Comparing TCP and UDP Speed and Packet Loss On LAN and WAN

Shane Spoor

Table of Contents

Abstract

The Transport Control Protocol (TCP) and User Datagram Protocol (UDP) are often presented as opposites: TCP is slow, reliable and connection oriented, while UDP is fast, unreliable and connectionless. This paper analyses the transfer speed and number of lost packets of each protocol on a wired Local Area Network (LAN), wireless LAN, and Wide Area Network (WAN) using a program designed for this purpose. The results indicate that TCP is highly reliable, that UDP is faster and less reliable than TCP on a WAN or wireless LAN, and that TCP can be faster than UDP on a wired LAN.

Introduction

There were a number of constraints for this experiment. First and most important was the need for a timing mechanism. This had to have a reasonable degree of accuracy and a high enough resolution to collect meaningful results. Secondly, the program needed the flexibility to change the packet size, number of packets to send, and protocol used to send through the interface to the collection of a variety of data. Thirdly, the program required some logging functionality to collect the statistics.

The program is implemented using the Windows API (see Appendices B and C for the program's state chart diagrams and pseudocode listings). To measure the transfer time, the program obtains a timestamp at the beginning and end of each transmission on both the receiver and sender sides. The timestamp has a resolution of 100 nanosecond intervals, which the program truncates to millisecond values. GUI items allow the user to select number of packets to send, packet size, protocol, and whether they are sending or receiving. The statistics for each transfer, including bytes transferred, packets expected, and transfer time are logged to a file for later review.

The interaction between the user, the sending program and the receiver is illustrated below. The experiment was carried out by sending 1KB, 4KB, 20KB and 60KB packets in bursts of 10 and 100, and each transfer was repeated 3 times. After each transfer, the data was logged for review.

Analysis

The following sections compare the average transfer times for TCP and UDP packets and the average packet UDP packet loss in a variety of environments. See Appendix A for the individual data points composing the averaged statistics.

Wired Local Area Network

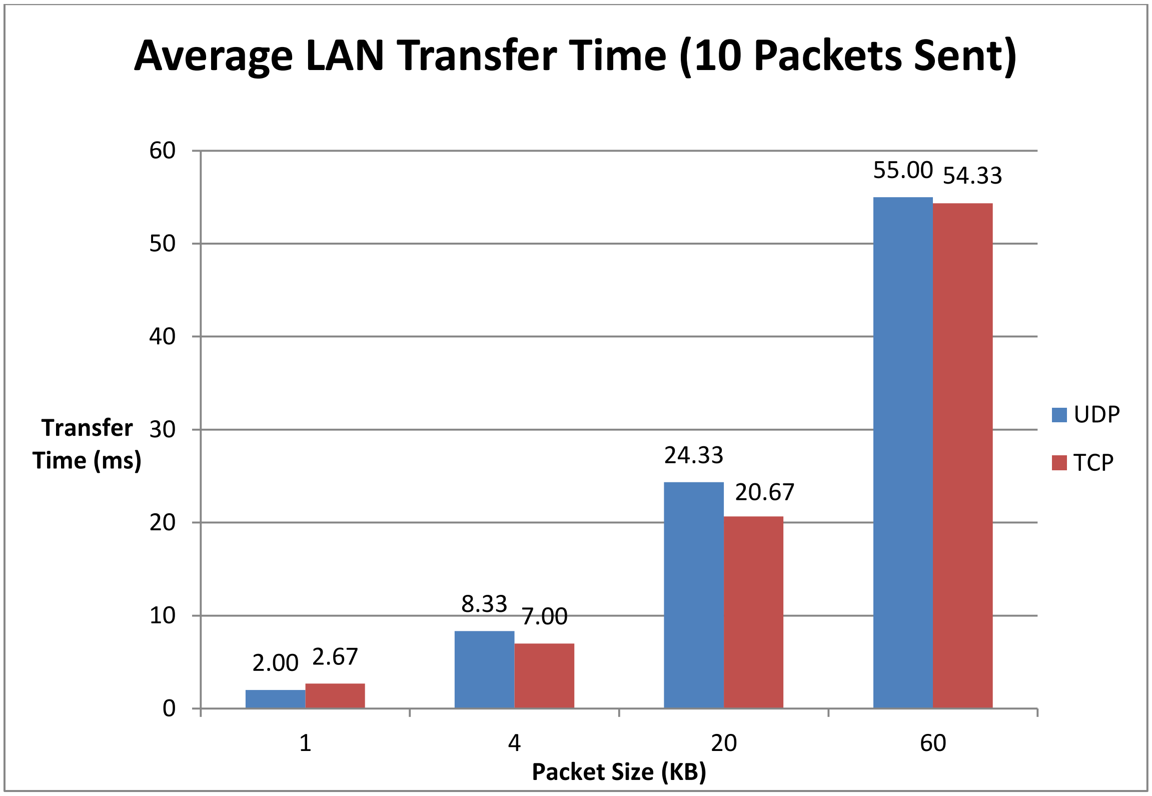


Figure 1: Average LAN transfer time for 10 packet bursts over wired LAN

Figure 1 compares UDP and TCP's average transfer times where each packet type was sent 10 times. The times grow steadily as the packet size increases, ranging from 2ms for the 1KB packets to 55ms for the 60KB packets. Both protocols demonstrate similar transfer speeds, with TCP being marginally faster for packets larger than 1KB.

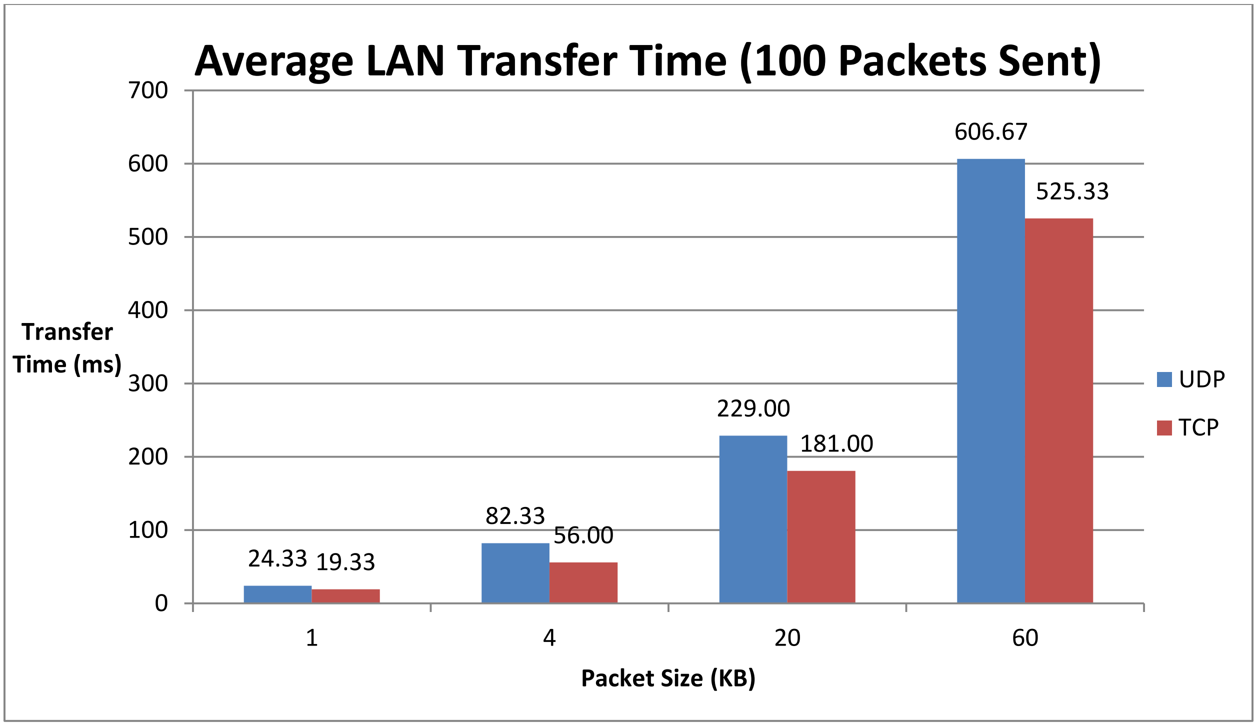


Figure 2: Average transfer time for 100 packet bursts over wired LAN

Figure 2 compares the two protocols sending 100 packet bursts. These results show UDP to be consistently and significantly slower than TCP on the wired LAN. The sending time for UDP also grows at a faster rate than that of TCP in this case.

The above results prove that under the correct circumstances, TCP can be faster than UDP. This contradicts the prediction that UDP would be faster in all circumstances. Some possible causes of this unexpected result are higher network usage at the time the UDP packets were sent and higher processing overhead in the UDP processing (the number of packets being read and the packet size is embedded into each test packet and is read at the receiving end).

Wireless Local Area Network

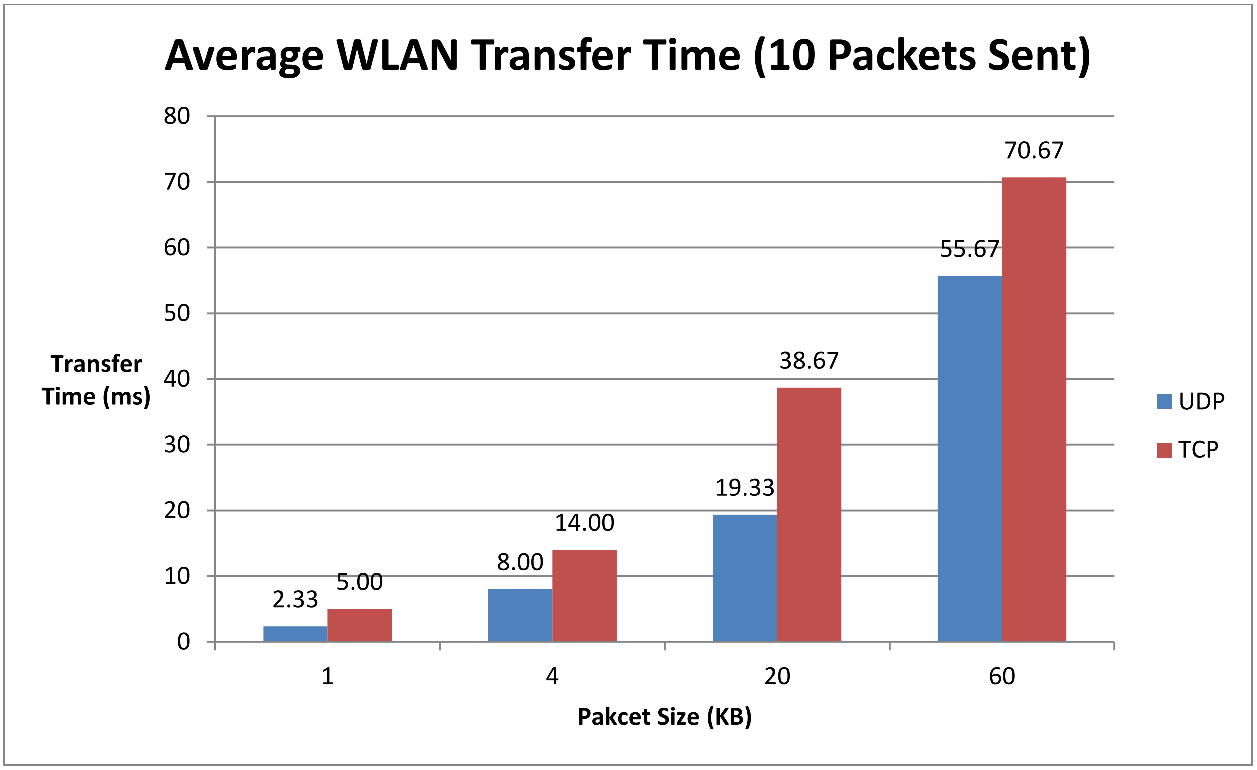


Figure 3: Average transfer time for 10 packet bursts over wireless LAN

Figure 3 above graphs the average transfer time for sending 10 packets on the wireless LAN. The UDP transfers in approximately the same period of time as in the previous example. However, TCP's transfer time has increased significantly. In sending 4KB and 20KB packets, the transfer time has nearly doubled, and the transfer time has increased by 16ms for the 60KB packets.

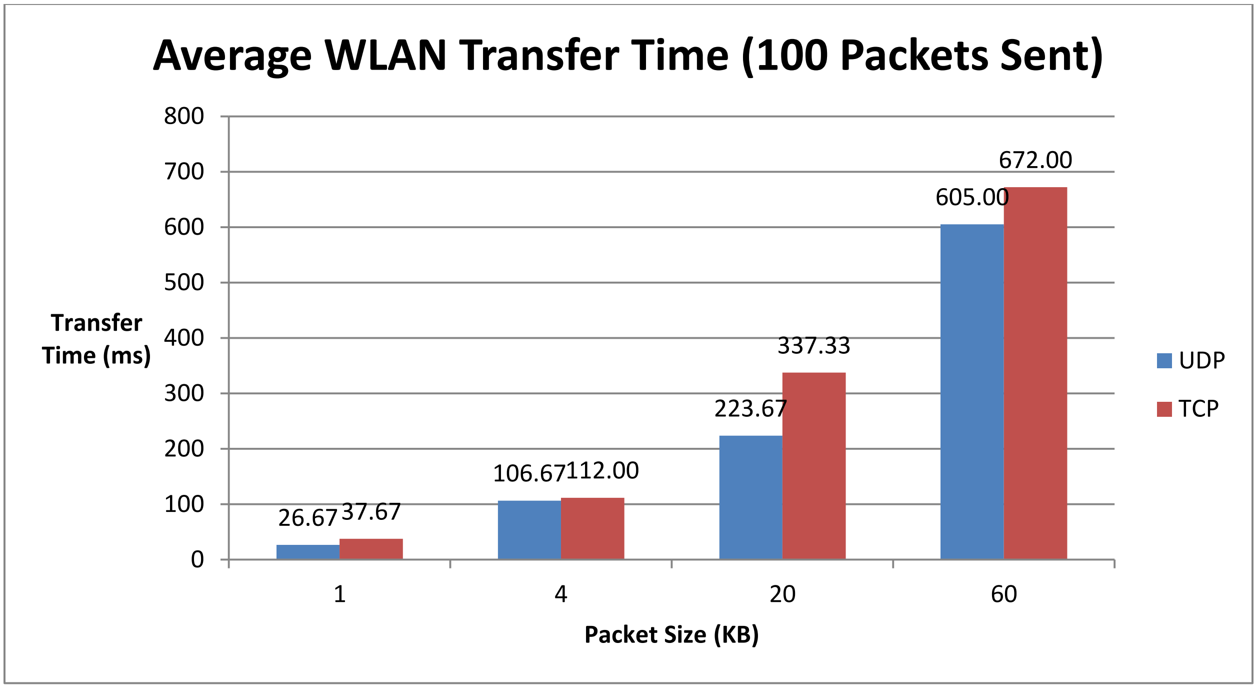


Figure 4: Average transfer time for 100 packet bursts over wireless LAN

The 100 packet bursts show a similar trend. UDP packets transfer in approximately the same time as in the wired LAN portion. The TCP packet transfer time has also doubled for 4KB packets and nearly doubled for 20KB packets.

This result is more in line with expectations. The larger header size and especially the ACKs are likely the biggest factors in the TCP slowdown.

Wide Area Network

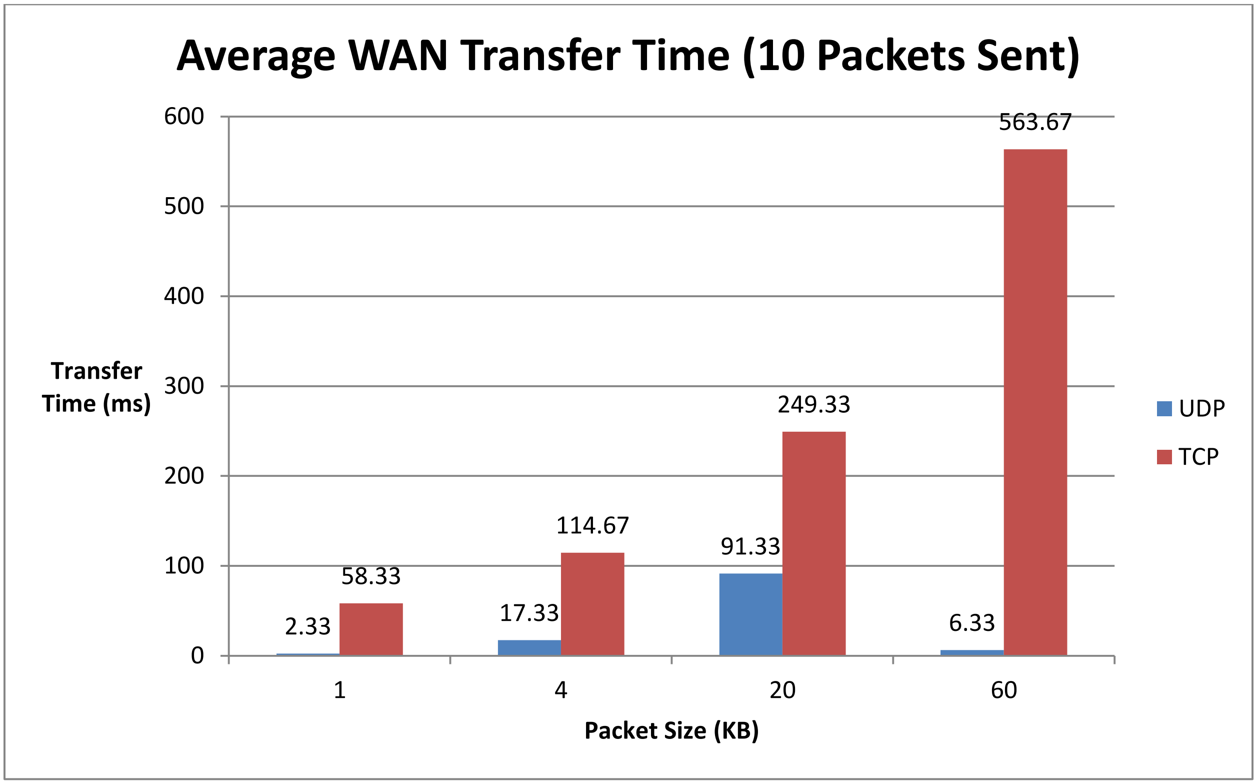


Figure 5: Average transfer time for 10 packet bursts over WAN

Figure 5 shows the transfer time for 10 packets over WAN. The trend seen in the wireless LAN data above has been exacerbated; the time to transmit 1KB 10 times is approximately 29 times longer than over the wired LAN; the 4KB packets, 16 times; the 20KB packets, 12 times; and the 60KB packets, 10 times slower. Conversely, the UDP packets transfer at approximately the same speed as the wired LAN for the 1KB packets, and 2 times as slow for the 4KB packets. The 20KB packets and the 60KB packets appear to transfer quickly, but there is significant packet loss in these bursts (discussed further in the UDP Packet Loss subsection).

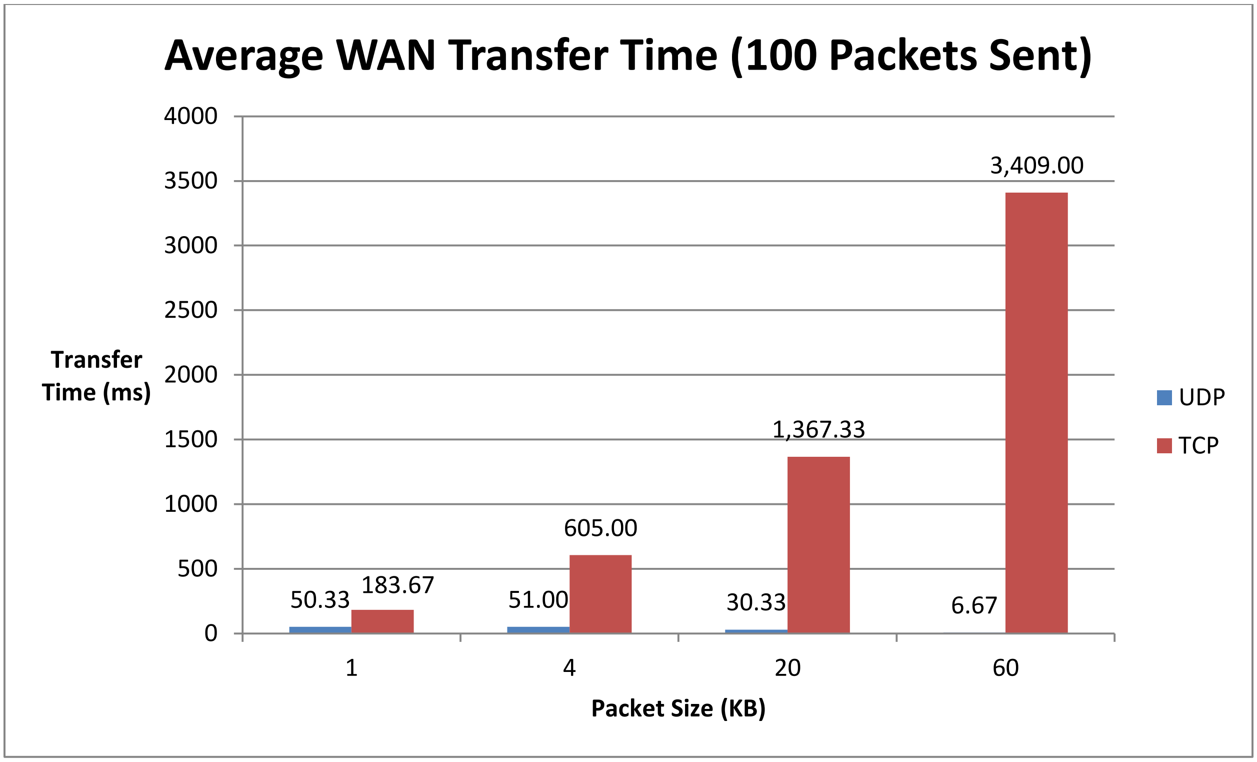
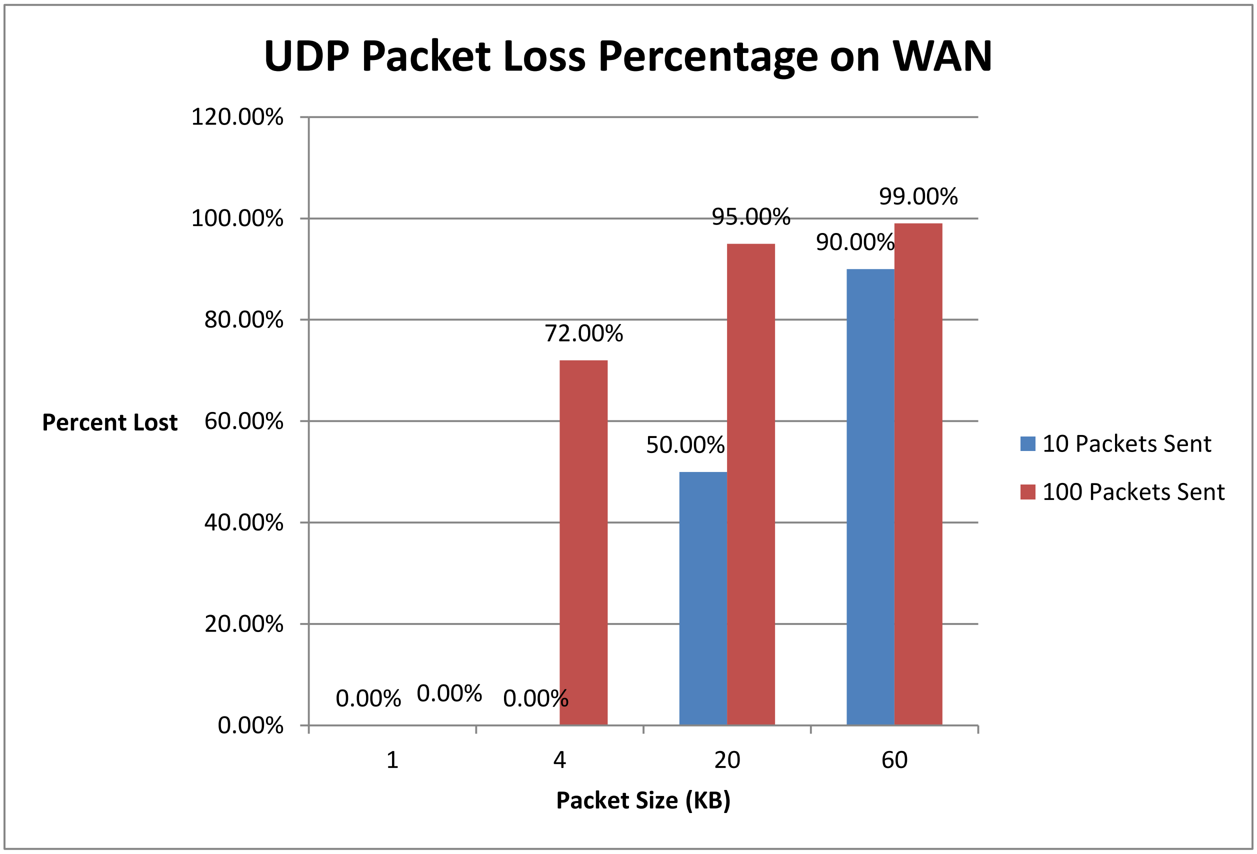


Figure 6: Average transfer time for 100 packet bursts over WAN

The 100 packet bursts over WAN demonstrate the extremes of the each protocol. TCP takes a full 3.4 seconds to transfer 100 60KB packets, which is approximately 62 times slower than on the wired LAN. UDP sends its 100 1KB packets in an average of 50.33ms, which is approximately 25 times slower than sending over the wired LAN. TCP sends the same packets in 183.67ms, which is over 90 times slower than on the wired LAN, meaning that TCP slowed down approximately 3.6 times more than UDP.

The increased traffic on the network, the greater distance to travel, and the extra processing time due to passing through more networking infrastructure all contribute to the extra transfer time. The necessity of ACKing packets increases the TCP transfer time dramatically, while the lack of reliability features in UDP do not incur the extra round-trip times and processing time, thus keeping its transfer speed relatively quick.

UDP Packet Loss



7: UDP packet loss percentage over WAN

While UDP sends much more quickly over the WAN, it also loses a considerable number of packets. The 1KB packets are the only ones not subject to packet loss, while the burst of 100 KB packets and all other packets lose at least 50%. I believe that the main contributor in this loss is the speed at which UDP packets are sent; the router buffer likely can't handle the massive influx of packets, and simply rejects packets. This is supported by the fact that small bursts of data, such as the 1KB packets and the 10 packet burst of 4KB packets, are able to transfer successfully, whereas larger packets are dropped at least half the time.

Conclusion

The tests confirmed that UDP is faster and unreliable (particularly over WAN) and that TCP is slower and reliable as originally speculated. They also revealed that TCP can be faster in certain situations, such as when sent over a wired LAN with little traffic.

TCP would then be suitable for sending small amounts of data on a wired LAN or any amount of data reliably over the WAN/wireless LAN. Applications such as file transfer, which require the reliability of TCP and can afford the large delay over wireless networks and wide area networks, and chat applications, which send relatively small amounts of data and require delivery.

UDP is suited towards applications which send data at slower intervals or smaller packets (as indicated by the success of 1KB packets and 10 4KB packets), such as media streaming applications, games, and GPS updates.

Appendix A: Transfer Data

	TRANSFER TIMES (ms)								
	LAN			WLAN			WAN		
Packets Sent	Test 1	Test 2	Test 3	Test 1	Test 2	Test 3	Test 1	Test 2	Test 3
1KB (UDP)									
10	2	2	2	2	2	3	2	3	2
100	25	27	21	27	22	31	55	48	48
4KB (UDP)									
10	9	8	8	9	7	8	19	18	15
100	93	99	55	103	97	120	49*	51*	53*
20KB (UDP)									
10	21	22	30	21	16	21	34*	179*	61*
100	219	245	223	235	206	230	32*	29*	30*
60KB (UDP)									
10	55	55	55	57	55	55	6*	6*	7*
100	600	621	599	611	604	600	7*	7*	6*
1KB (TCP)									
10	3	3	2	5	5	5	55	64	56
100	19	19	20	35	39	39	190	182	179
4KB (TCP)									
10	6	7	8	16	13	13	115	106	123
100	59	55	54	113	112	111	366	690	759
20KB (TCP)									
10	19	19	24	37	38	41	239	232	277
100	184	175	184	340	355	317	1644	1240	1218
60KB (TCP)									
10	55	54	54	73	71	68	646	531	514
100	522	524	530	716	644	656	3221	3206	3800

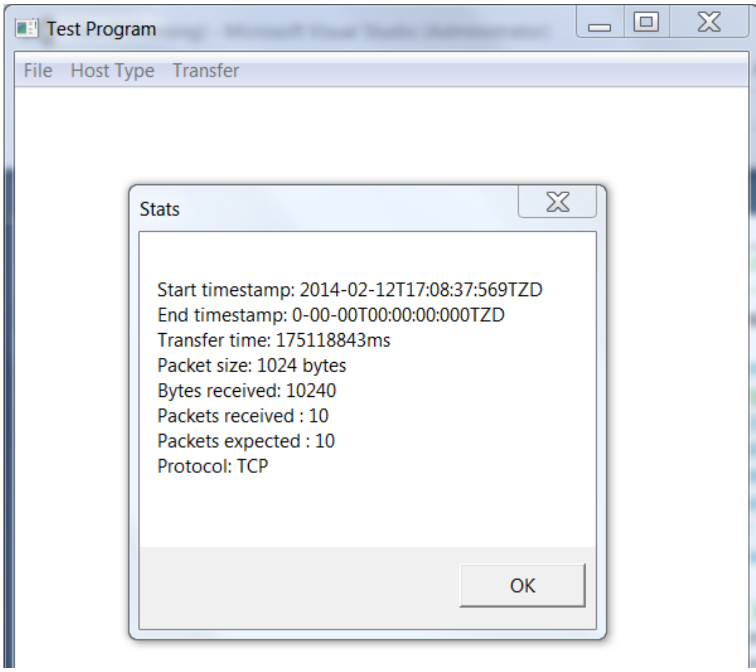
*There was some packet loss for these transfers. The times constitute the time from sending the first packet to receiving the last packet, NOT the total transfer time. See the table below for packet loss statistics.

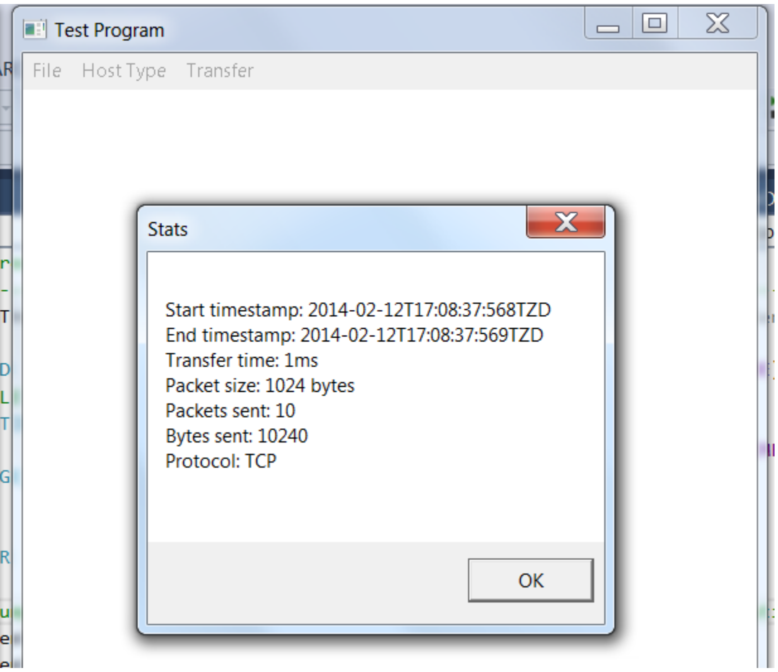
	UDP PACKETS LOST					
	WLAN			WAN		
Packets Sent	Test 1	Test 2	Test 3	Test 1	Test 2	Test 3
1KB						
10	0	0	0	0	0	0
100	0	0	0	0	0	0
4KB						
10	0	0	0	0	0	0
100	0	0	1	72	72	72
20KB						
10	0	0	0	5	5	5
100	0	0	0	95	95	95
60KB						
10	0	0	0	9	9	9
100	6	6	0	99	99	99

Appendix B: Tests

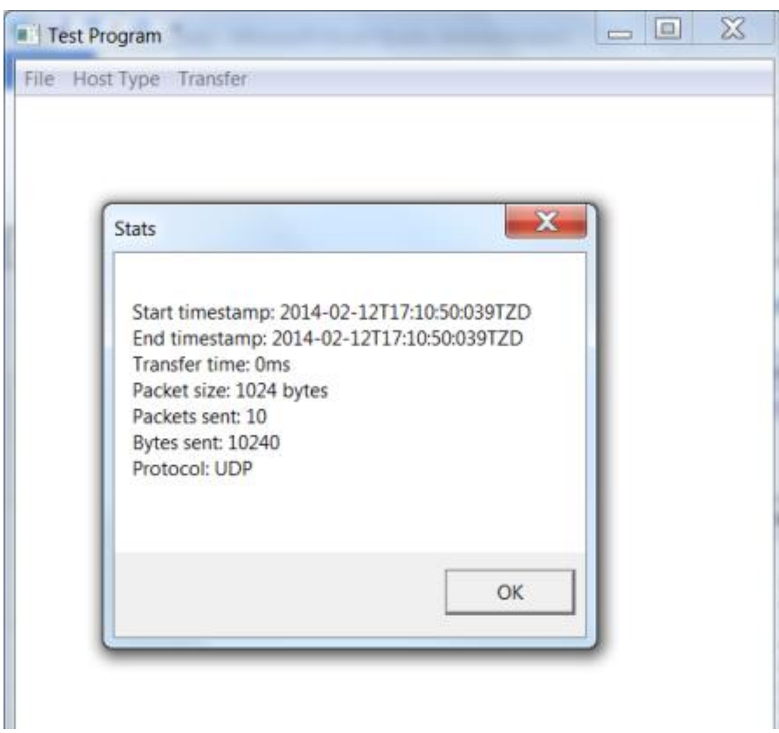
Test Number	Description	Tools Used	Expected Outcome	Result
1	Send generated data over TCP	Assn2	Sender and receiver display transfer statistics	Passed; see Test 1 image below
2	Send generated data over UDP	Assn2	Sender and receiver display transfer statistics	Passed; see Test 2 image below
3	Send file over TCP	Assn2	File received is the same as file sent	Passed; see Test 3 image below
4	Send file over UDP	Assn2	File received is the same as file sent	Passed; see Test 4 image below
5	Generate log file	Assn2	Records all relevant data	Passed; see Test 5 image below

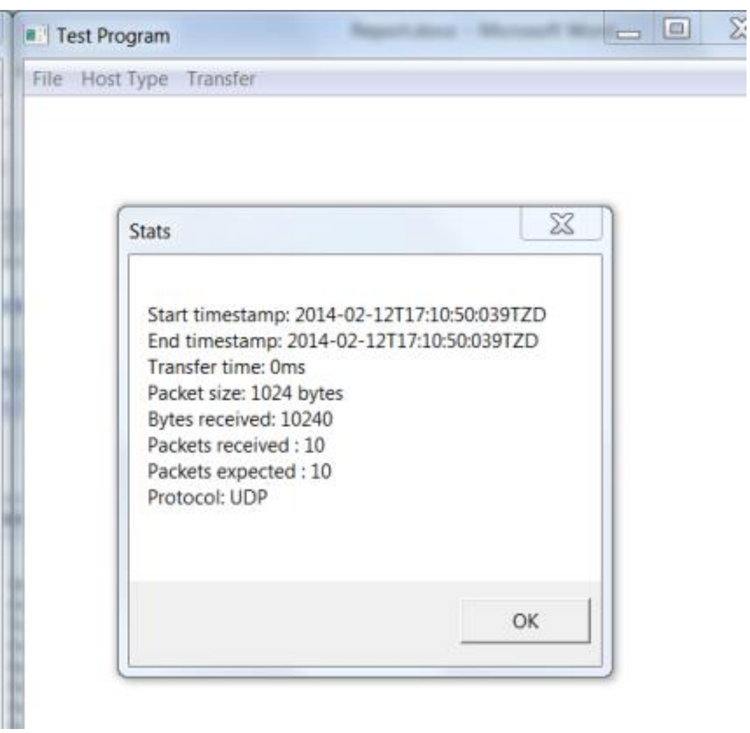
Test 1: Send Generated Data Over TCP



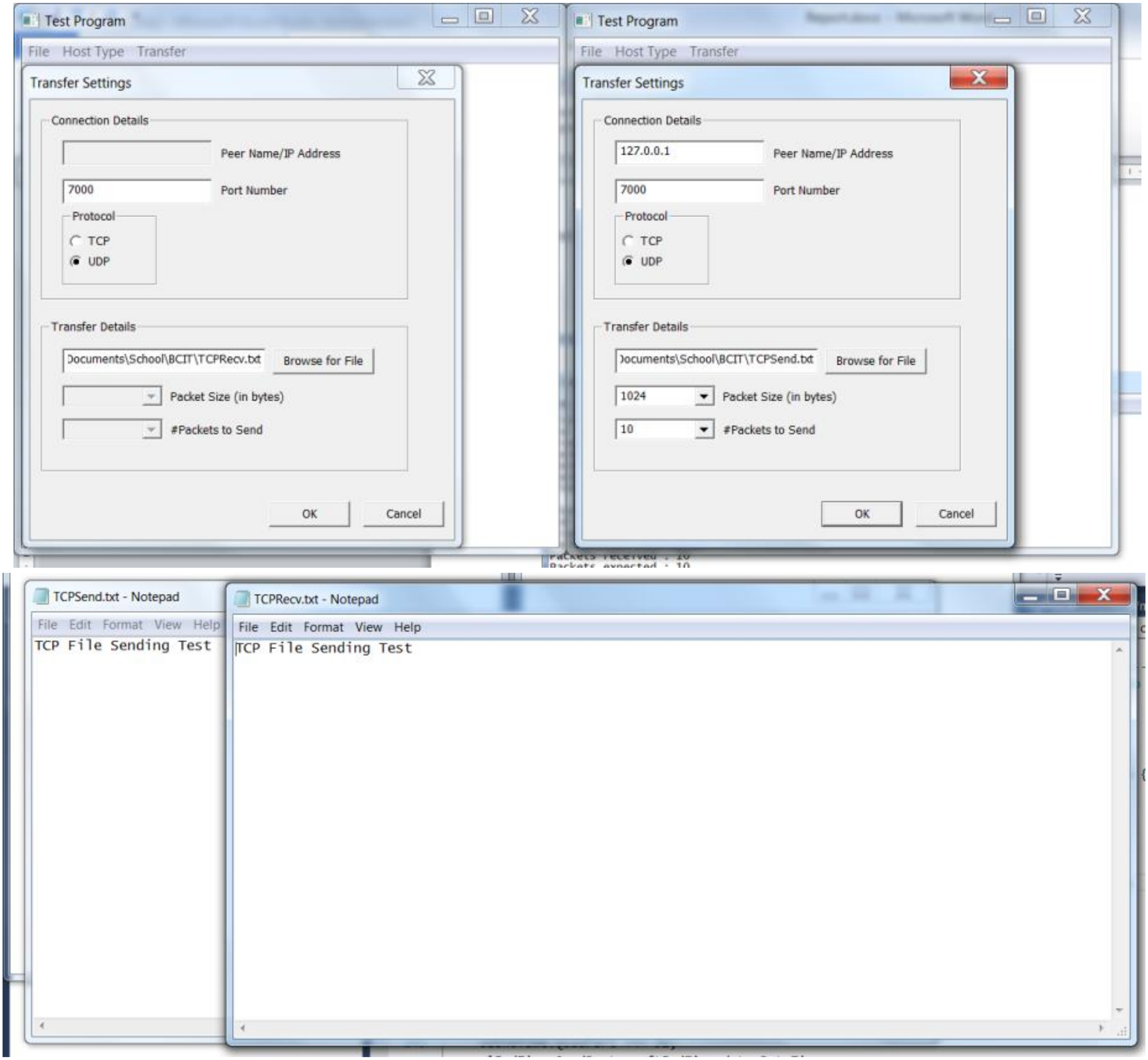


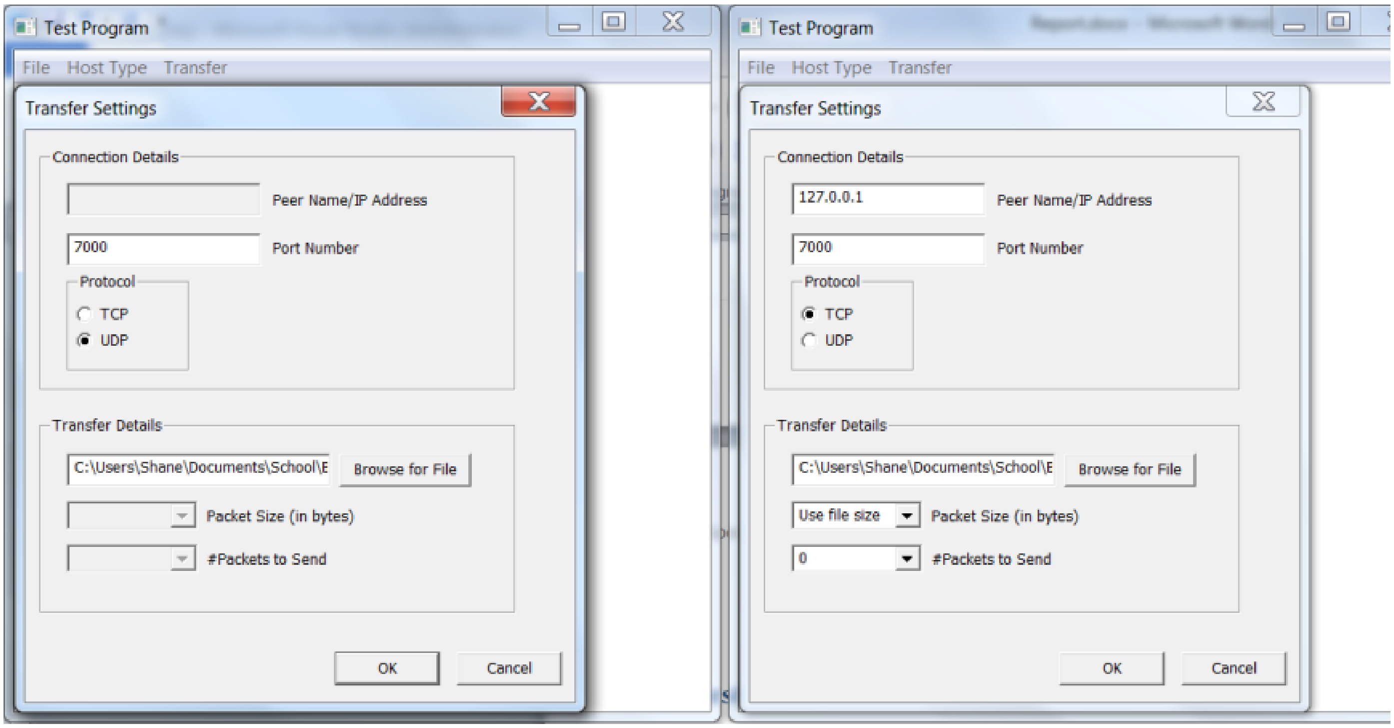
Test 2: Send Generated Data Over UDP

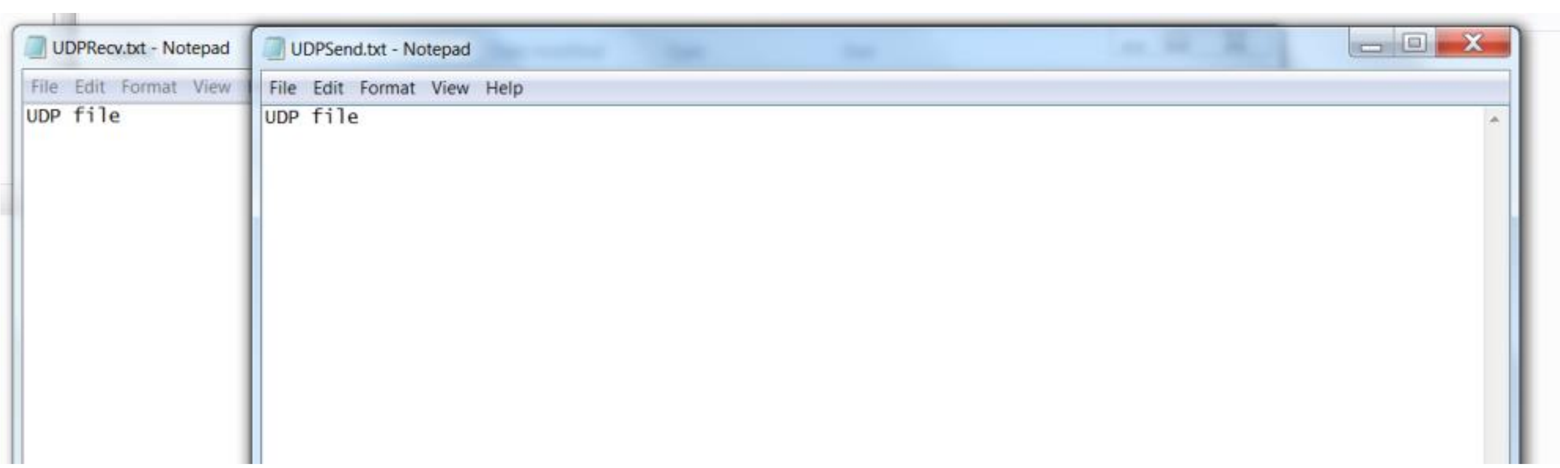




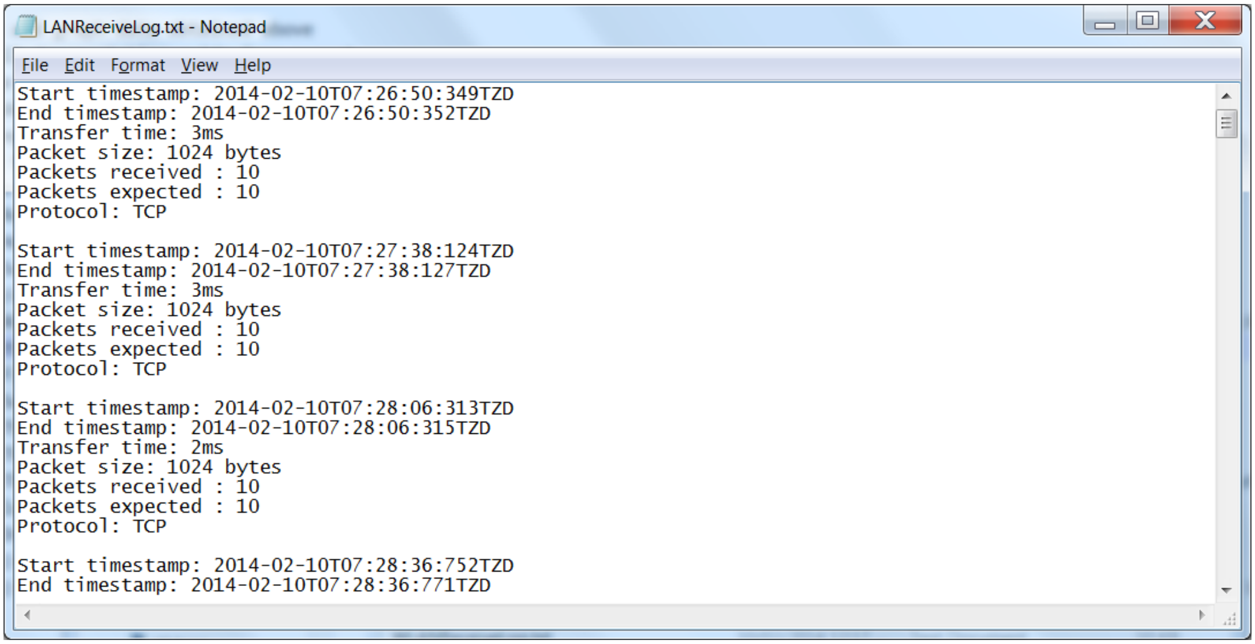
Test 3: Send File Over TCP




Test Program
File Host Type Transfer
Transfer Settings
Connection Details
Peer Name/IP Address
7000
Port Number
Protocol
TCP
UDP
Transfer Details
C:\Users\Shane\Documents\School\E
Browse for File
Packet Size (in bytes)
#Packets to Send
OK
Cancel
Test Program
File Host Type Transfer
Transfer Settings
Connection Details
127.0.0.1
Peer Name/IP Address
7000
Port Number
Protocol
TCP
UDP
Transfer Details
C:\Users\Shane\Documents\School\E
Browse for File
Use file size
Packet Size (in bytes)
0
#Packets to Send
OK
Cancel


UDPRecv.txt - Notepad
File Edit Format View
UDP file
UDPSend.txt - Notepad
File Edit Format View Help
UDP file

Test 5: Generate Log File

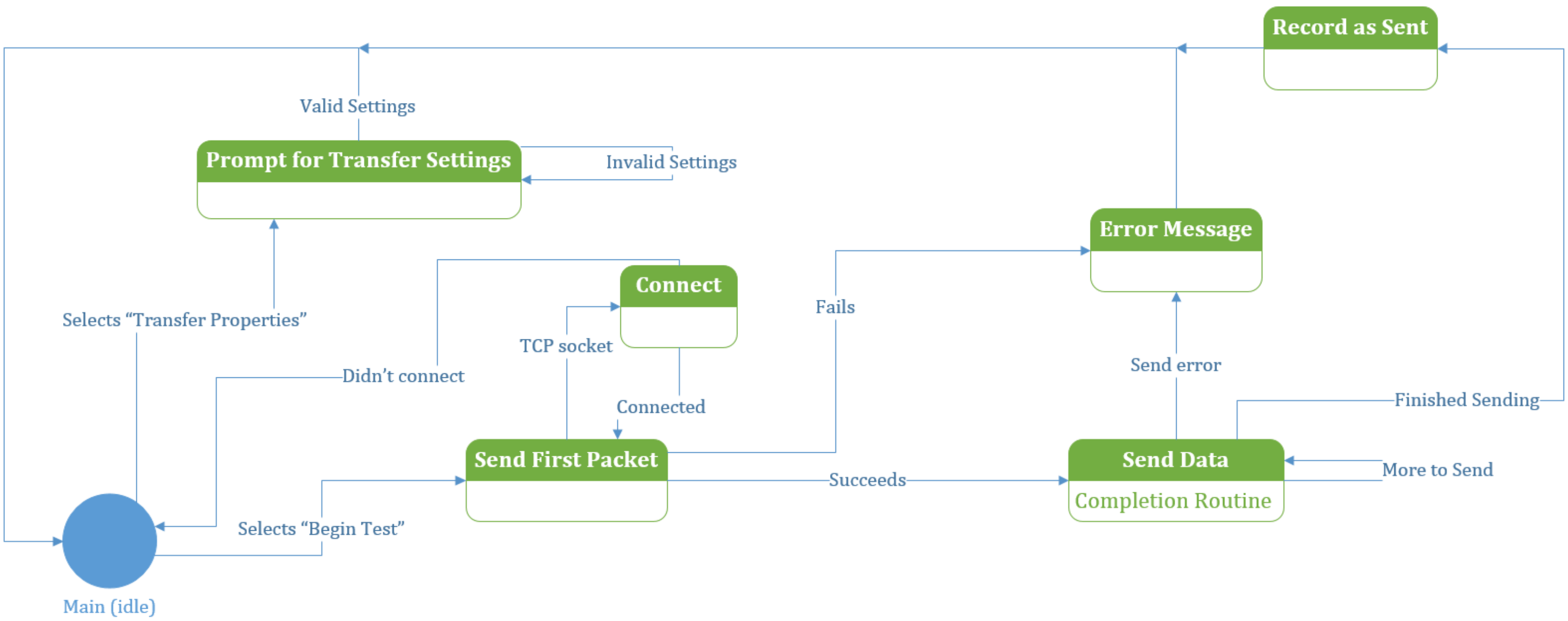



```
Start timestamp: 2014-02-10T07:26:50:349TZD
End timestamp: 2014-02-10T07:26:50:352TZD
Transfer time: 3ms
Packet size: 1024 bytes
Packets received : 10
Packets expected : 10
Protocol: TCP

Start timestamp: 2014-02-10T07:27:38:124TZD
End timestamp: 2014-02-10T07:27:38:127TZD
Transfer time: 3ms
Packet size: 1024 bytes
Packets received : 10
Packets expected : 10
Protocol: TCP

Start timestamp: 2014-02-10T07:28:06:313TZD
End timestamp: 2014-02-10T07:28:06:315TZD
Transfer time: 2ms
Packet size: 1024 bytes
Packets received : 10
Packets expected : 10
Protocol: TCP

Start timestamp: 2014-02-10T07:28:36:752TZD
End timestamp: 2014-02-10T07:28:36:771TZD
```

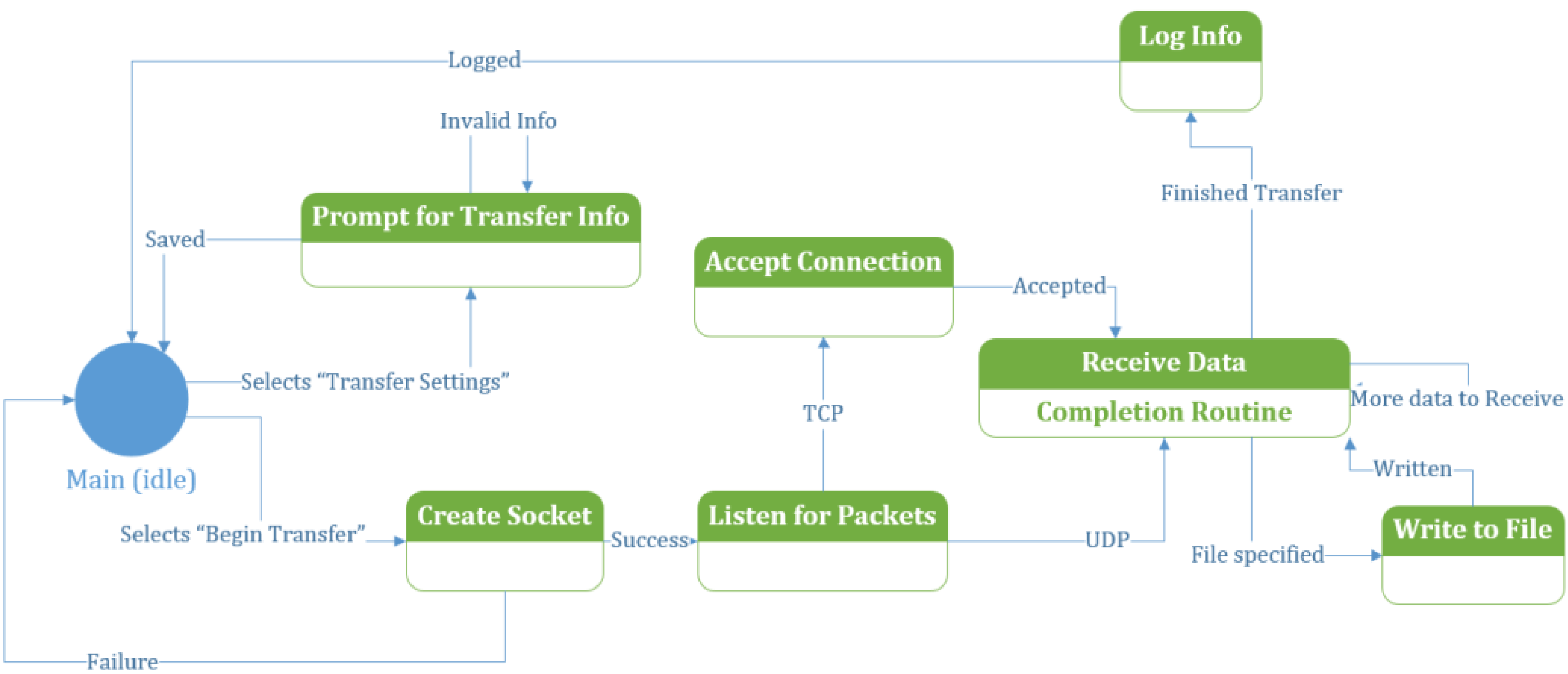
Appendix C: State Chart Diagrams

The state diagrams for the program are featured below. The user switches between Client and Server Mode with a menu item.

Client Side



Server Side



Appendix D: Pseudocode

Client Side

Client Initialise Socket Pseudocode

```
If they used a host name
      Attempt to resolve name/IP
      If resolution unsuccessful
            Display error message
            Return False
      Else
            Store IP address

If the IP address is set
      Create a socket
      If the socket isn't valid
            Display error message
            Return False

      Else
            Store socket descriptor
            Return True

Else
      Display "Please set a host name or IP address."
      Return False
```

Client Send Pseudocode

```
Populate the send buffer
If socket type is SOCK_STREAM
      Send First TCP packet
      If this fails
            Clean up
            Exit thread

Else If socket type is SOCK_DGRAM
```

```
        Send First UDP packet
        If this fails
              Clean up
              Exit thread

  Sleep in alertable wait state
  If the timeout expires
        Display "Connection timed out"
        Exit thread

  Write the transfer details to a log file
  Display message box with transfer details
  Return
```

TCP Send Completion Routine

```
If the error number isn't 0
      Print an error message
      Invalidate the socket
      Return

Add to number of bytes sent
If number of bytes sent == number to send
      Return
Else
      Post another send to the socket
```

UDP Send Completion Routine

```
If error number isn't 0
      Print error message
      Invalidate the socket
      Return

Increment number of bytes sent
If sent == the number to send
      Set last timestamp
      Return
Else
      Post another send to the socket
```

Send First UDP Packet

```
Attempt to WSASendTo the packet
If there was an error
      Display error message
      Return failure
```

Send First TCP Packet

```
Attempt to connect
If unable to connect
      Display "Connection to [host or IP] failed. Please check connection
settings and try again."
      Return

Attempt to WSASend the packet
If there was an error
      Display error message
      Return failure
```

Populate Send Buffer Pseudocode

```
If they specified a file name
      Load File
      If that failed
            Return failure
      Else
            Return success

Else
      Create Buffer
      If that failed
            Return failure
      Else
            Return success
```

Load File Pseudocode

```
If they specified a file name
```

```
	Attempt to open the file

	If the file can't be opened
		Display "Could not open file [file]. Make sure the file is
		spelled correctly or select another."
		Return
	Else
		Read the file into a buffer
```

Create Buffer Pseudocode

```
Create a buffer of user-specified packet size
If the buffer couldn't be created
	Return failure

Load the buffer with random data
Write the number of packets to send and the packet size into the start of the
buffer
```

Server Side

Server Initialise Socket Code

```
Create new listening socket (with correct protocol)
Bind the socket to the stack
```

Server Transfer Code

```
If socket type is SOCK_STREAM
	Listen for TCP connections
	If this fails
		Clean up
		Exit the thread

Else
	Listen for UDP packet
	If this fails
		Clean up
		Exit the thread
```

```
Sleep in alertable wait state
If the wait times out
      Break

Log transfer info
Clean up
Exit thread
```

TCP Receive Completion Routine

```
If the error number isn't 0
      Print an error message
      Set the timeout to 0
      Return
If number of bytes is 0
      Store last timestamp
      Set the timeout to 0

If user specified a file
      Write bytes to file

Add to number of bytes received
If the packet size hasn't been set
      Read the expected number of packets
      Read the packet size
```

UDP Receive Completion Routine

```
If the error number isn't 0
      Print an error message
      Invalidate the socket
      Return

Read int out of packet (number of packets to be received)
Store the packet size

Increment bytes received counter
If packets received == total packets
      Set the timeout to 0
      Return
```

```
If timeout is INFINITE  // this if the first packet
      Set start time to timestamp
      Set timeout to COMM_TIMEOUT
```

Log Transfer Info Pseudocode

```
Attempt to open the file for writing
If the file didn't open
      Display error message
      Return

If the socket is SOCK_STREAM
      Calculate the number of packets received

Record the start time, end time, and total transfer time
If it's the server
      Record the number of packets received and the number of packets
expected
Else
      Record the number of packets sent
Record the packet size and protocol
Close the file
```

Listen for TCP

```
Listen on socket until we get a connection request
WSAAccept

If user specified a file
      Write bytes to file

Create new socket to serve request
Store timestamp
Post WSARecv on socket
```

Listen for UDP

```
Receive first packet
Store timestamp
Post WSASendTo on socket
```