Extended sequence number (modulo-128) option for AX.25

Half-duplex packet radio links are operated at ever higher bitrates. When observing the efficiency of a half-duplex link, it is immediately obvious that a percentage of the maximum throughput is lost due to delays when changing the direction of the traffic. A transmitter needs a finite time to key-up, and the receiver needs time to lock on the received signal and provide stable data. At the end of a transmission, some extra flags are usually sent to overcome a difficulty in the commonmy used SCC chip, and to clear a scrambler that may be present in the path. When the link would be via satellite, propagation delay would be an extra factor.

To operate at reasonable efficiency, it is best to send at least so much data in each transmission that the changeover delay is less than, say, 10% of the transmit time. This is normally equivalent to transmitting about 1..3 seconds.

On a link between nodes operating using the NET/ROM protocol, there is only a single AX.25 connection that handles all the traffic. On its queue are both the user-data packets and the transport layer acknowledge- ments, in addition to the transport layer connection setup packets. In practice, this means there are often quite a number of small packets queued on the connection.

To get the best effciency (on a link with a reasonable bit-error rate), one would like to send many of these packets in a single transmission. For example, when a number of 50-byte packets is queued and the target transmit time is 1 second, one would need to send about 24 packets at 9600 bps.

However, the current AX.25 protocol uses only 3-bit sequence number fields, and therefore no more than 7 frames can be sent in a single transmission. This would be only a 300ms transmission in this case, which is often quite short when compared to the changeover delay.

To overcome this limitation, I have added the option of using extended (modulo-128) sequence numbers to the AX.25 handling in NETCHL over two years ago. It has operated very satisfactorily over that period, and this document describes how it was done, so that other software writers who desire to to the same can do it in a compatible way.

Protocol modifications

Basically, there is not much to it. The AX.25 protocol is based on HDLC, and in the HDLC standard there already exists the modulo-128 sequence number option. This essentially is what is being used.

A extended sequence connection is started using SABME (0x6f) instead of SABM (0x2f). When the connected station answers the SABME with an UA, the connection is established and will use modulo-128 sequence numbers. This results in a change in the control field in each I and S frame, frames that include a sequence number. The control field becomes 16 bits long in these frames. U frames, including SABM, SABME, UA, DM, and UI are sent with the normal 8-bit control field.

(the bytes are shown MSB-to-the-left here)

modulo-8:

N(r)			P/F	N(s)			0	(I-frame)

N(r)			P/F			0	1	(S-frame)

			P/F			1	1	(U-frame)

modulo-128:

N(s)							0	(I-frame)
N(r)							P/F	

0	0	0	0			0	1	(S-frame)
N(r)							P/F	

			P/F			1	1	(U-frame)

In the FRMR frame there is also a change in the data field of the frame:

modulo-8:

Control fld							
N(r)			C/R	N(s)			0
0	0	0	0	z	y	x	w

modulo-128:

Control fld byte 1							
Control fld byte 2							
N(s)							0
N(r)							C/R
0	0	0	0	z	y	x	w

When the rejected frame is a U-frame, the control field is put in the first byte of the FRMR frame, and the second byte is set to zero.

Monitoring

As the type of the connection (modulo-8 or modulo-128) is determined only at the start, by sending SABME instead of SABM, it is difficult for monitoring stations to know how to decode a received frame. Therefore, bit number 6 in the SSID field of the SOURCE call in the AX.25 header is cleared to indicate that the frame belongs to a modulo-128 connection. This is normally a reserved bit that should be set to 1.

The value of bit 6 is not used by the stations participating in the connection, but is only intended as information for decoding by a station tracing (monitoring) the connection.

C/R	6	5	SSID				E

bit 5 is cleared for DAMA master

bit 6 is cleared for modulo-128 frame

Implementation

Implementation of the extended sequence number (modulo-128) option should not be difficult when the existing AX.25 handler is reasonably well written.

- For each connection, an extra flag is required to indicate that modulo-128 sequence numbering is in use. The flag is set when SABME (0x6f) was received to setup the connection instead of SABM (0x2f).

- To setup a connection, SABME is sent when modulo-128 is desired, and the modulo-128 flag is set as well.

- The extraction of N(r), N(s) and P/F from the control fields of I and S frames must be made dependent on the setting of the modulo-128 flag.

- When constructing an I or S frame, the format must be selected depending on the modulo-128 flag.

- When incrementing a sequence number, the correct modulo must be selected depending on the flag.

- When constructing or interpreting a FRMR frame, the format of the frame is selected depending on the flag.

- Monitor (trace) code should be adapted to trace extended sequence number frames when bit 6 of the source SSID field is zero.

- A separate MAXFRAME value for modulo-128 connections ('EMAXFRAME') could be a settable parameter, when two types of connections are allowed on the same interface (port).

- The worst-case frame length (256-byte frame sent via 8 digipeaters) is one byte more than in modulo-8 mode. This will usually be no problem, as modulo-128 is most likely to be used on point-to-point links. For a really complete implementation, it may be that the buffer size has to be increased.

In (NETCHL) practice it has turned out to be convenient to use a variable in the connection control block, mmask, which is set to 7 for modulo-8 connections and to 127 for modulo-128. This variable is used as the "modulo-128" flag described above, and also as the "modulo mask" which is applied (ANDed) after all sequence number arithmetic.

Resequencing

It is obvious that, when sending so many frames in a single transmission, any lost frame has a large impact on throughput. In the standard AX.25 protocol, all frames following the missing frame have to be discarded and re-sent after the lost frame has been re-sent and successfully received.

It has been shown before that a resequencing queue ("framesammler") can be used to save the out-of-sequence frames, and use them after the lost frame has been retransmitted. Unfortuately, special tricks (checksumming the frame) were required to solve the ambiguity that results from the small sequence number range in standard AX.25.

With extended sequence connections, resequencing is possible without these tricks, when the EMAXFRAME (maximum number of unacknowledged frames) is kept below half the modulo. Therefore it is hereby specified that the value set for EMAXFRAME should be limited to 63. Then, for each incoming frame it can be unambiguously determined if it is a re-transmitted old frame or a frame which lies beyond the expected sequence number, and could be saved in a resequencing queue.

Resequencing can work without further extending the protocol (adding a Selective REJect frame type), using the following simple rules:

- when a frame with a sequence number beyond the expected number is received, it is saved on a resequencing queue for the connection. (unless it is already there, or memory is low)

- each time a frame has been received that matches the expected sequence number, the resequencing queue is examined to see if it contains one or more frames that fit in next

- when a reply is to be sent to report the next expected sequence number, the following type of frame is sent:

  RNR when the input queue is too long

  REJ when frames are present on the resequencing queue

  RR otherwise

- when a REJ reply is received, the next transmission will be only a single frame, namely the next expected frame at the other end.

This procedure is not as efficient as SREJ would be, but it is in successfull operation in a number of programs that implement resequencing. It results in a short transmission to recover from loss of a frame, but does not waste time retransmitting information the other end already has.

Compatability

NET implements compatability with old code using the following algorithm: For each connection configured to be in modulo-128 mode, SABME is sent first. When the response is DM or FRMR, a fallback is done to modulo-8 mode and SABM is sent instead.

Unfortunately it turns out that not all AX.25 implementations send DM or FRMR back when they receive SABME (not even with Poll bit set). This can be regarded as a violation of the protocol, but on the other hand sending an SABME could be regarded as a protocol violation as well, so... The solution has been to have a table of callsigns that are known to handle modulo-128 connections, and only attempt such a connection when the destination station is in the table.

On NET/ROM interlinks, which is where it is most useful, I think it should be sufficient to have a single configuration bit per interface (port) enabling use of modulo-128.

It is also no problem to implement it on incoming connections on local access interfaces. When it is desired to have modulo-128 operation on outgoing connects, on interfaces where many stations are likely to be present, a table of callsigns is required for compatability.

Rob PE1CHL - Mar 4, 1995