By Matti Aarnio, OH2MQK, Version 0.11

Table of Contents

1 APRX software Requirement Specification

This is *Requirement Specification* for a software serving in Amateur Radio APRS service.

Reader is assumed to be proficient with used terminology, and they are not usually explained here.

1.1 Purpose:

This describes algorithmic, IO-, and environmental requirements for a software doing any combination of following three tasks related to APRS service:

1. Listen on messages with a radio, and pass them to APRSIS network service
2. Listen on messages with a radio, and selectively re-send them on radio
3. Receive messages from APRSIS network, and after selective filtering, send some of them on radio

1.2 Usage Environments:

The **aprx** software can be used in several kinds of environments to handle multiple tasks associated with local APRS network infrastructure tasks.

1. License-free Receive-Only (RX) IGate, to add more “ears” to hear packets, and to pipe them to APRSIS
2. Licensed bidirectional IGate, selectively passing messages from radio channels to APRSIS, and from APRSIS to radio channels, but not repeating packets heard on a radio channel back to a radio channel.
3. Licensed bidirectional IGate plus selectively re-sending of packets heard on radio channels back to radio channels
4. Licensed system for selectively re-sending of packets heard on radio channels back to radio channels, and this without bidirectional IGate service.
5. Licensed system for selectively re-sending of packets heard on radio channels back to radio channels, and doing with with “receive only” IGate, so passing information heard on radio channel to APRSIS, and not the other way at all.

In more common case, there is single radio and single TNC attached to digipeating (re-sending), in more challenging cases there are multiple receivers all around, and very few transmitters. Truly challenging systems operate on multiple radio channels. As single-TNC and single-radio systems are just simple special cases of these complex systems, we consider the complex ones:

1. 3 different frequencies + in use, traffic is being relayed in between them, and the APRSIS network.
2. On each frequency there are multiple receivers, and one well placed transmitter.

2 Treatment rules:

Generally: All receivers report what they hear straight to APRSIS, after small amount of filtering of junk messages, and things which explicitly state that they should not be sent to APRSIS.

General rules for these receiving filters are described here:

http://www.aprs-is.net/IGateDetails.aspx

Gate all packets heard on RF to the Internet (Rx-IGate) EXCEPT

1. 3rd-party packets (data type '}').
   3rd-party packets should have all before and including the data type stripped and then the packet should be processed again starting with step 1 again.
2. generic queries (data type '?').
3. packets with TCPIP, TCPXX, NOGATE, or RFONLY in the header (last 2 are optional).

Gate message packets and associated posits to RF (Tx-IGate) if

1. the receiving station has been heard within range within a predefined time period (range defined as digi hops, distance, or both).
2. the sending station has not been heard via RF within a predefined time period (packets gated from the Internet by other stations are excluded from this test).
3. the sending station does not have TCPXX, NOGATE, or RFONLY in the header.
4. the sending station has not been heard via the Internet within a predefined time period.
   A station is said to be heard via the Internet if packets from the station contain TCPIP* or TCPXX* in the header or if gated (3rd-party) packets are seen on RF gated by the station and containing TCPIP or TCPXX in the 3rd-party header (in other words, the station is seen on RF as being an IGate).

Gate all packets to RF based on criteria set by the sysop (such as call sign, object name, etc.).

With more advanced looking inside frames to be relayed, both the digipeater and Tx-IGate can use filtering rules, like “packet reports a position that is within my service area.”

From multiple receivers + single (or fewer) transmitter(s) follows, than when a more usual system does not hear what it sent out itself, this one will hear, and its receivers must have a way to ignore a frame it sent out itself a moment ago.

Without explicit “ignore what I just sent” filtering, an APRS packet will get reported twice to APRSIS:

rx -> igate-to-aprsis + digi -> tx -> rx -> igate-to-aprsis + digi (dupe stops)

Digipeating will use common packet duplication testing to sent similar frame out only once per given time interval (normally 30 seconds.)

2.1 Low-Level Transmission Rules:

These rules control repeated transmissions of data that was sent a moment ago, and other basic transmitter control issues, like persistence. In particular the persistence is fine example of how to efficiently use radio channel, by sending multiple small frames in quick succession with same preamble and then be silent for longer time.

1. Normal digipeater duplicate packet detection compares message source, destination, and payload data against another expiring storage called “duplicate detector”. Digipeater and Tx-IGate will ignore packets finding a hit in this storage.
2. A candidate packet is then subjected to a number of filters, and if approved for it, the packet will be put on duplicate packet detection database (one for each transmitter.)
3. Because the system will hear the packets it sends out itself, there must be a global expiring storage for recently sent packets, which the receivers can then compare against. (Around 100 packets of 80-120 bytes each.) This storage gets a full copy of packet being sent out – a full AX.25 frame.

Also, transmitters should be kept in limited leash: Transmission queue is less than N seconds (< 5 ?), which needs some smart scheduling coding, when link from computer to TNC is considerably faster.

2.2 Low-Level Receiving Rules:

1. Received AX.25 packet is compared against “my freshly sent packets” storage, matched ones are dropped.
2. Received packet is validated against AX.25 basic structure, invalid ones are dropped.
3. Received packet is validated against Rx-IGate rules, invalid ones are dropped (like when a VIA-field contains invalid data.)
4. Packet may be rejected for Rx-IGate, but it may still be valid for digipeating! For example a 3rd party frame is OK to digipeat, but not to Rx-IGate to APRSIS!

2.3 Digipeater Rules:

Digipeater will do following:

1. Count number of hops the message has so far done
2. Figure out the number of hops the message has been requested to do (e.g. “OH2XYZ-1>APRS,OH2RDU*,OH2RDK*,WIDE7-5: ...” will report that there was request of 7 hops, so far 2 have been executed.)
3. If either of previous ones are over configured limit, the packet is ignored.
4. Multiple filters look inside the message, and can enforce a rule of “repeat only packets within my service area.”
5. Xxxx ???

2.4 Additional Tx-IGate rules:

The Tx-IGate can have additional rules for control:

1. Multiple filters look inside the message, and can enforce a rule of “repeat only packets within my service area,” or to “limit passing message responses only to destinations within my service area”
2. Xxxx ?