**Introduction**

My program sends and receives data from the internet and it uses my own UDP protocol. To be able to send a packet in UDP protocol, it needs to be packed in IP protocol, which means that bytes 0-33 will stay unmodified by my protocol. My header looks like this:

|  |  |  |  |
| --- | --- | --- | --- |
| **Byte num** | **1** | **2 - 8** | **9 - 16** |
|  | **Ack Num** | | **Checksum** |
|  | **Flags** |  | |

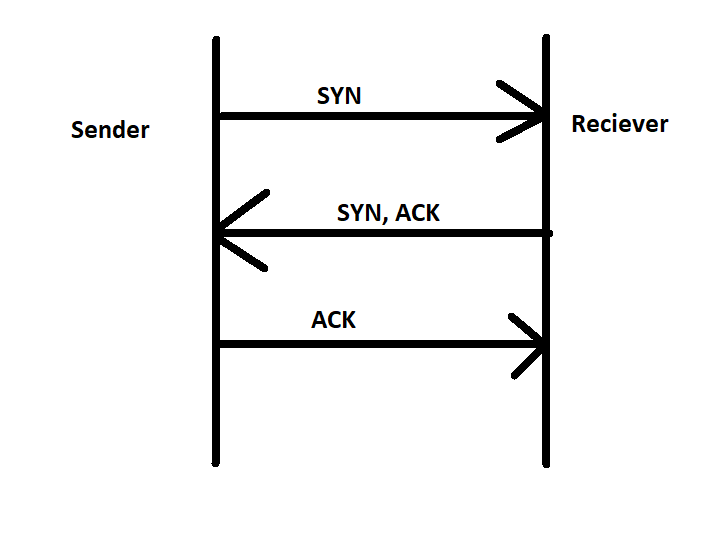
The maximum size of a payload for my program is 1500 – 20(Max IP Header size) – 8(UDP header size) – 17(My protocol adjustment size) = **1455 bytes.**

**Checksum:** a special number that controls that all data I a packet was transmitted full and uncorrupted. While a sender prepare a packet to send, the checksum is created based on different data that the packet consist. After packet was received, receiver uses the checksum with the same calculations as the sender. If the result of receiver’s calculation is same, that means that the packet was send full and uncorrupted.

**Ack number:** a special number generated by a initiator of connection to crate it.

**Flags:** special options to defy the specific of current packet. Possible flags are: SYN, ACK, RST, FIN and WAIT.

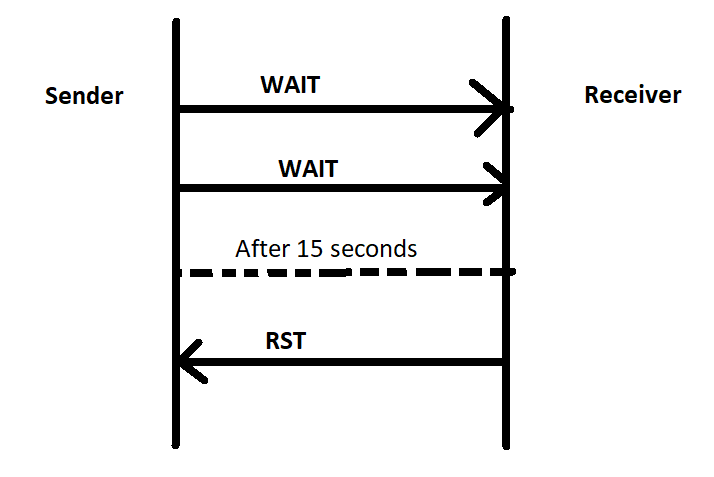
**Communication preparation**

To establish a communication using my protocol, firstly it need to create a connection. In such connection there are two sides: client(the one who receives data) and a server(the one who sends data). A diagram for it looks like this: 

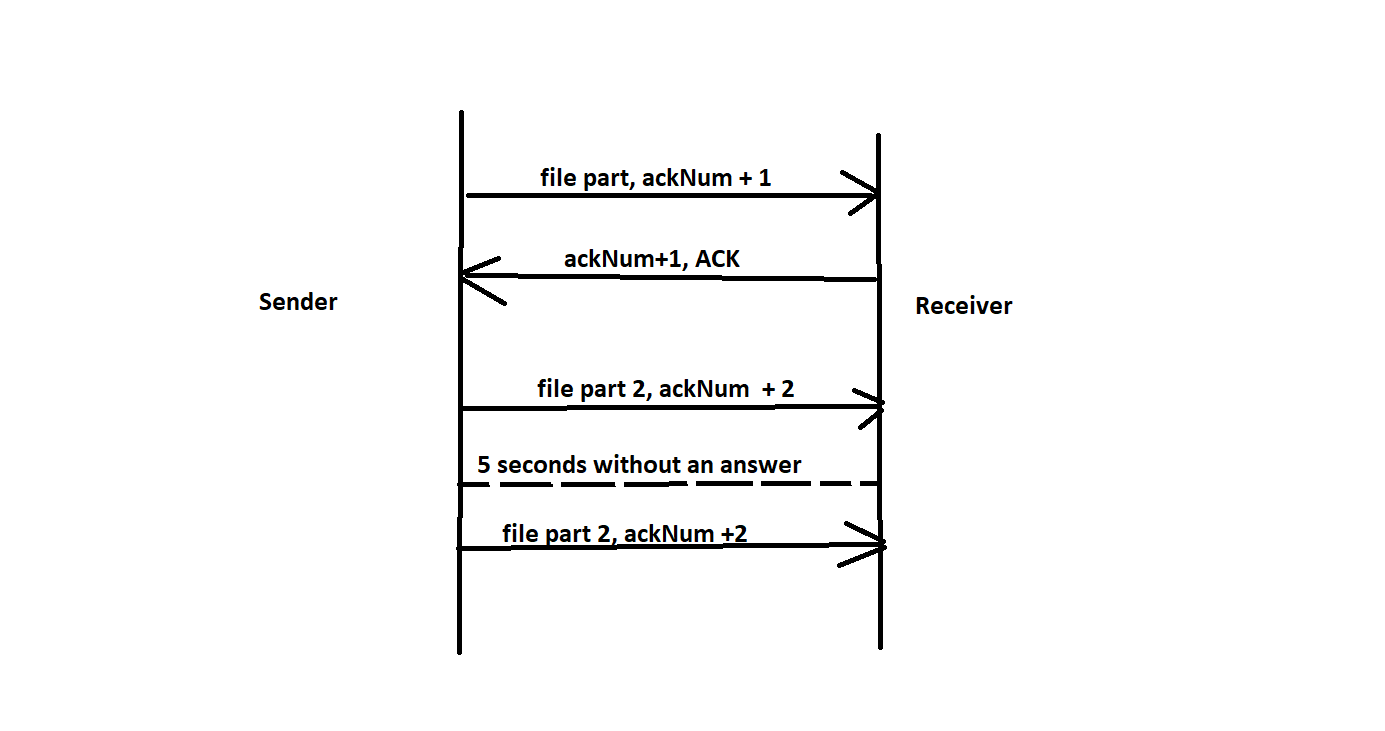
1. Client sends packet with SYN flag on. Client must have a newly created random Ack number in range <1;16383>.
2. Server replies to a client with SYN and ACK flags on. Server will send back client’s Ack number, but will make first bit to 1 (will add 32768 to Ack number).
3. Client finally replies to server with ACK flag on. Client will also send received Ack number + 1.

At this moment, the connection between the sender (server) and the receiver (client) is established.

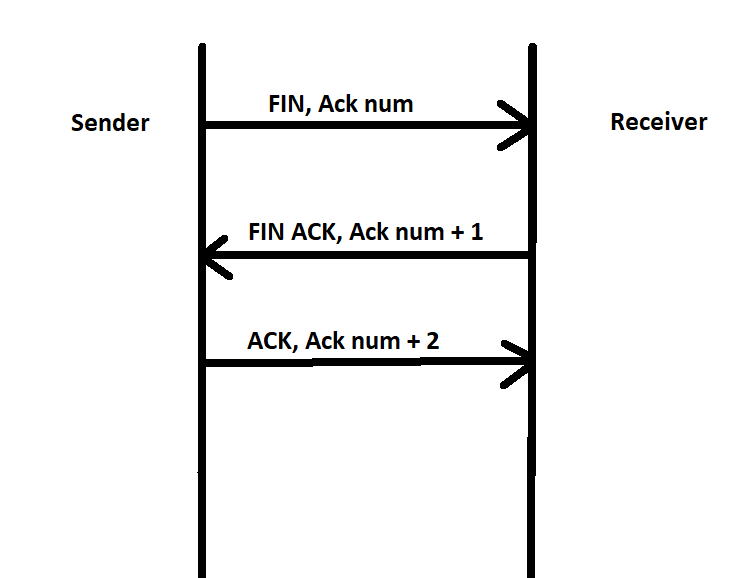
**Connection holding**

 While the sender is choosing what file to send and what the length of a packet, the connection can time out. For that, sender using threads is sending a package with a flag WAIT every 5 seconds to save the connection. If the receiver will not receive anything for 10 seconds, connection is broken.

**File transmitting**

 After sender and receiver created a connection, sender can send a special packet, that consist some important data about a file. It contains file name, file size, size of a data in every package and the amount of packages. After that sender start to send a file piece by piece. Sender can choose the size of parts that this file will be transmitted by, or, if the file is small enough, it can send full file with in one package. Maximum size of a one payload is 1468 bytes. Then all parts one by one will be sent to receiver in a proper order. Every package will also has a Ack number + number of a part of a file. After receiving any part of a file, a receiver will send back a package with ACK flag on and with the Ack number of a received package. If sender will not receive an ACK package for a part of a file for 5 seconds, it will send this part again. Sender will do this one more time, but for the third time it will send a package with a RTS flag on and with the last modified Ack number. For sender, the connection is over. A diagram for it looks like this (example with 5 parts of a file):

**Ending communication**

 If sender wants to end communication, it sends a packet with FIN flag on and last modified Ack number. After that, the receiver replies with a packet with FIN and ACK flags on and with Ack number of a sender +1. Then, the sender replies back with a package with a ACK flag on and with a Ack number for a receiver +1. A diagram for it looks like this: