README
Team Members: Fu Shen   fushen@gatech.edu
                   Wendi Tang   wtang45@gatech.edu

Class: CS3251 B
Date: Nov.25th
Assignment: Programming Assignment 2

Compile:
1, No need to change file path in FAA_UI when compiling on linux through terminal. Otherwise please make sure you have the correct path:
       FILE_PATH: ~/test_files/
       SERVER_DOWNLOAD_PATH: ~/uploads/
       CLIENT_DOWNLOAD_PATH: ~/downloads/

2, run "emulator" in one terminal: python NetEmu.py 5000

3, javac FAA_server.java; javac FAA_client.java

4, run FAA_server in different terminal: java FAA_server 8081 127.0.0.1 5000

5, run FAA_client in different terminal: java FAA_client 8080 127.0.0.1 5000

Commands:
FAA_server Command: Only allow when server is listening

       "window W":   to change max window size to W
       "terminate": to gracefully shut-down server

FAA_client Command:
       "connect":   connect to the server
       "get F":   download file F from server
       "post F":   upload file F to server
       "disconnect": close the connection

File Description:
       FAA_server.java: the server application
       FAA_client.java: the client application
       FAA_UI.java: the base abstract for server and client

AAPSocket.java: the socket of FAA protocol
AAPServerSocket.java: the server socket for FAA protocol
AAPPacket.java: packet structure of FAA protocol including header and data
AAPInputStream.java: inputstream for reading data
AAPOutputStream.java: outputstream for sending data
AAPUtils.java: help functions for AAPPacket
ByteBufferQueue.java: data structure for inputstream buffer

Exception java files:
ConnectionAbortEarlyException.java, FileTransferException.java, FlagNotFoundException.java, InvalidCommandException.java, PacketCorruptedException.java, PayloadSizeTooLargeException.java, ServerNotRespondingExceptino.java
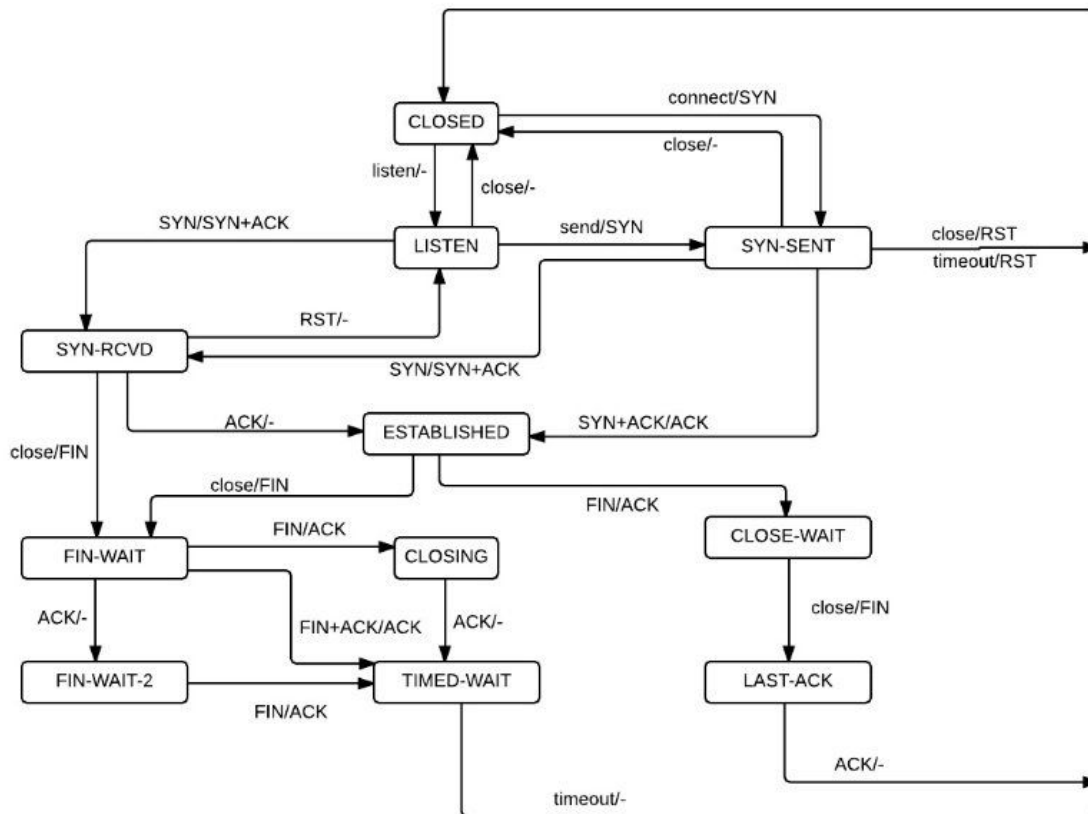
Folder Description:
test_files: file for test
downloads: where the client get the file and save to
uploads: where the client post the file and server save the file to

Updates and API description:
Updates:
      Used three-way hand shake instead of four-way hand shake described in the previous

CLOSED — connect/SYN — close/- — listen/- — close/- — SYN/SYN+ACK — LISTEN — send/SYN — SYN-SENT — close/RST — timeout/RST — RST/- — SYN-RCVD — SYN/SYN+ACK — close/FIN — ACK/- — ESTABLISHED — SYN+ACK/ACK — close/FIN — FIN/ACK — CLOSE-WAIT — FIN-WAIT — FIN/ACK — CLOSING — ACK/- — ACK/- — FIN+ACK/ACK — close/FIN — FIN-WAIT-2 — TIMED-WAIT — LAST-ACK — FIN/ACK — ACK/- — timeout/-

Documentation.
API description:
      AAPSocket

- AAPSocket(String server, int servPort, int localBindPort)
    - Constructor. It takes in the server address and server port number. It implicitly does connect() upon creation. It will throw an unknown host exception if the host is not found.It might throw illegal argument exception if the prot number is not valid. It will throw and IOException if any IO error occurs while opening the socket.
- getAAPInputStream()
    - Return the input stream for this socket which can be used to get input data.
- getAAPOutputStream()
    - Return the output stream for this socket which can be used to send data.
- close()
    - Close the connection

- ○ AAPServerSocket
  - ■ AAPServerSocket(int port)
    - ● Constructor. It takes in the port number and bound the server to that specific port. It implicitly does listen() upon creation. It might throw illegal argument exception if the prot number is not valid. It will also throw IOException if any IO error occurs while opening the socket.
    - ● A queue is used to stored multiple incoming connections. Any connection arrives when the queue is full is refused.
  - ■ accept()
    - ● Accept the connection to this socket and return AAPSocket. It blocks until a connection is made.
  - ■ close()
    - ● Close the socket.
- ○ AAPInputStream
  - ■ read()
    - ● Read the next byte from the stream
  - ■ read(byte[] recvBuffer)
    - ● Read some number of bytes from the stream and stored it in the buffer
  - ■ read(byte[] recvBuffer, int off, int length)
    - ● Read up to "length" bytes into the recvBuffer starts at offset "off"
  - ■ close()
    - ● Close the inputstream and release the resources
- ○ AAPOutputStream
  - ■ write(byte b)
    - ● Write a single byte to the outputstream
  - ■ write(byte[] bArray)
    - ● Write the byte array to the outputstream
  - ■ write(byte[] bArray, int off, int len)
    - ● Write "len" bytes to the outputstream from the "off" byte in the buffer
  - ■ close()
    - ● Close the outputstream and release the resources

Known bugs and limitations:
    It takes a while to transfer large size file. Error handling is not completely work.