

APRX Software Requirement Specification

Table of Contents

1	APRX Software Requirement Specification.....	2
1.1	Purpose:.....	2
1.2	Usage Environments:.....	3
2	Treatment rules:.....	4
2.1	Basic IGate rules:.....	4
2.2	Low-Level Transmission Rules:.....	5
2.3	Low-Level Receiving Rules:.....	6
2.4	Additional Tx-IGate rules:.....	6
2.5	Digipeater Rules:.....	7
2.6	Radio Interface Statistics Telemetry.....	8
2.7	Individual Call-Signs for Each Receiver, or Not?.....	9
2.8	Beaconing.....	10
2.8.1	Radio Beaconing.....	10
2.8.2	Network beaconing.....	10

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. Listen on messages with a radio, and selectively re-send them on radios on other frequencies
4. Receive messages from APRSIS network, and after selective filtering, send some of them on radio

Existing *aprx* software implements Receive-Only (Rx) IGate functionality, and the purpose of this paper is to map new things that it will need for extending functionality further.

23

24 **1.2 Usage Environments:**

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

27 On following one should remember that amateur radio transmitters need a licensed
28 owner/operator/license themselves, but receivers do not:

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

41

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

- 47 1. 3 different frequencies in use, traffic is being relayed in between them, and the
48 APRSIS network.
- 49 2. On each frequency there are multiple receivers, and one well placed transmitter.
- 50 3. Relaying from one frequency to other frequency may end up having different rules,
51 than when re-sending on same frequency: Incoming packet retains traced paths,
52 and gets WIDEN-N/TRACEN-N requests replaced with whatever sysop wants.

53

54

55 2 Treatment rules:

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

59 2.1 Basic IGate rules:

60 General rules for these receiving filters are described here:

61 <http://www.aprs-is.net/IGateDetails.aspx>

62
63 Gate all packets heard on RF to the Internet (Rx-IGate) EXCEPT

- 64 1. 3rd party packets (data type '}') should have all before and including the data
65 type stripped and then the packet should be processed again starting with
66 step 1 again. There are cases like D-STAR gateway to APRS of D-STAR
67 associated operator (radio) positions.
- 68 2. generic queries (data type '?').
- 69 3. packets with TCPIP, TCPXX, NOGATE, or RFONLY in the header, especially
70 in those opened up from a 3rd party packets.

71
72 Gate message packets and associated posits to RF (Tx-IGate) if

- 73 1. the receiving station has been heard within range within a predefined time
74 period (range defined as digi hops, distance, or both).
- 75 2. the sending station has not been heard via RF within a predefined time
76 period (packets gated from the Internet by other stations are excluded from
77 this test).
- 78 3. the sending station does not have TCPXX, NOGATE, or RFONLY in the
79 header.
- 80 4. the receiving station has not been heard via the Internet within a predefined
81 time period.

82 A station is said to be heard via the Internet if packets from the station
83 contain TCPIP* or TCPXX* in the header or if gated (3rd party) packets are
84 seen on RF gated by the station and containing TCPIP or TCPXX in the 3rd
85 party header (in other words, the station is seen on RF as being an IGate).

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

88
89 With more advanced looking inside frames to be relayed, both the digipeater and Tx-IGate
90 can use filtering rules, like “packet reports a position that is within my service area.”

91

92

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

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

98 rx \Rightarrow igate-to-aprsis + digi \Rightarrow tx \Rightarrow rx \Rightarrow igate-to-aprsis + digi (dupe filter stops)

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

101

102 **2.2 Low-Level Transmission Rules:**

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

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

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

120 Original KISS interface is defined as “best effort”: if TNC is busy while host sends a frame,
121 the frame will be discarded, and “upper layers” will resend. In APRS Digipeating, the
122 upper layer sends the packet once, and then declares circa 30 second moratorium on
123 packets with same payload.

124

125

126 **2.3 Low-Level Receiving Rules:**

- 127 1. Received AX.25 packet is compared against “my freshly sent packets” storage, and
128 matched ones are dropped. (Case of one/few transmitters, and multiple receivers
129 hearing them.)
- 130 2. Received packet is validated against AX.25 basic structure, invalid ones are
131 dropped.
- 132 3. Received packet is validated against Rx-IGate rules, forbidden ones are dropped
133 (like when a VIA-field contains invalid data.)
- 134 4. Packet may be rejected for Rx-IGate, but it may still be valid for digipeating!
135 For example a 3rd party frame is OK to digipeat, but not to Rx-IGate to APRSIS!
136 Also some D-STAR to APRS gateways output 3rd party frames, while the original
137 frame is quite close to an APRS frame.

138 Divide packet rejection filters to common, and destination specific ones.

139 **2.4 Additional Tx-IGate rules:**

140 The Tx-IGate can have additional rules for control:

- 141 1. Multiple filters look inside the message, and can enforce a rule of “repeat only
142 packets within my service area,” or to “limit passing message responses only to
143 destinations within my service area”
- 144 2. Basic gate filtering rules:
 - 145 1. the receiving station has been heard within range within a predefined time
146 period (range defined as digi hops, distance, or both).
 - 147 2. the sending station has not been heard via RF within a predefined time period
148 (packets gated from the Internet by other stations are excluded from this test).
 - 149 3. the sending station does not have TCPXX, NOGATE, or RFONLY in the header.
 - 150 4. the receiving station has not been heard via the Internet within a predefined time
151 period.

152 A station is said to be heard via the Internet if packets from the station contain
153 TCPIP* or TCPXX* in the header or if gated (3rd-party) packets are seen on RF
154 gated by the station and containing TCPIP or TCPXX in the 3rd-party header (in
155 other words, the station is seen on RF as being an IGate).

- 156 3. More rules ?

157

158

159

160 **2.5 Digipeater Rules:**

161 Digipeater will do following for each transmitter:

- 162 1. Compare candidate packet against duplicate filter, if found, then drop it. (Low-level
163 transmission rules, number 1)
- 164 2. Count number of hops the message has so far done, and...
- 165 3. Figure out the number of hops the message has been requested to do
166 (e.g. "OH2XYZ-1>APRS,OH2RDU*,WIDE7-5: ..." will report that there was request
167 of 7 hops, so far 2 have been executed – one is shown on trace path.)
- 168 4. If either of previous ones are over any of configured limits, the packet is dropped.
- 169 5. FIXME: WIDEn-N/TRACEn-N treatment rules: By default treat both as TRACE,
170 have an option to disable TRACE mode. Possibly additional keywords for cross-
171 band digipeating?
- 172 6. Multiple filters look inside the message, and can enforce a rule of "repeat only
173 packets within my service area."
- 174 7. FIXME: Cross frequency digipeating? Treat much like Tx-IGate?
175 Relaying from one frequency to other frequency may end up having different rules,
176 than when re-sending on same frequency: Incoming packet retains traced paths,
177 and gets WIDEn-N/TRACEn-N requests replaced with whatever sysop wants.
- 178 8. Cross band relaying may need to add both an indication of "received on 2m", and
179 transmitter identifier: "sent on 6m":
180 "OH2XYZ-1>APRS,RX2M*,OH2RDK-6*,WIDE3-2: ..."
181
182 This "source indication" may not have anything to do with real receiver identifier,
183 which is always shown on packets passed to APRSIS.

184

185

186 **2.6 Radio Interface Statistics Telemetry**

187 Current *aprx* software offers telemetry data on radio interfaces. It consists of following four
188 things. Telemetry is reported to APRS-IS every 10 minutes:

- 189 1. Channel occupancy average in Erlangs over 1 minute interval, and presented as
190 busiest 1 minute within the report interval. Where real measure of carrier presence
191 on radio channel is not available, the value is derived from number of received
192 AX.25 frame bytes plus a fixed Stetson-Harrison constant added per each packet
193 for overheads. That is then divided by presumed channel modulation speed, and
194 thus derived a figure somewhere in between 0.0 and 1.0.
- 195 2. Channel occupancy average in Erlangs over 10 minute interval. Same data source
196 as above.
- 197 3. Count of received packets over 10 minutes.
- 198 4. Count of packets dropped for some reason during that 10 minute period.

199 Additional telemetry data points could be:

- 200 1. Number of transmitted packets over 10 minute interval
- 201 2. Number of packets IGate:d from APRSIS over 10 minute interval
- 202 3. Number of packets digipeated for this radio interface over 10 minute interval
- 203 4. Erlang calculations could include both Rx and Tx, but could also be separate.

204

205

206

207 **2.7 Individual Call-Signs for Each Receiver, or Not?**

208 Opinions are mixed on the question of having separate call-signs for each receiver (and
209 transmitter), or not. Even the idea to use all 16 available SSIDs for a call-sign for
210 something does get some opposition.

- 211 • There is no license fee in most countries for receivers, and there is no need to limit
212 used call signs only on those used for the site transmitters.
- 213 • There is apparently some format rule on APRSIS about what a “call-sign” can be,
214 but it is rather lax: 6 alphanumerics + optional tail of: “-” (minus sign) and one or two
215 alphanumerics. For example OH2XYZ-R1 style call-sign can have 36 different
216 values before running out of variations on last character alone (A to Z, 0 to 9.)
- 217 • Transmitter call-signs are important, and there valid AX.25 format call-signs are
218 mandatory.

219

220 Transmitters should have positional beacons for them sent on correct position, and
221 auxiliary elements like receivers could have their positions either real (when elsewhere), or
222 actually placed near the primary Tx location so that they are separate on close enough
223 zoomed map plot.

224 Using individual receiver identities (and associated net-beaconed positions near the real
225 location) can give an idea of where the packet was heard, and possibly on which band. At
226 least the *aprs.fi* is able to show the path along which the position was heard.

227

228

229 **2.8 Beacons**

230 Smallest time interval available to position viewing at aprs.fi site is 15 minutes. A beacon
231 interval longer than that will at times disappear from that view. Default view interval is 60
232 minutes.

233 Beacon transmission time **must not** be manually configured to fixed exact minute. There
234 are large peaks in APRSIS traffic because of people are beaconing out every 5 minutes,
235 and every 10 minutes, at exact 5/10 minutes. Beaconing algorithm must be able to
236 spread the requests over the entire cycle time (10 to 30 minutes) evenly. Even altering
237 the total cycle time by up to 10% at random at the start of each cycle should be considered
238 (and associated re-scheduling of all beacon events at every cycle start). All this to avoid
239 multiple non-coordinated systems running at same rhythm.

240 The Bresenham's line plotting algorithm can be used to find smooth integer time intervals –
241 or the programmer can resort of floating point. Related algorithm is known as “Digital
242 Differential Analyser”. Both can be implemented using integer arithmetic, which may be of
243 interest on some cases.

244 Beaconing at quicker repetition rates is also possible by inserting same item multiple times
245 into the cycle.

246 Receiver location beacons need only to be on APRSIS, transmitter locations could be also
247 on radio.

248 **2.8.1 Radio Beacons**

249 “Tactical situation awareness” beaconing frequency could be 5-10 minutes, WB4APR does
250 suggest 10 minutes interval. Actively moving systems will send positions more often.
251 Transmit time spread algorithm must be used.

252 Minimum interval of beacon transmissions to radio should be 30 seconds. If more
253 beacons need to be sent in this time period, use of Persistence parameter on TNCs (and
254 KISS) should help: Send the beacons one after the other (up to 3?) during same
255 transmitter activation, and without prolonged buffer times in between them.

256

257 **2.8.2 Network beaconing**

258 Network beaconing cycle time can be up to 30 minutes.

259 Network beaconing can also transmit positions and objects at much higher rate, than radio
260 beaconing. Transmit time spread algorithm must be used.

261 Beacons could also be bursting similar to radio beacon Persistence – within a reason.

262

263