                    A REQUEST FOR COMMENTS ON
                      SOCKET NAME STRUCTURE

INTRODUCTION

        This RFC is in answer to a request (made at the
February NWG Meeting at the University of Illinois) that
we comment on several suggested socket name structures.
We apologize for the delay in getting out these comments
and we hope that you will respond more quickly with your
reactions.
        Please direct your replies via the standard RFC
mechanism.
        Two structures are presented in this RFC as shown
below.

```
                     31                    1
       +-----------------------------+-+
  1.   |         Arbitrary           | | <-- gender
       +-----------------------------+-+


                     24            7    1
       +-----------------------+------+-+
  2.   |        User ID        | tag  | | <-- gender
       +-----------------------+------+-+
```
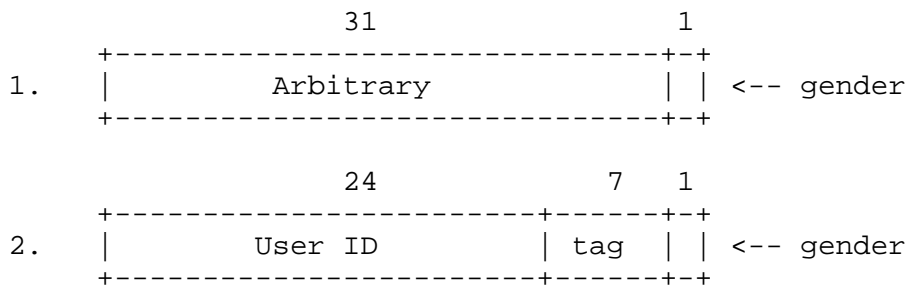
        Three variations are given for the way in which
socket names are assigned, as examples of use of the
first structure.
        1.   Users pick the arbitrary number arbitrarily
             and associate it with a process.
        2.   A logger chooses the arbitrary number dynamically
             and associates it with a process via a directory.
        3.   The arbitrary number is assigned outside of a
             logger but may be issued by a logger to the
             remote party.

The second format shown above associates sockets specifi-
cally with users as opposed to processes.

    The following discussion covers three different schemes
of socket identifier assignment using a simple example.
User A at Host A has agreed (by letter, telephone, etc.)
with User B at Host B for their respective processes to
establish a connection through the Network at a particular
time. User B is to be waiting for the connection attempt
initiated by User A. The issues to be faced are those of
addressing (how is User A to know to which socket to connect?),
and of security (how are both users to be confident that
they are talking each other, and not some interloper?).

    A fourth scheme follows which addresses another concept
of Network use--that connections are made between processes
and that processes not users should be identified via
Socket names.

FREELY SELECTED RANDOM SOCKET IDENTIFIERS (Scheme 1)

    Under this scheme a user is able to use any 32-bit
socket identifier he chooses. Two restrictions apply: the
least significant bit denotes the socket's gender (0-read,
1-write), and no more than one socket bearing a given iden-
tifier can be active at a host at a time.
    The two users select suitably random identifiers ("a"
and "b"). User A will attempt to activate his socket with
identifier "a" an connect it to socket "b" at Host B. There
is the possibility that somebody other than User B has
activated socket "b" at Host B so that User A will address
the wrong party. However, the possibility that some other
user has accidentally picked this particular identifier is
reasonably small, since there are about a billion different
identifiers. When the connection request from A gets to
User B, he examines the identifier of the calling socket.
If for some reason it is not "a" or not from Host A, he
rejects the request, because it is likely to be from some

outside party. If the calling socket is named, "a" and
from Host A, User B can be reasonably sure that it is from
User A. It is very unlikely that some other party will
accidentally address socket "b" from a socket named "a".
    The advantages of this scheme are: simplicity and
reasonable security in a non-malicious environment. The
disadvantages are that there are possibilities from annoy-
ingly unavoidable conflicts with other users and that each
pair of users must conduct a prior confidential private
communication (as opposed to a broadcast announcement in
more secure schemes).

HOST-SELECTED IDENTIFIERS PLUS DIRECTORY (Scheme 2)

    This system uses the same socket identifier structure
as presented above, except that the Host picks the identi-
fier at the time the socket is assigned, and the user has no
no prior knowledge or control of the assignment. By itself,
this system would be totally unusable, because there would
be no way for User A to address User B. However, it allows
certain service functions (such as the Network logger) to
have specifically assigned sockets.
    One of these is a Network Directory service. This
serves to relate a socket identifier at a particular host
to the name of the user operating it. This might either
be a single distributed service, or there might be a separ-
ate service at each host.
    Under this scheme, each user, A and B, first activates
his socket (or somehow gets his host to assign and tell
him of a socket identifier). Then he gets the Directory
module at his host to associate his name with the identi-
fier of the socket just activated. Following this, User A
in some manner gets the Directory Service at Host B to tell
him the socket identifier assigned to User B. Then User A
dispatches a connection request for this socket.

When User B gets the request, he similarly calls on the
Directory service at Host A to find out the name of the user
who is operating the socket User B was called by. If the
name is that of User A, User B can safely accept the request.
Otherwise, he rejects.

This scheme is rather cumbersome, but some directory
services must exist for Host-selected socket identifiers to
work. On advantage of the Directory Service is thst it
allows symbolic addressing. A sizeable disadvantage in view
of its complexity is that it does not provide absolute
security. (For exemple, after User A gets the identifier
of the socket he is to address, User B could deactivate it,
and somebody else could come along and get the same-named
socket.)

ADMINISTRATIVELY ASSIGNED USER IDENTIFIERS (Scheme 3)

This is the system that is put forth on page 5 of
Protocol Document 1(8/3/70). Under it a user is permanently
assigned a user identifier by his home host. There is a
user identifier subfield within the socket identifier, and a
user is permitted by an NCP to operate only those sockets
bearing his uder identifier. This gives the user a selec-
tion of 256 sockets operable by him.

In arranging for the connection the two Users A and B
tell each other their user identifiers (alternatively a user
ID could be read from a directory), and User B specifies
which of his sockets ("b") that he will "listen" on. At
connection time, User A selects one of his sockets and
requests connection for it to socket "b" specified by User B.
By protocol only User B can operate socket "b", so User A
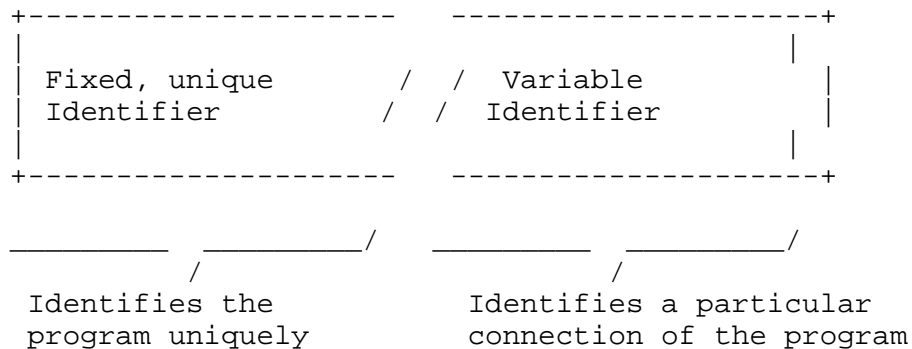can be certain of reaching the right party.

When User B receives the connection request, he examines
the user identifier subfield of the calling socket identifier.
If it is the user identifier of User A, User B accepts the
connection request, confident that it is actually User A at
the other end. Otherwise B rejects the request.

The advantages of this scheme are that if both hosts
involved in a connection enforce the user ID assignment,
the misconnection aspect of security is solved and there
can be no socket naming conflict between users. Also,
arrangements can be made openly and publicly between many
potential communicators. A disadvantage to this scheme is
that some systems may be incapable of insuring user ID
integrity.

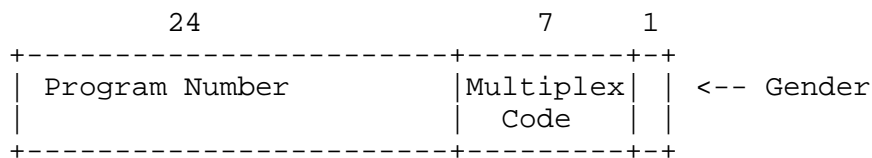A VIEW OF SOCKET NAME MEANING (Scheme 4)

     Another view of Network use is that programs will con-
nect to programs, via NCPs. Some of these programs may be
multi-access subsystems that are really agents for local
consoles (and TELNETs). Consoles will generally communicate
through some such software agent rather than directly to
an NCP.
     Programs, then, must have a fixed, unique identifier,
known to its remote users and perhaps to its local logger.
The identifier is constant; it does not change from day to
day. If such a program is to allow multiple concurrent
connections (for many or a single user) then it must have
a range of variable identifiers as well. It makes sense
to group these sockets in a contiguous range. The variable
identifiers are transient and are dynamically associated
with Network logical connections.

```
        +--------------------   --------------------+
        |                                          |
        | Fixed, unique       /  /  Variable        |
        | Identifier         /  /  Identifier        |
        |                                          |
        +--------------------   --------------------+


        _____ _____/   _____ _____/
                 /                      /
          Identifies the          Identifies a particular
          program uniquely        connection of the program
```

The above premise is that the program (or agent) is
doing the communicating with an NCP and thus needs to be
identified for message traffic routing from an NCP. In
the past it has been said that users can be mobile, i.e.,
log on from different sites, and thus it is the user that
needs identification. In many typical on-line systems the
user first requests a service and then identifies himself
to the service for purposes of accounting, etc. User IDs
can be transmitted after requesting a service and can thus
be elevated above the meaning of socket names.

     A program might typically associate the terminals, for
which it is an agent, with the variable part of the identi-
fier, i.e., the particular connection(s). For example,
the Network Services Program (NSP) at Rand now uses the
following format for socket names. The first 24 bits are
administratively assigned and would be known to a logger.
The multiplex code is normally chosen randomly. Predefined,
fixed multiplex codes are possible also.

```
              24                    7      1
     +------------------------+---------+-+
     | Program Number         |Multiplex| |  <-- Gender
     |                        |  Code   | |
     +------------------------+---------+-+
```

     The Socket name structure #1 (page 1) thus accomodates
the above example as well as other exploratory socket name
structures and various "standards" superimposed on the arbi-
trary field.

     [This RFC was put into machine readable form for entry]
        [into the online RFC archives by Simone Demmel 4/97]