Network Working Group
Request for Comments #143
NIC #6728
Categories: D.1, D.3
Obsoletes: None
Updates: 123, 145

W. Naylor
J. Wong
C. Kline
J. Postel
UCLA - NMC

3 May 1971

Regarding Proferred Official ICP

We should like to comment on a race condition discovered in the ICP as proposed in NWG/RFC #123. The problem arises when the server attempts to initiate a second connection to the user's receive socket and the first connection is not yet closed. Using a similar notation to that of NWG/RFC #123 the following table illustrates the sequence of events in the proferred and proposed ICP. The last two columns indicate which actions must be completed before the current action may be initiated. User and Server are third level programs, and UNCP and SNCP are the users NCP and Servers NCP respectively. Allocates have not been included since they add nothing to the argument.

			Required Predecessors	
Reference #	Action	Initiator	"Proferred"	Proposed
1	Listen(L,32)	Server	--	--
2	Init(U,L,32)	User	--	--
3	RTS(U,L,'l')	UNCP	2	2
4	STR(L,U,32)	SNCP	1 and 3	1 and 3
5	Send(L,S)	Server	4	4
6	SEND('l',S)	SNCP	5	5
7	RECEIVE('l',S)	UNCP	6	6
8	Receive(U,S)	User	7	7
9	Close(L)	Server	5	5
10	CLS(L,U)	SNCP	9 and 7	9 and 7
11	Close(U)	User	8	not used
12	CLS(U,L)	UNCP	11	10

Reference #	Action	Initiator	Required Predecessors "Proferred"	Required Predecessors Proposed
13	Init(S,U+1,B u)	Server	9	9
14	RTS(S,U+1,'l' 2)	SNCP	13	13
15	Init(S+1,U,B s)	Server	13	14 and 18
16	STR(S+1,U,B s)	SNCP	15	15
17	Init(U+1,S,B u)	User	11	12
18	STR(U+1,S,B u)	UNCP	17	17
19	Init(U,S+1,B s)	User	17	17
20	RTS(U,S+1,'l' 3)	UNCP	19	19

Note that in the Proferred Order column, 16 can occur before 12 in which case UNCP would find socket U in use and probably return a CLS (U,S+1). The Server would probably then assume the User was finished with the conversation.

The above problem is resolved by eliminating the Close from one side and causing that side to wait for the CLS from the other side before doing an Init. We propose that eliminating the user's Close (U) is the best solution. (The user NCP must of course return a CLS in response to the CLS sent by the server NCP).

The Server's Close (L) leads more quickly to the reuse of socket L thus the serving of another user.

To clarify the above discussion which may seem confusing at first glance, let us demonstrate the problem in the language of RFC #123.

Server		User	
(S1)	Listen(L,32)	(U1)	Init(U,L,32)
(S2)	[Wait for match]	(U2)	
(S3)	Send(L,S)	(U3)	Receive(U,S)
(S4)	Close(L)	(U4)	Close(U)
(S5)	Init(S,U+1,B) u	(U5)	Init(U+1,S,B) u
(S6)	Init(S+1,U,B) s	(U6)	Init(U,S+1,B) s

Notice that since server and user are independent (probably in different hosts), server could execute (S6) before user executes (U4) and could receive an error back from user's NCP that socket U is busy. Similarly, user could execute (U6) before server executes (S4) and could receive an error back from his own NCP that socket U is not yet closed (assuming an implementation where sockets are kept busy until a CLS match).

Various modifications could be made to ICP to solve this problem. We propose the following ICP:

```
          Server                              User
          ------                              ----
          Listen(L,32)                        Init(U,L,32)

          [Wait for match]

          Send(L,S)                           Receive(U,S)

          Close(L)                            [Wait for CLS]

          Init(S,U+1,B )                      Init(U+1,S,B )
                      u                                   u

          [Wait for match]                    Init(U,S+1,B )
                                                          s

          Init(S+1,U,B )
                      s
```

This ICP assumes the following:

1. The user can inquire or is notified of the fact that one of his connections has been closed.

2. The server can inquire or is notified that a connection for which he has done an Init (or Listen) is now open.

Both of the above seem basic to any NCP - user interface.

This race condition problem would not exist had the dynamic reconnection features of RFC #36 been included in the NCP protocol and had dynamic reconnection been used in this ICP.