Explorations of smartphone APRS environments

Don Rolph AB1PH

John Tarbox WA1KLI

Rev 1.1

Nov 28, 2021

Executive summary:

The enhancement of capabilities of smartphones have enabled the development for highly functional APRS apps which can run on smartphones and can use audio to couple to many transceivers. While the functionality of these APRS smartphone apps has significantly improved over the last few years, there still remain challenges with creating smartphone APRS apps which can effectively mediate the use of location services and sound services between the APRS apps and other apps. This makes these apps excellent choices for an introduction to APRS, and for light weight APRS usage. Further refinements are probably required before these smartphone apps are suitable for major APRS utilization, but we can foresee a day when smartphone apps could be effectively used in the AT Golden Packet effort.

Introduction:

It is perhaps useful to take a step back and examine APRS implementations in more detail. The classic model for APRS deployments is shown in figure 1, typified perhaps by the Kenwood D710A or similar devices.

Figure 1: classical APRS deployment model

It is suggested that in fact one of the earliest versions of this was a vic 20 running the applications interface against a TNC and a traditional transceiver fed through the audio chain. In actuality, if one explores the APRS deployment in more detail, we will see additional nuances to the design with the more developed model perhaps being shown in figure 2.

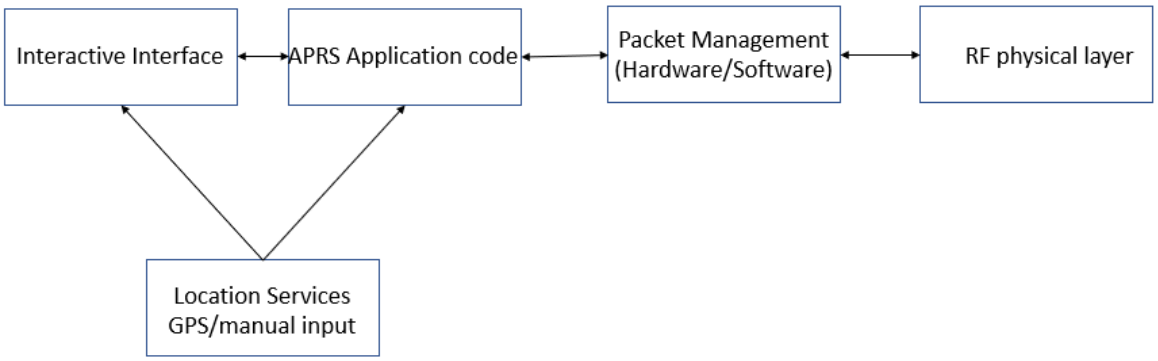

Figure 2: more nuanced model of APRS deployment

One can see this model more clearly perhaps if one looks at a D710A connected to a PC running APRSIS32. Indeed, the D710A in particular is notable in that for tracking, a GPS must be connected to the D710A AND if an application program such as APRSIS32 is used, then a second GPS typically must also be connected to the application, since the default KISS interface does not in general pass the GPS NMEA data for the station itself.

Exploring each of the blocks in the block diagram in more detail:

For APRS, the RF unit can be fairly generic. Tones are fed the audio chain (for 1200 baud communications), or directly to the FM modulator (for 9600 baud), with sound being returned from the transceiver either through the audio chain or directly from the discriminator. In principle any 2 meter FM transceiver can be used, although typically the 6 pin mini-din interface is required for 9600 baud communications.

Traditionally the AX.25 packet handling has been performed by some form of hardware Terminal Node Controller (TNC: perhaps the Kantronics KPC3+ is an example). As computing hardware increased in capability, more of the packet activity was passed off to software and increasingly the TNC only converted the packetized byte string into tones, or the tones into byte strings (what is sometimes referred to as a KISS TNC: perhaps the TNC-X is a good example). As computing capacity increased further, implementations where the computing hardware produced the audio encoding and passed it to a simple sound card began to evolve, creating what are sometimes termed software modems (Dire Wolf and UZ7HO sound modems are examples). In all designs, the packet information is encoded as sounds to be transmitted by the transceiver, and the received sound is converted back to byte strings to be consumed by the APRS application.

The APRS applications typical provide all or some subset of the following services:

- Beaconing
- Receiving and decoding APRS packets for display to the user
- Digipeating
- Sending of and receiving of messages

Again, with limited computing power, much of these application activities were performed directly by the TNC. As applications became more sophisticated, these functions could be implemented in software on the computer as opposed to coding on the TNC.

Initial APRS deployments typically had rudimentary interactive interface functionality. Sometimes this was a simple terminal connection to a TNC. Sometimes it was button and menu choices on a radio. As interactive interfaces became more sophisticated, they provided services such as:

- Moving map displays of the various APRS stations
- Listing of all heard stations
- Testing interface to the messaging functionality of APRS

The presence of the location services has often been neglected in the models, but location services have always been a critical component of an APRS deployment. The location services have evolved over time as:

- Initially manual entry of location
- Eventually integrated access to the GPS from the application or interactive modules
- As software environments began to support sophisticated location services, APRS has increasingly began to leverage this typically operating system level location interfaces

A brief note on initiation transmissions:

In any APRS deployment there needs to be a mechanism to initiate transmission, typically referred to as push to talk (PTT) functionality. This can be provided by many methods among them:

- Hardware PTT driven by serial port activity
- Hardware PTT from general IO functionality
- Explicit rig control
- VOX (voice operated transmission)

For typical packet (AX.25) activity, rapid turnaround from transmit to receive is required, and usually some form of hardware PTT is required for successful communication.

APRS however is a broadcast protocol. And as such, rapid turnaround is of lesser importance. By design AX.25 packet environment are designed to manage channel contention, so use of VOX consumes more of the channel time availability but should not break channel usage. In short, unlike connected AX.25 packet communication, VOX is arguably usable, although not preferred, for APRS.

Implications of smartphones on APRS usage:

A typical smartphone has:

- A sophisticated graphical user interactive interface
- Very high computing capability compared to the early computers used in amateur AX.25 packet work
- High quality sound systems
- Very effective location services

This suggests that the interactive interface, the APRS applications, the location services, and the packet management functionality can all be implemented on a smartphone and interfaced to the transceiver via an audio cable. Since we are focused on APRS, the transceiver VOX is functional for use as the PTT functionality, and indeed a large number of smartphone apps with software TNC functionality has begun to appear. The requirements for deployment are an audio cable (see figure 3) such as the BTech cable:

- https://baofengtech.com/product/aprs-k1/

Figure 3: BTech K1 interface cable; lightning port to 3.5 mm TRRS adaptor in middle

This cable has the Kenwood speaker-mic connection found on several radios and a 3.5 mm TRRS cable for connection to the smartphone. Modern iPhones will require the 3.5 mm TRRS to lightning port adaptor as well. A connected configuration then might look like figure 4.

Figure 4: connected smartphone APRS system: environment used for all test: Baofeng UV5R, BTech cable and iPhone

Software options for iPhone:

There are many software options for the iPhone, but I will focus in on three because the are informative of the present iPhone APRS app space.

Before using the interface cable for communications, it is useful to adjust the receive volume so that one is not overdriving the audio on the iPhone. One way to do this is to set your transceiver to 144.390 MHz, plug the cable into your transceiver and the iPhone and then using the iPhone voice memo app, record the sound of a received APRS packet. An approach is to set iPhone volume to max and adjust the transceiver volume until one sees an image such as in figure 5. You want to have the sound peaks just below maximum volume in the voice memo screen.

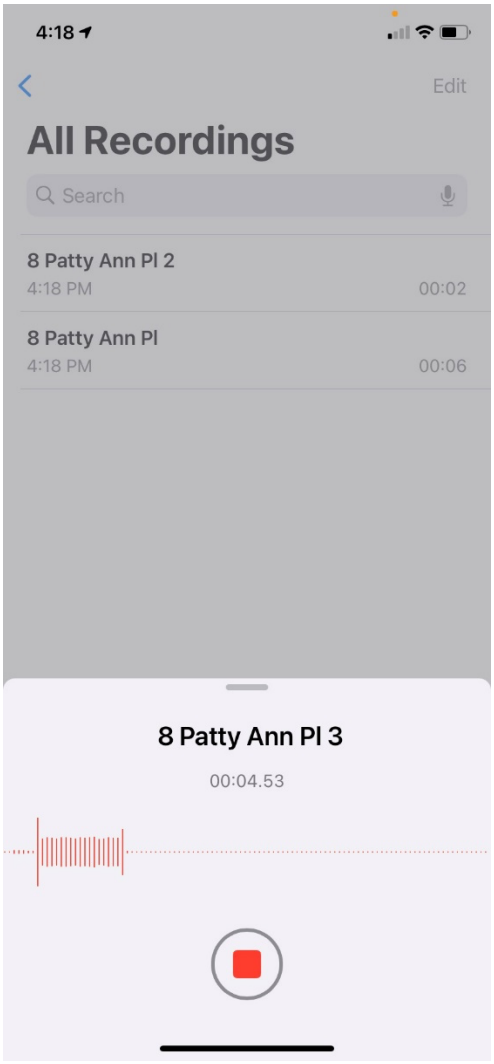

Figure 5: sound level setting

This will probably be about halfway on the volume setting for either a Baofeng UV5R or a Wouxun KG-UV6D (both have been tested). My thanks to Baofengtech tech support for this suggestion.

First, an application to avoid:

- APRS Pro: https://aprspro.com/v1/

This is a full featured very sophisticated APRS application on the iPhone, and under other conditions it might perhaps be the recommended app. However, at present:

- APRS Pro has an apparent bug in its sound reception so that received packets are not decoded
- APRS Pro does not appear to have any active software support

As such, APRS Pro should probably be avoided.

An application which can be used for APRS activity on a smart phone is PocketPacket:

- https://apps.apple.com/us/app/pocketpacket/id336500866

PocketPacket is free and supports:

- Beaconing
- Reception of packets over RF
- A map interface of APRS stations and your beaconing
- Messaging
- List of stations heard

It does not support digipeating.

For testing its functionality as an RF based APRS solution it is probably useful to only enable APRS activity over RF. The settings used in testing PocketPacket are shown in figure 6.

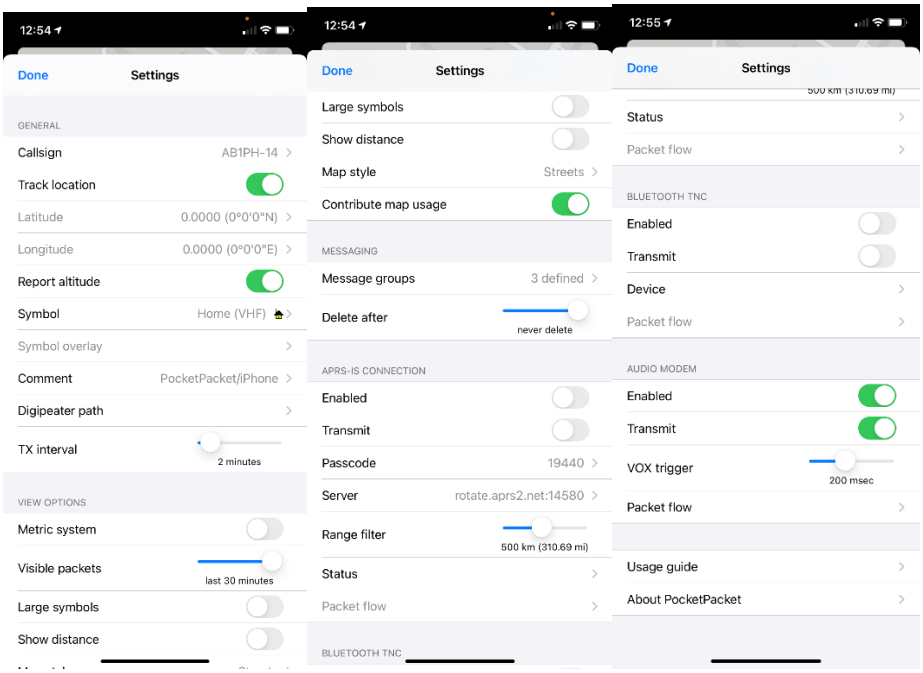

Figure 6: PacketPacket settings

Notice that APRS-IS is not enabled but the audio Modem is enabled. The beaconing over a two hours walking track is shown in figure 7.

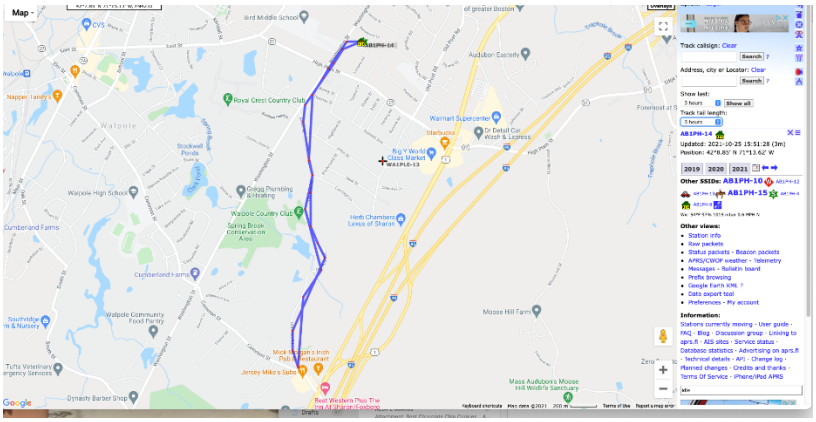

Figure 7: two hour walking track using PocketPacket

Some things to note:

- PocketPacket beacons at fixed time intervals: while the beacon locations are correct, when one assembles the beacons into a presumed track, turns which occur faster than the beaconing rate do not get captured cleanly, resulting in the rather “jagged” track
- To ensure that PocketPacket has access to the location services and the sound services, PocketPacket attempts to keep PocketPacket as the app in the foreground: per testing, switching to other apps can cause PocketPacket to stop beaconing and receiving
- Observationally when comparing decoding of APRS packets compared to a Kenwood D72, it was observed that the handheld would receive some of the packets but PocketPacket would not decode many packets which were decoded by the D72A.

The maintainers of the aprs.fi web site also make an iPhone app called aprs.fi:

- https://apps.apple.com/us/app/aprs-fi/id922155038

The aprs.fi app supports:

- Beaconing
- Reception of packets over RF
- A map interface of APRS stations and your beaconing
- Messaging
- List of stations heard

It does not support digipeating.

The settings used for testing aprs.fi are shown below in figure 8.

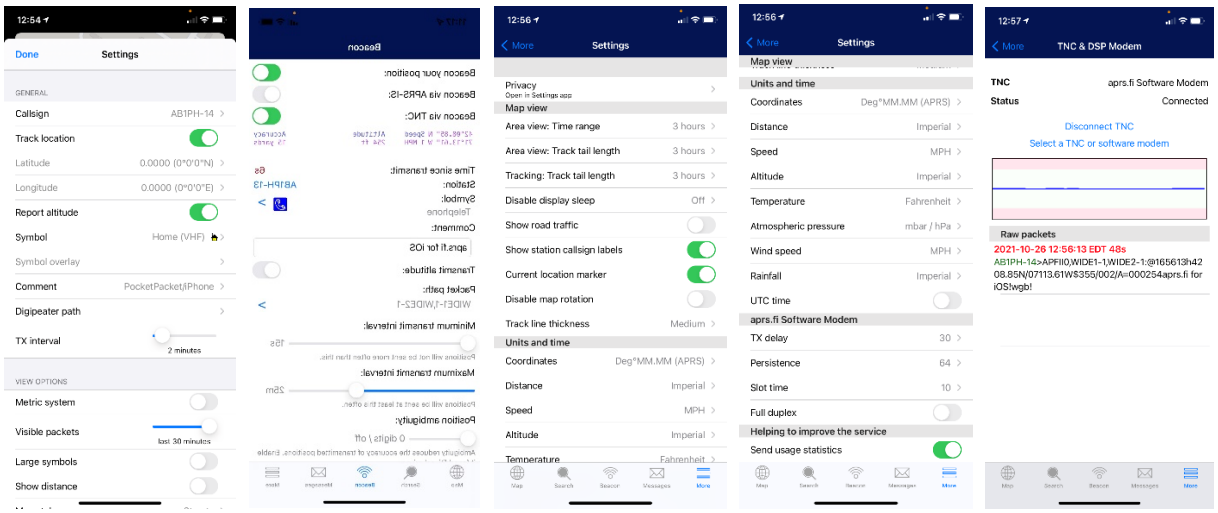

Figure 8: aprs.fi app settings: you need to set: beaconing, settings, and select software modem TNC

Make sure your call sign with SSID is set in beaconing: this setting can be quirky since aprs.fi allows creating profiles for different beaconing ID. In settings make sure aprs-is access is off, and that you have enabled transmit via RF. There is an extra fee for transmitting, which you can conveniently execute within the subscription option of settings.

A two hour track over the same course for aprs.fi is shown in figure 9.

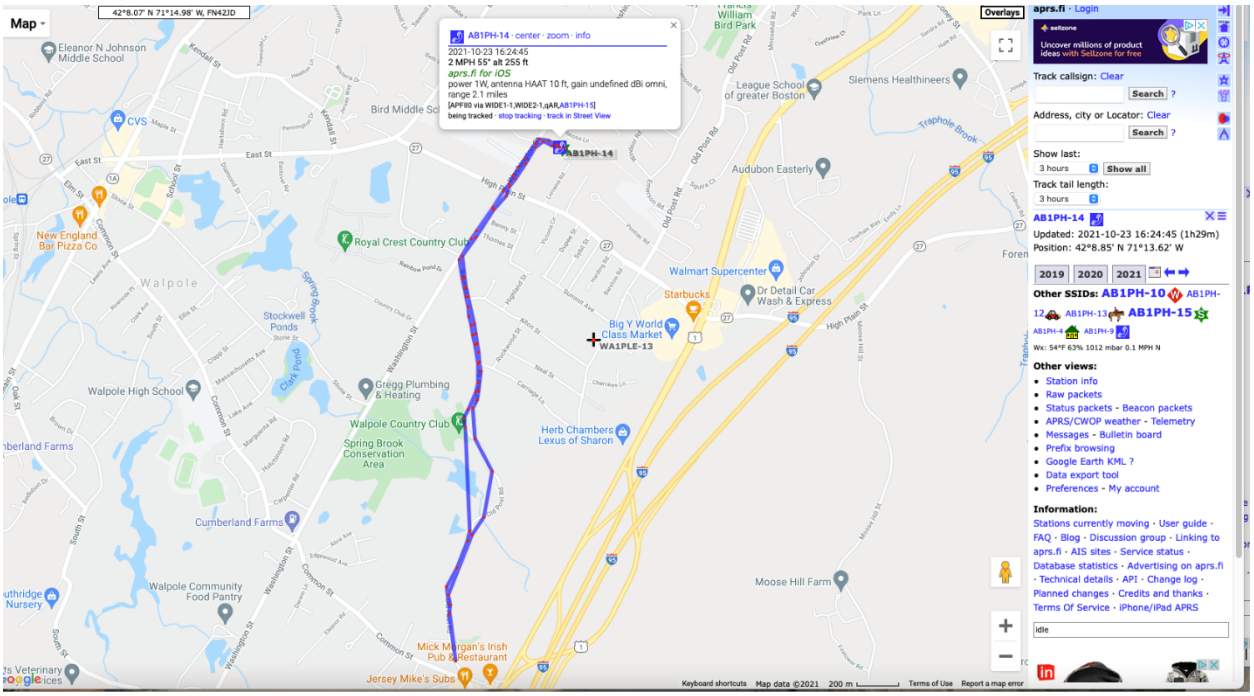

Figure 9: two hour track using the aprs.fi app

Some things to note:

- aprs.fi uses a dynamic beaconing model: while this does not result in a perfect track, it is much closer to the actual track than the track produced by PocketPacket
- aprs.fi also is cognizant of the challenge of using a smartphone for multiple apps. To assist in running in the background aprs.fi works best if the privacy protections on aprs.fi for locations services are set to always rather than the more typical while using app setting. Even with this setting use of another app accessing the sound system can disable beaconing
- the aprs.fi app did appear to effectively decode all packets heard when compared to the Kenwood D72A: it appears to have better software decoding of packets than the PocketPacket app
- aprs.fi has a nice feature in the software TNC screen that allows you to monitor and adjust your receive volume during operations.

Environment	Beaconing	Receiving	Digipeating	Messaging	Dynamic Beaconing	Packet Decoding
PocketPacket/Baofeng	X	X		X	N	fair
aprs.fi/Boafeng	X	X		X	Y	good
D74A	X	X		X	Y	good
D72A	X	X	X	X	Y	good
D710X	X	X	X	X	Y	good

Table 1: functional comparison of various APRS solutions

Comparison of Android apps:

The testing environment did not include an Android phone, so testing on androids was not feasible. It would be helpful to have others test android apps, and in particular test for behavior when the android is used for multiple apps including the APRS app simultaneously.

Observations/conclusions:

Based on this testing, it is clear that a smartphone APRS app combined with a simple handheld can provide functionality similar to that of dedicated APRS transceivers costing many times the price of inexpensive handhelds. We do not have data comparing the performance of such systems with dedicated APRS handhelds, and such an effort requires establishment of objective test protocols to measure APRS performance.

It is also clear that future APRS activity using a smartphone can potentially match the functionality of the most sophisticated APRS environments, and one can conjecture using smartphones in the AT Golden Packet in the future. For the moment, however, challenges in handling arbitration of location and sound resources with smartphone APRS apps would seem to suggest that they are presently appropriate for testing and learning. Smartphone APRS app usage for serious APRS activity probably will require additional refinement of the approach used to arbitrate sound and location resources for smartphone APRS apps.