RFC 768(N)

Nazmul Alam
IT Tralee
11 Nov 2016

Datagram-FTP

1.Introduction

A simple File Transfer Protocol based on RFC 768. This protocol is established to demonstrate my understanding of inter-process communication using User Datagram Protocol(UDP).

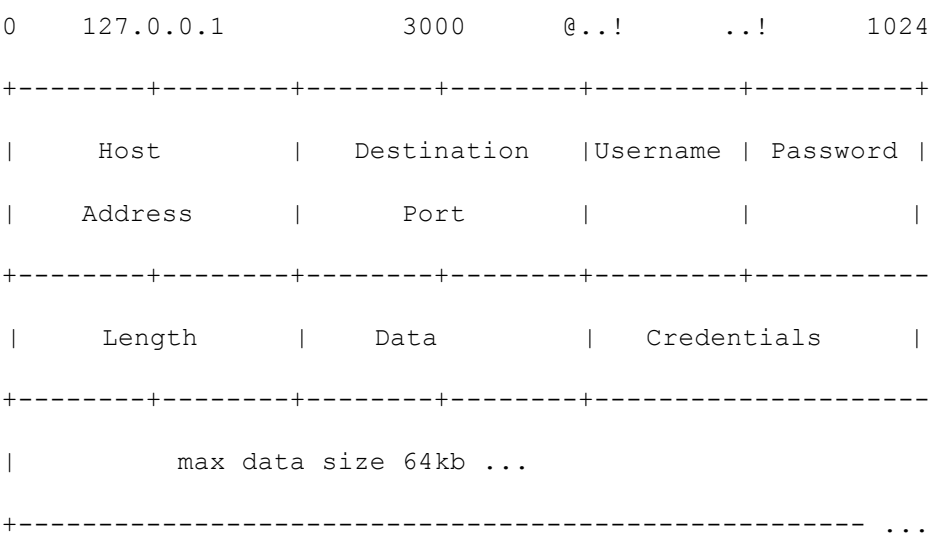
This protocol provides a procedure for application programs to send messages within the local network programs with a minimum of protocol mechanism.

2.Objectives

The objectives are:

- Establish an authentication mechanism for data transfers
- To specify a set of rules that must be observed by the client and the server during a session of a particular service
- Exchanging data within the Local Area Network

3.Data Format

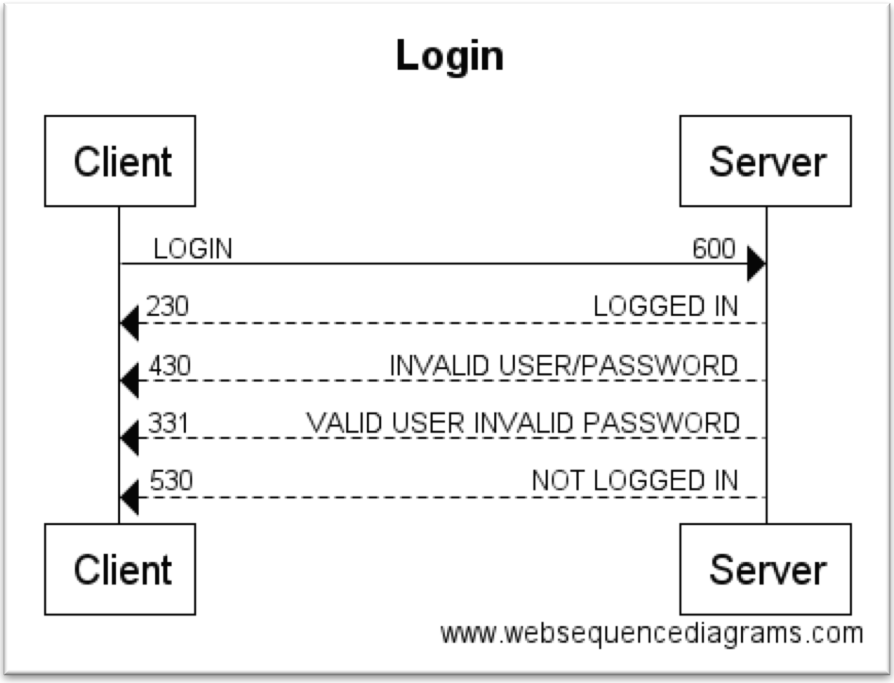


Data uploading and downloading format

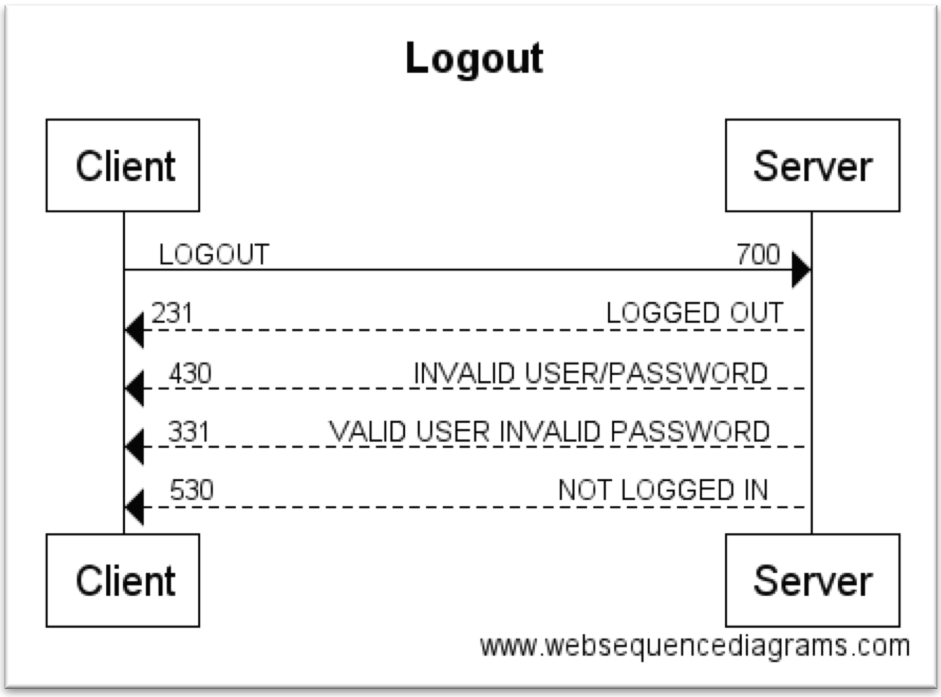
4. UML Sequence Diagrams

Basic sequence diagrams for the implemented protocol codes to describe the communication between the participating entities.

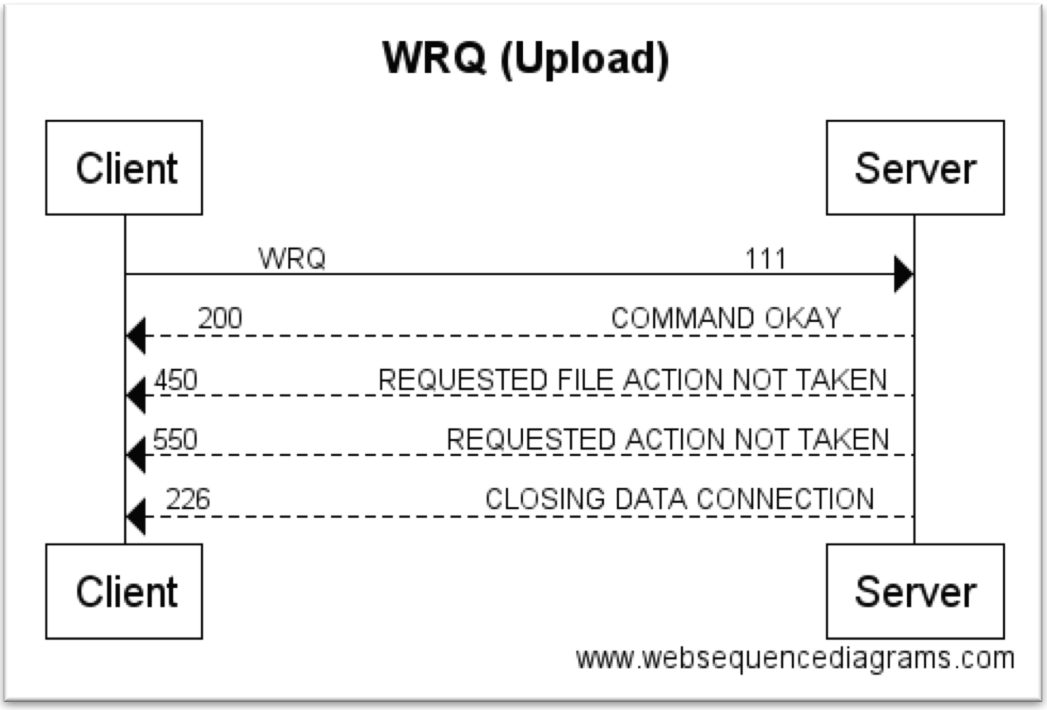
A. LOGIN Sequence Diagram



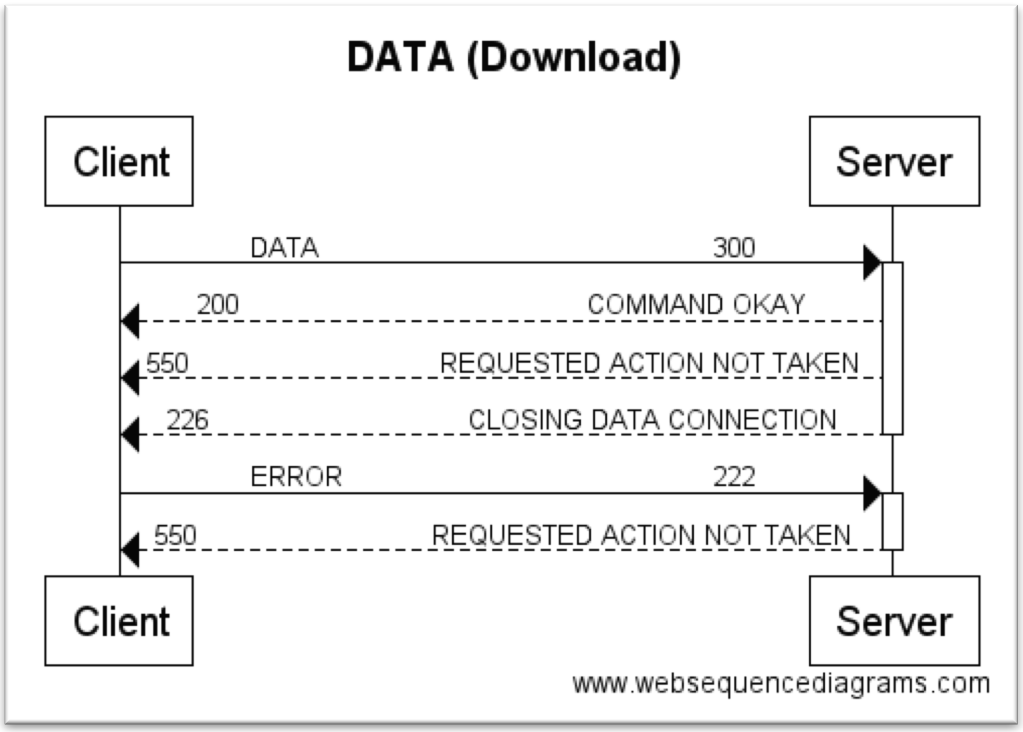
B. LOGOUT Sequence Diagram



C. WRQ (Upload) Sequence Diagram



D. DATA (Download) Sequence Diagram



5. Message Definitions

Message Passing Protocol Codes	
Code	**Description**
600	LOGIN operation code, used to login to the system
700	LOGOUT operation code, used to logout of the system
222	ERROR operation code, used to indicated erroneous data or message
400	ACK operation code, indicates a successful acknowledgement has been received
300	DATA operation code, used to download data from the server
111	WRQ operation code, used for uploading data to the server
200	Command okay, requested action has been successfully completed
226	Closing data connection. Requested file action successful (for example, file transfer or file abort)
230	User logged in, proceed. Logged out if appropriate
231	User logged out, service terminated
331	User name okay, need password
425	Cannot open data connection
430	Invalid username or password
450	Requested file action not taken
500	Syntax error, command not recognised and the requested action did not take place. This may include errors such as command line too long
550	Requested action not taken. File unavailable (for current directory or dataset)

6. Message Format

Each request must contain the server host address, the server port number, operation code, username, password and the message and/or data.

A. LOGIN

Message: Request login

Description: User wants to login to the server. (Username must start with '@' and end with '!' delimiters. Password must end with '!' delimiter.)

OpCode: 600

Message Parameter: [host] [port] [opcode] [@username!] [password!]

```
localhost,3000,600,@username!,password!
```

Message Response: On successful authentication, the server responds with code 230, otherwise 430 for invalid username or password, 331 for invalid password but valid user, 530 if not logged in.

B. LOGOUT

Message: Request logout

Description: User wants to logout of the server. (Username must start with '@' and end with '!' delimiters. Password must end with '!' delimiter.)

OpCode: 700

Message Parameter: [host] [port] [opcode] [@username!] [password!]

```
localhost,3000,700,@username!,password!
```

Message Response: On successful, the server responds with code 231, otherwise 430 for invalid username or password, 331 for invalid password but valid user, 530 if not logged in.

C. WRQ (Upload) Data

Message: Request to upload handshake

Description: User wants to send a handshake to the server to upload data. (Username must start with '@' and end with '!' delimiters. Password must end with '!' delimiter.)

OpCode: 111

Message Parameter: [host] [port] [opcode@username!password!]

```
localhost,3000,111@username!password!
```

Message Response: On successful handshake, the server responds acknowledgement code 200.

Message: Upload data to the server

Description: On successful handshake, send data to the server. (File size should not exceed 64 kilobytes.)

OpCode: 111

Message Parameter: [host] [port] [data]

```
localhost,3000,{binary data object}
```

Message Response: On successful upload, the server responds acknowledgement code 226, otherwise 450 on miscellaneous errors and 550 if file is unavailable

N.B. Binary data object must contain the source directory, destination directory, filename, file size, status and the file.

D. DATA (Download) data

Message: Request to download handshake

Description: User wants to send a handshake to the server to download data. (Username must start with '@' and end with '!' delimiters. Password must end with '!' delimiter.)

OpCode: 300

Message Parameter: [host] [port] [opcode@username!password!]

```
localhost,3000,300@username!password!
```

Message Response: On successful handshake, the server responds acknowledgement code 200.

Message: Download data from the server

Description: On successful handshake, download unrestricted data from the server. (File size should not exceed 64 kilobytes.)

OpCode: 111

Message Parameter: [host] [port] [data]

```
localhost,3000,{binary data object}
```

Message Response: On successful upload, the server responds acknowledgement code 226, otherwise 450 on miscellaneous errors and 550 if file is unavailable

N.B. Binary data object must contain the source directory, destination directory, filename, file size, status and the file.

Message: Request to download

Description: User wants to download restricted data from the server. (Username must start with '@' and end with '!' delimiters. Password must end with '!' delimiter.)

OpCode: 111

ErrorCode: 222

Message Parameter: [host] [port] [opcode@username!password!ErrorCode]

localhost,3000,111@username!password!222

Message Response: The server responds with 550

7. Pseudocode

```
login()

try {

   o validate host address
   o validate port number
   o validate user name
   o validate password
   o send authentication request

} catch {

   o invalid user name or password
   o user name okay, invalid password
   o invalid host
   o invalid password

} finally {

   o return
        o 230 - connected
        o 430 - invalid user or pass
        o 331 - invalid pass
        o 530 - not logged in

}
```

```
logout()

try {

   o  validate host address
   o  validate port number
   o  validate user name
   o  validate password
   o  send logout request

} catch {

   o  invalid user name or password
   o  user name okay, invalid password
   o  invalid host
   o  invalid password

} finally {

   o  return
        o  231 - logged out
        o  430 - invalid user or pass
        o  331 - invalid pass
        o  530 - not logged in

}
```

```
upload()

try {

   o validate host address
   o validate port number
   o validate user name
   o validate password
   o validate a file is selected
   o validate chosen file exists
   o validate file size does not exceed max size
   o send upload handshake
   o if handshake acknowledgement code from server is 200
        o send data to the server

} catch {

   o invalid user name or password
   o user name okay, invalid password
   o invalid host
   o invalid password
   o error in data
   o file exists

} finally {

   o return
        o 226 - data transferred
        o 450 - file transfer aborted for miscellaneous reason
        o 550 - file does not exist

}
```

```
download()

try {

   o validate host address
   o validate port number
   o validate user name
   o validate password
   o validate a file is selected
   o validate chosen file exists
   o validate file size does not exceed max size
   o validate accessible directory
   o if directory is not restricted
        o send download handshake
        o if handshake acknowledgement code from server is 200
        o download the file from the server to the client
   o else
        o send error code 222 to the server for restricted
          access

} catch {

   o invalid user name or password
   o user name okay, invalid password
   o invalid host
   o invalid password
   o error in data
   o file exists

} finally {

   o return
        o 226 - data transferred
        o 550 - file does not exist

}
```