Purpose

This paper is presented to document some of the major changes proposed over the last four years to the AX.25 draft. These changes have been collected by this author from various sources, and were recognized by the ARRL Digital Committee which met in July of 1988.

Background

The Amateur Radio Link Layer (ARLL) Standard was created by Douglas Lockhart, VE7APU. Since this author had already written several of the proposed ideas in the ARRL Newsletter, a draft of the new AX.25 Level 2 was begun.

This draft was almost completed when Tom Clark, then President of AMSAT, called a meeting in October of 1982 to establish a standard Link Layer Protocol to be used on AMSAT Satellites. At that meeting, several protocols were discussed, with the result being that a slightly modified version of the AX.25 Level 2 Draft-beeing adopted. The major alteration was to include fields for more than one digipeater. This draft was then used by the TNC-1 Terminal Node Controllers. The AX.25 draft quickly became the standard for packet radio due to its implementation by TAPR. It should be noted that the TNC-1 was capable of both AX.25 and the original Vancouver Protocol. The AX.25 Level 2 Protocol Specification was published in March of 1983 in the Second ARRL Amateur Radio Computer Networking Conference proceedings by the author.

The AX.25 Level 2 protocol was heavily based on the commercial X.25 Protocol Specification, with some revisions. One of these revisions was the removal of certain types of frames (S-Frames) as commands for link status and maintenance. Instead, Information Frames were used, along with a heavier emphasis on timers. This was done to simplify the protocol implementation, but as it turned out, this short-cut caused more problems than it relieved.

It became apparent that some of the changes, such as the removal of command S-Frames, were not working as well as had been expected. About the same time the ARRL and its Digital Committee expressed a interest to officially adopt the AX.25 Level 2 protocol. These factors led to the creation of a new version of the protocol, which more closely adhered to both the CCITT X.25 and ARRL X.25 standards. The original AX.25 Level 2 was expanded and refined to become AX.25 Version 2.0, which was adopted by the ARRL Digital Committee and the ARRL Board in October of 1984.

The original AX.25 was labeled Version 1.0 to indicate its operational differences.

Version 2.0 of AX.25 Level 2 has been in use for four years now, with estimates of between 50 and 80 thousand devices using it. During this time, some additional problems with it have been encountered. The remainder of this paper discusses alterations to the AX.25 Level 2 Version 2.0 Protocol Specification.

Backward Compatibility Issue

As just mentioned, there are an estimated 50 to 80 THOUSAND devices using the AX.25 Level 2 Version 2.0 Protocol. A fundamental issue that must be resolved is what happens to those users when alterations to the protocol are made. In Amateur Radio, there is no method of requiring this installed user base to migrate to a new protocol quickly. If changes are made that cause incompatibility between users, complaints are sure to follow. An example of this is found with the TAPR TNC-1. Due to an oversight in the software design, TNC-1s cannot be used with Version 2.0, even as digipeaters. This is because the TNC-1 code accidentally tests bits that were clearly marked as reserved in the protocol specification. If some users feel they are being forced to migrate to a new protocol that is not backwards-compatible, they may argue, the newer version causes enough on-air problems and crashes, the rest of the community must follow just to stay compatible, and that method of subterfuge, they argue, changes for the better should be made automatically without regard to on-air compatibility. In view of the large number of devices using Version 2.0, this author agrees that backward-compatible solutions should be implemented whenever possible. The guideline should be first and foremost. If a solution cannot be reached which is backward-compatible, it should be flagged and carefully weighed before implementation, since this author's recommendation that minor revisions are noted by the number after the period, while major alterations are noted by the number preceding the period.

Addressing; e.s.

One of the prime motivating forces behind the move to update AX.25 Level 2 is its limitations in the area of addressing. When this author originally proposed the use of an extended address field that included Amateur Radio callsigns, it was felt that six characters would be sufficient to contain the callsign, with a seventh tacked on to allow each Amateur to maintain more than one station. It now appears that both prefixes and suffixes should be sent in addition to the callsign. These additions are often that fit within the six octets allowed under Version 2.0.
Further, since Version 2.0 specifically calls for sub-fields of six characters plus SSID, extending the individual callSIGN address sub-fields is precluded.

After much individual research and the suggestions of many others, two methods of allowing adding the additional information have been recommended. The first method has been designed to be backward-compatible, but is rather inelegant. The second method is designed to cure regulatory-related problems outside the United States in addition to the above issue, but is not backward-compatible.

**Backward-Compatible Addressing Extension**

The first method of adding more addressing information to AX.25 Level 2 Version 2.1 is very much a kludge. It will be immediately obvious to older equipment exactly what is going on, but it will function properly with the older versions, assuming they followed the previous protocol standard, unlike the TNC-1 as described above. This unwitting inter-operation with the older standard makes it backward-compatible with them.

Simply put, method one defines additional address sub-fields called extension sub-fields which, if present, convey the additional addressing information required. They are placed after the destination and source addresses, but before any repeater address sub-fields. As with the other address sub-fields, these extension sub-fields must be seven bytes long. Both the placement and encoding of the bit in the extension sub-fields are a subterfuge to imply to an older version device that the extensions are digital repeater addresses, allowing the older version to ignore the extensions presence. Since Version 2.0 allows at most eight digital repeaters, any extension sub-fields must be subtracted from the number of allowed digital repeaters to keep the maximum number of repeater sub-fields at eight or less.

**Address Extension Information Encoding**

The additional information is encoded in the same manner as the other address information. It should be bit-shifted ASCII, stuffed with trailing ASCII spaces as required to six characters plus SSID. The only other requirement is that if the additional information is a prefix to a callsign, the slash (/) character is placed after the prefix. If the additional information is at the end of the callsign, the slash is placed before the postfix. For example:

<table>
<thead>
<tr>
<th>Amateur</th>
<th>Normal Callsign Address Field 1 Field 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>WB4JF1/KT-1</td>
<td>WB4JF11 /KT (none)</td>
</tr>
<tr>
<td>VPP2/VB4JF1</td>
<td>VPP2/VB4JF1 (none)</td>
</tr>
</tbody>
</table>

As implied above, more than one extension may be required, and may be used. If both pre- and post-fixes are required and are under six bytes in total length (included a shared slash), they may be combined in a single extension field.

The SSID number of the first address extension sub-field for each address (callsign) will be set to zero, the second to one, and so on as necessary. Not only will this aid in “gluing” the address back together, but will also indicate when one extension block ends and another begins.

The equivalent of the H-bit (bit 7) in the SSID octet of all extension sub-fields should be set to one at all times by a Version 2.1 device, indicating to a previous version device that this is a repeater field that has been repeated. This will allow a previous version device which is a destination to conclude that it is repeaters (including the extensions) have repeated the received frame, and may process the frame.

**Extension Information Indication**

Any address that has extension information will indicate this by resetting bit 6 (hereafter called the A-bit, for Address extension) of its SSID octet to zero (0). This bit has previously been reserved and should have been set to one (1) as indicated by both Versions 1.0 and 2.0 of the AX.25 protocol specification.

The extension sub-address fields should also have the A-bit set to zero to simplify the comparison of extension sub-fields with repeater sub-fields in Version 2.1 devices.

**Extension Information Placement**

In order to fake-out earlier versions of the protocol, the extension information cannot simply follow the base address. The only place this new information can be placed which will work with older versions is between the source address and the first of any repeater addresses. If placed there with the H-bit set to one, older versions may assume the field is for a repeater that has already repeated the frame.

The order of appearance of these extension sub-fields is the same as the main address sub-fields; any destination extensions come first, followed by any source extensions, then any repeater extensions in the same order as the repeater sub-fields themselves.

**Examples of Address Extension**

The following example will aid in indicating the proper operation of the proposed address extension recommendation.

The following example indicates how a single address sub-field may be extended. In it, the destination field has a post-fix modifier. The frame instead assumes the base address WB4JFI1 to K8MMO/KT-0. Note how the A-bit is set to zero both in the destination sub-field AND the address extension sub-field. Note also how the address extension sub-field has an SSID of 0000 indicating it is the first (and in this case only) extension sub-field.

```
! DA ! SA ! Ext. 1 ! CTL ! K8MMO ! WB4JFI1 /KT 0 ! UI !
```

The actual bit-encoding of these fields is as follows:

<p>| | | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>A1</td>
<td>K</td>
<td>1001</td>
<td>0110</td>
<td>96</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>B2</td>
<td>A</td>
<td>0011</td>
<td>0000</td>
<td>70</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C3</td>
<td>A3</td>
<td>1001</td>
<td>1010</td>
<td>9A</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>D4</td>
<td>A4</td>
<td>1001</td>
<td>1010</td>
<td>9A</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>E5</td>
<td>A5</td>
<td>0100</td>
<td>0000</td>
<td>40</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>F6</td>
<td>A6</td>
<td>0100</td>
<td>0000</td>
<td>40</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>G7</td>
<td>A7</td>
<td>1010</td>
<td>0010</td>
<td>50</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>H8</td>
<td>A8</td>
<td>1010</td>
<td>1110</td>
<td>AE</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>I9</td>
<td>A9</td>
<td>0100</td>
<td>0000</td>
<td>40</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>J10</td>
<td>A10</td>
<td>0100</td>
<td>0000</td>
<td>40</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>K11</td>
<td>A11</td>
<td>1001</td>
<td>0100</td>
<td>9A</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>L12</td>
<td>A12</td>
<td>0100</td>
<td>0000</td>
<td>40</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>M13</td>
<td>A13</td>
<td>0100</td>
<td>0100</td>
<td>9A</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>N14</td>
<td>A14</td>
<td>0100</td>
<td>0100</td>
<td>9A</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>O15</td>
<td>A15</td>
<td>0100</td>
<td>0110</td>
<td>5E</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>P16</td>
<td>A16</td>
<td>1001</td>
<td>0110</td>
<td>96</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Q17</td>
<td>A17</td>
<td>1010</td>
<td>0000</td>
<td>40</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>R18</td>
<td>A18</td>
<td>0100</td>
<td>0000</td>
<td>40</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>S19</td>
<td>A19</td>
<td>0100</td>
<td>0000</td>
<td>40</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>T20</td>
<td>A20</td>
<td>0100</td>
<td>0000</td>
<td>40</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>U21</td>
<td>A21</td>
<td>1010</td>
<td>0001</td>
<td>A1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>V22</td>
<td></td>
<td>0000</td>
<td>0011</td>
<td>03</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

If there are real repeater fields in addition to the extension information, the address field will look as follows:

```
Uplink to the Destination Station
! DA ! SA ! Ext. 1 ! CTL ! K8MMO ! WB4JFI1 /KT 0 ! UI !
```

```
WB4JFI1 ! K8MMO ! WB4JFI1 /KT 0 ! UI !
```

Link Back to the Original Station
If the link-back frame is passed through older versions of the Protocol Analyzer, the H-bit in the Extension sub-field and A-bits in both fields may not be set to indicate an extension is present. For example, the older device to receive the frame knows there may be extension information which may look like repeater sub-fields. Relating to the address information, the sender of the frame's repeater addresses (or count thereof) and the number of H-bits set compared to repeater fields used, to find out if the protocol analyzer knows there may be an extension. The main disadvantage of this new addressing scheme is that it totally alters the addressing of AX.25, rendering it absolutely incompatible with all earlier versions. This renders it from the AX.25 protocol, at least on the channel it will be used on, non-use of the E-bit means, among other things, that Protocol Analyzers can no longer be used to troubleshoot and analyze links and software implementations.

The second method is the subject of a separate paper found elsewhere in these proceedings and will not be discussed in depth here. It involves separating the addressing issues from the rest of the protocol, and totally redefining the address protocol, at least on the channel it will be used on. Will both versions be required to be available simultaneously? Time will tell.

State Description Logic (SDL) Diagrams

In the back of the AX.25 Level 2 Version 2.0 Document are three state tables, which are meant to describe the operation of the protocol based on various external actions taking place. There has been some concern as to whether these tables are actually part of the document or just an implementation guide. They were included to easily indicate to implementers how the protocol should operate, and therefore are part of the protocol description.

One of the problems with these state tables are that they cannot easily indicate the various steps taken when an external event happens. There is only enough room to indicate if a frame is to be sent as a result of the event, and any state transition made as a result of that event. If there is more to be learned, which is the case with AX.25 Version 2.0. This is meant to simplify software implementation of the protocol.

Another advantage is the addition of a hashed version of the next destination address as the first address field. This allows the implementation of selective addressing which is built into most modern synchronous hardware chips, reducing memory overhead.

In addition, Address sub-fields may be of variable length, and may include mnemonics in addition to Amateur call signs.
the AX.25 Level 2 Version 2.0/2.1 in SDL diagrams. If they are available in time, the next printing of the AX.25 Level 2 Document may include the SDL diagrams in place of the State Tables. The reason for any delay is that the SDL diagrams are quite a bit more complicated, and must be checked very carefully with the text to make sure they agree with each other.

**State Table Changes**

Given the preceding information, State Table changes may be a moot point. They are included here for completeness.

The main State Table change is the removal of States 14 and 16. After review, no one has ever found how a protocol machine could remain in either of these states for any length of time. Both have to do with the local station being busy, but having sent a REJ frame. The sending of the REJ frame itself indicates a non-busy condition by requesting a re-transmission.

**Unnumbered Information Operation**

There has been some discussion regarding the proper use of the Unnumbered Information, or UI frame. This is especially true when discussing the use of the Poll bit associated with UI frames. Some feel that it is possible to maintain a separate "UI Mini-State-Machine" by the use of Polls and that this use associates ONLY with UI frames. After careful review, it was deemed that in order for these P/F bits not to interfere with the normal protocol machine type operation, more processing overhead would be required than people believed. Therefore, UI frames are left as Commands, with no Poll bits used.

**Automatic Re-connect Elimination**

Perhaps the largest complaint heard regarding AX.25 Level 2 performance is when a "disconnected" connection keeps coming back. There are cases when a station has requested a disconnect, gone into timer recovery due to the disc frame getting lost, and then perceives I frames from the remote station. Eventually the first station will consider itself disconnected, and send DM frames. The second station will tell the data pending, so it will re-establish the connection and pass the data. This is NOT a bug, rather it is a deliberate attempt to make AX.25 re-establish failed links. For our shared RF medium, it now appears this tactic is too aggressive. The new recommendation will specify that the link is not re-established, instead an error message is passed to the higher layer protocol or program, which will decide what action the AX.25 Level 2 machine should take.

**Maximum Packet Size**

There have been many discussions regarding the maximum transmittable frame size. Some people feel the 256 byte limit of data in Information frames is too small. They site two reasons for an increase in the maximum transmittable data per frame. The first reason is that some higher-layer protocols require substantial overhead (such as TCP/IP), and this overhead must be added to the real user data, subtracting from the total transmittable user data per frame. The other reason is that higher speed channels should be able to transmit larger amounts of data per frame, increasing the overhead-to-user-data ratio, thereby using the channel more efficiently. Information field sizes of 1024, 2048, or larger have been suggested.

Making an ad-hoc change to the maximum frame size may have serious repercussions, however. Older devices that have hard limits to the maximum allowable frame size, reducing the chance of backward compatibility. An even greater potential problem is that implementations do not check for an allowable maximum frame size and crash if substantially large frames are received. Older devices may have had limits set to the maximum allowable frame size, and simply larger frames might also tie up too much memory, another potential for crashing implementations.

In order to make sure older equipment doesn't crash due to having limited allowed frame sizes, there needs to be more study done on this issue. Otherwise, drastic results may occur.

The result of discussions within the Digital Committee is an agreement that the 256 byte limit be maintained at this time, with an escape clause that Source, Destination, and Intermediate repeaters may use agreed-upon larger frame sizes on particular link connections and channels. Implementers should be aware of this, and make sure maximum allowable frame size is checked.

**Maximum Window Size of One**

Phil Karn has been suggesting that present AX.25 connections be limited to a single frame per RF transmission are actually using the channel inefficiently. He feels there should be only one I frame outstanding at a time, and that the direction per link, creating his Ack-Ack protocol. Even if there is some accuracy in his argument, altering the operation of the re-transmission would have to be the 2.0 scheme for now. It should be noted that this system follows the traditional X.25 approach.

**Forced Disconnect**

Franklin Antonio has raised an issue regarding the Disconnect Request state, S4. Presently while in this state, a Local Stop Command is received (from the higher-layer), no action is taken. He suggests a transition to the Disconnected state, S1, and the discontinuation of sending Disc frames. This appears to be reasonable behavior, and was recommended.

**Stop Timers During Channel Activity**

The main reason for the use of timers during AX.25 Level 2 links is to make sure the link is still valid during slack transmission periods and for error recovery. In half-duplex Amateur Radio use, errors are normally introduced whenever two or more stations transmit simultaneously, interfering with each other. The timers are presently run whether the channel is busy or not. Some Amateurs argue that they are a waste of valuable time on an efficiently used channel.

Forging a decision here is hard. Would the lost frame be the Ack frame, which could by itself cause the timer stoppage, or the I frame, which could also stop the timer? Some results may be needed before a decision is made one way or the other.

**Ack Prioritizing**

Another suggestion that has been made is to make sure that acknowledgments have a higher priority than other frames during connections. It is suggested that this will cause fewer retransmissions, since the shorter acks can be clobbered by longer data frames, causing error recovery procedures to be hastily implemented.

After reviewing the various requirements to implement this, it was found that this would cause the timers to be involved, it was felt that implementing Ack Prioritization could become extremely complicated. More study is needed before considering this subject, as it may still have advantages.

**Remote AX.25 Level 2 Parameter Control**

There have been a few Amateurs that have indicated a need to be able to remotely access and possibly alter various Level 2 parameters. This has been met with quite a bit of resistance, primarily because of the potential for link damage by others, either accidentally or on purpose. It remote parameter setting is to be performed, it
should be done within the confines of a higher protocol, preferably with some authentication.

Implementation Issues

There have been several questions asked regarding how to properly implement AX.25 Level 2. Sometimes, some "bugs" are introduced on the air due to its improper implementations. After discussion of a few of these implementation problems, it was decided to include an Appendix to the AX.25 Level 2 document discussing a few of these issues. In addition, the inclusion of the SDL diagrams discussed above should help resolve future questions. A few of the implementation problems are discussed below.

Queued Text When $\text{SI}$ or $\text{S_4}$ Entered

Franklin Antonio points out that some implementations (particularly TNC-1 and TNC-2) send any queued text left upon disconnection as Information in UI frames AFTER the disconnection. This should NOT be done. Either the link should be re-established, or the information should be discarded (the decision on which action to take is now left to the upper-layer). This is an implementation error.

PNR and Memory Usage Problems

Some Amateur indicate that there is a problem with some devices which use RAM memory to store, or log, received data or messages. After a certain amount of operation, these devices can become full, even if a disconnection is established while one of these devices is already full, that device will allow the connection, but the indicated space by sensing of frames until the user frees some memory. This should not happen. If a device cannot support a connection, it should reject the connection request (with a DM response). This is a clear implementation issue that is beyond the protocol document. The document is meant to describe how the protocol machine operates, not all possible implementation issues, such as mail storage in the same device as the protocol machine.

Bells, Clear Screens, Other Binary Data

A problem that is becoming more an issue every day is that of binary data being sent to the user terminal from the TNC device. Quite often, a user who is monitoring a packet channel suddenly hears bleeps, the screen clears, and other strange things start happening. This is often because more real network protocols are showing up on the air. These network protocols use binary data in their control fields (which are located in the information field of the AX.25 Level 2 frame), which can cause a terminal to go crazy if it receives the binary data.

When the AX.25 Level 2 protocol was designed, a Protocol Identifier (PID) was added to indicate what type of higher level protocol if any was being used. At that time, a PID of $0x$ was not issued to be used whenever no upper layer protocol was being used. Since then, additional PIDs have been assigned to Network Layer protocols. At the time it was hoped that if a device (or software between the user and the device) saw a PID other than $0x$, that device could selectively not allow the data to the terminal (or computer screen). Since this was not clearly stated as a purpose in the protocol document, it appears that was not implemented. Future software designers should allow this option, reducing the number of problems showing up on the user screen. This will become even more important as more network protocols show up on the air.

Level 1 and Level 1/2 Interface Issues

Many of the proposed changes received are not directly related to AX.25 Level 2 operation. Specifically, adjustment to L2 timers and channel access operations should be considered outside the scope of the AX.25 Level 2 protocol, since different channel access protocols require different settings and adjustments. There needs to be more work done in this area possibly as a separate AX.25 Level 1 document. The following is a list of ideas and suggestions that should be considered for a Level 1 document. As an addendum or an internal solution, an addendum to the AX.25 Level 2 document, or a separate working paper could be created to include the suggested operations. Another reason for this separation is that most of these adjustments are to fine-tune half-duplex CSMA or SMMCD operation. In full-duplex, slotting, or other channel access methods, these parameters do not necessarily come into play, and do not need to be specified or adjusted.

If there is separate document created for these Level 1 issues, wording should be added to the AX.25 Level 2 document indicating which parameters may be altered as a result of another Level's operation.

Persistence

Back in the old days of Vancouver boards running either V1 or AX.25, the computer was a simple case of automatically and aggressively retrying, but would wait before retransmitting. Unfortunately, the TNC-2 software did not implement this, and as a result there has been a beating-of-the-chest regarding schemes to take care of this "sudden" problem. Recently, Phil Karn and others have rediscovered this situation and has suggested we modify the Level 2 specification to include persistence.

Adding requirements to the AX.25 Level 2 specification regarding persistence would be a mistake. We will not APHAS be using half-duplex CSMA as the channel access protocol.

In our half-duplex environments, persistence should be used, with the value of the persistence being set to the half-duplex operation. The actual value and rate of any alterations made to the persistence are subject to further study.

Retransmission Backoff

Another Level 1 issue has to do with the adjustment of the retransmission timer, $T_1$. Originally, the Vancouver TNC also added some delay each time a message was transmitted. Exponential backoff was insisted upon by Phil Karn last year, he has since agreed that exponential backoff may be too aggressive. Tom Moulton has indicated that some have found that on marginal links a simple arithmetic backoff may operate much more efficiently. Previously, the ARQ process started with a continual monitoring of round-trip time, and suggests that a continual monitoring of round-trip timing be used, and a smoothed version of this value be used to adjust $T_1$. Retries should not be re-established, or the information should be somehow written to a log, and other binary data.

Regarding Round-Trip Timing

Yet another modification to the $T_1$ timer is that it automatically altered the value of $T_1$, based partially on the number of digipeaters used by the link. Phil suggests that a continual monitoring of round-trip timing be used, and a smoothed version of this value be used to adjust $T_1$. Retries should be re-established, or the information should be somehow written to a log, and other binary data.

Carrier Sensing

Franklin Antonio, N6KFX, has suggested that RF carrier sensing be used in half-duplex operation. He proposed is to have it's value adjusted based on an average of the round-trip time for information packets sent and ACKs received. Once again, the old Vancouver AX.25 code had a variation on this, in that it automatically altered the value of $T_1$, based partially on the number of digipeaters used by the link. Phil suggests that a continual monitoring of round-trip timing be used, and a smoothed version of this value be used to adjust $T_1$. Retries should be re-established, or the information should be somehow written to a log, and other binary data.

Carrier Sensing

Franklin Antonio, N6KFX, has suggested that RF carrier sensing be used in half-duplex operation. He proposed is to have it's value adjusted based on an average of the round-trip time for information packets sent and ACKs received. Once again, the old Vancouver AX.25 code had a variation on this, in that it automatically altered the value of $T_1$, based partially on the number of digipeaters used by the link. Phil suggests that a continual monitoring of round-trip timing be used, and a smoothed version of this value be used to adjust $T_1$. Retries should be re-established, or the information should be somehow written to a log, and other binary data.
Most TCPs have a variable delay between when the RF transmitter is first turned on, and when data actually starts being sent, to allow for the transmitter to stabilize. This `keyup` delay is often called `TXDELAY`, and Franklin suggests that it be required to be `user-settable`, and benedict suggests that if possible it be implemented (at least partially) in hardware, due to the wide variation of values based on channel speed. When its specification is needed, `keyup Delay` is another issue that belongs in a Level 1 document.

**When to Stop Transmitting**

Another timer issue Franklin has brought up is a transmit turn-off delay (TX-Tail). He notes that if some packet has been placed in the character buffer, and although the host CPU may think a packet is done, the actual data may not have cleared the hardware yet. If the CPU then turns off the transmitter, the last `packet` will not have been completed, and the whole packet is therefore corrupted. He further points out that present implementations do not adjust this timing, but rather make it arbitrarily long to be sure the whole packet clears, regardless of channel speed. He suggests this timer also be `user-settable`.

Others indicate different methods can be used to insure the end of the packet has actually been sent. If a particular chip has a `three-character buffer`, the start of another frame three characters can be sent to the chip, then an abort command is issued to the chip, `which will abort the last `timer` set and allow time for the last frame to be completely sent`. This removes the necessity of having yet another `special user-settable way of checking this`. Franklin suggests that this method not be used more likely. Implementers should take note of this trick in order to reduce channel overhead.

**Additional AX.25 Level 2 Issues**

In addition to the previous items, there are a few more that have come up whenever alterative allocations to the AX.25 Level 2 `protocol` is discussed. These have not been thoroughly digested yet, and may be subject to further implementers at a later time. Some of them are listed below.

**Parameter Negotiation**

Several methods of negotiating different AX.25 operating parameters at connection setup have been discussed. The parameters most frequently talked about are packet data, and window size. Alteration of the default values have been suggested either by adding an information field to the SABM frame, or by using the `XID frame` under the original Vancouver protocol. This seems to violate some basic field rules, including Unnumbered and Supervisory frames not containing data (except for `FRMR frames`). The general feeling seems to not use this method.

Using `XID frames` to transfer special requests also has problems. Do the `XID frames` come before or after the connection established? It has been suggested that an XID command frame with the `Poll bit` set be used to convey a special request, and an XID reject frame with the final bit set be used to indicate the response to the request. This seems more manageable, and should be researched further.

**Larger Window Sizes**

When the Amateur community starts using satellites and high speed links, they may be beneficial to use larger frames and more of them. Our practical limit of window size is seven due to the three bit frame numbering plan implemented. Commercial satellite users have found that this limitation is too small when round-trip transmissions are involved. While not allowed under X.25 other HDLC protocols do allow the use of selective reject frames. These are used when a station receives a multiple frame transmission, with a bad one in the middle. The selective reject frame indicates not only that a bad frame was received, but gives the number of that frame, indicating it only needs that frame to complete the group. While the use of selective reject is not particularly time-saving with smaller windows, if the above extended windowing suggestion is implemented, some form of selective reject should also be implemented.

There is a possibility that many frames are missed, with selective re'jects issued for each of those frames. This could actually perform slower than if all outstanding frames were retransmitted. Actual implementation should take this into account, relying on some mean value to determine which action to take.

**Multi-Reject**

Presently, once a Reject frame has been sent, another cannot be issued until the error condition has been cleared. Suggestions have been made that multiple rejects be allowed to be sent, simplifying error recovery. Franklin suggests that this with older devices is in question, and will also be subject to further study.

**Data Compression**

Franklin Antonio has also suggested that standard data compression techniques be implemented over AX.25 Level 2 links, to shorten transmission times. While this is a good idea, and should be further researched, it probably doesn’t belong in the actual `protocol` specification.

**Assignment of PIDs Based on Manufacturer**

GLB Electronics requested a couple of years ago that certain PID numbers be reserved and assigned to implementers. After much discussion over several meetings, this issue is still unresolved. The moment is that this could be dangerous, potentially triggering user wars between system types. Franklin remains unconvinced that the benefits would outweigh the potential for harm.

**Conclusion**

This paper outlines most of the issues brought up regarding AX.25 Level 2 Version 2.0 since its adoption in October, 1984. There may be some additional ones that have been missed, which will be picked up at future and Conferences. There have also been several suggestions and corrections to the text of the AX.25 Level 2 Version 2.0 document, which were left out for the sake of brevity. Some of these corrections have been indicated to the ARL Digital Committee.

The next step is for this author to add the suggested changes to the AX.25 document and distribute the changed version to the Digital Committee, where it will be held under further review. At that time the SDL diagrams will also be added to the document. After passing that review step, the Digital Committee will approve a final new version of the protocol, then it will be printed and distributed.

Those with any comments, suggestions, or complaints should send them to this author at the above address. They will be passed to the Digital Committee in addition to being placed in the permanent AX.25 Documentation File kept by the author, which is the main basis for further AX.25 Level 2 modifications.
References


Fox, T. "Level 2 Protocol Proposal", AMRAD Newsletter, March 1982


Karn, P. "Proposed Changes to AX.25 Level 2", Correspondence, June 1987


Antonito, F. "AX.25 Proposed Changes", Correspondence, 1988


Anderson, P. "AX25LVZ", Correspondence, 1987

Johnson, L. "Thoughts on AX.25 Level 2 Version 2.1", Correspondence, 1988


GSA "Proposed American National Standard for Advanced Data Communication Control Procedures (ADCCP)", GSA, 1977