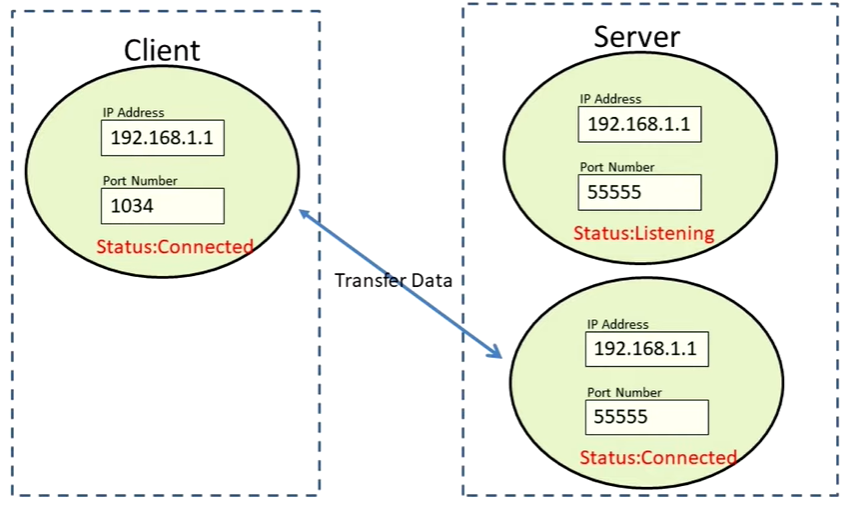
**C++ Network Programming**



“Client”, aynı IP adresine sahip “Server” a veri transferi için istek atar. “Server” ın ilgili portuna gelen mesaj, farklı bir soket oluştur. Oluşan bu soket, “Client” ile bilgi transferi yapabilecek duruma gelir. Bu sırada , “Client” ile iletişim içerisinde olan soket ise aynı IP adresi üzerindeki aynı veya farklı kullanıcılardan gelen veri taleplerini dinler.

-*Server Fonksiyonları:*

WSAStartup() (Windows Socket Application Programming Interface): Windows soketlerini başlatır.

Socket() :Soket oluşturur.

Bind() : Soket bağlantılarını kurmaya yarar.

Listen(): Server üzerindeki soketin dinlemesi için komut verir.

Accept(), connect(): Veri transferi için gereken onay işlemlerini gerçekleştirir.

Closesocket(): Server bağlantısını keser.

*-Client Fonksiyonları:*

WSAStartup() (Windows Socket Application Programming Interface): Windows soketlerini başlatır.

Socket() :Soket oluşturur.

Connect(): Server ile bağ kurar.

Recv(),send(),recvfrom(),sendto(): Client için veri transferi komutlarıdır.

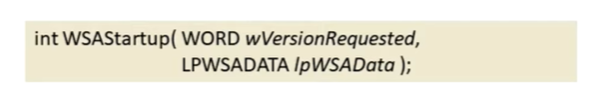
Disconnect() : Client, bağını keser.

**Port**

Ağdaki cihazların arasında veri iletişimini sağlayan numaralardır.Bir IP adresi ile birden fazla uygulamaya erişim sağlanabilir.her hizmet veya veri, bu kapı üzerinden kabul edilir.

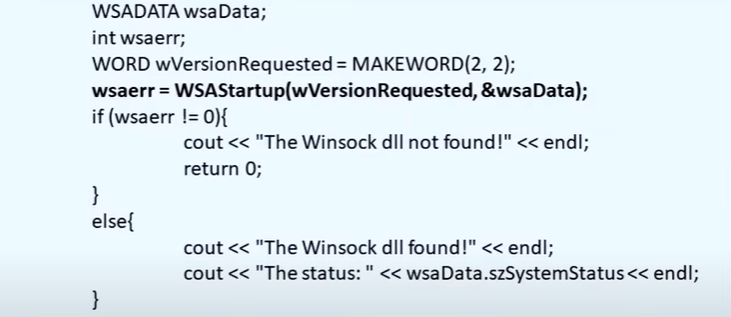
Bazi port numaraları standarttır. Mesela HTTP protokolü 80 numaralı portu kullanırken, HTTPS protokolü 443 numaralı portu kullanır.

**Soketlerin İşleyişi**

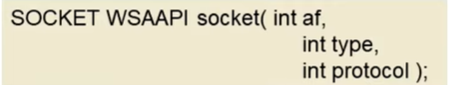


WSAStartup() sorgusu ile “Client” tarafından mesaj üretilir ve “Server” a iletilir. “WORD” bir sınıf değil türdür. 16bitlik bir tamsayıyı ifade eder. 32bitlik tamsayilar için “DWORD”kullanılır.

LPWSADATA kütüphanesiden oluşan obje, soketlerden gelen bilgiyi toplar.Eğer WSAStartup() fonksiyonu 0 dönerse işlem başarılıdır.



Herhangi bir soket oluturmak için;



soket fonksiyonu şekildeki gibi kullanılır.

“int af” parametresi, internetin adres ailesini temsil eder. 2 tane adres ailesi vardır.

* AF\_INET : IPv4 adreslerini temsil eden ailedir. 32 bitlik veri tutar.
* AF\_INET6: IPv6 adreslerini temsil eden ailedir. 128 bitlik veri tutar

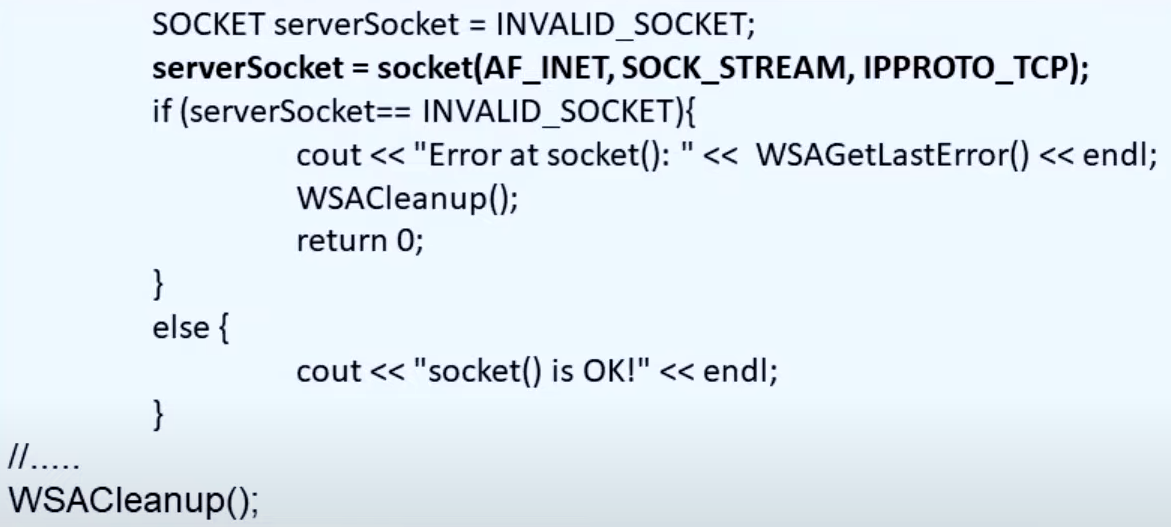
Her bir adres ailesi, belirli bir adres türünü ifade eder. Ağ programlamasında farklı protokolleri ve adres yapılarını temsil ederler.

“int type” parametresi, soket türünü ifade eder.

* TCP için SOCK\_STREAM, UDP için SOCK\_DGRAM kullanılır.

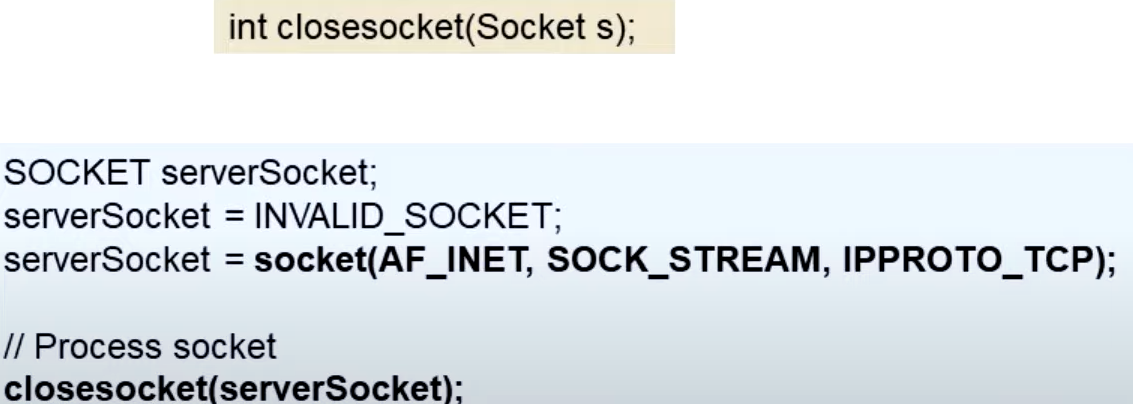
“int protocol” parametresi, bağlantının hangi protokolde olucağını temsil eder.

* IPPROTO\_UDP
* IPPROTO\_TCP



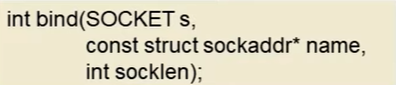
Soketler kullanıldıktan sonra içi boşaltılmalıdır. “WSACleanup()” bu yüzden kullanılır. Eğer “serverSocket” ifadesi -1 değerini döndürüyorsa “Server” soketi oluşturulamamıştır.

Veya, argüman olarak gönderilen soket nesnesi, “closesocket()” fonksiyonu ile sonlandırılmalıdır.



**Soketlerin Fonksiyonları**

1-bind(): “Server” ve “Client” soketlerini birbirine bağlayan fonksiyondur. Server.cpp içerisinde kullanılır.



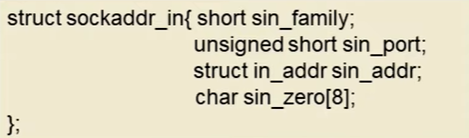
s: Bağlı olmayan bir server soketi temsil eder.

Name: Sokete gitmesi için verile adrestir.

socklen: Adresteki soketin boyutunu temsil eder.

Hata yoksa bind() fonksiyonu 0 döner. Aksi taktirde SOCKET\_ERROR hatası verilir.

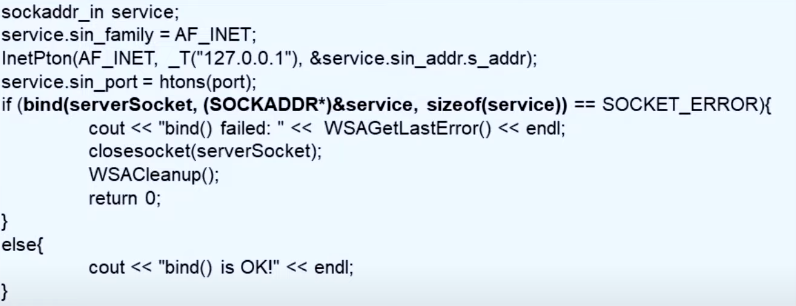
“const struct sockaddr\* name” içerisindeki “struct”;



sin\_family : AF\_INET deki adres ailesini temsil eder.

sin\_port : IP portunu ifade eder.

sin\_addr: IP adresini ifade eder.



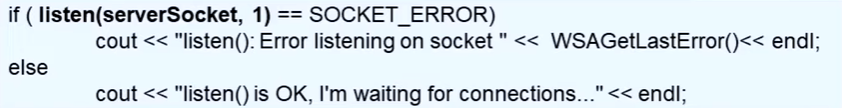
2-listen(): Sokete gelen bağlantıyı dinler.



s: Bağlanılmamış soketi temsil eder.

backlog: Izin verilen maksimum bağlantı sayısıdır.

“listen()” fonksiyonu eğer 0 dönerse hata yoktur, aksi taktirde SOCKET\_ERROR hatası alınır. TCP bağlantılarının kontrolü için kullanılır.



Şeklinde kontrol edilir.

3-recv(): TCP veya UDP bağlantılar için kullanılır. TCP bağlantısından gelen veriyi almak için kullanılır.

**İnt recv(SOCKET s, char\* buffer, int leght, int flags);**

S: Veri alınacak soketin nesnesidir. Genellikler “Server” soketidir.

buffer: Karakter iaşretçisinin gösterdiği alınacak verinin depolandığı bellek alanıdır.

Lenght: Beklenen verinin uzunluğudur.

Flags: İsteğe bağlı değiş

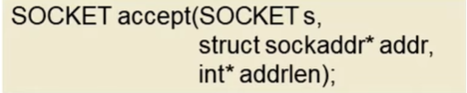
4-recvfrom(): recv() fonksiyonundan farklı olarak, UDP soketlerden veri alırken kaynağı da belirler.

**İnt recvfrom(SOCKET s, char\*buffer, int legth, int flags, struct sokaddr\*from, int\*fromlen);**

Sockaddr\*from: veri gelen sokete hangi soketten geldiğine dair bilgi içerir.

İnt\* fromlen: “from” yapsının boyutunu içerir.

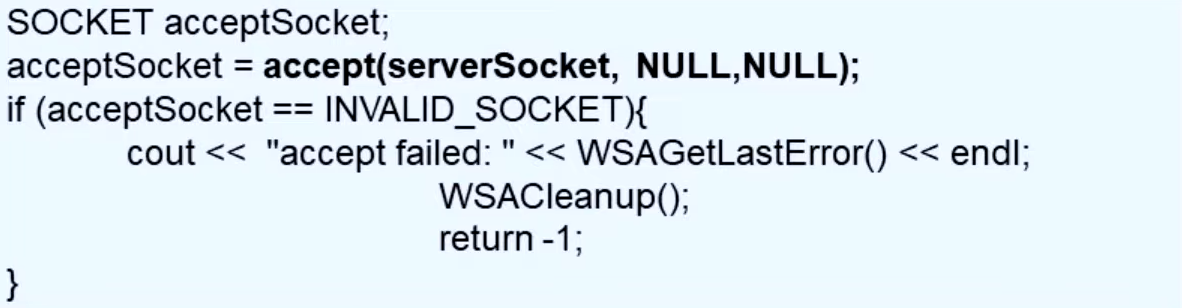
5-accept(): Sokete gelen bağlantıyı kabul eder. TCP bağlantıları için kullanılır , UDP bağlantısı için kullanılmaz.



s: Dinleyici soketi temsil eder.

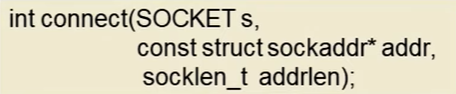
addr: “Client” IP adresini taşır.(opsiyonel)

addrlen: “Client” adresindeki “struct” ın boyutunu temsil eder.(opsiyonel)



Şeklinde kontrol edilir

6-connect(): “Client” ı “Server” a bağlayan komuttur.

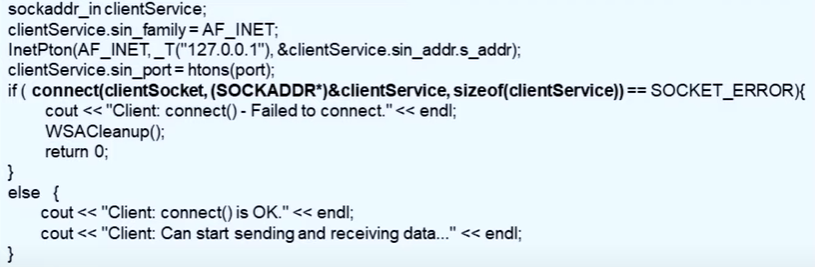


s: Bağlanılacak olan sokettir.

addr : Bağlanılacak olan “Server” ın IP adresini ve portunu içeren yapıdır.

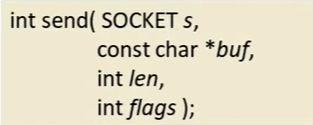
addrlen:Bağlanılacak sokete gelen verinin boyutunu döndürür.

“connect()” fonksiyonu eğer 0 dönerse doğru çalışıyordur, aksi taktirde “SOCKET\_ERROR” hatası alınır.



*Client Fonksiyonları*

1-send(): Bağlı sokete veri transferi yapmayı sağlar.



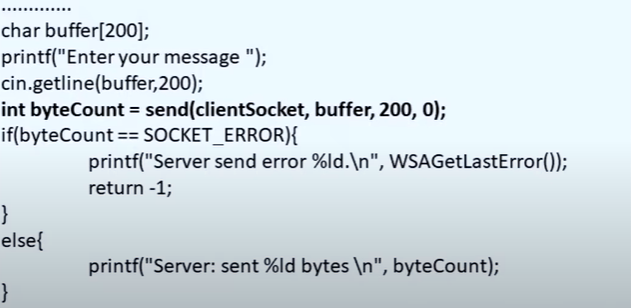
S:”Server” tarafından yollanan mesajı taşıyan nesnedir. Genellikle “clientSocket”bu görevi üstlenir.

Buf: “Buffer” a iletilen verileri gösteren işaretçidir.

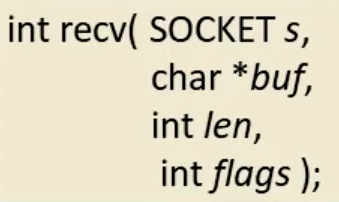
Len: “Buffer” a gelen verilerin boyutudur.

Flags: Bu fonksiyonun davranışını etkileyen etmenlerdir.

Eğer “send()” fonksiyonu doğru çalışıyorsa, gönderilen “byte” sayısını döner.Aksi taktirde SOCKET\_ERROR görülür.



2. recv(): Bağlı soketten gelen verileri taşıyan fonksiyondur.



S: Bağlı soketi tanımlayan nesnedir.

Buf: Gelen verilerin geçtiği “buffer” ı gösteren işaretçidir.

Len: “buf” işaretçisinin gösterdiği “buffer” ın “byte” türünden boyutudur.

Eğer “recv()” fonksiyonu düzgün çalışıyorsa , gelen verilerin boyutunu “byte” cinsinden döndürür. Bağlantı kesildiğinde fonksiyon 0 dönüyorsa doğru çalışıyordur, aksi taktirde SOCKET\_ERROR hatası alınır.

***Örnek bir Server;***

#include <iostream>

#include <winsock2.h>

#include <ws2tcpip.h>

#include <Windows.h>

#pragma comment(lib, "ws2\_32.lib")

int main() {

WSADATA wsadata;

WORD wVersionRequested = MAKEWORD(2, 2); //WORD, ağ sınıfından 16 bitlik bir tamsayı tutan veri türüdür, Ağ paketlerinde başlık bölümündeki değerleri temsil etmek için kullanılır.Bu değerler, paketin uzunluğu, protokol numarası, hata denetimi gibi bilgileri içerebilir.

int wsaerr = WSAStartup(wVersionRequested, &wsadata);

if (wsaerr != 0) {

std::cout << "Windows Server sockets are not found!" << std::endl;

WSACleanup();

}

else {

std::cout << "Windows Server sockets are found!" << std::endl;

}

SOCKET myServerSocket = socket(AF\_INET, SOCK\_DGRAM, IPPROTO\_UDP); //SOCK\_DGRAM gönderilen UDP dosyalarını temsil eder, SOCK\_DGRAM herhangi bir socket nesnesine atanmalıdır.

if (myServerSocket == -1) {

std::cout << "Server socket is not available!" << std::endl;

WSACleanup();

}

else {

std::cout << "Server socket is available!" << std::endl;

//std::cout << "Server port is: " << getservbyport << std::endl;

}

struct sockaddr\_in servaddr; //sockaddr\_in yapısı IPv4 tabanlı soket adreslerini temsil eder.Structın içerisinde IP adresi, port numarası ve IPv4 adresi yer alır.

servaddr.sin\_family = AF\_INET;

inet\_pton(AF\_INET, "127.0.0.1", &(servaddr.sin\_addr));

servaddr.sin\_port = htons(8000);

if (bind(myServerSocket, (struct sockaddr\*)&servaddr, sizeof(servaddr)) != 0) {

std::cout << "Binding has crashed! " << WSAGetLastError() << std::endl;

closesocket(myServerSocket);

}

else {

std::cout << "Binding is successful!" << std::endl;

}

char buffer[128];

struct sockaddr\_in clientaddr;

int client\_addr\_len = sizeof(clientaddr);

int receivedData = 0;

//int receivedData

//= recvfrom(myServerSocket,(char\*)buffer, sizeof(buffer), 0, (struct sockaddr\*)&clientaddr, &client\_addr\_len); //recvfrom() , alınan verinin uzunluğunu byte cinsinden döndürür.

//if (accept(myServerSocket, (struct sockaddr\*)&servaddr, &clientMessageLength) == -1) { //server bağlantı gerektirmediğinden dolayı UDP için, accept kullanılmaz.

//std::cout << "Connection is not accepted!" << WSAGetLastError() << std::endl;

//}

//else {

//std::cout << "Connection accepted by the server!" << std::endl;

//}

while (true) {

//if (listen(myServerSocket, 5) == 0) //listen() fonksiyonu, server soketinin bağlantıyı kabul etmeye hazır olup olmadığını kontrol eder fakat UDP soketleri bağlantı gerektirmezler! TCP bağlantıları için kullanılır.

receivedData = recvfrom(myServerSocket,buffer, sizeof(buffer), 0, (struct sockaddr\*)&clientaddr, &client\_addr\_len);

buffer[receivedData] = '\0';

if (receivedData > 0) {

std::cout << "Data successfully received!" << std::endl;

std::cout << receivedData << " bytes received!" << std::endl;

std::cout << buffer << std::endl;

}

};

closesocket(myServerSocket);

WSACleanup();

};

#include <iostream>

#include <winsock2.h>

#include <ws2tcpip.h>

#include <windows.h>

#pragma comment(lib, "ws2\_32.lib")

int main() {

WSADATA wsadata2;

WORD Request = MAKEWORD(2, 2); //WORD, ağ sınıfından 16 bitlik bir tamsayı tutan veri türüdür, Ağ paketlerinde başlık bölümündeki değerleri temsil etmek için kullanılır.Bu değerler, paketin uzunluğu, protokol numarası, hata denetimi gibi bilgileri içerebilir.

int wsaerr2 = WSAStartup(Request, &wsadata2);

if (wsaerr2 != 0) {

std::cout << "Windows Server sockets are not found!" << std::endl;

WSACleanup();

}

else {

std::cout << "Windows Server sockets are found!" << std::endl;

}

SOCKET myClientServer = socket(AF\_INET, SOCK\_DGRAM, 0);

if (myClientServer == -1) {

std::cout << "Client Socket was not created!" << std::endl;

return 1;

}

else {

std::cout << "Client socket was created!" << std::endl;

}

const char\* message = "Merhaba bu bir test mesaji.";

sockaddr\_in destinationAdress{};

destinationAdress.sin\_family = AF\_INET;

inet\_pton(AF\_INET, "127.0.0.1", &(destinationAdress.sin\_addr));

destinationAdress.sin\_port = htons(8000);

int sent\_bytes = sendto(myClientServer, message, strlen(message), 0, (struct sockaddr\*)&destinationAdress, sizeof(destinationAdress));

if (sent\_bytes == -1) {

std::cout << "Data was not sent!" << std::endl;

closesocket(myClientServer);

WSACleanup();

return 1;

}

else {

std::cout << "Data sent: " << sent\_bytes << std::endl;

}

closesocket(myClientServer);

return 0;

}

***Örnek bir Client;***

#include <iostream>

#include <winsock2.h>

#include <ws2tcpip.h>

#include <windows.h>

#pragma comment(lib, "ws2\_32.lib")

int main() {

WSADATA wsadata2;

WORD Request = MAKEWORD(2, 2); //WORD, ağ sınıfından 16 bitlik bir tamsayı tutan veri türüdür, Ağ paketlerinde başlık bölümündeki değerleri temsil etmek için kullanılır.Bu değerler, paketin uzunluğu, protokol numarası, hata denetimi gibi bilgileri içerebilir.

int wsaerr2 = WSAStartup(Request, &wsadata2);

if (wsaerr2 != 0) {

std::cout << "Windows Server sockets are not found!" << std::endl;

WSACleanup();

}

else {

std::cout << "Windows Server sockets are found!" << std::endl;

}

SOCKET myClientServer = socket(AF\_INET, SOCK\_DGRAM, 0);

if (myClientServer == -1) {

std::cout << "Client Socket was not created!" << std::endl;

return 1;

}

else {

std::cout << "Client socket was created!" << std::endl;

}

const char\* message = "Merhaba bu bir test mesaji.";

sockaddr\_in destinationAdress{};

destinationAdress.sin\_family = AF\_INET;

inet\_pton(AF\_INET, "127.0.0.1", &(destinationAdress.sin\_addr));

destinationAdress.sin\_port = htons(8000);

int sent\_bytes = sendto(myClientServer, message, strlen(message), 0, (struct sockaddr\*)&destinationAdress, sizeof(destinationAdress));

if (sent\_bytes == -1) {

std::cout << "Data was not sent!" << std::endl;

closesocket(myClientServer);

WSACleanup();

return 1;

}

else {

std::cout << "Data sent: " << sent\_bytes << std::endl;

}

closesocket(myClientServer);

return 0;

}