Packet White Paper: 249
Status: Informational
Date: January 2025
Author: Paula Dowie
Updates: PWP245

AX25 Packet Tracing Within XRouter Remote Host Protocol

Abstract

RHP is one of the XRouter API's. This document describes how to set up AX25 packet tracing within an XRouter RHPv2 (Remote Host Protocol version 2) session, and how to interpret the resulting JSON-format messages.

Status of this Memo

This memo provides information for the Packet Radio community. It is aimed at developers who wish to use RHP. It does not propose any form of standard. Discussions and suggestions are welcome. Distribution of this memo is unlimited.

Copyright Notice

Table of Contents

1. Introduction

This section gives a brief overview of RHP, packet tracing, and trace sockets. For more information on RHP, please see “RHP Version 2 AX25 Functionality" [PWP245].

1.1. Brief Overview of RHP

RHP is the acronym for REMOTE HOST PROTOCOL, so called because it allows a "host" application to be located remotely from XRouter, effectively using XRouter as a multi-protocol "packet engine".

It is a client-server protocol in which the server is XRouter. The client is usually, but not necessarily, some sort of application with a human interface.

The client and server interact by passing "messages" via a single, persistent TCP stream. In RHP version 2, those messages use JSON format. This document applies only to version 2.

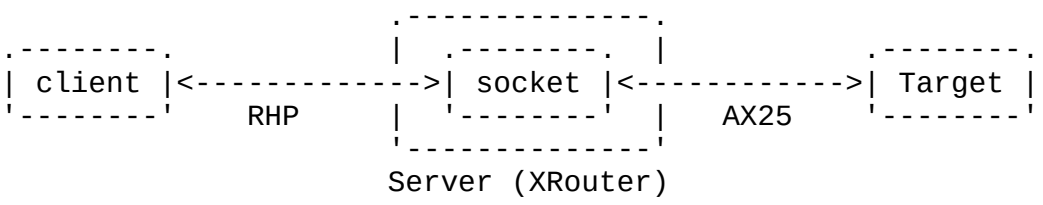
Within "normal" RHP2, JSON messages are "framed" by a simple two-byte "frame length", sent high byte first. e.g. the two bytes 0x01 0x20 indicate that the subsequent 288 bytes are a frame. The framing for RHP within WebSockets is similar, albeit with a more complex header.

The protocol allows clients to access the XRouter protocol stack at several layers. This document concerns only AX25 layer 2.

1.2. RHP Sockets

At the heart of RHP is the concept of "sockets", i.e. communication endpoints. These are functionally similar to Berkeley (BSD) sockets, but with some extra features.

The diagram below depicts a client using RHP to control a socket, and to exchange data with it. In turn the socket is interacting with a target system using the AX25 protocol. Data is passed between the client and target via the socket, whilst RHP control signals pass only between client and server.



Sockets must be "opened" before use, and "closed" after use. When a socket is opened, a numeric "handle" is returned, which must be used

for all subsequent operations via that socket.

Clients may open multiple sockets, provided there is no conflict between them, for example two sockets with exactly the same target address and port.

All sockets "owned" by a client are closed if the client disconnects.

When opening a socket, the client must specify a "protocol family", in this case "AX25", and a "mode".

There are 4 main socket "modes" applicable to AX25, as follows:

```
STREAM     = AX25 connected mode
DATAGRAM   = AX25 unconnected mode (UI frames)
RAW        = AX25 un-decoded raw packet data
TRACE      = AX25 decoded headers plus data
```

It is only the last of these modes with which this document is concerned.

1.2. Packet Tracing

Within the scope of this document, "Packet Tracing" refers to the process of decoding a communication protocol's packets, and presenting them in human-readable form.

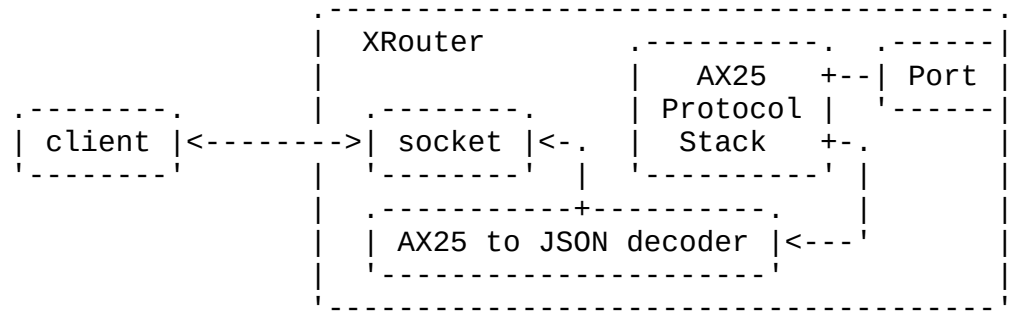
RHP includes a mechanism for tracing packets entering or leaving any of XRouter's communication "ports".

Several different protocols may be traced. This document is only concerned with the AX25 protocol.

1.4. Packet Trace Sockets

These are special sockets which have their MODE set to "trace". There is no equivalent in Berkeley Sockets.

Trace sockets have no "target" address. Instead they watch and decode traffic entering and leaving XRouter via its PORTS. The following diagram is a simplified representation:



In practice there may be multiple ports, multiple protocol stacks, multiple decoders, and multiple sockets per client.

2. Using Packet Trace Sockets

This section describes how to open and use XRouter's AX25 Packet Trace sockets, via Remote Host Protocol. It is assumed that the client has already established an RHP connection with XRouter.

For the purposes of this document, JSON requests and responses are shown in "human-friendly" form. In practice, the whites pace can be omitted.

2.1. Opening Packet Trace Sockets

Clients open AX25 trace sockets using "open" requests, specifying protocol family "ax25" and mode "trace". The client must also specify the number of the XRouter port to be watched, and some flags which control what types of packet are traced. For example, the client would send a JSON request like this:

```
{
"type": "open",
"id": 22,
"pfam": "ax25",
"mode": "trace",
"port": "4",
"flags": 7
}
```

The "id" field is an optional serial number, used to match commands and responses in "pipelined" situations. It may safely be omitted if the client only executes one command at a time.

The value for the "flags" field is the numerical sum of various options. The options for ax25 trace sockets are as follows:

```
0x01     Trace Incoming frames
0x02     Trace Outgoing frames
0x04     Trace Supervisory frames
```

Thus a "flags" value of 1 would trace only incoming information and unnumbered information frames, whereas a value of 7 would trace all incoming and outgoing frames.

2.2. Successful Open

If the socket is successfully opened, XRouter sends back a reply similar to this, containing the handle of the new socket:

```
{
"type": "openReply",
"id": 22,
"handle": 3,
"errcode": 0,
"errtext": "ok"
}
```

Thereafter, XRouter will send JSON messages with "type": "recv" to the client whenever a packet is traced. The format of these

messages is discussed later in this document.

2.3. Unsuccessful Open

If the socket cannot be opened, e.g. because the specified PORT doesn't exist, XRouter returns an error message to the client, similar to this:

```
{
"type": "openReply",
"id": 22,
"errcode": 10,
"errtext": "No such port"
}
```

A full list of error codes can be found in PWP 245

2.4. Closing Packet Trace Sockets

Packet trace sockets are closed in exactly the same way as any other type of RHP socket, i.e. by sending an RHP "close" request. In RHP2 this consists of a JSON message similar to this:

```
{
"id": 23,            (optional)
"type": "close",     (required)
"handle": 3          (required)
}
```

2.5. Successful Close

If the socket is successfully closed, XRouter replies with a "closeReply" message, similar to this:

```
{
"id": 23,
"type": "closeReply",
"handle": 3,
"errcode":0,
"errtext":"Ok"
}
```

2.6. Unsuccessful Close

If the close request was unsuccessful, e.g. because the specified handle did not refer to an open socket, XRouter responds with an error message similar to this:

```
{
"id": 23,
"type": "closeReply",
"handle": 0,
"errcode": 12,
"errtext": "Invalid handle"
}
```

2.7. List of Error Messages

The following error messages are applicable to TRACE sockets.

You are advised to parse the error CODE, not the error text, as the latter may change in future versions.

Code	Text	Notes
0	"Ok"	No error
1	"Unspecified"	Catch-all error, might be transient
2	"Bad or missing type"	Unrecognised frame type, don't retry
3	"Invalid handle"	Invalid socket handle, don't retry
4	"No memory"	No memory, try later
5	"Bad or missing mode"	Invalid "mode" in SOCKET or OPEN
8	"Bad or missing family"	Unsupported address family
9	"Duplicate socket"	Socket / connection already exists
10	"No such port"	Invalid port number in OPEN
11	"Invalid protocol"	(in OPEN or SOCKET)
12	"Bad parameter"	Bad or missing parameter
14	"Unauthorised"	Request requires AUTHorisation
16	"Operation not supported"	e.g. SEND on a TRACE socket

3. Packet Trace Message Formats

This section describes the format of the JSON messages in detail. As mentioned above, white space is included for clarity and may be safely omitted.

3.1. Example Packet Trace

The "AX25" in "AX25 Trace Socket" refers to the protocol layer at which the packets are sampled. AX25 packets may contain other protocols, such as NetRom, as shown in the following example:

```
{
"type": "recv",
"handle": 5,
"action": "sent",
"port": 2,
"srce": "G8PZT-1",
"dest": "G8PZT",
"ctrl": 140,
"frametype": "I",
"rseq": 4,
"tseq": 6,
"cr": "C",
"ilen": 33,
"pid": 207,
"ptcl": "NET/ROM",
---- this section present for ptcl=NetRom frames only -----
"l3type": "NetRom",
"l3src": "G8PZT-1",
"l3dst": "G8PZT",
"ttl": 25,
"l4Type": "INFO",
```

```
   "toCct": 16199,
   "txSeq": 0,
   "rxSeq": 0,
   ------------------------------------------------------------
   "data": "Hello World!\r"
   }
```

The above frame might conventionally be displayed as follows:

```
   *** Port 2 sent:
   AX25: G8PZT-1>G8PZT <I C R4 S6> ilen=33 pid=207 NET/ROM
     NTRM: G8PZT-1 to G8PZT  ttl=25 cct=16199 <INFO S1 R1>:
     Hello World!
```

At the date of writing, the AX25 trace sockets in XRouter v504b only trace up to NetRom layer 4. Other protocols will be traced in future versions.

3.2. Layer 2 (AX25) Trace Fields

These are all the possible fields which may occur within an AX25 Layer 2 trace. Most of these fields will be present in all frames, but fields such as "digis", and all those below "frameType" may be present only in some frames.

Field	Type	Comments
"type"	string	RHP frame type - always "recv"
"handle"	integer	Handle of the trace socket
"action"	string	"sent" or "rcvd" (*1)
"port"	integer	XRouter port number
"srce"	string	Source callsign+ssid, e.g. "G8PZT-1"
"dest"	string	Destination callsign+ssid, e.g "M0AHN-9"
"digis"	string	List of digipeater calls (*2)
"ctrl"	integer	AX25 control field numeric value. (*3)
"frametype"	string	Frame type mnemonic, e.g. "I", "RR" (*4)
"rseq"	integer	Receive sequence number (i.e. expected)
"tseq"	integer	Transmit sequence number
"cr"	string	Command/Response bit (*5)
"pf"	string	Poll/Final bit, either "P" or "F" (*6)
"ilen"	integer	Length of information field (*7)
"pid"	integer	Next layer protocol number in decimal (*8)
"ptcl"	string	Next layer protocol in words (*9)
"data"	string	L2 payload. Only present if "ptcl" is "DATA"

(*1) "sent" indicates outgoing packets, i.e. those transmitted by XRouter onto the air or onto the packet network. "rcvd" indicates incoming packets, i.e. those received off-air or from the packet network.

(*2) The "digis" field is only present if there are digipeaters in the destination path. If present, the field value is a JSON array of digipeater objects. Each digipeater object contains two fields as follows:

Field	Type	Comments
"digiCall"	string	Digipeater's callsign+SSID
"repeated"	boolean	True if digi repeated the packet

(*3) Applications may wish to decode the control field themselves instead of using the strings provided by XRouter.

(*4) Possible values for the L2 "frameType" field:

Mnemonic	Meaning
"SABME"	Set Asynchronous Balanced Mode Extended
"C"	Non-extended connnect request (AKA SABM)
"D"	Disconnect Request
"DM"	Disconnected Mode / Busy
"UA"	Unnumbered Acknowledgement
"UI"	Unnumbered Information frame
"I"	Numbered information frame
"FRMR"	Frame Reject (serious error)
"RR"	Receiver Ready
"RNR"	Receiver Not Ready
"REJ"	Reject (Frame not the expected one)
"?"	Unknown type

(*5) Possible values for the "cr" (Command/Response) field are as follows:

"C"	Command
"R"	Response
"V1"	AX25 version 1

(*6) Possible values for the "pf" (Poll/Final) field are as follows:

"P"	Poll
"F"	Final

(*7) The "ilen" field is only present in frame types "I" and "UI".

(*8) The "pid" value includes the two high order bits which are usually set to one, but are sometimes used for layer 2 fragmentation.

(*9) Possible values for the "ptcl" (Layer 3 protocol) field:

Mnemonic	Meaning
"SEG"	Intermediate segment of a fragmented packet
"DATA"	No layer 3, i.e. payload contains normal data
"NET/ROM"	Payload contains NetRom/INP3 information
"IP"	Payload contains IP datagram or part thereof
"ARP"	Payload contains ARP data
"?"	Unknown layer 3 protocol

3.3. Layer 3/4 (NetRom) Trace Fields

These fields are only present if PID is 207 (ptcl=Net/ROM). Some of these fields are only present in a few frame types.

Field	Type	Comments
"l3type"	string	Layer 3 frame type (see below) (*1)
"l3src"	string	layer 3 source callsign
"l3dst"	string	Layer 3 destination callsign
"ttl"	integer	Layer 3 Time To Live
"l4type"	string	NetRom L4 Frame Type (*2)
"fromCct"	integer	Source circuit number (*3)
"toCct"	integer	Destination circuit number (*4)
"txSeq"	integer	Transmit sequence number (*6)
"rxSeq"	integer	Receive sequence number (*7)
"infoLen"	integer	Information length (INFO frames only)
"srcUser"	string	Callsign of originating user (*3)
"srcNode"	string	Callsign of originating user's node (*3)
"service"	integer	NetRomX service number (*5)
"window"	integer	Proposed window (*3)
"accWin"	integer	Acceptable window ("CONN ACK" only)
"l4t1"	integer	Layer 4 T1 timer in seconds (*3)
"bpqSpy"	integer	BPQ extension (*3)
"chokeFlag"	boolean	True if CHOKE flag is set
"nakFlag"	boolean	True if NAK flag is set
"moreFlag"	boolean	True if MORE flag is set

(*1) layer 3 frame types for "l3type":

Value	Meaning
"NetRom"	Netrom L4 control/data
"Routing info"	Layer 3 routing data
"Routing poll"	Request for nodes broadcast
"Unknown"	Shouldn't happen

(*2) Possible values for "l4type":

Value	Meaning
"CONN REQ"	Connect Request
"CONN REQX"	Extended Connect Request
"CONN ACK"	Connection Acknowledgement
"CONN NAK"	Connection Negative Ack (refusal)
"DISC REQ"	Disconnect request
"DISC ACK"	Disconnect Acknowledgement
"INFO"	Information-bearing frame
"INFO ACK"	Acknowledgement for an INFO frame.
"RSET"	Circuit Reset (kill)
"PROT EXT"	Protocol Extension (e.g. IP, NCMP etc)
"unknown"	Unrecognised type (shouldn't happen)

Protocol extension indicates that the frame does NOT contain Netrom layer 4 control or data.

(*3) These fields are only present in L4 frame types "CONN REQ" and "CONN REQX".

(*4) The "toCct" field is only present in L4 frame types "CONN ACK", "INFO", "INFO ACK", "DISC REQ" and "DISC ACK".

(*5) Field "service" is only present in L4 frame type "CONN REQX".

(*6) The "txSeq" field is only present in "INFO" frames.

(*7) The "rxSeq" field is only present in "INFO" and INFO ACK" frames.

4. Security Considerations

Packet traces may contain sensitive data such as IP addresses, usernames, passwords, clues to LAN topology etc. It is therefore recommended that RHP is not opened up to the wider Internet.

By default, RHP allows unauthorised connections from LAN addresses, but not from non-LAN addresses.

Clients with non-LAN IP addresses MUST send a valid AUTH packet before they can access any other RHP commands. The AUTH packet MUST contain a username/password pair which matches one stored in USERPASS.SYS.

Non-LAN clients MAY be granted access without AUTH by including the client's IP address in ACCESS.SYS. Beware of granting such access to ranges of IP addresses, unless you have control of that range.

5. References

[PWP144] Dowie, P., "Remote Host Protocol", PWP 144, December 2004.

[PWP222] Dowie, P., "Remote Host Protocol Version 2", PWP 222, June 2023.

[PWP245] Dowie, P., "RHP Version 2 - AX25 Functionality", PWP 245, May 2024.

6. Author's Address

Paula Dowie
g8pzt[at]blueyonder.co.uk (replace ‘[at]’ with ‘@’)