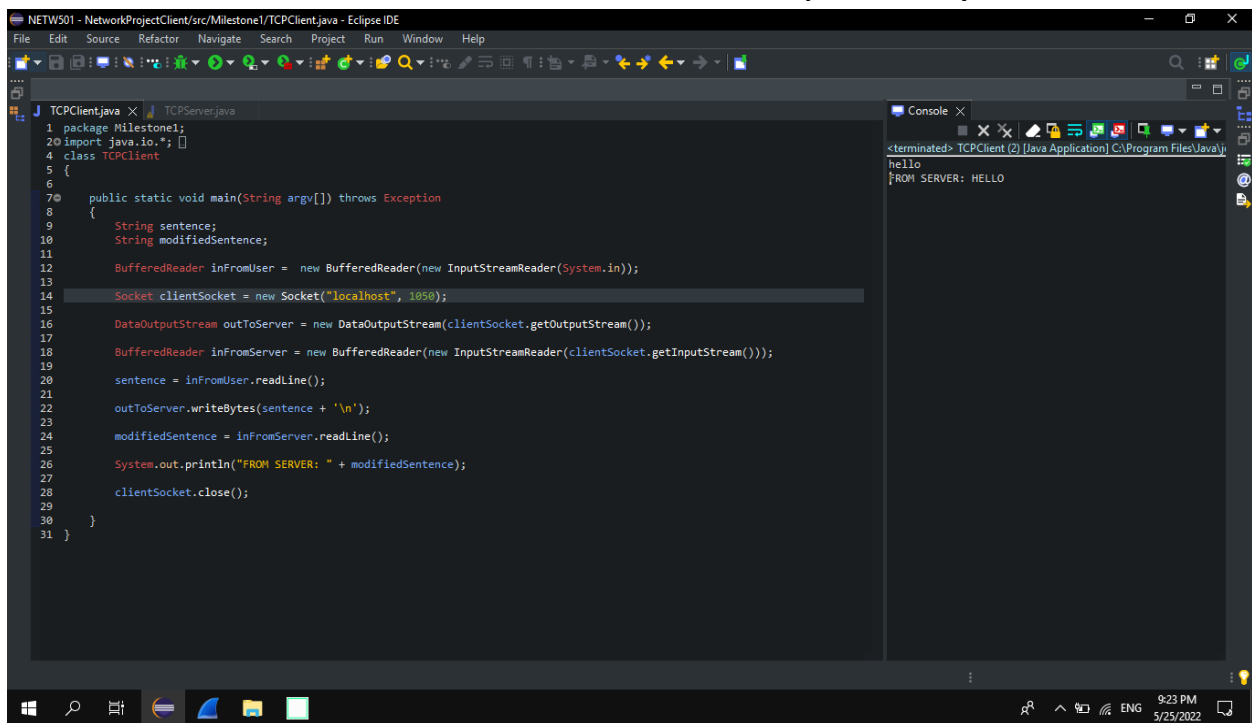


Project report

In this project we aimed to introduce a bi- direction chatting application using the “java socket programming” with imported libraries

MILESTONE1

1. We created a java project and implemented two packages in it
2. The first package concerning our first milestone which goes by the following steps
3. We created two Class (TCPClient , TCPServer) and pasted the code found In the slide but replacing the word “hostname” with “local host” as we are using the same PC.
4. We used a distinctive port number for each run and this is the code for the the client class and the Server respectively

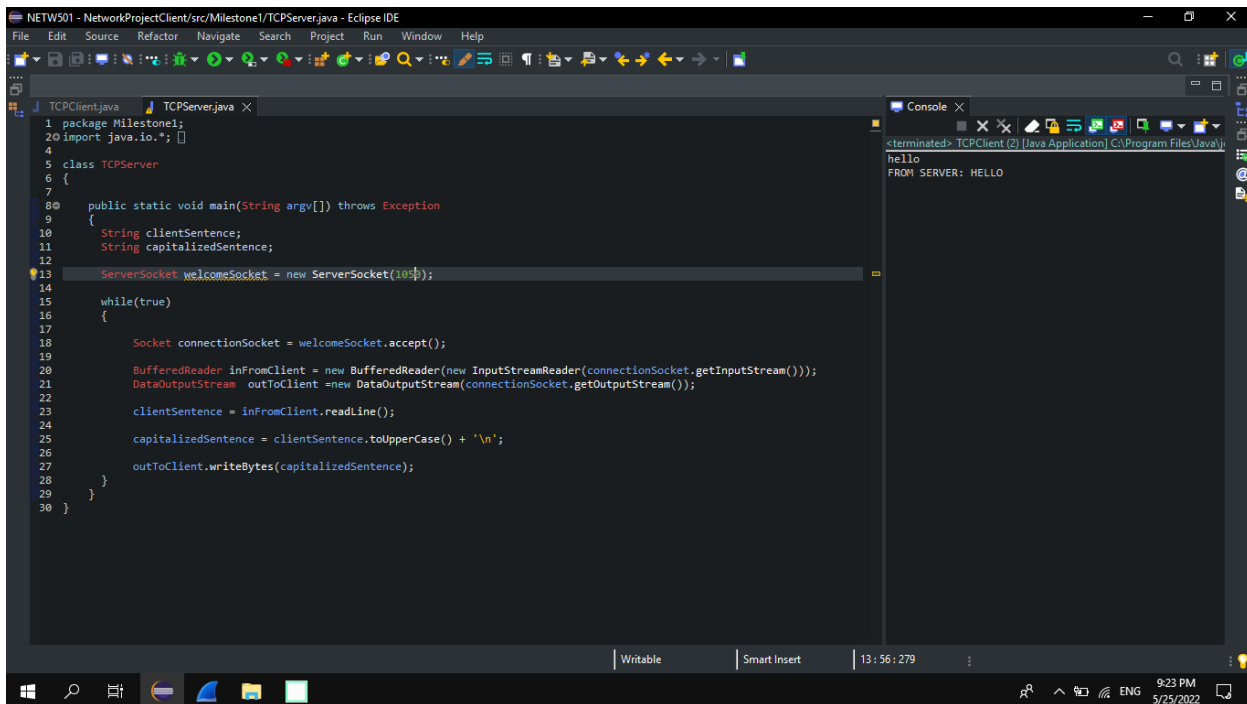


The screenshot shows the Eclipse IDE interface. The main editor displays the code for `TCPClient.java`. The code is as follows:

```
1 package Milestone1;
2 import java.io.*;
3
4 class TCPClient
5 {
6
7
8     public static void main(String argv[]) throws Exception
9     {
10         String sentence;
11         String modifiedSentence;
12
13         BufferedReader inFromUser = new BufferedReader(new InputStreamReader(System.in));
14
15         Socket clientSocket = new Socket("localhost", 1050);
16
17         DataOutputStream outToServer = new DataOutputStream(clientSocket.getOutputStream());
18
19         BufferedReader inFromServer = new BufferedReader(new InputStreamReader(clientSocket.getInputStream()));
20
21         sentence = inFromUser.readLine();
22
23         outToServer.writeBytes(sentence + '\n');
24
25         modifiedSentence = inFromServer.readLine();
26
27         System.out.println("FROM SERVER: " + modifiedSentence);
28
29         clientSocket.close();
30     }
31 }
```

The console window on the right shows the output of the program:

```
<terminated> TCPClient (2) [Java Application] C:\Program Files\Java\j
hello
FROM SERVER: HELLO
```

The screenshot shows the Eclipse IDE interface. The main editor window displays the code for TCPServer.java. The code is as follows:

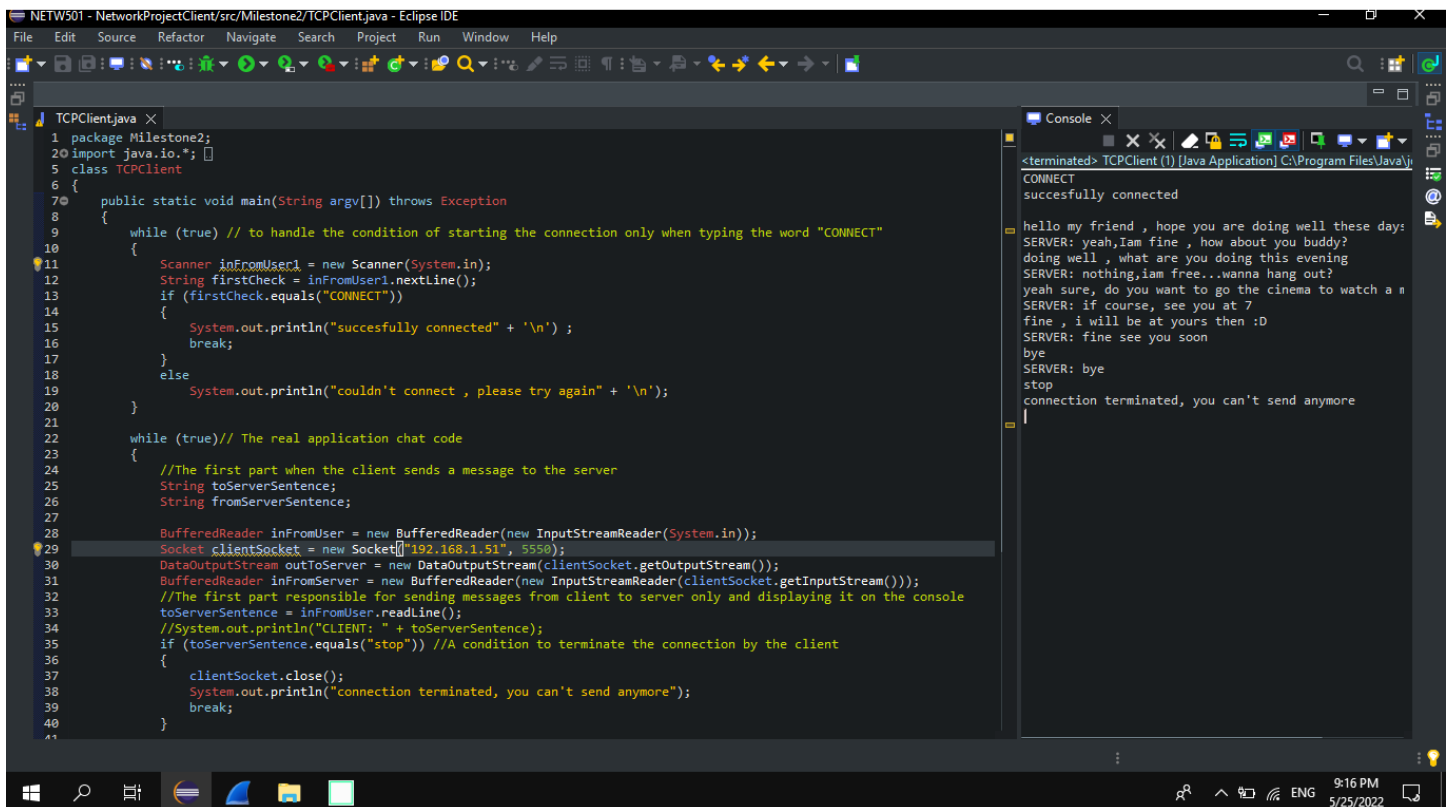
```
1 package Milestone1;
2 import java.io.*;
3
4 class TCPServer
5 {
6
7
8     public static void main(String argv[]) throws Exception
9     {
10         String clientSentence;
11         String capitalizedSentence;
12
13         ServerSocket welcomeSocket = new ServerSocket(1808);
14
15         while(true)
16         {
17
18             Socket connectionSocket = welcomeSocket.accept();
19
20             BufferedReader inFromClient = new BufferedReader(new InputStreamReader(connectionSocket.getInputStream()));
21             DataOutputStream outToClient = new DataOutputStream(connectionSocket.getOutputStream());
22
23             clientSentence = inFromClient.readLine();
24
25             capitalizedSentence = clientSentence.toUpperCase() + '\n';
26
27             outToClient.writeBytes(capitalizedSentence);
28         }
29     }
30 }
```

The console window on the right shows the output of the program. It displays the message "hello" from the client and "FROM SERVER: HELLO" from the server. The status bar at the bottom indicates the file is "Writable", "Smart Insert" is active, and the time is 13:56:279 on 5/25/2022.

MILESTONE2

1. In this milestone we tend to establish the real chatting application by editing our code as follows:
2. Firstly, the client can only establish the connection by typing the word "CONNECT" in the console of the client, otherwise no connection will be established so I'm using a loop to fulfill this case.
3. Secondly, once a connection is established the client send the first message to the server in one line and then presses enter , so the statement will apparat immediately in the console of the server
4. The Client and the server will keep sending the message alternatively each message per turn for only one party of the server and client.
5. The Chat will keep going till one of the party enters the word "stop", after that the socket will be closed from both sider and the connection will be terminated i.e. no one of the party will send any thing any more or an exception will be thrown.

The code for the client and the server as shown down here respectively

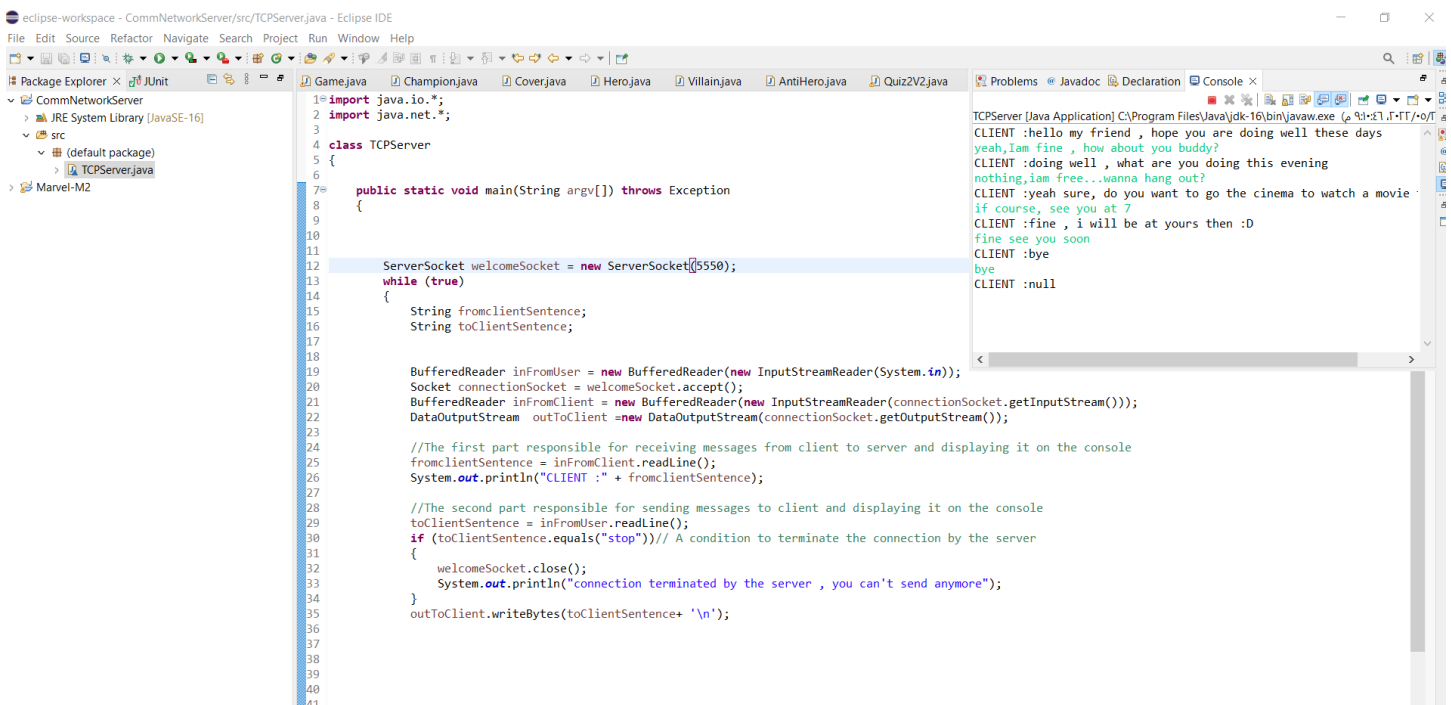


The screenshot shows the Eclipse IDE with the file `TCPCClient.java` open. The code is as follows:

```
1 package Milestone2;
2 import java.io.*;
3
4 class TCPCClient
5 {
6
7     public static void main(String argv[]) throws Exception
8     {
9         while (true) // to handle the condition of starting the connection only when typing the word "CONNECT"
10        {
11            Scanner inFromUser1 = new Scanner(System.in);
12            String firstCheck = inFromUser1.nextLine();
13            if (firstCheck.equals("CONNECT"))
14            {
15                System.out.println("succesfully connected" + '\n');
16                break;
17            }
18            else
19                System.out.println("couldn't connect , please try again" + '\n');
20        }
21
22        while (true) // The real application chat code
23        {
24            //The first part when the client sends a message to the server
25            String toServerSentence;
26            String fromServerSentence;
27
28            BufferedReader inFromUser = new BufferedReader(new InputStreamReader(System.in));
29            Socket clientSocket = new Socket("192.168.1.51", 5550);
30            DataOutputStream outToServer = new DataOutputStream(clientSocket.getOutputStream());
31            BufferedReader inFromServer = new BufferedReader(new InputStreamReader(clientSocket.getInputStream()));
32            //The first part responsible for sending messages from client to server only and displaying it on the console
33            toServerSentence = inFromUser.readLine();
34            //System.out.println("CLIENT: " + toServerSentence);
35            if (toServerSentence.equals("stop")) //A condition to terminate the connection by the client
36            {
37                clientSocket.close();
38                System.out.println("connection terminated, you can't send anymore");
39                break;
40            }
41        }
42    }
43 }
```

The console output shows the following sequence of events:

```
<terminated> TCPCClient (1) [Java Application] C:\Program Files\Java\j
CONNECT
succesfully connected
hello my friend , hope you are doing well these days
SERVER: yeah,iam fine , how about you buddy?
doing well , what are you doing this evening
SERVER: nothing,iam free...wanna hang out?
yeah sure , do you want to go the cinema to watch a m
SERVER: if course, see you at 7
fine , i will be at yours then :D
SERVER: fine see you soon
bye
SERVER: bye
stop
connection terminated, you can't send anymore
```



The screenshot shows the Eclipse IDE with the file `TCPServer.java` open. The code is as follows:

```
1 import java.io.*;
2 import java.net.*;
3
4 class TCPServer
5 {
6
7     public static void main(String argv[]) throws Exception
8     {
9
10
11
12        ServerSocket welcomeSocket = new ServerSocket(5550);
13        while (true)
14        {
15            String fromClientSentence;
16            String toClientSentence;
17
18
19            BufferedReader inFromUser = new BufferedReader(new InputStreamReader(System.in));
20            Socket connectionSocket = welcomeSocket.accept();
21            BufferedReader inFromClient = new BufferedReader(new InputStreamReader(connectionSocket.getInputStream()));
22            DataOutputStream outToClient = new DataOutputStream(connectionSocket.getOutputStream());
23
24            //The first part responsible for receiving messages from client to server and displaying it on the console
25            fromClientSentence = inFromClient.readLine();
26            System.out.println("CLIENT : " + fromClientSentence);
27
28            //The second part responsible for sending messages to client and displaying it on the console
29            toClientSentence = inFromUser.readLine();
30            if (toClientSentence.equals("stop")) // A condition to terminate the connection by the server
31            {
32                welcomeSocket.close();
33                System.out.println("connection terminated by the server , you can't send anymore");
34            }
35            outToClient.writeBytes(toClientSentence + '\n');
36        }
37    }
38 }
39
40
41
```

The console output shows the following sequence of events:

```
TCPServer [Java Application] C:\Program Files\Java\jdk-16\bin\javaw.exe (a 9d11f1f1-0f1
CLIENT :hello my friend , hope you are doing well these days
yeah,iam fine , how about you buddy?
CLIENT :doing well , what are you doing this evening
nothing,iam free...wanna hang out?
CLIENT :yeah sure , do you want to go the cinema to watch a movie
if course, see you at 7
CLIENT :fine , i will be at yours then :D
CLIENT :bye
CLIENT :bye
CLIENT :null
```

MILESTONE3

- In this milestone we use the wireshark sniffing application to trace the packet we sent.
- We chose wifi then set ip.addr = our ip address
- We trace any packet we find that is using the TCP layering

Capturing from Wi-Fi

File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

ip.addr == 192.168.1.51

| No. | Time | Source | Destination | Protocol | Length | Info |
|-----|-----------|----------------|----------------|----------|--------|--|
| 1 | 0.000000 | 192.168.1.48 | 192.168.1.51 | TCP | 55 | 53156 → 5552 [PSH, ACK] Seq=1 Ack=1 Win=256 Len=1 |
| 2 | 0.270157 | 192.168.1.51 | 192.168.1.48 | TCP | 54 | 5552 → 53156 [ACK] Seq=1 Ack=2 Win=513 Len=0 |
| 3 | 0.270262 | 192.168.1.48 | 192.168.1.51 | TCP | 57 | 53156 → 5552 [PSH, ACK] Seq=2 Ack=1 Win=256 Len=3 |
| 4 | 0.566808 | 192.168.1.51 | 192.168.1.48 | TCP | 54 | 5552 → 53156 [ACK] Seq=1 Ack=5 Win=513 Len=0 |
| 7 | 17.366418 | 192.168.1.49 | 224.0.0.251 | MDNS | 130 | Standard query 0x0000 PTR _companion-link_tcp.local, "QU" question PTR _homekit_tcp.local, "QU" question OPT |
| 9 | 18.390369 | 192.168.1.49 | 224.0.0.251 | MDNS | 130 | Standard query 0x0000 PTR _companion-link_tcp.local, "QM" question PTR _homekit_tcp.local, "QM" question OPT |
| 11 | 20.734545 | 192.168.1.48 | 64.233.167.188 | TCP | 55 | 53076 → 5228 [ACK] Seq=1 Ack=1 Win=256 Len=1 |
| 12 | 21.052109 | 64.233.167.188 | 192.168.1.48 | TCP | 54 | 5228 → 53076 [RST] Seq=1 Win=0 Len=0 |
| 13 | 22.486206 | 192.168.1.51 | 192.168.1.48 | TCP | 55 | 5552 → 53156 [PSH, ACK] Seq=1 Ack=5 Win=513 Len=1 |
| 14 | 22.527242 | 192.168.1.48 | 192.168.1.51 | TCP | 54 | 53156 → 5552 [ACK] Seq=5 Ack=2 Win=256 Len=0 |
| 15 | 22.530245 | 192.168.1.51 | 192.168.1.48 | TCP | 56 | 5552 → 53156 [PSH, ACK] Seq=1 Ack=5 Win=513 Len=2 |

Frame 3: 57 bytes on wire (456 bits), 57 bytes captured (456 bits) on interface \Device\NPF_{8CF1EAEA-0154-49F5-91FA-0FB48BA3F326}, id 0

Ethernet II, Src: LiteonTe_3e:42:10 (74:e5:43:3e:42:10), Dst: Chongqin_cd:47:a9 (e8:6f:38:cd:47:a9)

Destination: Chongqin_cd:47:a9 (e8:6f:38:cd:47:a9)

Address: Chongqin_cd:47:a9 (e8:6f:38:cd:47:a9)

... .. = LG bit: Globally unique address (factory default)

... .. = IG bit: Individual address (unicast)

Source: LiteonTe_3e:42:10 (74:e5:43:3e:42:10)

Address: LiteonTe_3e:42:10 (74:e5:43:3e:42:10)

... .. = LG bit: Globally unique address (factory default)

... .. = IG bit: Individual address (unicast)

Type: IPv4 (0x0800)

Internet Protocol Version 4, Src: 192.168.1.48, Dst: 192.168.1.51

Transmission Control Protocol, Src Port: 53156, Dst Port: 5552, Seq: 2, Ack: 1, Len: 3

Source Port: 53156

Destination Port: 5552

[Stream index: 0]

[Conversation completeness: Incomplete (12)]

[TCP Segment Len: 3]

0000 e8 6f 38 cd 47 a9 74 e5 43 3e 42 10 08 00 45 00 c0 6f 38 cd 47 a9 E..

0010 00 2b 24 64 40 00 00 06 52 b5 c0 a8 01 30 c0 a8 5d 00 R...0..

0020 01 33 cf a4 15 b0 41 e8 f9 96 28 1b 20 27 50 18 3...A...('P-

0030 01 00 4a 8c 00 00 6d 73 0a ...J...ms

Wi-Fi: <live capture in progress> | Packets: 273 · Displayed: 192 (70.3%) | Profile: Default

9:18 PM 5/25/2022

Wireshark · Packet 3 · Wi-Fi

Address: Chongqin_cd:47:a9 (e8:6f:38:cd:47:a9)

... .. = LG bit: Globally unique address (factory default)

... .. = IG bit: Individual address (unicast)

Source: LiteonTe_3e:42:10 (74:e5:43:3e:42:10)

Address: LiteonTe_3e:42:10 (74:e5:43:3e:42:10)

... .. = LG bit: Globally unique address (factory default)

... .. = IG bit: Individual address (unicast)

Type: IPv4 (0x0800)

Internet Protocol Version 4, Src: 192.168.1.48, Dst: 192.168.1.51

Transmission Control Protocol, Src Port: 53156, Dst Port: 5552, Seq: 2, Ack: 1, Len: 3

Source Port: 53156

Destination Port: 5552

[Stream index: 0]

[Conversation completeness: Incomplete (12)]

[TCP Segment Len: 3]

Sequence Number: 2 (relative sequence number)

Sequence Number (raw): 1105787286

[Next Sequence Number: 5 (relative sequence number)]

Acknowledgment Number: 1 (relative ack number)

Acknowledgment number (raw): 672866343

0101 = Header Length: 20 bytes (5)

Flags: 0x018 (PSH, ACK)

Window: 256

[Calculated window size: 256]

[Window size scaling factor: -1 (unknown)]

Checksum: 0x4a8c [unverified]

[Checksum Status: Unverified]

Urgent Pointer: 0

[Timestamps]

[SEQ/ACK analysis]

TCP payload (3 bytes)

Data (3 bytes)

0000 e8 6f 38 cd 47 a9 74 e5 43 3e 42 10 08 00 45 00 c0 6f 38 cd 47 a9 E..

0010 00 2b 24 64 40 00 00 06 52 b5 c0 a8 01 30 c0 a8 5d 00 R...0..

ACK] Seq=1 Ack=1 Win=256 Len=1

Seq=1 Ack=2 Win=513 Len=0

ACK] Seq=2 Ack=1 Win=256 Len=3

Seq=1 Ack=5 Win=513 Len=0

000 PTR _companion-link_tcp.local, "QU" question PTR _homekit_tcp.local, "QU" question OPT

000 PTR _companion-link_tcp.local, "QM" question PTR _homekit_tcp.local, "QM" question OPT

Seq=1 Ack=1 Win=256 Len=1

Seq=1 Win=0 Len=0

ACK] Seq=1 Ack=5 Win=513 Len=1

Seq=5 Ack=2 Win=256 Len=0

ACK] Seq=1 Ack=5 Win=513 Len=2

CF1EAEA-0154-49F5-91FA-0FB48BA3F326}, id 0

7:a9)

Packets: 303 · Displayed: 210 (69.3%) | Profile: Default

9:18 PM