RFT Protocol Documentation

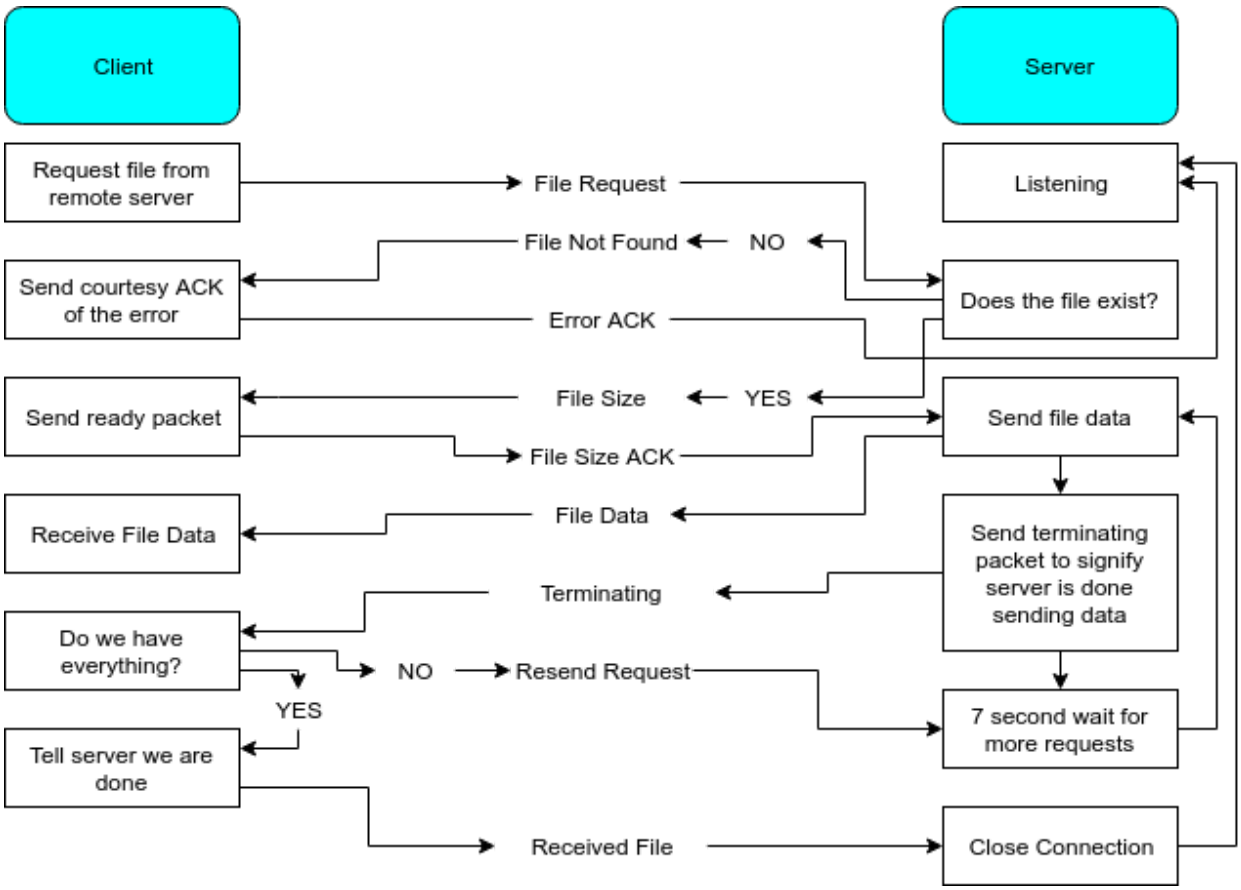
Josiah Keller and Gabriel Gonzalez

May 8th, 2018

Reliable File Transfer Protocol

The purpose this protocol was developed was in order to achieve a reliable way to transfer data to a client from a server, mitigating packet loss and/or out-of-order sequencing, across the already developed user datagram protocol. The way we achieved this was by utilizing sequencing numbers and waits to make sure the file was received with no packets loss, and using it to check that the datagram chunks were in the correct order as the original file.

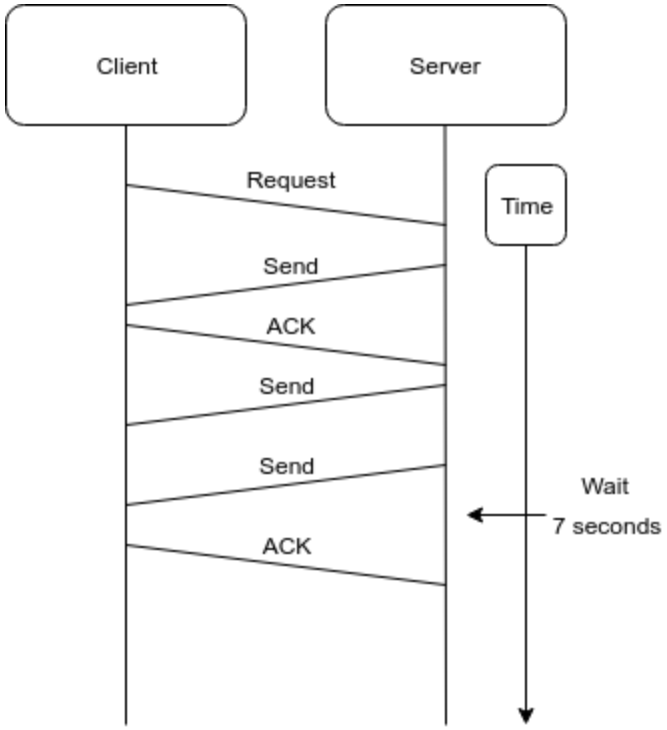
When the server is started, it will take the first command line argument as its port, and start listening on it. Then we can start the client using the arguments for specifying the IP address, the port number, the file we want to grab from the remote server, and the local directory address to store the downloaded file.



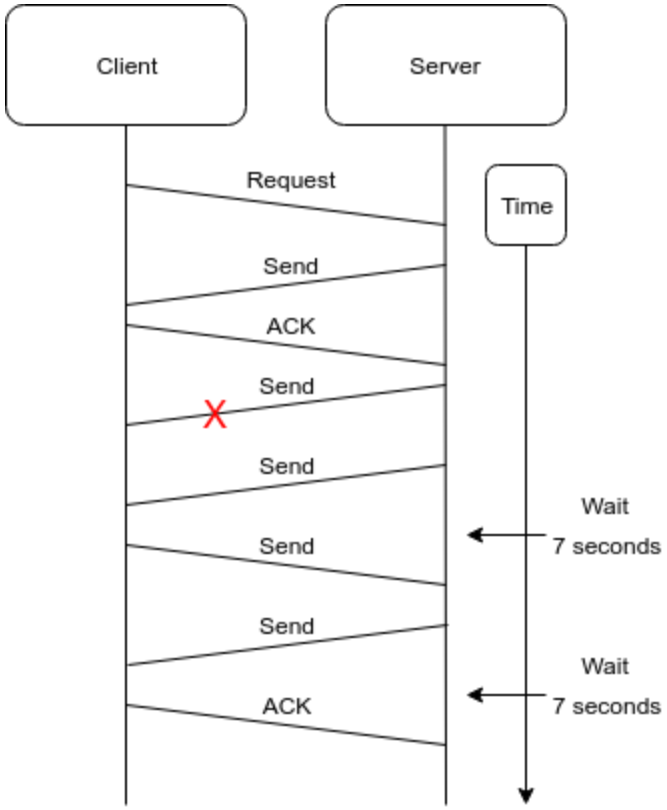
Our protocol was designed so that we would send a datagram of size 512 between the client and the server. Of these 512 bytes, the first 4 bytes would be our header number, which we use to let the client and server tell each other what they want, or what they are sending. The next 512 bytes will be the data from the file or the server.

File Request Packet:	Header = 0x01
	Data = File Path
File Size Packet:	Header = 0x02
	Data = File Size
File Not Found Packet:	Header = 0x0194
	Data = Error Message
Error Acknowledgement Packet:	Header = 0x80000194
	Data = Padding
File Size Acknowledgement Packet:	Header = 0x03
	Data = File Size
File Data Packet:	Header = 0x04
	Data = File Data
Terminating Packet:	Header = 0x04
	Data = 0xFFFFFFFF
Resend Request Packet:	Header = 0x05
	Data = Sequence Number
Received File Packet:	Header = 0x05
	Data = 0xFFFFFFFF

We have 9 different types of datagram packets being sent between the client and server. The first is the **file request packet**, which tells the server what file we want. The next is the **file size packet** which is sent to the client so it can allocate memory for the file, and after doing so the client will send the a **file size acknowledgement packet** to the server saying it is good to go and ready to receive the file data. Next comes the **file data packets.** Each file data packet contains a header, followed by a sequence number, and then followed by the file data itself. The client sees these and puts them in order as it receives the file chunks. A **terminating packet** is sent to tell the client it is done sending the files. The client will send the **received file packet**, telling the server that it has successfully received the file and it can close the connection and start listening again.



However, in the case that the server can not find the file requested, it will send a **file not found packet**, to which the client will acknowledge with an **error acknowledgement packet** once received, and the server will go back to listening. If the server is done sending the file and the client realizes it is missing a certain packet (i.e sequence number 29), it will send a **resend request packet** to the server with the sequence number it wants, and the server will send that file data packet back. The client sends the received file packet, and it is finished.


Client
Server
Time
Request
Send
ACK
Send
Send
Send
Wait
7 seconds
Send
Wait
7 seconds
ACK