SCAMP (Sound Card Amateur Message Protocol)

By Rick Muething, KN6KB (rmuething@cfl.rr.com)

Abstract
Digital modes enjoy increasing popularity and performance thanks to a better understanding of Digital Signal Processing (DSP) and continual improvement in the performance of modern computers, sound cards, and operating systems. SCAMP is a new experimental "wide-band" (2 KHz) digital sound card message protocol suitable for HF. SCAMP leverages the work by Barry Sanderson, KB9VAK and employs an ARQ "wrapper" around Barry's Redundant Digital File Transfer (RDFT) scheme to provide the error-free automatic operation necessary for today's modern digital message systems. This paper documents work in process developing and testing a new sound card mode that promises Pactor-like performance for HF channel transmission. Key words: RDFT, HDSSTV, Pactor, MT63, PSK-31, Winlink, DIGTRX, Sound Card Modes, Pipelining.

Background
Hams now can select from a number of software implemented digital modes for monitoring and sending information using the sound card found in virtually every personal computer. These modes offer not only lower cost solutions (no modem expense) but also an excellent experimental environment ....one of the cornerstones of our ham radio hobby.

Some modes support a listen or monitor only mode (e.g. We fax reception, FEC broadcast monitoring etc) essentially replacing the hardware "decoder" with a sophisticated DSP program running on a PC. There are also now a number of very popular conversational modes like PSK-31 and MT63 that have sprung up and serve the keyboarder very well.

However when it comes to the near perfect error-free reception required to automatically forward messages and binary file attachments used in modern amateur message systems (1) most of these "conversational" modes fall short in throughput and robustness. The rather strict synchronous Transmit / Acknowledge timing constraints of Pactor also make it very difficult to implement a reliable Pactor *connection* (as opposed to only m*onitoring*) with anything but the very fastest computers or dedicated DSP processors.

Barry Sanderson presented a paper at the 2001 Dayton Hamvention and released software utilities that implemented an effective robust transport mechanism initially called HDSSTV for transmitting digital image files over noisy HF channels (2). Barry's fine work later expanded and renamed RDFT has sparked other programmers like Roland Zurmely, PY4ZBZ to develop programs like DIGTRX based on Barry's RDFT technology and software primitives (3).

After experimenting with several alternative approaches (e.g. ARQ wrappers around multi channel PSK31/63 or MT63) and working with DIGTRX it appeared Barry's RDFT modulation scheme and software primitives could be adapted to reliably forward error-free binary data in automatic and semiautomatic message systems.

FEC and ARQ

It is important to clarify and understand the two primary mechanisms for achieving error- free digital data forwarding. First by "error-free" I mean that the probability of an uncorrected error is sufficiently low (generally less than 10^{-6}) not to impact the practical use of the system. Any coding or retry scheme can fail but we can, by proper design and coding, reduce that probability of failure to an acceptably miniscule level.

FEC stands for Forward Error Correcting and essentially is a coding mechanism that adds redundancy to the data stream to aid in detecting and hopefully correcting errors. In some modes (e.g. FEC broadcasting) we can achieve FEC by appending simple parity or sum check bits and redundantly transmitting (duplicating) the data. These schemes are simple and effective especially for broadcast data (ARRL bulletins, NAVTEX weather bulletins etc) but achieve robustness at the expense of significant reduction in the theoretical channel capacity. Without a "back channel" there is no way the FEC sending station can be sure the receiver received the data error-free or adjust the level of FEC redundancy.

ARQ stands for Automatic Retry reQuest and employs a back channel (from the "receiving" station to the "sending" station) that acknowledges (ACKs) or negatively acknowledges (NAKs) indicating the data was received correctly or not. In some cases a full repeat is requested or perhaps just some portion of the message (or additional error correcting coding) is requested. In a half duplex ARQ system (typical on HF) the timing can be rigid and synchronous as in Pactor or rather loose and asynchronous as in packet. All ARQ systems must have a robust mechanism for error detection and most also employ at least some level of FEC as well to try and correct some errors without retries.

Since ARQ keeps the sending station "aware" of the received message progress it can insure that not only the message is received "error free" it also allows the sending station to know by acknowledge that a message (or how much of a message) has been received and processed. Within a typical digital message system using FEC and ARQ there are also normally higher forwarding protocols used.

Challenges of DSP processing on the PC

One of the big challenges in creating an efficient sound card based message protocol is the inherent lag or delay in sound card processing. Sound cards basically capture digital audio samples (to memory or a file) in a continuous stream. The PC software must break this stream into groups of samples (usually a power of two from 1024 to 8192 sample) to process with DSP algorithms to do filtering, demodulation, and decoding. Some sophisticated decoding algorithms take considerable computational effort and even with modern high-speed processors this adds significant delay in the ACK or NAK signaling required for ARQ. For synchronous ARQ systems like Pactor this requires significant peak computing loads that have to date required dedicated hardware or DSPs. If we accept an asynchronous ARQ system that allows the receiving end to delay the ACK/NAK we can solve this problem but at the expense of significantly degrading the precious channel capacity. For SCAMP I looked closely at the typical computational and timing demands common to popular HF protocols. What I found was that through the use of parallel threads it was possible to overlap the low computational tasks of waveform capture and transmission with the high computational loads of decoding and encoding.

Pipelining

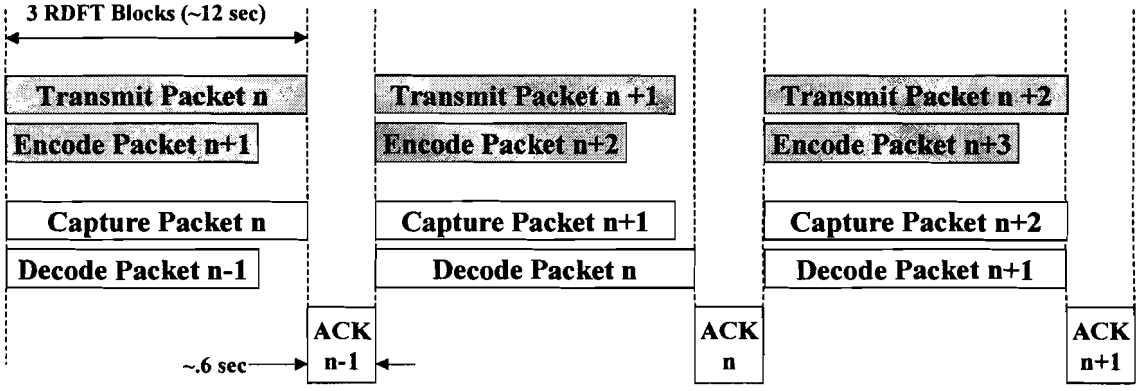
To accommodate the delay introduced by the sound card capture, DSP and decoding delays without degrading the channel throughput I came up with a pipelining scheme and an asynchronous ARQ protocol. Together these work to reduce the ARQ and sound card processing demands while still

providing a way for slower processors to participate but at reduced channel capacity. Pipelining is possible because of two important characteristics of digital message systems:

1) The sender and receiver are automated so a delay of several seconds in the message due to the pipeline is not a significant problem as it might be with a "live" keyboard QSO.
2) The typical message consists of many packets or frames allowing messages to be pipelined with minimal net overhead. The inefficiency of starting and ending the pipeline is less when there are many packets or frames.

The following figure shows a typical pipelining and Asynchronous ARQ mode used in SCAMP. You can see the CPU intensive processing functions (encoding and decoding) are done in parallel with the low intensity transmission and capture tasks. When the sending and receiving CPUs can complete the processing in less than the actual transmission time this is a transparent and efficient mechanism. Essentially pipelining levels out the processing "peaks" and this allows us to use modest processors and operating systems with limited real-time interrupt capability...our typical PC. One requirement of pipelining is that it does requires the software (and programmer!) to efficiently support multiple parallel threads...something early DOS and Windows systems could not but today's OS and development frameworks can do with relative ease.

SCAMP Pipelining
Long Data Packet Timing with all correct reception



Overlaps low Demand Transmit with CPU intensive Encoding on Transmit side
Overlaps low Demand Capture with CPU intensive Decoding on Receive side

RDFT and SCAMP

Barry's RDFT has some very important characteristics that make it well suited for this type of asynchronous ARQ scheme. RDFT is basically a "batch" process that operates independently on each captured wave file from the sound card. Barry developed sophisticated math and signal processing stand-alone utilities that generate or decode sound card wave files. Essentially RDFT provides two DLL or batch programs (for DOS/Windows or Unix) that perform the complex and CPU intensive encode and decode functions. The encode function takes a binary file and encodes it (with a selectable degree of FEC redundancy) to a sound card wave file...suitable for directly modulating a HF SSB transmitter. The more CPU intensive decoding utility does the reverse, decoding the captured wave file (with all its HF channel noise and distortion!) from the sound card and producing the (hopefully error-free) binary file. RDFT can work effectively on both short and long binary files. RDFT's multi-level FEC Reed-Solomon encoding and sophisticated soft decision decoding make the mechanism robust while still

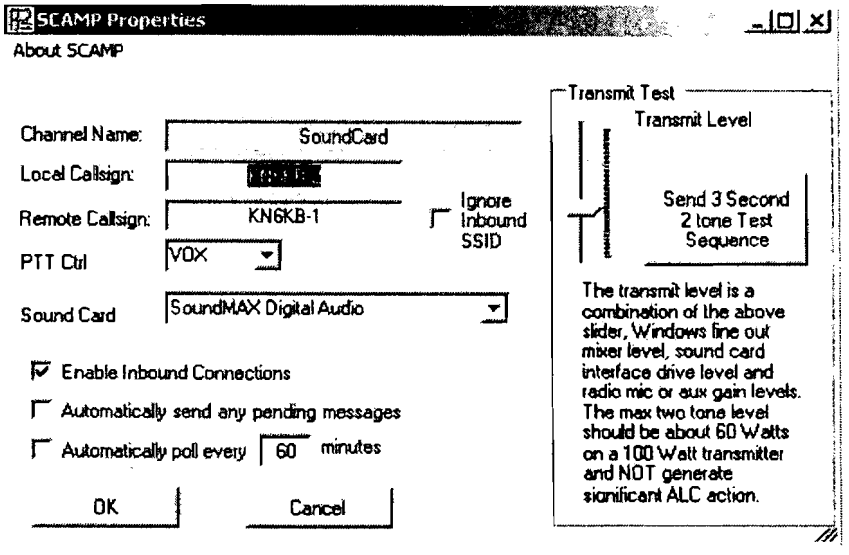
achieving a reasonably high information content vs. bandwidth. The variable level of redundancy (10-70 %) allows adapting the encoding to fit both good and poor radio channels.

An ARQ "wrapper" however must be added to the basic RDFT transport mechanism. This wrapper breaks the messages to be forwarded into standardized serialized "packets" and implements a back channel for the required ACK or NAK signaling success or failure during reception. The ARQ mechanism must also implement a robust mechanism to control the start and stop of the sound card capture process to "frame" each of the packets. In SCAMP I used a very robust .6 second burst of 10 tones to signal packet start, packet stop, ACK and NAK. These tone bursts have a very low correlation to the data and each other and are very effective even in marginal channels.

The ARQ wrapper also implements two modes to optimize throughput. In the non pipelined short-packet mode the sending and receiving station asynchronously wait to encode, send, capture, decode and ACK/NAK short high redundancy blocks (80-260 information bytes). This is used to control the link and the higher-level forwarding protocol where pipelining would be inefficient due to link turn around and pipeline startup overhead. In the pipelined mode SCAMP uses "packets" of 3 RDFT blocks yielding 260-800 information bytes depending on coding. This provides better overall throughput for messages from 1000 to more than 20,000 bytes.

Example Implementation with Paclink

Developing SCAMP has certainly been a learning and challenging experience. To begin experimentation I used DIGTRX to become familiar with RDFT. Later I developed a special version of the WL2K Paclink program (4) written in VB.NET to support the sound card using MS Direct X utilities. The capability of VB.NET to launch and manage other processes and threads was necessary to permit overlapping the time-critical threads of playback, capture, start/stop burst correlation, encoding and decoding. The following screen shows the simple setup required in Paclink to setup and run SCAMP.



Once a connection is manually initiated, SCAMP, using Winlink's B2F protocol (5), automatically forwards messages requesting repeats as required and monitoring for error conditions and timeouts. The performance of SCAMP depends of course on the quality of the HF channel. To perform realistic and repeatable tests a DSP based HF channel simulator (6) is used in addition to on-the-air RF tests.

The repeatability of the channel simulator and the flexibility of easily modifying the simulated channel (S/N, fading, multi-path etc) are very helpful in optimizing SCAMP's ARQ mechanism and measuring and comparing performance.

Initial tests indicate SCAMP should achieve net binary message throughput (after all ARQ and forwarding protocol overhead but before any B2F compression gains) in the range of 3-4 Kbytes per minute over typical HF channels. This will probably be between Pactor II (500 Hz bandwidth) and Pactor III (2.2 KHz bandwidth) in similar conditions and 4-6 times faster than typical Pactor I throughput. When the development and testing is complete I plan to publish detailed comparison data of these and other modes based on actual measurements made using the HF channel simulator.

Future work

SCAMP is not yet ready for deployment or extensive beta testing. The ARQ and pipelining mechanism must be made virtually bullet proof with all required timeout mechanisms. Although SCAMP is intended for manually initiated auto forwarding it will still require an effective busy channel "detector" to prohibit interference to existing signals in the channel bandwidth. Barry is also working on more advanced RDFT modulation schemes (7) that may prove an effective way to increase throughput during good propagation and possibly allowing effective application in VHF SSB or FM channels as well. I plan to fully document the SCAMP protocol and algorithms allowing others to duplicate and build on this effort. Barry Sanderson's RDFT is documented and the source code released under the GNU General Public License (GPL).

Once SCAMP is ready, the interfaces to systems like Winlink 2000 must be put in place to support it. There is however considerable motivation for implementations like SCAMP as good performance lower-cost alternatives to modes requiring dedicated hardware or DSP processors.

Finally to fully utilize these new types of sound card modes our current band planning regulation must be restructured. We can no longer accept the regulatory time lag of segregating signals based on mode (e.g. CW, RTTY, PSK31, Pactor, SSB voice etc). The ARRL is now in the process of developing a proposal to the FCC to segregate modes by bandwidth, grouping all modes of *similar bandwidth* to specific HF band segments. This will not only reduce interference but also promote and encourage the development of new digital modes and protocols.

Summary

SCAMP represents a new approach in the implementation of sound card modes where the CPU intense DSP and encoding/decoding functions are first segregated from and then overlapped with the functions of user interface, sound card control, and higher level message protocols. This approach should encourage more collaboration and experimentation in this segment of our hobby by partitioning these often-diverse skills

Acknowledgements and References:

This work was made possible by the contributions and encouragement of several fellow hams. As Sir Isaac Newton said "...it is by standing on the shoulders of giants". Thank you for your time and support: Barry Sanderson KB9VAK, Roland Zurmely PY4ZBZ, Dave Jones KB4YZ, Vic Poor W5SMM, Steve Waterman K4CJX, and Scott Thile K4SET.

(1) *Winlink 2000* ...A Global Ham Message Transfer and Delivery System, DCC 2000. **http://www.winlink.org/News/dcc2000.htm**

(2) RDFT (HDSSTV) by Barry Sanderson, KB9VAK
http://www.svs.net/wyman/examples/hdsstv/index.html

(3) DIGTRX by Roland Zurmely, PY4ZBZ
http://planeta.terra.com.br/lazer/py4zbz/hdsstv/teste1.html#digtrx

(4) Telpac and Paclink- Streamlined AX.25 Packet Server and Client for a full service Ham Radio Messaging Network, DCC 2003.
http://www.winlink.org/News/DCC2003.htm

(5) Winlink 2000 Message Structure and B2 Forwarding Protocol
http://www.winlink.org/B2F.htm

(6) A Low-Cost HF Channel Simulator for Testing and Evaluating HF Digital Systems. DCC 1999 Johan B. Forrer, KC7WW
http://www.johanforrer.net/SIMULR/

(7) New Digital Modes Proposed by Barry Sanderson, KB9VAK
http://www.tima.com/~djones/digmodes.htm