Programming Assignments 2 and 3

CS 5283 - Computer Networks

“TCP” over UDP: 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. The assignment is broken into two milestones, representing programming assignments numbered 2 and 3. 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**.

Milestone 1 (Programming Assignment 2)

- Connection setup & teardown
- Check the **1 Establish a connection** and **3 Tear down the connection** sections below.

Milestone 2 (Programming Assignment 3)

- Data transfer with retransmission of lost packets
- Bonus points for any extra features of TCP like flow control.

Following the completion of milestone 1, we will release client.py and server.py solution files for milestone 1, as well as a 3rd program that simulates a lossy channel, to ease testing (so you would run 3 programs, the client, the server, and a 3rd Python program we will provide that drops/reorders messages).

Details

In essence, you will implement TCP-like protocol on top of UDP. UDP payload data will consist of your “TCP” protocol's header and 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 a 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 as discussed in class, or e.g. here: https://en.wikipedia.org/wiki/Transmission_Control_Protocol#Connection_establishment

- 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. For milestone 1, no retransmission or reliability aspects are necessary. For milestone 2, at least message drops and reorders should be handled (this is tricky depending on how you implement it, we recommend just using stop and wait).

You are free to choose a protocol for retransmission, as discussed in the reliable data transfer and TCP sections, such as stop and wait.

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

Submission – Programming Assignment 2 (Milestone 1)

Please submit to Brightspace for this assignment:

1) Your client and server source files (and any others needed, such as utils.py if modified),
2) A screenshot of the client and server with a successful (1) connection establishment, (2) data transfer of some form (e.g., some ASCII text), and (3) connection teardown.

Submission – Programming Assignment 3 (Milestone 2)

Please submit to Brightspace for this assignment:

1) Your client and server source files (and any others needed, such as utils.py if modified),
2) Screenshots of the client and server with a successful (1) connection establishment, (2) data transfer of some form (e.g., some ASCII text) illustrating it is appropriately re-ordered and with appropriate retransmissions due to message losses from the simulated lossy channel, and (3) connection teardown. **Special note**: what happens if messages are lost/reordered during connection establishment or teardown?