

**TECHNICAL DESIGN**

*Project: Virtualisation*

|  |  |  |  |
| --- | --- | --- | --- |
| **Client: Plaintech** |  | **Filename:** | Technical Design v1.0.7 team 1.docx |
| **Project: Virtualisation** |  | **Version:** | 1.0.7 |
| **Author: Willem Westerhof, Pieter Dieleman, Eddy van der Steen, Kjell Zijlemaker, Rodney Lanuzga** |  | **Date:** | 10-11-2014 |

# **Documentproperties**

## History

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Version** | **Date** | **Changes(concept/final)** | **Client** | **Author(s)** |
| 0.1 | 16-9-2014 | Concept | Plaintech | All Teammembers |
| 0.2 | 17-9-2014 | Charts added | Plaintech | All Teammembers |
| 0.3 | 17-9-2014 | Descriptions added | Plaintech | All teammembers |
| 0.4 | 17-9-2014 | Grammer check | Plaintech | Willem Westerhof |
| 0.5 | 23-9-2014 | Changed Archimate diagrams to new versions | Plaintech | Eddy van der Steen |
| 0.6 | 2-10-2014 | Added IP plan and global network topology | Plaintech | Eddy van der Steen |
| 0.7 | 10-11-2014 | Added models for network and database models | Plaintech | Kjell Zijlemaker |

## Approval

This document needs the following signatures of approval:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | **Role** | **Autograph** | **Document Date** | **Version** |
| Arjen Jansen | Project manager |  | 10-11-2014 | 1.0.7 |
| CEO/CFO | Client |  | 10-11-2014 | 1.0.7 |

Inhoudsopgave

Documentproperties 2

History 2

Approval 2

1. Introduction 4

2. Archimate System Model 5

3. Network Design 7

4. Technology model: 8

5. Server Tier Model 9

5.1 Deeper into the layers: 10

6. ERD 11

7. VM Management 12

8. IP Plan 13

# 1. Introduction

In this document you will find all the technical functionalities for the new virtualization platform as well as the required infrastructure. The technical functionalities also include the models for the network with all needed hardware and software components as required for plaintech.   
  
This document is spliced in 2 major parts. An schematic overview made with the Archimate modelling tool and a written explanation of these schematics.

# 2. Archimate System Model

This archimate model has two layers: the technology and application layer. These two layers are in conjunction with each other, with the sole purpose of defining the newly designed architecture for Plaintech.

As seen in the model below, we first have the technology layer for indicating the servers, that will run in support for the applications. As you can see, we have one server node where Debian will run on. In the Debian enviroment there will be a database that has the purpose to support several applications, higher up in the layerstructure.

Higher up in the layerstructure you’ll see that there are several services that will help realize the underlying structure, for these services will support the application layer.

* Database service
* Python/Bash service
* KVM Hypervisor service

The hypervisor service is the most important since this will be the foundation for the virtualization platform. As agreed upon, the KVM will be Linux based.

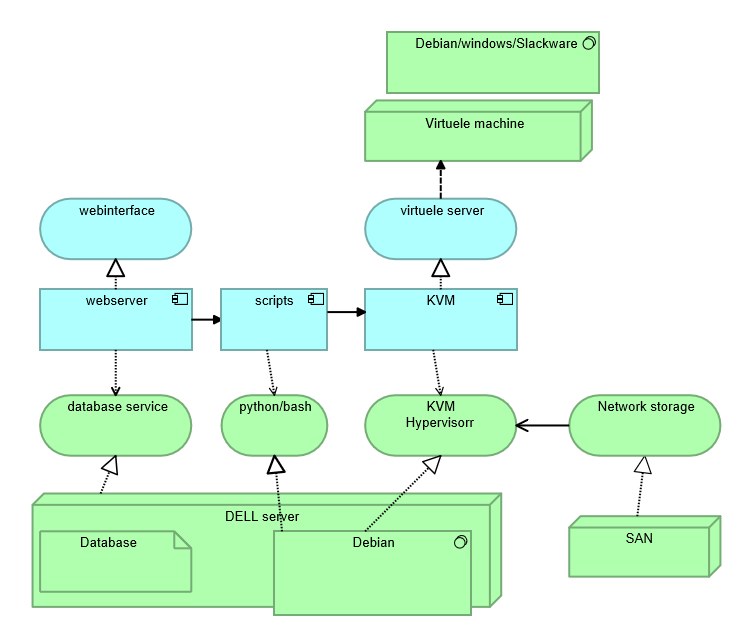
In the application layer, we have selected the applications that will need to run on the server and will trigger another application for execution. The following applications will trigger another:: **Webserver → Scripts for the KVM → KVM**

Furthermore, the webserver itself will also realize the webinterface for the customers that’ll login for managing their servers. Also, the KVM will realize the virtual server that will be the hosting part of the virtual machine.

The last part (and layer) for the Archimate system model is (are) the virtual machines that will be created by the virtual server. These virtual machines will be able to support the following operating systems:

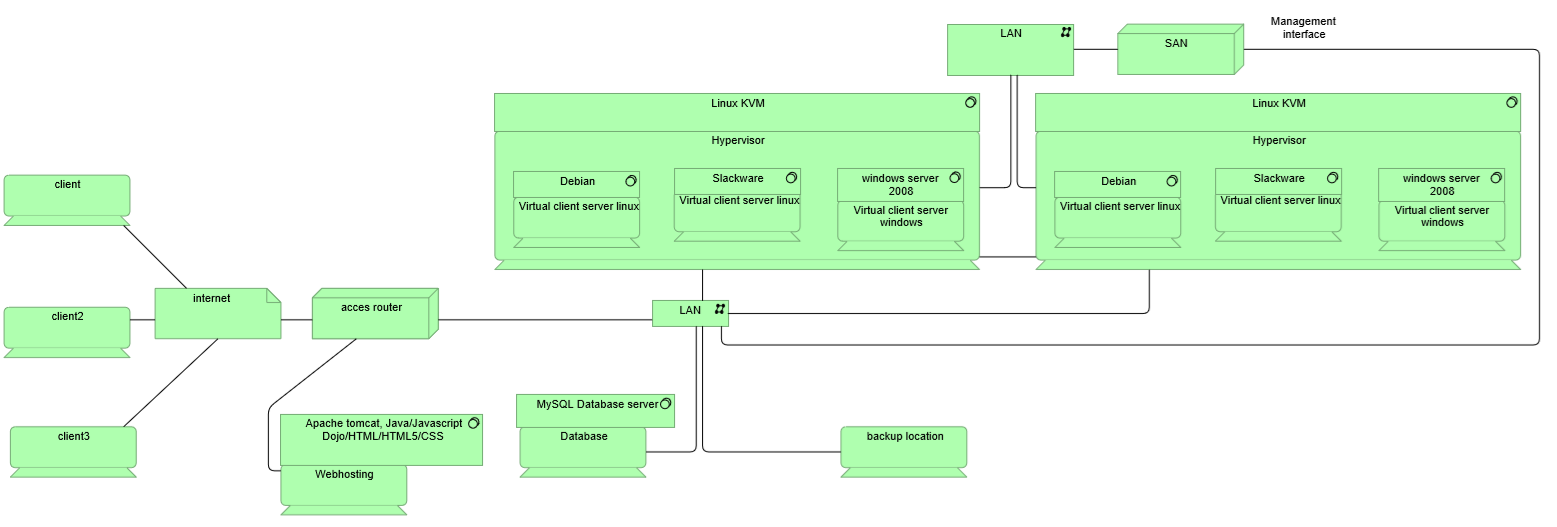
* Debian
* Windows
* Slackware

The customer will be able to choose the specifications for the host they want to buy. Of course there will be differences in cost per operating system, although this will not be discribed here.



# 3. Network Design

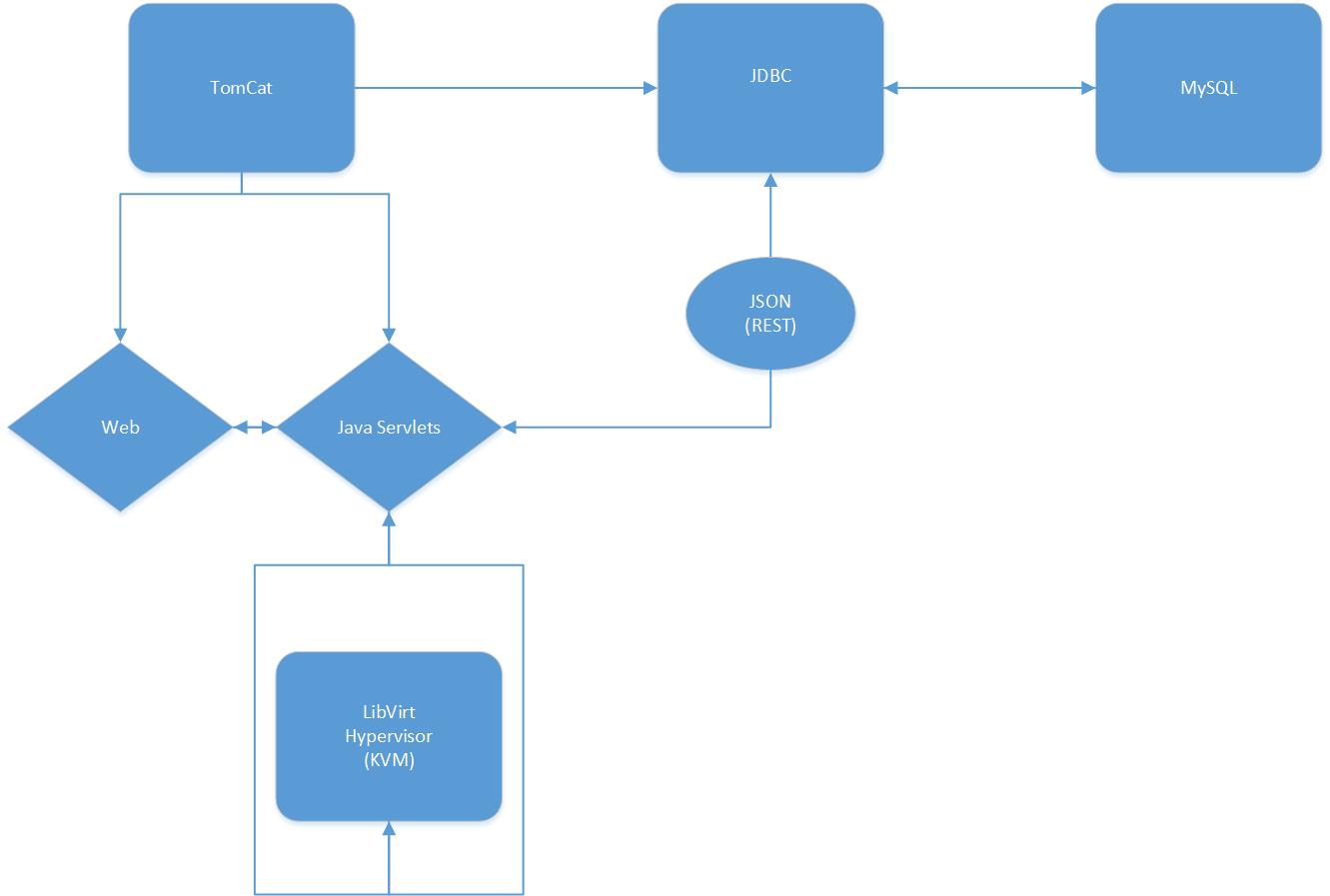
In the schematic below you can find detailed information about the network setup. All the way on the left side of the diagram there are a number of “clients.” These clients are customers on any kind of devices. These clients can then connect to the plaintech website to browse on the website and order a server.   
  
The computer that hosts the website is connected to a local area network. Within this network there are 5 categories of machines. First off the webhosting machine, this is the machine that hosts the plaintech website and communicates with the database and hypervisor machine.  
  
The next part of the network is the database machine. This machine is dedicated to database coverage and making sure all data is processed quickly.  
  
Then there is the backup server. All data and virtual machines that have the daily backup package are backed up to this machine.   
  
Finally there is the hypervisor. This is the machine that creates new virtual machines as requested by the user. Technically this isn’t just one machine, it is a node of servers that combine their power. This greatly improves the flexibility and scalability of the overall system.

The last machine all the way to the right in the diagram is a server node which holds an exact copy of all Virtual servers with the high availability package. This way if anything happens to the Virtual server it can be restored instantly by switching over to the secondary server.  


# 4. Technology model:

The technology model is a small overview of how the technology will flow through the application. For example, all of the logic (and html) is kept in the TomCat container. In the middle of that, there will be JSON (needed with dynamic content), but can also go straight to the JDBC for making the connection to MySQL.

The data will go back through the JDBC connection and will go back, either by JSON, or without if preferred. The servlets will maintain some control for the data that will be send to LibVirt, for example, the scripts that LibVirt will need to receive for making new connection to VM’s.



# 5. Server Tier Model

In the Server Tier model, we see 3 tiers. These tiers will make sure that all the data will be in it’s designated place. The presentation tier will be responsible for showing all the generated data with the designated servlets. The Data logic tier will be responsible for the logic part of the data, coming in from the database. All the checks and complicated things (such as objects) will be done here.

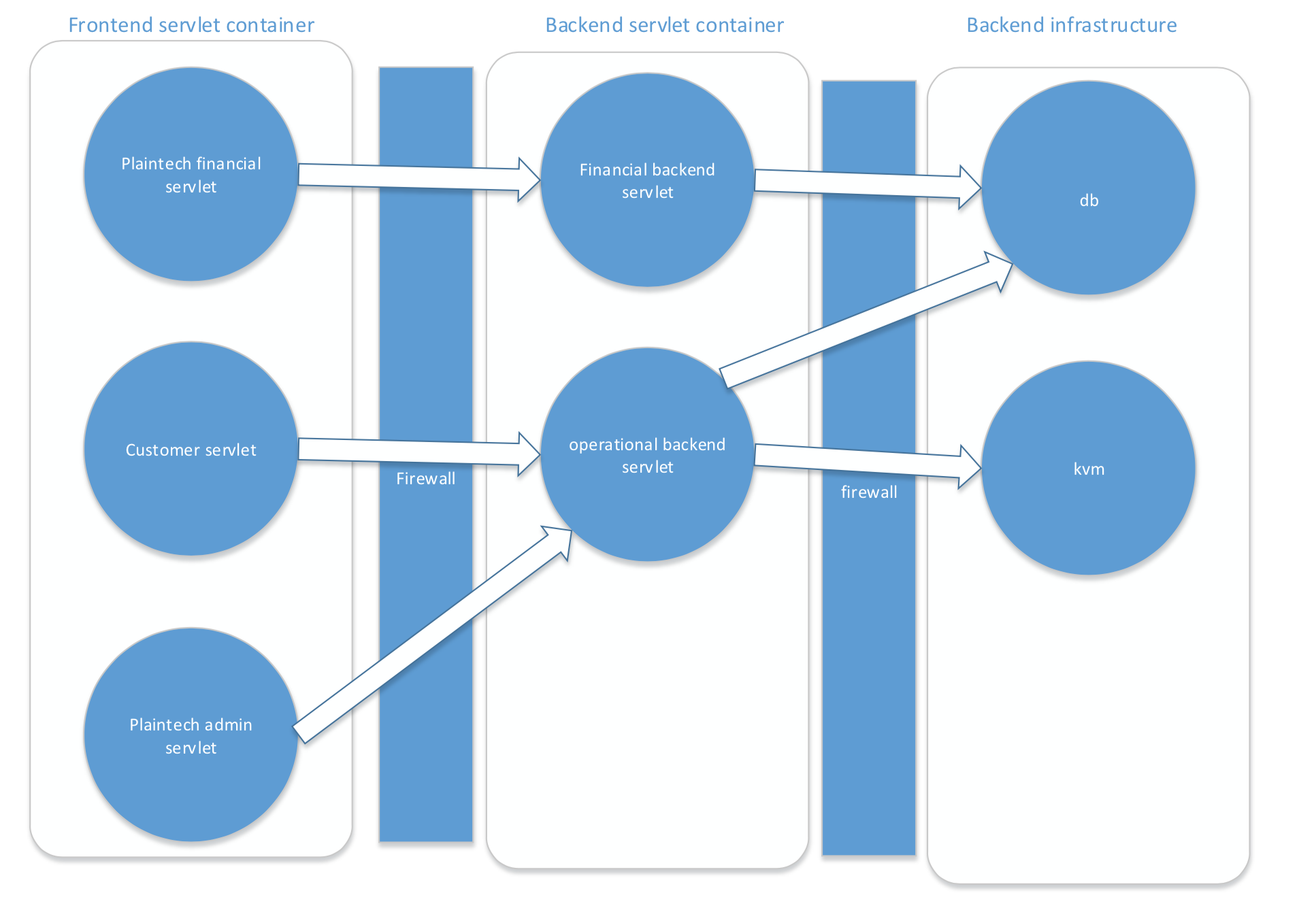
The Infrastructure tier will be responsible for making the connection to the database and getting the data for the data logic tier.

If done correctly, all the data will be nicely encapsulated and the server tier will be easy to manage and expand.

Macintosh HD:Users:kjellzijlemaker:Downloads:tiered design.pdf

## 5.1 Deeper into the layers:

In the diagram showed next, we zoomed into the layers some deeper. Here, the firewalls and servlets are shown that will power the application. There can also be several containers for improvement of the security, but also the expandability. On top of that, the servlets will also communicate with RESTFUL services.



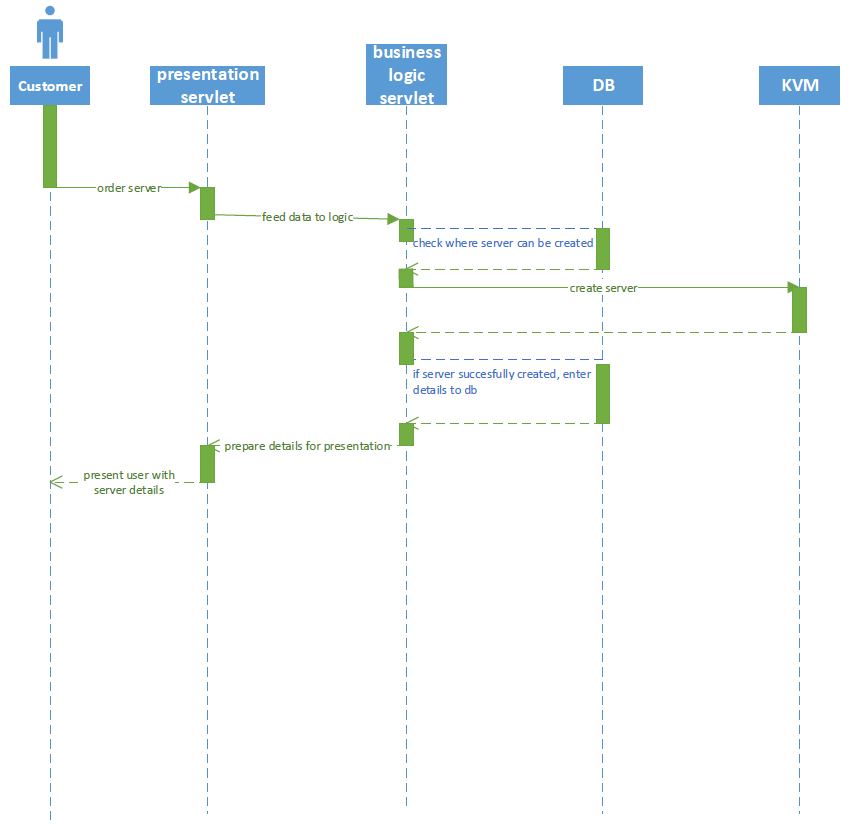
# 6. ERD

At the ERD we can see that the logical data structure has taken a physical place. Eventually when managed, one user has one or many VM’s that he/she can work with. All the data from the VM will be managed from there.

Macintosh HD:Users:kjellzijlemaker:Downloads:ERD.pdf

# 7. VM Management

The VM management is a bit like the server tier model. This particular case for the VM management is creating a new VM. As seen below, the user will need to go to the presentation servlet for ordering a server. When done, the business logic makes sure that this data will be processed. The business logic will then check if the server can be created, if it can be created (conditions). If done and everything goes well, the servlet will make a request to the KVM to create the server. If a response is given from the KVM, the new information will be written to the database. Of course, all the data will be presented to the customer now.

****

# 8. IP Plan

Seeing as Plaintech has at least 50.000 customers, with at least 1 server each, there must be at least 65.000 ip addresses available in the network infrastructure. As Plaintech expects a 100% growth within 5 years, even this 65.000 ip addresses is not enough. This is a massive amount, and must be segmented properly to ensure the network won’t be suffering from congestion and traffic does not slow down.

Subnetting

To be able to use all of the ip addresses required, we need a class A network with several different subnets. A class A network has a total maximum number of 16,777,216 addresses minus 2.

There should first of all be a division between the customer servers and the virtualised servers that Plaintech uses for their own business ends (ie, webserver, databases etc). On top of that, both of these ranges should be separate from the range of physical servers on which KVM runs.

As the servers will be put in 19" racks with a height of 47U, there will be 20 servers with roughly 100 virtual machines each in a rack. This means that each rack needs about 2000 IP addresses. 2000 IP addresses equals a /21 network per rack. The initial situation will require 30 racks each with their own /21 subnet.

Data-network

All of the physical hosts should be interconnected to the SAN on a second, completely separate, network. This means that all 556 initial hosts should be connected to all initial SAN devices. This network is entirely connected with Fiber optics, meaning that each physical server has to be connected to the SAN via Fiber HBA. On the following diagram, the SAN is depicted as a cloud network without IP addresses. This is to ensure that the concept of a SAN remains true to its nature as the disks are shared at block level instead of being shared at the filesystem level.

Additionally, since it's on fiber optics and ip-less, there is no encapsulation of any data between the SAN and the physical servers, which further increases the throughput of all data traffic.

