FONEBridge2 Installation Guide

Version 1.0

May 2009

Objective

The present document is intended to explain in the greatest possible detail all steps required to install, configure and maintain FONEBridge2® devices.

Intended audience

It is assumed an intermediate knowledge of Asterisk, specially related to TDM Channels, Linux knowledge of basic tools is also required.

Software Covered in this guide:

All commands and instructions are based on latest version of `Fonulator`, for the present document is `Fonulator` V.2.0.1

How do I know `Fonulator` installed version?

Login onto server(s) running `fonulator` and type:

```
# fonulator –V
fonulator 2.0.1
Copyright (C) 2007 Redfone Communications, LLC.
Build Number: 37
```

The first line shows `fonulator´`s version, in this case is 2.0.1.

If version is prior 2.0.1 it is strongly recommended to upgrade from Redfone´s support site: http://support.red-fone.com/download/`fonulator`

1. FONEBridge2 Overview

Redfone is a hardware and software development Company that created and manufactures FONEBridge2®.

FONEBridge2 is a solid state device that converts TDM to TDMoE and is intended as a link between PSTN trunks and Asterisk platforms.

What makes Redfone unique is the fact of being a separate device, independent of server's hardware, making it a better choice for any kind of installations and the cheapest option to create distributed load and high availability systems.

Because FONEBridge2 integrates seamlessly with Asterisk TDM channels, it is viewed from Asterisk as a typical PCI board.

What is TDMoE?

TDMoE is a data transport protocol intended to send over Ethernet links telephony trunks, as much transparent as possible.

Redfone´s TDMoE

Main limitation in TDMoE is scalability. Due to the fact that TDMoE uses an Ethernet frame per each trunk, so having multiple trunks over same Ethernet segment creates enormous volumes of lan traffic, that can degradate call quality or generate lost of synchronization on Asterisk.

Redfone has developed and improved TDMoE driver (ethmf) that improves drastically the amount of traffic over Ethernet. Redfone's improved TDMoE driver packs four trunks on each Ethernet frame, reducing the total Ethernet traffic in a 1:8 factor.

Detailed information of this driver is in www.thrallingpenguin.com

Software compatibility

FoneBrisge2 tools and drivers are supported for any version of Asterisk starting at 1.2 up to the newest stable version. This means that drivers where adapted both for Zaptel and DAHDI channels.

Repositories and files

All required software needed to compile, run and configure FONEBridge2 is at:

http://support.red-fone.com/downloads root directory for downoalds

http://support.red-fone.com/downloads/dahdi dahdi stack patched with Redfone's improved ethmf driver

http://support.red-fone.com/downloads/zaptel zaptel stack with improved ethmf driver

http://support.red-fone.com/downloads/`fonulator` sources, packed distributions and config examples of `fonulator` tool.

2. Installing Software

This section explains how to configure and install all required software to run FONEBridge2 into an Asterisk platform.

Introduction

In order to connect to FONEBridge2, Asterisk's drivers must be compiled again, using the version provided by Redfone's that includes the ethmf.c driver.

Once compiled and installed, timing for Asterisk is derivated from FONEBridge2 devices, so if ztdummy is running on current installation it must be removed.

Although not required, it is recommended compile all Asterisk packages, as explained in commonly available Asterisk literature.

Checklist

1. Download zaptel (or DAHDI depending on current Asterisk version) tarball from http://support.red-fone.com/downloads
2. Download `fonulator` software and utilities from http://support.red-fone.com/downloads
3. Compile zaptel/DAHDI
4. Install Zaptel/DAHDI
5. Compile `fonulator` and `fonulator`_tools
6. Install `fonulator` and `fonulator`_tools
7. Check that all software is installed and running properly

Installation step-by-step

- Logging into Asterisk server as root (or sudo -).

- Select a place to store sources, usually is /usr/src

  – Download and uncompress Redfone software:

```
# wget http://support.red-fone.com/downloads/dahdi/dahdi-linux-redfone-2.1.0.4.tar.gz
# tar -xvf dahdi-linux-redfone-2.1.0.4.tar.gz
#
# wget http://support.red-fone.com/downloads/dahdi/dahdi-tools-redfone-2.0.0.tar.gz
# tar -xvf dahdi-tools-redfone-2.0.0.tar.gz
#
# wget http://support.red-fone.com/downloads/fonulator/fonulator-2.0.1.tar.gz
# tar -xvf fonulator-2.0.1.tar.gz
#
# wget http://support.red-fone.com/downloads/fonulator/libfb-2.0.0.tar.gz
# tar –xvf libfb-2.0.0.tar.gz
#
# wget http://support.red-fone.com/downloads/fonulator/redfone.conf
```

Assuming as base installation directory /usr/src a directory listing must show:

```
# ls –l
drwxr-xr-x  6 root    src         4096 2009-04-24 09:16 dahdi-linux-redfone-2.1.0.4
-rw-r--r--  1 root    src      1267962 2009-04-24 10:27 dahdi-linux-redfone-2.1.0.4.tar.gz
drwxrwxr-x  7 root    root        4096 2009-05-15 10:53 dahdi-tools-2.0.0
-rw-r--r--  1 root    src       428511 2009-02-03 10:19 dahdi-tools-redfone-2.0.0.tar.gz
drwxr-xr-x  4 root    root        4096 2009-05-15 13:54 fonulator-2.0.1
-rw-r--r--  1 root    src       468881 2009-05-14 17:13 fonulator-2.0.1.tar.gz
drwxr-xr-x  6 root    root        4096 2009-05-15 13:53 libfb-2.0.0
-rw-r--r--  1 root    src       923533 2008-06-13 16:10 libfb-2.0.0.tar.gz
-rw-r--r--  1 root    src          720 2009-01-27 17:53 redfone.conf
```

Compile drivers (in this example DAHDI-based, but it is the same for zaptel)

```
# cd dahdi-linux-redfone-2.1.0.4
# make
# make install
# cd ..
```

Compile DAHDI Tools

```
# dahdi-tools
# ./configure
# make
# make install
# cd ..
```

Copy configuration file example to /etc (where `fonulator` expects the config file is)

```
# cp /usr/src/redfone.conf  /etc/redfone.conf
```

If the installation is intended to connect several FONEBridge2 on same server, each `fonulator` will require a specific file, so a copy and rename must be done, for instance:

```
# cd /etc
# cp redfone.conf redfone_unitA.conf
# cp redfone.conf redfone_unitB.conf
```

Check installation

1- During compilation phases no error should ne issued on the process of ./configure, make and make install.

2- To check if `fonulator` is correctly installed type:

```
# fonulator -V
fonulator 2.0.1
Copyright (C) 2007 Redfone Communications, LLC.
Build Number: 37
```

At this point, there is likely no connectivity with FONEBridge2, issuing fonualtor without parameter will rise the error "Connection to device timed out! Check network or device power"

3. Connecting FONEBridge2 to server

Being installed all required software and checked that is running properly (as explained in previous section), next step consist in connect FONEBridge2 with server(s).

This section covers the needed step to connect the Ethernet or Data side of FONEBridge2 to an Asterisk server, section 4 explains TDM interconnection.

Introduction

FONEBridge2 has two Ethernet 100bT ports and 1, 2 or 4 trunk TDM ports depending on model; also, an echo cancellation optional board can be installed.

FoneBrgidge implements two levels of data communications:

- UDP/IP to exchange status and configuration files.
- MAC to MAC: to exchange TDM traffic using TDMoE improved protocol.

FONEBridge2 is delivered with two preprogrammed IPs that can be changed as many other FONEBridge2 parameters using `fonulator` tool.

Executing `fonulator` command with no parameters causes `fonulator` reads `/etc/redfone.conf` file and writes parameters to FONEBridge2 device(s), this traffic is accomplished using UDP/IP protocol.

Once configured, all TDM traffic in and out will flow using a raw, layer 2 transport, independent of any IP running on same LAN segment. This means that Data (IP-level) and TDM (layer-2 or MAC level) traffic can share same Ethernet FONEBridge2 port or be on separate ones.

Configuration is written onto FONEBridge2 device in volatile RAM, until a “write config” command is issued using `fonulator`. This causes actual configuration to being stored into device’s flash memory. Not issuing this command will cause that, on a device reset (for instance a power off situation) when device is up again, configuration is the default one, and requires issuing `fonulator` to reload configuration

Because FONEBridge2 has independent power source, it will continue working even in the case of server outgages, this means that, even with servers down or powered off, telco will see trunks connected without alarms. This kind of isolation is one of the distinctive FONEBridge2 features.

Software Stack

There are three layers of software to be configured before using FONEBridge2:

Layer (top-down)	Zaptel technolgy	DAHDI technology
Asterisk side of TDM interfaces	`/etc/asterisk/zapata.conf`	`/etc/asterisk/chan_dahdi.conf`
Driver side of TDM interfaces	`/etc/asterisk/zaptel.conf`	`/etc/dahdi/system.conf` `/etc/dahdi/modules.conf`
Device-level configuration	`/etc/redfone.conf`	`/etc/redfone.conf`

The following steps are shown in bottom-up order.

Using `fonulator`

`Fonulator` provide command line access to FONEBridge2 devices. For an "out of the box" installation, invoking the command "`fonulator`" causes that redfone.conf is read and written into FONEBridge2 device (volatile memory only).

Albeit simple, `fonulator` allows change device behavior in any possible way, the easiest way to get the up to date list of available commands is:

```
# fonulator --help
usage: fonulator [-hvqsVgR] [--write-config] [--reset-defaults] [--upload=<file>]
[FILE] [--set-ip=x.x.x.x] [--fb2]

        -h, -H, --help           this help information
        -v, --verbose            verbose output
        -q, --query              query FONEBridge2 to check availability
        -s, --stats              query FONEBridge2 link statistics
        -V, --version            get version information
        --write-config           write configuration to the FONEBridge2
        --reset-defaults         reset default configuration on the FONEBridge2
        --upload=<file>          upload new firmware image
        -g, --gpak               specify that the uploaded firmware is a GPAK binary
        -R, --reboot             reboot the FONEBridge2
        FILE                     config file (default: /etc/redfone.conf)
        --set-ip=x.x.x.x         set new ip
        --fb2                    specify that ip to be changed is fb2
```

`-v` Makes command execution more explicit, the command `fonulator` –v causes the default configuration file is written into volatile memory.

`-q` Query FONEBridge2 and shows current FONEBridge2 firmware version, line configuration, MAC and IPs. Only FB1 port´s MAC is shown, the second one (FB2) is always FB1+1.

`-s` Retrieves information from FONEBridge2 about span configurations (line encoding, framing), MAC address, IPs, firmware version.

`-V` Shows `fonulator`'s current version and build.

`--write-config` Stores current configuration into non-volatile memory.

`--reset-defaults` Restores factory configuration, including IP ports addresses, so if alive configuration is in different networks than 192.168.1.254 and .255 connection to FONEBridge2 will be lost.

`-R` Reboot FONEBridge2, returning device to last saved configuration. If no **`write-config`** command was issues, it will reset to factory defaults.

`FILE` Specifies configuration file other than default **`redfone.conf`**. Specially usefull when having more than one device attached to same server. Without **`FILE`** parameter, command **`fonulator`** equals to **`fonulator redfone.conf`**.

`--set-ip=x.x.x.x` set new IP on FB1 port

`--fb2` meaningful only with –set-ip, changes FB2 IP instead of default FB1, for instance: `fonulator -set-ip=10.10.10.150 -fb2`

To access devices in a multi-device installation, a separate .conf file must be created and specified when running `fonulator`, specifying the wrong config file can change parameters on the wrong device.

Using the example given in previous section

```
# fonulator redfone_unitA.conf
# fonulator redfone_unitB.conf
```

The `redfone.conf` file

This file contains all required information to set FONEBridge2's parameters, the example file has all possible definitions.

For the extent of this section (data side) it is shown only the section required to connect to FONEBridge2 over Ethernet link.

```
# cat /etc/redfone.conf

  [globals]
  # IP-address of the IP Configuration port
  # Factory defaults are; FB1=192.168.1.254 FB2=192.168.1.253
  fb=192.168.1.254
  # Which port to use for TDMoE Traffic (1 or 2)
  port=1
  # Which Asterisk server destination MAC address for TDMoE Traffic?
  server=00:11:22:33:44:55
  # .......line encoding parameters listed here ..............
```

Checking connectivity

Having configured `/etc/redfone.conf` , both server and FONEBridge2 are able to talk each other, this is the next checkpoint for installation.

Checking connectivity is simple, it requires trying to load FONEBridge2 configuration,as shown:

```
# fonulator -v
Detecting FONEBridge2
DSP Status: Bypassed
Detecting current FONEBridge2 link configuration
Stopping FONEBridge2 TDMoE transmission
WPLL Enabled
Line configurations differ for link 3
Line configurations differ for link 4
Updating FONEBridge2 link configuration
Starting FONEBridge2 TDMoE transmission
FONEBridge2 reconfigured!
```

The first line after the command is issued shows the intent of `fonulator` to contact FONEBridge2, if everything was properly configured it will show the previous message (with actual configuration).

Driver configuration

Next step to link FONEBridge2 and server is the configuration of Asterisk drivers, in this example is used DAHDI on a Debian Etch installation.

Driver-level configuration steps includes:

1. Configure /etc/dahdi/system.conf
2. Configure /etc/dahdi/modules
3. Load driver
4. Check installation

The `/etc/dahdi/system.conf` file

For the scope of this guide, only FONEBridge2 related configurations will be shown, check Asterisk documentation if other devices must be configured.

```
# cat /etc/dahdi/system.conf
#
# ……. other configuration lines out of scope ……
# Dynamic Spans
# ^^^^^^^^^^^^^
# Next come the dynamic span definitions, in the form:
#
#   dynamic=<driver>,<address>,<numchans>,<timing>
#
# Where <driver> is the name of the driver (e.g. eth), <address> is the
# driver specific address (like a MAC for eth), <numchans> is the number
# of channels, and <timing> is a timing priority, like for a normal span.
# use "0" to not use this as a timing source, or prioritize them as
# primary, secondard, etc.  Note that you MUST have a REAL DAHDI device
# if you are not using external timing.
#
#   dynamic=eth,eth0/00:02:b3:35:43:9c,24,0
#
# If a non-zero timing value is used, as above, only the last span should
# have the non-zero value.
#
dynamic=ethmf,eth1/00:50:c2:65:d6:64/0,31,0
dynamic=ethmf,eth1/00:50:c2:65:d6:64/1,31,1
bchan=1-15
bchan=17-30
dchan=16
bchan=31-45
bchan=47-60
dchan=46
```

This example shows the configuration of a FONEBridge2 device with 2xE1s.

First, definition of dynamic spans:

```
dynamic=ethmf,eth1/00:50:c2:65:d6:64/0,31,0
dynamic=ethmf,eth1/00:50:c2:65:d6:64/1,31,1
```

Note that driver name is ethmf because FONEBridge2 expects this driver to exchange TDM traffic.

Eth1 is the interface name where TDM traffic will be exchanged

MAC corresponds to FONEBridge2 MAC

To define more than one span, a subaddress notation is used, /0, /1 etc.

Number of channels corresponds to channels associated to line type (E1, T1, etc) and encoding.

Timing source: this is a non-trivial parameter. It differs from legacy TDMoE driver. Because each FONEBridge2 encapsulates all channels information into only one Ethernet frame, driver must be informed about the last expected span into each frame, this span will be marked with a non-zero value, for a 4xE1 device:

```
# cat /etc/dahdi/system.conf
#
# …… other configuration lines out of scope ……
dynamic=ethmf,eth1/00:50:c2:65:d6:64/0,31,0
dynamic=ethmf,eth1/00:50:c2:65:d6:64/1,31,0
dynamic=ethmf,eth1/00:50:c2:65:d6:64/2,31,0
dynamic=ethmf,eth1/00:50:c2:65:d6:64/3,31,1
# …… other configuration lines out of scope ……
```

In the case of two FONEBridge2 boxes a suggested configuration should be:

```
# cat /etc/dahdi/system.conf
#
# …… other configuration lines out of scope ……
dynamic=ethmf,eth1/00:50:c2:65:d6:80/0,31,0
dynamic=ethmf,eth1/00:50:c2:65:d6:80/1,31,2
dynamic=ethmf,eth1/00:50:c2:65:d6:64/0,31,0
dynamic=ethmf,eth1/00:50:c2:65:d6:64/1,31,1
# …… other configuration lines out of scope ……
```

Some remarks:

1- MAC addresses corresponds to each FONEBridge2 device.

2- Timing mark must be non zero for each connected FONEBridge2, but with increasing integer, meaning that as a master clock it will selected the “1” driver and if this fails, it will be used the “2” and so on.

3- Virtual span must start on /0 for each FONEBridge2, be careful with this issue, because a wrong numbering will cause kernel panic on TDMoE initialization.

Channels definition is the usual definition of bearer (bchan) and signaling (dchan) time slots, and , greatly depends on other side’s configuration, usually a telco (information must be gathered from telco technical staff).

The `/etc/dahdi/modules` file

For the scope of this guide, only FONEBridge2 related configurations will be shown, check Asterisk documentation if other modules are configured.

```
# cat /etc/dahdi/modules
#
# …… other configuration lines out of scope ……
#
# Refone's FONEBridge2 TDM dybamis drivers
dahdi_dynamic_ethmf
dahdi_dynamic
```

Those two modules must be loaded by dahdi in order to be able to communicate with FONEBridge2

Loading driver

In order to test driver installation and checking TDMoE dialog with FONEBridge2, the following command will load dahdi driver:

```
# modprob dahdi
#
# lsmod | grep dahdi
dahdi_dynamic_ethmf    11460  2
dahdi_dynamic           9808  1 dahdi_dynamic_ethmf
dahdi                 183688  1 dahdi_dynamic
crc_ccitt               2304  2 dahdi,irda
```

Loading dahdi drivers from startup is configured when `make config` is issues during dahdi-tools compilation and installation process.

Be careful with Dahdi_dummy !

`Dahdi_dummy` is required to generate Asterisk timebase when no TDM hardware is installed, using TDMoE is an exception that must be checked for proper function.
If command `lsmod|grep dahdi` shows dahdi_dummy running, it must be removed because ethmf driver will provide time base to Asterisk.

On command line: `rmmod dahdi_dummy`

Startup scripts will load dahdi_dummy because it do not check if dynamic drivers are configured, incorrectly assuming that dahdi_dummy must be installed. The startup script hence must be changed as shown below (for a Debian Etch installation):

```
# cat /etc/init.d/dahdi

……. Lines removed ……..

  while [ ! -d /dev/dahdi ] ; do
      sleep 1
      TMOUT=`expr $TMOUT - 1`
      if [ $TMOUT -eq 0 ] ; then
         echo "Error: missing /dev/dahdi!"
```

```
          exit 1
       fi
       done

      xpp_startup

      if [ -e /proc/dahdi/dynamic-ethmf ]; then
         echo "Dynamic driver encountered, skipping dahdi_dummy"
      elif [ ! -e /proc/dahdi/1 ]; then
         echo "No hardware timing source found in /proc/dahdi,loading dahdi_dummy"
         modprobe dahdi_dummy 2> /dev/null
      fi

…… lines removed …….
```

Checking TDMoE connectivity

Having dahdi successfully loaded, there is a simple check to see if TDMoE traffic is flowing, at this point there are already TDMoE traffic on both sides:

```
# tcpdump -I eth1 ethernet proto 0xd00d
09:31:22.952139 00:40:63:d7:bc:e3 (oui Unknown) > 00:50:c2:65:d6:64 (oui Unknown),
ethertype Unknown (0xd00d), length 572:
        0x0000:  8002 0802 8628 001f 0802 8618 001f bbbb  .....(..........
        0x0010:  bbbb bbbb bbbb bbbb bbbb bbbb 0bbb bbbb  ................
        0x0020:  bbbb bbbb bbbb bbbb bbbb bbbb 0bbb ffff  ................
        0x0030:  ffff ffff ffff ffff ffff ffff ffff ffff  ................
        0x0040:  ffff ffff ffff ffff ffff ffff ffff ffff  ................
        0x0050:  ffff                                     ..
09:31:22.952843 00:50:c2:65:d6:64 (oui Unknown) > 00:40:63:d7:bc:e3 (oui Unknown),
ethertype Unknown (0xd00d), length 572:
        0x0000:  8002 0802 647b 001f 0802 647b 001f 0000  ....d{....d{....
        0x0010:  0000 0000 0000 0000 0000 0000 0000 0000  ................
        0x0020:  0000 0000 0000 0000 0000 0000 0000 ffff  ................
        0x0030:  ffff ffff ffff ffff ffff ffff ffff ffff  ................
        0x0040:  ffff ffff ffff ffff ffff ffff ffff ffff  ................
        0x0050:  ffff                                     ..
.
.
.
.
70 packets captured
420 packets received by filter
258 packets dropped by kernel
```

This command will cause the screen floods with captured packets, so hit CTRL-C immediately, the basic check to be done is that MAC addresses of traced packets swaps between receiving and transmitting ones, this means that TDMoE traffic (defined as protocol 0xd00d) is established in both senses.

Checking Driver Setup

For this test, dahdi_tool must be used.

```
# dahdi_tool

  |     Alarms          Span                                                  |
  |                                                                           |
  |     OK              Dynamic 'ethmf' span at 'eth1/00:50:c2:               |
  |     OK              Dynamic 'ethmf' span at 'eth1/00:50:c2:               |
  |                                                                           |
```

This ensures proper installation and configuration at driver (dahdi/zaptel) level.

Alarms types:

Red Alarm
Your T1/E1 port will go into red alarm when it cannot maintain synchronization with the remote switch. A red alarm typically indicates either a physical wiring problem, loss of connectivity, or a framing and/or line-coding mismatch with the remote switch. When your T1/E1 port loses sync, it will transmit a yellow alarm to the remote switch to indicate that it's having a problem receiving signal from the remote switch.

Yellow Alarm
Your T1/E1 port will go into yellow alarm when it receives a signal from the remote switch that the port on that remote switch is in red alarm. This essentially means that the remote switch is not able to maintain sync with you, or is not receiving your transmission.

Blue Alarm
Your T1/E1 port will go into blue alarm when it receives all unframed 1s on all timeslots from the remote switch. This is a special signal to indicate that the remote switch is having problems with its upstream connection. dahdi_tool and Asterisk don't correctly indicate a blue alarm at this time.

Channel allocation

One of the most common errors is due to incorrect channel allocation among driver-level configuration (/etc/system.conf or /etc/asterisk/zaptel.conf) and channel-level configuration (/etc/asterisk/chan_dahdi or /etc/asterisk/zapata.conf).

The following is a quick reference to match correctly both configuration files

/etc/asterisk/chan_dahdi.conf	/etc/dahdi/system.conf
group = 1	
channel => 1-15	bchan = 1-15
channel => 17-31	bchan = 17-31
	dchan = 16
channel => 32-46	bchan = 32-46
channel => 48-62	bchan = 48-62
	dchan = 47

This example shows two E1s configured for group 1, dotted lines marks the lines that must match between both files.

Network considerations

There are no special considerations on how FONEBridge2 **data** traffic flows on the LAN because usage is too small and does not risk other traffic. Nevertheless, TDMoE traffic must be considered as LAN-resource demanding and hence must be kept separated from data traffic.

TDMoE transport is accomplished at MAC (or layer-2) level, not involving IP on any matter, for multi-trunk installations the amount of TDMoE frames transmitted on both sizes can be important, even using the improved ethmf driver. Due to Asterisk design, it expects an external time base for processing media. This timing source is provided for TDM cards or ztdummy in the case of no-TDM installations. In this aspect, FONEBridge2 acts like a legacy PCI card, providing the time source for Asterisk. The rate of timing is 1000 interrupts per second on each direction, per each FONEBridge2 device attached, meaning a total of 2,000 ethernet packets per second. This is the reason that Redfone recommends to allocate a dedicated LAN segment for TDMoE traffic.

Another consideration to keep LAN segment isolated is that, when the target server gets disconnected (for any reason) to the TDMoE LAN segment, FONEBridge2 keeps sending TDMoE traffic at 1,000 frames per second. Some switches broadcast packets on all LAN ports trying to find the destination MAC, generating a packet storm at 1000 packets per second.