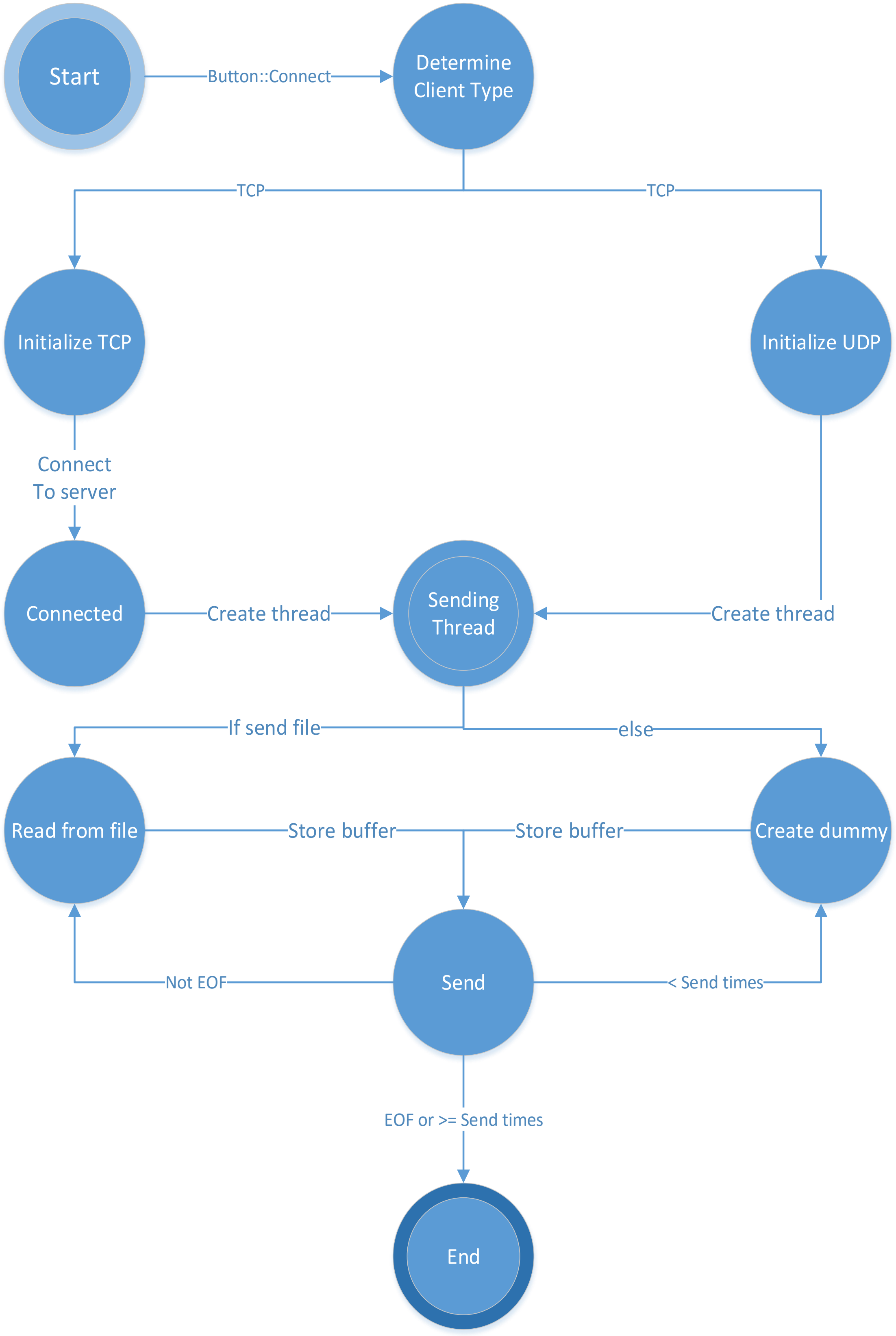
Client


Start
Button::Connect
Determine Client Type
TCP
TCP
Initialize TCP
Initialize UDP
Connect To server
Connected
Create thread
Sending Thread
Create thread
If send file
else
Read from file
Store buffer
Store buffer
Create dummy
Not EOF
Send
< Send times
EOF or >= Send times
End

Global defined structs

```
typedef struct _SOCKET_INFORMATION
{
	OVERLAPPED Overlapped;
	SOCKET Socket;
	CHAR Buffer[DATA_BUFSIZE];
	WSABUF DataBuf;
	DWORD BytesSEND;
	DWORD BytesRECV;
} SOCKET_INFORMATION, *LPSOCKET_INFORMATION;

typedef struct _TRANSMISSION_INFORMATION
{
	DWORD PacketSize;
	DWORD PacketsExpected;
	DWORD PacketsRECV;
	SYSTEMTIME StartTimeStamp;
	SYSTEMTIME EndTimeStamp;
	LPSTR ProtocolType;
} TRANSMISSION_INFORMATION, *LPTRANSMISSION_INFORMATION;
```

Client Pseudocode

Determine Server Type

```
Waits for windows messages received from WNDPROC
      Check the message and see if TCP radio button is selected
            If it is
                  Close winsock session if its already opened
                  transition to Initialize TCP
            Else
                  Close winsock session if its already opened
                  Transition to Initialize UDP
```

Initialize TCP

```
Start a WINSOCK session
Create a TCP socket for sending packet streams
Initialize the server address structure
      -  Get IP input from GUI
      -  Get Port number input from GUI
      -  Set sin_familiy to AF_INET
      -  Set IP address and port number
```

```
Transition to Connect
```

Connect

```
Call WSAConnect by passing in the server address and port number
If succeeded, create a thread to send packets to server
Transition to Send Thread
```

Initialize UDP

```
Start a WINSOCK session
Create a UDP socket for sending datagrams
Initialize the server address structure
     -  Get IP input from GUI
     -  Get Port number input from GUI
     -  Set sin_familiy to AF_INET
     -  Set IP address and port number
Bind the server address to the socket
Create a thread to send packets to server
Transition to Send Thread
```

Send Thread

```
Retrieve packet size from GUI
Retrieve send times from GUI
Allocate memory space for SOCKET_INFORMATION structure
Check if we’re sending a file
Send an initial control message with the format of “Packetsize.sendtimes”
If so
     Transition to Read from File
Else
     Transition to Create Dummy
```

Read from file

```
Open the file name specified by the GUI
While the file has not reached EOF
     Read packet size buffer into a buf
     Store the buf into the SOCKET_INFORMATION struct
     Transition to Send
     Clear the buf
Send a last packet with null characters to indicate EOT
```

```
Wait until server has done processing data
     Close the socket
     Exit
```

Create Dummy

```
Create a dummy packet of size packet size
Store the dummy packet into the SOCKET_INFORMATION struct
While I is less than send times
     Transition to Send
Send a last packet with null characters to indicate EOT
Wait until server has done processing data
     Close the socket
     Exit
```

Send

```
If we’re using TCP
     Call WSASend
If we’re using UDP
     Call WSASendTo, with the server address structure specified
```