Value Added Services and Content Platforms

M.Sc. Thesis Report

Author: Adrian Mahdavi d98-ama@nada.kth.se
Date: 2003-06-25

Academic Advisor: Professor Gerald Q. Maguire Jr.
Department of Microelectronics and Information Technology
Royal Institute of Technology, Stockholm, Sweden

Industrial Advisor: Alexander Pigall, Tele2 AB

Abstract

Value-Added Services and Content Platforms (VAS and Content platforms) was carried out with in a group with same name at Tele2 AB in Kista, Stockholm. This group is responsible for network design, capacity planning, dimensioning, Acceptance testing (ATP test), and introducing of new functionality in Tele2's VAS-platforms.

Acceptance testing is performed on new devices (servers and other network components) in order to verify their capacity and performance guaranteed by their manufactures. Every platform has a guaranteed upper bound performance (based on the license a buyer has paid for), measured by different approaches. For instance for Short Message Service Center (SMSC) platforms, the measurement is based on the maximum number of SMS messages processed per second (SMS/sec), for Multimedia Messaging Service Center (MMSC) platforms the metric is the maximum number of MMS messages processed per second (MMS/sec), and for WAP Gateways it is the maximum number of WAP Transactions Per Second (TPS).

This M.Sc. thesis project involved creating two graphical load generators for load testing of SMSC and MMSC platforms. These application-programs are not allowed to occupy unnecessary resources, or cause additional traffic on the radio network (when they are deployed), but they must be powerful enough in order to send and receive traffic in order to derive statistical data about the system's performance. This data will be used for behavioral analysis of these systems, and finally for verifying the guaranteed capacities. These tests are very important and decisive for service providers, who want to be able to offer good quality of service, guarantee availability, and offer reliability.

In order to measure the performance and verify the guaranteed performance, two main scenarios were of great importance:

- Sending 5 messages per second during a interval of 5 minutes. This case will simulate a TV-contest in which the TV audiences submit messages to a predefined number in order to join the contest.

- Sending 3 Multimedia Messages per second during 30 minutes (for the MMSC performance measurement), and 7 SMS-messages per second during 120 minutes (for the SMSC performance measurement). This case attempts to simulate the traffic that will be generated in the minutes before and after Christmas or New Year.

For behavioral analysis and performance measurement of the MMSC and the SMSC platforms an Open Queueing Network model is employed. In this model each server system is considered as a network, consisting of nodes, where each node represents one component inside the system. By considering each node as a single-server queueing system we can take advantage of queueing theory in order to drive several performance results.

Acknowledgements

First I would like to thank Tele2 AB, especially my industrial adviser for giving me the opportunity to perform my thesis project and for their support and helpfulness.

Special thanks goes out to my academic supervisor and examiner Professor Gerald Maguire who has shown great interest, given me suggestions, and guided me throughout the project.

Special thanks also goes to my family who have been very supportive throughout my education at KTH.

Table of Contents

List of Figures

List of Tables

Acronyms

AMQ	Active Message Queue
AMR	Adaptive Multi-Rate
BHSM	Busy Hour Short Messages
BBS	Base Station System
EI	External Interface
ESME	External Short Message Entity
GSM	Global Standard for Mobile
GPRS	General Packet Radio Service
HLR	Home Location Register
HTTP	Hypertext Transfer Protocol
MAP	Mobile Application Part
MM	Multimedia Message
MMS	Multimedia Messaging Service
MMSC	Multimedia Messaging Service Center
MS	Mobile Station
MSC	Mobile Switching Center
NP	Number Portability
PDU	Protocol Data Unit
PLMN	Public Land Mobile Network
RFC	Request For Comments
SFE	Store and Forward Engine
SMIL	Synchronized Multimedia Integration Language
SMPP	Short Message Peer to Peer
SMS	Short Messaging Service
SMSC	Short Messaging Service Center
SMTP	Simple Mail Transfer Protocol
SS7	Signaling System 7
TPS	Transaction Per Second
3G	Third Generation
3GPP	Third Generation Partnership Project
UMTS	Universal Mobile Telecommunication Systems

VAS	Value Added Services
VLR	Visitor Location Register
VMS	Voice Mail System
WAP	Wireless Application Protocol
WSP	WAP Session Protocol
WAP GW	WAP GateWay

1. Introduction

System providers in the telecom industry today, are designing their infrastructure as a multi-layer application architecture. The major driving force for this is, of course, the Internet and the services that are rapidly being made available. One question that is raised is whether the hardware applications can handle the load, generated by a large number of simultaneous users, both with respect to performance and quality of service (QoS). Applications execute in a continuously changing environment with increased end user activity, application upgrades, new hardware, etc. In this context performance assurance of scalability and quality of service become unavoidable issues that should be considered carefully. In order to minimize the cost of correcting performance and scalability problems, these problems must be found as early as possible. Finding a performance or scalability problem during the deployment of an application could have a cost that includes new application and/or hardware upgrades, and loss of profit. After application deployment, quality, availability, and performance assurance should be maintained, by monitoring response times and availability from a user perspective. Monitoring response times from a user's perspective could also give early indication that the infrastructure needs more resources. By monitoring response times it is possible to see whether the resources are reaching their limits, or if application components have scaling problems, memory leakage, etc.

The purpose of this document is to give a more detailed description of the project, its components, specifically the Multimedia Message Service Center and Short Message Service Center, and the protocols involved. It will also provide essential information about how to design and create load generator applications that can be used to stress test platforms such as an MMSC and SMSC. The remainder of this report is structured as follow:

Section 2 *Project definition* describes the project and its goals.

Section 3 *Audience* contains information about the basic knowledge, essential for understanding this report.

Sections 4 *Multimedia Messaging System*, and 5 S*hort Message Service* (and their subsections) present the necessary background for the remainder of the report. These sections are essential for those who fully want to be able to understand the report.

Section 6 *Development and Evaluation* (and its subsections) contains important issues about queueing theory, software-development, and evaluation-results.

Section 7 *Future works* is devoted to areas in which more work should or shall be done. This section also includes aspects that can be considered in order to improve the software-applications and testing results.

Section 8 *Conclusion* presents the conclusion of the thesis work, including the final results of the MMSC and SMSC load testing.

2. Project definition

The goal of the project was to facilitate acceptance testing and performance testing, this involved designing and implementing Load Generators that could be deployed for testing of systems such as an SMSC or MMSC. The Load Generators enable us to measure how well these servers can handle large numbers of concurrent users. Scalability issues, such as response time and processing bottlenecks, are also of importance and can be examined.

The Load Generators generate traffic, which we use to monitor the response time from a user perspective, and to drive statistical data about the system's performance. By making use of these data, it is then possible to characterize the behavior of these systems in detail, and through this to identify bottlenecks. It is also very important to verify whether these platforms meet the manufacture's guarantees when processing large numbers of requests.

Another aspect is that hardware behaves differently with different parameter settings and configurations. The aim of the operator is to find an optimal configuration and set of parameter settings in order to be able to predict how the system will behave under the expected load. Another important issue is, ensuring that the generated load (SMS or MMS messages) from the load-generators will not actually be transmitted since these platforms are connected to the operational net and it is not desirable to occupy resources or to cause problems for paging customers.

3. Audience

This document is meant for students, (who have knowledge of Telecommunication or Computer architecture), designers, system developers (who are new in this area), and others, who are interested in understanding and creating MMS respectively SMS applications. The aim of this document is to give the reader knowledge of:

1. Multimedia Messaging Services, protocols, and hardware involved in an MMS network.

2. Hardware and components inside an MMSC server, and how they interact.

3. Short Message Services, protocols involved, and the architecture of an SMS network.

4. Hardware and components inside an SMSC, and their interactions.

5. How different components of an MMS – or an SMS network interact with each other, using different types of transport protocols.

6. How external applications interact and utilize functionality and services offered by these networks.

7. Performance evaluation techniques.

8. Utilization of queueing theory for behavioral analysis.

If the reader is familiar with the following areas then he/she has the required knowledge for understanding the rest of this report.

- Internetworking
- Computer communication and computer networks
- Operating systems
- Queueing theory

Otherwise knowledge of following areas is essential (sources for this information are cited):

- TCP/IP protocol stack, its functionality, possibilities and boundaries [TCP/IP]
- WAP protocol stack and how it collaborates with the TCP/IP protocol stack [WTI]
- WAP architecture and protocol specification [WAPA]
- SMTP protocol specification [RFC0821]
- SMPP protocol specification version 3.4 [SMPPD]

4. Multimedia Messaging System

4.1 Definition

A Multimedia Messaging System (MMS) is a non-real-time, IP-based, message delivery system. It is comparable to other existing messaging systems such as e-mail and SMS. MMS provides messaging capabilities for the delivery of multimedia messages, composed of text, pictures, audio, and other media types, by utilizing the network capabilities provided by Internet, GPRS, 3G and GSM as well.

4.2 MMS Network Architecture

The figure below gives an abstract picture of a MMS network and potential components. The connection between components utilizes the Internet Protocol (IP) and its associated messaging capabilities, as described in [WAP205] and [TSG].

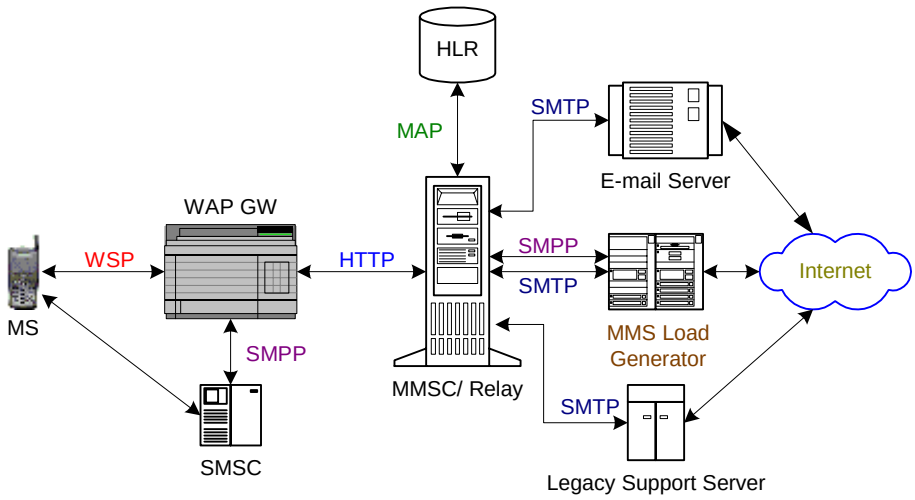

Figure 1: MMS network architecture

4.2.1 Multimedia Messaging Service Center (MMSC)

The MMSC provides MMS message-transfer between sender and receiver. It is responsible for storing and handling incoming and outgoing MMS messages. The MMSC is also responsible for transmission of MM messages between different messaging systems such as e-mail server, Content Converter, and even other MMSCs, using the SMTP protocol, as presented in [SYMSOFT].

The MMSC is a notification delivery system (unlike SMSC, see chapter 5), which means that the MMSC does not directly send the MM messages to the receiver. But rather MM messages reside on the MMS server. Before delivery of each MM message the MMSC submits a notification (binary SMS) saying that there is a message available to download. The recipient chooses whether or not to download the message at that time or later. If the recipient has not downloaded the message after a predefined period of time, the message will be discarded by the server (MMSC).

The MMSC uses a dedicated SMSC for delivery of notifications. By default all notifications (binary SMSs) will be submitted to their destinations, through a WAP gateway, by this SMSC. It is also possible to configure the MMSC to use the SMSC directly (i.e. not through the WAP gateway). This option is used by the MMS Load-Generator to remove the need of using the real SMSC and the radio network.

The MMSC architecture [3GPPTS] is specified by 3GPP and includes the following elements:

MMS Proxy-Relay is responsible for interaction between, Mobile Station (user agents), other messaging systems such as Voice Mail Server (VMS), and even other MMSCs. It is also responsible for communication with the MMS Server (see below).

MMS Server is responsible for storing MM messages while trying to make contact with potential recipients.

4.2.1.1 Nobill MMSC

The specific Multimedia Messaging Service Center (MMSC) used on Tele2's Multimedia network is Sysmsoft's Nobill MMSC. It is fully IP-based, and complies with Third Generation Partnership Project (3GPP) and WAP Forum standards. It is compatible with other IP-based technologies such as GPRS and UMTS.

The hardware platform of the MMSC is a SUN V880 server, consisting of a single 900 MHZ CPU, 4 GByte RAM, several 72 GByte disks, and 2 Ethernet interfaces [SYMHARD]. The selection of hardware-components was based upon an integrity testing procedure, performed by Symsoft (Tele2 didn't receive any details about these tests).

The following figure shows the architecture of the Nobill MMSC as shown on page 7 in [SYMHARD].

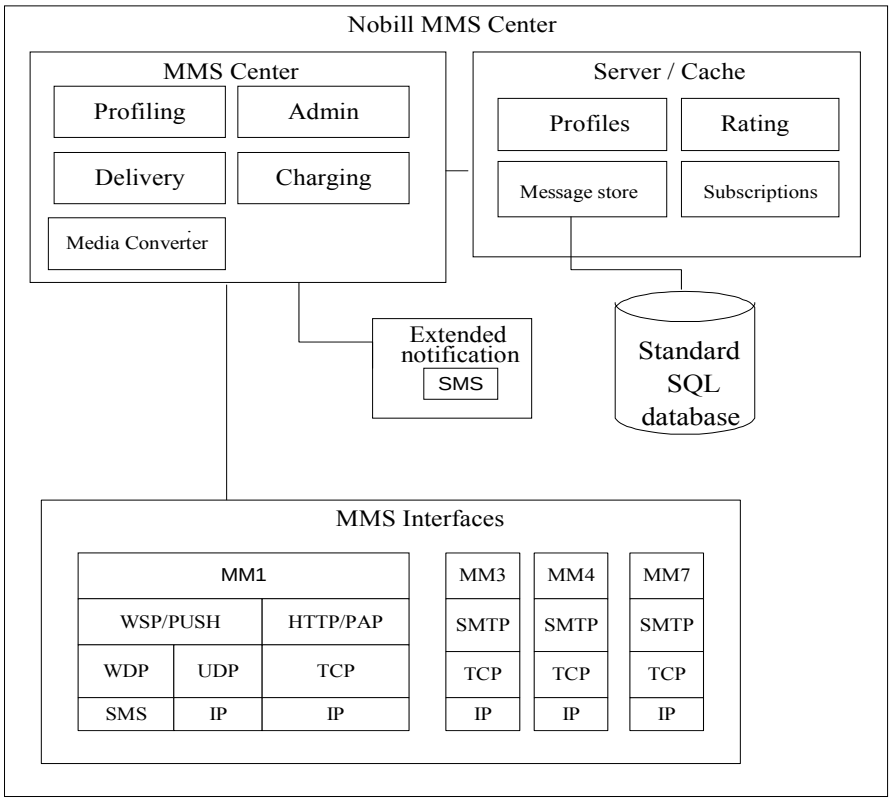

Figure 2: Nobill MMSC Architecture

Some explanation of the figure:

- MM1 is the mobile station (MS) to MMSC interface.
- MM3 is the MMSC to external messaging system (e-mail) interface.
- MM4 is the MMSC to MMSC interface.
- MM7 is the MMSC to VASP (Value Added Service Provider) interface.

4.2.2 WAP Gateway (WAP GW)

The WAP GW offers standard WAP services, needed for deployment of the MMS services. All multimedia messages and notifications, exchanged between MS clients and the MMS Proxy-Relay, pass (by default) through the WAP GW.

MM messages may be transmitted between MSs and the WAP GW through the Wireless Session Protocol's (WSP) methods, such as POST or GET. The messages are then transmitted over Hyper Text Transport Protocol (HTTP) to/from the WAP GW from/to the MMSC-Proxy.

All mobile traffic between MSs and the MMSC goes through an IP based, Gateway GPRS Support Node (GGSN).

4.2.3 Home Location Register (HLR)

When an MM message arrives at the MMSC, the MMSC interrogates a local database (Number Portability (NP) database, which contains MSISDN numbers of all operator-specific subscribers) to see if the destination number belongs to the local network operator or if it belongs to another operator. If the destination number belongs to the local network then the MMSC sends a notification, which is a binary SMS message containing source address, destination address, subject, and data, to an SMSC for further transmission to the destination mobile subscriber. The SMSC interrogates the HLR in the GPRS network, to find out where to send that notification. The HLR is a database that is used for permanent storage of subscriptions and service profiles.

If the destination address belongs to another operator, then the MM message will be sent to the MMSC of that operator via the MM4 interface.

4.2.4 E-Mail Server

The communication between the MMSC-Proxy and all external applications e.g. an E-mail server is via the SMTP protocol. MM messages destined to an e-mail address or MM messages destined to an MS (originated from an e-mail client) are examples of applications that directly utilize the SMTP protocol.

4.2.5 Legacy Support Server

If a subscriber wants to receive MM messages from another subscribers via an MMSC, then he/she must first register himself/herself with that MMSC (by sending an MMS message to the MMSC). The MMSC stores information about subscribers and therefore can forward MM messages to them. If you don't have a MMS capable handset, or if you have a MMS capable handset but still are not registered with the MMSC, then the MMSC cannot deliver MM messages to you (i.e. you are a legacy user). Instead the MM messages will be sent to a legacy server. The legacy server will send a legacy message (i.e. binary SMS message) containing an URL to you. You may use that URL to view your message via a web-browser.

Legacy server communicates with the MMSC via SMTP. Its primary task is to store MM messages and to send URLs to legacy users. Thus all subscribers get the opportunity of receiving and sending MM messages.

4.2.6 Voice Mail Server

The Voice Mail Server (VMS) is another example of an external application. As mentioned above the VMS communicates with the MMSC-Proxy via SMTP. Receivers get an SMS message telling that there is a voice message waiting. After connection to the VMS, voice messages encapsulated as MM messages can be downloaded by the receivers. The result is that voice messages can be delivered as packets, and they don't have to arrive in real-time while retrieval of voice mails requires a circuit connection to be set up and dedicated to that user.

4.2.7 MMS User Agent located at Mobile Station (MS)

The MMS User Agent is an application layer function that provides users with the ability to view, compose, and handle MM messages. This allows also operators to consolidate access to multiple applications from a single architecture (e.g. SMSC, Email, Unified Messaging, etc.).

4.2.8 MMS VAS Providers

MMS VAS Providers offer value-added services to the MMS users. In many ways MMS VAS applications behave like a fixed MMS User Agent. MMS VAS Providers are able to generate Call Data Records (CDRs) when receiving MM messages from the MMS Relay and when submitting MM messages to the MMS Relay.

4.3 Structure of the MMS PDU

In an MMS capable network, what actually is being sent, are MMS Protocol Data Units (MMS PDUs). The structure of the MMS PDU is specified in [WAP-209].

Each MMS PDU that is submitted from a sender to a receiver via an MMSC must contain two parts:

1. MMS header: contains a field with the name "**Content_type**" whose value depends on whether or not the presentation part exists. If the presentation part exists, then the value of this field will be set to "*application/vnd.wap.multipart.related*". Additionally there are two other fields "**type**" (defines the type of the presentation), which must be set to "application/smil", and "**start**" that defines the location of the presentation part, which must be set to "<0000>".

   Otherwise (i.e. when there is no presentation part) the value of the "**Content_type**" will be set to "*application/vnd.wap.multipart.mixed*".

2. MMS multipart message body: The message body part of the MMS PDU consists of a number of pages (or slide shows) where each page contains two or more regions (for example three, if there are text, image, and audio). These regions (i.e. text, image, and audio) are packaged as separate elements within the message body as described in [RFC2387].

   The first element of the multipart message body is the presentation part (if there is one). The presentation part contains instructions about the layout, ordering, and how the multimedia content should be presented on an MS. This arrangement is implemented by a presentation language called SMIL (see 4.3.1).

   The rest of the message body consists of headers with corresponding contents (body). The parameters "**Content-Type**" and "**Content-Location**" in each header provide information about the content type (which may be text, image, or audio) and the content identification (which is the name of the file).

The MMS PDU is placed in the content section of the WSP PDU, HTTP PDU, or SMTP PDU, depending on which transport protocol is being used (WSP, HTTP, or SMTP). If WAP is used (as shown in figure 3) the value of the field "**Content_type**" in the WSP header is set to "*application/vnd.wap.mms-message*". When SMTP is used then that parameter will be set to "*application/vnd.smtp.mms-message*".

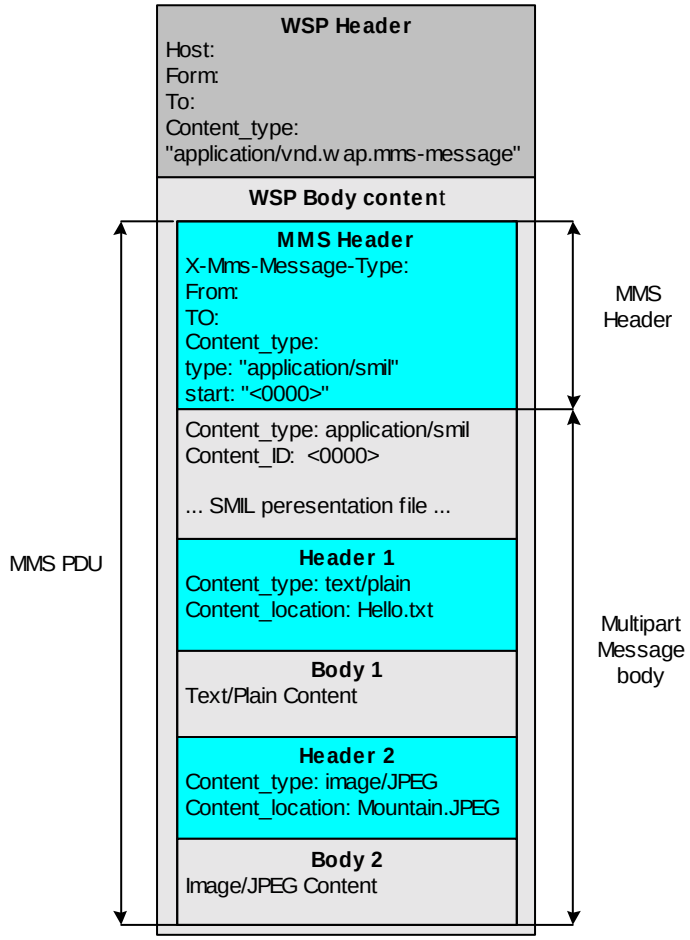

Figure 3: WSP Protocol Data Unit

4.3.1 Presentation of SMIL

The Synchronized Multimedia Integration Language [SMIL] defines the layout and ordering of MM messages. This presentation language is used to present MM messages on different terminals with different display capabilities and to synchronize the multimedia contents (if necessary).

Several companies, including Ericsson, Nokia, Motorola, Siemens, Logica, and others, have written a document called "MMS Conformance" ([MMSCON]) about the presentation language of MM messages supported by their mobile handsets.

According to this document the supported formats for the contents of MM messages are:

1. Text: US-ASCII, utf-8, and uft-16
2. Image: JPEG, GIF, and WBMP
3. Audio: Adaptive Multi-Rate [AMR]

For more details see [AMR] and [MMSCON].

4.4 Sending and receiving a multimedia message

For different tasks (requests/responses) different types of MMS PDUs may be used. The type of a PDU is indicated by the value of the parameter "**X-Mms-Message-Type**" in the MMS header (e.g. **X-Mms-Message-Type**: M-Send.req).

The information that must or may be passed within the MMS PDU depends on the parameter "**type**" (in the MMS header), as mentioned above. The MMS PDU header is mandatory, but the MMS PDU Body is optional, depending on the type of MMS PDU.

The following MMS PDUs are used in the sending and receiving scenario, described below:

A1. Message creation, and destination setting, on originating ESME
A2. *M-Send.req* (from originating ESME to MMSC)
A3. *M-Send.conf* (from MMSC to originating ESME)

B1. *M-Notification.ind* (from MMSC to MMS recipient, indicating that an MMS message is waiting)

Immediate message retrieval by the receiver:

C1. *WSP-GET.req* (from receiver to MMSC)
C2. *M-Retrieve.conf* (from MMSC to receiver)
C3. *M-NotifyResp.ind* (from receiver to MMSC)

The receiver is not active or chooses to download the MMS later:

C1. *M-NotifyResp.ind* (from receiver to MMSC)
C2. *WSP-GET.req* (from receiver to MMSC)
C3. *M-Retrieve.conf* (from MMSC to receiver)
C4. *M-Acknowledge.req* (from receiver to MMSC)

D1. *M-delivery.ind* (from MMSC to the originating ESME)

I will now describe the above scenarios in more detail. In this scenario a sending application sends an MMS message to a receiving application via an MMSC. If the

transmission of the MMS message from the MMSC application to the receiving application was successful then an Acknowledgement will be sent from the MMSC to the sender. The WAP Wireless Session Protocol (WSP) is used for transmission of MMS messages from a mobile handset to a WAP GW, and from the WAP GW to a receiving mobile handset [WAPWSP]. The HTTP is used for transmission of MMS messages between the WAP GW and the MMSC.

The following key steps summarize the scenario (from [WAP-206]):

A. Sending application:

A1. The message originator on a MS creates a multimedia message and addresses it to the receiver.
A2. The originator MS makes a WAP connection (via for example GPRS) to the MMSC and sends the message by using a WSP POST request. (This is the *M-Send.req*)
A3. The MMSC receives the message and sends an ACK to the originator over the same WAP connection by using WSP POST response indicating that the message was received. The sender's terminal displays "message sent". (This is the *M-Send.conf)*

B. MMSC:

B1. The MMSC stores the message and sends a SMS message to the addressed recipient, by using WAP PUSH (*M-Notification.ind)*, notifying him/her about the message.

C. Receiver:

C1. If the receiver is active and wants to accept the message, then he/she initiates a WAP connection (over for example GPRS) and uses a WSP GET request (*WSP-GET.req)* to retrieve the message from the MMSC.
C2. The MMSC sends the message to the receiver in the body of the WSP GET response (*M-Retrieve.conf)* over the same WAP session. The receiver's terminal displays, "Message received".
C3. The receiver acknowledges the received message using WSP POST (*M-NotifyResp.ind)* over the same WAP session.

But if the receiver doesn't fetch the MM-message, then that message will be kept in the MMSC for a predefined period of time (normally one week). When this period elapses that message will be discarded.

D. MMSC:

D1. The MMSC sends a SMS message to the originator by using a WAP PUSH (*M-delivery.ind)*, informing that the message was delivered. Sender's terminal displays, "Message delivered".

The following diagram illustrates this scenario:

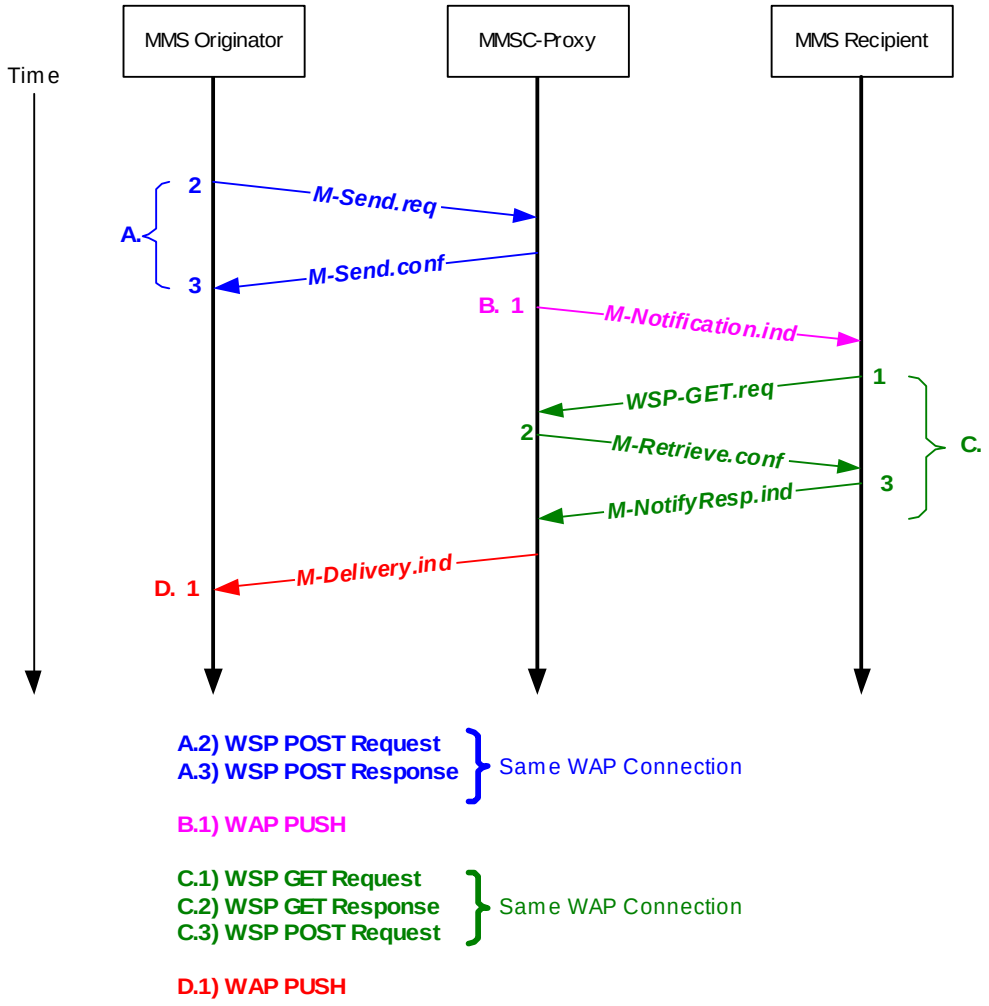

Figure 4: Sending and Receiving an MMS message

4.5 External applications

All applications physically outside the MMSC are called external applications. External applications can be categorized as Originating applications, Terminating applications, and Filtering application.

4.5.1 Originating application

These kinds of applications are the source of multimedia messages. They create and send multimedia messages to an MMSC.

4.5.2 Terminating applications

These applications receive MMS messages from an MMSC. The terminating applications, depending on their implementations, can be either synchronous or asynchronous.

A synchronous application can handle just one message at the time, i.e. when such an application receives a message, it either consumes the message, or sends it back to the MMSC, before accepting a new message. While asynchronous applications are those that can handle several messages at the same time.

4.5.3 Filtering applications

This kind of application receives, for example, a MM message from the MMSC, processes it, and then sends it back to the MMSC for further processing. Depending on how filtering applications are implemented, they are either synchronous or asynchronous as described above (see 4.5.2).

4.6 Related Works

Even though the concept of the MMS and its services are not new, deployment of MMS is still new. This is one of the major reasons why there are few applications or even documents about MMS applications. Despite increasing number of MMSC servers in use and increasing number of subscribers, little is definitively known about the performance of these systems.

Recently Vodafone and Tele2 introduced MMS services to their customers (I was actively involved in the introduction of the MMS services by Tele2). Even though the deployment of the MMS services has taken long time, there are still some interoperability problems between these two company's MMS networks.

Several companies, including Ericsson, Nokia, Motorola, Siemens, Logica, and others, have written a document called "MMS Conformance" ([MMSCON]) about format, size, coding language, and presentation language of MM messages supported by their mobile handsets. It is thought that, a conformant MM message can be received and presented on the display of all MMS capable mobile stations, manufactured by these companies.

5. Short Messaging Service (SMS)

5.1 Definition and Background

Short Messaging Service (SMS) is an accepted wireless service, which was introduced in the digital wireless networks, known as the Global Standard for Mobiles (GSM) in 1991. SMS provides a means of transferring Short Messages (SMSs) of up to 160 octets/characters from an External Short Message Entity (ESME) to a mobile subscriber via an SMSC.

SMS utilizes a point-to-point delivery technique, which is used for transmission of short text messages between cellular terminals (MSs) themselves and ESMEs such as e-mail, paging, and voice-mail systems.

SMS guarantees reliable delivery of short messages by the network, which means that errors and temporary delivery-attempt failures (i.e., when the receiving MS is not active) are identified by a Short Message Service Center (SMSC). The SMSC is a middleman between a sender and a destination. The SMSC acts as a store-and-forward system for SMS messages and takes care of delayed transmission of short messages when the destination MS (Mobile Station) is not active. This means that short messages, whose recipients are not available (active), will be stored temporarily in the SMSC and will be transmitted when the recipients are active again.

SMS is characterized by out-of-band packet delivery and low-bandwidth message transfer, which results in a highly efficient means for transmitting short bursts of data [IEC]. However, the delivery delay may be long.

5.2 SMS advantages

The benefits of the SMS are based upon the fact that users are able to send and receive text messages anywhere with their mobile handsets. The advantages are:

- Delivery of message errors or/and message notifications to senders.
- Reliable transmission service, but with a very long tailed distribution of delivery times.
- Provision of value-added services such as e-mail, voice mail, stock and currency quotes, etc. These services are in used today, and additional services will be introduced in the future.
- Integration and collaboration with other external applications such as Voice Mail System (VMS) and MMS.

The SMS also provides a number of service elements that effect the performance of the SMSC. These are:

- Message Expiry time: When configuring an SMSC, an expiry time may be defined, that tells, how long a message can be stored in the SMSC before it is discarded gracefully. The SMSC will store short messages until they are delivered successfully or the expiry time runs out.
- Priority: This information is provided by ESMEs to differentiate urgent messages from normal messages. Urgent messages have higher level of delivery priority than the normal messages. The SMSC processes messages with higher priority before those messages with lower priority.

The services offered by SMS are specified by the European Telecommunications Standards Institute (ETSI) GSM Standard and by the Telecommunications Industry Association (TIA) IS41 Standard.

5.3 SMS Network Elements and Architecture

An overview of the SMS Network architecture and components (elements) involved (based on GSM) is shown below (in figure 5). The architecture is presented at a general level, independent of the exact interfaces and protocols used between the network elements.

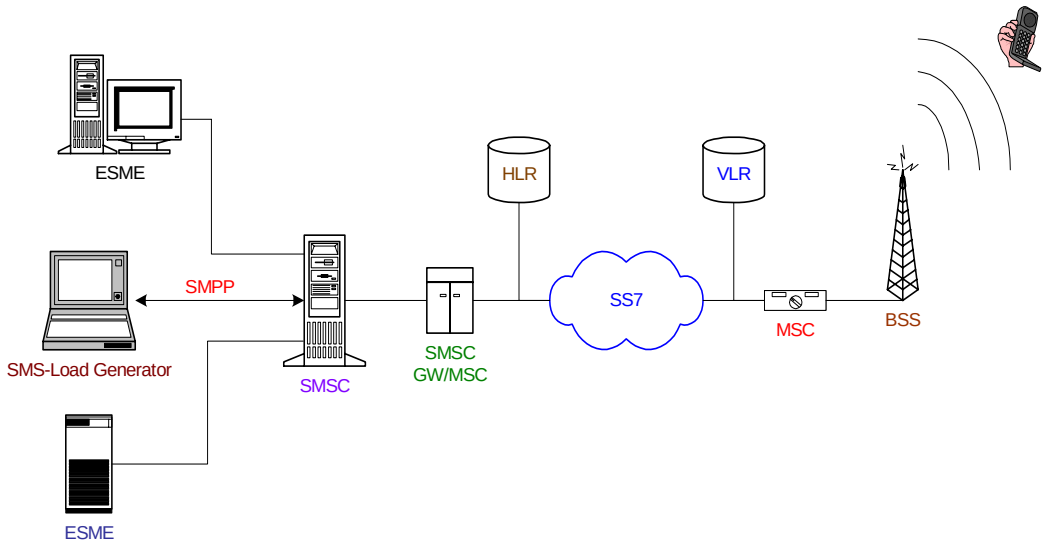

Figure 5: SMS network architecture

5.3.1 External Short Message Entity (ESME)

An ESME is an application or a non-mobile device that may send, receive, or both send and receive short messages to/from an SMSC (see 5.3.2).

The ESME may be a network-connected device, a mobile phone, or just a client based application. Examples of ESME include:

- Voice mail system (VMS) is responsible for receiving, storing, and forwarding of voice messages, sent to subscribers, who did not answer the call when it was made.
- E-mail server, which is used to receive or/and send e-mails to SMS capable terminals through an SMSC. The SMSC must support configuration and interconnection with e-mail servers.
- Client Application (e.g. my Load Generator), which is an application residing on a computer (e.g. a PC) for sending and receiving short messages to/from the SMSC.

5.3.2 Short Message Service Center (SMSC)

An SMSC is responsible for relaying, and storing-and-forwarding of SMS messages between external entities and Mobile Stations (MSs). It must be scalable in order to support growing use and to introduce new services for subscribers.

The SMSC must have high reliability, large message capacity, and high message throughput in order to handle growth.

5.3.2.1 Intelligent Short Message Service Center (ISMSC)

The Intelligent Short Message Service Center (ISMSC), manufactured by Comverse Network Systems, is the SMSC that is used in Tele2's SMS network. ISMSC provides a means of relaying short text messages and icons between mobile stations and the SMSC of a digital cellular network [COMTECH].

The test ISMSC platform, used in the performance testing, is built on a Multi-purpose Pentium Assembly (MPA) platform consisting of the following components [COMOP]:

1. Processor: Dual-Pentium III with 512 MB of RAM and a 500-MHz CPU.
2. Hard Disk drive: The hard disk drive in the server contains, UnixWare OS, and Unix applications. The hard disk provides 9.1 GByte of storage.

ISMSC configurations are classified as either medium or large, based on the following characteristics:

- Medium Capacity: 110K Busy Hour Short Messages (BHSM).
- Large Capacity: 235K BHSM (This configuration is used by Tele2 AB).

Figure 6 shows the architecture of the ISMSC. The architecture is independent of the exact interfaces and protocols, used between the ISMSC elements.

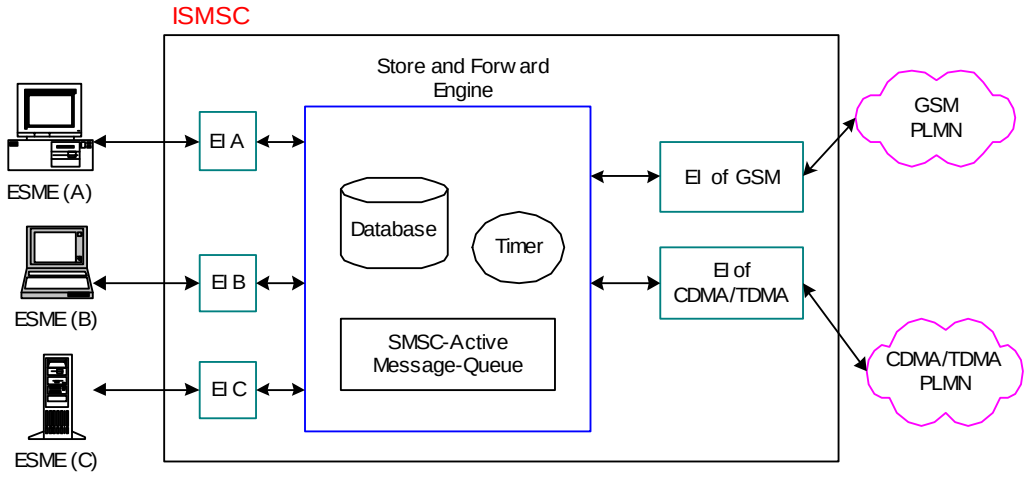

Figure 6: The Architecture of the ISMSC

Some comments on the figure:

- The main component "Store and Forward Engine (SFE)" performs the store-and-forward procedure and interacts with a number of different external interfaces (EIs). Its main elements are a database, a retry mechanism (based on a Timer) and an active message queue.
- Each external Interface (EI) is responsible for interacting with an ESME, and mediating between the ESME and the SFE. There is always a specific EI for each ESME (e.g. an e-mail server).

5.3.3 SMS Gateway/Mobile Switching Center

The SMSC GW/MSC is responsible for receiving short messages from the SMSC and searching for routing information in the Home Location Register (see section 5.3.4), and delivery of the messages to appropriate Mobile Switching Centers (see section 5.3.7).

5.3.4 Home Location Register (HLR)

The HLR is a database that is used for permanent storage of subscriptions and service profiles. The HLR provides the SMSC with routing information about the destination MSC, which in its turn serves the addressed mobile station.

When the receiver's mobile handset is not active, then message delivery from the SMSC will fail. When the receiver's handset is active again, the HLR will inform the SMSC and the message can be transmitted.

5.3.5 Signaling System 7 (SS7)

SS7 is a powerful multi-layered signaling protocol (which is used to provide and to maintain communication services) by which all components of a phone network exchange information. As phone users we cause exchange of signals by network elements, using the SS7. For example dialing phone numbers, getting dial ton, sending/receiving voice messages, getting busy tone and so on.

This part was beyond the scope of this document, for more information see [NW].

5.3.6 Visitor Location Register (VLR)

The visitor location register is a database that contains temporary information about visiting subscribers. The VLR provides the MSC (see section 5.3.7) with routing information about the visitor's location and makes it possible for the visited MSC to offer services to visiting subscribers.

5.3.7 Mobile Switching Center (MSC)

The MSC is responsible for switching functions. It controls and switches calls between different mobile stations and subsystems. The MSC delivers short messages to the destination MS through a proper BSS (see section 5.3.8).

5.3.8 Base Station System (BSS)

The BSS takes care of the transmission of messages (via radio signals) between the MSC and the destination MS.

The BSS consists of a Base Station Controller (BSC) and Base Transceiver Stations (BTSs) each of which services a "cell". Each BSC controls one or more BTS and takes care of resource assignment, when a cellular terminal moves from one cell to another cell.

5.4 Mobile Application Part (MAP)

MAP defines a number of operations, necessary to support end-to-end SMS functionality. These operations can be summarized as follow [IEC]:

- Routing Information Request: The SMSC must receive routing information from the HLR, to determine the serving MSC for the mobile device at the time of the delivery attempt. This information is needed before a message can be delivered by the SMSC.
- Point-to-Point Short Message Delivery: After obtaining the MSC's address, the short message can be delivered directly to that MSC for further transmission to the specified MS. The outcome of this operation may be success or failure (caused by one of several possible reasons).

- Short Message Waiting indication: If the delivery of a message to its destination MS fails due to the absence of the MS, then the message will be stored in the SMSC. The SMSC requests the HLR to notify the SMSC when the indicated MS is registered (active again).
- Service Center Alert: When the destination MS is registered by the mobile network, the HLR will send a notification to the SMSC informing it that the specified MS is now available. In this case the SMSC submits the message to the destination MSC for further transmission.

5.5 SMS Message Delivery

For the transmission of short messages between ESMEs and an SMSC the Short Message Peer to Peer protocol (SMPP) is use. For more information about this protocol and its functionality see [SMPPD].

SMS utilizes two different types of Point-to-Point communication services:

1. Mobile Originated Short Messages (MO) are messages initiated from an MS, destined to another MS or an ESME via an SMSC. I will explain the later case in more details in section 5.5.4.
2. Mobile Terminated Short Messages (MT) are messages that are received and terminated by MSs. The originator of these messages is either an MS or an ESME. You can read about the later case in section 5.5.4.

5.5.1 SMPP Protocol Definition

SMPP [SMPPD] is an application layer protocol, which operates on the top of TCP/IP. The SMPP utilizes transport functions (such as reliable delivery, packet encoding, sliding window, flow control, and error handling), offered by TCP/IP. SMPP provides a communication interface between ESMEs, outside the mobile network and an SMSC. This protocol defines operations needed for sending and/or receiving of SMS messages. It also defines the message format, and the data that is exchanged during an SMPP operation.

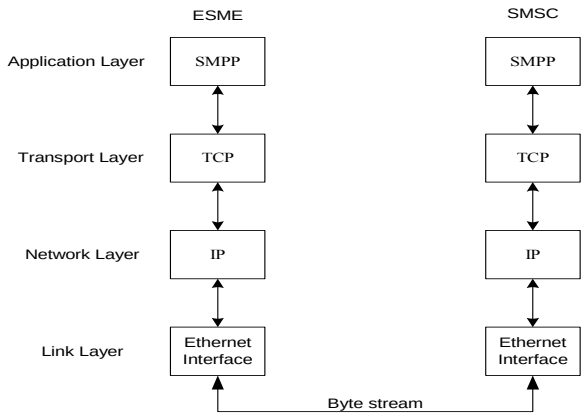

Figure 7: SMPP Protocol Stack

SMPP is based on the exchange of Request- and Response Protocol Data Units (PDUs). Thus each SMPP operation consists of a Request PDU and the corresponding Response PDU. A unique sequence number is used in order to identify each SMPP Request PDU with its corresponding Response PDU. The corresponding SMPP Response PDU uses the same sequence number as the Request PDU, thus making identification possible.

The exchange of SMPP PDUs between an ESME and an SMSC needs one network connection and one SMPP session. The message transmission may be performed via three different types of bind operations:

1. Transmitter: when the sending application binds to the SMSC as transmitter, SMS messages are sent from that application to the SMSC (see section 5.5.2). The sending application can receive responses from the SMSC.

2. Receiver: an application that binds to the SMSC as receiver, can receive SMPP PDUs from the SMSC. The SMPP PDUs originate from an ESME, the SMSC itself, or an MS. The receiver can still send responses to the SMSC (see section 5.5.2).

3. Transceiver: when an application binds as tranceiver, PDUs can be sent in both directions (Duplex) over the same session. For this only one network connection and one SMPP session are needed (see 5.5.2). I will use this bind operation for my solution.

5.5.2 SMPP Session

An SMPP Session between an ESME and an SMSC is always started by the ESME, first by establishing a network connection with the SMSC, and then sending a "*bind_request*" PDU to the SMSC, where the "*request*" may be one of following:

- Transmitter
- Receiver
- Transceiver

The SMSC must respond, by sending an acknowledgement "*bind_request_resp*", where the "*request*" is as above (see the figure 8), or the SMSC can reject the request by sending a "*generic_nack*" PDU containing an appropriate error status. For a complete explanation of all commands and parameters see [SMPPD].

During a session many requests and responses may be sent, for instance for each request, initiated by the ESME, the SMSC must respond with the corresponding response and vice versa. The corresponding response may be a generic negative acknowledgement (i.e. "*generic_nack*" PDU), which indicates that the header of the submitted SMPP PDU was invalid. Generally a "*generic_nack* " response is returned in the following cases:

- Invalid "**command_length**": If the receiving SMSC detects that the value of the "**command_length**" field is either too short or too long, it should assume that the message is corrupt. In such cases a "*generic_nack*" must be submitted to the message originator.

- Unknown "**command_id**": If the value of the "**command_id**" field is unknown or invalid, a "*generic_nack*" PDU must also be sent to the message originator.

The session will terminate when the ESME sends a "*unbind*" request as it is shown by the figure 8. The SMSC responds with the corresponding "*unbind_resp*", then terminates the session. This procedure is also implementable in opposite direction.

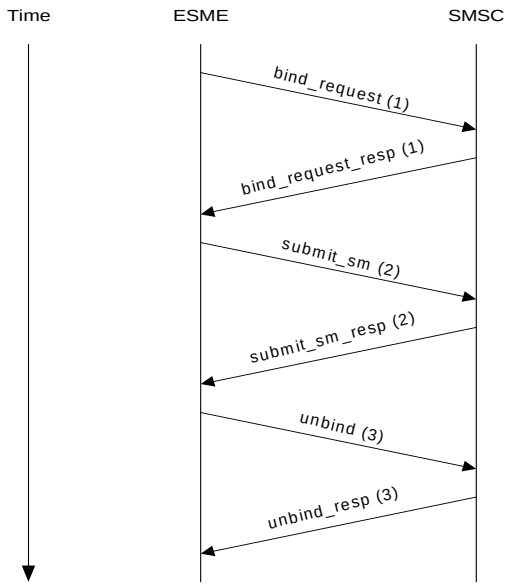

Figure 8: SMPP Session (initiation, submission, and termination)

When the SMSC receives a short message destined to an ESME, it must first establish a network connection with that ESME (see the figure 9). For binding to that ESME and initiating the session the SMSC must send an "*outbind*" request to the ESME. The ESME will answer with a "*bind_receiver*" request, which will be acknowledged by the SMSC with a "*bind_receiver_resp*". Now the session is established and the SMSC may deliver the message by issuing "*deliver_sm*" request. When the ESME receives the message it acknowledges that message by sending an "*deliver_sm_resp*" response.

The session can be terminated at any time as mentioned above (the figure 9 shows this procedure). As we see from the figure 9, the ESME sends an "*unbind (3)*" request in order to terminate the session. The SMSC responds with an "*unbind_resp (3)*" and

terminates the session. If there are additional messages to the ESME, the SMSC will submit them before transmission of the "*unbind_resp (3)*" response.

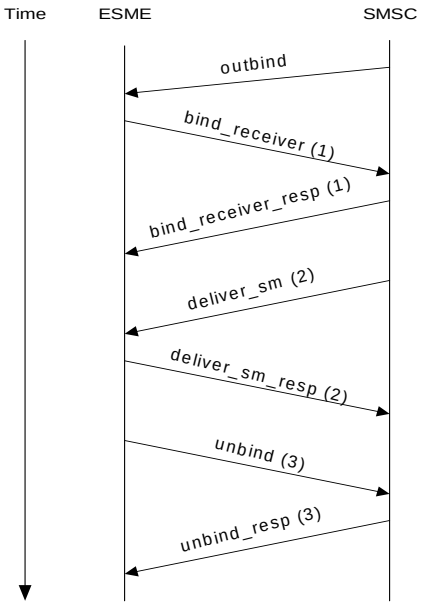

Figure 9: SMPP messages sent from SMSC to ESME using Outbind

5.5.3 SMPP messages destined to ESME via SMSC

For delivery of SMPP messages from an SMSC to an ESME, the ESME must first establish a network connection to the SMSC and then bind to it by sending either *bind_receiver* or *bind_transceiver*. After initiation of the session (as described above), the following SMPP PDUs may be sent from the SMSC to the ESME:

- *deliver_sm*
- *data_sm*

The receiving ESME must acknowledge, by sending the corresponding SMPP PDU response, which is one of the following:

- *deliver_sm_resp*
- *data_sm_resp*

The acknowledgement from the ESME to the SMSC must contain the same sequence number as the sequence number used in the receiving PDU. The acknowledgement also contains information about the delivery status. If the PDU was corrupted then an error must be sent to the SMSC.

The exchange of SMPP requests and responses between the SMSC and the receiving ESME may occur:

- Synchronously: Just one request at time. (See section 5.5.4)
- Asynchronously: Multiple requests at time. (See section 5.5.4)

This exchange follows the same time line as shown in Figure 9.

5.5.4 SMPP PDUs sent from ESME to SMSC

Before an ESME sends short messages (SMPP PDUs) to an SMSC it must first establish a network connection to that SMSC and then bind to it by initiating a *bind_transmitter,* or a *bind_transceiver* session. Once the session is started the ESME may send one of the following PDUs:

- *submit_sm*
- *data_sm*

Each SMPP message PDU, sent from the ESME must be acknowledged by the SMSC by sending the corresponding response (containing the same sequence number as the sequence number inside the received message PDU, delivery status, time, etc.)

In addition to above-mentioned operations there are a couple of operations that may be requested by the sending ESME that use the sequence number, contained in the corresponding response (i.e. identical sequence number in each pair of request and response PDUs), sent by the SMSC. These additional operations are:

- *query_sm*: asking about the status of the previous sent PDU
- *replace_sm*: replace the previous sent PDU
- *cancel_sm*: cancel or discard the previous PDU

Note that the exchange of the SMPP requests and responses (i.e. the session) may occur synchronously or asynchronously. In the synchronous mode the ESME sends just one request at time and waits for the corresponding response before sending a new one. In the case of asynchronous mode, the ESME may send multiple requests, before getting any responses from the SMSC. The SMSC will subsequently send responses using the corresponding sequence number. Note that identification through sequence numbers is a very important issue here.

The following figure illustrates the procedure. For more information see [SMPPD].

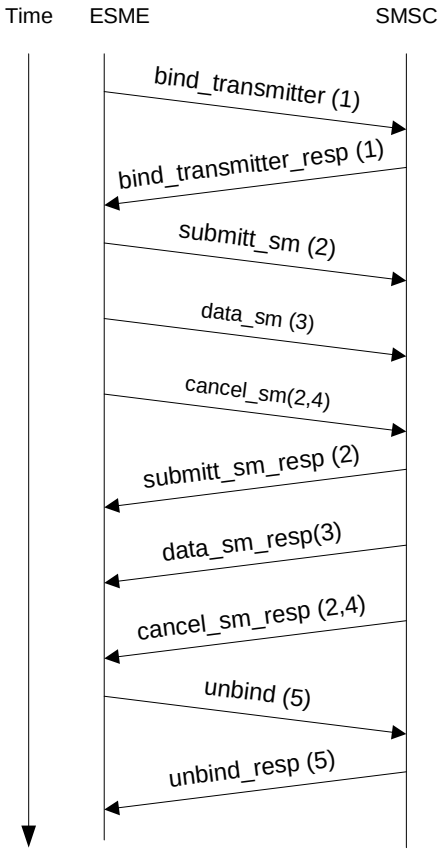

Figure 10: SMPP PDUs from ESME to SMSC in asynchronous mode

5.5.5 Format of SMPP PDU

The SMPP PDU consists of a mandatory header and an optional body as shown in the following figure.

1. A mandatory header: Every SMPP PDUs has a mandatory header. The header contains four different fields, where each field is 4-octets (32 bits) long. A short explanation of each field is as follow [SMPPD]:

    1.1. **command_length** is 4 octets long. This field indicates the total length of the PDU packet, i.e. both the header and the body.
    1.2. **command_id** identifies the type of each SMPP PDU (e.g. *submit_sm* or *data_sm*) uniquely. The SMPP request's **command_id** is identical to the SMPP response's **command_id**, but with bit 31 set. The complete list of all **command_id**s can be found in ([SMPPD], section 5.1.2.1 page 110).
    1.3. **command_status**: This field is only relevant in the SMPP response PDU and indicates the delivery status (success or failure). This field is assigned the value NULL in the request PDU (see [SMPPD], section 5.1.3).
    1.4. **sequence_number**: The value of this field is used to associate requests and responses, i.e. to specify which response belongs to which request. The

originator of the SMPP PDU must assign this value. The allowed **sequence_number** range is between 0x00000001 and 0x7FFFFFFF.

2. An optional body: The body of an SMPP PDU, if present, consists of two parts:

   2.1. Mandatory parameters: If presence of a body is necessary, then all the parameters in this part of body must be valid. The lists of parameters differ depending on the value of the **command_id** field (mentioned above) of the SMPP PDU header. Examples of mandatory parameters include **system_id**, **password**, **source_addr**, **dest_addr**, etc. For more information see [SMPPD, section 5.2 page 116].

   2.2. Optional parameters: The list of optional parameters also depends on the value of the **command_id** field. These parameters may be included in the body of a SMPP PDU. Optional parameters always appear at the end of the PDU. Some of these parameters are **ms_validity**, **sms_signal**, **display_time**, etc. For more information see section 5.3.2, page 132 of [SMPPD].

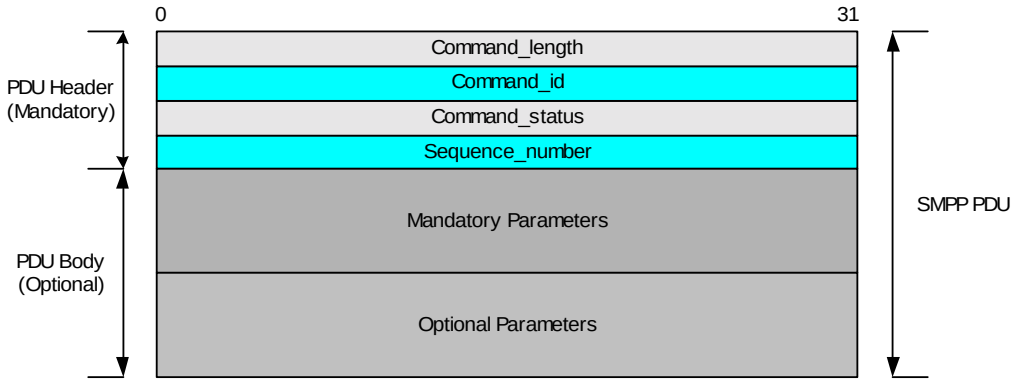

Figure 11: Format of the SMPP PDU

5.6 Related works

During my research on SMS, I found much relevant material on the SMS Forum [SMSF]. I also found an SMPP Java library [Logica], implemented by Logica that is suitable for implementation of SMS applications.

The SMPP Java library implements SMPP version 3.4. This Java Library provides the basic functionality needed to create and encode/decode SMS messages. I utilize this Java library for the implementation of my "SMS Load Generator".

There are also a couple of SMS load-tester applications that can be bought via the Internet. A short explanation of existing load-testing products follows in the subsections below. Deployment of these products is not desirable, since using these products will cause additional traffic on radio network and will occupy extra network resources. Implementation of software that doesn't occupy unnecessary resources or causes any traffic on the radio network was one of the challenges in my project.

5.6.1 SMS Testing suite v2.0

The SMS Testing Suite [OPP] consists of SMS simulators (such as SMPP Simulator, Bad SMSC Simulator, Desktop SMSC, Web SMSC Simulator, SMPP-SMTP Forwarder, and SMS Load Tester) and tools that can be used by developers and service providers for testing of SMS applications using a variety of protocols such as SMPP, SMTP, and HTPP. The simulator operates both via a graphical interface and via command-line interface. This simulator implements SMPP v3.4 and it is backward compatible with version 3.3. The load testing performed by this software computes the number of successful transactions per second.

5.6.2 Temia's GSM and GPRS Application Software

This software consists of a Mobile Switching Center (MSC) simulator developed by Temia [TEM], which provides testing, monitoring, and management tools for GSM and/or GPRS networks. The simulator simulates the functionality of the MSC towards a GSM interface and hence enables testing the performance of individual BSS, MS, and GSM network elements.

The MSC simulator is based on the GSM specifications and includes an SMS message editor that enables the user to create both standard and corrupted SMS messages. It can also provide simulation capabilities for Mobile originated and Mobile terminated short messages. The SMS load generator utilizes the functionality of an SMS Timer. This timer allows the simulator to create and to send SMS messages during a predefined period of time (the duration time must be specified by the user). This software is free of charge and could have been downloaded from [TEM] if their links would have worked.

5.6.3 Netcool Wireless Service Monitors

Micromuse Inc. has implemented a product called “Netcool WSMs”, which monitors the entire voice and data transactions in a wireless network to ensure end-to-end performance and availability of wireless applications [Micromuse]. Netcool also has support for SMS, which allows mobile subscribers to send text messages over GSM or GPRS. This software provides enhanced functionality to monitor and to determine availability, quality of text messages, usability, and response time of SMS- and WAP wireless applications.

Information regarding wireless application performance (e.g. SMS- and WAP performance) and even availability of these applications can be viewed in network status report, delivered by the Netcool Wireless Service Monitor.

6. Development and Evaluation

In this section the performance evaluation in combination with the underlying queueing theory is presented. The purpose of the application software programs developed were to evaluate the performance of the MMSC and SMSC systems, by carrying out the following actions:

- Generating traffic (i.e. Messages).
- Submission and receiving of messages
- Collecting statistics about the performance of the components of interest.

As we can see in the following sections, as the load increases the time required to serve a message increases imperceptibly up to a point. Thereafter, it increases asymptotically toward infinity. This asymptote defines a clear upper bound on the performance of the server. As the load on the server nears this boundary, a minor increase in the load can rapidly plunge the server into deadlock (loop), where it attempts to serve more and more messages at slower and slower speed such that no messages are served successfully. The majority of UNIX based servers, which allows a large number of simultaneous connections are susceptible to this problem while Windows based servers with limited number of simultaneous connections are more resistant to this problem. In order to solve this problem, the UNIX based servers may be equipped with a monitoring facility, which controls the number of simultaneous connections.

In the following sections I will also explain the testing tools (software applications) i.e. MMSC-, and SMSC-Load Generator, how they measure the performance, what components are of interest, and the testing results.

In the performance analysis of the MMSC and SMSC platforms it is important to understand these system's operating characteristics, as well as to fully exploit the components they contain (in order to analyze their behavior and identify the bottleneck of each system), and the services they offer. The performance of the MMSC and SMSC is affected by several factors, such as the level of services offered by these systems (for example creation of trace files at each level, consumes a lot of CPU cycles), implementation and parameterization of the communication protocols, error control at different protocol layers, implementation and parameterization of components located inside these systems, and interconnection and settings of communication equipment inside the network. All these factors and their interactions affect data communication services, offered to end-users.

In this paper I measured performance as seen by my application programs for two different platforms i.e. an MMSC and an SMSC. In the MMSC case, for data transmission, Synchronous mode with constant message size was used. This mode was chosen since the MMSC implementation works only in this mode. In the SMSC measurements I used the Asynchronous mode and the Transceiver service, since the SMSC application program implements SMPP v.3.4. These choices of parameters make the analysis more reliable and simulate the real world.

In this section the underlying queueing theory will also be considered and applied to the server systems, since all messages sent to these servers share resources like processor time and message access (i.e. I/O). Since only one job may use a resource at any time (because each server has just one processor), all other jobs must wait in a queue for their turn at the resource. As jobs complete, they are removed from the queue mean while new jobs enter the queue. Queueing theory helps us to compute the size of such queues, the utilization of each component, the service time, and the time that jobs spend in such queueing systems.

In my application programs I used the TCP/IP protocol suite to accomplish the communication between the load generators and the MMSC- and SMSC platforms.

The performance metrics that I'm primarily interested in are the time taken by the servers to send responses (i.e. response time), and the time taken by the servers to process messages (i.e. the service time). These metrics characterize the performance and can be used to identify the bottleneck of the each system.

6.1 Server model and Queueing theory

For modeling and analyzing the behavior of a complex system (such as a SMSC or a MMSC), a single queueing system is very insufficient tool. In this study I model the systems as an open queueing network in which each queue represents one node (component) inside these systems. This solution was chosen since the sequence numbers inside acknowledgements and messages received from these servers were in the right order. By doing so I'm able to analyze the performance of each component inside the system. Jobs arrive from outside the network and may eventually depart from the network. In these models the low-level details of the SMPP, SMTP, and TCP/IP protocols are ignored.

For utilizing the open queueing network model following assumptions were made:

- Jobs (messages) arrive according to independent Poisson processes.
- Impact of a single job on the performance of the system is very small.
- A single-server with its own queue and exponentially distributed service time at each node.
- All jobs are independent and routing decisions do not depend on past history.

In such an open queueing network every node behaves like an independent M/M/1 queueing system, according to the Jackson theorem [QMF]. Thus the performance measures (i.e. resource occupancy ρ_i and response time T_i) at node i are given by applying the M/M/1 expressions at the corresponding node.

It is important to know that associated with every queue are:

- The arrival rate (A): the average rate at which new jobs arrive at the queue.
- The service time (Ts): the average amount of time taken by a node to process such jobs.
- The queueing time (Tq): the average amount of time a job spends in the queue.
- The average response time (T): that is T = Ts + Tq

If the arrival rate (A) is less than the service rate, which is 1/Ts, then the queueing system is said to be stable. This means that all jobs that enter the queue will eventually be processed and the average queue size is bounded. Otherwise if A is bigger than 1/Ts then the queueing system is unstable and the queue will grow without bound. Thus:

1. $A < \frac{1}{Ts}$ indicates stable queueing system,

Or

2. $A > \frac{1}{Ts}$ indicates unstable queueing system.

The resource occupancy or the utilization of each node is given by the equation:

$$\rho_i = \frac{A_i}{\mu_i} \qquad \textbf{(Equation 6.1.1)}$$

Where $\rho < 1$ is the condition for stability, and the service rate $\mu = \frac{1}{Ts}$.

The service rate for each node is driven from the corresponding response time, by the equation:

$$\mu_i = \frac{1 + (T_i * A_i)}{T_i} \qquad \textbf{(Equation 6.1.2)}$$

Finally for the study of the average queueing size of the entire network (system), Little's theorem [QMF] is used. Little's theorem states that, in any stable system the average number of jobs (N) waiting in the queue can be determined from the following equation:

$$N = A * T \qquad \textbf{(Equation 6.1.3)}$$

Where:

$$T = \sum_{1}^{n} T_i \qquad \textbf{(Equation 6.1.4)}$$

Where n is the number of nodes in the model (system) and T_i is the response time of each node.

For each case (server system) below an open queueing network model will be provided and discussed.

6.2 Performance evaluation of the MMSC Platform

The MMSC-server has a maximum performance of processing 300 MM-messages per minute (according to the vendor). To be sure that the server is capable of handling this amount of workload, we need to measure and analyze the response time of each component inside the server.

6.2.1 MMSC Application

The MMSC offers services by which subscribers can send text, pictures, video clips, and sound to each other.

In order to conduct a performance evaluation for the MMSC, we need to study components inside the MMSC and analyze interactions between them. Components that are of interest in this study and their functions are:

- **MM3-Interface (Initialization node)**

The MM3 interface constitutes the interface to the outside world. All one-time initialization and processing are performed at this node.

All SMTP messages sent to the port 25 of the MMSC will arrive at the MM3 interface. This node is responsible for accepting socket connections to the port 25, SMTP handshaking, receiving of SMTP messages, and scanning the MM-header of the messages (in this order). If the processing in one of these steps (above) fails then the corresponding MM-message will be discarded, and an NACK will be sent. Otherwise the node will send an acknowledgement, and thereafter the NP & Routing database will be contacted.

- **Number Portability (NP) and Routing database**

NP database contains information about service providers (such as Tele2, Telia, and Vodafone). It checks whether the receiver's telephone number is ported or not, while the Routing database indicates where to send notifications (i.e. receiver addresses).

When a Multimedia Message is successfully received by the initialization node and an acknowledgement is sent to the sender, the receiver field of the message header will be used to query the NP database to determine whether the MSISDN of the receiver is ported or not, thereafter the routing database looks up the MSISDN presented in the receiver field to determine where to send the notification. If the look up process doesn't succeed then the receiver of the message is unknown and the message will be discarded. Otherwise the message will be stored in the MM database.

- **Multimedia Message database (MM)**

MM database (also called the Store-and-Forward engine), stores and forwards multimedia messages. The MM database has a queue for incoming messages. This queue has a capacity of three hundred messages (according to a license agreement between Tele2 and Symsoft), which means that if the MM database is not fast enough

to process messages, the queue will grow up to its limit i.e. three hundred messages. In this situation all other incoming messages will be dropped.

If the Routing database found the destination of the MM-message, the message will be stored in the MM database. Thereafter the Subscriber database will be contacted.

- **Subscriber database**

This database contains information about whether the receiver of a message has an MMS-capable Mobile Station (MS) or not. You must have an MMS-capable terminal in order to receive MM-messages.

When a message is stored in the SF engine, the Subscriber database will be contacted to see if the receiver of the message has an MMS-capable MS or not. If the outcome is "yes" then a notification will be sent to the receiver via the SMPP-client (see below) telling him/her that an MM-message is waiting, otherwise (i.e. if the outcome is "no") a legacy interface will be used to send a legacy SMS to the receiver and thereafter the MM-message will be discarded by the MM database and a message indicating the failure will be sent to the original sender, if a delivery report was requested.

- **SMPP Client**

The SMPP client sends binary SMSs to receivers telling them that there are messages waiting for them. Binary SMSs contain information about the size and location of the MM-messages. They also contain parameters and other information for configuration the receiver's MS.

For behavioral analysis and performance measurement of the MMSC, we model the entire system as an Open queueing network (see 6.1).

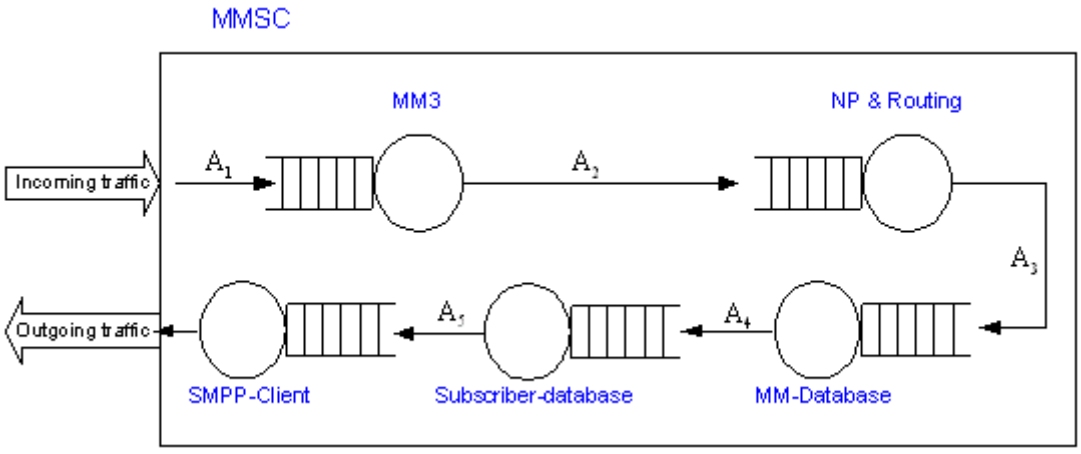

Figure 12: Open Queueing Network Model of the MMSC.

The figure 12 presents a diagram of our open queueing network model. This network model consists of five nodes, where each node represents one of the components

inside the MMSC (described above). Each node (with the queue belonging to it) is considered as a single-server queueing system.

In the following sections we will study the behavior of these components in detail and analyze how they impact the overall performance of the system.

The reader should remember that the MMSC works only in Synchronous mode, which means that when a message is sent to the MMSC, the client **must** wait for corresponding acknowledgement before sending another message.

6.2.2 MMS Load Generator & Data Collector (MMSLG)

MMS-Load Generator and Data Collector (MMSLG) is the name of the software that I created. It is written in Java (using JSDK 1.4.1), since Tele2 wanted platform-independent software.

MMSLG works according to the traditional Client-Server model. It uses SMTP for submission of MIME-type MM-messages to the MMSC via the MM3 (SMTP interface) interface. The SMTP and MIME-Type messages are defined in [RFC0821], [RFC2822], and [RFC2045]. The SMTP protocol runs on the top of the TCP/IP protocol stack.

Since the MMSC only works in synchronous mode the MMSLG submits a message, and it has to wait for the corresponding acknowledgement, before sending another message.

The functions of the MMSLG can be summarized as:

1. Creation of MM-messages: The MM-messages are created with respect to the guidelines specified in [MMSCON]. Each MM-message consists of text, image, and SMIL presentation of the contents. In the header of each MM-message there is a “Subject”-field that is used for message-identification (see below).

2. Putting a unique sequence number in the “Subject”-filed of each MM-message header: These unique sequence numbers are used for identification of acknowledgements and notifications.

3. Adding an SMTP header to each MM-message.

4. Opening TCP/IP connections via sockets to the port 25 of the SMTP interface. These socket connections are open during the message submission until either the MMSC closes the connections (due to its performance bound) or the termination of the test.

5. Dynamic creation of threads for submission of SMTP-messages. Each thread simulates a real MMS-subscriber (i.e. a person sending MM-messages via his/her MS). This is because the MMSC allows multiple connections to port 25.

6. Creation of a timer thread: This thread takes care of the test duration.

7. Registration of the time and the sequence number when:

    a. An SMTP-message is submitted: The sending time and the sequence number of each message will be written in a file, called “Sending-times”.

    b. An acknowledgement is received: When an acknowledgement for the submitted MM-message arrives, the receiving time and the sequence number belonging to it, will be written in a file, called “Response-times”.

    c. A notification for submitted message is received: When a notification for the submitted MM-message arrives (this is identified by the translated Subject-field of the binary SMS) the receiving time and its sequence number will be written in a file, called “notification-times”.

8. An SMSC Simulator (implemented and embedded inside the application software): The purposes of the SMSC Simulator are to eliminate the risk of overflowing the radio network with binary SMSs, and to identify the notifications. The SMPP client (inside the MMSC) makes connection to the SMSC Simulator at start up time and submits binary SMSs to the simulator. The SMSC Simulator runs in a separate thread. The functions of the SMSC Simulator are:

    a. Receiving notifications (binary SMSs) from the SMPP client, via port 3700 (i.e. default SMPP-port) over TCP/IP.

    b. Translation of binary SMSs into human readable format: This translation is necessary in order to read the sequence number inside the notification.

    c. Submission of acknowledgements for each received SMS.

9. Workload definition: The workload can be specified by giving the number of message that must be submitted per second and the period (in minute) during which the test must be run.

10. Creation of three files, containing data about the CPU-performance, SMPP client-performance, and the round trip time (i.e. the time taken to submit a message to the MMSC and receive the corresponding notification from the MMSC).

The connection between the MMSLG and the MMSC is shown by the figure 1 (see section 4.2).

6.2.3 Development Problems and Solutions

The creation of a powerful and reliable load generator for the MMSC platform was a great challenge for me, I had to investigate and implement many different protocols and standards in order to create a program, which could send and receive traffic to/from the MMSC.

This job became much harder because the MMSC behaved abnormally in some situations. It was very difficult to tell what was happening since the implementation of the MMSC diverged from the MMS specification in [3GPPST], and the available documents about this particular server were incomplete. The situation became more frustrating when the vendor couldn't answer my questions and wouldn't tell how they had implemented the MMSC software. During my work with the MMSC, I found many bugs in the server system that forced the vendor to put new patches in the system, which in turn took time. These problems sometimes ended in changes and modifications of my solution to work around problems in their implementation of the MMSC.

Because of the divergence from the MMS specification [3GPPST], I couldn't use my initial design, which was based on MM3-to-MM3 or MM3-to-MM7 interrelation. The only possible way to implement the load generator was to use the interrelation between MM3 and MM1, which in its turn forced me to find a smart way of handling binary SMSs (Notifications or legacy), since I wasn't allowed to overload the radio network by actually sending these binary SMSs. To solve this problem I created an SMSC simulator, which works as a real SMSC and implements SMPP v.3.4. The SMSC simulator is embedded within the MMSLG application and can be started by giving a port number (default is 3700) and pushing a button. The SMSC simulator accepts connections from the MMSC's SMPP-Client, receives SMSs from that client, and submits acknowledgements (for each received SMS) to the SMPP-Client.

Unfortunately there was no possibility of fetching MM-messages from the MMSC since the MMSC-server's MM1 interface is connected to the GPRS via a WAP Gateway. This means that the connection to the MM1 interface is only possible through the radio network. On the other hand, since the WAP Gateway understands HTTP it would be possible to fetch MM-messages, by initiating HTTP-Get requests directly to the WAP Gateway. I tried this option and succeeded to contact the WAP Gateway, but something went wrong that made the request incomprehensible. This behavior could have something to do with the configuration of the WAP Gateway. I don't know for sure because, when I tried to solve this problem I didn't get permission to look inside the WAP Gateway. That was because, according to my industrial adviser, such an operation would involve other employees, who worked with that WAP Gateway, and it could require a lot of time and efforts for them.

The only way to overcome this problem was the use of three MMS-capable mobile stations for fetching MM-messages. In this way the performance of the SF-engine could be measured.

Another important problem was determining which response and which notification belongs to which MM-message. To solve this problem the MMSLG puts a unique

sequence number in the Subject-field of each message; hence this sequence number is used for identification of both the binary SMSs and the acknowledgements. In the case of the binary SMS, I put a function in the SMSC simulator, which makes it possible to decode the binary SMS to the human readable format and thus the sequence number can be placed in the output file for later matching.

6.2.4 Evaluation Results

In the following sections the results of the MMSC load testing will be shown and discussed. To characterize the performance of the MMSC, the components inside the server (see 6.2.1) were examined, and their response times were measured.

The reader should notice that, in all measurements presented here the transmission delay in the core network is neglected, since the MMSC-server is located in the operator's domain and the core network is well dimensioned.

In the last section 6.2.11 the conclusion of these tests will be presented. In this section a number of statistic methods are presented and used in order to accept or reject the vendor's statement about the performance of the MMSC.

In the MMSC performance measurements there are two main cases that are of interest. In the first case we are interested to see how the MMSC works when sending 5 MM/s during an interval of 5 minutes. This case will simulate a TV-contest in which the TV audiences submit MMS messages to a predefined MSISDN in order to join the contest. The second case examines sending 3 MM/s during 30 minutes (i.e. 60% of the peak load). It was very hard to define an exact test-duration for this case since MMSC is new and the actual load-generation will depend mainly on how many MMS-capable MS have been sold. This case attempts to simulate the traffic that will be generated in the minutes before and after New Year.

Table 1 gives a detailed description of the parameters used in the load testing.

Parameter	Value
Maximum Transfer Rate	10 Mbits/Second
Network bandwidth	2 Mbits/Second
Message size	24100 Bytes
Memory assigned to the JVM	240 Mbytes
Socket Buffer size (Send Buffer)	194560 Bytes
Socket Buffer size (Receive Buffer)	6144 Bytes
Maximum Received bytes in one call	131072 Bytes

Table 1: parameters used in MMS load testing

The maximum heap-size used by the Java Virtual Machine (JVM) is 64 Kbytes (by default). In order to increase the performance of the JVM, this size was increased to 240 Mbytes. Of this amount of memory, 194560 bytes was assigned to the sending buffer, and 6144 bytes was assigned to the receiving buffer in order to increase the performance of the MMSLG.

The result of the MMSC load testing for the first case i.e. when submitting 5 MM/s during 5 minutes (i.e. at the maximum load) is given by the table 2. I have chosen to repeat the test with 1, 2, 3, and 4 minutes durations in order to characterize the behavior of the server. A further test with 30 minute duration was also performed since it was interesting to determine the boundary at which the server saturates.

Messages per second	Duration in minutes	Sent Messages	Received Acknowledgements	Losses	Received Notifications	Discarded
5	1	300	300	0	300	0
5	2	600	600	0	535	65
5	3	900	900	0	815	85
5	4	1200	1200	0	1068	132
5	5	1500	1500	0	1242	258
5	30	9000	8029	971	6834	1195

Table 2: The result of MMSC testing, when submitting 5MM/s during variable period of time.

The column “Losses” shows the difference between the number of sent messages and the number of received acknowledgements. The column “Discarded” shows the difference between the number of acknowledgements (of each message sent), and the number of binary SMS-messages received from the MMSC.

As we can see from the table, the MMSC could successfully receive and process five messages per second, during a single minute, i.e. without any losses. As the duration period increases the message-losses increases too, since the queue of the SF-engine has a constant size of 300 MM-messages. If the SFE is not fast enough to process messages, the queue will grow up to its maximum limit and each message beyond this limit will be discarded.

If we look at those five first cases we can clearly see that a doubling of the period causes more than double message-losses (the CPU works according to our expectations). But in the sixth case i.e. when the MMSC runs at the peak load for 30 minutes, 971 MM-messages were discarded by the MMSC. This is because the CPU became slower and slower. In this situation all socket (TCP/IP) connections to the MMSC were terminated (by the MMSC) and the MMSC restarted itself.

Now we will examine the second case in which the arrival rate is 60% of the maximum load (i.e. 3 MM/s) during variable period of time. The table 3 shows the results of the testing.

Messages per second	Duration in minutes	Sent Messages	Received Acknowledgements	Losses	Received Notifications	Discarded
3	10	1800	1800	0	1800	0
3	20	3600	3600	0	3600	0
3	30	5400	5400	0	5400	0

Table 3: The result of MMSC testing, when submitting 3 MM/s during variable period of time.

As we can see from the table 3, when the workload is about 60% of the maximum load, the MMSC can receive and process all MM messages successfully, without any losses. This is because the size of the MM-database's queue never exceeds its limit of 300 MM, because the MM-database's service rate exceeds the arrival rate (see 6.2.7).

In the following sections the response times of the components (described in section 6.2.1) inside the MMSC will be presented, and analyzed. These data are then used to calculate the overall performance of the entire system, and to make a comparison with the numbers received from the MMSLG. Finally in the last section we will utilize the statistical hypothesis testing technique in order to present the result of the MMSC load testing.

6.2.5 MM3-Interface Performance

The MM3 response time is computed by taking the time-difference between when a MM-message is submitted and when the corresponding acknowledgement is received. The measuring method is based on the fact that the MMSC processes messages in synchronous mode.

T_i is the time when message i is sent.

T_i' is the time when the acknowledgement for the message i is received.

$$\text{Response time for the message } i = T_i' - T_i$$

I must point out that when all traces (in the MMSC) are on, about 50% of the CPU-power is consumed by the traces. It means, when the MMSLG sends 3 MM-messages per second, only 1.5 messages (on average) will be processed by the MMSC. To take advantage of the maximum CPU-power, I turned this option off.

Table 4 shows the MM3 response time for the different cases shown in table 2. All times are in Milliseconds.

Load	Max	Min	Mean MM3 Response Time	Standard deviation
5MM/sec in 1 minute	3425	1352	1530.11	276.962
5MM/sec in 2 minutes	7441	1362	1612.40	670.793
5MM/sec in 3 minutes	6239	1331	1543.64	479.850
5MM/sec in 4 minutes	7230	1342	1595.08	537.106
5MM/sec in 5 minutes	7892	1362	1577.84	572.710
5MM/sec in 30 minutes	265101	1322	3542.53	11394.70

Table 4: Mean MM3-Response time, when sending 5 MM/s during variable period of time.

The Table 4 shows that the range (difference between the maximum MM3-response time and the minimum MM3-response time) grows when the testing-duration grows. The standard deviation indicates that there are fluctuations in the response times. The mean MM3-response times for the five first cases show that the service times are stable. But in the last case we can see that the mean response time and the standard deviation increase rapidly. To see why, we need to look at the graph of the last case.

The figure 13 shows the response time of the MM3 interface when sending 5 MM/sec during 30 minutes.

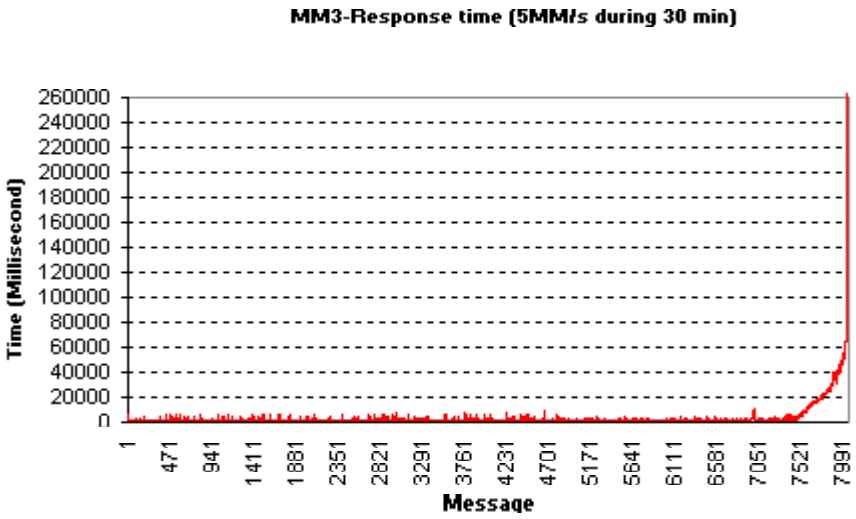

Figure 13: Response time of the MM3 when sending 5 MM/s during 30 minutes.

From the figure 13 we can see that the response time of the MM3 has a stable behavior when the test has been going on for 25 minutes (i.e. about 7500 MM-messages). From this time forward the response time grows almost exponentially, up to the message 8016, thereafter it grows suddenly asymptotically. This asymptote defines an upper bound on the serving capacity of the MM3. In this situation the MMSC is not able to process messages any more, it cuts all the socket connections and restarts. To see why, we need to use queueing theory in order to calculate the utilization degree of the MM3 (when submitting 5 MM/s). The equations 6.1.2 and 6.1.1 give:

Workload	μ	Utilization of the MM3
5MM/s during 1 min	5.65	88.44%
5MM/s during 2 min	5.62	88.96%
5MM/s during 3 min	5.648	88.53%
5MM/s during 4 min	5.627	88.86%
5MM/s during 5 min	5.63	88.75%
5MM/s during 30 min (From message 1 to 7500)	5.58	89.66%
5MM/s during 30 min (From message 7500 to 8016)	5.04	99.20%

Table 5: The occupancy of the MM3 when the workload is 5 MM/s during variable period of time.

As we can see from the table 5, as the duration period increases the MM3 approaches full utilization (100%), and it becomes gradually unstable, which in its turn results in the response time (T) growing suddenly and asymptotically.

The conclusion of this part of my report is that the MMSC can run at the peak load for 25 minutes. When this boundary is passed the MM3 will become slower and slower until it crashes.

The table 6 presents the mean MM3-response times for the cases described in the table 3.

Load	Max	Min	Mean MM3 Response time	Standard deviation	μ	Utilization
3MM/sec in 10 minutes	5828	1362	1484.61	254.3874	3.674	81.66%
3MM/sec in 20 minutes	5828	1321	1488.72	243.061	3.672	81.70%
3MM/sec in 30 minutes	5838	1322	1481.78	251.584	3.675	81.64%

Table 6: Mean MM3-Response time when sending 3 MM/s during variable period of time.

As the table 6 shows, the measured mean MM3-Response times are very close to each other. For all these cases the average service rate (see the equation 6.1.1) is higher than the average arrival rate (i.e. 3 MM/s), and the MM3 utilization is about 82% of the maximum occupancy. This behavior indicates that the queueing system is stable, and the MM3 can catch up with the arrival rate. The standard deviations for these three cases are very close to each other; this indicates that the behavior of the response time-curves is stable. The standard deviation for the first case (i.e. when the duration is 10 minutes) is little higher than the two other cases. This deviation can be explained by the stronger fluctuations in the MM3-response times. The figure 14 shows the MM3-response time for this case (it is just interesting for the comparison between this case and the case with 30 minutes duration).

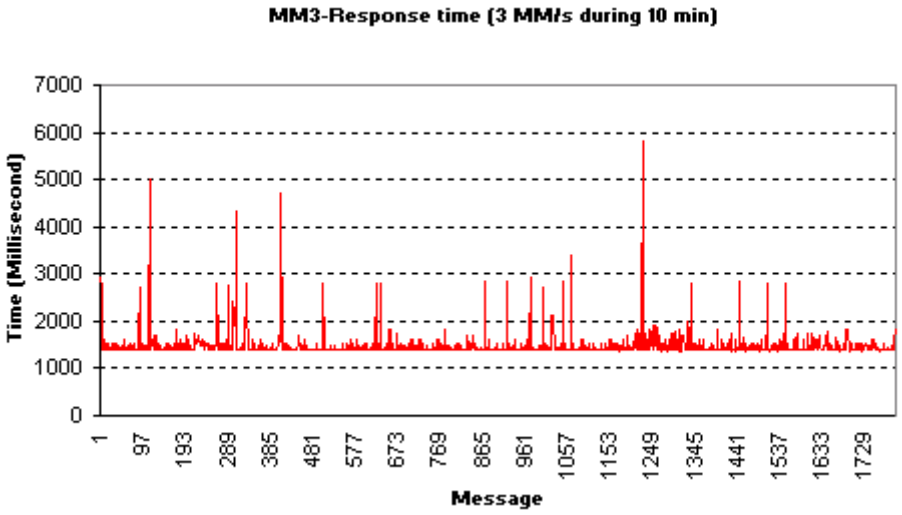

Figure 14: Response time of the MM3, when loaded at 3 MM/s during 10 minutes.

The figure 15 shows the response time of the MM3 when submitting 3 MM/s during 30 minutes.

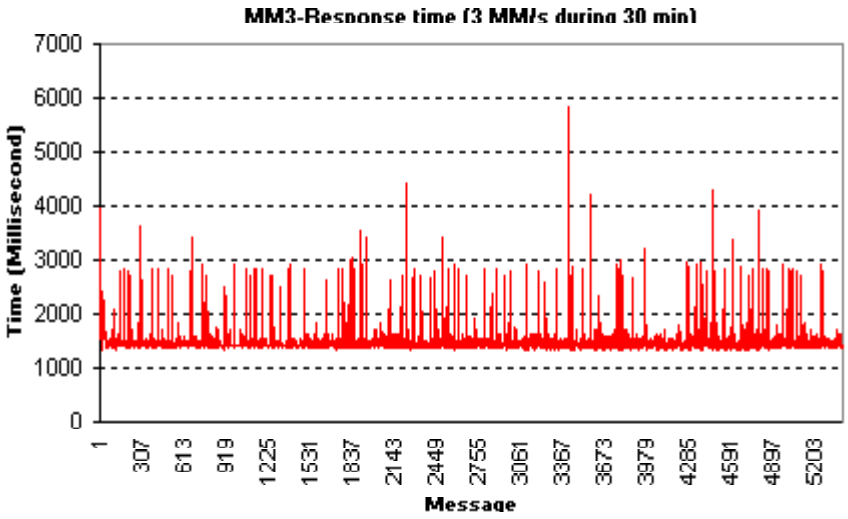

Figure 15: Response time of the MM3 when loaded at 3 MM/s during 30 minutes.

The comparison between figure 14 and 15 shows that, the fluctuations in the figure 14 are more stable/uniform than the fluctuations in the figure 15. That's the reason why the standard deviation for the figure 14 is larger than the figure 15.

The measurements for this case (i.e. with 3 MM/s) shows that the MM3 is capable of handling 3 MM/s during 30 minutes, since the queueing system for this case has shown to be stable.

6.2.6 Performance of the NP and Routing database

Sending pairs of messages with invalid destination address measures the service time of the NP and Routing database. The goal of this approach is to measure the time taken by the NP and Routing database to find the receiver of a MM-message.

If we submit two messages with invalid MSISDN at about the same time, the NP & Routing database will submit two NACKs, saying that the receiver could not be found. By taking the time difference between the departure times of these two NACKs, we can compute the service time.

t_i : is the departure time of the NACK for MM-message i

t_{i+1} : is the departure time of the NACK for message $i+1$

Service time of the NP & Routing database= $T_s = t_{i+1} - t_i$

The performance of the NP & Routing database was measured by sending 420 MM messages with invalid MSISDN (i.e. 210 pairs). This load was chosen in order to get suitable amount of statistic data for computation of the service time. The table 7 shows the result of these measurements.

Workload	Max	Min	Mean NP & Routing Service time	Standard deviation
420 MM-messages	278	5	19.29	34.327

Table 7: Mean service time (in milliseconds) of the NP and Routing database.

As we see, the NP and Routing database has a mean service time of about 19.29 milliseconds. The queueing system is stable, since the arrival rate (A) is less than the service rate 1/Ts (which is 51.84).

The theoretic average response time of the NP & Routing database is about 20.47 milliseconds (calculated by the equation 6.1.2).

6.2.7 Performance of the MM database

The performance evaluation of the MM-database is quite complex. For doing the evaluation, we need to measure and study the response time of the MM-database, and analyze the queue belonging to it.

For evaluation of the performance of the MM-database I used four different MMS-capable MSs to fetch MM-messages form the MMSC. The MMSLG submitted 420 MM-messages to the MMSC, and the MSs could successfully receive all these messages. This workload was chosen in order to get suitable amount of statistic data for computation of the service time of the MM-database. Before submission of these messages I turned all system-traces on. By doing so I was able to see the exact times that it took the MM-database to find and process MM-messages, when fetching requests from the specified destinations had arrived.

The table 8 shows the result of this test (all times are in millisecond).

Workload	Max	Min	Mean MM-database service time	Standard deviation
420 MM-messages	318	223	252.34	14.383

Table 8: Mean service time (in milliseconds) of fetching messages from the MM-database.

The average service time of the MM-database is about 252 milliseconds. The value of the standard deviation shows that the differences between the collected data are very small.

The reader should notice that 252 ms is not really the fetching time, but it is the time taken to find a message and send it to the MM1 interface for delivery.

I also measured the storage time, for the 420 messages, of the MM-database (from the traces). This time includes the time taken to write a message into a proper section of the database, the time taken to give an unique id to each message, and the time taken to send a request (containing the message's id, the MSISDN of the receiver, and the subject field of the MM message) for each message to the subscriber database. The differences between the storage-values and fetching-values were in the interval 0-10 millisecond, i.e. they were in the range (223…318), which will not affect the coming discussion about the behavior of the queueing system of the MM-database.

The figure 16 shows the service-time of the MM-database. As we see the response time curve is stable.

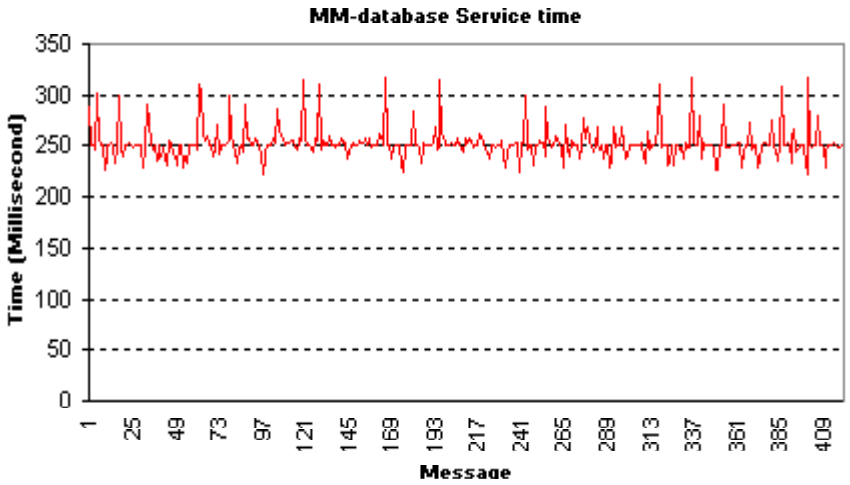

Figure 16: Service time of fetching messages from the MM-database.

When the NP & Routing database has computed the destination address of a message (i.e. when the receiver is identified), the message will arrive the MM-database's queue for being served (i.e. for storage). When the MM-database receives the CPU, it will process (store) the message and thereafter remove that message from the queue.

If the arrival rate (A) is less than the service rate then the queueing system is stable and the average queue size is bounded. Otherwise the queueing system is unstable and the queue will grow to its maximum limit (which is 300). In this situation all incoming jobs are discarded.

The table 7 shows that the average service time of the NP & Routing database is about 19.29 milliseconds. This means all incoming jobs at this unit are processed fast and the messages are handed to the MM-database's queue at the same rate as they were submitted to the MMSC.

Now we will analyze and examine the test results, explained in section 6.2.4 with help of the measured service time from table 8.

The results from the table 2 (i.e. when the workload was 5 MM/s) show that the MM-database can process 5 MM/s during a single minute (i.e. without any losses). This is because of the size of the MM-database's queue (which is 300 MM). Here A = 5 MM/s and the Ts of the MM-database is about 252 milliseconds, then

$$\frac{1}{\left(252/1000\right)} \approx 3.97 < A$$

As it is shown by the equation the arrival rate A is bigger than the service rate. This means that the queueing system is unstable and the queue will grow to its maximum boundary, which is 300 MM. When we submit 5 MM/s during a single minute we actually send only 300 MM to the MMSC (i.e. the size of the queue). That is the reason why the MMSC can process this amount of job without any losses. Thereafter when we increase the duration (to 2, 3, 4, and 5 minutes), the MM-database can't catch up with the arrival rate and the queue grows fast into its maximum limit. Thereafter all incoming messages will be discarded.

When the arrival rate is 3 MM/s (i.e. A=3) during variable period of time, and Ts is 252 milliseconds, then:

$$\frac{1}{\left(252/1000\right)} \approx 3.97 > A$$

This means that the queueing system is stable and the average queueing size is bounded. In this case the MM-database manages to process messages inside the queue fast enough to prevent the queue grows beyond its boundary. This is the reason why the MMSC can handle 3 MM/s (i.e. 60% of its maximum load) during variable period of time without any losses. The resource occupancy or the MM-database utilization is about 75.6% (it is given by the equation 6.1.1), and the average response time is about 1038.52 milliseconds (by the equation 6.1.2).

6.2.8 Performance of the Subscriber database

When eventually a message is stored in the MM-database a request will be sent to the Subscriber database to see whether the receiver has MMS-capable MS or not (for sending either Legacy or Notification SMS). If the receiver has such a MS then the Subscriber unit creates a Notification (binary SMS), and passed it to the SMPP client for further transportation. Otherwise a Legacy SMS will be created, and sent to the SMPP client.

For performance measurement of the Subscriber unit the SMSLG submitted 420 MM-messages (with valid MSISDN) to the MMSC. Before submission I turned all traces on for gathering data of importance. The mean service time of the subscriber unit consists of the average time that it takes to find information about the receivers, and the average time, needed to create and process the data (which is used inside the binary SMS) to the SMPP client.

Table 9 shows the result of the test (all times are in millisecond).

Workload	Max	Min	Mean Subscriber Service time	Standard deviation
420 MM-messages	494	4	37.13	50.362

Table 9: Mean Service time of the Subscriber database.

There are no message-losses here; because when the MM-database saves a message it will eventually result in submission of a binary SMS. The mean service time of the Subscriber database shows that the queueing system for this node is stable (since $A < 1/Ts$).

The theoretic average response time of the subscriber database is 41.78 milliseconds (calculated by the equation 6.1.2).

6.2.9 Performance of the SMPP Client

When the Subscriber database has found information about the receiver's MS, data about the location of the MM-message will be submitted in the body of a binary SMS to the receiver (by the SMPP client).

The service time of the SMPP Client was measured by repeatedly taking the time difference between two consecutive binary SMSs sent from the SMPP client to the SMSC simulator.

Table 10 shows the computed mean service time of the SMPP Client (all times are in millisecond).

Load	Max	Min	Mean SMPP-Client service time	Standard deviation	Mean response time
3MM/sec in 10 minutes	70	10	32.16	15.497	35.59
3MM/sec in 20 minutes	68	10	37.24	18.709	41.92
3MM/sec in 30 minutes	70	10	35.66	20.388	39.93

Table 10: Mean Service time of the SMPP-Client.

In these cases the queueing system shows stable behavior, since the arrival rate is less than the service rate. The average utilization of the SMPP client for above mentioned cases are 9.6%, 11.2%, and 10.7%, which shows that the workload has no serious impact on the component. The response time for each of the cases is also given. The values of the standard deviations show that the differences between measured values are very small.

6.2.10 Performance evaluation of the entire system (MMSC)

For the performance evaluation of the entire system I had only one choice. This choice was to put a unique identification number in the subject filed of each MM-message before submission. Since the binary SMSs (sent from the SMPP client) contain the value of the Subject field, the SMSC-simulator can translate the incoming binary SMSs and thereafter read the value of this filed. By doing so I can see which messages have actually been discarded or successfully have been processed.

The MMSLG calculates the performance of the entire system (i.e. the response time) by taking the time difference between when a MM-message is sent to the MMSC and when the corresponding SMS (i.e. notification) is received.

First I will present the values from the testing, then I will use the Little's theorem (see 6.1) to check these values, and to analyze the queueing system.

Table 11 shows the results of the MMSC performance testing, using MMSLG.

Load	Max t_{Max}	Min t_{Min}	Mean MMSC Response time	Standard deviation
3MM/sec in 10 minutes	7922	2192	2633.78	714.203
3MM/sec in 20 minutes	7271	2156	2757.96	646.384
3MM/sec in 30 minutes	7010	2171	2757.59	674.471

Table 11: Mean response time (in milliseconds) of the MMSC, when sending 3 MM/s during variable period of time.

As we see from the table the average response times of the MMSC for those different cases are very close to each other. This indicates that the server has stable behavior. Let these values be T^*.

The figure 17 shows the response time of the last case, i.e. when sending 3 MM/S during 30 minutes.

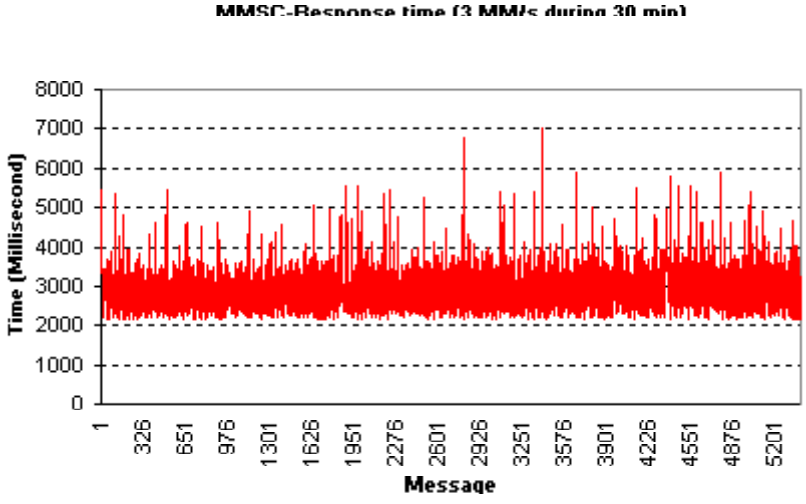

Figure 17: Response time of the MMSC when loaded at 3 MM/s during 30 minutes.

As we see from the figure, the response time of the MMSC has stable behavior. This figure conforms our previous statement about the stability of the queueing system.

In order to characterize the behavior of the MMSC we need to apply queueing theory to calculate the average service rate, the occupancy of the MMSC, and the average number of messages in the waiting queue (using Little's theorem). The Table 12 shows the result of the calculations.

Workload	Average service rate (μ)	Status of the queueing system	Messages in the waiting queue (N)	Utilization (ρ)
3MM/sec in 10 minutes	3.380	Stable	7.9	88.77%
3MM/sec in 20 minutes	3.363	Stable	8.3	89.22%
3MM/sec in 30 minutes	3.363	Stable	8.3	89.22%

Table 12: Average service rate of the MMSC, when sending 3 MM/s during varible period of time.

As the table 12 shows, the average utilization of the server is about 89%, the average service rates are higher than the arrival rates, and the average number of messages in the waiting queue N is constant. This means that the queueing system is stable (for all these cases) and the average queue size is bounded (since N is constant). The MMSC is able to catch up with the arrival rate, and all messages will eventually be delivered to their destinations. Now by using the equation 6.1.4 we calculate the theoretic sum of the response times that we had measured before (i.e. from the component testing). Let these values be T .

Table 13 shows the results of these measurements (all times are in millisecond).

Case	Mean MMSC Response time (T^*)	Theoretic mean MMSC Response time (T)
3MM/sec in 10 minutes	2633.78	2640.83
3MM/sec in 20 minutes	2757.96	2631.41
3MM/sec in 30 minutes	2757.59	2622.48

Table 13: Mean MMSC Response times in practice and in theory

As we can see from the table 13 the differences between the real values and the theoretical values are insignificant, and we also realize that the values of T are:

$$T \in [t_{Min} .. t_{Max}]$$

By using the Little's theorem (equation 6.1.3) and the values of T (theoretic values), we see that the average number of messages in the system is 8 (as we saw in the table 12).

The MMSC testing has proved that when the arrival rate was 60% of the maximum load the server system worked well, and all submitted messages were processed successfully.

6.2.11 Conclusion of MMSC Testing

Tele2 wanted to see the behavior of the MMSC system in two different cases:

1. How many messages the MMSC is able to process when it is running under the maximum load (i.e. 5MM/s) during five minutes. This test is very important for verification of the performance offered by the vendor.

2. How many messages the MMSC is able to process when it is running under 60% of its maximum load (i.e. 3 MM/s) during thirty minutes.

The testing of the second showed that the MMSC can process this amount of traffic without any losses, but in the first case we saw that the MMSC could not handle 5 MM/s during any period longer than one minute.

The testing shows also that the bottleneck of the system is the MM-database with the queue belonging to it. If the database was faster, then the system could handle messages much faster. This would prevent the queue growth.

For rejection or acceptance of the vendor's statement about the maximum performance, we need to perform statistical analysis to determine the number of trials. For doing this we will use three different methods.

The first method is Significance testing. This method is used for determining the number of trials. Before using this method we need first to introduce a number of parameters:

- H0 (Null hypothesis): The MMSC is capable of handling 5 MM/s during 5 minutes.

- $\theta_0 = 0.5$ is the probability of accepting H0 (i.e. the Null hypothesis)

- $\alpha = 0.05$ is the Type I Error rate. This is the probability of rejecting H0, when it is actually true.

- X is the number of observed unsuccessful trials.

- H1: The MMSC cannot handle 5MM/s during 5 minutes.

- $\theta_1 > 0.5$ is the probability of accepting H1.

- β is the Type II Error rate. This is the probability of rejecting H1, when it is actually true.

- N is the total number of trials.

The aims are to reject H0 in favor of H1, and to determine the size of N.

If X > k (where k is the number of unsuccessful trials in N independent trials), then X has binomial distribution i.e.

$$X \in \text{Bin}(N, \theta)$$

Let P(X>k) be the probability of rejecting H0, then:

P(X>k) = P(Z>k), where $Z \in \text{Normal}(n\theta, \sqrt{n\theta.(1-\theta)})$

$$\Rightarrow P(X>k) = P(Z>k) = 1 - \Phi\left(\frac{k - n\theta}{\sqrt{n\theta.(1-\theta)}}\right) = g_{(k,n)}(\theta)$$

$$g_{(k,n)}(\theta) = 1 - \Phi\left(\frac{k - n\theta}{\sqrt{n\theta.(1-\theta)}}\right) \qquad \textbf{(Equation 6.2.11.1)}$$

By using the equation 6.2.11.1 and inserting the values of θ_0 and θ_1 we will get an equation system consisting of two equations and two variables, i.e.

1. α = P(Reject H0 even if it is true) = 0.05, and $\theta_0 = 0.5$ then

$$g_{(k,n)}(\theta_0) \Rightarrow 0.05 = 1 - \Phi\left(\frac{k - (n * 1/2)}{\sqrt{n * 1/2(1-1/2)}}\right)$$

   The Normal distribution table gives the value of 0.05, which is 1.6449. By inserting this value:

$$1.6449 = \frac{k + 1/2 - n/2}{1/2\sqrt{n}}$$

2. $\gamma = 1 - \beta$, where γ is the statistical power. This means that the H0 can be reject with γ% confidence. If $\theta_1 = 0.80$ and $\gamma = 90\%$ then $\beta = 0.1$. The Normal distribution table gives the corresponding value of β, which is 1.2816, then:

$\Phi(0) = \frac{1}{2}$ (i.e. if there isn't any error)

$$g_{(k,n)}(\theta_1) \Rightarrow \beta = 1 - \Phi\left(\frac{k - (n * \theta_1)}{\sqrt{n * \theta_1(1-\theta_1)}}\right)$$

$$1.2816 = \frac{\theta_1 n - k - 1/2}{\sqrt{n} * \sqrt{\theta_1(1-\theta_1)}}, \text{ where } \theta_1 = 0.80$$

This equation system gives n =19.7 (i.e. N = n = 19.7). This means that I must run the test (5 MM/s during 5 minutes) 20 times before I can say something about the performance of the MMSC. But since the highest value of n is 19 (values larger than 19 don't exist in the table of the Binomial distribution, which I used [PROB]), I will choose n to be 19.

Approximation theory is the second method that is used to determine the power of the significance testing (described above).

- Condition: Reject H0 if x > c (c is a constant that must be calculated)

We know that $P(X > c) \leq 0.05$, and $X \in$ Bin (n, 0.5)

If n =19 and $\theta_0 = 0.5$, then c can be obtained by $P(X \leq c) \geq 0.95$ (for 1−0.05=0.95).
The Binomial distribution table gives c=13, and P=0.96822

$\Rightarrow P(X \leq c) = 0.96822$

The power of the test can be obtained by:

$P(X > 13) = P(19-X \leq 19-13) = P(19-X \leq 6)$

Since $(19-X) \in$ Bin (19, 0.2) for $1 - \theta_1 = 0.2$

$\Rightarrow P(19-X < 5) = 0.83694$ is the answer (see below).

Third method calculates the confidence interval of the tests (when n is19).

Let P·(1−P) be the variance of the H0, when P = 0.5 then the confidence interval can be obtained by [PROB]:

$$P^* \pm \frac{P.(1-P)}{\sqrt{n}} \text{ When } n = 19$$

The value for the confidence interval (given by the equation) is ±0.057. This value shows that the confidence interval is very narrow, and for n =19 our estimation will be very accurate (i.e. if n is larger we can be very certain that the result will be a very accurate one).

Finally with respect to our results from the statistical analysis the test concerning 5MM/s during 5 minutes was run (independently) 19 times (it was actually run 25 times, for safety's sake), and in none of the cases could the MMSC handle the workload without any losses. The conclusion (based on n =19) is that the MMSC

server can't handle this amount of traffic, and the Null hypothesis can be rejected with more than 84% power.

Consequently the vendor was contacted, and informed (by me) about the result of the tests (under supervision of my industrial adviser). The vendor started an investigation, and shortly after that they confirmed my results and made an official agreement to upgrade the system.

6.3 Performance evaluation of the SMSC Platform

The SMSC offers services by which subscriber can send text-based messages to each other.

For performance evaluation of the SMSC server we need to study the server's operating characteristic in detail. The SMSC-server has a maximum performance of processing 10 messages per second (according to the vendor). To be sure that the server is capable of handling this amount of traffic, we need to measure, and analyze the hardware and software inside the server.

6.3.1 SMSC Application

The hardware, existing inside the SMSC server contains mainly of an External Interface (EI), and a Store-and-Forward Engine (SFE). In order to conduct a performance evaluation we need to investigate the behavior of these components, and analyze their interactions.

- **External Interface (EI)**

EI is responsible for all initialization tasks when an SMS arrives. There is always a specific EI for each External Short Message Entity (ESME). The EI is responsible for interacting with an ESME, and mediating between the ESME and the Store-and-Forward Engine.

All messages that are submitted to the SMSC will be placed in a queue waiting for their turn to be processed. Since the SMSC is a single processor system, at any time just one message can use the CPU. When eventually a message is processed, it will be removed from the queue, an acknowledgement will be sent, and the message will be handed to the SFE.

- **Store and Forward Engine (SFE)**

All messages entering the SFE will first be placed in a queue, called Active Message Queue. The SFE brings messages (one at the time) from this queue for processing, which means that the SFE tries to find the destinations of the messages, and if they are accessible (their MSs are on) the messages will be submitted to them.

In order to analyze the behavior of the SMSC we need to model the server as an Open queueing network (see figure 18).

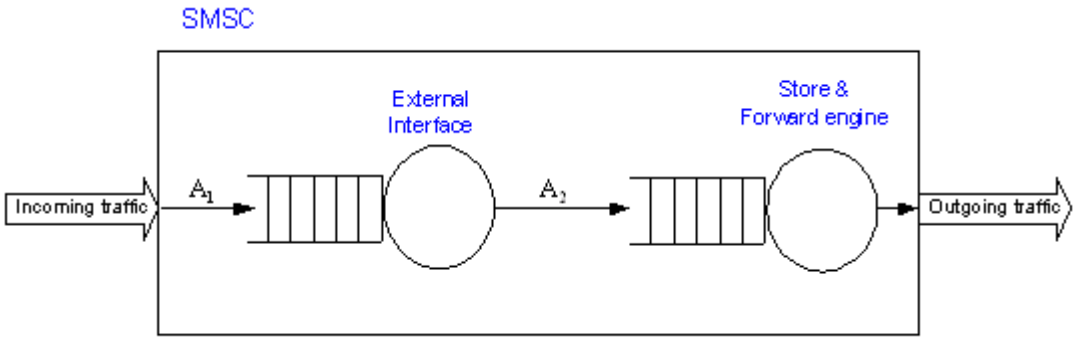

Figure 18: Open Queueing Network Model of the SMSC.

This model abstracts all components inside the server, and is detailed enough to produce significant performance results (see section 6.1). In this model each node represents one component inside the SMSC (presented above), and is considered as a single-server queueing system. In the following sections we will study and analyze this model in more detail.

The reader should remember that the SMSC works both in Synchronous mode, and Asynchronous mode.

6.3.2 SMSC Load Generator & Data Collector (SMSLG)

I implemented the SMS Load-Generator and Data Collector (SMSLG), using Java 1.4.1 (JSDK 1.4.1), since Tele2 wanted platform-independent software. The development environment was Emacs 20.7 running on a PC with Windows NT OS.

SMSLG works according to the traditional Client-Server model. It uses Short Message Peer to Peer (SMPP) protocol version 3.4 on top of the TCP/IP protocol stack for submission, and receiving of text-based messages to/from the SMSC via port 3700 (default port for the SMPP). The choice of the port number depends on the implementation of the SMSC software.

SMSLG is capable of submitting SMS-messages in two different modes (depending on the task); either Synchronous mode or Asynchronous mode (see 5.5.4). All messages are sent and received by the SMSLG. This means that the source address and the destination address are the same.

The functions of the SMSLG can be summarized as:

1. Creation of SMS-messages: SMS-messages are created in accordance with instructions, and rules, specified in [SMPPD]. Each message consists of a header and a body (see 5.5.5). In the mandatory part of the header there is a field called “Sequence_number”, which is used for identification of messages and their responses.

In the mandatory part of the body there is field called “Short_message”, which is used for identification of messages sent to the SMSC and received from the SMSC.

2. Inserting a unique sequence number in the both “Sequence_number” and “Short_message” field of each message (i.e. same sequence number in the both fields): These unique sequence numbers are used for identification of messages (sent to the server), their corresponding acknowledgements, and their corresponding messages (i.e. when messages are received from the server). Since each message, submitted from the SMSC will receive a new sequence number, the SMSLG puts the sequence number even in the “short_message” field of each message, which will remain unchanged.

3. Opening TCP/IP connections (via sockets) to the port 3700 of the SMSC. These socket connections remain open during the message exchanges until either the SMSC closes the connections (due to its performance bound) or the user submits an “unbind_request” PDU by pushing a button.

4. Creation of three threads, one thread submits SMS-messages to the SMSC, the second thread listens for incoming acknowledgements and SMS-messages from the SMSC, and the third thread, which is a timer thread that keeps track of the testing duration. All threads run independent from each other. Message-submissions occur during the same session, because it increases the performance of the application software.

5. Registration of the sequence number and the time, when:

    a. An SMS-message is submitted: The sending time and the sequence number of each message will be written to a file, called “Sending-times”.
    b. An acknowledgement is received: When an acknowledgement for the submitted SMS-message arrives, the receiving time and its sequence number will be written to a file, called “Response-times”.
    c. An SMS-message is received: When an SMS-message is arrived, the receiving time will be registered in a temporary variable. Then the sequence number of the message is extracted from the “short_message” field of the message. Thereafter the extracted sequence number will be looked up in a hash table, containing the sequence numbers of all received messages so far. This check must be done (since the SMSC may submit same SMS more than once in case of software crash) to be sure that the same message is not registered more than once. If the sequence number is not in the table, it will be inserted to the table. Thereafter both the sequence number, and the receiving time will be written to a file, called “Receiving_times”. Otherwise nothing happens.

6. Submission of acknowledgement for each received SMS: When an SMS is received (from the SMSC) an acknowledgement must be submitted in order to stop further transmission of that message.

7. Workload definition: The workload can be specified by giving the number of messages that must be submitted per second, and the period (in minute) during which, the test must be run.

8. Creation of three files, containing data about the EI-performance, SFE-performance, and the SMSC-performance.

The connection between the SMSLG and the SMSC is shown by the figure 5 (see section 5.3).

6.3.3 Development Problems and Solutions

The creation of a powerful, fast, reliable, and memory effective load generator that could give correct data about the performance was the goal of this part of the project. It was very important to use both effective and fast data structures in order to decrease the running time of the threads.

It was much easier to find, and correct bugs in the SMSLG, since the implementation of the SMSC follows almost exactly the SMPP protocol (unlike the MMSC). However, it was not easy to make effective contact with the vendor since they are outside Sweden. Every time I had questions about the SMSC hardware or software I was forced to make many phone-calls, and sometimes wait for weeks in order to get answers. However, in the end their answers were very satisfactory.

In the implementation phase I faced two major problems. The first problem was knowing, which acknowledgement belongs to which SMS. To solve this problem the SMSLG uses sequence numbers. Since each SMS has its own sequence number (which is also unique) it is possible to recognize the corresponding acknowledgement, sent from the SMSC. This is because the SMSC puts the same sequence number in the "sequenceNumber" field of each corresponding acknowledgement. The second problem was about knowing, which SMS message from the SMSC corresponds to which SMS message sent from the SMSLG (i.e. to make the round-trip-time measurements possible). To solve this problem the SMSLG puts even the sequence number in the "short_message" field of each SMS, before submission. This is because when the SMSC receives an SMS message it will remove the old header, and add a new header (with new sequence number) to the body of the message before submission to the destination (i.e. the SMSLG). Since the body part of the message remains unchanged it is appropriate to put the sequence number even there. It is very important to not confuse the terms "acknowledgement" and "SMS message", submitted from the SMSC. With this solution the SMSLG is capable of measuring the round-trip-time for each submitted message.

6.3.4 Evaluation Results

In the SMSC performance measurements there are two main cases that are of interest (for Tele2). In the first case we are interested to know how the SMSC acts when sending 10 SMS/s (i.e. peak load) during 5 minutes. This case will simulate a TV-

contest in which TV audiences submit SMS messages to a predefined SMPP account in order to join the contest. The second case is about sending 7 SMS/s (i.e. 70% of the peak load) for 120 minutes. This case attempts to simulate the traffic that will be generated in the hours before and after Christmas or New Year. In this section the result of the SMSC testing for above-mentioned cases are presented.

The reader should notice that, in all measurements, the transmission delay in the core network is neglected, since the SMSC-server is located in the operator's domain and the core network is well dimensioned.

Table 14 gives a detailed description of the parameters, used during the load testing.

Parameter	Value
Maximum Transfer Rate	10 Mbits/Second
Network bandwidth	2 Mbits/Second
SMS PDU size	600 Bytes
Memory assigned to the JVM	128 Mbytes
Socket Buffer size (Send Buffer)	204800 Bytes
Socket Buffer size (Receive Buffer)	204800 Bytes
Maximum Received bytes in one call	153600 Bytes

Table 14: Parameters used in the SMS load testing by SMSLG

The maximum heap-size used by the Java Virtual Machine (JVM) is 64 Kbytes (by default). In order to increase the performance of the JVM and consequently the performance of the SMSLG, this size was increased to 128 Mbytes. Of totally 128 Mbytes memory available, 205 Kbytes was assigned to the sending buffer, and the same amount of memory was assigned to the receiving buffer in order to increase the speed of the MMSLG. This means that each buffer can contain up to 342 SMS messages.

The result of the SMSC load testing when sending 10 SMS/s during 5 minutes (i.e. at the maximum load) is presented in the table 15. Further tests with 20, 30, and 40 minutes duration were also run, Since it is interesting to determine the actual performance of the SMSC. Another test with 11 SMS/s during 30 minutes was also run, since as we see later the SMSC behaves interesting in this case.

Messages per second	Duration in minutes	Sent Messages	Received Acknowledgements	Losses	Received Messages	Discarded
10	5	3000	3000	0	3000	0
10	20	12000	12000	0	12000	0
10	30	18000	18000	0	18000	0
10	40	24000	18698	5302	18121	577
11	30	19800	20357	–557	19800	0

Table 15: The result of SMSC testing, when sending 10 SMS/s during variable period of time.

The column “Losses” shows the difference between the number of sent messages and the number of received acknowledgements. The column “Discarded” shows the difference between the number of acknowledgements (for each sent message), and the number of messages received by the SMSLG.

As we see from the table 15, the SMSC could process the generated workload without any losses when the duration periods were 5, 20, and 30 minutes. But when the duration was increased to 40 minutes the SMSC became slower, and began to lose messages. It could receive 18698 messages of total 24000 messages, and of 18698 messages only 18121 were successfully submitted, which means that 577 messages were discarded inside the system.

In the case of 11 SMS/s, we see that the SMSLG submitted 19800 messages, but it received 20357 acknowledgements i.e. 557 acknowledgements more, and of total 19800 messages, received by the SMSC all messages were successfully transmitted to the SMSLG. The difference between the number of sent messages (to the SMSC), and the number of received acknowledgements (from the SMSC) is because of a software crash. The SMSC crashes, when it attempts to process more and more acknowledgements at slower and slower speed such that no acknowledgements are processed successfully. The only way to get out of this state was to restart the SMSC.

The SMSC testing results for the second case, when the workload is 70% of the maximum load (7 SMS/s) is shown by the table 16. In this case I began with a period of 60 minutes, after that I increased the period to 120, and finally to 180 minutes.

Messages per second	Duration in minutes	Sent Messages	Received Acknowledgements	Losses	Received Messages	Discarded
7	60	25200	25200	0	25200	0
7	120	50400	50400	0	50400	0
7	180	75600	57188	18412	56999	189

Table 16: The result of SMSC load testing, when sending 7 SMS/s during variable period of time.

The SMSC is capable of handling 7 SMS/s during 120 minutes. But when the test duration is increased from 120 to 180 minutes, only 57188 acknowledgements could be received by the SMSC, and of this amount of messages only 56999 messages could successfully be submitted to the SMSLG (i.e. 189 messages were discarded inside the SMSC).

In the following sections the performance of the components (described in section 6.3.1) inside the SMSC are measured, and discussed. In the last section the conclusion of the SMSC load testing is presented. In that section some statistical methods are used in order to reject or not reject the vendor’s statement about the SMSC’s performance.

6.3.5 External Interface (EI) Performance

All SMS messages submitted to the SMSC arrive at this interface, and will enter a queue, waiting for their turn to be processed. When a message receives service, the EI sends it to the SFE for further transportation, and an acknowledgement will be submitted to the sender.

The service time of the EI is measured by taking repeatedly the time difference between the departure times of two consecutive responses. If the acknowledgement for message i is submitted at time t_i and the acknowledgement for message i+1 is submitted at time t_{i+1} then the service time of the EI is:

$$\text{EI-service time} = t_{i+1} - t_i$$

We begin with the first case i.e. when the SMSLG submits 10 SMS/s during variable period of time, and the case of 11 SMS/s. Table 17 shows the mean EI response time for the cases, presented in the table 15 (all times are in millisecond).

Load	Max	Min	Mean EI Service Time	Standard deviation
10 SMS/s in 5 minutes	461	10	98.99	32.529
10 SMS/s in 20 minutes	6349	10	99.91	81.611
10 SMS/s in 30 minutes	4437	10	143.79	219.055
10 SMS/s in 40 minutes	19919	10	215.28	442.418
11 SMS/s in 30 minutes	8152	10	189.40	376.758

Table 17: The mean EI service time, when sending 10SMS/s during variable period of time.

As we see from the table 17, when the period increases from 20 to 30, and then 40 minutes (for 10 SMS/s), the average service time of the EI increases rapidly, this means that with the time the EI becomes slower and slower. The standard deviations for these cases indicate that the differences between the measured service times become larger and larger.

Figures 19, and 20 show the service time of the EI, when sending 10 SMS/s during 30, and 40 minutes. Figure 21 shows the service time for the case of 11 SMS/s during 30 minutes.

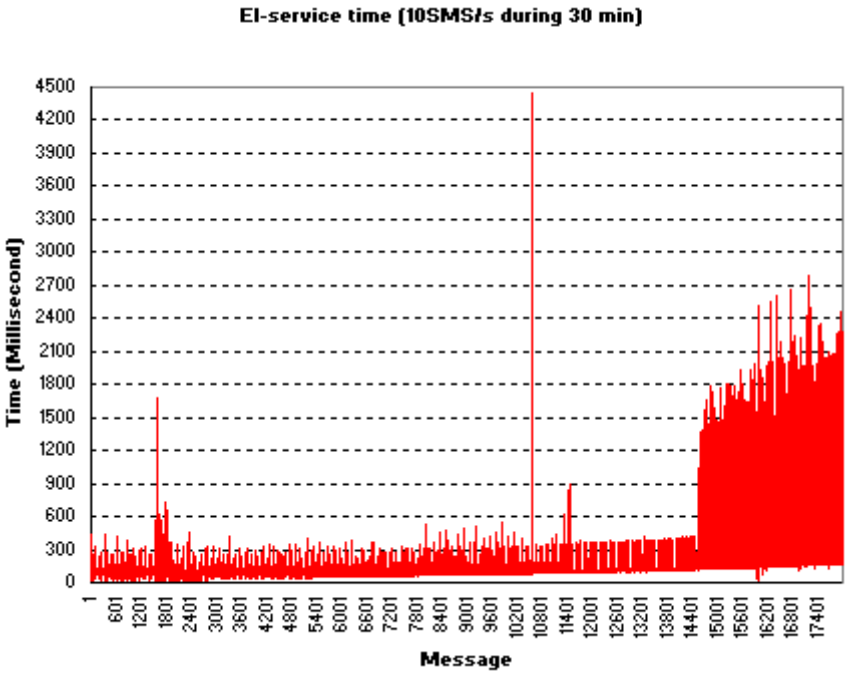

Figure 19: Service time of the EI, when loaded at 10 SMS/s during a period of 30 minutes.

We see in the figure 19 that the average service time of the EI is about 90 milliseconds, up to the message 5401. From this message to message 14401 the average time increases slowly and continuously, and from that message to the last message the average time increases rapidly.

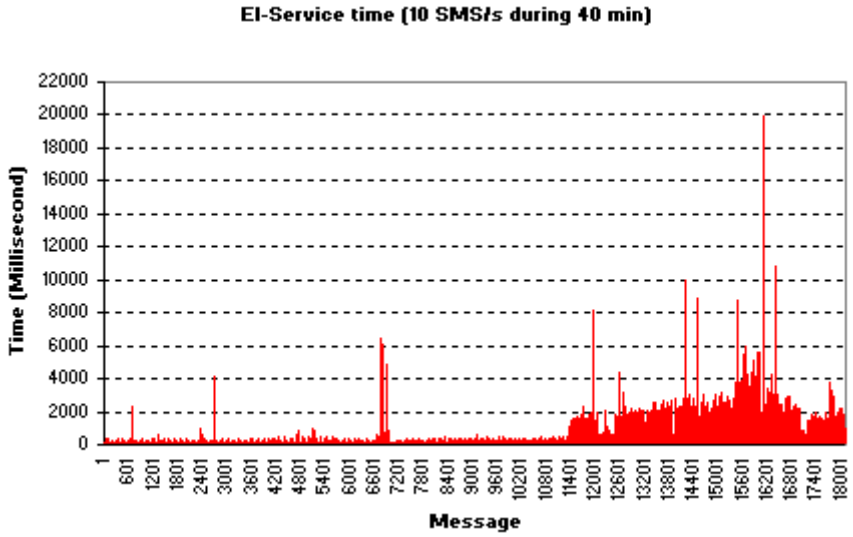

Figure 20: Service time of the EI, when loaded at 10 SMS/s during a period of 40 minutes.

As the time being passed the EI becomes slower and slower, and consequently the average service times become larger and larger. As we see of total 24000 messages the EI could receive, and process just 18698.

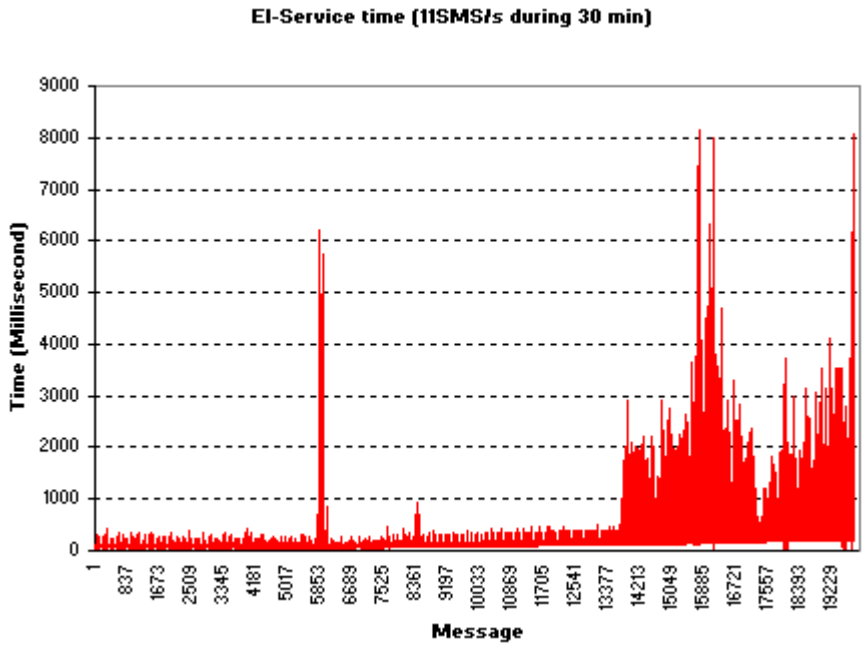

Figure 21: Service time of the EI, when loaded at 11 SMS/s during a period of 30 minutes.

The same motivation is valid even for this case. As the time goes on, the EI becomes slower and slower, and consequently the average service time grows.

Now in order to analyze the behavior of the EI, we need to utilize queueing theory (see 6.1). For doing so we need to categorize the average service rate of the EI, determine the status of the queueing system, and calculating the occupancy of the EI for the cases described so far.

Workload	Arrival rate (A)	Average service rate (μ)	Status of the queueing system	Utilization (ρ)
10SMS/s during 5 min	10	10.10	Stable	98.99%
10SMS/s during 20 min	10	10.01	Stable	99.90%
10SMS/s during 30 min	10	6.95	Unstable	Unstable
10SMS/s during 40 min	10	4.65	Unstable	Unstable
11SMS/s during 30 min	11	5.28	Unstable	Unstable

Table 18: Average service rate of the EI, when the work load is at the maximum load

The table 18 shows that, the average service rate of the EI (i.e. $1/T_s$) for the first two cases are bigger than the arrival rate (i.e. $A < 1/T_s$), which means that the queueing system is stable and the average size of the queue is bounded. The occupancies of the EI are also less than one, which is another indication of the stability. In the third case $A > 1/T_s$, this means the queueing system is unstable, and the queue will grow without bound. In this case (and the remaining cases) the occupancy of the EI is bigger than one, since the queueing system has shown to be unstable, but the duration

period is not long enough to cause system-crash. When the period is increased from 30 to 40 minutes the EI can receive and process 18698 messages of total 24000. This number of messages corresponds to 31.16 minutes, which means an increase by just about 2 minutes is long enough to plunge the server into saturation.

Table 19 shows the mean EI service times for the cases, where the arrival rate were 70% of the maximum load (i.e.7 SMS/s) during variable period of time.

Load	Max	Min	Mean EI Service Time	Standard deviation
7 SMS/s in 60 minutes	6229	4	119.17	111.536
7 SMS/s in 120 minutes	8042	4	119.18	187.419
7 SMS/s in 180 minutes	66916	4	203.82	1064.495

Table 19: Mean EI Service time, when sending 7 SMS/s during variable period of time.

As the table 19 shows, the mean EI service times for the two first cases have almost the same values. The fluctuations in the service time are also not so high; therefore the values of the standard deviation are low. But in the last case the average service time grows rapidly to 204 milliseconds, and the standard deviation grows to 1064. This indicates that the EI behaves abnormally in this situation.

Now for analyzing this behavior we use queueing theory in order to calculate the values of the average service rates, and the occupancies of the EI (see the equation 6.1.1) for the cases described in the table 19.

Workload	Arrival rate (A)	Average service rate (μ)	Status of the queueing system	Utilization (ρ)
7 SMS/s during 60 min	7	8.3914	Stable	83.42%
7 SMS/s during 120 min	7	8.3907	Stable	83.43%
7 SMS/s during 180 min	7	4.91	Unstable	Unstable

Table 20: Average service rate of the EI, when the work load is 70% of the maximum load

As the table 20 shows, the queueing system for the two first cases is stable, and the occupancy of the EI is about 83%. But in the last case $A > \mu$ i.e. the queueing system is unstable, which eventually results in system-crash.

The EI-performance testing shows that this component is capable of handling the desired workloads (i.e. 10 SMS/s during 5 min, and 7 SMS/s during 120 min). Further we have seen that in those cases, when the SMSC wasn't able to process the workloads, the number of acknowledgements was larger than the number of SMS-messages, submitted from the SMSC. This means that the SFE may be the bottleneck of the system, but we don't know yet. We need to examine this component too.

6.3.6 Store-and-Forward-Engine (SFE) Performance

All messages processed by the EI will enter the Active Message Queue (AMQ), and must wait for their turn to be served by the SFE. The SFE processes messages (one by one) and sends them to their destination.

The performance of the SFE is measured by taking repeatedly the time difference between the departure times of two consecutive SMS-messages. If the departure time of the message i is t_i and the departure time of the message $i+1$ is t_{i+1} then the service time of the SFE is measured by:

$$\text{SFE-service time} = t_{i+1} - t_i$$

For performance evaluation of the SFE, the cases described in the tables 14, and 15 are examined here. First the case concerning 10 SMS/s, and second the case with 7 SMS/s during variable period of time will be studied.

Table 21 shows the mean SFE service times, described in the table 15 (all times are in millisecond).

Load	Max	Min	Mean SFE Service Time	Standard deviation
10 SMS/s in 5 minutes	480	10	99.94	31.723
10 SMS/s in 20 minutes	6149	10	99.97	79.842
10 SMS/s in 30 minutes	4507	10	148.10	219.537
10 SMS/s in 40 minutes	23313	10	221.24	450.521

Table 21: Mean SFE Service time, when sending 10 SMS/s during variable period of time.

As the table shows, the mean service times for the two first cases are very close to each other, and the standard deviations are not too large. In the third case (30 minutes duration) the mean value increases fast, and the standard deviation shows strong fluctuations in the measured service times. This can be interpreted as a warning signal, indicating that the system is about to be instable. In the forth case we can see that the mean value has increased rapidly to 221 milliseconds, since the SFE has become slower. The fluctuations in the measured data are also high, since the service time increases continuously with the time.

The figures 22, 23, and 24 show the service time of the SFE when the workload is 10 SMS/s.

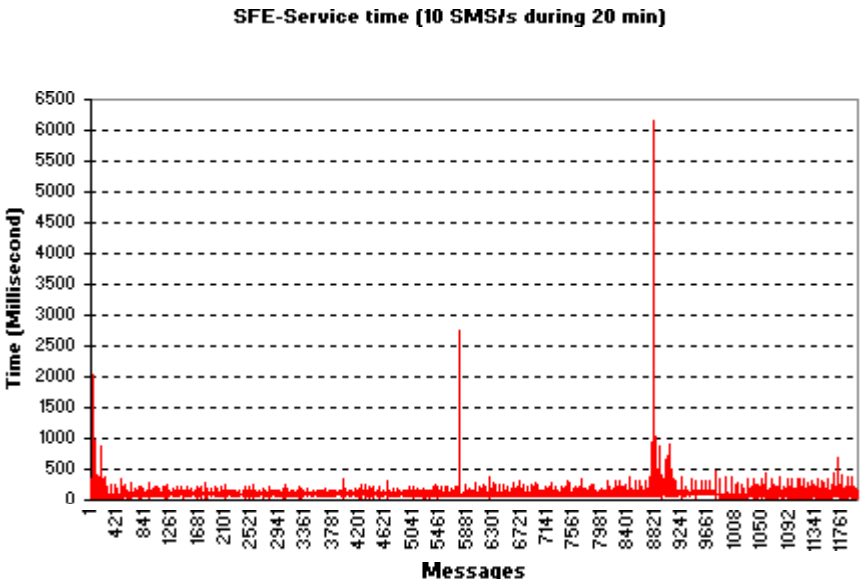

Figure 22: Service time of the SFE, when loaded at 10 SMS/s during a period of 20 minutes.

As the figure 22 shows, the service time of the SFE (when the period is 20 minutes) is stable. This indicates also that the queueing system is stable too.

But when the duration period is increased to 30 minutes, at the end of the time period the SFE starts to be unstable. The figure 23 shows the time of this case.

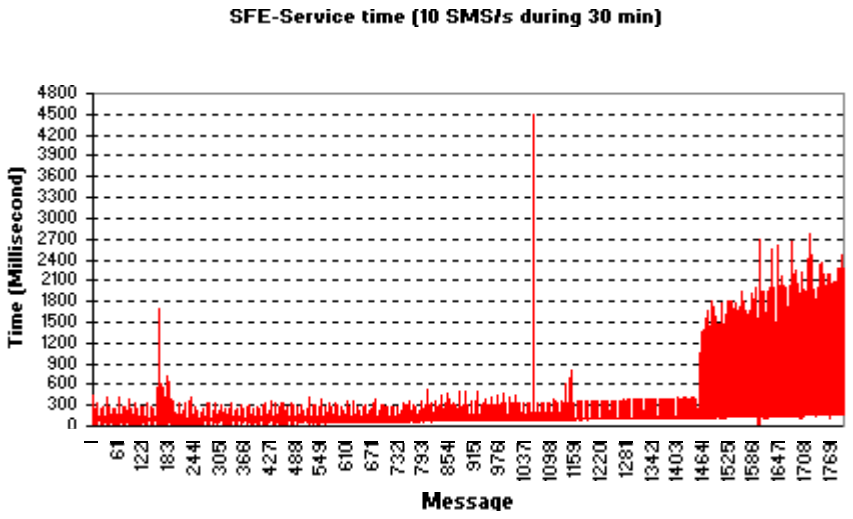

Figure 23: Service time of the SFE, when loaded at 10 SMS/s during a period of 30 minutes.

We see here that the service time of the SFE grows rapidly (beginning at about the message 14641 and forward). In this case the testing duration is not long enough to cause system crash, but it is clear that the SFE has become slower.

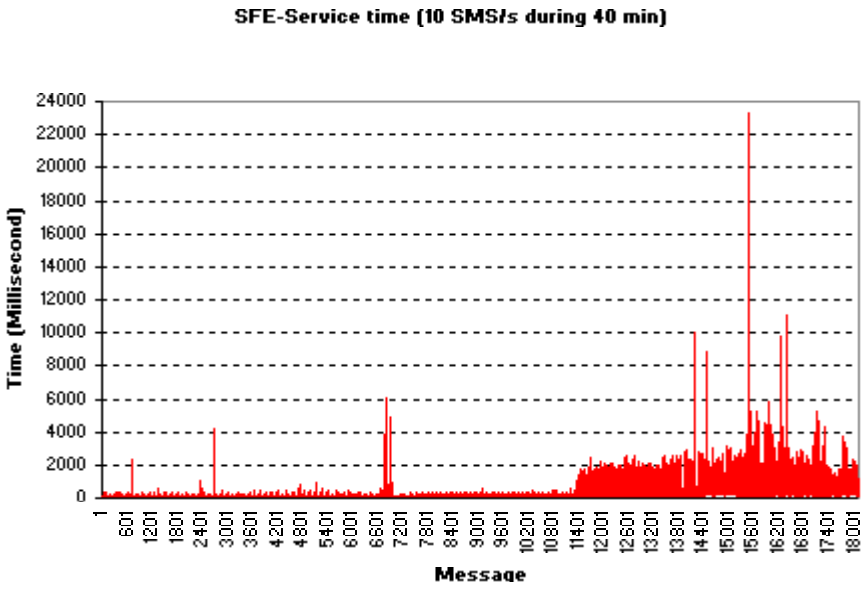

Figure 24: Service time of the SFE, when loaded at 10 SMS/s during a period of 40 minutes.

By increasing the testing period from 30 to 40 minutes, we actually cross the performance boundary and cross into saturation and instability. In this case the server system is plunged into saturation, and the SMSC begins to drop messages. In this situation the number of messages dropped by the SFE is higher than the number of messages dropped by the EI.

In order to analyze the behavior of the SFE (for these cases) we need apply queueing theory. To do this we need to determine the average service rates, status of the queueing system, and the utilization of the SFE (see the equation 6.1.1). Table 22 shows these values.

Workload	Arrival rate (A)	Average service rate (μ)	Status of the queueing system	Utilization (ρ)
10 SMS/s during 5 min	10	10.006	Stable	99.94%
10 SMS/s during 20 min	10	10.003	Stable	99.97%
10 SMS/s during 30 min	10	6.721	Unstable	Unstable
10 SMS/s during 40 min	10	4.520	Unstable	Unstable

Table 22: Average service rate of the SFE, when the work load is 10 SMS/s

We see from the table 22 that the average service rates of the two first cases are higher than the average arrival rates, which means that the queueing system is stable. In the third case the average arrival rate is higher than the average service rate, which in its turn results in that the waiting queue tends to grow without bound, but the testing duration is not long enough to cause system-crash. When the duration period is increased from 30 to 40 minutes, 18121 messages of total 18698 can be processed successfully by the SFE. This amount of messages corresponds to 30.20 minutes of duration. This means when the duration period is increased by just about 20 seconds, the queueing system of the SFE collapses, and consequently the system crashes.

Now we will study the second scenario in our testing procedure i.e. when the workload was 7 SMS/s during variable period of time. The table 23 shows the values of the mean SFE service time (all times are in millisecond).

Load	Max	Min	Mean SFE Service Time	Standard deviation
7 SMS/s in 60 minutes	5899	10	122.19	99.807
7 SMS/s in 120 minutes	8002	10	122.43	186.072
7 SMS/s in 180 minutes	66916	10	203.98	1056.760

Table 23: Mean SFE service time when submitting 7 SMS/s during variable period of time.

The table 23 shows that, the mean service times for the two first cases are almost equal, since the measured service times have stable behavior. But in the last case the mean value increases rapidly to 204 milliseconds, and the differences between the measured service times become larger (larger standard deviation). This indicates that the SFE has become slower, and the system is not stable.

To analyze the behavior of the SFE we need to study the queueing system (see 6.1). The table 24 shows the average service rate, status of the queueing system, and the occupancy of the SFE for the cases described in the table 23.

Workload	Arrival rate (A)	Average service rate (μ)	Status of the queueing system	Utilization (ρ)
7 SMS/s during 60 min	7	8.184	Stable	85.53%
7 SMS/s during 120 min	7	8.168	Stable	85.70%
7 SMS/s during 180 min	7	4.902	Unstable	Unstable

Table 24: Average service rate of the SFE, when the work load is 7 SMS/s

As it was expected the average service rates (for the two first cases) are larger than the average arrival rate, which indicates that the queueing system is stable, and the average queue size is bounded. In the last case the average arrival rate is higher than the average service rate. This indicates that the queueing system is unstable, and it will eventually collapse.

The SFE testing shows that the SFE is also capable of processing the desired workloads (10 SMS/s during 5 min, and 7 SMS/s during 120 minutes). But in those cases that the SMSC couldn't handle the generated workload (the total number of received acknowledgements were higher than the total number of received SMS-messages), messages were discarded inside the server by the SFE. This can be explained because the mean service rate of the SFE is little higher than the mean EI service rate, which in its turn causes message losses inside the system during long run times.

6.3.7 Performance evaluation of the entire system (SMSC)

For performance measurement of the SMSC, the SMSLG calculates the round trip time of each submitted SMS-message. The round-trip-time is the time taken to submit a message to the SMSC, and to receive the corresponding message from the SMSC.

If:

t_i is the submission-time of the message i

t_i' is the receiving-time of the message i

Then:

SMSC-Response time = $t_i' - t_i$

In this section the response time of the SMSC for the cases described in the tables 15, and 16 will be presented (see 6.3.4). Then the behavior of the queueing system will be studied and analyzed.

The table 25 shows the mean response times of the SMSC, when the workload is 10 SMS/s during variable period of time.

Load	Max	Min	Mean SMSC Response Time	Standard deviation
10 SMS/s in 5 minutes	25427	50	2691.45	6497.337
10 SMS/s in 20 minutes	25297	50	2538.39	6135.676
10 SMS/s in 30 minutes	793942	50	86208.17	180794.420
10 SMS/s in 40 minutes	2088173	50	362220.86	596236.862

Table 25: Mean SMSC Response time, when sending 10 SMS/s during variable period of time.

As the table shows, the average response times of the SMSC for the two first cases are very close to each other. But in the third and forth case these values increase rapidly, and the standard deviations show strong fluctuations in the response time.

Here we will directly use the M/M/1 queueing theory to analyze the behavior of the queueing system, and the Little's theorem to determine the average number of messages in the waiting queue. For doing so we need to determine the average service rate (using equation 6.1.2), status of the system, and the occupancy of the SMSC (using equation 6.1.1).

Table 26 shows the results of the calculations.

Workload	Average service rate (μ)	Status of the queueing system	Messages waiting in the queue (N)	Utilization (ρ)
10 SMS/s during 5 min	10.37	Stable	26.9	96.42%
10 SMS/s during 20 min	10.39	Stable	25.4	96.21%
10 SMS/s during 30 min	10.01	Stable	862.1	99.88%
10 SMS/s during 40 min	10.00	Unstable	Infinity	100.00%

Table 26: Average service rate of the SMSC, when sending 10 SMS/s during varible period of time.

As the table 26 shows, the queueing system of the SMSC is stable for the two first cases, and the average number of messages in the queue, N are constant. In the third case we can see that the average service rate is very close to the arrival rate, and N has grown very fast (compared to the pervious case). This indicates that the SMSC has become slower, and messages must wait longer (i.e. instability begins to show up), but the testing duration is not long enough to cause system crash. In the last case the average arrival rate and the average service rate are equals, which results in that the waiting queue tends to grow into infinity (division by zero), and consequently the server system eventually crashes.

The conclusion is that as the time goes on messages tend to wait longer and longer, and the queue tends to grow larger and larger. To see how the SMSC behaves we need to look at graphs for the cases shown in the table 25.

The figures 25, 26, and 27 show the response time of the SMSC, when the duration periods are 20, 30, and 40 minutes.

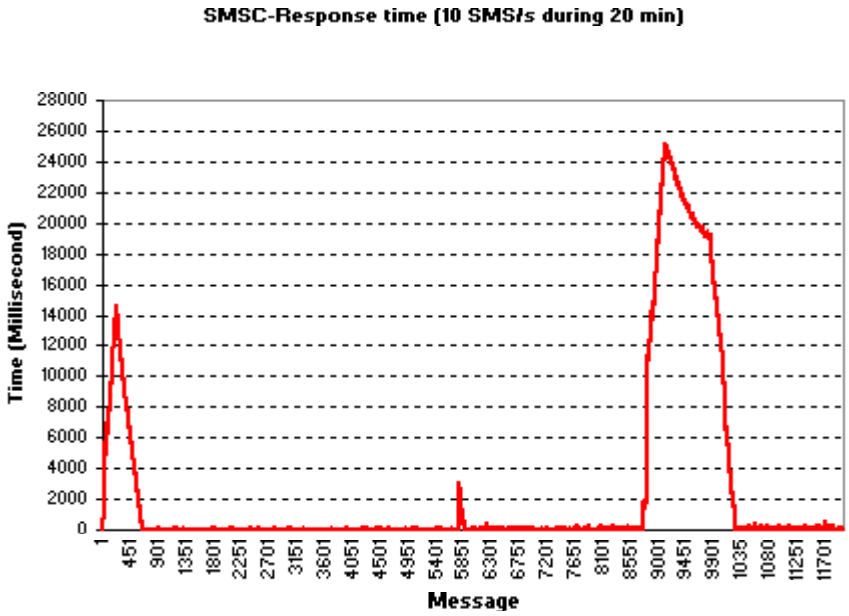

Figure 25: Response time of the SMSC, when loaded at 10 SMS/s during 20 minutes.

As the graph shows, the response time of the SMSC indicates that the queueing system is stable, and the SMSC works according to the expectations.

The next graph shows the situation when the duration period is increased to 30 minutes.

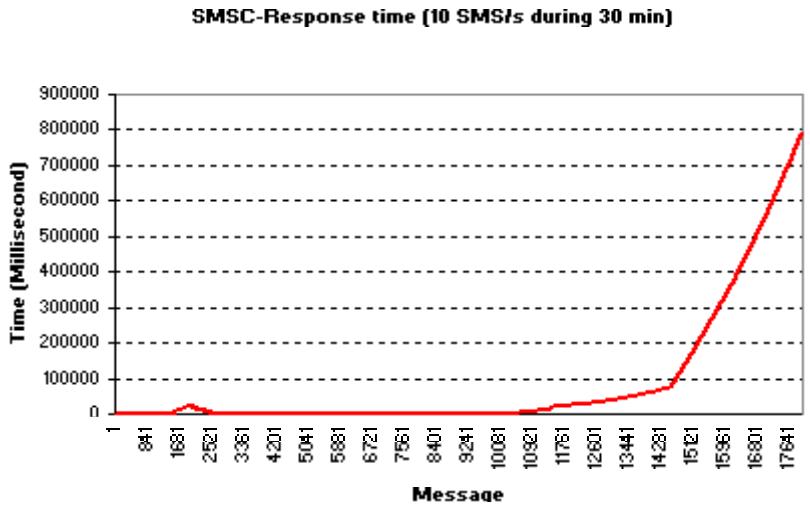

Figure 26: Response time of the SMSC, when loaded at 10 SMS/s during 30 minutes.

As we can see from the figure 26, the queueing system shows stable behavior up to message 14500 (approximately). Later on, the queueing system becomes instable and the response time curve begins to grow linearly.

The next graph shows the situation in which the duration period is increased to 40 minutes.

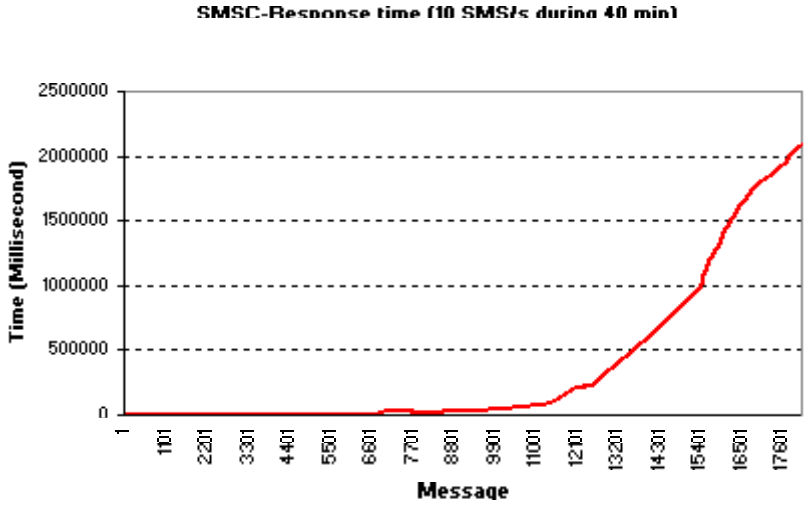

Figure 27: Response time of the SMSC, when loaded at 10 SMS/s during 40 minutes.

As we can see the response time curve is stable at the beginning. Eventually the queueing system becomes unstable, and consequently the response time grows almost exponentially.

The figure 28 shows the response time of the SMSC when the workload is increased to 11 SMS/s during an interval of 30 minutes. This figure is interesting for comparison to the figures 26 and 27.

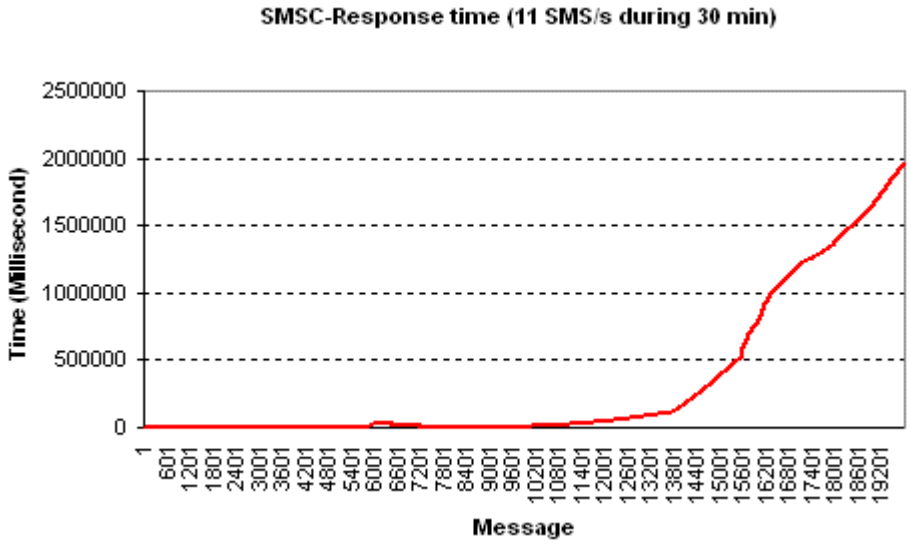

Figure 28: Response time of the SMSC, when loaded at 11 SMS/s during 30 minutes.

The comparison of these three figures (i.e. 26, 27, and 28) shows that the behaviors of the response time curves are very similar to each other, which indicates that our statistics are very accurate.

Now we will examine the SMSC for the second scenario i.e. when the workload is 7 SMS/s during variable period of time. Table 27 shows the results of the testing (all times are in millisecond).

Load	Max	Min	Mean SMSC Response Time	Standard deviation
7 SMS/s in 60 minutes	16113	50	628.95	2098.623
7 SMS/s in 120 minutes	18857	50	702.20	2243.409
7 SMS/s in 180 minutes	3512090	50	182573.29	387866.666

Table 27: Mean SMSC Response time, when sending 7 SMS/s during variable period of time.

As we see the mean response time of the SMSC is stable for the two first cases. But in the third case, the mean value increases rapidly, which indicates that the system is unstable. To analyze these behaviors we use queueing theory.

Table 28 shows the average service rate, occupancy, and other important parameters for this study.

Workload	Average service rate (μ)	Status of the queueing system	Messages waiting in the queue (N)	Utilization (ρ)
7 SMS/s during 60 min	8.59	Stable	4.4	81.49%
7 SMS/s during 120 min	8.42	Stable	4.9	83.09%
7 SMS/s during 180 min	7.00	Unstable	Infinity	100.00%

Table 28: Average service rate of the SMSC, when sending 7 SMS/s during variable period of time.

For the two first cases (as the table shows) the average service rate of the queueing system is larger than the arrival rate, the utilization is about 82%, and the number of messages in the queue N is constant. To sum up, the queueing system is stable. But when the duration period is increased to 180 minutes the service rate becomes almost as large as the arrival rate, and the queue tends to grow into infinity (division by zero). This means that the SMSC has become slower, and consequently messages must wait longer, which in its turn will cause message losses, and eventually leads up to the system crash.

The performance testing of the entire system demonstrates that the SMSC is capable of handling the desired workloads.

6.3.8 Conclusion of SMSC Testing

As I mentioned before, Tele2 wanted to see the behavior of the SMSC system in two different scenarios:

1. How many messages the SMSC can process when it is running under the maximum load (i.e. 10 SMS/s) during five minutes. This test is very important for verification of the performance guaranteed by the vendor.

2. How many messages the SMSC is capable of handling when it is running under 70% of the peak load (i.e. 7 SMS/s) during two hours.

Our testing regarding the second scenario proved that the SMSC system is actually able to process this amount of traffic excellently. The server system behaves very well and all measured values show that the system is stable.

Regarding the cases when the SMSC crashed, we saw that the number of received acknowledgements were higher than the number of received SMS-messages from the server, since a number of messages were discarded by the SFE. At the first glance one may think that the SFE can be the bottleneck of the server system (which is true), and should be replaced. But in order to increase the performance of the server it is not enough to put a faster SFE (hardware) there, since the performance of the SFE is very close to the performance of the EI (i.e. the server system is well dimensioned). In order to improve the performance more powerful components must replace both the EI, and the SFE.

In order to reject or not reject the vendor's statement about the maximum performance, we need to use some statistical methods in order to determine the number of trials. To do so we will perform three different calculations.

The first method is called significance testing, which is used for calculation of the number of trials. To start the calculation we need first to introduce a number of parameters.

- H0 (Null hypothesis): The SMSC cannot handle 10 SMS/s during 5 minutes.

- $\theta_0 = 0.5$ is the probability of accepting H0.
- $\alpha = 0.05$ (Type I Error rate): This is the probability of rejecting H0 (by mistake), when it is actually true, i.e. α=P(Reject H0, when it is true)=0.05.
- X (stochastic variable) is the number of observed successful trials.
- H1: The SMSC is capable of processing 10 SMS/s during 5 minutes.
- $\theta_1 > 0.5$ is the probability of accepting H1.
- β (Type II Error rate): This is the probability of rejecting H1 (by mistake), when it is actually true.
- N is the total number of trials (must be calculated).
- m is the number of successful trials in N independent trials.

The aims are to reject H0 by accepting the H1, and to determine the value of N.

If X > m then X has binomial distribution i.e.

$$X \in \text{Bin}(N, \theta)$$

Let P(X>m) be the probability of rejecting H0, then:

P(X>m) = P(Z>m), where Z is a stochastic variable that has normal distribution i.e.

$$Z \in \text{Normal}(n\theta, \sqrt{n\theta.(1-\theta)})$$

$$\Rightarrow P(X>m) = P(Z>m) = 1 - \Phi\left(\frac{m - n\theta}{\sqrt{n\theta.(1-\theta)}}\right) = g_{(m,n)}(\theta)$$

$$g_{(m,n)}(\theta) = 1 - \Phi\left(\frac{m - n\theta}{\sqrt{n\theta.(1-\theta)}}\right) \qquad \textbf{(Equation 6.3.8.1)}$$

By using the equation 6.3.8.1 and inserting the values of θ_0 and θ_1 we will get an equation system, consisting of two equations and two unknown variables (m and N).

1. α = P(Reject H0 even if it is true) = 0.05 and $\theta_0 = 0.5$ then

$$g_{(m,n)}(\theta_0) \Rightarrow 0.05 = 1 - \Phi\left(\frac{m - (n * 1/2)}{\sqrt{n * 1/2(1-1/2)}}\right)$$

The table of the Normal distribution gives the value of 0.05, which is 1.6449. Insertion of this value gives:

$$1.6449 = \frac{m + 1/2 - n/2}{1/2\sqrt{n}}$$

2. $\gamma = 1 - \beta$, where γ is the statistical power. This means that H0 can be reject with γ% confidence. If $\theta_1 = 0.90$ and $\gamma = 99\%$ then $\beta = 0.01$. The Normal distribution table gives the corresponding value of β, which is 2.3263, then:

$\Phi(0) = ½$ (i.e. if there isn't any error)

$$g_{(m,n)}(\theta_1) \Rightarrow \beta = 1 - \Phi\left(\frac{m - (n * \theta_1)}{\sqrt{n * \theta_1(1-\theta_1)}}\right)$$

$$2.3263 = \frac{\theta_1 n - m - 1/2}{\sqrt{n} * \sqrt{\theta_1(1-\theta_1)}}, \text{ where } \theta_1 = 0.90$$

This equation system gives n = 14.45 (i.e. N = n =14.45). This means that I must run the test 15 times (independently) before I can say something about the performance of the SMSC.

The second calculation is about computing the power of the significance testing (above), by using approximation theory.

- Condition: Reject H0 if x > c (c is a constant that must be calculated)

We know that $P(X > c) \leq 0.05$, and $X \in$ Bin (n, 0.5)

If n =15 and $\theta_0 = 0.5$, then c can be determined by $P(X \leq c) \geq 0.95$ (i.e. 1–0.05=0.95). The Binomial distribution table gives c=10, and P=0.94077

$\Rightarrow P(X \leq c) = 0.94077$

The power of the test can be obtained by (when c=10):

$P(X > 10) = P(15 - X \leq 15 - 10) = P(15 - X \leq 5)$

Since $(15 - X) \in$ Bin (15, 0.1) for $1 - \theta_1 = 0.1$

$\Rightarrow P(15 - X < 4) = 0.98728$ is the answer.

The third procedure (which is very important) computes the confidence interval of the tests when n = 15.

Let P·(1−P) be the variance of the H0, and P = 0.5 the probability of accepting H0, then the confidence interval can be obtained by [PROB]:

$$P^{*} \pm \frac{P.(1-P)}{\sqrt{n}} \text{ When } n = 15$$

The confidence interval (given by the equation) is ±0.065. This value shows that the confidence interval is very narrow, which means that for n = 15 our estimation will be very accurate, and the sampling distribution will be centered around 0.5.

Finally with this background, the test concerning 10 SMS/s during 5 minutes was run (independently) 15 times (I actually run it 25 times, for safety's sake), and in all of the cases could the SMSC handle the generated workload without any losses.

The conclusion (which is based on n = 15) is that the SMSC is capable of processing 10 SMS/s. This leads to the fact that the Null hypothesis can be rejected with 98.7% power, and the H1 will be accepted.

7. Future work

In this paper I presented a high-level view of the SMSC and MMSC servers, modeled as open queueing networks. These models abstract all hardware inside these systems, and the relationship between them. The primary aim of these models is to analyze results regarding the performance of the systems. Two applications software were also created in order to gather data about the performance. Since these models have ignored the impact of the low-level details of the protocols like SMPP, SMTP, and TCP/IP, the future versions may contain these refinements.

At the physical layer and the link layer there are several factors which affect the performance: parameters of the TCP/IP protocol suite, transfer rate, frame size, and bulk data transfer. Thus the impact of the network bandwidth and signal-links on the overall performance should also be studied. Since these factors are the bottleneck in almost all computer networks and radio networks, it would be very interesting to see how these parameters effect the response time of the servers.

The MMSLG application software can be improved, to include functionality for fetching MM-messages from the MMSC.

8. Conclusion

In the past few years the mobile-based services (such as SMS and MMS), and the number of subscribers have growth rapidly. Despite increasing number of MMSC and SMSC servers little is known about their performance characteristics. Server vendors often exaggerate about the actual performance of their products and telecom industry doesn't have access to effective tools for performance analysis.

In this paper the performance evaluation of the SMSC and MMSC platforms were presented. In this study the results of the load testing were analyzed by using queueing theory. This analysis of the results yields several interesting results.

In the case of the MMSC we began our testing with an arrival rate that was as large as the MMSC's specified maximum performance. In this case the MMSC could process incoming messages during just the first minute, thereafter it began to lose messages (messages were discarded inside the system) as the testing duration grew. This happened since the MM-database was not fast enough to store (process) the messages, and consequently the waiting queue grew to its maximum limit (300 messages). We saw also that as the load on the server increased by time, the response time increased gradually up to a level. Thereafter it increased rapidly, and then asymptotically. In this situation the server couldn't process the generated load fast enough, which in its turn results in growing waiting times. Consequently the server cut all the socket connections and restarted. When the arrival rate was 60% of the maximum load for the testing duration 10, 20, and 30 minutes the MMSC worked well, all messages were processed successfully, and the response time curve was stable. By these results we drew the conclusion that the real maximum performance of the MMSC was lower than the guaranteed performance by the vendor (i.e. 5 MM/s), and the performance of the server could be improved by replacing the MM-database with a faster one.

In the SMSC performance evaluation we began our testing with an arrival rate that was as large as the maximum performance. In this case we saw that the response time of the SMSC had a stable behavior when the testing period was increased from 5 to 20 minutes. But when the testing period was increased to 30 and then 40 minutes the response time began to increase gradually up to a point, thereafter it increased exponentially. In this situation the SMSC had been plunged into deadlock (loop), where it attempts to serve more and more messages at slower and slower speeds such that no messages were successfully processed. In this situation the number of correctly received acknowledgements was higher than the number of correctly received SMS-messages from the SMSC (i.e. a number of messages was discarded inside the system). This happened since the SFE was slower than the EI, and that the waiting queue (AMQ) grew to its maximum limit. When the arrival rate was decreased to 70% of the maximum performance the SMSC worked well, and the response time had a stable behavior for the periods 60, and 120 minutes. But when the testing duration was increased from 120 to 180 minutes the response time began to increase gradually (at the beginning), and then exponentially. In this situation the SMSC had been plunged into saturation. The conclusion was that in order to improve

the performance of the SMSC both the EI and SFE must be replaced by faster components, since the performance of the EI and SFE were very close to each other. The statement of the vendor concerning the maximum performance of the server (i.e. 10 SMS/s) was also verified, and accepted.

9. References

[AMR] Third Generation Partnership Project, "*Technical Specification Group Services and System Aspects, Mandatory Speech Codec Speech processing functions, Adaptive Multi-Rate speech codec, Transcoding functions*", 3GPP TS 26.090, Jun 2002, URL: http://www.3gpp.org/ftp/Specs/archive/26_series/26.090/

[COMOP] Comverse Network System, "*Operation and Maintenance Manual*", September 2001, URL: http://www.comverse.com

[COMTECH] Comverse Network System, "SMS Technical Bulletin", Mars 2002, URL: http://www.comverse.com

[IEC] International Engineering Consortium, "*Wireless Short Message Service (SMS)*", April 1999, URL: http://www.iec.org/online/tutorials/wire_sms/index.html

[Logica] Logica, "*SMPP Documentation*", 11-28-2001, URL:http://opensmpp.logica.com/CommonPart/Documentation/Documentation.htm

[Micromuse] Micromuse Inc., "Netcool/Wireless Service Monitor", Micromuse Inc., 2002-09-18, URL: http://www.micromuse.com/products/product_information.html

[MMSCON] Nokia Forum, "*MMS Conformance document Version 2.0.0*", 04-30-2002, URL: http://www.forum.nokia.com/main/1,,1_2_7_1,00.html

[NM] Alan Zeichick, "*Signaling System 7*", Network Magazine, 01-12-1998, URL:http://www.networkmagazine.com/article/NMG20000727S0020

[OPP] Open Path Products, "*SMS Testing Suite v2.0*", April 2002, URL: http://www.openpathproducts.com/OpenPath.jsp?content=products_tools

[PROB] Gunnar Blom, "*Probability and Statistics: Theory and Applications*", Student literature, ISBN: 91-44-03594-2, forth edition, 1999

[QMF] Ng Chee Hock, "*Queueing Modelling Fundamentals*", John Wiley & Sons Ltd, ISBN: 0471 968196, 1997

[RFC0821] J. Postel, "*Simple Mail Transfer Protocol*", 08-01-1982, URL: http://www.ietf.org/rfc/rfc0821.txt?number=821

[RFC2045] IETF; "*Multipurpose Internet Mail Extensions (MIME) Part One: Format of Internet Message Bodies*", URL: http://www.ietf.org/rfc/rfc2045.txt

[RFC2387] E. Levinson, "*The MIME Multipart/related content type*", August 1998. URL: ftp://ftp.isi.edu/in-notes/rfc2387.txt

[RFC2822] IETF; STD 0011, "*Internet Message Format*", URL: http://www.ietf.org/rfc/rfc2822.txt

[SMIL] W3C, "*Synchronized Multimedia Integration Language (SMIL) Boston Specification*", Working Draft SMIL 2.0, September 2000, URL: http://www.w3.org/TR/smil20/

[SMPPD] SMS Forum, "*SMPP Protocol Specification v3.4*", URL: http://smsforum.net/doc/public/Spec/

[SMSF] SMS Forum, URL: http://smsforum.net

[SYMSOFT] Symsoft AB, "*Symsoft Software Specification*", October 2002, URL: http://www.symsoft.se/Files/MMSC.html

[SYMHARD] Symsoft AB, "*Symsoft Hardware Composition*", October 2002, URL: http://www.symsoft.se/Files/MMSC.html

[TEM] Temia, "*GSM/GPRS software*", Jun 2002, URL: http://195.124.167.141/en/products/te966/gsm_gprs.htm

[3GPPTS] Third Generation Partnership Project, "*3GPP Technical Specification 23.140 v5.4.0",* September 2002. URL: http://www.3gpp.org

[TSG] Nokia, "*TSGT2_04*", submitted to 3GPP, 1999-06-16, URL:http://www.3gpp.org/ftp/tsg_t/WG2_Capability/TSGT2_04/Docs/T2-99500.doc

[TCP/IP] W.Richard Stevens, "*TCP/IP illustrated, Volume 1*", Addison-Wesley Publication Co, ISBN: 0201633469 1st edition, January 1994

[WAPA] WAP Forum, "*WAP Architecture Specification*", WAP-210-WAPArch-20010712, 2001-07-12, URL: http://www1.wapforum.org/tech/documents/WAP-210-WAPArch-20010712-a.pdf

[WAP205] WAP Form, "*WAP MMS Architecture Overview*", WAP-205 MMS Architecture Overview, 2001-04-25,
URL: http://www1.wapforum.org/tech/terms.asp?doc=WAP-205-MMSArchOverview-20010425-a.pdf

[WAP206] WAP Forum, "*WAP MMS Client Transactions*", WAP-206 MMS Client Transaction, 2002-01-15,
URL: http://www1.wapforum.org/tech/terms.asp?doc=WAP-206-MMSCTR-20020115-a.pdf

[WAP209] WAP Forum, "*WAP MMS Encapsulation Protocol*", WAP-209 MMS Encapsulation Protocol, 2002-01-05,
URL: http://www1.wapforum.org/tech/terms.asp?doc=WAP-209-MMSEncapsulation-20020105-a.pdf

[WAPWSP] WAP Forum, "*Wireless Application Protocol, Wireless Session Protocol Specification*", WAP-203-WSP Specification, 2000-06-20,
URL: http://www1.wapforum.org/tech/terms.asp?doc=WAP-203_001-WSP-20000620-a.pdf

[WTI] xNet Project, "*Simulation of Wireless Application Protocol*", July 2001, URL: http://quark.it.iitb.ac.in/~xnet/wap/default/first-page.html?46,16