XRouter Update (XRlin502e)

Paula G8PZT has released a NEW version of XRouter, XRLin502e. Why there is not a release of Xrpi “that question remains”
You can Download it at https://groups.io/g/xrouter/files (must be registered)

Or Download it from here https://ham.packet-radio.net/packet/xrlin/xrlin502e
And the Release notes here. https://ham.packet-radio.net/packet/xrlin/V502%20Release%20Notes.txt

XRLin (and XRPi) Release Notes For Version 502
##

Version 502 has been a long time coming, far too long. This is because a great deal has been added or tweaked since the last release, and the aim was to get it all finalised, documented and thoroughly tested before it became v502.

If new or experimental features are released too soon, future development of them can be stifled or prevented by compatibility issues. During alpha testing it is easy to change things, for example protocols, without compatibility issues. But once released, any further changes have to be backward-compatible to avoid breaking earlier versions.

Several inter-dependent sub-sections of the project were in parallel development when work on it was unexpectedly curtailed at the end of August 2020. There was no time to tidy up loose ends or make notes, and it has not been possible to work on the project since then.

It will take a long time to piece together what work had been in progress at the time, and to write documentation. As that process would further delay the release of v502, I have decided to release it "as-is", with no documentation, no guarantees that it will work, and no guarantees that it won't be broken by the next version.

It may crash. It may not work on your flavour of operating system. It may contain alpha-test diagnostics that fill up your disk or mess up your screen. Use it at your own risk, and PLEASE report the bugs.

Also, there is much unfinished work where the human interface could probably be improved. If you have any suggestions, please let me know.

What's New in 502?

* SYN Cache
* FTP client.
* NFTP (Netrom File Transfer Protocol) server and client.
* NetRomX "Standard Services"
* Packet SMS (Short Message Service) client & server
* IP Packet capture
* Private sysop chat/messaging.
* Bug fixes.
* More colour
* Sounds
* And lots more...

Changelog

The last full release was v501c, back in May 2019. The following is a list of some (but possibly not all) of the changes since then:

- Fixed: "MON ?" now displays syntax help instead of "Error!".

- Fixed: If no Ethernet PORT was defined, the Telnet command always
returned "System Busy" for a non-localhost destination, even if a kernel route has been defined.

- Fixed: F3 didn't correctly set the monitored ports.

- Fixed: UDP multicasts generated ICMP "port unreachable"

responses.

- Fixed: Bug in chat server which had been causing crashes.

- Fixed: Chatserver /ECHO command didn't toggle the echo. /ECHO is no
longer a toggle, the form is "/ECHO [on | off]"

- Fixed: TCPKISS connection bug.

- Fixed: The TIME command was reporting "GMT" even in Australia.
Removed the timezone for now.

- Fixed: SYN scan notification displayed a random number instead of IP
source address.

- Fixed: If nodes table was sorted by callsign, using the SORTBYCALL
config directive, some of the nodes were omitted from nodes broadcasts.

- Fixed: Missing timestamp on pi version of chat keepalive

- Fixed: Some bugs in PPP, which now works as intended.

- Fixed: Sent/rcvd bytes stats were zero on some types of interface.

- Fixed: TCP RESET command for case when IP is banned.

- Fixed: The LOG command wasn't working. It is now working, and accepts
a numeric argument in addition to the previous ON | OFF.

- Fixed: KISS over TCP was sending incorrect frame format.

- Fixed: Extremely long filenames could cause segfault on FTP.

- Fixed: KISSTCP link was manically retrying.

- Fixed: AXTCP now works again.

- Fixed: INP option 16 (position) tracing now works.

- Fixed: AXIP using 44-net addresses would not work if a TUN interface
was used instead of an Ethernet one. AXIP using non-44-net addresses was OK, but used the Linux IP address NOT the XRPi one. This is now fixed. 44-net addresses can now be used for AXIP if either Ethernet or TUN interface/port combination exists.

- Fixed: Argument to INFO command was case sensitive.

- Fixed: In chat server, "Version" now reports XRPi or XRLin instead of
generic "Xrouter".

- Added: TUN (tunnel) interface type.

- Added: UDP interface, capable of KISS.

- Added: TCP server interface, supporting many protocols.

- Added: "TNC" protocol, for controlling external TNC's in command mode.

- Added: IGMP tracing.

- Added: IPv6 and ICMPv6 tracing.

- Added: Wireshark-compatible packet capture

- Added: MNDP tracing (seems to be used by HamNet).

- Added: ACL command, with subcommands permit, deny, log, remove and
view.

- Added: "IP BAN SAVE" to save the list of banned IP's. This can be

used in crontab to save the list from time to time in case of power cuts.

- Added: IP BAN ADD and IP BAN DROP and IP BAN PORT SAVE subcommands.

- Added: TCP/UDP/ALL options to the "IP BAN PORT" (honeypot) command,
for creating protocol specific bans.

- Added: IP BAN PORT LIST to list the banned ports.

- Added: PC[ap] [on |off] command, to control packet capture.

- Added: MON IDS [on | off] command, to control IDS monitoring.
If MON is ON *and* MON IDS is ON, IDS warnings are echoed to console or remote monitoring session as appropriate. The default for a remote monitor session is ON, but the default for consoles is OFF.

- Added: WAIT < (secs<60) | (ms>60) > command for use in BOOTCMDS.SYS.

- Added: "TNC" command for controlling TNC-style peripherals, e.g. real
TNC's or any applications that use plain text commands.

- Added: IFACE command, with subcommands ADD, DROP and LIST. This is
still a work in progress. Interfaces can be added and deleted on the fly, but the code to configure them is incomplete.

- Added: IDS subcommand to STATS command, to display brief IDS stats.

- Added: Chatserver "/STAT[us]" command to set status/presence indication. Experimental, it might not work.

- Added: Lots of layer-specific error codes, making it easier to locate

problems.

- Added: NOPARMS to the list of KISSOPTIONS. Setting this option
prevents XRouter from sending KISS parameters to the TNC.

- Added: SYN Cache, plus associated TCP CACHE sub commands.

- Added: SYNCACHESIZE and SYNCACHELIFE configuration directives, which
are used in XROUTER.CFG to set the default cache size and lifetime. These parameters can also be changed on the fly using TCP CACHE SIZE and TCP CACHE LIFE commands. If the directives are not present, they default to 1000 slots and 10 seconds.

- Added: Smurf and Fraggle attack detection.

- Added: Hit counts and last_hit display to banned ports display.

- Added: Honeypot hits to IDS statistics display.

- Added: UDP honeypot (tcp honeypot already existed)

- Added: TCP port scan detection (udp detection already existed)

- Added: Many new commands to TNC2 emulation.

- Added: The ability to ban IP ranges using netmask.

- Added: Saving and reloading of banned ports across restarts.

- Added: Option to enter plain text password using "@" command

- Added: TCP service port into PMS NODE entry (CHAT already done).

- Added: Timestamp, TCP service port and node capability flags to INP3.

- Added: NFTP (Netrom File Transfer Protocol) server, client and
command.

- Added: NFTPROOT configuration directive, defaulting to "FTP".

- Added: APRS, CHARGEN, CHAT, DISCARD, ECHO, INFO, NFTP, NTTY, PMS,
SMS, TIME and WX NetromX services.

- Added: FTP client, with colour, scripting, and 67 commands.

- Added: FTP command, to invoke the FTP client.

- Added: FTPCLI.SYS, containing details of frequently used connections.
All fields in that file except nickname and hostname are optional, so we can use it simply as a nickname to hostname translator, or we can login with USER alone and supply PASSword manually, or we can automate the password and send the account manually, or we can automate the entire logon.

- Added: FEAT, SIZE and MFMT commands to FTP server.

- Added: SMS server, client and command.

- Added: Security window now displays separate bad login stats for FTP,
Telnet, Telnet proxy and RLogin, plus FTP directory hacks, TCP scans and total notifications.

- Change: Ports and interfaces are now listed in strict numerical
order, instead of the order in which they are created.

- Change: The PORT DROP command had been disabled long ago because the
code was unfinished and possibly dangerous. This has now been completed, so the command should be safe to use.

- Change: IDS logs are now stored in LOG directory instead of the main
working directory.

- Change: LOG flags rearranged, and the ability to save logs with CRLF
line endings is added. Bit 0 controls logging on/off as
before. If bit 1 is set logs are written with CRLF (Windows /
DOS) instead of LF (Unix) format. This is useful for those
who view the logs on a Windows machine.
Session logging is now controlled bit bit 2 instead of bit 1.

- Change: The chatserver "/W *" command now shows the user's status /
presence indication.

- Change: If someone logs onto channel 1234 on local chat server, they
show up on the sysop chat window's "online" list, and when
they log off, they are instantly removed from the list
instead of slowly timing out.

- Change: If a local chat user (i.e. one logged directly onto our chat
server) changes status and they are logged onto channel 1234,
the status is echoed to the sysop chat window.

- Change: As chat user connections can sometimes die, even when
keepalives are enabled, keepalives now include a timestamp
allowing the user to verify that the connection really is
alive.

- Change: Chatserver "channel-leave" messages for channel 1234 now
remove users from the "online" list on sysop chat window.
This should reduce the number of long-gone connections that
persist on the chat window.

- Change: TCP sockets stuck in SYN_RECEIVED state are now purged after

a period of inactivity. This is not part of RFC793, but seems sensible.

- Change: Tidied up wording of TCP/UDP scan reporting.

- Change: INP3 unicasts no longer include IP address if it is private.

- Change: NetromX connections now show the service number, e.g. G8PZT::18

- Change: A lot more information is saved to XRNODES file, e.g. QTH,
locator, software type and flags, version, timezone, etc.
And the data is reloaded again at start up
.
- Change: Reinstated NCMP infocasts, but much improved on 2001.

- Change: Re-enabled PEERS command in much expanded form. This displays
the closest XRouter neighbours.

- Change: IDS now reports source callsign for valid but unsolicited AXUDP.

- Change: IDS log is now daily, and depends on LOG_IDS flag (256)

- Change: Isolated IDS-reported incidents, such as entering an incorrect
password are not necessarily suspicious and might not be detected as abuse. But when they are repeated over time to form patterns they become suspicious. IDS can now recognise some suspicious patterns of behaviour, and automatically ban the caller's IP.

- Change: The chat /keepalive command now accepts ON | OFF | , so
the interval can be tailored to the user's particular needs.

- Change: Chat keepalives are now sent only when there is no activity
on the user's connection, instead of rigidly every 10 mins.

- Change: All PZTDOS and NFTP client functions now cope with arguments
that contain spaces, providing such args are enclosed in
double quotes, e.g. COPY "old file" "new text file".
Note that the superfluous use of double quotes is allowed,
e.g. 'GET "fred.txt"', although many functions don't require
them, e.g. 'DEL my big fat file' is ok because DEL only
has one argument so there's no ambiguity.

- Change: CHAT, DISCARD, ECHO, INFO, PMS, TIME and WX commands now
accept a nodecall or alias argument, allowing them to access
those services on other XRouter nodes.

- Change: ECHO, DISCARD and CHARGEN servers can now be aborted using /x
instead of disconnection. This was not part of the original
protocols as defined in the RFC's, but it saves having to log
back in after ending one of these sessions. Ending the session
by disconnection is still allowed.

- Change: Tidied up the columns in INP3 tracing.

- Change: Tidied up ARP trace display.

- Change: Big rewrite of interface routines, which might break some
things and fix others?.

- Change: Prevented invalid protocols from being specified on
interfaces that don't support then, to avoid confusion.

- Change: Major improvements to TNC2 emulation, especially
monitoring.

- Change: DHCP tracing, which was disabled way back in XR16 to save

space, is reinstated because space is no longer an issue.

- Change: Reception of IP multicasts was previously blocked, but is now
unblocked. This will allow the sysop to detect potential
attacks instead of being oblivious to them.

- Change: 3 unsucessful FTP login attempts during one session used to
result only in disconnection. It now bans the user's IP as
well. This mitigated brute force and dictionary attacks.

- Change: 5 incorrect FTP logins within a short interval now result in
banning of caller's IP address.

- Change: STATS syntax help improved, and it now displays syntax help if
an invalid stats sub-command is entered.

LOG Flags
~~~~~~~~~

The flags used in the argument to the LOG directive and command are in a state of flux. It hasn't yet been decided which flags are useful and what order they should be in, so they may change again. The eventual aim is for the lower numbers to be used for routine logging and the higher ones for occasional debugging (which may generate a lot of logging).

The flags recognised in v502 are as follows:

Bit Value Function
----
0 1 Enable/disable logging
1 2 Use CRLF instead of LF line endings
2 4 Log session activity
3 8 Unused
4 16 Unused
5 32 Unused
6 64 Unused
~~~~~~~~~

7 128 Log ODN activity.
8 256 Log IDS activity to IDS log
9 512 Log AXTCP activity
10 1024 Generate PCAP file (use with caution)

SYN Cache
~~~~~~~~~~

To combat the ever-growing problem of TCP port scanning, which was using up a lot of TCP resources, a SYN cache has been added.

In "normal" RFC793 TCP, a received SYN elicits a SYN|ACK, and the TCP state machine allocates resources then moves to SYN_RECEIVED state, awaiting an ACK from the caller. The nastier port scans tend to send a SYN, wait for the ACK, then move on without sending a reset. This leaves dozens of half-open connections, which eventually prevent new connections from being formed.

With a SYN cache, a received SYN elicits a SYN|ACK as before, but no resources are allocated. The SYN is cached for a short while. If an ACK is received, a normal connection is created, otherwise the SYN is discarded. This makes the TCP stack far more resistant to port scans.

The TCP command has been modified to accommodate the SYN cache - see later.

****
****

Standard Services
==========

In my view, a major limitation of NetRom is the fact that each "application" (BBS, PMS, CHAT, DXCLUSTER etc.) that has a direct network presence, rather than being accessed via a node's comand line, requires a separate SSID, and only 16 SSID's are possible.
~~~~~~~~~~

Not only does this limit the maximum number of connectible services per node callsign to 16, but it also clutters up the node table with multiple instances of the same callsign, each with a different SSID. So far it hasn't been too much of an issue, but it is stifling development.

Another problem is that no-one agrees on a standard for SSID's. Some people think SSID 0 should be for users, -1 for nodes, -2 for PMS, -3 for BBS, -8 for chat and so on. Others think -0 should be the node, -8 for the PMS, -11 for chat etc. It's a pig's ear!

If you see "G6XYK-5" in your nodes table, what is it? Maybe it's a NOS system. Maybe it's a "Ping-Pong" chat node. Or is it his FBB BBS?

I don't have the answer to that one, but one thing that can be standardised, at least on XRouters, is "Service Numbers". This is the NetRom equivalent of TCP "port" numbers, which allow multiple services to co-exist on a single IP address.

No matter what the node's call, alias or SSID, you can be sure that, if it's an XRouter, service 1 will always give you information about the node. Likewise, service 2 will always be the PMS, and service 80 will always be the HTTP server.

So if the alias contains "XR", e.g. DOTXR, XRPKT etc, and you want to connect to their PMS, you know that if it's enabled, it's going to be service 2.

This is made possible by an extension to NetRom, using opcode 8 = CREQX, i.e. Extended Connection Request". This opcode allows up to 65535 destination service numbers to be requested. I call this, for want of a better word, "NetromX".

Nodes that don't understand CREQX simply reject the connection (or crash if they have been badly written. lol :-) It's been in use since 2013, and I'm not aware of any problems so far.

To make life simple, TCP and UDP services have "Well Known Ports", and I proposed a similar scheme for NetromX as

follows:

No. Service Description
----
1 INFO Information server
2 PMS Personal Message System
3 BBS Bulletin Board System
4 DX DX cluster/dx-spot feed
5 TPP "Tampa Ping-Pong" chat server
7 ECHO Echoes data back to sender
8 CHAT Chat server
9 DISCARD Data sink
10 RMS (winlink RMS}
11 BPQCHAT BPQ chat server
13 DAYTIME Local date/time (similar to RFC867)
14 APRS APRS Server
16 WX Local weather information
18 SMS Short Message System
19 CHARGEN Generates a test pattern
20 NDATA Reserved for FTP data
21 NFTP Netrom File Transfer Protocol
22 SSH Secure login (if legal?)
23 TELNET Normal login
25 SMTP Simple Mail Transfer Protocol
80 HTTP NetromWeb server
87 NTTY Netrom TTY - Keyboard to keyboard chat

There are many other possibilities, not yet finalised.

Standard services facilitate simple commands such as "TIME ", to discover the time, zone and daylight saving at a distant node. Or "PMS " to connect directly to someone's PMS.

It is envisaged that some of the services may be used by network crawlers (human and machine) to harvest data without needing to know the exact format of the commands on all the different types of software.

Standard service numbers cannot be changed. That's the point - they are always the same, like "well-known" TCP/IP numbers, so people can remember and rely on them.

****

NetromX Services Implemented in XRLin/XRPi v502 (numeric order)
==========

Information Service
~~~~~~~~~~~~~~~~~~~

Isn't it frustrating to connect to a node, only to find that there is no INFO command, or the sysop hasn't bothered to set up any information?

Where in the world is it? We aren't all familiar with country prefixes!

What software is it running? That affects what commands we must use to find information. What sort of node is it? Just a router, or is it the node underlying a BBS or DX cluster? Where can I find more information about the system, the group, the software etc?

The NetRomX "Information Service" (service no. 1) is intended to answer some of your questions. It is accessed by connecting to service 1 on any XRLin or XRPi running version 501v or later. For example:

C G8PZT 1

The information is provided in a form which is readable by both humans and machines. It is of the ": " form, with each piece of information on a separate line. Every keyword is unique. Only a few have been implemented so far, although others are planned. Your suggestions would be welcome.

The INFO command has been modified to accomodate this feature, so you don't even have to remember the service number or add the "stay" flag. The syntax is now:

I[nfo] [nodecall | nodealias | topic]
~~~~~~~~~~~~~~~~~~~

Here's a typical session:

```
info kidder
VK2DOT-1:DOTXR} Info:
Trying G8PZT::1...

VK2DOT-1:DOTXR} Connected to G8PZT::1

Node-Type: Host / Router
Node-Callsign: G8PZT
Node-Alias: KIDDER
QTH: Kidderminster, Worc's
Maidenhead: IO82VJ
Position: 52.400002 / -2.250000
Sysop-Callsign: (to be done)
Software-Type: AX25+NetRom+IP Router/Node
Software-Name: XrPi
Software-Version: 501y (19/7/20)
Software-Author: Paula G8PZT
Software-Info: http://vk2dot.dyndns.org/xrpi
Software-Uptime: 4d, 11h, 3m, 58s
Local-Time: Fri, 21 May 2021 03:27:06 +0000
Local-Sunrise: 05:05
Local-Sunset: 21:08
Netrom-Known: 250
Netrom-Neighbours: 6
AX25-Links: 6
Amprnet-IP: 44.136.16.6
Amprnet-Name: g8pzt.ampr.org
Netrom-Services: 1,2,7,8,9,13,14,16,18,19,21,23,80,87

VK2DOT-1:DOTXR} Welcome back
```

PMS Service
~~~~~~~~~~~

NetromX service 2 is a direct connection to a node's PMS (Personal Message System).

You may not know what SSID a sysop is using for his PMS this week. Or his PMS might be out of Netrom visibility because its
~~~~~~~~~~~

quality is set low. But if you know the nodecall, and it's an XRouter, you always know that the PMS will be on service 2.

The PMS is well documented, so no further explanation is required here.

ECHO Service
~~~~~~~~~~~~

The NetromX ECHO service simply reflects back a copy of any data it receives, for test and measurement purposes. It behaves the same as the ECHO command, and uses the same service number (7) as the TCP/IP version.

The service terminates upon receipt of "/x" followed by newline, or when the caller disconencts.

CHAT Service
~~~~~~~~~~~~

Can't remember what SSID G8PZT's chat server is on? No problem, just connect to service number 8 using the following command:

c g8pzt 8

Add the "stay" flag if you don't want to be disconnected after leaving the chat...

c g8pzt 8 s

If you can't even remember the service number, just type CHAT G8PZT, and it will do both of the above steps for you...

G8PZT-1:MOBILE} Trying G8PZT::8...

G8PZT-1:MOBILE} Connected to G8PZT::8
Hello Paula
Welcome to the chat server at KIDDER. There are 5 users
Type /HELP if you need help

/b

G8PZT-1:MOBILE} Welcome back

DISCARD Service
~~~~~~~~~~~~~~~

The NetromX DISCARD service is a "data sink", i.e. it throws away any data it receives. It is intended for test and measurement purposes. It behaves the same as the DISCARD command, and uses the same "well-known" service number (9) as the TCP/IP version.

The service terminates upon receipt of "/x" followed by newline, or when the caller disconects.

CHARGEN Service
~~~~~~~~~~~~~~~

The CHARGEN service generates a stream of characters for testing purposes. It is very bandwidth hungry and should be used with caution. It uses the same "well known" service number (19) as the TCP/IP version.

Upon opening a connection to NetromX service 19, the server starts sending lines of characters to the caller and continues until the caller types /x followed by newline, or closes the connection. The server accepts no other commands.

Conforming to the de-facto standard, each line of 72 characters is offset by one. This creates a pattern which makes it easy to spot dropped characters. Here's what it looks like:

c g8pzt 19
G8PZT-1:MOBILE} Trying G8PZT::19...
G8PZT-1:MOBILE} Connected to G8PZT::19

"#$%&'()*+,-
./0123456789:;<=>?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^_`abcdefghi
#$%&'()*+,-

```
./0123456789:;<=>?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^_`abcdefghij
$%&'()*+,-
./0123456789:;<=>?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^_`abcdefghijk
%&'()*+,-
./0123456789:;<=>?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^_`abcdefghijk
l
&'()*+,-
./0123456789:;<=>?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^_`abcdefghijk
lm
'()*+,-
./0123456789:;<=>?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^_`abcdefghijk
lmn
()*+,-
./0123456789:;<=>?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^_`abcdefghijk
lmno
)*+,-
./0123456789:;<=>?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^_`abcdefghijk
lmnop
```

and so on...

DAYTIME Service

The DAYTIME service returns the node's current local date and time, for test or general interest purposes. It is accessed via NetromX standard service 13.

Upon connection to service 13, the server sends the local time as a single line of text. The response is currently like this, but may change to include timezone and DST:

```
Thu May 20 16:37:24 2021
```

When the caller has received the text, the server terminates the connection.

APRS Service

Until now, the APRS server has only been accessible to TCP/IP

callers. It is now also available as NetromX service 14.

```
c g8pzt 14
G8PZT-1:MOBILE} Trying G8PZT::14...
G8PZT-1:MOBILE} Connected to G8PZT::14

MB7UYL>APXR01:=5224.00N/00215.00Wn Kidderminster, Worc's
{Xrouter 501y}
VE2PKT-4>ID:!4625.96N/07237.66WB 147.435Mhz 1.2Kb, VE2PKT-4,
XRPI Router
VK2DOT-1>ID:!3323.21S/15121.42E# Niagara Park Node -
(VK2DOT-1)
44.136.16.1
```

This server is already well documented on the XRPi website.

Weather Service

Have you ever wondered what the weather is like at someone else's site? Or maybe you are the sysop of a remote site and you need to know what the weather is doing up there?

The weather service returns local weather information, if available, from the target node. The information is returned in machine-readable plain text form. The server uses NetromX service number 16.

A typical response would look like this:

```
Observed: Mon 17 May 2021 06:01:23 +0001
Pressure: 1007.2 mB
Temperature: 11.2 C
Humidity: 67 %
Wind: 23 mph
Direction: 178 deg
Gust: 31 mph
Rain-24h: 24.7 mm
Rain-Today: 19.7 mm
Rain-1h: 3.2 mm
```

When the client has recieved the information, the server terminates the connection.

Some fields may be missing if there is no information in them.

The information is derived from APRS broadcasts, or from the WXNOW or CUMULUS format files generated by home weather stations or weather applications. The filename is specifed by the WXFILE directive in XROUTER.CFG.

Once per minute, XRLin checks the file specified by the WXFILE directive. If the file is present, and the information therein is more up to date than the previous data, the contents are imported.

A WXNOW.TXT file only has 2 lines, for example:

Feb 01 2009 12:34
272/010g006t069r010p030P020h61b1050

where: 272 = Wind direction in degrees.
010 = Average wind speed in MPH.
g006 = Gust speed in MPH
t069 = Temperature in degrees Farenheit
r010 = Rainfall in the last hour in hundreths of an inch.
p030 = Rainfall in the last 24H in hundreths of an inch.
P020 = Rainfall since midnight in hundreths of an inch.
h61 = Humidity in percent.
b1050 = barometric pressure in millibars.

The Cumulus "realtime.txt" file is a text file with a single line of space separated values. It contains a list of key values of the sensors and is re-created frequently. After creation, it can be set to automatically upload to a website (or to XRLin) via FTP.

An example of the format is as follows (all one line):

18/10/08 16:03:45 8.4 84 5.8 24.2 33.0 261 0.0 1.0 999.7 W 6 mph
C mb mm 146.6 +0.1 85.2 588.4 11.6 20.3 57 3.6 -0.7 10.9 12:00

7.8 14:41 37.4 14:38 44.0 14:28 999.8 16:01 998.4 12:06 1.8.2
448 36.0 10.3 10.5 13 0.2 14 260 2.3 3 1 1 NNW 2040 ft 12.3
11.1
420.1 1 13.6

XRLin currently reads the following fields from realtime.txt:

1 Observation date
2 Observation time
3 Temperature
4 Humidity
6 Wind speed
8 Wind direction
9 Rain rate per hour
10 Rain today
11 Barometric pressure
14 Wind units - m/s, mph, km/h, kts
15 Temperature units - degree C, degree F
16 Pressure units - mb, hPa, in
17 Rainfall units - mm, in
41 Gust speed
48 Rainfall last hour

There is another (non-standard) format that XRLin can import from a WX file. This consists of ": " pairs as follows:

Temperature: 22.7
Barometer: 1015
Wind: 6
Direction: 178
Gust: 12
Humidity: 67

The fields can be in any order, and need1 not all be present. They may be seperated by spaces ot tabs.

At present, temperature must be in Centigrade, pressure in millibars, and wind speeds in MPH. There are plans to make this accept other units. It's all a work in progress, so any suggestions are welcome.

So you haven't got a weather station that outputs one of the above formats? No problem...

Using suitable software or scripts you could extract weather information from your WX station and write it to a file in one of the above 3 formats.

You could even pull weather information for your area off the web and write it to a file. But that is beyond the scope of this document.

In my case I am using a cheap RTL dongle and "rtl_433" software on the Pi to receive and decode the 433.9 MHz OOK transmissions from my weather station to its base unit.

SMS Service

This is a short messaging service for XRouter sysops, and has nothing to do with telephone SMS.

The purpose of this service is to receive short, single-line messages (up to 160 characters), and to deliver them to the sysop. It is service number 18.

Although the protocol is plain text, it is intended for use by automatons, not humans.

It is a one-way protocol. Nodes "push" messages to each other, by connecting to each other's SMS service. They cannot poll for mail.

The protocol is not yet finalised. It works, but may need to be tweaked.

The sysop is alerted to the presence of an unread mesage by a flashing "S" on the top status bar. He can then type SMS to view the messages. See SMS command in the "New or modified commands" section below.

NFTP Service

~~~~~~~~~~~~

NFTP (Netrom File Transfer Protocol) is basically a form of FTP over NetRom. It uses fairly standard FTP server commands, but in it's simplest form it shares a single stream for both commands and data. It is service 21, the same number as the TCP version.

Strictly speaking, NFTP is a misnomer, because the protocol can be used on any reliable byte-ordered stream, be that AX25, NetRom or TCP.

Although NFTP can be used by anyone, it is primarily intended for sysops to exchange files with each other. The protocol permits limited access to "public" files without login.

The server can be accessed in one of 3 ways:

If the source node is NetromX-capable, simply connect directly to service 21 on the target node. For example, "C G8PZT 21".

```
c g8pzt 21
DOTXR:VK2DOT-1} Trying G8PZT::21...
DOTXR:VK2DOT-1} Connected to G8PZT::21
220 G8PZT NFTP ready - Restrictions apply
```

Alternatively, **if you are the sysop of the source node**, use the "NFTP " command to invoke a client which connects to the target node. See the "New Commands" section below for client details.

If the source node is *not* NetromX-capable, you must connect to the target node then issue the NFTP command, supplying the target node's callsign as an argument. For example, if you were connected to VK2DOT and you wanted to connect to G8PZT's server you would issue "C G8PZT", wait for connection then issue "NFTP G8PZT":

```
c g8pzt
DOTXR:VK2DOT-1} Connected to G8PZT
nftp g8pzt
```
~~~~~~~~~~~~

```
G8PZT:KIDDER} Trying server
220 G8PZT NFTP ready - Restrictions apply
```

Once connected to the server, the HELP command reveals the available commands and their syntax:

```
help
211-Cmds: ABOR CWD HELP LIST MDTM NLST NOOP
211-Cmds: PASS PWD QUIT REST RETR STOR TYPE USER
211-(Additional commands available after login)
211-
211 Use HELP for syntax etc.

help stor
211-STOR Upload file to server
211 Syntax: STOR
```

Login (using USER and PASS commands) is optional, and intended only for sysops.

Unlogged users are restricted to a "public" directory, which by default is located at FTP/public. The true location is not shown to users. They cannot climb out of this directory, nor create any directories within it. It is purely a space for sysops and unlogged users to place publicly accessible files, such as useful documents, software etc. For example:

```
04-19-2021 02:41AM
06-20-2020 04:07AM
04-19-2021 02:41AM 49375 repeaterlist.csv
04-19-2021 02:36AM 93162 rfc8200.txt

4 file(s) 150,729 bytes
2 dir(s) 398,048 bytes free
226 File sent ok
```

The main protocol differences between FTP and NFTP are in the commands and responses associated with the transfer of data, i.e. commands like LIST, STOR and RETR.

When a file or directory listing is requested using LIST or

RETR, the server replies with the line "150 OK n", where n is the exact filesize in bytes. After receiving this line, the client must treat the next "n" received bytes as data, to be saved or displayed. After sending the data, the server sends the line "226 File sent ok", and is ready for the next command.

The syntax to upload a file is "STOR <bytes> <filename>", for example, "STOR 96507 xrpi-manual.txt". If this is acceptable to the server, it responds "150 OK <bytes>". Upon receiving this line, the client must send exactly <bytes> bytes of data. The server will not return to command mode until it has received at least the specified number of bytes. Any excess bytes sent by the client are discarded by the server.

Thus you can read text files directly to your monitor, and create them directly from your keyboard if required.

Service 20 is reserved for possible future use as a separate "data" channel.

HTTP Service
~~~~~~~~~~~~

Service 80 accesses XRLin's HTTP server, allowing HTTP over Netrom.

HTTP over NetRom? Shock horror! What a phenomenally stupid idea!!! Well maybe it is, maybe not? It's just another communication tool.

This service is not new - it has been in XRouter since Jan 2013. If you access an XRouter's web interface via TCP/IP, it's what allows you to click on certain nodes in the nodes list, and go to their default pages.

Yes it might be slow, but if the HTML is kept clean and tight, it is workable, and unlike the Internet, the addresses are consistent. My node's "NetromWeb" address is simply my callsign.
~~~~~~~~~~~~

The service number is not affected by the setting of HTTPPORT, which only affects TCP/IP access to the server.

NTTY Service
~~~~~~~~~~~~~~~

NetromX Service 87 is Netrom access to TTYLink - keyboard to keyboard chat with the sysop. It is equivalent to telnetting to TCP service 87, or using the YELL command on the target node.

Upon connection, the sysop of the target node is paged, and has 90 seconds to respond. The page consists of a pop-up dialog on top of the sysop's current window, and an audible sound. Note the latter only works on older-style PC's at present.

At any point during or after the 90 seconds, the caller has the option to leave a short one-line message (160 chars max) or to abort the call.

If the sysop takes more than 90 seconds to respond, and the user has not disconnected, the sysop can still use the "talk" command to initiate a chat with the user.

Here's an example session where the sysop responds:

```
c mobile 87 s

VK2DOT-1:DOTXR} Trying G8PZT-1

VK2DOT-1:DOTXR} Connected to G8PZT-1

TTYLINK at MOBILE:G8PZT-1
Calling sysop for 90 seconds..
Type 'M' at any time to leave a short message, or 'Q' to quit...

*** G8PZT-1 has come online to talk ***
Hello, why are you calling me?
Because I want to, silly!
```
~~~~~~~~~~~~~~~

```
Fair enough, it's mad talking to oneself you know...
I know, but no-one else is around.
Ok, bye for now

*** The other end terminated the chat

VK2DOT-1:DOTXR} Welcome back
```

And here's a session where the sysop didn't respond. Although not strictly TTYlink, I feel it's a useful compromise, rather than allow an important call to be missed:

```
c mobile 87 s

VK2DOT-1:DOTXR} Trying G8PZT-1

VK2DOT-1:DOTXR} Connected to G8PZT-1

TTYLINK at MOBILE:G8PZT-1
Calling sysop for 90 seconds..
Type 'M' at any time to leave a short message, or 'Q' to quit...
No response, please type a short (1 line) message now...
(or enter a blank line to skip this step)

Can you call me back to discuss xrpi release asap?

Message saved for sysop
Goodbye
VK2DOT-1:DOTXR} Welcome back
```

The message is stored in the sysop's PMS, and a flashing asterisk on the top left corner of the status bar alerts him to its presence. The sysop can then access his PMS to read the message like this:

```
CMD(B/H/K/L/R/S/?)>
l
Msg# Stat Rcvd Time To From Subject
6 PR 22/05 03:25 SYSOP G8PZT TTYLink msg from G8PZT@VK2DOT-1
```

```
CMD(B/H/K/L/R/S/?)>
r 6

Msg#: 6
Rcvd: 22/05 03:25
From: G8PZT
To: SYSOP
Subj: TTYLink msg from G8PZT@VK2DOT-1

Can you call me back to discuss XRPi release asap?

CMD(B/H/K/L/R/S/?)>
```

It's all a bit untidy at present, but will hopefully be tidied up in future revisons.

****

Modified / New Commands (alphabetical order):
==========

This may not be a commplete list:

ACL Command
~~~~~~~~~~

This new command allows the IP Access Control List to be modified on the fly, without having to edit and reload the IPROUTE.SYS file. It has the following sub-commands, revealed by typing ACL by itself:

DENY, LOG, PERMIT, REMOVE, VIEW

Of these ACL PERMIT and ACL DENY were already used in IPROUTE.SYS, and are documented at https://ohiopacket.org/xrpi/docs/ipacl-cmd.htm

Both commands require the following arguments:

<srcip>[/mask][:port] <dstip>[/mask][:port] [proto]
~~~~~~~~~~

Typing "ACL PERMIT" or "ACL DENY" displays the syntax for the command.

The "ACL VIEW" command displays the access control list. The first column shows the "rule number", and the second column is the "action" for that rule, i.e. permit or deny. The order of the rules is important - rules are applied in the order they are shown. For each rule, the source and destination IPs and netmasks are shown, along with protocol number. The service port numbers are only shown if they are non-zero.

The following example has been horizontally compressed to fit the page:

```
acl view
G8PZT:KIDDER} IP Access Control List:

# Action Source IP [Mask] SvcPort Dest IP [Mask] SvcPort Proto
1 Permit 192.168.0.0 [255.255.0.0] 0.0.0.0 [0.0.0.0] 0
2 Permit 44.0.0.0 [255.0.0.0] 0.0.0.0 [0.0.0.0] 0
3 Deny 0.0.0.0 [255.255.255.255] 44.0.0.0 [255.0.0.0] 0
```

The rule number is used by "ACL REMOVE <rule_number>", which removes a rule. After removal, the remaining rules are renumbered.

The "ACL LOG [level]" command displays or sets the ACL logging level. The only levels so far defined are:

```
Level Actions
-----------------------------------------
0 No ACL logging
1 Log denial events
2 Display denial events on IDS window
3 log and display denial events
```

if ACL logging is enabled, ACL events go into the main daily log. Be aware that in some cases this might generate a lot of logging, and in other cases virtually nothing. It depends on how strict your rules are, what your IP routing table is like, how open your system is to the outside world, and how much it

is attacked.

Logging defaults off, but the ACL LOG command may be used in IPROUTE.SYS.

There is currently no mechanism to save a modified ACL back to the IPROUTE.SYS file, as the command is intended only for on-the-fly changes.

CHAT Command
~~~~~~~~~~~~

The CHAT command has been modified to allow easy connection to the chat service on other nodes. The new syntax is:

CH[at] [nodecall | nodealias]

CHAT by itself invokes the local chat server (i.e. the one where the user is logged on), as before.

Using "CHAT <nodecall>" is roughly equivalent to issuing the command
"C <nodecall> 8 S". It attempts a connection to the chat service at <nodecall>. If the target node is not NetromX or chat-capable, the connection is refused.

DISCARD Command
~~~~~~~~~~~~~~~

The DISCARD command has been modified to allow easy connection to the discard service on other nodes. The new syntax is:

DIS[card] [nodecall | nodealias]

DISCARD by itself invokes the local discard server (i.e. the one where the user is logged on), as before.

Using "DISCARD <nodecall>" is roughly equivalent to issuing the command
"C <nodecall> 9 S". It attempts a connection to the discard service at <nodecall>. If the target node is not NetromX or

discard-capable, the connection is refused.

If no argument is supplied, or the argument is the callsign or alias of the node where the user is logged on, the local chat server is invoked instead.

ECHO Command
~~~~~~~~~~~~

The ECHO command has been modified to allow easy connection to the echo service on other nodes. The new syntax is:

EC[ho] [nodecall | nodealias]

ECHO by itself invokes the local echo server, as before.

Using "ECHO <nodecall>" is roughly equivalent to issuing the command
"C <nodecall> 7 S". It attempts a connection to the echo service at <nodecall>. If the target node is not NetromX or echo-capable, the connection is refused.

If no argument is supplied, or the argument is the callsign or alias of the node where the user is logged on, the local echo server is invoked instead.

FTP Command
~~~~~~~~~~~

The FTP command is new. It invokes the FTP client.

Some may wonder why an FTP client has been added, when Linux already has one? The answer is simply, "Because I had a need for it!". I won't elaborate why, just to say it has proved very useful to me. It might even prove useful to you. If not, just ignore it.

One good reason for including an FTP client is that it allows FTP over Amprnet, which is something that's not easy to set up in Linux itself.

The client is started by "FTP" or "FTP <hostname | ipaddr | nickname>",
e.g.

FTP 192.168.1.3
FTP g8pzt.ath.cx
FTP gb7kr

Nicknames are specified in FTPCLI.SYS, along with connection details allowing automated login. This and the keepalives were two of the features I personally find most useful.

XRouter's FTP client currently has 67 commands, all of which have brief help and syntax. It will take a while to fully document them all!

ftp
G8PZT-1:MOBILE} FTP client started

FTP> ?
Commands may be abbreviated. Command list:

? < abort account append
ascii bell binary bye case
cd cdup close dir delete
get hash help idle keep
lcd ldel ldir ls list
lmkdir lpwd lrmdir lren lview
mdelete mget mkdir modtime mput
newer nlist open passive preserve
progress prompt put pwd quit
quote reset restart recv reget
rename rhelp rmdir rstatus runique
send site size sndport status
sunique system timeout type user
verbose view

Use "? <cmd>" for specific help on <cmd>

FTP> ? ldir
LDIR List files in local directory

Syntax: LDI[r] [local-path]

FTP> quit

G8PZT-1:MOBILE} Welcome back

IFACE Command
~~~~~~~~~~~~~

The new IFACE command can be used to add, modify, remove, list, start and stop XRouter's interfaces on the fly. It has the following sub-commands:

A[dd], D[isplay], DR[op], L[ist], STA[rt], STO[p]

The A[dd] sub-command is used to create a new interface, for example to create interface 3 (providing iface 3 doesn't already exist):

IFACE ADD 3

The DR[op] sub-command deletes an interface, for example:

IFACE DROP 3

The L[ist] sub-command lists the interfaces:

iface l

G8PZT:KIDDER} Interfaces:

Num Type Prtcl MTU Description
1 external ether 1064 eth0
2 axudp axudp 256
3 axtcp axtcp 340
(End of list)

The D[is[;ay sub-command displays the details of a single interface:

iface d 1
~~~~~~~~~~~~~

G8PZT:KIDDER} Interface 1:
Type=external, Ptcl=ether, MTU=1064, Descr=eth0
Used by port(s): 1
MAC Address=B8:27:EB:35:5C:5B

The STA[rt] sub-command is used to bring an interface into use, e.g.:

IFACE START 2

The STO[p] sub-command takes an interface out of use, for example:

IFACE STOP 2

You cannot stop an interface which has dependent ports until all those ports have been stopped. At present, stopping an Ethernet interface may cause a segfault.

Certain interface parameters can be changed on the fly, using the general command form:

IFACE <number> <command> [args]

e.g. IFACE 4 ID Paula's Interface

The following parameters are accepted, but as this is a work in progress, they are not all fully functional...

COM CONFIG FLOW ID INTNUM IOADDR KISSOPTIONS MTU NUMBER PROTOCOL SPEED TYPE

(CONFIG, PROTOCOL and MTU are not yet working)

INFO Command
~~~~~~~~~~~~

The INFO command has been modified to allow it to query the INFO server of other nodes. The syntax is now:

I[nfo] [nodecall | nodealias | topic]
~~~~~~~~~~~~

As before INFO by itself displays a list of topics. Sadly, sysops rarely bother to provide any topics!

The "INFO <nodecall>" form attempts a connection to the INFO service on the target node, with the STAY option set.

If the target node is NetromX capable, and has the INFO server, various data about the target node are returned (see "INFO Service" section above). Otherwise the connection is refused.

The point of this is that the provision of basic useful or interesting information about a node requires absolutely no effort by the sysop.

IP Command
~~~~~~~~~~

New sub-commands "IP BAN ADD" and "IP BAN DROP" allow manual editing of the banned IP list.

IP BAN ADD <ipaddr> [netmask]

IP BAN DROP <ipaddr>

The short forms of those commands, IP BAN and IP UNBAN, still work....

IP BAN <ipaddr> [netmask]

IP UNBAN <ipaddr>

The new sub-command "IP BAN SAVE" saves the list of banned IP addresses to the file IPBAN.SYS. The contents of the file are reloaded at boot-up.

The ban list is always saved at closedown, but this command can be used in CRONTAB.SYS to additionallly save the ban list from time to time in case of power cuts.

New sub-commands IP BAN PORT ADD, IP BAN PORT DROP, IP BAN
~~~~~~~~~~

PORT LIST, and IP BAN PORT SAVE allow management of the "honeypot" ports, as shown in the following examples:

```
ip ban port add
G8PZT:KIDDER} Syntax: ADD <TCP|UDP|ALL> <start> [end]

ip ban port add tcp 445
G8PZT:KIDDER} Ok

ip ban port list
G8PZT:KIDDER} Banned (honeypot) ports:
Start End Last-hit Hits Protocols
445 445 xx/xx xx:xx 0 TCP
(End of list)

ip ban port save
G8PZT:KIDDER} Ok

ip ban port drop
G8PZT:KIDDER} Syntax: DROP <TCP|UDP|ALL> <start> <end>

ip ban port drop tcp 445 445
G8PZT:KIDDER} Ok

ip ban port list
G8PZT:KIDDER} Banned (honeypot) ports:
Start End Last-hit Hits Protocols
(End of list)
```

(I think IP BAN PORT ADD was present in earlier versions, but without the portocol field which allows protocol-specific honeypots).

What's a honeypot? In this context, it's a sticky trap, set up on popular TCP or UDP ports, for catching internet low-life.

Hackers generally start their attacks by probing for open TCP ports, and to save time they often start with the most popular ones - telnet, SSH, HTTP, VNC and so on. If they find an open port, they tend to inform each other, then they all concentrate their attacks on that port.

Unless you have a particular service port open, the chances are that anyone who tries to connect to it is up to no good. So the honeypot is a mitigation measure. It LOOKS like an attractive open port, but it's not! Anyone who connects to it gets their IP address logged and banned. After that they can't do any more attacking unless they change their IP address.

OK, it's not foolproof. Nation states and sophisticated hackers have access to virtually unlimited IP addresses. But it slows them down, and the IDS alerts you that there is problem.

When I was running XR16 v187 as my main internet router a few years ago, I put all my sensitive services on non-standard ports, and honeypotted their normal ports. Thus it was very unlikely that anyone would find my FTP or VNC ports for instance, because they always got trapped on port 21 and 5900. It was an extra layer of security.

Now, I guess no-one will be running XRrouter as their main internet router these days, so the honeypot is mainly for protecting XRPi and gathering data on attackers and their tactics.

LOG Command
~~~~~~~~~~~

The LOG command now accepts a numeric argument in addition to the previous ON or OFF.

Syntax: LOG [on | off | n]

Where n is a number between 0 and 65535 formed by adding together the flags corresponding to the desired options. (See LOG Flags above)

For example "LOG 0" disables all logging. "LOG 5" turns on session-layer logging, with CRLF line endings. "LOG 1029" does the same, but adds packet capture.

LOG by itself reports the current state. If logging is
~~~~~~~~~~~

enabled, "LOG OFF" temporarily turns it off, but doesn't change the choice of flags. "LOG ON" restores the previous state.
For example:

```
log
G8PZT:KIDDER} Logging is enabled
Logging level = 1029

log off
G8PZT:KIDDER} Logging is disabled

log on
G8PZT:KIDDER} Logging is enabled
Logging level = 1029
```

Admittedly the terminology and ergomnomics of this command could be improved.

MON Command
~~~~~~~~~~~

There have been some modifications to the MON command. The permitted arguments are now ON, OFF and IDS. Any other argument now displays syntax help instead of the unhelpful "Error!" message.

```
mon ?
G8PZT-1:MOBILE} Syntax: MO[n] [ON | OFF | IDS [on | off]]
```

The MON IDS [on | off] sub-command controls IDS monitoring as follows:

If MON is ON *and* MON IDS is ON, IDS warnings are echoed to the console (or remote monitoring session if you are logged on remotely). This allows you to keep an eye on the IDS while doing other things.

This was mainly intended for use on remote monitorng sessions, so the default for a *remote* monitor session is ON. The default for consoles is OFF.
~~~~~~~~~~~

NFTP Command

The NFTP command is new. It invokes the NFTP client, used to exchange files with other sysops.

Syntax is "NF[tp] <target>"

If <target> is the callsign or alias of the node where the user is logged on, it invokes the local NFTP server at that node. This mode is available to non-sysops, and uses fairly standard FTP server commands, such as LIST, STOR, RETR etc.

If <target> is the callsign or alias of any other node, and the user is a verified sysop, it invokes a client which connects to the target server.

The client uses standard FTP client commands such as DIR, GET, PUT etc.

```
NFTP> help
Commands may be abbreviated. Command list:
? abort ascii binary bye
cd cdup close dir delete
get hash help lcd ldel
ldir ls list lmkdir lpwd
lrmdir lren lview mkdir modtime
nlist open put pwd quit
restart recv rename rmdir send
status user verbose view

Use "HELP <cmd>" for specific help on <cmd>
NFTP>
```

Most of these commands should be familiar to FTP users.

The NFTP client has full access to the local system, so is available only to sysops.

Note that the client uses a "standard" L4 connection to the server, rather than an "extended" (L4X) one. This results in a

rather clumsy connection sequence, but allows other software authors to incorporate NFTP, if they so desire, without needing to implement L4X. The next version of XRLin/XRPi will use L4X if the target node supports it, otherwise it will use normal L4.

A typical usage scenario would be like this:

Sysop ALICE is chatting with sysop BOB on XR sysop chat.
She wants his help, because her TUN interface doesn't work.

Bob says "send me your XROUTER.CFG and I'll have a look".

Alice types "NFTP BOB", then "PUT XROUTER.CFG" to send it, informing BOB via the chatserver.

BOB studies Alice's file, but sadly can't find the error.

Meanwhile sysop CHARLES has downloaded the file from BOB, and spots the error. He corrects it and uses "NFTP ALICE", then "PUT XRNEW.CFG" to send it back to ALICE.

BOB downloads the new file and is enlightened. ALICE copies it from FTP/public to her XRPI working directory and reboots. The TUN interface now works. CHARLES is a genius!

CHARLES writes a useful HOWTO about the TUN interface, and uploads it to ALICE's public directory, from where DICK, EDDIE,
FRANK etc can download it.

No-one needed to know anyone else's email address. No-one needed to copy files from their node machine to an email machine and back. There was no need to set up accounts and passwords. No-one had privileged access to anyone elses machine. It could all be done from XRouter's console.

PCAP Command
~~~~~~~~~~~~

The new PCAP command controls IP packet capture. The syntax
~~~~~~~~~~~~

is:

PC[ap] [on | off]

If enabled, every IP datagram entering or leaving XRLin's IP stack is recorded to a "pcap" file which can be examined with Wireshark.

This can be useful when chasing obscure crashes or doing a security audit. Other authors will no doubt use it to reverse-engineer XRouter's protocols, which incidentally are not secret, just so-far undocumented.

The default packet capture state is controlled by the LOG_PCAP bit (1024) in the argument to the "LOG" command, or in the "LOG=" directive in XROUTER.CFG. If the bit is set by the LOG= directive, packet capture is on by default, otherwise it is off.

Packet capture can generate huge files, especially if you are using FTP to transfer large amounts of data, so it is not recommended for long-term use.

PEERS Command
~~~~~~~~~~~~~

The sysop-only PEERS command, disabled since 2001, has been re-enabled. This command displays the closest NCMP-capable neighbours. At the moment, only XRouters (XR16/XR32/XRLin/XRPi) are NCMP-capable.

peers
G8PZT:KIDDER} Peers:
Nodecall RTT Hop Last-cnfrmd Last-update Latitude Longitude S/ware
G8PZT 0 1 26/05 04:18 26/05 04:18 0000.00N 00000.00E ?
ZL2BAU-3 248 3 26/05 04:10 xx/xx xx:xx 4444.26S 17103.23E ?
VK2DOT-1 31 1 26/05 04:18 xx/xx xx:xx 3323.21S 15121.42E ?
G8PZT-1 6 1 26/05 03:50 26/05 03:50 5008.75N 00507.50W XRLin
VE2PKT-4 8 1 26/05 04:17 xx/xx xx:xx 4625.96N 07237.66W ?
VE3UIM-7 16 1 26/05 04:16 xx/xx xx:xx 4453.53N 07600.74W ?
~~~~~~~~~~~~~

(End of list)

RTT is the smoothed round trip time in 1/100ths of a second, e.g 6 = 60ms. It is a running average, so it may show longer round trip times if traffic is heavy.

Hop is the no. of hops to the peer, where 1 = direct link.

Last-cnfrmd shows when the peer was last confirmed to be NCMP-capable, by the reception of any form of NCMP message from it. Over time, sysops may change from one brand of software to another, keeping the same callsign and alias. So we can't assume that a PEERS entry is valid unless the last-comfirmed date/time is relatively up to date.

Last-update shows the most recent time when the peer responded to a neighbour discovery request.

S/ware show what software the peer is running, if known.

The old DOS and Windows versions of XRouter are NCMP-capable, so they will show up in the peers list. But they use an earlier version of the protocol which doesn't support the "last-update" feature and don't report the software version.

PMS Command
~~~~~~~~~~~

The PMS command has been modified to allow easy connection to the PMS service on other nodes. The new syntax is:

PM[s] [nodecall | nodealias]

PMS by itself invokes the local PMS (i.e. the one where the user is logged on), as before.

Using "PMS <nodecall>" is roughly equivalent to issuing the command
"C <nodecall> 2 S". It attempts a connection to the PMS service at <nodecall>. If the target node is not NetromX or chat-capable, the connection is refused.
~~~~~~~~~~~

PORT Command

The PORT DROP sub-command had been disabled long ago because the code was unfinished and possibly dangerous. The code has now been completed, so the command should be safe to use.

The existing online documentation doesn't mention these PORT subcommands:

PORT ADD <portnum> <ifacenum> <id>
PORT DROP <portnum>
PORT LIST
PORT START <portnum>

So typing "? P[orts]" or "P[orts] ?" now produces the syntax help:

Syntax: P[orts] [A[dd] | D[rop] | L[ist] | S[tart]]

SMS Command

The format and function of this command have been changed. The new syntax is as follows:

SMS [nodecall [text]]

Examples: SMS
SMS G8PZT
SMS G8PZT I'll be on sysop chat at 3pm UTC

The first form invokes the SMS client, from where you view the conversations, read, send, and delete messages.

The second form initiates the sending of a message, and prompts for the message body.

The third form sends a message instantly and returns to the command line.

This is a short messaging service for XRouter sysops, and has nothing to do with telephone SMS. Nor is it to be confused with the SMS command you may remember from the original DOS XRouter (XR16).

History: In that original form, the SMS command sent short messages as UI frames on a specified port. It had been envisaged as part of a system for users to send short messages to anyone, whatever node they were on. Protocols for transporting and delivering the messages were developed in April 2001, but to be truly effective it needed other software authors to incorporate the protocols. Unfortunately it was around that time that the UK packet network began to disintegrate. The number of users plummeted as Broadband Internet became widespread, and no-one seemed to be developing packet any longer. The command persisted mainly for test purposes, and was eventually disabled in 2011.

Nowadays, at least on the XRouter network, it appears there are no "users" as such - only the node operators themselves. But those sysops have a need to communicate with each other. Some of them leave messages for each other on the sysop chat, but that tends to scroll off the screen and get lost. It is also very public, and unecessarily broadcasts the message to every chat server. The PMS can be used, but it isn't very "instant", and has no sense of "conversation"; only a random list of incoming messages.

To try and solve this problem, the SMS idea was resurrected as a sysop-only private instant-messaging system.

Unlike the BBS system, SMS messages are delivered directly from the source node to the recipient node, without intermediate store and forward. Thus it curently only works with nodes that are in your list. However, as most XRouter nodes are known to each other, it's not much of a problem.

There is a "read receipt" mechanism, so the sender knows whether or not his message has been received and read.

The recipent is alerted to the presence of a mesage by a

flashing "S" on the top status bar. He can then type SMS to view the conversation list, which displays the new message(s). If there is more than one new message in a conversation, it displays the first unread one.

```
sms
G8PZT-1:MOBILE} Starting SMS Client..

SMS Conversations:

Nodecall New Date/Time Text
G8PZT-14 0 09/06 02:56 another test back from g8pzt-14
G8PZT 0 18/05 03:27 test at 03:25 tuesday

R)ead S)end A)rchive D)elete E)xit SMS
```

Note that unlike a PMS or BBS, this is CONVERSATION not MESSAGE oriented.

Each {nodecall} in the above list indicates a conversation with the sysop of that node. A conversation consists of one or more messages between you and the other party in strict chronological sequence. Both sides of the conversation are included.

The argument to each command, with the exception of "E)xit", is a {nodecall}, i.e. a conversation. So "A)rchive G8PZT" archives the conversation to a file, removing it from the list.

```
R)ead Reads a conversation
S)end Sends a message to a conversation
A)rchive Removes conversation & archives it to disk.
D)elete Permanently deletes a conversation.
E)xit Exits the client and returns to the node.
```

"R G8PZT" reads the conversation between you and G8PZT, starting at the first unread message. If there are no unread messages it displays the last message.

In this example there are both previous and subsequent

messages, which can be displayed using the Next and P)revious commands:

```
r g8pzt

Sent: 18/05/21 03:27:20
To: G8PZT

test at 03:25 tuesday

S)end, N)ext, P)revious, C)onversations, E)xit SMS
```

The "S)end" option is a little ambiguous in this menu, and shows how unfinished the user interface is!! Things like that would have been ironed out if I hadn't been forced to abandon the project prematurely. It really means "create a new message", not "send the above message".

The C)onversations option returns you to the list of conversations.

STATS Command
~~~~~~~~~~~~~

A new IDS sub-command displays brief IDS (Intrusion Detection System) statistics. This is mainly for use by remote sysops, because all the information is already available to console sysops via the "Security Monitor" window.

```
s ids

G8PZT:KIDDER} Security stats:

TCP Scans: SYN=837 FIN=0 ACK=152 NUL=0 MAI=37 XMS=0 OTH=1632
Bogus SYNs: 2083
Rejected Logins: Telnet=77, Rlogin=0, FTP=7, TelProxy=0
Malicious Logins: Telnet=334, Rlogin=0, FTP=47, TelProxy=0
FTP directory hacks: 0
IDS Notifications: 132682
IP frames from banned senders: 3155
IP Fragmentation attacks: Tiny=5 Huge=0 Overlapped=3
```
~~~~~~~~~~~~~

Smurfs: 0 Fraggles: 0
Honeypot Hits: 27

Note that if your node is AX25-only, or completely hidden behind a firewall, many of these stats will be 0, which is a good thing :-)

The syntax help for STATS was badly out of date, so this has now been updated, and is also displayed if you type an unrecognised stats subcommand, e.g. "S ?"

Syntax: S[tats] [IDS | IP | L1 | L2 | L3 | L4 | TCP | *]

TALK Command
~~~~~~~~~~~~

The TALK command now accepts either a session number or a node callsign, the latter facilitating private sysop-to-sysop chat.

I.e. the new syntax is: TALK <sessnum | nodecall>.

If the argument is a session number, e.g. "talk 123", it allows the sysop to talk to a logged-in user as before. The following applies only to the new mode:

If the argument is a node callsign or alias, e.g. "talk VK2DOT" it initiates a private keyboard to keyboard chat with the sysop of that node, if he is available. This is not related to the "sysop chat" channel on the chat server. It is the NetRom equivalent of the TTYLINK command.

Note this currently only works if the target node is running XRLin or XRPi. Other types of node will not respond to the request.

The sysop of the target node has 90 seconds to respond. At any point during or after the 90 seconds, the caller has the option to leave a short one-line message (160 chars max) or to abort the call. Messages are saved into the sysop's PMS.

If the sysop takes more than 90 seconds to respond, and the
~~~~~~~~~~~~

user has not disconnected, the sysop can still use the "talk <sessnum>" form of the command to initiate a chat with the user.

TCP Command
~~~~~~~~~~~

A new CACHE subcommand has been added to the TCP command, with the following syntax:

TCP CACHE <LIFE | SIZE | STATUS>

The LIFE subcommand displays or sets the lifetime of a cached SYN. The default is 10 seconds.

```
tcp cache life
G8PZT:KIDDER} Lifetime = 5 seconds

tcp cache life 10
G8PZT:KIDDER} Lifetime = 10 seconds
```

The SIZE subcommand displays or sets the maximum no of SYNS that can be simultaneously cached. The deault is 1000. Note you can't have more than 9999 slots, but 9999 slots and a 10 sec lifetime would cope with 999 SYNs per second.

```
tcp cache size
G8PZT:KIDDER} Size = 9999 slots

tcp cache size 1000
G8PZT:KIDDER} Size = 1000 slots
```

The STATUS subcommand displays the status and cache statistics:

```
tcp cache status
G8PZT:KIDDER} SYN cache status:

1264 Qualifying SYNs rcvd
0 Dropped - invalid MSS
0 Dropped - RST flag
```
~~~~~~~~~~~

0 Dropped - cache full
11 Duplicate SYNs
1253 SYNs cached
489 Successful connections
238 Timeouts - valid MSS
526 Timeouts - bad MSS

Cache size: 1000 slots, Slot lifetime: 10 secs
Slots currently active: 0, Peak occupancy: 4

TIME Command
~~~~~~~~~~~~

The TIME command is no longer sysop-only, and has been slighly modified to allow it to query the time servers on other nodes. It still requires sysop privileges to change the time.

The new syntax is:

TI[me] [nodecall | hh:mm]

Examples:

TIME
TIME G8PZT
TIME 12:47

TNC Command
~~~~~~~~~~~

The new "TNC" command is for controlling TNC-style peripherals, e.g. real TNC's or any applications that use plain text commands.

Syntax: TNC <portnum>

Where <portnum> references a PORT whose PROTOCOL is "TNC".

After issuing the TNC command, the session talks directly to

the TNC in plain text mode. This could be used for example to interface with an ARDOP TNC.

WAIT Command
~~~~~~~~~~~~

The new WAIT command is intended for use only in BOOTCMDS.SYS. It causes execution to pause for the specified time. This can be used between commands that need time to execute.

Syntax: WAIT <(secs<60) | (ms>60)>

If the argument to WAIT is less than 60, it is treated as SECONDS, otherwise it is treated as milliseconds.

The following example shows 300ms pauses betwen Linux shell commands when setting up a tunnel interface. The sequence does not work if the pauses are removed.

```
# BOOTCMDS.SYS
#
# For setting up tunnels:
# Assign the address 192.168.0.98 to the Linux end of tun99:
shell ip address add 192.168.0.98 dev tun99
#
wait 300
#
# Tell Linux that 192.168.0.99 is on the other end of tun99:
shell ifconfig tun99 dstaddr 192.168.0.99
#
wait 300
#
# Bring up the Linux end of the interface:
shell ip link set dev tun99 up
```

WX Command
~~~~~~~~~~

The WX command has been slightly modified, to allow it to

query other nodes' weather servers.

As before, typing "WX" by itself lists up to 5 APRS weather stations from which data has been heard, in order of proximity.

Also as before, typing "WX <callsign>" returns the most recent weather information from that station.

However, if <callsign> is not in the list of stations returns by WX", a connection to that node's weather service is attempted. If successful, the weather information is returned, like this:

```
wx g8pzt
G8PZT-1:MOBILE} Trying G8PZT::16...
G8PZT-1:MOBILE} Connected to G8PZT::16

Observed: Mon 17 May 2021 06:01:23 +0001
Pressure: 1007.2 mB
Temperature: 11.2 C
Humidity: 67 %
Wind: 23 mph
Direction: 178 deg
Gust: 31 mph
Rain-24h: 24.7 mm
Rain-Today: 19.7 mm
Rain-1h: 3.2 mm

G8PZT-1:MOBILE} Welcome back
```

The special case "WX LOCAL" returns - yes you've guessed it - the local WX information.

Random Notes:
~~~~~~~~~~~~~

On the "Session Monitor" window, the amount of space available for "watching" a specific session is limited by the terminal height. It's about 3 rows with a 25 row terminal, but grows by one row for every additional terminal row. e.g. a 40-row
~~~~~~~~~~~~~

terminal gives 18 rows of "watch"

Sound Output

Raspberry Pi's and modern laptops do not have the traditional "PC speaker", so up till now XRLin/XRPi have been unable to make the usual sounds. This has now changed. A sound device can be used instead.

To enable sounds (assuming a suitable sound device is present), put the following new directive anywhere in XROUTER.CFG

```
# Audio device for sound output:
# Default OSS device is /dev/dsp (/dev/audio on Rasp pi)
#
AUDIODEVICE=/dev/dsp
#
```

This uses OSS, (a) because I have found OSS to work much better than ALSA, and (b) because ALSA requires libasound to be installed, and I hate dependencies!

In order to use sound you must first run the command

```
sudo modprobe snd-pcm-oss
```

The downside of OSS is that it won't share the audio device, so if you have XRouter's audio enabled but another app is using the sound device, XRouter will fail at boot.