- Шлюз на debian 6.0.4 squeeze 2.6.32-5-686;
  - isc-dhcp-server, bind9, iptables...
- 3 интерфейса;
  - eth0 смотрит на провайдера интернет, там же идет вещание iptv 172.31.242.0/24;
  - eth1 смотрит в локальную сеть пользователя 192.168.0.1/24;
  - eth2 смотрит в другую локальную сеть, там тоже есть интернет 192.168.0.0/24;
- Iptv приставка MAG250, пока одна, но положение изменится, внесу поправки в конфиг;
- И маловажный факт, приставка со шлюзом соединена с помощью двух адаптеров D'link DHP-501AV, он конечно подпортил жизнь, но об этом в другой раз;

Прежде чем приступить к настройке нужно понять, что за зверь такой multicast, и так, маленький "ликбез".

Существует несколько вариантов передачи пакетов в сетях tcp/ip

- unicast, пакеты передаются одному определенному адресату;
- broadcast, пакеты передаются всем;
- multicast, пакеты передаются некому подмножеству адресов, входящих в определенную группу;
- anycast, появился в tcp/ip v6, пакет передается до первого принявшего узла из некого подмножества.

Подробнее можно почитать в википедии, пока ее не закрыли *привет реестр сайтов

Для работы iptv использует протокол igmp, по этому нам нужно разрешить его на шлюзе и научить транслировать все это добро в локальную сеть. Будем использовать программу igmpproxy. Её нет в репозитории debian и нам придется ее собирать. Для сборки понадобится gcc и make со всеми зависимостями.

Качаем igmpproxy:

```
wget
http://sourceforge.net/projects/igmpproxy/files/igmpproxy/0.1/igmpproxy-0.1.
tar.gz
```

На сегодня 0.1 последняя версия, распаковываем и переходим в папку с программой:

```
tar -xzf igmpproxy-0.1.tar.gz
cd igmpproxy-0.1
```

Собираем и устанавливаем, не забываем выполнять действия от имени администратора (sudo):

```
sudo ./configure
sudo make
sudo make install
sudo checkinstall -D -y --pkgname=igmpproxy --pkgversion=0.1-beta2 --nodoc
```

Отредактируем конфигурационный файл igmpproxy

```
sudo nano /etc/igmpproxy.conf
```

У вас он может оказаться в другом месте, например в `/usr/local/etc/igmpproxy.conf`

Важно выяснить с каких адресов идет вещание, для этого вооружимся программой tcpdump, она входит в стандартные репозитории debian, и если еще не установлена, то:

```
sudo apt-get install tcpdump
```

Используем:

```
sudo tcpdump -i eth0 -n -t port 1234
```

Немного поясню:

- -i eth0 интерфейс который собираемся анализировать
- -n заменяет хост ip адресом
- -t не выводим метку времени, для удобства port 1234 отображаем информацию переданную исключительно на порт 1234, порт на котором сидит igmpproxy

Посыпется информация из которой нужно выудить адреса вещания, точнее диапазон который будет переправляться к нам за nat.

Вывод будет примерно следующий:

```
172.31.242.63.2352 > 239.32.1.15.1234: UDP, length 1316
172.31.242.32.1418 > 239.32.1.90.1234: UDP, length 1316
```

Соответственно диапазон вещания 172.31.242.0

С полученной информацией идем редактировать конфиг, повторюсь:

```
sudo nano /etc/igmpproxy.conf
```

Он должен иметь примерно такой вид:

```
quickleave phyint eth0 upstream ratelimit 0 threshold 1 # Интерфейс с которого забираем трафик
altnet 172.31.242.0/24 # Узнаем из вывода tcpdump
altnet 224.0.0.0/24 # Узнаем из плейлиста скаченного у провайдера, открыв его любым текстовым редактором
phyint eth1 downstream ratelimit 0 threshold 1 # eth1 интерфейс на который перенаправляем трафик
phyint tun0 disabled # Интерфейсы которые нужно игнорировать
phyint vmnet8 disabled
```

В iptables нужно добавить следующие строки (все исходя из вашего плейлиста, тут описаны сугубо мои настройки)

```
iptables -A INPUT -d 224.0.0.0/240.0.0.0 -i eth0 -j ACCEPT
iptables -A FORWARD -p udp -d 224.0.0.0/240.0.0.0 -j ACCEPT
```

Так же проверьте что бы были разрешены multicast, igmp и порты udp с 5242 по 5352, и порт 1235 для получения списка каналов.

Далее необходимо добавить igmpproxy в автозагрузку, напоминаю, что у меня debian. Создаем файл `/etc/init.d/igmpproxy`

```
sudo nano /etc/init.d/igmpproxy
```

В него записываем следующее содержание, не помню у кого украдено:

```
#!/bin/sh
### BEGIN INIT INFO
# Provides: igmp proxy
# Required-Start: $remote_fs
# Required-Stop: $remote_fsx
# Default-Start: 2 3 4 5
# Default-Stop: 0 1 6
# Short-Description: Example initscript
# Description: IGMP Proxy init file
 ### END INIT INFO
 # Do NOT "set -e"
 # PATH should only include /usr/* if it runs after the mountnfs.sh script
 #PATH=/sbin:/usr/sbin:/bin:/usr/bin
 DESC="IGMP Proxy"
 NAME=igmpproxy
 DAEMON=/usr/local/sbin/$NAME
 DAEMON_ARGS="/usr/local/etc/igmpproxy.conf"
 PIDFILE=/var/run/$NAME.pid
 SCRIPTNAME=/etc/init.d/$NAME
 # Exit if the package is not installed
 [ -x "$DAEMON" ] || exit 0
 # Read configuration variable file if it is present
 [ -r /etc/default/$NAME ] && . /etc/default/$NAME
 # Load the VERBOSE setting and other rcS variables
 . /lib/init/vars.sh
 # Define LSB log_* functions.
 # Depend on lsb-base (>= 3.0-6) to ensure that this file is present.
 . /lib/lsb/init-functions
 #
 # Function that starts the daemon/service
 #
 do_start()
 {
 # Return
 # 0 if daemon has been started
 # 1 if daemon was already running
 # 2 if daemon could not be started
 start-stop-daemon --start --quiet --pidfile $PIDFILE --exec $DAEMON --test
> /dev/null \
 || return 1
 start-stop-daemon --start --quiet --background --make-pidfile --pidfile
```

```
$PIDFILE --exec $DAEMON -- \
 $DAEMON_ARGS \
 || return 2
 # Add code here, if necessary, that waits for the process to be ready
 # to handle requests from services started subsequently which depend
 # on this one. As a last resort, sleep for some time.
 }
 #
 # Function that stops the daemon/service
 #
 do_stop()
 {
 # Return
 # 0 if daemon has been stopped
 # 1 if daemon was already stopped
 # 2 if daemon could not be stopped
 # other if a failure occurred
 start-stop-daemon --stop --quiet --retry=TERM/30/KILL/5 --pidfile $PIDFILE
--name $NAME
 RETVAL="$?"
 [ "$RETVAL" = 2 ] && return 2
 # Wait for children to finish too if this is a daemon that forks
 # and if the daemon is only ever run from this initscript.
 # If the above conditions are not satisfied then add some other code
 # that waits for the process to drop all resources that could be
 # needed by services started subsequently. A last resort is to
 # sleep for some time.
 start-stop-daemon --stop --quiet --oknodo --retry=0/30/KILL/5 --exec
$DAEMON
 [ "$?" = 2 ] && return 2
 # Many daemons don't delete their pidfiles when they exit.
 rm -f $PIDFILE
 return "$RETVAL"
 }
 #
 # Function that sends a SIGHUP to the daemon/service
 #
 do_reload() {
 #
 # If the daemon can reload its configuration without
 # restarting (for example, when it is sent a SIGHUP),
 # then implement that here.
 #
 start-stop-daemon --stop --signal 1 --quiet --pidfile $PIDFILE --name $NAME
 return 0
 }
 case "$1" in
 start)
 [ "$VERBOSE" != no ] && log_daemon_msg "Starting $DESC" "$NAME"
 do_start
```

```
case "$?" in
0|1) [ "$VERBOSE" != no ] && log_end_msg 0 ;;
2) [ "$VERBOSE" != no ] && log_end_msg 1 ;;
esac
;;
stop)
[ "$VERBOSE" != no ] && log_daemon_msg "Stopping $DESC" "$NAME"
do_stop
case "$?" in
0|1) [ "$VERBOSE" != no ] && log_end_msg 0 ;;
2) [ "$VERBOSE" != no ] && log_end_msg 1 ;;
esac
;;
#reload|force-reload)
#
# If do_reload() is not implemented then leave this commented out
# and leave 'force-reload' as an alias for 'restart'.
#
#log_daemon_msg "Reloading $DESC" "$NAME"
#do_reload
#log_end_msg $?
#;;
restart|force-reload)
#
# If the "reload" option is implemented then remove the
# 'force-reload' alias
#
log_daemon_msg "Restarting $DESC" "$NAME"
do_stop
case "$?" in
0|1)
do_start
case "$?" in
0) log_end_msg 0 ;;
1) log_end_msg 1 ;; # Old process is still running
*) log_end_msg 1 ;; # Failed to start
esac
;;
*)
# Failed to stop
log_end_msg 1
;;
esac
;;
*)
#echo "Usage: $SCRIPTNAME {start|stop|restart|reload|force-reload}" >&2
echo "Usage: $SCRIPTNAME {start|stop|restart|force-reload}" >&2
exit 3
;;
esac
:
```

Затем

```
sudo chown root:root /etc/init.d/igmpproxy # Меняем владельца
sudo chmod 755 /etc/init.d/igmpproxy # Меняем права и делаем исполняемым
sudo update-rc.d igmpproxy defaults # Добавляем в загрузку
sudo /etc/init.d/igmpproxy start # Стартуем
```

Теперь нужно проверить, включена ли поддержка multicast в ядре, для этого выполните команду :

```
sudo ifconfig eth0
```

И найдите слово MULTICAST, если есть, значит все нормально, в ином случае надо пересобрать ядро с поддержкой multicast

```
CONFIG_IP_MULTICAST=y
```

Все должно работать, всех благ.