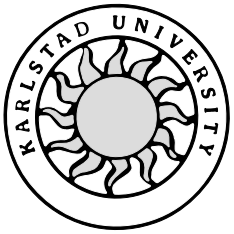



Computer Science

**Lisa Andersson and Katarina Nilsson**

# Improvement of an SS7 Protocol Tool Developed in Java

Bachelor's Project

2004:15

# Improvement of an SS7 Protocol Tool Developed in Java

**Lisa Andersson and Katarina Nilsson**

This report is submitted in partial fulfillment of the requirements for the Bachelor's degree in Computer Science. All material in this report which is not our own work has been identified and no material is included for which a degree has previously been conferred.

Lisa Andersson

Katarina Nilsson

Approved, 2004-06-03

Advisor: Per Strömgren

Examiner: Stefan Lindskog

# Abstract

The main area of TietoEnator Telsys AB is development of Signaling System No 7, a SS7 stack, an international protocol and worldwide standard for telecommunications.

The purpose of this Bachelor's Project was to improve an existing tool that traces a log file produced by the SS7 stack. TietoEnator Telsys AB had stated improvements for the trace tool. The goal was to implement as many as possible of these improvements and list further requirements for the existing tool. Two of the listed assignments, Terminate Reading Log File and Save Filter Options, were implemented and a number of new improvements were documented.

This Bachelor's Project at TietoEnator Telsys AB has resulted in wisdom about assignments that can occur when software is maintained and improved. The experience consists of both problems and pleasures.

# Contents

# List of Figures

# 1 Introduction

This Bachelor's Project is a 10-point course at the Department of Computer Science at Karlstad University. The Bachelor's Project has been performed by Lisa Andersson and Katarina Nilsson at TietoEnator Telsys AB in Karlstad and the work began 2004-01-21.

The first contact was taken in November 2003. A meeting was held, where the company was briefly presented and our applications were submitted. After three weeks an assignment was defined, which implied improvements of an existing tool developed in Java. We found the assignment interesting because Java was a new programming language. Also learning about Signaling System No 7 was a challenge and an excellent opportunity to apply our theoretical knowledge in reality.

The tool that should be improved is used by designers, testers, and people who work with maintenance and support. It is a graphical tool that presents a view of the internal traffic within the SS7-stack i.e. messages exchanged between the different layers within the SS7 stack.

This report describes the purpose and the goal of this Bachelor's Project as well as our approach and results. The approaches are presented in the chapter Work Packages.

# 2 Purpose and Goals and Background

This chapter describes the purpose of this Bachelor's Project, gives an overview about Signaling System No 7, Ericsson Infotech's Portable SS7 and the Protocol Tool that has been improved. It also contains a concise presentation of TietoEnator AB.

## 2.1 Purpose and Goals

The purpose of this Bachelor project was to improve an already existing SS7 protocol tool developed in Java. TietoEnator AB had stated improvements for the Trace Viewer Tool, TVTOOL, in a document. See Appendix F. The advisor at TietoEnator AB also requested documentation of recommendations for further improvements of TVTOOL.

## 2.2 Background

In this chapter the organization of TietoEnator AB is briefly presented to get an understanding why the company maintains the Ericsson SS7 product. Further the protocol SS7, Ericsson Infotech's Portable Signaling System No 7 and the programs Trace Viewer, TV, and TVTOOL, are introduced in this section.

### 2.2.1 TietoEnator AB

TietoEnator AB is one of the architects in building a more efficient Information Society with about 14000 employees operating in 20 countries. TietoEnator AB focuses on the following areas; Banking and Finance, Telecom and Media, Public and Healthcare, Production and Logistics, see figure 2-1. TietoEnator AB supplies services to the Telecom and Media industry in Europe. In the Telecom and Media sectors TietoEnator AB employs 2800 people in China, Kazakhstan and nine European countries. TietoEnator AB's headquarter is located in Esbo, Finland.

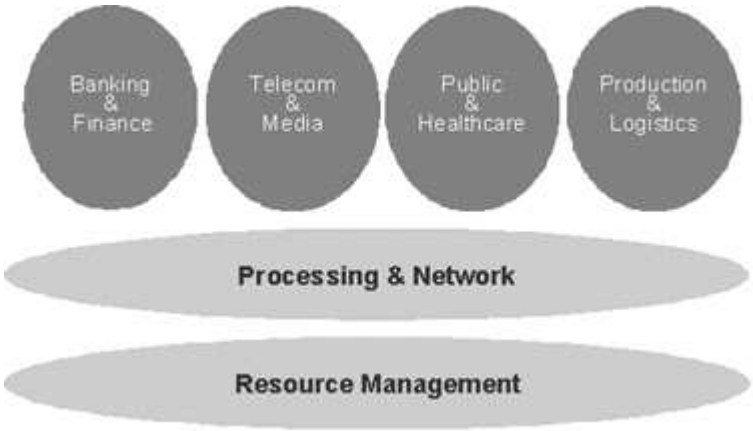



*Figure 2-1 Organization of TietoEnator*

### 2.2.2 TietoEnator Telsys AB

The history of TietoEnator Telsys AB started 1981, see figure 2-2. The company name was at that time Ericsson Programatic Sweden AB, and was intended as a software company. The ownership was 50 % L M Ericsson and 50 % Programator. The company was managed by Programator. In 1992 Ericsson Programatic Sweden AB introduced an activity to develop their own products. One of these products was the Portable SS7. In 1993 Ericsson Programatic Sweden AB became Programatic Sweden AB. A year later the company became Ericsson Infocom Consultants AB. In 1997 the company changed its name again to Ericsson Infotech AB. Ericsson Infotech AB became a part of Ericsson AB in April of 2002 and this part was sold to TietoEnator AB in November of 2002.

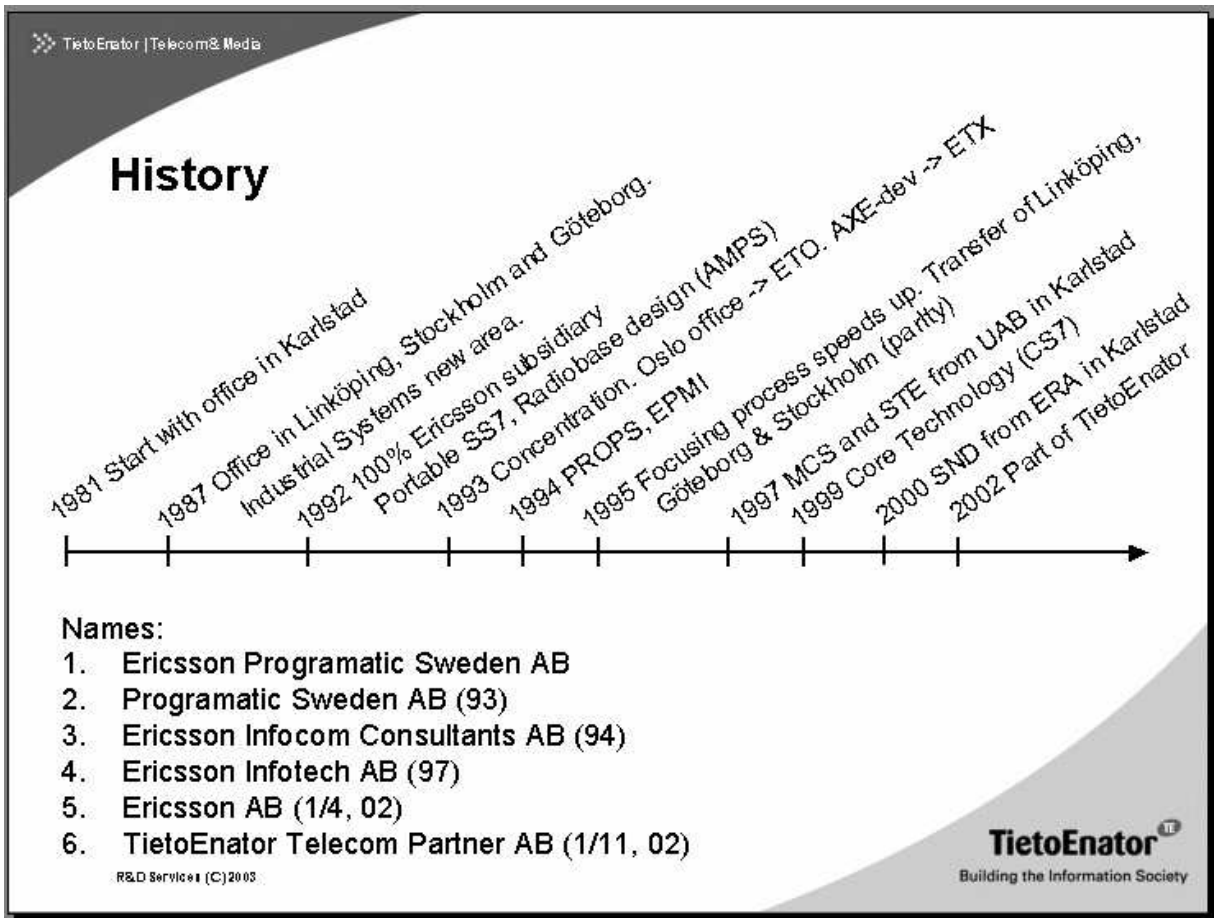



*Figure 2-2 History of TietoEnator Telsys AB*

TietoEnator Telsys AB is a division within the R&D services organization. Their office is located in Karlstad. Telsys includes five sub Business Units, SBUs. Signaling Products, Service Network Development, Radio Systems Development, Maintenance & Customer Support and Signaling Solutions.

- Signaling Products – responsible for developing SS7 for Ericsson. This SBU is the sole supplier of signaling on open platforms, for Ericsson.
- Service Network Development – responsible for development of Service Networks products for Ericsson. This SBU develops the platform on which Intelligent Network-services, IN-services, are based.
- Radio Systems Development – responsible for development of parts of the radio base station within Ericsson.
- Maintenance & Customer Support – develop, support, and provide Ericsson with an infrastructure within maintenance and customer support of Telecom systems.

- Signaling Solutions – responsible for marketing and sales of signaling solutions to customers worldwide outside of Ericsson.

### 2.2.3 SS7 – Signaling System No. 7

Signaling for telecommunication consists of two essential components, the data that should be exchanged between endpoints and the information that instructs the communication line to establish the connection and route the data to the appropriate destination.

Common Channel Signaling System No. 7 (SS7 or C7) is an international protocol and worldwide standard for telecommunications. SS7 defines the procedures and the protocols by which network elements in the telephone network exchange information. The signaling process is using a digital network for executing functions such as call setup, routing and control of wired and wireless calls.

The forerunner of SS7 signaling methods was Channel Associated Signaling, CAS. It was inefficient, because all telephone connections were managed by different techniques based on "in band" signaling, see figure 2-3. This meant that all telephone connections required a bearer trunk to transport the data between sender and receiver; a single wire reserved for the connection. The wire carried both data and signaling information necessary for the connection. Even if the receiver was unable to accept the incoming call, a complete bearer was reserved from the caller to the destination point.

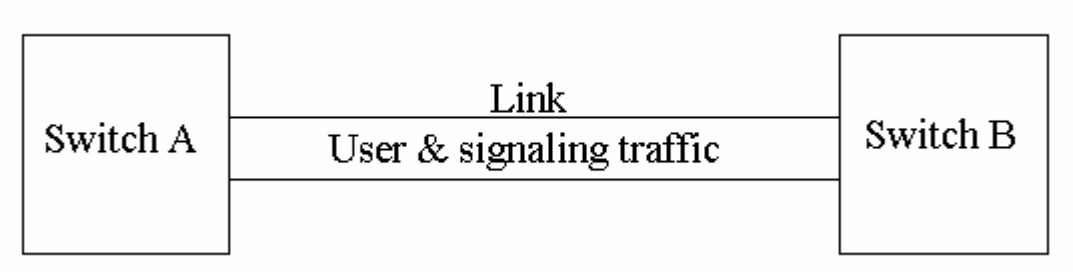



*Figure 2-3 In Band Signaling*

The next generation of signaling used a more resource effective way to establish connections. The wire was not reserved until the connection was accepted and the sender and the receiver could exchange data. This meant that valuable resources were occupied only during data

transmission. This concept, separating the data transmission from the signaling traffic, is called Common Channel Signaling, CCS, or “out of band signaling”, see figure 2-4.
Another important advantage of CCS is the essentially independency of switch and transmission technology.

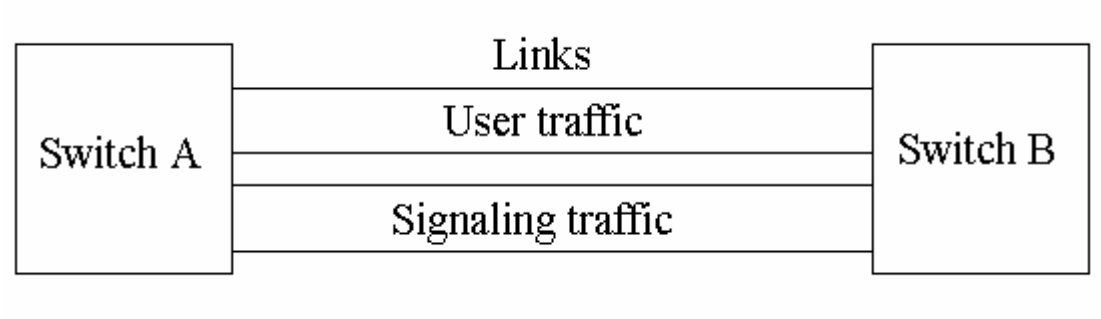



*Figure 2-4 Out of Band Signaling*

SS7 is a highly sophisticated and powerful form of CCS. Due to the independency of CCS, the development of SS7 standards is not affecting the underlying equipment and vice versa. SS7 is responsible for routing calls across countries, between countries, and has a central role in mobile networks. SS7 has a layered architecture. Each layer has a specific role and responsibility, see figure 2-5.

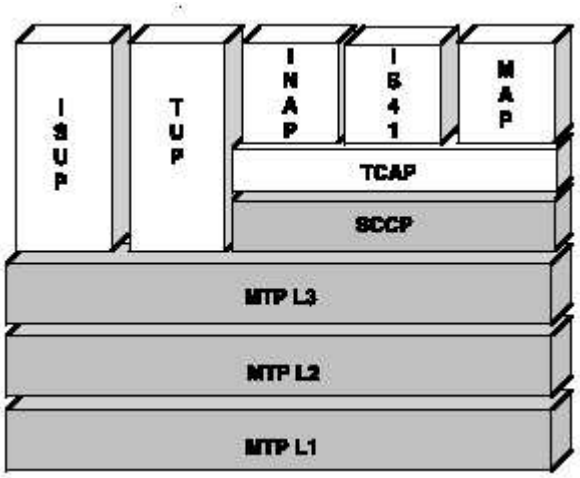



*Figure 2-5 Overview of a SS7 Stack*

The three lowest layers consist of Message Transfer Part, MTP. The main purpose of the MTP is reliable routing of messages and managing the SS7 links. Together the three layers are responsible for secure robust routing of messages. The links enables communication

between Signaling Points, SP, within a network. These SS7 links are fundamental for the security features and make the connection recoverable.

MTP Layer 1 corresponds to the physical layer of the Open System Interconnection, OSI, model, see figure 2-6. This layer is responsible for the connection of the entities in the network over which the SP's communicate. MTP Layer 1 will also handle the conversion of messaging into electrical signals and maintenance of the physical links.

MTP Layer 2 supplies reliable data transmission. It examines transmitted data, checks for errors and corrects them if possible. When a large set of data is sent, this layer handles the flow control. The flow control makes sure that neither side of a connection overwhelms the other by sending too many packages too fast. Further Layer 2 creates the outgoing messages into packages so called signaling units. There are three types of messages that Layer 2 creates, Message Signal Unit, MSU, Link Status Signal Unit, LSSU, and Fill-In Signal Unit, FISU. For a more detailed information about the three message types, see Appendix B Signaling Units.

MTP Layer 3 is responsible for message routing. MTP Layer 3 will also handle the control of traffic routing, the links which bear the traffic and errors.

The protocols above Layer 3 are, see figure 2-5:

- Signaling Connection Control Part, SCCP
- Integrated Services Digital Network User Part, ISUP
- Telephone User Part, TUP
- Intelligent Network Application Protocol, INAP
- Electronic Industries Association Interim Standard 41, IS41
- Mobile Application Part, MAP
- Transaction Capabilities Applications Part, TCAP

These Layer 4 protocols are presented in section 2.2.4.

The layers at level four are either circuit related protocols (e.g. ISUP and TUP) or non-circuit-related protocols (e.g. SCCP).

- Circuit-related protocols – the resources needed along a path e.g. buffers, link bandwidth, are reserved for the duration of the communication session, [8].
- Non-circuit related protocols – a session's messages use the resources on demand, i.e. there are no resources reserved. As a consequence, the transmission has to wait for access to a communication link.

For further information see [1] and [11].

The SS7 Protocol correspond to physical, data link, network and application layer in the OSI see figure 2-6

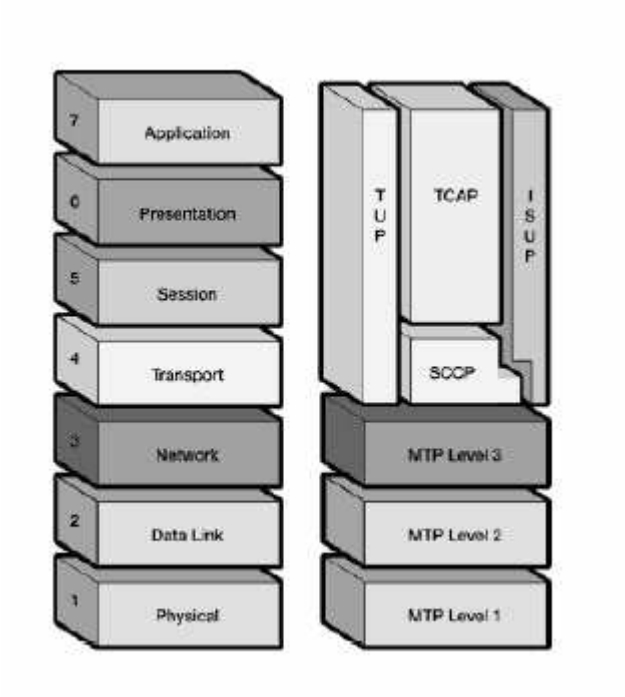



*Figure 2-6 OSI-model compared to SS7*

### 2.2.4 Ericsson Infotech's Portable Signaling System No 7

Ericsson Infotech's Portable SS7 is an implementation of the SS7 protocol and can be used on several operating systems. The product is intended for Computer Vendors, System Integrators, Network Operators and Equipment Manufactures, i.e. companies who need to interconnect with telecom networks. Ericsson Infotech's Portable SS7 has been split into functional blocks to support the different standards, i.e. International Telecommunication Union, ITU-T, the standard of Europe, American National Standard Institute, ANSI, the US

standard and finally The Telecommunications Technology Committee, TTC, the available standard in Japan.

Figure 2-7 shows which functional blocks that are supported for each standard.

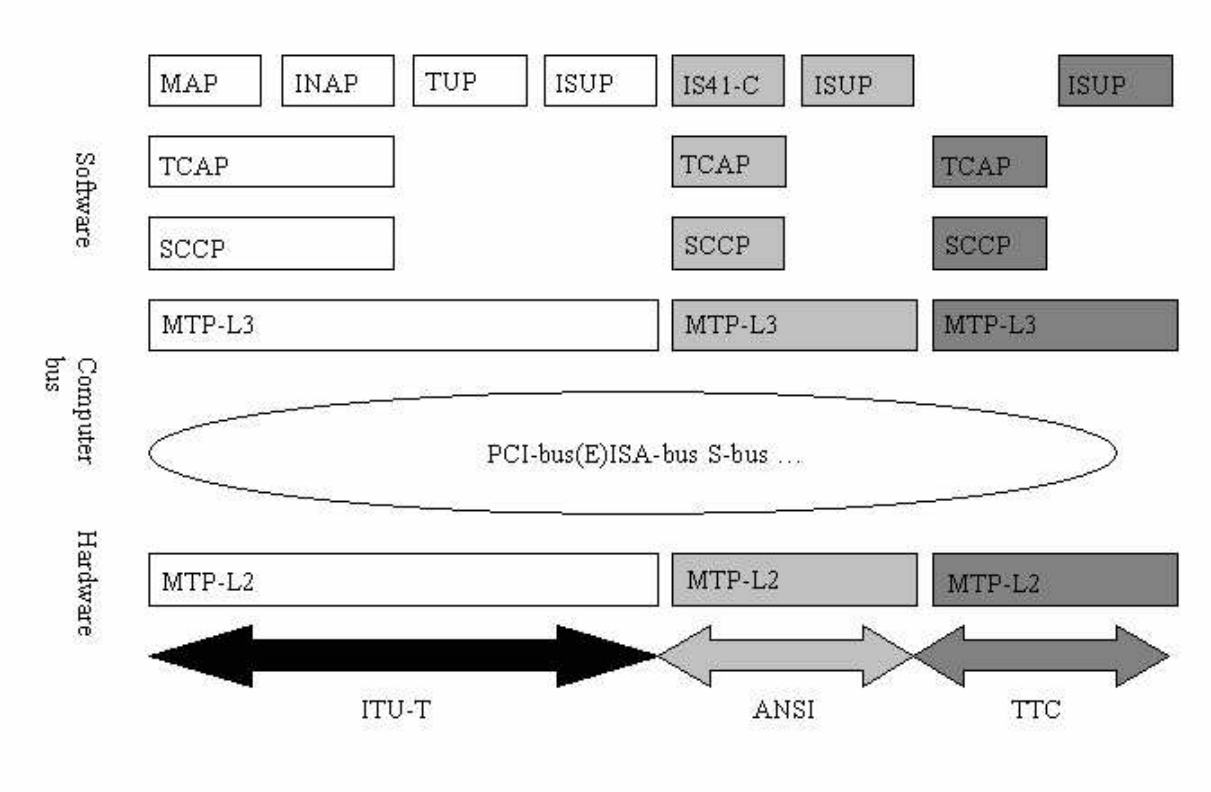



*Figure 2-7 Support of Portable SS7*

The management of Ericsson Infotech's Portable SS7 can be solved in a number of different ways depending on the required management functionality and the possibility to coordinate with existing management system and application.

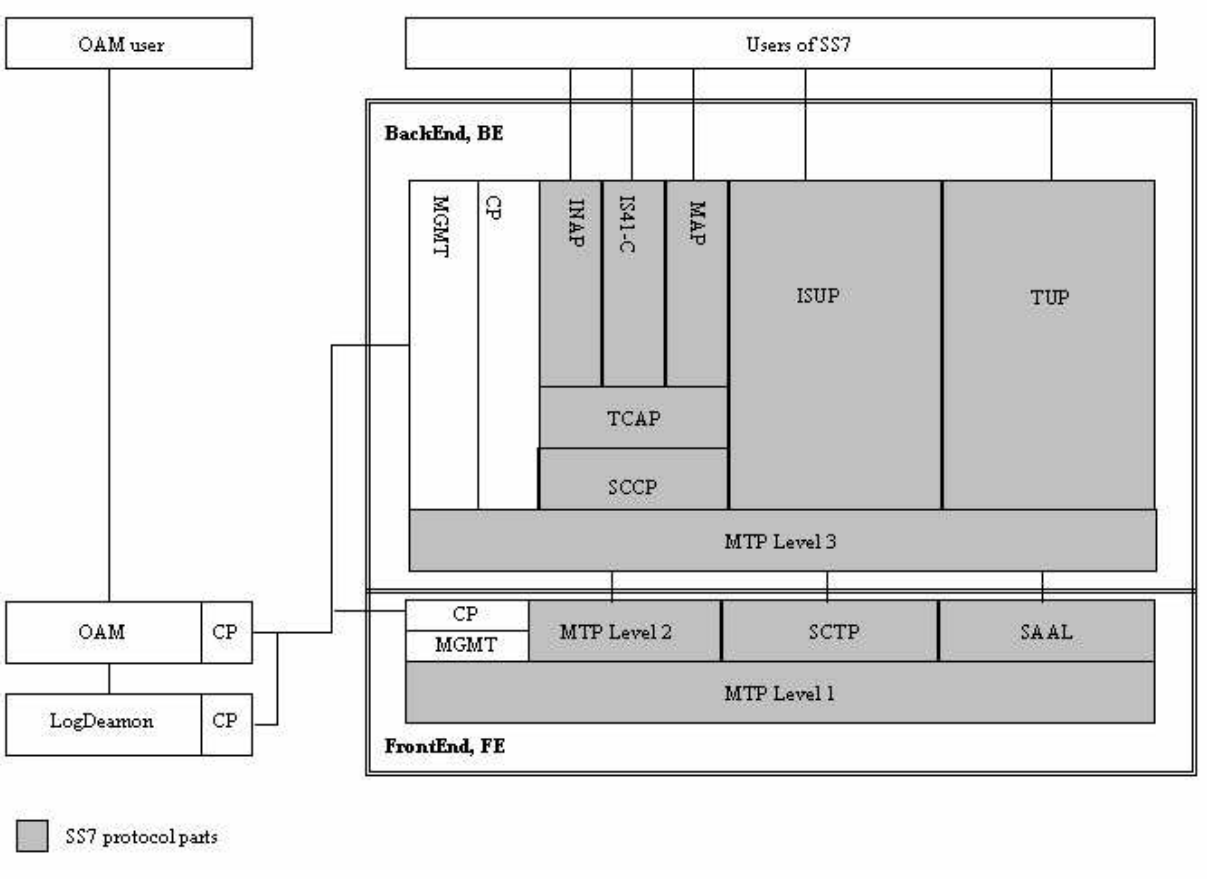



*Figure 2-8 SS7 layer module; Portable SS7*

## SS7-layers

In this section, each module in the SS7 stack is presented, see figure 2-5.

- **MTP-L1** is described in section 2.2.3.
- **MTP-L2** is described in section 2.2.3.
- Stream Control Transmission Protocol, **SCTP**, is an end-to-end, connection-oriented protocol that transports data in independent sequenced streams. SCTP provides applications with enhanced performance, reliability and control functions. This protocol is essential where detection of connection failure and associated monitoring are mandatory.
- Signaling ATM Adaptation Layer, **SAAL**, is equivalent to MTP-2 for Asynchronous Transfer Mode, ATM, high-speed links. ATM is a technique for digital transmission of cells with a defined size, e.g. pictures, voice, video, sound and data transmission.

- **MTP-L3** is described in section 2.2.3.
- **SCCP** provides an OSI Network layer service, i.e. network routing, flow control, segmentation/reassembly of packages and error control, to its users (i.e. TCAP). The SCCP software package is intended for use in an end-node, i.e. intermediate routing is not supported.
- **TCAP** offers the transaction control service. It is used by applications such as messaging in mobile networks and in IN applications involved in data transactions.
- **INAP** specifies the information flow that is exchanged between different entities of the IN functional model.
- **IS41-C** provides mobile operators a format to perform intersystem hand-off and roaming operations for their subscribers. It is the USA standard, equivalent of Global System for Mobile communications, GSM, for the European Digital Cellular Phone system
- **MAP** provides signaling functions for mobile voice and non-voice applications on mobile networks. The main task of MAP is to exchange information with mobile network units and allow a mobile station to roam around freely.
- **ISUP** handles basic call set-up and tear down. It is used in e.g. voice mail and voice response system.
- **TUP** defines the international telephone call control signalling function to be used over SS7 network.

### Ericsson Infotech's Portable SS7 specific implementations

In this section, Ericsson Infotech's Portable SS7 specific modules are briefly described, see figure 2-8.

- Common Parts, **CP**, provide communication interface between the layers and interaction with the OS and other processes in the system.
- The Operation, Administration and Maintenance module, **OAM**, process provide a single access point for managing the system. The OAM distributes orders, statistics and alarm requests to the correct handling process within the system.

- The **LogDeamon** function logs all data to the file system in a log file. The log file provides detailed information about messages such as time stamps, sender/receiver, primitive name, message data etc.
- The Management, **MGMT** or Management Module, **MM**, acts as an interface between the SS7 functional blocks for different protocol servers and the external user of the protocol stack. When the management module receives a message from the external module, it will set the destination and distribute the message to the correct SS7 functional block.
- **SS7 user** serves as the application user interface for SS7 system users. It consists of Application Programming Interface, API's, linked together with application specific SS7 interface software.
- **OAM user** is an interface for the SS7 Horizontal Distributed, HD, system management applications. It provides and activates configuration, starts the system, initiates orders, requests statistics and subscribes alarms.

## Configurations of Ericsson Infotech's Portable SS7

Ericsson Infotech's Portable Signaling System No 7 can be created in three different basic configurations, Single SS7 stack, High Availability SS7 stack, HA and HD SS7 stack.

- **Single portable SS7.** One SS7 stack where Frontend, FE (MTP-L1 and MTP-L2), and Backend, BE (MTP-L3 and Layer 4), i.e. the whole stack, are running on the same SS7 Server. Another solution; is FE running on one SS7 Server and BE is running on another. A disadvantage with this solution is if the stack breaks down, the whole system will stop.
- **High Availability SS7**. It is possible to use the portable SS7 stack with two SS7 Servers, and use one as active and the other as "cold" stand-by, i.e. the Master and Slave concept. The advantage of this configuration is the possibility to switch over to the Slave server during e.g. maintenance and upgrading.
- **Horizontal Distributed SS7 stack**. More than one SS7 stack is running in parallel, one BE on one SS7 Server can use one FE of another SS7 Server, if congestion occurs. The advantage of this solution; the resources are used in an

efficient way and congestion will be avoided. Figure 2-9 is an overview of a Horizontal Distributed SS7 stack. Each box in the figure corresponds to a module. Several modules are linked together into processes within a certain function. For further information about SS7 and Portable SS7 [6] and [7]. Detailed information about the modules in the FE and BE processes is found in Appendix. C.

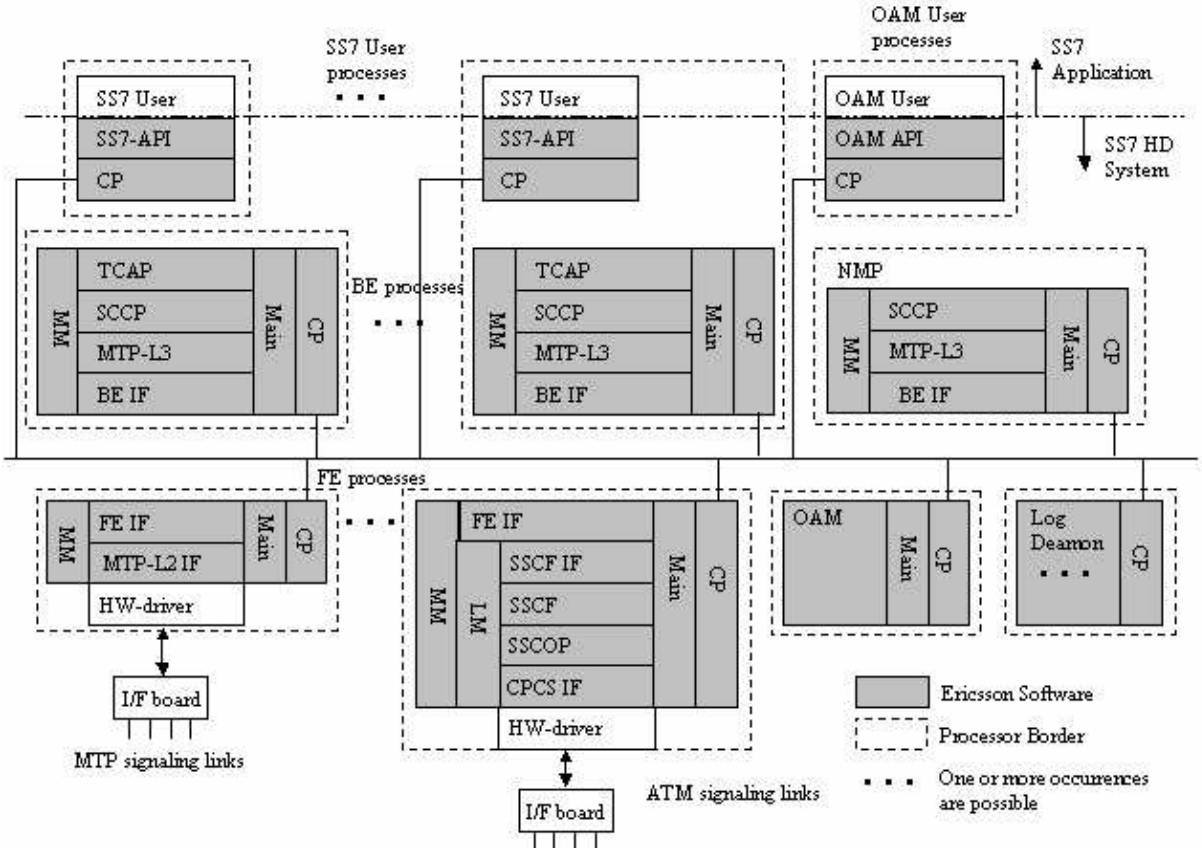



*Figure 2-9 Horizontal Distributed SS7 stack*

### 2.2.5 Logging Tools

While the SS7 protocol is running, the LogDeamon writes information about events and unexpected failures to a trace log file. The trace log file may either be in hexadecimal or decimal format. TV and TVTOOL are programs that make it possible to trace events in the SS7 stack, for example sending and receiving messages between layers in the SS7 stack.

Trace Viewer –TV

All log information sent between the different protocols in the SS7 stack is written to a trace log file by the LogDeamon, see figure 2-11. Interpreting this hexadecimal or decimal output by hand is a very time-consuming task. This is the reason why TV was developed. TV parses and converts the SS7 specific trace log files into plain text, see figure 2-10. The plain text can be written to a text file or to the screen.

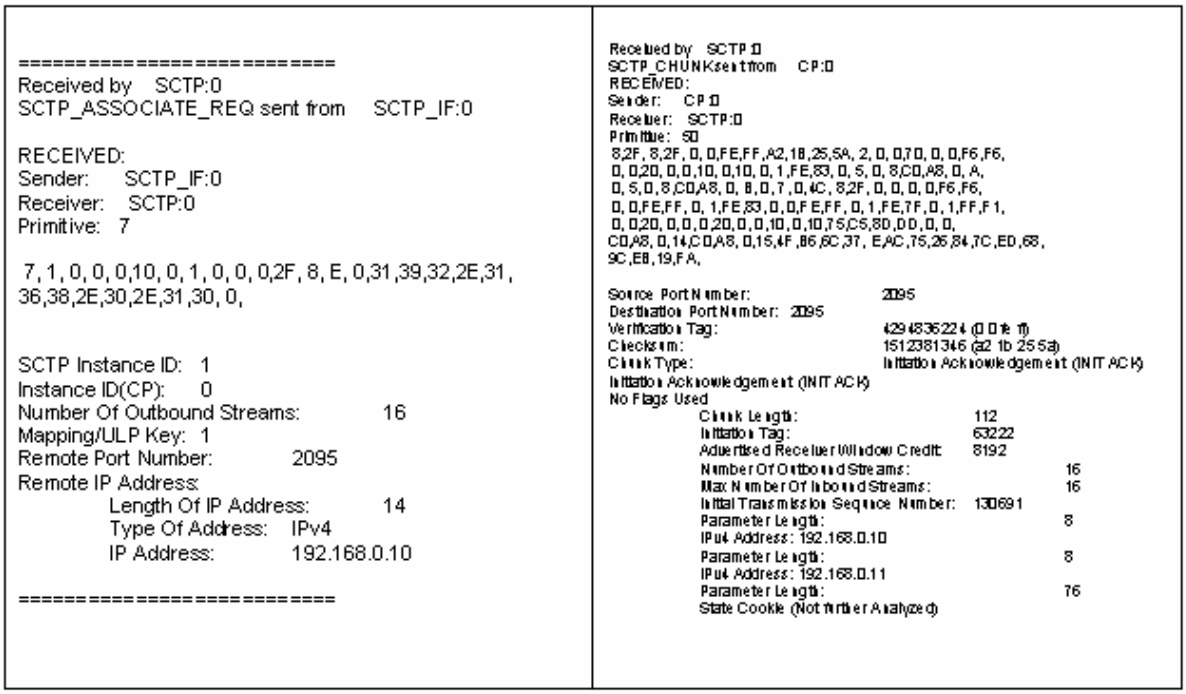



*Figure 2-10 Example of the output for TV*

The test efficiency and test quality will increase by using TV. It is also easier to track errors. TV is successfully used during design, test, support and maintenance. It gives a good overview of small trace log files, but large log files can be complex.

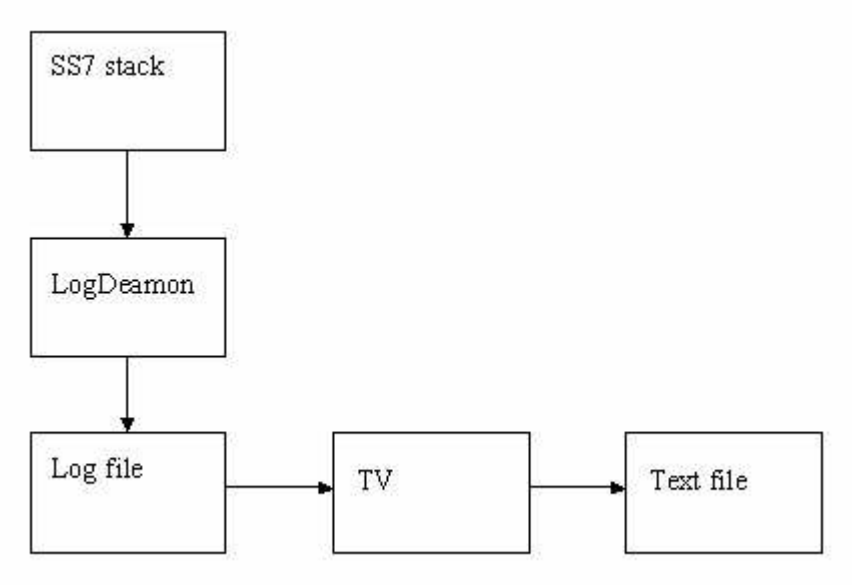



*Figure 2-11 TV tracing log file*

## Trace Viewer Tool – TVTOOL

TVTOOL creates a graphical view of messages or primitives that are sent internally within a SS7 stack. There are two different techniques to create a graphical view. The first technique is to create the view from the trace log file that the LogDeamon has created, see figure 2-12.

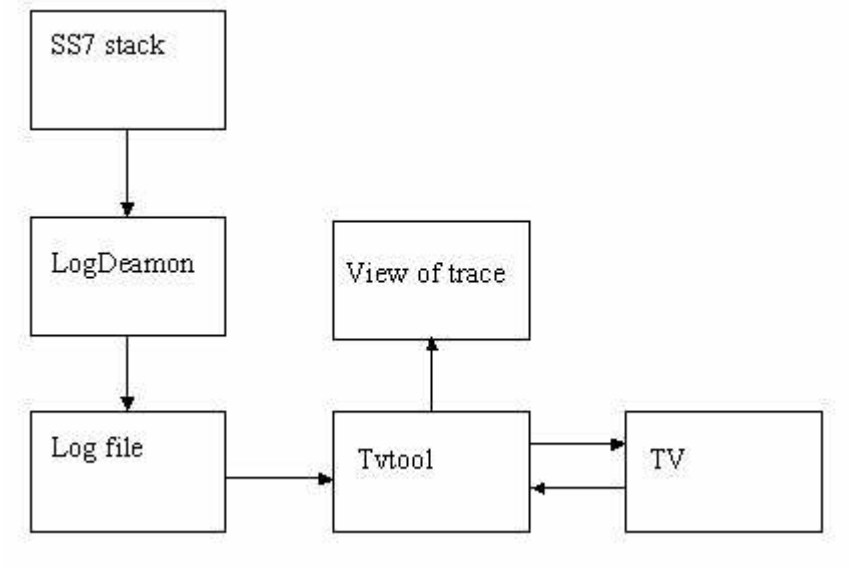



*Figure 2-12 Graphical view created from the LogDeamon*

Using the second technique, TVTOOL is used in real time, i.e. TVTOOL replaces the LogDeamon function and reads directly from the SS7 stack via an User Datagram Protocol, UDP, socket, see figure 2-13.

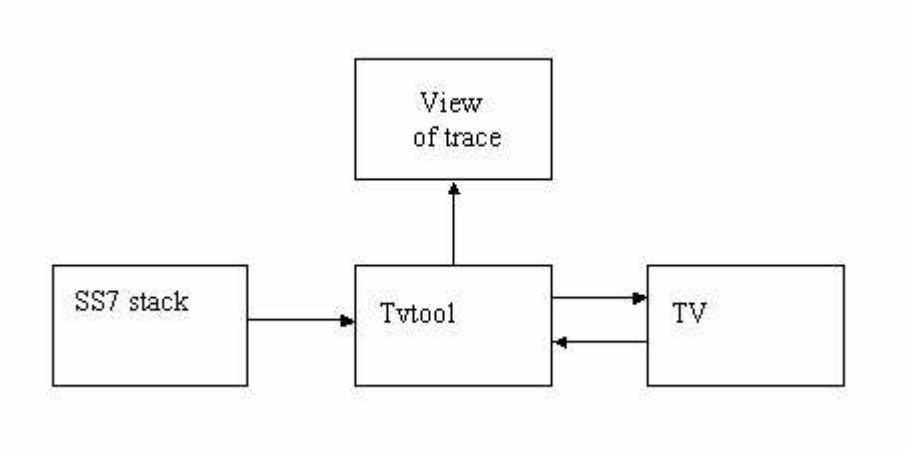



*Figure 2-13 The graphical view created in real time*

TVTOOL is a sophisticated tool, but it is not useful without TV. TV translates the sender, receiver and error primitives from hexadecimal or decimal format to plain text.

- The sender primitive contains information about the sending protocol.
- The receiver primitive corresponds to the addressed protocol.
- The error primitive specifies which state and event that have occurred.

If TVTOOL does not recognize a primitive when it displays the graphical view, a request is sent to TV for translation. The basic idea with this design is that all dependencies are located in TV, leaving TVTOOL independent i.e. TVTOOL can be used on different stacks.

When a trace log file is loaded, the user can see which protocols have exchanged primitives in the SS7 stack. The user has the possibility to get additional information about the primitive by clicking on the arrow displaying the current primitive. (The left part of the figure 2-14.) TVTOOL sends a request to TV, which translates the primitive to plain text and TVTOOL displays the result, see figure 2-14 to the right. TVTOOL uses TV's translations methods concerning sender-, receiver- and error primitives. The user can receive additional data about Trace, Xtrace, Timer and Debug. TVTOOL fetches this information directly from the trace log file or from the UDP socket.

- The information of Trace and Xtrace primitives resemble the Error primitive without being an error. See description of Error primitive earlier in this section.
- The Timer secures that the primitives reach their destination protocol. If the timer is expired the primitive is sent again.
- Debug is an extension of Xtrace. The user can trace on different levels.

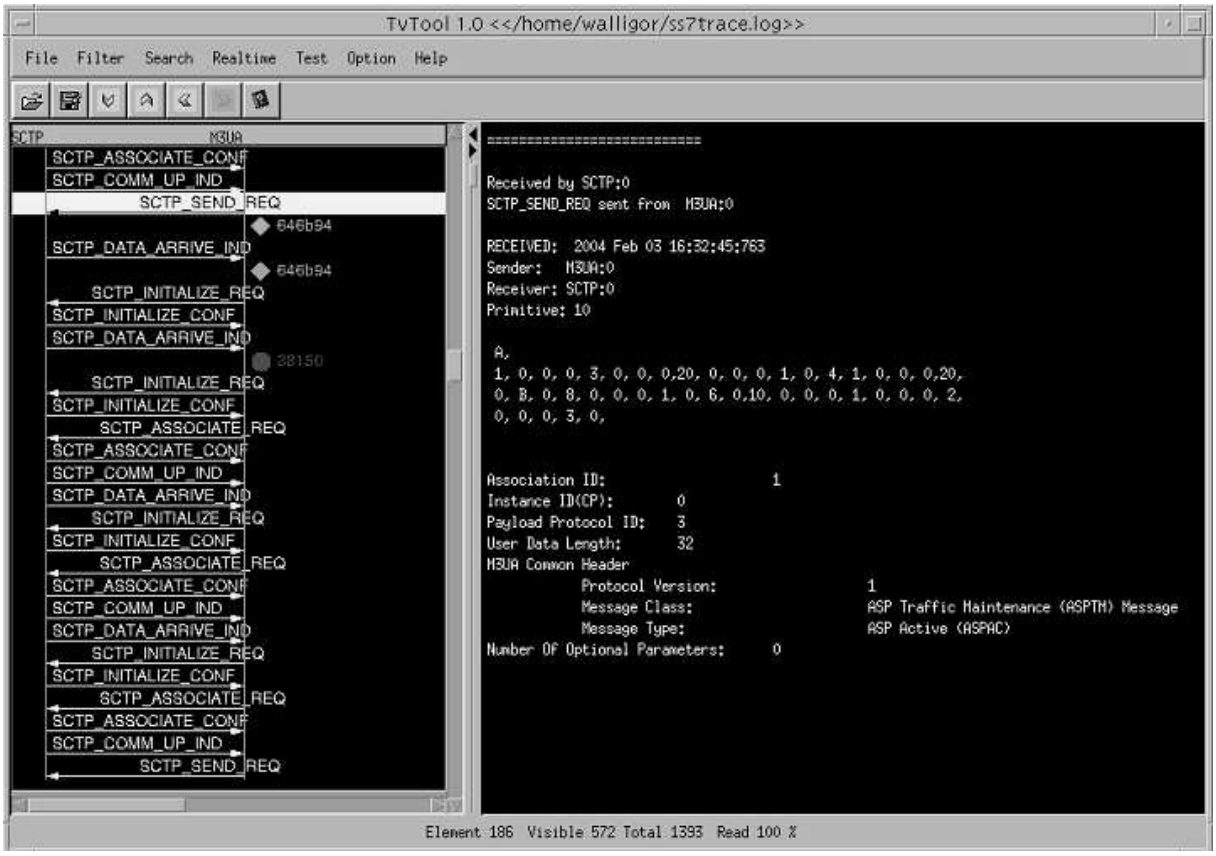



*Figure 2-14 View of TVTOOL*

Because TVTOOL is a graphical tool it gives a better view of the SS7 stack's condition, compared to TV that presents its result in plain text. TVTOOL has an additional functionality to filter the trace log file information on different levels, i.e. the user can filter the information by protocol, message, type, instance or by primitives. This means that the user can customize the view depending on what information is needed.

# 3 Work Packages

The work to be performed was planned as a number of work packages. The work packages are presented together with necessary preparations to solve the specific task and problems that have occurred during implementations of the improvements of the TVTOOL. Alternative solutions and other ideas are also described, [9] and [10].

## 3.1 Preparation

The first weeks were spent learning about SS7 in general, in order to get an overview of the environment in which TVTOOL is used. Our time was spent reading and understanding the implementation of the tool. This increased the comprehension of how the program files collaborated. Discussions with the supervisor at TietoEnator AB resulted in details of the current problems and suggestions of the solutions were verified.

## 3.2 Work Package 1, Terminate Reading Log File

In the earlier version of TVTOOL the user had no possibility to stop the present reading of the log file. The used alternatives were either reading the entire log file or opening a new log file that was read in parallel. When reading parallel files, the last opened file became the valid one in the view of the user. These two solutions were time consuming processes. Reading the entire log file, in some cases the size of the log file is about 100 Mb, can take up to three or four minutes. Reading large log files in parallel is a computer power consuming process. The first assignment was to add the functionality; stop reading a log file on demand of the user.

### 3.2.1 Preparation

In order to understand how to carry out this assignment, it was necessary to learn how Java implements active objects, i.e. parallel processes. There are two ways to implement active objects in Java. The first is using the predefined Java Class java.lang.Thread and the second is using class javax.swing.Timer.

In the package javax.swing there is a class named Timer. An object of this class can execute as its own thread in the background. A Timer object can generate events of the type ActionEvent. Parameters that are sent to the objects constructor will decide how often this

event is generated. A reference to a listener, an object that implements the interface ActionListener, is also sent to the constructor. An example of a call to the constructor;

Timer t = new Timer(200,l).

This definition creates an object that generates an event every 200:th millisecond. The second argument “l” is a reference to the listener. A start() method has to be called, it makes it possible for the Timer object to generate events.
The class Thread is found in the java.lang package. By defining a Thread object, a new activity can be created.
An example of declaration of a Thread object:

Thread myThread = new Thread();

To start an execution of a Thread object the method start has to be called. In the class Thread there is a method called run. This method will be called automatically when start is called. The run method should define the real activity. The thread will exist and be active until it has executed its task to the end.

In the implementations of TVTOOL the Thread method was used. The run method was defined but there existed no method to stop or interrupt the Thread. Therefore the entire log file had to be loaded to the end.

### 3.2.2 Solution

To stop the loading of a log file, a regular button was created. When this button is pressed the reading of the log file shall stop. When the program start reading a log file, a thread is created and the run method is called.

Using threads is a good solution for this task. Because the log files can sometimes be up to 100 Mb in size and loading these files can take a few minutes. Using Threads the program can therefore be able to receive other commands from the user during reading the file.

To be able to stop the reading the correct thread had to be found. The specific thread inherited from the Thread class, but the existing thread method was not used completely. A flag was assigned a value but the Thread was never stopped or interrupted. The consequence of this solution with a flag containing the state of the Thread was parallel running threads.

The solution was to interrupt the started Thread when the user pressed the stop button or started reading a new log file. For sequence diagram see Appendix D.

The button is only available when TVTOOL is reading a log file. When the reading is finished, the stop button will be disabled for the user and it can no longer be pressed, see figure 3-1.

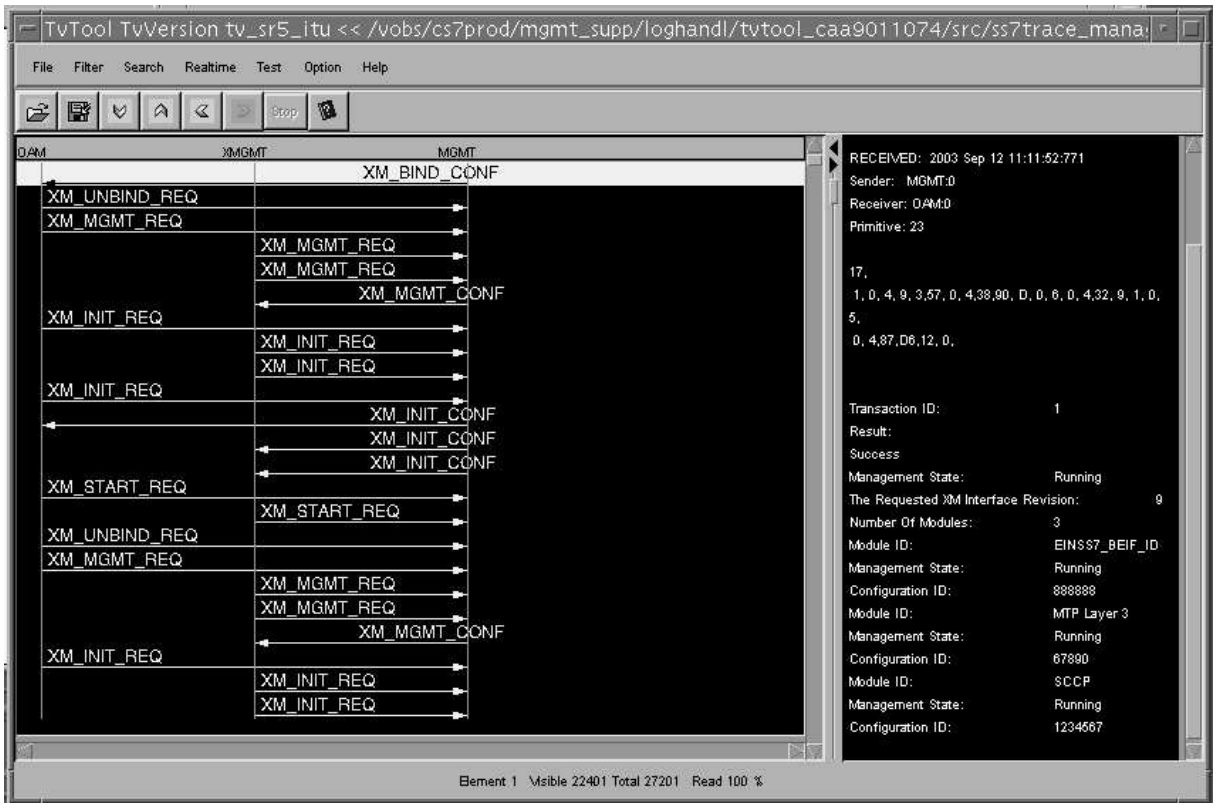



*Figure 3-1 TVTOOL adding Stop Reading Functionality*

### 3.2.3 Problems

Terminate reading log file was the first problem to solve. No experience of reading other developers implementations resulted in us spending several hours reading the program code.

TVTOOL consists of over thirty program files. This was a larger amount of files compared to the university laborations in programming. The program files contains between 500 to 1300 rows. Due to the inexperience of handling large amount of program files, it was complicated to find in which class and in which function the modifications was supposed to be done.

Problems occurred how to understand using threads in general and how the previous developers had implemented it. Reading and discussing with more experienced people created conditions to manage the Thread implementation.

When the reading of the log file was interrupted, the user should have the possibility to look at the information the program has loaded so far. The first implementation deleted the information about the primitives, when the stop read button was pressed.

Deleting this important information destroyed the functionality of TVTOOL. Many hours were spent to find the failure. Finally a former developer of TVTOOL informed us about the efficiency problem of Java application generally. To make programs less time consuming it is recommended to close files when they are not used. This was the cause of the problem, the log file was closed and the program tried to read from a closed file. When the log file was closed later in the program the assignment was finished.

## 3.3 Work Package 2, Save Filter Options

TVTOOL offers the service to filter the information created by the LogDeamon. The users of TVTOOL requested an opportunity to save their customized filter options between sessions. The assignment was to save these options in a properties-file, [2].

### 3.3.1 Preparation

Java offers several ways to save information to a file. There are two ways to access a file, sequential or direct. Direct access files are handled by specific classes e.g. Hashtables. Sequential files can be described as streams. A stream can be compared to a communication channel between a source and a destination. If the data flows into a program it is called an input stream and if the data flows out, it is an output stream. The destination of flow can be e.g. a file or a remote computer. The fundamental idea with streams is that the data should always be handled in the same way in a program, independent of the source or the destination.

Java has a built-in class Properties. This class is an extension of the Hashtable class. The Properties Class can be used for e.g. creating tables or Configurations files. The keys, the identity, and the corresponding values must be of the type String. The main advantage of the Properties class is that it is designed to quickly and easily retrieve and store the keys and values from a text file, allowing quick configuration changes to the code without requiring a re-compile. Other advantages with the Properties Class are that properties can be loaded and stored to a file. Any form of output stream can be used to save the properties and any form of input stream can be used to load the properties.

The structure of a properties file is simple. Each line is either a comment, identified with a # character, or a key-value pair. The key-value pair row begins with a key followed by an equals sign or a colon and then the value of the key.

There are methods defined to get and set values, getProperty() and setProperty(). There are two versions of the getProperty() method: one that provides a default value, if the given key does not exist and another that will return null if the key does not exist.
The setProperty() method accepts two parameters. The first parameter is a String representing the key and the second parameter is a String representing the value. If the key already exists within the properties class, its value will be changed to the new value.

### 3.3.2 Solution

The first part of this assignment was to change the existing header. Before changes, the header contained the version of TVTOOL and the path of the log file. The header was modified to also include the TV-version.

In the previous version, the program used an ordinary text file, TVTOOL.cnf, that worked like a configuration file. To make use of the built-in Java-properties the TVTOOL.cnf was changed to TVTOOL.properties. Example of TVTOOL.properties see figure 3-2. The TVTOOL.properties file is automatically saved in the users home directory.

```
#Testing properties
#Mon Apr 19 16:29:34 MEST 2004
Log.Debug=ON
Log.TESTREADER=ON
Log.Width=150
Log.Path=/vobs/cs7prod/mgmt_supp/loghandl/tvtool_caa9011074/src
Log.Textwidth=15
Log.Error=ON
Log.Trace=ON
Log.MTPL3=ON
Log.MTPL2=ON
Log.MGMT=ON
Log.Instance.2=ON
Log.Instance.1=ON
Log.XTrace=ON
Log.XMGMT=ON
Log.Instance.0=ON
Log.port=6999
Tv.Path=/proj/portss7/bin/tv_sr5_itu
Log.SCCP=ON
Log.Message=ON
Log.OAM=ON
Log.MGMT_FE=OFF
Log.Timer=ON
```

*Figure 3-2 Example of properties file*

To solve this assignment, variables were created. The variables store the value of the options during the execution of the program. To handle the choices of the user options, set and get functions were created for each variable. The set functions were also used when the program was started and the options were fetched from the properties file. The GUI retrieved the value of the variables from the get functions and displayed these values in the graphical view.

An easily accessible button was placed in the toolbar, this making the saving filter options available to the user. See figure 3-3. To get an overview see sequence diagram in Appendix E.

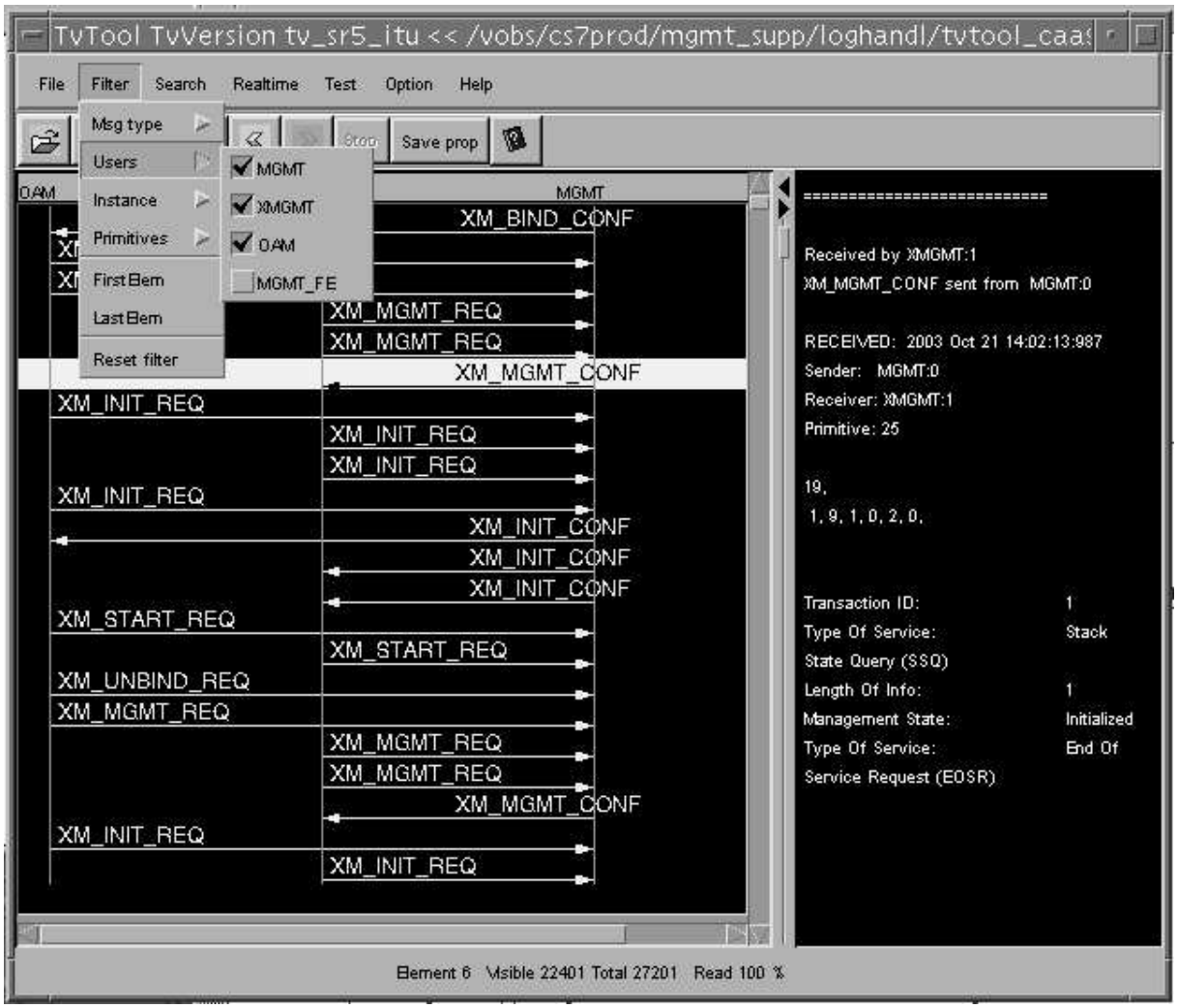



*Figure 3-3 TVTOOL Save Properties Functionality added*

### 3.3.3 Problems

A first attempt was made to display the result of fetching the content of the properties file using examples from the literature. The implementations were never displayed in the GUI. The cause was, a new defined class. The new class had its own definition of displaying values in the GUI, called later in the program. When these own defined functions were used, the program managed to display the values from the properties file in the GUI.

## 3.4 Other Work Packages on TVTOOL

Additional work packages documented by TietoEnator AB are presented in this section. The ambition was to implement these work packages but there was no time left. The work packages were:

- Improve the user interface input for e.g. filter options.
- Additional function of search (possibility to interrupt current search).
- Design hierarchical sequence diagram.
- Create a help function and a user manual.

## 3.5 Experiences

This section summarizes the experiences during the Bachelor's Project.

TVTOOL traces the traffic within the SS7-stack and it was necessary to get an overview of SS7. SS7 is a complex product and in some cases difficult to comprehend.

TVTOOL is implemented in the programming language Java. Unfortunately Java courses were not included in the ordinary Bachelor's of Computer Science at Karlstad University. This was a new language to learn, which increased the complexity. Java literature was read to understand the language. Experience in this language has been a positive effect.

Many designers have developed TVTOOL. They have extended the functionality for their own purposes. The consequence of many different developers has had a negative effect of the program structure. This makes the implementation of the program sometimes difficult to understand. Because TVTOOL was not developed to be a product, there is no existing documentation and the code only contains a small number of comments. Reading and understanding other designer's implementation style shows how important it is to include comments and documentation. It makes it easier to maintain the program in future.

This Project has given wisdom about how important Analysis and Design phases are when software is developed. The purpose of the Analysis phase is to achieve a more precise understanding of the requirements and to achieve a description of the requirements that is easy to maintain. This structures the whole system including its architecture.

In the Design phase the shape of the system is developed, including its architecture. All requirements are checked, i.e. programming language, component reuse, operations system, user interface technologies and so on.

# 4 Improvement of TVTOOL

The advisor at TietoEnator AB requested documentation of recommendations for further improvements during the implementation of the assignments. This chapter contains our suggestions for improvements of TVTOOL. In other words it is not from an ordinary user perspective but from a maintainers view.

- Create documentation containing an overview of the TVTOOL system including pictures showing the collaboration between the program files. This gives a system overview and facilitates for future developers to e.g. locate the function to modify.
- Add comments to each file header containing the main purpose of the program file.
- Add comments to complex functions and give the variables describing names. This will increase readability and program comprehension.
- Reorganize the implementation. Similar parts should be placed in sections to improve the readability and maintenance.
- Optimize TVTOOL to handle large trace files.

# References


[1] http://www.c7.com/ss7/whitepapers/brooktrout_into_to_ss7.pdf

[2] http://www.codenotes.com/articles/articleAction.aspx?articleID=373

[3] http://www.cisco.com/univercd/cc/td/doc/product/tel_pswt/vco_prod/ss7_fund/ss7fun04.htm#80740

[4] http://www.bsnl.co.in/service/inteligent_network.htm

[5] Generic Application System HD SEP R1, 1/15517-1/ANF 101 04 Uen FUNCTIONAL SPECIFICATION Rev A

[6] SS7 in Brief, Basic Concepts, Ericsson Infocom AB, 1997

[7] Portable SS7, Ericsson Infotech AB

[8] James F. Kurose and Keith W. Ross, Computer Networking A TopApproach Featuring the Internet, Addison Wesley, Second Edition, 2002

[9] Jan Skansholm, Java direkt med Swing, Studentlitteratur, 2003

[10] Deitel and Deitel, How to program Java, Prentice Hall, Fourth Edition, 2002

[11] Ericsson Telecom AB och Telia AB, Att förstå telekommunikation, Ericsson Telecom AB, Telia AB och Studentlitteratur, 1996

# A Abbreviations

| | |
|---|---|
| ANSI | American National Standard Institute |
| API | Application Programming Interface |
| ATM | Asynchronous Transfer Mode |
| BE | Back End |
| BIB | Backward Indicator Bit |
| BSN | Backward Sequence Number |
| CAS | Channel Associated Signaling |
| CCS | Common Channel Signaling |
| CP | Common Parts |
| CPCS | Common Part Convergence Sub-layer |
| FE | Front End |
| FIB | Forward Indicator Bit |
| FISU | Fill In Signaling Unit |
| FSN | Forward Sequence Number |
| GSM | Global System for Mobile communication |
| GUI | Graphical User Interface |
| HA | High Availability |
| HD | Horizontally Distributed |
| HW | Hardware |
| IF | Interface |
| INAP | Intelligent Network Application Protocol |
| IN | Intelligent Network [4] |
| IS41 | Electronic Industries Association Interim Standard 41 |
| ISUP | Integrated Services Digital Network User Part |
| ITU-T | International Telecommunication Union |
| LM | Layer Management |
| LSSU | Link Status Signal Unit |
| MAP | Mobile Application Part |
| MGMT | Management |
| MM | Management Module |

| | |
|---|---|
| MSU | Message Signal Unit |
| MTP | Message Transfer Part |
| NMP | Network Management Process |
| OAM | Operation, Administration and Maintenance |
| OS | Operating System |
| OSI | Open System Interconnection |
| R&D | Research and Development |
| SAAL | Signaling ATM Adaptation Layer |
| SBU | Sub Business Unit |
| SCCP | Signaling Connection Control Part |
| SCTP | Stream Control Transmission Protocol |
| SIF | Service Information Fields |
| SIO | Service Indicator Octet |
| SP | Signaling Point |
| SS7 | Signaling System No. 7 |
| SSCF | Service Specific Coordination Function |
| SSCOP | Service Specific Connection Oriented Protocol |
| TCAP | Transaction Capabilities Application Part |
| TTC | The Telecommunications Technology Committee (Standardization body in Japan) |
| TUP | Telephone User Part |
| TV | Trace Viewer |
| TVTOOL | Trace Viewer Tool |
| UDP | User Datagram Protocol |

# B Signaling Units

In this Appendix detailed information about the three types of messages created by Layer 2, MSU, LSSU and FISU are presented. Information about the different fields [3] of the message types is presented after the figures.

- **MSU** the package that contains the actual SS7 message from a higher level in the SS7 stack.

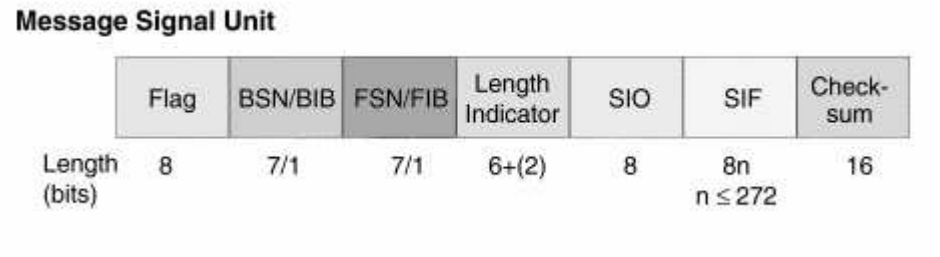



*Figure 0-1 MSU*

**Flag** - Delimiter in a signal unit, which marks the end of one signal unit and the beginning of another. All signal units begin with a distinct 8-bit pattern, 0111 1110.
**Checksum** - An 8-bit sum is calculated by the transmitting SP from the transmitted message. The checksum is inserted in the message. The receiving SP recalculates the sum. If it is corrupted, a retransmission is requested.
**Length Indicator -** The number of octets between itself and the checksum. Checks the integrity of the signal unit and discriminates between different types of signal units. The default values are: FISU=0, LSSU=1 or 2, MSU<2
**BSN/BIB FSN/FIB -** Octets that hold the Backward Sequence Number, BSN, and Backward Indicator Bit, BIB, the Forward Sequence Number, FSN and the Forward Indicator Bit, FIB. These fields are used to confirm receipt of signaling units and to ensure that they are received in the same order that they were transmitted. They also provide the flow control.

The functionality of the MSU lies in the contents of the Service Indicator Octet, SIO, and the Service Information Fields, SIF.

The **SIO** contains three types of information.

- The first four bits define the type of information the service information field consists of, e.g. Signaling Network Management and Signaling Network Testing and Maintenance.
- The two following bits indicates if the message is used in a national or international network.
- The last two bits are the priority of the message.

**SIF** consists of two sub fields.

- The first field is source and destination addresses.
- The last field is the user data that should be sent.

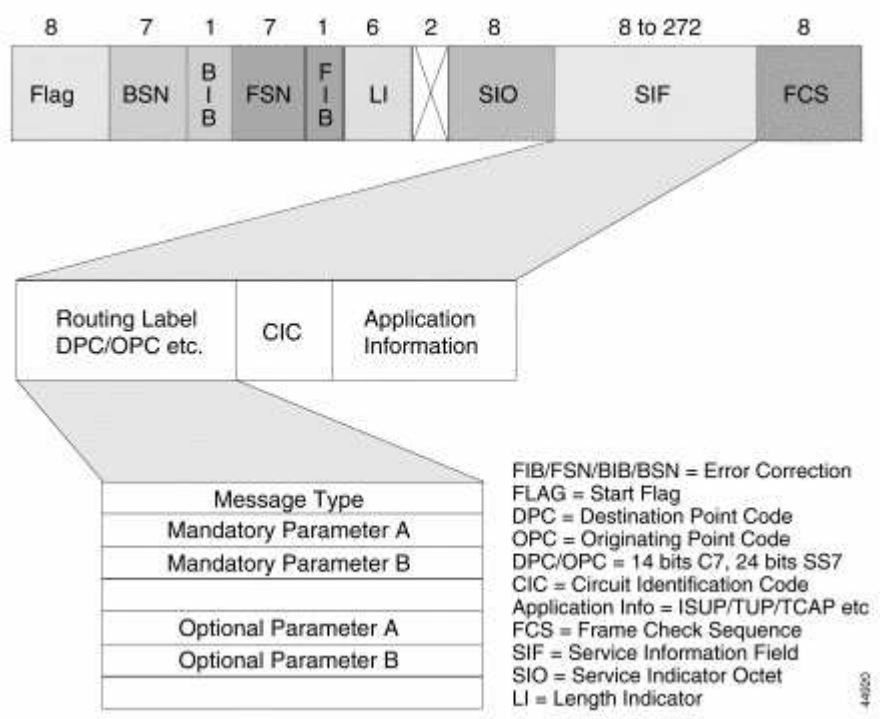



*Figure 0-2 Detailed MSU*

- **LSSU** carries the information about the links.

**Link Status Signal Unit**

| | Flag | BSN/BIB | FSN/FIB | Length Indicator | Status Field | Check-sum |
|---|---|---|---|---|---|---|
| Length (bits) | 1 | 7/1 | 7/1 | 6+(2) | 8 or 16 | 16 |

*Figure 0-3 LSSU*

**Status Field -** Contains the information about the link e.g. the status of the processors at either end of the link.

- **FISU** the packages are transmitted across the network when no other packages are sent. The purpose of sending FISU continuously is to detect link failure quickly.

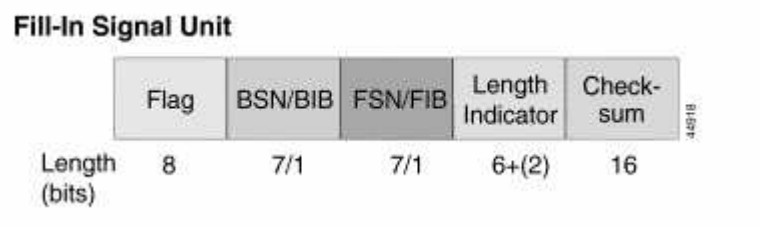



*Figure 0-4 FISU*

# C Detailed information about Backend and Frontend

This Appendix presents detailed information about BE and FE and detailed figures are shown to get an overview of which modules BE and FE are created by. Further information about HD can be found in [5].

The Backend function, BE, handles SS7 User traffic and registration/deregistration of SS7 Users and provides SS7 Users with information about status of local and remote subsystems.

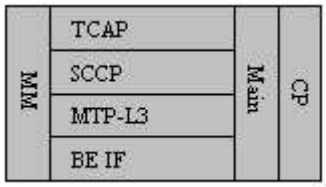



*Figure 0-5 Backend modules*

Figure 0-5 shows the modules of the Backend module. TCAP, SCCP, MTP-L3, CP and MM are described in sections 2.2.4.

- Main consists of a generic and a platform specific part. The generic part is an interface between the modules. The platform specific part provides the generic part with necessary start-up information e.g. process type and instance number. The generic part is Operating System, OS independent.
- Backend Interface, BE IF, controls the communication with the FE processes in the system. Each BE in the system has a connection with every FE in the system.

A FE process control one or several signaling boards with either narrowband, see figure 0-6, or broadband, see figure 0-7, signaling links and serve as a distributor of incoming traffic. Depending on the FE type the FE process consists of different modules.

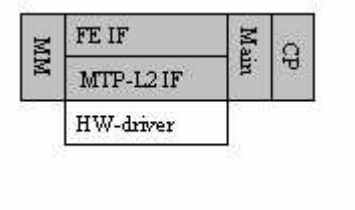



*Figure 0-6 Narrowband FE (MTP)*

- Frontend Interface, FE IF, handles the communication with BE IF in the Network Management Process, **NMP**. NMP handles the SP's networks management it ensures that the SP acts as one common entity in the SS7 network, even if there are several instances of the BE's executing in parallel.
- Hardware-driver, **HW-driver**, The HW specific MTP-L2-IF accesses the HW signaling board driver and thereby communicate with the MTP-L2 located on the HW signaling board.

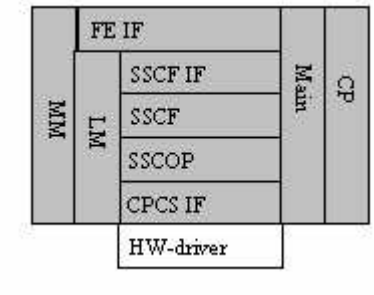
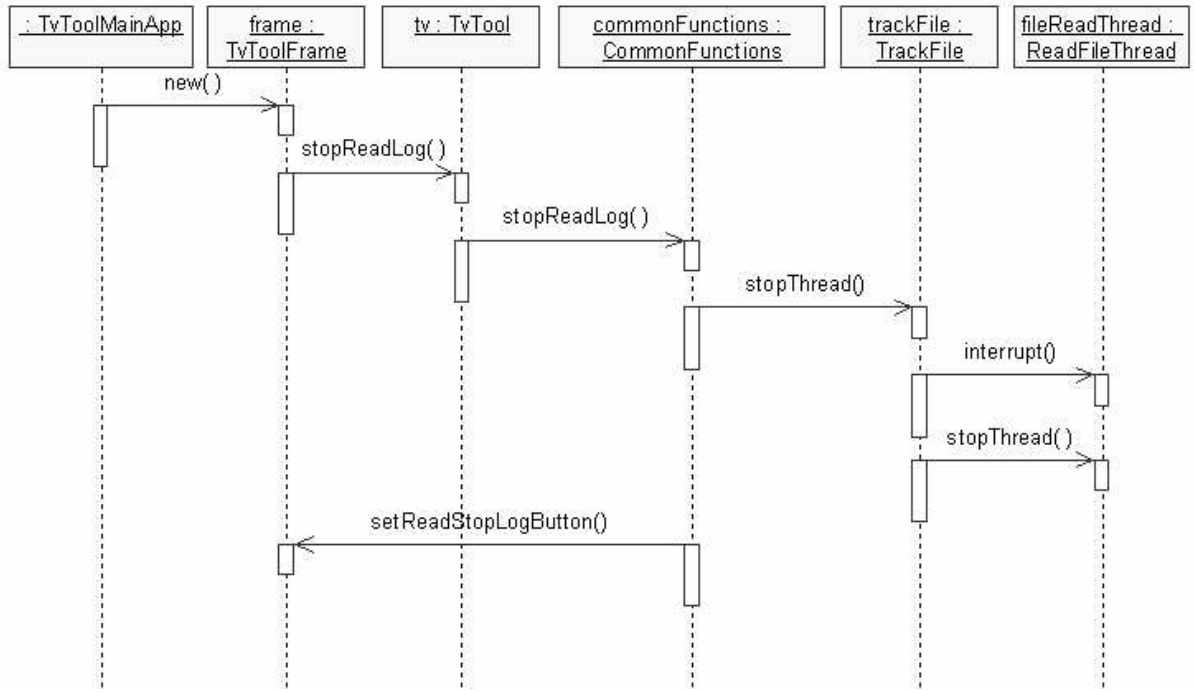



*Figure 0-7 Broadband FE (SAAL)*

- Service Specific Coordination Function Interface, **SSCF IF**, is a mapping layer that converts the messages from and to FE IF and MTP-L3, making them understandable for Service Specific Coordination Function, SSCF.
- **SSCF**, provides coordination between the MTPL-3 Users and the data transfer services provided by Service Specific Connection Oriented Protocol, SSCOP.
- **SSCOP** is a peer-to-peer connection oriented protocol that provides a generic reliable data transfer service, including error, recovery for the SSCF.
- Common Part Convergence Sub-layer Interface, **CPCS IF**, is a HW specific adaptation layer that must be adapted to access the HW specific signaling board driver and thereby communicate with the implementation on the HW board
- Layer Management, **LM**, provides management coordination, link quality control and measurement of statistics for the different SAAL operations.

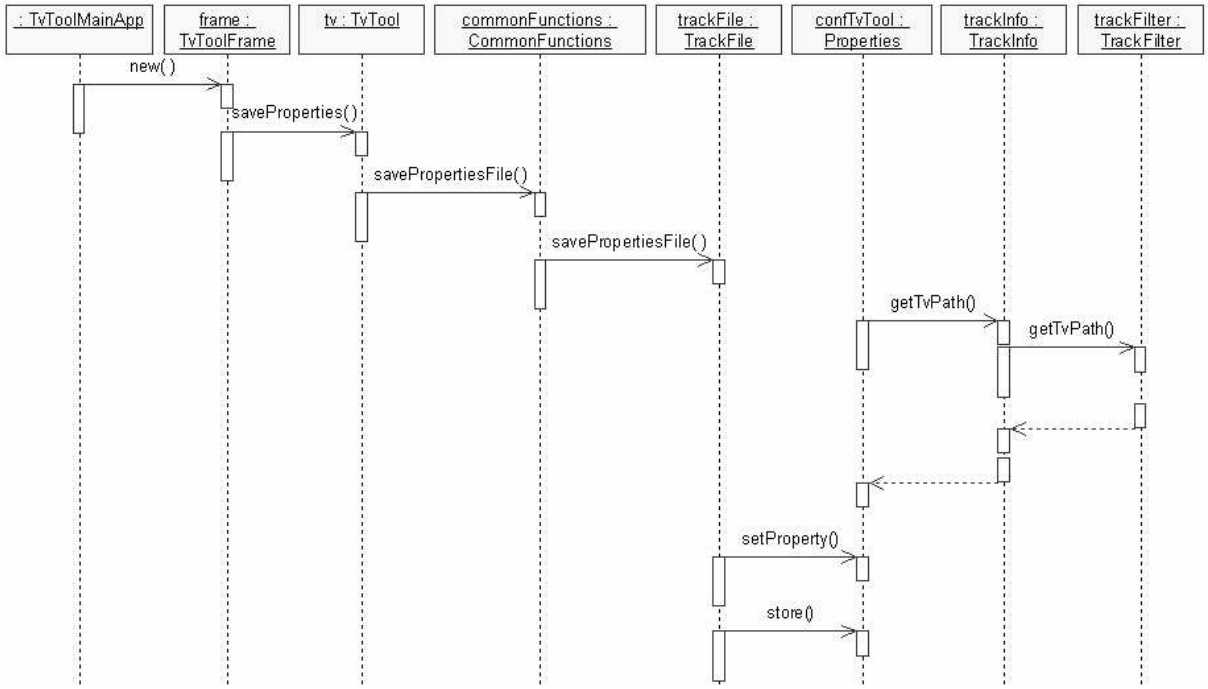
# D Sequence Diagram StopReadLog

To interrupt the started Thread when the user has pressed the stop button or has started reading a new log file.



# E Sequence Diagram SaveProperties

The user has pressed the Save Filter Options button.



# F Bachelor's Project Specification (in Swedish)

## Exjobbets titel

Improve a SS7 Protocol Tool Developed in Java

## Bakgrund

Ett telekommunikationsnät består logiskt sett av två separata nät. Dels består det av ett bärarnät för tal och data, och dels ett signaleringsnät med huvuduppgiften att förmedla den styr-information som krävs för uppkoppling, övervakning och nedkoppling av tele-förbindelser (mobilnät är det dessutom signaleringsnätet som möjliggör mobiliteten) som dessutom utgör grundvalen för majoriteten av alla teletjänster som t ex återupp-ringning och vidarekoppling.

Inom divisionen Telsys hos TietoEnator AB i Karlstad utvecklas signaleringslösningar till bl a Ericsson. De har t ex utvecklat en protokollstack för så kallade SS7-baserade signaleringsnät (SS7 är en förkortning för ”Signaling System Nr. 7”): ”Portable SS7”.

Under test och utveckling av ”Portable SS7” utnyttjas ett antal protokollanalysverktyg. Ett av dessa verktyg går under benämningen ”tvtool” . Detta är ett verktyg, utvecklat i Java, som är framtaget internt inom Telsys. Tvtool parsar den loggfil som skapas av ”Portable SS7” och åskådliggör i ett sekvensdiagram, flödet av meddelanden mellan olika komponenter i ”Portable SS7” under en session. Vidare innehåller tvtool funktioner för filtrering och sökning.

## Mål

På senare tid har krav framförts på utökad och förbättrad funktionalitet hos ”tvtool”. Dessa krav framgår under rubriken Resultat.

## Resultat

Vår uppgift är att implementera delar av dessa krav. Mer specifikt ska följande krav implementeras:

1. Utökad sökfunktion.
2. Hierarkisk uppritning av sekvensdiagram.
3. Förbättrat användargränssnitt för inmatning av t ex filteroptioner.
4. Möjlighet att avbryta pågående sökningar.
5. Möjlighet att avbryta pågående inläsning av loggfil.
6. Möjlighet att spara filterinställningar o dyl mellan sessioner.
7. Utöka hjälpfunktionen.
8. Ta fram en användarmanual.

## Arbetsuppgifter

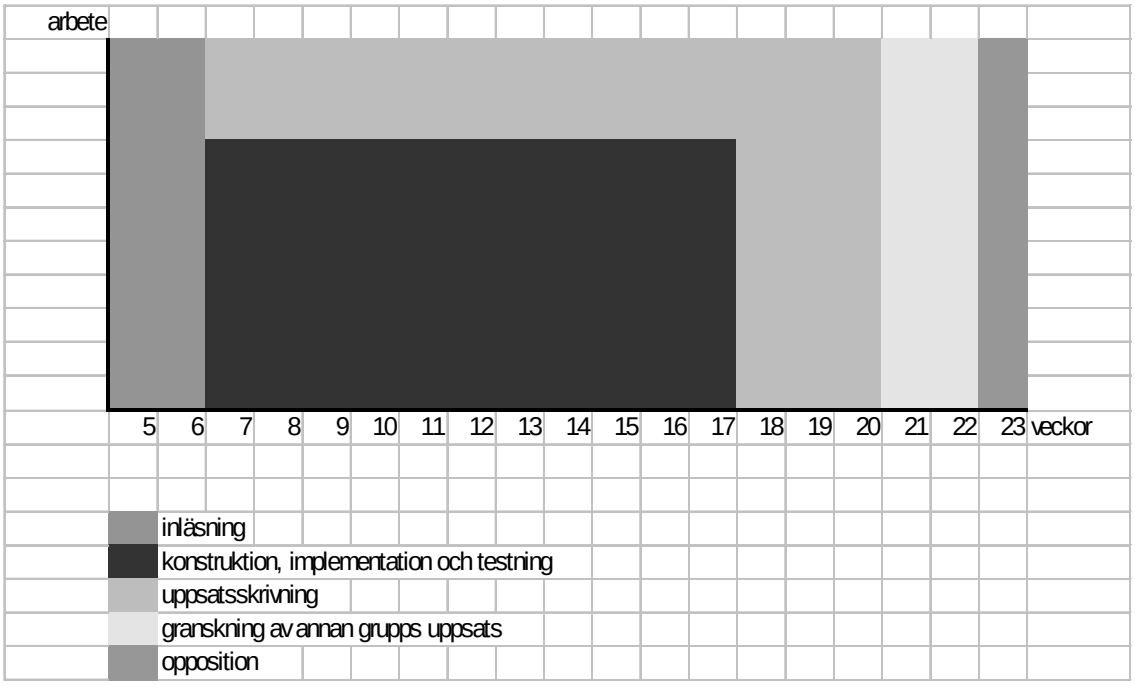
**Inläsning**
Vilket innebär att vi skall få en förståelse för vertygen som vi skall använda samt grunder i Java. Få en grundläggande förståelse för SS7 samt det befintliga verktyget Tvtool.

**Konstruktion, implementation och testning**
Exjobbets utforming är specifiserat i olika punkter (se resultat ovan). Dessa olika punkter kommer att konstrueras, implementeras och testas var för sig. Dvs varje punkt kommer att vara en egen iteration.

**Dokumentation**
Veckorapporter kommer att skrivas enligt mall varje vecka och skickas med e-post till handledare vid universitet.

Uppsatsen kommer att skrivas kontinuerligt under arbetets gång.

## Tidplan




Beräknad total tid för exjobbet är 360 timmar varav 120 timmar är avsatt till uppsats-skrivning, 20 timmar till granskning inför opposition då återstår 220 timmar för själva arbetet.

**Omfattning**

Inläsning

Möjlighet att avbryta pågående inläsning av loggfil 32-40 timmar

Förbättrat användargränssnitt för inmatning av t ex filteroptioner 32-40 timmar

Möjlighet att spara filterinställningar o dyl mellan sessioner 40-52 timmar

Utökad sökfunktion (Möjlighet att avbryta pågående sökningar) 60-72 timmar

Hierarkisk uppritning av sekvensdiagram 40-60 timmar

Utöka hjälpfunktionen och ta fram en användarmanual 32-40 timmar

Totalt 236-304 timmar, ovanstående punkter har placerats i prioritetsordning och utförs i mån av tid enligt ovanstående tidplan.

## Resurser

| | |
|---|---|
| Lokaler: | Vi kommer att arbeta i TE:s lokaler |
| Utrustning: | Dator som TE tillhandahåller, samt accessmöjligheter till nödvändig programvara. |
| Programvara/Miljö: | Java SDK 1.4.2, XEMACS, Rational ClearCase<br>Solaris, Windows, Linux |
| Litteratur: | Java direkt med Swing, Jan Skansholm<br>Ytterligare litteratur tillkommer senare |

## Principer för avrapportering

Avstämningsmöte genomförs med handledare (Per), varannan vecka. Veckorapport enligt mall, skickas via e-post till Per i slutet av varje vecka.

Avstämningsmöte genomförs med uppdragshandledare (Admir), varannan vecka eller efter behov.

## Ansvarsuppdelning

Teknisk hjälp tillhandahålls av uppdragshandledare och frågor runt uppsatsen besvaras av handledare vid universitetet.

## Rimlighetsbedömning

Vår ambition är att utföra alla punkter enligt ovanstående punktlista under rubriken Tidplan: Omfattning.

Följande två punkter är dock något komplexa
Utökad sökfunktion (Möjlighet att avbryta pågående sökningar) och Hierarkisk uppritning av sekvensdiagram. Detta innebär att tidsåtgången kan bli något mer än planerat.