Christopher LaJon Morgan
Brother Zappala CS460
2/28/2013

Lab3: TCP Reliability

Introduction

Overview of the Socket Layer

The basic Socket Layer that I have built is linked into my network simulator via the node class. Each node in my network simulator has both a unique IP as well as an OS. The OS class has three functions, its first function is to issue and keep track of sockets, its second function is to keep track of a set of address/port bindings, and its third function is to create new sockets for incoming connections.

Each Socket has the power to request from the OS a unique binding. For example, a socket could ask the OS to associate with itself the binding (125.225.134.2, 8080). The OS would then reserve the binding and associate the requesting socket with the requested binding. For TCP socket this functionality is extended to bind a tuple consisting of (source address and port, destination address and port) with a connected socket.

Once a socket is created and bound it can then make requests of the virtual network to both send and receive data. For TCP connections, special features have been made to provide reliability across an unreliable network. TCP connection handshakes, cumulative ACKs, and connection teardown were all implemented.

To test the accuracy of TCP reliability my link class was augmented to probabilistically drop packets.

def bind()

Bind in my Socket API is implemented in the following way.

1) Bind ensures that the requested address and port are not already bound.
2) Bind then requests the OS to reserve the requested address and port for use.
3) Bind then assonates the reserved address and port with itself.

This is accomplished by query and adding an entire to the OS's python dictionary, where the address and port are used as the key, and the value is the socket performing the bind request.

def connect()

Connect in my Socket API is implemented in the following way.

1) Checks Connect Attempt: Connect can be call up to three time requesting the same address and port. Once it is called over three times the connection process terminate with a failure to connect.
2) Set the Remote Address and Port: Connect is used to set the remote address and port for the host in which the current client will be communicating.
3) Associate a Current Address and Port: If the current socket is asked to connect with a remote socket but it has not yet been bound to its own machine, connect will request from the OS a random port on which to communicate. It will then set its own address and port to the address and port requested.
4) Create SYN Packet: Connect will then create and send off a SYN packet to the remote host.

5) Schedule Time OUT: Connect will then set an appropriate time for which it should receive a SYN+ACK, if none is received it will call connect again.

def Send

Send in my Socket API is implemented in the following way.

1) Breaks down the data into segments.
2) Puts each segment into the send buffer.
3) Schedules a send-data-event if the send buffer is currently empty.

def Accept

Accept in my Socket API is implemented in the following way.

1) Checks that the state of the socket is consistent with receiving an accept request.
2) Sets the sockets remote address and port.
3) Sends a SYN+ACK for the request.
4) Set a timer to resend SYN+ACK if needed.

def Ready

Ready in my Socket API is implemented in the following way.

1) Clears any timers.
2) Sets the status of the socket to connected.
3) Notifies any registered applications that the socket is ready.
4) If called with a failed flag, ready logs and error and cleans up any binds.

def Recv

Recv in my Socket API is implemented in the following way.

1) Sends received data to any bound application.

def Close

Close in my Socket API is implemented in the following way.

1) Sets the status of the Socket to closing.
2) Sends FIN to connected host.
3) Sets timeout to hear back from the sent FIN

def Closing

Closing in my Socket API is implemented in the following way.

1) Stops any timers.
2) If the socket is closing it calls done.

def Done

Done in my Socket API is implemented in the following way.

1) Stops any timer.
2) Terminates any connections
3) Cleans up bindings in the os.

Packet Object

My packet object may be described by defining its most important parts. Therefore, a packet has the following.

length	The length or size of the packet in bits.
sqNum	SqNum is a unique identifier for the packet, as to distinguish it across the wire and throughout the system; currently, the sqNum is being generated by creating a hash number from the packet's data.
src	Is the address and port from which the packet is originating.
des	Is the address and port to which the packet wishes to be sent.

data	Is the information contained in the packet.
packetType	Packet type represents the type of packet that can be sent. The following are valid types, SYN = 1 ACK = 2 FIN = 4 DATA = 8
ackNum	The sequence number being ACKed.

It may also be important to note that a packet also keeps track of its delays as it is sent through the virtual network.

Explanation of Timers and Retransmission

A single timer was added to the TcpSocket class. The timer is used to reschedule SYN, FIN, SYN+ACK, FIN+ACK, and DATA packets. A maximum retransmit rate is set for SYN and FIN packets.

For DATA packets, the timer is set for the first initial packet. Upon firing the timer will look for its packet in the send window. If the packet is still in the send window the packet will be resent and the timer will be reset. If the packet is not in the window, the timer will reset itself to fire on the packet with the lowest sequence number in the send window. This process continues till no more packets are in the send window.

Demonstrating Accuracy

Opening a Connection

EST. Connection

```
0 SendingConnect from ('125.225.53.1', 609) to ('125.225.53.2', 80) attempt 1
0 PacketRecieved 0 125.225.53.1
0.001005 PacketRecieved 0 125.225.53.2
0.001005 PacketDone 0 0.001005
0.011005 Accept SYNRecieved from [('125.225.53.1', 609)] to [('125.225.53.2', 80)] SENDING SYN|ACT
0.011005 PacketRecieved 0 125.225.53.2
0.02101 PacketRecieved 0 125.225.53.1
0.02101 PacketDone 0 0.010005
0.03101 incomeSocketEvent 0 ('125.225.53.1', 609, '125.225.53.2', 80)
0.04101 recieved 125.225.53.1 type: SYN|ACK
0.04101 connectConfirmed from ('125.225.53.1', 609) to ('125.225.53.2', 80) sending ACK
0.04101 READY 125.225.53.1
0.04101 Client Send File.
0.04101 PacketRecieved 0 125.225.53.1
0.04101 PacketRecieved 0 125.225.53.1
0.04101 PacketRecieved 1500 125.225.53.1
0.04101 PacketRecieved 3000 125.225.53.1
0.042015 PacketRecieved 0 125.225.53.2
0.042015 PacketDone 0 0.001005
0.052015 incomeSocketEvent 0 ('125.225.53.2', 80, '125.225.53.1', 609)
0.062015 recieved 125.225.53.2 type: ACK
0.062015 ACT_Recieved 125.225.53.2
0.062015 READY 125.225.53.2
```

SYN Lost
SYN+ACK Lost

SYN Lost:

```
0 SendingConnect from ('125.225.53.1', 422) to ('125.225.53.2', 80) attempt 1
0 PacketRecieved 0 125.225.53.1
0.001 LossPacket 0 SYN
3.0 SendingConnect from ('125.225.53.1', 422) to ('125.225.53.2', 80) attempt 2
3.0 PacketRecieved 0 125.225.53.1
3.001005 PacketRecieved 0 125.225.53.2
3.001005 PacketDone 0 0.001005
3.011005 Accept SYNRecieved from [('125.225.53.1', 422)] to [('125.225.53.2', 80)] SENDING SYN|ACT
3.011005 PacketRecieved 0 125.225.53.2
3.02101 PacketRecieved 0 125.225.53.1
3.02101 PacketDone 0 0.010005
3.03101 incomeSocketEvent 0 ('125.225.53.1', 422, '125.225.53.2', 80)
3.04101 recieved 125.225.53.1 type: SYN|ACK
3.04101 connectConfirmed from ('125.225.53.1', 422) to ('125.225.53.2', 80) sending ACK
3.04101 READY 125.225.53.1
```

SYN+ACK Lost:

```
0 SendingConnect from ('125.225.53.1', 661) to ('125.225.53.2', 80) attempt 1
0 PacketRecieved 0 125.225.53.1
0.001 LossPacket 0 SYN
3.0 SendingConnect from ('125.225.53.1', 661) to ('125.225.53.2', 80) attempt 2
3.0 PacketRecieved 0 125.225.53.1
3.001005 PacketRecieved 0 125.225.53.2
3.001005 PacketDone 0 0.001005
3.011005 Accept SYNRecieved from [('125.225.53.1', 661)] to [('125.225.53.2', 80)] SENDING SYN|ACT
3.011005 PacketRecieved 0 125.225.53.2
3.021005 LossPacket 0 SYN|ACK
6.0 SendingConnect from ('125.225.53.1', 661) to ('125.225.53.2', 80) attempt 3
6.0 PacketRecieved 0 125.225.53.1
6.001 LossPacket 0 SYN
6.011005 resendAccept onNode[125.225.53.2] attempt [1]
6.011005 PacketRecieved 0 125.225.53.2
6.02101 PacketRecieved 0 125.225.53.1
6.02101 PacketDone 0 0.020005
6.03101 incomeSocketEvent 0 ('125.225.53.1', 661, '125.225.53.2', 80)
6.04101 recieved 125.225.53.1 type: SYN|ACK
6.04101 connectConfirmed from ('125.225.53.1', 661) to ('125.225.53.2', 80) sending ACK
6.04101 READY 125.225.53.1
6.04101 Client Send File.
6.04101 PacketRecieved 0 125.225.53.1
6.04101 PacketRecieved 0 125.225.53.1
6.04101 PacketRecieved 1500 125.225.53.1
6.04101 PacketRecieved 3000 125.225.53.1
6.04201 LossPacket 0 ACK
7.5420151 PacketRecieved 0 125.225.53.2
7.5420151 PacketDone 0 1.5010051
7.5520151 incomeSocketEvent 0 ('125.225.53.2', 80, '125.225.53.1', 661)
7.5620151 recieved 125.225.53.2 type: DATA|ACK
7.5620151 DumpingData 125.225.53.2 NOT_Connected
7.5620151 ACT_Recieved 125.225.53.2 Est.ing_Connection
7.5620151 READY 125.225.53.2
```

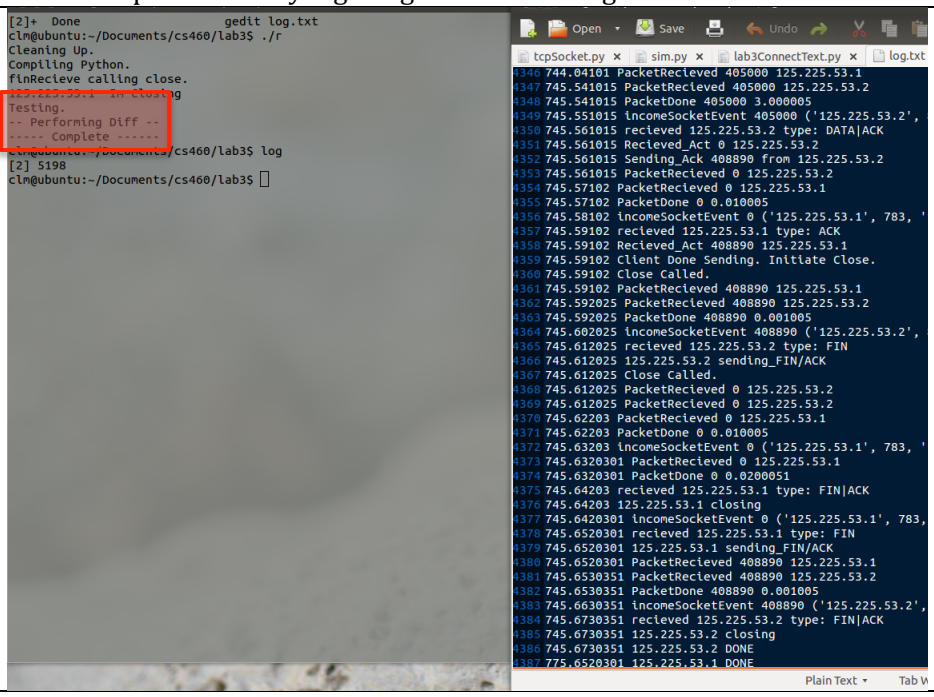
FAILED Connection		0 SendingConnect from ('125.225.53.1', 131) to ('125.225.53.2', 80) attempt 1 0 PacketRecieved 0 125.225.53.1 0.001 LossPacket 0 SYN 3.0 SendingConnect from ('125.225.53.1', 131) to ('125.225.53.2', 80) attempt 2 3.0 PacketRecieved 0 125.225.53.1 3.001 LossPacket 0 SYN 6.0 SendingConnect from ('125.225.53.1', 131) to ('125.225.53.2', 80) attempt 3 6.0 PacketRecieved 0 125.225.53.1 6.001 LossPacket 0 SYN 9.0 NOT_READY 125.225.53.1 9.0 125.225.53.1 DONE

Transfer of a 10,000 Line File

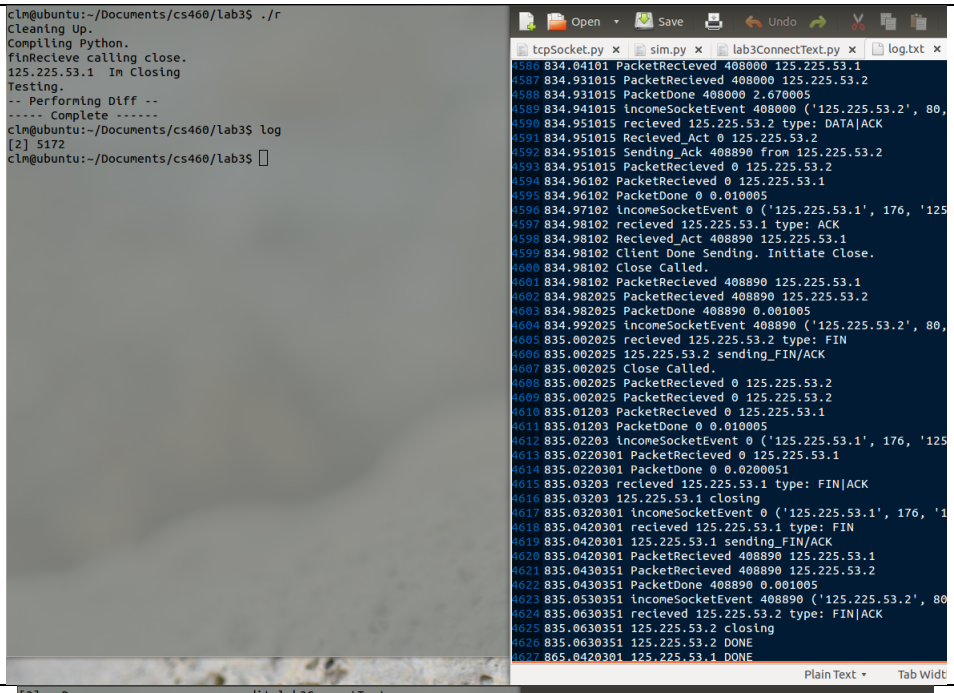
Below I am transferring a file that is comprised of 10,000 lines of text. The file is transmitted across my network and then saved once it is received. I have a script that runs my program and then compares my original file with the transmitted one. The red box signifies the beginning and end of the diff test. Nothing in between the "Performing Diff" and "Complete" tags means the files are the same.

On the right side is a small portion of my log file generated during the network simulation.

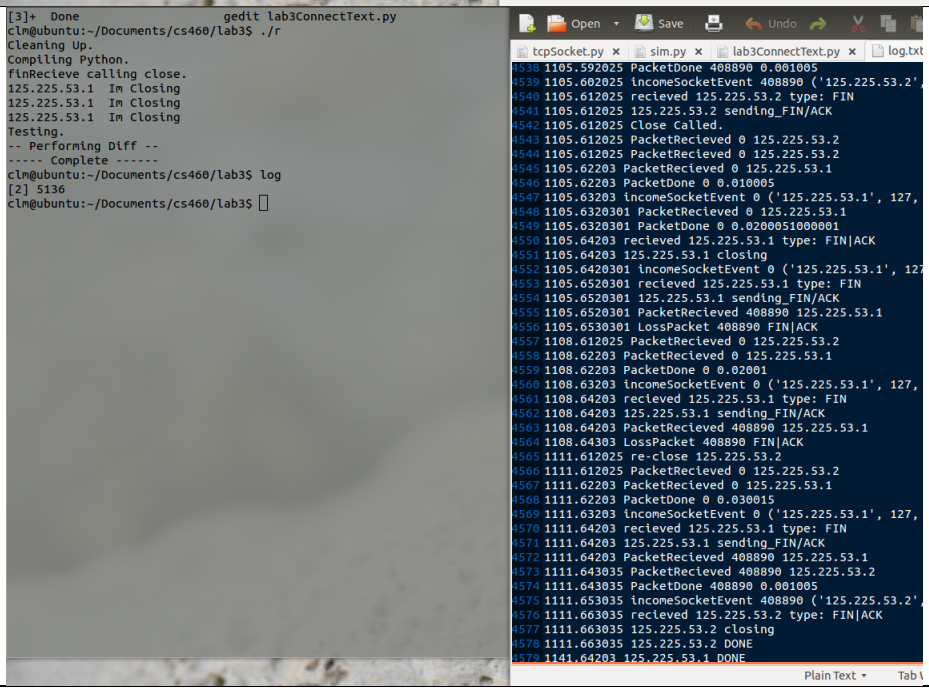
25% Loss Rate



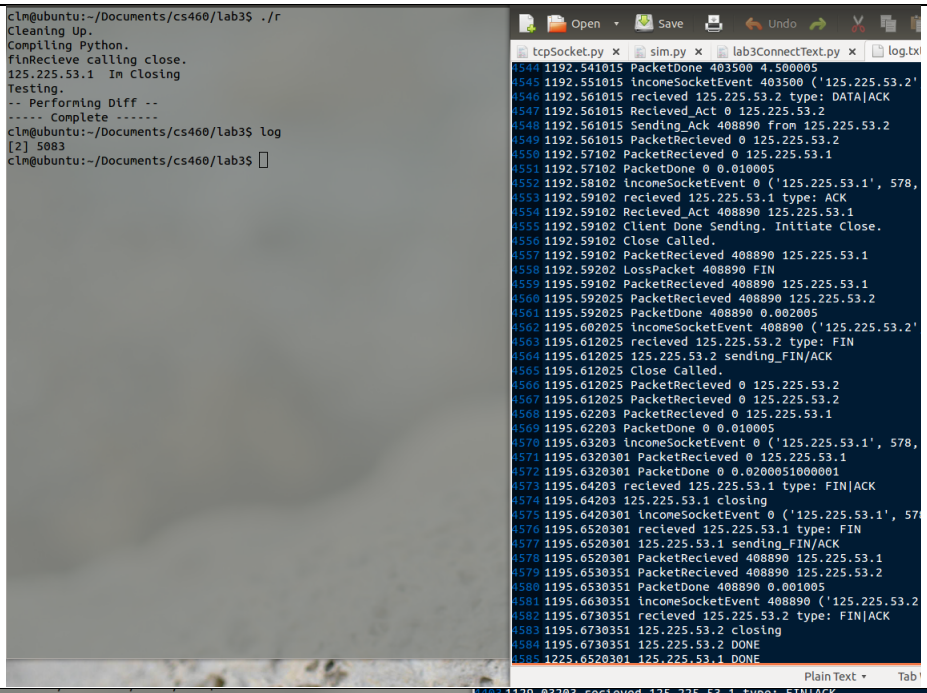
30% Loss Rate



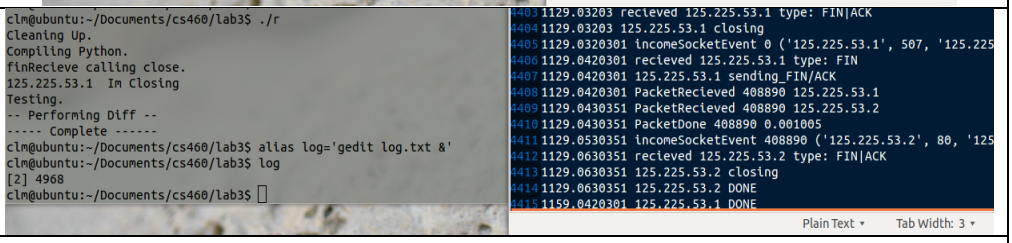
40% Loss Rate



45% Loss Rate



50% Loss Rate



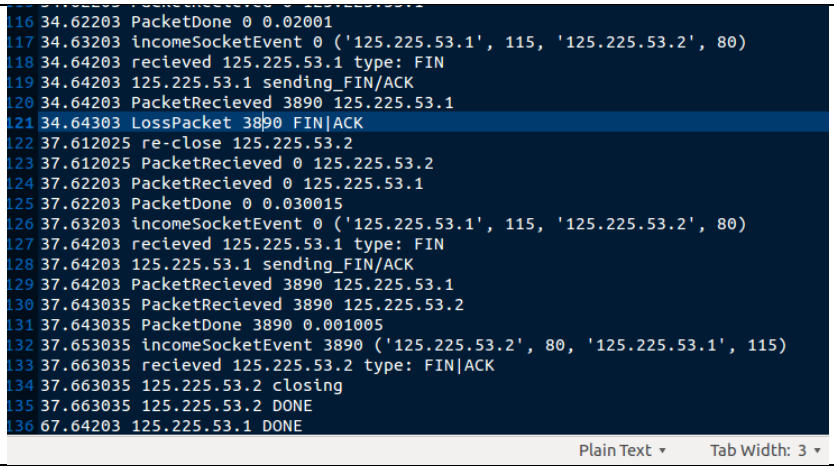
Closing a Connection

FIN Lost
FIN+ACK Lost

Lost FIN:

```
 94 37.59102 Client Done Sending. Initiate Close.
 95 37.59102 Close Called.
 96 37.59102 PacketRecieved 3890 125.225.53.1
 97 37.59202 LossPacket 3890 FIN
 98 40.59102 PacketRecieved 3890 125.225.53.1
 99 40.592025 PacketRecieved 3890 125.225.53.2
100 40.592025 PacketDone 3890 0.002005
101 40.602025 incomeSocketEvent 3890 ('125.225.53.2', 80, '125.225.53.1', 111)
102 40.612025 recieved 125.225.53.2 type: FIN
103 40.612025 125.225.53.2 sending_FIN/ACK
104 40.612025 Close Called.
105 40.612025 PacketRecieved 0 125.225.53.2
106 40.612025 PacketRecieved 0 125.225.53.2
107 40.62203 PacketRecieved 0 125.225.53.1
108 40.62203 PacketDone 0 0.010005
109 40.6320251 LossPacket 0 FIN
110 40.63203 incomeSocketEvent 0 ('125.225.53.1', 111, '125.225.53.2', 80)
111 40.64203 recieved 125.225.53.1 type: FIN|ACK
112 40.64203 125.225.53.1 closing
113 43.612025 PacketRecieved 0 125.225.53.2
114 43.62203 PacketRecieved 0 125.225.53.1
115 43.62203 PacketDone 0 0.020005
116 43.63203 incomeSocketEvent 0 ('125.225.53.1', 111, '125.225.53.2', 80)
117 43.64203 recieved 125.225.53.1 type: FIN
118 43.64203 125.225.53.1 sending_FIN/ACK
119 43.64203 PacketRecieved 3890 125.225.53.1
120 43.64303 LossPacket 3890 FIN|ACK
121 46.612025 re-close 125.225.53.2
122 46.612025 PacketRecieved 0 125.225.53.2
123 46.62203 PacketRecieved 0 125.225.53.1
124 46.62203 PacketDone 0 0.03001
125 46.63203 incomeSocketEvent 0 ('125.225.53.1', 111, '125.225.53.2', 80)
126 46.64203 recieved 125.225.53.1 type: FIN
127 46.64203 125.225.53.1 sending_FIN/ACK
128 46.64203 PacketRecieved 3890 125.225.53.1
129 46.643035 PacketRecieved 3890 125.225.53.2
130 46.643035 PacketDone 3890 0.001005
131 46.653035 incomeSocketEvent 3890 ('125.225.53.2', 80, '125.225.53.1', 111)
132 46.663035 recieved 125.225.53.2 type: FIN|ACK
133 46.663035 125.225.53.2 closing
134 46.663035 125.225.53.2 DONE
135 76.64203 125.225.53.1 DONE
```

Lost FIN+ACT:


116 34.62203 PacketDone 0 0.02001
117 34.63203 incomeSocketEvent 0 ('125.225.53.1', 115, '125.225.53.2', 80)
118 34.64203 recieved 125.225.53.1 type: FIN
119 34.64203 125.225.53.1 sending_FIN/ACK
120 34.64203 PacketRecieved 3890 125.225.53.1
121 34.64303 LossPacket 3890 FIN|ACK
122 37.612025 re-close 125.225.53.2
123 37.612025 PacketRecieved 0 125.225.53.2
124 37.62203 PacketRecieved 0 125.225.53.1
125 37.62203 PacketDone 0 0.030015
126 37.63203 incomeSocketEvent 0 ('125.225.53.1', 115, '125.225.53.2', 80)
127 37.64203 recieved 125.225.53.1 type: FIN
128 37.64203 125.225.53.1 sending_FIN/ACK
129 37.64203 PacketRecieved 3890 125.225.53.1
130 37.643035 PacketRecieved 3890 125.225.53.2
131 37.643035 PacketDone 3890 0.001005
132 37.653035 incomeSocketEvent 3890 ('125.225.53.2', 80, '125.225.53.1', 115)
133 37.663035 recieved 125.225.53.2 type: FIN|ACK
134 37.663035 125.225.53.2 closing
135 37.663035 125.225.53.2 DONE
136 67.64203 125.225.53.1 DONE
Plain Text
Tab Width: 3