THE myftp PROTOCOL

Daniel Dobson & Jarryd Kaczmarczyk

Introduction

myftp is a simple network protocol used for communication between a client and a server and to transfer files. The protocol functions on top of the underlying transport layer, the Transmission Control Protocol (TCP) [1]. This paper assumes knowledge of TCP and serves as the specification of the myftp protocol.

Overview of the Protocol

The myftp protocol is designed to allow a remote connection that permits access and navigation of file systems and the read and write of files using TCP from/to a remote server. The protocol does not provide an authentication process and passes 8-bit bytes of data. The connection established via TCP uses the default server listening port 41147. The following commands are available as part of the specification:

- pwd
  Display the current directory of the server serving the client.

- dir
  Display the file names under the current directory of the server serving the client.

- cd <pathname>
  Change the current directory of the server that is serving the client.

- get <filename>
  Retrieve the named file from the current directory of the remote server serving the client.

- put <filename>
  Send the named file from the current directory of the client to the current directory of the remote server serving the client.

Each request sent through the myftp protocol is initiated by the client. In all instances, the server is the receiver of requests.

Initial Connection Protocol

A TCP connection stream is first established between the client and the remote server. Requests can then be sent to the remote server via the established connection. The command requests will specify the nature of the operation to be performed and the server should “listen” to such requests on the specified port 41147. It is the responsibility of the client to terminate the established connection to the remote server when no further requests are required.

myftp Packets & Messages

The myftp header consists of a one byte opcode (an ASCII character), indicating the type of the packet as described in Table 1 below:

Table 1. Opcode Definitions

Opcode	Command
P	pwd
D	dir
C	cd
G	get (initial request)
H	get (accept file)
U	put (initial request)
V	put (accept file)

Depending on the opcode, an acknowledgement (ACK) packet may be sent/received from the client or the server. The message will contain a one byte opcode which is an ASCII character followed by a one byte acknowledgement code.

1 byte	1 byte
Opcode	Ack Code

Figure 1-1: ACK packet

For a client requesting a list of file names in the server's current working directory (dir), the ACK packets are acknowledged by the dir packets.

Client requesting the servers current working directory (pwd):

1 byte	2 bytes	n bytes	1 byte
Opcode	Length	Current Working Directory	0

Figure 1-2: pwd packet

- Client sends one message to the server with a one byte opcode which is ASCII character ‘P’.
- Once the server receives a message with opcode ‘P’, it responds with the following message:
  - one byte opcode which is ASCII character ‘P’ followed by,
  - two-byte integer value in two's complement and in network byte order, which represents the total length of the current working directory to be sent to the client followed by,
  - the sequence of ASCII characters representing the current working directory terminated by a zero byte.

Client requesting a list of file names in the servers current working directory (dir):

1 byte	2 bytes	n bytes	1 byte	1 byte
Opcode	Total Length	Filenames in Current Working Directory	0	Ack Code

Figure 1-3: dir packet

- Client sends one message to the server with a one byte opcode which is ASCII character ‘D’.
- Once the server receives a message with opcode ‘D’, it responds with the following message:
    - one byte opcode which is ASCII character ‘D’ followed by,
    - two-byte integer value in two's complement and in network byte order, which represents the total length of the file names (including the newline characters) to be sent to the client followed by,
    - the sequence of ASCII characters separated by a newline character representing each filename in the current working directory terminated by a zero byte followed by,
    - one byte acknowledgment code which is one of the following ASCII characters:
        - 0 - the server processed the request successfully.
        - 1 - the server did not process the request successfully.

Client request to change the servers current working directory (cd)

1 byte	2 bytes	n bytes	1 byte	1 byte
Opcode	Length	New Working Directory	0	Ack Code

Figure 1-4: cd packet

- Client sends one message to the server. The message format is given below:

    - one byte opcode which is ASCII character ‘C’ followed by,
    - two-byte integer value in two's complement and in network byte order, which represents the total length of the new working directory to be sent to the server followed by,
    - the sequence of ASCII characters representing the current working directory terminated by a zero byte.
- Once the server receives a message with opcode ‘C’, it responds with the following message:
    - one byte opcode which is ASCII character ‘C’ followed by,
    - one byte acknowledgment code which is one of the following ASCII characters:
        - 0 - the server has successfully changed the working directory.
        - 1 - the server cannot set the new working directory.

Client retrieving a file from the server (get):

1 byte	2 bytes	n bytes	1 byte
Opcode	Length	File	0

Figure 1-5: get packet

- Client sends one message to the server. The message format is given below:
    - one byte opcode which is ASCII character 'G' followed by
    - two-byte integer value in two's complement and in network byte order, which represents the length of the filename of the file to be retrieved from the servers current working directory followed by,
    - the sequence of ASCII characters representing the filename terminated by a zero byte.
- Once the server receives a message with opcode 'G', it responds with the following message:
    - one byte opcode which is ASCII character 'G' followed by

    - one byte acknowledgment code which is one of the following ASCII characters:
        - 0 - the server is ready to send the file
        - 1 - the server cannot send the file as the filename doesn't exist in the current working directory.
        - 2 - the server cannot send the file as it doesn't have permission to send.
        - 3 - the server cannot send the file due to other reasons.

- Once the client receives the 'G' message from the server and if the acknowledgment code is 0, it responds by sending the following message:
    - one byte opcode which is the ASCII character 'H' followed by,
    - one byte acknowledgement code which is one of the following ASCII characters:
        - 0 – client is ready to accept the file
        - 1 – client is not ready to accept the file
- Once the server receives the 'H' message from the server and if the acknowledgement code is 0, it responds by sending the following message:
    - one byte opcode which is the ASCII character 'H' followed by,
    - two-byte integer value in two's complement and in network byte order which represents the length of the file.
    - a sequence of N bytes which is the content of the file terminated by a zero byte.

Client sending a file to the server (put):

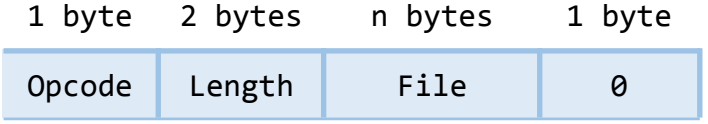

Figure 1-6: put packet

- Client sends one message to the server. The message format is given below:
  - one byte opcode which is ASCII character ‘U’ followed by,
  - two-byte integer value in two's complement and in network byte order, which represents the length of the filename of the file to be sent to the server followed by,
  - the sequence of ASCII characters representing the filename terminated by a zero byte.
- Once the server receives a message with opcode ‘U’, it responds with the following message:
  - one byte opcode which is ASCII character ‘V’ followed by,
  - one byte acknowledgment code which is one of the following ASCII characters:
    - 0 - the server is ready to accept the named file
    - 1 - the server cannot accept the file as there is a filename clash
    - 2 - the server cannot accept the file because it does not have permission to download the file.
    - 3 - the server cannot accept the file due to other reasons.
- Once the client receives the ‘V’ message from the server and if the acknowledgment code is 0, it responds by sending the following message:
  - one byte opcode which is the ASCII character ‘V’ followed by,
  - two-byte integer in two's complement and in network byte order which represents the length of the file followed by,
  - a sequence of N bytes which is the content of the file terminated by a zero byte.

References

[1] Postel, J. (ed.), "Transmission Control Protocol - DARPA Internet Program Protocol Specification", RFC 793, USC/Information Sciences Institute, September 1981.