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. Also, the callsign-ssid you assign to the interface also will in a sense become the MAC ADDRESS of the interface, so the SSID -must- remain unique to the rest of your SSIDs.

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
```

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 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.

FTP this document.