby
```

The file containing the tool is a compressed file that needs to be unpacked, which can be done with the following command.

```
tar -xvvf packgen-0.2.tar.bz2
```

To install packgen, enter the unpacked folder and compile the tool using the ruby-script called “setup.rb”. This can be done with the following command.

```
ruby setup.rb
```

This tool comes preinstalled with scripts that send packages of different types and listens for the response. The preinstalled scripts can be configured to increase or decrease packet-size, bandwidth and also change the packet-type. This tool needs to be installed on each client and server since the packages that is sent, is sent to a destination-port that needs to listen for the traffic. Installing this tool on Windows however in not needed. By configuring the script to use the port 53(Default DNS-port), the packages will not be blocked. There is also a possibility when using this tool to log the packages into a log-file to see if they were sent correctly. The command to use the tool to send packages, with logging, is as follows:

```
packgen -i send.yml -l logfile.log
```

The file called “send.yml” is used as a configuration file with arguments defining the packages. The content of the “send.yml” file is as follows:

```
SEND:
  udp:
   -
     name: Voice
     host: 192.168.186.111:5000
     bandwidth: 700Kb
     packet_size: 512B
     dscp: cs6
     from..to: !ruby/range 0.0..60.0
   -
     name: Video
     host: 192.168.186.111:5001
     bandwidth: 2.8Mb
     packet_size: 750B
     dscp: cs4
     from..to: !ruby/range 10.0..60.0
   -
  tcp:
   -
     name: Best Effort
     host: 192.168.186.111:5002
     bandwidth: 3.2Mb
     packet_size: 1KB
     from..to: !ruby/range 20.0..60.0
   -
     name: Video
     host: 192.168.186.111:5003
     bandwidth: 3.2Mb
```

```
  packet_size: 1KB
  dscp: cs1
  from..to: !ruby/range 30.0..60.0
```

The destination IP and port to send the packages is configured under the attribute called “host”. The attributes that can be altered in order to gain different results are the rows with the “bandwidth” and “packet_size”. In order for the sending of packets to function when sending to computer with Linux, the receiver must be configured to listen on the same ports that are defined in “send.yml”. The ports to listen too can be defined in the “listen.yml” file:

```
LISTEN:
  udp:
   -
     ports:  !ruby/range 5000..5004
  tcp:
   -
     ports:  !ruby/range 5000..5004
```

The command to use when wanting to listen for incoming packets is:

```
packgen -i listen.yml
```

In order to simulate several users and to be able to scale up the tests, a bash-script is created to send multiple instances of “send.yml” at the same time.

The bash-script is configured as follows:

```
#!/bin/bash
while (true)

do
packgen -i send.yml -l logfile1.log &
packgen -i send.yml -l logfile2.log &
packgen -i send.yml -l logfile3.log &
packgen -i send.yml -l logfile4.log &
packgen -i send.yml -l logfile5.log &

wc -l logfile1.log
wc -l logfile2.log
wc -l logfile3.log
wc -l logfile4.log
wc -l logfile5.log

sleep 10s
done
```

The script is configured to first send packets through five different sessions, which simulates five users, and write the results into the five different log-files. Since the log-files remain empty unless the packets were successfully sent, the remaining rows are used to count the number of rows in the logs to check for potential errors. This specific bash-script simulates five users but by adding additional rows of the “packgen -i send.yml” command, more users will be simulated. The whole script is encapsulated in a while-loop that sends all these packages every tenth second.

5.6 Installation of LTSP

The installation of the LTSP server has a set of different guides depending on which operating system the user prefers, which in this case is Ubuntu. The installation of the

LTSP server is an easy process if Ubuntu is chosen due to the fact that LTSP has an own installation-mode on the Ubuntu alternate image.

To install the LTSP server, the alternate Ubuntu image needs to be configured to be bootable from a USB-memory (see 5.3) and then inserted into to the computer. When the image has booted and the user is prompted with the menu to choose what to do, there is a small row of text at the bottom of the screen where there are additional options for the installation. By pressing "F4" during this time, the user is prompted with additional modes whereas one of these modes is the installation of an LTSP server. After this mode has been chosen, the rest of the installation is conducted like a standard Ubuntu installation.

It is important that the LTSP server has DHCP (Dynamic Host Configuration Protocol), DNS (Domain Name System) and TFTP (Trivial File Transfer Protocol) installed. These services are needed in order for the server to function. DHCP is used to set IP-addresses of connecting hosts, DNS is used to point names to IP-addresses and TFTP is used for "pxe boot" or transferring files. While DNS and DHCP are required to be installed, TFTP is in this case considered a "best-practice". Due to this, DHCP and DNS were installed during the installation of Ubuntu while TFTP was considered unnecessary.

5.6.1 Configuration

After the LTSP server had been installed, it still needed configuration to work. A static network interface needs to be set up where the thin-clients should be attached. To do this, additional packages needed to be installed and the thin-client environment needs to be created. The following two commands were used:

```
sudo apt-get install ltsp-server-standalone openssh-server
sudo ltsp-build-client
```

After these packages were installed, the LTSP-system needs to be updated with the ssh-key, which is done with the following command:

```
sudo ltsp-update-sshkeys
```

All the available configuration options regarding the clients and other administrative tasks are conducted through the program called "ltspadmin".

The clients need to be added into the LTSP-tree to be allowed to contact the server, which can be configured in ltspadmin. In order for the clients to be authenticated, their DSA-key needs to be copied into the servers LTSP-tree. After the key has been placed there, the clients can start to use the LTSP tunnel.

This concludes the configuration of LTSP on the server; however, the Zabbix agent still needs to be installed. For more information regarding the installation of the agent, see section 5.4.1.

5.6.2 Performance

The first step to measuring the baseline of this configuration is to add the hosts into Zabbix and monitor the hosts for a couple of hours without any generated load or traffic to see how the system behaves normally. For more information about the other measured attributes, see appendix B. The baseline of one hour has shown the following CPU-Load:

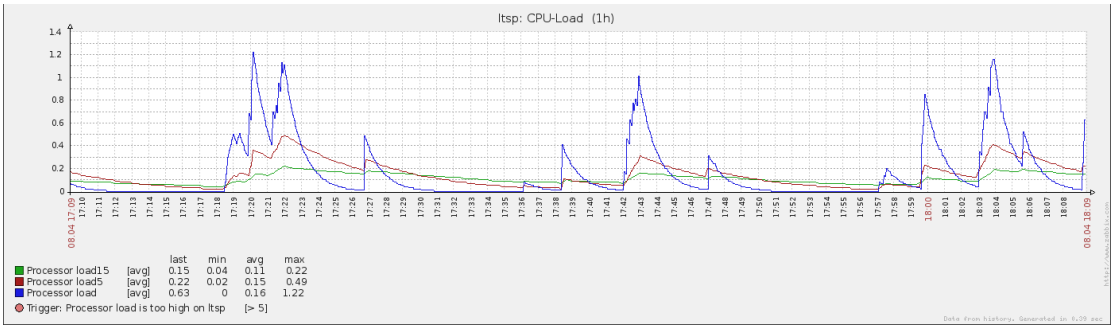

Figure 2: LTSP-server baseline CPU performance

The graph shows the processor load for one minute (displayed in blue), processor load over 5 minutes (displayed in red) and processor load over 15 minutes (displayed in green). The horizontal axis displays time in minutes while the vertical axis shows the load of the processor.

The baseline measurements of LTSP shows that once every 15-20 minutes there seems to be some kind of background process which produces load on the processor. It is important to have that in mind when comparing this figure with the results from the performance tests.

The first performance measurement of the LTSP-solution is conducted by using the bash script to simulate five users with the normal size and bandwidth configured in the script (see 5.5). For information about all the monitored attributes, see appendix C. For detailed information regarding the tests (see 4.4). This test has shown the following CPU-Load:

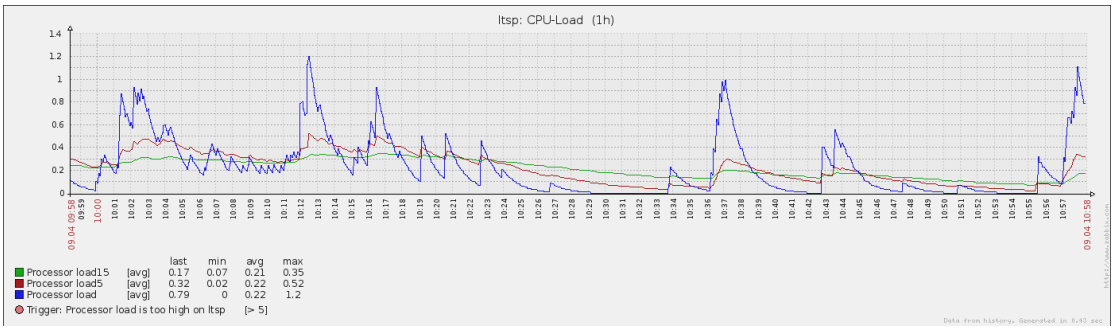

Figure 3: LTSP-server test#1 CPU performance

When comparing the measurement from the baseline, the results show that the load on the processor increased slightly but it seems overall unaffected. The peaks shown from the baseline measurements seem a bit more irregular which indicates that the server might have trouble returning to the normal state.

The second performance measurement of the LTSP-solution is conducted by using the same script after increasing the number of simulated users to seven and modifying the "send.yml" file to double the "bandwidth" and "packet_size" attributes. This test has shown the following CPU-Load:

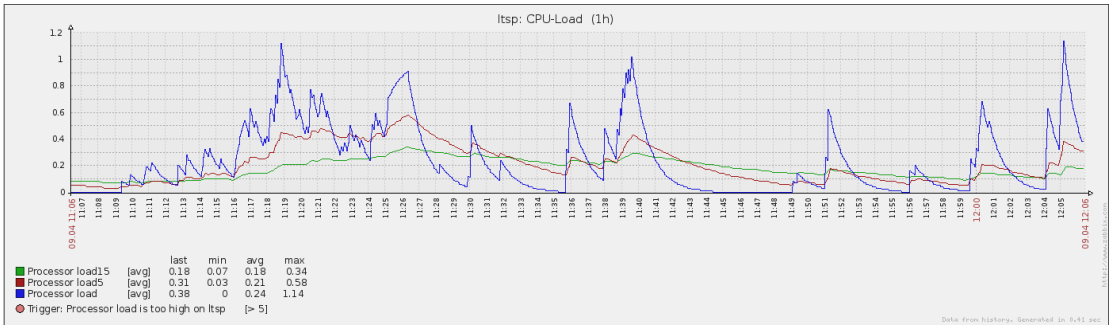

Figure 4: LTSP-server test#2 CPU performance

When comparing the measurement from the first performance test, the results show that the load on the processor has increased slightly. The graph shows an increasing irregularity in the processor load and the baseline becomes hard to read out.

The final performance measurement of the LTSP-solution is conducted by increasing the number of simulated users to 15 and by using the original script after modifying the "send.yml" file to increase the "bandwidth" and "packet_size" attributes ten times. This test has shown the following CPU-Load:

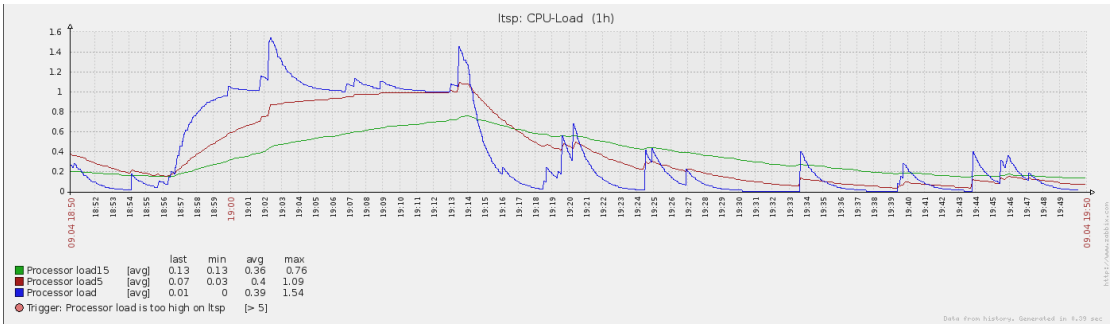

Figure 5: LTSP-server test#3 CPU performance

When comparing the measurement from the baseline, the results show a large increase of the processor-load. The graph shows that the load from the background process is quite low in comparison to the load generated from the test. The baseline is hard to read out in this graph which indicates that the system is having some difficulties returning to the normal behavior.

5.6.3 Cost

This solution is an open-source product that is under the license called "GNU GPL" which means that it does not cost anything to get access to the source-code. This also means that downloading and installing the thin-client software does not cost anything.

LTSP themselves do not appear to sell preconfigured hardware appliances with their solution, but there are available options from disklessworkstation.com. This homepage also provides with three different support levels which are called: incident, project support and managed support. In the cost-calculation, the chosen support level is incident. For more information, see appendix F.

LTSP offers three different thin-client hardware boxes. They differ in their hardware and their cost. It may be important to choose a hardware box that has a Gigabit interface and a DVI-port installed to support an expanding business. The chosen

hardware box in the cost-calculation is the cheapest hardware box with a Gigabit interface which in this case is the model called LTSP Term 1620[6].

Cost	Estimation
Initial cost	The initial cost will consist of buying a separate server and choosing clients from among the available hardware boxes. **Estimation: Low to medium cost**
Running cost	Power consumption ranged from 10 Watt to 14 Watt depending on the chosen hardware. **Estimation: Very low cost**
Support cost	Since there are three levels of available support, the cost varies greatly depending on the chosen level. The chosen level in this case is the cheapest one. This price-model is an annual cost per server and per client. **Estimation: Medium to High**

Table 3: LTSP Costs

5.7 Installation of Openthinclient

The installation of Openthinclient is a flexible installation that can be conducted on all the Linux distributions that support Java version 1.6 and it also supports Windows XP and Windows 2003 Server. The provided instructions for installing Openthinclient suggest that the server needs to have a graphical user interface installed so for that reason, the chosen operating system was Windows 2003 Enterprise server. When choosing a different base operating system for the comparison, it is important to note that the results will become harder to interpret since the system behaves differently. As stated earlier in this thesis, however, to ensure the validity of the chosen method it is important to follow the instructions given from the respective company.

Since Windows 2003 was the chosen operating system, the directory service to be installed is Active Directory (AD). To install AD on a server, open the Start-menu and click on "Manage your server". This will open a window with all the configuration-options available for the Windows 2003 server. In this window, click on "Add or remove a role". This menu shows a list of available roles where in this case the chosen role is: "Domain Controller" (Active Directory). Choosing this option and clicking next will start the installation for AD. During the installation, the choice for domain name was the following: "otc.local". The rest of the installation was conducted by using the default-values at all the options.

As mentioned above, Openthinclient needs a Java version of at least 1.6, which can be obtained by using the browser to download the newest version for Windows and installing it. The installation-file for Openthinclient-server was available to download as a java-file from their respective homepage. The installation-file is very similar to

[6] For more information, see http://www.disklessworkstations.com/200122.html.

that of a normal Windows program installation; accept the license, choose what to install and where to install it. During this installation, all the default options were chosen.

5.7.1 Configuration

To configure the Openthinclient-server, there is a web-interface that can be reached by using the browser to enter “http://localhost:8080”. This interface will only have one button which downloads and installs a java-file that will open the “thin-client manager”. This is the program where all the available configurations are made to the thin-client server and where the clients are managed. Thin-client manager requires a connection to AD and this is conducted by clicking the button called “Connect to an existing realm”. The program then will prompt the user for AD credentials. The user must then choose a “realm” where all the information is stored from Openthinclient. The realm points to an Organizational Unit (OU) in AD where information such as users and computers can be stored. The client configuration is similar to that of the normal AD configuration. There is a program called “Active Directory Users and Computers” where hosts and users are configured. It is important in this stage to make sure that the new clients and users are created under the correct OU.

5.7.2 Performance

The measurement of the baseline on Openthinclient-solution is conducted the same way as on the LTSP-solution: to add the hosts into Zabbix and monitor the hosts for a couple of hours without any load or traffic to see how they behave normally. For more information about the other measured attributes of the Openthinclient-solution, see appendix D. The baseline of one hour has shown the following CPU-Load:

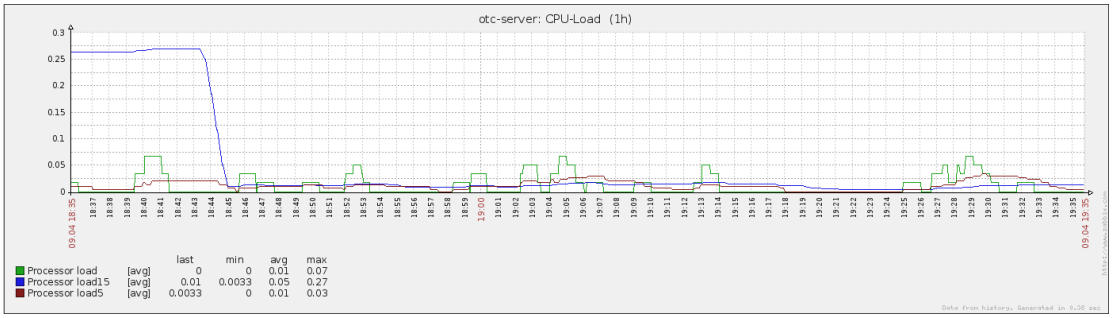

Figure 6: Openthinclient-Baseline: CPU-Load

The graph shows the processor load for one minute (displayed in green), processor load over 5 minutes (displayed in red) and processor load over 15 minutes (displayed in blue). The horizontal axis displays time in minutes while the vertical axis shows the load of the processor.

The baseline measurements of Openthinclient show that there seems to be some kind of background process which produces a small load on the processor. It is important to have that in mind when comparing this figure with the results from the performance tests. The blue line in the beginning of the baseline is a result of starting up the server a moment earlier.

The first performance measurement of the Openthinclient-solution is conducted the same way as on the LTSP-solution: The measurements are conducted by using the bash script to simulate five users with the normal size and bandwidth configured in the script (see 5.5). For information about all the monitored attributes of the

Openthinclient-solution, see appendix E. For detailed information regarding the tests (see 4.4). This test has shown the following CPU-Load:

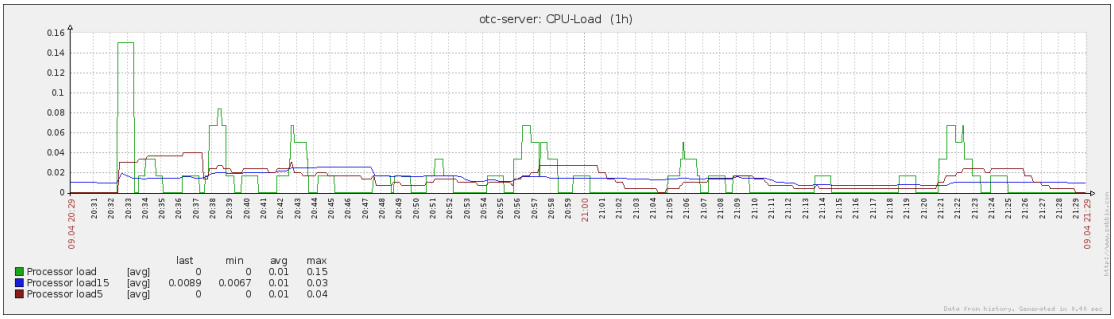

Figure 7: Openthinclient-Test#1: CPU-Load

When comparing the measurement from the baseline, the results show that the overall load on the processor increased slightly but it during the performance test (the first quarter of the graph), it seems largely unaffected. The peaks shown from the baseline measurements seem a bit more irregular which indicates that the server might have trouble returning to the normal state.

The second performance measurement of the Openthinclient-solution is conducted the same way as on the LTSP-solution: The measurements are conducted by using the same script after increasing the number of simulated users to seven and modifying the "send.yml" file to double the "bandwidth" and "packet_size" attributes. This test has shown the following CPU-Load:

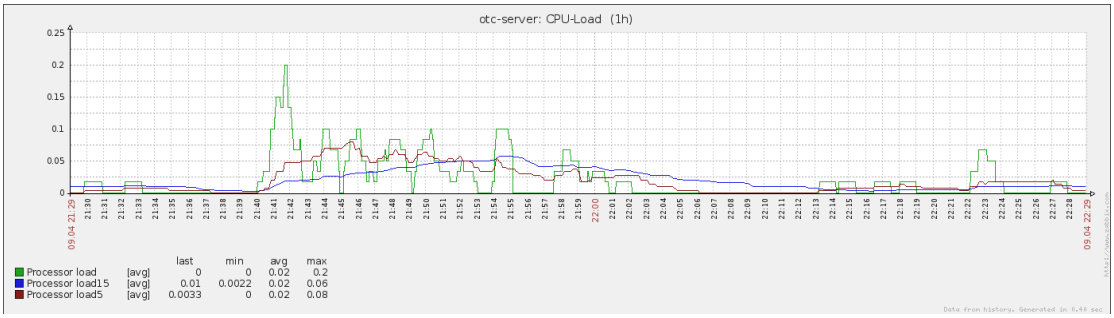

Figure 8: Openthinclient-Test#2: CPU-Load

When comparing the measurement from the first performance test, the results show that the load on the processor has increased slightly. The graph shows that during the test, the processor load increased by an amount that made the baseline hard to read out.

The final performance measurement of the Openthinclient-solution is conducted the same way as on the LTSP-solution: The measurements are conducted by increasing the number of simulated users to 15 and by using the original script after modifying the "send.yml" file to increase the "bandwidth" and "packet_size" attributes ten times. This test has shown the following CPU-Load:

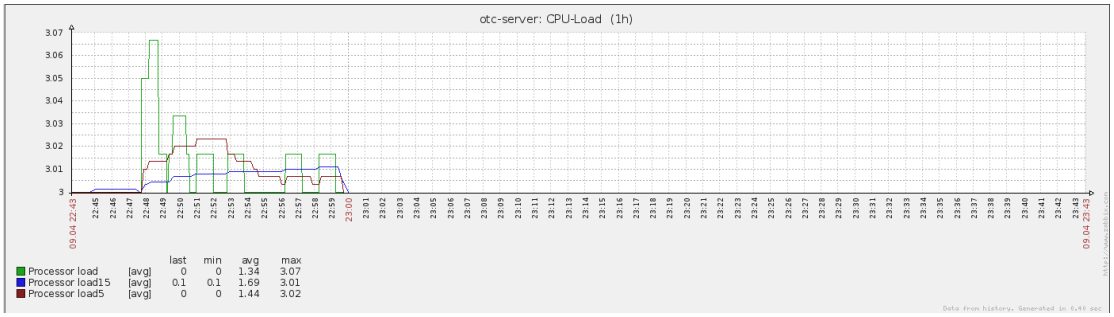

Figure 9: Openthinclient-Test#3: CPU-Load

When comparing the measurement from the baseline, the results show a large increase of the processor-load. The vertical axis is on a completely different scale, but the server seemed to manage the performance test. The baseline is impossible to read out in this graph which indicated that the load from the background process is negligible in comparison to the load generated from the test. Even though the latency during this test is higher than that of the baseline measurement, the server could cope with this amount of traffic.

5.7.3 Cost

This solution is an open-source product that goes under the license called GNU GPL v2, which means that the software is free to download and install.

Openthinclient offers three levels of support: basic, business and enterprise. These levels differ greatly in support-medium, service-hours, response-time during service hours and cost. Depending on the size of the company and the requirements of availability on support, the choice may vary. In the cost-calculation, the chosen support level is Basic. For more information, see appendix F.

Openthinclient offers many different thin-client hardware boxes. They differ in their hardware and their cost. It may be important to choose a hardware box that has a Gigabit interface and a DVI-port installed to support an expanding business. The chosen hardware box in the cost-calculation is the cheapest hardware box with a Gigabit interface which in this case is "ThinClients 1697"[7].

Cost	Estimation
Initial cost	The initial cost will consist of buying a separate server and choosing clients from among the available hardware boxes. **Estimation: Low to medium cost**
Running cost	Power consumption ranged from 10 Watt to 50 Watt depending on the chosen hardware. The client with 50 Watt power consumption is a client that supports two displays. **Estimation: Low cost**
Support cost	Since there are three levels of available

[7] For more information, see http://openthinclient.com/pdf/openthinclient_Pricelist-TCs_en.pdf.

	support, the cost varies greatly depending on the chosen level. The chosen level in this case is the cheapest one. This price-model is an annual cost per server and per client. There is also an option for support on third-party hardware. **Estimation: Medium to very high cost**

Table 4: Openthinclient costs

5.8 Creating Scenarios

This chapter contains three different scenarios with the proposed solution. The following scenarios are created based on my own experience as a system administrator and have no connection to any company.

5.8.1 First scenario

A small sized company is looking to decrease their cost by replacing their current fat-client environment with thin-clients using a Linux server. They have about 30 office workers that would need this solution and requires good performance without paying much for the solution. Their requirements are the following:

- The users should be able to read and write files
- Must be scalable for at least ten more users
- The solution should be as cheap as possible
- The solution should fulfill minimum performance required regarding latency

Proposed solution:
Since the company wanting to invest into thin-clients is looking for a cheap solution that is mildly scalable, the proposed solution is the LTSP-solution. Due to the fact that the LTSP-solution does not cost much and has a system that can handle an increase in both users and traffic without any particular hardware, this system seems the most suited for this particular purpose.

Since the customer requires a cheap solution with low latency, Openthinclient is not a fitting solution. The baseline showed a higher latency than that of the LTSP-solution.

5.8.2 Second scenario

A medium sized company is looking to invest into thin-client solutions. Their main reason to invest is not due to costs but instead to support many new users at the same time and also to have an available support option. They are not looking to replace their current fat-client solution but rather expand their business. Their requirements are the following:

- Must support many users at the same time
- Must be a scalable solution
- Must be a support-option available
- Must be able to interact with the fat-client solution and existing infrastructure

Proposed solution:
Since the company looking to invest is not trying to replace their current fat-client solution, the proposed solution is the Openthinclient-solution. Due to the fact that the Openthinclient-solution is easily integrated into either LDAP or AD makes it easy for the new system to interact with the old one. It is a scalable system that supports multiple users that can be added with the normal tools used in the fat-client solution. There is also available support with three different levels to suit their needs.

The requirements from the customer state that the system must be able to interact with the existing solutions. This makes LTSP a poor solution for this scenario since it uses a local database for accounts and clients.

5.8.3 Third scenario

An enterprise sized company has just got a new customer that requires them to invest into a new type of open-source thin-client solution. They are currently looking for a solution that has the whole package: available hardware, available support, good performance and has a scalable solution. They are not interested in mixing in third-party software or hardware since that would make their environment even more complex. Their requirements are as follows:

- Available hardware
- Available 24-hour support
- No performance issues
- Scalable solution

Proposed solution:
Since the company looking to invest is trying to invest into a solution that can provide the customer with the whole deal, the proposed solution is the Openthinclient-solution. Due to the fact that the Openthinclient-solution sell its own hardware, its own software and has several support options available, they are the most suited company to invest in.

The requirements from the customer states that there must be available 24-hour support. The different support level to LTSP differs in their response time but even the level with the highest availability does not have 24-hour support.

6 Analysis

Both solutions were tested in the same way, by editing the scripts shown in 5.4 to scale up the tests to measure the performance of the server. The baseline for the solutions is shown in appendix B and appendix D, while the results are shown in appendix C and appendix E.

6.1 Linux Terminal Server Project

The LTSP-solution is a product that supports several different Linux-distributions but there seems to be no available support for Windows. This makes LTSP a poor choice if the company looking to invest into thin-clients has thoughts about investing into Windows also. On Linux however, LTSP seems to work quite well.

The baseline measurement shows that the only activity is on the CPU once every fifteen minutes which seems to be some kind of background computation (see figure 1). By comparing the baseline to the first test, it is clear that the server handles five users without any trouble. Even when scaling up the tests to fifteen users sending with quite a large bandwidth, the server seemed to handle the traffic quite well. The latency increased from about 0.5ms (milliseconds) to about 4ms but there were no loss of packets. When looking at the results they show that the server can most likely handle at least double the amount of users without having any kind of trouble.

As mentioned before, since LTSP is an open-source project it is free to download and install. LTSP themselves do not sell any hardware but they reference to disklessworkstations.com which both provides with available hardware and support. There are three different hardware boxes that have quite a low energy consumption ranging from 10 to 14 Watt. There does not seem to be an available 24-hour support available which makes LTSP a poor choice when the company is larger than a medium sized company.

6.2 Openthinclient

The Openthinclient-solution is a product that supports every Linux-distribution that supports java version 1.6 and higher, and also Windows XP and Windows 2003 Server. Openthinclient do however require a GUI on the server since the only installation option seem to be based on a graphical installation. For that reason, Openthinclient seem better suited for a Windows installation. This solution requires an integration with either LDAP or AD in order to function as it should, which makes this installation able to integrate with existing solutions. This makes the solution a flexible choice for a company looking to expand their business with a thin-client solution.

The baseline measurements show that the background-activity of the Openthinclient is much higher than that of the LTSP-server. The background activity is highly likely to be that of the base operating system rather than the Openthinclient program. A ping-test shows that the latency of the baseline is much higher than that of the LTSP-solution: 0.5ms for LTSP and 2.2ms for Openthinclient.

While conducting the performance tests, the server seems largely unaffected from the first test. The CPU and memory has a slightly increased fluctuation but the effects are minor. When scaling up the tests to about fifteen users the CPU and memory has notably increased fluctuation, but from the results of these tests it shows that this

server would be able to handle at least two or three times the amount of users without any problem.

Openthinclient has available options for hardware and support. The available hardware has a low energy consumption which makes the running cost much lower than that of a Fat-client environment. There are available hardware boxes that support multiple screens which have an energy consumption of fifty Watt. Even when choosing among those models, the running cost is much lower than a normal Fat-client environment. There are three available support levels: basic, business and enterprise. These different levels range from a small company to a multi-national company depending on the need of the company.

6.3 Cost comparison

When one wants to invest into one of these solutions, the cost should be taken into account. In the following calculation, 1 server and 20 clients are counted. Peripherals such as power cables, network cables or network equipments are not covered by the calculations. The running cost is in this case counted without eventual replacement of hardware and maintenance. Since none of the solutions offers a server to their solution, the cost of a server is counted as **€2000** with a power consumption of 400 Watt. Both of the solutions are counted with the cheapest version of support with the cheapest available hardware with DVI and Gigabit-Ethernet. The chosen hardware model for the LTSP client is "LTSP Term 1620" which costs **$269.95** (194,67 U.S. Dollars(2011-06-03)) and has a power consumption of 14 Watt per client. The chosen hardware model for Openthinclient is "ThinClients 1697" which costs **€355** and has a power consumption of 25 Watt per client. Counting with 20 clients per solution:

$269,95*20 = **$5399** (€3893[8]) (LTSP)
€355*20 = **€7100** (Openthinclient)

14 Watt*20 = 280 Watt/hour (LTSP)
25 Watt*20 = 500 Watt/hour (Openthinclient)

The initial cost for LTSP is: **$5399** + **$**2902.6 = **$8301,6** (€5986,44)
The initial cost for Openthinclient is: **€7100 +** €2000 = **€9100**

The running cost for LTSP is: 280 + 400 = **680 Watt/Hour**
The running cost for Openthinclient is: 500 + 400 = **900 Watt/Hour**

Cost	LTSP	Openthinclient
Initial cost	**€5986,44**	**€9100**
Running cost	**680 Watt/Hour**	**900 Watt/Hour**
Support cost	Medium to high	Medium to very high

Table 5: Cost comparison summary

[8] Translated from U.S Dollars to Euro, 2011-06-03.

7 Conclusion

According to Cantoria (2010), one of the weaknesses of thin-client solutions are that thin-clients work poorly when the latency of the network is high. This thesis does, however, show a network with a fairly high latency during the performance tests without any lack of functionality. The results indicate that both of the solutions would be able to handle double that amount of traffic and simulated users without any problems. The comparison then comes down to the functionality and cost.

The functionality of the LTSP-solution are quite limited. The options they have are limited to Linux-distributions without the availability of Windows. The functionality of the Openthinclient-solution however are quite flexible. They support Linux and Windows and they have the ability to integrate their system with an existing directory service. Both of the solutions have available hardware for sale with different specification depending on the needs of the company.

Giving a final answer to which product is the best is a challenging task since they both have strong and weak sides. Depending on what the customer is after when looking to invest into an open-source alternative of thin-client solutions, the result will be different. Openthinclient is strong when it comes to functionality due to the possibility to integrate the system with an existing infrastructure. LTSP is a cheaper solution regardless of the cost-type and both Openthinclient and LTSP are strong when it comes to performance. Openthinclient has an available 24-hour support option while LTSP does not. LTSP has On-Site support while Openthinclient only has remote support.

7.1 Discussion

The result from this thesis shows that both LTSP and Openthinclient are strong candidates when choosing a thin-client solution, but are the tests valid?

The results show that the solutions work when constructed in an isolated environment but when putting the solutions in an environment with existing traffic and normal users, the results may vary greatly. It does not necessarily mean that the solutions work in a production-environment even if they perform well in a laboratory setting. Moreover, the traffic used during the experiments does not model the behavior of real users, since packgen does not provide such functionality. The users modeled present thus, a static behavior. To improve these results to a more realistic environment, it would be preferable to conduct these tests in a production-environment of a company, with normal users using both solutions. This does, however, mean that the results might become very company-specific. While the results might not be valid in a production-environment, it does show that both of these solutions work adequately in a laboratory setting.

The results of this thesis can, however, be used as a methodology for comparing thin-client solutions in general, since it provides a detailed description on how to construct test scenarios, how to set the different environments and how to compare thin-client solutions.

7.2 Future work

Since the majority of the tests in this thesis are based on network traffic, it would be preferable if another test could be conducted where the memory-usage is put to test. This can be conducted by opening many instances of the web-browser Firefox from

the clients and at the same time monitor the performance of the server with a monitoring tool. The comparison would also benefit from a large-scale performance test with many clients to show how much the solutions can handle.

It would also be preferable to construct realistic scenarios by contacting different companies. This would give a more correct view of the solutions in a production-environment.

Since the compared solutions is an open-source alternative and no commercial alternatives were compared, this is what the thesis is lacking. These open-source alternatives should be compared with more commercial products such as Citrix and Sun-ray to see if the commercial products live up to their names or if an open-source product can be a valid competitor.

8 References

Berndtsson, M. Hansson, J. Olsson, B. and Lundell, B. (2008). *Thesis Projects: A Guide for Students in Computer Science and Information Systems*. 2nd ed. London Limited: Springer-Verlag.

Cantoria, C.s. (2010). *Thin Client Review- The Pros and Cons of Thin Client Computing* [online]. [Accessed 23rd Jan 2011] Available from: http://www.brighthub.com/environment/green-computing/articles/66417.aspx.

Chiorean, C. (2009). *FirstDigest* [online]. [Accessed 1st June 2011]. Available from: http://www.firstdigest.com/2009/12/great-tool-for-testing-qos-implementation/.

Greenberg, S. (2005). *Thin Client Computing: Tech info* [online]. [Accessed 4th April 2011]. Available from: http://www.thinclient.net/thinclient_history.html.

Sobotta, A. Sobotta, I. Gøtze, J. (2009). Cloud Computing. *In*: Sobotta, A. Sobotta, I. Gøtze, J., (ed). *Greening IT*, London: Gøtze Consulting, p68-93

Hemmendinger, D., Ralston, A. and Reilly, E.D. (1998). *Client/Server Term Definition* [online]. [Accessed 8th April 2011]. Available from: http://www.maffeis.com/articles/research/client_server.pdf.

Kanter, J. (2000). *Understanding Thin-Client/Server Computing*. 1st. ed. London: Microsoft Press.

LLC Books (2010). *Thin Clients: Thin Client, Ncomputing, Blade PC, Sun Ray, Thincan, Openthinclient, X Terminal, Blit, Ndiyo*. First. ed. London: BOOKS LLC.

McDonough, M. (2009). *Is SaaS Really a New Concept?* [online]. [Accessed 23rd Jan 2011] Available from: http://www.brighthub.com/computing/windows-platform/articles/52867.aspx.

McKenna, F. (2002). *Thin-Client VS Fat-Client Computing* [online]. [Accessed 23rd Jan 2011] Available from: http://www.knowledgeonecorp.com/news/pdfs/Thin%20client%20vs%20Fat%20client%20Computing.pdf.

Reese, G. (2000). Distributed Application Architecture. In: *Database Programming with JDBC and Java*. 2nd ed [online]. [Accessed 23rd Jan 2011] Available from: http://java.sun.com/developer/Books/jdbc/ch07.pdf: O'Reilly Media.

Rizwan, A. (2011). *Upgrading and Performance Analysis of Thin Clients in Server Based Scientific Computing* [online]. [Accessed 20th April 2011]. Available from: http://liu.diva-portal.org/smash/get/diva2:400152/FULLTEXT02.

Schmidt, B.K., Monica, S., Lam, J. and Northcutt, D. (1999).*The interactive performance of SLIM: a stateless, thin-client architecture* [online]. [Accessed 23rd Jan 2011] Available from: http://labs.oracle.com/techrep/Perspectives/PS-01-5.pdf#search=%22thin-client%20history%22.

Tykesson, A. Ericsson, T. (1999). *Tunna eller tjocka klientapplikationer - en studie i Client/Server-teknik* [online]. [Accessed 20th April 2011]. Available from: http://gupea.ub.gu.se/handle/2077/1360.

Tyson. B (2010). *The History of Thin Clients* [online]. [Accessed 27th Jan 2011] Available from: http://www.brighthub.com/environment/green-computing/articles/71173.aspx.

Homepage of Zabbix : An Enterprise-Class Open Source Distributed Monitoring Solution [online]. (2011) [Accessed 20th April 2011]. Available from: http://www.zabbix.com/.

Openthinclient [online]. (2011) [Accessed 20th April 2011]. Available from: http://openthinclient.com/.

The Linux Terminal Server Project [online]. (2009) [Accessed 20th April 2011]. Available from: http://ltsp.org/.

Appendix A – Topology

The following picture shows the chosen network design for the solutions.

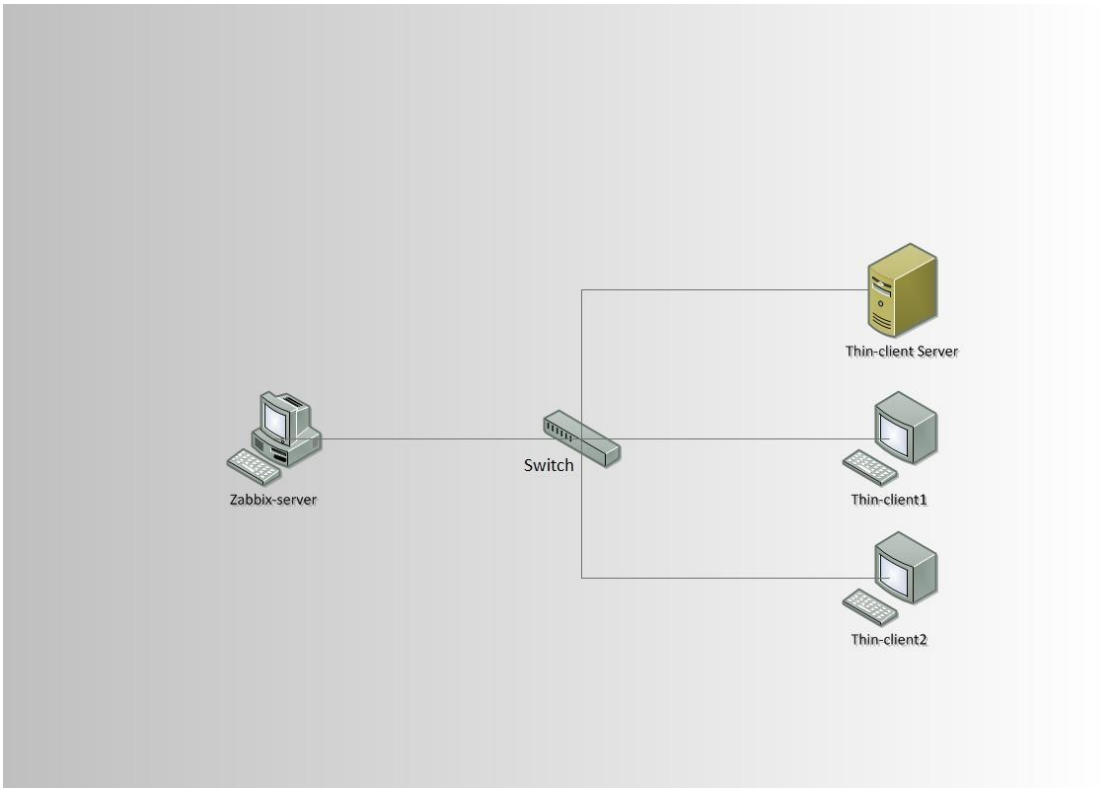

Figure 10: Network Topology

Appendix B: LTSP server - baseline

The following graphs show the baseline of the LTSP-server.

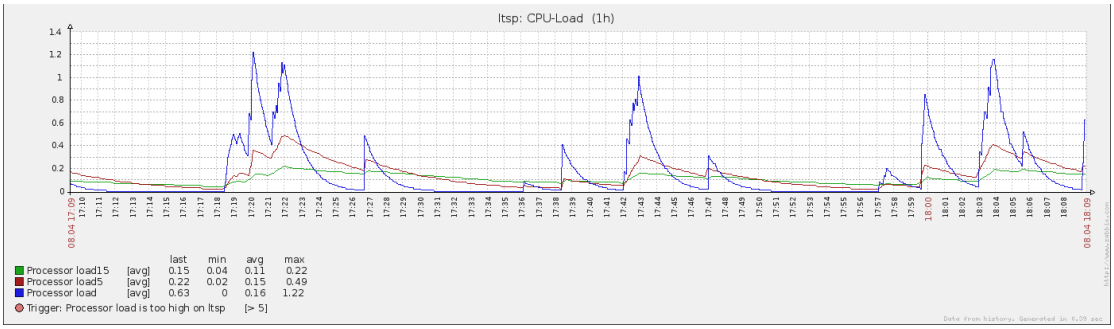

Figure 11: LTSP-Baseline: CPU

Figure 10 shows the processor load for one minute (displayed in blue), processor load over 5 minutes (displayed in red) and processor load over 15 minutes (displayed in green). The horizontal axis displays time in minutes while the vertical axis shows the load of the processor. If the processor-load reaches 1, it means that one core of the CPU has reached 100% load.

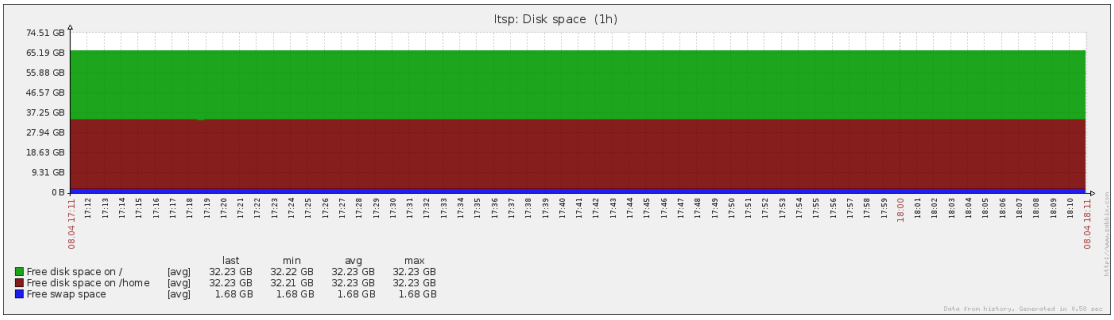

Figure 12: LTSP-Baseline: Disk Space

Figure 11 shows the free swap space (displayed in blue), free disk space on /home (displayed in red) and free disk space on the root-catalogue (displayed in green). The horizontal axis displays time in minutes while the vertical axis shows the amount of space.

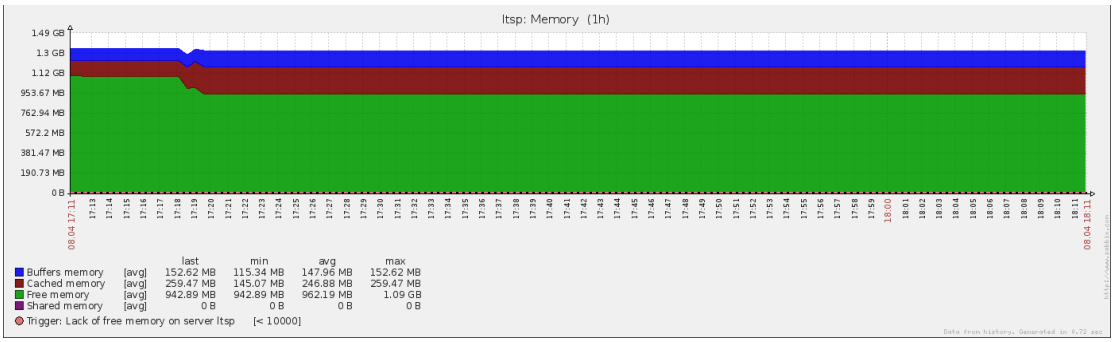

Figure 13: LTSP-Baseline: Memory

Figure 12 shows the buffered memory (displayed in blue), cached memory (displayed in red), free memory (displayed in green) and shared memory (displayed in purple). The horizontal axis displays time in minutes while the vertical axis shows the amount of memory.

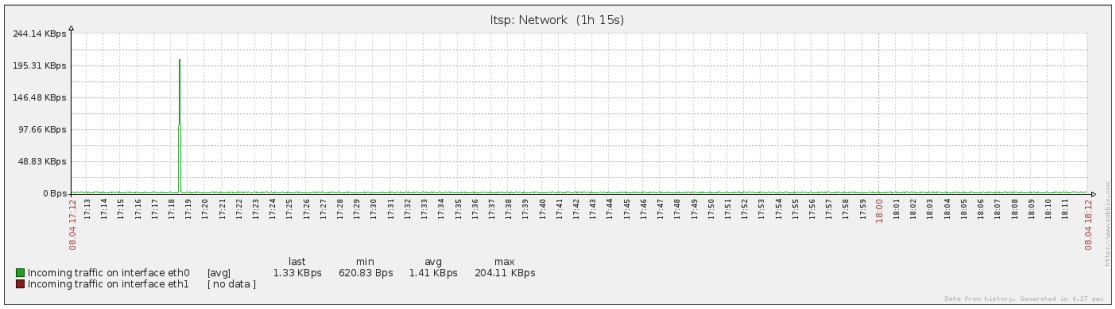

Figure 14: LTSP-Baseline: Network

Figure 13 shows the network traffic incoming on the interface called eth0 (displayed in green) and the traffic incoming on the interface called eth1 (displayed in red). The horizontal axis displays time in minutes while the vertical axis shows the amount of traffic.

--- 192.168.186.111 ping statistics ---

138 packets transmitted, 138 received, 0% packet loss, time 136997ms

rtt min/avg/max/mdev = 0.307/0.527/7.086/0.572 ms

Appendix C: LTSP server - Performance tests

The following graphs show the performance graphs of the LTSP-solution during the tests. The test was conducted for fifteen minutes, so the first quarter of the graphs is the relevant parts. The other part of the graph is the Server returning to the normal state. Below is the first test conducted with five simulated users. The first test showed unaffected memory and disk-space so those graphs will not be shown.

First test:

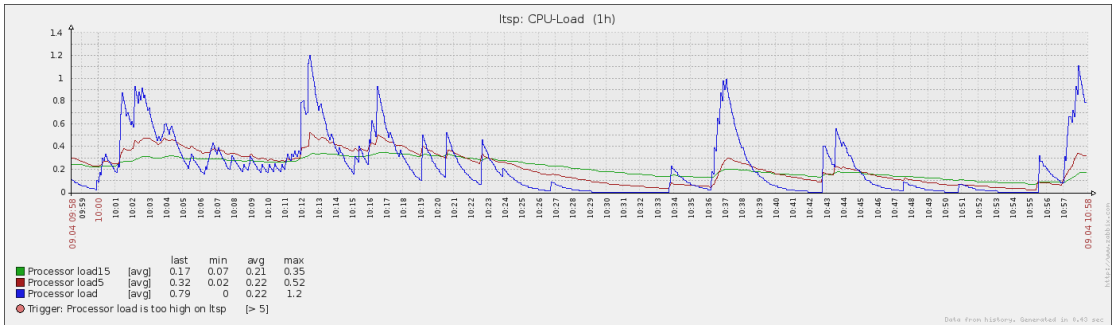

Figure 15: LTSP-Test#1: CPU

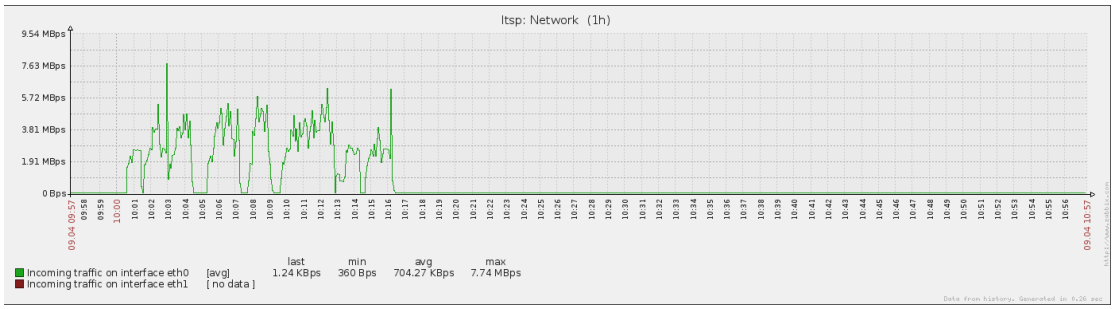

Figure 16: LTSP-Test#1: Network

The ping test from thin-client2 show in the topology (see appendix A) showed that the latency became higher but there were no loss of packets. Below follows the second test conducted with seven simulated users.

Second test:

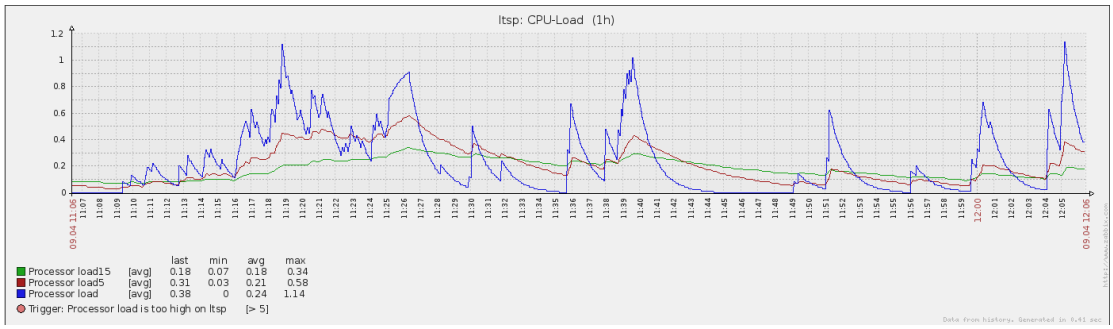

Figure 17: LTSP-Test#2: CPU

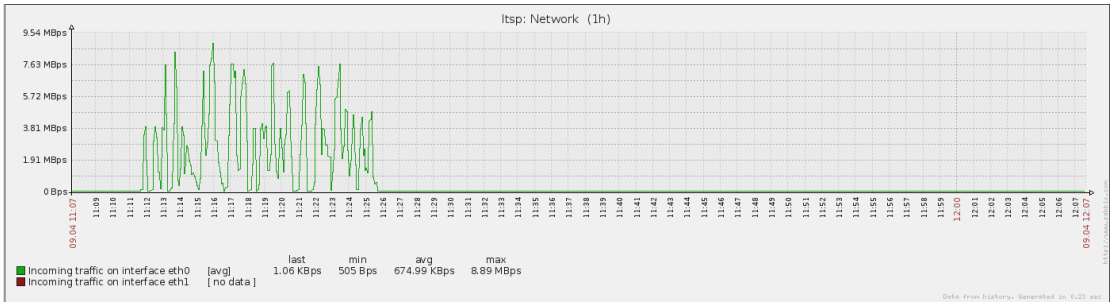

Figure 18: LTSP-Test#2: Network

The second test also showed unaffected memory and disk-space so those graphs will not be shown. The ping results showed that the latency became slightly higher but there was no loss of packets. Below follows the final test configured to simulate 15 users. Since memory and disk-space still showed to be unaffected, these will not be shown.

Third test:

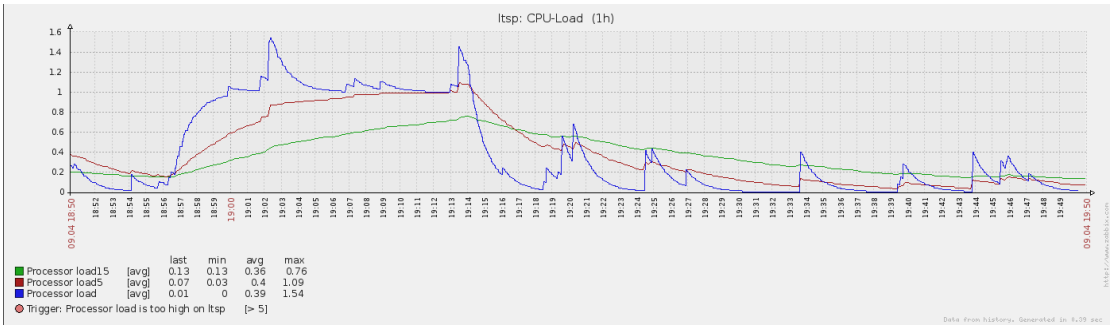

Figure 19: LTSP-Test#3: CPU

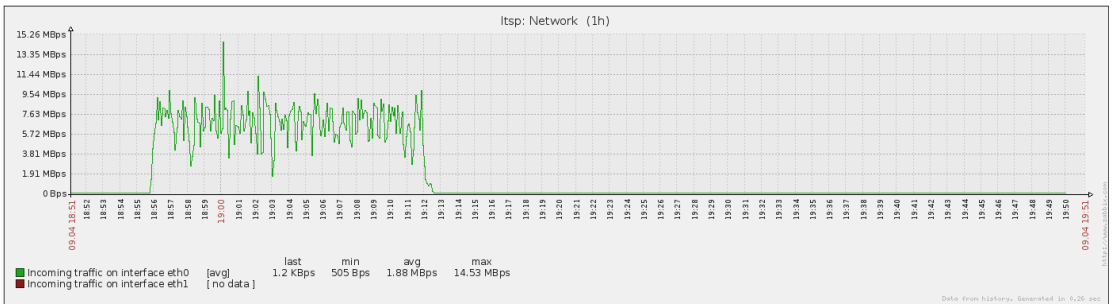

Figure 20: LTSP-Test#3: Network

--- 192.168.186.111 ping statistics ---

380 packets transmitted, 380 received, 0% packet loss, time 379108ms

rtt min/avg/max/mdev = 0.234/2.611/22.357/4.282 ms

The ping test from thin-client2 show in the topology (see appendix A) showed that the latency became slightly higher but there were no loss of packets.

Appendix D: Openthinclient-server - baseline

The following graphs show the baseline of the Openthinclient-server.

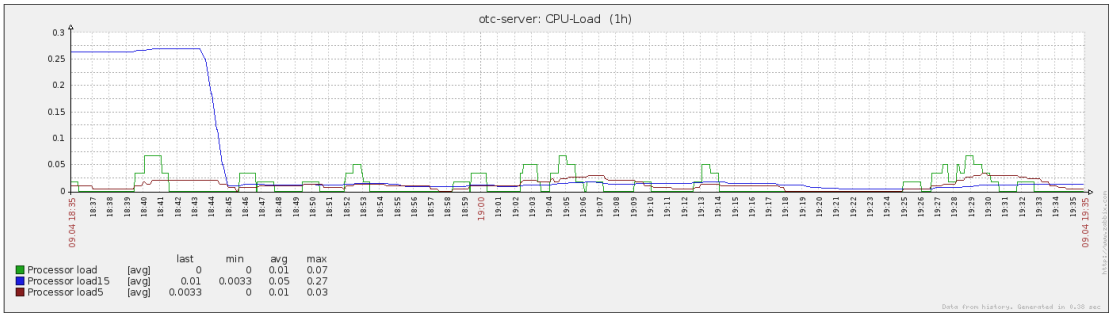

Figure 21: Openthinclient-Baseline: CPU-Load

Figure 20 shows the processor load for one minute (displayed in green), processor load over 5 minutes (displayed in red) and processor load over 15 minutes (displayed in blue). The horizontal axis displays time in minutes while the vertical axis shows the load of the processor. If the processor-load reaches 1, it means that one core of the CPU has reached 100% load.

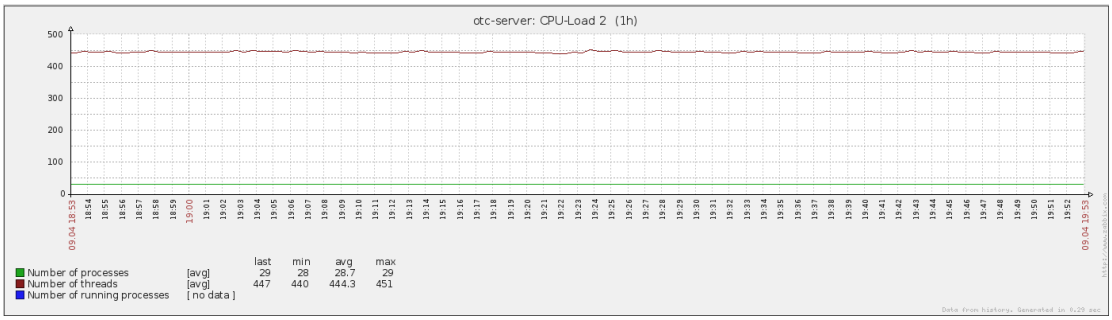

Figure 22: Openthinclient-Baseline: CPU Threads and Processes

Figure 21 shows the number of processes (displayed in green), number of threads (displayed in red) and number of running processes (displayed in blue). The horizontal axis displays time in minutes while the vertical axis shows the amount of threads.

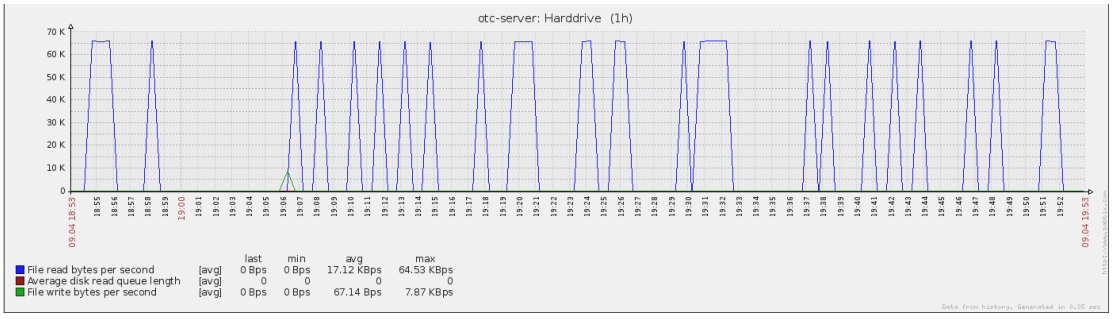

Figure 23: Openthinclient-Baseline: Hard-drive read and writes per second

Figure 22 shows the amount of bytes read per second (displayed in blue), the average disk read queue length (displayed in red) and amount of bytes written per second (displayed in green). The horizontal axis displays time in minutes while the vertical axis shows the amount of bytes.

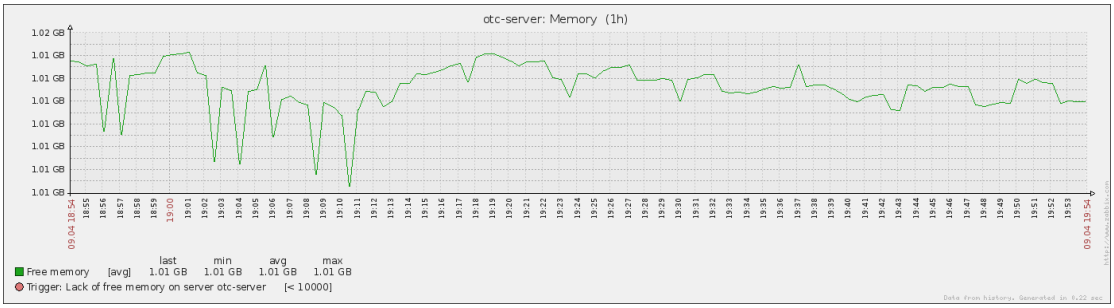

Figure 24: Openthinclient-Baseline: Memory

Figure 23 shows the amount of free memory (displayed in green. The horizontal axis displays time in minutes while the vertical axis shows the amount of memory.

--- 192.168.186.88 ping statistics ---

174 packets transmitted, 174 received, 0% packet loss, time 172985ms

rtt min/avg/max/mdev = 0.261/0.516/28.385/2.139 ms

Appendix E: Openthinclient-server - Performance tests

The following graphs show the performance graphs of the Openthinclient-solution during the tests. The test was conducted for fifteen minutes, so the first quarter of the graphs is the relevant parts. The other part of the graph is the Server returning to the normal state. Below is the first test conducted with five simulated users. The first test showed unaffected CPU-threads and Hard-drive so those graphs will not be shown.

First test:

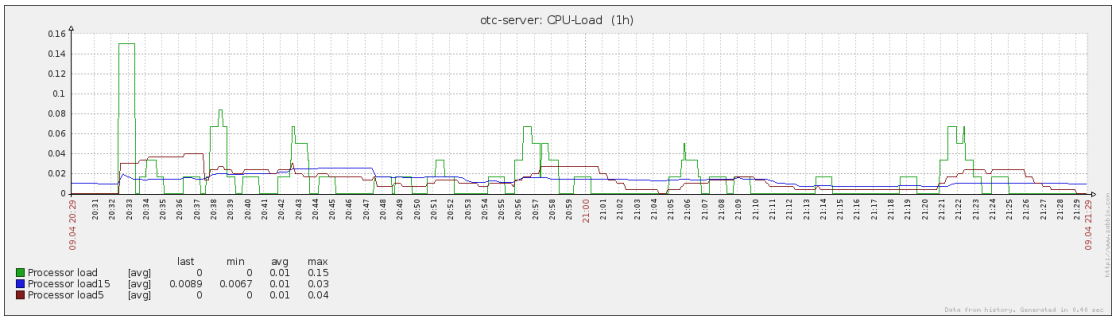

Figure 25: Openthinclient-Test#1: CPU-Load

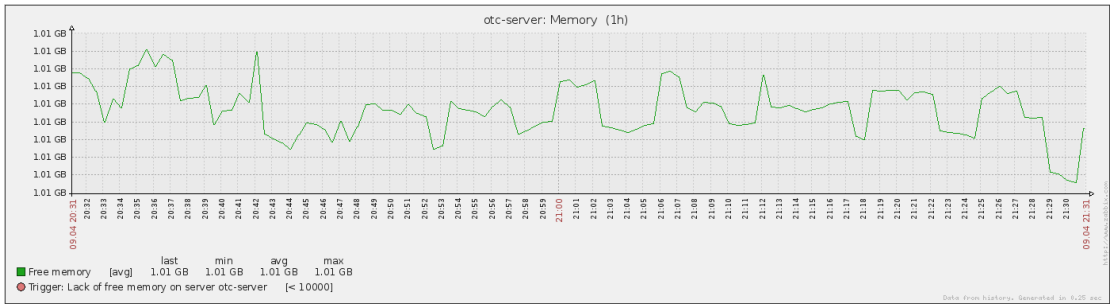

Figure 26: Openthinclient-Test#1: Memory

--- 192.168.186.88 ping statistics ---

622 packets transmitted, 622 received, 0% packet loss, time 621137ms

rtt min/avg/max/mdev = 0.237/0.947/45.727/2.028 ms

The ping test from thin-client2 show in the topology (see appendix A) showed that the latency became higher but there were no loss of packets. Below follows the second test conducted with seven simulated users.

Second test:

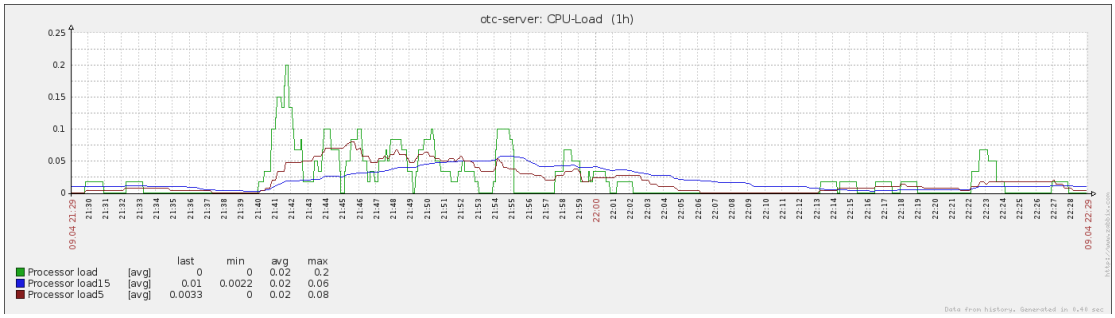

Figure 27: Openthinclient-Test#2: CPU-Load

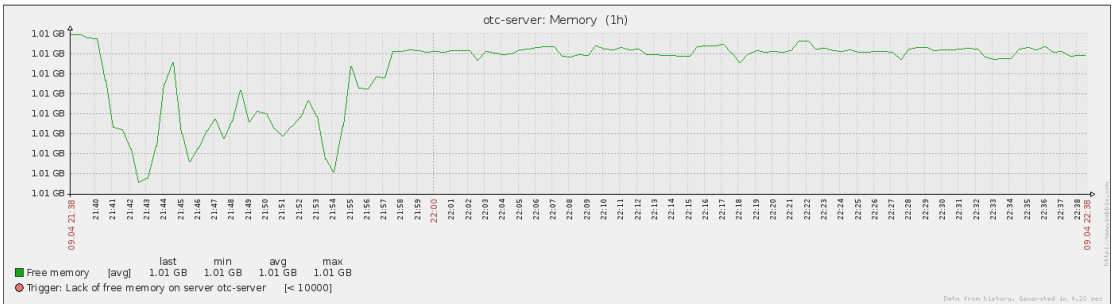

Figure 28: Openthinclient-Test#2: Memory

The second test also showed unaffected CPU-threads and Hard-drive so those graphs will not be shown. The ping results showed that the latency became slightly higher but there was no loss of packets. Below follows the final test configured to simulate 15 users. Since memory and disk-space still showed to be unaffected, these will not be shown.

Third test:

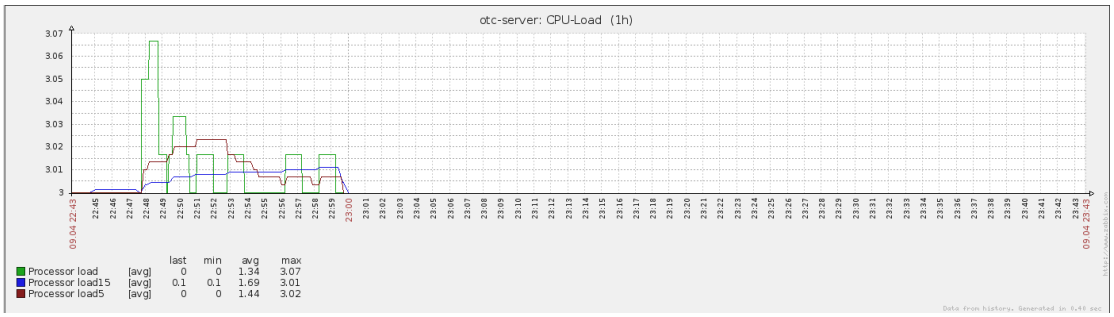

Figure 29: Openthinclient-Test#3: CPU-Load

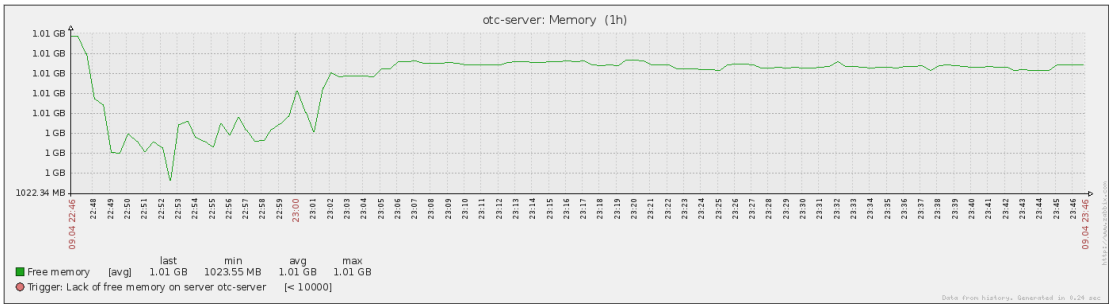

Figure 30: Openthinclient-Test#4: Memory

The ping test from thin-client2 show in the topology (see appendix A) showed that the latency became slightly higher but there were no loss of packets.

Appendix F: Support costs

The following table shows the different support levels with their respective costs.

Support level	Basic	Business	Enterprise
Support over	Web (support portal, E-mail), telephone, remote hands	Web (support portal, E-mail), telephone, remote hands	Web (support portal, E-mail), telephone, remote hands
Response times during service hours	Category 1[9] :48 h Category 2 : 4 bd Category 3 : 30 bd	Category 1 : 12 h Category 2 : 48 h Category 3 : 14 bd	Category 1 : 8 h Category 2 : 24 h Category 3 : 10 bd
Service hours	Monday to Friday except weekends and holidays	Monday to Friday except weekends and holidays	Monday to Sunday including holidays
Annual basic rate for 1 server	1.550 EURO	2.950 EURO	12.000 EURO
Annual addition price for a further server	750 EURO	1.350 EURO	5.200 EURO
Annual charge per ThinClient (up to 100 clients)	5 EURO	8 EURO	30 EURO
Annual charge per ThinClient (above 100 clients)	4 EURO	6 EURO	20 EURO
Annual surcharge for support of third-party hardware per hardware type	420 EURO	650 EURO	2.500 EURO

Table 6: Openthinclient support

[9] These categories references to the criticality of the incident where category 1 is the highest incident.

Support Level	Incident	Project	Managed
Response Time	48 Hours (Mon - Fri)	24 Hours (Mon - Fri)	8am - 5pm EST (Mon - Fri)
Method	Email, Chat	Email, Chat, Phone	Email, Chat, Phone, On-Site
Scope	None	- Application customization - Application development - Maintenance - OS Upgrades - Remote Server Access - Research & Development	- Application customization - Application development - Maintenance - OS Upgrades - Remote Server Access - Research & Development - Backup solution - Server Monitoring - SLA

Table 7: LTSP support