Network Working Group
Request for Comments: 133
NIC 6710
[Categories C.4, C.5, C.6, D.4, D.7, D.7]

R. L. Sunberg
Harvard University
27 April 1971

FILE TRANSFER AND ERROR RECOVERY

1 FILE TRANSFER PROTOCOL

1A Handshaking

I think that Mr Bhushan(RFC #114, NIC 5823) is not strict enough in his concept of a transaction sequence. Every transaction should prompt a response from its recipient (recall Kalin's crates -- RFC #60, NIC 4762). Control should pass back and forth until the server terminates. The server _always_ gets the last word (more on error recovery later).

Some sample interchanges are given.

User		Server	Comments
<...>	==>		Establish a connection
	<==	<...>	
<I><...>	==>		Identify self
	<==	<+>	Ok, ready
<R><...>	==>		Retrieval request
	<==	<rs>	I've got your file
<rr>	==>		Send it
	<==	<,><...>	Here's the first part
<rr>	==>		Got it
	<==	<+>	All done
<S><...>	==>		Store request
	<==	<rr>	Ok, go ahead
<#><...>	==>		Here's some protection stuff
	<==	<rr>	Ok
<*><...>	==>		Here's the file
	<==	<+>	Got it. All done.

See section 2B, below, for examples of error recovery.

1B Extensions to the file transfer protocol

The file transfer protocol needs a mechanism for accessing individual records of a file. This will be particularly useful when very large data bases appear on the network. The following definitions should be added to the protocol:

The store(S) and retrieve(R) requests have the data field format <key>, where <key> has the syntax:

```
 <key>::=<devicename>RS<filename>US<keyname> | <filename>US<keyname>.
                        --          --                      --
```

The <pathname> syntax is changed to:

```
    <pathname>::=<devicename> | <filename> | <pathname>RS<filename>.
                                                            --
```

If a retrieve(R) request is given with a data field with <key> syntax rather than <pathname> syntax, then the returned data will consist of the record following the matching <key>. If a store(S) request is given with a data field of <key> syntax, then the supplied data will replace the record following the matching <keyname>. If the keyname does not exist, the record will be appended to the named file. The individual installation must provide the linkage between the <keyname> and the record it references.

In addition, the lookup(L) request will provide a list of keynames into a file (or the name of a file which contains the keynames).

Transaction code F (request File directory) requests a listing of available files. The data field of the F transaction is of the form: <pathname>GS<pathname>GS... All files in the server system
                -- --
which match one or more of the given <pathname> specifiers are listed in a return file. The format of the data fields of this file is: <pathname>GS<pathname>GS... If a <pathname> field in
                    -- --
the request transaction does not include a <name> field, the default is all files on the given device. Some examples are given:

```
    <F><DC1 DSK[62,50]] GS JOE>
        ---             --
```

This example requests a list of all files on the disk specified by [62,50] plus all files named JOE. The response could contain in the data field:

```
<DC1 DSK[62,50] RS ALPHA RS BETA RS JOE GS DC1 DSK[10,50] RS JOE>
 ---            --       --      --     -- ---            --
```

This message states that in the [62,50] area of the disk there are files ALPHA, BETA, and JOE, and that JOE is also a file in the [10,50] area of the disk.

2 ERROR RECOVERY

2A Error recovery procedures have been noticeably lacking to date. The usual approach has been to close the connection and start from scratch. Mr Bhushan proposes a third level abort but doesn't really detail the implementation. I propose a multilevel error recovery procedure as follows.

2B If an error occurs which does not cause a loss of third level transaction boundaries and only affects one side of a duplex connection, a third level recovery is possible via a transaction sequence abort. An example is given:

User		Server	Comments
<R><...>	==>		Send me this file
	<==	<rs>	Ok, I've got it
<rr>	==>		Ready
	<==	<*><...error>	Here it is (with an error)
<-><D>	==>		No. (data) error
	<==	<-><D>	Sorry, forget it
<R><...>	==>		Send the file (again)
	|<==	<rs>	Ready (doesn't get there)
	...		(waiting)
<-><0>	==>		Error, timeout
	<==	<-><0>	Sorry, forget it
<R><...>	==>		Send the file (third time)
	<==	<rs>	Got it
<rr>	==>		Ready
	<==	<*><...>	There it is
<rr>	==>		Got it
	<==	<+>	Done (finally>

Note that the server always gets the last word in error situations as well as normal transmission.

2C Although the above examples are given in terms of Bhushan's transaction codes, this form of error recovery is implementable in any protocol which uses flagged blocking and duplex connections.

2D If errors cannot be recovered as above, then some means must be available to clear the link completely and resynchronize. I suggest that an 8-bit argument be appended to the interrupt-on-link NCP message (INR, INS). The receiver would send <INR><error> to indicate that the block boundaries were lost and all incoming data is being discarded. The sender, upon receiving the INR, would flush all queued output and wait for the link to clear. The NCP would then send a <INS><newsync> message and, when it was received (RFNM returned), a negative termination would be sent on the link. The receiver begins accepting data again when the INS is received. This assumes that any process can flush untransmitted data and detect a clear link. Note that this method is useable on any simplex connection.

2E If all else fails, one can resort to closing the faulty socket.

3 NCP VERSION NUMBERS

3A I suggest that the NCP be given a version number and the next version include two new message types: <WRU> ('Who aRe yoU?') requests a version number from the receiving host and <IAM><version> ('I AM') supplies that number.

3B The messages would probably be initially used in a 'can I talk to you?' sense or not at all. Eventually, it would take on a 'what can you do?' meaning. Accordingly, the <version> field should be large (32 bits?) for expansion.