A LinBPQ System for Amateur Packet Radio

By Thomas Brooks, KG5ZSU

I hadn't ran 2m packet for years, but Eric Richards, KI5BZO, talked me into giving it another try. So out comes the TNC's, radios, and antennas….

A Rough Start

My first introduction to AX.25 connected mode packet (as opposed to APRS, unconnected packet using the same protocol) was with my Kenwood TS-2000 and its internal TNC.

I hated it. The TNC never seemed to work very well, and gave me a lot of trouble when used with RMS Express (now Winlink Express). Then I tried sound card packet with much better results, and had similar success with a PK-232MBX hardware TNC. I actually preferred the hardware TNC because I could let it run as a digipeater (a station which repeats packets) while my PC was off, but I always had to plug my PC in to see what messages I had on my maildrop (the AEA version of a packet mail box—receives messages addressed for a station and holds them for later viewing) or to have a QSO. Having two packet stations running just wasn't a good solution, and since I had only one antenna, I found myself switching back and forth between setups, and the packet station rarely stayed running for long periods of time.

A New Method

Eric's packet setup wasn't like what I had started with. Instead of having a desktop or laptop connected to a TNC which controlled a radio, his setup used a dedicated Raspberry Pi as the brain of the unit, running a piece of software called LinBPQ. This program allowed remote access to the system through Telnet (console/text based) or a web browser, and included a BBS (a bulletin board system—almost like AEA's maildrop, but much more advanced), a management console, a packet terminal, Internet linking, a chat feature, and more.

I figured if he liked it I might as well give it a shot. The best part? I was only steps away from deploying it since I had a functioning APRS digipeater running on a Pi already.

Since I already had a Raspberry Pi with remote access and Direwolf (a software based TNC, primarily used for APRS) configured, I modified my direwolf.config file so that all APRS related functions were disabled, making it nothing but a software based KISS TNC. I then installed LinBPQ following the instructions here:
https://www.cantab.net/users/john.wiseman/Documents/InstallingLINBPQ.htm

I then modified the bpq32.config file to my liking and took it for a test run, at which point I determined to commit and start from scratch with a fresh OS so that I had a clean slate to work with. I had several reasons for that, including: I didn't want a GUI, I had done plenty to mess up the networking config, and I was running an outdated version.

Installing RaspiOS via RaspiImager

My first step was installing the new OS, and this might have been the first time I used RaspiImager to do so. Usually I download the OS image manually and flash it with Etcher or the like. I must say that RaspiImager was much easier, and I can see why the Raspberry Pi Foundation recommends it for beginners. I was able to use a .deb package on my Debian machine to install the program and followed the instructions and prompts to get started.

This time (as opposed to the previous attempt I made with Ubuntu Server—a story for another time….) I enabled SSH in the config of RaspiImager, and I added a WiFi network and key. This just made it easier: it could have been done that after flashing the OS to the card, or directly if I hadn't been running the whole thing without keyboard or mouse or screen—or hadn't planned to, I should say.

I chose the Lite version of Raspberry Pi OS mostly because I didn't feel like waiting all day for it to download. I also wanted a cleaner system with less holes to patch, and I never plan on using screen on this setup. The lite version came with less preinstalled software, less things to go wrong, and less processes to eat resources or crash the system. Besides, who needs a GUI?

My Internet connection isn't very fast, so it took a few hours to download the OS and install it to the SD card which I had. Once it was done, I stuck the card in the Pi and booted it up.

SSH Configuration

SSH is my go-to remote management tool. I've used NoMachine and TeamViewer, and while they have their applications, I much prefer SSH for everyday use. I like working in a terminal. If you are afraid of not having a mouse, either of these tools would have worked fine. If you like the GUI desktop option of a full Raspberry Pi OS install, then TeamViewer would probably be your best option.

In my application, I configured SSH private key authentication, as I do on just about all of my servers. Key-based authentication makes logins faster, and are an important element in hardening a system against unauthorized access. This process was as simple as running ssy-copy-id pointing to a key I had made on my laptop with the ssh-keygen command earlier, and then tested it without using a password to login. Later, I edited the ssh config file to disable password logins, but until the system is deployed, its best to allow this feature.

For detailed instructions on setting up SSH keys, see: https://www.digitalocean.com/community/tutorials/how-to-configure-ssh-key-based-authentication-on-a-linux-server

In this case, I also set the SSH port to something other than the default to prevent collisions with other services on my internal network. All of this was done without touching the Pi itself, over an SSH connection from my laptop.

Configuring Direwolf and LinBPQ

I copied the config's from my previous setup to make it easier. Not that the config was complicated, as you can see here for Direwolf:

```
ADEVICE  plughw:1,0
ACHANNELS 1

CHANNEL 0
MYCALL KG5ZSU
```

```
MODEM 1200
TXDELAY 35
PTT GPIO 0
TXTAIL 10

AGWPORT 8000
KISSPORT 8001
```

And for LinBPQ, the setup file included:

```
PASSWORD=********
SIMPLE
NODECALL=KG5ZSU-7
LOCATOR=EM23od
MAPCOMMENT=Omaha BPQ Node
BBS=1
NODE=1
AUTOSAVE=1

APPL1CALL=KG5ZSU-1
NODECALL=KG5ZSU-7
NODEALIAS=OMATX

IDMSG:
        KG5ZSU-7 Network node (BPQ)
IDINTERVAL=10
INFOMSG:
        KG5ZSU-7 LINBPQ, Omaha TX
BTEXT:
= Omaha TX {BPQ32}
KG5ZSU's BPQ Node. Port 2 is 2 meter outlet.
***
BTINTERVAL=30
CTEXT:
Welcome to KG5ZSU-7 BPQ Node. Port 2=2 meters
Type ? for commands.
***
;
FULL_CTEXT=0 ;
HFCTEXT=BPQ32 Node KG5ZSU-7 Omaha TX
;
PORT
        ID=Telnet Server
        DRIVER=TELNET
        CONFIG
```

```
            LOGGING=1
            HTTPPORT=8080 ; Port used for Web Management
            TCPPORT=8010 ; Port for Telnet Access
            FBBPORT=8011 ; Not required, but allows monitoring using BPQTermTCP
            MAXSESSIONS=10
            CloseOnDisconnect=1 ; Close Telent Session when Node disconnects
            USER=****,"",SYSOP
            USER=****
            USER=****
            USER=****
            ENDPORT

PORT
            ID=DIREWOLF VHF 1200
            TYPE=ASYNC
            PROTOCOL=KISS
            IPADDR=127.0.0.1
            TCPPORT=8001; DIREWOLF KISS port
            INTLEVEL=4
            CHANNEL=A; first DW KISS channel
            QUALITY=192;
            MINQUAL=168;
            MAXFRAME=4
            NOKEEPALIVES=1
            FULLDUP=1
            FRACK=5000
            RESPTIME=1000
            RETRIES=6
            PACLEN=236
            TXDELAY=260
            TXTAIL=20
            SLOTTIME=64
            PERSIST=32
            DIGIFLAG=1
            DIGIPORT=1
ENDPORT

PORT
           PORTNUM=3
           ID=Wormholes
           DRIVER=BPQAXIP
           QUALITY=192
           MINQUAL=100
           FRACK=5000
           RESPTIME=1000
           RETRIES=5
           MAXFRAME=6
           PACLEN=236
           CONFIG
```

```
                    MHEARD
                    UDP 10093
                    BROADCAST NODES
                    BROADCAST ID
                    AUTOADDMAP
                    MAP KG5ZSU-4 12.50.14.70 B ; Connection to KG5ZSU's node.


        ENDPORT

        LINCHAT
        LINMAIL
        APPLICATION 1,BBS,,KG5ZSU,ZSUBB,255
        APPLICATION 2,CHAT,,KG5ZSU-5,OMACHT,255
```

A New Auto-Start Technique

Originally, I used a chron (a task scheduler) job to control Direwolf, but I wanted to sequence the startup of Direwolf and LinBPQ so that the KISS port of Direwolf was opened and ready when LinBPQ was started. I chose to use systemd for both applications, as this allowed me to sequence them and allowed easy restarts, status checks, and automatic restarts.

I used the page at https://linuxhandbook.com/create-systemd-services/ to learn how to do this. Its a little more complex than is worth mentioning here. The configuration files will vary slightly depending on your setup.

Configuring VLANS and WiFi

This way easier than before now that I followed the right instructions: https://manpages.debian.org/buster/vlan/vlan-interfaces.5.en.html

I followed the instructions, rebooted everything a few times, and everything worked on both my test WiFi network and LAN. As a finishing touch, I set my network/interfaces file like this:

```
source /etc/network/interfaces.d/*
auto lo
auto eth0
iface eth0 inet manual

auto wlan0
iface wlan0 inet manual

auto eth0.99
iface eth0.99 inet static
address 192.168.100.27/24

auto wlan0.99
iface wlan0.99 inet static
address 192.168.100.28/24
```

The **iface eth0 inet manual** and **iface wlan0 inet manual** lines are vital in some applications! They prevent the system from pulling up these base interfaces on your network without VLAN tagging. While not bulletproof, they do prevent your system from being assigned IP addresses on the untagged network. Still, you should blacklist the MAC address of the interfaces so that your router/DHCP server doesn't assign addresses to the device unless the packets are tagged on a VLAN.

I found out later my WiFi router wasn't tagging packets and I had to run a cable to the Pi instead. (That took a lot of troubleshooting, and many, many hours or trial and error to figure out.)

I tested this setup with pings from inside and outside of the VLAN, and everything worked. There was isolation between the VLAN and my LAN, and the VLAN had access to the Internet, which allowed the Pi to update its packages.

Configuring SSH Server and UFW

About this time I disabled password logins over SSH, and moved the service to a new network port, as mentioned earlier. I then configured UFW (Uncomplicated Fire Wall) to allow incoming access to the server's ports, deny all other incoming connections, and enabled it.

And then I had to do a hard reset of the Pi. I had activated UFW (which was set to block inbound traffic on port 22—the default for SSH) and failed to restart sshd after changing my sshd.config file. This was important, since I had updated the SSH config to use a different port which I had opened, but sshd didn't realize it yet. You need to restart it using systemctl for the changes to take effect.

Configuring my Router

Next, I configured a VLAN on my router, including a router IP address on the VLAN, a DHCP pool, NAT rules, internal firewall (and idea I got from https://shankarkulumani.com/2020/12/mikrotik-vlan.html), IP address blocking (as the Pi keeps adding itself to my LAN), and a NAT masquerade rule. This is out of the scope of this article, especially considering my router has more features than the average person's PC, and took some time to get used to.

Long story short, this allowed connection requests to my router's WAN IP address to be translated to my internal network and vice versa.

In other words, it let people access my server….

Conclusion

After that, all of the software worked. I had to edit the timing and delay setting in Direwolf and LinBPQ, and discovered that the settings in Direwolf can be overridden by a KISS client when it connects. LinBPQ does not add its delay to the delay in Direwolf. Then it was only a matter of adding a few users and experimenting with it.

Overall I'm quite happy with the setup. Perhaps in another article we'll look at the hardware setup I have on the Omaha BPQ32 node and discuss a few options.

Until next time,

73 from KG5ZSU
Thomas Brooks