Packet White Paper: 232
Status: Informational
Date: July 2023
Author: Paula G8PZT

XRouter MQTT API For Raw Packet I/O

Status of This Memo

This memo provides information for the Packet Radio community. It describes an existing API, but does not propose a standard. Distribution of this memo is unlimited.

Abstract

This memo describes part of the Application Programming Interface (API) of the "XRouter" networking software. This fragment of the larger API is concerned with the sending and receiving of "raw" packets, in which client <> server interactions make use of the MQTT protocol.

Table of Contents

1. Introduction

XRouter's "raw packet" API allows MQTT client applications to send and receive raw binary packets at different ISO layers. This could be used for example to trace / log all channel activity, send and receive APRS, build a network crawler, or even a complete MQTT based node, using XRouter as a dumb "packet engine" controlling the TNCs.

1.1. Raw Packets

In this context "raw" packets are self-contained blocks of binary data, consisting of a "header" portion and a "payload" portion. The format of the packet varies according to the ISO layer.

For example, a raw AX25 packet contains at least 7-byte source and destination callsigns, an optional list of digipeater callsigns, a one or two-byte "control" field, and for information-bearing packets there will also be a PID (Protocol IDentifier) plus the payload data.

XRouter currently provides access to raw packets at the KISS, AX25, NetRom and IPv4 layers. Such packets may originate / destinate "on-air", or within XRouter itself.

1.2. Enabling the API

In order to use the raw packet API, the application must connect to XRouter's inbuilt MQTT broker. The broker is enabled using a non-zero MQTTPORT directive in XROUTER.CFG. The usual TCP port for MQTT is 1883.

1.3. Sending Raw Packets

To send a raw packet, the client simply publishes it to the appropriate MQTT topic. For example, to send a raw AX25 packet, the topic would be:

```
xrouter/put/{nodecall}/ax25/{portnum}
```

where:

{nodecall} is the node callsign (including SSID if non-zero), e.g. G8PZT-1

{portnum} is the number of the XRouter PORT upon which the packet is to be sent.

(Note that XRouter identifies PORTs by number, but the topic format does not preclude the use of other forms of port identifier, e.g. "VHF1", if this protocol was to be adopted by other node software.)

The broker does not respond to the client, unless the client specifies a non-zero Quality of Service (QOS). In which case the broker returns a PUBACK if the publish was successful.

1.4. Receiving Raw Packets

To receive raw packets from XRouter, an MQTT client connected to XRouter's broker must first "subscribe" to a "topic", by sending an MQTT "subscribe" message.

For example, to enable the reception of "incoming" KISS frames, i.e. those received off air by an interface, the general form of the topic would be as follows:

```
xrouter/kiss/{nodecall}/rcvd/{ifacenum}

where: {ifacenum} is the INTERFACE number.
```

Thereafter, the broker publishes all the KISS frames received via the specified interface, using the same topic. For example if the {nodecall} was G8PZT-1 and the interface number was 3, received frames would be published to the topic:

```
xrouter/kiss/G8PZT-1/rcvd/3
```

To enable the reception of "outgoing" KISS frames, i.e. those transmitted to air via an interface, the general form of the topic would be as follows:

```
xrouter/kiss/{nodecall}/sent/{ifacenum}
```

1.5. Wildcards

The MQTT multi-level wildcard (#) may be used in topics, for example:

```
xrouter/kiss/G8PZT-1/rcvd/#  - Rcvd frames from all interfaces

xrouter/kiss/G8PZT-1/#       - All sent and rcvd frames
```

The single-level wildcard is not yet supported.

2. Interfacing at KISS Layer

Raw KISS frames begin with a "control" byte, which defines the type of data that follows it. This may be data, TNC settings, polls, ACKs etc.

For KISS "data" frames, the upper nybble indicates the "channel" (0-15) on the TNC, if the TNC supports it. Note that the "TNC" may be a real hardware device, or just a piece of software. In XRouter, a TNC is considered to be an INTERFACE with the outside world, and each of its "channels" may be associated with an XRouter PORT.

2.1. Enable Reception of Incoming KISS Frames

Topic: xrouter/kiss/{nodecall}/rcvd/{ifacenum}

Subscribing to this topic causes XRouter to publish MQTT frames, to the same topic, whose payloads contain KISS frames received on the specified interface. These frames include the KISS "control" byte, but do not include the KISS framing and escape sequences.

2.2. Enable Reception of Outgoing KISS Frames

Topic: xrouter/kiss/{nodecall}/sent/{ifacenum}

Subscribing to this topic causes XRouter to publish MQTT frames, to the same topic, whose payloads contain KISS frames transmitted on the specified interface. These frames include the KISS "control" byte, but do not include the KISS framing and escape sequences.

2.3. Transmit a KISS Frame

Topic: xrouter/put/{nodecall}/kiss/{ifacenum}

Payload: Raw KISS, including "control" byte, but excluding KISS framing and escape sequences.

Publishing to this topic causes the KISS payload to be transmitted on the specified interface. No response is returned, other than a PUBACK if QOS is greater than 0.

3. Interfacing at AX25 Layer 2

Frames transmitted or received at this layer may contain higher level protocols such as NetRom, IP, INP3 etc. The interface has no concept of state, so applications are free to implement their own LAPB layer if so desired.

3.1. Enable Reception of Incoming AX25 Frames

Topic: xrouter/ax25/{nodecall}/rcvd/{portnum}

Subscribing to this topic causes XRouter to publish MQTT frames, to the same topic, whose payloads contain AX25 frames received on the specified port. Frames do not include HDLC framing or CRC.

3.2. Enable Reception of Outgoing AX25 Frames

Topic: xrouter/ax25/{nodecall}/sent/{portnum}

Subscribing to this topic causes XRouter to publish MQTT frames, to the same topic, whose payloads contain AX25 frames transmitted on the specified port. Frames do not include HDLC framing or CRC.

3.3. Transmit an AX25 Frame.

Topic: xrouter/put/{nodecall}/ax25/{portnum}

Payload: Raw AX25, without CRC or HDLC framing.

Publishing to this topic causes the AX25 payload to be transmitted on the specified port. No response is returned, other than a PUBACK if QOS is greater than 0.

4. Interfacing at NetRom Datagram Layer

Frames transmitted or received at this layer include only "routable" NetRom L3 datagrams. L3RTT, INP3, and Nodes broadcasts are not part of this layer, and can be handled via the raw AX25 interface. There is no concept of "port" at the datagram layer, where everything goes via the NetRom router.

A NetRom "datagram" consists of a 15 byte header portion, and at least 5 bytes of payload. The header contains layer 3 source and destination callsigns, plus a TTL (Time To Live) byte.

4.1. Enable Reception of Incoming NetRom Datagrams

Topic: xrouter/nr3b/{nodecall}/rcvd

Subscribing to this topic causes XRouter to publish MQTT frames, to the same topic, whose payloads contain received NetRom datagrams.

4.2. Enable Reception of Outgoing NetRom Datagrams

Topic: xrouter/nr3b/{nodecall}/sent

Subscribing to this topic causes XRouter to publish MQTT frames, to the same topic, whose payloads contain outgoing (sent) NetRom datagrams.

4.3. Transmit A NetRom Datagram.

Topic: xrouter/put/{nodecall}/nr3b

Payload: Raw NetRom.

Publishing to this topic causes the NetRom payload to be transmitted via whatever route the NetRom router deems is best. No response is returned, other than a PUBACK if QOS is greater than 0.

5. Interfacing at IPV4 Layer

This layer sends and receives raw IP version 4 datagrams. Such datagrams may contain any IP protocol, such as ICMP, UDP, TCP etc. Link-layer protocol headers are removed upon receive and added upon transmit. Incoming fragments are not reassembled. Outgoing datagrams are routed according to XRouter's routing table, and are fragmented to suit the interface MTU, providing the DF (Don't Fragment) bit is not set. If the DF bit is set and the datagram is too big for the egress interface, it is silently discarded.

5.1. Enable Reception of Incoming IPv4 Datagrams

Topic: xrouter/ipv4/{nodecall}/rcvd

Subscribing to this topic causes XRouter to publish MQTT frames, to the same topic, whose payloads contain received IPv4 datagrams.

5.2. Enable Reception of Outgoing IPv4 Datagrams

Topic: xrouter/ipv4/{nodecall}/sent

Subscribing to this topic causes XRouter to publish MQTT frames, to the same topic, whose payloads contain Outgoing (sent) IPv4 datagrams.

5.3. Transmit an IPv4 Datagram.

Topic: xrouter/put/{nodecall}/ipv4

Payload: Raw IPv4 datagram

Publishing to this topic causes the IPv4 payload to be transmitted via whatever route XRouter's IP router deems is best. No response is returned, other than a PUBACK if QOS is greater than 0.

6. Limitations

At the time of writing, this implementation has a few minor limitations. For example, the MQTT broker does not yet support Quality of Service levels 2 and 3. And it does not yet support the single-level wildcard (+). These limitation will be rectified in later versions of XRouter.

7. Security Considerations

At the time of writing, there is no security on the MQTT server, as it was intended solely for use on a LAN. As this API can send and receive AX25, NetRom and IP packets, the MQTT server MUST NOT be exposed to the open Internet.

8. References

[MQTT-OASIS-Standard-v5] Banks, A., Ed., Briggs, E., Ed., Borgendale, K., Ed., and R. Gupta, Ed., "MQTT Version 5.0", OASIS Standard, March 2019, <https://docs.oasis-open.org/mqtt/mqtt/v5.0/mqtt-v5.0.html>.

9. Author's Address

Please send comments, criticisms, suggestions, hate mail etc. to the following email address:

g8pzt[at]blueyonder.co.uk