# Programming Assignment 2

In this assignment, you will develop a simplified version of the Transmission Control Protocol (TCP) that operates over User Datagram Protocol (UDP). Your implementation will include a three-way handshake for connection establishment, error logging, and a FIN request to signal the end of data transmission. The network environment is emulated using a provided link emulator, "newudpl," which simulates packet loss, delay, corruption, and reordering. Now, let's dive into the details of the assignment.

Assignment Details:

Your task is to write the sender (client) and receiver (server) transport-level code for a simplex version of TCP. The sender and receiver will communicate via UDP, using the provided link emulator (newudpl version 1.7) as a proxy. The emulator can introduce network impairments, but acknowledgments are assumed to arrive without delay or loss.

The sender and receiver processes, along with the link emulator, can run on one, two, or three separate machines. The sender sends packets to the emulator, which forwards them to the receiver. The receiver sends acknowledgments directly to the sender.

You will implement a three-way handshake for connection establishment and include error logging to track issues during execution. Additionally, your program will handle one set of packets (one "file") and use a FIN request to signal the end of transmission. Sequence numbers start at zero, and there is no need for congestion or flow control.

To test your implementation, run the link emulator with the following command:

```
newudpl -iinput_host_address -ooutput_host_address -L 50
```

The TCP data receiver should be invoked as follows:

```
tcpserver file listening_port address_for_acks port_for_acks
```

The server receives data on the `listening_port`, writes it to the file `file` and sends ACKs to `ip_address_for_acks`, port `port_for_acks`.

The client (data sender) has the following interface:

```
tcpclient file address_of_udpl port_number_of_udpl windowsize
ack_port_number
```

The client reads data from the specified file, sends it to the emulator's address and port, and receives acknowledgments on the ack_port_number. The window size is measured in bytes and can be set to any reasonable default value. The TAs will specify all arguments when testing your client and server.

TCP Header Format:

You need to implement the 20-byte TCP header format without options. You do not have to implement push (PSH flag), urgent data (URG), reset (RST), or TCP options. Set the port numbers in the packet to the correct values, but otherwise, you can ignore them. The TCP checksum is computed over the TCP header and data (with the checksum set to zero). This approach is a close approximation of the correct method, which includes parts of the IP header.

You can choose a reasonable value for the maximum segment size (MSS), such as 576 bytes.

**Extra Credit:**

Extra credit is available for well-documented code. It can boost your score to a maximum of 100, compensating for any lost points in the written part of the assignment.

**Submission Requirements:**

Please include the following in your submission (ZIP file):

1. README.txt file that lists:
    - All submitted files and a short description of each.
    - Commands needed to run the programs.
    - List of any known bugs, features, etc.
    - The TA should be able to run your program on any host and port number by providing this information from the command line. If you've tested your code on a specific machine that you want the TA to use, mention this in your README. Programs requiring code editing by the TA will be rejected. Incomplete READMEs may incur a penalty.
2. A separate (typed) document describing:
    - The overall program design.
    - A verbal description of "how it works."
    - Design tradeoffs considered and made.
3. A program listing with in-line documentation.
4. A screen dump of a typical client-server interaction.

Downloading the Link Emulator:

The link emulator tool, "newudpl," is available on the Courseworks website under the "Files" page. Two files are provided: "newudpl" (a Linux binary) and "newudpl-1.7.tar" (installation files for version 1.7). If your operating system is compatible, you can download and run the "newudpl" binary directly. Otherwise, download and unzip the "newudpl-1.7.tar" file and follow the directions in the INSTALL file to install the emulator.

**Grading Rubric for Programming Assignment 2: Simplified TCP over UDP**

Total Points: 100 + 10 Extra Credit

- Functionality (80 points):
    - Three-way handshake implementation (15 points)
    - Data transmission and reception (15 points)
    - FIN request to signal the end of transmission (10 points)
    - Error logging (10 points)
    - Correct implementation of the 20-byte TCP header format (10 points)
    - Correct handling of sequence numbers (5 points)
    - Retransmission timer adjustment as per the TCP standard (5 points)
    - Correct computation of the TCP checksum (5 points)
    - Correct handling of command-line arguments (5 points)
- Documentation (15 points)
    - README.txt file with complete and accurate information (5 points)
    - Separate document describing program design, functionality, and tradeoffs (5 points)
    - Screen dump of a typical client-server interaction (5 points)
- Extra Credit (10 points)
    - Well-documented code (5 points)
- Additional features or enhancements (5 points)

**Some Notes on the Grading Rubric:**

- The rubric is a guideline for grading. Instructors and TAs may use their discretion in awarding points.
- Partial credit may be awarded for incomplete or partially correct implementations.
- Extra credit can boost the score beyond 100 points, up to a maximum of 110 points.