File Transfer

COMP4985

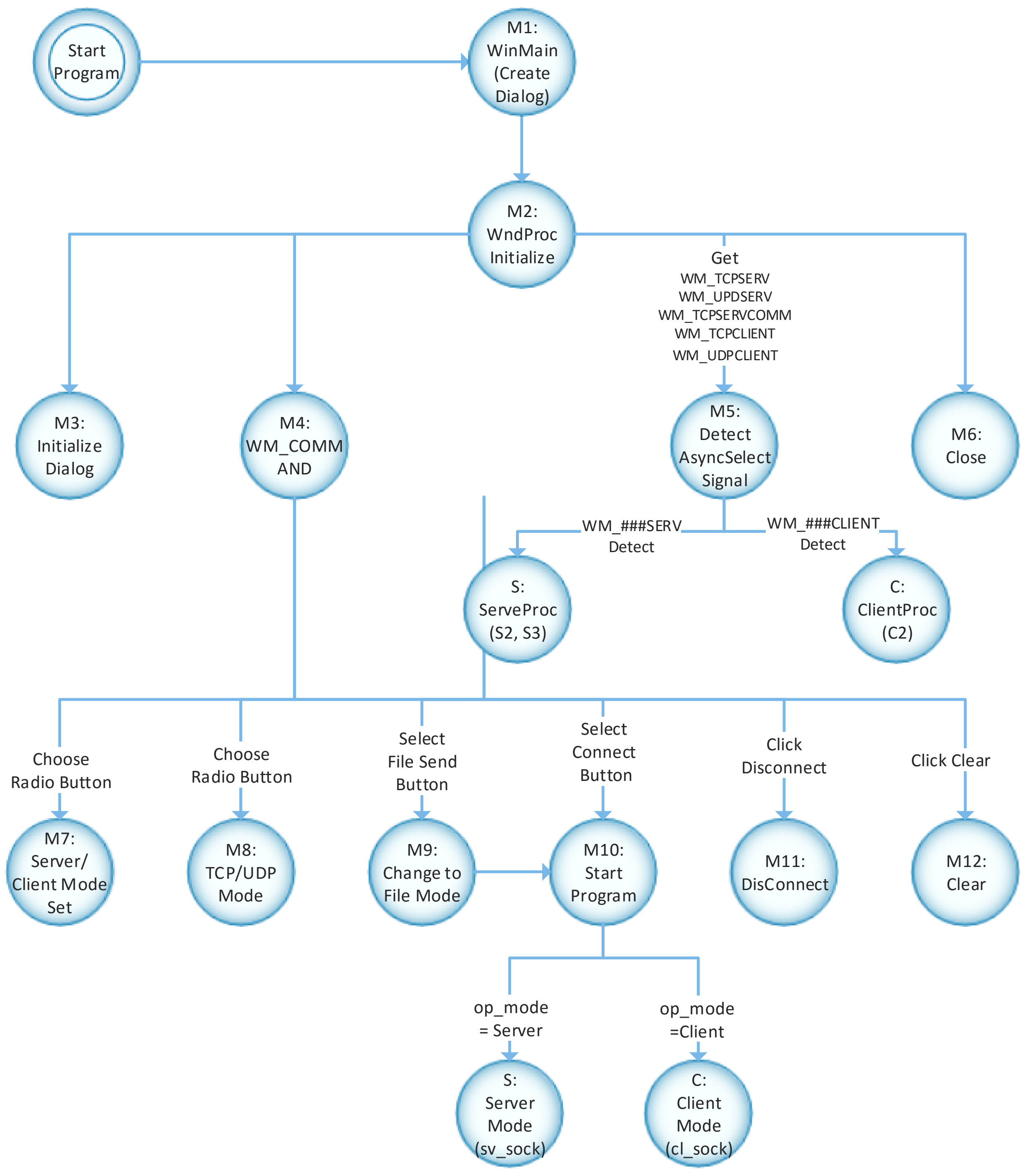
Assignment2

A00907822

Moon Eunwon

DESIGN

State Diagram



DESIGN

Psudocode

```
[M1]
Function WinMain()
        Create Dialog
        Show dialog
        Initialize all Hwnd of each item in Dialog  (initHwnd function)

[M2]
Function WndProc(message, wparam, lparam)
        switch(message)
          [M3] case WM_INITIDIALOG:
                        init dialog value function
                        break
          [M4] case WM_COMMAND
                        call menuAction function to operate menu selection
                        break
          [M5]  case WM_TCPSERV, WM_UDPSERV,  WM_TCPSERVCONN:
                        Call ServerWinProc Function (SERVER part) [S2. S3]
                        break
                case WM_TCPCLIENT,  WM_UDPCLIENT:
                        Call ClientProc Function (CLIENT part) [C2]
                        break
         [M6] case WM_DESTROY
                        quitwindow
                        shotdown and close socket depending on Mode
                        break

[M3]
Function dialogInit(hwnd)
        //initialize default valie/
        Set Type radiobutton : Server
        type  option : SERVER
        Set Protocol radio button : Tcp
        Protocol option  :  TCP
        Message type option : MESSAGE ( MESGSEND)

        Set port initial value : 7000
        Lock Edit Text field : Address, message Size, Number of Send, FileName
```

DESIGN

[M4]

```
Function menuAction(hwnd, wParam)
    [M7]
    choose 'RADIO_SERVER'
        change op_type to SERVER
        changeDialogType to SERVER mode
        break
    choose 'RADIO_CLIENT'
        change op_type to CLIENT
        changeDialogType to CLIENT mode
        break
    [M8]
     choose 'RADIO_TCP'
        change op_protocol to TCP
        break
    choose 'RADIO_UDP'
        change op_protocol to UDP
        break
    [M9]
    Click 'File Transfer' button
        Get filename
        If no name, display error message
            break
        Else change message type 'FILE
    Click 'CONNECT' button
        Get port Number
        If no value in port
            Set port number to DEFAULTPORT value(7000)
        If client mode
            Get IP addresss, MessageSize, Number of Message send
            If  any texts are not valid
                display error
                break
            else
                convert port, Number of Message  to number

            start client winsock mode (cl_winsock)
        else if server mode
            start server winsock mode (sv_winsock)
```

[S10]

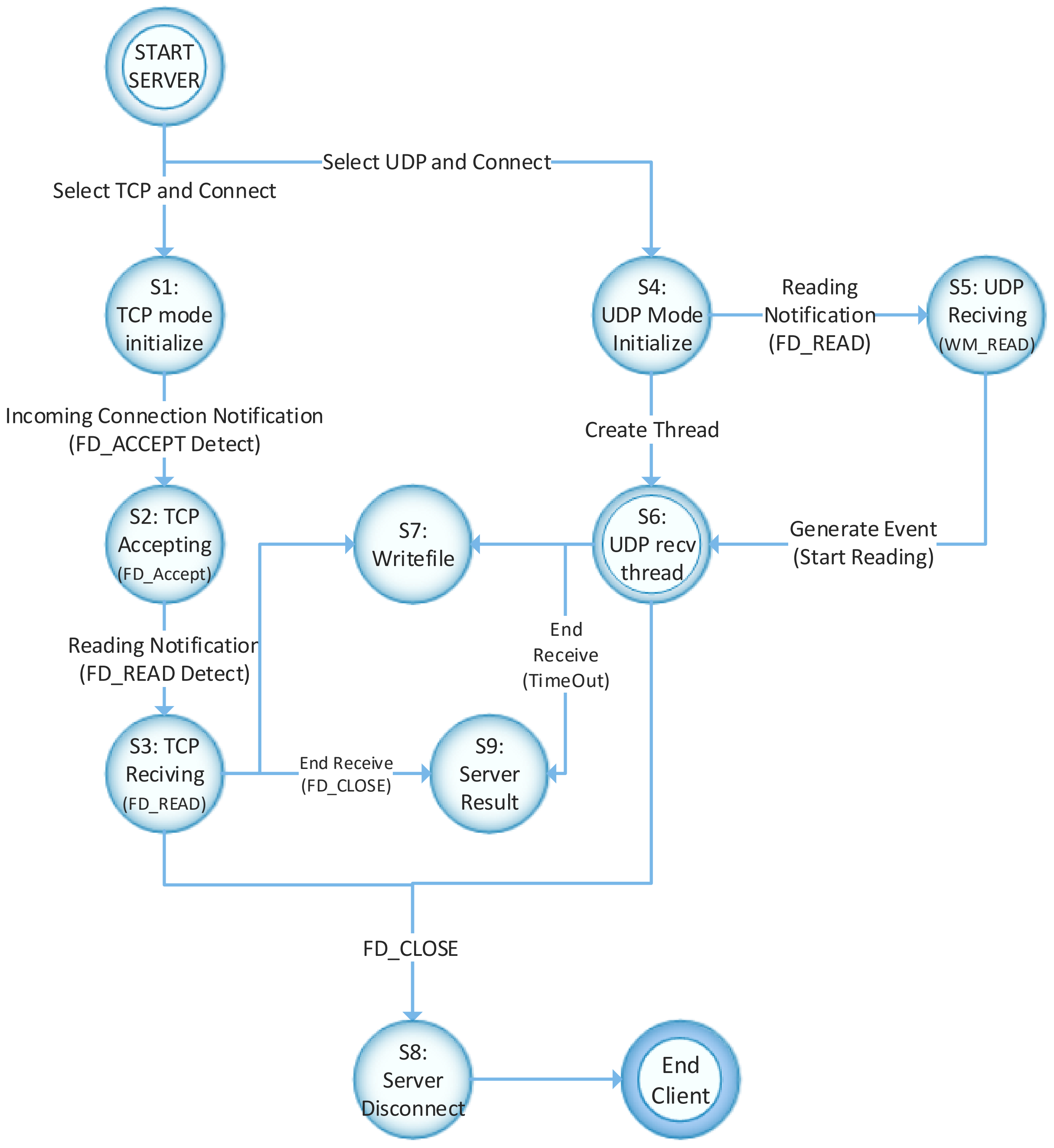
```
Click ‘Clear button
    Send RECETCONTENT message to hList
    break
```

[S11]

```
Click ‘DISCONNECT’ button
    Showdown and close socket depending on type(op_type)
    WSACleanup
    Enable the SERVER, CLIENT button UNBLOCK
    break
```

DESIGN

State Diagram



Assumption

** S2, S3, S4 if detect FD_CLOSE event, close socket and end client*

DESIGN

Psudocode

```
[START]
Function sv_winsock(HWND, socket,  op_no, port_number)
        open socket session info using WSAStartup
        ReadNotStart to ture;
        [S1]
        Open socket in TCP MODE
                (use WSASocket,   SOCK_STREAM, IPPROTO_TCP, OVERLAPPED flag)
                        If fail, display error message and return
                Set the event 'WM_TCPSERV'  to detect FD_ACCEPT|FD_CLOSE

                Initialize SOCKADDR_IN

                Bind socket
                        If error display error message and return
                Listen SOCKET
                        If error display error message and return
                Break
        [S4]
        Open socket in UDP MODE
                (use WSASocket,   SOCK_DGRAM, IPPROTO_UDP, OVERLAPPED flag)
                        If fail, display error message and return
                Set the event 'WM_UDPSERV'  to detect FD_READ|FD_CLOSE

                Udpthread is Make Event do detect message arriving
                        (to block error in UDP recv thread)

                Initialize SOCKADDR_IN

                Bind socket
                        If error display error message and return

                Create Thread UDPServThread, pass socket pointer  (recv message)
                        If error display error message and return
                break
```

```
Function ServWndProc(hwnd, message, wParam, lParam, socket)
     Switch(message)
       case WM_TCPSERV:
          if detect error message
               error message and return
          else
               switch(WSAGETSELECTEVENT)
               [S2]
               case FD_ACCEPT
                    send accept to client (WSAAccept)
                    update socket to new Accept return value
                    initialize ReceiveData to 0
                    Set the event 'WM_TCPSERVCONN'  to detect READ|CLOSE
               case FD_CLOSE
                    close socket
                    break
       case TCP_SERVCONN:
          if get WSAGETSELECERROR value, error message and return
          else
            switch(WSAGETSELECTEVENT)
               [S3]
               case FD_READ
                    allocate SocketInfo structure (LPSOCKET INFORMATION type)
                    initialize SocketInfo value(socket, DataBuf, overlapped)
                    Receive from client (WSARecv)
                    If error detect and not IO_PENDDING or WSAWOULDBLOCK
                         Error message and return
                    If first packet is false
                         Get system time
                         First packet change to true
                    Add the size of message received to RecvData
                    Get system finished time(keep updating)
                    break

               case FD_CLOSE
                    call ServerResult function[S9]
                    change firstPacket change to false
                    break
```

```
        [S4]
        case UDPSERV:
            if get WSAGETSELECERROR value, error message and return
            else
                switch(WSAGETSELECTEVENT)
                    [S5]
                    case FD_READ
                        if readingNotStart is true
                            change readingNotStart to false
                            SetEvent 'UDPevent'
                        break

                    case FD_CLOSE
                        closeSocket
                        break
```

[S6 - Thread]

```
Function UDPServThread(socket)

    Get Port number and IP address
    If Port is not empty, update port value

    allocate SocketInfo structure (LPSOCKET INFORMATION type)

    initialize socketaddress info (SOCKADDR_IN)
    initialize SocketInfo value(socket, DataBuf, overlapped event)

    forever loop
        waitforsingleobject : FD_READ  [S3]
        reset updread event
        if  startrecv is false save start time and change to true

        forever loop
            initialize overlapped
            Receive from client (WSARecv)
                If error detect and not IO_PENDDING or WSAWOULDBLOCK
                    Error message and return
            If WSAWiatForMultipleEvents of OVERLAPPED is TIMEOUT
                Call ServerResult function to display result. [S9]
                startRecv to false
            add the received bytes amount to SocketInfo BytesRECV
```

[S8]

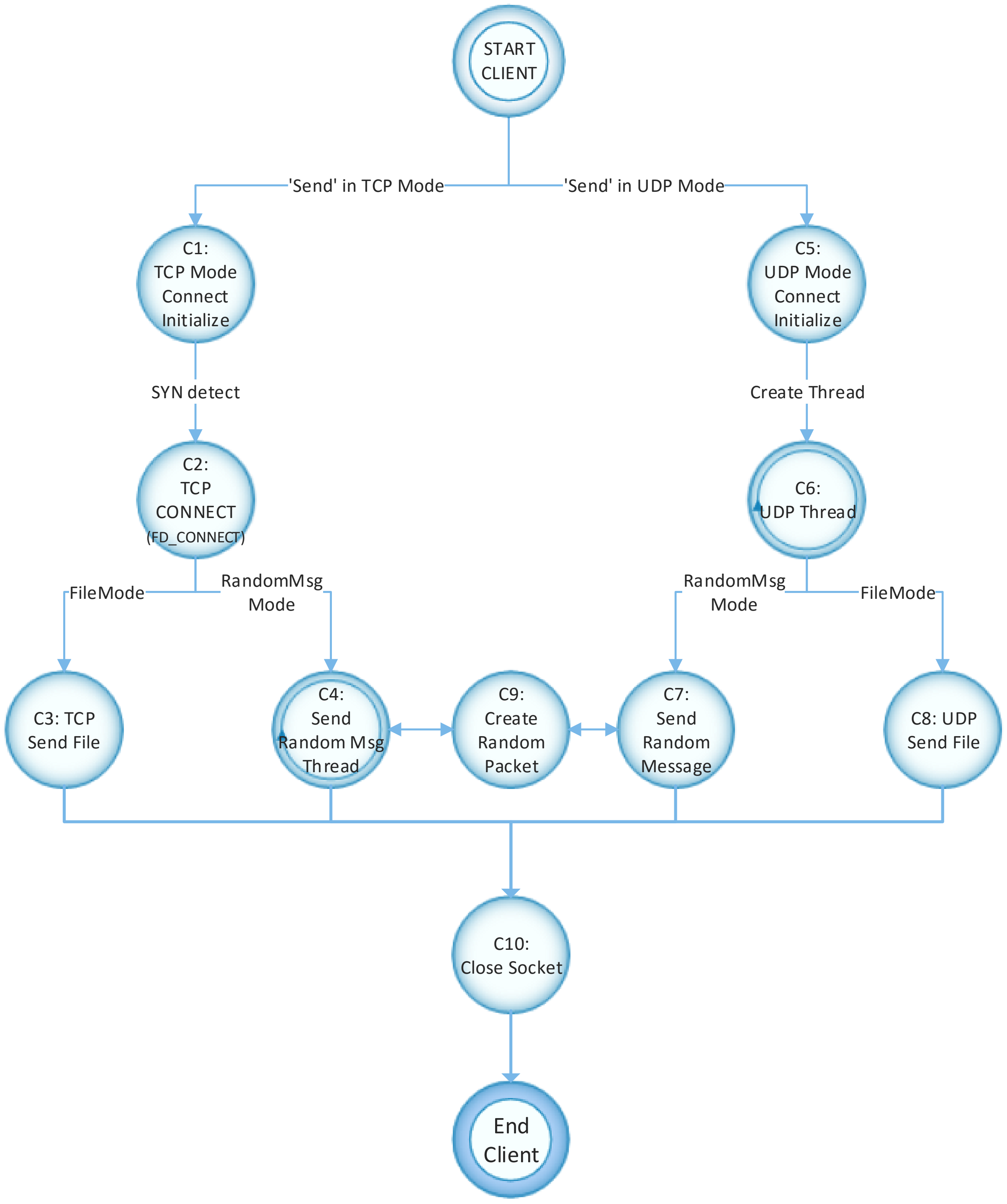
```
Function sv_disconn(socket)
    shotdown and close socket depending on Mode
    WSACleanup
```

[S9]

```
Function ServerResult()
    calculate data time (Last system time – first packet time)
    display result of transfer
```

DESIGN

State Diagram



**Assumption : FD_CLOSE close*

Psudocode

```
[START]
Function cl_winsock(HWND, socket,  op_protocol,  ip_address, port_number, messageType)

        Set ProtoMode to op_protocl
        Set MesgType to messageType

        open socket session info using WSAStartup
        ReadNotStart to ture;
        [C1]
        Open socket in TCP MODE
                (use WSASocket,   SOCK_STREAM, IPPROTO_TCP, OVERLAPPED flag)
                        If fail, display error message and return
                Set the event 'WM_TCPCLENT'  to detect FD_CONNECT|FD_CLOSE

                Get host info using gethostbyname
                        If nothing, display error message and return

                Initialize SOCKADDR_IN

                WSAConnect (Connect to Server : SYN+ACK)

                Break
        [C5]
        Open socket in UDP MODE
                (use WSASocket,   SOCK_DGRAM, IPPROTO_UDP, OVERLAPPED flag)
                        If fail, display error message and return
                Set the event 'WM_UDPCLIENT'  to detect FD_CLOSE


                Get host info using gethostbyname
                        If nothing, display error message and return

                Initialize SOCKADDR_IN

                Bind socket
                        If error display error message and return

                Create Thread UDPThread, pass socket pointer  (recv message)
                        If error display error message and return
                break
```

DESIGN

```
[C1]
Function ClientProc(hwnd, message, wParam, lParam, socket)
        Switch(message)
           case WM_TCPCLIENT:
                if detect error message
                        error message and return
                else
                        switch(WSAGETSELECTEVENT)
                        [C2]
                        case FD_CONNECT
                                Set the event 'WM_TCPCIENT'  to detect FD_WRITE|FD_CLOSE
                                Break

                        case FD_WRITE
                                if MessageType is MESGSEND
                                        createThread to TCPSendThread and pass socket info
                                                if fail, display errir message and return
                                else
                                        call TCPFileSend Function
                                break
                        case FD_CLOSE
                                close socket[C10]
                                break
            [C4]
            case UDPCLENT:
                if get WSAGETSELECERROR value, error message and return
                else
                    switch(WSAGETSELECTEVENT)
                        [C5]
                        case FD_CLOSE
                                closeSocket[C10]
                                break
```

[C3]

```
Function TCPFileSend(hwnd, socket)
        get packet size and filename to send from the dialog

        malloc character point 'packet' using  packetsize

        open file using filename to read in binary mode(rb+)
                if fail, display error message and return
        allocate SocketInfo structure (LPSOCKET INFORMATION type)

        initialize SI(SocketInfo)

        while(file not end : !feof())
                read file stream as much as 'packetsize' value
                update DataBuf.buf value in SI(SocketInfo)
                initialize Overlapped

                Send Message from client (WSASend)
                        If error detect and not IO_PENDDING or WSAWOULDBLOCK
                                Error message, close file and return
                If WSAWiatForMultipleEvents of OVERLAPPED is TIMEOUT
                        Close file, Disconnect socket and return[C10]
                add the send bytes amount to SocketInfo BytesSEND

        display Send information
        free packet memory
        close file
        close Socket[C10]
```

[C4 - Thread]

```
Function TCPSendThread(socket)
    get packet size and times to send from dialog

    malloc character point ‘packet’ using  packetsize

    allocate SocketInfo structure (LPSOCKET INFORMATION type)

    make ‘packet’ using  DummyPacket Function[C9]
    initialize SI(SocketInfo)

    while(i is from 0 to less than times to send)
        initialize Overlapped
        Send Message from client (WSASend)
            If error detect and not IO_PENDDING or WSAWOULDBLOCK
                Error message and return
        If WSAWiatForMultipleEvents of OVERLAPPED is TIMEOUT
            Disconnect socket and return
        add the send bytes amount to SocketInfo BytesSEND
    display Send information
    close Socket[C10]
    free packet memory
```

[C7]

```
Function UDPMesgSend(LPSOCKET_INFORMATION SI, SOCKADDR_IN netAddr, pksize, times)
    malloc character point ‘packet’ using  pksize
    make ‘packet’ using  DummyPacket Function[C9]
    initialize SI(SocketInfo)

    while(i is from 0 to less than times to send)
        initialize Overlapped

        Send Message from client (WSASend)
            If error detect and not IO_PENDDING or WSAWOULDBLOCK
                Error message and return
        If WSAWiatForMultipleEvents of OVERLAPPED is TIMEOUT
            Disconnect socket and return
        add the send bytes amount to SocketInfo BytesSEND

    display Send information
    free packet memory
```

[C8]

```
Function UDPFileSend(hwnd, socket)
    get packet size and filename to send from the dialog
    malloc character point ‘packet’ using  packetsize

    open file using filename to read in binary mode(rb+)
        if fail, display error message and return

    while(file not end : !feof())
        initialize Overlapped
        read file stream as much as ‘packetsize’ value
        update DataBuf.buf value in SI(SocketInfo)

        Send Message from client (WSASend)
            If error detect and not IO_PENDDING or WSAWOULDBLOCK
                Error message, close file and return
        If WSAWiatForMultipleEvents of OVERLAPPED is TIMEOUT
            Close file, Disconnect socket and return
        add the send bytes amount to SocketInfo BytesSEND
    Display result
    free packet memory
    close file
```

[C9]

```
Function DummyPacket(packetsize, *packet)
    Loop i from 0 to packetsize
        packet[i] = ‘a’;
    packet[i] = ‘\0’
```

[C10]

```
Function ClDISCONNECT (socket)
    Shutdown and close socket
    WSACleanup
```