IERG4180 Report 2

This project is to demonstrate the Code Design Architecture, Functions in project 2.
To achieve configuration consistency and concurrent work on different sockets, the socet_default is design to accept the new connection. For the reliability communication, I choose the TCP as configuration sending protocol.

```
typedef struct DOKI_packet {
    int mode;
    int proto;
    int pktsize;
    int pktnum;
    int pktrate;
    int port;
}DOKI_packet;
```

Initially, the port is set to -1, which means unused. The Server just extract the top 5 label.
For the TCP sending and receiving mode for Server, I can just keep the socket which accept() returns and use this socket to do sending or receiving.
For the UDP sending and receiving, because we don't do connection, the receive side should send its port number to sending side. When server recognize this is udp protocol, if this is sending mode, the server use explicit early binding and get port number by getsockname() and use the socet_default to sending port number.

The architecture for server, processNewConnection() is design to put the socket to socketHandles, in this function, getUdpSenderSock() and getUdpReceiverSock() is the implementation of the previous paragraph.

```
typedef struct SockClock{
    clock_t current;
    clock_t previous;
    long threshold;
    int stat;
    long send;
    long pktnum;
    long pktsend;
} SockClock;
```

For sending rate control, I create the SockClockList to record each sock.

For cross-platform maintenance, I will encapsulate functions in different formats for win and linux in utils.cpp and replace them with the prefix cp_. This approach reduces the coupling between codes, and layers the code, so that the business code does not need to care about the underlying implementation, which greatly increases the scalability of the code.