ARDOP (**A**mateur **R**adio **D**igital **O**pen **P**rotocol)

A next generation digital protocol for HF and VHF/UHF

Rick Muething KN6KB Matthew Pitts N8OHU John Wiseman GM8BPQ

Abstract:

The popularity of low cost PCs and tablets with substantial DSP processing power and an increasing awareness of digital signal processing in the amateur community have created an explosion of digital modes. Some of the challenges this poses are lack of portability, inconsistent "virtual TNC" interfaces and protocols optimized for single uses. ARDOP is a new protocol development which was targeted to address these challenges. The development started in 2014 and Alpha testing of the ARDOP_Win TNC (Windows version) was begun in April 2015. From the beginning the protocol was designed to cover a wide spectrum of amateur uses and be fully documented with open sourced code to encourage learning, experimentation, evolution and portability to other platforms both software and hardware.
Key words: ARQ, FEC, 4FSK, 8FSK, 16FSK, 4PSK, 8PSK, WINMOR, cyclic prefix, bandwidth negotiation, automatic timing, open source and sound card protocols.

Introduction:

Today's computing platforms (PCs, laptops, tablets and smart phones) pack more numeric processing capability than expensive dedicated DSP hardware of just 10 years ago. This with simple low cost sound cards/interfaces and modern radios with built in "sound cards" combine to make the setup and experimentation of software generated digital modes an important part of our amateur radio hobby. These modes range from simple keyboard and weak signal modes such as PSK31 and JT65 to more complex high speed message/file modes with the ability to automatically adapt to changing signal strength and propagation conditions. WINMOR [1] developed by one of the authors in 2008 has seen good acceptance as a low-cost Pactor alternative in various messaging systems like Winlink 2000 and BPQ32. Each generation of protocol and increase in low cost DSP equipment provides an opportunity to expand both the performance and flexibility of software controlled digital modes. But the development, optimization and support of a full featured digital protocol require a substantial contribution of time and skills that should be spread over many applications, operating systems and years of use. The development of ARDOP started with a short list of target objectives:

- Open Design promoting targeting to various computer, OS, and hardware platforms
- Wide range of bandwidths to optimize spectrum usage
- Automatic channel adaptability…ability to adjust to changing propagation and S:N
- Support for both connected **ARQ** (**A**utomatic **R**etry re**Q**uest) and multicast **FEC** (**F**orward **E**rror **C**orrecting) transmission modes.
- Minimize interference potential (bandwidth negotiation and effective busy channel detection)

- Flexible operating modes and radios. Compatibility with popular voice grade HF and VHF/UHF transceivers using modulation optimized for the frequency of use.
- Full binary transmission and support for multi-language UTF-8 character sets.

These expanded over a period of months to first a skeleton specification and finally a full detailed specification with detailed spread sheets showing the composition, bandwidth, robustness and speed of several modes across a 200 to 2000 Hz (audio bandwidth) spectrum [2]. In deriving the specification care was taken to provide avenues to encourage experimentation yet not impact the compatibility of compliant implementations.

Virtual TNC with Host concept

The experience with hardware TNCs and the portability of virtual (software) TNCs such as WINMOR has confirmed the benefit and flexibility of separating the "TNC" or modem function from the host (user) application. This promotes portability and allows the same TNC code or hardware implementation to be used (without change) in a variety of diverse applications such as keyboard clients, messaging systems, tracking functions, sounding systems, emergency beacons, etc. We chose this path to allow us to focus first on the protocol and TNC and not the final user host application. To the user the virtual ARDOP TNC operates similar to a hardware TNC and like a hardware TNC can display operating parameters or hidden away to avoid clutter. Figure 1 is an example of the ARDOP Win TNC showing a small but rich panel to display operating parameters, transmission progress and for the entertainment of the operator!

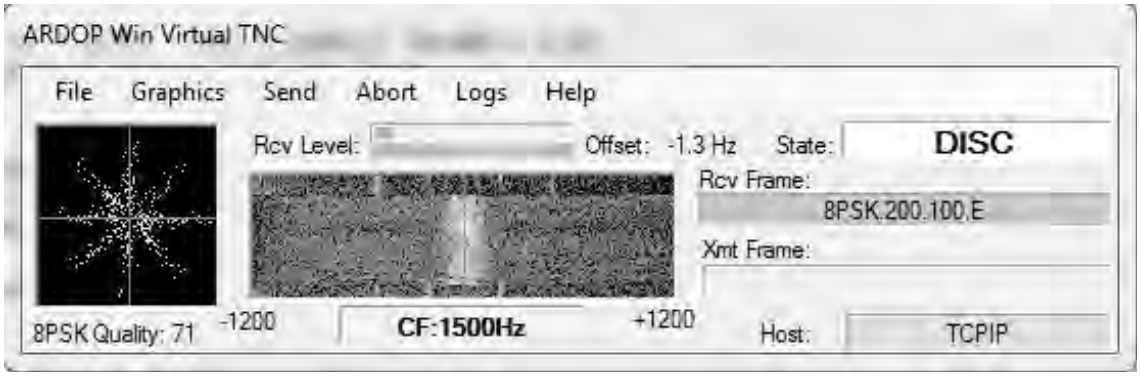

Fig 1. Screen capture of the ARDOP_Win TNC user interface/display

The virtual TNC can interface to the host program via a TCPIP connection (wired or wireless), a high speed serial connection or a wireless Bluetooth connection. This permits not only flexibility but the ability to operate the TNC/Radio combination remotely from the host software. Likewise a hardware implementation of the protocol (e.g. PIC microprocessor with sound card chip) could interface to the same host software and provide functional equivalence and compatibility. A simplified block Diagram of the ARDOP TNC is shown in Figure 2.

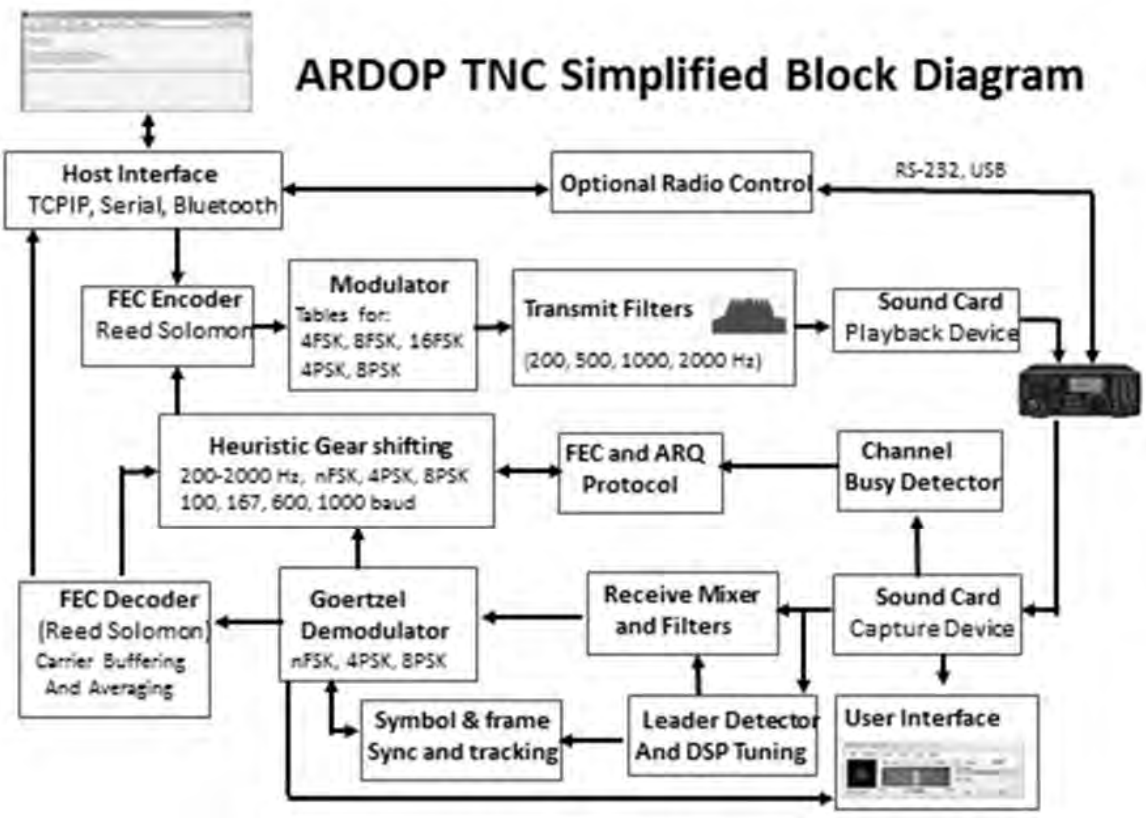

Fig 2. ARDOP TNC Simplified Block Diagram

ARDOP Performance

Most amateurs familiar with digital modes are aware of the tradeoffs required when it comes to robustness, bandwidth, signal strength and propagation quality. Some specialized modes can work deep into the negative S:N regions but they are very slow…sometimes exchanging only call signs. High speed modes permit sending large files and images but need fairly good signals, wider bandwidths and minimal multipath propagation or path compensation. Even within a 10 minute QSO or forwarding session wide variations in signal strength and propagation quality are often observed. One solution to this problem is for the sending station to adjust modulation type and bandwidth based on the *current* propagation channel. ARDOP uses a simple but effective mechanism to send the received decode quality (basically a "score" of the last received frame's symbol constellation) back to the sending station with every ACK or NAK. During the development of ARDOP a HF channel simulator was often employed to develop the modes, mechanisms and optimum FEC level to allow the sending station to rapidly home in and maintain near optimum modulation (and bandwidth in some cases). This allows the session to proceed with the highest data rate that can be supported with the *current* S:N, propagation channel and bandwidth. Figure 3 shows two typical net throughput measurements (200 Hz channel and 2 kHz channel) made during Alpha testing using long ARQ sessions on an HF channel simulator across various HF channel types.

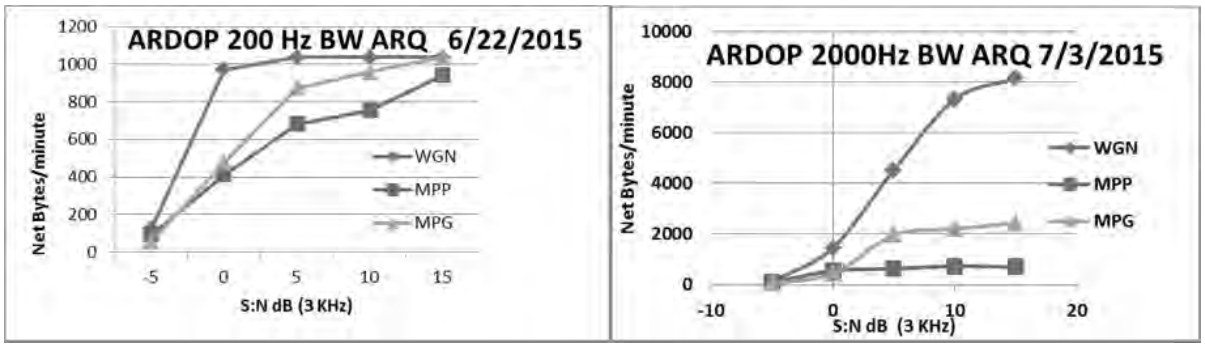

Fig 3. ARDOP performance over WGN (white Gaussian noise), MPP(multi-path poor) and MPG (multi-path good) channel types for 200 and 2000 Hz bandwidths.

Future plans include experimenting with “training sequences” and DSP path compensation techniques to allow higher performance during poor channel conditions.

Porting ARDOP to Other languages, OS and Platforms

Three significant challenges for this project from a programming perspective are as follows:
1) Porting the code from the initial rapid prototyping language used (Visual Basic .NET) to another language more readily usable on the various target platforms.
2) Targeting multiple platforms, such as Linux, Mac OS X, iOS and Android.
3) Finding alternatives to specialized interfaces (sound card, Internet and I/O) that were used for development of previous generations of applications by the same developers.
An interesting thing happens when you start looking into the various options and taking a hard look at the source code. For this project, conversion of the VB.NET code to C# was chosen, as a few of the source files were actually very similar to the original C/C++ code that can be found on the Internet from hams that developed software a decade or two ago. And while conversion to an alternate language may appear difficult the Internet is a good source of free tools to do rough conversions. The online code converter from Telerik [3] is the one chosen by one of the designers of ARDOP for this purpose. When the code is converted by the online tool, it is quite likely that it's not going to be ready to compile; the designer had to do a great deal of hand editing of the results to get it to compile and that is compounded by the number of files that interact in subtle ways. It also uncovers a lot of corner cases where VB.NET specific functionality has to be removed for a more workable product. This also provides an opportunity to lay the ground work for the second and third challenges; targeting multiple platforms and alternative interfaces.

When targeting multiple platforms, it is often best to understand the way each one handles user specific files such as application configuration files. It is also good to know what options exist for handling the interface to the sound hardware in the device the application will be running on. In the past, and this is often still done by application authors, configuration files have been placed in the same directory as the application executable is. This is fine in a single user environment, but not when installed on a multiuser system. The proper procedure is to place the configuration files in a folder

(also called a subdirectory) in the user's home directory. Global files with basic parameter values can be installed as well, if desired. Audio device detection can be handled one of two ways; with a custom library that is used by the software on the alternative platforms instead of the default library, or one that is available on all target platforms.

Typical Host programs

For initial on-air testing of the ARDOP protocol we needed a fairly simple host program where users could send beacons, basic keyboard text, and small files with ARDOPs FEC and ARQ modes exercising various bandwidths and modulation modes. Existing code from a prior project (V4Chat) was modified to interface to the virtual ARDOP TNC using a robust TCP IP interface. Fig 5 shows the basic ARDOP Chat host that provided setup for the ARDOP TNC, keyboard interaction, received data display and file editing and transmission along with a few conveniences like ADIF logging, beacon setup, and basic radio control (PTT and Frequency).

Fig 5. Basic ARDOP Chat host program used for initial keyboard testing.

Following initial debugging of the ARDOP Virtual TNC and ARDOP_Chat host programs John Wiseman GM8BPQ adapted his BPQ32 [4] host program to interface to the ARDOP Win TNC. This allowed additional functions including radio email and binary file transmission through the WL2K system using the existing B2 forwarding protocol. The following diagram shows how the BPQ Host interfaces to the ARDOP TNC along with conventional Packet and Pactor hardware TNCs.

This interface approach (separating the TNC DSP code from the host and interfacing through standard TCPIP , Serial or Bluetooth interfaces) allows the TNC code to be host application independent similar to the way a typical hardware TNC is.

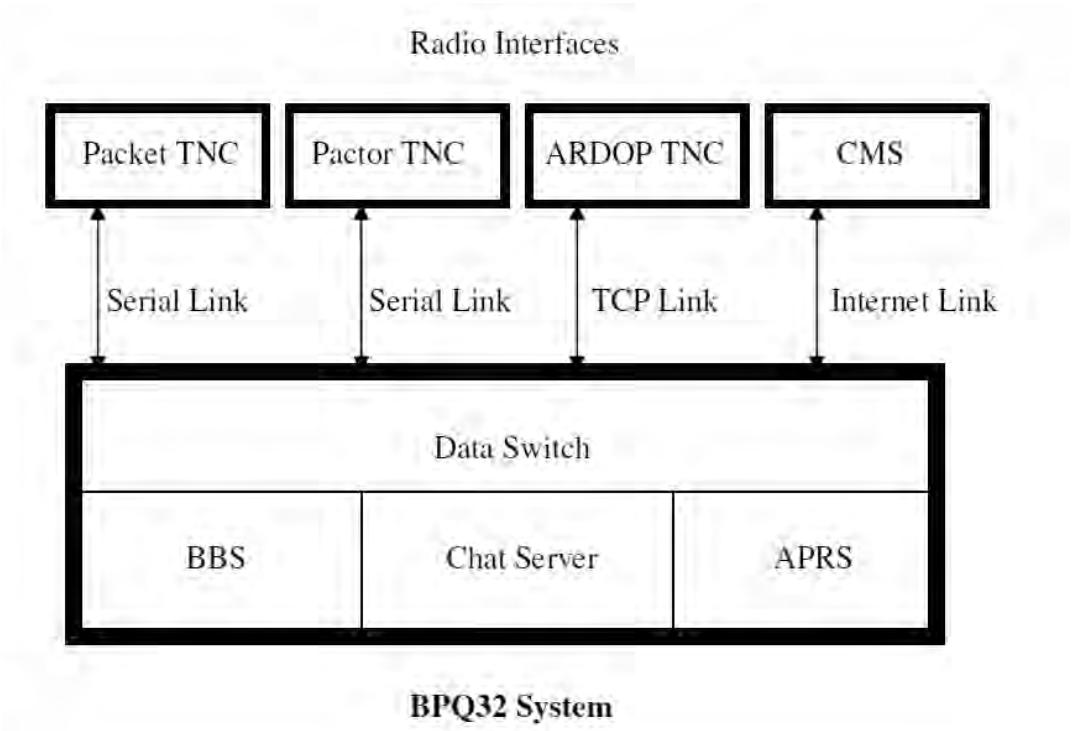

Fig 6. ARDOP Virtual TNC Interface to the BPQ32 System

Figure 7 shows a basic screen capture of an ARDOP B2F protocol session with the BPQ32 ARDOP TNC interfaced as described in Fig 6 above.

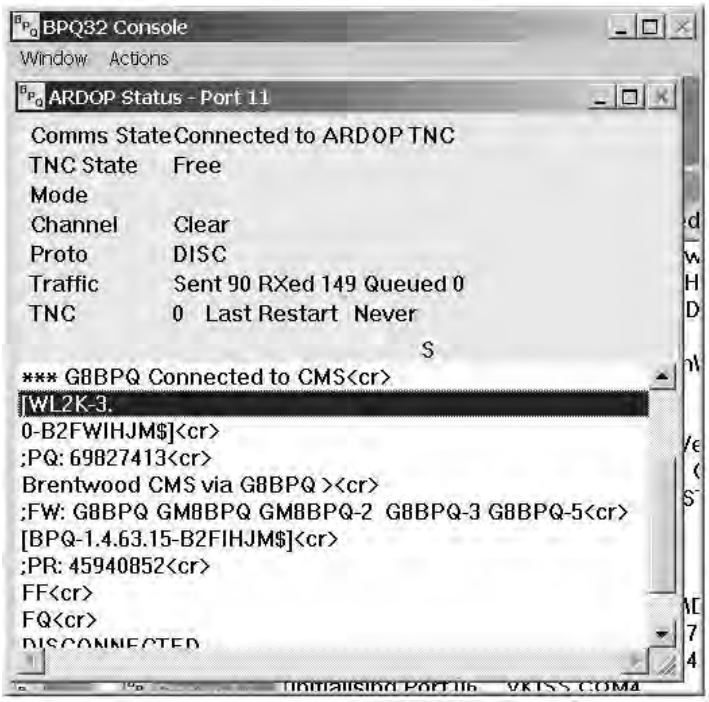

Fig 7. BPQ32 ARDOP ARQ session showing the interface to the WL2K Ham radio email system.

Project Status

The ARDOP project began Beta testing using both the ARDOP_Chat and BPQ 32 host programs in July 2015. The ARDOP Protocol spec is complete and the ARDOP virtual TNC is operational on both the Windows (Win XP- Win 10 using DirectX) and Linux (x86-Debian and ARM-Raspbian systems using MONO and the ALSA sound library) platforms and on Apple using the popular dual boot systems. A wide range of data modes covering speed and robustness ranges in excess of 40:1 are optimized for both HF (baud rate < 200 baud) and VHF/UHF FM (baud rate > 600 baud) are operational. As the Beta phase completes we will release the open source code along with a detailed testing and conformance document to allow those adapting or extending the protocol to insure basic compatibility with prior implementations. We have also initiated an effort to develop small low-cost hardware to allow wireless interfacing (Wi-Fi and Bluetooth) of small computing platforms (tablets, smart phones) to HF and VHF/UHF radios that would use the ARDOP protocol.

Credits

The authors wish to thank all those ARDOP Alpha and Beta testers from across the globe that have contributed to the development and testing of this new amateur protocol. Acknowledgement is also given to those programmers that wrote public DSP and encoding/decoding routines that were used in the ARDOP TNC. Specific reference of these is included in the commented source code.

References:

[1] (WINMOR…A Sound Card ARQ Mode for Winlink HF Digital Messaging, Rick Muething, KN6KB, 27th ARRL and TAPR Digital Communications Conference 2008)

[2] ARDOP Documentation and Code in Yahoo groups:
https://groups.yahoo.com/neo/groups/ardop_development/files
https://groups.yahoo.com/neo/groups/ardop_users/files

[3] Telerik on-line code converter. http://converter.telerik.com

[4] BPQ Host program by John Wiseman GM8BPQ. http://g8bpq.org.uk
https://groups.yahoo.com/neo/groups/BPQ32/info