Asterisk SS7 Installation

1 Abstract

Asterisk is an open source software private branch exchange (PBX). It has an extensive large functionality representing a traditional PBX system, voice mail, interactive voice response system, automatic call distributor or signalling and media gateways. Asterisk can interface to both traditional TDM based systems (PSTN networks) and packet based systems (VoIP networks). In our project we have focused on interconnecting Asterisk PBX with a PSTN using a Signalling System #7 (SS7) in a role of a call control mechanism.

Keywords: Asterisk, SS7, chan_ss7, libss7 2 Introduction 2.1 Asterisk

Asterisk is a software private branch exchange (PBX). Unlike the traditional PBX being bound to vendor specific equipment, Asterisk can be run on almost any kind of hardware which uses Linux, BSD or Mac OS as their operating system.

Figure 1: Asterisk logo

Asterisk offers a rich and flexible set of features including classical PBX services as well as advanced features. For example voice mail, interactive voice response, conference calling, automatic call distribution, call forwarding, identity services and call detail records.

Asterisk can interface both traditional TDM based systems (PSTN networks) and packet based systems (VoIP networks). For interconnection with Voice over IP (VoIP) networks, Asterisk doesn't need any additional hardware. However, for interconnection with PSTN networks and almost all standards-based telephony equipment an additional hardware is needed (vendors such as Digium or Sangoma). These hardware devices come in a form of PCI cards offering analog or digital telephony interfaces (ISDN BRI, PRI).

Asterisk supports a wide range of protocols. Among natively supported VoIP protocols are Session Initiation Protocol (SIP), Inter Asterisk Exchange protocol (IAX), Media Gateway Control Protocol (MGCP), H.323 and CISCO Skinny protocol (SCCP). Other protocols can be supported via third-party add-ons.

The Asterisk TDM protocol group includes most of the European and American standard signalling types, for example analog E&M, FXO, FXS and multifrequency tones as well as digital (ISDN) signalling protocols.

For encoding of voice for transmission over IP networks, Asterisk offers a number of codecs - ADPCM, G.711, G.723.1, G.726, G.729, GSM, iLBC, LPC-10 or Speex - being capable of transcoding between most of these.

One conclusion from the capabilities named above - Asterisk can perform as a signalling and media gateway, between VoIP providers using different signalling protocols and codecs, but also as a gateway between PSTN and VoIP world.

2.2 Signalling System #7

Common Channel Signalling System No. 7 (also referred as SS7, CCS7 or C7) is a suite of control protocols used to provide instructions between elements within a telephony network.

These instructions include information about how to route calls within a network, requested services by subscribers, identities of participants or, for example, management information about signalling network.

Generally, two types of signalling exist - signalling between a subscriber and the local exchange referred to as subscriber signalling and a signalling between telephony exchanges. SS7 falls into the second group called network signalling.

SS7 is a digital signalling method that uses a separate (from voice&data network) packet-switched network to transfer messages. That allows reliable and faster connection setup and teardown of a connection and information exchange parallel to the current call.

SS7 provides two types of services: (voice-) circuit related and non-circuit related services. First group is used for setup and teardown of voice connections, while non-circuit related services are for example network management or database access for number translations and subscriber information retrieval.

As the digital telephony networks evolved, a demand for new services emerged. As a result, the concept of Intelligent Network (IN) was specified moving the service logic out of the switching platform onto specific servers. The IN still utilizes SS7 for call control and network management mechanisms and newly adds the database access to support these new functions.

Several organizations have written standards for SS7 networks, for example ITU, ANSI or IETF. On the international level of interconnection between signalling networks the ITU-T standard of the SS7 protocol is used, while the national level of the network can use any standard that exists within the country, for example the ANSI standard in the United States. The most common SS7 standard used in Europe is ITU variant.

All nodes in SS7 network are called signalling points and they have assigned a unique number the Signalling Point Code (SPC).

Nodes in signalling network can be of two types: a Signalling Point (SP) and a Signalling Transfer Point (STP). SP is a source and a destination of signalling messages. STP on the other hand only transfers messages in the signalling network.

All signalling nodes are interconnected via signalling data links. These links are bidirectional with typical speed of 64 kbps or multiples. Links are placed into groups called linksets. A collection of linksets used to reach a destination is called a route. And a collection of routes is known as routeset, where multiple of these are necessary to ensure reliable message transfer between signalling endpoint.

Figure 2: SS7 network.

The SS7 protocol stack used in a conventional TDM-based network consists of four layers. The lowest three layers are called Message Transfer Part (MTP).

MTP layer 1 represents a physical network interface responsible for conversion of a bit stream into an electrical or optical signal transmitted over the medium.

MTP layer 2 ensures a reliable message transfer between adjacent signalling points. Functions of MTP2 include error detection and correction mechanisms and sequencing of messages between nodes.

The network level - MTP layer 3 is responsible for message routing and network management. The routing function analyzes the signalling messages based on calling and called party addresses and delivers it to a higher level protocol or forwards the message to the next signalling point in its way. MTP3 network management function monitors the signalling network for cases of congestion, processor outages or link breakdowns and reroutes the traffic around these links and nodes.

The service of MTP layers is being used by the upper User Part layer (UP). Three types of UP are defined: Telephony User Part (TUP), Data User Part (DUP) and ISDN User Part (ISUP). TUP supports basic call setup and tear-down on analog circuits only. DUP defines the necessary call control for circuit-switched data transmission services. TUP and DUP are used only in some parts of the world and are being replaced by ISUP.

ISUP is the signalling protocol used to establish, maintain, and release circuit-switched connections across an SS7 network. The most common ISUP messages include IAM, ACM, ANM, REL and RLC. Initial Address Message (IAM) is a message sent to establish a connection. The message body includes Circuit Identification Code (CIC) informing the partner switch about which voice circuit the call will be placed on, information identifying calling parties - called and calling party number and type of service (speech or data) and many more optional parameters. Address Complete Message (ACM) is a message sent by the terminating side of a connection after collecting enough information to reach the called party. Answer Message (ANM) is sent when called party answers the call and charging is started. Release (REL) clears the call and Release complete (RLC) message acknowledges the call release and the physical timeslot can be used again.

Figure 3: SS7 protocol stack.   3   SS7 support in Asterisk 3.1   LIBISUP

Libisup was the first SS7 solution intended for use with Asterisk. It represents a commercial product developed by Cosini Technologies and licensed under the Digium commercial license. Libisup replaces in the Asterisk architecture the libpri (a standard ISDN PRI library) and enables applications to make use of standard Asterisk features and applications.

Hardware that can be used as an interface to the PSTN with SS7 signalling is represented by Digium E1 cards.
3.2   ss7box

A Digium rival Sangoma introduced SS7 solution called ss7box that comes together with ss7boost. The software is used for interconnecting PSTN and IP networks in general and utilizes Sangoma hardware cards.

ss7box and ss7boost functionality can be described as SS7 router and signalling gateway for both SS7 variants (ITU and ANSI) and enables call setup and intelligent network functions over an IP network. 3.3   chan_ss7

Asterisk SS7 channel driver (chan_ss7) is an open source software developed by Danish company SIFIRA A/S. The license for chan_ss7 is GPL (General Public License). It is not officially certified for the SS7 interoperability.

This solution includes an implementations of MTP2 layer, bare essentials of MTP3 and a large subset of ISUP functions. The support for multiple linksets with different DPCs and multiple hosts configuration with a load sharing and failover could be interesting from the network management point of view. 3.4 libss7

Libss7 is the latest implementation of SS7 support for Asterisk released by Asterisk developers in summer 2006. Like libisup, it represents a library replacing libpri and making use of chan_zap as channel driver for Asterisk and a zaptel hardware as an interface to SS7 network. The functionality covered in this solution is the ITU variant of SS7 implementing MTP2, MTP3, and ISUP protocols. 4 Installation of Asterisk

We have used Debian GNU/Linux for our tests of Asterisk and SS7 signalling. Before we started the Asterisk installation process, we had to satisfy several preconditions, namely the following software packages:

- linux 2.6 kernel headers (for kernel version 2.6. Otherwise for kernel 2.4 we need linux 2.4 kernel sources)
- bison and bison-dev
- ncurses and ncurses-dev
- zlib1g and zlib1g-dev (data compression)
- libssl and libssl-dev (cryptography)
- libnewt-dev (zaptel tool)
- initrd-tools, cvs and procps

We used the 1.2 he version of Asterisk obtained as source code from the Asterisk official web page. Three packages were required for the basic installation - libpri, zaptel and Asterisk itself.

The libpri library implements ISDN PRI for E1/T1/J1 interfaces. Zaptel is a kernel interface device driver for analog or digital interface cards. Asterisk supports variety of hardware interface cards, but we used natively supported Digium cards (model TE110P, a digital single E1 port card). For other hardware, related device drivers have to be installed.

We unpacked downloaded packages and installed them in following order using the standard make procedure.

(Note: Command lines starting with '#' symbol represent super user (root) mode, while '$' indicates non-root user.)

```
# cd /usr/src/asterisk/libpri
# make
# make install

# cd /usr/src/asterisk/zaptel
# make linux26 # using kernel version 2.6 (for 2.4 use make only)
# make install

# cd /usr/src/asterisk/asterisk
# make mpg123 # mpg123 for mp3 music on hold play
# make
# make install
# make samples # creation of samples configuration files
```

5 Asterisk SS7 channel driver 5.1 Installation

We have decided to use in our project the open source implementation of SS7 support in Asterisk, the "SS7 channel driver" (chan_ss7). Source code of version 0.8.3 was downloaded from the SIFIRA web page.

During the installation process we encountered several problems. The compile process reported missing header file. zaptel.h is by default placed in /usr/include/linux directory. We solved this problem by editing header definitions in chan_ss7.c and mtp.c files. Similarly, when fasthdlc.h header file was missing we copied the file from zaptel source code directory to /usr/include.

Successful compilation process created the chan_ss7.so module. With "make install" we moved the module into the Asterisk module directory located in /usr/lib/asterisk/modules/. 5.2   Configuration

Configuration of the SS7 channel driver is stored in a text file ss7.conf. We have created this configuration file based on a template that comes with the chan_ss7 source code and placed it to the Asterisk default configuration directory (/etc/asterisk). As the file includes configuration for both ends of the signalling link, it can be used on both hosts.

Content of the file is divided into three sections. The first one defines a signalling linkset and its properties. The next section describes a signalling link. It defines properties such as which signalling linkset the link belongs to or which channels are used for user data and which for signalling. In our scenario, we defined a signalling linkset called "siuc" including two signalling links "l1" and "l2" both using time slot number 16 for signalling and remaining time slots for user traffic. Last section defines the signalling point itself. It specifies the name of the host, signalling links it can use and signalling point codes that belong to this SP. Here is a sample configuration:

```
[linkset-siuc]
enabled => yes ; The linkset is enabled
enable_st => no ; The end-of-pulsing (ST) is not used to determine
             ; when incoming address is complete
use_connect => yes ; Reply incoming call with CON
               ; rather than ACM and ANM
hunting_policy => even_mru ; The CIC hunting policy is even CIC
                    ; numbers, most recently used
context => ss7 ; Incoming calls are placed in the ss7 context
             ; in the asterisk dialplan
language => da ; The language for this context is da

[link-l1]
linkset => siuc ; This link belongs to linkset siuc
channels => 1-15,17-31 ; The speech/audio circuit
                  ; channels on this link
schannel => 16 ; The signalling channel
firstcic => 1 ; The first CIC
enabled => yes ; The link is enabled

[link-l2]
linkset => siuc
channels => 1-15,17-31
schannel => 16
firstcic => 1
enabled => yes

[host-adela]
; chan_ss7 auto-configures by matching the machines'
; host name with the host-<name> section
; in the configuration file, in this case 'adela'.
enabled => yes ; The host is enabled
opc => 0x1 ; The point code for this SS7 signalling point is 0x1
dpc => siuc:0x2 ; The destination point (peer) code is 0x2
             ; for linkset siuc
links => l1:1 ; Syntax links => link-name:digium-connector-no

[host-bolek]
enabled => yes
opc => 0x2
dpc => siuc:0x1
links => l2:1
```

After configuring the chan_ss7 module, we had to configure the Digium hardware interface card driver - zaptel. We have defined the interface as a single E1 interface card with ccs signalling (SS7) and a standard hdb3 encoding using a remote timing source. Time slots were labeled as "B" channels (traffic channels) meaning no conversion or signalling is performed, raw data is available on the master and the channels are treated individually (not bundled). Even the timeslot number 16 used for signalling is configured as a B channel, because the signalling is handled entirely in the userspace in chan_ss7.

The system configuration is placed in /etc/zaptel.conf and the file contains the following lines:

```
# span=<span num>,<timing source>,<line build out>,<framing>,<coding>[,yellow]
span=1,1,0,ccs,hdb3
# <device>=<channel list>
bchan=1-31
```

Details about configuration can be found in Appendix.

5.3 Startup

We had to initialize the hardware by Zaptel Configurator (ztcfg) before starting the Asterisk. ztcfg parsed the configuration file (zaptel.conf) and set the correct values. Then we loaded the modules controlling hardware interface using the standard modprobe command. We have first loaded the generic zaptel module and then the card specific module (wcte11xp for Digium WildCard TE110P).

```
# modprobe zaptel
# modprobe wcte11xp
$ lsmod
Module         Size   Used by
wcte11xp      24864   31
zaptel       224260   68 ztdummy,wcte11xp
crc_ccitt      2432   1 zaptel
```

When the kernel modules were successfully loaded, we initiated the Asterisk startup. The SS7 channel driver module (chan_ss7.so) is automatically loaded with Asterisk if "autoload" is set to "yes" in /etc/modules.conf. Successful initiation of the SS7 connection was reported by Asterisk on the command line interface (CLI) by informing that MTP was UP and INSERVICE and a group reset of circuits had happened.

Figure 4: SS7 channel module load   at Asterisk CLI. (large image)

5.4 Testing

We have tested the functionality of the SS7 channel driver in two stages.

The first scenario used two Linux servers with installed Asterisk application combined with chan_ss7 module support. These servers were equipped with the TE110P interface cards and interconnected by an E1/PRI crossover cable. After the Asterisk startup we loaded manually the SS7 channel driver on both servers. The initial procedure of MTP alignment and a group reset of E1 circuits run successfully. Then we passed calls in both directions. The connection was quick and reliable and the voice quality was excellent.

In the second phase of our tests, we interconnected an Asterisk with the SS7 channel driver to a live PSTN switch - the Ericsson AXE platform. After solving initial problems with SPCs (Ericsson using decimal point codes, while chan_ss7 uses hexadecimal), we have successfully interconnected the exchanges. Initialization process of the SS7 connection was identical to the previous case and the successful outcome was confirmed by Asterisk on the console. Calling in either direction was successful and the connection was reliable for the whole period of testing - approximately a week.

Besides the basic call setup we tested several supplementary services (SS). The implementation of calling line identity services - presentation and restriction (CLIP, CLIR) - follows the recommendation, while the connected line presentation and restriction seem not to be supported. The call waiting and call hold SS were also successfully tested with successful.

Figure 5: Example of SS7 call   displayed in Asterisk CLI. (large image)

6 Native Asterisk SS7 support

6.1 Installation

Libss7 was the second SS7 solution for Asterisk we selected for testing. It represents the native SS7 support for Asterisk implemented by Digium developers. Libss7 comes in a form of a Linux library replacing the standard ISDN PRI library (libpri). The solution makes use of zaptel channel driver (chan_zap) to control incoming and outgoing calls via the zaptel hardware. It supports only the ITU variant of SS7 implementing MTP2, MTP3 and ISUP protocols.

Libss7 library doesn't support older versions of Asterisk (1.2 or older). That leaves us with either using Asterisk 1.4-beta version or a development branch called "trunk". We have decided to use the trunk version of Asterisk and a trunk version of both of its components - libpri and zaptel.

To obtain the current version of the source code, we had to install Subversion on the Linux system. Using it, we downloaded the source code, compiled it and installed on the system.

libss7-trunk:

```
$ svn co http://svn.digium.com/svn/libss7/trunk libss7-trunk
$ cd libss7-trunk
$ make
# make install
```

libpri trunk: $ svn co http://svn.digium.com/svn/libpri/trunk libpri-trunk
```
$ cd libpri-trunk
$ make
# make install
```

zaptel trunk: $ svn co http://svn.digium.com/svn/zaptel/trunk zaptel-trunk
```
$ cd zaptel-trunk
$ ./configure
$ make linux26 # using 2.6 kernel
# make install
```

asterisk trunk: $ svn co http://svn.digium.com/svn/asterisk/trunk asterisk-trunk
```
$ cd asterisk-trunk
$ ./configure
$ make
# make install
```

Successful installation of the libss7 library is confirmed by the configure script of asterisk-trunk by finding libss7 and then during process of compilation by creating a chan_zap.so module with libss7 support. 6.2   Configuration

Two Asterisk configuration files had to be modified to setup the SS7 connection. The configuration of zaptel device in zaptel.conf is similar as for ISDN PRI. We defined the physical interface parameters (number of E1s, synchronization, framing and coding, etc.), specified the channels used for user traffic and signalling. An example configuration:
```
# span=<span num>,<timing source>,<line build out>,<framing>,<coding>[,yellow]
span=1,0,0,ccs,hdb3
# <device>=<channel list>
bchan=1-15
dchan=16
bchan=17-31
```

The second configuration file zapata.conf is based on a template enclosed with asterisk-trunk branch that can be found in the configs/zapata.conf. We defined the signalling type, SS7 variant, number of linksets, point codes (OPC, DPC) and traffic and signalling channels as shown in the following example.
```
; Signaling type SS7
signalling = ss7
; Variant of SS7 signaling:
; Options are itu and ansi
ss7type = itu
; All settings apply to linkset 1
linkset = 1
; Point code of the linkset
pointcode = 2
; Point code of node adjacent to this signaling link
; (Possibly the STP between you and your destination)
adjpointcode = 1
; Default point code that you would like to assign to
; outgoing messages (in case of routing through STPs,
; or using A links)
defaultdpc = 1
; What the MTP3 network indicator bits should be set to.
; Choices are national, national_spare, international,
; international_spare
networkindicator=international
; First signaling channel
sigchan = 16
; Begin CIC (Circuit indication codes) count with this number
cicbeginswith = 1
; Channels to associate with CICs on this linkset
channel = 1-15
; another cicbeginwith, so channel 16 is used for signalling
```

```
cicbeginswith = 17
channel = 17-31
```

After we configured the zaptel.conf and zapata.conf, we could initialize the zaptel hardware device and load the kernel modules.

```
# ztcfg -vvv
# modprobe zaptel
# modprobe wcte11xp
```

After the modules were successfully loaded, we initiated the Asterisk startup. The SS7 link is brought up automatically when Asterisk starts.

6.3 Testing

We have already tested a basic call setup with the libss7 solution and other tests are still underway.

As a test platform we have interconnected two Linux servers, one using Asterisk with the SS7 channel driver (chan_ss7) and the other hosting Asterisk application including libss7 and using standard zaptel channel (chan_zap). After initial configuration, module loading and application startup, the SS7 link was automatically initialized with a successful outcome and calls were exchanged in both directions.

7 Conclusions

We have successfully experimented with two solutions that enable Asterisk PBX to use the SS7 network signalling system. These solutions were Asterisk SS7 channel driver (chan_ss7) and the native Asterisk SS7 support (libss7). Both are open source and we not aware of previous experience with them in the Czech Republic.

Asterisk SS7 channel driver (chan_ss7) was successfully tested for call services and related supplementary services against a public PSTN switch - the Ericsson AXE platform. The native Asterisk SS7 support (libss7) is still being tested and has so far proved to be a solution capable of handling calls using standard SS7.

The future steps in this activity will cover further tests of both solutions concerning interoperability in the network and supplementary services support according to the international recommendations.

Another area of interest are the possibilities of transmission of SS7 messages over the Internet Protocol.

References and Links

-

Tomá&scaron; Wija, David Zukal and Miroslav VozHák, "Asterisk a jeho použití", CESNET Technical reports, December 2005,

http://www.cesnet.cz/doc/techzpravy/2005/voip/asterisk.pdf

-

T. Russell, "Signalling System #7", McGraw-Hill (2002), ISBN 0-07-138772-2

-

Jim Van Meggelen, Jared Smith, and Leif Madsen, "Asterisk: The Future of Telephony", O'Reilly Media (2005),

http://asterisk.stablehosting.net/AsteriskTFOT.zip

-

The Official Asterisk Community Pagehttp://www.asterisk.org/

-

SIFIRA A/S home page

http://www.sifira.dk/chan-ss7/

-

The VoIP-Info Wiki

http://voip-info.org/wiki/

-

Asterisk Guru tutorials

http://www.asteriskguru.com/tutorials/ 8 Appendix

```
# First come the span definitions, in the format
# span=<span num>,<timing source>,<line build out>,<framing>,<coding>[,yellow]
#
# All T1/E1 spans generate a clock signal on their transmit side.
# The <timing source> parameter determines whether the clock
# signal from the far end of the T1/E1 is used as the master source
# of clock timing. If it is, our own clock will synchronise to it.
# T1/E1's connected directly or indirectly to a PSTN provider (telco)
# should generally be the first choice to sync to. The PSTN will
# never be a slave to you. You must be a slave to it.
#
# Choose 1 to make the equipment at the far end of the E1/T1 link
# the preferred source of the master clock. Choose 2 to make it
#the second choice for the master clock, if the first choice port
# fails (the far end dies, a cable breaks, or whatever).
# Choose 3 to make a port the third choice, and so on. If you have,
# say, 2 ports connected to the PSTN, mark those as 1 and 2.
# The number used for each port should be different.
#
# If you choose 0, the port will never be used as a source of timing.
# This is appropriate when you know the far end should always be
# a slave to you. If the port is connected to a channel bank,
# for example, you should always be its master. Any number of ports
# can be marked as 0.
#
# The line build-out (or LBO) is an integer, from the following table:
# 0: 0 db (CSU) / 0-133 feet (DSX-1)
# 1: 133-266 feet (DSX-1)
# 2: 266-399 feet (DSX-1)
# 3: 399-533 feet (DSX-1)
# 4: 533-655 feet (DSX-1)
# 5: -7.5db (CSU)
# 6: -15db (CSU)
# 7: -22.5db (CSU)
# The coding is one of "ami" or "b8zs" for T1 or "ami" or "hdb3" for E1
#
span=1,1,0,ccs,hdb3
#
# Next come the definitions for using the channels. The format is:
# <device>=<channel list>
# "clear" : Channel(s) are bundled into a single span. No conversion
# or signalling is performed, and raw data is available on the master.
# "indclear": Like "clear" except all channels are treated
# individually and are not bundled. "bchan" is an alias for this.
# The channel list is a comma-separated list of channels or ranges,
# for example:
# 1,3,5 (channels one, three, and five)
# 16-23, 29 (channels 16 through 23, as well as channel 29
#
bchan=1-31
```

{mos_fb_discuss:18}