



**Department of Computer Science and Engineering**  
**Islamic University of Technology (IUT)**  
A subsidiary organ of OIC

**Laboratory Report**

**CSE 4412: Data Communication and Networking Lab**

**Name** : M M Nazmul Hossain  
**Student ID** : 200042118  
**Section** : 1  
**Semester** : 4th  
**Academic Year** : 2021-2022  
**Date of Submission** : 16.01.2023  
**Lab No** : 2

**Title:** Understanding the basics of OSI Model

**Objective:**

1. Examine HTTP Web Traffic
2. Display Elements of the TCP/IP Protocol Suite

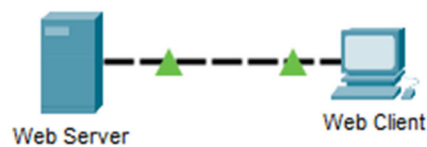
**Devices/ Software Used:**

Cisco Packet Tracer: Web Server and Web Client

**Working Procedure:**

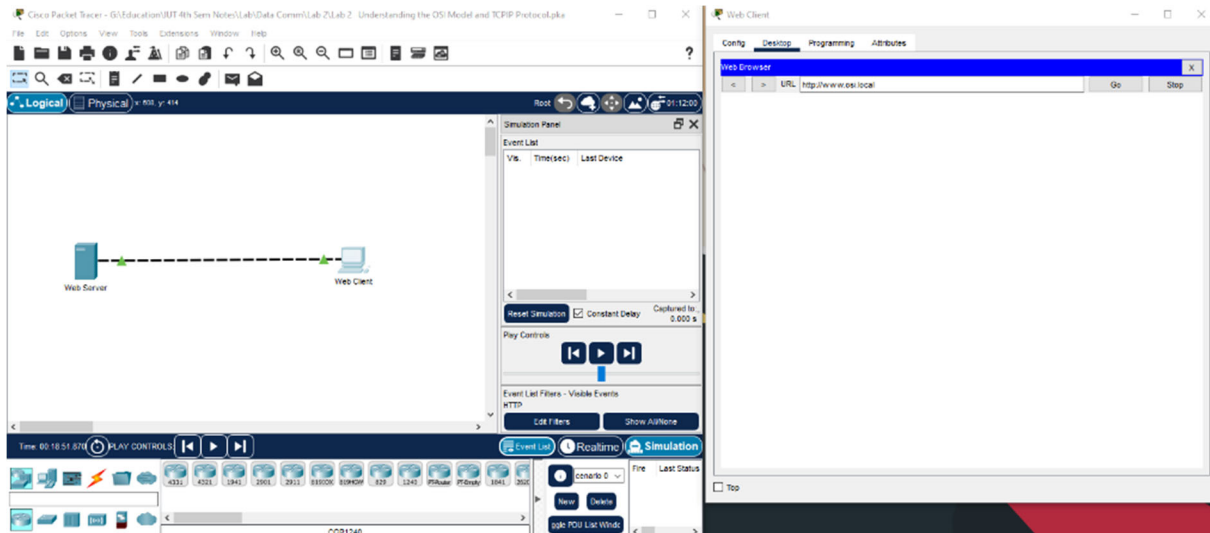
There is no working procedure for this lab as the setup was provided to us. The procedures were demonstrated.

**Diagram of the experiment:**



## Observation:

### Event – 1



PDU Information at Device: Web Client

**OSI Model**    Outbound PDU Details

At Device: Web Client Source: Web Client Destination: HTTP CLIENT	
<b>In Layers</b>	<b>Out Layers</b>
Layer7	Layer 7: HTTP
Layer6	Layer6
Layer5	Layer5
Layer4	Layer 4: TCP Src Port: 1034, Dst Port: 80
Layer3	Layer 3: IP Header Src. IP: 192.168.1.1, Dst. IP: 192.168.1.254
Layer2	Layer 2: Ethernet II Header 0060.47CA.4DEE >> 0001.96A9.401D
Layer1	Layer 1: Port(s):

1. The HTTP client sends a HTTP request to the server.

Challenge Me    << Previous Layer    Next Layer >>

The Web Client sends out an HTTP request to the server. In the different layers, the data is stored as packets by TCP, and source and destination addresses are determined. The source port is found to be **1034** and the destination port is found to be **80**. The source IP address is found to be **192.168.1.1**, and the Destination IP address is **192.168.1.254**. The destination MAC address is also found to be **0060.47CA.4DEE** for the source and **0001.96A9.401D** for the destination. The device encapsulates the PDU into an Ethernet frame.

## Event – 2

The image displays the Cisco Packet Tracer simulation environment. The main workspace shows a network topology with a Web Server and a Web Client connected. The Event List panel on the right shows a visible event at 0.005 seconds. The Web Client configuration window shows the URL `http://www.osi.local`. Below the main workspace, the PDU Information at Device: Web Client window is open, showing the Outbound PDU Details for an HTTP client request.

**PDU Information at Device: Web Client**

**OSI Model**    **Outbound PDU Details**

At Device: Web Client  
Source: Web Client  
Destination: HTTP CLIENT

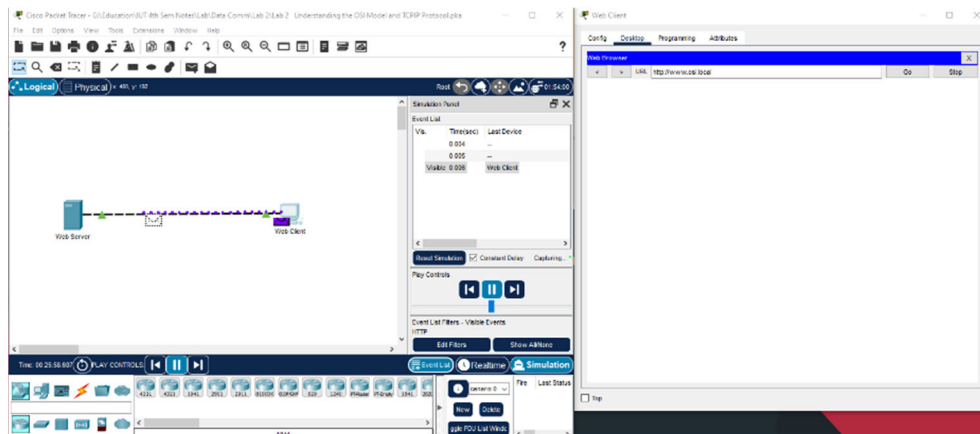
In Layers	Out Layers
Layer7	Layer7
Layer6	Layer6
Layer5	Layer5
Layer4	Layer4
Layer3	Layer3
Layer2	Layer2
Layer1	Layer 1: Port(s): FastEthernet0

1. The device takes out this frame from the buffer and sends it.  
2. FastEthernet0 sends out the frame.

Challenge Me    << Previous Layer    Next Layer >>

The device takes out the frame from the buffer and sends the request at the desired address, to the Web Server in this event. FastEthernet0 sends out the frame.

## Event – 3



PDU Information at Device: Web Server

OSI Model   Inbound PDU Details   Outbound PDU Details

At Device: Web Server  
Source: Web Client  
Destination: HTTP CLIENT

In Layers	Out Layers
Layer 7: HTTP	Layer 7: HTTP
Layer 6	Layer 6
Layer 5	Layer 5
Layer 4: TCP Src Port: 1034, Dst Port: 80	Layer 4: TCP Src Port: 80, Dst Port: 1034
Layer 3: IP Header Src, IP: 192.168.1.1, Dst, IP: 192.168.1.254	Layer 3: IP Header Src, IP: 192.168.1.254, Dst, IP: 192.168.1.1
Layer 2: Ethernet II Header 0060.47CA.4DEE >> 0001.96A9.401D	Layer 2: Ethernet II Header 0001.96A9.401D >> 0060.47CA.4DEE
Layer 1: Port FastEthernet0	Layer 1: Port(s): FastEthernet0

1. FastEthernet0 receives the frame.

Challenge Me   << Previous Layer   Next Layer >>

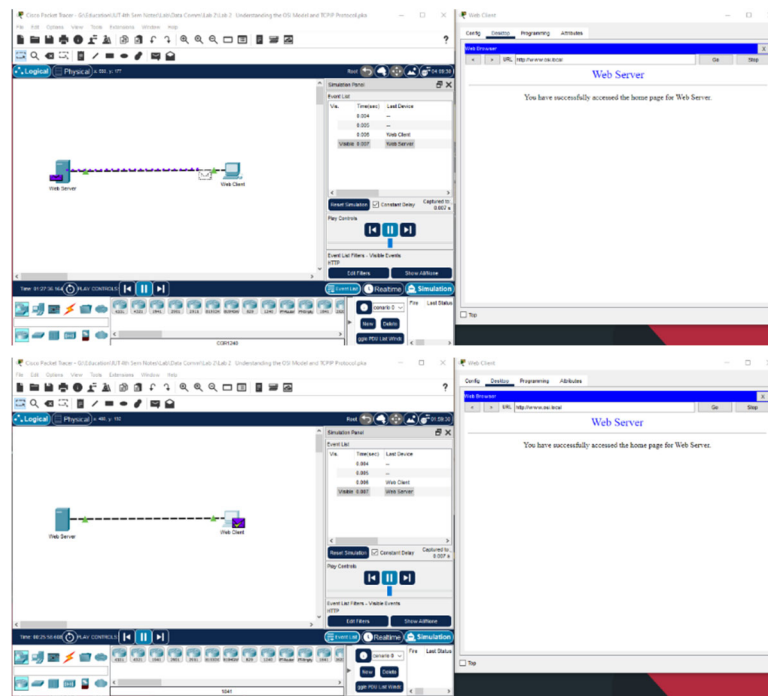
### In Layers

The frame is received at the Web Server and decrypted to match the addresses. The Mac address, IP Address and Port addresses are matched. The source is found to be our Web Client and the Destination is found to be the Web Server. Finally, the Web Server receives the Web Clients request after TCP decrypts the sent data.

### Out Layers

After receiving the request, the server sends back an HTTP reply to the client. The process is similar to event-1 except this time around the source and destination addresses are switched. The source port is found to be **80** and the destination port is found to be **1034**. The source IP address is found to be **192.168.1.254**. and the Destination IP address is **192.168.1.1**. The destination MAC address is also found to be **0001.96A9.401D** for the source and **0060.47CA.4DEE** for the destination. The device encapsulates the PDU into an Ethernet frame and FastEthernet0 sends out the frame.

## Event – 4



PDU Information at Device: Web Client

OSI Model	Inbound PDU Details
At Device: Web Client	
Source: Web Client	
Destination: HTTP CLIENT	
<b>In Layers</b>	
Layer 7: HTTP	
Layer 6	
Layer 5	
Layer 4: TCP Src Port: 80, Dst Port: 1034	
Layer 3: IP Header Src. IP: 192.168.1.254, Dst. IP: 192.168.1.1	
Layer 2: Ethernet II Header	
0001.96A9.401D >> 0060.47CA.4DEE	
Layer 1: Port FastEthernet0	
<b>Out Layers</b>	
Layer 7	
Layer 6	
Layer 5	
Layer 4	
Layer 3	
Layer 2	
Layer 1	
1. FastEthernet0 receives the frame.	

Challenge Me << Previous Layer Next Layer >>

The Web Client receives the frame. The frame is decrypted to get the addresses stored in it. The Mac address, IP address, and port address are all following the source and destination. The source is identified as our Web Server, and the destination is identified as the Web Client. Finally, the Web Client receives the HTTP response from the web Server and renders the page in the Web Browser.

## Challenges:

I encountered no difficulties in this lab because the procedure was a demonstration. All we had to do was jot down all the points raised during the discussion. It's difficult for me to understand why Event-2 is distinct from Event-1. As seen in Event-3, the HTTP request could be sent automatically in the same event. It is most likely due to the frame buffer. FastEthernet0 most likely had to check the buffer before sending the request.