ASTERISK И LINUX: МИССИЯ IP-ТЕЛЕФОНИЯ
ДЕЙСТВИЕ 2

МИХАИЛ ПЛАТОВ

В прошлой статье [1] мы познакомились с основными возможностями Asterisk PBX, настроив небольшую мини-АТС, работающую по протоколу SIP. Сегодня мы значительно расширим ее функциональность, добавив поддержку протокола H323 и обеспечив связь с городской сетью.

Альтернативный межгород

В прошлый раз мы предоставили нашим пользователям возможности дешевой междугородной связи. Для этого мы подключили наш сервер к сети одного из провайдеров по «родному» для Asterisk протоколу SIP. Однако, как показывает практика, провайдеры в большинстве своем не спешат переходить на SIP и при предоставлении своих услуг продолжают использовать более традиционный H323. Сильно расстраиваться по этому поводу не стоит – рано или поздно все равно все перейдут. А пока этого не произошло, решим эту проблему со своей стороны – установим H323-модуль Asterisk. На данный момент мне известно о существования четырех различных H323-драйверов (см. **таблицу 1**)для Asterisk:

- chan_oh323 – исторически первый H323-драйвер для Asterisk. При работе использует RTP-стек библиотеки OpenH323, со всеми вытекающими отсюда последствиями (поддерживаемые кодеки, стабильность, совместимость). Поддерживается компанией inAccess Networks [2].
- chan_h323 – отличительной особенностью данного драйвера является более тесная интеграция с Asterisk (используется Asterisk-реализация протокола RTP), что позволяет достичь большей производительности по сравнению с chan_oh323. Платой же за это является несколько меньшая функциональность. Так, например, chan_h323 не имеет собственного буфера дрожания (jitter buffer) [3], а также не поддерживает некоторые кодеки.
- chan_ooh323c – драйвер компании Objective Systems Inc [4], основанный на их открытом H323 стеке – Objective Open H.323 for C. И хотя количество поддерживаемых кодеков и H323-настроек несколько меньше, чем у конкурентов, драйвер вполне можно использовать. На данный момент статус драйвера – «beta», но со слов разработчиков, модуль в скором времени должен войдет в состав официальной версии Asterisk.
- chan_woomera – драйвер, позволяющий связать Asterisk с сервером Woomera. На данный момент через woomera можно совершать только H323-звонки, хотя в будущем разработчики обещают обеспечить поддержку универсального уровня абстракции OPAL.

Что же выбрать? Ответ на этот вопрос нужно искать исходя из конкретных задач.

Если вы используете STABLE-версию Asterisk, то варианты ограничиваются первыми двумя представителями. При этом, если планируемое количество одновременных звонков, совершаемых через H323, достаточно велико, то более предпочтительным будет chan_h323, если желательна более «качественная» реализация H323, то лучше использовать chan_oh323.

Драйверы chan_ooh323 и chan_woomera можно рекомендовать к использованию разработчикам и «очень продвинутым пользователям», желающим заглянуть в завтрашний день (H323 и SIP поверх IPv6 и т. д.). Для наших же задач возможностей chan_h323 или chan_oh323 будет более чем достаточно. Определенности ради будем считать, что мы выбрали chan_oh323, хотя приведенные ниже настройки с небольшими изменениями будут работать и с chan_h323.

Устанавливаем chan_oh323

Для канальных драйверов Asterisk отсутствуют готовые бинарные пакеты, поэтому воспользуемся единственно возможным способом установки – компиляцией из исходного кода.

При этом будем использовать машину следующей конфигурации:

- CPU AthlonXP 1500+
- MB Epox 8KHA+ (VIA KT266A)
- 512 Мб RAM
- HDD 40 Гб IDE Samsung
- Gentoo Linux Linux 2005.0 (ядро 2.6.12, gcc 3.4.3 и glibc 2.3.4)

Нам понадобятся следующие версии программ и библиотек:

- pwlib 1.6.6
- openh323 1.13.5
- chan_oh323 0.6.6
- Asterisk 1.0.9

Таблица 1. Сравнение H323-драйверов для Asterisk

Драйвер	Версия		Входит в поставку Asterisk	H323-стек	Характерные минусы	Характерные плюсы
	STABLE	HEAD				
chanoh323	+	+	–	OpenH323	Производительность	Совместимость, поддержка STABLE
chan_h323	+	+	+	OpenH323	Проблемы со сборкой, функциональность	Производительность, поддержка STABLE
chan_ooh323c	–	+	–	OOH323	Бета-версия, поддерживаются меньше кодеков	Производительность
chan_woomera	–	+	–	OpenH323/OPAL	Бета-версия, необходим сервер Woomera	Поддержка woomera, большой потенциал функциональности

Очень большое количество проблем при установке канальных драйверов H323 возникает именно из-за использования других версий pwlib и openh323. Кроме того, у некоторых людей наблюдались трудности при использовании готовых бинарных пакетов, входящих в состав у некоторых дистрибутивов.

Взять нужные версии лучше всего с сайта inAccessNetworks (к сожалению, у меня не получилось собрать openh323 с сайта inAccessNetworks, поэтому я взял эту же версию библиотеки из репозитория исходных кодов Gentoo) [5, 6, 7]. Итак, загружаем чуть меньше 4 Мб исходных кодов в /root/src/oh323 и переходим к установке. Начинать необходимо с pwlib.

```
# cd /root/src/oh323
# tar xfz ./pwlib-Janus_patch4-src-tar.gz
# cd pwlib
# ./configure
# make opt
# make install
```

Оптимизированная версия pwlib установлена, переходим к openh323.

```
# cd ..
# tar xfz ./openh323-v1_13_5-src.tar.gz
# cd openh323
# ./configure
# make opt
# make install
```

Библиотека OpenH323 собирается достаточно долго (в моем случае на это ушло чуть больше часа). Кроме того, во время компиляции вам понадобятся около 150 Мб дискового пространства и 300 Мб swap.

Теперь перейдем к канальному драйверу chan_oh323:

```
# cd ..
# tar xfz ./asterisk-oh323-0.6.6.tar.gz
# cd ./asterisk-oh323-0.6.6
# make
# make install
```

Драйвер установлен. Запустим Asterisk с режимом консоли (asterisk -cvvvvv) и выполним команду:

```
* CLI> show modules
```

Модуль chan_oh323 должен присутствовать в списке загруженных модулей (см. **рис. 1**).

Настраиваем chan_oh323

Прежде чем приступить к использованию chan_h323 в Asterisk, нам необходимо предварительно произвести настройку, а также внести соответствующие изменения в номерной план.

Параметры chan_oh323 хранятся в файле /etc/asterisk/oh323.conf. Итак, в контексте [general] определим следующие параметры (для остальных оставим значения по умолчанию):

```
[general]
listenAddress=IP_адрес_интерфейса
fastStart=yes
inBandDTMF=yes
gatekeeper=IP_адрес_GateKeeper
gatekeeperPassword=secret
context=generic-inc
```

Здесь мы указали:

- IP-адрес, который chan_oh323 будет использовать для работы с H323-устройствами (очень полезно, если у Linux-машины имеется несколько адресов в различных сетях);
- IP-адрес GateKeeper, пароль, с которым мы будем регистрироваться;
- контекст Asterisk, в который будут попадать все звонки, приходящие со стороны H323;
- включили режим faststart для более быстрой установки H323-соединений;
- сообщили серверу, что тоновые сигналы (DTMF) необходимо передавать внутри RTP-пакетов (inBandDTMF=yes).

Заметим, что приведенные значения параметров не являются «лучшими для всех случаев жизни», просто в моей ситуации именно этот набор вызывает наименьшее количество проблем.

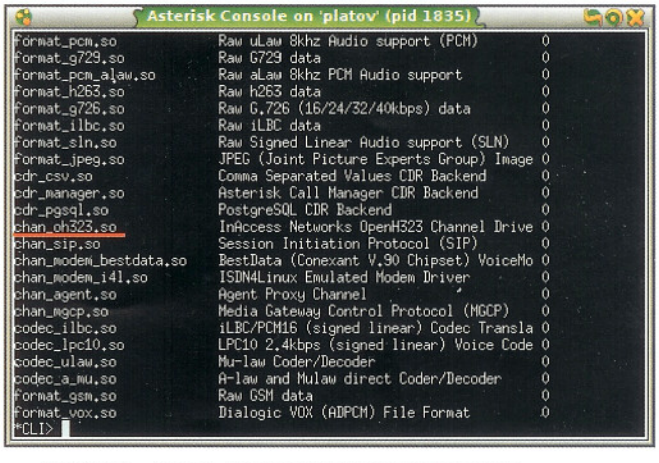

Рисунок 1. Список загруженных модулей Asterisk

Если провайдер предоставляет нам телефонные номера для входящих звонков, запишем их в секцию [register]:

```
[register]
context=generic_inc
alias=телефонный_номер_1_от_провайдера
alias=телефонный_номер_2_от_провайдера
```

Звонки, приходящие на указанные номера, будут попадать в контекст generic_inc (при необходимости звонки на разные номера можно направить в различные контексты).

Для кодеков определим следующие параметры:

```
[codecs]
codec=G729
frames=6
codec=G711U
frames=20
codec=G729A
frames=6
codec=G729B
frames=6
codec=G729AB
frames=6
codec=GSM0610
frames=4
codec=G7231
frames=4
```

Данные настройки определяют количество голосовых кадров, упаковываемых в один IP-пакет. Значение по умолчанию («1») слишком расточительно, поэтому поставим здесь типичные значения для голосового оборудования H323 – «4,6». Заданные параметры будут применены после следующего перезапуска Asterisk (см. «Перезагрузка конфигурации»). Перейдем к изменениям номерного плана. Для входящих звонков (со стороны H323) подойдет контекст, который мы создавали в прошлый раз (ведь по большому счету нам все равно посредством какой технологии до нас дозвонились):

```
[generic-inc]
exten => s,1,Wait, 1
exten => s, 2, Answer
exten => s,3, BackGround(local-welcome)
exten => s,4, WaitExten
exten => 200,1, Macro(stdexten,200,SIP/200)
exten => 201,1, Macro(stdexten,201,SIP/201)
exten => 202,1, Macro(stdexten,202,SIP/202)
exten=> 8500,1, VoiceMailMain
exten=> 8500,n, Hangup
```

А вот для исходящих звонков потребуются некоторые изменения. Добавим в контекст [office] следующую запись:

```
exten => _6 Dial(OH323/{EXTEN:1},20,rT)
```

При наборе номера с префиксом «6» звонок будет осуществляться с помощью chan_oh323, т.е. согласно номерному плану провайдера определенному на его GateKeeper.

Если же провайдер не использует Getekeeper (схема терминации на шлюз), то в файле oh323.conf необходимо указать:

```
gatekeeper=DISABLE
```

А в номерном плане написать так:

```
exten => _6 Dial(OH323/prefix{EXTEN:1}@gw_addr,20,rT)
```

Здесь prefix – служебный номерной префикс провайдера IP-телефонии, а gw_addr – IP-адрес его H323-шлюза.

Выход в город

Как вы могли заметить, на данный момент наша система обладает одним недостатком, несколько затрудняющим ее широкое использование, – отсутствие связи с городской телефонной сетью. Действительно довольно сложно представить себе ситуацию, в которой связь с ТФОП (или с уже существующей мини-АТС) не требовалась. Сейчас мы исправим это недоразумение.

Как вы помните из [3], существуют два основных способа подключения к городской телефонной сети: аналоговый (через стандартный телефонный провод) и цифровой (например, 30 каналов через интерфейс E1). Также нам известно, что «переводом» звонков из телефонной сети в VoIP занимаются шлюзы. Так вот, с Asterisk можно не только использовать обычные шлюзы H323, SIP с портами FXO, FXS, E1, но и специализированные модули выпускаемые специально для Asterisk. Фактически они представляют собой PCI-платы с соответствующими разъемами, необходимой электроникой «на борту», а также драйверами, позволяющими всему этому добру работать с Asterisk.

В город вместе с Digium

Наиболее известным (но не единственным [8]) производителем таких плат является компания Digium [9] – основной спонсор Open Source-проекта Asterisk. Спектр выпускаемых ею плат достаточно велик – от однопортовых FXS и FXO адаптеров, до модулей с четырьмя интерфейсами E1/T1. Платы с аналоговыми портами наиболее привлекательны для малых предприятий (2 платы 4FXO = 8 аналоговых линий (**рис. 2**)), цифровые модули наиболее интересны средним предприятиям (1 плата 4E1 – 120 цифровых каналов!).

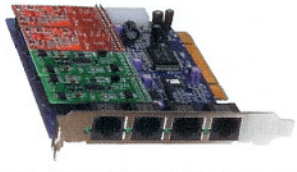

Рисунок 2. Модуль Digum TDM22B (2 FXO + 2 FXS)

С точки зрения стоимости особенно привлекательными являются платы Digium Wildcard X100P/X101P. С ними связана одна интересная история. Первоначально Wildcard X100P/X101P продавалась Digium по цене около 100$. Однако через какое-то время пользователи заметили, что на самом деле эти платы есть ни что иное, как обычные PCI-софт-модемы на широко распространенных чипсетах (Intel 537PU, 537PG, Ambient MD3200, Motorola 62802-51, первые 3 определятся как X101P или generic clone, последний как X100P), продаваемые в обычном компьютерном магазине за 10-15$. После того как эта информация стала общеизвестной, Digium ничего не оставалось, кроме как свернуть продажи линейки X100P/X101P (в последствии ее место занял внешний модуль IAXy). Впрочем, это не означает, что остальные поступили также. Наоборот, во многих online-магазинах эту плату (X100P, X101P

или их клоны) по-прежнему можно купить по цене 10-15$. К сожалению, в России софт-модемы на нужных чипсетах не продаются в широкой рознице (у нас более популярны модемы на микросхемах Conexant и Lucent, а не Intel и Motorola), поэтому я заказал эту плату (X101P) через Интернет [10, 11]. Итак, вставим модем в компьютер и настроим его с Asterisk.

Первое, что нам необходимо сделать, – убедиться, что модем определился системой. Для этого выполним команду:

```
# lspci
```

Если среди вывода этой утилиты присутствует строчка, похожая на эту:

```
0000:00:09.0 Network controller: Tiger Jet Network Inc. Intel 537
```

можете считать, что ядро Linux PCI-плату увидело и можно пробовать загружать ее драйверы:

```
# modprobe zaptel
# modprobe wcfxo
```

Если модули загрузились без ошибок, то в /var/log/messages мы увидим следующее:

```
Found a Wildcard FXO: Generic Clone
```

Данная строчка является лучшей иллюстрацией того, что с «железом» у нас все в порядке.

Однако прежде чем перейти к настройке Asterisk остановимся на еще одном важном моменте. Дело в том, что платы Digium реализуют «телефонный» интерфейс (FXO, FXS, E1, T1) программно (в zaptel-драйвере), используя для этого вычислительные ресурсы компьютера (в то время как производители других систем телефонии используют специализированные (и более дорогие) сигнальные процессоры – DSP). При этом ввиду специфики задачи (быстрая обработка сигналов) работа с платой ведется в режиме прерываний. Поэтому для нормальной работы очень важно настроить систему так, чтобы платы Digium ни с кем не делили одно прерывание, в противном случае возможны ухудшения качества звука. Посмотрим, как с этим обстоит ситуация у нас:

```
# cat /proc/interrupts
```

```
  0:  384085652          XT-PIC  timer
  1:     287894          XT-PIC  i8042
  2:          0          XT-PIC  cascade
  8:          2          XT-PIC  rtc
  9:          0          XT-PIC  acpi
 10:     181360          XT-PIC  wcfxo
 11:   48942244          XT-PIC  hci_hcd, uhci_hcd, uhci_hcd, eth0
 12:    2690249          XT-PIC  i8042
 14:     810676          XT-PIC  ide0
 15:     699164          XT-PIC  ide1
NMI:          0
LOC:          0
ERR:          0
```

Все нормально, плата монопольно использует прерывание 10. Перейдем к настройке Asterisk. Первым делом отредактируем файл /etc/zatel.conf:

```
fxsls=1
loadzone=us
defaultzone=us
```

Первая строчка означает, что для платы 1 необходимо использовать сигнализацию FXS loop-start (в Asterisk для плат FXO используется сигнализация FXS, и, наоборот; если при указанных настройках сервер не будет определять сигнал «занято», попробуйте другие разновидности fxs-сигнализации: fxs_ks, fxs_gs). Если плат в системе несколько (например, две), то в первой строчке нужно написать fxsks=1-2.

Строки 2 и 3 определяют параметры телефонной сети для используемого оборудования (частоты тоновых сигналов, их длительность и т. д.). К сожалению, в списке доступных стран России нет, поэтому, для того чтобы в вашем конкретном случае все заработало, возможно, придется перебрать несколько стран. Список всех доступных вариантов проще всего посмотреть в исходном тексте – файл zonedata.c библиотеки zaptel.

Следующим шагом при настройке нашего сервера является определение параметров карты в файле /etc/asterisk/zapata.conf. Выбор значения того или иного параметра определяется конкретной АТС, к которой мы подключаем Asterisk. Как известно, в России могут встречаться различные типы АТС. Привести универсальную конфигурацию, подходящую для всех случаев жизни, достаточно сложно, поэтому ограничимся описанием двух наиболее типичных – с импульсным набором (для старых АТС), с тоновым набором (для более-менее новых АТС). Итак, вначале определим общие параметры, подходящие практически для всех АТС:

```
[channels]
context=[generic-inc]
signalling=fxs _ ls
group=1
callgroup=1
pickupgroup=1
busydetect=yes
busycount=5
channel=1
```

Замечание. Приводимые параметры сгруппированы для более удобного изложения. При фактическом редактировании файла Zapata.conf настоятельно рекомендуется сохранять порядок, принятый в файле.

Итак, мы задали контекст, в который будут попадать все телефонные звонки, приходящие со стороны ТФОП. Определили тип сигнализации, используемой Asterisk при работе с FXO-портом.

Кроме того, мы создали группу каналов (состоящую из одной телефонной линии), включили распознавание сигнала «занято» (параметр busycount определяет количество коротких гудков, необходимое для того, чтобы Asterisk освободил линию), а также разрешили перехват звонков с ТФОП.

Теперь перейдем к параметрам, определяемым в зависимости от возможностей конкретной АТС:

```
callerid=asreceived
callprogres=yes
usecallerid=yes
usecallingpres=yes
callwaitingcallerid=yes
threewaycalling=yes
transfer=yes
canpark=yes
cancallforward=yes
```

Первой строкой мы сказали, что сигналы «европейского АОН», приходящие из города, нужно передавать в Asterisk без изменений. Вторая строка включает отслеживание состояния линии (сигналы вызова, ответ, занято) при звонках. (На данный момент эта опция является экспериментальной и может работать некорректно). Следующие 3 строки актуальны, если телефон, подключенный к данной линии, может нормально работать с flash (извещение о входящем вызове, конференц-связь, АОН должны поддерживаться и быть активизированы на АТС). Последние два параметра разрешают парковку и передачу звонков с ТФОП.

Заметим, что приведенные выше настройки актуальны для более-менее современных АТС (электронные, цифровые). Для их престарелых собратьев (координатных, шаговых, декадно-шаговых) вышеперечисленные параметры, скорее всего, придется установить в «no». Кроме того, для активизации импульсного набора нужно будет добавить следующую строчку:

```
pulsedial=yes
```

В независимости от АТС нам, скорее всего, понадобится включить подавление «эха» для zaptel-каналов. Для этого добавим следующие две строчки:

```
echocancel=yes
echocancelwhenbridging=yes
```

Если же с помощью этих настроек полностью избавиться от эха не удается, можно дополнительно задействовать следующие:

```
echotraining=yes
txgain=0.0
rxgain=0.0
```

Первой строкой мы включаем режим тренировки эхоподавления (полезно, если «эхо» регулярно слышно в начале разговора), а последними двумя задаем усиление для входного и выходного сигналов. Для определения конкретных значений последних можно воспользоваться утилитой ztmonitor, входящей в модуль zaptel.

Перейдем к настройке номерного плана. В качестве префикса для выхода в город будем использовать цифру «5». В уже знакомый нам контекст [office] добавим следующую строку:

```
exten=> _5.,1,Dial(ZAP/g1/EXTEN:1,20,rT)
```

Этим мы указали Asterisk, что если все номера, начинающиеся с цифры 6, должны передаваться на группу каналов 1 zaptel-интерфейса. Первая цифра набранного номера будет отбрасываться сервером Asterisk.

Необходимо заметить, что способ маршрутизации звонков с использованием префиксов не является единственным. Так ничто не мешает использовать нам следующую конструкцию:

```
[pstn-dialout]
exten => s,1,Wait, 1
exten => s,2, Answer
exten => s,3, BackGround(pstn-announce)
exten => s,4, Read(ext)
exten => s,5, Dial(ZAP/g1/${ext:1},20,rT)

[office]
exten => 101,1,Goto(pstn-dialout,s,1)
```

Таким образом, при наборе номера 101 Asterisk перейдет в контекст pstn-dialout, зачитает приветствие pstn-announce (которое необходимо предварительно записать), после чего совершит звонок в город по введенному нами номеру. Подобную схему можно с успехом использовать и для других технологий (H323, SIP).

Выход через SIP-шлюз

К сожалению, далеко не всегда имеется возможность использования оборудования Digium. Например, в ввиду физических ограничений иногда бывает несколько затруднительно установить PCI-платы в стандартные стоечные серверы (1U, 2U). Кроме того, рынок оборудования IP-телефонии в России все еще находится в состоянии, близком к зачаточному, и найти в продаже платы Digium по разумным ценам может быть несколько проблематичным.

Для решения этих проблем рассмотрим альтернативный вариант сопряжения Asterisk с городской сетью – с помощью «внешних» голосовых шлюзов. Теоретически совместно с Asterisk можно использовать любые из шлюзов, поддерживающих протоколы SIP/H323/SCCP, однако, по моему мнению, на практике совместно с Asterisk лучше использовать шлюзы с протоколом SIP. Наиболее известными производителями таких устройств (совместимых с Asterisk) являются – Cisco, Mediatrix, Quintum, AddPac, VegaStream и D-Link.

В качестве примера рассмотрим сопряжение Asterisk с городской сетью с использованием шлюза AddPac AP200D. Данный шлюз представляет собой отдельное устройство, имеющее 2 порта FXO (**рис. 3**).

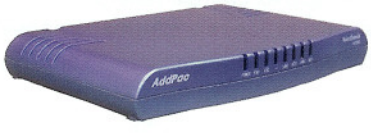

Рисунок 3. Шлюз IP-телефонии AddPac AP200D (2FXO)

Устройство поддерживает все наиболее популярные способы конфигурации (console, telnet, web, EMS), а его командный интерфейс очень сильно похож на интерфейс устройств Cisco (таким образом, с некоторыми поправками приведенное ниже применимо и к голосовому оборудованию Cisco). Для базовой конфигурации устройства необходимо определить сетевые и голосовые параметры. И если с первыми все достаточно просто (нужно всего лишь определить стандартные настройки – IP-адрес, маску подсети и маршрут по умолчанию), то со вторыми могут возникнуть затруднения. Остановимся на них более подробно.

При описании голосовых параметров ключевым является понятие dial-peer – точка, участвующая в голосовом соединении. Существует два типа dial-peer (см. **рис. 4**):

- POTSdial-peer – описывает параметры соединения с традиционной телефонной сетью. POTS dial-peer «привязываются» к аппаратным портам (FXO, FXS) шлюза.
- VOIPdial-peer – описывает параметры соединения с VoIP-сетью. Для простоты понимания VOIP dial-peer можно ассоциировать с Ethernet-разъемом шлюза.

Полезные советы

Запись звуков

При создании голосовых меню часто возникает необходимость использования собственных файлов озвучки. Звуковые файлы, используемые в Asterisk, хранятся в формате gsm. Наиболее просто и быстро записать такие файлы можно с помощью самого Asterisk. Для этого в номерном плане можно создать специальный контекст со следующим содержанием:

```
[gsm-record]
exten => 150,1,Wait(2)
exten => 150,2,Record(testrecord:gsm)
exten => 150,3,Wait(2)
exten => 150,4,Hangup
exten => 151,1,Playback(testrecord2)
exten => 151,2,Wait(2)
exten => 151,3,Hangup
```

Позвонив на номер 150, мы сможем наговорить необходимый текст, который будет сохранен в стандартной папке звуков Asterisk (/var/lib/asterisk/sound) в файле ivrrecording.gsm. Прослушать содержимое этого файла можно, позвонив по номеру 151.

Помимо записи с помощью Asterisk можно использовать специальные утилиты конвертации форматов, например sox [12]:

```
# sox inputfile.wav -r 8000 -c 1 outputfile.gsm resample -ql
```

Перезагрузка конфигурации

Как вы могли заметить, после редактирования конфигурационных файлов мы всегда перезапускали сервер. Действительно для некоторых модулей (например, zaptel и oh323) такая операция обязательна. Однако при незначительных изменениях (добавление нового устройства, изменение номерного плана и т. д.) полного перезапуска сервера можно избежать. Для этого в консоли asterisk достаточно выполнить следующую команду:

```
*CLI > reload
```

Ежедневная перезагрузка

Некоторые пользователи [13] отмечали проблемы с утечками памяти при очень интенсивном использовании Asterisk. Эта проблема в той или иной степени присуща всем продуктам и вызвана ошибками в исходном коде. Безусловно, самым правильным способом ее решения является исправление исходного кода. Однако в качестве временной меры (до тех пор пока ошибки в коде не будут исправлены разработчиками) можно предложить регулярно (для профилактики) перезапускать сервер с Asterisk в свободное время, например, ночью. Для этого можно добавить в файл /etc/crontab строку следующего содержания:

```
0  0  * * *     root    /etc/init/asterisk restart> /dev/nu
```

Восстановление после сбоев

Не исключено, что во время работы процесс Asterisk «совершит недопустимую операцию и будет закрыт». (И хотя такая ситуация для STABLE-версии является из ряда вон выходящей, не учитывать ее появление для «боевой» системы все же нельзя.) Как это отразится на пользователях?

Думаю, что не нужно быть пророком, чтобы понимать, что в этом случае связи не будет. Возможным решением этой проблемы может быть сокращение общего времени простоя сервиса путем его быстрого перезапуска. Для Asterisk эта возможность является практически стандартной – достаточно просто для запуска использовать файл asterisk_safe (вместо asterisk). Теперь при «авариях» сервис будет перезапускаться автоматически. В принципе аналогичного результата можно добиться и с помощью простого shell-скрипта, выполняемого по cron.

Для минимальной настройки нам понадобится сконфигурировать два dial-peer (по одному каждого типа), а также определить правила маршрутизации между ними внутри нашего шлюза. POTS dial-peer «привяжем» к FXO-линии, а VOIP dial-peer «настроим» на Asterisk. Звонок из ТФОП на наш сервер будут делаться с помощью донабора внутреннего номера. При наборе городских номеров из Asterisk будем добавлять префикс «9» перед исходным номером.

Для конфигурирования воспользуемся telnet-интерфейсом шлюза. Для этого подключимся к устройству и перейдем в интерфейс конфигурирования с помощью команды conf:

```
AP200# conf
Enter configuration commands, one per line. End with CNTL/Z
AP200(config)#
```

Создадим POTS-dial-peer с номером 0 (для второго порта нужно произвести аналогичные действия):

```
AP200(config)# dial-peer voice 0 pots
AP200(config-dialpeer-pots-0)#
```

Определим для него следующие параметры:

```
dial-peer voice 0 pots
 destination-pattern 9T
 port 0/0
 user-name 100
 user-password potsfxopwd1
```

Только что мы сказали, что все звонки, приходящие на шлюз и начинающиеся с номера «9», должны уходить на первый порт FXO, ассоциированный с dial-peer 0. При звонках из ТФОП на указанный FXO-порт шлюз будет аутентифицироваться на Asterisk с использованием указанных имени пользователя и пароля. Правда, для этого устройству предварительно необходимо указать, где именно находится Asterisk, которому будут передаваться звонки из ТФОП. Опишем VOIP dial-peer:

```
AP200(config)# dial-peer voice 1000 voip
AP200(config-dialpeer-voip-1000)#
```

Определим следующие параметры:

```
dial-peer voice 1000 voip
```

```
 destination-pattern T
 session target sip-server
 session protocol sip
 dtmf-relay rtp-2833
 no vad
```

Перейдем в контекст определения настроек sip user-agent:

```
AP200(config)# sip-ua
AP200(config-sip-ua)#
```

Зададим следующие параметры:

```
sip-ua
 user-register
 sip-server 192.168.0.20
 register e164
```

Этим мы сказали шлюзу, что SIP-сервер находится по адресу 192.168.0.20. и при регистрации на нем необходимо использовать e164-номер (международный стандарт, определяющий правила нумерации абонентов в телефонных сетях) и пароль, указанными для POTS dial-peer. Теперь разберемся со «служебным» префиксом «9». Очевидно, что передавать его в ТФОП нельзя, поэтому его необходимо отбрасывать. Делать это можно как на Asterisk (см. выше примеры для межгорода и плат Digium), так и непосредственно на шлюзе. Воспользуемся вторым способом. Для этого создадим translation-rule и применим его к POTS dial-peer:

```
AP200(config)# translation-rule 0
AP200(config-translation-rule#0)# rule 0 9T T
AP200(config-translation-rule#0)# exit
AP200(config)# dial-peer voice 0
AP200(config-dialpeer-pots-0)# translate-outgoing ↵
      called-number 0
```

Теперь при наборе номера через первый голосовой порт FXO префикс «9» будет отбрасываться.

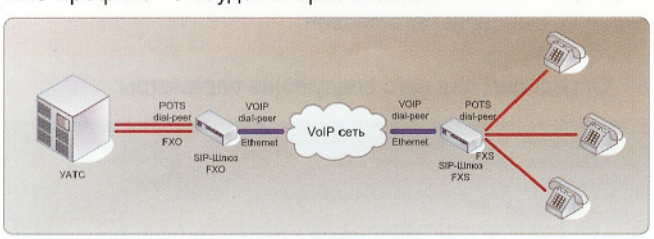

Рисунок 4. POTS и VOIP dial-peer

Итак, с настройками шлюза мы закончили. Применим конфигурацию и перейдем к настройке Asterisk:

```
AP200(config)# write
Do you want to WRITE configuration ? [y|n] y
Writing configuration....done
AP200(config)# reboot
System Reboot...
```

Для того чтобы использовать шлюз из Asterisk, нам понадобится создать для него учетную запись SIP, а также внести необходимые изменения в номерной план. Добавим в файл /etc/asterisk/sip.conf следующие строки:

```
[100] ;AP200
type=friend
host=dynamic
username=100
secret= potsfxopwd1
dtmfmode=rfc2833
context=generic-inc
```

Звонки из ТФОП будем направлять в наш стандартный контекст для входящих звонков – generic_inc.

Для исходящих звонков добавим в контекст [office] следующую строчку:

```
exten => _ 9,Dial(SIP/EXTEN@100,20,rT)
```

Перезапустим Asterisk.Через некоторое время шлюз зарегистрируется на сервере, о чем можно будет узнать, выполнив на нем следующую команду:

```
AP200# show sip
Proxyserver Registration Information
   proxyserver registration option = e164
   Proxyserver list :
      ------------------------------------------------------------------
        Server address          Port    Priority        Status
      ------------------------------------------------------------------
        192.168.0.20            5060    128     Registered(E.164)

   Proxyserver registration status :
      ------------------------------------------
        UserName        Regist          Status
      ------------------------------------------
        100             yes         Registered
```

Теперь мы можем звонить не только нашим соседям по офису, но и любым абонентам городской телефонной сети!

Заключение

Оглядываясь на проделанное, можно смело сказать – теперь у нас есть полноценная (по функциональности) мини-АТС. Изучение Asterisk на этом не заканчивается! В следующий раз мы более подробно остановимся на способах облегчения управления сервером, рассмотрев установку различных веб-интерфейсов.

До встречи!

Литература и ссылки:

1. Платов М.В. Asterisk и Linux: миссия IP-телефония. – Журнал «Системный администратор», №6, 2005 г. – 12-19 с.
2. http://www.inaccessnetworks.com/projects/asterisk-oh323.
3. Платов М. Что важно знать об IP-телефонии. – Журнал «Системный администратор», №5, 2005 г. – 20-25 с.
4. http://obj-sys.com/open/index.shtml.
5. http://www.inaccessnetworks.com/ian/projects/asterisk-oh323/Libraries/pwlib-Janus_patch4-src-tar.gz.
6. ftp://ftp.gtlib.cc.gatech.edu/pub/gentoo/distfiles/openh323-v1_13_5-src.tar.gz.
7. http://www.inaccessnetworks.com/projects/asterisk-oh323/download/asterisk-oh323-0.6.6.tar.gz.
8. http://www.voipsupply.com/index.php?cPath=99.
9. http:/www.digium.com.
10. http://www.goods2world.com.
11. http://www.iaxtalk.com.
13. http://sox.sourceforge.net.
14. http://www.voip-info.org/tiki-index.php?page=Asterisk+administration.