                        RELIABLE LINK LAYER PROTOCOLS

Status of This Memo

   This RFC discusses protocols proposed recently in RFCs 914 and 916,
   and suggests a proposed protocol that could meet the same needs
   addressed in those memos. The stated need is reliable communication
   between two programs over a full-duplex, point-to-point communication
   link, and in particular the RFCs address the need for such
   communication over an asynchronous link at relatively low speeds.
   The suggested protocol uses the methods of existing national and
   international data link layer standards. This RFC suggests a
   proposed protocol for the ARPA-Internet community, and requests
   discussion and suggestions for improvements. Distribution of this
   memo is unlimited.

Introduction

   This RFC is motivated by recent RFCs 914 and 916, which propose new
   standards for protocols that transfer serial data reliably over
   asynchronous communication lines. In this note, I summarize
   widely-used standards that have been in existence for some time that
   might be appropriate for this environment. I hope that the existing
   standards will be found to meet the needs the new proposals seek to
   address.

   In both the US and international standards areas, there are two major
   categories of serial data communication standards for the link layer.
   These categories are character-oriented and bit-oriented. The first
   is the older class; it is standardized in the US standard ANSI
   X3.28-1976 (which superseded the original version of 1971), and in
   the ISO standards IS 1745, IS 2111, IS 2628 and IS 2629. Although
   frequently used in synchronous environments, wherein the name binary
   synchronous (or bisynch) is used, these standards use the term "basic
   mode" to describe their procedures. The latter class is standardized
   in the US standard ADCCP (Advanced Data Communication Control
   Procedures), ANSI X3.66- 1979, and in the ISO standard HDLC
   (High-level Data Link Control procedures), in IS 3309, IS 4335 and IS
   7809.

   Other international standards, draft standards and vendor standards
   follow the ADCCP/HDLC procedures. Among these are SDLC (IBM), X.25
   LAPB (CCITT), IEEE 802.2/ISO 8802.2 LLC (LAN Logical Link Control)
   and ISDN LAPD (CCITT). Many vendors have built equipment which meets

Robinson   [Page 1]

the character-oriented standards, in both synchronous and
asynchronous environments, including all the major US mainframe
manufacturers.

The only other serial link layer protocol known to the author in as
wide use as these is DEC's DDCMP (Digital Data Communications Message
Protocol). This protocol uses a character count instead of framing
characters, but is in other respects a character-oriented protocol.

The next sections of this note will compare the three protocols above
on several bases, paying particular attention to the characteristics
that make particular aspects of the protocol appropriate to the
low-speed, asynchronous, serial environment.

Frame Structure

All serial protocols divide the data to be transmitted into units
known as frames. A frame is typically one to several hundred
characters in length. The frame structure is the major difference
used above to divide the protocols into three classes.

Character-Oriented Framing

Character-oriented protocols use two techniques for defining a frame.
First, it is necessary to determine where characters start and stop.
The technique used for this purpose is to transmit a number of unique
characters prior to the start of a frame. The character generally
used for this is the SYN character.

Note that this is not required when using asynchronous transmission.
Since each character is itself framed by start and stop bits, there
is never a question of where characters begin and end.

The main technique for structuring a frame is the use of special
framing characters to delineate the start and end of a frame, and to
delineate portions of the frame (such as header and text). Some uses
of character-oriented protocols require that these characters never
appear in the header or text of the frame, while others allow
"transparent" transmission. Transparency is obtained by preceding
each framing character by a unique control character, typically DLE.
In this way, all characters may be sent as header or text, except for
DLE. In order to allow DLE to be sent in the header or text, the DLE
is doubled.

Bit-Oriented Framing

   Bit-oriented protocols also use a unique character (technically, it
   is just an arbitrary bit-string) for frame delineation, which is the
   FLAG. This character provides frame synchronization. All bits
   between two occurrences of FLAGs constitute a frame. The FLAG is a 0
   bit, followed by six 1 bits, followed by another 0 bit. In order
   that the FLAG character not appear mistakenly in the data of the
   message, the sender inserts (and the receiver removes) an extra 0 bit
   after any five successive 1 bits in the data stream.

   Because this insertion of bits ("stuffing") results in arbitrary
   frame bit-lengths, bit-oriented protocols are generally useful only
   in synchronous transmission environments. Although it has never been
   attempted, however, one could imagine an asynchronous environment
   where each FLAG character that appears in the data is translated into
   a two- character sequence that avoids FLAGs, and at least one other
   character is similarly translated. For example, one could frame data
   with FLAGS, and send DLE-F to mean FLAG and DLE-DLE to mean DLE when
   these characters occur within the frame.

   Note that bit-oriented procedures do not require that the number of
   bits between FLAGs be an exact number of 8-bit characters, in
   distinction to character-oriented protocols and DDCMP. The necessity
   for character-alignment may be imposed at higher layers, as it is,
   for example, in X.25 Network Layer.

Frame Structure in DDCMP

   DDCMP uses a third approach to frame structure. Like
   character-oriented protocols, it uses a SYN character to achieve
   character synchronization prior to starting a frame, but one cannot
   dispense with this over asynchronous lines (see below). Contained
   within the fixed-length header portion of the frame is a count field,
   which reports how many characters are contained in the
   variable-length text portion. Since no framing characters are
   required at all, transparency is not a problem. However, because the
   count must be received properly for the sender and receiver to stay
   in frame synchronization, the header is protected with a separate
   error control checksum, which is typically two characters long (see
   below). Also, once a header error has been detected, the count field
   must be assumed to be invalid, and so there must be a unique
   character sequence that introduces the next header in order that the
   receiver can regain synchronization with the sender.

   Therefore, the SYN characters preceding a frame are required even on
   asynch lines.

Error Detection

   Several types of error control may be used, and the various protocols
   above are similar. Most synchronous uses require a cyclic redundancy
   check sequence be attached to each frame. This is a 16-bit sequence
   which can be easily generated and checked in hardware using a shift
   register, and can be somewhat more tediously done in software with
   about 5-6 instructions per character sent or received, and a 256 by
   16-bit lookup table. DDCMP and Bit-oriented protocols all require
   use of such a sequence. As noted above, DDCMP uses a check sequence
   twice, once for the header and once for the data.

   In some environments, weaker checks are used on character-oriented
   links. These take two forms. If the the number of significant bits
   per character is only 7, then the parity bit can be set to achieve
   either odd or even parity. ANSI standard X3.16-1976 specifies that
   odd parity should be used on synchronous links and even parity on
   asynchronous links. The second type of check is "longitudinal
   parity", wherein one character is added to the frame so that the
   number of 1 bits in each bit position summed over all the characters
   of the message and the check character is even. In other words, the
   exclusive-or of all the characters is 0. Character parity and
   longitudinal parity may be used together.

   Note also that most character-oriented control messages, such as
   those that poll, select, and acknowledge, are sent with only parity
   for error control.

Sequence Control

   All these protocol provide reliable transmission by sequencing the
   frames and providing positive and (in some cases) negative
   acknowledgments. Senders can ask the receiver for status if a reply
   is late.

   In character-oriented protocols, frames are implicitly numbered
   (typically) and only one may be outstanding at a time.
   Acknowledgments are explicitly numbered. One variant allows each
   block (frame) to be explicitly numbered as well; in this case up to 7
   may be outstanding.

   In bit-oriented protocols, frames are explicitly numbered and up to 7
   may be outstanding at a time. Optional control field extension
   allows for up to 127 outstanding. An alternate procedure that has
   been defined for use both in the ISDN LAPD environment and in IEEE

802 LAN environments uses, in effect, a one-bit sequence number and
one outstanding frame. Also, unsequenced, unacknowledged information
frames can be used when frames need not be sent reliably.

In DDCMP, the frames are explicitly numbered and up to 255 may be
outstanding.

Addressing

All of these protocols allow for addressing stations on a multipoint
link separately. In LAN environments, both a sending and receiving
address are required, whereas multipoint environments use a single
address and assume one master station communicating with multiple
addressed slave stations. In bit-oriented protocols, the address
provides extra information in that frames can be categorized as
commands or responses; in this sense, the address provides another
control bit per frame. However, it is possible to operate without
needing this distinction.

Addresses are typically one character long; bit-oriented protocols
allow for extension of this field to arbitrary length.
Character-oriented protocols use two-character (controller and
terminal) addresses.

For point-point operation, the address is clearly superfluous (except
to distinguish commands and replies in bit-oriented protocols); one
might imagine dispensing with it.

The Asynchronous Environment

Which of these protocols is best for the asynchronous environment?
This depends on the definition of "best", of course. One means of
judging is to compare the amount of overhead that each protocol would
add to each frame sent.

We will examine the overhead costs in two groups:

    framing/transparency/error checking,

    and addressing/control.

The two groups of functions are independent of each other, even
though the protocols mentioned above use specific combinations of
techniques from these two groups. Also, hardware available on
minicomputer-class and larger machines today supports the first group
of functions completely for these standard protocols; this fact
should allow for far greater performance from the gateway machine.

To the extent that such hardware becomes available for personal
computers, it can also be used there to reduce the protocol
processing overhead. Here's a breakdown of framing costs in
characters. RATP is also included for comparison.

Protocol	Frame	Check	Transp.	Total	F+C
char-or.	4	2	2	8	6
bit-or.	1	2	2	5	3
DDCMP	4	4	0	8	8
RATP	2	3	0	5	5

The transparency column indicates the anticipated cost in inserted
characters to achieve transparency across a 256-byte frame. The
figure for bit-oriented protocols is a pessimistic guess, because I
don't know the exact answer; it is between 0 and 8 characters, with
the worst case occurring when the data is all one bits. For
character-oriented protocols, we would expect on average one DLE
character in a 256-byte frame; the worst case overhead (256 DLEs) is
256 bytes.

Because transparency is so much a function of the user data, and
because we have ignored the cost of loss of frame synchronization in
the counting protocols (DDCMP and RATP), I argue that we should base
the comparison on the frame and checksum costs only. For these two
columns, character-oriented framing costs one more character per
frame than RATP. This, plus its wide use and hardware chip support,
create a strong case for its use in preference to RATP for framing.

Bit-oriented framing, as noted previously, is appropriate only on
synchronous links. The character oriented variant I postulated above
would have the same costs, but as it is not a standard, it is not
proposed here. So we now have constructed the following frame
format:

    DLE STX <control and data ...> DLE ETX CRC CRC

One objection to using character-oriented protocols as opposed to
character-count protocols is that it is necessary to examine every
character as it arrives. I respond to this objection as follows:

    1. Under some circumstances, such as when a header has been hit
    with an error, it will be necessary for the receiver to look at
    every character anyway.

    2. The environment for this protocol is a 1200 baud link; thus

120 characters per second need to be examined at a maximum. Even
on a relatively slow personal computer, this should not present a
problem.

We now turn our attention to the content and format of the control
information preceding each link frame. There are three components to
this cost, control, address, and acknowledgment. The address field
allows multipoint configurations and is superfluous for the
point-to-point environment proposed, but it is present in the public
standards and we restrict ourselves to those.

Acknowledgments are shown if they are required explicitly by the
protocol. A "0" indicates that the acknowledgments may be included
in the control information for traffic in the opposite direction, and
only need be sent explicitly when no reverse traffic is present (and
thus are assumed to take no required overhead). Only
character-oriented protocols have required acknowledgments.

```
              Cont.    Addr.    Ack    Total
    char-or.    0        3       2       5
    bit-or.     1        1       0       2
    DDCMP       3        1       0       4
    RATP        1        0       0       1
```

Again, the bit-oriented procedures provide the lowest overhead among
the public standards, but in this case there is no conflict in using
them in the asynchronous environment. In fact, even if all the other
aspects of RATP were to be adopted, I believe the control field
codings of the bit- oriented procedures represent a more efficient
use of the channel, are widely implemented, and allow for addition of
more functions later if desired. As stated above, there are several
protocols in the bit-oriented family. I would recommend use of LAPB,
since this is the most widely known of the family.

For those not familiar with bit-oriented control procedures, I have
included a quick summary of these procedures in Appendix A. Refer to
the standards listed at the end of this note for more detail.

RATP Compared to Public Protocols

   As can be seen from the above tables, RATP does not represent a
   significant savings compared to other widely-used protocols.

   Given frame lengths of 13 bytes, which appears to be the minimum for
   Thinwire II (RFC 914), 8 characters' overhead using the public
   standards represents 61% versus 46% for RATP's 6 characters. On a
   1200 baud line, the bandwidth available assuming only such short
   frames is thus 74 versus 82 characters per second, respectively.
   Since 1/13 of these are actually user data, the typing rates
   supported by these protocols using TCP/IP are pretty low, like 5.6
   versus 6.3 characters per second. Clearly a bigger cost is still
   found in the 12 characters overhead in Thinwire II (or 40 for TCP/IP
   with no compression).

   The costs improve dramatically when the number of user characters per
   frame increases. Thus, file transfer, or even line-blocked typing,
   should perform adequately. As frame size grows, the cost of the
   extra 2 characters per frame to use standard protocols rapidly drops
   to a few percent or less.

   RATP does allow one optimization which cannot be achieved in the
   standard protocols - the use of a one-character format that reduces
   the per-frame overhead to 3 characters (or 4 if a 16-bit CRC is
   used). However, in the scenario wherein single-character messages
   make sense, a user typing characters (with no higher layer
   protocols), the extra overhead is probably not a problem since the
   link is still lightly enough loaded that the extra overhead is still
   a small percentage of the available bandwidth. Also, allowing
   multiple frames in flight helps reduce the bottleneck caused by
   having one frame at a time outstanding.

On Check Sequences

   Both RFCs 914 and 916 propose to use relatively simple check
   sequences, which can be easily computed in a general-purpose
   processor. In one case, this is an additive check and in the other
   it is an exclusive-or (or parity) check. Although the additive check
   is slightly more powerful than the exclusive-or, both are relatively
   weak compared to CRC techniques.

   Since the intended network-layer protocol (IP) provides for similar
   checks on its header, and the transport layer (TCP) checksums its
   header and data, one might question whether the protection should
   also be provided at the link layer at all, or if it should, then are
   these checks good enough? Providing for recovery at the TCP layer

leads to slow recovery times, so this approach will probably yield
too poor a level of service for noisy links. More importantly, the
link layer control field needs a certain degree of protection to
prevent needless loss or duplication of frames in the face of line
errors.

A CRC check, in combination with the additive checks provided by IP
and TCP, yield an error-protection that is greater than that afforded
by either check by itself. This is because the two techniques
address fundamentally different characteristics of the possible
errors. The degree of increase is substantial compared to that of
two additive checks. That is, if two additive checks are cascaded,
there are many types of two-bit failures that will pass both the link
layer and TCP/IP checking.

Although I don't wish to include a detailed error analysis in this
note, I would support the use of a CRC type of error check because of
the far greater level of protection it affords. As I pointed out,
the cost per character is roughly 5-6 instructions, assuming the use
of a 256 by 16-bit lookup table. Again, at 120 characters per
second, the increased cost is not deemed to be too great.

Moreover, use of a standard CRC allows for the possibility that the
serial line chip's own CRC generation and checking hardware may be
used. Although such chips may not be present in the PCs in the
environment envisioned, they are likely to be available in the
gateway machine to which the PCs talk.

Data Compression: An Aside

I find the proposed methods of data compression of RFC 914
particularly interesting. I see these as independent of the choice
of the underlying link layer protocol, as it is in RFC 914. I am
aware of no such character-oriented compression that is in common use
in the communication world. The only techniques that come close are
in statistical multiplexing devices, which sometimes also include an
adaptive Huffman-coding to reduce link bandwidth. Since the Thinwire
II approach can recognize much longer repeated sequences than a
Huffman code, I expect that the potential savings are correspondingly
greater.

I would like to see a version of the Thinwire protocols which allows
for the template idea, but which keeps it independent of the higher
layer protocols in use. One way to achieve this is to allow
templates to be encoded and exchanged between the communicating
parties when they start up, and perhaps adaptively as conditions
warrant. I would anticipate that this sort of approach might well

have widespread applicability beyond the TCP/IP environment addressed
in RFC 914. The most important gain for this environment is removal
of the apparent exorbitant overhead that IP and TCP seem to present
for use over slow links.

Summary

   The link layer protocol I would advocate for asynchronous, dialup
   communication between PCs would use transparent, character-oriented
   framing, a 16-bit CRC error check, and the control procedures of
   LAPB. The CRC should be either CRC-16 or the CCITT CRC used in X.25,
   with the latter probably being preferred; modern link chips tend to
   support both of these if they support either.

   Evolution of integrated circuits that directly implement all of the
   public standards will dramatically drop the cost and raise the
   reliability of new implementations of standard protocols. Chip
   manufacturers have little motivation to address standards that are
   not widely used.

   Overhead for the suggested protocol is 8 characters per frame. Up to
   7 frames may be outstanding at a time in either direction of
   transfer. Choice of an appropriate maximum frame size is
   application-dependent, and would certainly be influenced by the
   quality of the physical connection; however, I believe that IP
   datagrams are acceptable frames for dialup 1200 baud service.

   Non-standard modifications that would save a little link overhead
   would be to dispense with the one-character address field, and to use
   the RATP count-oriented frame structure. These are not recommended,
   because they depart from common practice and yield modest
   improvements at best.

Postscript

   Those familiar with the early history of the Telenet Public Data
   Network should recognize that this proposal is essentially the same
   as the original link layer protocol specification for that network,
   circa 1976, except that the control procedures used at that time,
   known as LAP, have now been superseded by the more powerful and
   efficient LAPB, and their access links, as all X.25 access links, are
   synchronous rather than asynchronous. I did not set out to achieve
   this result, but just note it in passing.

   My personal view of where the world of personal computer access to
   data networks is heading is that X.25 will rapidly become the
   protocol of choice. One already sees third-party (for IBM PC) and

vendor (for Wang PC) implementations of X.25. CCITT is circulating a
proposal for accessing an X.25 data network using a dial-up X.25
connection, as recommendation X.32. Thus, I feel that the type of
communication proposed in this RFC and RFCs 914 and 916 will be used
for a relatively short period of time. The future holds, I believe,
LAN or X.25/X.32 data link layer and access, X.25 and/or ISO IP
network layer, and TCP and/or ISO TP4 transport layer.

References

   RFC 914, "Thinwire Protocol", Farber, Delp and Conte, 1984.

   RFC 916, "Reliable Asynchronous Transfer Protocol", Finn, 1984.

   "Technical Aspects of Data Communication", McNamara, Digital Press,
   1977.

   ANSI X3.4-1968, "American National Standard Code for Information
   Interchange (ASCII)", American National Standards Institute, Inc.,
   1968.

   ANSI X3.16-1976, "American National Standard Character Structure and
   Character Parity Sense for Serial-by-Bit Data Communication in the
   American National Standard Code for Information Interchange",
   American National Standards Institute, Inc., 1976.

   ANSI X3.28-1976, "American National Standard Procedures for the Use
   of the Communication Control Characters of American National Standard
   Code for Information Interchange in Specified Data Communication
   Links", American National Standards Institute, Inc., 1976.

   ANSI X3.66-1979, "American National Standard for Advanced Data
   Communication Procedures (ADCCP)", American National Standards
   Institute, Inc., 1979.

   International Standard 1745, "Information Processing - Basic mode
   control procedures for data communication systems", International
   Organization for Standardization (ISO), 1975.

   International Standard 2111, "Data Communication - Basic mode control
   procedures - Code independent information transfer", ISO, 1973.

   International Standard 2628, "Basic mode control procedures -
   Complements", ISO, 1973.

   International Standard 2629, "Basic mode control procedures -
   Conversational information message transfer", ISO, 1973.

International Standard 3309, "Data Communication - High-level data
link control procedures - Frame structure", ISO, 1982.

International Standard 4335, "Data Communication - High-level data
link control procedures - Consolidation of elements of procedures",
ISO, 1982.

International Standard 7809, "Data Communication - High-level data
link control procedures - Consolidation of classes of procedures",
ISO, 1984.

Recommendation X.25, "Interface between Data Terminal Equipment (DTE)
and Data Circuit Terminating Equipment (DCE) for Terminals Operating
in the Packet Mode and Connected to Public Data Networks by Dedicated
Circuit", The International Telegraph and Telephone Consultative
Committee (CCITT), 1980 (to be revised, 1984).

Recommendation Q.920, "ISDN User-network Interface Data Link Layer -
General Aspects", CCITT, 1984 (draft E).

Recommendation Q.921, "ISDN User-network Interface Data Link Layer
Specification", CCITT, 1984 (draft E).

Draft International Standard 8802.2, "Local Area Network Standards,
Logical Link Control", ISO, 1984 (draft).

Draft Proposed Addendum to DIS 8802.2, "Single Frame Service", IEEE,
1984 (Eighth Draft).

Appendix A - Bit-Oriented Control Field Contents

   There are three control field formats. The primary one is used for
   data frames (called "information frames" in the standards), and is
   coded as follows:

```
   8  7  6  5  4  3  2  1  <- bit number,  1 sent first
                     0      (signifies data frame)
               S  S  S      send seq , bit 2 low-order
            P/F            poll/final bit, for recovery
      R  R  R              receive seq  (ACK)
```

   Acknowledgments are cumulative. Recovery is typically to back up and
   continue from the lost frame. Use of the poll/final bit is beyond
   the scope of this note.

   Acknowledgments may also be sent in supervisory frames, coded as
   follows:

```
   8  7  6  5  4  3  2  1  <- bit number,  1 sent first
                  0  1      (signifies supervisory frame)
               T  T         frame type (see below)
            P/F            poll/final bit, for recovery
      R  R  R              receive seq  (ACK)
```

   Up to four frame types are possible; only two are required. The
   first is called RR, for "receive ready", and indicates acknowledgment
   and that the receiver is prepared to process more frames. The other,
   RNR for "receive not ready", is used for flow control of the sender.
   If flow control is not necessary, I suppose even this frame could be
   dispensed with.

   The other supervisory frames, "reject" and "selective reject", are
   varieties of negative acknowledgement that ask for retransmission of
   all and one (respectively) of previously transmitted frames.
   Positive acknowledgment and retransmission are the only really
   necessary procedures, however.

   The third frame format is called Unnumbered. Many possible functions
   are specified in the various standards for these frames, including
   initializing the link, reset sequence numbers, etc. The basic format
   is:

```
   8  7  6  5  4  3  2  1  <- bit number,  1 sent first
                  1  1      (signifies unnumbered frame)
         T  T  T     T  T         frame type
            P/F              poll/final bit, for recovery
```