Aprx Manual

A highly configurable APRS I-gate/Digipeater Daemon

Matti Aarnio, OH2MQK and Kenneth Finnegan, W6KWF

January 8, 2017

Contents

1 What is Aprx?

The Aprx program is for amateur radio APRS networking.

The Aprx program is available at: http://thelifeofkenneth.com/aprx/

Discussion forum is at: http://groups.google.com/group/aprx-software

The Aprx program can do job of at least three separate programs: 1. APRS iGate 2. APRS Digipeater 3. DPRS-to-APRS gateway

The digipeater functionality can also be used for other forms of AX.25 networking, should a need arise.

The program has ability to sit on a limited memory system, it is routinely run on OpenWRT machines with 8 MB of RAM and Linux kernel. 128 MB RAM small PC is quite enough for this program with 64 MB ram disk, a web-server, and much much more.

The program is happy to run on any POSIX compatible platform, a number of UNIXes have been verified to work, Windows needs some support code to work. On Linux platform the system supports also seamlessly the Linux kernel AX.25 devices. This program will also report telemetry statistics on every interface it has. This can be used to estimate radio channel loading, and in general to monitor system and network health.

The telemetry data is viewable via APRSIS based services, like http://aprs.fi

The message flows inside the Aprx are as follows, note also where you have filtering opportunities.

TODO: figure

Aprx was originally written by Matti Aarnio, OH2MQK ¡oh2mqk (at) sral.fi¿

Aprx is currently maintained by Kenneth Finnegan, W6KWF ¡kennethfinnegan2007 (at) gmail.com¿

Email list for support: aprx-software (at) googlegroups.com

2 Configuration Examples

Basis of Aprx configuration is in understanding how interfaces and message flows depend on each other:

In following chapters there are examples for:

1. Minimal Configuration of Rx-iGate
2. Minimal Configuration APRS Digipeater
3. Controlling New-n-paradigm
4. Filtering at APRS Digipeater
5. Combined APRS Digipeater and Rx-iGate
6. Doing Transmit-iGate
7. Digipeater and Transmit-iGate

Safest possible setup not requiring transmit licenses is first: the Rx-iGate. Full feature bi-directional APRS-IS ↔ RF I-gate is the last one.

At chapter 4, Advanced Configuration Examples, you can find more advanced things with multiple radios at same or at different frequencies, controlling what kind of packets and from what source locations you want to relay to your current transmitter (or not relay.)

2.1 Minimal Configuration of Rx-only I-gate

To make a receive-only iGate, you need simply to configure:

1. mycall parameter
2. APRSIS network connection
3. Interface for the radio

```
mycall  NOCALL-1

<aprsis>
  passcode  -1
  server    rotate.aprs2.net
```

```
</aprsis>

<interface>
  serial-device   /dev/ttyUSB0  19200 8n1    KISS
</interface>
```

You need to fix the "N0CALL-1" callsign with whatever you want it to report receiving packets with (it must be unique in global APRSIS network!) and the APRS-IS passcode field. Contact the Aprx maintainer with the callsign used to receive an APRS-IS passcode if one has not already been issued to you for the used call. You will also need to fix the interface device with your serial port, network TCP stream server, or Linux AX.25 device. Details further below. In usual case of single radio TNC interface, this is all that a receive-only APRS iGate will need. Detail configuration part tells about the available options to initialize the TNC in case it isn't always talking using KISS.

You need to look at http://www.aprs2.net/ for possible other suitable servers to use. The "rotate.aprs2.net" uses global pool of servers, however some regional pool might be better suited for example: noam.aprs2.net, euro.aprs2.net, etc.

In most common case of wanting to make an I-gate, you do not want to use APRSIS service port parameter at all! Using "all traffic" port of 10151, or anything else than "user port" 14580 is bound to cause all sorts of troubles. This includes excessive APRSIS to Aprx data traffic, severe bloat of packet history tracking memory database, etc.

2.2 Minimal Configuration APRS Digipeater

Note: The "Interface Rx filters" feature has not been implemented, and does exist only in these pictures. To make a single interface digipeater, you will need:

1. mycall parameter
2. `<interface>` definition
3. `<digipeater>` definition

This is not an I-gate as there is no APRSIS section. Additional bits over the Rx-iGate configuration are highlighted below:

```
mycall  N0CALL-1

<interface>
  serial-device /dev/ttyUSB0  19200 8n1    KISS
  tx-ok         true
</interface>

<digipeater>
  transmitter $mycall
  <source>
```

```
    source  $mycall
  </source>
</digipeater>
```

The interface must be configured for transmit mode (default mode is receive-only) Defining a digipeater is fairly simple as shown. Do not put “alias WIDE1-1” or any other of that type on the `<interface>` definitions! Aprx understands the “new-n-paradigm” digipeater routing at a fundamental level, and does not need kludges that old AX.25 digipeater systems needed. The digipeater handles AX.25 UI frames with APRS packet types using APRS rules, including duplicate detection, “new-n-paradigm” processing, etc. At the same time it also handles all kinds of AX.25 frames as basic AX.25 digipeater matching next-hop with interface callsigns and aliases (see chapter 3.6 “The “¡interface¿” sections”)

2.3 Controlling New-n-paradigm

By default the New-n-paradigm processing uses stems of “TRACE”, “WIDE” and “RELAY” on which it adds an original request indicating n value suffix character, plus actual hop-by-hop processing counter on SSID field: WIDE2-2 -¿ WIDE2-1 -¿ WIDE2*

The Aprx has following default settings on TRACE and WIDE rules. TRACE rules are always taking precedence.

```
<digipeater>
  transmitter \$mycall

  <trace>
    maxreq  3 # in range: 1 .. 7, default: 4
    maxdone 3 # in range: 1 .. 7, default: 4
    keys    TRACE,WIDE,RELAY
  </trace>
  <wide>
    maxreq  3 # in range: 1 .. 7, default: 4
    maxdone 3 # in range: 1 .. 7, default: 4
    keys    TRACE,WIDE,RELAY
  </wide>

  <source>
    source  \$mycall
  </source>
</digipeater>
```

The request will be discarded from digipeating, if sum of requested steps or sum of accounted steps exceeds the configured limit (default on both: 4). That is, if a request packet is received with excessively large WIDEn-N request, like WIDE7-6, such packet will never be digipeated at all. There is one exception: If a packet is observed to have no

executed steps done when it is being received by Aprx, then Aprx marks all VIA fields executed (H-bit set), and sends it back to radio. This is by Bob Bruninga's request, that pathological packets should be treated so on first appearance on radio network to give sending user a change to observe that his packet needs fixing. WIDE7-7 -¿ WIDE7-7*

Note: The Rx-iGate processing will happen in all cases regardless of digipeater filtering.

2.4 Filtering at APRS Digipeater

Note: The "Interface Rx filters" feature has not been implemented, and does exist only in these pictures. To make a single interface digipeater, you will need: 1. mycall parameter 2. myloc parameter (for "filter m/100" only) 3. ¡interface¿ definition 4. ¡digipeater¿ definition

Additional bits over the minimal digipeater are highlighted below:

```
mycall  N0CALL-1
myloc  lat  ddmm.mmN  lon  dddmm.mmE

<interface>
  serial-device /dev/ttyUSB0  19200 8n1    KISS
  tx-ok         true
</interface>

<digipeater>
  transmitter \$mycall
  <source>
    source   \$mycall
    filter   p/prfx # additive filters
    filter    -p/prfx # substractive filters
  </source>
</digipeater>
```

Without any filter parameters, the system will not inspect packet content to determine if it should be digipeated or not. It will digipeat everything. When filter parameters are defined, there are two kinds: Additive: Packet matching on this may be eligible for digipeat Subtractive: Packet matching on this will be rejected even if it matched on Additive. In particular an outside range or outside area filters may be of interest to you. There can be unlimited number of both kinds of filters. Note: These filters are applied only on APRS type packets.

2.5 Combined APRS Digipeater and Rx-iGate

Note: The "Interface Rx filters" feature has not been implemented, and does exist only in these pictures. Constructing a combined APRS Digipeater and Rx-iGate means

combining previously shown configurations:

```
mycall  N0CALL-1

<aprsis>
  passcode  -1
  server    rotate.aprs2.net
</aprsis>

<interface>
  serial-device /dev/ttyUSB0  19200 8n1    KISS
  tx-ok         true
</interface>

<digipeater>
  transmitter \$mycall

  <source>
    source  \$mycall
  </source>
</digipeater>
```

It really is as simple as that. When an ¡aprsis¿ section is defined, all declared ¡interface¿s are Rx-iGate:d to APRSIS in addition to what else the system is doing.

2.6 Doing Transmit-iGate

The APRSIS source on a digipeater will enable APRSIS -¿ RF iGate functionality. Mandatory parts are “source APRSIS”, and “relay-type third-party”. No filters are needed in usual case, and this example shows none. Everything else is optional.

```
mycall  N0CALL-1

<aprsis>
  passcode  -1
  server    rotate.aprs.net    14580
</aprsis>

<interface>
  serial-device  /dev/ttyUSB0  19200 8n1    KISS
  tx-ok          true
</interface>

<digipeater>
  transmitter \$mycall
```

```
  <source>
    source     APRSIS
    relay-type  third-party
    via-path    WIDE1-1 # default: none
    msg-path    WIDE1-1 # default: none
  </source>
</digipeater>
```

2.7 Digipeater and Transmit-iGate

Note: The "Interface Rx filters" feature has not been implemented, and does exist only in these pictures. This is fairly simple extension, but shows important aspect of Aprx's ¡digipeater¿ definitions, namely that there can be multiple sources!

```
mycall  N0CALL-1

<aprsis>
  passcode  -1
  server    rotate.aprs.net    14580
</aprsis>

<interface>
  serial-device  /dev/ttyUSB0  19200 8n1    KISS
  tx-ok          true
</interface>

<digipeater>
  transmitter \$mycall
  <source>
    source  \$mycall
  </source>
  <source>
    source      APRSIS
    relay-type  third-party
  # viscous-delay 5
  </source>
</digipeater>
```

Using both the radio port, and APRS-IS as sources makes this combined Tx-iGate, and digipeater.