Minimizing APRS Collisions

John Langner, WB2OSZ

Problem

APRS is designed for many stations to share a single frequency and transmit at unpredictable times. It would be counterproductive if multiple stations transmitted at the same time and interfered with each other. How can we minimize collisions and give everyone a fair chance?

Human Group Conversion

Suppose you were in a group of around 5 or 10 peers having a discussion. What would you do if you had something to say?

(1) Blurt it out immediately, even while someone else is talking?

(2) Start talking, and don't stop, when there is silence for a few milliseconds?

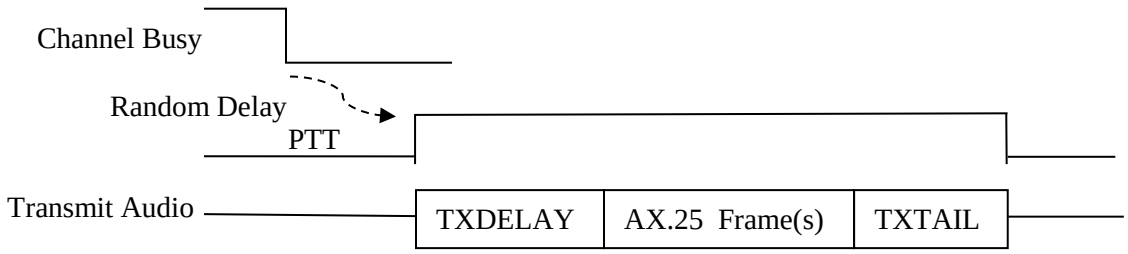
(3) After one person is finished, wait a fraction of a second to see if someone else jumps in.

Choice (1) is very rude. Choice (2) is not much better; we all know someone who totally dominates a conversation and doesn't give anyone else a chance to participate. Choice (3) gives everyone a fair chance.

There are similar rules for how a well behaved APRS station should wait before transmitting.

Channel Access Algorithm

APRS is designed for many stations to share a single frequency and transmit at unpredictable times. There is no central station telling everyone when it is their time to speak. Obviously you should wait for the channel to be clear before transmitting. If everyone transmitted as soon as the channel becomes clear, there would be many collisions. These collisions can be minimized by waiting a minimum amount of time and then waiting a random amount of time then transmitting if the channel is still clear. The technical term for this is p-persistent Carrier Sense Multiple Access (CSMA).



Transmit timing is determined by 4 parameters with these traditional names and typical default values.

SLOTTIME 10	x 10 mSec per unit = 100 mSec.
PERSIST 63	probability for transmitting after each slottime.
TXDELAY 30	x 10 mSec per unit = 300 mSec.
TXTAIL 10	x 10 mSec per unit = 100 mSec.

Some implementations might use milliseconds, rather than units of 10 milliseconds, so read the documentation if not using the defaults provided by the application.

- For digipeated frames the transmission should begin immediately after the channel is clear. Having multiple digipeaters transmit at the same time is intentional to minimize the amount of time taken for multiple digipeaters retransmitting the same packet. The strongest signal will win due to the FM capture effect.

- For other frames, SLOTTIME and PERSIST parameters are used to generate a random delay to minimize the chances of two different stations starting to transmit at the same time. The process is:

  (a) Wait for channel to be clear (i.e. no signal detected by modem)

  (b) Wait for SLOTTIME.

  (c) If the channel is busy (i.e. signal detected by the modem), go back to step (a).

  (d) Generate a random number in the range of 0 to 255. If it is less than or equal to PERSIST (Typically 25% probability) , start to transmit. Otherwise go back to step (b).

  For the typical default values, we have delays with the following probabilities:

Delay, mSec	**Probability**	
0	0	= 0%
100	.25	= 25%
200	.75 * .25	= 19%
300	.75 * .75 * .25	= 14%
400	.75 * .75 * .75 * .25	= 11%
500	.75 * .75 * .75 * .75 * .25	= 8%
etc.	...	

It is important to notice that there is a 0% chance of transmitting with less than a 100 millisecond delay. This time slot is reserved for digipeaters.

The Push to Talk (PTT) control line is asserted.

The frame data can't be sent immediately because the transmitter takes a little while to stabilize and reach full power after being activated. The receiving system needs time to lock on to the incoming bit timing.

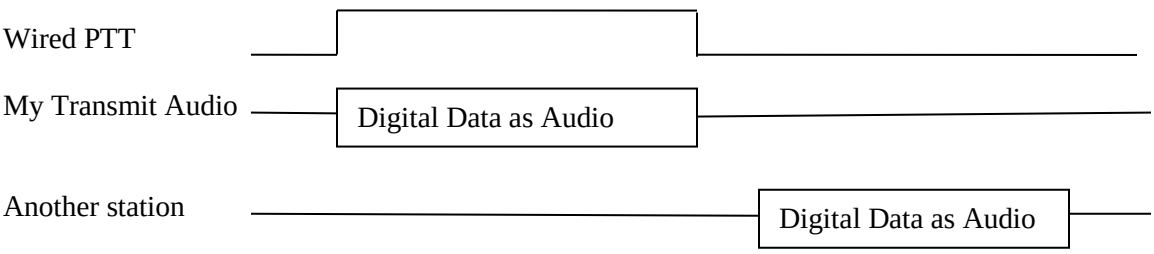
The HDLC "flag" pattern (01111110) is sent for a time period of TXDELAY. For historical reasons, going back to the 1980s, the configuration numbers are often expressed in 10 millisecond (mS) units so 30 actually means 300 milliseconds.

When sending with a hardware modem, you know exactly when the audio for the frame is finished so the transmitter can be turned off fairly soon after that. TXTAIL can be very small.

When sending audio out through a "soundcard" there is latency between sending an audio waveform to the audio output device and when the sound comes out. We can't be sure precisely when the queued up sound has been completed so we need to keep the PTT on a little longer. The HDLC "frame" pattern is also sent during this time to keep the channel busy.

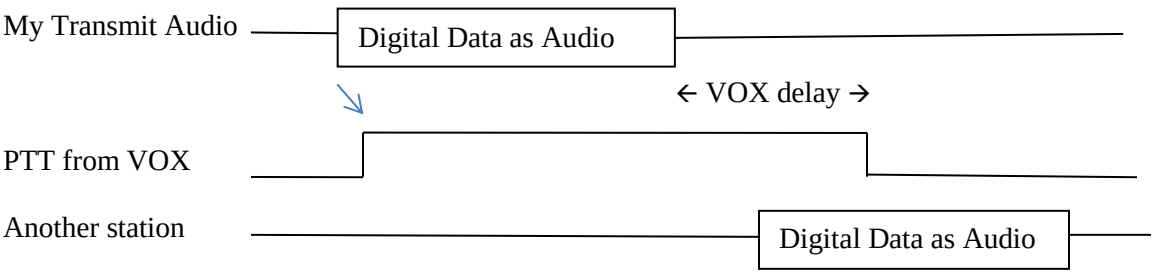
Using VOX Instead of Wired PTT

First let's review the timing for case where we have a wired connection to activate the transmitter. The transmitter is activated, we send our digital data as an audio signal, and then turn off the transmitter when the audio is finished. Another station detects no other signal, waits a random amount of time (usually less than 1/3 of a second), and starts transmitting.



VOX stands for **Voice** Operated Transmit. It is designed for voice which contains small gaps between words and sentences. It responds quickly when speech begins so it doesn't chop off too much from beginning of the first word. We don't want the transmitter going on and off for every little gap in speech so there is a built in delay before turning the transmitter off again. This is usually referred to as the "VOX delay." On one popular HT, the default is 500 milliseconds and the minimum setting is 250. Another popular HT doesn't allow the delay time to be configured and the documentation doesn't mention how long it is.

Keeping the transmitter on a half second after the sound ends is fine for voice. But what about digital data? We saw in the previous section that another station, waiting for a clear channel, will usually transmit less than 1/3 second after it no longer hears another digital signal.



Using VOX in this case might be easier but, it is:

- Inconsiderate of the community:

  You are interfering with others by sending a quiet carrier, possibly overpowering other stations trying to be heard.

- **BAD** for APRS:

  In this case, you will probably miss the beginning of the next station transmitting because you haven't switched back to receive yet.

- ***DISASTER*** for connected mode packet:

  Connected mode uses a rapid back-and-forth exchange to acknowledge that information was received and retry if something gets lost. The next frame will often be a response to something you sent. If that response frame is lost, your station will keep trying over again and eventually give up.

My recommendation is to avoid using VOX, built into a transceiver, unless you can be sure the transmitter will turn off very soon (e.g. less than 50 mSec) after the transmit audio signal is no longer present. If you stubbornly insist on using VOX, be sure the "VOX delay" setting is at the shortest setting to do the least damage.

The SignaLink USB has a built in VOX circuit but it is adjustable into an appropriate range. Turn the delay down to the minimum (fully counterclockwise). According to the documentation, this should turn off the transmitter around 15 or 30 milliseconds after the transmit audio has ended.

Who is Playing by the Rules?

The 20th century legacy TNCs (e.g. Kantronics KPC-3+) implement SLOTTIME and PERSIST.

Kenwood APRS-equipped radios have SLOTTIME and PERSIST parameters.

The Dire Wolf software TNC implements SLOTTIME and PERSIST.

These parameters are so important they were part of the extremely minimalistic KISS protocol.

I haven't noticed any mention of SLOTTIME and PERSIST for the newer APRS implementations. It seems they have forgotten, or chose to ignore, well-established rules, from the previous century, to minimize collisions. If I missed any, that do follow the rules, please let me know.