Network Working Group
Request for Comments: 937

M. Butler
J. Postel
D. Chase
J. Goldberger
J. K. Reynolds
ISI

Obsoletes: RFC 918

February 1985

POST OFFICE PROTOCOL - VERSION 2

Status of this Memo

This RFC suggests a simple method for workstations to dynamically access mail from a mailbox server. This RFC specifies a proposed protocol for the ARPA-Internet community, and requests discussion and suggestions for improvement. This memo is a revision of RFC 918. Distribution of this memo is unlimited.

Introduction

The intent of the Post Office Protocol Version 2 (POP2) is to allow a user's workstation to access mail from a mailbox server. It is expected that mail will be posted from the workstation to the mailbox server via the Simple Mail Transfer Protocol (SMTP). For further information see RFC-821 [1] and RFC-822 [2].

This protocol assumes a reliable data stream such as provided by TCP or any similar protocol. When TCP is used, the POP2 server listens on port 109 [4].

System Model and Philosophy

While we view the workstation as an Internet host in the sense that it implements IP, we do not expect the workstation to contain the user's mailbox. We expect the mailbox to be on a server machine.

We believe it is important for the mailbox to be on an "always up" machine and that a workstation may be frequently powered down, or otherwise unavailable as an SMTP server.

POP2 is designed for an environment of workstations and servers on a low-delay, high-throughput, local networks (such as Ethernets). POP2 may be useful in other environments as well, but if the environment is substantially different, a different division of labor between the client and server may be appropriate, and a different protocol required.

Suppose the user's real name is John Smith, the user's machine is called FIDO, and that the mailbox server is called DOG-HOUSE. Then

we expect the user's mail to be addressed to JSmith@DOG-HOUSE.ARPA (not JSmith@FIDO.ARPA).

That is, the destination of the mail is the mailbox on the server machine. The POP2 protocol and the workstation are merely a mechanism for viewing the messages in the mailbox.

The user is not tied to any particular workstation for accessing his mail. The workstation does not appear as any part of the mailbox address.

This is a very simple protocol. This is not a user interface. We expect that there is a program in the workstation that is friendly to the user. This protocol is not "user friendly". One basic rule of this protocol is "if anything goes wrong close the connection". Another basic rule is to have few options.

POP2 does not parse messages in any way. It does not analyze message headers (Date:, From:, To:, Cc:, or Subject:). POP2 simply transmits whole messages from a mailbox server to a client workstation.

The Protocol

The POP2 protocol is a sequence of commands and replies. The design draws from many previous protocols of the ARPA-Internet community.

The server must be listening for a connection. When a connection is opened the server sends a greeting message and waits for commands. When commands are received the server acts on them and responds with replies.

The client opens a connection, waits for the greeting, then sends the HELO command with the user name and password arguments to establish authorization to access mailboxes. The server returns the number of messages in the default mailbox.

The client may read the default mailbox associated with the user name or may select another mailbox by using the FOLD command. The server returns the number of messages in the mailbox selected.

The client begins a message reading transaction with a READ command. The read command may optionally indicate which message number to read, the default is the current message (incremented when a message is read and set to one when a new folder is selected). The server returns the number of characters in the message.

The client asks for the content of the message to be sent with the RETR command. The server sends the message data.

When all the data has been received the client sends an acknowledgment command. This is one of ACKS, ACKD, and NACK.

ACKS means "I've received the message successfully and please keep it in the mailbox".

ACKD means "I've received the message successfully and please delete it from the mailbox".

NACK means "I did not receive the message and please keep it in the mailbox".

In the case of ACKS or ACKD the server increments the current message indicator. In the case of NACK the current message indicator stays the same.

In all cases the server returns the number of characters in the (now) current message.

The client terminates the session with the QUIT command. The server returns an ok.

The Normal Scenario

```
        Client                    Server
        ------                    ------
                                  Wait for Connection
   Open Connection  -->
                        <--  + POP2 Server Ready
                                  Wait for Command
   HELO Fred Secret -->
                        <--  #13 messages for you
                                  Wait for Command
   READ 13          -->
                        <--  =537 characters in that message
                                  Wait for Command
   RETR             -->
                        <--  (send the message data)
                                  Wait for Command
   ACKS             -->
                        <--  =0 no more messages
                                  Wait for Command
   QUIT             -->
                        <--  + OK
   Close connection --> <--  Close connection
                                  Wait for Connection (go back to start)
```

Conventions

Arguments

These arguments have system specific definitions.

user - A login account name.

password - The password for the login account.

mailbox - A mailbox name (also called a mail folder).

Default Mailboxes

TOPS-20

MAIL.TXT.1 - from login directory

UNIX

both
/usr/spool/mail/user
and
/usr/user/Mail/inbox/*

where "user" is the user value supplied in the HELO command.

End of Line

End of Line is Carriage Return (CR) followed by Line Feed (LF). This sequence is indicated by "CRLF" in this document. This end of line convention must be used for commands and replies.

Message Length

The reply to the READ command or an acknowledgment command (ACKS, ACKD, NACK) is the length (a character count) of the next message to be transmitted. This includes all the characters in the data transmitted. CRLF counts as two characters. A length of zero means the message does not exist or is empty. A request to transmit a message of zero length will result in the server closing the connection. The message is transmitted in the standard internet format described in RFC-822 [2] and NVT-ASCII. This may be different from the storage format and may make computing the message length from the stored message non-trivial.

Message Numbers

The reply to the HELO and FOLD commands is a count of the number of messages in a the selected mailbox. The READ command has a message number as an optional argument. These numbers are decimal, start at one, and computed with respect to the current mailbox. That is, the first message in a mailbox is message number 1.

Numbers

All numbers in this memo and protocol are decimal.

Quoting

In a few cases, there may be a need to have a special character in an argument (user, password, or mailbox) that is not allowed by the syntax. For example, a space in a password. To allow for this, a quoting convention is defined. Unfortunately, such quoting conventions "use up" another otherwise uninteresting character. In this protocol the back slash "\" is used as the quote character. To include a space in an argument the two character sequence "back-slash, space" is transmitted. To include a back-slash in an argument the two character sequence "back-slash, back-slash" is transmitted. This quoting convention is used in the command arguments only, it is not used in the mail data transmitted in response to a RETR command.

Reply Strings

The first character is required to be as specified (i.e., "+", "-", "=", "#"). The optional strings that follow can be whatever the implementer thinks is appropriate.

Definitions of Commands and Replies

Summary of Commands and Replies

Commands	Replies
HELO user password	+ OK
FOLD mailbox	- Error
READ [n]	#xxx
RETR	=yyy
ACKS	
ACKD	
NACK	
QUIT	

Commands

HELO user password

The Hello command identifies the user to the server and carries the password authenticating this user. This information is used by the server to control access to the mailboxes. The Hello command is the "HELO" keyword, followed by the user argument, followed by the password argument, followed by CRLF.

Possible responses:

"#nnn"

where nnn is the number of messages in the default mailbox,"

"- error report" and Close the connection.

FOLD mailbox

The Folder command selects another mailbox or mail folder. The server must check that the user is permitted read access to this mailbox. If the mailbox is empty or does not exist, the number of messages reported is zero. The Folder command is the "FOLD" keyword, followed by the mailbox argument, followed by CRLF.

Possible responses:

"#nnn"

where nnn is the number of messages in this mailbox.

READ [nnn]

The Read command begins a message reading transaction. If the Read command is given without an argument the current message is implied (the current message indicator is incremented by the ACKS or ACKD commands). If an argument is used with the Read command it is the message number to be read, and this command sets the current message indicator to that value. The server returns the count of characters in the message to be transmitted. If there is no message to be read, the count of zero is returned. If the message was previously deleted with the ACKD command, the count of zero is returned. The Read command is followed by the RETR command, the READ command, the FOLD command, or the QUIT command. Do not attempt to RETR a

message of zero characters. The Read command is the "READ" keyword, optionally followed by the message number argument, followed by CRLF.

Possible responses:

"=ccc"

where ccc is the number of characters in this message.

RETR

The Retrieve command confirms that the client is ready to receive the mail data. It must be followed by an acknowledgment command. The server will close the connection if asked to transmit a message of zero characters (i.e., transmit a non-existent message). The message is transmitted according to the Internet mail format standard RFC-822 [2] in NVT-ASCII. The Retrieve command is the "RETR" keyword, followed by CRLF.

Possible responses:

the message data

Close the connection

ACKS

The Acknowledge and Save command confirms that the client has received and accepted the message. The ACKS command ends the message reading transaction. The message is kept in the mailbox. The current message indicator is incremented. The server returns the count of characters in the now current message to be transmitted. If there is no message to be read or the message is marked deleted, the count of zero is returned. The Acknowledge and Save command is the "ACKS" keyword, followed by CRLF.

Possible responses:

"=ccc"

where ccc is the number of characters in the next message.

ACKD

The Acknowledge and Delete command confirms that the client has received and accepted the message. The ACKD command ends the message reading transaction. If the user is authorized to have write access to the mailbox, the message is deleted from the mailbox. Actually, the message is only marked for deletion. The actual change is made when the mailbox is released at the end of the session or when the client selects another mailbox with the FOLD command. The messages are not renumbered until the mailbox is released. If the user does not have write access to the mailbox no change is made to the mailbox. The response is the same whether or not the message was actually deleted. The current message indicator is incremented. The server returns the count of characters in the now current message to be transmitted. If there is no message to be read or the message is marked deleted, the count of zero is returned. The Acknowledge and Delete command is the "ACKD" keyword, followed by CRLF.

Possible responses:

"=ccc"

where ccc is the number of characters in the next message.

NACK

The Negative Acknowledge command reports that the client did not receive the message. The NACK command ends the message reading transaction. The message is kept in the mailbox. The current message indicator remains the same. The server returns the count of characters in the current message. Since the count to be returned is for the message just transmitted it the message must exist and not be marked deleted, and the count must be positive (non-zero). The Negative Acknowledge command is the "NACK" keyword, followed by CRLF.

Possible responses:

"=ccc"

where ccc is the number of characters in this message.

QUIT

The Quit command indicates the client is done with the session. The server sends an OK response and then closes the connection. The Quit command is the "QUIT" keyword, followed by CRLF.

Possible responses:

"+ OK" and Close the connection

Replies

Greeting

The greeting is sent by the server as soon as the connection is established. The greeting is a plus sign, followed by the protocol name ("POP2"), followed by the server host name, optionally followed by text, and ending with a CRLF.

+

The success or plus sign response indicates successful completion of the operation specified in the command. The success response is a plus sign, optionally followed by text, and ending with a CRLF.

-

The failure or minus sign response indicates the failure of the operation specified in the command. The failure response is a minus sign, optionally followed by text, and ending with a CRLF.

=

The length or equal sign response tells the length in characters of the message referenced by the command. The length response is a equal sign, followed by a number, optionally followed by text, and ending with a CRLF.

#

The count or number sign response tells the number of messages in a folder or mailbox referenced by the command. The count response is a number sign, followed by a number, optionally followed by text, and ending with a CRLF.

Timeouts

In any protocol of this type there have to be timeouts. Neither side wants to get stuck waiting forever for the other side (particularly is the other side has gone crazy or crashed).

The client expects a reply to a command fairly quickly and so should have a short timeout for this. This timeout is called T1.

For some servers, it may take some processing to compute the number of messages in a mailbox, or the length of a message, or to reformat a stored message for transmission, so this time out has to allow for such processing time. Also care must be taken not to timeout waiting for the completion of a RETR reply while a long message is in fact being transfered.

The server expects the session to progress with some but not excessive delay between commands and so should have a long timeout waiting for the next command. This time out is T2.

One model of use of this protocol is that any number of different types of clients can be built with different ways of interacting with the human user and the server, but still expecting the client to open the connection to the server, present a sequence of commands, and close the connection, without waiting for intervention by the human user. With such client implementations, it is reasonable for the server to have a fairly small value for timeout T2.

On the other hand, one could easily have the client be very human user directed with the user making decisions between commands. This would cause arbitrary delays between client commands to the server, and require the value of timeout T2 to be quite large.

Implementation Discussion

Comments on a Server on TOPS-20

On TOPS-20, a mailbox is a single file. New messages are appended to the file. There is a separator line between messages.

The tricky part of implementing a POP2 server on TOPS-20 is to provide for deleting messages. This only has to be done for the mailboxes (files) for which the user has write access. The problem is to avoid both (1) preventing other users from accessing or updating the mailbox for long periods, and (2) accidentally deleting a message the user has not seen.

One suggestion is as follows:

> When a mailbox is first selected, if the user has write access, rename the mailbox file to some temporary name. Thus new messages will be placed in a new instance of the mailbox file. Conduct all POP2 operation on the temporary mailbox file (including deleting messages). When the POP2 session is over or another mailbox is selected, prepend any messages left undeleted in the temporary file to the new instance of the mailbox file.

Sizes

The maximum length of a command line is 512 characters (including the command word and the CRLF).

The maximum length of a reply line is 512 characters (including the success indicator (+, -, =, #) and the CRLF).

The maximum length of a text line is 1000 characters (including CRLF).

ISI has developed a POP2 server for TOPS-20 and for Berkeley 4.2 Unix, and a POP2 client for an IBM-PC and for Berkeley 4.2 Unix.

Extensions Not Supported

POP2 does not examine the internal data of messages. In particular, the server does not parse message headers.

The server doesn't have any state information (i.e., it doesn't know from one session to the next what has happened). For example, the server doesn't know which messages were received since the last time the user used POP2, so it can't send just the "new" messages.

Examples

Example 1:

```
        Client                    Server
        ------                    ------
                                Wait for connection
   Open connection  -->
                        <--  + POP2 USC-ISIF.ARPA Server
   HELO POSTEL SECRET -->
                        <--  #2 messages in your mailbox
   READ             -->
                        <--  =537 characters in message 1
   RETR             -->
                        <--  [data of message 1]
   ACKD             -->
                        <--  =234 characters in message 2
   RETR             -->
                        <--  [data of message 2]
   ACKD             -->
                        <--  =0 no more messages
   QUIT             -->
                        <--  + OK, bye, bye
   Close connection --> <--  Close connection
                             Go back to start
```

Example 2:

```
        Client                    Server
        ------                    ------
                                Wait for connection
   Open connection  -->
                           <--  + POP2 ISI-VAXA.ARPA server here
   HELO smith secret -->
                           <--  #35 messages
   FOLD /usr/spool/mail/smith -->
                           <--  #27 messages
   READ  27          -->
                           <--  =10123 characters in that message
   RETR              -->
                           <--  [data of message 27]
   ACKS              -->
                           <--  =0 no more messages
   QUIT              -->
                           <--  + bye, call again sometime.
   Close connection --> <--  Close connection
                                Go back to start
```

Example 3:

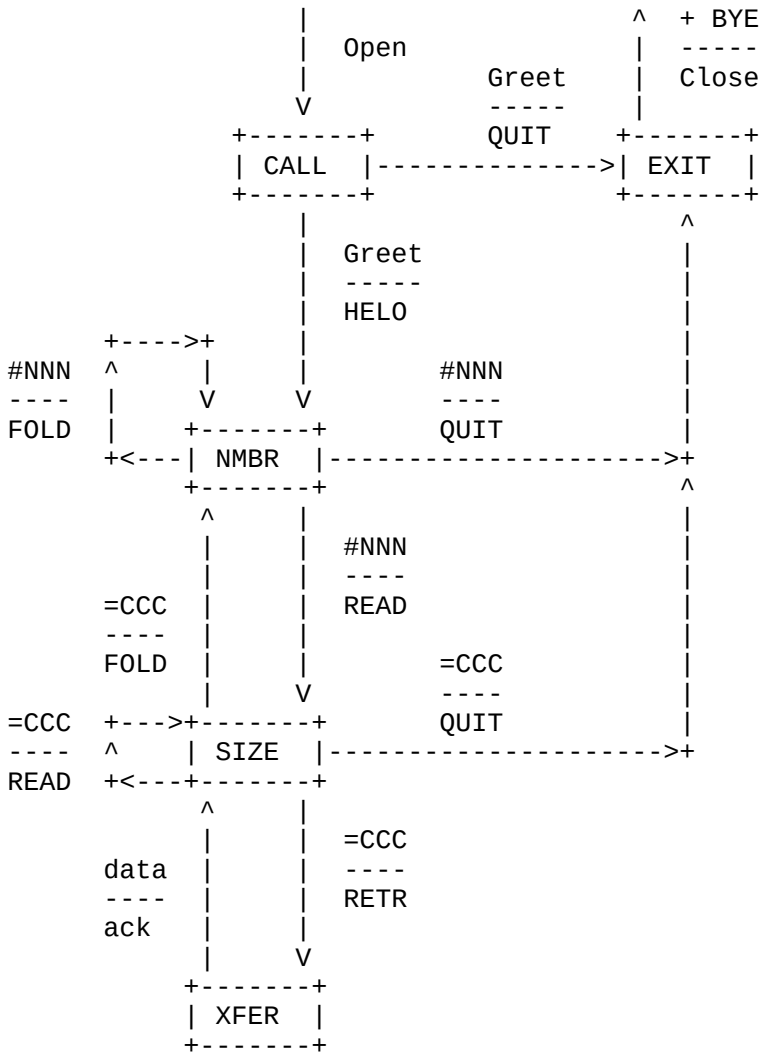
```
        Client                    Server
        ------                    ------
                                Wait for connection
   Open connection  -->
                           <--  + POP2 ISI-VAXA.ARPA server here
   HELO Jones secret -->
                           <--  #0 messages
   READ              -->
                           <--  Close connection
   Close connection -->
                                Go back to start
```

Formal Syntax

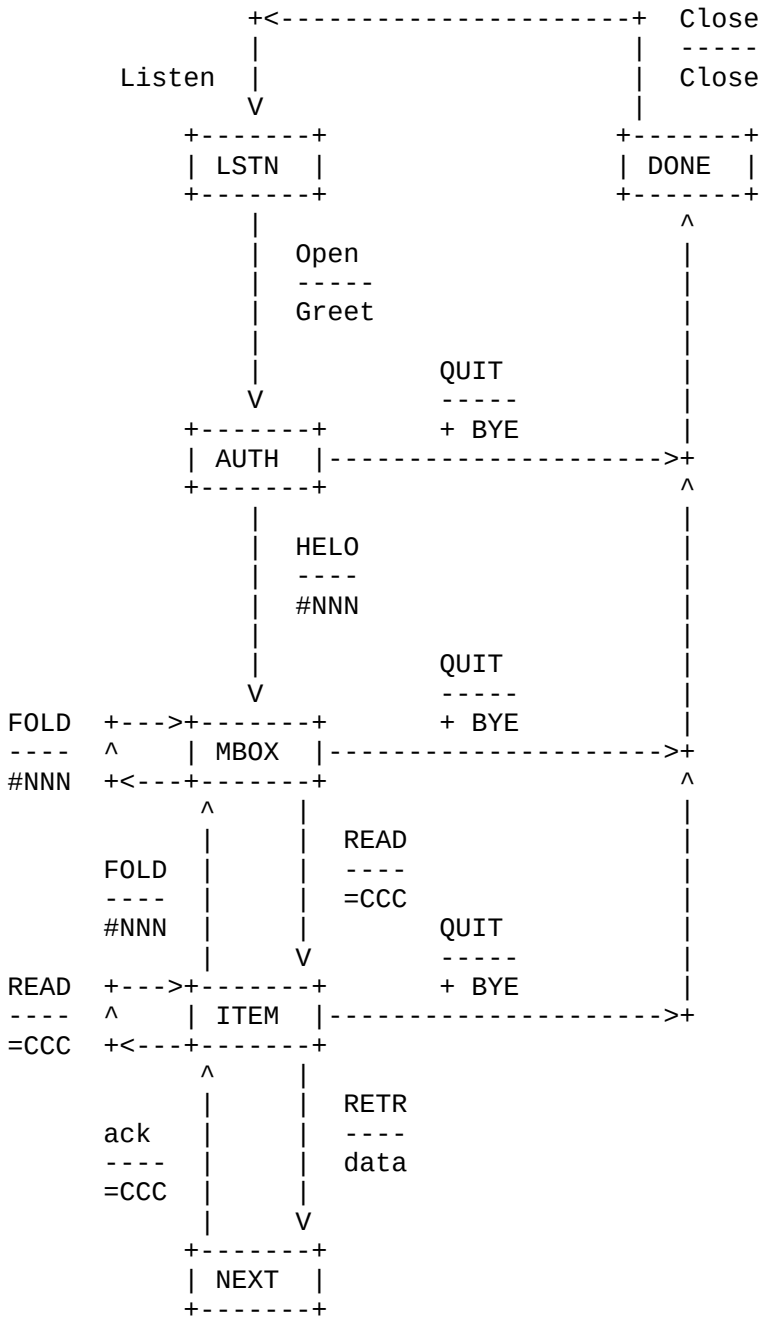
```
   <digit>    = 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9

   <letter>   = A | B | C | ... | Z
                a | b | c | ... | z

   <punct>    = ! | " | # | $ | % | & | ’ | ( | ) | * |
                + | , | - | / | : | < | = | > | ? | @ |
                [ | ] | ^ | _ | ‘ | { | | | } | ~

   <quote>    = \

   <any>      = any one of the 128 ASCII codes

   <CR>       = carriage return, code 10

   <LF>       = line feed, code 13

   <SP>       = space, code 32

   <CRLF>     = <CR> <LF>

   <print>    = <letter> | <digit> | <punct> | <quote> <any>

   <char>     = <print> | <SP>

   <word>     = <print> | <print> <word>

   <string>   = <char> | <char> <string>

   <ld>       = <letter> | <digit>

   <ldh>      = <letter> | <digit> | -

   <ldhs>     = <ldh> | <ldh> <ldhs>

   <name>     = <letter> [ [ <ldhs> ] <ld> ]

   <host>     =  <name> | <name> . <host>

   <user>     = <word>

   <password> = <word>

   <mailbox>  = <string>

   <number>   = <digit> | <digit> <number>
```

```
   <helo>     = HELO <SP> <user> <SP> <password> <CRLF>

   <fold>     = FOLD <SP> <mailbox> <CRLF>

   <read>     = READ [<SP> <number>] <CRLF>

   <retr>     = RETR <CRLF>

   <acks>     = ACKS <CRLF>

   <ackd>     = ACKD <CRLF>

   <nack>     = NACK <CRLF>

   <quit>     = QUIT <CRLF>

   <ok>       = + [<SP> <string>] <CRLF>

   <err>      = - [<SP> <string>] <CRLF>

   <count>    = # <number> [<SP> <string>] <CRLF>

   <greet>    = + <SP> POP2 <SP> <host> [<SP> <string>] <CRLF>

   <length>   = = <number> [<SP> <string>] <CRLF>

   <command>  = <helo> | <fold> | <read> | <retr> |
                <acks> | <ackd> | <nack> | <quit>

   <reply>    = <ok> | <err> | <count> | <length> | <greet>
```

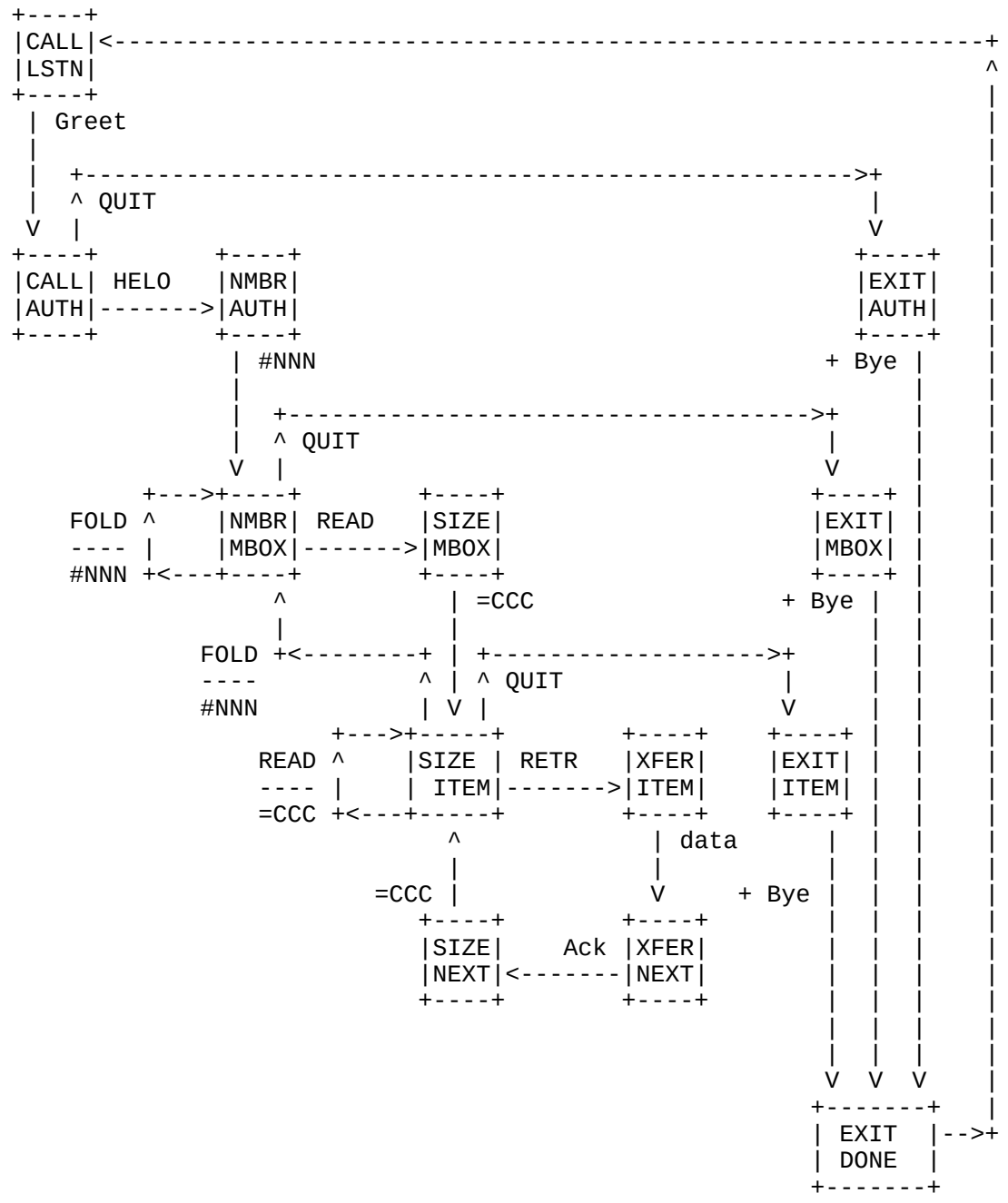
Client State Diagram



Server State Diagram



Combined Flow Diagram



Client Decision Table

	STATE				
INPUT	CALL	NMBR	SIZE	XFER	EXIT
Greet	2	1	1	1	6
#NNN	1	3	1	1	6
=CCC	1	1	4	1	6
data	1	1	1	5	6
+ Bye	1	1	1	1	6
Close	1	1	1	1	6
other	1	1	1	1	6
Timeout	1	1	1	1	6

Actions:

1. This is garbage. Send "QUIT", and go to EXIT state.

2. (a) If the greeting is right then send "HELO"
   and go to NMBR state,
   (b) Else send "QUIT" and go to EXIT state.

3. (a) If user wants this folder and NNN > 0
   then send "READ" and go to SIZE state,
   (b) If user wants a this folder and NNN = 0
   then send "QUIT" and go to EXIT state,
   (c) If user wants a different folder
   then send "FOLD" and go to NMBR state.

4. (a) If user wants this message and CCC > 0
   then send "RETR" and go to XFER state,
   (b) If user wants a this message and CCC = 0
   then send "QUIT" and go to EXIT state,
   (c) If user wants a different message
   then send "READ" and go to SIZE state.

5. (a) If user wants this message kept
   then send "ACKS" and go to SIZE state,
   (b) If user wants a this message deleted
   then send "ACKD" and go to SIZE state,
   (c) If user wants a this message again
   then send "NACK" and go to SIZE state.

6. Close the connection.

Server Decision Table

	STATE					
INPUT	LSTN	AUTH	MBOX	ITEM	NEXT	DONE
Open	2	1	1	1	1	1
HELO	1	3	1	1	1	1
FOLD	1	1	5	5	1	1
READ	1	1	6	6	1	1
RETR	1	1	1	7	1	1
ACKS	1	1	1	1	8	1
ACKD	1	1	1	1	8	1
NACK	1	1	1	1	8	1
QUIT	1	4	4	4	1	1
Close	1	1	1	1	1	9
other	1	1	1	1	1	1
Timeout		1	1	1	1	1

Actions:

1. This is garbage. Send "- error", and Close the connection.

2. Send the greeting. Go to AUTH state.

3. (a) If authorized user then send "#NNN" and go tp MBOX state,
   (b) Else send "- error" and Close the connection.

4. Send "+ Bye" and go to DONE state.

5. Send "+NNN" and go to MBOX state.

6. Send "=CCC" and go to ITEM state.

7. If message exists then send the data and got to NEXT state,
   Else Close the connection.

8. Do what ACKS/ACKD/NACK require and go to ITEM state.

9. Close the connection.

Acknowledgment

We would like to acknowledge the helpful comments that we received on the first version of POP described in RFC 918, and the draft of POP2 distributed to interested parties.

References

[1] Postel, J., "Simple Mail Transfer Protocol", RFC 821, USC/Information Sciences Institute, August 1982.

[2] Crocker, D., "Standard for the Format of ARPA-Internet Text Messages", RFC 822, University of Delaware, August 1982.

[3] Reynolds, J.K., "Post Office Protocol", RFC 918, USC/Information Sciences Institute, October 1984.

[4] Reynolds, J.K., and J. Postel, "Assigned Numbers", RFC 923, USC/Information Sciences Institute, October 1984.