Network Working Group
Request for Comments #98
Network Information Center #5744

                    Logger Protocol Proposal

                      Edwin W. Meyer, Jr.
                       Thomas P. Skinner
                       February 11, 1971

        With the ARPA Network Host-to-Host Protocol specified and at
least partially implemented at a number of sites, the question of what
steps should be taken next arises. There appears to be a widespread
feeling among Network participants that the first step should be the
specification and implementation of what has been called the "Logger
Protocol"; the Computer Network Group at project MAC agrees. The term
"logger" has been commonly used to indicate the basic mechanism to gain
access (to "login") to a system from a console. A network logger is
intended to specify how the existing logger of a network host is to
interface to the network so as to permit a login from a console attached
to another host.

        To implement network login capability now seems   quite
desirable.In the first place, it is natural for Network participants to
wish to learn more about the remote systems in the immediate fashion
afforded by direct use of those systems. In the second place, the
technical problems introduced by remote logins are probably less complex
than those involved with such further tasks as generalized file
transfer; thus, a Logger Protocol could be implemented relatively
quickly, furnishing additional impetus and encouragement for taking
still further steps.

        In order to furnish at least a basis for discussion (and at most
an initial version of a Logger Protocol), we have prepared this
document, which attempts to present a minimal set of conditions for
basing a Logger Protocol. This proposal covers only the mechanism for
accomplishing login. What occurs following login is not discussed here,
because we feel more experimentation is necessary before any protocol
for general console communication can be established as standard. In its
absence, each site should specify its own experimental standards for
console communications following login.

        Some of the points raised in this document have already reached
a certain level of consensus among network participants while at least
one point is rather new. It should be clearly understood, however, that
we feel regardless of the disposal of particular issues, Networkwide

                                                              [Page 1]

agreement should be reached as soon as possible on some general
protocol. This is all the more desirable in view of the fact that it is
quite likely that certain points which should be covered in this
protocol will only become apparent during the course of implementation;
therefore, the sooner a common basis for implementation can be reached,
the sooner a more rigorous protocol can be enunciated.

Before turning to 1) a discussion of the points with which to
decide the protocol should deal, and 2) specifications for the current
state of the protocolm we feel that we should acknowledge the
consideration that a case could be made for avoidingthe difficulty of
generating a Logger Protocol by simply declaring that each host may
specify its own, perhaps unique, preferences for being approached over
the Network. Although such a course is certainly possible, it does not
seem to us to be desirable. One reason for avoiding such a course is
simply that following it hamper general Network progress, in that
adressing the task of interfacing with some 20 systems is bound to more
time-consuming than to interface with "one" system, even though each
indivudual one of the former, multiple interfaces might be in some sense
simpler than the latter, single interface. Another consideration is less
pragmatic, but nonetheless important: agreement on a common protocol
would tend to foster a sense of Network "community", which would tend to
be fragmented by the local option route. After all, the Host-to-Host
Protocol could have been handled on a per-host basis as well; assumedly,
one reason why it has not had something to do with similar, admittedly
abstract considerations.

Context

Structurally, the mechanism serving to login a user over the Network
consists of two parts, one part at the using host, the other at the
serving host. The using or local host is the one to which the users
typewriter is directly connected; it contains a modulewhich channels and
transforms communications between the Network connection and the
typewriter. The serving or foreign host provides the service to be used;
it contains programming that adapts the logger and command system to use
through the Network rather than a local typewriter.

There are three different phases to a login through the network.

1. During the connection phase the users console is connected to
   the serving logger through the network. This is, of course,
   the most important phase from the protocol viewpoint.

2. The second or dialog phase consists of a sequence of exchange
   between the user and the logger that serves to identify the
   user and verify his right to use the system. In some hosts,
   this phase may be minimal or non-existent.

3. The admission phase occurs after the user has successfully completed the login dialog. It consists of switching his network typewriter connections from the logger to an entity providing a command processor of some sort. In some hosts this switching may be totally conceptual; in others there may be a real internal switching between entities.

The Connection Phase

        The issues involved in specifying a protocol for implementing login can be separatedintop two major parts: how to establish and maintain the network connection between the typewriter and the logger, and how to conduct a dialog after the connection is made. The first part is called the Initial Connection Protocol by Harlem and Heafner in RFC 80. It in turn consists of two subparts: how to establish a connection and how and when to destroy it.

        We endorse the proposal for establishing a connection made in RFC 80, which we summarize briefly for convenience. It is a two-step process utilizing the NCP control messages to effect a connection between the logger and the console of a potential user. First, the user causes the hosts NCP to send out a "request for connection" control message destined for the serving hosts loggers contact socket. The two purposes of this message are to indicate to the logger that this user wishes to initiate a login dialog and to communicate the identifiers of the and send socket he wishes to operate for this purpose. The logger rejects this request to free its contact socket. As the second step the logger choses two sockets to connect to the user's sockets, and dispatches connection requests for these. If the user accepts the connection within a reasonable period, the connection phase is over, and the dialog phase can begin. If the user does not respond, the requests are aborted and the logger abandons this login attempt.

        There is another part to an NCP: when and how to disconnect. There are two basic situations when a logger should disconnect. The first situation may arise of the serving host's volition. The logger may decide to abandon a login attempt or a logged-in user may decide to log out. The second situation may be due to the using host's volition or network difficulties. This situation occurs when the serving host receives a "close connection" control message or one of the network error messages signifying that further transmission is impossible. This may happen for either the "read" or the "write" connection, Disconnecting involves both the deletion of the network connections and the stoppage of any activity at the serving host related to that user. If the login is in progress, it should be abandoned. If the user is already logged in, his process should be stopped, since he no longer has control over what it is doing. This is not intended to restrict absentee

(i.e. consoleless) jobs.

The Dialog Phase

     The second major part other than getting connected is how to
conduct the login dialog. This resolves itself into two parts: what to
say and in what form to say it. The login dialog generally consist of a
sequence of exchanges, a prompting by the logger followed by a user
reply specifying a name, a project, or password. However, exactly what
information is desired in what sequence is idiosyncratic to each host.
Rather than attempt to specify a standard sequence for this dialog, we
have taken the approach that each host may specify its own sequence, so
long as it is expressible as an exchange of messages in a basic
transmission format. A message is a set of information transmitted by
one of the parties that is sufficient for the other party to reply.By
host specification, either the logger or the user sends sends the first
message of the dialog. After that, messages are exchanged sequentially
until the dialog is completed. In this context "message" has no relation
to "IMP message".

     The other issue involved in the login dialog is the format for
transmitting a message. We propose that it be transmitted as a sequence
of ASCII characters (see Specificarions) in groupings calle transaction
blocks.

   1. Character Set, We feel that there should be a standard
      character set for logging-in. The alternative, requiring a
      using host to maintain different transformation between its set
      and of each serving host, is a burden that can only narrow the
      scope of interhost usage, The character set proposed, ASCII is
      widely used standard. Each host must define a transformation
      sufficient to transform an arbitrary character sequence in the
      host's code into ASCII and back again, without any ambiguity,
      The definition of ASCII sequences to express characters not
      contained in ASCII is appropriate.

   2. Transaction Blocks. A message is transmitted as an arbitrary
      integral number of transaction blocks. A transaction block
      consists basically of a string of ASCII characters preceeded
      by a character count. (It also contains a code field. See
      below.) The count is included as an aid to efficiently
      assembling a message. Some systems do not scan each character
      as it is input from the console. Rather, such systems have
      hardware IO controllers that place input characters into a
      main memory buffer and interrupt the central processor only
      when it receives an "action" character (such as "newline").
      This reduces the load on the central processor. Because such
      a hardware facility is not available for interpreting

network messages this scheme is proposed as a substitute. It
helps in two ways. First, a system need take no action until
it receives all characters specified in the count. Second, it
need not scan each character to find the end of the message.
The message ends at the end of the of a transaction block.

Other Issues

        There are several other issues involved in the area of remote
logins   which   we   feel   should   be   raised,   although most need not
necessarily have firm agreements reached for an intial protcal.

1.   "Echoplex". Echoplex is a mode of typewriter operation in which
     all typed material is directed by the computer. A key struck by
     a user does not print directly. Rather the code is sent to the
     computer, which "echoes" it back to be printed on the typewriter.
     To reduce complexity, there is to be no option for network
     echoplexing (for the login only). A using system having its
     typewriters operating in echoplex mode must generate a local
     echo to its typewriters. However, a serving system operating
     echoplexed should suppress the echo of the input during the login
     phase.

2.   Correction of Mistakes. During the login dialog the user may make
     a typing mistake. There is no mistake correction ecplicitly
     proposed here. If the message in error has not yet been
     transmitted, the user can utilize the input editing conventions
     of either the using or the serving host. In the first case, the
     message is corrected before transmission; in the second, it is
     corrected at the serving host. If the user has made an
     uncorrectlable mistake, he should abort the login and try again.
     To abort, he instructs the local (using) host to "close" one of
     the connections. The connections are disconnected as specified in
     the Initial Connection Protocol.

3.   "Waiting". It may happen that the user may get into a login dialog
     but for some reason does not complete it. The logger is left
     waiting for a response by the user. The logger should not wait
     indefinitely but after a reasonable interval (perhaps a minute)
     abort the login and "close" the connections according to the
     provisions of the Initial Connection Protocol.

4.   Socket Assignments. The Initial Connection Protocol does not
     specify the ownership of the sockets to be used by the logger in
     connecting to the user. (The use code field of the socket
     identifier determines ownership.) The sockets may belong to the
     logger or may have an arbitraryuser code not used by another
     process currently existing at the serving host. Under this initial

scheme, it is not possible to implement administratively assigned
user codes, because the logger must assign permanent sockets
before the identity of the user is verified. A future connection
protocol can avoid this problem by implementing a socket
connection as a part of the admission phase. The logger would talk
to the user over the logger's sockets. Following identification it
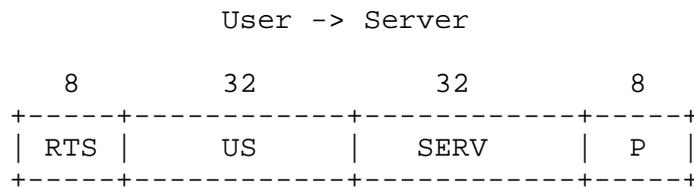would transfer the connections to the sockets belonging to the
user.

5.   General Console Communications. A companion paper under
      preparation outlines a protocol for general console communcations
      between hosts. This paper will seek to adress most of the
      problems associated with typewriter like communications. This
      includes discussion of full and half duplex, character escapes,
      action characters and other pertinent topics. Such a protocol
      might not be suitable for all terminals and host systems but
      would include solutions to problems for many. It is not
      intended as a monolithic standard, but rather as a recommendation
      for those sites who wish to implement a common protocol. The
      important point is that we feel quite a bit of actual network
      usage is required before all the problems are better understood.
      This is a prerequisite for devising a standard.

                          SPECIFICATIONS

Initial Connection Protocol - Connection Phase

      The following sequence is as presented in RFC 80. It is restated
      here for completeness.

1.   To intiate contact , the using process requests a connection of
      his receive socket (US) to a socket (SERV) in the serving host.
      By convention, this socket has the 24-bit user number field set
      to zero. The 8-bit tag or AEN field is set to one indicating
      the socket gender to be that of a sending socket. There is no
      restriction on the choice of the socket US other than it be of
      of the proper gender; in this case a receive socket. As a result
      the using NCP sends:

                       User -> Server

                 8   32   32   8
              +-----+------------+------------+-----+
              | RTS | US | SERV | P |
              +-----+------------+------------+-----+

over the control link one, where P is the receive link assigned
by the user's NCP.

2. The serving host now has the option of accepting the request for
    connection or closing the the connection.

    a. If he sends a close it is understood by the user that the
        foreign host is unable to satisfy a request for service at
        this time. The serving host's NCP would send:

                          Server -> User

                 8   32   32
              +-----+-----------+------------+
              | CLS | SERV | US |
              +-----+-----------+------------+

        with the user's NCP sending the echoing close:

                          User -> Server

                 8   32   32
              +-----+-----------+------------+
              | CLS | US | SERV |
              +-----+-----------+------------+

    b. If the serving host is willing to provide service it will
        accept the connection and immediately close the connection.
        This results in the the serving host's NCP sending:

                          Server -> User

                 8   32   32
              +-----+-----------+------------+
              | STR | SERV | US |
              +-----+-----------+------------+

                 8   32   32
              +-----+-----------+------------+
              | CLS | SERV | US |
              +-----+-----------+------------+

with the user's NCP sending the echoing close. It sends:

                          User -> Server

            8   32   32
        +-----+-----------+------------+
        | CLS | US | SERV |
        +-----+-----------+------------+

It should be mentioned that the echoing closes are required
by the host-to-host protocol and not by the logger initial
connection protocol.

Character Set

    The character set used in conducting the login dialog is
standard ASCII as documented in "American National Standard Code for
Information Interchange", ANS X3, 4-1968, American National Standard
Institute, October, 1968. A logger at a serving host may demand any kind
of input that can be expressed as a string of one or more ASCII
characters. It similarly, it may output any such string.

    All ASCII characters are legal, including the graphics and
control characters. However, it is proposed that the only standard way
of indicating the end of a console line be the line feed character
(012). This is in accordance with an anticipated change to the ASCII
standard.

    Currently the ASCII standard permits two methods of ending a
line. One method defines a single character, line feed (012), as
incorporating the combined functions of line space and carriage return
to the lefthand margin. The second method, implicitly permitted by
ASCII, uses the two character sequence line feed (012) and carriage
return (015) to perform the same function.

    There is a proposal that the ASCII standard be changed to
include a return to the left-hand margin in all vertical motion
characters of at least one full space (line feed, vertical tab and new
page). This will disallow the dual character sequence to end a line.

    It is suggested that a character in a hostst character set not
having any ASCII equivalnet be represented by the ASCII two character
sequence ESC (033) and one of the ASCII characters. Each host should
publish a list of the escape sequence it has defined.

Transaction Block Format

        All textual messages exchanged between user and logger are to
consist of one or more "transaction blocks". Each transaction block is a
sequence of 8-bit elements in the following format:

                 <code> <count> <char1> ... <charn>

<code>   is an 8-bit element that is not interpreted in this
              protocol. In the proposed general console communications
              protocol, this field specifies communication modes or
              special characteristics of this transaction block. Here
              <code> is to be zero.

<count>   is an 8-bit element that specifies the number of character
              elements that follow in this transaction block. It is
              interpreted as a binary integer which has a permissible
              range between 0 and 127. The most significant bit is zero.

<chari>   is an 8-bit element containing a standard 7-bit ASCII
              character right-adjusted. The most significant bit is
              zero. The number of <chari> in the transaction block is
              governed by the <count> field. A maximum of 127 and
              minimum of zero characters are permitted in a single
              transaction block.

        The most significant bit of each of these elements is zero,
effectively   limiting   each   of   these elements to seven bits of
significance. The reason for doing this is twofold: the eighth bit of
the <chari> elements is specifically reserved for future expansion, and
it was desired to limit all the elements so as to permit certain
implementations to convert the incoming stream from 8-bit elements to
7-bit elements prior to decoding.

        With one exception, there is to be no semantic connotation
attached with the division of a logger-user message into one or more
transaction blocks. The character string comprising the message to be
transmitted may be divided and apportioned among multiple transaction
blocks according to the whim of the sending host. If less than 128
characters in length, the message may be sent as a single transaction
block. The exception is that separate messages may not appear in the
same transaction block. That is, a message must start at the beginning
of a transaction block and finish at the end of one. Note also that
there is no syntactic device for specifying the last transaction block
of a message. It is presumed that the logger end user both have
sufficient knowledge of the format to know when all of a message has
arrived

Note that the first 8-bits of data transmitted through a newly
established connection must be a type code as specified in Protocol
Document 1. This type code must be sent prior to the first transaction
block and should be discarded by the receiving host.

Acknowledgments

Robert Bressler, Allen Brown, Robert Metcalfe, and Michael
Padlipsky contributed directly to the establishment of the ideas
presented here. Thanks are due Michael Padlipsky and others for
editorial comments.

[This RFC was put into machine readable form for entry]
[into the online RFC archives by Carl Moberg 1/98]