|  |
| --- |
| Comp 4985 |
| Assignment 2 |
| Client/Server Application in Windows |

|  |
| --- |
| Mateusz Siwoski A00758640  2/12/2014 |

Table of Contents

[Abstract: 1](#_Toc379935929)

[Introduction: 2](#_Toc379935930)

[Analysis: 3](#_Toc379935931)

[Sending/Receiving Through TCP: 3](#_Toc379935932)

[Sending 1 Packet: 3](#_Toc379935933)

[Sending 10 packets: 4](#_Toc379935934)

[Sending 100 Packets: 4](#_Toc379935935)

[Sending/Receiving over UDP: 5](#_Toc379935936)

[Sending 1 Packet: 5](#_Toc379935937)

[Sending 10 Packets: 5](#_Toc379935938)

[Sending 100 Packets: 6](#_Toc379935939)

[Loss of Packets over UDP: 6](#_Toc379935940)

[Interesting Occurrences: 6](#_Toc379935941)

[Conclusion: 7](#_Toc379935942)

[State Diagram: 8](#_Toc379935943)

[Pseudocode: 9](#_Toc379935944)

[Test Cases: 11](#_Toc379935945)

[Figures: 12](#_Toc379935946)

[Figure 1: 12](#_Toc379935947)

[Figure 2: 12](#_Toc379935948)

[Figure 3: 13](#_Toc379935949)

[Figure 4a/4b/4c/4d: 13](#_Toc379935950)

[Figure 5a/5b/5c: 14](#_Toc379935951)

[Figure 6a/6b: 15](#_Toc379935952)

[Figure 7: 16](#_Toc379935953)

[Tabular Data: 17](#_Toc379935954)

[UDP: 17](#_Toc379935955)

[TCP: 18](#_Toc379935956)

# Abstract:

An analysis on the TCP/IP protocol suite, specifically the UDP and TCP protocols, has shown that TCP is more reliable when sending datagrams between two Window workstations over the LAN. The analysis has shown that as the number of times the packet being sent increases, or packet size increases, the reliability decreases for UDP. This reliability is measured by the number of packets sent versus the number received. TCP, although has more reliability, has more overhead to ensure reliability such as accepting a connection, which meant that transfer times for TCP were longer than UDP.

# Introduction:

The application is a Windows program that is able to generate TCP/UDP datagrams and transfer the data using the TCP/IP protocol suite between two Windows machines. The application has three general states: Control, Client and Server.

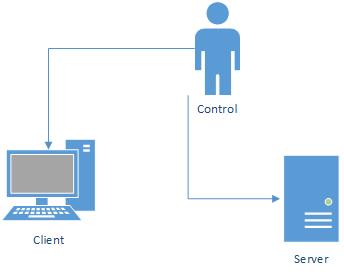
****

Figure 1: Three States of the Client Server Application

The Control state is just the initial state the program enters. The Client state allows datagrams to be sent to a specified user. The user is able to specify the port and protocol being used, the size of the datagrams and the number of times to send the datagram. As well, the user will be able to send data from a file. The Server state allows for datagrams to be received. The user will specify the port that the server will be open on and the type of protocol being used. Upon receiving all the packets, the user will be able to save the data as a file.  
 Further to the application, an analysis was done to test whether TCP or UDP is more reliable. The tests that were run were on the data being sent and received among different packet sizes (1024 Bytes, 4096 Bytes, 20 Kb, 60 kb and higher). As well, the packets were sent in different multiples, from 1 to 100. Analysis has shown that TCP is more reliable, compared with UDP and that as the packet sized increased or the number of packets being sent between the server and client increased, UDP reliability decreased.   
 The application was written in C++ with the Windows Forms .net framework.

# Analysis:

The purpose of the application is to test the reliability of different TCP/IP protocol suites, specifically the UDP and TCP protocols. The analysis of the two different protocols involved sending datagrams of different sizes, with a minimum of 1024 bytes per packet and sending the packet multiple times, specifically 1, 10 and 100 times. The data being sent was done over the LAN between two different machines. Wire Shark was used to confirm the data being sent and received by the client and the server. Control packets were always sent before the actual data was sent so that the server would know how many packets they should expect and what size each packet should be. Due to the implementation of the control packet, the Client/Server program works with Jordan Marling’s program as he is implementing a control packet with similar stats.

## Sending/Receiving Through TCP:

The files were sent at 1 KB, 10 KB, 20 KB, 40 KB, 60 KB, 80 KB and 100 KB and they were sent in multiples of 1, 10 and 100 packets at a time. The tabular data for TCP is located on page 18.

### Sending 1 Packet:

### Sending 10 packets:

### Sending 100 Packets:

The three charts show that as the packet size increased, the time for the packets to be sent increased linearly. Furthermore, testing over TCP showed that there was no packet loss even though the packet size or number of times increased.

## Sending/Receiving over UDP:

The files were sent at 1 KB, 10 KB, 20 KB, 40 KB, 60 KB and 64 KB. Many of these tests were run multiple times as the reliability of UDP and the implementation of a control packet meant that the data being sent would not necessarily arrive in order or at all, causing data gathering to be quite difficult. The tabular data for UDP is located on page 17.

### Sending 1 Packet:

### Sending 10 Packets:

### Sending 100 Packets:

### Loss of Packets over UDP:

The data gathered during the loss of packets phase of testing shows that UDP, although quicker, was extremely unreliable and packet loss increased considerably as the packet size increased or the number of times increased.

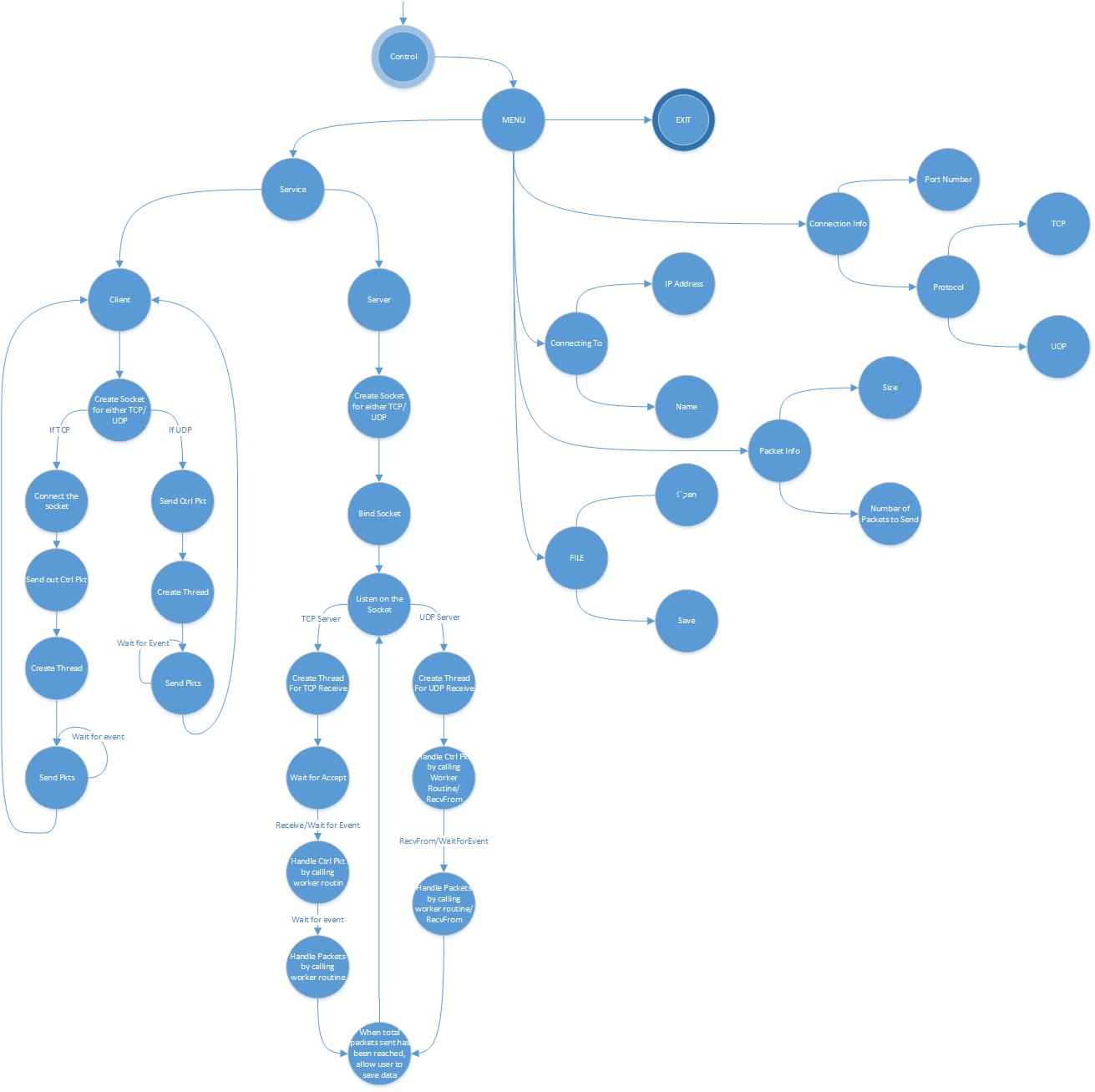
## Interesting Occurrences:

When sending 1 packet over TCP only once, the total time taken for the process from start to finish was greater than sending 10 or 100 packets at a time. It is suspected that Nagle’s Algorithm is the cause for this delay. As well, during UDP transmission, receiving packets of 40 kb at 100 times produced some irregular results, which was unexplainable.

# Conclusion:

Through the data gathered, it is shown that TCP is more reliable with no packet loss, while UDP, although quicker, will have packet loss. UDP packet loss increases as the packet size increases and the number of times the packet gets sent to the server. When running the application for TCP, no packets were lost in the process, but as the packet sized increased or the number of times the packet was sent, the time for receiving all the packets increased linearly. These results support the process of TCP as TCP must ensure there is a 3 way handshake before any data is sent.  
 It is conclusive that UDP is less reliable than TCP but UDP is much faster as there is no 3 way handshake occurring. Due to the speed of UDP, the application of UDP would be ideal for situations such as multiplayer games or other time sensitive applications because waiting for packets is not preferred. TCP would be important for applications where data must arrive completely to ensure accuracy such as web browsing. The circumstance in which a user would use TCP or UDP would really be determined whether the information being sent must be accurate and error checked or not, as TCP will do this while UDP will not.

# State Diagram:



# Pseudocode:

System::Void MyForm::MyForm\_Load(System::Object^ sender, System::EventArgs^ e){

Set the WSAStartup

}

System::Void MyForm::clientButton\_Click(System::Object^ sender, System::EventArgs^ e){

Send button is active

Receive button is unactive

}

System::Void MyForm::serverButton\_Click(System::Object^ sender, System::EventArgs^ e){

Send button is unactive

Receive button is active

}

System::Void MyForm::openButton\_Click(System::Object^ sender, System::EventArgs^ e){

Open the dialog box

Store the file into a buffer file

}

System::Void MyForm::saveButton\_Click(System::Object^ sender, System::EventArgs^ e){

Open the save dialog box

Save the file to the system

}

System::Void MyForm::receiveButton\_Click(System::Object^ sender, System::EventArgs^ e){

Initialize socket information

Store information in control packet

If TCP is selected

Create a socket for TCP

else

Create a socket for UDP

Bind on the newly created socket

if TCP

Listen on the socket

create a thread for TCP

else

create a thread for UDP

}

System::Void MyForm::sendButton\_Click(System::Object^ sender, System::EventArgs^ e){

Initialize socket information

Store information in control packet

Create an event

if TCP

Create a socket for TCP

Connect the socket

Send the control packet

create the thread for TCP

else

Create a socket for UDP

Send the control packet

Create the thread for UDP

}

DWORD WINAPI TCPWorkerThread(LPVOID){

while true{

Accept the socket

Initialize all the socket information

WSARecv and pass in the TCPCtrlWorkerRoutine

Wait for multiple events

Reset the event

}

return true;

}

DWORD WINAPI UDPWorkerThread(LPVOID){

Initialize all the socket information

WSARecvFrom and pass in the UDPCtrlWorkerRoutine

Wait for multiple events

Reset the event

}

VOID CALLBACK TCPCtrlWorkerRoutine(DWORD, DWORD, LPWSAOVERLAPPED, DWORD){

Get the control packet information and save it in a struct

WSARecv and pass in the TCPWorkerRoutine

Wait for multiple events

Reset the event

}

VOID CALLBACK TCPWorkerRoutine(DWORD, DWORD, LPWSAOVERLAPPED, DWORD){

if the packets received is the same as the control packet amount

print off the packet info

WSARecv (pass in the TCPWorkerRoutine

Wait for multiple events

Reset the event

}

VOID CALLBACK UDPCtrlWorkerRoutine(DWORD, DWORD, LPWSAOVERLAPPED, DWORD){

Get the control packet information and save it in a struct

WSARecvFrom and pass in the UDPWorkerRoutine

Wait for multiple events

Reset the event

}

VOID CALLBACK UDPWorkerRoutine(DWORD, DWORD, LPWSAOVERLAPPED, DWORD){

if the packets received is the same as the control packet amount

print off the packet info

WSARecvFrom and pass in the UDPWorkerRoutine

Wait for multiple events

Reset the event

}

DWORD WINAPI TCPSenderThread(LPVOID){

If packets sent is less than the amount set by the user

Send the packet

Wait for multiple events

Reset event

}

DWORD WINAPI UDPSenderThread(LPVOID){

If packets sent is less than the amount set by the user

Send the packet

Wait for multiple events

Reset event

}

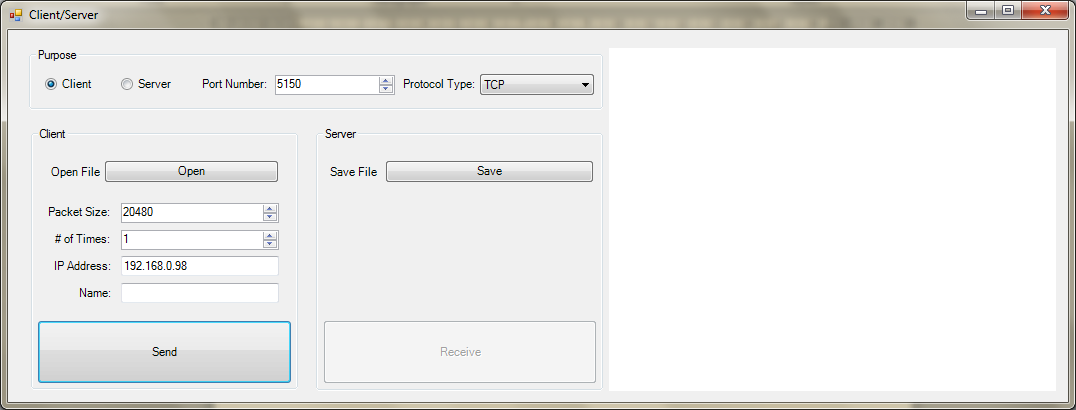
long delay (SYSTEMTIME t1, SYSTEMTIME t2){get system time}

# Test Cases:

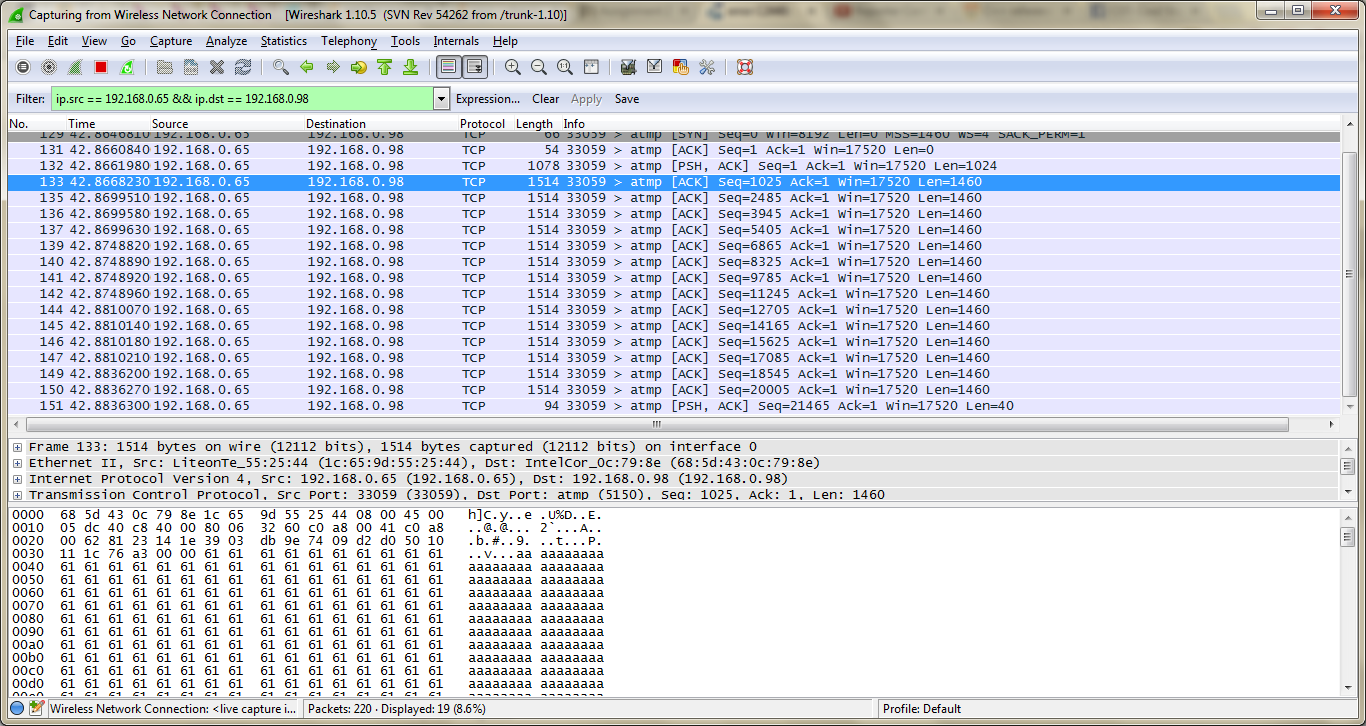
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Test** | **Tests Description** | **Tools Used** | **Expected Result** | **Pass/Fail** | **Notes** |
| 1 | Open an Initial Window | Windows Application | The initial window should open with the menu options | Pass | See Figure 1 |
| 2 | Send Data over TCP | Windows Application/ WireShark | The data is suppose to be sent to the client. 3 way handshake occurs and data is sent. | Pass | See Figure 2 |
| 3 | Send Data over UDP | Windows Application/ WireShark | The data is suppose to be sent to the client. | Pass | See Figure 3 |
| 4 | Receive Data over UDP | Windows Application/ WireShark | Data is received by the client over UDP | Pass | See Figure 4a/4b/4c/4d |
| 5 | Receive data over TCP | Windows Application/ WireShark | Data is received by the client over UDP | Pass | See Figure 5a/5b/5c |
| 6 | Connect to Server (TCP) | Windows Application | The client should be able to connect to a server. | FAIL | See Figure 6a/6b |
| 7 | Connect to Server (UDP) | Windows Application | The client should be able to connect to a server | Pass | See Figure 7 |

# Figures:

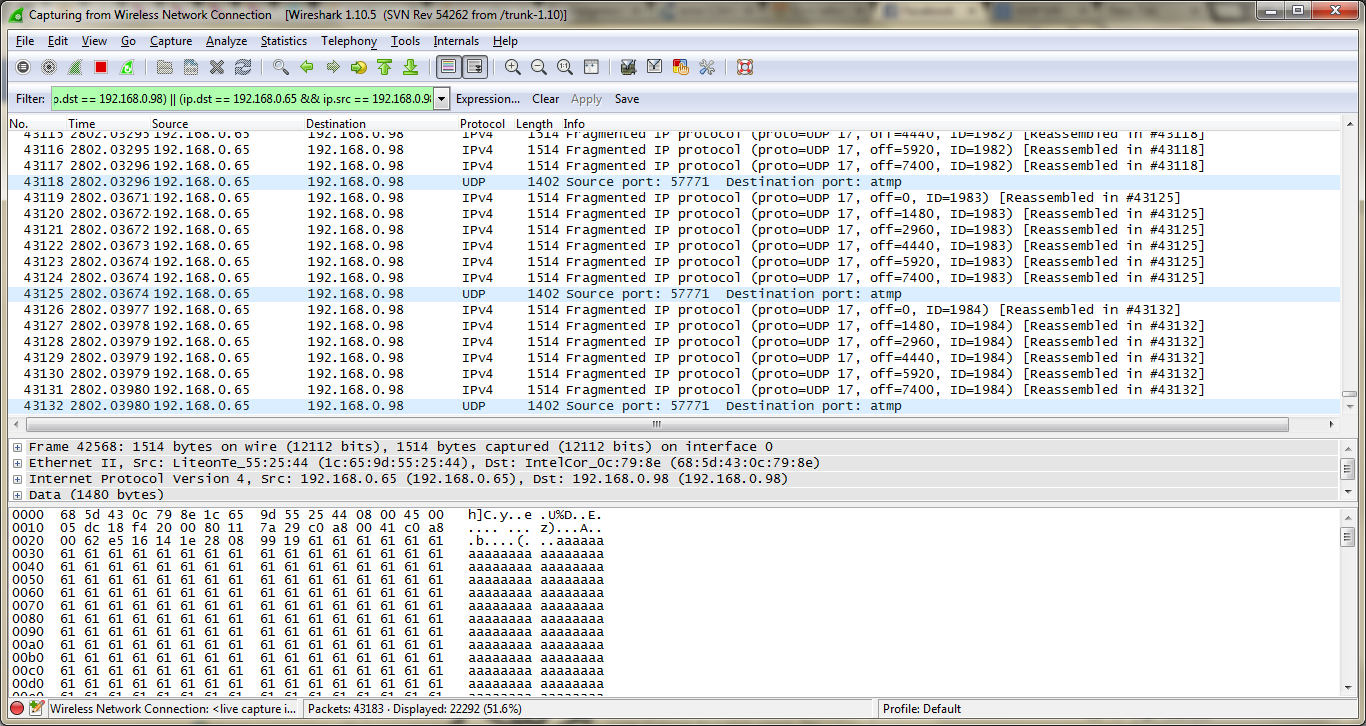
## Figure 1:



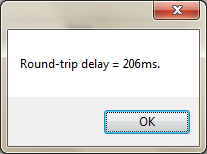
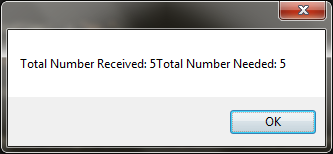
## Figure 2:

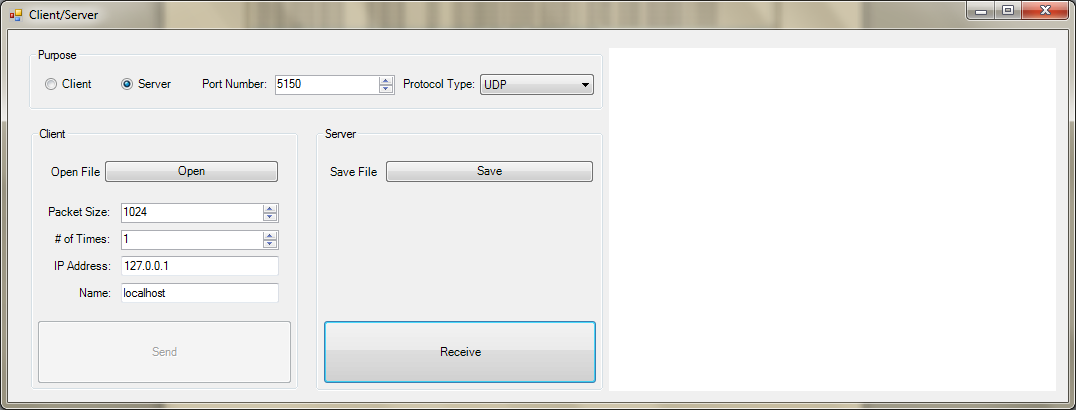


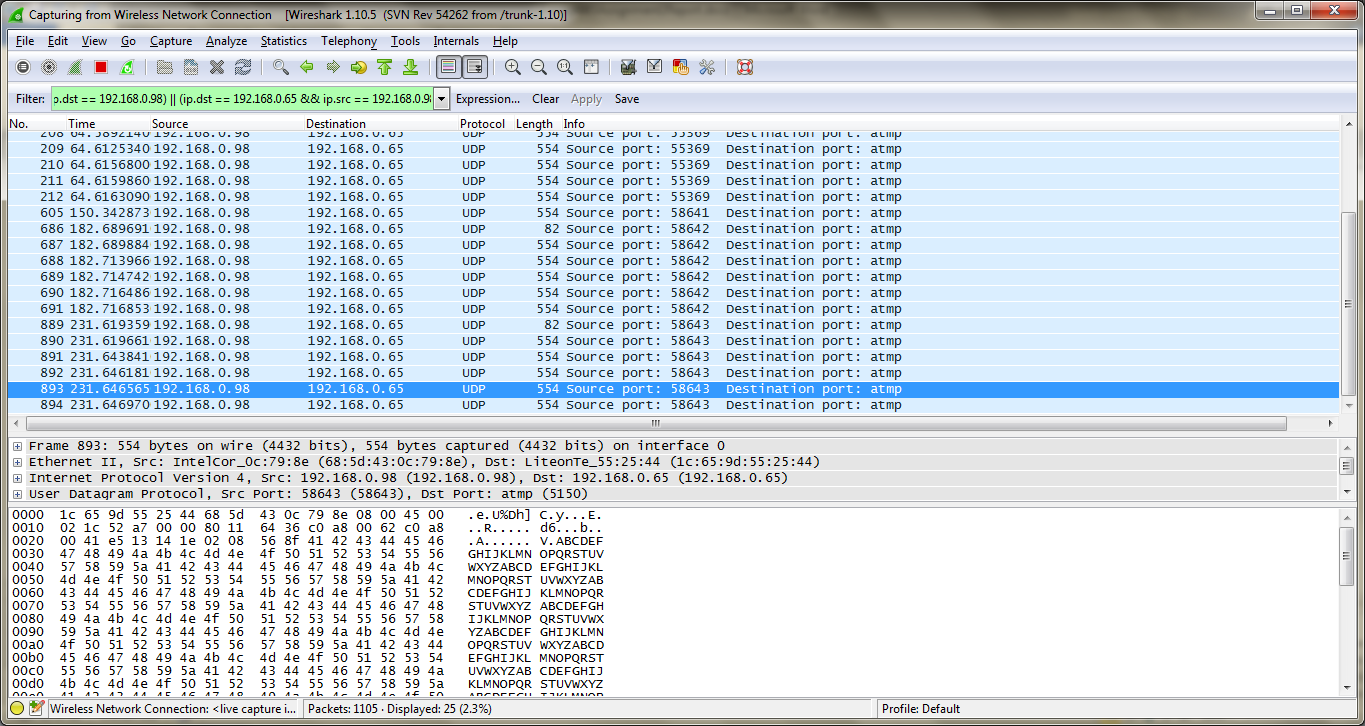
## Figure 3:



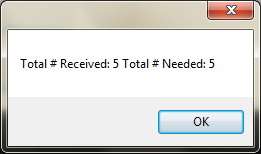
## Figure 4a/4b/4c/4d:

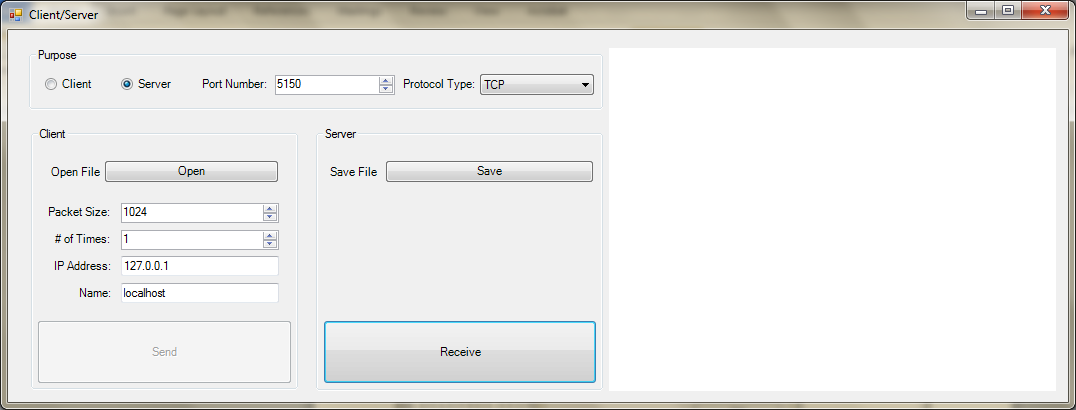
 

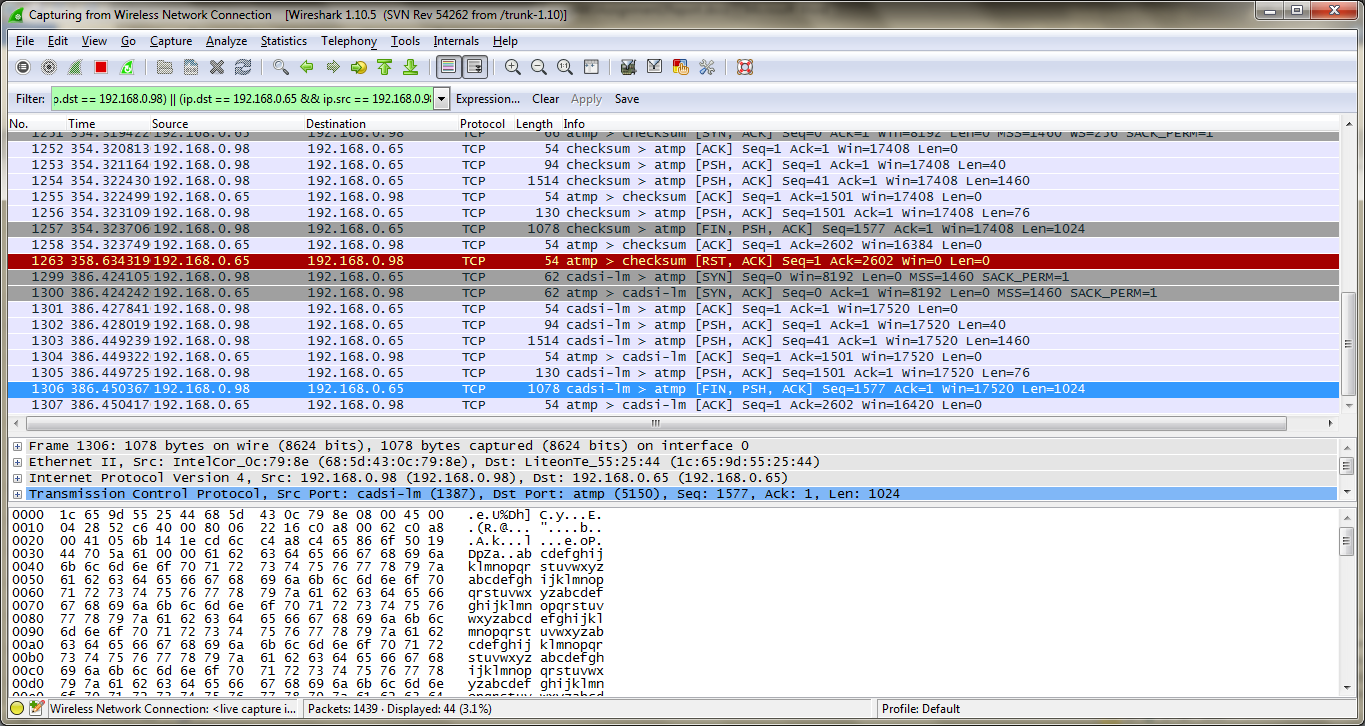




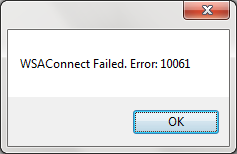
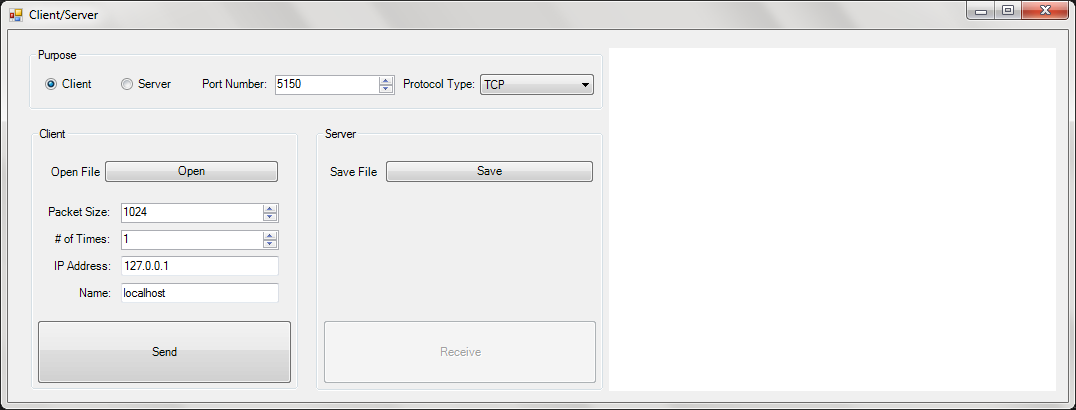
## Figure 5a/5b/5c:



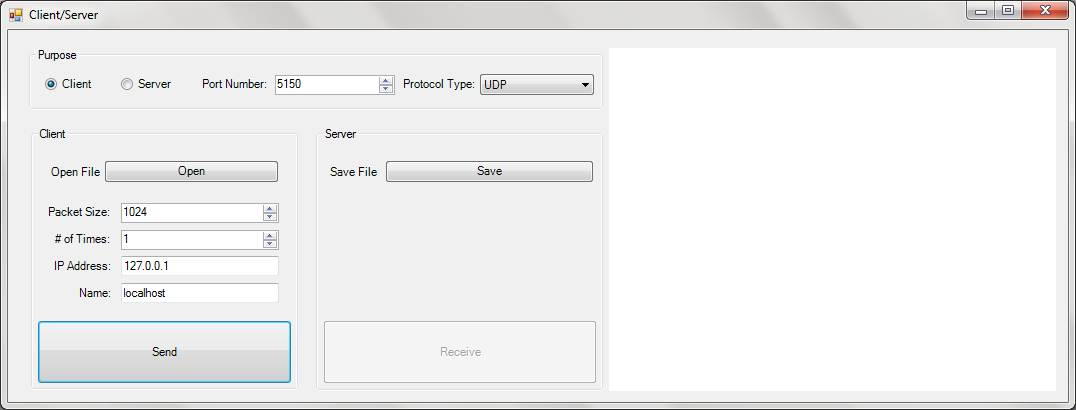




## Figure 6a/6b:

## Figure 7:



# Tabular Data:

## UDP:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| UDP | | | | |
| Times Sent | KB | Milliseconds | Packets Received | Times Sent |
| 1 | 1 | 2 | 1 | 1 |
| 10 | 1 | 21 | 10 | 10 |
| 100 | 1 | 110 | 100 | 100 |
| 1 | 10 | 8 | 1 | 1 |
| 10 | 10 | 149 | 10 | 10 |
| 100 | 10 | 763 | 36 | 100 |
| 1 | 20 | 18 | 1 | 1 |
| 10 | 20 | 249 | 10 | 10 |
| 100 | 20 | 731 | 12 | 100 |
| 1 | 40 | 35 | 1 | 1 |
| 10 | 40 | 412 | 7 | 10 |
| 100 | 40 | 2212 | 15 | 100 |
| 1 | 60 | 69 | 1 | 1 |
| 10 | 60 | 377 | 4 | 10 |
| 100 | 60 | 435 | 4 | 100 |
| 1 | 64 | 70 | 1 | 1 |
| 10 | 64 | 548 | 5 | 10 |
| 100 | 64 | 833 | 7 | 100 |

## TCP:

|  |  |  |
| --- | --- | --- |
| TCP | | |
| Times Sent | KB | Milliseconds |
| 1 | 1 | 213 |
| 10 | 1 | 7 |
| 100 | 1 | 98 |
| 1 | 10 | 14 |
| 10 | 10 | 121 |
| 100 | 10 | 1066 |
| 1 | 20 | 63 |
| 10 | 20 | 201 |
| 100 | 20 | 2322 |
| 1 | 40 | 62 |
| 10 | 40 | 387 |
| 100 | 40 | 3163 |
| 1 | 60 | 60 |
| 10 | 60 | 429 |
| 100 | 60 | 4293 |
| 1 | 80 | 136 |
| 10 | 80 | 604 |
| 100 | 80 | 5038 |
| 1 | 100 | 127 |
| 10 | 100 | 765 |
| 100 | 100 | 6303 |