Dire Wolf Internal Packet Routing

- Rough Draft 2 - Jan. 2025 -

Dire Wolf can support multiple audio channels which correspond to a physical audio interface, a virtual audio pipe, stdin (receive only), or a UDP port (receive only). Each of these channels is configured to have its own modem (AFSK, GMSK, PSK) and framing type (AX.25, FX.25, IL2P, AIS, EAS SAME).

Dire Wolf can support many concurrent client applications using serial KISS, KISS over TCP, and AGW network protocol over TCP. The serial port connection can also be used for Bluetooth.

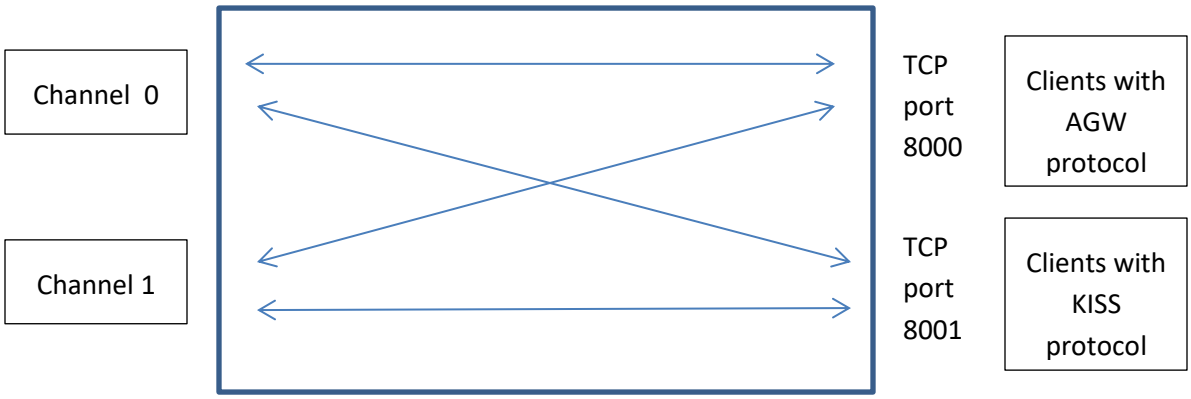
Dire Wolf started out very simple, but over time, many new options were added for creative use cases. All of the different options, are in various places in the User Guide, and can be confusing. This is an attempt to reduce the confusion.

Use as a TNC by client applications - Ethernet or WiFi

By default, KISS over TCP is available on port 8001 and AGW network protocol is available on port 8000. Most modern applications will support a TCP/IP connection to a network TNC.

All packets received on any radio channel are sent to all attached client applications. A field contains the channel number where it came from.

All packets received, from any client application, are sent to the specified channel.



If you don't want to use the default ports, they can be changed with this in the configuration file:

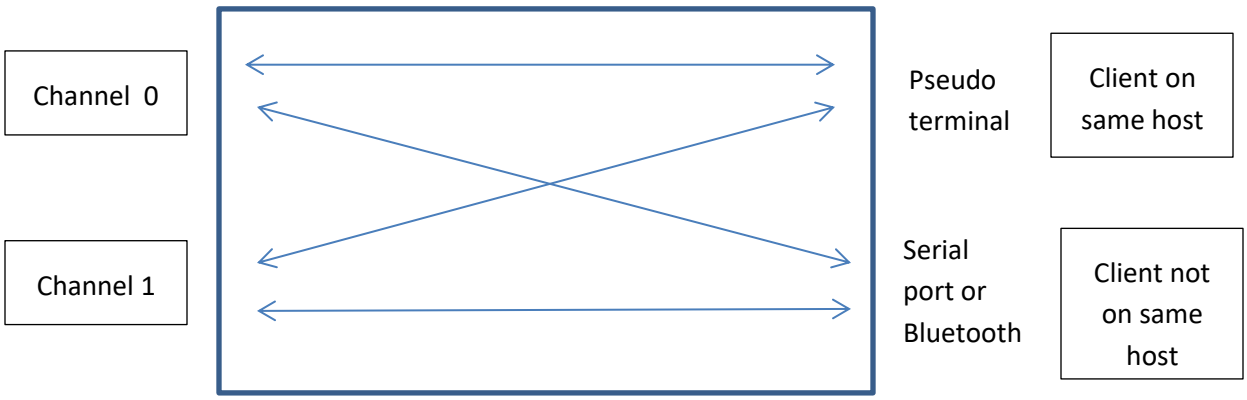
```
KISSPORT 12345
AGWPORT  54321
```

Use as a KISS TNC by client applications - Serial ports

Serial ports are available for older applications without network capability. For an application running on the same host, "-p" can be added to the command line to enable a pseudo terminal (Linux only).

Do not use the "-p" option unless you have an application attached to the pseudo terminal.

We can't pick the device name; it is assigned by the operating system. A symbolic link "/tmp/kisstnc" is created so applications can use a predictable name.



A serial port can also be used for applications not running on the same host. Examples:

MORE

```
SERIALKISSPOLL   /dev/rfcomm0
```

This can be a physical serial port or the serial port profile of a Bluetooth interface. A complete example can be found in "***Bluetooth KISS TNC***."

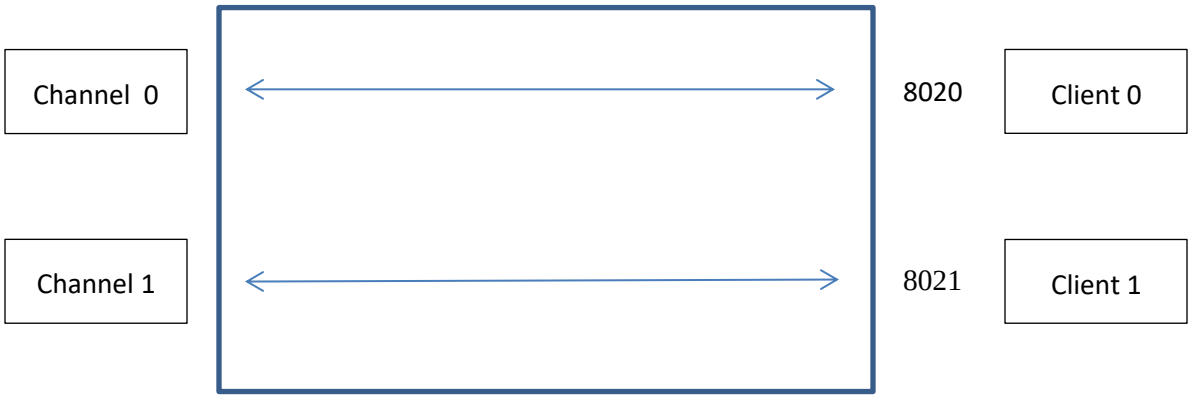
Only the KISS protocol is available over serial ports.

Single channel applications

Many Packet Radio / APRS applications were written assuming that a TNC could only be attached to a single radio. They always set the transmit channel to 0. They might ignore the channel number on receive or ignore any packets where it is not 0.

Additional TCP KISS ports can be created corresponding to a single radio channel, rather than conveying all of them.

```
KISSPORT  8020 0
KISSPORT  8021  1
```



- Receive channel 0 → goes to only port 8020 and channel is set to 0.
- Receive channel 1 → goes to only port 8021 and channel is set to 0.
- Client 0 transmit → port 8020 → channel is forced to 0.
- Client 1 transmit → port 8021 → channel is forced to 1.

Client 1 thinks it is using channel 0 but it is really mapped into channel 1 in both directions.

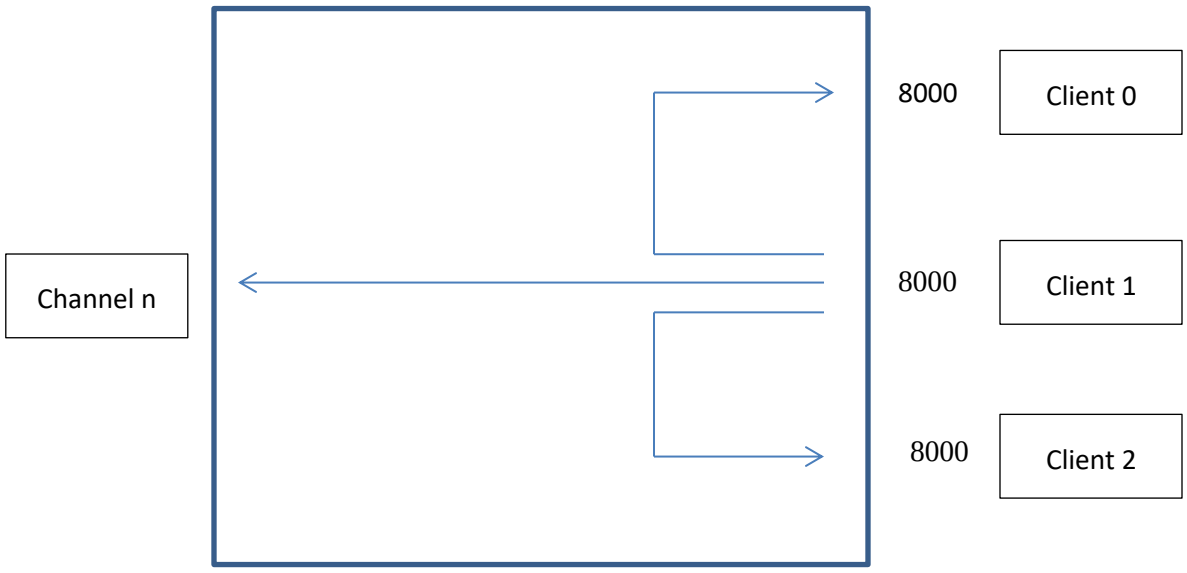
This feature is only available for KISS over TCP. It is not available for AGWPORT or serial port KISS.

Communication among TCP KISS client applications

You might have a situation where you want data, generated by one application, to be available to other applications. This feature is enabled by added the following to the configuration file:

```
KISSCOPY
```

A packet from any TCP KISS client will be sent to all other TCP KISS clients in addition to the specified radio channel. This does not apply to serial port KISS.



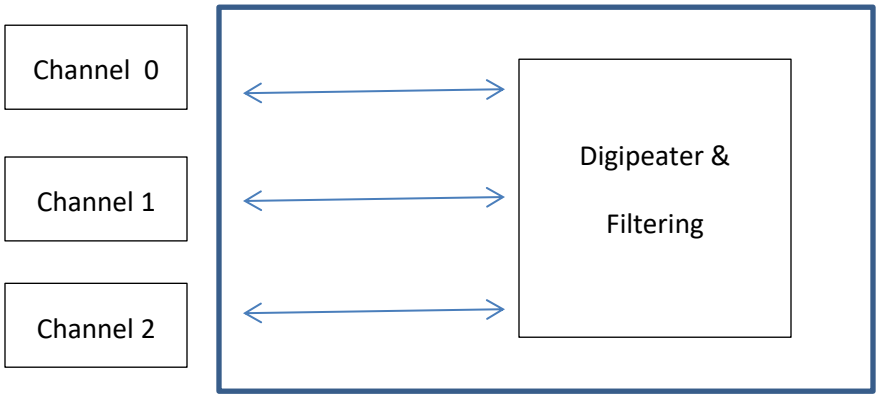
In the example above, a packet sent by Client 1 is sent to both a radio and all other attached TCP KISS clients. This might be useful if you have one application generating object reports and you want to see them, combined with received data, in a different application.

APRS Digipeater

The purpose of a digipeater is to extend the range of the original station. Dire Wolf implements both the APRS and connected mode digipeater algorithms which are not the same. In theory, you could probably run both at the same time on the same channel. The cabability is enabled with the following commands:

DIGIPEAT *from-chan to-chan aliases and other options*
CDIGIPEAT *from-chan to-chan*

The important part, for this discussion, is that you have a separate configuration line for each combination of "from" and "to" channel. This allows eligible packets, from any channel, to be resent on any combination of same or different channels.



You can limit which packets are allowed by adding a filter. For example, you could block any packets from a particular station. Again, you specify the "from" and "to" channels for each of the channels. FILTER is for APRS and CFILTER is for connected mode.

```
FILTER      from-chan  to-chan   filter pattern
CFILTER     from-chan  to-chan   filter pattern
```

More details can be found in the ***Dire Wolf "User Guide***" and "***APRS Digipeaters***."

Receive-Only (RF>IS) IGate

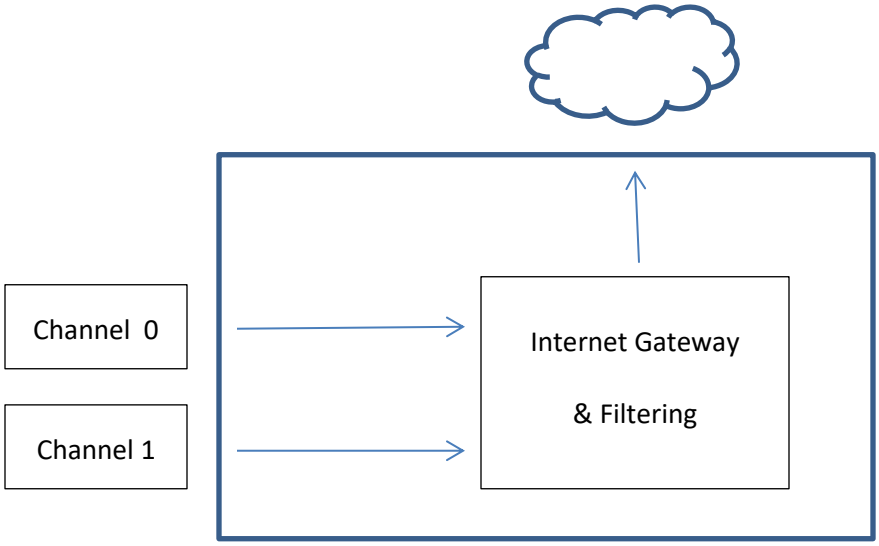
The purpose of an Internet Gateway ("IGate") is to link radio networks that can't hear each other.

The receive direction is enabled with the configuration:

X

X

Everything received on all channels is repackaged and sent to the global network of APRS-IS servers.



Filtering is also possible; similar to the digipeater filtering except the radio channel number is replaced by "IG" short for Internet Gateway. Normally you want everything to go from RF to IS. The most useful form would be:

```
FILTER  1  IG  0
```

This means that packets received on channel 1, going to the IGate, are filtered by the expression 0 which means false. Only packets from other channels would be sent to the APRS-IS servers.

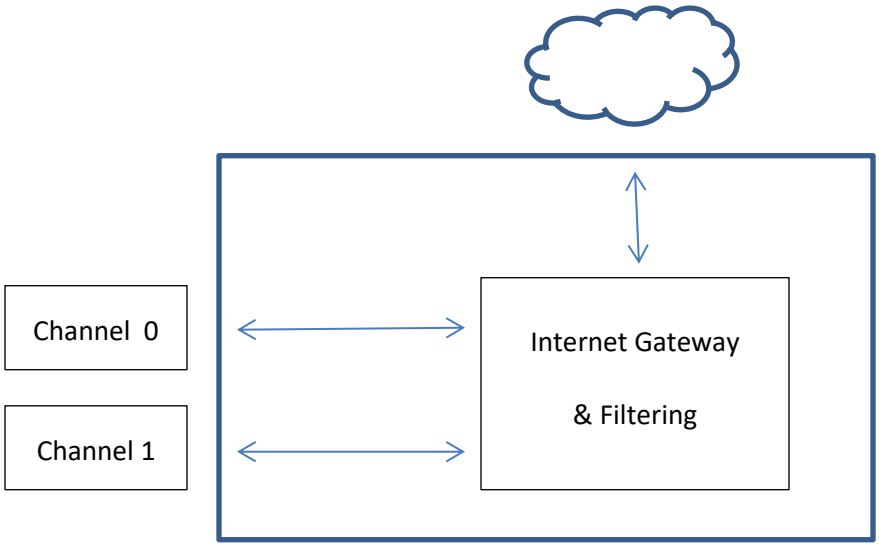
It is strongly recommended that IGates are bidirectional if legal in your country. Receive-only IGates break APRS messaging.

Bidirectional (RF>IS & IS>RF) IGate

Bidirectional capability is added by enabling transmit.

X

X



Eligible packets, from APRS-IS, are wrapped in a third-party header and transmitted over the radio.

If you transmit everything that the APRS-IS sends to you by default, you will be very unpopular for clogging up the airwaves. The default filter, for the IS>RF direction, allows only APRS "messages" for users recently heard in your area. This is appropriate for nearly everyone so don't mess with it unless you have a good reason and a good understanding of how this all works.

This topic is covered in great detail in "***Successful IGate Operation***."

Connect application to APRS-IS

(New in version 1.7)

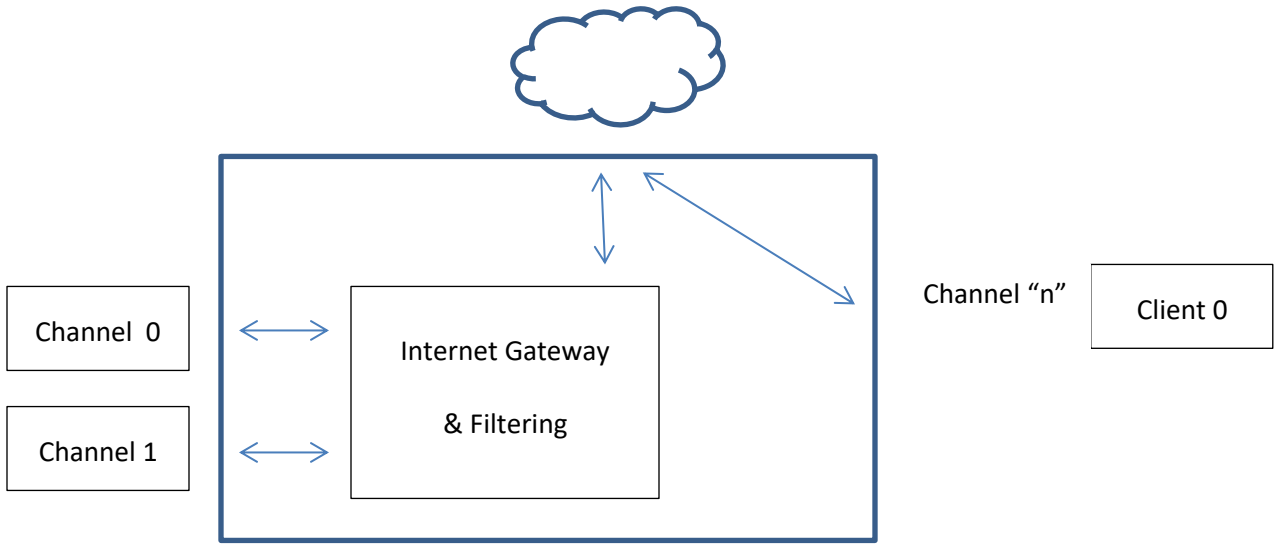
Sometimes you might want an application to talk directly to the APRS-IS without any RF involvement.

After configuring a bidirectional IGate, add this to your configuration, where *n* is some channel number not used by a radio channel.

```
ICHANNEL    n
```

Anything that comes from APRS-IS is sent to attached client(s) with the channel number specified above. A properly written application can distinguish packets from APRS-IS and those from RF. There is no filtering along this path so the application needs to ignore packets it does not care about.

In the other direction, when an application sends to that channel, the packet will go to APRS-IS rather than a radio channel.



Here is a working example, to get you started with this feature.

```
# An audio device needs to be present but we are not using a radio.
#ADEVICE  plughw:1,0
ADEVICE  0

MYCALL yourcall


# Establish a bidirectional IGate
# Use closest servers your geographical location.
```

```
IGSERVER noam.aprs2.net
IGLOGIN yourcall 123456

# Request something besides the default just to see if it is working.
# Note: this is the server side filter.

IGFILTER b/AB1OC-10

# We need to configure a bidirectional IGate.

IGTXVIA 0

# We might want to prevent the IGate from interacting with RF.
# Block anything from radio channel 0 to the IGate.
#FILTER 0 IG 0
# Block anything from IGate to radio channel 0.
#FILTER IG 0 0

# Map channel 15 to the IGate.
# Packets from APRS-IS are routed to client apps with channel 15.
# When a client app sends to channel 15, it goes to APRS-IS rather than
a radio channel.

ICHANNEL 15

# Send a beacon directly to APRS-IS with no RF involvement.
# This has been around for a long time.

OBEACON sendto=IG objname="Santa C" delay=0:20 every=10
symbol=snowmobile lat=42^37.14N long=071^20.78W comment="Ho Ho Ho"

# New in 1.7:  Send beacon to channel 15 after ICHANNEL is defined.
# A client app can do the same thing by sending to that KISS channel.

OBEACON sendto=15 objname="El perro" delay=0:25 every=10 symbol=dog
lat=42^37.17N long=071^20.78W comment="guau guau guau"
```

Notice that the beacon can go to either IG or the channel defined by ICHANNEL. The result should be the same.

Here I run **kissutil** to simulate what you might do with your application.

Type:

```
[15] WB2OSZ-14>TEST::WHO-IS   :W1AW{002
```

Note that there are 3 spaces after WHO-IS. The APRS “message” format requires exactly 9 characters for the addressee.

The WHO-IS server responds with two packets.

```
[15] X>X:}WHO-IS>APJIW4,TCPIP*,qAC,AE5PL-JF::WB2OSZ-14:ack002
[15] X>X:}WHO-IS>APJIW4,TCPIP*,qAC,AE5PL-JF::WB2OSZ-14:C/ARRL HQ
OPERATORS CLUB/CT/United States{471
```

The first is an acknowledgement that my message was received.

The second contains information about the callsign sent in my message.

The prefix of “X>X:}” probably comes as a surprise. The response needs to be wrapped in a third party header because the sender address "WHO-IS" does not follow the rules for an AX.25 RF address.

Occasionally we see something from the station mentioned in the server side filter.

```
[15] X>X:}AB1OC-10>APWW11,TCPIP*,qAC,T2SPAIN:>FN42er/-DX: WA1PLE-13
43.7mi 154▒ 02:35 4208.36N 07113.50W<0x20>
```

The KISS frame uses AX.25 formatting for addresses. This means the addresses are constrained to maximum 6 upper case letters or digits then an SSID in the range of 0 to 15. APRS-IS has looser restrictions. Lower case can be used, station name can exceed 6 characters, and SSID can be up to 2 alphanumeric characters.

When something from APRS-IS is conveyed over KISS, we add a fake third party header so the original addresses can be preserved.

There is no radio involved. The client app communicates directly with APRS-IS.

Going in the opposite direction, you would need a fake third party header only if any of the addresses violate the AX.25 restrictions. It might be a good idea to always use it so you don’t need to examine the addresses and have special cases.

Direwolf will simply discard any third party header before forwarding the rest to APRS-IS.

I hope this is helpful to you. Let us know about any interesting ways you can use this new feature.

---------- END OF DOCUMENT ----------

The following are notes for sections not written yet.

GPS Tracker

DTMF

EAS SAME

> EAS>APDW16:{DEZCZC-WXR-RWT-033019-033017-033015-033013-033011-025011-025017-033007-033005-033003-033001-025009-025027-033009+0015-1691525-KGYX/NWS-

"***EAS SAME to APRS MESSAGE Converter***"

AIS translation

APRStt Gateway

kissutil