PACSAT Broadcast Protocol

Harold E. Price, NK6K
Jeff Ward, G0/K8KA

ABSTRACT

The case for a broadcast protocol for use on PACSATs is made, and a suitable protocol is proposed. In the proposed protocol, files of general interest are chopped into <UI> frames and repeatedly sent by the PACSAT in round-robin fashion or on request. Each <UI> frame datagram contains enough information for groundstation software to place the frame in the correct position in the appropriate file. Information indicating the type of file being received is also included in each frame.

A protocol for groundstations to request retransmissions of specific frames is included.

1.0 Background

PACSAT is a generic term used to describe a digital store-and-forward satellite mission in the Amateur Radio Service. Two of four Microsats and one of two UoSATs launched in January 1990 will have PACSAT as a primary missions.

PACSATs will use the AX.25 frame as the basic link layer protocol, either in the full AX.25 connected mode, or as unconnected <UI>-frame datagrams.

A PACSAT will have several types of data to send. Some of these are:

1) Forwarded mail messages. These are messages are not destined for PACSAT as an endpoint, but are in transit between forwarding gateway stations.

2) Personal electronic mail messages. These are messages to and from individuals who are using the satellite as a BBS; entered either directly by a human-run connection, or by using a program that pre-formats messages for fast or unattended transmission. These messages use PACSAT as an endpoint. It can be argued that all mail should be forwarded mail, that no one use PACSAT as a direct BBS. There are three counter arguments:

a) For access in remote areas without a terrestrial infrastructure in place.

b) For emergency access by minimal ground stations.

c) To permit large numbers of people to have a direct hands-on experience with satellite communications.

3) Realtime Telemetry. These are current spacecraft telemetry values, such as solar array power, internal temperature, etc.

4) Stored Telemetry. This is a file of one or more telemetry values stored over time, for example. the output of the solar arrays once a second over the last orbit. This is usually called "Whole Orbit Data" or "WOD".

5) Bulletins. These are items of general interest. orbit predictions, AMSAT News, Gateway, etc.

6) Point-to-multipoint messages. These are messages which one PACSAT user wishes to send to several other PACSAT end users. This may be implemented using "mail groups" or "CC" lists.

Items 1) and 2) above are primarily point to pointoperations, either an individual reading his

own mail. or a gateway forwarding mail to be picked up by another gateway.

Item 3) is a self-contained frame that needs no other frames to convey its information.

The other items are germane to this discussion. Both are data types that would he of interest to a large number satellite users people. Any time that more than one user is interested in the same information, it will generally make more sense to broadcast it (send it once for reception by many) rather than send the information separately to each individual.

Current terrestrial packet radio use is highly inefficient. If 160 people in southern California want to read the 20k byte ARRL Gateway newsletter, they each log into a BBS and read it separately. This requires 3.2MB of data to be sent. Because of packet overhead, collisions, acks, etc., the actual number of bytes sent, and the total channel time, is much higher.

Assuming a very optimistic average throughput of 120 bytes/second, it takes 7.5 hours of channel time to allow 160 people to get a copy of the bi-weekly Gateway. Taking the standard loaded aloha 18% of that figure, 42 hours of channel time are required. Because there are 24 hours in a day, and there are seven packet channels in use on 2 meters alone in southern California, it is possible to accumulate 42 hours of channel time in less than a day. The inefficiency of the system is masked by the large amount of time and RF spectrum available.

Now assume that one wished to distribute Gateway on a Microsat, an environment where time and spectrum are at a premium. The Microsat is visible for about 14 minutes a pass, or about 60 minutes a day. It would then take 42 days to accumulate 42 hours of channel time, since even though the Microsat has multiple uplinks, it has only one downlink. UoSAT-3 (UO-14), with its 9600 baud downlink, would require only 5 days to move the same data.

Even if we assume a very disciplined set of users who access the satellite one at a time, we can move the efficiency from 18% to nearly 95%, enabling UO-14 to service the 160 users in 1.5 days, and Microsat in 8.5. This is assuming all of the single downlink frequency was devoted to the activity of downloading Gateway to individual users.

There are several conclusions that can be drawn from this. One is that a PACSAT is useless for dealing with a large number of individual users if they all want individual copies of the same file. It would be far better to have a gateway in the local area read a copy, then distribute it on the ground for everyone else. This is, however, not much different than what we have now, and we're looking to use the satellite to improve general conditions.

Another conclusion is that if there were a way to send a single copy of Gateway that could be seen by all 160 users at the same time, this would reduce both the load on the satellite, and the load on the terrestrial network. Since the satellite CAN see all 160 users in southern California at one time, the solution is now clear.

A broadcast protocol for items of general interest is clearly desirable. A strong case for one can be made in the terrestrial network as well, but the multiplicity of frequencies and the 24 hour availability of VHF/UHF networks makes the advantages harder to see. The single downlink of a PACSAT, and the tyranny of orbital mechanics makes the need more evident in the satellite environment. In a broadcast mode, UO-14 can transmit at least 3.2 million bytes of data to a station in the mid latitudes in an average day, AO-16 and LO-19 can each send at least 432,000. That is a lot of mail.

There is another advantage to a broadcast protocol, and again, it is more telling in the satellite service. A broadcast protocol, being one way, does not require a transmitter at the receiving site. The complexity of the ground station is reduced as the 2 meter transmitter and its antenna are not required to receive the broadcast.

Finally, the PACSAT environment gives another encouragement to broadcast protocols: because the transmit and receive frequencies are on different bands, it is inherently full-duplex. Since PACSAT will be the only transmitting station on the downlink, a major source of data-loss, collisions with other stations, is removed. When the link is good, there is no need for retransmissions, making the broadcast protocol more efficient than the normal ACK/timeout AX.25 protocol.

2.0 Attributes of a Broadcast Protocol

The AX.25 protocol, in its connected mode, makes the following guarantees:

1) All bytes are received once, in the order they were sent, with no gaps.

2) No bytes are received in error (within the limits of CRC16).

This means that the two stations can establish a connection, and then send a file. When the expected number of bytes or an end of file marker is received the receiving station can assume that it received the entire file (or message) correctly.

The general philosophy is that the sending and receiving stations work together to retransmit each frame (or small group of frames, **1-7)** repeatedly until it is received correctly, before moving on to the next frame.

The connected mode requires two-way point to point transmissions, and is not suitable for a broadcast protocol.

The unconnected mode of AX.25 make these guarantees:

1) Byte order is maintained within a frame only.

2) No bytes in a frame are received in error (within the limits of CRC16).

One or more frames may be missing within a group of frames. Although some TNCs allow you to receive frames with CRC errors, you can receive no reliable indication that you have missed a frame. AX25 does not provide frame sequence numbers in these frames. You can not know if the frame you have just received immediately follows the previously received frame, or if you missed some.

Although on average, the new PACSATs have the strongest amateur satellite signals to date, the receiving station will still get occasional dropouts from local conditions, polarity reversals due to the spacecraft's changing orientation, 70cm radar, etc. Loss of signal as the spacecraft goes over the horizon will also cause a loss of frames.

All of this means that if you receive a file as a set of broadcast < UI > frames, AX.25 itself does not provide enough information to allow you to tell if the entire file has been received, or if it has gaps. A broadcast protocol must provide this missing information. The broadcast protocol wouldride inside a standard AX25 < UI > frame.

Ideally, the broadcast protocol would have the following attributes:

1) Any frame, when received independently, can be placed in the proper location within the file to which it belongs.

2) When the all frames have been received, the receive station can tell that the file is complete.

3) For file types where it makes sense, partial files should be usable. For example, if a data compression scheme is used, the file should be able to be incrementally decompressed, e.g., a **20000** bytes of a 20256 byte file should not be useless simply because the first 256 bytes are missing.

To carry this to full frame independence, this means that compression algorithms should be such that the decompression of one frame does not depend on the receipt of any other frame. This may require the use of a less-efficient compression algorithm for files which are to be incrementally decompressed, but partial files, and perhaps all of the information, can be extracted from a smaller set of received frames.

Files that are not meant to be incrementally decompressed may use any compression scheme, the broadcast protocol maintains data transparency.

Four rules can be derived from the above desires:

1) Each frame must contain a file identifier

2) Each frame must contain something that identifies this frame's position in the file.

3) The file must contain a special record that defines file attributes, particularly file size. Other items of interest would be the **actual** file name, creation date, etc.

4) If a partial file is to be usable, especially if compression is used, each frame must contain enough information to allow use of a partial file.

2.0.1 Additional Error Protection

It is most likely that TNCs operating in KISS mode will be used to receive the UI frames which make up PACSAT Broadcasts. The authors' experiments indicate that the link between the KISS TNC and the user's personal computer may be prone to errors caused by a lack of flow control. This is especially true when using high radio data rates such as UoSAT-OSCAR-14's 9600 baud link.

Although the AX.25 CRC assures that only correctly received frames will be passed on by the KISS TNC to the host computer, we are now not certain that the frame reaching the host computer is error free. If we process incorrectly received frames, even occasionally, files will be lost or corrupted. Thus, we must add some error protection to the broadcast frame.

After experimentation, a 16-bit CRC was selected.

2.1 Frame Header Contents

The above rules define a frame structure that looks like this:

<flags> <file id> <file type> <offset> <data> <crc>

Each field is discussed below.

2.1.1 file id

At first glance, the simplest file id would be the actual name of the file. Since the PACSAT file system permits 8 bytes of name and 3 bytes of extension, this would lead to **11** bytes of overhead per frame, or 4%. This is somewhat large. There is also the problem that the contents of a file named "error.log" may change, and so the file name is not truly a unique file identifier.

To overcome this problem **PACSAT** assigns each created file a "file number", which is 4-bytes long and will uniquely identify the file. This number is used as the file-id in the Broadcast Protocol header.

The receiving station will not know the actual name of the file until the file header record is received. This record could be transmitted more frequently than other records, increasing the chances that it would be available in any partially-received file.

2.1.2 offset

Each frame of broadcast data must contain the position in the file where the data came from. Each frame, then, contains an offset field. The offset is the byte number of the first byte in that frame, relative to the first byte in the file. For PACSAT files, byte 0 is always the first byte of the PACSAT File Header.

To place a received data frame into a file, one need only:

```
lseek( (long) offset);
write(handle, data, frame size)
```

While it is easy to insert the frame bytes into the file, it is much harder to know when all the bytes have been received. There are several methods, one is maintaining a list of holes in the file, much as some TCP/IP implementations do when reassembling IP packets.

The size of the offset field is important, too small and the size of broadcast files are limited, too large and the overhead for each file is increased. We've chosen 24 bits as a compromise between efficiency and maximum file size. This allows for files up to 16Mb.

2.1.3 Flags

The flag field is a byte which provides option bits. Two bits are currently defined. "Length" says that an option length field is present in the frame header. "EOF" says that this is the last frame of the associated file.

2.1.4 File type

If a partial file, which could be just a single frame, is to be useful, some information on the file type must be provided with each frame. Examples of file types are:

ASCII text bulletins
Images from UoSAT-E CCD

Stored telemetry
Digitized voice
Machine-ready Keplerian element updates
Incrementally decompressable files
Non-decompressable files

The file type can also inform the groundstation software that a file is incrementally decompressable, or not compressed at all.

The file type byte is the same byte which is found in the file header record in the file. File types are assigned and defined in a separate document.

2.1.5 data

This part of the frame is the file data, compressed or uncompressed. The file type can be used to tell what type of data is present.

2.1.6 CRC

Reception errors in the AX.25 UI frame can be detected using the 16-bit CRC which is part of the HDLC frame, and this low-level error detection would, ideally, be enough to protect the Broadcast protocol frames. However, in reality. the UI frames are usually received by a KISS TNC, and the link between the KISS tnc and the host computer may be unreliable (especially at high speed) due to the lack of flow control in the KISS standard. For this reason, we have chosen to append a 16-bit CRC to the broadcast frame WITHIN the AX.25 UI frame data field. This CRC will be calculated using the XMODEM CRC specification, which has been successfully and efficiently implemented on many microcomputers.

2.1.7 File Header

Clearly, the 8-bit file type and the 32-bit file id cannot convey all of the information which- a groundstation needs to know about a file (e.g. time of creation, complete file name, file size). This information is in a structure at the beginning of each file. The file header may be of variable length, but would contain an offset to the first user data byte. The header is always at start of the file, with a byte offset of 0.

A proposal for a standard PACSAT File Header is provided in a separate document.

2.2 Binary Data

For greatest efficiency, all header information is coded in binary. This is a departure from previous amateur digital satellite philosophy.

Currently, UO-9, AO-10, UO-11, and AO-13 transmit telemetry and bulletins in uncompressed Ascii. This was, in part, to make this information available to the widest possible audience. The audience was a late 1970/early 1980 audience, and was assumed to have only the modem and a terminal, with little or no computer assistance. Telemetry on all of the above satellites can be read in Ascii and decoded with pencil and paper. The amount of telemetry generated is small, and the broadcast data is bi-weekly bulletins.

PACSATs are being designed for a 1990s environment. A new piece of hardware is required, either an external TNC, or an add-on card for your computer. In almost all cases, a computer is present at the ground station. The number of users who must deal directly with the raw data is reduced, perhaps to zero.

An increased emphasis on compressed data will lead to downlink which is mostly binary anyway. Several ground station programs are planned or already available which will allow users to monitor the downlink and acquire the data. The authors plan to make public domain source and shareware programs available for the IBM PC, with proceeds going to AMSAT-NA and AMSAT-UK.

3.0 PACSAT Broadcast - How it works in practice

PACSAT command stations would upload files tagged with a broadcast rotation priority and an expiration date. On-board programs would also be able to tag or build files with this data. An on-board broadcast administrator task maintains a list of active broadcast files and assigns a file-id. Files would be broadcast based on their rotation priority, i.e., files with low priority would be sent less often that those with high priority. Files under some threshold priority would only be sent during otherwise idle downlink time, those above the threshold would be mixed in with whatever else is on the downlink. Since parsing binary data (the header) in the standard monitor mode of TNCs is non-trivial, we can assume that the KISS mode, or

other host mode will be used to implement the ground program. Therefore the PID byte in the frame can easily be used to identify broadcast protocol frames. The exact value is Oxbb.

Ground users would run a program that monitors the downlink for frames. This program would probably use the KISS mode of the TNC. As each frame was received, the file_id and offset field of the frames are used to build up an image of the file. Duplicate frames are discarded. Several files can be active at once.

A utility program is provided to convert the received files into a usable form. It can decompress compressed files, and handle partial files if possible. It will display the contents of the PACSAT File Header record in received files. It can also generate a bole list or bit map, as required, to send to PACSAT and request retransmission of missing parts of the file.

4.0 Extensions

So far this discussion has assumed that all files on PACSAT can be divided into two groups: broadcast files which are sent with the broadcast protocol, and point-to-point files which are sent using AX.25 connected mode. There is also a gray area: point-to-multipoint files, which are not of interest to all PACSAT groundstations, but might be of interest to more than one station in the same PACSAT footprint. For instance, a message to all TCP/IP users would not warrant a continuous broadcast, but it might be downloaded by several groundstations. One would like to arrange things so that when the first groundstation downloads "TCPIP.UPD", other users in the broadcast-listen mode would also receive and avoid downloading it themselves (if they're interested). There are two ways of achieving this: by using the broadcast mode when the first groundstation requests the file, or by embedding broadcast mode datagrams in the connected-mode frames.

Using the first method, the user would connect to PACSAT and request that a file or files be put in the broadcast rotation. The user would then disconnect and use his broadcast-listen program to receive the desired files. (The user could even avoid connecting in the first place by transmitting the file request in a <UI> frame on the uplink.)

Files added to the broadcast list this way would be sent at a high priority for the length of an average pass.

Users who do not receive the file completely while it is being broadcast can send a description of the missed data (hole-list) to the PACSAT and those blocks would placed in the broadcast rotation. Retransmission requests for a particular file can come from many stations, but by the nature of the round-robin broadcast? requests are not likely to be synchronized and cause a mass uplink collision.

Thus, we use the broadcast mode for all but explicitly point-to-point files.

The alternative is for the PACSAT to connect to one station and commence standard AX.25 transfer, but embed enough information into the <I> frames to allow "eavesdropping" stations to build up copies of the file being transferred. Thus, to every <I> frame PACSAT adds the standard broadcast mode header, for the benefit of stations which may be listening in on the connection. Unfortunately, the connected mode user has no idea where each <I> frame begins and ends, so his software cannot strip out the unwanted header information. Thus, we set the L bit in the <flags> field and place frame length in the <length> field. Now the connected mode user knows where each frame begins and can strip out the header.

This may stick in the gullet of pure layered protocol designers. In defense of the proposal we offer the facts that (1) the broadcast <UI> frame listener already relies on knowing where the level-2 frames begin and end; and (2) we can look on our frame format as a higher-level packet format, and it just so happens that each packet fits in one frame (for the benefit of the "eavesdroppers").

5.0 Incremental Decompression

Incremental decompression is the ability to decompress partial files. In the most general case, any frame can be decompressed independently of any other frame.

It should be noted that the desire for incremental decompression has several deleterious effects. Primarily, it forces the use of non-optimal compression techniques.

Some compression schemes, such as Huffman coding, require that a table be sent along with the file that provides deciding information. The file is not decodable without this table, so a partial file that did not include the table would be useless. To allow use of Huffman-style coding, files would have to be compressed with a small number of standard tables that can be encoded in the file type.

Many compression methods, including Huffman coding, turn a file into tokens of variable bit length. That is, a token will not necessarily be an integral number of octets long, whereas an AX.25 frame must be an integral number of octets long. An arbitrary frame can be decoded only if the start of the frame is also the start of a token, and we know the number of meaningful bits in the frame.

Also, the program doing the broadcast framing may not be the program that did the compression. This would require, for instance, that PACSAT decompress the file as it broadcasts it, so it could properly align frames and tokens.

Decompression of partial files is an area for further discussion. To allow maximum flexibility in future implementation, the proposed standard includes the possibility of variable length frames. If the length bit in the flag byte is set, the two bytes following the standard frame header are the number of valid bits in the block. This allows for bit tokens of a non-integer number of octets.

It should be noted that the WO-18 image transmission format is a form of broadcast with incremental decompression, allowing even a single frame of the compressed image to be placed in it proper location on the display screen.

6.0 Broadcast Advantages

Receive-only stations with just an omni antenna and an eavesdrop program can accumulate bulletins, telemetry, and other items of general interest just by listening.

For transmit and receive stations, if one other station is interested in the same file, then overhead is reduced by at least 100%. This should justify the additional few percent overhead of the broadcast information.

7.0 Broadcast Disadvantages

Nearly everything broadcast from PACSAT would have several bytes of binary data in the front of each frame. This would make it difficult to monitor with just a TNC and terminal. Since the assumption is made that the data will also be compressed, a computer and proper program would be required in any case, so it is unlikely there will be any TNC-only stations. We also note that most of current fleet of spacecraft are already transmitting telemetry in raw binary form.

Due to the difficulty of parsing binary data from the standard monitor mode of TNCs, only TNCs with the KISS protocol, or other host-mode protocol, will be suitable. It may be possible, depending of the availability of true 8-bit transparency, to use the monitor mode *on* some TNCs, since a known, fixed string would precede each monitored frame, e.g., " > QST-1:".

8.0 Implementation Status

UO-14 has been broadcasting files using the protocol defined in Appendix A. since mid summer. AO-16 and LO-19 should begin using this format by the end of August.

9.0 Proposed Broadcast Protocol

The protocol specification is provided as Appendix A to this paper.

10.0 Correspondence

Address comments to:

Telemail:	HPRICE or UOSAT
Compuserve:	71635,1174
Packet:	NK6K @ WB6YMH
	or G0K8KA @ GB2UP
Internet:	71635.1174
	@COMPUSERVE.COM
Mail:	Jeff Ward
	UoSAT Unit
	University of Surrey
	Guildford, Surrey GU2 5XH
	UK

Appendix A.

A.1 PACSAT Broadcast Protocol Description

This protocol describes a method where files can be sent from a PACSAT to an unknown number of ground-stations, using the unconnected < UI > frame **mode** of AX.25. The protocol could also **be** used in purely terrestrial applications.

Each file that is to be broadcast is assigned a 32-bit file id. This number must be unique among all other files **broadcast by this station.** A **station** must be able to later match the file id with the file if the station is to handle retransmission requests.

When a file is broadcast, it **is** broken down into frames. Each frame is identified with enough information so that the data can be placed in the **proper** location and in the proper **file** by a receive-only groundstation. Each frame is sent within an AX.25 < UI > frame, and is totally contained within that < **UI** > frame.

Some files **are** automatically retransmitted enough times over several passes so that the likelihood of a station receiving all parts of the file at least once with no errors is high. A facility is also provided to allow a ground station to request the transmission of certain frames to allow fills of missing data.

The PACSAT Broadcast Protocol has two elements, the File Transmission Frame, use to send data; and the **File** Request Frame, used to request retransmission of all or part of a file.

A.2 File Transmission frame format

A.2.1 Frame

A broadcast frame consists of a header, data and a CRC . The header is variable length, depending on bits in the flag byte. The data may also be of variable length.

The broadcast frame is wholly contained within a standard AX.25 < UI > frame. The total length of the broadcast frame may not exceed the length of a legal AX.25 c UI > frame.

A < UI > frame containing a broadcast frame uses a PID of Oxbb.

A < UI > frame containing a broadcast frame uses a source address of the transmitting station, and a destination address of QST-1.

A.2.1.1 Frame Header

Each frame contains a frame header, which occupies the first n bytes of the frame. The structure of the frame header is as follows:

< flags > < file id> <file type> < offset > [< length >]

This structure follows the PACSAT Data Specification Standards as regards field size and byte order, which is defined in a separate document.

```
struct FRAME_HEADER {
    unsigned char flags;
    unsigned long file_id:
    unsigned char file_type;
    unsigned int offset:
```

```
    unsigned char offset-msb;
    };
```

or

```
struct FRAME_HEADER_EXT {
    unsigned char flags;
    unsigned long file-id;
    unsigned char file-type;
    unsigned int offset;
    unsigned char offset msb;
    unsigned int length;-          /* only present if L bit set */
    };
```

flags A bit field as follows:

7	6	3	4	3	2	1
*	*	0	V	V	0	L

L 1 length field is present
0 length field not present

E 1 Last byte of frame is the last byte of the file.
0 Not last.

VV Two bit version identifier. This version is 0.

0 Always 0.

* Reserved, must be 0.

file id A number which identifies an active broadcast file. All frames which are part of the same file are tagged with this number.

file type The file_type byte from the file header. Provided so that partial files received without the header can be decoded. File types are defined in a separate document.

length If the L bit is set, this field is present it in the beader. It is the number of bits that are to be used in the data field. This field has two intended uses: when variable length blocks are used with the broadcast carried inside a higher level protocol (so that the frame length is lost)? and when a non-integer number of octets of data are used.

offset If the 0 bit is not set, this is the block number of the block. If the 0 bit is set, this is the offset from the start of the file for the first byte in this frame. This field is the lower 16 bits of a 24 bit integer.

offset-msb The high order 8 bits of the offset.

A.2.1.2 Data

If the block mode is used, the length of the data must be fixed. It may be any lengtb, but the following must be true:

file offset_of first_byte_in_frame = offset * this frame size

All frames in a particular transmission of a file should be the same length, to avoid having the receiver resize his bit map too often.

If byte numbers rather than block numbers are used. the data may be any length. If the length field is present, the data must contain only that number of bits, rounded up to the next octet boundary.

A.2.1.3 CRC

A two-byte CRC as defined by the **XMODEM** protocol follows the data. The crc covers all data bytes in the AX.25 UI frame, including the broadcast frame header.

An inefficient definition routine for checking the **XMODEM CRC** when receiving a frame is:

```
unsigned sbort crc;              /* global crc register.              */
crc = 0;                         /* clear crc register                */
for (all received data bytes)
    gen_crc(rx_byte);            /* crc the incoming data bytes       */
gen_crc(1st_crc_byte);           /* crc the incoming first crc byte   */
if(gen_crc(2nd_crc_byte))        /* crc the incoming second crc byte  */
 bad-crc ( );                    /* non-zero crc reg. is an error     */
else
 good-crc ( );                   /* zero crc is a good packet.        */
/* Function to take a byte and run it into the XMODEM crc generator.  */
/* Uses a global CRC register called ''crc''.                         */
gen_crc(data)
char data;
{
    int y;
    crc ^= data << 8;
    for(y=0; y < 8; y++)
        if( crc & 0x8000)
            crc = crc << 1 ^ 0x1021;
        else
            crc <<= 1;
    return crc;
}
```

Further information on generation of **XMODEM CRC** via more efficient byte-wise methods can be found in BYTE Magazine, September 1986, pp. 115-124.

A.3 File Retransmission Request Format

Each request to retransmit portions of a file is sent in a retransmission request frame. If more data is being requested than will fit in one request, multiple requests may be sent.

This protocol is intended to service retransmissions of already-broadcast files. The manner is which a file is first requested to be broadcast is a more complex topic, including, presumably, the ability to request files based on name, date, content, keywords, mail message copy list, etc.

A.3.1 Request Frame

A request frame consists of a header and data. The header is fixed length. The data depends on the type field in the header, and can be of variable length.

The request frame is wholly contained within a standard AX.25 <UI> frame. The total length of the broadcast frame may not exceed the length of a legal AX.25 < UI > frame.

A <UI> frame containing a request frame uses the same PID as a broadcast frame, Oxbb. The source address is the address of the transmitting station, the destination address is the address of the station to which the request is directed. Thus, if a frame has the PACSAT Broadcast Protocol PID , if the destination is QST-1, it is a broadcast frame, otherwise it is a request frame.

A.3.2 Request Format

A.3.2.1 Header

Tbe format of request is:

<type><file_id><data>

```
struct REQUEST-HEADER{
        char flags;
        unsigned long file-id;
        int block-size;
        };
```

flags - A bit field as follows:

7	6	5	4	3	2	1
*	*	1	V	V	C	C

CC Two bit field as follows:
- 00 start sending file <file_id >
- 01 stop sending file < file_id >
- 10 Frame contains a hole list.

VV Two bit version identifier. This version is 0.

1 Always 1.

* Reserved, must be 0.

block size Requests that the broadcast use this value as a maximum size.

file id File id of the requested file.

A.3.2.1 Data

If the CC field is 2, a hole list is present.

The format of the hole list is pairs of start addresses and lengths.

```
struct PAIR (
    unsigned int offset:
    unsigned char offset msb;
    unsigned int length:-
    );
```

The length of the hole list is determined by received frame size.

A.4 PROCEDURES

A.4.1 Identification of a broadcast protocol data frame.

The broadcast protocol will use a special PID Oxbb. Broadcast frames with be sent with a source callsign of the station's call, and a destination callsign of < QST-1 > .

A.4.2 File transmission

Multiple broadcast files may be transmitted simultaneously, frames from one file can be interleaved with another. Frames can be transmitted in any order and repeated at any time.

A.4.3 File completion

The file header contains the total number of bytes in the file. Once the ground station has received that number of bytes. the file is complete. The file header also contains checksums which can be used to verify the correct reception of the file.

Appendix B.

B. Legalities Within the Amateur Satellite Service

The use of the words "broadcast" and "compression" sometimes raise the hackles of certain members of the amateur community, "Broadcast" is mentioned in the Amateur Rules (Part 97 of the FCC regulations), and compression is sometimes equated with encryption.

B.1 Broadcast

To establish the bona-rides of a broadcast protocol:

97.113(d)(2) specifically permits information bulletins consisting solely of subject matter relating to amateur radio.

The prohibited items are communications intended to be received directly by the public (1200/4800/9600 baud PSK HDLC should take care of that), or for newsgathering for broadcast purposes.

B.2 Compression

97.117 specifically permits the use of abbreviations or signals where the intent is not to obscure the meaning but only to facilitate communications. Compression using published algorithms to increase data throughput would thus be permitted.