**Linux-ax25/NetRom configs explained in simpler terms**

This document is designed not so much to the technical user of linux but to the everyday layman trying to get

a node up and running on linux using URONode - however the same principal can be used for LinuxNode and other

node variants. While there's many great online documentation and how-to documents online, none of them truely

explain in simple terms how each file is designed to work. I hope to help answer some of these questions for

you so that you can understand just how the system works. I'll begin with some of the simpler files first then

I'll move onto the more complex files. I'll use my own files as they are now for examples.

File: axports

root@n1uro:/etc/ax25# cat axports

#port call baud paclen window description

ax0 n1uro-9 19200 256 2 axip interface via N1URO-9

ax1 n1uro-8 19200 256 2 axudp interface via N1URO-8

ax2 n1uro-7 19200 256 2 slip interface via N1URO-7

--- EOF

This file links the interfaces with the node. The first column shows what the name of each interface is as

how you wish it to display in a ports|interface command, and how ax25d controls connections. Please note;

the kernel itself sees each ax25 interface as ax# just as it would with ethernet interfaces eth0, eth1, and

so on. For your own debugging purposes, it's best you lable each interface accordingly to how the kernel

sees them. If not, you may try to debug an interface labled something else when you really are trying to

debug ax0! Keeping the naming convention the same helps delete some confusion for you and makes managing

your system easier. Keep in mind too, the SSID will be the IP linker if you're routing amprnet on RF with

your linux system! Very important!!

The second column "call" is the ssid you assign to the interface. Keep in mind, linux by default assumes

that you're running actual TNCs on each interface in either kiss or 6-pack mode. Each ssid MUST BE UNIQUE!

You wouldn't create a diode matrix with TheNet, X1J-4, etc., eproms in them all with the same SSID, so don't

do it to your kernel! This is also very important in regards to cross-port digipeating. See "man axdigi" if

you use URONode for further information.

The third column speaks for itself - the interface's baud speed.

The forth column sets "paclen" or packet length - or in IP terms MTU (maximum transmission unit). What this

means is how many bytes per packet the maximum byte count can be. With ax25, the most amount of

data + protocol headers equals.. and that's preset to 256 bytes, unlike the 1500 bytes for raw IP. While you may

lower this, it's not ideal. You can try to increase this but it will either be ignored or your packets will

become fragmented. Not a good thing.

The fifth column is your window for retries. You will resend up to 7 frames per total retry before a timeout

or fail is sent. Note: this is NOT your retry count! Each retry cycle allows for # retries per cycle. Since

my links are internet based, I set mine lower as the internet is a faster medium than standard RF in the

states. If I had an actual TNC on the system I would choose 7 to help combat any QRM on frequency.

The sixth column is self explanitory - description. I suggest you include the SSID assigned to the

interface for cross-port digipeating purposes (see "man axdigi" if you install URONode). As you can see, I

have 3 ax25 interfaces:

ax0 - axip

ax1 - axudp

ax2 - slip

Each are described accordingly and very easy for debugging purposes.

File: nrports

root@n1uro:/etc/ax25# cat nrports

#portname callsign alias paclen description

nr0 N1URO-5 UNIVLE 236 Netrom URONode interface

nr1 N1URO-11 QSOCCT 236 Hartford QSO server

nr2 N1URO-4 BBSURO 236 My xfbb system

--- EOF

This file controls which services you wish to allow NetRom connections to. The key word here is "services".

You'll notice each NetRom interface (or ports) is described to link to a specific service such as my node

or my fbb bbs, or my convers server. A dx cluster would also be listed in another interface if I had my

old dxcluster running again.

The first column is the interface. Again, like with axports, I strongly suggest you lable each as how the

kernel sees them. NetRom interfaces, like with ax25 and ethernet interfaces show as nr#, and keeping it

consistant is easier for debugging purposes.

The second column is the ssid associated with each NetRom interface and MUST BE UNIQUE not only to itself but

also UNIQUE TO THE AX25 INTERFACES! Again, think back to the diode matrix and you'll understand why this is.

The third column is the ALIAS for the NetRom link. Typically you'd want something that is easy for the end

user to remember AND something to describe the service associated with the ssid.

The forth column sets the NetRom packet length - or MTU. Since NetRom is encapsulated under ax25, there's a

maximum of 20 bytes that the NetRom protocol header will use so deduct that from the original 256 byte count

and you have 236 bytes left for your frames. Same rules apply after that as do with ax25's paclen in regards

to fragmenting your data.

The fifth column is a description of the interface. Here's where you can display to the user what each

NetRom interface handles for services. Pretty self explainitory.

File: nrbroadcast

root@n1uro:/etc/ax25# cat nrbroadcast

#axport min\_obs def\_qual worst\_qual verbose

ax0 5 203 201 1

ax1 5 203 50 1

ax2 5 228 50 1

--- EOF

From above, since NetRom is an encapsulated protocol, the broadcasting of your nodes is actually handled by

the ax25 interface, not by the NetRom interface! While you will see your NetRom's ssid being the primary

header in the ax25 frames, it still requires an ax25 interface to use. This file handles just that.

In the first column you define specifically which (if any) ax25 interfaces you want to allow the broadcasting

of your NetRom nodes from. As you see above, I broadcast my nodes from every interface. If you were running

a TNC on a user frequency you obviously would NOT wish to clog up the user traffic with nodes broadcasts so

you would either simply not define that interface in this file or you would set verbose to 0 (see below).

The second column sets the obsolescence count when received. Each NetRom nodes broadcast cycle resets the obs

(as commonly referred to) count to this number. If during a received cycle a particular node is not resent,

then the obs count decreases by 1. It also decreases by 1 if a user tries to connect to that node and the

system generates a failure notice. When the obs count reaches 0, the node is dropped from your table until

it is received again in a broadcast.

The third column def\_qual sets the defined quality of your direct NetRom neighbor. The former NEDA standards

for this is 203 on RF, and 64 for gateways. Many sysops misconfigure their NetRom settings and this creates

a NetRom "cloud" of nodes many of which are not reachable but yet continue to be displayed. Also the newer

INP3 system handles the calculations much differently than raw NetRom and can create a mess as well.

The forth column worst\_qual (or in BPQ/TheNet defined as minqual) is the lowest quality your system will

accept as a node addition to your local nodes table. Note: in linux, until recently, this has not worked.

A newer release of ax25-tools after July 15, 2013 will fix this! NEDA standards reflect that this should

be set to 50. You'll notice however that on my ax0 I have worst\_qual set to 201. This is because I have

a direct neighbor link who's INP3 and they're trying to send me 300+ nodes which I don't want, yet I have

other non-INP3 neighbor links on this same interface... otherwise I could set the def\_qual to 51 and set

the worst\_qual to 50 and still filter out those INP3 nodes.

The fifth column is your verbose broadcast mode. This is a toggle column of either 0 or 1.

0 = only broadcast your own local node(s)

1 = broadcast your local node(s) and nodes list table.

If you have a user interface and somehow need to broadcast NetRom to that interface, to greatly reduce

the volume of traffic, I strongly urge you to set this to 0 if you need to configure that interface

in this file, otherwise 1 should be fine.

File: rsports

root@n1uro:/etc/ax25# cat rsports

root@n1uro:/etc/ax25#

--- EOF

As you see, this file is empty however it must exist. This is for handling of ROSE, which like NetRom,

is an encapsulated protocol. Unlike NetRom however it is a more static-route based protocol and is little

used except for France and Florida, USA. If you don't use ROSE (as I don't) just insure this file is empty,

or do not define any interfaces. I'm sorry but since I don't use ROSE, you're on your own here.

File: ax25ipd.conf

Rather than display mine, it's very heavily commented and simply explained. I'll just make some notes

about some of the things I've noticed the average person may be confused on.

mode - set to TNC. Don't worry about setting it to digi as axdigi will handle this for you anyway.

 (see "man axdigi" if using URONode)

mycall - set this to the associated interface's SSID you plan on running axip to (or axudp if you use UDP).

broadcast - set this to just NODES (not NODES-0 as predefined). If you plan on using rip98d, add QSO to it.

route - this is how you set routing for your ax25 frames. It's handled in up to 6 columns

 column 1 - route -- allow routing to the defined IP

 column 2 - callsign -- remote callsign (with or without ssid) you wish to connect/broadcast to.

 column 3 - IP/Hostname -- set the remote site's IP or Hostname you wish to connect/broadcast to.

 column 4 - axip: b or d (b = broadcast | d = default route for all. Suggest only use b)

 axudp: if using UDP, set this to udp

 column 5 - axip: unused

 axudp: udp port of remote side you need to send your frames to.

 column 6 - axip: unused

 axudp: b or d (see column 4 notes for axip)

Note: you can run multiple instances of ax25ipd and point each to it's own config file by using the -c switch.

Example:

/usr/sbin/ax25ipd -c /etc/ax25/ax25ipd.conf

/usr/sbin/ax25ipd -c /etc/ax25/ax25udp.conf

/usr/sbin/ax25ipd -c /etc/ax25/ax25slip.conf

and so on. This would be set in your startup script for ease of loading up your ax25 stuff. I strongly suggest

however, that you use axip as axudp is NOT a connectable protocol. At first, axudp was developed as a hack

work-around to those who weren't on the amprnet and couldn't get axip working. While it's intention was

good, it's not always the better one to use. I suggest you visit https://portal.ampr.org/ , make an account,

and request an IP or small block. Your coordinator should be able to assist you with your IP block, dns, and

any config help you may require. Expect weirdness with UDP - you have been warned!

Now the big one:

File: ax25d.conf

This file is quite large in size to post as a whole so I'll only post sections of interest for sample purposes as

I describe what each section does. Here is how you process incoming connects and tell your system how to handle

each accordingly. Also, you can configure budlists (blocks) accordingly as well. Also note, this is how you can

prevent your node from appearing as a user when you're encapsulating NetRom/ROSE when a user on your node makes

an outbound connect.

First let's look at how to handle ax25 connections:

[n1uro-2 via ax0]

parameters 7 4 \* \* \* \* \*

NOCALL \* \* \* \* \* \* L

default 7 2 2 60 300 5 0 root /usr/sbin/uronode uronode

[univle via ax0]

parameters 7 4 \* \* \* \* \*

NOCALL \* \* \* \* \* \* L

default 7 2 2 60 300 5 0 root /usr/sbin/uronode uronode

Here you'll notice I call the ax25 side of my node -2. This is how I hook to flexnet (see "man flexd.conf" if you

run URONode). Connections inbound via ax25 from flexnet (which is vanilla ax25) to n1uro-2 will get automatically

routed to my system and will launch uronode accordingly. You define the connects with brackets []'s as

[callsign-ssid via ax25-interface]. This tells ax25d to handle an incoming connect to that SSID on that interface

accordingly. You do NOT want to use the SSID associated in "ifconfig ax#" because that would interfere with any

IP routing you plan on doing!

Note: you CAN use your NetRom alias here but do NOT use your NetRom SSID! If you do, your node will also try

to establish a connect as a user as well as encapsulate NetRom to the remote end. By NOT configuring your

NetRom SSID here you can prevent this from occurring and also prevent unwanted loop traffic as your node will

continue to spawn itself to the remote end. This is where many MANY people make an error! Don't let this happen

to you!

The parameters line sets ax25 mode configurations. 7 for example sets the number of retry cycles used in

conjunction with the retry config in axports. Since I configure 2 there, for every two retries, I allow up to

7 full cycles, for a total of 14 attempts.

The NOCALL sets a budlist/block for someone who doesn't define a callsign/ssid to their tnc, and locks the

the interface from an attempted connect. In a sense, this is a callsign check.

the default sets what the system should do for all other connects. As you can see, I reset the parameters

to which after that I tell the system to spawn the appropriate service. In the above example connects to

either n1uro-2 OR to univle will spawn uronode. The first parameter is the user id to use to do the launch,

the second is the full path to the binary you're going to spawn, including the binary, and the third is the

binary itself. Some binaries allow for further parameters. See the appropriate manpages for the service

binary you plan to allow to be spawned for such parameters. URONode doesn't require any.

[n1uro-11 via ax1]

parameters 1 10 \* \* \* \* \*

NOCALL \* \* \* \* \* \* L

default \* \* \* \* \* \* - root /usr/sbin/axconv axconv -c 88 -n %u -p 236 44.88.0.9

Here you see I allow incoming connections to n1uro-11 on interface ax1 to spawn axconv which is an ax25

based linker to the conversd daemon. Here I set a default-login channel, user (%u) paclen, and the IP

address for axconv to telnet to. A similar program is available for dxspider.

You will wish to repeat each spawned service to each interface accordingly such as:

[univle via ax1]

parameters 7 4 \* \* \* \* \*

NOCALL \* \* \* \* \* \* L

default 7 2 2 60 300 5 0 root /usr/sbin/uronode uronode

and so on. Remember NOT to use your NetRom SSID to spawn a node! You'll see why below.

Inbound NetRom connections:

With instructing ax25d for NetRom based connections you use another set of brackets <> to define a NetRom

interface handler. The same set of rules apply to ax25 configurations. Here's my example:

# Node (Note: do not include the ' 's they're there to make <> print on the web)

'<'nr0>

parameters 1 10 \* \* \* 3 \*

NOCALL \* \* \* \* \* \* L

default \* \* \* \* \* \* 0 root /usr/sbin/uronode uronode

# Convers

'<'nr1>

parameters 1 10 \* \* \* \* \*

NOCALL \* \* \* \* \* \* L

N1URO-1 \* \* \* \* \* \* L

default \* \* \* \* \* \* - root /usr/sbin/axconv axconv -c 88 -n %u -p 236 44.88.0.9

You'll notice all you need in the brackets is the NetRom interface in nrports to activate the service via NetRom.

It will automatically associate the callsign-ssid and alias with this service upon seeing an incoming NetRom

request. If you have a flexnet neighbor you'll wish to monitor their polling and set a budlist for that specific

flexnet callsign-ssid or else flexnet will login as a user when doing it's poll. This doesn't prohibit flex from

polling your interface (same rule applies to ax25 interfaces), it will get it's RTT fine, however your system

won't spawn the service to it. If you don't have flexnet in your region than this is moot.

Also note: FBB does NOT require you to add it in here. FBB very nicely has it's own internal NetRom handler and

doesn't require ax25d to handle it.

Inbound ROSE connections:

ROSE uses the parethesis {} to define it's handling. It also uses ax25 configs as well. To define an outbound

ROSE linker:

{\* via rose}

NOCALL \* \* \* \* \* \* L

default \* \* \* \* \* \* - root /usr/sbin/rsdwnlnk rsdwnlnk 4800 n1uro-6

#

To allow incoming rose connects:

#

[N1URO-6\* via ax0]

NOCALL \* \* \* \* \* \* L

default \* \* \* \* \* \* - root /usr/sbin/rsuplnk rsuplnk rose

#

I'm not that familiar with ROSE to further explain this however if you don't use ROSE (like I don't)

simply do not add a configuration for it in ax25d.conf.

I hope this white page has helped.

AX25 Generic

#!/bin/sh

# Start AX25/Netrom networking daemons.

# Script written and modified by Brian Rogers N1URO

# Date of last modification: 10/2013

### BEGIN INIT INFO

# Provides: ax25

# Required-Start: $network $remote\_fs $syslog

# Required-Stop: $network $remote\_fs $syslog

# Default-Start: 2 3 4 5

# Default-Stop: 0 1 6

# Description: AX25 server

### END INIT INFO

set +e # Don't exit on error status

PATH=/usr/local/sbin:/usr/local/bin:/sbin:/bin:/usr/sbin:/usr/bin

# DAEMON=/usr/local/bin/ax25

# NAME=ax25

DESC="AX25 server and amprnet IPEncap tunnelling"

# Most configuration options in /etc/default/dnsmasq are deprecated

# but still honoured.

ENABLED=1

test -f /usr/sbin/ax25d || /usr/local/sbin/ax25d || exit 0

case "$1" in

 start)

 COMIP=<public.ip>

 AMPRIP=<44.x.x.x>

 IPMASK=<255.255.x.x>

 # Install modules and configure your amprnet tunnel w/rip.

 modprobe -a ipip mkiss ax25 netrom

 ifconfig tunl0 $AMPRIP netmask $IPMASK multicast arp up

 sleep 1

 ip tunnel change ttl 64 mode ipip tunl0

 /usr/sbin/ampr-ripd -t 1 -a $COMIP -p pLaInTeXtpAsSwD -i tunl0 -v -s -r

 sleep 1

 #echo "AX25: starting loopback..."

 echo "Starting: "

 # kissnetd local loopback for fbb

 #socat PTY,link=/dev/ptysf PTY,link=/dev/ptyse &

 #sleep 1

 # for ax0 (vhf)

 socat PTY,link=/dev/ttyq0 PTY,link=/dev/ptyq0 &

 sleep 1

 # for ax1 (hf)

 socat PTY,link=/dev/ttyq1 PTY,link=/dev/ptyq1 &

 sleep 1

 # for ax2

# socat PTY,link=/dev/ttyq2 PTY,link=/dev/ptyq2 &

# sleep 1

 # for ax3

# socat PTY,link=/dev/ttyq3 PTY,link=/dev/ptyq3 &

# sleep 1

 # dual port TNC

# /usr/sbin/mkiss -s 9600 /dev/ttyS1 /dev/tty@0 /dev/tty@1

# sleep 1

 echo "Attaching AX25 to the interfaces..."

 sleep 1

 # ax0 (axip)

 /usr/local/sbin/kissattach /dev/ptyq0 ax0 $AMPRIP

 /usr/local/sbin/kissparms -p ax0 -t 10 -s 100 -r 255

 /usr/local/sbin/axparms -setcall ax0 <callsign>-1

 # ax1 (axudp)

 /usr/local/sbin/kissattach /dev/ptyq1 ax1 $AMPRIP

 /usr/local/sbin/kissparms -p ax1 -t 10 -s 100 -r 255

 /usr/local/sbin/axparms -setcall ax1 <callsign>-2

 # (ax2)

# kissattach /dev/ptyq2 ax2 $AMPRIP

# kissparms -p ax2 -t 10 -s 100 -r 255

# /usr/local/sbin/axparms -setcall ax2 <callsign>-3

 # (ax3)

# kissattach /dev/ptyq3 ax3 $AMPRIP

# kissparms -p ax3 -t 10 -s 100 -r 255

# /usr/local/sbin/axparms -setcall ax3 <callsign>-4

 # NetRom

 echo "Attaching NetRom to the interfaces..."

 sleep 1

 /usr/local/sbin/nrattach -i $AMPRIP -m 512 nr0

 echo "done."

 # Ifconfig the interfaces

 echo -n "Bringing up ax25 interfaces: "

 sleep 1

 ifconfig ax0 $AMPRIP netmask $IPMASK up

 echo -n "ax0 "

 sleep 1

 ifconfig ax1 $AMPRIP netmask $IPMASK up

 echo -n "ax1 "

 sleep 1

 ifconfig ax2 $AMPRIP netmask $IPMASK up

 echo -n "ax2 "

 sleep 1

 ifconfig ax3 $AMPRIP netmask $IPMASK up

 echo "ax3 "

 sleep 1

 echo -n "Bringing up NetRom interfaces: "

 ifconfig nr0 $AMPRIP netmask $IPMASK up

 sleep 1

 echo -n "nr0 "

 sleep 1

# ifconfig nr1 $AMPRIP netmask $IPMASK up

# echo "nr1 "

# sleep 1

# ifconfig nr2 $AMPRIP netmask $IPMASK up

# echo -n "nr2 "

# sleep 1

# ifconfig nr3 $AMPRIP netmask $IPMASK up

# echo "nr3."

 # Creating the tunnel interface

 echo -n "Bringing up IPENCAP interface "

 ip tunnel change ttl 64 mode ipip tunl0

 sleep 1

 echo "tunl0."

 echo "done."

 sleep 2

 # Load the daemons

 echo -n "Starting AX25 and NetRom daemons: "

 echo -n "ax25d "

 /usr/local/sbin/ax25d -c /etc/ax25/ax25d.conf -l

 sleep 1

 echo -n "ax25ipd "

 /usr/local/sbin/ax25ipd -c /etc/ax25/ax25ipd.conf

 sleep 1

 echo -n "ax25udp "

 /usr/local/sbin/ax25udp -c /etc/ax25/ax25udp.conf

 sleep 1

 echo -n "axdigi "

 /usr/sbin/axdigi &

 sleep 1

 echo "netromd."

 /usr/local/sbin/netromd -c -i -t 15 -p 1

 sleep 1

 echo -n "Helper daemons: "

 echo -n "mheardd "

 /usr/sbin/mheardd -f

 sleep 1

 echo -n "beacon "

 /usr/sbin/beacon -c <callsign>-2 -t 10 -d ID ax0 "<alias:node>} URONode in <location> <CALLSIGN>" &

 /usr/sbin/beacon -c <callsign>-2 -t 10 -d ID ax1 "<alias:node>} URONode in <location> <CALLSIGN>" &

# /usr/sbin/beacon -c <callsign>-2 -t 10 -d ID ax2 "<alias:node>} URONode in <location> <CALLSIGN>" &

# /usr/sbin/beacon -c <callsign>-2 -t 10 -d ID ax3 "<alias:node>} URONode in <location> <CALLSIGN>" &

 sleep 1

# echo -n "conversd "

# /opt/conversd/bin/conversd

# sleep 1

 # Configure amprnet iptables rules if needed:

# iptables -I INPUT 1 -j ACCEPT --proto 4

# iptables -I INPUT 1 -j ACCEPT --proto 93

# iptables -I OUTPUT 1 -j ACCEPT --proto 4

# iptables -I OUTPUT 1 -j ACCEPT --proto 93

# iptables -I FORWARD 1 -j ACCEPT --proto 4

# iptables -I FORWARD 1 -j ACCEPT --proto 93

# iptables -I FORWARD 1 -j ACCEPT -s 0.0.0.0/0 -d 44.0.0.0/8

 sleep 1

 echo ""

 # no need to touch this, let rip handle routing.

 echo -n "Installing routing: "

 ip route add default via 169.228.66.251 dev tunl0 onlink src $AMPRIP table 1

 ip rule add to 44/8 pref 1 table 1

 ip rule add from 44/8 pref 1 table 1

 echo "AX.25 and related processes started."

 # Proc file system tweeks

 echo "10" > /proc/sys/kernel/panic

 echo "5" > /proc/sys/net/netrom/obsolescence\_count\_initialiser

 echo "203" > /proc/sys/net/netrom/default\_path\_quality

 echo "2000" > /proc/sys/net/netrom/transport\_acknowledge\_delay

 echo "1" > /proc/sys/net/netrom/transport\_requested\_window\_size

 echo "600000"> /proc/sys/net/netrom/transport\_no\_activity\_timeout

 echo "1" > /proc/sys/net/ax25/ax0/ip\_default\_mode

 echo "0" > /proc/sys/net/ax25/ax0/backoff\_type

 echo "7" > /proc/sys/net/ax25/ax0/standard\_window\_size

 echo "5000" > /proc/sys/net/ax25/ax0/t1\_timeout

 echo "5" > /proc/sys/net/ax25/ax0/maximum\_retry\_count

 echo "1" > /proc/sys/net/ax25/ax0/t2\_timeout

 echo "90000" > /proc/sys/net/ax25/ax0/t3\_timeout

 echo "600000" > /proc/sys/net/ax25/ax0/idle\_timeout

 echo "1" > /proc/sys/net/ax25/ax1/ip\_default\_mode

 echo "0" > /proc/sys/net/ax25/ax1/backoff\_type

 echo "5" > /proc/sys/net/ax25/ax1/maximum\_retry\_count

 echo "7" > /proc/sys/net/ax25/ax1/standard\_window\_size

 echo "5000" > /proc/sys/net/ax25/ax1/t1\_timeout

 echo "1" > /proc/sys/net/ax25/ax1/t2\_timeout

 echo "90000" > /proc/sys/net/ax25/ax1/t3\_timeout

 echo "600000" > /proc/sys/net/ax25/ax1/idle\_timeout

# echo "1" > /proc/sys/net/ax25/ax2/ip\_default\_mode

# echo "0" > /proc/sys/net/ax25/ax2/backoff\_type

# echo "4" > /proc/sys/net/ax25/ax2/standard\_window\_size

# echo "5000" > /proc/sys/net/ax25/ax2/t1\_timeout

# echo "10" > /proc/sys/net/ax25/ax2/maximum\_retry\_count

# echo "1" > /proc/sys/net/ax25/ax2/t2\_timeout

# echo "30000" > /proc/sys/net/ax25/ax2/t3\_timeout

# echo "600000" > /proc/sys/net/ax25/ax2/idle\_timeout

# echo "1" > /proc/sys/net/ax25/ax3/ip\_default\_mode

# echo "1" > /proc/sys/net/ax25/ax3/backoff\_type

# echo "10" > /proc/sys/net/ax25/ax3/maximum\_retry\_count

# echo "2" > /proc/sys/net/ax25/ax3/standard\_window\_size

# echo "7000" > /proc/sys/net/ax25/ax3/t1\_timeout

# echo "1" > /proc/sys/net/ax25/ax3/t2\_timeout

# echo "90000" > /proc/sys/net/ax25/ax3/t3\_timeout

# echo "600000" > /proc/sys/net/ax25/ax3/idle\_timeout

 echo 0 > /proc/sys/net/ipv4/conf/all/log\_martians

 exit 0

 ;;

 stop)

# Kill daemons for shutdown. May cause kernel panic!

 killall -TERM socat

 killall -TERM ampr-ripd

 killall -TERM beacon

 killall -TERM mheardd

 killall -TERM flexd

 killall -TERM netromd

 killall -TERM ax25udp

 killall -TERM ax25ipd

 killall -TERM axdigi

 killall -TERM ax25d

 killall -TERM kissattach

 killall -TERM kissnetd

 ifconfig ax3 down

 ifconfig ax2 down

 ifconfig ax1 down

 ifconfig ax0 down

# ifconfig nr3 down

# ifconfig nr2 down

# ifconfig nr1 down

 ifconfig nr0 down

 ifconfig tunl0 down

 modprobe -r netrom

 modprobe -r mkiss

 modprobe -r ax25

 modprobe -r ipip

 modprobe -r tunnel4

 # insure routing is flushed for amprnet upon shutdown

 ip rule del from 44.0.0.0/8 pref 1 table 1

 ip rule del to 44/8 pref 1 table 1

 exit 0

 ;;

 restart)

 service ax25 stop

 sleep 4

 service ax25 start

 exit 0

 ;;

 \*)

 echo "Usage: /usr/local/bin/ax25 {start|stop|restart}"

 exit 0

 ;;

esac

exit 0

# Set up an AMPR Net Gateway

## Introduction

The AMPRNet or 44.0.0.0/8 network is a global Internet-routed network reserved for use by Amateur Radio. The URL names are suffixed with .ampr.org and are resolvable by almost any properly configured DNS name server you wish to use. There's also some exclusively for use on the AMPRNet if you know where they are.

While some softwares available for Internet routing uses public network IPs and dynamic hostname resolvers such as ddns.org or no-ip.org, there is NO security on these networks. You are at the mercy of your ISP and any port filtering they may impose. By using the AMPRNet, you can bypass the blocks your consumer-grade ISP places on you and create a whole learning experence for yourself - such as network security, web server configurations, mail server management, and more.

## What's the secret?

The AMPRNet uses a special protocol not a port, so there are no ports to forward in your router! AMPRNet uses IP protocol number 4, in simple terms IPEncapsulation. This is a means of tunneling IP packets through an existing IP path. to bypass the filters and blocks on consumer ISP circuits.

If your OS is incapable of running protocol 4, you will not get connectivity. This includes all Microsoft products. This is because Microsoft improperly mislabeled protocol 4 as IP Version 4.
See <http://support.microsoft.com/kb/949848> for more details.

## Setting up an AMPRNet account.

With your Browser, go to the AMPRNet Portal [http://portal.ampr.org](http://portal.ampr.org/) and open an account by registering.

Log on, go to **Home > Networks** and follow the instructions to request a block of addresses. Typically a small subnet of 6 or 14 usable IPs is all you need. A /29 subnet is 6 addresses and a /28 subnet is 14 addresses.

You will receive your address allocation by email.

Don't forget to fill in the bottom part of that form.

## Setting up a Router.

I suggest using an older computer which runs the Linux OS. You can configure your AMPRNet Gateway and link your Microsoft machine, smartphone, etc to it.

The Raspberry Pi is an excellent choice for a router. It is a small $35 unit with a 100BaseT Ethernet connection and at least 2 USB interfaces, one Port may be used for a Wifi dongle. You can plug your Pi into your Main Router and use wifi out... or vice-versa.

If you link your RPi as a Client you can use an external router as a Bridge to supply Ethernet and/or Wifi to Hosts you wish to put on the AMPRNet. You can get as creative as you want configuring a DHCP server, turn Linux/Pi into a full access point Wifi router, etc.

## Installing the RIP device.

The RIPv2 device is used to automatically import AMPRNet routing into your system so you'll have point-to-point tunnelling with others on 44-net. I like to use ampr-ripd by *Marius YO2LOJ* and have a copy of it on : my ftp server

Compile and install ampr-ripd in the /usr/local/sbin/ directory.

## Setting the address of your Router.

Assign a Static IP address to your AMPRNet Router's Host and set the AMPR Net Router in DMZ of your Commercial Router.
This is a 192.168.x.x address, some may use 10-net space. Either address scheme will work depending on your brand of consumer router. Which is sometimes referred to as CPE equipment.

## Configure the Router.

Copy and paste the script (below) and edit in the information on the 4 lines to include your addresses.

AMPRIP='x.x.x.x' # Your Gateway address ex: 44.1.2.3

IPMASK='x.x.x.x' # Your Netmask ex: 255.255.255.248

COMMIP='x.x.x.x' # Your Router's Ip address ex: 24.35.13.28

NOSIP is only needed if you have installed one of the xNOS programs - such as JNOS, maintained by Maiko VE4KLM. However with a Linux based OS, xNOS is no longer needed for packet services.

NOSIP='x.x.x.x' # Next: 44.1.2.4

Place the "dotun.sh" script in /usr/local/bin/ and flag it executable.
**chmod +x /usr/local/bin/dotun.sh**

To Initialize your AMPRNet router: enter the command **/usr/local/bin/dotun.sh start**

Use the command **/usr/local/bin/dotun.sh stop** to shutdown the AMPRNet router.

Comments are included in the script so you can follow step by step what the script is doing. If you have questions, please join the 44-net list and someone will be happy to assist you.

To have true global internet routing on the AMPRNet submit DNS entries for your gateway to your coordinator.

# --- dotun.sh ---

## Installation:

## name this file: /usr/local/bin/ipip

## chmod +x !$

## ln -s /etc/init.d/ipip

## yum install/apt-get install chkconfig

## chkconfig ipip 2345

## service ipip restart

#! /bin/bash

# Start ipip amprnet networking daemons.

# Script written and modified by Brian Rogers N1URO

# Date of last modification: 1/21/2015

### BEGIN INIT INFO

# Provides: ipip

# Required-Start: $syslog

# Required-Stop: $syslog

# Default-Start: 2 3 4 5

# Default-Stop: 0 1 6

# Short-Description: Starts and stops ipencapsulation

# Description: Starts and stops ipencapsulation for ham services

### END INIT INFO

# Author: Brian Rogers

#

AMPRIP='x.x.x.x' # ampr IP of this specific machine.

IPMASK='x.x.x.x' # netmask of your amprnet block.

COMMIP='x.x.x.x' # the IP of your router/modem.

NOSIP='x.x.x.x' # if you run xNOS, what's it's ip?

test -f /usr/local/sbin/ampr-ripd ||

 echo "You need ampr-ripd. You may find it at:"

 echo "ftp://n1uro.ampr.org/packet"

 exit 0;

case "$1" in

 start)

 # Load your ipencap module in the kernel:

 modprobe ipip

 # Allow ip forwarding from amprnet to your ethernet interface

 echo "1" > /proc/sys/net/ipv4/ip\_forward

 # load RIPv2 routing using the ampr-ripd daemon

 /usr/local/sbin/ampr-ripd -t 1 -a $COMMIP -p -i tunl0 -v -s -r

 # Configure your ipencap tunnel interface - required for the amprnet

 ifconfig tunl0 $AMPRIP netmask $IPMASK up

 # Allow traceroutes to work on the amprnet:

 ip tunnel change tunl0 mode ipip ttl 64 pmtudisc

 # If you run xNOS, configure a tun/tap interface:

# ifconfig tun0 $AMPRIP pointopoint $NOSIP up

 # configure your rointing accordingly:

 # Note, if this node is behind an existing gw on your

 # lan, change the "via" ip below in the default line to

 # the lan IP of your gateway... and add a route in table 1

 # on your gateway to this node.

# ip route add $NOSIP dev tun0 onlink table 1 src $AMPRIP

 ip route add default via 169.228.66.251 dev tunl0 src $AMPRIP onlink table 1

 # configure policy routing so that frames from/to your 44-net IP

 # know how to route accordingly:

 ip rule add from 44/8 pref 1 table 1

 ip rule add to 44/8 pref 1 table 1

 # script is done, exit as a clean flush.

 echo -n "Amprnet routing complete. "

 \_sleep 1

 echo "Script by N1URO."

 exit 0

 ;;

 stop)

 # Unload what we loaded above:

 ip rule del to 44/8 pref 1 table 1

 ip rule del from 44/8 pref 1 table 1

 ifconfig tunl0 down

 ifconfig tun0 down

 killall -TERM ampr-ripd

 modprobe -r ipip

 echo "Amprnet routing disabled."

 exit 0

 ;;

 restart)

 echo "Restarting/resetting amprnet routing..."

 ipip stop

 sleep 3

 ipip start

 exit 0

 ;;

 \*)

 echo "Usage: ipip {start|stop|restart}"

 exit 0

 ;;

esac

exit 0

--- EOF ---

Documentation rewrite by Charley K4GBB.