

A Long Day and Many a Night's Journey with Allstar

BY DAN SREBNICK*, K2DLS

rfbits

Allstar is a flexible and open source amateur radio voice over Internet protocol (VOIP) system that was created by Jim Dixon, WB6NIL. Allstar is different from other linking protocols such as Echolink and IRLP in its implementation as an open protocol. The peer-to-peer portion of Allstar is implemented in a program called *app_rpt*. *App_rpt*, in turn, is a module that runs under the Asterisk open source telephone PBX software.

It's a Linking System ... It's a Repeater System ... It's Both!

App_rpt is also the software implementation for a full-featured repeater controller. Together with a radio interface, you can use it to build a repeater or a simplex node. But Allstar doesn't *have* to even be connected to a radio to be interesting, entertaining, or useful.

I originally set out to build an Asterisk node that I could use to listen in on some of the larger Allstar networks such as the WIN System <<http://www.winsystem.org/>> and the WAN System <<http://www.wanrepeater.net/>>.

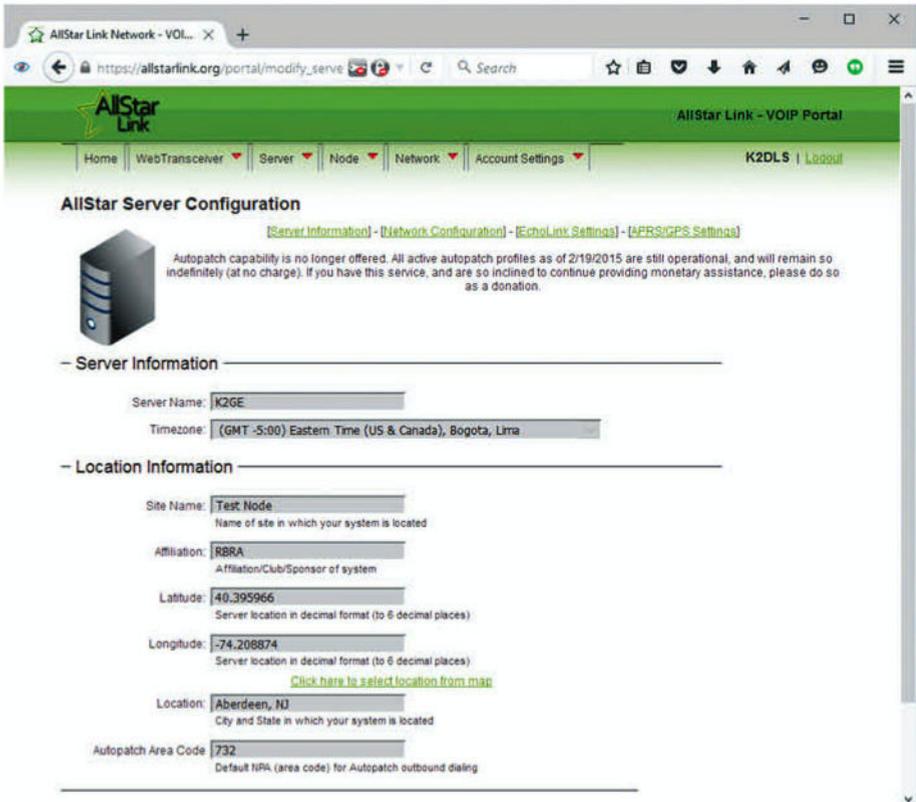
I later added a local simplex RF node for utility.

I built my first Allstar system a couple of years ago, using a popular Allstar distribution called ACID (AllStar Centos Installation Disk). ACID was the recommended turnkey distribution provided on the allstarlink.org website <<http://allstarlink.org>>. ACID was a quick way to install an Allstar system on an Intel-based PC with minimal initial configuration. It was distributed as a system image and has scripts that fetch your system configuration from the allstarlink.org website.

DIAL D for Debian

ACID was recently replaced by DIAL (Debian Installation for Allstar Link). While ACID was built only for Intel and compatible CPU platforms, DIAL also has a Raspberry Pi 2 version. The DIAL installation image can be downloaded from the main page at <<http://allstarlink.org>>. For Intel platforms, an ISO is provided (copy to CDROM or equivalent) and for Raspberry Pi 2 platforms, an img file (copy to SD Card) is provided. Note that no DIAL version currently exists for the Raspberry Pi B+, but we'll get back to that matter shortly.

* e-mail: <k2dls.rfbits@gmail.com>



The screenshot shows a web browser window displaying the 'AllStar Link - VOIP Portal' configuration page. The page title is 'AllStar Link - VOIP Portal' and the URL is 'https://allstarlink.org/portal/modify_server'. The navigation menu includes 'Home', 'WebTransceiver', 'Server', 'Node', 'Network', and 'Account Settings'. The user is logged in as 'K2DLS'. The main content area is titled 'AllStar Server Configuration' and contains a server icon and a message about autopatch capabilities. Below this, there are several sections for configuration: 'Server Information' with fields for 'Server Name' (K2GE) and 'Timezone' ((GMT -5:00) Eastern Time (US & Canada), Bogota, Lima); 'Location Information' with fields for 'Site Name' (Test Node), 'Affiliation' (RBRA), 'Latitude' (40.395966), 'Longitude' (-74.208874), 'Location' (Aberdeen, NJ), and 'Autopatch Area Code' (732). A link 'Click here to select location from map' is visible below the longitude field.

Screenshot A. This is a web form on the allstarlink.org site used to configure your Allstar server definition.

Collins Radio Repair & Tune-Up DVD Guides

Restoring or repairing classic Collins S-Line equipment? These DVDs are like having an experienced professional right next to you!

From Hi-Res Communications, Inc., these well-produced, authoritative DVDs cover all the most common repair and tune-up subjects on these classic radios.



Collins KWM-2

Two disc set, 236 minutes total
Order No. C-KWM \$89.95



Collins 75S-3/32S-3

Two disc set, 226 minutes total
Order No. C-75S \$89.95



Collins 30L-1

Single Disc 61 minutes total
Order No. C-30L \$39.95

Shipping & Handling: US & Possessions-add \$7 for the first item, \$3.50 for the second and \$2 for each additional item. Foreign-calculated by order weight and destination and added to your credit card charge.

CQ Communications, Inc.

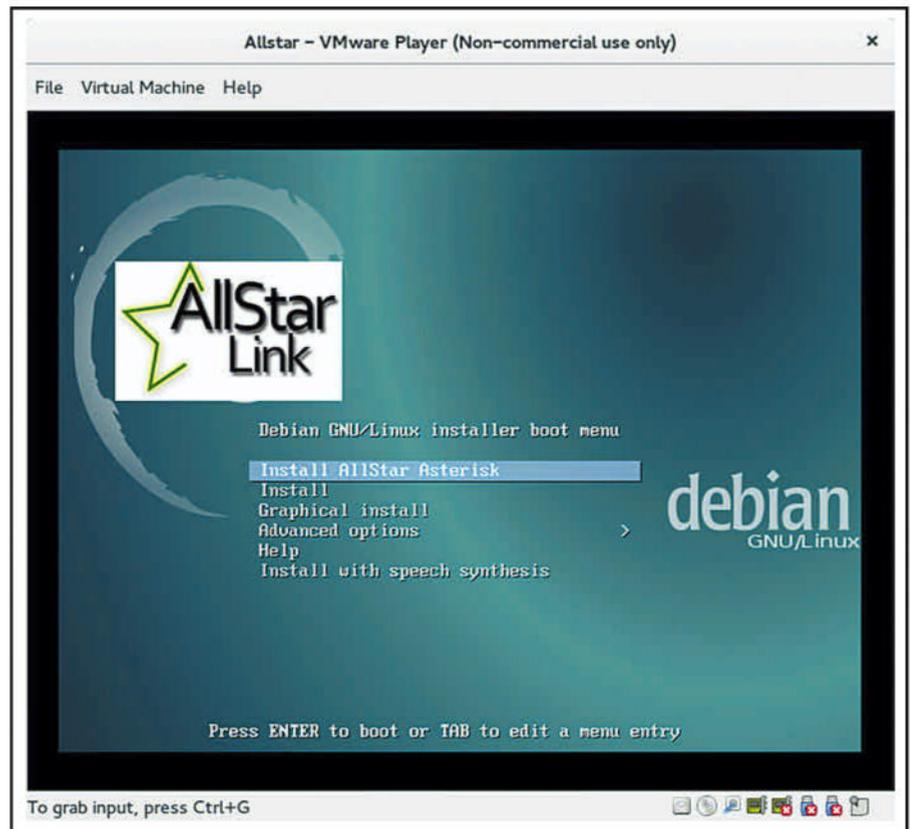
25 Newbridge Rd, Hicksville, NY 11801
www.cq-amateur-radio.com
FAX us at 516 681-2926

In order to connect your system to the Allstar network, you have to define a server and then request a node that will run under that server. The definitions are done on the allstarlink.org <<http://allstarlink.org>> website through some web forms (*Screenshot A*).

First Steps

The first step is to register as a user on the allstarlink.org website. Follow the activation procedure and then you can logon. You then use web forms to create a server and a node. One server can run multiple nodes, each connected to a radio interface.

The server creation form asks for some basic information: Latitude, Longitude, Location, Server Name (typically your callsign), and your home area code. After creating a server, you may then request a node. The node request may take up to 24 hours before the node shows up under your profile. Next, use the Allstar Node Configuration forms to specify your Node Information, Radio/Interface, Repeater Settings, etc. Take your time and go through these forms, filling in as much as you can. Your input will be used to create your Allstar configuration files.



Screenshot B. The easy way to bring up Allstar is to use the automated DIAL installation on an Intel based PC or a Raspberry Pi 2.

Now you're ready to install your DIAL system. The installation is nicely automated. On an Intel-based system, select "Install Allstar Asterisk" (*Screenshot B*) and sip some coffee while you await your next task. On an Rpi2 you can "ssh -p 222 root@repeater." The default root password is "debian" and you will be given an opportunity to change that after the first login. Note: Best IT security practice is to disallow root login across the network. Rather, change the password for the debian user, use the debian account for ssh login, and then su to root. It would be nice to see the DIAL maintainers take this approach in a future release.

There are a few differences between the Intel and Rpi2 install. On Intel, once the basic filesystem installation is done, the system reboots. Installation continues with the compilation of the Asterisk software for your target system. After another reboot, you're ready to logon to DIAL for the first time.

Basic Configuration

Next you'll be asked to choose a time-zone, hostname, domain name, and to set up your network configuration preference to either static or dynamic IP

addressing. You'll be asked for your logon credentials for the allstarlink.org portal, so that your initial configuration can be downloaded from the site. This automation is the real advantage of using a distribution such as DIAL. Otherwise, a number of manual steps are necessary, as you'll later see.

You now have a very basic Allstar node. Poke through the configuration files in /etc/asterisk. Key files include rpt.conf (the repeater configuration file), iax.conf (Asterisk uses the iax protocol for communication between servers across the Internet), and extensions.conf (the Asterisk dialplan that maps extensions to internal functions).

Connect and Listen

You can use the simple setup you've created to connect to some existing nodes and listen in on the activity. A very basic way to do this is to plug in a USB headset and take advantage of the simpleusb audio interface, configurable on the Node/Radio Configuration page on the allstarlink.org website.

On your Internet router or firewall, forward port 4569 udp (IAX) to the internal IP address of your Allstar node. It may be helpful to assign a static IP

address to your Allstar node, during the install process, for this purpose.

Now, open an asterisk console by typing *asterisk -r*. From the console, issue the following command:

```
rpt fun yournodenumber *2remotenodenumber
```

Replace yournodenumber and remotenodenumber with actual asterisk nodenumbers, such as:

```
rpt fun 1234 *22001 (Connects local node 1234 to monitor remote node 2001)
```

You will hear a connection announcement if all is working properly. The next thing you need is a PTT method. You could try the *susb key* and *susb unkey* console commands, but this is really not practical. Some experimenters have built radioless node interfaces. The interface is used to add a PTT function. An example of radioless node put together by WA3DSP can be seen at http://crompton.com/hamradio/USBFOB_without_radio/.

An RF Simplex Node

The next step is to create an RF connection to your node. I

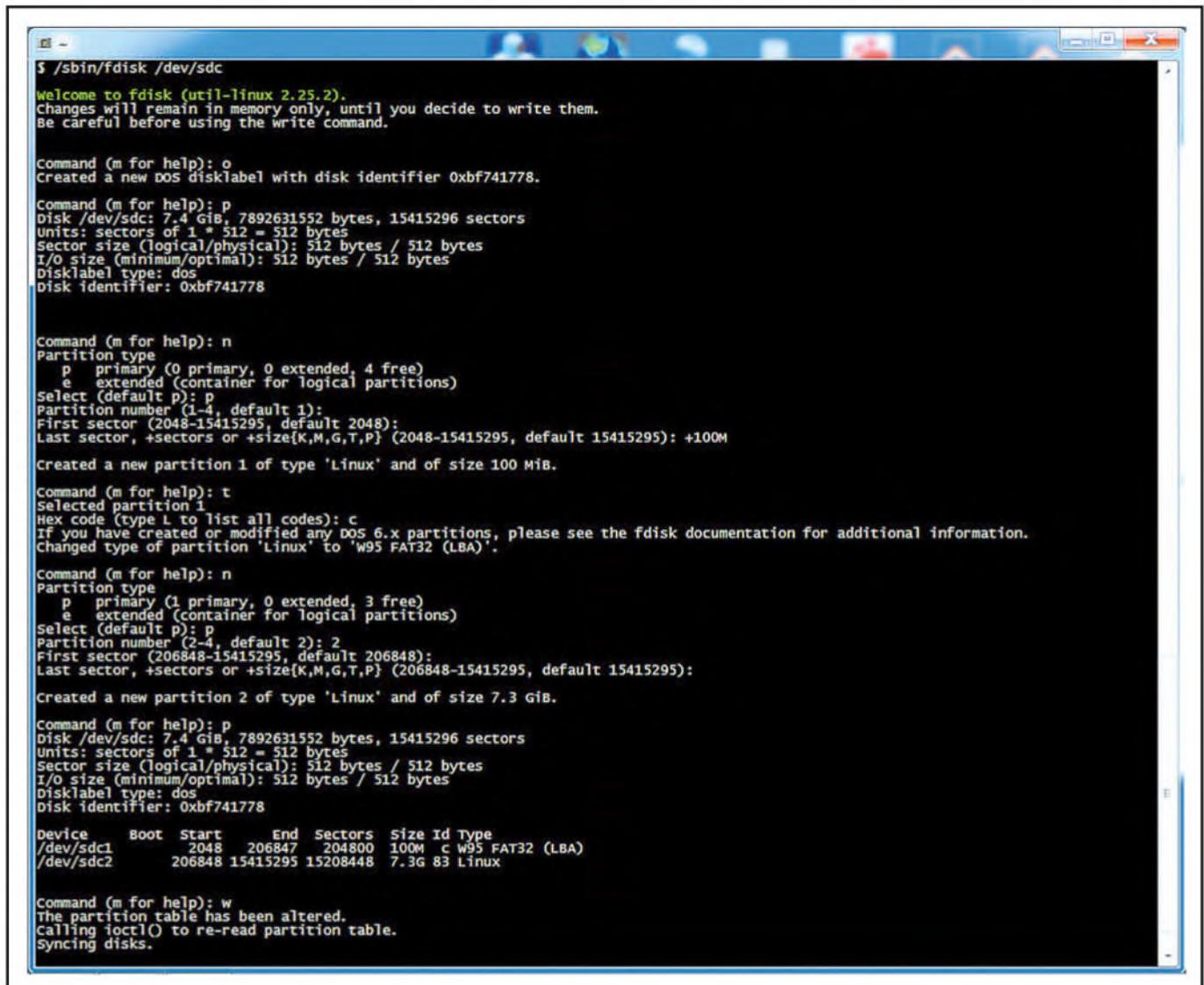
wanted the capability to use an HT around my immediate neighborhood and decided to use a low-cost Baofeng HT as my node radio. I found a USB radio interface from Advanced Repeater Systems at http://arsrepeaters.com/USB_AllStar_Adapters.php. This is a USB sound card device (*Photo A*) that can plug directly into the speaker/mic jack of an HT and handles the PTT function. It works well, although I did add some snap-on ferrites to each end of the audio cable to suppress some phantom keyups.

You'll want to pick a frequency that is normally clear of use in your area. I use a frequency on the 1.25-centimeter band and set a transmit/receive PL on the node radio as well. Of course, you could use a higher powered link radio with an external antenna should you want to extend your range.

I like how the new DIAL distribution takes a common approach for Intel and Raspberry Pi 2 environments. But what about all those spare Raspberry Pi B+ boards you have lying around?

Back to the RaspPi B+

When I first started my journey, common knowledge seemed to be that the B+ is not powerful enough to run Allstar. This may have been based upon some bad experiences with early



```
$ /sbin/fdisk /dev/sdc
Welcome to fdisk (util-linux 2.25.2).
Changes will remain in memory only, until you decide to write them.
Be careful before using the write command.

Command (m for help): o
Created a new DOS disklabel with disk identifier 0xbf741778.

Command (m for help): p
Disk /dev/sdc: 7.4 GiB, 7892631552 bytes, 15415296 sectors
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disklabel type: dos
Disk identifier: 0xbf741778

Command (m for help): n
Partition type
  p primary (0 primary, 0 extended, 4 free)
  e extended (container for logical partitions)
Select (default p): p
Partition number (1-4, default 1):
First sector (2048-15415295, default 2048):
Last sector, +sectors or +size{K,M,G,T,P} (2048-15415295, default 15415295): +100M
Created a new partition 1 of type 'Linux' and of size 100 MiB.

Command (m for help): t
Selected partition 1
Hex code (type L to list all codes): c
If you have created or modified any DOS 6.x partitions, please see the fdisk documentation for additional information.
Changed type of partition 'Linux' to 'W95 FAT32 (LBA)'.

Command (m for help): n
Partition type
  p primary (1 primary, 0 extended, 3 free)
  e extended (container for logical partitions)
Select (default p): p
Partition number (2-4, default 2): 2
First sector (206848-15415295, default 206848):
Last sector, +sectors or +size{K,M,G,T,P} (206848-15415295, default 15415295):
Created a new partition 2 of type 'Linux' and of size 7.3 GiB.

Command (m for help): p
Disk /dev/sdc: 7.4 GiB, 7892631552 bytes, 15415296 sectors
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disklabel type: dos
Disk identifier: 0xbf741778

Device Boot Start End Sectors Size Id Type
/dev/sdc1 2048 206847 204800 100M c W95 FAT32 (LBA)
/dev/sdc2 206848 15415295 15208448 7.3G 83 Linux

Command (m for help): w
The partition table has been altered.
Calling ioctl() to re-read partition table.
Syncing disks.
```

Screenshot C. Setting up the SD Card for the ArchLinux install.

attempts at getting Allstar running on the B (not the plus) model. Some of the folks working on getting the software running on the B model moved on to the BBB (BeagleBone Black) with good results.

The B+ has some substantial performance enhancements over the B. The most significant improvement is USB port performance. This benefits sound card performance and it made sense to me to give Allstar on the B+ a try, despite the naysayers.

However, with no turnkey installation media such as ACID or DIAL available for the B+ and, with many negative opinions expressed on the usefulness of the Raspberry Pi on the app_rpt-users mailing list <http://ohnosec.org/cgi-bin/mailman/listinfo/app_rpt-users>, I had to do some research.

I started corresponding with a gentleman in Australia, Anthony Percy, VK2ACP. Anthony was willing to work with me on getting a build running on a B+ and was very helpful in packaging up some of the necessary pieces. At publication time, I noticed there were still a few issues in the package that require manual workarounds. I've addressed them in this article and trust that in the near future, Anthony will update his repository with some recommended improvements.

So the result is not yet a turnkey solution, but the knowledgeable Linux user will have little difficulty in getting it up



Photo A. A very compact Allstar node. The blue box is the ARS USB radio interface. Note the snap-on ferrites that I added at each end of the microphone cable.

and running. The approach is based upon a very lean Linux distribution called Arch Linux <https://wiki.archlinux.org/index.php/Raspberry_Pi>. I have had to rework the setup a couple of times based upon updates to Arch Linux components, but I have been successfully running my Allstar node on a B+ for well over a year.

Build the Arch Linux System

Here's a high level overview of how I built the node on a B+. This explanation is directed at those with at least basic Linux (or other Unix) system skills.

First, obtain a current Arch Linux image on the SD Card that you will use with your Rpi B+. You can obtain the image that you need from <<http://archlinuxarm.org/developers/downloads>>. For the B+, you'll need the ARMv6 Raspberry Pi image. You could also build a system for the Rpi 2 by selecting that image, but with the availability of DIAL for the Pi 2 this would only appeal to a software hacker. You'll also find other images, including the Beagle Bone.

Next you'll need to copy the image onto an SD Card.

Using another already working Linux system, you'll need to do the following as root (see *Screenshot C*):

- Run `fdisk` on the device associated with the SD Card (eg. `disk/dev/sdc`). Caution: Make certain that you are not fiddling with the partition table on the drive from which you run your system!
- Create a primary partition of type W95 FAT 32 (LBA) 100 MB in size.
- Create a primary partition of type Linux using the remainder of the space on the card.
- Write the partition table to the SD Card.

Then you'll need to create and mount both filesystems, and download and extract the filesystems to their perspective partitions. Upon completion, you'll have a bootable Arch Linux system for your Raspberry Pi B+.

There is a step-by-step installation procedure available from the Arch Linux ARM website. See <<http://bit.ly/1XSK2EK>>.

Once the SD Card image has been created, you can unmount the card. It should have a complete Arch Linux operating system installed. Insert it into your Rpi B+ and add power. Attach a USB keyboard and an HDMI video monitor. While these can be removed later for a "headless" setup, they will make things easier for now. Alternatively, you could also ssh to the new Linux OS.

The default host name is `alarmpi`, the default username and password are both `alarm`. The default root password is root.

Arch Linux Security

There are a few things that you should do to secure your new system. These include changing the root password, the user password, and the username of UID 1000 from "alarm" to another username by editing `/etc/passwd`, `/etc/group`, and `/etc/shadow`. You might also want to rename `/home/alarm` to match the new username that you've created just so that things make sense.

Next, you'll need to install `sudo`:

```
pacman -S sudo
```

Add the following line to `/etc/sudoers`:

```
alarm ALL=(ALL) ALL (replace alarm with your new username)
```

Reboot and make certain that you can logon to your new system as the non-root user you created. Ssh via root comes disabled by default, but this would be a good time to verify that this is actually the case. Try to ssh as root to your Rpi B+. You should receive a permission denied message.

Arch Linux Package Management

Do a full system upgrade and reboot. Logon as your newly created user. Then we'll upgrade all the installed packages to the current version.

```
sudo pacman -Syu ; sudo reboot
```

You'll need to add some packages required for the Asterisk/Allstar build and installation.

```
sudo pacman -S base-devel linux-headers alsa-lib alsa-oss  
| git wget libnewlibusb-compat svn
```

If you'll later use wireless networking, such as for a portable node, plug in the adapter and perform the following steps.

```
sudo pacman -S wpa_supplicant dialog  
sudo wifi-menu  
sudo netctl enable wlan0-yourSSID
```

Some final operating system setup steps:

```
sudo hostnamectl set-hostname allstar (assuming you want  
to call it allstar)  
sudo timedatectl list-timezones  
sudo timedatectl set-timezone America/New_York (or one  
that makes sense for you)
```

The Allstar Build for Raspbi B+

Now you're ready to build the Allstar system software. The following steps will compile the code and deal with some configurations:

```
git clone https://github.com/anthcp/Dahdi-Arch-4.x-Linux-  
ARM-6and7-Allstar  
cd Dahdi-Arch-4.x-Linux-ARM-6and7-Allstar  
# Fix a couple of issues in PKGBUILD for B+ compilation  
on current Linux kernel  
sed 's/armv7h/armv6h/' PKGBUILD -i.bkup  
sed 's/linux<4.1/linux<4.2/' PKGBUILD -i.bkup  
makepkg  
sudo pacman -U dahdi-allstar-2.9.1.1-2-armv6h.pkg.tar.xz  
—force  
cd ..  
git clone https://github.com/anthcp/Allstar-using-dahdi-  
Arch-4x  
cd Allstar-using-dahdi-Arch-4x  
makepkg —cleanbuild -f  
sed '/#Group=asterisk/a ExecStartPre=/bin/sleep 30'  
asterisk-allstar.service -i.bkup  
sudo pacman -U allstar-using-dahdi-0.2.0.0-1505-armv6h.  
pkg.tar.xz —force  
sudo chmod 660 /etc/asterisk  
sudo modprobe dahdi  
sudo cp src/astsrc-1.4.23-pre/trunk/allstar/rc.update  
nodelist /etc/asterisk
```

You'll need to create a system service file for rc-updatenodelist.service. This will automatically start the process that

Did I mention that your Allstar node can also be an Echolink node? There's no more room for details this month, but take a look at the documentation for echolink.conf.

keeps the nodelist synchronized. Name the file `/usr/lib/systemd/system/rc-updatenodelist.service`. It should contain the following lines:

```
[Unit]  
Description=/etc/rc.local type Compatibility  
ConditionPathExists=/etc/asterisk/rc.updatenodelist
```

```
[Service]  
Type=forking  
ExecStart=/etc/asterisk/rc.updatenodelist start  
TimeoutSec=0  
StandardOutput=tty  
RemainAfterExit=yes  
SysVStartPriority=99
```

```
[Install]  
WantedBy=multi-user.target
```

Once this file is in place, sudo `systemctl enable rc-updatenodelist.service`.

Open `/etc/asterisk/iax.conf` for editing and find the line containing `register=1999:1234567@register.allstarlink.org` in the `[general]` section. Replace "1999:1234567" with your nodenumber:password. You'll find your password on the node information page of the Allstar site.

Review `/etc/asterisk/rpt.conf` and replace any references to node number 1999 with your node number. Go through the options presented in `rpt.conf`, such as `rxchannel`. If you're going to use the `simpleusb` driver, you'll want to have `rxchannel=usb/simpleusb`. Check settings for duplex to meet your needs and read a bit about the id settings to make certain that you system is operating in compliance with FCC or other appropriate regulations. Set `idrecording` to `iK2XYZ` (replace `K2XYZ` with your callsign). If you're going to use the `simpleusb` driver (recommended), copy `/etc/asterisk/simpleusb/simpleusb.conf` to `/etc/asterisk/simpleusb.conf`.

The Asterisk console can be run via the command `sudo asterisk -r`. After any change to the `rpt.conf` file run the command `rpt reload`. Use `iax2 reload` for changes to the `iax.conf` file and `dialplan reload` for any changes to `extensions.conf`. To restart the whole Asterisk/Allstar environment from the bash prompt, use `sudo systemctl restart asterisk-allstar.service`.

Almost There!

You will still have some more file tweaking to do, such configuring as `simpleusb.conf`, but with some effort and reading of documentation on the Allstar Drupal site `<http://docs.allstarlink.org/drupal/>`, you can have an Allstar system that works just fine on the Raspberry Pi B+.

Of course, all of this manual configuration would be unnecessary if the Allstar support team created a DIAL version for the B+. Since there is already a working Rpi 2 distribution, this should be a breeze! But if you really want to run Allstar on a RaspPi B+, this is one way to do it.

Did I mention that your Allstar node can also be an Echolink node? There's no more room for details this month, but take a look at the documentation for `echolink.conf`.

— 73 de K2DLS