Conferencing with Sangoma Wanpipe

Building a Simple, Cost-Effective Conferencing Server Using Sangoma Wanpipe and YATE (Yet Another Telephony Engine)

September 18, 2008

Prepared By
Jim Van Meggelen
Core Telecom Innovations Inc.

CONFERENCING WITH SANGOMA WANPIPE 1

BUILDING A SIMPLE, COST-EFFECTIVE CONFERENCING SERVER USING SANGOMA WANPIPE AND YATE (YET ANOTHER TELEPHONY ENGINE) 1

EXECUTIVE SUMMARY 3

YET ANOTHER TELEPHONY ENGINE 3

WHY YATE? 3

YATE'S DESIGN PHILOSOPHY 3
Stability 3
Performance 3
Flexibility 4

PUTTING YATE TO WORK FOR YOU 4

STAND-ALONE CONFERENCING SERVER 4
EMBEDDED CONFERENCING ENGINE IN AN EXISTING ASTERISK INSTALLATION 4

INSTALLATION OF WANPIPE FOR YATE 5

HARDWARE REQUIREMENTS 5
LINUX REQUIREMENTS 5
DOWNLOADING THE DRIVERS 5
DIVIDING YOUR T1/E1 CARD IN TWO 7
D-Channel 7
B-Channels 7
Configuring Wanpipe 8
STARTING WANPIPE 13

INSTALLING YATE 14

DOWNLOADING YATE 14
INSTALLING YATE 14
RUNNING YATE AT STARTUP 14

CONFIGURING YATE AS A CONFERENCING SERVER 15

YSIGCHAN.CONF 15
WPCARD.CONF 15
REGEXROUTE.CONF 16
USING RMANAGER FOR A YATE CONSOLE 17

SOME THINGS TO TRY 18

CALL GENERATOR 18
INSTALLING THE YATE CLIENT ON YOUR PC 19

CONCLUSION 19

APPENDIX: SOME YATE MESSAGE TYPES 20

CHAN.STARTUP 20
CALL.PREROUTE 20
CALL.ROUTE 20
CALL.EXECUTE 20
CALL.RINGING 20
CHAN.DTMF 20
CALL.CDR 21
USER.AUTH 21
USER.REGISTER 21

GLOSSARY 22

Executive Summary

Yet Another Telephony Engine

YATE is a next-generation, open-source telephony engine that deserves your attention. With more than four years of development behind it, this mature telephony engine offers powerful features and capabilities.

In YATE, each channel is an object. All channel objects are built from the same base class, with channel-specific parameters added as required.

In this white paper we will give you an introduction to YATE and provide you with step-by-step instructions on how to get YATE up and running.

Why YATE?

YATE was designed from the ground up to be carrier-grade. For carriers and large operators, stability and performance take priority over features. While YATE has more than its fair share of features, at its heart it is a switching engine, and as such, it can be trusted with tasks that would overwhelm lesser software.

YATE's Design Philosophy

At the heart of the decision to develop YATE was a desire to ensure that external control of the engine would be logical and simple and would not compromise stability.

YATE allows you to change parameters such as the called party, the caller, privacy options, and protocol settings on the fly.

Stability

The YATE team is passionate about stability and about the importance of testing for any new release.

Stability in YATE comes in part because of the belief that if you design software correctly, you don't need to change it as often. Software that doesn't change has its bugs sorted out early on, and stability is the reward.

YATE's core doesn't have to change much because it was designed in such a way that new features can be provided through the creation of external modules. New modules may have bugs, but the core is protected; the core engine therefore remains stable.

Performance

Because the core of YATE is small and lean, the engine can fearlessly take on any software telephony engine in a performance contest. In our testing, using an Intel Core 2 Duo CPU with 1 Gig of RAM, we were able to mix 200 separate audio streams in a conference room with full duplex audio, and an average CPU

load of less than 75%. This kind of performance allows a single YATE-based server to handle large numbers of connections. Recent testing has shown that 1400 SIP connections can easily be switched through the engine with full audio.

Flexibility

YATE is wonderfully flexible. Its message-passing architecture means that modules can be created to handle all sorts of tasks. As an example, YATE can support multiple routing engines running at the same time. Each routing engine can be responsible for responding to different requests, and if there is a conflict (such as two engines responding to a routing request for the same call), YATE's priority system will ensure that only one module responds to the request.

Putting YATE to Work for You

YATE can be used for everything from massive PBX installations serving thousands of users to a simple softphone client running on your PC. For the purposes of this white paper, we will focus on using YATE to deliver a simple application that will showcase its simplicity, flexibility, and power.

To illustrate the use of YATE, we will show you how to build a conferencing server. For the sake of brevity, the server will be fairly simple. It is important to note, however, that YATE is more than capable of handling much more complex requirements, such as scheduling, database control, web-based GUIs, automated callout to clients, and more.

Stand-Alone Conferencing Server

The conferencing server we are going to build will be designed to connect either directly to the PSTN, or to a PBX via the PRI protocol. We will walk you through the process of downloading, compiling, and installing the Sangoma T1 card, then show you how to download, install, and configure YATE to utilize the card and provide conferencing.

Embedded Conferencing Engine in an Existing Asterisk Installation

The process that we discuss here could just as easily be used to embed YATE in an existing Linux installation (even one that is running Asterisk). The only difference is that instead of using PRI to provide the incoming connections, you would use SIP instead. At the end of the PRI examples, we will provide some tips on how to configure YATE as a SIP-based conferencing server within an existing Asterisk installation.

Installation of Wanpipe for YATE

Before installing YATE, you must configure your Sangoma card.

> Please note that from this point you will need to have some familiarity with command-line operations in Linux. If you are comfortable with the basics of the Linux shell and with basic text editing, you should be fine.

Hardware Requirements

You will need to have a Sangoma AFT card with at least one T1/E1 port, and a Linux-compatible computer.

Linux Requirements

You should be running a server-based version of Linux, and ideally this server should not be running the X windowing interface or any other Linux GUI. You will also need to have the Linux kernel sources and a basic set of development tools installed. On a CentOS server install, the following commands should ensure that you have the correct packages:

Having said that, by far the easiest way to ensure that your CentOS server has the packages you need is to install the Development Packages and Development Tools during the Linux install process.

Downloading the Drivers

The latest drivers for your Sangoma card can be found at:
http://wiki.sangoma.com/wanpipe-linux-drivers

Download the latest version of the Stable drivers, which is currently 3.2.x. You must install the drivers as the root user, so make sure you can run commands as root before you proceed. We always download drivers to the `/usr/src` folder, so unless you have a different preference, you can issue the following commands from the shell to obtain the drivers:[1]

```
# cd /usr/src
# wget ftp://ftp.sangoma.com/linux/current_wanpipe/wanpipe-3.2.7.1.tgz
```

If the drivers do not download, check the URL to make sure it is correct and working.[2]

Once the download is complete, you can extract the drivers with the following command:

```
# tar zxvf wanpipe-3.2.7.1.tgz
```

---

[1] YATE is tested on the CentOS distribution of Linux, and that is the distro that was used in the making of this white paper. You are free to use whatever flavour of Linux `you prefer`, but if you're using a different distribution the instructions given here may not be totally compatible with your system.

[2] Please note that 3.2.7.1 was the current version as of this writing. You should substitute the latest stable driver version for the one in this command and the commands that follow.

This will create a folder like `/usr/src/wanpipe-3.2.7`. Issue the following commands to begin your installation:[3]

```
# cd wanpipe-3.2.7
# ./Setup install
```

The interface will verify that all required tools are installed and ask you if you want to install Wanpipe now. Clearly, the answer is '**Y**' (Yes). Next, you will be asked if you would like to build the WANPIPE kernel driver modules. Answer '**Y**' again.

You will then be asked to specify the absolute pathname of your Linux directory. You can press **Enter** to accept the default. If this does not work for you, contact Sangoma support for further assistance.

You will again be asked to confirm that you wish to build the kernel drivers. Unless you've changed your mind about installing this card, answer **'Y'**.

Now comes the important question: the Setup program wants to know what compilation mode you wish to use. If you are used to setting up Sangoma cards for Asterisk, here is where things start to diverge. You'll want to select the option for the TDM API, which in the case of wanpipe-3.2.7 is option 6. Look for the option that specifically mentions YATE, and you should be OK. Again, if you have any trouble with this step, contact Sangoma support.

You will have to press **Enter** a few more times, and then you will see the message *`Compiling General WANPIPE Driver for 2.6.X Kernel`*. Depending on the speed of your system, compilation will take anywhere from a few seconds to several minutes. At the end, the Setup program will ask you to 'visually confirm that driver compilation was successful.' Take the time to make sure that there were no errors, and if all looks well, press the **'Y'** key one more time.

Press **Enter** at the next few prompts, and the Setup program will compile some of the utilities and libraries that will be used later to configure and run your card.

Again, you'll be asked to confirm that the compilation sequence was successful. Check for errors and then press **Enter**, and you will come to the final question*:* ***Would you like to install WANPIPE start-up scripts?*** Answer **'Y'** one last time, and your Sangoma drivers are compiled.

[3] If you have used a program such as `yum update` to ensure that your Linux system is running the latest patches and kernel version, you should reboot the system before continuing or your installation will run into trouble.

Dividing Your T1/E1 Card in Two

Once you've compiled the drivers, utilities, and libraries, it's time to configure your card so that YATE can use it.

YATE handles most of the signaling over the PRI connection. This means that when you configure the Sangoma Wanpipe card, you need to set it up to allow YATE to manage the PRI connection.

PRI circuits consist of two elements: the D-channel (delta channel), which carries the signaling; and the B-channels (bearer channels), which carry the actual phone calls. When setting up Wanpipe, you have to configure two separate timeslot groups for each PRI port: one for the D-channel, and one for all of the B-channels.

In the early days of the PRI protocol, the D-channel was carried on a completely different circuit from the B-channels. A full T1 was allocated to the B-channels, and the D-channel was run over a physically separate 64K connection (often just a serial link). Later, technology was developed to use one of the timeslots of the T1 to carry the D-channel. The important thing to note is that the D-channel is a totally different type of channel from the B-channels; it just happens to run over the same T1 for convenience's sake. The ISDN protocol does not require this, so when you are configuring PRI circuits, you will generally have to define where the D-channel is in relation to the B-channels.

D-Channel

YATE takes advantage of the HDLC engine that is built into Wanpipe, but in order for it to do so; one of the timeslots on the card must be configured appropriately. In North America, channel 24 of the T1 is generally used for the D-channel. In the rest of the world, channel 16 of the E1 circuit services the D-channel.

B-Channels

The bearer channels carry the audio portion of each telephone call. In North America, channels 1-23 are used for this purpose. In the rest of the world, channels 1-15 and 17-31 are designated as B-channels.[4]

[4] In an E1 circuit, channel 32 is not used as a B-channel.

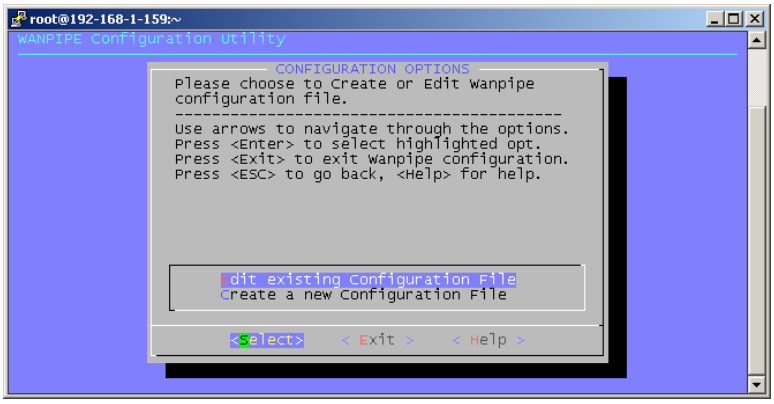
Configuring Wanpipe®

Follow the procedure outlined here to configure Wanpipe into two timeslot groups for YATE.

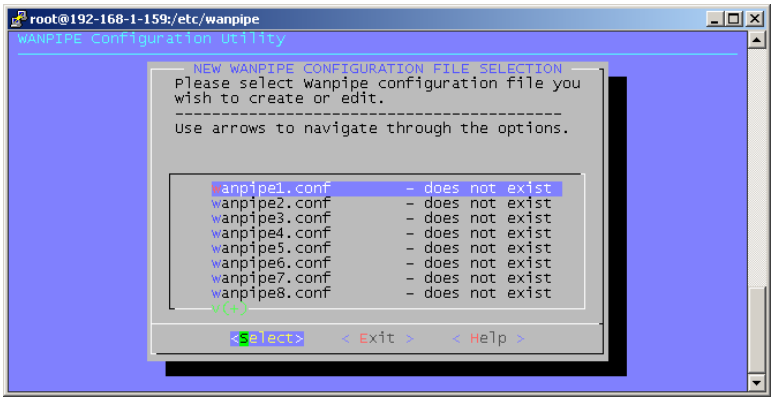
First, as the root user, issue the following command:

```
# wancfg
```

This will start the WANPIPE Configuration Utility, which will build the Wanpipe configuration files. Press **OK** at the first screen, and you should see something like this:



Select **Create a new Configuration File** and press **Enter**.
You will see a list of the configuration files that are possible, and their states. For a new installation, you should see this:

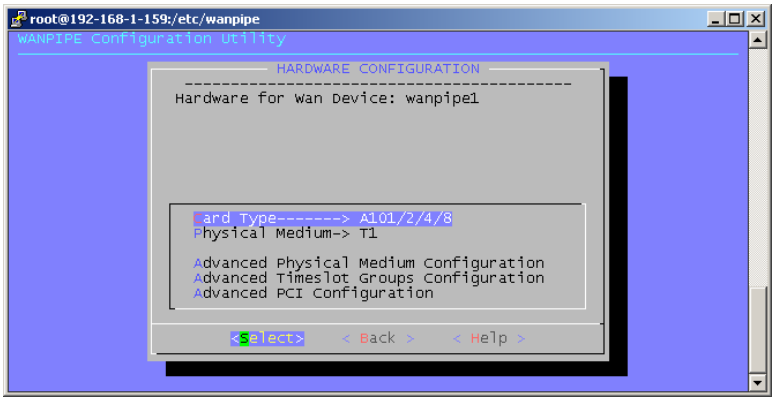


Press **Enter** to select **wanpipe1.conf**.

Select the following from the list of detected cards:

```
AFT-A101-SH SLOT=7 BUS=0 IRQ=5 CPU=A PORT=1 HWEC=32 V=34
```

Next, you will see this screen:

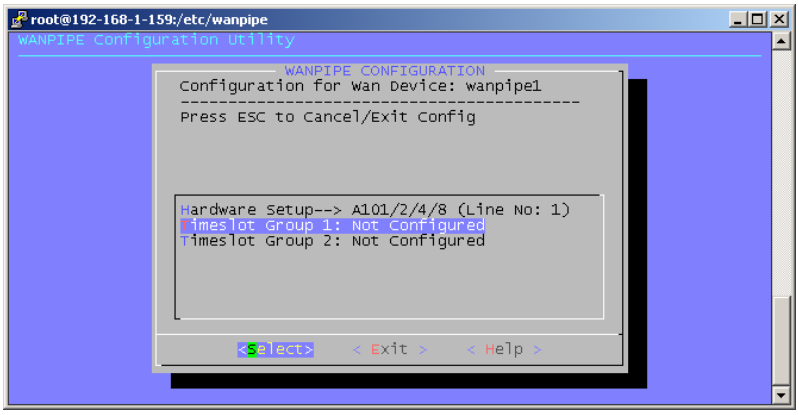


If you are not in North America, you will probably need to change the physical medium from **T1** to **E1** using the first submenu.

Next, you'll have to configure the timeslot groups using the second submenu. You will need to define two timeslot groups: one for the D-channel, and the other for all the B-channels.

World (E1)	**North America (T1)**
Timeslot Group 1 Configuration Timeslots in Group-> ALL HDLC engine--------> Enabled Idle char ---------> 0x7E MTU ---------------> 1500 MRU ---------------> 1500 **Timeslot Group 2 Configuration** Timeslots in Group-> 1-15.17-30 HDLC engine--------> Disabled Idle char ---------> 0x7E MTU ---------------> 1500 MRU ---------------> 1500	**Timeslot Group 1 Configuration**-> 24 Timeslots in Group-> ALL HDLC engine--------> Enabled Idle char ---------> 0x7E MTU ---------------> 1500 MRU ---------------> 1500 **Timeslot Group 2 Configuration**-> 1-23 Timeslots in Group-> ALL HDLC engine--------> Disabled Idle char ---------> 0x7E MTU ---------------> 1472 MRU ---------------> 1472

Once you have defined the timeslots, you should see this screen:



You'll need to tell Wanpipe what each timeslot will be used for. The timeslots will be referred to as **`w1g1`** and **`w1g2`** (Wanpipe 1 group 1 and Wanpipe 1 group 2). If you have a multiport Wanpipe card, you may have other group names as well.

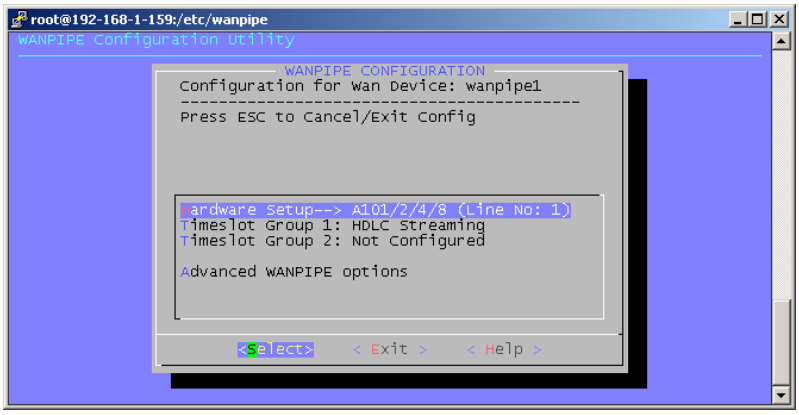
For your current purposes, you'll want to be running HDLC streaming on each timeslot, with an operation mode of API. The timeslot group config screens are a little confusing to navigate, but with a bit of back-and-forth you should be able to figure them out. For each timeslot group, set the following:

`Protocol:` **`HDLC Streaming`**
`Interface Setup--> 1 defined` (select and press **Enter**)
`Interface 1 - w1g1` (select and press **Enter**)
`Operation Mode--> WANPIPE` (select and press **Enter**)

You want to change this to:

`Interface Operation Mode:` **`API`**

Press **Exit/Back** until you return to this screen:



Then configure Timeslot Group 2 in the exact same way:

Protocol: **HDLC Streaming**

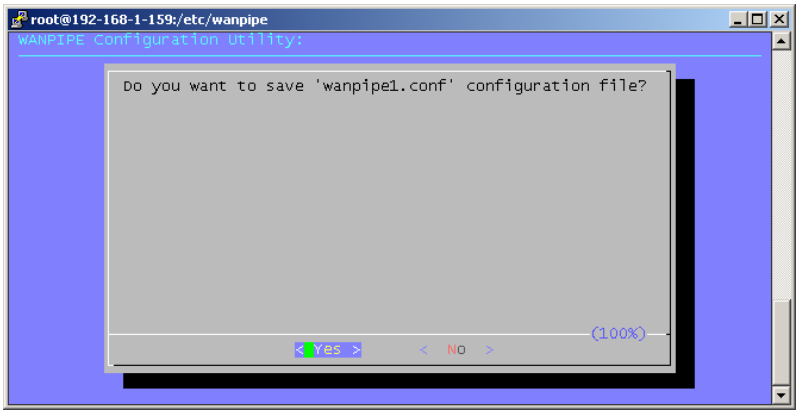
Interface Setup--> 1 defined (select and press **Enter**)

Interface 1 - w1g2 (select and press **Enter**)

Operation Mode --- WANPIPE (select and press **Enter**)

Interface Operation Mode: **API**

Press **Exit** until you get to this screen:



Success! Save the file, and your Wanpipe is ready to go.

When you're all done, you should have the following in your */etc/wanpipe/wanpipe1.conf* file:

World (E1)	**North America (T1)**
[w1g1] HDLC_STREAMING = YES ACTIVE_CH = 16 MTU = 1500 MRU = 1500 DATA_MUX = NO TDMV_HWEC = NO [w1g2] HDLC_STREAMING = NO ACTIVE_CH = 1-15.17-30 IDLE_FLAG = 0x7E MTU = 1500 MRU = 1500 DATA_MUX = NO TDMV_HWEC = YES	[w1g1] HDLC_STREAMING = YES ACTIVE_CH = 24 MTU = 1500 MRU = 1500 DATA_MUX = NO TDMV_HWEC = NO [w1g2] HDLC_STREAMING = NO ACTIVE_CH = 1-23 IDLE_FLAG = 0x7E MTU = 1472 MRU = 1472 DATA_MUX = NO TDMV_HWEC = YES

You can edit this file manually as well (without using the ***wancfg*** program), but this is not recommended by Sangoma.

Starting Wanpipe

If all went well during the *wancfg* process, Wanpipe should start automatically as soon as your system boots. You can also start it manually with the following command:

```
# wanrouter start
```

Watch for errors during this process, as they will provide you with information on how to resolve any problems that occur. If you need to make changes, don't forget to run wanrouter stop before you edit the configuration files.

If your Wanpipe is properly configured and you have a successful T1 connection to the other end (the PBX or CO that is connected to your new conferencing server), this command:

```
# wanrouter status
```

should show output similar to the following:

```
Devices currently active:
        wanpipe1
Wanpipe Config:
Device name | Protocol Map | Adapter  | IRQ | Slot/IO | If's | CLK | Baud rate |
wanpipe1    | N/A          | A101/1D/A102/2D/4/4D/8| 58  | 2       | 2    | EXT
| 0    |
Wanrouter Status:
Device name | Protocol | Station | Status        |
wanpipe1    | AFT HDLC | N/A     | Connected     |
```

The key is the Status, which should be `'Connected'`. This does not mean your PRI circuit is up, but it does mean that the T1/E1 circuit that will carry your PRI is connected, which is what you're looking for at this stage. If you have any Status other than `'Connected',` you should probably resolve this before you continue.[5] On the PBX or CO side, a problem with the T1/E1 connection will be represented by either a 'Red' or 'Yellow' alarm. These alarms need to be corrected (although please note that PRI alarms are still to be expected at this point, since you have not configured the PRI circuit yet).

[5] Please contact Sangoma support if you require any assistance with this process.

Installing YATE

YATE is simple to install. You can even run it on a system that is already running Asterisk.

We used the CentOS6 distribution of Linux for our YATE installation, so the following instructions are based on that distribution. If you are using a different version of Linux (especially one where the YATE RPM does not work), you may need to compile and install YATE from the sources.

Downloading YATE

Issue the following commands from the shell to obtain the YATE package from http://yate.null.ro:[7]

```
# cd /usr/src
# wget http://yate.null.ro/tarballs/yate2/mdv2008.0/i586/yate-2.0.0-1mdv2008.0.i586.rpm
```

Installing YATE

After downloading the RPM, you can install it with this command:

```
# rpm -Uhv yate-2.0.0-1mdv2008.0.i586.rpm
```

Running YATE at Startup

If you wish to have YATE start each time you boot, the following command will set it to start at normal boot:

```
# chkconfig --level 345 yate on
```

[6] CentOS is binary compatible with RedHat Enterprise Linux which is closely related to Fedora Linux. If you have worked with any RedHat-style Linux distribution, you should find the very popular CentOS distro familiar.
[7] Note that the version number specified in the wget command may not be the most current release. Check the YATE project's downloads page to make sure you are getting the most current version.

Configuring YATE as a Conferencing Server

The basics of running YATE as a conferencing server are simple. We'll go through this exercise so that you can quickly get a feel for what YATE sounds like in production. You can then expand the capabilities of your conferencing server to provide more services to your users.

The yate RPM installs the configuration files for YATE in the folder `/usr/local/etc/yate`. The files that you'll find there are full of useful information on how to configure various features of YATE. This section will walk you through the changes you'll need to make to those files for the current purposes.

ysigchan.conf

The signalling (sig) channel in YATE is where you configure interactions with traditional PSTN services. In this case, you need to configure the sig channel for PRI.

Edit the ysigchan.conf file to match the following:

World (E1)	**North America (T1)**
`[general]`	`[general]`
`[link1]`	`[link1]`
`type=isdn-pri-net`	`type=isdn-pri-net`
`enable=yes`	`enable=yes`
`sig=wanpipe1`	`sig=wanpipe1`
`voice=wanpipe1`	`voice=wanpipe1`
`switchtype=euro-isdn-e1`	`switchtype=national-isdn`
`strategy=increment`	`strategy=lowest`
`format=alaw`	`format=mulaw`

You can edit the sample file and simply change fields as needed, or make a backup of the sample file and create a new file with these contents. Keep a copy of the sample file with all the comments somewhere, as this file contains useful information about how to define signalling channel parameters.

wpcard.conf[8]

The sig channel in YATE needs to know where to find the hardware to which it's connecting. In the case of Sangoma hardware, the wpcard.conf file contains the information the sig channel needs to communicate with the Sangoma Wanpipe card.

[8] There is a similar file named *wpchan.conf*; make sure you edit the correct one.

World (E1)	North America (T1)
[general] [wanpipe1] type=E1 siggroup=w1g1 voicegroup=w1g2	[general] [wanpipe1] type=T1 siggroup=w1g1 voicegroup=w1g2

There are other options in this file that may be required under special circumstances, but the above settings should be suitable for most configurations.

regexroute.conf

By far the most popular routing module for YATE thus far is ***regexroute***, the regular expression routing module. While there are other routing modules available for YATE (you could even write your own if you were so inclined), the ***regexroute*** module is the one we will be using in this example.

Issue the following commands to create a copy of the sample file and create a working ***regexroute.conf*** file:

```
# cd /usr/local/etc/yate/
# cp regexroute.conf regexroute.conf.sample
```

Now, you will want to edit ***regexroute.conf***. There are many text editors that you can use to do this, but the easiest way to add text to the end of the file is as follows:

```
# cat >> regexroute.conf
^701$ = conf/sampleroom;lonely=true;billing=true;maxusers=150
Ctrl-D
```

Then type `service yate restart`, and your conferencing server will be ready to go.

Using rmanager for a YATE Console

If you are interested in opening up a console on YATE, you need to configure the remote manager. This module allows you to use the telnet application on your server to connect to the running YATE system and interact with it.

Open the ***rmanager.conf*** file in a text editor, and uncomment the line that says `port=5038` and the line that says `addr=127.0.0.1.`

Restart YATE, and you will be able to connect to a basic console with this command:

```
# telnet localhost 5038
```

If you want, you can bind to an IP address on the YATE server as well; however, this is probably not a good idea since telnet is not very secure. It's better to SSH into your system and then run telnet from the command line.

Some Things to Try

You've now completed all of the necessary configuration, and your conferencing server should be ready to use. To give you a sense of what you have at your disposal, here are some things that you can try to test out your new server.

Call Generator

When you call in to test your new conference room, you'll find that you are the only one there. You are going to notice silence. This is not a bad thing from a technical perspective, but it would be nice to hear something more. The following procedure will help you test the performance of your conference room.

First, you need a recording to play. You can download one using these commands:

```
# mkdir –p /usr/share/yate/sounds
# cd /usr/share/yate/sounds
# wget http://www.coretel.ca/files/yate/sounds/georgecarlin.slin
```

This will save to your */usr/share/yate/sounds* directory an audio file that is useful for testing audio mixers.

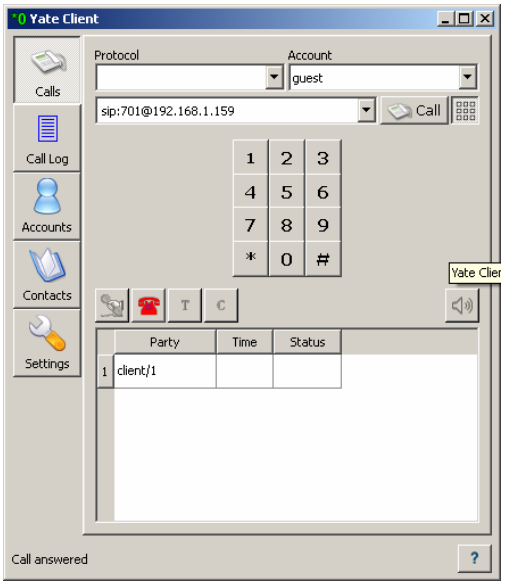
You now need to create a configuration file for the call generator. Do this as follows:

```
# cd /etc/yate
# cat >> callgen.conf
[parameters]
callto=sip/sip:701@localhost
source=wave/play//usr/share/yate/sounds/georgecarlin.slin
numcalls=400
avgdelay=1000
maxlife=15000
maxcalls=50
Ctrl-D
```

Telnet into your YATE console and type callgen start. Now when you call into your conference room, you should hear the result of this audio file being mixed into the conference room. To stop the call generator, enter callgen stop.

Installing the YATE Client on Your PC

The YATE engine is flexible and will run on all kinds of platforms. As an example of this, the YATE client for Windows uses the same YATE engine to deliver a softphone client.



Download the YATE client application for Windows from http://yate.null.ro/tarballs/yate2/yate2.exe and install it on your PC.

You will be able to run a little client app similar to the one you see here. Simply input the address of your YATE server and the extension of the conference server (e.g., sip:701@192.168.1.159) and you will be connected to your conference room.

Conclusion

The open source telecom landscape is rich with resources. As open source telecom software matures, hardware manufacturers must begin to offer products that do more than just provide drivers for specific software products. A visionary approach to existing and emerging technologies marks the difference between merely making hardware, and creating lasting value.

Appendix: Some YATE Message Types

To help you get a feel for the messaging system in YATE, here's a brief look at a few typical messages in the order in which they might occur in a call flow.

chan.startup

The `chan.startup` message begins a new call.

call.preroute

The capabilities of this message type are not commonly used, but they can be useful in special situations. For example, a `call.preroute` message can apply filters to deal with bugs in other systems, or perhaps add special parameters for special circumstances (such as adding a privacy flag to an incoming analogue channel, which would not arrive with such information).

Each channel module must send the `preroute` message. Even if no changes are being made, it is the `call.preroute` message that starts the call in YATE.

call.route

The `call.route` message is required on all calls. As its name implies, it defines where the call is to be sent. The called parameter is the key in this message. The reply to a `call.route` message will contain the parameter `callto`, which will define where the YATE engine should connect the call.

Most commonly, the `regexroute` module will be used to listen to and reply to these messages; however, other routing modules (such as the `callfork` module) could also be used.

call.execute

Once routing has been determined, the `call.execute` message requests the outgoing call leg. The callto parameter is the key to this message. Every module sees the message, but only the appropriate module (such as the SIP channel module, for example) will respond to it.

call.ringing

This message will be generated by the outgoing call module as soon as ringing has been established.

chan.dtmf

If any touch-tone digits are received on a channel, a `chan.dtmf` message will be generated.

call.cdr

`call.cdr` messages can be generated from anywhere in the system, enabling detailed call records to be built from many data sources. Call records can be initialized, updated, and finalized by the use of this message.

user.auth

This message is a request from a module to have a user authenticated. Typically, the auth module will respond to these requests. The `user.auth` message will contain a field that indicates who the user claims to be, but this information will not be trusted until the auth module has made a decision and replied. The auth module will determine what password is valid for that user and send that password back to the requesting module, which can then determine if it is a match.

user.register

This message is intended for the registering module. It tells the register module where the user is located and what technology the user is using (SIP, H.323, IAX).

Glossary

HDLC

High-Level Data Link Control is a synchronous data-link layer protocol that is used to carry the D-channel.

ISDN

The Integrated Services Digital Network is a group of protocols that were designed in the 1970s and 1980s to allow for the delivery of powerful, integrated features and services over subscriber telephone lines. There are two main forms of ISDN, called Basic Rate Interface (BRI) and Primary Rate Interface (PRI). This protocol was in part intended to bring the power of the SS7 network to end users.

ISDN-PRI

Primary Rate Interface ISDN (commonly called PRI) was designed to provide higher-density digital trunking to larger organizations.

Regular Expression (RegEx)

A regular expression is used in computing to search through strings of text. Regular expressions are powerful ways to define the rules by which information is identified within a string. As an example, a regex can be used to define concepts such as any line containing the string 416 or 647, or any line containing two of the word 'and' in sequence, or even lines containing the words Amsterdam, Boston, or Chicago, but not London or Tokyo. In this way, programmers making use of regular expressions can precisely define search criteria for content within other content9.

TDM

Time Division Multiplexing is a method of carrying multiple signals on a single circuit. The circuit is divided up into timeslots (in a T1 there are 24 timeslots, while in an E1 there are 32). Each channel is allowed to transmit only during its timeslot. The data rate in each timeslot is a fraction of the total bandwidth of the whole circuit, so, for example, a timeslot on a T1 circuit provides 64 kbps to its channel, while the whole T1 runs at 1544 kbps (there is 8k of overhead).

Timeslot

In any TDM circuit, each channel is allocated a timeslot. Therefore, in a digital telephony circuit, timeslots and channels can be considered the same thing.

[9] Regular expressions are generally considered difficult to learn, but they are also recognized as being extremely powerful. If you are serious about learning regular expressions, there is no better book than *Mastering Regular Expressions* by Jeffrey E. F. Friedl (O'Reilly Media)