Final Thesis

ToIP functionality in Asterisk:
Making ToIP communication more available and interoperable by integrating it in the IP-PBX Asterisk

by

Sara Hörlin

LITH-IDA-EX--07/045--SE

2007-08-17
Final Thesis

**ToIP functionality in Asterisk:**
Making ToIP communication more available and interoperable by integrating it in the IP-PBX Asterisk

by

**Sara Hörlin**

LITH-IDA-EX--07/045--SE

2007-08-17

Supervisor: Gunnar Hellström
Examiner: Mariam Kamkar
Title
ToIP Functionality in Asterisk: Making ToIP communication more available and interoperable by integrating it in the IP-PBX Asterisk

Author
Sara Höran

Abstract
In the thesis the advantages with Text over IP (ToIP) is explained and it is motivated why it is a good idea to integrate this in Asterisk. It also presents an implementation of a ToIP extension in Asterisk.

ToIP means communicating over a network based on Internet protocols with real-time text. Real-time text means a character is sent to the receiving terminal as soon the sender has typed it or with a small delay.

In the thesis IM and ToIP is compared in a survey. The result point at IM is not better than ToIP even though it is much more commonly used. VoIP can not replace ToIP either because there are occasions when ToIP is better for instance if the person using it is deaf or if a person want to make a private conversation in a noisy room.

Asterisk is an IP-PBX. PBX stands for Private Branch Exchange which means a private telephone system which is part of a larger network system that exchange information.

An IP-PBX is a PBX based on the Internet. Asterisk and many other IP-PBX can also exchange calls between the PSTN and the Internet. By including ToIP in Asterisk it will be possible to exchange ToIP calls.

The implementation described is not only including ToIP in Asterisk but also a translation function between the text format called T.140 and another text format called T.140 with redundancy.

The idea is to extend the translation function in the future to more text formats.

Keywords
Asterisk, ToIP, PBX, T.140, RTP, SIP
Abstract

In the thesis the advantages with Text over IP (ToIP) is explained and it is motivated why it is a good idea to integrate this in Asterisk. It also presents an implementation of a ToIP extension in Asterisk.

ToIP means communicating over a network based on Internet protocols with real-time text. Real-time text means a character is sent to the receiving terminal as soon the sender has typed it or with a small delay. There is a presentation level standard for real-time text conversation called T.140 (ITU-T 1998). There is also an Internet standard for transmission of real-time text called RFC 4103 which uses T.140 (Hellström 2005).

There are other ways of communicating by the Internet. Most common is with Voice over IP (VoIP) which is the same as using the telephone but on the Internet. Another way of communicating is with Instant Messaging (IM) which is a message oriented text communicating method. In the thesis IM and ToIP is compared in a survey. The result point at IM is not better than ToIP even though it is much more commonly used. VoIP can not replace ToIP either because there are occasions when ToIP is better for instance if the person using it is deaf or if a person want to make a private conversation in a noisy room.

Asterisk is an IP-PBX. PBX stands for Private Branch Exchange which means a private telephone system which is part of a larger network system that exchange information. Traditionally the larger network system is the Public Switched Telephone Network (PSTN). The concept of the PBX was that companies could manage their own private telephone system and by that save money by not paying for internal calls. An IP-PBX is a PBX based on the Internet. Asterisk and many other IP-PBX can also exchange calls between the PSTN and the Internet. By including ToIP in Asterisk it will be possible to exchange ToIP calls.

The implementation described is not only including ToIP in Asterisk but also a translation function between the text format called t140 and another text format called t140 with redundancy. When using the new implementation it will be possible to make calls through Asterisk where one terminal using plain t140 and the other t140 with redundancy. The idea is to extend the translation function in the future to more text formats. In this way ToIP will be more available to people because you do not need to use the same application to call with as the person you make the call to and hopefully not even the same type of network.
Acknowledgement

This work was partially funded by the National Institute on Disability and Rehabilitation Research, US Dept under Grant H133E040013 as part of a co-operation between the Telecommunication Access Rehabilitation Engineering Research Center of the University of Wisconsin – Trace Center joint with Gallaudet University, and Omnitor. People I would like to recognize for participating in the work is Benjamin Larsson and Gunnar Hellström at Omnitor. Benjamin has oriented me through Asterisk and Gunnar has come up with designs ideas and ideas for subjects to the report. I would also like to thank my examiner Mariam Kamkar for examining me and my opponent Anna Vapen for reviewing my work. Finally I would like to thank John Martin at AuPix for hints about Asterisk and Patrik Persson who also has reviewed the report.
# Table of Contents

1 Introduction ........................................................................................................................................... 1  
1.1 About the thesis ................................................................................................................................. 1  
1.2 Purpose ............................................................................................................................................... 1  
1.3 Method ............................................................................................................................................... 1  
1.4 Problem ............................................................................................................................................. 2  
1.5 Goal .................................................................................................................................................. 2  
1.6 Delimitations .................................................................................................................................... 2  

2 Documentation overview .......................................................................................................................... 3

3 Different Types of Networks .................................................................................................................. 4  
3.1 Circuit Switched Network ..................................................................................................................... 4  
3.2 Packet Switched Networks .................................................................................................................. 4  
  3.2.1 Connection Oriented Packet Switched Network ............................................................................. 4  
  3.2.2 Connectionless Packet Switched Network .................................................................................... 4  

4 Communication over Internet Protocol .................................................................................................... 6  
4.1 Voice over Internet Protocol (VoIP) .................................................................................................... 6  
4.2 Text Based Communication over IP .................................................................................................. 6  
  4.2.1 Instant Messaging ......................................................................................................................... 6  
  4.2.2 Text over Internet Protocol (ToIP) ............................................................................................... 6  
  4.2.3 Difference between ToIP and VoIP .............................................................................................. 7  

5 Comparison between Real-Time Text Conversation and Message Oriented Text Communication ........................................................................................................................................................................ 8  
5.1 Background and goal ............................................................................................................................ 8  
5.2 Survey implementation ......................................................................................................................... 8  
5.3 Trial users ........................................................................................................................................... 9  
5.4 Questions asked after the trial ............................................................................................................. 9  
5.5 Result ............................................................................................................................................... 9  
5.6 Conclusions ..................................................................................................................................... 10  

6 Protocol used for VoIP and ToIP ............................................................................................................. 12  
6.1 Session Initiated Protocol (SIP) ......................................................................................................... 12  
  6.1.1 Components in a SIP call ............................................................................................................... 12  
  6.1.2 Call set up with SIP...................................................................................................................... 12  
6.2 Session Description Protocol (SDP) .................................................................................................. 13  
6.3 Real-time Transport Protocol (RTP) ................................................................................................ 14  
6.4 RTP Payload for Text Conversation ................................................................................................ 14  
  6.4.1 T140 with redundancy ................................................................................................................ 15  
  6.4.2 How t140 with redundancy is packed ......................................................................................... 15  
  6.4.3 T140 with redundancy compared to message oriented methods ........................................... 16  
  6.4.4 UTF-8 ......................................................................................................................................... 17  
  6.4.5 Applications that supports RFC 4103 ......................................................................................... 17  

7 Private Branch Exchange (PBX) ............................................................................................................. 19  
7.1 Functionality .................................................................................................................................... 19  
7.2 IP-PBX .............................................................................................................................................. 19  

8 Asterisk ................................................................................................................................................. 20  
8.1 Functionality .................................................................................................................................... 20  
8.2 Architecture ................................................................................................................................... 20  
  8.2.1 Asterisk's API ............................................................................................................................... 21  
  8.2.2 The Dialplan ............................................................................................................................... 21  
  8.2.3 Core .......................................................................................................................................... 22  

9 Interoperability ..................................................................................................................................... 25  
9.1 Standards ....................................................................................................................................... 25
List of Figures

Figure 1 The user interface of the real-time conversation application Allan eC. ..........................7
Figure 2 Call setup using SIP.............................................................................................................13
Figure 3 Illustration of how T140 data is packet in a RTP-packet (Hellström 2005)......................15
Figure 4 Illustration a RTP packet with three blocks of T140 data (Hellström 2005). ..................16
Figure 5 Asterisk core and its four API (Spencer 2003)..................................................................21

List of Tables

Table 1 Distribution in preference. .....................................................................................................9
Table 2 The distribution of answers about how easy it was to solve the task depending on the text communication method used. .................................................................10
Table 3 How UTF-8 is coded. (wikipedia 2007:d). ..........................................................................17
1 Introduction

Today there are good possibilities for a hearing person to talk with good quality with other persons by telephone using the Internet, so-called voice over Internet protocol (VoIP). For a deaf or hard of hearing person an alternative is to communicate by real-time text over the Internet. This is also called Text over Internet Protocol (ToIP). The definition of real-time is that a character is popping up on the receiver’s terminal right after the sender has typed it or with a very small delay. Gunnar Hellström the MD of the businesses company Omnitor has designed a standard for real-time text transmission which is described in RFC 4103 (Hellström 2005). Even though there is an Internet Standard for real-time text transmission not all manufactures developing communication systems for real-time text follow the standard. Products that are not following the same standard will not be interoperable. This could be seen as a problem for the end-user and also for service providers because then the user of a communication application is dependent on the persons they want to communicate with using the same type of application.

1.1 About the thesis

This is a final thesis for the education Master of Computer Science and Engineering at Linköpings Universitet. The work is a part of an ongoing collaborative effort to promote mainstreaming and functional enhancements toward telecommunications access for all. The work was partially funded by the National Institute on Disability and Rehabilitation Research, US Dept under Grant H133E040013 as part of a co-operation between the Telecommunication Access Rehabilitation Engineering Research Center of the University of Wisconsin – Trace Center joint with Gallaudet University, and Omnitor. Omnitor is a Swedish company which designs solutions for total conversation adjusted for people with special needs. Total conversation includes voice, video and real-time text.

1.2 Purpose

The purpose of this thesis is partly to show there is a necessity for real-time text communication. More specific, to show instant messaging which is a more commonly used text communicate method can not replace real-time text. Also the thesis wants to show the importance of interoperability between systems. The most important thing with the thesis is to simplify the use of real-time text communication over the Internet protocol by designing a solution for making it possible to make text based calls through Asterisk which is an IP-PBX, and there by be able to use the features Asterisk offers and also preparing for making different real-time text communication systems compatible with each other.

1.3 Method

In order to investigate if there is a market for real-time text communication, a small investigation where instant messaging and real-time text communication is compared was performed. The main purpose was not to find out exact how the necessity differs between these methods but only if there is a preference for real-time text in any situation.

The description of Asterisk, what it is, how it works and its feature is built on different literature made by people working with or interested of Asterisk. Though the documentation
of Asterisk architecture is quite general, questions have been asked to people involved in Asterisk and also source code have been studied.

The implementation which is a big part of the thesis work follows the description in RFC 4103. Other design decisions are based on preferences from Omnitor, the architecture of Asterisk and sometimes just the simplest alternative has been chosen.

Other facts in the report are based on different documents written by people who developed and studied Asterisk, different RFC’s and other documents and books about different techniques important for the subject. Some basic definitions come from the Internet forum Wikipedia.

1.4 Problem

Asterisk as it is today can receive SIP-calls from a SIP-terminal, where the media is real-time text with the format t140, and without doing anything with the packet send it to another SIP-terminal. By adding functionality for receiving packets with t140-media it will be possible to check the media is correct before sending it to the receiver and also it will be possible to translate the media to other formats. Many telecom products are not compatible even though it has the same purpose. This problem could be overcome by a translator in Asterisk. The implementation will include functionality for translating between plain t140 and t140 with redundancy. A packet with redundancy includes t140-data from earlier packets which can be used if packets are lost. The form of this packet is described in RFC 4103 (Hellström 2005). A user communicating with an application supporting t140 is supposed to be able to choose whether she wants to use redundancy or not. When using t140 with redundancy more data has to be transferred but if the local network is bad the increase in reliability could be very useful.

1.5 Goal

The goal with this final thesis is to design and implement a prototype in Asterisk for handling real-time text media in a call that is set up by SIP. The solution shall support transforming of media with the text-format t140 into the format t140 with redundancy. Also a test application for showing that it is possible to manipulate text data in Asterisk is to be created.

1.6 Delimitations

The result is to be seen as a prototype. Getting things to work has been prioritized over making good code. Also everything in RFC 4103 has not been implemented because of lack of time.
2 Documentation overview

In the chapter Different Types of Networks a description is presented of what kind of networks there exist. This background is necessary to understand the difference between communication over the Internet and common telephone services.

In the chapter Communication over the Internet Protocol more details will be presented of different ways to communicate over the Internet. This chapter also discusses the reasons for using real-time text communication.

In the chapter Comparison between Real-Time Text Conversation and Message Oriented Text Communication, the discussion in previous chapter about why using real-time text communication is followed up by a survey where instant messaging and real-time text communication are compared.

At this point in the thesis the concepts of VoIP and ToIP is presented. In the chapter Protocol Used for VoIP and ToIP the phenomena is discussed in more detail by presenting what protocol is used above the Internet Protocol and the User Datagram Protocol when communicating. The protocols presented are the ones included in the implementation. The chapter also mentions some products following RFC 4103.

In the chapter PBX this technique is described. This information is necessary for the understanding of what Asterisk is.

In the chapter Asterisk, Asterisk is presented. The implementation is built on Asterisk and it is important to understand what Asterisk is and how it is working to understand the implementation.

One purpose with this work is to make real-time text communication more available by making it interoperable with other systems. The chapter Interoperability discusses this issue and what perquisites there are to making systems interoperable and also what benefits there are with compatible systems.

In the chapter Analysis conclusion of the thesis so far is drawn.

In the chapter Implementation the implementation and the design behind is presented. Also a test case study for the implementation is presented.
3 Different Types of Networks

In order to understand the difference of Internet based communication methods and traditional telephone services it is necessary to know about the different networks which the techniques are built on. There are two main types of networks, namely the circuit switched networks and the packet switched networks.

3.1 Circuit Switched Network

In a circuit switched network a connection between source and destination must be set up before any data can be send. When setting up the connection the source use the unique address to the destination and send this to its local exchange which uses the address to set up a connection to the destinations local exchange. When the destination is reached it returns a message that the connection is free to use. When all data has been transferred the source or the destination sends a message so the connection will be cleared. With this type of network there is always a time delay when setting up the connection. The circuit switched network operates in a time dependent way. It is a synchronous communication channel since it provides a constant bit service at a specified rate. The public switched telephone network (PSTN) which is the traditionally telephone network is circuit switched. (Halsall 2006)

3.2 Packet Switched Networks

In a packet switched network data is packed into chunks called packets and transferred over the network from node to node by routers. The routers reads source and destination address and then send the packet to next router or to the destination. The packet switched network is asynchronous since it uses the maximum bandwidth between two nodes. The packet switched network could be divided into two types. These are the connection oriented and the connectionless. (Halsall 2006)

3.2.1 Connection Oriented Packet Switched Network

The connection oriented works in a similar way as the circuit switched and also calls a virtual circuit switched network. Before any data can be transferred a connection is set up between source and destination. The main different to a real circuit switched network is that the first only use a variable portion of the bandwidth of each link while the latter always use the hole bandwidth. By setting up the connection beforehand less work needs to be done in the routers. Each router only needs to send the packet to the next node in the established connection. Also while the connection is up, packet does not need to wait for free links. This type of net is used for services that need high data transfers rate. An example is ATM which is used in bank systems and telecommunications. (Halsall 2006)

3.2.2 Connectionless Packet Switched Network

In a connectionless network a connection set up is not required and the source could start sending data at any time. This type of network is therefore time independent. Data is transferred in packets which include addresses to the destination and source. When a packet arrives to a router it will be checked. The router searching for bit and transmission errors by looking at a control bit in the header of the packet. If the packet has been damage it
discards. In a connectionless network there is never a guarantee packet will reach end
destination. This is the reason why this type of network is called best effort service. This
type of network also suffering for some delays that occur when a packet in a network node
waiting for outgoing link to be free. The Internet is a connectionless packet switched
network. (Halsall 2006)
4 Communication over Internet Protocol

This thesis is about real-time text communication over the Internet but it is also interesting to know about the alternatives. When discussing communication over the Internet the most common media being discussed is voice. The commonly way of using text communication is by instant messaging. These concepts will be explained in this chapter and also real-time text communication will be described.

4.1 Voice over Internet Protocol (VoIP)

Voice over Internet Protocol (VoIP) is a concept for audio communication between users using a packet switched network and the Internet Protocol, normally the Internet. All IP-calls can be seen as local calls since no circuit needs to be set up. While the traditional telephone networks and services are expensive to use, most IP services are cheap or for free. (wikipedia 2007:a) Van Meggelen who is one of the principal contributors to the Asterisk Documentation Project, explain that in this way. The telecom sector traditionally is developed by companies and business men with commercial interest while the Internet and what comes with it was developed in an academic community which value sharing of knowledge over profit (van Meggelen 2005).

4.2 Text Based Communication over IP

Of course there is also ways of communicating with text over the Internet Protocol. For more rapidly conversation, email excluded, exist instant messaging and Text over Internet Protocol (ToIP). This thesis is about ToIP which will be discussed soon but first instant messaging is to be explained.

4.2.1 Instant Messaging

A commonly used method for text communication over the Internet is with a message oriented text method so-called instant messaging (IM). This method is used in chat channels as ICQ, MSN messenger and Skype. When using IM characters are not sent when typed, but when the sender gives a send-commando like pressing the enter button (wikipedia 2007:b). An advantage with this method over ToIP is that it is easier for the user to think through what message to send. An earlier version of ICQ did use real-time text but as it seems it did not turn out so well. One reason could be many people who use IM uses a voice telephone when they need a more interactive conversation. For deaf people this is not an alternative.

4.2.2 Text over Internet Protocol (ToIP)

Text over Internet Protocol (ToIP) stands for real-time text communication. The definition is that a character must be sent to the receiving terminal as soon the sender has typed it or if the sending person is writing fast, characters can be buffered and sent together after a small delay which maximum is allowed to be 500 ms (Hellström 2005). There is an ITU-T standard called T.140 which is a presentation level standard for real-time text conversation (ITU-T 1998). There are then two standards describing how to transmit T.140 in IP-networks. In RFC 4103 (Hellström 2005), the media type t140 is specified to be the only media in a RTP-stream of type text and in RFC 4351 (Hellström 2006) the media type t140c
is one among other media in a RTP-stream of type audio. RFC 4351 was developed for text-telephone traffic between VoIP-gateways. Text-telephones are used over the public switched network where the gateways may have a limited access to ports. Number of ports needed is decreasing when text and audio is in the same stream. If the access to ports is not very limited the first way of transporting the media is better. In fact today RFC 4103 is the only one of the two RFC’s that is used. The reason has to do with how loss of data is being controlled which will be described in the section RTP Payload for Text Conversation. The implementation in Asterisk will follow RFC 4103. Figure 1 shows the user interface of the real-time conversation application Allan eC. Allan eC works with video, text and audio. In the picture only text is in use. The left window shows what the person the user is talking with is typing and the right window show what the user of the application is typing.

![Figure 1](image.jpg)

**Figure 1** The user interface of the real-time conversation application Allan eC.

### 4.2.3 Difference between ToIP and VoIP

When using SIP, there is really no difference in principle between setting up a call with text media and a call with audio media. So what is the different by sending audio and text? VoIP require higher bandwidth than ToIP because audio generates more data. ToIP on the other hand has higher demands on data delivery. A text communication is hard to read even with small losses while for example a voice call preserves pretty good quality with the same loss frequency. Other things that differs is routing algorithms, how to translate to other formats and there is probably other functions as well (Hellström 2005).
5 Comparison between Real-Time Text Conversation and Message Oriented Text Communication

As already mentioned IM is a more commonly used method for text communication than ToIP. Since IM seems more popular than ToIP raises the question if there really is a need for extending support for ToIP or if IM can fulfil the same purpose. There are reasons to believe this is not the case. ToIP is more similar to voice telephone and there are opportunities when the interaction that comes with a phone call is desirable without a voice phone call is an alternative. One clear example is if the persons in the conversation are deaf. Another example is a person working in an office landscape wishing to make a private call. To make this call with text is then another alternative and to use real-time text should seem natural because the interaction in a real-time text call is more similar to a voice call than an IM conversation is. To investigate if there is a market for real-time text communication a survey was made where these methods are compared.

5.1 Background and goal

This is a description of an investigation aiming at clarifying how different methods for two-way text communication over data or telecommunications networks are perceived by the users. The compared methods are real-time text conversation and message oriented text communication. When using real-time text conversation, the user does not need to do any other specific action than typing the characters to transmit them. In message oriented communication, characters are collected until the user gives a send-command like presses enter. Real-time text is currently used mainly in services aiming at deaf and hard-of-hearing users. Message oriented text communication is used in many text chat services.

The investigation is small and limited. That leads to the persons participating in the trials being few and not randomly selected. The result can therefore just be seen as an indication.

5.2 Survey implementation

The survey was performed by two persons at a time communicating with each other through one computer each using the text communication software SipCon1 (SipCon1 2007). This program can be set to communicate in real-time as characters are typed or sentence by sentence. It can also be set to present the dialogue in two columns or in a one column chat style with real-time preview. The real-time preview mode was used in the trial. Each pair of persons in the trial made two trial sessions, one in each mode.

The persons got one drawing on a paper each. The drawings contained parts of a common picture. The users were instructed to try to complement their drawings by adding the contents of the counterpart’s drawing as described in the text conversation session. Each test session was run for five minutes or until the drawings were completed. The goal of this task was to create a need for intensive interaction both ways in the conversation in a way that resembles a telephone call.
5.3 Trial users
The trial users can be divided in two groups.

- Deaf or hard-of-hearing persons
- Hearing persons

This classification is based on the fact that services with real-time text often are aimed at deaf and hard-of-hearing users who therefore might have been exposed to that kind of services more than hearing persons in general.

5.4 Questions asked after the trial
After the trial, the trial persons were asked the following questions:

- Are you deaf or hard-of-hearing?
- Do you have experience of message oriented text communication?
- Do you have experience with real-time text oriented text communication?
- Which of the presented communication methods do you prefer?
- Why do you prefer that method?
- What advantages and disadvantages do you see with each method?
- Are there conditions when you prefer one or the other of these methods?
- How much easier was it to solve the task depending on which text method you used?
  - Much easier with real-time text method.
  - A little bit easier with real-time text method.
  - There was no difference in difficulty between the two methods.
  - A little bit easier with message-oriented text method.
  - Much easier with message-oriented text method.
- Further comments if any?

5.5 Result
11 persons performed the trial. Four of them answered they were deaf or hard-of-hearing. All persons in the trial had experience from message-oriented text communication. 10 of these also had experience from real-time text communication. Nine persons answered they preferred the real-time text method. One of the two persons preferring message-oriented text communication had never experienced real-time text communication before.

<table>
<thead>
<tr>
<th>Prefer real-time text</th>
<th>Prefer message-oriented text</th>
</tr>
</thead>
<tbody>
<tr>
<td>9</td>
<td>2</td>
</tr>
</tbody>
</table>

Table 1 Distribution in preference.

The persons who preferred real-time text conversation thought this mode was more efficient.
because you could see all the time what the other person was typing. The real-time communication was experienced to be more interactive and giving a better perception of the other person's feelings and presence.

The persons who preferred to communicate with message-wise text thought the conversation became easier to follow and understand with this mode and communicating with this method made it easier to focus on the task.

On the question “if there are conditions when one method are preferable” one person mentioned that a message-wise chat was preferred if there were more than two persons involved in the call. One person expressed the opposite opinion. Two persons claimed to prefer real-time text if the conversation was performed under time pressure.

One person had the opinion that real-time text communication can be confusing. One person found it hard to maintain a one-threaded discussion using real-time text, aiming at the experience that the other person placed a new question before the previous one was completely answered. On such occasions the questions and answers did not match. The same person claimed when using message oriented communication, it was more natural to wait for a response before continuing with next question. Several persons thought that spelling mistakes were a problem when using the real-time text method.

Several persons stated that a disadvantage with communicating with the message-oriented text method was that it is slow and inefficient. Also several persons answered that this method made the conversation less natural and fluid.

The question “how much easier do you think it was to solve the task depending on which method you used” gave the following result. Two persons thought it was much easier with the real-time text method. Five persons thought it was a little bit easier with this method. Two persons thought there was no difference in the degree of difficulty. One person thought it was a little bit easier with message-oriented text method and one person thought it was there were a lot easier with this method.

<table>
<thead>
<tr>
<th>Real-time a lot easier</th>
<th>Real-time a bit easier</th>
<th>Equal</th>
<th>Message a bit easier</th>
<th>Message a lot easier</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>5</td>
<td>2</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

Table 2 The distribution of answers about how easy it was to solve the task depending on the text communication method used.

No difference could be found in the answers from the test persons if they were hearing, hard-of-hearing or deaf.

5.6 Conclusions

The result shows that most of the people participating in the trial prefer communicating with the real-time text method. On the question “how much easier do you think it were to solve the task depending on which method you used” there were some more difference in the answers comparing to the more general questions. This could have to do with difficulties solving the problem with real-time text using the specific program the persons used during the trial and that the answers to the more general questions were based on earlier experiences.

Finally, this test shows there exists an interest for using real-time text communication, both
among deaf and hard of hearing people and hearing people. The extent of this interest is hard to assess exactly based on this trial because there were so few participants.
6 Protocol used for VoIP and ToIP

To use the Internet protocol for communication, higher level protocol in the protocol stack is needed for media transmission and for setting up sessions between nodes that wishes to communicate. A node is a sender or a receiver and a session includes all nodes and data streaming between these nodes. A session could be a multimedia conference, a VoIP-call, streaming video, a ToIP-call etc. Important protocols for the thesis are SIP, SDP and RTP and therefore these will be described here. Also RFC 4103 is described in this chapter.

6.1 Session Initiated Protocol (SIP)

When exchanging real-time streaming media like audio, text and video over IP, a protocol for establishing a connection between the communicating parts is necessary. The Session Initiated Protocol (SIP) is a signalling protocol and it is used to set up, modify and tear down calls. SIP is typically using the port 5060 and it is designed to fit with different protocols. A summary of what SIP is responsible for is: (Rosenberg 2002)

- Determinate which end system to be included in a call.
- Determinate the willingness of the called party to engage in the call.
- Determinate which media and media parameters to be used in a call.
- Establish the session parameters at all participants to be used in the call.
- Transfer and terminate sessions, modify session parameters and invoke services.

6.1.1 Components in a SIP call

There are three main components involved in a SIP call. These are user agents, servers and location servers. A user agent is the end device in a sip call for instance a soft phone. A user agent acts as both a user agent client and a user agent server. The client sends all requests while the server sends the responses. There are three types of SIP servers. These are only used during the call set up. A SIP proxy receives requests from the user agent or another proxy and directs the message to other servers or to the user agent at the end. A redirect server receives a question from a proxy or user agent and return which location the question should be sent to. An example of a redirect server is a Domain Name System (DNS) server. The last server type is a registrar server which receives SIP registration request and updates the user agent’s information into a database for instance a location server. Examples of information being stored are IP-addresses and URI. (Sinnreich 2006)

6.1.2 Call set up with SIP

When setting up a call with SIP the caller sends an invite to the person she wants to call. The invite is sent to the persons SIP URI. A SIP URI is similar to an email address and has the form: sip:name@my-domain.com. The invite includes information like media capabilities, addresses, an identifier for the call and which ports and protocol to use for the media transfer (Rosenberg 2002). The media capabilities are described with help of the session description protocol (SDP). A call set up between user A and user B could be done like this:
A wants to make a call to B. B uses the address sip:userB@b-domain.com. The using agent of A sends a request to a DNS server to look up the address to b-domain.com. The response from the DNS includes an IP-address which goes to a sip proxy with the address sip-proxy@b-domain.com. A then sends a SIP-invite to the proxy server. The proxy server looks up B’s destination address by contacting a location server who returns the registration for B which is 123456. The proxy server then contacts a DNS server who returns the corresponding URI for 123456 which is sip:userB@100.101.102.103. The proxy server then forwarding A’s invite message to the address. (Sinnreich 2006) When B gets the invitation, the user server of B sends a response back to the proxy server who directs it directly to user A. If B decides to accept the call the user server sends an OK response. With the response message also the invite message user A sent is included. The invite message is then extended with information of user B’s addresses and media capabilities. Once user A gets the response it continues by sending an acknowledge message directly to user B and then the stream of media can begin. (Rosenberg 2002)

This was an explanation of a simple call set up. It could be more advanced by using several proxy servers before reaching the end destination. Also a request can be directed to a gateway which is necessary if the participants in the call using different networks. For instance could the caller use a soft phone using the Internet while the receiver of the call using a traditional telephone and the PSTN. Also if the destination of the end user is not known a request can be forked to several addresses where the end user might be located. In figure 2, user A with the domain address there.com calls user B with the domain address there.com (Sinnreiched 2006)

![Figure 2 Call setup using SIP.](image)

### 6.2 Session Description Protocol (SDP)

When a new session is initialize all participating nodes must agree upon which media types, transport addresses to use and other important session data. SDP is a standard for how this
type of information is suppose to be defined. SDP is not dependent on a special protocol but
can be used in different environment and applications. SIP uses SDP when initiate a new
session. Information the SDP is carried includes following data (Handley 2006):

- The name and purpose of the session.
- The time when the session is active.
- Which media the session handling.
- Which bandwidth the session requires.
- Type of media for instance text.
- Which transport protocol to use, usually UDP or TCP.
- The format of the media for instance t140.

### 6.3 Real-time Transport Protocol (RTP)

The real-time transport protocol (RTP) was designed to deliver audio and video over the
Internet. The commonly used protocol for data transfer transmission control protocol (TCP)
was not suited for this type of media because it acknowledges and resend packets. User
datagram protocol (UDP) was then a better choice but still there was some features missing.
RTP does not provide any services for flow control or congestion control but it provides
information the application can use to discover this. Also it is designed only to use one
medium in a session. The advantage that comes with this is less bandwidth is needed. RTP
only transfer the media in a session while the initiating and tear down of the session often
are handled by SIP. The service RTP offers includes: (Schulzrinne 2003)

- Payload-type identification which means it shows what kind of content is being
carried.
- Sequence numbering
- Time stamping which is made for synchronization and jitter calculations.

RTP is incorporating another protocol called Real Time Control Protocol (RTCP). Its
primary function is to provide feedback on the quality of service being provided by RTP for
instance the quality of the reception. RTCP packets are delivered now and then when RTP is
used (wikipedia 2007:c).

### 6.4 RTP Payload for Text Conversation

RTP Payload for Text Conversation is the name for RFC 4103. RFC 4103 is a standard for
how to transmit data in a real-time text conversation. As the title says the protocol for
transporting the media is RTP. This protocol is not enough though for transferring real-time
text. RFC 4103 makes use of the presentation level standard for real-time text conversation
called ITU-T T.140 (ITU-T T.140). In order to signal use of T.140 the Multipurpose
Internet Mail Extension (MIME) type t140 has been registered. MIME describes standards
for media types and is encoded in UTF-8. RFC 4103 puts demands on the transport channel
delivering all packets and in right order and without copies. A special feature of T.140 is the
erasure function. It should be possible to erase a character after typing it. In order to
minimize character loss caused by packet loss there is a way to send t140 with redundancy.
This packet looks a bit different and is described in next section. The media type t140 is designed for use with SIP and RTP and is delivered in blocks in a RTP-packet of normally 1-4 characters. As mentioned in a previous chapter characters are buffered a certain time before heading to the receiver. This is why the size of the block differs. When a new call session is set up a few blocks with a symbol for empty is sent for marking the conversation has started. The first packet in a session and the first packet after an idle period should have a marker bit in the RTP-header. Empty packets may be sent with regular intervals for marking the conversation is not ended yet. As the picture show the t140 data is in the RTP-packets payload and its payload type is described in the RTP-header. (Hellström 2005)

```
0                   1                   2                   3
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|V=2|P|X| CC=0  |M|   T140 PT   |       sequence number         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                      timestamp (1000Hz)                       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|           synchronization source (SSRC) identifier            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                      T.140 encoded data                       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 3 Illustration of how T140 data is packet in a RTP-packet (Hellström 2005).

### 6.4.1 T140 with redundancy

When t140 with redundancy is used the RTP-payload not only contains primary t140 data but also blocks of data sent in previous packets. They are called redundant data. The redundant data is always from the latest previous packets and it is therefore easy to decide their sequence number by checking the sequence number for the primary data which is stored in the RTP-header. At least this is valid when t140 is the only media in the stream. When the media is alternating with other media types as described in RFC 4351 (Hellström 2006), each block needs its own header including a sequence number. This method takes more bandwidth and also does not follow the rules of RTP. If a packet is received with a higher sequence number than expected the redundant data in the packet is used to fill the gap. If there is no redundant data a symbol for missing data is inserted in the text delivered to the application. Every block of data is sent three times. In a SIP-call the invitation message needs to include a description for how many redundant generations each RTP-packet will carry. (Hellström 2005)

### 6.4.2 How t140 with redundancy is packed

All data related to the packet with t140 with redundancy is in the RTP-payload. Before the blocks of the t140 data there is one header for each data block. The first header represents the oldest data. It begins with a marker bit where one means redundant data and zero means primary data. After this bit the payload type is described as t140. Then a timestamp follows. The timestamp shows how old the data is compared to the primary data. If the data is old it will be discarded. Next declaration in the header tells how many bytes the data consist of. Each redundant data has this type of header. The primary data which header starts with a zero only have the payload type described in its header. After all headers the data comes.
First the oldest redundant data and last the primary data. By writing the block length in the header it is easy to figure out which data belongs to which header. Also the sequence number of the primary data is written in the RTP-header and the redundant data has the preceding sequence numbers. As the picture show the type of data in the RTP-header is RED, which is the format for T140 with redundancy. (Hellström 2005)

Figure 4 Illustration a RTP packet with three blocks of T140 data (Hellström 2005).

6.4.3 T140 with redundancy compared to message oriented methods

Another method for discovering and compensating for lost packets is to use a message oriented method which among others is implemented in the Transmission Control Protocol (TCP). In this method, when a packet arrives the receiver sends an acknowledge message to the sender. If the sender does not receive an acknowledge message it sends the same packet again.

There are several reasons for why this is not a good solution for real-time text communication. First, traditionally in real-time systems there is standard to not acknowledge and resend any packets because this process takes extra time which makes delays not acceptable in a real-time system. With media like audio and video also small losses will not affect the quality to much. When it comes to text small packet losses actually do matter but the possibility of sending several media at the same time increasing if the same transport protocol is in use. There are also other reasons for why the TCP-solution is not good for real-time text or other real-time media. A TCP session may get hung. This could cause long delays after which it may need to be restarted. Then there is a risk of duplication of data. With Asterisk it is possible to make conferencing call. This implies that a packet is sent to several nodes. If all this nodes should acknowledge the packet it will be much information for the sender to keep in mind for knowing who to send a repeated message to. The last reason for not using this method is when the data passes through Asterisk incoming packets are unpacked and the data is moved into a new packet. If the receiver shall acknowledge the packet all the old sequence numbers must be saved in Asterisk and matched to the new ones.
6.4.4 UTF-8

T140 is encoded in UTF-8. UTF-8 is a transform of the international 16-bit character code ISO 10646-1. Characters encoded according to UTF-8 could be up to four bytes. This means there is a capability of using a lot of characters when using this code system and for example there is not necessary to restrict to the Latin alphabet. For characters encoded with seven or less bits UTF-8 is equivalent to ASCII. (wikipedia 2007:d)

<table>
<thead>
<tr>
<th>Code range hexadecimal</th>
<th>Scalar value binary</th>
<th>UTF-8 binary/hexadecimal</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>000000–00007F</td>
<td>00000000 00000000 0zzzzzzz</td>
<td>0zzzzzzz(00-7F)</td>
<td>ASCII equivalence range; byte begins with zero.</td>
</tr>
<tr>
<td>128 codes</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>000080–00007FF</td>
<td>00000000 00000yy  yyzzzzzz</td>
<td>10yyzzzzz(80-BF)</td>
<td>First byte begins with 110, the following byte begins with 10.</td>
</tr>
<tr>
<td>1920 codes</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>000800–00FFFF</td>
<td>00000000 xxxxxxxx  yyyyyyy  yyzzzzzz</td>
<td>1110xxx(80-EF) 10yyyyyy 10zzzzzz</td>
<td>First byte begins with 1110, the following bytes begin with 10.</td>
</tr>
<tr>
<td>63488 codes</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>010000–10FFFF</td>
<td>00000000 xxxxxxxx  yyyyyyy  yyzzzzzz</td>
<td>11110www(80-F4) 10xxxxx 10yyyyyy 10zzzzzz</td>
<td>First byte begins with 11110, the following bytes begin with 10.</td>
</tr>
<tr>
<td>1048576 codes</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 3 How UTF-8 is coded. (wikipedia 2007:d).

6.4.5 Applications that supports RFC 4103

There are a few products supporting real-time text transmission as described in RFC 4103. Most of them are intended to be used by hard of hearing people according to the manufactures websites. With reservation for the possibility of missing an example here follows a description of products and companies supporting RFC 4103.

Omnitor has developed two products for total conversation over Internet. They are both using T.140 for text communication. The biggest product is called Allan eC which is an acronym for “All languages electronic Conversation” (Omnitor 2007). The low layer real-time text transmission in Allan eC is handled by eConf which is integrated in the system and developed by France Telecom (France Telecom 2007). eConf is a softphone whit video, voice and text according to RFC 4103 implemented.

The other product Omnitor is developing in a joint project with University of Wisconsin Trace Centre is called SipCon1 (SipCon1 2007). This is similar to Allan eC but made as a Java application with open source code. The main purpose with this product is to act as a developing tool, a source for product developers and as a test application.

RNID is a British organization which provides supporting products for persons that are deaf or hard of hearing. Among the products RNID offers are text telephones and a product for
real-time text conversation using a mobile phone. These products support RFC 4103. (RNID 2007)

Another British company is AuPix which has a videophone called AP-100 that implements RFC 4103. (AuPix 2007)
7 Private Branch Exchange (PBX)

Private Branch Exchange means a private telephone system which is part of a larger network system that exchange information. Traditionally the larger network system is the Public Switched Telephone Network (PSTN). The concept of the PBX was that companies could manage their own private telephone system and by that save money by not be paying for internal calls. Also each person does not need to call to the public network all the time and by having a PBX people can share links and the company does not have to pay for every people at the company having their own link. A drawback with PBX is that it is not so simple to set up and manage and also the hardware is often quite expensive. Today bigger telephone companies offer the service the PBX offer and is an alternative for companies who do not want to manage their own, this is called hosted PBX. When Internet arrived in the middle of the 90's PBX for packet switched networks was invented where the calls are carried by the Internet Protocol (IP-PBX). (pbxinfo 2007)

7.1 Functionality

The core functionality in a PBX is:

- Establish a connection between two terminals during a phone call.
- Maintain a connection as long as the users require it.
- Provide information for accounting purposes.

Other common calling feature is forwarding call, managing conferencing, transferring call, call pick up and voice mail. (wikipedia 2007:d)

7.2 IP-PBX

As mentioned, IP-PBX is used for packet switched networks and using the Internet Protocol. Many IP-PBX also have functionality for the circuit switched network with ability to transfer calls between the networks. The architecture of an IP-PBX can be described as IP, hybrid IP, and Legacy Time Division Multiplier (TDM). TDM is a type of multiplexing where a communication channels time domain is dividing into timeslots and where each terminal who wants to communicate on the channel gets one timeslot. This is used in some circuit switched network. An IP architecture only handle IP-based calls. A hybrid IP can transfer calls between the circuit switched network and the packet switched network. The third architecture is only made for handling TDM-based calls. (Black 2007) Asterisk is a hybrid IP.
8 Asterisk

Asterisk is an IP-PBX in software. It is an open source project which is developed under the GNU General Public License (GPL). It was first invented by Mark Spencer at Digium and was designed for Linux. Today it is also runnable on other operative systems for example Unix Solaris and Mac OS. (Asterisk 2007) Asterisk is implemented in C. The goal of Asterisk is to offer a design that makes it simple to include new interfaces and technologies and in the future be able to support every kind of telephony technology possible (Spencer 2003). It supports both traditional telephone services and the newer one using the internet protocol and can be configured to send traffic between different types of computer networks but also to handle the registration of necessary data to make IP based telephones compatible with traditional telephone networks (Spencer 2003). A big advantage with Asterisk being open source is that it is free to use. Anybody can download the source code and add changes to it and if some Asterisk authority approves the new code will be uploaded and available to other. This makes it very easy to adapt to new ideas. Another advantage is its open architecture that makes it compatible with many different IP-phones. A disadvantage with Asterisk is the lack of documentation. Writing documentation is not a priority to most people developing in their spare time which often is the case since it is an open source project. This makes it harder for a beginner to Asterisk to learn it. Also the complexity of Asterisk makes it harder to learn than simpler systems.

8.1 Functionality

A lot can be done with Asterisk. Van Meggelen mentions some of the things above the ordinary PBX-functionalities in his book (van Meggelen 2005). One thing is that Asterisk can be used as a gateway between a PBX based on traditionally telephone technologies and to IP based telephone technologies. It is also possible to integrate Asterisk into another PBX by letting Asterisk replacing some of the functionalities in the other PBX. Asterisk could be managed to bridge an incoming call to several numbers and it can also be used for conferencing calls. Other features includes possibility to control what actions to take when an incoming call is from a specific number, controlling which number that it is okay to call out to and controlling for how long a user is authorized to use the system just for mentioning a few.

8.2 Architecture

Asterisk is a middleware between telephony application at the top and telephony technologies in the bottom. The interfaces of the telephony technologies could be divided into three categories (Spencer 2003). These are Zaptel hardware, non-Zaptel hardware and packet voice. The first two provides integration with traditional digital and analogue telephone interface including connection to the public telephone network (PSTN). The third one including standard protocols for communication over packet switched networks and is the only interfaces that do not require specialized hardware. Examples of these standard protocols are SIP and a protocol similar to SIP called H.323. When Asterisk is started drivers providing channel drivers, file formats, codec application and more, dynamically is loaded and initialized.
8.2.1 Asterisk's API

Asterisk consists of four main application programming interface (API) (Spencer 2003). These are:

**Asterisk Application API:** all telephony application is written in separately modules that is loaded into Asterisk when it starts up. This makes it very simple to add new application files. Example of telephony application is Dial and Voicemail.

**Asterisk Codec Translator API:** a codec is a module that translate data to another type. Before a frame of data is leaving Asterisk it checks which type the frame has. If it is audio, Asterisk checks if the format type is the same as the outgoing channel request for. If it is not the same Asterisk checks if there is a codec for translating the original format to the requested one. In this way channels carrying audio compressed in different way can communicate. Theoretically this could be used for text and video as well but functionality for this is not implemented in Asterisk today. (Hitchcock 2006)

**Asterisk File Format API:** formats are modules that read or write media files to disk. With other words, this is the codec translator for internal use in Asterisk. For instance when an application ask for a particular file the File Format API is responsible for the returning file comes in the most optimal format for the application.

**Asterisk Channel API:** this includes all the technology drivers as PSTN and SIP. When a call is set up through Asterisk, incoming data carry information of which protocol to use in the call. The corresponding technology driver is then loaded. All specific protocol details are handled here.

![Asterisk core and its four API](Spencer2003)

8.2.2 The Dialplan

Asterisk is controlled by the dialplan. Here the administrator of the system can specify how to rout incoming calls and what actions to take by writing extensions. Each extension invokes an application or a function. A function could be a wait statement while an application is a telephone technology as Dial. The extensions are sorted in contexts. A context could be all incoming calls from SIP or just what default action to take and it includes one or more extension line. An extension line is written like this:

```
Exten => <exten>,<priority>,<application>,[<args>]
```

Incoming calls is routed based on its destination address, source address, time of the day
and so on. For making the dial plan easier to read several contexts could be included in each other. Below is an example of a dialplan including the contexts default, daytime and nighttime.

\[
[\text{default}]
\]

\[
\text{include} \Rightarrow \text{daytime|8:00-17:00|mon-fri}
\]

\[
\text{include} \Rightarrow \text{nighttime}
\]

\[
[\text{daytime}]
\]

\[
\text{exten} \Rightarrow \text{s/tediousperson@sip.asipserver.se, 1, Congestion}
\]

\[
\text{exten} \Rightarrow \_9XXXXXX, 2, \text{Dial(Sip/REMOTE_SERVER,20)}
\]

\[
\text{exten} \Rightarrow \text{s, 3, Dial(Sip/sara@sip.mysipserver.se,20)}
\]

\[
[\text{nightime}]
\]

\[
\text{exten} \Rightarrow \text{s,1,Playback(after_hours_msg)}
\]

The dialplan above specifies that an incoming call is handled by the default context. If the call is occurring on a Monday to Friday between 8 am and 5 pm the daytime context is executed, else the nighttime context is executed. In the nighttime context all that happens is that a message is played to the person making the call. During the day the daytime context is executed. The first extension line check if the incoming call is from the address tediosperson@sip.asipserver.se and if so executes the application Congestion which produces a congestion tone. If the call is not coming from tediosperson@sip.asipserver.se next line is executed which checks whether the destinations address begin with a 9. If so the call is routed to another server that continues rout the call. If not the call is directed to sara@sip.mysipserver.se. In the Dial application at the last line in the daytime context, SIP specifies the driver to be used and the number 20 specifies for how long to wait for the host to answer. (Spencer 2003)

8.2.3 Core

This section will explain some core functionality in Asterisk relevant for the implementation. When an incoming call arrives the Asterisk dialplan is executed. The dialplan then invokes some applications. From an application, functions in the core are invoked. In the Asterisk file system there is a catalogue called main. In this catalogue a lot of Asterisk core files are implemented.

8.2.3.1 Channel

When making a call a connection between the parties in the call is established. This is called a channel. Outgoing data is transferred on one channel while incoming data is transferred on another. To be able to use a channel a session also needs to be established. In the file main/channel there is functions handling reading and writing data from the channel. When a call is set up a structure called ast_channel is initialized for each participant in the call using an own terminal. This structure contains all the data needed during a session like addresses and data and a pointer to a technology driver structure. During a call Asterisk will in a regular manner check incoming channel for new data. Data is stored in a structure called ast_frame. An ast_frame not only stores the data but also which type of data it is, which type of format it is and its sequence number just for mentioning a few other things. The function ast_read checks if data is available on the channel and if so check what type the
data-frame has. If the data frame has the type voice and the format is not handled by the core it will be translated. The translation function can be found in the file Translation. To write data to outgoing channel the function ast_write is used. If the data frame has the type voice its format will be compared with outgoing channels format and translated if needed.

8.2.3.2 Technology Driver for SIP

The technology driver for the SIP protocol is called channel/chan_sip. This is a very complex file that includes the sip stack and all functionality the protocol is offered. In this file there is a definition of the technology structure which is reachable from the ast_channel structure. In Sip this structure is called Sip_tech. The structure includes a pointer to which function in the Sip file to use for writing data, reading data, that it is a Sip call, and so on. Another import structure is the sip_pvt. This holds information about data needed in one dialog as which media format that can be used in the dialog, which rtp-packet has been sent, the addresses to the parties and much more information. When incoming SIP data arrives a sip_pvt is initialized before the data is identified as a sipcall whereupon the session object, ast_chan is initialized.

8.2.3.3 Reading process in detail

In more detail the reading process is invoked from an application or sometimes from other functions in the core by calling the function ast_read in the file main/channel. This function reads from the incoming data channel. From ast_read the technology driver function for reading is invoked. In a Sip call this function is sip_read and this is written in channels/chan_sip. From sip_read the function sip_rtp_read is invoked. This function reads the RTP from the network by invoking ast_rtp_read. This function can be found in the file main/rtp. The function handle all data in an rtp-packet including header information and payload.

8.2.3.4 Writing process in detail

The writing process is done in a similar way as the reading process. From the file channel the function ast_write writes data to the outgoing channel by invoking the technology driver function for write. In Sip this function is called sip_write which sends the frame to the channel by invoking ast_rtp_write in the file main/rtp. This function first builds the rtp_packet with help from the function ast_rtp_raw_write and send the packet to the underlying layer in the protocol stack which is UDP.

8.2.3.5 Call set up with Sip in Asterisk

A call setup in Asterisk using SIP is working like this. The part making the call sends an invite with SIP. The invite tells information about which socket to use and an SDP description of which media format the calling part wishes to and can use. The invite is sent to Asterisk. Asterisk will look at the invite and change the SDP-information by adding formats Asterisk is able to use and clear those formats it does not recognize. Then Asterisk sends the invite to the part the call is attended to. The receiving part returns the invite after clearing those formats it can not handle. By Asterisk adding formats it is possible for one of the parties communicating with Asterisk with one format while the other part communicates with Asterisk using another format.
8.2.3.6 Translating audio data

In the session structure there is information about what format the outgoing channel requires in the data frame. Before reaching the outgoing channel, Asterisk uses this information and translates the frame. How to translate the frame is specified in a codec file. In Asterisk there is a catalogue called codecs which includes files for how to translate different format to other formats. The codec file is automatically compiled into Asterisk and it is easy just to create a new file. The codecs only work for audio.
9 Interoperability

One purpose with the implementation described in this thesis is to increase the use of real-time text communication by offer a solution for non compatible applications to communicate. Even though there is a standard for real-time text transmission, manufactures make their own design solutions. This chapter will describe what kinds of possibilities there exist for achieving interoperability, who is working with this and what purpose interoperability fulfil.

9.1 Standards

A way of achieving interoperability is to create a standard document for a system or process that everybody manufacturing a product based on this type of system or process will follow. Today there are two kinds of standards. These are De facto and De jure (Forouzan 2006). De facto are standards that have not been approved by an official recognized organization but are widespread and used. A De jure standard is approved by some official organization.

9.1.1 Standardization organizations

There are many organizations around the world working with developing and maintaining of standards. Here are some of the bigger ones providing standards for the Internet and telecommunication.

International Standards Organization (ISO): this is an international organization which member comes from other standardization organization around the world. Its purpose is to come up with word wide agreement of standardizations. (ISO 2007)

International Telecommunication Union Standardization Sector (ITU-T): this is the leading United Nation agency for information and communication technology. ITU-T mission is to act as a platform for governments and the private sector to coordinate the development of international standards for global telecom networks and services. (ITU-T 2007)

European Telecommunications Standards Institute (ETSI); this is an independent, non-profit organization, whose mission is to produce telecommunications standards for today and for the future. (ETSI 2007)

The Internet Society (ISOC); this is a non-profit, non-governmental, international, professional membership organization. ISOC provides support for the Internet standards process but also works with public policy, education and training, and membership (ISOC 2007). The standardization work is handled by the Internet Engineering Task Force (IETF). IETF is responsible for developing and reviews specifications intended as Internet standards. (IETF 2007)

9.1.2 Internet Standards

An Internet standard is a specification of how things concerning the Internet are to be specified, for instance the definition of a protocol. To become an Internet standard a specification needs to go through a strict procedure. First a working document called an Internet draft is created. If an Internet authority approves the draft it will be published as a request for comment (RFC) after about six months. When the specification first becomes an
RFC it gets the status proposal standard. The specification is then usually tested and implemented by several groups. After at least two independent and interoperable implementations and about six months the request reaches the status draft standard. After demonstrations of a successful implementation which takes about four months the specification finally reaches the status Internet standard. There are also three other status modes a RFC can have. An experimental RFC only affect an experimental environment and does not affect the operations on the Internet. An informational standard contains information like historical and tutorial related to the Internet. An historical standard are not in use anymore or never have been in use. (Forouzan 2006)

### 9.2 Why Interoperability is important

The telecom and Internet market is manufactured by several companies and organizations. The users do not want products that are not working with similar products made by other manufactures. For instance would you care for a mobile phone that only could communicate with other phones with that specific type? The European Telecommunications Standards Institute (ETSI) states that poor interoperability means bad publicity fall in manufactures stock price, loss of investor confidence and loss of customer (van deer Veer 2006). Using standards is a way to attain interoperability and maintaining the market open and competitive. A reason why some companies not follows the standards is they have their own standards which they have no attention to spread because it brings them money. Also the process of working with standards can be time consuming and there by cost money.

Not all standards are required to be implemented and also there could be functions in a standard that is recommended but not necessary to implement. It is quite obvious that all system not needs to follow standards that are not relevant for the system. An example of a required standard for Internet use is the IP protocol. The advantage with classify some functions in a standards as only recommended is the system can be working with less effort on the implementation process. The drawback is that all system must be able to handle systems without the recommended functions.

### 9.3 Interoperability according to the implementation

The implementation this thesis describes is made after an RFC with status Internet Standard. As explained this standard are not followed by all manufactures. The translator functionality in Asterisk is a way of achieving interoperability. There is a lot work on standardization processes around the world and this chapter has given examples of this which motivates the work of achieving interoperability in Asterisk. Without a compatible product the customers disappear. With no standards at all there is a risk one manufactures raises a monopoly.
10 Analysis

In the first chapters of the thesis the purpose with the thesis and relevant techniques has been presented. Also the meaning of Asterisk has been explained and what use Asterisk can fulfil. Two important conclusions have been made. First, real-time text conversation or ToIP is a relevant technique that has its market. Even if some persons seem to prefer instant messaging, there are situations when ToIP is a better alternative according to a lot of people. Also today IM is a way of communicating people is more familiar with which also could be a possible reason why some people prefer this method. Another conclusion drawn so far is the importance of a system being interoperable. To be an interesting system to the users a system has to be compatible with other systems. This motivates the extended translation functionality in Asterisk which is made for this purpose. Finally the thesis has addressed some other benefits with using an IP-PBX as Asterisk which also is a base for why functionalities for ToIP should be implemented in Asterisk.
11 Implementation

In this chapter the implementation of the extended real-time text functionalities in Asterisk is described.

11.1 Prerequisites

The implementation is made on the platform Linux Ubuntu and all code is implemented in C. The implementation is built on a special edition of Asterisk 1.2. This version of Asterisk already had support for the media type t140. The format was defined as a payload type in Asterisk which resulted in the format being accepted in a SIP-negotiation. Asterisk did not do anything with the t140 data but letting the RTP-packet with this type pass through Asterisk.

11.2 Requirements

The extensions in Asterisk shall contain functionality for taking care of incoming RTP-packets with t140 data and terminate these packets. Before reaching the outgoing media channel the data is to be put into a new RTP-packet. The end-users using the system shall be able to communicate through Asterisk with the format t140, t140 with redundancy and with one of the users using t140 while the other uses t140 with redundancy. Asterisk shall handle the translation. Asterisk shall also handle packets loss according to ITU-T T.140 and RFC 4103 where also the format t140 with redundancy is specified. The result of the implementation is to be seen as a prototype. Finally to demonstrate the implementation a test application is to be made for showing it is possible to manipulate text contents in Asterisk. The application shall present a menu with these alternatives:

- Writes a character with description. The character is typed by the user.
- Writes a sentence to the terminal which is typed by the user and ended by an enter.
- Writes a sentence to the terminal.
- Writes a character to the terminal.
- Writes a character to the terminal and returns a character typed by the user.
- Writes a sentence to the terminal and returns a sentence that is typed by the user and ended with a special character.

11.3 Design

This section describes the design of the implementation. The design was made to be simple to implement and to follow the architecture of Asterisk.

11.3.1 Design issues

Before the implementation could start, some problem needed to be solved. These were:

- Where to put the extension of the code?
- When to check for missing data and when creating the new frame?
• In what form shall the data be transferred in the core of Asterisk and the application?
• Data for redundancy needs to be stored somehow, where is a good place?
• How to implement the translator?
• How to implement the application?
• What rules shall the SIP negotiation follow when it comes to t140 and t140 with redundancy?

11.3.2 Design decisions
This section describes solutions to the design issues above and also discusses some alternatives to these design decisions.

11.3.2.1 The place of the extension of the code
The extended implementation is put in the file channel and is invoked from the functions ast_write and ast_read in the same file. In Asterisk incoming data flows on one channel and outgoing data flows on another channel and incoming data is passing through the function ast_read and outgoing through the function ast_write. The handling and the translation of audio data is invoked in ast_read and ast_write. Therefore it seemed natural to do the text translation in the same functions.

11.3.2.2 When to check for missing data and when to create the new frame
The functionality handling incoming data must be totally separated and independent from the functionality handling outgoing data. The reason is that it should be possible to let applications handle the data in between. These applications could drop frames or create new ones. Compensating for missing data must therefore be done both when data is coming into Asterisk and when it leaves Asterisk. The new sequence number for outgoing packets must be created on the way out.

11.3.2.3 How is data to be transferred through the core of Asterisk and the application
Even though the new application only was supposed to handle data, an idea was to send a string of data to the application and to other new functions in Asterisk. But in the reality, Asterisk always works with frames of data. Data from the RTP-packet is picked up and placed in a frame, when leaving Asterisk, data is packed back to the same packet or into a new one. So what ever needs to be done with the data it has to flow through Asterisk in a frame. A frame has certain attributes for instance “type: text” and “format: t140”. There are then two ways of handling the translation between t140 and t140 with redundancy. One is to first check what format outgoing channel is requested for and then on the way out translate incoming data frame to a new frame. In this way the translation would only be done one time. Another way to do the translation is to always translate incoming packets to the same format and then on the way out from Asterisk translate the new frame to the type of frame the outgoing channel is requested for. The implementation is made according to the second alternative. The reason is if Asterisk would be extended to handling 20 text formats it would be less translation function to do if all formats always was translated to the same text formats and then from there was translated to a new one. In the solution both frames with
t140 and frames with t140 with redundancy are translated to frames with the format t140. Another reason for this solution is that it is easier to add applications to Asterisk if the applications just have to support one format, which is the case if all incoming frames are translated to this before entering the rest of the core.

11.3.2.4 Storing of data

All received data must be stored somewhere for the possibility of sending it as redundant data later. The storing place needs to be maintained during the whole session. There are then two alternatives, to store the data in an already existing structure or to create a new one. The advantage with using an already existing structure is that none of Asterisk’s functions need to be changed to take a new parameter. On the other hand one of Asterisk’s structures needs to be modified. The choice fell on using an old structure because the changes would only have to take place in one structure instead of several functions. Structures including arrays of data frames were put in the session structure ast_channel. Data is stored in the structure ast_frame but the real data is only represented by a pointer in this structure. Therefore places for holding the real data were also allocated in ast_channel. A really better design would have been to only make one new structure in ast_channel and put all other new structures in there.

11.3.2.5 Translation of data

The best solution for making the translation would be to follow the same pattern as the audio translation and to create a new codec. Information about which text format to translate could be written in the session structure just as the information about what audio format to translate to were found here. And the translator functions could be invoked if the format was text as well as audio. In the end a simpler solution was implemented. The information about what format to translate to was added to the session structure but none information on how. Instead of creating a new codec all translating functions were put into the file channel. One reason for not implementing a codec was the translator structure is not adopted for text. To fix the translator functions to be more like the other design in Asterisk is probably not that hard but probably time-consuming because a lot of code needs to be moved and restructured.

11.3.2.6 How to implement the application

In Asterisk there is a catalogue called apps including all applications belonging to Asterisk. The applications are invoked from the dialplan. It seemed natural to put the test application in this catalogue because it followed the standard. The application transmits all responses to the same channel the requests are coming from. With this design only one terminal is needed when testing the application.

11.3.2.7 SIP negotiation

When using t140 with redundancy, the SDP which follows with the SIP invitation should specify the payload type being t140 with redundancy and how many generations of t140 is included in the packet. This is specified in the SIP driver. In the SIP negotiation there are three possible ways of deciding which text format to use. These are:

- Asterisk only allows a format if it is supported by Asterisk, the sender and the receiver.
Asterisk allows a format if it is supported by both sender and receiver but not necessary Asterisk.

Asterisk allows a format if it is supported by Asterisk and the receiver or the sender. The first case could seem as a natural way of handling the negotiation. Asterisk will be able to check if the packet is correct but it can also decide to just let the packets pass through.

In the second case it would be possible for two clients to communicate through Asterisk with any type of formats without anything has to be added into Asterisk. The drawback is that Asterisk want be able to do anything with the packet as check if the data is correct and also it want be possible to communicate through Asterisk with different formats.

In the third case Asterisk must be able to translate all formats it supports to each other. The pro with this method is that clients can communicate with each other with different formats. The cons are that it will not be possible to communicate through Asterisk without Asterisk being familiar with the format. The negotiation in Asterisk is working like this today and the main reason is because it is desirable that it should be possible to communicate through Asterisk with different types of text formats.

11.3.3 Description of the implementation

This is a summary of what happens with the data entering Asterisk. The incoming data frame with the format t140 or t140 with redundancy is translated to a t140 frame. Before the new frame is created, a check for missing data is done. If data is lost and the incoming packet includes redundancy the redundant data and the primary data is put together and become data in the new t140 frame. If there is no redundant data a character representing lost data is included in the new data. On the way out from Asterisk the new t140 frame is again translated into a new t140 frame or a t140 with redundancy frame. A check for missing data is done and the frame is getting a new sequence number and is then saved in the session for later use. To know which format to use, a new parameter is included in the session structure.

11.3.4 Test

In order to verify the translation is working and the format t140 with redundancy is implemented in a correct way, a test has been carried out which is presented in this section.

11.3.5 About the test

When t140 with redundancy is used is possible to loose two packets in a row without the person at the receiving terminal noticing. If more packets are lost the receiver will get a character for lost data. If packets are lost on the way into Asterisk, Asterisk is compensating for the loss. If the losses happen on the way out from Asterisk the compensating for the lost packets is handled by the receiving application. The test was performed with the caller and the receiver of the call using one out of two different applications to show the solution is not dependent on one special application. The applications used in the test are Omnitor’s application Allan eC and Omnitor’s and Trace Center’s open source application SipCon1.

11.3.6 Test case parameters

The test cases are built on the following:
- The application initiating the call.
- The application receiving the call.
- How many generations of redundant data the calling application sends.
- How many generations of redundant data the receiving application expects.
- How many packets in a row are lost on the incoming channel.
- How many packets in a row are lost on the outgoing channel.

11.3.7 The implementation of the test
During a test case it is controlled that expected data is shown on the receiving terminals application and that it is possible to erase text and change line. Also the content of the RTP-packet and the SIP-packet is checked with the program Wire Shark. Both packets to Asterisk and from Asterisk are checked. To simulate packet loss some incoming packets are destroyed when reaching Asterisk and some are destroyed when leaving Asterisk.

11.3.8 Limitations
There are some limitations in the source code which will be ignored during the test. First there is no timer which decides when packet should be sent after an idle period. Redundant data will therefore not arrive until next packet from the sender is coming into Asterisk. Secondly packets that are created in Asterisk will not have any timestamp.

11.3.9 About the testcases
The test cases are written on this form:

- \(<\text{calling application}> \rightarrow <\text{receiving application}>,\)
- \(<\text{number of data copy's calling application handles}>,\)
- \(<\text{number of data copy's receiving application handles}>,\)
- \(<\text{number of lost packets on incoming channel}>,\)
- \(<\text{number of lost packets on outgoing channel}>\)

Not all combinations are tested. The application Allan eC it is only possible to choose between sending no redundant data or two redundant data. Also all combination of the caller and the receiver using different application is not tested and finally, more test cases are constructed for checking the result for data loss on incoming channel than on outgoing channel.

11.4 Testcases
Allan eC -> Allan eC, 2, 2, 0, 0
SipCon1 -> SipCon1, 0, 0, 0, 0
SipCon1 -> SipCon1, 0, 2, 0, 0
SipCon1 -> Allan eC, 2, 2, 0, 0
Allan eC -> Allan eC, 2, 2, 1, 0
SipCon1 -> SipCon1, 0, 0, 1, 0
The test cases above were successful with following remarks.

- The function for text erasure will not be in synchronous on the transmitting part’s terminal and the receiving part terminal. This is because of the applications sometimes sends empty characters. The application want be able to know if it is a real character that is lost or an empty character and will erase even if it is an empty character that is missing. This problem is not really associated to Asterisk, but a result of idle traffic included in the applications.

- When t140 with redundancy is used by the sending application and the sending application is Allan eC sometimes data with the type t140 is sent instead of data with type t140 with redundancy. When this occurs one generation of redundant data will never be sent. The phenomenon only happens occasionally and it is not a big problem.
12 Conclusions

In the thesis ToIP and Asterisk has been explained. It has been described why an extension for real-time text is good to have in Asterisk. Firstly real-time text cannot be substituted with instant messaging because real-time text is more interactive and similar to a telephone call. Secondly the importance of interoperable systems has been addressed and finally the benefits with using Asterisk have been presented. The result is a prototype of Asterisk where RFC 4103 is implemented with a few exceptions. The prototype is working but some of the design must be modified if it should be possible to upload the code at the official Asterisk homepage.

12.1 What is left to do

In RFC 4103 it is defined that after an idle period, if packet with redundant data has not been sent yet it must be sent. A timer is supposed to keep in mind for how long the channel has been idle. This is not implemented in the prototype. The code for the translation in the prototype is not written in the correct form. Ideally should all the translation function be integrated in a new codec module. Finally, in the new t140 packet with redundancy the payload type in the header is static. It should be the same payload type as the receiving application expects. Often this mean the same number but it does not have to be that way.

12.2 Future Work

The translating between t140 and t140 with redundancy is just one example for text translation. In the future this could be extended with translating functions for other formats. By always translating incoming data with the type text to t140 a lot of translating functions between t140 and other formats could be done. An example of another format is modem sounds adapted for text telephones. Omnitor already develops this type of translator and with it integrated in Asterisk it would be possible to use t140-media to communicate with the ordinary telephone network by using an arbitrary VoIP-gateway. Other formats to translate to could be formats used in instant messaging applications. With more translating functions the availability of ToIP communication would increase.
13 References

The references are divided into four categories. The first and most important one includes books, publications and informational websites. The second category includes standards. The third one is official websites from company and organization which is mentioned in the report. The fourth one includes sites from Wikipedia where no author to the text is official. Only general information is referenced to this category.

13.1 Books and publications

Halsal, Fred (2006), Computer Networking and the Internet 5 edition, Addison Wesley

13.2 Standards


13.3 Information about specific products and companies based on their official websites

France Telecom (2007) (Information about products supporting T140),
Netwise (2007) (Information about products supporting T140),
Omnitor (2007) (Information about Allan eC) <http://www.omnitor.se/>
PBX info (2007) (Information about PBX), <www.pbxinfo.com>
RNID (2007) (Information about products supporting T140),
<http://www.rnid.org.uk/howwehelp/research_and_technology/communication_and_broadc
asting/text_and_voice_communications/textphones_and_interactive_texting/>
SipCon1 (2007) (Free for downloading), <http://sourceforge.net/projects/tipcon1/>

13.4 Information from Wikipedia
<http://en.wikipedia.org/wiki/Real_time_control_protocol>
A How to configure and run Asterisk

The appendix describes how to compile and run Asterisk. It also describes which configuration files that have been changed for the new code to work. The code is located at the T-hybrid server at Omnitor.

A.1 Compilation and Installation

When changing something in the source code Asterisk needs to be compiled and installed. This is done from the directory ~/funkar2/vidcaps. To compile and install write the command “make” followed by the command “make install”. When only small changes are made like a line is made to a comment it can happen that it will not be included in the compilation. It can then help to remove the .o file for the current file and then recompile.

A.2 Running and testing

To run Asterisk, type the command “Asterisk”. To start the Asterisk console type the command “Asterisk –r”. When Asterisk is recompiled the server needs to be shut down. This is done by the command “stop now” from the Asterisk console. There is more than one way to print information from the code when it is running. One is to add this line in the code:

\[\text{ast\_verbose(“a message”);}\]

This is working approximately as printf and will write “a message” to the console when executed if first the command “core set verbose 10” is written to the Asterisk console. This is one way of testing the code.

A.3 Configuration files

This section describes what changes in configuration files that were necessary for the new code to work. Extended configuration files are:

- extensions.conf
- sip.conf

A.3.1 extensions.conf

This file includes the dialplan. This file can be changed for testing the new functionality. For the moment the added test functionalities is in the context “incoming from SIP” where this line is added:

\[\text{Exten=> 2345,1, Dial(SIP/sara2@sip.omnitor.se)}\]
\[\text{Exten=> 5678,1, Dial(SIP/sara.horlin@sip.omnitor.se)}\]
\[\text{Exten=> 6789, Goto(T140test, s, 1)}\]
\[\text{[T140test]}\]
\[\text{Exten=> s,1,Answer()}\]
\[\text{Exten=> S,n,Wait(5)}\]
\[\text{Exten=> S,n,T140test()}\]
A.3.2 sip.conf

In this file the port-number and the IP-address SIP is using on the machine is specified. SIP always uses the port 5060 which is specified under bindport. The IP-address 192.168.0.34 is the IP-address to the machine where the SIP server is located in Omnitons local network. In this file also all formats SIP is suppose to accept must be specified. “allow=t140” means Asterisk should allow the format t140 in a SIP-call.

[general]
bindport = 5060
bindaddr = 192.168.0.34
allow=t140
allow=red
B How to add a new application

To add a new application in Asterisk, create a new file in the directory ~/funkar2/vidcaps/apps and name the file to app_<application name>.c. To make it possible for Asterisk to load and unload the new file in the Asterisk core create the functions load_module and unload_module. This function looks the same in all application file and can be copied from another application. With this file the new application will dynamically be compiled and loaded into Asterisk when Asterisk is compiled and installed. The function load_module takes a function pointer to a function that executes the application. This function is called <application name>_exec. From this function all other functions in the application is invoked directly or indirectly. In order to be able to read and write data from the channel into the application and then back on a channel, use the functions ast_read and ast_write. The application is invoked from the dialplan in the file ~../etc/asterisk/extension.conf. If the application only handles data frames with the type text incoming frames must been controlled and dropped if the frame has another format.
C Changes in the source code

The appendix describes changes made in the source code. The new code could be found on the server called T-Hybrid owned by Omnitor under the directory ~/funkar2/vidcaps.

C.1 Source code

Changes are committed into these files:

- main/channel.c
- main/channel.h
- main/rtp.c
- main/frame.c
- main/frame.h
- channels/chan_sip.c

The test application is in a new file:

- apps/app_t140test.c

The changes to and the new code is presented in this format:

<table>
<thead>
<tr>
<th>Function/Structure</th>
<th>&lt;name&gt;</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>&lt;Description of the functionality&gt;</td>
</tr>
<tr>
<td>Description of changes</td>
<td>&lt;What does the new code include&gt;</td>
</tr>
</tbody>
</table>

C.1.1 Changes in the source code

This section describes what have been changed in the different files.

C.1.1.1 main/rtp.c

<table>
<thead>
<tr>
<th>Function</th>
<th>ast_rtp_write</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>Writes the content into an RTP-packet inclusive the header.</td>
</tr>
<tr>
<td>Description of changes</td>
<td>If the data is T140 or RED the sequence number will be set to the one in the ast_frame. The marker bit in the RTP-header is set to one if at least 330 ms has gone since the last frame passed or if it is the first packet in a session.</td>
</tr>
</tbody>
</table>
Function
ast_rtp_read

Description
Reads the content in an RTP-packet.

Description of changes
In ast_frame there is a variable called ts. This is set to the timestamp from the rtp-header.

Struct
mimeTypes

Description
Define the MIME media types.

Description of changes
The type AST_FORMAT_RED is added.

C.1.1.2 main/frame.c

Struct
static RTP_PT

Description
RTP payloadtypes.

Description of changes
Red is added with the number 99 and T140 is changed to the number 98. Also h264 is changed to 100. The changes were necessary to make it compatible with SipCon1. If a bug is fixed in SipCon1 the number can be changed to something else.

Struct
AST_FORMAT_LIST

Description
Definition of supported media formats.

Description of changes
The format RED is added to this list.
### C.1.1.3 include/asterisk/frame.h

<table>
<thead>
<tr>
<th>Constant</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AST_FORMAT_RED</td>
<td>Media format.</td>
</tr>
</tbody>
</table>

### C.1.1.4 channels/chan_sip.c

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
<th>Description of changes</th>
</tr>
</thead>
<tbody>
<tr>
<td>sip_read</td>
<td>Read SIP RTP from channel.</td>
<td>The parameter ast-&gt;textformat is set.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
<th>Description of changes</th>
</tr>
</thead>
<tbody>
<tr>
<td>add_tcodec_to_sdp</td>
<td>Ads supported text formats to the SDP.</td>
<td>If the format is RED, the fmp line will be set in SDP according to RFC 4103.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
<th>Description of changes</th>
</tr>
</thead>
<tbody>
<tr>
<td>process_sdp</td>
<td>Process SDP and activates RTP channels.</td>
<td>The RED-format is added to the text-rtp structure.</td>
</tr>
</tbody>
</table>
### C.1.1.5 main/channel.c

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
<th>Description of changes</th>
</tr>
</thead>
<tbody>
<tr>
<td>ast_channel_alloc</td>
<td>Initialize the session structure ast_channel.</td>
<td>All new variables added to ast_channel is initialized here.</td>
</tr>
<tr>
<td>__ast_read</td>
<td>Listen for data on incoming channel and handle data when it is available.</td>
<td>There is a case statement based on frametype. Under the alternative text new code is added. The new code checks whereas the frame is T140 or RED and translates the frame.</td>
</tr>
<tr>
<td>ast_write</td>
<td>Handles data to outgoing channel.</td>
<td>There is a case statement based on frametype. Under the alternative text new code is added. The new code checks whereas the frame is T140 or RED and translates the frame.</td>
</tr>
<tr>
<td>t140_create_empty</td>
<td>Makes a T140 frame empty.</td>
<td>This is a new function.</td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
<td>Description of changes</td>
</tr>
<tr>
<td>-------------------</td>
<td>------------------------------------------</td>
<td>------------------------</td>
</tr>
<tr>
<td>bin</td>
<td>Convert an integer into binary.</td>
<td>This is a new function.</td>
</tr>
<tr>
<td>char_to_bin</td>
<td>Convert a char into a binary.</td>
<td>This is a new function.</td>
</tr>
<tr>
<td>transform_from_red_to_t140</td>
<td>Transform a RED frame to a T140 frame.</td>
<td>This is a new function.</td>
</tr>
<tr>
<td>transform_from_t140_to_red</td>
<td>Transform a T140 frame to a RED frame.</td>
<td>This is a new function.</td>
</tr>
<tr>
<td>save_redundant_data</td>
<td>Saves a data frame for later use.</td>
<td>This is a new function.</td>
</tr>
</tbody>
</table>


<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
<th>Description of changes</th>
</tr>
</thead>
<tbody>
<tr>
<td>save_redundant_data_with_data</td>
<td>Saves a data frame at a specific index.</td>
<td>This is a new function.</td>
</tr>
<tr>
<td>create_new_t140_frame</td>
<td>Creates a new T140 frame.</td>
<td>This is a new function.</td>
</tr>
<tr>
<td>check_for_missing_data</td>
<td>Check if there is any lost data.</td>
<td>This is a new function.</td>
</tr>
<tr>
<td>look_up_diff</td>
<td>Check if there is a gap between the new sequence number and the previous one.</td>
<td>This is a new function.</td>
</tr>
<tr>
<td>create_new_seqno</td>
<td>Creates a new sequence number.</td>
<td>This is a new function.</td>
</tr>
</tbody>
</table>
C.1.1.6 include/asterisk/channel.h

<table>
<thead>
<tr>
<th>Structure</th>
</tr>
</thead>
<tbody>
<tr>
<td>ast_channel</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Includes all data needed in a session.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Description of changes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Parameters added are: prev_frames, red_data0, red_data1, red_data2, red_data20, red_data21, red_data22, index, index2, index3, index4, index5, textformat, red_packet, prev_seqn2, prev_seqno3, new_seqno and new_seqno2.</td>
</tr>
</tbody>
</table>

C.1.2 The test application
The test application is in the catalogue apps.

C.1.2.1 apps/app_t140test.c

<table>
<thead>
<tr>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>t140test_exec</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Executes the application and writes a menu to the terminal. Wait for the user to make a choice.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Description of changes</th>
</tr>
</thead>
<tbody>
<tr>
<td>This is a new function.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>t140_skriv</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Writes to the channel.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Description of changes</th>
</tr>
</thead>
<tbody>
<tr>
<td>This is a new function.</td>
</tr>
<tr>
<td>Function</td>
</tr>
<tr>
<td>------------------</td>
</tr>
<tr>
<td>op_t140</td>
</tr>
<tr>
<td>t140_get_string</td>
</tr>
<tr>
<td>t140_get_string_in</td>
</tr>
<tr>
<td>t140_get_char</td>
</tr>
<tr>
<td>Function</td>
</tr>
<tr>
<td>-------------------</td>
</tr>
<tr>
<td>t140_get_char_in</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>t140_send_char</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>t140_send_string</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>skriv_tecken</td>
</tr>
<tr>
<td>Function</td>
</tr>
<tr>
<td>----------------</td>
</tr>
<tr>
<td>skriv_mening</td>
</tr>
<tr>
<td>T140_tom</td>
</tr>
<tr>
<td>t140_return</td>
</tr>
<tr>
<td>T140_endstring</td>
</tr>
<tr>
<td>T140_specialchar</td>
</tr>
</tbody>
</table>


<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
<th>Description of changes</th>
</tr>
</thead>
<tbody>
<tr>
<td>unload_module</td>
<td>Unload the application driver from Asterisk.</td>
<td>This function looks the same in all application files.</td>
</tr>
<tr>
<td>load_module</td>
<td>Load the application driver into Asterisk.</td>
<td>This function looks the same in all application files.</td>
</tr>
</tbody>
</table>