```cpp
#define _WINSOCK_DEPRECATED_NO_WARNINGS
#include <iostream>
#include <winsock2.h>
#include <string>
#include <thread>
#pragma comment(lib, "Ws2_32.lib")

int const MAX_SOCKETS = 5;
SOCKET Aux_Socket;
SOCKET ClientSockets[MAX_SOCKETS + 1] = { SOCKET_ERROR };
bool Active_Sockets[MAX_SOCKETS + 1] = { false };

int find_available_socket(void) {
    int socket_number = MAX_SOCKETS;
    for (int i = 0; i < MAX_SOCKETS; i++) {
        if (!Active_Sockets[i]) {
            socket_number = i;
            break;
        }
    }
    return socket_number;
}

void Run(int Index) {
    std::cout << "Thread Started at Index " << Index << std::endl;
    Active_Sockets[Index] = true;
    while (true) {
        char RxBuffer[128] = { };
        memset(RxBuffer, 0, sizeof(RxBuffer));
        recv(ClientSockets[Index], RxBuffer, sizeof(RxBuffer), 0);
        if (sizeof(RxBuffer) != 0);
        {
            std::cout << "From Thread " << Index << " : ";
            std::cout << RxBuffer << std::endl;
            send(ClientSockets[Index], "Ok", sizeof("Ok"), 0);
            if (std::string(RxBuffer) == "[q]")
                break;
        }
    }
    std::cout << "Closing Connection" << std::endl;
    closesocket(ClientSockets[Index]);
    Active_Sockets[Index] = false;
}

int main(int argc, char* argv[]) {

    int Socket_Number;

    WSADATA wsaData;
    if (WSAStartup(MAKEWORD(2, 2), &wsaData) != 0) {
        std::cout << "Could not start DLLs" << std::endl;
        return 0;
    }

    SOCKET ListenSocket;
    ListenSocket = socket(AF_INET, SOCK_STREAM, IPPROTO_TCP);
    if (ListenSocket == INVALID_SOCKET) {
        std::cout << "Could not create socket" << std::endl;
        WSACleanup();
        return 0;
    }

    struct sockaddr_in SvrAddr;
    SvrAddr.sin_family = AF_INET;
    SvrAddr.sin_addr.s_addr = inet_addr("127.0.0.1");
    SvrAddr.sin_port = htons(27000);
    if (bind(ListenSocket, (struct sockaddr*)&SvrAddr,
        sizeof(SvrAddr)) == SOCKET_ERROR) {
        std::cout << "Could not bind socket to port" << std::endl;
        closesocket(ListenSocket);
        WSACleanup();
        return 0;
    }

    if (listen(ListenSocket, 1) == SOCKET_ERROR) {
        std::cout << "Could not start to listen" << std::endl;
        closesocket(ListenSocket);
        WSACleanup();
        return 0;
    }

    while (true) {
        std::cout << "Ready to accept a connection" << std::endl;
        Aux_Socket = accept(ListenSocket, NULL, NULL);
        if (Aux_Socket == SOCKET_ERROR) {
            return 0;
        }
        else {
            Socket_Number = find_available_socket();
            if (Socket_Number < MAX_SOCKETS) {
                ClientSockets[Socket_Number] = Aux_Socket;
                send(ClientSockets[Socket_Number], "Welcome",
                    sizeof("Welcome"), 0);
                std::thread(Run, Socket_Number).detach();
            }
            else {
                send(ClientSockets[MAX_SOCKETS], "Full",
                    sizeof("Full"), 0);
                std::cout << "Connection Fail" << std::endl;
            }
        }
    }
    closesocket(ListenSocket);
    WSACleanup();
    return 0;
}
```