Reliable protocol over UDP

User Datagram Protocol (UDP) is a minimal protocol running over IP. In this assignment you will implement a reliable message protocol over UDP. You will emulate a subset of TCP protocol. Features to be implemented are:

- Establish a connections
- Ordered data transfer
- Retransmission of lost packets

We provide you templates for client and server applications for **Python 3**.

We plan to update this document as it's needed. If there's any major update we'll make an announcement.

Milestone 1 - due March 29

Connection setup & teardown, due 3/29

Check the **1 Establish a connection** and **3 Tear down the connection** sections below.

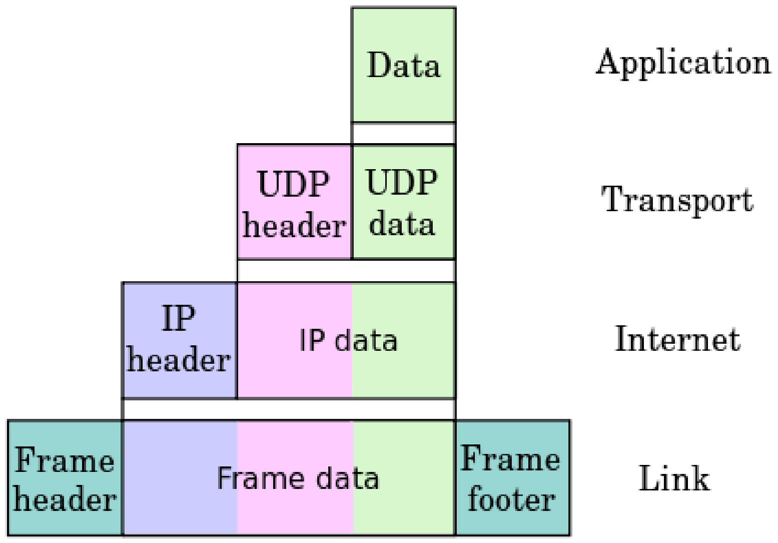
Milestone 2 - due April 10

Data transfer with retransmission of lost packets

Bonus points for any extra features of TCP like flow control.

Details

In essence, you will implement TCP-like protocol within UDP data. UDP data will consist of TCP header and TCP body.



Provided template code

Please, read and understand the provided template code. There are three python files are provided.

utils.py

This file implements States Enum type, header type for your own protocol and some helper functions. You will need to extend these classes for your needs as they are missing some states or fields.

DEBUG

If you set `DEBUG = True`, it will print extra information as your application is running. If you need to print debug information as you develop application, check this variable before printing. Set this value to `False` when you're submitting the code.

States

`class States`: implements Enum types to represent TCP protocol states.

Please refer to

server.py

Read the comments in the file

client.py

`MSS = 12`: we use pretty small segment size to make it easier to test and demo. We will need to divide long messages into a number of packets.

Read the comments in the file

Header

This class represents the Header of your own protocol and provides some helper methods.

The header has 12 bytes. First 4 bytes represent *sequence_number*, the second 4 bytes represent *acknowledge_number* and the next two bits represent the syn and ack header fields. The remaining 30 bits are currenctly unused. You can utilize the rest for your needs.

Steps

1 Establish a connection

Implement the TCP 3-way Handshake Protocol

Please refer to Connection establishment section: https://en.wikipedia.org/wiki/Transmission_Control_Protocol

- Server is waiting for a connection
- Client sends a SYN
- Server replies with SYN-ACK
- Client replies with ACK

2 Transfer the data

UDP is an unreliable protocol. Some of the packages might be lost or received in different order. Implement the TCP-like protocol to retransmit the missing packages. Client should order the packets correctly.

3 Tear down the connection

Bonus: TIME_WAIT state implementation is bonus points. You won't lose points if you don't implement TIME_WAIT.

Resources

https://en.wikipedia.org/wiki/Transmission_Control_Protocol#Protocol_operation
https://tools.ietf.org/html/draft-gg-udt-03