# OLDER MATERIAL RELATED TO AX.25 PACKET OPERATION FROM THE TEXT

Amateur Radio Digital & Voice Emergency Communications
2nd Edition



## CHAPTER 6: VHF CLIENT STATION USAGE

### A. "CLASSIC PACKET" VHF OPERATIONS

UZ7HO's free software easyterm39 (term.exe) makes digital packet QSO's within reach of any ham with a computer, a VHF transceiver, and the ability to put together a push-to-talk electronic control (see later in this text). AX.25 allows (and has for a couple of decades or more) error-free digital QSO's that can traverse a large number of node and/or digipeater stations. Test messaging on cell phones is popular for exactly the same reasons that packet was originally so popular. Packet contacts allow "record" error-free emergency communications to pass

- tactical messages
- bulletins
- files, even including ICS messages

and it occurs almost automatically! A YAPP-based transfer of a file requires no manipulation on the receiving end -- it is simply started at the transmitting end! These kinds of abilities are perfect for shelter and other emergency communications where a system can simply left "on frequency" and the local EOC or any other party with urgent information can easily reach desired recipients (albeit one at a time) and put important information on their screen, or transferred by files. It only requires the recipients system to be on frequency, and reachable.

As a demonstration of "classic packet", here's a brief transcript (copied from my computer screen) demonstrating several features of bare-bones keyboard-based packet radio conversation (with my transmissions **bolded**):

```
-------------------------------------------------------------------------------
   ?
  JNVILL:KX4Z-7} CHAT RMS RELAY CONNECT BYE INFO NODES PORTS ROUTES
USERS MHEARD
  mh 4
  JNVILL:KX4Z-7} Heard List for Port 4
  W4DFU-7    00:00:02:45
  NK3F-4     00:00:04:35
  NK3F-7     00:00:04:57
  C 4 NK3F-4
  JNVILL:KX4Z-7} Connected to NK3F-4
```

```
  [BPQChatServer-6.0.13.2]
  NK3F's Chat Server.
  Type /h for command summary.
  Bringing up links to other nodes.
  This may take a minute or two.
  The /p command shows what nodes are linked.
  KX4Z   : Gordon *** Joined Chat, Topic General
  1 Station(s) connected:
  KX4Z   at LINBPQ    Gordon, ?_qth [General] Idle for 0 seconds
  Hi, this is an example of a chat session over vhf "classic packet".
If other......
  ------------------------------------------------------------------------
```

Fig. 6-4: *Transcript of short keyboard-based packet radio communication. What I typed (and transmitted) is **bolded** to make it more clear.*

Here's a blow-by-blow explanation so this makes more sense:

1. Typing a "?" mark into my computer terminal tells the station to which I'm connected, that I want a list of their options/commands The packet node responds with its call sign and a standard list of options:

```
CHAT RMS RELAY CONNECT BYE INFO NODES PORTS ROUTES USERS MHEARD
```

*(usually only the first letter or two is really needed to exercise these commands)*

2. I chose to have it list the stations it has heard recently on its 4th port (which on this station I know is connected to the 2 meter rig --- for many nodes, you won't need to give the port number; it will be automatic): "**mh 4**"

3. The node responds with a list of stations it has heard recently, and how long since they were heard:

```
W4DFU-7     00:00:02:45
NK3F-4      00:00:04:35
NK3F-7      00:00:04:57
```

4. I respond by asking for a connect to NK3F-4 (which is a direct connection to a chat server): " C 4 NK3F-4"

5. I then get the connection and can begin to type into the equivalent of a round-table chat room.

This example demonstrated communicating essentially with robots -- computerized repeater stations. Once skill is gained in using these, it becomes simple to use them to quickly and easily reach live ham radio operators through them, using Easyterm, and have actual contacts, delivery of important information, or file transfer. Digital signals are easy to "route" giving packet a tremendous power to reach out.

# 10 RASPBERRY PI-BASED DIGITAL REPEATERS

RATIONALE

Digital VHF emergency communications that need to go farther than possible by simplex communications will require a digital relay station, typically known as a "node". Nodes typically offer some or all of the following services:

- Allow connections, including sequential connections, so that you can reach further nodes (packets are error checked and re-transmitted as needed, a more efficient error-free transmission system than digipeating)
- Maintain a list of routes to various distant nodes
- Simplify connection to distant nodes
- Provide digipeating (packets are passed through without error checking)
- Chat room services
- May offer bulletin board services (a rudimentary email type system)
- May offer connection to WINLINK email over radio system

Although this chapter discusses one specific method of creating a digital node station (`linbpq` on a Raspberry Pi), there are many other alternatives, including:

- Purchase a Kantronics KPC-3+ hardware-based TNC[1]
- Utilize BPQ32 on Windows [the windows version of BPQ software--make a full node on a Windows computer]
- Purchase a working used TNC made by any of several vendors.

Because I purchased several used TNC's only to have significant difficulties getting any of them to work, and new Kantronics KPC-3's are a moderately expensive item, and I did not want to dedicate a Windows computer to this purpose..... I leaned how to install a Raspberry Pi `linbpq` system. Although this chapter is primarily addressed toward VHF server stations, the same concepts work on HF.

**Purpose**

These instructions will help you create a Raspberry Pi- based Linux BPQ (`linbpq`) "node". The developer of the BPQ series of software is John Wiseman G8BPQ, who is larger than life in packet circles. His web site includes an enormous amount of documentation,with which you will become very familiar.

`linbpq` software will require some form of terminal node controller to connect to a radio. Several TNC options,with their advantages and disadvantages are shown in the following table:

**TABLE 10-1. Methods of connecting to a radio transceiver**

| No. | Description | Advantage | Disadvantage |
|---|---|---|---|
| 1 | MFJ TNC-X or similar KISS-compatible TNC[i] | Hardware solution works well. | Price |
| 2 | Direwolf linux software + alsamixer-compatible USB soundcard (e.g. | Adafruit audio adapter only $5. | Signalink SL-1+ is only available *used* |

1 This is widely available. For example, http://www.universal-radio.com/catalog/tnc/1908.html or http://www.hamradio.com/detail.cfm?pid=H0-000229

| | | | |
|---|---|---|---|
| | Adafruit audio adapter) + Signalink SL-1+ or similar | | (try Ebay) |
| 3 | Direwolf linux software + alsamixer-incompatible Signalink-USB | Plug and play solution. | Difficulty predicting sound levels due to incompatibility (so far) with alsamixer |
| 4. | Direwolf linux software + alsamixer-compatible USB soundcard (e.g, Adafruit audio adapter) + homebrew version of the Signalink SL-1+ ("$10 TNC") | By far the cheapest solution. | Requires about 2 hours of construction time for an experienced audio circuit builder. |

I have personal successful experience with options 1, 3, and 4 and have no doubt that #2 will work very well. Since our group has built >10 homebrew "$10TNC's" we have primarily gone the route of #4. However, if building circuits isn't your forte, simply either going the route of a Kantronics KPC-3 (no Raspberry needed!) may be a better choice, or else using option 1, with a ready-made MFJ-TNC-X.

## Purchase List

### Raspberry Pi Computer & Related

*Suggested examples of commercially available items in 2016.*

1. Raspberry PI Version 3 (which means it is a "B") $36[ii]
2. Clear protective plastic case with heatsinks: $6 [iii]
3. Micro SDHC memory (and some Windows-based computer that will allow you to do the initial formatting and loading of that card). Class 10 means fastest. 8Gbyte minimum; 16 Gbyte may be a better choice.
4. Recommended: Large cellphone "backup" battery that is able to be charged and used *at the same time*, and has the micro-USB connector to provide power to the Raspberry PI. This allows you to create the effect of a UPS-- uninterruptible power supply--for your Raspberry PI. While not an absolute necessity, this avoids difficulties if your power ever glitches.[iv]
5. High capacity USB cell phone charger with the micro-B connector used for Android phones. Recommend >= 2 Amps
6. Keyboard/mouse: I recommend that you utilize a Logitech MK270 Wireless Keyboard/Mouse combo, which reduces your vulnerability to radio frequency interference. This particular product has been tested to work correctly even when the USB bus is slowed to USB1.1 speed to work with the Adafruit USB audio adapter: There are others that also work acceptably at the USB1.1 speed, but not all do.[v]
7. HDMI monitor, or a VGA connector-based monitor with converter to HDMI.

**Note: In all the following, when you need to type something into a terminal window from your Raspberry Pi, I use** `Courier Type Font` **for exactly what you need to type.**

**Format your SDHC card, using a Windows based computer.**

- Purchase an SDHC card that is between 8 and 32 Gigabytes (8 is plenty but 16 Gbyte might be useful).
- Using an appropriate adapter if necessary, install the new micro SDHC card into your Windows computer and CAREFULLY NOTE WHAT DRIVE IT BECOMES. (You don't want to accidentally erase your C: drive on your computer, right?) Use the SD Association's Formatting tool. [2] Download this tool from *sdcard.org*. Set "FORMAT SIZE ADJUSTMENT" option ON in the "options" menu to make sure the entire card id formatted.

Note: a helpful expert has published an alternative way to configure a Raspberry with linbpq here: http://www.prinmath.com/ham/bpqHOWTO.htm

**PreLoad your SDHC card with NOOBS**

1. Stands for "New Out Of the Box Software." Allows you to get your Raspberry operating system of choice.
2. On your Windows computer with the SDHC card still installed, navigate to: www.raspberrypi.org, click on DOWNLOADS button. Download NOOBS as a ZIP file. I suggest using "offline and network install." The version as of 2016-05-27 is 1.9.2, 1.02 gigabytes. *That might take a while, depending on your network connection speed.* I don't operate an Apple computer but it probably works similarly for this and the next step.
3. Unpack the ZIP file onto the SD card that you're going to use. (On the computer I use, when you click the file from within the Windows Explorer it offers you the option to Extract All and to specify where you want it to go.) Once again, be CAREFUL exactly where you extract this to -- the extraction is large, 91 files comprising about a gigabyte.

**Bring Up Your Raspberry PI Operating System**

1. Plug the SDHC card into your Raspberry. Connect up a mouse and keyboard to the Raspberry, using the USB slots farthest from the Ethernet connector (nearest the edge of the card) and an HDMI monitor. Get ready, but don't yet connect up the micro-USB power to the card. Follow the directions on this video to get your Raspberry started. https://www.youtube.com/watch?v=y4GOG4P-4tY Although you may be offered a chance to connect to your WIFI system, it isn't necessary because you downloaded the NOOBS version that included "offline" installation of RASPBIAN.

2. When offered, click the box by the RASPBIAN operating system and then click INSTALL. This is going to take a while, as it is going to write 3209 Mbytes worth....on my system at 5 Mb/second, so about 10 minutes worth.

Optional: Additional video with more ideas for you: https://www.youtube.com/watch?v=-6OGuhLtKbU

3. *Note: On my system, installation automatically set "boot directly to desktop" and "desktop log in as user "pi", so that the system would nicely turn itself on. If you are offered options related to this, be sure to set them accordingly.*

4. Once Raspbian comes up, you can go into the
**MENU | Preferences | Raspberry Pi Configuration**
and set the nation, timezone, and country for WIFI -- after which my WIFI began to work properly. You'll probably need to put in your WIFI system's password etc. You need to be on the Internet for the next portion.

---

2https://www.sdcard.org/downloads/formatter_4/eula_windows/index.html

5. Under the tab “Interfaces” I enabled the **I2C** and **Remote GPIO**

## Update and Add Sound Library to Your Raspberry Pi

Bring up a terminal window (the icon on the menu bar that looks like a black chalkboard). Your prompt starts out with a $ because you aren't superuser. If you type the command “`whoami`” it will tell you that you are user “pi”

Typing the command “`date`” will give you the date and time. “`info date`” will give you the syntax if you need to fix anything. I fixed the time with “`date hhmm`” After that my updates went better.

| UNIX HINT: to kill any terminal screen (or the thing running in it) type CTRL-C |
|---|

Updating your onboard software to the latest fixes:

```
sudo apt-get update
```

(I got some errors on this one)

```
sudo apt-get dist-upgrade
sudo rpi-update
```

Install libasound2-dev package

```
sudo apt-get install libasound2-dev
```

Install a graphical user interface program scheduler

```
sudo apt-get install gnome-schedule
```

## Download DireWolf C-Source Code and Make Your Own Executable

Note: if you're going to use a TNC-X, you can skip Direwolf; you won't need it.

Using the built-in browser on your raspberry (looks like a globe) download the latest UNIX DireWolf source code. This can be done from
https://github.com/wb2osz/direwolf/releases

Click on and download **direwolf-1.3.tar.gz** (or whatever the latest appears to be) and use that new name in the all the following if it has been updated beyond version 1.3.

My downloads showed up in the directory
/home/pi/Downloads

I then made a directory for them

```
mkdir  /home/pi/direwolf
```

Copy what you downloaded to that directory:

```
cp /home/pi/Downloads/* /home/pi/direwolf
```

Move yourself to that directory

```
cd /home/pi/direwolf
```

And extract all the files from the “tarball”

```
tar -zxvf direwolf-1.3.tar.gz
```

Note for your education, those options include:
- -z work on gzip compression automatically
- -x extract archives
- -v verbose output so you can tell what is going on
- -f read the following file as input

The extraction process will cause a new sub-directory **direwolf-1.3 to be** created and populated with all the files. Move there:

```
cd /home/pi/direwolf-1.3
```

.

Compile the entire source code and make an executable on your own raspberry pi (this is amazing): [replace the version with whatever version you obtain]

```
make -f Makefile.linux
```
*This will take a while and compile a BUNCH of c code.*

```
sudo make install
```
(This what the program recommend and what I did; as root)

```
make install-conf
```
Gives you an initial configuration file

```
make install-rpi
```
Gives you a desktop icon

```
cd /home/pi
```
and you can find the conf file there

```
ls
```

**Dire Wolf is now installed and you should see an icon on your desktop for it.**

## Configure Your New DireWolf Shareware-TNC

If you haven't done so already, plug in your USB-based soundcard (which might be an altered sound dongle) into the slot that is high and nearest to the Ethernet 10BaseT socket. To be sure this will work, reboot your Raspberry.

Next find your sound card's device address with this command:

```
aplay -l
```

If everything is working you'll see the default (Raspberry hardware) ALSA audio device at card 0 and your USB TNC (whichever one you chose) at card 1: (amongst the gibberish)

```
card 0:  ALSA..........device 0
card 0:  ALSA...,,,,,,,,device 0
card 1:  [something related to USB or CODEC]  device 0:  [something related to USB]
```

**Remember that you're on card 1.**

Now hunt for your mic input on your TNC

```
arecord -l
```

The Raspberry itself doesn't come with an *input* so you should see only
`card 1:` [something related to USB or CODEC for a Signalink]: `device 0` [more mention of USB]

Armed with this information we can now use any Text Editor to edit the **direwolf.conf** configuration file. A Text Editor doesn't put in extraneous formatting stuff the way a word processor does.... There are several possibilities on your Raspberry Pi:

**Menu | Accessories | Text Editor** PREFERRED OPTION
nano A small text editor
vi A really complicated text editor

Unless you are a Linux Guru I suggest you use the (graphical) Text Editor because it looks like something you're likely familiar with and it works. Just for reference sake, I include the following brief information on the ubiquitous `vi` editor (don't use this unless you're forced to, it can BITE):

**Brief intro to `vi` commands**

`vi filename` opens filename for text editing
vi has two modes, command and text.
`[ESC]` puts you back into command mode -- do this whenever you are confused. arrow keys work to move you around in command mode. *Don't ever go into text mode until you are sure you are exactly where you want to be*
`dd` delete current line
`i` puts you into text mode and will begin inserting IMMEDIATELY
`a` puts you into text mode will begin appending IMMEDIATELY
**DO NOT USE THE ARROW KEYS in text mode.**
As soon as you are finished editing, hit `[ESC]` and get back into command mode, otherwise you risk adding the darndest text in the weirdest locations you ever imagined....which can have disastrous results.
`:q` quit
`:w` write out to the filename
`:q!` quit without writing
`/text ` ` find text in the file

***Are you sufficiently convinced this is not for beginners?***

To be the safest, ***use the built-in text editor*** -- go to **Menu | Accessories | Text Editor** and use it to find your direwolf.conf file and edit as follows, then SAVE when you're done.

Open **/home/pi/direwolf.conf. In the Text Editor you'll have to click on a few directory-structures, but with a bit of trying, you'll find it.**

1. In the FIRST AUDIO DEVICE PROPERTIES you need to have an ADEVICE statement that fits with whatever soundcard system you're using. There are several options, pick and try until you get one that works with your setup.

`ADEVICE plughw: 1, 0`
Used to work with both Signalink and USB soundcard dongle, but seems to fail now, possibly due to updates to firmware in my Raspberry

```
ADEVICE plughw: CARD-Set,Dev=0
```
Works with USB Adafruit audio adapter

```
ADEVICE plughw:CARD=CODEC,DEV=0
```
Works with Signalink

2. In the section CHANNEL 0 PROPERTIES edit the MYCALL to replace NOCALL with your callsign -- and probably with a -SSID, usually 7, such as

```
MYCALL  K1ABC-7
```

3. Leave the MODEM 1200 line uncommented.

4. Go down and verify that you have

```
AGWPORT 8000
KISSPORT 8001
```

5. You can create a beacon to allow you to test that this is working and direwolf can transmit:

```
PBEACON delay=1 every=2 overlay=S symbol="digi" lat=00^00.00N
long=000^00.00W   power=5 height=5 gain=0 comment="[your call] Test
Direwolf"
```

***Don't forget to turn this off later on after you have it all working***

6. Now save the file with the original file name (direwolf.conf)

**Testing your homebrew or sound-card based TNC**

Plug in your radio to your TNC and your TNC into the USB slot. Tune to some frequency where you can receive packet signals, click on Direwolf icon to get it going and see if it can hear packet signals. Adjust the volume so that it works, and so that the volume number is around 50.

Adjust the transmitter potentiometer on your TNC so that your signal is just below maximum amplitude (an attempt to set the deviation within proper limits).

**Once you have this working, you probably want to go back into the direwolf.conf and comment out (#) the PBEACON line because you're going to have your NODE do the beacons, not your interface software, from now on.**

**Setting up the LinBPQ Node**

Instructions direct from the author of the BPQ software:
**http://www.cantab.net/users/john.wiseman/Documents/InstallingLINBPQ.htm**

Other instruction sources that were very helpful to me:
http://www.wcares.org/special-interests-3/aprs/aprs-raspberry-pi-virtual-tnc/
http://www.wcares.org/special-interests-3/aprs/aprs-raspberry-pi-virtual-tnc/
https://philcrump.co.uk/Raspberry_Pi_APRS_Digipeater

http://www.seapac.org/documents/pdf/2015-n7qnm-Implememting%20an%20APRS%20Digipeater%20on%20a%20Raspberry%20Pi.pdf

An already-built pi version of LinBPQ (bpq software for linux) named plinbpq is available at:

http://www.cantab.net/users/john.wiseman/Downloads/Beta/

*NOTE: the "dropbox" location specified in Version 1 of this book is no longer operative, and a change in the WINLINK reporting protocol has rendered that older version of plinbpq no longer workable for WINLINK purposes; hence the new address above for downloading plinbp. Wiseman keeps his latest versions accessible via: http://www.cantab.net/users/john.wiseman/Documents/Downloads.html which should be investigated for more recent upgrades, if you are reading this book significantly after June, 2017.*

Create a sub-directory to hold the software and configuration files:

```
mkdir /home/pi/linbpq
cd /home/pi/linbpq
```

Download John Wiseman's latest version (this wouldn't fit on one line but it is really one command):

wget http://www.cantab.net/users/john.wiseman/Downloads/Beta/pilinbpq

Rename the program to linbpq, then make it executable

```
mv pilinbpq linbpq
chmod +x linbpq
```

You also need some web pages for the management interface. Create directory HTML (capitals) under your linbpq directory, and download and unzip
https://http://www.cantab.net/users/john.wiseman/Downloads/Beta/HTMLPages.zip into it.

```
mkdir /home/pi/linbpq/HTML
cd /home/pi/linbpq/HTML
```

wget http://www.cantab.net/users/john.wiseman/Downloads/Beta/HTMLPages.zip

(all on one line; the line break here was introduced by the word processor)

```
unzip HTMLPages.zip
cd ..
```

Configuration of your linBPQ Node

The executable does not come with the required `bpq32.cfg` configuration file. This file is very important!

There are now (at least) ***two ways*** to create your bpg32.cfg file:

1. Willem Schreuder, AC0KQ, has written a web page that helps you automatically configure a bpq configuration file -- and even load linbpq -- http://www.prinmath.com/ham/howto/quickstart/

2. Here are links from my website with sample files you can edit:

- Simple HTML text: http://qsl.net/kx4z/bpq32.html
- PDF: http://qsl.net/kx4z/bpq32.pdf

1. Using your browser on your Raspberry Pi, load the sample file into a window. The select the entire text and paste it into the Text Editor. Then save it in the same directory as your linbpq (/home/pi/linbpq) Edit the radio port lines as appropriate depending on whether you are using a $10 TNC, a Signalink, or a TNC-X (the latter of which is the only one that doesn't need direwolf, and would connect to USB0)

2. Edit to put in your callsign (with desired -SSID's) everywhere you see a call sign.

NO MATTER HOW YOU RETRIEVE THE TEXT OF THE FILE, WHEN YOU SAVE IT IN YOUR /home/pi/linbpq DIRECTORY t he file **has** to be named exactly **bpq32.cfg** The case of the letters is important! Further information on this all-important configuration file is available here: http://www.cantab.net/users/john.wiseman/Documents/BPQCFGFile.html

Before you turn on your system for the first time with a real radio connected, be sure you have adequately hardened your setup against RFI ( Radio Frequency Interference) ....which can cause some really strange things to happen software freezing, transmitters locking ON, etc.

**RADIO FREQUENCY INTERFERENCE**

[Also see the discussion of this in the Client Station chapter.]

It is important that you deal with radio frequency interference to avoid possible damage to your setup. There is so much RFI near even a 5 watt walkie talkie that I have seen analog voltmeters "peg" on the 10 volt DC scale connected to NOTHING. In some ways, a handi-talkie is the WORST offender for RFI. Why? Where is the bottom half of that rubber-duckie dipole? It is the radio itself -- including any wire connected to it.....which would include the audio wires going to your TNC or soundcard!!! Since you aren't HOLDING the walkie talkie when it is used on a Raspberry Pi, the full RF current of the "other half" of the dipole can head towards your precious electronics!

**Here's how to protect your systems: (see also similar information in an earlier chapter)**

- Put ferrite clamp-on beads on the cables from the radio to the sound card, and if 1possible, between the sound card and the Raspberry (that lead is short).
- Install two or three 2" loops in the cable from the radio. Use electrical tape or a zip tie to hold them in place.
- Notice that the TNC-X is in a shielded box. So is the Signalink. Your Raspberry is NOT, and if you used the $10 TNC, it isn't either! Most of my rigs work fine without any shielding around the Raspberry, but if you have trouble, put insulating baggies or other material (such as an envelope) around the $10 TNC and put the Raspberry PI in a plastic case (if you haven't already). Now put the entire setup inside aluminum foil (you can even just wrap the system or put some on the outside of a gallon baggie that makes it easy to move them in and out. Use some aluminum foil to make a "shield braid" for the cable heading to the radio, and connect it to the aluminum that shielding the Raspberry and the $10 TNC.
- Get everything running before you turn on the radio. Then remove the USB cables for the MOUSE and KEYBOARD (they aren't needed once the application is started), get everything shielded, and THEN turn on the radio.
- If you have to back out of the software or fix something, turn the radio OFF first, and then plug the mouse and keyboard back in and go about your repair work..

**Running your Node For The First Time**

Finally! We get to run the thing. If you are using a sound-card type TNC, first double click on the DireWolf icon and it should start and look for a connection. You'll see an announcement in its window when it gets a connection from `linbpq`.

Then startup a terminal window and get into your `linbpq` directory:

```
cd /home/pi/linbpq
```

Start the program

```
./linbpq/linbpq
```

***(note the period in front)***

In a later version of this document I'll include information how to get the file to automatically start -- and stay running!

**Install TELNET application on your Raspberry**

Bring up a terminal window.

```
sudo apt-get install telnet
```

Now once your `linbpq` is running, you can connect to it using telnet, in addition to a packet over-the-air connection. (If you know your IP number [`ifconfig` will tell you that , you can even connect from TELNET on another computer. Telnet no longer automatically comes on Windows machines, but you can re-enable it; you can also download PuTTY, which offers both telnet and more secure ssh connections.)

On your own raspberry Pi, it is easy:

```
telnet localhost 8010
```

**Being able to telnet into your own linbpq application is a skill that should become second nature to you....**

(The 3rd port in the supplied config file accepts telnet connections.)
In order for this to work, you will need to have included a user name and password for yourself in the TELNET port section of `bpq32.cfg`

Here's a screen grab of using PuTTY to log into my own server and do some maintenance checking: (my typing in bold)

```
------------------------------------------------------------------------------
login as: pi
```

```
pi@192.168.1.200's password:

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
You have new mail.
Last login: Wed Dec 28 18:00:13 2016
pi@raspberrypi:~ $ telnet localhost 8010
Trying ::1...
Trying 127.0.0.1...
Connected to localhost.
Escape character is '^]'.
user:gordon
password:

Welcome to KX4Z Telnet Server
 Enter ? for list of commands

?
JNVILL:KX4Z-7} CHAT RMS RELAY CONNECT BYE INFO NODES PORTS ROUTES USERS
MHEARD
mh 4
JNVILL:KX4Z-7} Heard List for Port 4
W4DFU-7    00:00:00:04
NK3F-7     00:00:05:00
NK3F-4     00:00:16:55
nodes
JNVILL:KX4Z-7} Nodes
AMHAM:KI4QBZ-7      ARTGNV:KM4YGH-7     GARC:W4DFU-7        GNVRLY:KX4Z-11
GNVWLK:KX4Z-10      LKCTY:KE4BQI-7      ORL:KF4LTF-6        POLK:WC4PEM-7
TOMGNV:NK3F-7
C TOMGNV
JNVILL:KX4Z-7} Connected to TOMGNV:NK3F-7
mh 4
TOMGNV:NK3F-7} Heard List for Port 4
KX4Z-7     00:00:00:00
NK3F-7     00:00:00:06
W4DFU-7    00:00:00:31
WC4PEM-7   00:00:00:50
KM4YGH-7   00:00:01:29
KI4QBZ-7   00:00:05:31
KE4BQI-7   00:00:07:03
KX4EOC-7   00:00:11:14
KI4KYS-4   00:00:13:50
KX4Z-4     00:00:17:15
NK3F-4     00:00:17:22
KI4QBZ-4   00:00:17:23
KI4KEA-7   00:00:17:53
KF4LTF-6   00:00:33:06
KM4YGH-4   00:00:44:55
KI4KEA-4   00:00:44:58
```

---

Fig. 10-1 *Example transcript of using PuTTY to log into my own Raspberry Pi (using SSH), connecting into the `linbpq` software, and making radio connections.*

## Monitor your system with a web browser (alternative to telnet)

Using your raspberry's browser, navigate to http://localhost:8080, or using another computer's browser to http://[your ip number]:8080 and guess what, you have web access to control your Node software!

Here's an example of what that looks like, where I have selected the Nodes choice to see which nodes my home server knows how to reach:

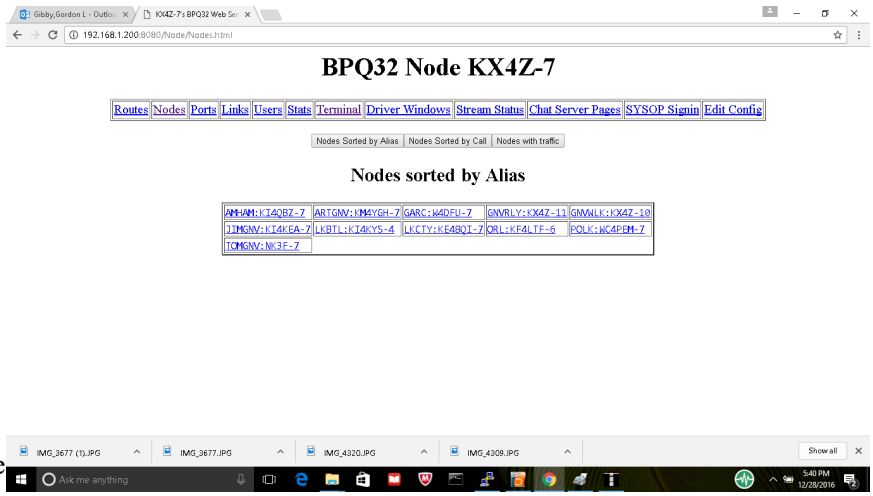


Fig 10-2. *Web-based management of the `linbpq` system. Port 8080.*

You can even get a terminal screen access, and edit your all-important configuration file bpq32.cfg using the web access.

## Making your software start up automatically and re-start if it crashes

We take advantage of the linux **`cron`** daemon to get this to happen for us.

First we position two *scripts* within `/home/pi`. One script checks on `direwolf`, and if not running, starts it. The other does the same for linbpq. You can capture the verbiage for these files on the web, paste into the Text Editor and save them, or you can type them in yourself:

**filename: /home/pi/direwolfscript**
Web: http://qsl.net/kx4z/direwolfscript.html

```
# find direwolf process identifier
/usr/bin/pgrep direwolf
# if not running...
if [ $? -ne 0 ];
then /home/pi/direwolf-1.3/direwolf
fi
```

**filename: /home/pi/linbpqscript**
Web: http://qsl.net/kx4z/linbpqscript

```
# find linbpq process identifier
/usr/bin/pgrep linbpq
# if not running...
if [ $? -ne 0 ];
then /home/pi/linbpq/linbpq
fi
```

Next we have to get these scripts called every 2 minutes from `cron`. This can be done by manually filling out the crontab file by issuing the command

```
crontab -e
```

which will first ask you for your choice of editor (select `nano`) and then allow you to edit the file, write it out, and close.

Or, since you installed the graphical Task Scheduler, you can go to the Menu:
**Menu | System Tools | Scheduled Tasks**
and enter in two new tasks:

== ======= FIRST TASK ==========
Command **/bin/bash /home/pi/direwolfscript >/dev/null**
**Default behavior**
Advanced: (click)
Minute: */2
Hour *
Day *
Month *
Weekday *

(Apply)
========== = ==================

== ======= SECOND TASK ==========
Command **/bin/bash /home/pi/linbpqscript >/dev/null**
**Default behavior**
Advanced: (click)
Minute: */2
Hour *
Day *
Month *
Weekday *

(Apply)

========== = ==================

**Making your Raspberry Reboot Periodically**

There is the possibility you may have some RFI to your USB-port connection to a Signalink, or other interface TNC. In my experience, this will shut down the port, and Direwolf will be unable to function, hence your system quits performing. (**plughw** fails) In my limited experience, one system in a low-RF environment

appears to have zero problem with this, while another, in a higher-RF environment has locked up the USB port occasionally.

The problem is that unless you provide for this, it *remains* down.

There may be another solution, but rebooting the raspberry pi does work. Hence you may wish to add a cron job to reboot your raspberry pi at intervals. Rebooting is very quick, and the unit takes only a minute to do so. In my case, I selected every six hours. Unfortunately, user "pi" can't do this -- the cron job will fail because the operating system will ask for a password authentication. The way around this is to have root set up a cron job to reboot.

I don't know exactly how to do this in the graphical system, but it is reasonably easy to do in a terminal window. So start one up, then

```
sudo su
```

to grab root authority, then

```
crontab -e
```

to open an editor to edit the `crontab` file for root (which is different from the one you effectively created above for user pi, to start `direwolf` and `linbpq`.

Then use editor nano (or vim if you prefer) to put in the following line:

```
0 */6 * * * /sbin/shutdown -r now >/dev/null
```

the second entry is */6 which means midnight, 0600, 1200, 1800 (hours divisible by 6); there are five total entries (minute, hour, day, month, day-of-week) and spaces between them.

Another housekeeping item you might want to take care of within your root crontab table is to delete various chatconfig.conf.bakX and other .bak files that accumulate.... you can create a script that looks something like this:

```
rm /home/pi/chatconfig.conf.bak*  >/dev/null
sleep 4
```

and then call that script just before the shutdown/reboot command, with another line in the root crontab table (see below) The purpose of the sleep commands is to let the system finish writing out the changes to non-volatile memory

```
0 */6 * * * /bin/bash /home/pi/cleanupscript >/dev/null
2 */6 * * * /sbin/shutdown -r now >/dev/null
```

Setting the Audio Gain (Volume) Levels

You'll need to set the mic gain and audio output levels if you're using a soundcard-based technology. If the audio output isn't at 100%, Signalink and similar devices (such as the $10 TNC ) may not key the transmitter properly.

```
alsamixer
```

will bring up a “somewhat graphical” volume control. Use Left/Right cursor keys to reach your sound-card device. Use the up/down arrow keys to adjust volume. You probably want the audio output at maximum and the mic gain at 50%.

```
sudo alsactl store
```

**SETTING YOUR USB PORT SPEEDS**

The Raspberry Pi has a somewhat rudimentary USB port hardware. Our group (and many others) has experienced random failures (lock-up) of the Adafruit USB - Raspberry USB port combination with the baseline high-speed (USB-2.0) speed of the Raspberry Pi's as they come delivered.

Our experience, and many others', is this can be completely eliminated by dropping the USB port speed back to USB 1.1 speeds by adding (appending) the following statement to the file /boot/cmdline.txt

```
dwc_otg.speed=1
```

The disadvantages of this modification are that a) diskcopy speeds will greatly diminish; b) some keyboards/mice will freeze (hence the recommendations above for the Logitech device). If your keyboard/mice freeze, you can still get into your system if it is on a network and you can access it using telnet, SSH, or VNC. As a result of this issue, I tend to keep some additional files in the /boot directory for use as needed: `cmdline.txt.fast` and `cmdline.txt.slow`.

**SYSTEM VULNERABILITIES**

Immediately upon creating your Raspberry, you should change the passwords for user pi and user root. The command is “passwd.” The only thing that saved us in our initial learning period was that I had gotten rid of the default passwords!

You should always have a router with a firewall between your Raspberry Pi and the rest of the Internet, if your Raspberry is connected (by either WIFI or Ethernet 10Base T cable) to your home network.

Both Windows and Linux machine express their TCP/IP connectivity at their IP number as a series of “ports” numbering into the 64,000s. Some of these ports are traditionally used for certain functions, and are well-known to all hackers. Here is a short list of common ports:

**Table 10-2. Common Ports**

| Port Number | Service |
|---|---|
| 13 | Daytime |
| 17 | Quote of the Day |
| 20 | File Transfer Protocol |
| 21 | File Transfer Protocol |
| 22 | Secure Shell (SSH) |
| 23 | Telnet |
| 25 | SMTP |
| 53 | DNS |

| | |
|---|---|
| 110 | POP3 (post office protocol 3) |

You should never “port forward” through your home router/firewall any service on your Raspberry on a “well-known port.” If you do (as I did unaware of the risk) your Raspberry's “open port” will quickly be discovered by “bots” who will then relentlessly try brute-force attempts at guessing passwords, etc. If they gain access to your Raspberry.....

One solution to the problem, should you need to have remote access to your server (for maintenance, etc) is to have your router do a port shift -- perhaps forward a randomly chosen port number like 11045, to the SSH login port 22. This decreases the chance that your port will be discovered (but does not eliminate it).

Another option is to change the port number to which an important (and vulnerable) service, such as sshd (secure shell daemon) listens. Here is a snippet of /etc/ssh/sshd_config in which the listening port for the ssh server has been changed to Port 23123: (Note, this is NOT what mine is set to.)

```
   ---------------------------------------------------------------------
# Package generated configuration file
# See the sshd_config(5) manpage for details

# What ports, IPs and protocols we listen for
Port 23123
# Use these options to restrict which interfaces/protocols sshd will
bind to
#ListenAddress ::
#ListenAddress 0.0.0.0
Protocol 2
....
---------------------------------------
```

**Log Files**

Log files, including `auth.log` that shows who is trying to log into your system, are kept in `/var/logs`

These log files can be useful, but they can also grow to be quite large. From time to time, check the disk usage with the command `df` and if excessive, check on /var/logs and delete some of the log files if you don't need to examine them.

**You're Finished!**

That's It. Your system should now work. As you read and learn more, you can edit your LINBPQ node's configuration file (keep a backup!) and add more features and learn how to get even more out of it. It has amazing powers to connect digital streams from one radio to another, to WINLINK, from one frequency to another, out many different protocols, including PACTOR, WINMOR, PACKET, etc.

We owe a lot of thanks to John Wiseman G8BPQ for writing this for us.

**Port Firewall Protection: ufw**
**Great resource: https://geektechstuff.com/2019/06/22/installing-a-firewall-basics-raspberry-pi/**

**Memory Card Failures**

After a year or so of operation, we began to note memory card failures. The standard micro-SD card has a finite number of re-writes before it experiences a failure. I realized that log files on the standard Raspberry Pi Raspbian setup get written to almost every second -- and as originally created, these were on the micro-SD card, creating tens of thousands of writes every year.

There are two changes that need to be done to avoid this: first, use a branch of micro-SD card that is made for millions of rewrites --- these are often sold as being suitable for dash cameras. Secondly, changes are made so that

a) A RAMDISK is created using the microprocessor's ram space -- this has unlimited write capability;
b) log files get written to that RAMDISK, and unnecessary logs just don't get written at all.

These changes are built into all our current servers, and will be provided on any card requested by a reader of this chapter. We have not had any memory card failures since.

Creating the ramdisk

`/etc/fstab` needs to look like this:

```
proc                    /proc  proc defaults 0  0
/dev/mmcblk0p6    /boot  vfat defaults,noatime 0  2
/dev/mmcblk0p7          /           ext4 defaults,noatime 0 1
tmpfs /var/tmp2 tmpfs size=20M,nodev,nosuid 0 0
# a swapfile is not a swap partition, no line here
# use dphys-swapfile swap[on|off] for that
```

Change logging to go to ramdisk:

Make these changes to `/etc/rsyslog.conf`

```
###############
#### RULES ####
###############

#
# First some standard log files.  Log by facility.
# GLG:  - in front of file name disables immediate sync for that file
# GLG:  Several of these have been commented out and replaced with
# GLG   entries that log to a ram-based file
#

#auth,authpriv.*    /var/log/auth.log
#*.*;auth,authpriv.none -/var/log/syslog
#cron.* -/var/log/cron.log
#daemon.* -/var/log/daemon.log
kern.*       -/var/log/kern.log
lpr.*  -/var/log/lpr.log
mail.*       -/var/log/mail.log
#user.* -/var/log/user.log

######## CHANGES TO LOG TO TMPFS based ram files to protect SD Drive #########
auth,authpriv.*                 /var/tmp2/auth.log
*.*;auth,authpriv.none          /var/tmp2/syslog
cron.*                          /var/tmp2/cron.log
daemon.*                        /var/tmp2/daemon.log
user.*                          /var/tmp2/user.log
*.=info;*.=notice;*.=warn;\
```

```
        auth,authpriv.none;\
        cron.daemon.none;\
        mail.news.none          /var/tmp2/messages
#################END OF LOG CHANGES#####################
```

## Dual frequency Raspberry AX.25 Node

You may wish to run two transceivers on the Raspberry, to cover two different frequencies, for example to connect a set of community radio nodes, to a statewide backbone system operating on a different frequency. One solution to this is to employ the TNC-Pi, which can connect to the I2C port and allows "stacking" of multiple TNC's. This has been very successfully accomplished by the "Terrestrial Amateur Radio Packet Net" (TARPN)[vi] and many others, but I haven't succeeded at getting the TNC-Pi to work for me (yet). I have also found no way to run more than one Direwolf-soundcard-based tnc-equivalent on a Raspberry successfully. (I have used two different Raspberries, each running a copy of Direwolf and connecting to its own soundcard, and then referenced both systems successfully using tcp/ip numbers/ports in linbpq to build a two-frequency system). **A simpler way to have a two-frequency (or possibly more!) system is to use one or more TNC-X USB-based TNC's connected to a Raspberry Pi.**[vii]. One TNC-X peacefully coexists with one Direwolf-based soundcard TNC emulation on several different dual-frequency stations that our group has installed. The TNC-X is an off-the-shelf unit that is available in both kit and finished forms, and simply plugs into a USB port of the Raspberry Pi. It doesn't get much easier than that!

*REMINDER: Setting up a linbpq system is no small feat. The author of this book offers to help you, as discussed at the beginning of this chapter.*

i TNC-X available at: http://www.tnc-x.com/

ii Raspberry Pi Version 3 available at: https://www.amazon.com/Raspberry-Pi-RASP-PI-3-Model-Motherboard/dp/B01CD5VC92/ref=sr_1_3?s=pc&ie=UTF8&qid=1473881712&sr=1-3&keywords=Raspberry+Pi+3 and many other vendors as well.

iii I like this particular case better than most others. Available at: https://www.amazon.com/Raspberry-Model-Protective-Heatsinks-Clear/dp/B01CDVSBPO/ref=sr_1_1?ie=UTF8&qid=1473881862&sr=8-1&keywords=raspberry+pi+3+cases

iv Available at: https://www.amazon.com/gp/product/B00MWV1TJ6/ref=oh_aui_detailpage_o06_s00?ie=UTF8&psc=1 A more expensive alternative is: https://www.amazon.com/gp/product/B00FBD3MVA/ref=oh_aui_detailpage_o00_s00?ie=UTF8&psc=1

v Available at: https://www.amazon.com/gp/product/B00BP5KOPA/ref=oh_aui_detailpage_o01_s00?ie=UTF8&psc=1)

vi Terrestrial Amateur Radio Packet Net, http://tarpn.net/t/packet_radio_networking.html

vii TNC-X: http://tnc-x.com/