Network Working Group
Request for Comment #167
NIC #6784

Socket Conventions Reconsidered

Athay Bhushan (MAC)
Bob Metcalfe (Harvard)
Joel Winett (LL)

24 May 1971

Category: C1, C3, C8
Related RFCs: #147, #129
Related Functional Documents: #1

The current NCP Protocol says nothing about how hosts should assign socket numbers to process ports, except that the low-order bit is to specify socket gender (i.e., send or receive). Two recent proposals call for additional network-wide conventions on the 32-bit socket-number. The first proposal asks that a portion of the socket number be reserved for a network-unique user number for accounting and access control. The second proposal asks that the high-order 16 bits of the socket number be zero to assist smaller hosts in reducing the space required for socket number tables.

It is recommended that both of these proposals be set aside. Because a large perturbation of the current NCP Protocol is required to provide adequate handles for accounting and access control, and because the socket number is already underpowered for its use, it is recommended that both proposals be set aside until serious consideration can be given to a major NCP Protocol overhaul.

DISCUSSION

The socket number, as it is used in the current NCP Protocol is a small number with a big function. It will probably be found that a substantially more powerful identification mechanism (e.g., a hierarchical naming scheme with arbitrarily long names) is required to satisfactorily manipulate process ports. Two features of such a mechanism will be (1) that it treats accounting and access control with the respect they deserve, and (2) that it is part of a simpler NCP Protocol more easily implemented under the existing size and complexity restrictions of smaller hosts.

Socket numbers are process port identifiers used in establishing connections between processes. It is essential that they be UNIQUE to avoid ambiguity during connection. It is important that their assignment to specific processes be REPEATABLE for reconnection on a regular basis.

To assure that process port identifiers are unique and repeatable it is necessary to subject their allocation to access controls. The simplest of access controls assuring uniqueness is that provided by NCPs which check their tables of active connections for duplication when a process requests the use of a specific socket number.

There is real difficulty in constructing schemes for allowing socket number assignments to be repeatable. Some socket numbers are to be universally known and associated with processes operating with specified protocols (e.g., a logger socket, an RJB socket, a file transfer socket). Other socket numbers might not be universally known, but given to their users in a transmission over a universally known socket (e.g., the socket pair specified by the transmission over the logger socket using the Initial Connection Protocol (ICP)). Concurrently running

instances of a program will require distinct process port identifiers. Therefore, socket numbers will in general need to be dynamically assigned via some system controlled allocation function.

There are a number of ways of providing for potentially repeatable socket number assignments. One bad way is to have the NCP keep a list of all assigned socket numbers with some indication of who is permitted to use them and for how long -- like keeping track of magnetic tape reels. If there were few available socket numbers (e.g., 16 bits worth) this bad method or one comparably distasteful and logistically foreboding would have to be adopted. With an abundance of socket numbers it is possible, using sparse socket number assignment, to devise simple algorithms for deciding whether a socket numbers being requested by a process can be allocated freely. Such algorithms might take into account (1) the dynamic status of the socket (i.e., its association with a currently active connection), (2) its reserved status as a standard service port address, and (3) its access control attributes in relation to those of the requesting process.

One good strategy for controlling socket numbers is to partition the full socket space at a host among its network users. Under this scheme a user could be assured of having the repeatable use of his partition. It might also be helpful to designate a utility partition for use in socket number allocations where repeatability is not essential. Such socket numbers could be selected from the utility partition by some clever construction on the date and time.

It will often be the case that a program will be written to use several connections. Remembering that this program might find itself being executed concurrently by several processes belonging to several users, it might be convenient to code with socket tags which are to be extended with runtime user and process identifier fields.

Socket numbers will tend to be viewed -- should be viewed -- as having three fields: a user field to assist in providing repeatability, a process field to assure uniqueness for concurrent instances of a program, and a tag field to enable the convenient referencing of multiple connections to a single process.

Although fields will be helpful in dealing with socket number allocation, it is not essential that such field designations be uniform over the network. In all network transactions the 32-bit socket number is handled with its 8-bit host number. Thus, if hosts are able to maintain uniqueness and repeatability internally, socket numbers in the network as a whole will also be unique and repeatable. If a host fails to do so, only connections with that offending host are affected.

Because the size, use, and character of systems on the network are so varied, it would be difficult if not impossible to come up with an agreed upon particular division of the 32-bit socket number. Hosts have different internal restrictions on the number of users, processes per user, and connections per process they will permit.

It has been suggested that it may not be necessary to maintain socket uniqueness. It is contended that there is really no significant use made of the socket number after a connection has been established. The only reason a host must now save a socket number for the life of a connection is to include it in the CLOSE of that connection. If such is really the case, then the NCP Protocol might be improved by inventing a new CLOSE which uses the host-line pair associated with the connection. Hosts which are short on space could then forget a socket number immediately after successful connection.