**Northeastern University**

CS 5700 Fundamentals of Networking

**FINAL PROJECT REPORT**

**Team members:** Sanjivani Jethwaney, Pranjal Nimse, Paavan Gopala Reddy

**Title:** File Transfer Using TCP/IP Protocol Implementation with Raw Sockets

**Introduction:**

The objective of the project is to implement a TCP/IP protocol using raw sockets to transfer files from one host to another. Normally, the headers for network transfer requests are provided by the kernel/OS network stack. In this project, we have used raw sockets to provide custom headers for the transfer protocol. Here, we can provide customization for various header fields as opposed to the headers provided by the underlying operating system. Raw socket programming is often used in applications related to network security.

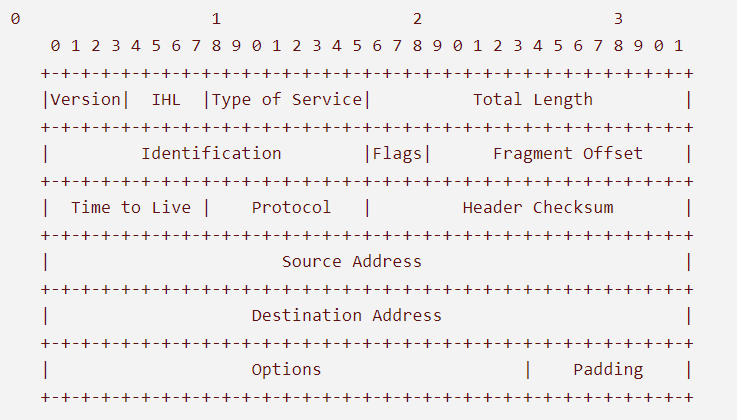
**TCP** is a highly efficient and reliable protocol designed for end-to-end transmission over an unreliable network. Transferring a file over TCP will also guarantee in-order delivery of data packets. This project is an application to transfer data files between network connected hosts. The execution modes in application are - Sender and Receiver.

**Methods and Algorithm Design:**

In the project, we have implemented the following customized fields for headers using raw socket programming:

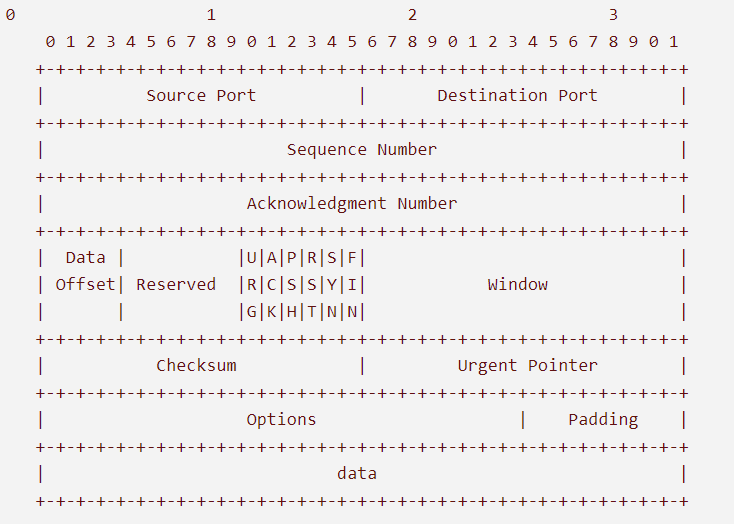
* IP Header:

According to RFC 791:



* + ip\_ihl: header length
  + ip\_ver: version of protocol
  + ip\_tos: type of service
  + ip\_tot\_len: total length
  + ip\_id: ID of the packet
  + ip\_frag\_off: fragmented offset
  + ip\_ttl: time to live
  + ip\_proto = socket.IPPROTO\_TCP
  + ip\_check: header checksum
  + ip\_saddr: source IP address
  + ip\_daddr: destination IP address
* TCP Header

According to RFC 793:



* + tcp\_source: source port
  + tcp\_dest: destination port
  + tcp\_seq: sequence number
  + tcp\_ack\_seq: acknowledgement sequence number
  + tcp\_doff: size of tcp header
  + TCP Flags
    - tcp\_fin
    - tcp\_syn
    - tcp\_rst
    - tcp\_psh
    - tcp\_ack
    - tcp\_urg
    - tcp\_window: maximum allowed window size
    - tcp\_check
    - tcp\_urg\_ptr

**Implementation:**

**Execution:**

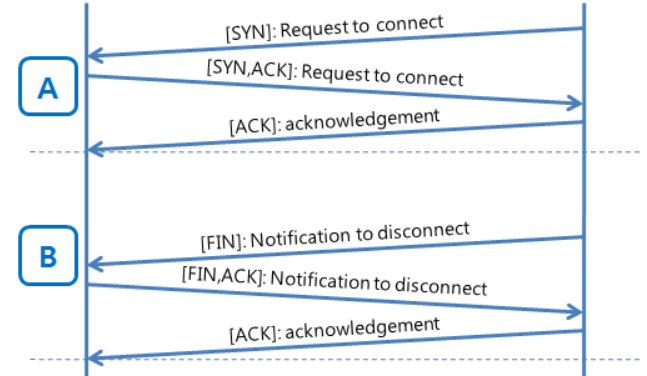
* Sender sends the file mentioned in cmd args to the Receiver whose ip address is specified. The application wraps the data packet to be sent with TCP and IP headers.
* The application uses raw sockets to communicate with the Sender and waits for the Sender to establish connection using TCP handshake.
* After connection establishment, the Receiver receives the data packet, extracts the TCP and IP headers and verifies its integrity using checksum in TCP header.
* If all goes well, the Receiver sends ACK to the Sender and awaits for the next packets to be received.
* When a FIN packet is received, the Receiver closes the output file and closed the connection.

**Accessing a raw socket:**

* Sockets are how programs on Linux way talk to the internet.
* The socket system call creates a file descriptor that can be written to and read from.
* The connect system call can then be used to connect the socket to some remote address.
* Writing to the socket sends data to that remote address, while reading from the socket file descriptor reads data sent from the remote address.
* Raw sockets operate at the network OSI level, which means that transport-level headers such as TCP or UDP headers will not be automatically decoded. If you are implementing a a transport-level protocol, you’ll have to write code to decode and encode the transport-level headers in your application.
* The main steps involved in programming with raw sockets:
  + Opening a raw socket
  + Reception of the network packet
  + Extracting the ethernet header
  + Extracting the TCP/IP header
  + Extracting the data

**Structure of client-server file transfer using socket programming:**

**Server Client**



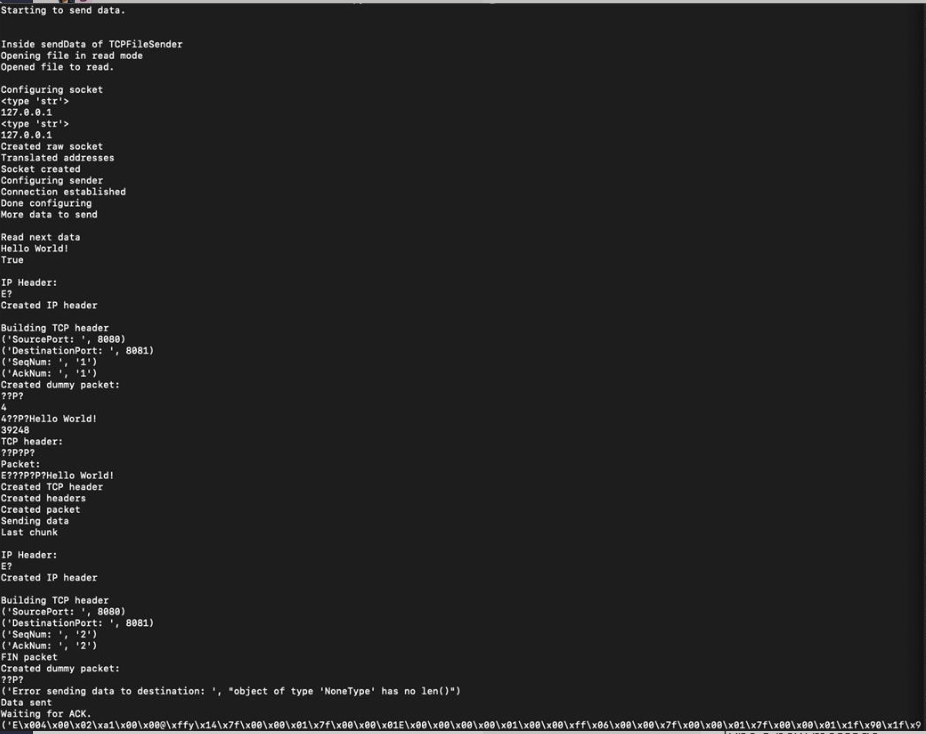
The application will close the TCP connection by sending a FIN packet after all of the data read from the file is transferred and end the application. When a FIN packet is received, the Receiver closes the output file and the connection.

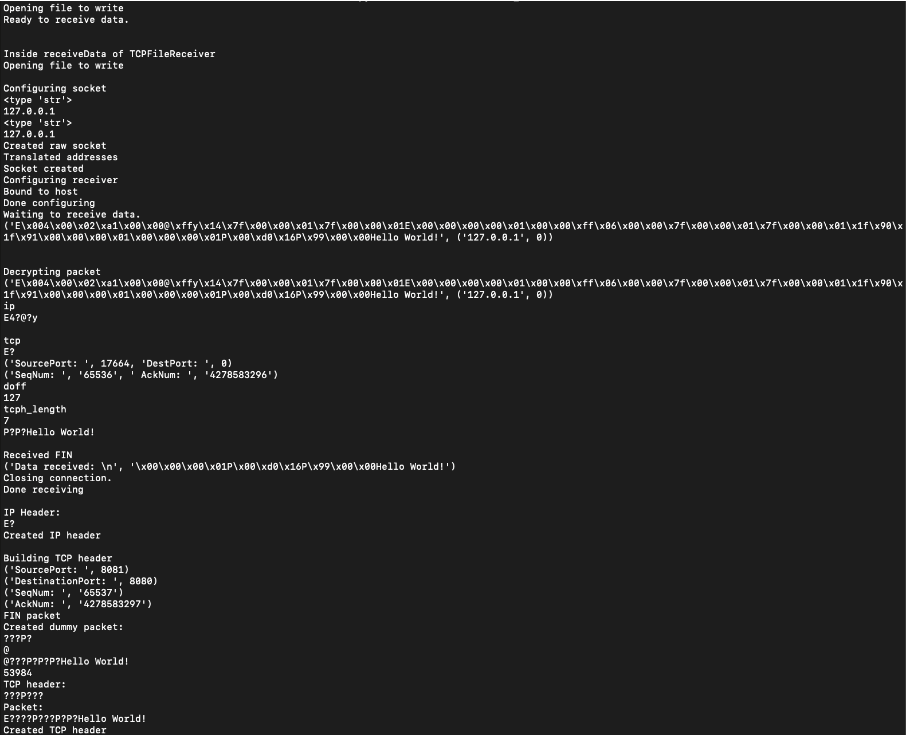
**Coding language:** Python

**Libraries**:

* sys
* socket
* signal
* os
* time

**Results and Discussion:**





**References:**

* <https://www.binarytides.com/raw-socket-programming-in-python-linux/>
* <https://medium.com/@yoursproductly/tcp-vs-udp-38b10bb1bbf3>
* <http://books.gigatux.nl/mirror/securitytools/ddu/ch06lev1sec3.html#:~:text=TCP%2FIP%20Header&text=The%20IP%20protocol%20header%20contains,and%20options%20for%20the%20packet>.