Table of Contents

1. Introduction to FitVT

After a few months working in our labs we want to introduce you to the FitVT, our low cost/low power server for the Village Telco based on FitPC2i low power computer.

After testing different low power solutions based on commodity laptops, we discovered the FitPC[1]. The Fitpc2i comes with two Gigabit network interfaces and the CompuLab SBC-FITPC2 series board that uses a fanless Atom Z550 32bits CPU.

In most of the scenarios the VT server will sit behind a village telco super-node and the Internet. Although we have tested commodity laptops using a USB-Ethernet converter as second interface, we have seen very bad performance in the data transfers of some USB dongles (USB 1.1). So if you plan to provide also Internet access from the mesh nodes, consider a solution with two built-in network interfaces.

1 http://www.fit-pc.com/web/fit-pc/fit-pc2i-specifications/

Installing Ubuntu 10.04 in the fitpc2i[2] was as easy as installing Ubuntu in any other computer. Use a bootable USB image or attach an external USB-CD Drive. Just make sure that you choose a 32 bits image as the Atom CPU runs 32 bits.

In our laboratory we have tested the Fitpc2i with a Intel(R) Atom(TM) CPU Z550@ 2.00GHz[3] and 2 GB of RAM. The unit runs at 35-40 C when idle so make sure that you do not forget it under a pile of books as I did. There are at least two revisions of this hardware but the process of installation should be the same.

CompuLab, the company behind the FiTVT provides support for Linux Mint as the hardware seems to be targeted to those that want to run a nice multimedia station at home (i.e. stream audio/video obtained from good friends to a television). We warned that the unit does not come with VGA output and has a DVI Digital output up to 1920 x 1200 through HDMI connector instead, so you will need a LCD/TV screen with HDMI or DVI input to connect the FitPC.

There are two useful accessories that you can consider, an analog VGA converter is powered from the DVI port and requires no external power source and the external heat-sink.

The main glitch in the installation process is that the stock kernel that ships with Ubuntu 10.04 has buggy R8169 network driver that has broken auto-negotation. In my case, only one of the interfaces was able to negotiate the speed, while the other interface required the *ethtool* to force the speed to 10 Mbps. Fixing the speed of the second card to 10 Mbps was the only solution so you can upgrade your kernel.

Once you upgrade the kernel from version 2.6.32-33 to 2.6.32-38 you will discover that the DADHI drivers that are used Asterisk are now broken and you will have to rebuild them. The DADHI drivers depend on kernel version (i.e. wrong symbols). In a nutshell, in order to get the network cards to work you will have to (1) upgrade your kernel and (2) rebuild DADHI Asterisk drivers.

The default linux kernel uses the PAE extensions that allows to run 32bits with lots of RAM.

2 http://www.fit-pc.com/wiki/index.php/Fit-PC2i_Revisions

3 http://www.cpu-world.com/CPUs/Atom/Intel-Atom%20Z550%20AC80566UE041DW.html

2. What is in the FitVT server image?

To facilitate the work to other developers we have created a bootable ISO that you can install directly in the server. The bootable ISO is based on *remastersys*[4]

1) The Village Telco Server Package (http://dev.villagatelco.org)
2) The Simple Unified Dashboard (SPUD) monitoring (http://spud.villagatelco.org)
3) Coova Captive portal

The Village Telco Server package is composed of four main components:

- **asterisk realtime** : Based on asterisk 1.6 and mysql realtime support.
- **a2billing village telco** : Based on a2billing version 1.7.0, a2biling incorporates our Village Telco simplified GUI, a SOAP webservice and a few extra patches to work in Ubuntu 10.04. A2billing is written in PHP+smarty and interfaces with Asterisk using AGI and AMI.
- **a2billing installation wizard** : The wizard interfaces with the SOAP webservice and performs a full installation of a2billing for Village Telco in 5 steps. The wizard is written in PHP+Cake MVC and interfaces with a2billing with SOAP.
- **a3glue** : The a3glue is a webservice that follows the REST architecture and delivers JSON provisioning and monitoring information to other components of the Village Telco as the Afrimesh monitoring panel. A3glue is written in PHP and interfaces with Asterisk using Asterisk AJAM (MXML).

The Simple Unified Dashboard (SPUD) monitoring is a wireless mesh network visualisation tool for BATMAN mesh networks and its users.

SPUD is a PHP based dashboard that communicates with the BATMAN visualization server and displays real time wireless link status. The software is written in CakePHP (a PHP-based MVC framework) and uses Google Maps API 1.3 for visualization.

SPUD is designed to be as simple as possible to use, and to enable teams, that have installed large amount of mesh nodes, to visualize their networks quickly.

Some of the core features of SPUD are:

- **Client management:** Bulk import of clients from CSV file, Edit client

4 http://www.geekconnection.org/remastersys/

position with Google Maps, Tracking of new clients
- **Link monitoring:** Easy overview of active wireless links, Mesh quality in each direction of a wireless link
- **Customization**: Colours and threshold values for link quality

And the Coova captive portal that integrated an universal access method (UAM) for wireless Internet access.

What is the default network setup?

Our default network setup uses B.A.T.M.A.N. in layer3 and looks as follows:

- The mesh potatoes use the IP network 10.130.1.0/24 by default. One of the mesh potatoes is the gateway to the wired network (Super Node). In our laboratory setup with have configured the super node (the wired mesh potato) with the IP address 10.130.1.2 (ath0/wirelesss) / 192.168.130.2 (eth0 /wired).
- Our super-node is installed in a Ubiquity NS2 with modified openwrt firmware that includes BATMAN and the legacy JSON output from the VIS server. Firmware images are available here:

```
http://dev.villagetelco.org/svn/villagetelco/ubnt2/
```

- The Village Telco Server runs the IP 192.168.130.1 (eth1, ETH2) and the upstream interface runs DHCP (eth0, ETH1).

Two major configuration steps are needed:

- Advertising the server network attached to the wired interface of the super node: you need to advertise the 192.168.130.0/24 network inside of the mesh via the super node. In the super node edit the configuration files

```
/etc/config/batmand
/etc/config/network

root@OpenWrt:/etc/config# cat batmand
config batmand general
        option interface                ath0
        option announce                 192.168.130.0/24
        option gateway_class
        option originator_interval
        option preferred_gateway
        option routing_class
        option visualisation_srv
        option policy_routing_script


root@OpenWrt:/etc/config# cat network

config 'interface' 'loopback'
        option 'ifname' 'lo'
        option 'proto' 'static'
```

```
        option 'ipaddr' '127.0.0.1'
        option 'netmask' '255.0.0.0'

config 'interface' 'lan'
        option 'ifname' 'eth0'
        option 'proto' 'static'
        option 'netmask' '255.255.255.0'
        option 'ipaddr' '192.168.130.2'
        option 'dns' '192.168.130.1'
        option 'gateway' '192.168.130.1'

config 'interface' 'wifi0'
        option 'ifname' 'ath0'
        option 'proto' 'static'
        option 'netmask' '255.255.255.0'
        option 'ipaddr' '10.130.1.2'
```

- In the server we need to add a route back to the mesh network and MASQUERADE all outgoing connections to the Internet.

3. Installing all dependencies

The village telco server bundle runs a2billing, asterisk, spud and coova. The packages have dependencies on apache, mysql, php, curl, espeak, soap, freeradius, etc. We have compiled all package dependencies in this list:

```
apt-get install libapache2-mod-php5 php5 php5-common php5-cli \
php5-mysql mysql-server apache2 php5-gd php5-curl php5-mcrypt php-soap \
php-pear espeak asterisk subversion dpkg-dev linux-headers-2.6.32-38 \
linux-headers-2.6.32-38-generic-pae linux-image-2.6.32-38-generic-pae \
freeradius freeradius-mysql coova-chilli libapache2-mod-auth-mysql libssl-dev ssl-
cert gcc
```

IMPORTANT! Remember your mysql admin/root password as you will need it at a later stage.

4. Building and installing village-telco package

Detailed information of how to build the village-telco debian package is available here:

http://dev.villagetelco.org/trac

As discussed in the introduction section you will need to update your kernel to ensure that the drivers for the Realtek gigabit cards are updated.

```
apt-get dist-upgrade
apt-get install linux-headers-generic-pae
```

if you see error messages like the following it means that you need to rebuild your asterisk DAHDI drivers. DAHDI (former Zaptel) is the open source device interface technology used to control Digium and other legacy telephony interface. Although you do not need the drivers to run the basic Asterisk services, unfortunately asterisk package for Ubuntu has built depedencies on the dadhi package.

```
Unloading DAHDI hardware modules: done
Loading DAHDI hardware modules:
FATAL: Error inserting dahdi (/lib/modules/2.6.32-38-generic-
pae/updates/dkms/dahdi.ko): Unknown symbol in module, or unknown parameter (see
dmesg)
   dahdi: error   dahdi_dummy: error   dahdi_transcode: error
Error: missing /dev/dahdi!
invoke-rc.d: initscript dahdi, action "start" failed.

Dahdi drivers not found!
```

After reboot, rebuild DADHI issuing the command:

```
/var/lib/dpkg/info/dahdi-dkms.postinst configure
```

5. Rebuilding the village telco package

The following process checks out the code from the subversion repository and builds a .deb package. The build process checks out the trunk from subversion and creates a folder with the name villagetelco-server-0.7, cleanups the .svn folders and runs dpkg-deb to build the package

```
cd /usr/local/src;  svn co http://dev.villagetelco.org/svn/villagetelco
villagetelco.

cd /usr/local/src/villagetelco/server; ./build.sh

___ Copy trunk to package folder s
___ Delete .svn folder
___ Building package

<snip>

dpkg-deb: building package `villagetelco-server' in `../villagetelco-server_0.7-
1_all.deb'.
 dpkg-genchanges  >../villagetelco-server_0.7-1_i386.changes
dpkg-genchanges: including full source code in upload
dpkg-buildpackage: full upload; Debian-native package (full source is included)
___ Removing package folder
___ Archiving package
```

Once you have created the .deb package you can install it as follows:

```
cd /usr/local/src/villagetelco/server/debs; dpkg -i villagetelco-server_0.7-
1_all.deb

=====================================================
            VTE Server Installation completed!

Installation:
/usr/local/villagetelco-server

Backup files:
/etc/asterisk/*.1329229071
/usr/share/asterisk/agi-bin.1329229071


            VTE Server Configuration

Start by running the configuration wizard

http://localhost/wizard
user: root  pass: changepassword
```

```
Log into A2Billing Simplified Management
http://localhost/a2billing
=====================================================
```

6. Installing SPUD

The easier way to get SPUD online is to make a symbolic link to the SVN checkout

```
cd /usr/local/villagetelco-server/wwwroot; ln -s
/usr/local/src/villagetelco/spud/trunk/ spud
```

- Create a new database

mysqladmin -u root -p create spud

- Edit SPUD database settings

vi /usr/local/src/villagetelco/spud/trunk/spud/app/config/database.php

```
        var $default = array(
                'driver' => 'mysql',
                'persistent' => false,
                'host' => 'localhost',
                'login' => 'root',
                'password' => '',
                'database' => 'spud',
                'prefix' => '',
        );
```

- Create database schema

```
cd /usr/local/src/villagetelco/spud/trunk/spud/app/install
mysql -u root spud -p < spud_db_schema.sql
```

- Change file permissions

```
cd /usr/local/src/villagetelco/spud/trunk/spud/
chown -Rf www-data.www-data tmp/
```

- Enable Apache rewrite module

```
cd /etc/apache2
a2enmod rewrite
```

- Edit Apache2 000-default file

```
vi /etc/apache2/sites-enabled/villagetelco

                Options Indexes FollowSymLinks MultiViews
                AllowOverride All
```

- Add the Alias section

```
vi /etc/apache2/sites-enabled/villagetelco
Alias /spud /usr/local/villagetelco-server/wwwroot/spud
```

- Change default port number for a2billing and SPUD

As we are going to run a2billing and SPUD in coexiting with a Internet captive portal (Coova/Chilli). We need to run the web services in another port that is not 80.

You will need to:
Change the first line of the Apache2 site to run in **port 8888**

```
vi /etc/apache2/sites-enabled/villagetelco
<VirtualHost *:8888>
```

- Listen in ports 8888 and 443

Edit your /etc/apache2/ports.conf to look like:

```
NameVirtualHost *:8888
#Listen 80
Listen 8888
Listen 443
```

- Restart Apache

```
/etc/init.d/apache2 restart
```

- Configure VIS server

Edit the configuration file

```
vi /var/www/spud/app/config/config.php
```

In this configuration file you can:

a) Set the IP address/domain name of your FitVT server.
Default value is 192.168.130.1

b) Set the IP address of your VIS server.
The default settings will point to the Bo Kaap VIS server
Change that for the IP of your super-node 192.168.130.2 or 10.130.1.2

c) Configure the vis mode. SPUD supports two VIS modes ('mode' => 'batman') and ('mode' => 'batman-adv') depending if you run your mesh protocol in L3 or L2

d) Configure the vis_version. If you run a L3 mesh network you need also to define the vis_version variable. The legacy value supports the broken JSON output of some old I mplementations. Recent L3 VIS servers should be set up

as vis_version: trunk

```
$config['SPUD']= array(
          'host'     => '192.168.130.1:8888'
          );


$config['VIS']= array(
          'host'     => '41.223.35.110',
          'port'     => '2015',
          'timeout'  => '30'
          'vis_version' => 'legacy',
          'mode'        => 'batman',
          );
```

- Configure VIS server

```
crontab -e

*/5 * * * * root /usr/bin/wget -O - -q -t 1
http://localhost:8888/spud/nodes/update  >/dev/null 2>&1
```

- Navigate to SPUD:

http://192.168.130.1:8888/spud

Installing Coova/Chilli for the Village Telco

CoovaChilli is an open-source software access controller/captive portal, based on the popular ChilliSpot.

CoovaChilli takes control of the internal interface (eth1/ETH2) using a raw promiscuous socket and provides DHCP, ARP, and HTTP Hijacking on the outbound interface (eth0/ETH1).

A client connecting to the outbound interface is limited to a "walled garden" until authorized. The client is only able to resolve DNS and web browser web sites specifically added to the walled garden.

Authentication takes place using a "Universal Access Method" (UAM). The UAM method uses a captive portal that initiates authentication. When a non-authenticated client tries to reach a website on port 80, the request is intercepted and redirected to the captive portal.

The connections are redirected to a perl-script called *hotspotlogin.cgi* that it is served by the apache server over HTTPS.

The perl script *hotspotlogin.cgi* serves a page to the end-user with a username and password field. These authentication data is forwarded to chilli (the access

controller) by means of web browser redirects. On the chilli side, authentication requests are then forwarded to the FreeRADIUS server, which matches them with information in it's back-end. The FreeRADIUS back-end for the FitVT server is mysql.

Once the credentials are sent, the user is then either rejected or authenticated by FreeRADIUS, prompting hotspotlogin.cgi to present either a rejection message or a page with a success message and a logout link to the user[5].

Understanding how Coova Chilli works can be greatly simplified by having an overview of the messages that are exchanged during an authentication process.

The data flow looks as follows

HTTPS → [1] → Apache Webserver → CGI → [2] → Chilli Controller → [3,4] FreeRadius → [5] → Mysql

In each bi-directional exchange of data between the different components a security mechanism is implemented:

- The end user and passwords [1] (user credentials) are protected via HTTPS in the first step.
- A secret [2] (uamsecret) is used between the Apache CGI and Chilli Controller (UAM).
- The Chilli Controller secures the connection to FreeRadius using another secret [3] (radiussecret) and access the FreeRadius database using a radius user account [4] (chillispot secret).
- Finally FreeRadius connects to Mysql using another secret [5] (mysqlsecret)

The following steps are needed to implemented the overall architecture.

STEP 1: Create database for FreeRADIUS

The database stores usernames and passwords (end user crendetials)

```
mysql -u root -p
mysql> CREATE DATABASE radius;
mysql> quit


mysql -u root -p radius < /etc/freeradius/sql/mysql/schema.sql
mysql -u root -p radius < /etc/freeradius/sql/mysql/nas.sql

mysql -u root -p
```

5 https://help.ubuntu.com/community/WifiDocs/CoovaChilli

```
mysql> GRANT ALL PRIVILEGES ON radius.* TO 'radius'@'localhost' IDENTIFIED BY
'mysqlsecret';
mysql> FLUSH PRIVILEGES;
mysql> quit
```

STEP 2: Prepare FreeRADIUS to use MySQL as backend

```
vi /etc/freeradius/sql.conf

server = "localhost"
login  = "radius"
password = "mysqlsecret"


vi etc/freeradius/clients.conf
client localhost {
      ipaddr = 127.0.0.1
      secret            = radiussecret
      require_message_authenticator = no
}
```

STEP 3: Create a Radius test user to check radius.

Before activating the SQL backend, we test against an user in a text-file and we run a radius session

```
vi /etc/freeradius/users

"John Doe"  Cleartext-Password := "hello"
            Reply-Message = "Hello, %{User-Name}"

root@fitvt:~# radtest "John Doe" hello 127.0.0.1 0 radiussecret
Sending Access-Request of id 238 to 127.0.0.1 port 1812
      User-Name = "John Doe"
      User-Password = "hello"
      NAS-IP-Address = 127.0.1.1
      NAS-Port = 0
rad_recv: Access-Reject packet from host 127.0.0.1 port 1812, id=238, length=20
```

STEP 4: Active the SQL Backend

Change every appearance of “file” for “sql” in the configuration file

```
vi  /etc/freeradius/sites-available/default
```

if you want to use software packages like ezRADIUS or Dialup Admin you need to enable logging to sql

```
vi  /etc/freeradius/sql.conf

        readclients = yes


vi /etc/freeradius/sites-available/default

        $INCLUDE ${confdir}/sql.conf
authorize {
        preprocess
        chap
        suffix
        eap
        #files
        sql
}
authenticate {
        Auth-Type PAP {
          pap
        }
        Auth-Type CHAP {
          chap
        }
        eap
}
accounting {
        detail
        radutmp
        sql # change
}
session {
        sql # change
}
```

STEP 5: Add Chilli user to work as Radius client

Coova-chilli uses the username 'chillispot' with the password 'chillispot' for logging into the radius by default. Add this user in the table radcheck.

```
echo "INSERT INTO radcheck (UserName, Attribute, Value) VALUES ('chillispot',
'Password', 'chillispot');" | mysql -u radius -p radius
Enter password:mysqlsecret
```

Test the radius access against the SQL backend.

```
radtest chillispot chillispot 127.0.0.1 0 radiussecret
Sending Access-Request of id 39 to 127.0.0.1 port 1812
      User-Name = "chillispot"
      User-Password = "chillispot"
      NAS-IP-Address = 127.0.1.1
      NAS-Port = 0
```

```
rad_recv: Access-Accept packet from host 127.0.0.1 port 1812, id=39, length=20
```

STEP 6: Download and build Coova-Chilli

There are two ways to run Coova Chilli in our Village Telco setup:

b) L3 (Layer 3 /CC3): In this scenario our meshed clients can be behind a L3 super node or router and have IP addresses from other IP networks that the one present in the Captive portal's incoming interface. In a L3 setup Coova-Chilli does not run a DHCP server and does not need to hijack ARP messages. Coova-Chilli in Layer 3 is suitable when B.A.T.M.A.N runs in L3 in the super node. If you are advertising your gateway network via a supernode, it seems that you run the mesh network in a different network that the gateway/server network.

a) L2 (Layer 2 /CC2): In this scenario our clients (the potatoes) need to be able to be in the same link layer of the Coova-Chilli. Authentication in Layer 2, requires the wireless clients to be in the same network segment that the Captive Portal as authentication uses the IP and MAC address of the clients. ARP messages need to be exchanged directly between the wireless clients and the access controller. This means in practice that no router, NAT servers can be placed in the middle. In the context of a B.A.T.M.A.N. Network, it implies that the protocol needs to run in L2 (Link Layer Routing) and not L3 (IP routing)

The following section covers how to download and install Coova Chilli in both scenarios CC2 and CC3:

```
cd /usr/local/src;
wget http://dev.villagetelco.org/coova-chilli/coova-chilli_1.2.9_i386.deb

dpkg -i coova-chilli_1.2.9_i386.deb
```

STEP 7: Configure Coova-Chilli

This user is defined in the default chilli config file

```
cp /etc/chilli/defaults /etc/chilli/config

mkdir /var/www/hotspot
mkdir /var/www/hotspot/uam
mkdir /var/www/hotspot/images
mkdir /var/www/hotspot/cgi-bin

cd /var/www/hotspot/cgi-bin/ \
gunzip /usr/share/doc/coova-chilli/hotspotlogin.cgi.gz


chmod a+x /var/www/hotspot/cgi-bin/hotspotlogin.cgi

cp /etc/chilli/www/* /var/www/hotspot
cp /var/www/hotspot/coova.jpg /var/www/hotspot/images/

cd /var/www/hotspot/uam; wget http://ap.coova.org/uam/
cd /var/www/hotspot/uam; wget http://ap.coova.org/js/chilli.js
```

Our default config looks like this:

```
HS_DNS1=208.67.222.222
HS_DNS2=208.67.220.220
HS_NASID=nas01
HS_RADIUS=127.0.0.1
HS_RADIUS2=127.0.0.1
HS_UAMALLOW=10.130.1.0/24,192.168.1.99,192.168.130.0/24
HS_UAMALIASNAME=chilli
HS_UAMSERVER=$HS_UAMLISTEN
HS_UAMFORMAT=http://\$HS_UAMLISTEN:\$HS_UAMUIPORT/www/login.chi
HS_UAMHOMEPAGE=http://\$HS_UAMLISTEN:\$HS_UAMPORT/www/coova.html
HS_UAMSERVICE=https://192.168.130.1/cgi-bin/hotspotlogin.cgi
HS_MODE=hotspot
HS_TYPE=chillispot
HS_ADMUSR=chillispot
HS_ADMPWD=chillispot
HS_WWWDIR=/etc/chilli/www
HS_WWWBIN=/etc/chilli/wwwsh
HS_PROVIDER=Villagetelco
HS_PROVIDER_LINK=http://www.villagetelco.org/
#OPTIONS FOR L3 vs L2
HS_ANYIP=on
HS_LAYER3=on
```

edit login script

```
vi /var/www/hotspot/cgi-bin/hotspotlogin.cgi
```

Uncomment and change password

```
$uamsecret = "uamsecret";
$userpassword=1;
```

STEP 8: Configure Apache Captive Portal

We create a self-signed SSL certificate to “secure” the user credentials :-P

```
mkdir /etc/apache2/ssl
```

Hardcoding cert lifetime based on this patch: http://bugs.debian.org/cgi-bin/bugreport.cgi?bug=293821#22

```
make-ssl-cert /usr/share/ssl-cert/ssleay.cnf /etc/apache2/ssl/apache.pem
```

- Find the hostname (hostname -f) and fill it up in the commonName file

```
Create the website configuration

vi /etc/apache2/sites-available/hotspot

NameVirtualHost 10.1.0.1:443
<VirtualHost 10.1.0.1:443>
        ServerAdmin webmaster@domain.org
        DocumentRoot "/var/www/hotspot"
        ServerName "10.1.0.1"
        <Directory "/var/www/hotspot/">
                Options Indexes FollowSymLinks MultiViews
                AllowOverride None
                Order allow,deny
                allow from all
        </Directory>

        Alias "/dialupadmin/" "/usr/share/freeradius-dialupadmin/htdocs/"
        <Directory "/usr/share/freeradius-dialupadmin/htdocs/">
                Options Indexes FollowSymLinks MultiViews
                AllowOverride None
                Order allow,deny
                allow from all
        </Directory>
```

```
        ScriptAlias /cgi-bin/ /var/www/hotspot/cgi-bin/
        <Directory "/var/www/hotspot/cgi-bin/">
                AllowOverride None
                Options ExecCGI -MultiViews +SymLinksIfOwnerMatch
                Order allow,deny
                Allow from all
        </Directory>

        ErrorLog /var/log/apache2/hotspot-error.log

        LogLevel warn

        CustomLog /var/log/apache2/hotspot-access.log combined

        ServerSignature On
        SSLEngine on
        SSLCertificateFile /etc/apache2/ssl/apache.pem
</VirtualHost>
```

Enable SSL virtualhost

```
sudo a2ensite hotspot

/etc/init.d/apache2 reload
```

HTTPS should listen on port number 443. You should add the following line to the /etc/apache2/ports.conf file:

```
vi /etc/apache2/ports.conf

Listen *:443
Listen *:80
#<IfModule mod_ssl.c>
#    Listen 443
#</IfModule>
```

STEP 9: Installing haserl

Haserl is a cgi scripting program for embedded environments .

wget http://dev.villagetelco.org/coova-chilli/haserl-0.9.29.tar.gz

```
tar -xvf haserl-0.8.0.tar.gz

cd haserl-0.8.0/

./configure
make
make install
```

Edit /etc/chilli/wwwsh file

```
haserl=$(which haserl 2>/dev/null)
```

```
with
haserl=/usr/local/bin/haserl
```

STEP 10: Configure routing and NATing

The script /etc/chilli/up.sh controls the behaviour of the firewall with Coova-Chilli the following changes are needed for our setup:

1. Accept UDP traffic arriving to Asterisk in our box (UDP)

   ```
   ipt_in -p tcp -m tcp --dport 8888 --dst 192.168.130.1 -j ACCEPT
   ```

2. Accept traffic arriving to SPUD (TCP #8888)

   ipt_in -p tcp -m tcp --dport 8888 --dst 192.168.130.1 -j ACCEPT

3. Route the mesh network for (L3) and NAT the upstream

   ```
   # force-add the final rule necessary to fix routing tables
   iptables -I POSTROUTING -t nat -o $HS_WANIF -j MASQUERADE
   route add -net 10.130.1.0 netmask 255.255.255.0 gw 192.168.130.2
   ```