Readme

Zetao Yu, Chuanqi Xiong

Introduction

This is a documentation for Rutgers CS352 Spring 2019 Project 1. The project demonstrates a simple version of Go-Back-N protocol similar to TCP. As implementing in Python, we built it based on UDP transport layer.

Usage and Arguments

Users should run server2.py and client2.py on different terminals (either on different machines or not)

Arguments:

-f <the file name to send or to save>
-d <the destination host to send to>
-u <the UDP port to use for receiving>
-v <the UDP port to use for sending> (optional)

Example Usage:

python server2.py -f recv.txt -u 8888 -v 9999
python client2.py -d localhost -f send.txt -u 9999 -v 8888

Detailed Description

In order to imitate TCP protocol in each step of socket communication, our functions in project map the existing Python socket built-in methods. For example, connect() and accept() function would first complete a “three way handshake” communication, then the socket connection would be established. In recv(), the server would only accept packets with correct sequence number, and send back ACKs. If any packets lost, the server would block and wait until the client retransmit the packet after timeout. In send(), we use the technique of multi-threading to allow the client sending packets and receiving ACKs simultaneously. When one thread realizes there is a timeout, it will notify the other thread immediately by modifying a shared global variable. And the sending thread will resend from the specific packets. In close() function, we implement “two double handshakes” algorithm which ensures the termination can be done from both sides. Notice the termination would fail if the transmission is still going on.

sock352.py function list:

```
def init(UDPportTx, UDPportRx):
def fileDivder(buffer):
        return list(segments)

class socket():
        def __init__(self):
        def connect(self, address):
        def accept(self):
                return (clientsocket, address)
        def close(self):
```

```
def send(self, buffer):
        return len(buffer)
def recv(self, nbytes):
        return bytesreceived
def autopack(self):
def __assemble_chunk(self, segement):
        return str(chunk)
```

Dependencies:

-Python 2.x (https://www.python.org/downloads/release/python-272/) (using python3 may cause crash)

-Currently the project is not using any outside packages, but we may use in the future version to extend the functionality.