Comments on HF Digital Communications
Part 2 -- Data Protocol Issues

Tom Clark, W3IWI
6388 Guilford Road
Clarksville, MD 21029

1. Introduction: In part 1 we discussed some of the link-level issues in terms of the effects imposed by the ionosphere on HF signals and possible Digital Signal Processing (DSP) approaches to those issues.

In part 2 we assume that bits can be reliably delivered at reasonable speed and turn to questions like

- How can more information be crammed into each bit sent?
- How can the overhead associated with each transmission be minimized?

  How can the number of transmissions (and hence the number of times the overhead must be paid) be minimized?

2. Connected mode AX.25 -- the *WRONG* solution to the HF problem: Having watched literally thousands of attempts to move packet mail, I have become convinced that connected-mode AX.25 is a bad choice for use on HF. The U.S. stations on the HF nets operate under the ARRL "SKIPNET" STA (wherein the FCC authorizes unattended operation); most nets are "closed" with each station allowing connections only from other net members .

Despite the "closed" nets and well-equipped stations about half the attempts to forward mail result in timeouts due poor propagation and QRM. One of the most serious sources of QRM is from other net members. It appears to me that nets like 14109 kHz have sufficient activity that its channel capacity is reduced to the "ALOHA limit" of ≈ 18 % which is shared among all the net members.

Messages longer than about 2 kbytes in size carried on the busier HF nets have a significant probability that they will result in a time-out and hence must be re-sent. The number of time they will need to be re-tried is proportional to the message size, and the time required for each attempt is also proportional to the message size. Therefore when a message exceeds a critical size, the channel time required to send that message will increase as the SQUARE of the message size. As a result many HF SYSOPs have adopted message size limits of 2-3 kbytes for traffic that will be handled.

Because of the combined difficulties associated with the ionosphere, plus QRM and QRN, plus the current modem technology used on HF (see Part l.), a typical HF link has a bit error rate (BER) in the range of 1: 10^2 to 1: 10^3. Thus any packet frame longer than ≈ 500-1000 bits will probably not work; this has led to stations using PACLEN parameters in the range 40 to 80. Thus AX.25's per-frame overhead is about one-third of all bits sent.

In the present AX.25 protocol, if 4 frames are transmitted and the receiving station gets good copy on frames number 1, 3 and 4, then the inability of the protocol to reassemble frames requires frames 2, 3 and for to be resent. Because of this deficiency in the protocol, the typical HF SKIPNET station operates with MAXframe set to 1 or 2. The present protocol lacks any frame reassembly capability for historical reasons; the original TNCs circa 1983-84 didn't have enough computing capability or RAM to support the function.

Eric Gustafson, N7CL has developed an improved "PRIACK" modification to AX.25 which is now available for TNC-2 (and clones) and AEA TNCs. PRXACK gives channel priority to < ack > frames and uses p-persistent CSMA algorithms for sending I-frames. Despite PRIACK being available for nearly 2 years, it has not found wide acceptance. Many HF operators say "it slows down MY transmissions too much". Even if it were accepted, PRIACK is only a band-aid applied to an inappropriate protocol.

Another inefficiency (not intrinsic to AX.25) comes from the fact that all messages sent on HF are plain ASCII text and yet a full 8-bit byte is used to send the data. Only about 6.5 bits are needed for each character, corresponding to ≈ 20% loss of channel utilization. Even better would be to use data compression techniques (like .ZIP or .ARC) with full binary data transmission, which would make ≈ 50% improvement.

Add to all these factors the wasted keyup time for amplifiers (if used) on each end of the path required for each frame sent, and the time for the other station to send an <ack> and it becomes apparent why the real data throughput on HF channels is only a few tens of bits/second.

These factors may be summarized:

1. There is a need for new improved modem technology outlined in Part 1.

2. Radical protocol surgery is needed to solve the problems of timeouts, multiple re-transmission of messages, channel sharing data compression, etc.

3. HEHFPRO -- Another Connectionless Protocol: HEHFPRO means "High Efficiency HF PROtocol. With HEHFPRO it is proposed to make use of unconnected AX.25 <UI> datagrams as an alternative to the present connected mode protocols.

Suppose that the W3IWI BBS has 23 messages to be sent to the European mail gateway at 4XlRU. W3IWI would collect all the messages into a single export file which might be 9132 bytes long. Although not required, for efficiency W3IWI compresses the first file with PKZIP into a new file 3932 bytes long. W3IWI then transmits a <UI> frame addressed to 4X1RU that says (in appropriate computerese):

> Hello 4X1RU. W3IWI Calling. I have a binary .ZIPed mail file for you which is my number RU11367. It is 3932 bytes long will be sent in blocks of 64 bytes/block. Let me know when you are ready.

If 4XlRU doesn't acknowledge W3IWI, the same <UI> frame is re-sent a few minutes later. When 4X1RU finally hears W3IWI, he responds with a response <UI> frame acknowledging the request. 4X1RU knows that the data portion of the blocks it receives subsequently will be the 64 data bytes long, plus 7 bytes to flag this data as a part of message RU11367, plus a two byte frame number or a total of 73 bytes long.

Since (3932 = 61*64 + 28), 4X1RU knows to expect a total of 62 such blocks and that the last block will have only 28 data bytes, and allocates space to hold the message. He prepares an response message with enough bits to flag each of the incoming blocks (in this case 8 bytes = 64 bits is the appropriate size) looking like

```
00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000011
```

with 62 zeroes. This 8-byte field plus a few bytes of housekeeping information is sent as a <UI> frame response.

W3IWI then sends a suitable number of the 64-byte data blocks in one transmission, uith each block corresponding to a separate <UI> frame with length 73 (including the message ID and the two-byte frame sequence

number). The number of <UI> frames sent in one transmission can be tailored to suit conditions, but lets assume 24 is chosen.

Of the first 24 frames sent, suppose that 4X1RU copies 14 frames numbered 0, 3-10, 14-17, 19, and 22 and would send a response <UI> frame with the 8-byte field now reading (from left to right):

10011111 11100011 11010001 00000000 00000000 00000000 00000000 00000011

The next tranmission would re-send the missing frames and append some new ones. After a few such attempts the acknowledge frame might look like:

11011111 11111111 11111111 11111101 11111111 11101111 11111101 11111111

with only frames numbered 2, 30, 43 and 54 missing. W3IWI would concentrate all efforts at filling in the missing pieces until an "all ones* response was received from 4X1RU. 4X1RU would then use PKUNZIP to de-compress the file and re-post the messages to their respective destinations on his "conventional" BBS.

At this point 4XlRU has informed W3IWI that he has the message complete. On receipt of this W3IWI would mark the outgoing file as delivered and initiate the <UI> datagram equivalent of a disconnect.

4. A few other design concepts: Even if no data transmissions were heard for a while from W3IWI, 4X1RU would continue to send <UI> frames indicating the current status of message RU11367 every few minutes so that when conditions permit data transfer can resume. If W3IWI encountered other activity on frequency, his software would enforce an automatic backoff. This process might take a few minutes or it might take hours but the messages would get through and there would be no such thing as a timeout.

Although a file containing mail was used for this example, the procedure is independent of data type, so it could also be used for file transfers. If the file is .ZIPped or .ARCed, then the file name is recovered on de-compression.

In this example we showed only one message file being transmitted. There is no reason that a given station could not have several sessions in progress at a given time in either direction. All "hooks" to implement these concepts (but not the HEHFPRO protocol) already exist in KA9Q's NOS software "engine".

Another potential application for an HEHFPRO-like protocol would be the amateur development of point-to-point meteor scatter links. The meteors in this case are not the "pings" associated with VHF DX operation which occur during meteor showers; rather they are the steady background of meteors which occur all the time. These meteors are most useful on forward scatter paths ≈ 800-1200 km long at frequencies like 6M and 10M. Under such conditions the meteor signals will rise from the noise for 1-2 seconds with little Doppler shift; such bursts will occur at a rate of about once per minute. To capitalize on such bursts, individual packet frames (blocks) must be shorter than about 0.5 seconds and it is desimble to be transmitting as much as possible to increase the probability of catching a meteor.

5. Acknowledgements: HEHFPRO has a lot of similarities to other protocols in use. 'In particular it draws heavily on TCP/IP's use of datagrams and I acknowledge KA9Q's patient tutorials on how things should be done. The main difference is that HEHFPRO has, in essence, a much longer sliding window and more extensive use of frame reassembly.

The use of a series of response bytes, with each bit corresponding to a fixed portion of the associated data/message, has been extensively used to load data and DCE messages into the UoSAT spacecraft for a number of years. Packet mail is handled by the DCE gateways by the same file export/import route described here.
G0/K8KA and NK6K have prepared detailed specifications for a similar datagmm protocol to be used with the PACSAT experiments on UoSAT and MICROSAT satellites. The main difference between HEHFPRO and the satellite protocols is that all terrestrial users in the satellite's footprint can hear the satellite and the satellite can pace data transfers over full-duplex links. On HF networks, all stations are presumed to be peers and the half-duplex links are plagued with "hidden terminal" problems.

I'd also like to offer special thanks to N4HY for serving as a sounding board for many if the ideas in both parts one and two of this tome.