Inter-Office Memorandum

To	Communicaton Protocols	Date	November 12, 1980
From	David Boggs	Location	Palo Alto
Subject	CopyDisk Protocol (Version 3)	Organization	Parc

XEROX

Filed on: <Pup>CopyDisk.bravo, -.press

The CopyDisk protocol is a means for making a bit-for-bit copy of a disk through the network. The first version of the protocol was designed and implemented in the summer of 1976. The protocol was redesigned during the summer of 1977 to handle Trident disks, and extended during the spring of 1980 to speak to IFSs and (eventually) tape servers.

Overview

There are two parties to a disk copy: a *server* which listens for connection requests at a well-known socket, and a *user* which initiates a connection to a server and thereafter controls the server by commands. The direction of data transfer over the connection is determined by the user and is orthogonal to who is user and who is server.

Commands and responses take the form of blocks with *type*, *length*, and *data* fields. The format of the data field is determined by the block type and sometimes the disk type. A Byte Stream Protocol connection is used since blocks can exceed the maximum size of a Pup, but a form of Reliable Packet Protocol which preserved the block boundries would be more suitable.

This protocol may be used to copy any type of disk. The data within some commands are intended for consumption by the concerned disk driver modules and are transported without interpretation by the protocol. CopyDisk programs need not implement all disk types; the protocol includes a mechanism for negotiating the type of disk to copy.

Two guiding principles were used in the design of this protocol: 1) the server should be passive, responding to commands from the user, but never initiating action on its own, and 2) the user should be responsible for checking the reasonableness of operations while the server just does what it is told, protecting itself only against self-inflicted damage. The similarity of the CopyDisk protocol to the File Transfer Protocol is intentional. In this design I am experimenting with some variations which are likely to show up in the next major revision to FTP.

The Protocol

The details of command formats are given in following sections. This section describes the order of commands and their general semantics. Copying a disk involves the following steps:

Connection establishment The user opens a BSP connection with the server and sends a [Version] command with a protocol version number and herald text as data. The server sends a [Version] response with its version and herald text. If the versions are not the same, the user should close the connection.

Unit negotiation The user sends a [SendDiskParams] command with a unit name as data. The server attempts to parse the unit name, and if successful sends a [HereAreDiskParams] response with the disk parameters as data. If the server doesn't understand the unit name, or is unable to reference

the unit (because, for example, it is not ready), it sends a [No] response.

This step may be repeated several times as the user pokes around trying to find the right disk. The server should remember the last disk for which parameters were sent. Subsequent transfer commands apply to that disk.

Data Transfer If the server responded with disk parameters, the user may now send a transfer command with transfer parameters as data. The transfer parameters are a message from the user's disk driver to the server's disk driver describing the sectors to be transferred. The server should respond [Yes] or [No]. If the response was [Yes], disk pages begin flowing; user-to-server if the transfer command was [StoreDisk], or server-to-user if the command was [RetrieveDisk]. After the source transmits the last page, it should send an [EndOfTransfer] command.

The data transfer step may be repeated several times and the direction in which data flows may not be the same each time (for example it may be [StoreDisk] the first time and [RetrieveDisk] the second time). These commands always refer to the disk last decribed by a [HereAreDiskParams] response from the server. The format and ordering of disk pages within a transfer is private to the disk drivers involved.

Error Reporting The user sends a [SendErrors] command to the server who responds with [HereAreErrors] with errors as data. The format is private to the disk drivers involved.

Command and Response Descriptions

Commands and responses are sent in blocks with a type, length, and (possibly empty) data field. Implementors are cautioned to flush the byte stream in situations where no other data follows a block that must be received by the other party to prevent a deadlock.

[Version] <code> <string>
Identifies the sender's protocol version. <Code> is the protocol version, and <string> is arbitrary text. The CopyDisk user should issue this command immediately upon opening a connection, and the server should respond with a [Version] reply. It is the user's responsibility to check for compatible protocol versions. I recommend that the herald text be displayed.

[Login] <userName> <userPsw> <connName> <connPsw>
Sent by the user to authenticate access to the server. The four strings are in Bcpl string format, and begin on word boundaries (so there will be a byte of garbage following a string with an even number of characters). A word of zeros is a legal Bcpl string with zero characters and should be used where no string is available. The server should respond with [Yes] or [No]; inclusion of the proper [No] subcode is strongly recommended since the user program can then automatically prompt its client to correct the problem.

[SendDiskParamsW] <string>
[SendDiskParamsR] <string>
Sent by the user, this command requests the server to supply disk parameters for the unit corresponding to <string>. SendDiskParamsW means that the user intends to write on the specified unit. This is necessary for servers such as IFSs which must verify that the user has write access to the file, or can create it if it doesn't exist. The server should respond with [HereAreDiskParams] or [No].

[HereAreDiskParams] <disk parameters>
Sent by the server in response to [SendDiskParams] to describe the disk mentioned in the previous [SendDiskParams] command. Sent by the user just before a [StoreDisk] command. This is necessary for servers such as IFSs which must store the parameters in the file so that they can be returned in response to a [SendDiskParams] command during retrieval. Servers that copy to a real disk whose shape can be determined may ignore these.

[StoreDisk] <transfer parameters>
Sent by the user to initiate a data transfer from user to server. The server should respond [Yes] or [No]. The <transfer parameters> are private format and describe the pages to be transferred in the

jargon of the concerned disk drivers.

[RetrieveDisk] <transfer parameters>
Sent by the user to initiate a data transfer from server to user. The server should respond [Yes] or [No]. The <transfer parameters> are private format and describe the pages to be transferred in the jargon of the concerned disk drivers.

[HereIsDiskPage] <page>
Transfers a page in either direction. The format of <page> is private to the disk drivers participating in the current transfer.

[EndOfTransfer] (no data)
Sent by the source of disk pages to mark the end of a transfer. If the destination disk driver encounters an [EndOfTransfer] before the last page (which it knows from the transfer parameters), then the transfer was aborted.

[SendErrors] (no data)
Sent by the user, it commands the server to report any errors that occurred during the last data transfer. The data field of this command is empty. The server should respond with [HereAreErrors]. [SendErrors] is legal after a data transfer command and up to the next [SendDiskParams] command.

[HereAreErrors] <errors>
Sent by the server in response to [SendErrors]. The format of <errors> is private to the disk type of the last data transfer command.

[Yes] <code> <string>
A positive response. <Code> may supply additional machine-readable information of possible interest to the receiver, or it may be zero; however the meaning of [Yes] in terms of the protocol is determined by the context and not the <code>. <String> is arbitrary text.

[No] <code> <string>
A negative response. <Code> may supply additional machine-readable information of possible interest to the receiver, or it may be zero; however the meaning of [No] in terms of the protocol is determined by the context and not the <code>. <String> is arbitrary text which I recommend be displayed when received.

[Comment] <string>
Used to supply commentary, indicate non-fatal errors, etc. <String> is arbitrary text which should be displayed for the human user. This command is a no-op with respect to protocol interactions and may occur in any context.

An Example

In the following example, the user wishes to overwrite the server's DP0, and then check that the bits arrived safely by reading it back (presumably comparing the incoming copy against the original).

User: [Version] <3> "CopyDisk User 3.00, April 20, 1980″
Server: [Version] <3> "CopyDisk Server 3.00, April 20, 1980
User checks compatibility of protocol and possibly closes connection
User: [SendDiskParamsW] "DP0"
Server: [HereAreDiskParams] <disk parameters>
or [No] <1> "Unit not ready"
If the server sent parameters, the user checks their consistancy
User: [HereAreDiskParams] <disk parameters>
User: [StoreDisk] <transfer parameters>
Server: [Yes] <0> "Ready to overwrite DP0"
or [No] <2> "Overwriting DP0 is not allowed"
If the server responded [Yes] then...
User: [HereIsDiskPage] <disk page>

. . .
User: [HereIsDiskPage] <disk page>
User: [EndOfTransfer]
Copy is complete now. If the user wishes to check it then...
User: [HereAreDiskParams] <disk parameters>
User: [RetrieveDisk] <transfer parameters>
Server: [Yes] <0> "Sending DP0"
Server: [HereIsDiskPage] <disk page>
. . .
Server: [HereIsDiskPage] <disk page>
Server: [EndOfTransfer]
Finally the user tells the server to report any errors
User: [SendErrors]
Server: [HereAreErrors] <errors>

Public Block Formats

All command blocks have the same header to permit handling by agents without knowledge of their contents:

word 0:	length in words including this one
word 1:	block type

The format of the [SendDiskParams] command is:

words 0-1:	standard header
remainder:	Bcpl-format unit name string

The unit name should conform to the unit name conventions of the server. This specification recommends a unit name syntax for each disk type specified.

The format of the [HereAreDiskParams] response is part public and part private. The disk type is in a fixed place so that, for example, a TFS disk driver can detect receipt of BFS parameters and refuse the copy.

words 0-1:	standard header
word 2:	registered disk type (see below)
remainder:	private format

An IFS or tape server should respond with a disk type of 0 when asked for disk parameters by a user who intends to write; users should interpret this wildcard disk type as compatible with all real disk types.

Block types [Yes], [No], and [Version] have a common format:

word 0-1:	standard header
word 2:	subcode
remainder:	Bcpl-format string

The subcode in [Yes] and [No] commands may contain a machine-readable argument to facilitate mechanical handling of exceptional conditions. Zero should be used in the absense of any code. The subcode in a [Version] command is the protocol version.

Alto Diablo (BFS) Private Formats

These formats are used by disk drivers which copy Diablo disks formated as Alto file systems. I recommend that unit ’n’ be referred to as "DPn". Sectors are sent in increasing real disk address order during a data transfer.

The format of the disk parameters in a [HereAreDiskParams] response is:

words 0-1:	standard header
word 2:	12B (registered disk type)
word 3:	# of cylinders
word 4:	# of heads
word 5:	# of sectors
word 6:	# of disks

The format of the transfer parameters in [RetrieveDisk] and [StoreDisk] commands is:

words 0-1:	standard header
words 2:	first real disk address
words 3:	last real disk address

The format of a [HereIsDiskPage] command is:

words 0-1:	standard header
words 2-3:	header record
words 4-11:	label record
words 12-267:	data record

Free pages (those with -1 in the serial and version number fields of the label record) may omit the data record, reducing the size of IFS files and speeding network transfers. I recommend that pages which are marked incorrigable (even if CopyDisk can read them without errors) and pages which give unrecoverable read errors be transmitted as free pages.

The format of the errors in a [HereAreErrors] response is:

words 0-1:	standard header
word 2:	number of hard errors
word 3:	number of soft errors

Alto Trident (TFS) Private Formats

These formats are used by disk drivers that copy Trident disks formated as Alto file systems (this includes Interim File Systems). Trident disks with different formatting (such as Juniper disks) should be treated separately. I recommend that unit 'n' be referred to as "TPn". Sectors are sent in increasing real disk address order during a data transfer.

The format of the disk parameters in a [HereAreDiskParams] response is:

words 0-1:	standard header
word 2:	2 (registered disk type)
word 3:	# of cylinders
word 4:	# of heads
word 5:	# of sectors

The format of the transfer parameters in [RetrieveDisk] and [StoreDisk] commands is:

words 0-1:	standard header
words 2-3:	first real disk address
words 4-5:	last real disk address

Real page 0 should not be copied, so as not to overwrite the destination disk's bad page list. CopyDisk should verify that the destination disk has a well-formed bad page list, and if not, an empty one should be created.

The format of a [HereIsDiskPage] command is:

words 0-1:	standard header
words 2-3:	real disk address

words 4-13:	label record
words 14-1037:	data record

Free pages (those with -1 in the serial and version number fields of the label record) may omit the data record, reducing the size of IFS files and speeding network transfers. I recommend that pages which are marked incorrigable (even if CopyDisk can read them without errors) and pages which give unrecoverable read errors be transmitted as free pages.

The format of the errors in a [HereAreErrors] response is:

words 0-1:	standard header
word 2:	number of hard errors
word 3:	number of soft errors

Opcode Assignments

All numbers are octal.

The well-known socket for a CopyDisk server is 25.

The following block types are assigned:

1	[Version]
2	[SendDiskParamsR]
3	[HereAreDiskParams]
4	[StoreDisk]
5	[RetrieveDisk]
6	[HereIsDiskPage]
7	[EndOfTransfer]
10	[SendErrors]
11	[HereAreErrors]
12	[No]
13	[Yes]
14	[Comment]
15	[Login]
16	[SendDiskParamsW]

The following [No] codes are assigned:

1	Unit not ready
2	Unit write protected (hardware write-protect)
3	Overwrite not allowed (software write-protect)
4	Unknown command
20	Illegal user name
21	Illegal or incorrect user password
23	Illegal connect name
24	Illegal or incorrect connect password

The following disk types are assigned:

12	Diablo disk with standard BFS formatting
2	Trident disk with standard TFS formatting
3	Trident disk with Juniper formatting
0	Wildcard (see above)

Revision History

June 21, 1977
Draft release for comments.

July 22, 1977
Changed TFS protocol: transfer parameters now contains real disk addresses and disk pages now include the header record.

December 18, 1977
Changed BFS protocol: transfer parameters now contains real disk addresses and disk pages now include the header record.

April 20, 1980
Incompatible protocol extentions to work with IFSs and (eventually) tape servers. BFS disk parameters now contain number of heads and sectors.

November 12, 1980
Changed BFS protocol: BFS disk parameters now contain number of disks.