

Course Code Assignment #

Student Number

Student Number



# Objective:

A computer code to analyse network packets.

# Method:

Here you will explain the steps you need to take in order to analyse the data. Take help from the posted assignment file.

# Program:

## Block Diagram or Pseudocode

Here you will provide either a block diagram, or a flow chart or pseudocode of your program.

Created 3 dictionaries, which hold the hex value and the associated protocol / port name for each

Created a HexToDecimal(x) function that takes a list as an argument

Define a getFrameInformation(x, y) function that takes a list and frame file as an input

Open the text file that contains the packet information

Read, split and store the contents of the text file in the list that was passed into the function

Use slicing and indices to separate and store the mac addresses and the upper layer protocol

Lookup the hex of the upperlayer protocol in the dictionary, and get the protocol and run the respective function.

getIPv4FrameInformation(x), where x is a list of all the hex byte data after the upper layer protocol (ie, everything after 0800).

Use indices and slicing to get the source and destination IP address, and store them in variables

Use indexing to get the byte that holds the Encapsulated IPv4 protocol, and store it in a variable, look it up in its respective dictionary, and call the function.

Use indexing and slicing to get the relevant information (src and dest port), store in a variable and print

## Code

1. def convertHexToDecimal(HexList):
2. i = 0
3. for HexByte in HexList:
4. newDecimalInt = int(HexByte,16)
5. HexList[i] = newDecimalInt
6. i += 1
7. result = '.'.join(str(i) for i in HexList)
8. return(result)
10. print("\_" \* 48)
11. frame1HexList = []
12. frame2HexList = []
13. frame3HexList = []
14. UpperLayerProtocolDict = {
15. "0800": "IPv4",
16. "87DD": "IPv6",
17. "0806": "ARP"
18. }
19. ProtocolsEncapsulatedByIPv4Dict= {
20. "01": "ICMP",
21. "02": "IGMP",
22. "06": "TCP",
23. "11": "UDP"
24. }
25. PortNumbersEncapsulatedByTcpUdpDict = {
26. "0014": "FTP",
27. "0016": "SSH",
28. "0017": "TELNET",
29. "0019": "Email SMTP",
30. "0035": "DNS",
31. "0050": "HTTP",
32. "008F": "IMAP",
33. "00A1": "SNMP",
34. "00B3": "BGP",
35. "0208": "RIP",
36. "01BB": "HTTPS",
37. "0387": "VMware Remove Console"
38. }
39. mac\_address\_length = 6
40. ipv4\_header\_length = 20
41. def getFrameInformation(frame\_file, hex\_list):
42. with open(frame\_file) as frame:
43. hex\_list.extend(frame.read().split())
44. # *this block below gets src and dest mac address*
45. DestinationMac = '-'.join(hex\_list[:mac\_address\_length])
46. SourceMac = '-'.join(hex\_list[mac\_address\_length : 2 \* mac\_address\_length])
47. print("\n\n\n")
48. print(frame\_file.center(48))
49. print("-" \* 48)
50. print("Ethernet Protocol".center(48))
51. print("Destination MAC: {:>28}".format(DestinationMac.upper())) # *for debug*
52. print("Source MAC: {:>33}".format(SourceMac.upper())) # *for debug*
54. # *this block gets the upper layer protocol*
55. upperLayerProtocolHex = hex\_list[12] + hex\_list[13]
56. protocol = UpperLayerProtocolDict[upperLayerProtocolHex]
57. print("-" \* 48)
58. print("{} Protocol".format(protocol).center(48))
60. # *This block gets the IP header information, gets everything after the 14'th item*
61. # *upperLayerInformation = hex\_list[14 : 14 + ipv4\_header\_length]*
62. upperLayerInformation = hex\_list[14:]
63. #*calls thed IPv4 frame information function*
64. if protocol == "IPv4":
65. getIPv4FrameInformation(upperLayerInformation)
66. elif protocol == "IPv6":
67. getIPv6FrameInformation(upperLayerInformation)
68. else:
69. getARPFrameInformation(upperLayerInformation)
71. def getIPv4FrameInformation(IPv4Frame):
73. SrcIpAddr = convertHexToDecimal(IPv4Frame[12:16])
74. DestIpAddr = convertHexToDecimal(IPv4Frame[16:20])
75. print("Source IPv4 Address: {:>20}".format(SrcIpAddr))
76. print("Destination IPv4 Address: {:>16}".format(DestIpAddr))

79. protocol = ProtocolsEncapsulatedByIPv4Dict[IPv4Frame[9]]
80. print("-" \* 48)
81. print("{} Protocol".format(protocol).center(48))
83. ProtocolEncapsulatedByIPv4ByteList = IPv4Frame[20:]
84. if protocol == "TCP":
85. tcpHeader(ProtocolEncapsulatedByIPv4ByteList)
86. elif protocol == "UDP":
87. udpHeader(ProtocolEncapsulatedByIPv4ByteList)

90. def getIPv6FrameInformation(IPv6Frame):
91. None

94. def getARPFrameInformation(ARPFrame):
95. None

98. def tcpHeader(tcpFrame):
99. SrcPort = ''.join(tcpFrame[:2])
100. DestPort = ''.join(tcpFrame[2:4])
102. print("Source Port: {:>18} ".format(int(SrcPort, 16)))
103. print("Destination Port: {:>15} ".format(int(DestPort, 16)))
105. port = PortNumbersEncapsulatedByTcpUdpDict[SrcPort.upper()]
107. print("-" \* 48)
108. print("\n{} Protocol\n".format(port).center(48))
109. print("-" \* 48)
111. def udpHeader(udpFrame):
112. SrcPort = ''.join(udpFrame[2:4])
113. DestPort = ''.join(udpFrame[:2])
115. print("Source Port: {:>22} ".format(int(DestPort, 16)))
116. print("Destination Port: {:>14} ".format(int(SrcPort, 16)))
118. port = PortNumbersEncapsulatedByTcpUdpDict[SrcPort.upper()]
120. print("-" \* 48)
121. print("\n{} Protocol\n".format(port).center(48))
122. print("-" \* 48)
123. getFrameInformation('frame1.txt', frame1HexList)
124. getFrameInformation('frame2.txt', frame2HexList)
125. getFrameInformation('frame3.txt', frame3HexList)

## Data Structures used

Here you will mention all the data structures that were used. Give references where applicable

The data structures I used are Dictionaries and Lists.

I used dictionaries to store the protocols with their associated hex values to reference when searching for the protocol.

I used lists to store the hex data, each item being 1 byte.

## Functions

Here you will explain the entire flow of program , mentioning and explaining any functions that have been created. Provide citation with the text that is referenced from another source.

I have 4 functions:

getFrameInformation(x, y)

this gets and processes the frame information, and gets the src and dest mac address, along with the upper layer protocol used, then calls getIPv4FrameInformation()

getIPv4FrameInformation(x)

this gets the ipv4 src and destination address, along with the upper upper layer protocol used, then calls either tcpHeader() or udpHeader()

tcpHeader(x) / udpHeader(x)

prints out the src and dest port, along with the protocol

## File Operations

Here you will explain which files were used and why they were used. You will also list all the operations that are performed on the file. You may copy paste relevant lines from your code and explain their functionality. Provide citation for the text that is referenced from another source

The files used in this program are frame1.txt, frame2.txt and frame3.txt. These are the 3 text files with hold the frames, which I then open in the program for processing. I used the read operation to open the file, and then used .extend() and .split() to add them to a list.

# Data Files:

### Name of data file: frame1.txt, frame2.txt, frame3.txt

### Purpose: hold the frame data

### Contents: hexadecimal bytes

# Conclusions

Here you will summarize your work

Took many hours to complete, first drew up a schema of all the steps I needed to take to come to the final product, changed the layout of the program a couple of times, and added in a couple of functions.

# References

Numbered references in IEEE citation format , if any external resources were used to solve this assignment. You should include this section even if the only source you refer to is your textbook.

No external resources used ☺