



Birzeit University
Faculty of Engineering & Technology
Department of Electrical & Computer Engineering

Socket Programming

Prepared By:
Abdallah Abbasi 1232951
Jamal Shehadeh 1192316
GROUP: T051

Supervised By:
ibrahim nemer

Birzeit
May , 2025

Abstract

Basic socket programming and computer networking skills were practiced in this project. Three primary tasks included the work. In the first task, Wireshark was used to capture packet information and explain and use key network commands. The second task involved using socket programming to create a basic web server that served HTML and CSS files. In the third task, a hybrid system of TCP and UDP sockets was used to create a multiplayer guessing game. Screenshots were taken and every component was tested. The project's objective was to teach students about network programming and collaboration. Every task was finished and recorded in this report.

Table of Contents

English Abstract	I
Table of Contents	II
List of Tables	IV
List of Figures	V
1 Theory	1
1.1 Theory	1
1.1.1 Socket Programming	1
1.1.2 TCP and UDP	2
1.1.3 HTTP Protocol	2
1.1.4 HTML and CSS	2
1.1.5 Wireshark and DNS	3
1.1.6 Multithreading	3
1.1.7 HTTP Redirection	3
2 Results and Discussion	4
2.1 Task 1 – Network Commands and Wireshark	4
2.1.1 Part A – Explanation of Network Commands	4
2.1.2 Part B – Practical Tests	7
2.2 Task 2 – Implement a Simple Web Server Using Socket Programming	19
2.2.1 Task 2a – Main English Webpage (<code>main_en.html</code>)	19
2.2.2 Task 2b – Local Webpage (<code>mySite_1232951_en.html</code>)	22
2.2.3 Task 2c – Arabic Versions (<code>main_ar.html</code> and <code>mySite_1232951_ar.html</code>)	24
2.3 Task 2 (d) – Server Functionality	39
2.4 Task 2 (e) – Default and Error Responses	42
2.5 Task 2 – Experimental Cases	43

2.5.1	Case 1: Successful Image Request	43
2.5.2	Case 2: Successful Google Image Redirect (PNG)	43
2.5.3	Case 3: Successful Google Video Redirect (MP4)	44
2.5.4	Case 4: File Not Found (404 Error)	45
2.6	Task 3 – TCP/UDP Hybrid Client-Server Game Using Socket Programming	47
2.6.1	a) Game Features	47
2.6.2	c) Game Workflow	47
3	Conclusion	60

List of Tables

List of Figures

1.1	Difference between TCP and UDP communication	2
1.2	Example of HTTP request and response	2
1.3	Captured DNS query and response using Wireshark	3
2.1	ip config example	5
2.2	ping output	6
2.3	tracert output	6
2.4	telnet output	7
2.5	ns lookup output	7
2.6	ipconfig result – page 1	8
2.7	ipconfig result – page 2	9
2.8	ipconfig result – page 3	9
2.9	Ping to local device at 192.168.1.22	10
2.10	Ping to <code>gaia.cs.umass.edu</code> (128.119.245.12)	10
2.11	Traceroute showing path to <code>gaia.cs.umass.edu</code>	11
2.12	NSLookup of <code>gaia.cs.umass.edu</code> showing resolved IP	11
2.13	ASN topology and Tier 1 ISP connection overview	12
2.14	Raw WHOIS information for 128.119.0.0/16	13
2.15	DNS response from BGP.tools for <code>gaia.cs.umass.edu</code>	13
2.16	BGP registration data for UMass ASN	14
2.17	ASN prefixes originated from AS1249	15
2.18	Contact and technical details for AS1249	16
2.19	Detailed routing and registry metadata	17
2.20	DNS query for <code>gaia.cs.umass.edu</code> sent to 192.168.1.1	18
2.21	DNS response received with IP address 128.119.245.12	18
2.22	HTML code – Team members section	20
2.23	HTML code – FDM vs TDM topic section	21

2.24	HTML code – Protocol stack section and external links	22
2.25	Final rendered output of <code>main_en.html</code> in browser	22
2.26	HTML code of <code>mySite_1232951_en.html</code> – file request form	23
2.27	Rendered view of local file request page	24
2.28	HTML code – Arabic main page (team member section)	25
2.29	HTML code – Arabic main page (topic and image sections)	26
2.30	HTML code – Arabic main page footer	27
2.31	HTML code – Arabic local request page (<code>mySite_1232951_ar.html</code>) . . .	28
2.32	Rendered view of Arabic main page (<code>main_ar.html</code>)	29
2.33	Rendered view of Arabic local page (<code>mySite_1232951_ar.html</code>)	29
2.34	CSS base variables and body styling (<code>Style.css</code>)	30
2.35	Navigation bar, team sections, and headings	31
2.36	Team card design, avatar styling, and responsive layout	32
2.37	Topic section layout and styling	33
2.38	Topic content spacing, image layout, and hover styles	34
2.39	Footer styles and small screen adjustments	35
2.40	General layout and team member section CSS	36
2.41	Form design and padding for mobile screens	37
2.42	Right-to-left (RTL) layout rules for Arabic pages (<code>rtl.css</code>)	38
2.43	Server setup and 404 response generation	40
2.44	Routing logic and redirection handling	41
2.45	Image/video redirect logic and MIME detection	42
2.46	Image successfully retrieved (<code>test1.png</code>)	43
2.47	Terminal log showing successful image request and 200 OK response	43
2.48	Image redirection example for <code>abdallah.png</code> (status 307)	44
2.49	Redirected search result for <code>jamal.mp4</code> to YouTube (307 Temporary Redirect)	44
2.50	Redirect behaviour for video file (<code>jamal.mp4</code>) shown in browser and developer tools	45
2.51	Client-side display of 404 Not Found error for <code>jamal.txt</code>	45
2.52	Server-side log output for 404 Not Found (<code>jamal.txt</code>)	46
2.53	Client-side socket setup	48
2.54	UDP game loop and message handling	49
2.55	Server TCP ping-pong and client monitoring	50
2.56	UDP player management and gameplay round	51
2.57	UDP guess handling and feedback logic	52

2.58	Round broadcasting and result distribution	53
2.59	TCP handling thread logic	54
2.60	Final game loop and result tracking	55
2.61	Two players join and the game starts. A bug happens.	56
2.62	Client inputs a guess outside valid range and receives a warning.	57
2.63	Server detects disconnected player and handles it gracefully.	58

Chapter 1

Theory

Contents

2.1 Task 1 – Network Commands and Wireshark	4
2.1.1 Part A – Explanation of Network Commands	4
2.1.2 Part B – Practical Tests	7
2.2 Task 2 – Implement a Simple Web Server Using Socket Programming	19
2.2.1 Task 2a – Main English Webpage (<code>main_en.html</code>)	19
2.2.2 Task 2b – Local Webpage (<code>mySite_1232951_en.html</code>)	22
2.2.3 Task 2c – Arabic Versions (<code>main_ar.html</code> and <code>mySite_1232951_ar.html</code>)	24
2.3 Task 2 (d) – Server Functionality	39
2.4 Task 2 (e) – Default and Error Responses	42
2.5 Task 2 – Experimental Cases	43
2.5.1 Case 1: Successful Image Request	43
2.5.2 Case 2: Successful Google Image Redirect (PNG)	43
2.5.3 Case 3: Successful Google Video Redirect (MP4)	44
2.5.4 Case 4: File Not Found (404 Error)	45
2.6 Task 3 – TCP/UDP Hybrid Client-Server Game Using Socket Programming	47
2.6.1 a) Game Features	47
2.6.2 c) Game Workflow	47

1.1 Theory

1.1.1 Socket Programming

Socket programming was used to make communication between devices possible. A socket is a software object that was used to send and receive messages. In this project, TCP and UDP sockets were used. TCP was used when a reliable connection was needed, and UDP was used when fast communication was more important than accuracy.

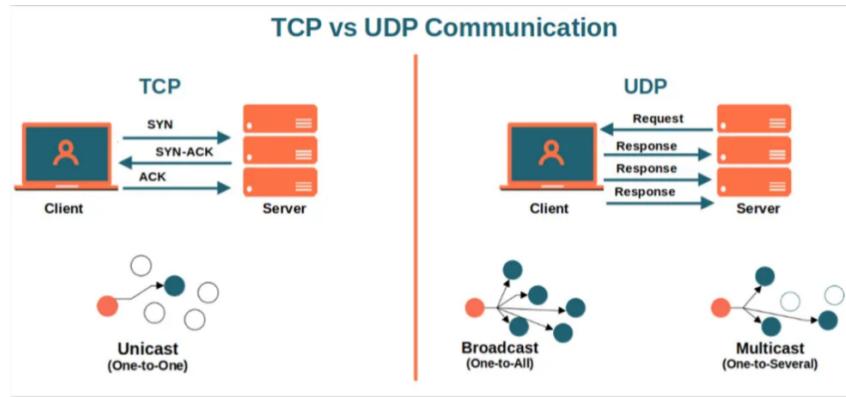


Figure 1.1: Difference between TCP and UDP communication

1.1.2 TCP and UDP

TCP (Transmission Control Protocol) is a connection-based protocol that was used to send data in order. It was helpful when messages had to arrive correctly. UDP (User Datagram Protocol) is a connectionless protocol that was used to send data quickly, but messages could be lost or arrive out of order. Both protocols were used in the game task.

1.1.3 HTTP Protocol

The HTTP (HyperText Transfer Protocol) was used to allow web browsers to talk to the server. When the browser sent a request, the server sent back a response with a file such as an HTML or CSS file. The response also included a status code like 200 (OK), 404 (Not Found), or 307 (Redirect).

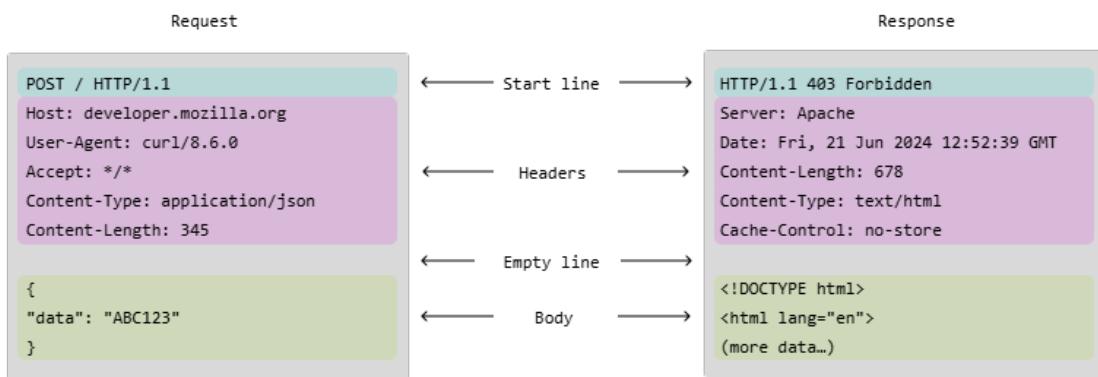


Figure 1.2: Example of HTTP request and response

1.1.4 HTML and CSS

HTML was used to build the content of the web pages. CSS was used to style the pages by changing colours, fonts, and layout. These files were stored in folders and were sent by the server when requested by the browser.

1.1.5 Wireshark and DNS

Wireshark is a tool used to capture data packets sent over the network. It's used in this project to capture a DNS (Domain Name System) query and response when a website was visited. DNS is a system that was used to change a website name into an IP address.

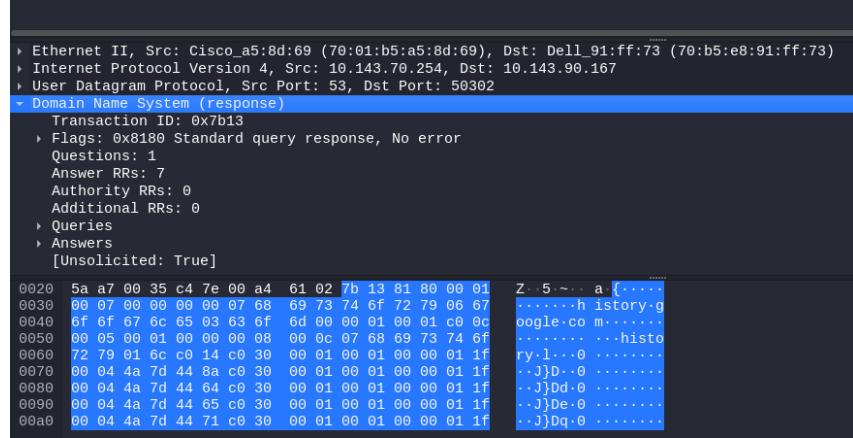


Figure 1.3: Captured DNS query and response using Wireshark

1.1.6 Multithreading

Multithreading is a technique that allows a program to do many things at the same time. In this project, threads were used so that the server could handle multiple players or requests at once without waiting.

1.1.7 HTTP Redirection

HTTP redirection was used when a file was not found. The server sent back a 307 redirect response. This told the browser to go to another URL. For example, if an image or video was not found, the browser was redirected to a Google search page.

Chapter 2

Results and Discussion

2.1 Task 1 – Network Commands and Wireshark

2.1.1 Part A – Explanation of Network Commands

- **ipconfig:** This command displays Internet protocol details for both IPv4 and IPv6 with their addresses, subnet masks, default gateway and DNS suffix. All of this information linked to network adapters.

```

Select Administrator: Command Prompt
C:\Windows\system32>ipconfig

Windows IP Configuration

Ethernet adapter Ethernet:
   Media State . . . . . : Media disconnected
   Connection-specific DNS Suffix . :

Ethernet adapter Ethernet 2:
   Connection-specific DNS Suffix . :
      Link-local IPv6 Address . . . . . : fe80::2c36:376a:41c8:75e%3
      IPv4 Address . . . . . : 192.168.56.1
      Subnet Mask . . . . . : 255.255.255.0
      Default Gateway . . . . . :

Wireless LAN adapter Local Area Connection* 1:
   Media State . . . . . : Media disconnected
   Connection-specific DNS Suffix . :

Wireless LAN adapter Local Area Connection* 10:
   Media State . . . . . : Media disconnected
   Connection-specific DNS Suffix . :

Wireless LAN adapter Wi-Fi:
   Connection-specific DNS Suffix . :
      Link-local IPv6 Address . . . . . : fe80::c697:df0b:8c6d:6baa%5
      IPv4 Address . . . . . : 192.168.1.21
      Subnet Mask . . . . . : 255.255.255.0
      Default Gateway . . . . . : 192.168.1.1

Ethernet adapter Bluetooth Network Connection:
   Media State . . . . . : Media disconnected
   Connection-specific DNS Suffix . :

Select Administrator: Command Prompt
C:\Windows\system32>ipconfig /?

USAGE:
  ipconfig [/allcompartments] [/? | /all |
    /renew [adapter] | /release [adapter] |
    /release6 [adapter] | /release6 [adapter] |
    /flushdns | /displaydns | /registerdns |
    /showclassid adapter |
    /setclassid adapter [classid] |
    /showclassid6 adapter |
    /setclassid6 adapter [classid] ]

where
  adapter          Connection name
                  (wildcard characters * and ? allowed, see examples)

Options:
  /?
  /all             Display this help message
  /release         Release the IPv4 address for the specified adapter.
  /release6        Release the IPv6 address for the specified adapter.
  /renew           Renew the IPv4 address for the specified adapter.
  /renew6          Renew the IPv6 address for the specified adapter.
  /flushdns        Purges the DNS Resolver cache.
  /registerdns    Refreshes all DHCP leases and re-registers DNS names
  /displaydns     Displays the contents of the DNS Resolver Cache.
  /showclassid    Displays all the dhcp class IDs allowed for adapter.
  /setclassid     Modifies the dhcp class id.
  /showclassid6   Displays all the IPv6 DHCP class IDs allowed for adapter.
  /setclassid6   Modifies the IPv6 DHCP class id.

The default is to display only the IP address, subnet mask and
default gateway for each adapter bound to TCP/IP.

For Release and Renew, if no adapter name is specified, then the IP address
leases for all adapters bound to TCP/IP will be released or renewed.

For Setclassid and Setclassid6, if no ClassId is specified, then the ClassId is removed.

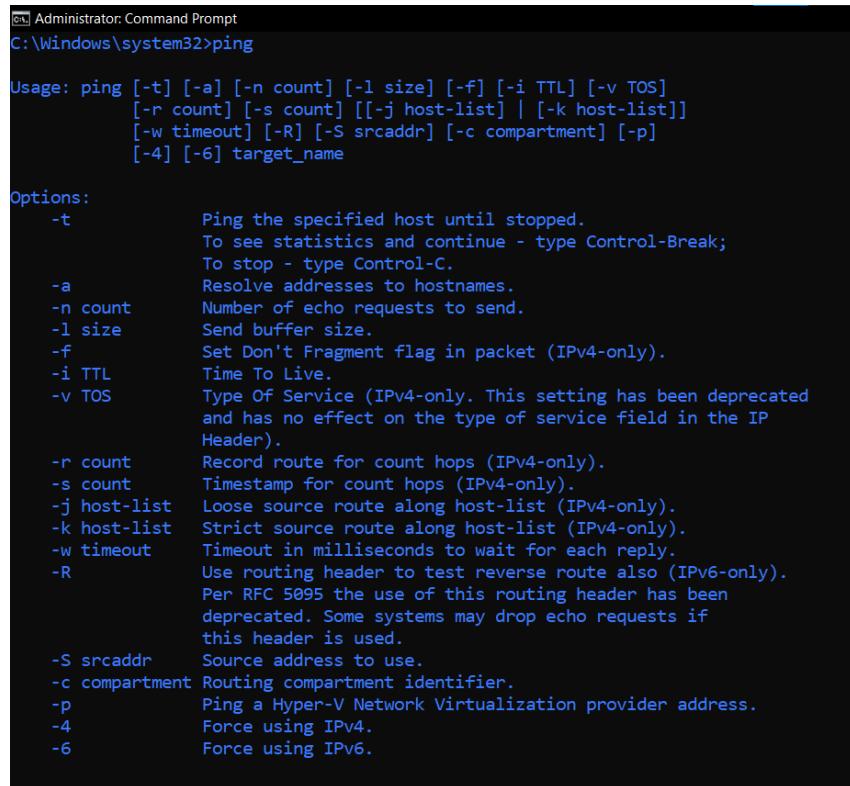
Examples:
  > ipconfig           ... Show information
  > ipconfig /all       ... Show detailed information
  > ipconfig /renew     ... renew all adapters
  > ipconfig /renew EL* ... renew any connection that has its
                         name starting with EL
  > ipconfig /release *Con* ... release all matching connections,
                             eg. "Wired Ethernet Connection 1" or
                             "Wired Ethernet Connection 2"
  > ipconfig /allcompartments ... Show information about all
                                compartments
  > ipconfig /allcompartments /all ... Show detailed information about all
                                    compartments

```

Figure 2.1: ip config example

- **ping(Packet Internet Grouper)**: writing this command itself displays its manual ,how to use it and what it does if a parameter has been used. Its main use is testing

internet/host reachability, internet connectivity, network interface card and DNS issues.



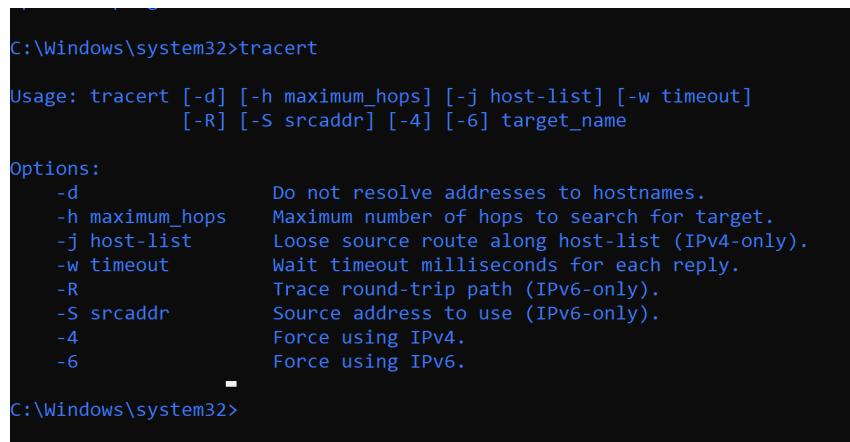
```
C:\Windows\system32>ping

Usage: ping [-t] [-a] [-n count] [-l size] [-f] [-i TTL] [-v TOS]
            [-r count] [-s count] [[-j host-list] | [-k host-list]]
            [-w timeout] [-R] [-S srcaddr] [-c compartment] [-p]
            [-4] [-6] target_name

Options:
  -t          Ping the specified host until stopped.
              To see statistics and continue - type Control-Break;
              To stop - type Control-C.
  -a          Resolve addresses to hostnames.
  -n count    Number of echo requests to send.
  -l size     Send buffer size.
  -f          Set Don't Fragment flag in packet (IPv4-only).
  -i TTL      Time To Live.
  -v TOS      Type Of Service (IPv4-only. This setting has been deprecated
              and has no effect on the type of service field in the IP
              Header).
  -r count    Record route for count hops (IPv4-only).
  -s count    Timestamp for count hops (IPv4-only).
  -j host-list Loose source route along host-list (IPv4-only).
  -k host-list Strict source route along host-list (IPv4-only).
  -w timeout  Timeout in milliseconds to wait for each reply.
  -R          Use routing header to test reverse route also (IPv6-only).
              Per RFC 5095 the use of this routing header has been
              deprecated. Some systems may drop echo requests if
              this header is used.
  -S srcaddr  Source address to use.
  -c compartment Routing compartment identifier.
  -p          Ping a Hyper-V Network Virtualization provider address.
  -4          Force using IPv4.
  -6          Force using IPv6.
```

Figure 2.2: ping output

- **tracert / traceroute:**It displays the path that packets from my computer take to reach the destination. It is a step-by-step overview of each network hop along the route, using this command without parameters displays the manual.



```
C:\Windows\system32>tracert

Usage: tracert [-d] [-h maximum_hops] [-j host-list] [-w timeout]
                [-R] [-S srcaddr] [-4] [-6] target_name

Options:
  -d          Do not resolve addresses to hostnames.
  -h maximum_hops Maximum number of hops to search for target.
  -j host-list Loose source route along host-list (IPv4-only).
  -w timeout   Wait timeout milliseconds for each reply.
  -R          Trace round-trip path (IPv6-only).
  -S srcaddr   Source address to use (IPv6-only).
  -4          Force using IPv4.
  -6          Force using IPv6.

C:\Windows\system32>
```

Figure 2.3: tracert output

- **telnet:**It is a terminal emulation program that is used to access remote servers and control them. writing the command without any parameters takes you to the telnet command line itself where you can write commands to connect to a specified server.

```

Administrator: Command Prompt
Welcome to Microsoft Telnet Client

Escape Character is 'CTRL+['

Microsoft Telnet> quit

C:\Windows\system32>telnet/?

telnet [-a][-e escape char][-f log file][-l user][-t term][host [port]]
-a      Attempt automatic logon. Same as -l option except uses
       the currently logged on user's name.
-e      Escape character to enter telnet client prompt.
-f      File name for client side logging
-l      Specifies the user name to log in with on the remote system.
       Requires that the remote system support the TELNET ENVIRON option.
-t      Specifies terminal type.
       Supported term types are vt100, vt52, ansi and vtnt only.
host   Specifies the hostname or IP address of the remote computer
       to connect to.
port   Specifies a port number or service name.

C:\Windows\system32>

```

Figure 2.4: telnet output

- **nslookup:** Displays information about DNS (Domain Name System), including default DNS server, name and the address (IP).

```

C:\Windows\system32>nslookup/?
Usage:
  nslookup [-opt ...]          # interactive mode using default server
  nslookup [-opt ...] - server  # interactive mode using 'server'
  nslookup [-opt ...] host     # just look up 'host' using default server
  nslookup [-opt ...] host server # just look up 'host' using 'server'

C:\Windows\system32>nslookup
Default Server: Unknown
Address: 192.168.1.1

> hi
Server: Unknown
Address: 192.168.1.1
*** Unknown can't find hi: Non-existent domain
>
> google.com
Server: Unknown
Address: 192.168.1.1

Non-authoritative answer:
Name: google.com
Addresses: 2a00:1450:4028:805::200e
           142.250.75.78

```

Figure 2.5: ns lookup output

2.1.2 Part B – Practical Tests

1. **IP Configuration** The `ipconfig /all` command was executed to find network configuration details. The following data were obtained:

- IP Address: 192.168.1.18
- Subnet Mask: 255.255.255.0
- Default Gateway: 192.168.1.1
- DNS Server: 192.168.1.1

```
Administrator: Command Prompt
Microsoft Windows [Version 10.0.19045.5737]
(c) Microsoft Corporation. All rights reserved.

C:\Windows\system32>color 9

C:\Windows\system32>ipconfig/all

Windows IP Configuration

 Host Name . . . . . : DESKTOP-RCTHIVK
 Primary Dns Suffix . . . . . :
 Node Type . . . . . : Hybrid
 IP Routing Enabled. . . . . : No
 WINS Proxy Enabled. . . . . : No

Ethernet adapter Ethernet:

 Media State . . . . . : Media disconnected
 Connection-specific DNS Suffix . . . . . :
 Description . . . . . : Realtek Gaming GbE Family Controller
 Physical Address. . . . . : C8-5A-CF-B2-F7-F7
 DHCP Enabled. . . . . : Yes
 Autoconfiguration Enabled . . . . . : Yes

Ethernet adapter Ethernet 2:

 Connection-specific DNS Suffix . . . . . :
 Description . . . . . : VirtualBox Host-Only Ethernet Adapter
 Physical Address. . . . . : 0A-00-27-00-00-03
 DHCP Enabled. . . . . : No
 Autoconfiguration Enabled . . . . . : Yes
 Link-local IPv6 Address . . . . . : fe80::2c36:376a:41c8:75e%3(Preferred)
 IPv4 Address. . . . . : 192.168.56.1(Preferred)
 Subnet Mask . . . . . : 255.255.255.0
 Default Gateway . . . . . :
 DHCPv6 IAID . . . . . : 889847847
 DHCPv6 Client DUID. . . . . : 00-01-00-01-2F-46-45-8C-C8-5A-CF-B2-F7-F7
 DNS Servers . . . . . : fec0:0:0:ffff::1%1
                           fec0:0:0:ffff::2%1
                           fec0:0:0:ffff::3%1
 NetBIOS over Tcpip. . . . . : Enabled
```

Figure 2.6: ipconfig result – page 1

```
Wireless LAN adapter Local Area Connection* 1:  
Media State . . . . . : Media disconnected  
Connection-specific DNS Suffix . :  
Description . . . . . : Microsoft Wi-Fi Direct Virtual Adapter  
Physical Address. . . . . : F4-26-79-D5-00-A3  
DHCP Enabled. . . . . : Yes  
Autoconfiguration Enabled . . . . . : Yes  
  
Wireless LAN adapter Local Area Connection* 10:  
Media State . . . . . : Media disconnected  
Connection-specific DNS Suffix . :  
Description . . . . . : Microsoft Wi-Fi Direct Virtual Adapter #2  
Physical Address. . . . . : F6-26-79-D5-00-A2  
DHCP Enabled. . . . . : No  
Autoconfiguration Enabled . . . . . : Yes  
  
Wireless LAN adapter Wi-Fi:  
Connection-specific DNS Suffix . :  
Description . . . . . : Intel(R) Wi-Fi 6 AX201 160MHz  
Physical Address. . . . . : F4-26-79-D5-00-A2  
DHCP Enabled. . . . . : Yes  
Autoconfiguration Enabled . . . . . : Yes  
Link-local IPv6 Address . . . . . : fe80::c697:df0b:8c6d:6baa%5(Preferred)  
IPv4 Address. . . . . : 192.168.1.18(Preferred)  
Subnet Mask . . . . . : 255.255.255.0  
Lease Obtained. . . . . : Thursday, May 8, 2025 8:58:39 PM  
Lease Expires . . . . . : Friday, May 9, 2025 8:58:38 PM  
Default Gateway . . . . . : 192.168.1.1  
DHCP Server . . . . . : 192.168.1.1  
DHCPv6 IAID . . . . . : 66332281  
DHCPv6 Client DUID. . . . . : 00-01-00-01-2F-46-45-8C-C8-5A-CF-B2-F7-F7  
DNS Servers . . . . . : 192.168.1.1  
NetBIOS over Tcpip. . . . . : Enabled
```

Figure 2.7: ipconfig result – page 2

```
Ethernet adapter Bluetooth Network Connection:  
Media State . . . . . : Media disconnected  
Connection-specific DNS Suffix . :  
Description . . . . . : Bluetooth Device (Personal Area Network)  
Physical Address. . . . . : F4-26-79-D5-00-A6  
DHCP Enabled. . . . . : Yes  
Autoconfiguration Enabled . . . . . : Yes
```

Figure 2.8: ipconfig result – page 3

- 2. Ping to a Local Device** A ping test was done to a phone connected to the same Wi-Fi network. All packets were received with no loss and an average latency of 22ms. This showed the connection was working well.

```
C:\Windows\system32>ping 192.168.1.22

Pinging 192.168.1.22 with 32 bytes of data:
Reply from 192.168.1.22: bytes=32 time=70ms TTL=64
Reply from 192.168.1.22: bytes=32 time=5ms TTL=64
Reply from 192.168.1.22: bytes=32 time=7ms TTL=64
Reply from 192.168.1.22: bytes=32 time=6ms TTL=64

Ping statistics for 192.168.1.22:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 5ms, Maximum = 70ms, Average = 22ms
```

Figure 2.9: Ping to local device at 192.168.1.22

3. Ping to gaia.cs.umass.edu The server `gaia.cs.umass.edu` was pinged successfully. Four packets were received with 0% loss. The average round-trip time was 136ms. This means the server was reachable and responded directly.

```
C:\Windows\system32>ping gaia.cs.umass.edu

Pinging gaia.cs.umass.edu [128.119.245.12] with 32 bytes of data:
Reply from 128.119.245.12: bytes=32 time=137ms TTL=41
Reply from 128.119.245.12: bytes=32 time=136ms TTL=41
Reply from 128.119.245.12: bytes=32 time=137ms TTL=41
Reply from 128.119.245.12: bytes=32 time=134ms TTL=41

Ping statistics for 128.119.245.12:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 134ms, Maximum = 137ms, Average = 136ms

C:\Windows\system32>
```

Figure 2.10: Ping to `gaia.cs.umass.edu` (128.119.245.12)

based on the information we received, here's what we learned: When we attempted to connect locally, the latency was very low, similar to that of the smartphone. However, when we tried connecting to a WAN server, the latency increased. This indicates that the packets successfully reached the server without encountering a firewall or any other obstacles, as all four packets were received with 0 loss.

4. Traceroute to gaia.cs.umass.edu The traceroute command showed that 25 hops were needed to reach the server. The route passed through different international networks and internet exchange points.

```
C:\Windows\system32>tracert gaia.cs.umass.edu

Tracing route to gaia.cs.umass.edu [128.119.245.12]
over a maximum of 30 hops:

 1   1 ms    2 ms    1 ms  192.168.1.1
 2   5 ms    4 ms    4 ms  10.74.32.246
 3  53 ms   54 ms   53 ms  10.74.19.25
 4   5 ms    4 ms    5 ms  10.74.19.178
 5  54 ms   53 ms   52 ms  10.74.59.158
 6  47 ms   47 ms   47 ms  et-4-0-13.edge1.Marseille3.Level3.net [213.242.111.121]
 7 113 ms   98 ms   109 ms  ae2.3605.edge4.mrs1.neo.colt.net [171.75.8.225]
 8   *       *       * Request timed out.
 9 186 ms   203 ms   223 ms  be2779.ccr41.par01.atlas.cogentco.com [154.54.72.109]
10  65 ms   68 ms   66 ms  be3684.ccr51.lhr01.atlas.cogentco.com [154.54.60.170]
11 137 ms   134 ms   134 ms  be3393.ccr31.bos01.atlas.cogentco.com [154.54.47.141]
12 137 ms   135 ms   135 ms  be8038.rcr71.orh02.atlas.cogentco.com [154.54.169.254]
13 140 ms   140 ms   140 ms  be8628.rcr51.orh01.atlas.cogentco.com [154.54.164.126]
14 131 ms   130 ms   131 ms  38.104.218.14
15 161 ms   136 ms   136 ms  69.16.0.8
16 136 ms   139 ms   136 ms  69.16.1.0
17 140 ms   142 ms   133 ms  core1-rt-et-8-3-0.gw.umass.edu [192.80.83.109]
18 137 ms   138 ms   138 ms  n1-rt-1-1-rt-0-0-0.gw.umass.edu [128.119.0.216]
19 136 ms   134 ms   134 ms  n1-fnt-fw-1-1-1-31-vl1092.gw.umass.edu [128.119.77.233]
20   *       *       * Request timed out.
21 134 ms   134 ms   135 ms  core1-rt-et-7-2-1.gw.umass.edu [128.119.0.217]
22 140 ms   137 ms   137 ms  n5-rt-1-1-xe-2-1-0.gw.umass.edu [128.119.3.33]
23 132 ms   131 ms   130 ms  cics-rt-xe-0-0-0.gw.umass.edu [128.119.3.32]
24   *       *       * Request timed out.
25 137 ms   138 ms   137 ms  gaia.cs.umass.edu [128.119.245.12]

Trace complete.

C:\Windows\system32>
```

Figure 2.11: Traceroute showing path to gaia.cs.umass.edu

5. NSLookup of gaia.cs.umass.edu The nslookup command was used to get DNS information. The IP address 128.119.245.12 was found for gaia.cs.umass.edu. The DNS server used was 192.168.1.1.

```
C:\Windows\system32>nslookup gaia.cs.umass.edu
Server: UnKnown
Address: 192.168.1.1

Non-authoritative answer:
Name:  gaia.cs.umass.edu
Address: 128.119.245.12
```

Figure 2.12: NSLookup of gaia.cs.umass.edu showing resolved IP

6. Telnet to Port 80 The telnet gaia.cs.umass.edu 80 command was used. A black screen appeared, meaning the connection to port 80 was successful. This showed that the web server was active and accepting connections.

7. Autonomous System (AS) Information Information about the AS that owns gaia.cs.umass.edu was gathered using <https://bgpview.io> and <https://bgp.tools>. The ASN is AS1249 and it belongs to the University of Massachusetts – AMHERST.

- AS Number: AS1249
- AS Name: FIVE-COLLEGES-AS
- Description: University of Massachusetts – AMHERST
- IP Prefixes:
 - 128.119.0.0/16
 - 72.19.64.0/18
 - 192.80.83.0/24
 - 192.189.138.0/24
- Peers: AS1968 (UMASSNET), AS168 (UMass Amherst)
- Upstream: AS1968 (UMASSNET)
- Tier 1 ISP: Not directly connected, but reachable via upstreams like Cogent and GTT
- Number of IPs: 65,536

Figures:

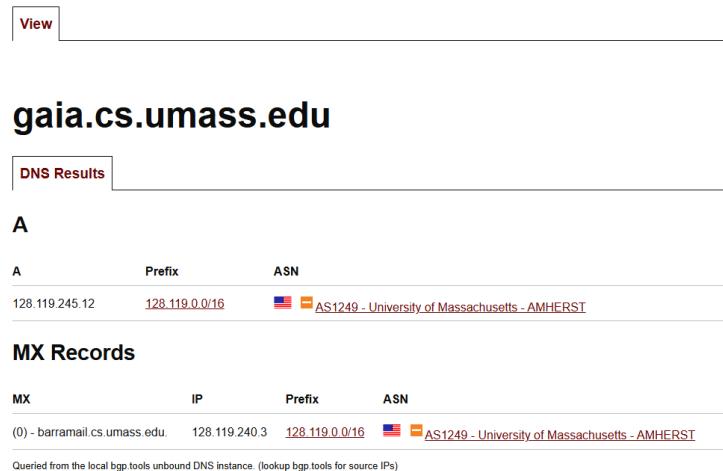


Figure 2.13: ASN topology and Tier 1 ISP connection overview

128.119.0.0/16
UNIVERSITY OF MASSACHUSETTS - AMHERST

Announcing ASNs: 1 Parent Prefix: Abuse: abuse@umass.edu RIR: IANA

Prefix	Routing	Raw Whois

Announcing ASNs

Country	ASN	Name	Description
	AS1249	FIVE-COLLEGES-AS	University of Massachusetts - AMHERST

128.119.0.0/16 Summary

PREFIX: 128.119.0.0/16	REGIONAL REGISTRY: IANA
NAME: UMASS-NET	ALLOCATION STATUS:
DESCRIPTION: University of Massachusetts - AMHERST	PARENT PREFIX:
COUNTRY:	IP ADDRESSES: 65,536

Contacts

EMAIL CONTACTS: hostmaster@umass.edu abuse@umass.edu	ABUSE CONTACTS: abuse@umass.edu	ADDRESS: University Computing Services, Lederle Graduate Research Center, Amherst, MA, 01003, US
---	---	--

POWERED BY **SecurityTrails**
A Recorded Future Company

Figure 2.14: Raw WHOIS information for 128.119.0.0/16

[View](#) [Edit](#)

University of Massachusetts - AMHERST

AS Number 1249
Website <http://umass.edu>

**BE BOLD. BE TRUE.
BE YOU.**
An iconic University of Massachusetts Amherst graduation scene featuring students in caps and gowns. The text "BE BOLD. BE TRUE. BE YOU." is overlaid on the bottom right, with a small fine print note below it.

[Overview](#) [Prefixes](#) [Connectivity](#) [Whois](#)

ASHandle:	AS1249
OrgID:	UNIVER-6
ASName:	FIVE-COLLEGES-AS
ASNNumber:	1249
RegDate:	1991-04-18
Updated:	2024-11-06
TechHandle:	PG138-ARIN
Source:	ARIN

OrgID:	UNIVER-6
OrgName:	University of Massachusetts - AMHERST
CanAllocate:	
Street:	University Computing Services
Street:	Lederle Graduate Research Center
City:	Amherst
State/Prov:	MA
Country:	US
PostalCode:	01003
RegDate:	1986-04-03
Updated:	2024-11-25
OrgAdminHandle:	JFD16-ARIN
OrgTechHandle:	ZU64-ARIN
OrgAbuseHandle:	ONS2-ARIN
OrgNOCHandle:	ZU64-ARIN
Source:	ARIN

Figure 2.15: DNS response from BGP.tools for gaia.cs.umass.edu

```
OrgAdminHandle: JFD16-ARIN
OrgAdminName: John Doyle
# Email removed, Log into bgp.tools to see whois emails
```

```
OrgTechHandle: ZU64-ARIN
OrgTechName: Hostmaster
# Email removed, Log into bgp.tools to see whois emails
```

```
OrgAbuseHandle: ONS2-ARIN
OrgAbuseName: OIT Network Services
# Email removed, Log into bgp.tools to see whois emails
```

```
OrgNOCHandle: ZU64-ARIN
OrgNOCName: Hostmaster
# Email removed, Log into bgp.tools to see whois emails
```

```
TechHandle: PG138-ARIN
TechName: Peter Gutierrez
# Email removed, Log into bgp.tools to see whois emails
```

Last Update: 2025-05-08T03:26:52Z UTC

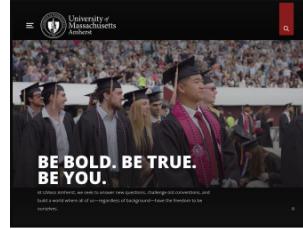
Member of the following AS-SETS

	Member	ASN Count
RADB	as10578-as-umassnet	9
LEVEL3	as-level3-origins-na	0
RADB	as-comcast-edia	2780
RADB	as198949-as-sd	3951
LEVEL3	as-as209-cust	1818
RADB	as-prole	1215
RIPE	as-aunanel	313
RIPE	as-ono-ups	316

Figure 2.16: BGP registration data for UMass ASN

University of Massachusetts - AMHERST

AS Number 1249
Website <http://umass.edu>



[Overview](#) [Prefixes](#) [Connectivity](#) [Whois](#)

Prefixes Originated
4 IPv4, 0 IPv6

Addresses Originated
322 /24's of IPv4
0 /48's of IPv6

Prefix	Description
72.19.64.0/18	University of Massachusetts - AMHERST
128.119.0.0/16	University of Massachusetts - AMHERST
192.80.83.0/24	University of Massachusetts - AMHERST
192.189.138.0/24	University of Massachusetts - AMHERST

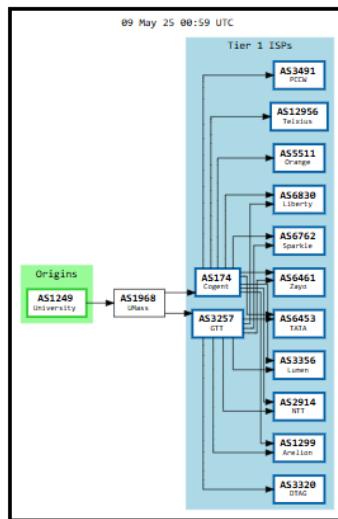
Figure 2.17: ASN prefixes originated from AS1249

<u>Peers</u>	<u>Upstreams</u>	<u>Downstreams</u>
2	1	1 (Cone: 2)

Network Policy i

[Click here to learn more about what this graph means and what makes up a network policy](#)

Global Aggregation



► Chart Display Options

[How are upstreams and downstreams calculated?](#)

Upstreams ⚡

	ASN	Description	IPv4	IPv6
🇺🇸	AS1968	UMASSNET	✓	✓

Peers ⚡

	ASN	Description	IPv4	IPv6
🇺🇸	AS1968	UMASSNET	✓	✓
🇺🇸	AS168	University of Massachusetts - AMHERST	X	✓

Downstreams ⚡

	ASN	Description	IPv4	IPv6
🇺🇸	AS168	University of Massachusetts - AMHERST	X	✓

Figure 2.18: Contact and technical details for AS1249

The screenshot shows the BGP View Countries Report interface. At the top, there is a search bar with the placeholder "Name, ASN, IP, Prefix" and a magnifying glass icon. Below the search bar, the IP address "128.119.0.0/16" is displayed next to the United States flag, with the text "UNIVERSITY OF MASSACHUSETTS - AMHERST".

Below the IP address, there are several sections:

- Announcing ASNs:** 1
- Parent Prefix:** (empty)
- Abuse:** abuse@umass.edu
- RIR:** IANA

On the left side, there is a sidebar with links: "Prefix", "Routing", and "Raw Whois". The "Raw Whois" link is currently selected.

The main content area displays detailed routing and registry metadata for the IP prefix. It includes sections for the prefix itself and its associated organization, along with various contact and administrative details.

```

# start
NetRange: 128.119.0.0 - 128.119.255.255
CIDR: 128.119.0.0/16
NetName: UMASS-NET
NetHandle: NET-128-119-0-1
Parent: NET128 (NET-128-0-0-0-0)
NetType: Direct Allocation
OriginAS:
Organization: University of Massachusetts - AMHERST (UNIVER-6)
RegDate: 1986-04-03
Updated: 2021-12-14
Ref: https://rdap.arin.net/registry/ip/128.119.0.0

OrgName: University of Massachusetts - AMHERST
OrgId: UNIVER-6
Address: University Computing Services
Address: Lederle Graduate Research Center
City: Amherst
StateProv: MA
PostalCode: 01003
Country: US
RegDate: 1986-04-03
Updated: 2024-11-25
Ref: https://rdap.arin.net/registry/entity/UNIVER-6

OrgTechHandle: ZU64-ARIN
OrgTechName: Hostmaster
OrgTechPhone: +1-413-545-0595
OrgTechEmail: hostmaster@umass.edu
OrgTechRef: https://rdap.arin.net/registry/entity/ZU64-ARIN

OrgNOCHandle: ZU64-ARIN
OrgNOCName: Hostmaster
OrgNOCPhone: +1-413-545-0595
OrgNOCEmail: hostmaster@umass.edu
OrgNOCRef: https://rdap.arin.net/registry/entity/ZU64-ARIN

OrgAbuseHandle: ONS2-ARIN
OrgAbuseName: OIT Network Services
OrgAbusePhone: +1-413-545-4809
OrgAbuseEmail: abuse@umass.edu
OrgAbuseRef: https://rdap.arin.net/registry/entity/ONS2-ARIN

RTechHandle: ZU64-ARIN
RTechName: Hostmaster
RTechPhone: +1-413-545-0595
RTechEmail: hostmaster@umass.edu
RTechRef: https://rdap.arin.net/registry/entity/ZU64-ARIN

```

Figure 2.19: Detailed routing and registry metadata

Part C – DNS Query and Response (Wireshark)

Wireshark was used to capture DNS traffic between the local machine and the DNS server. Before starting the capture, the DNS cache was cleared to make sure that a new DNS query would be sent.

The website `gaia.cs.umass.edu` was typed into the browser. A DNS query was sent from the computer to the DNS server at `192.168.1.1`. The server responded with the IP address `128.119.245.12`.

This process can be seen in the following two figures. The first figure shows the DNS query that was sent. The second figure shows the DNS response with the resolved IP address.

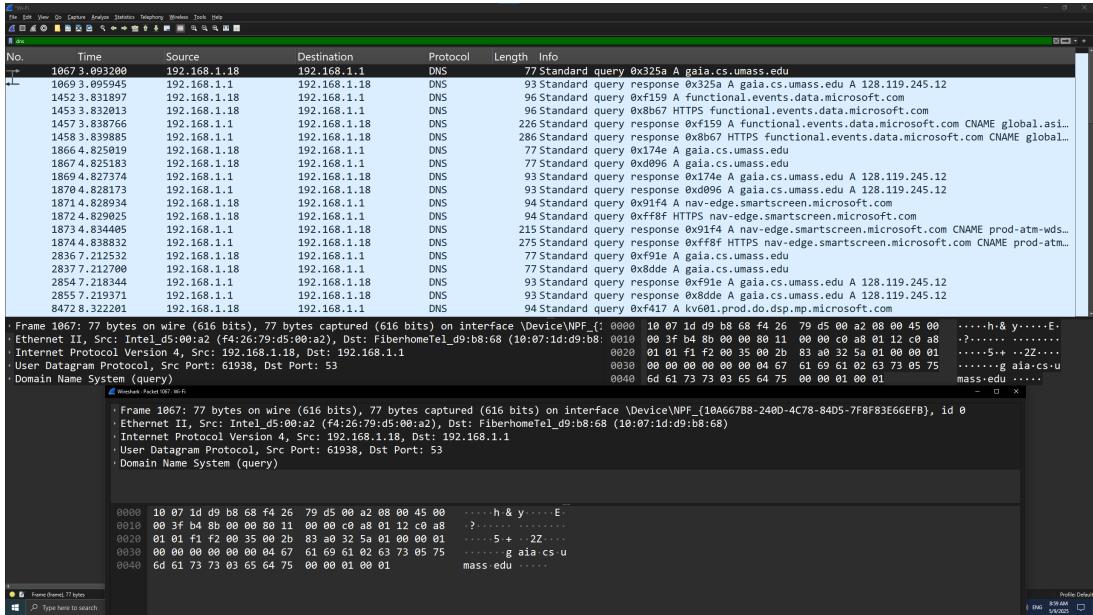


Figure 2.20: DNS query for `gaia.cs.umass.edu` sent to 192.168.1.1

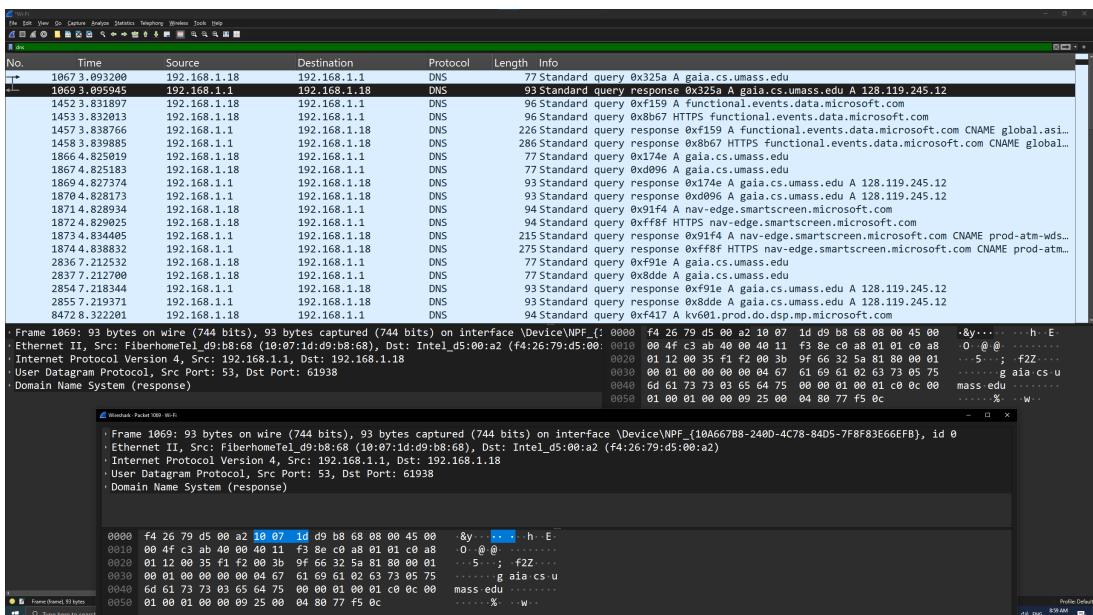


Figure 2.21: DNS response received with IP address 128.119.245.12

This capture confirmed that a DNS query and response were successfully exchanged, and the domain name was resolved properly by the local DNS server.

2.2 Task 2 – Implement a Simple Web Server Using Socket Programming

2.2.1 Task 2a – Main English Webpage (`main_en.html`)

A complete HTML and CSS webpage was created to serve as the main English page for the ENCS3320 web server. It included all the required components and was styled for clarity and visual appeal.

- The HTML page title was set as "**ENCS3320-Webserver**". Although the browser tab title was written correctly in red style within the HTML, it could not be displayed in red due to browser theme limitations. Modern browsers render tab titles using system or browser-defined colours, which cannot be overridden with HTML or CSS.
- The header text "**Welcome to ENCS3320 – Computer Networks Web-server**" was displayed in blue at the top of the page using CSS.
- Team members were displayed in boxes with photos, names, student IDs, and a short list of projects, skills, and hobbies.
- A networking topic from the textbook's first chapter was included — comparing Frequency Division Multiplexing (FDM) and Time Division Multiplexing (TDM). The section used headings, bullet points, and diagrams.
- An additional section showed the Internet Protocol Stack using an ordered list and an image.
- Three links were added at the bottom of the page:
 - Textbook site: https://gaia.cs.umass.edu/kurose_ross/index.php
 - Birzeit University: <https://www.birzeit.edu/>
 - Local page: `mySite_1232951_en.html`

Code Screenshots:

```

1  <!DOCTYPE html>
2  <html lang="en">
3
4  <head>
5      <meta charset="UTF-8" />
6      <meta name="viewport" content="width=device-width,initial-scale=1.0" />
7      <title>ENCS3320-Webserver</title>
8      <link rel="stylesheet" href="Style/Style.css" />
9  </head>
10
11 <body>
12
13     <nav class="navbar">
14         <a class="lang-switch" href="main_ar.html">عربى</a>
15         <h1 class="nav-title">Welcome to ENCS3320 - Computer Networks Webserver</h1>
16         <div class="nav-spacer"></div>
17     </nav>
18
19     <main>
20
21         <section class="team-section">
22             <h2 class="section-heading">Team Members</h2>
23
24             <div class="team-grid">
25                 <article class="team-card">
26                     <div class="avatar-wrapper">
27                         
28                     </div>
29
30                     <ul class="member-list">
31                         <li><strong>Name:</strong> Abdallah Abbasi</li>
32                         <li><strong>ID:</strong> 1232951</li>
33                         <li><strong>Projects:</strong> Shopping Cart System</li>
34                         <li><strong>Skills:</strong> HTML, CSS, JavaScript, React, Node.js, MongoDB, Networking,
35                                         Cybersecurity, Operating Systems</li>
36                         <li><strong>Hobbies:</strong> Programming, All things computer-related</li>
37                     </ul>
38                 </article>
39
40                 <article class="team-card">
41                     <div class="avatar-wrapper">
42                         
43                     </div>
44
45                     <ul class="member-list">
46                         <li><strong>Name:</strong> Jamal Shehadeh</li>
47                         <li><strong>ID:</strong> 1192316</li>
48                         <li><strong>Projects:</strong> Smart Home System for People with Disabilities, Electrical
49                                         Projects</li>
50                         <li><strong>Skills:</strong> Electrical installation, Home Assistant</li>
51

```

Figure 2.22: HTML code – Team members section

```

1           <li><strong>Hobbies:</strong> Reading, Piano</li>
2           </ul>
3       </article>
4   </div>
5 </section>
6
7 <section class="topic-section">
8     <h2 class="section-heading">
9         Frequency Division Multiplexing (FDM)&nbsp; &nbsp;vs&nbsp; &nbsp;Time Division Multiplexing (TDM)
10    </h2>
11
12   <div class="topic-grid">
13     <aside class="topic-card">
14         <h3>Frequency Division Multiplexing (FDM)</h3>
15         <ul>
16             <li>Optical / electromagnetic frequencies divided into narrow bands.</li>
17             <li>Each call allocated its own band and can transmit at the max rate of that band.</li>
18         </ul>
19
20         <h3>Time Division Multiplexing (TDM)</h3>
21         <ul>
22             <li>Transmission time divided into slots.</li>
23             <li>Each call allocated periodic slot(s); it can use the full bandwidth only during its slot(s).</li>
24         </ul>
25     </aside>
26
27     <aside class="topic-card topic-image">
28         
29     </aside>
30   </div>
31 </section>
32
33
34
35 <section class="topic-section">
36     <h2 class="section-heading">Internet Protocol Stack</h2>
37
38     <div class="topic-grid">
39     <aside class="topic-card">
40         <ol>
41             <li><strong>application:</strong> supporting network applications<br>IMAP, SMTP, HTTP</li>
42             <li><strong>transport:</strong> process-process data transfer<br>TCP, UDP</li>
43             <li><strong>network:</strong> routing of datagrams from source to destination<br>IP, routing
44                 protocols</li>
45             <li><strong>link:</strong> data transfer between neighboring&nbsp;network elements<br>Ethernet,
46                 802.11 (WiFi), PPP</li>
47             <li><strong>physical:</strong> bits "on the wire"</li>
48         </ol>
49     </aside>
50
51

```

Figure 2.23: HTML code – FDM vs TDM topic section

```

1           <aside class="topic-card topic-image">
2             
3           </aside>
4         </div>
5       </section>
6     </main>
7
8   <footer class="page-footer">
9     <p>
10       External resources:
11       <a href="https://gaia.cs.umass.edu/kurose_ross/index.php" target="_blank">Textbook site</a> |
12       <a href="https://www.birzeit.edu/" target="_blank">Birzeit University</a> |
13       <a href="mySite_1232951_en.html">Local page (mySite_1232951_en.html)</a>
14     </p>
15   </footer>
16
17 </body>
18
19 </html>

```

Figure 2.24: HTML code – Protocol stack section and external links

Rendered Page:

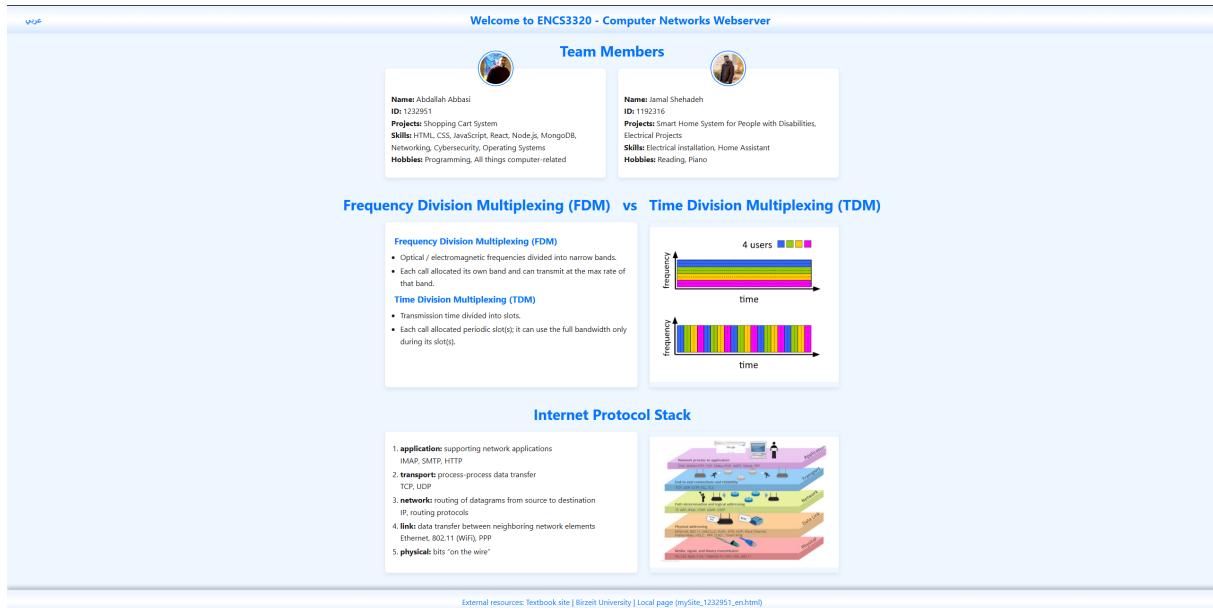


Figure 2.25: Final rendered output of main_en.html in browser

2.2.2 Task 2b – Local Webpage (mySite_1232951_en.html)

A secondary local webpage was created to allow users to request files related to the topic presented in the main page. The user was asked to enter the exact name of an image or video file into an input field.

The form used the GET method and submitted the input as a URL query. When the form was submitted, the file was searched for in the server's local directory.

If the requested file was not found, the server responded with a **307 Temporary Redirect**. The redirection was based on the file extension:

- If the file ended with .png, .jpg, or other image formats, the client was redirected to a Google Image search using the provided file name as a keyword.
- If the file ended with .mp4 or other video formats, the client was redirected to a Google Video search.

The user was also shown a message indicating that if the file was not available locally, a redirection would occur.

Form Page HTML Code:



```

1 <!DOCTYPE html>
2 <html lang="en">
3
4 <head>
5   <meta charset="utf-8" />
6   <meta name="viewport" content="width=device-width,initial-scale=1.0" />
7   <title>Media Finder - ENCS3320</title>
8   <link rel="stylesheet" href="Style/Style.css">
9   <link rel="stylesheet" href="Style/mySite.css">
10 </head>
11
12 <body>
13
14
15   <nav class="navbar">
16     <a class="lang-switch" href="mySite_1232951_ar.html">عربي</a>
17     <h1 class="nav-title">Request an Image / Video</h1>
18     <div class="nav-spacer"></div>
19   </nav>
20
21 <main>
22
23
24   <section class="team-section" style="padding:2rem 0">
25     <div class="team-grid" style="grid-template-columns:minmax(280px,400px)">
26       <article class="team-card" style="padding:2rem 1rem">
27         <p style="margin-bottom:1rem">
28           Type the <strong>exact file name</strong> you want
29           (e.g. &nbsp;<em>os1_layers.png</em> or <em>intro.mp4</em>), then hit <b>Fetch</b>.
30           If the file isn't found locally, the server will redirect you to
31           a Google Images / Videos search for it.
32         </p>
33
34         <form action="/getImage" method="get" class="member-list" style="font-size:1rem">
35           <label for="imageName">File Name:</label><br>
36           <input id="imageName" name="imageName" type="text" required
37             style="width:100%;padding:.5rem;border:1px solid #ccc;border-radius:4px;margin:.5rem 0 1rem">
38           <button type="submit"
39             style="padding:.5rem 1.25rem;border:none;border-radius:4px;background:var(--accent);color:#fff;cursor:pointer">
40             Fetch
41           </button>
42         </form>
43       </article>
44     </div>
45   </section>
46
47 </main>
48
49
50   <footer class="page-footer">
51     <p>
52       <a href="main_en.html">Back to Home</a>
53     </p>
54   </footer>
55
56 </body>
57
58 </html>

```

Figure 2.26: HTML code of mySite_1232951_en.html – file request form

Rendered Page:

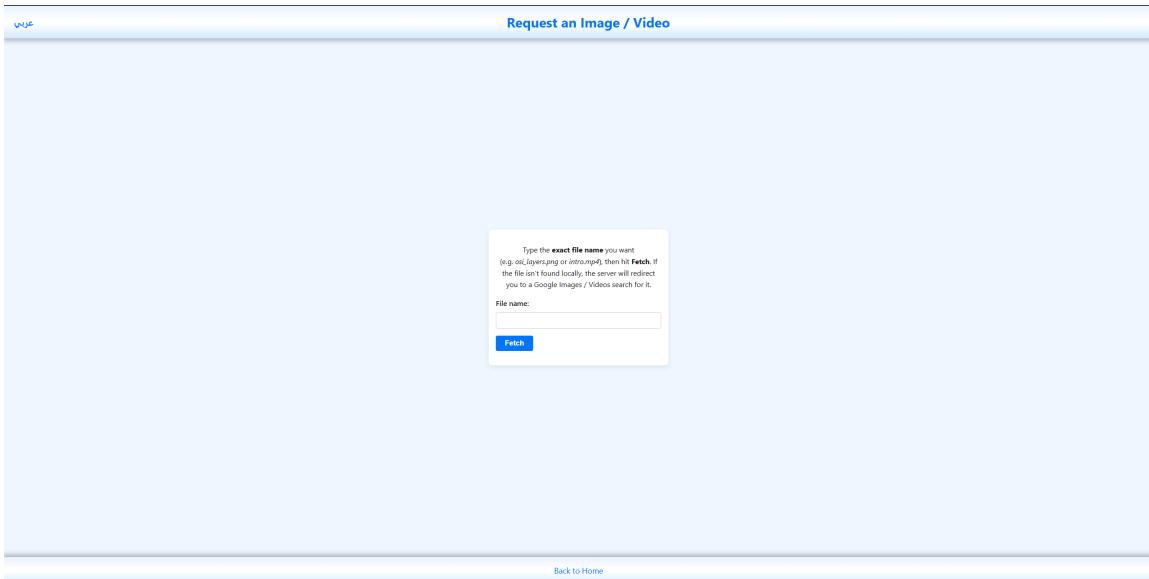


Figure 2.27: Rendered view of local file request page

The form was fully functional, and the redirection was correctly handled by the server logic. This helped simulate real-world search fallback behaviour for missing media content.

2.2.3 Task 2c – Arabic Versions (`main_ar.html` and `mySite_1232951_ar.html`)

Arabic versions of the main and local HTML pages were created. These pages were named `main_ar.html` and `mySite_1232951_ar.html`.

Both pages used Arabic text and followed a right-to-left (`rtl`) layout. The visual design, content, and structure were kept similar to the English versions for consistency.

- The direction was set using `<html lang="ar" dir="rtl">`.
- All text content was translated into Arabic, including headings, labels, button text, and descriptions.
- External links to the textbook site and Birzeit University were included.
- A language switch link allowed users to go back to the English version.

`main_ar.html:`

- Contained the same sections as the English version (team members, FDM vs TDM, Internet protocol stack).
- CSS styling and layout were preserved using mirrored flex/grid alignment.

`mySite_1232951_ar.html:`

- Allowed Arabic-speaking users to input a file name and request it.
- The same 307 redirect behaviour was implemented if the file was not found locally.

Code Screenshots:



```
1  <!DOCTYPE html>
2  <html lang="ar" dir="rtl">
3
4  <head>
5      <meta charset="utf-8" />
6      <meta name="viewport" content="width=device-width,initial-scale=1.0" />
7      <title>ENCS3320-خادم الويب</title>
8
9      <link rel="stylesheet" href="Style/Style.css" />
10     <link rel="stylesheet" href="Style/rtl.css" />
11 </head>
12
13 <body>
14
15
16     <nav class="navbar">
17         <a class="lang-switch" href="main_en.html">English</a>
18         <h1 class="nav-title">مرحباً بكم في ENCS3320 - خادم شبكات الحاسوب</h1>
19         <div class="nav-spacer"></div>
20     </nav>
21
22     <main>
23
24
25         <section class="team-section">
26             <h2 class="section-heading">أعضاء الفريق</h2>
27
28             <div class="team-grid">
29                 <article class="team-card">
30                     <div class="avatar-wrapper">
31                         
32                     </div>
33                     <ul class="member-list">
34                         <li><strong>الاسم:</strong> عبد الله</li>
35                         <li><strong>الرقم:</strong> 1232951</li>
36                         <li><strong>نظام سلة تسوق:</strong> المفاصيل</li>
37                         <li><strong>المهارات:</strong> HTML, CSS, JavaScript, React, Node.js, MongoDB,</li>
38                         <li><strong>أنظمة التشغيل:</strong> Linux</li>
39                         <li><strong>الهوايات:</strong> البرمجة، كل ما يتعلق بالحواسوب</li>
40                     </ul>
41                 </article>
42
43                 <article class="team-card">
44                     <div class="avatar-wrapper">
45                         
46                     </div>
47                     <ul class="member-list">
48                         <li><strong>الاسم:</strong> جمال</li>
49                         <li><strong>الرقم:</strong> 1192316</li>
50                         <li><strong>نظام:</strong> Linux</li>
51                     </ul>
52                 </article>
53             </div>
54         </section>
55     </main>
56
57     <script src="https://code.jquery.com/jquery-3.5.1.min.js"></script>
58     <script src="Style/rtl.js"></script>
59
60 </body>
```

Figure 2.28: HTML code – Arabic main page (team member section)

```

1          <li><strong> ، التمديدات الكهربائية</strong></li>
2          <li><strong> القراءة، العزف على البيانو</strong></li>
3      </ul>
4  </article>
5 </div>
6 </section>
7
8
9 <section class="topic-section">
10 <h2 class="section-heading">
11     تقسيم الزمن (FDM) مقابل (TDM)
12 </h2>
13
14 <div class="topic-grid">
15     <aside class="topic-card">
16         <h3>(FDM) تقسيم التردد</h3>
17         <ul>
18             <li> يُقسم الترددات الكهرومغناطيسية إلى نطاقات ضيقة</li>
19             <li> يُخضّم لكل اتصال نطاق مستقل ويمكنه الإرسال بأقصى معدل لتلك النطاق</li>
20         </ul>
21
22         <h3>(TDM) تقسيم الزمن</h3>
23         <ul>
24             <li> يُقسم الزمن إلى خانات زمنية</li>
25             <li> يُخضّم لكل اتصال خانة دورية ويمكنه استخدام كامل النطاق فقط خلالها</li>
26         </ul>
27     </aside>
28
29     <aside class="topic-card topic-image">
30         
31     </aside>
32 </div>
33 </section>
34
35
36 <section class="topic-section">
37     <h2 class="section-heading">طبقات بروتوكول الإنترنت</h2>
38
39 <div class="topic-grid">
40     <aside class="topic-card">
41         <ol>
42             <li><strong>Application:</strong> تطبيقات الشبكة<br>IMAP، SMTP، HTTP</li>
43             <li><strong>Transport:</strong> نقل بيانات بين العمليات<br>TCP، UDP</li>
44             <li><strong>Network:</strong> توجيه العزم بين المصدر والوجهة<br>IP</li>
45             <li><strong>Link:</strong> نقل البيانات بين عناصر الشبكة المترابطة<br>Ethernet، Wi-Fi، PPP</li>
46             <li><strong>Physical:</strong> نبضات على المدى</li>
47         </ol>
48     </aside>
49
50     <aside class="topic-card topic-image">
51

```

Figure 2.29: HTML code – Arabic main page (topic and image sections)

```
● ● ●
1      
2      </aside>
3      </div>
4      </section>
5
6  </main>
7
8
9  <footer class="page-footer">
10     <p>
11         مصادر خارجية:
12         <a href="https://gaia.cs.umass.edu/kurose_ross/index.php" target="_blank">موقع الكتاب</a> |
13         <a href="https://www.birzeit.edu/" target="_blank">جامعة بيرزيت</a> |
14         <a href="mySite_1232951_ar.html" target="_blank">صفحة محلية (mySite_1232951_ar.html)</a>
15     </p>
16  </footer>
17
18 </body>
19
20 </html>
```

Figure 2.30: HTML code – Arabic main page footer

```
1 <!DOCTYPE html>
2 <html lang="ar" dir="rtl">
3
4 <head>
5   <meta charset="utf-8" />
6   <meta name="viewport" content="width=device-width,initial-scale=1.0" />
7   <title>مدونة البوساطة - ENCS3320</title>
8
9
10  <link rel="stylesheet" href="Style/Style.css">
11  <link rel="stylesheet" href="Style/mySite.css">
12 </head>
13
14 <body>
15
16
17  <nav class="navbar" style="direction:rtl">
18    <a class="lang-switch" href="mySite_1232951_en.html">English</a>
19    <h1 class="nav-title">طلب مورة / بيدرو</h1>
20    <div class="nav-spacer"></div>
21  </nav>
22
23  <main>
24
25
26  <section class="team-section" style="padding:2rem 0">
27    <div class="team-grid" style="grid-template-columns:minmax(280px,400px);direction:ltr">
28      <article class="team-card" style="padding:2rem 1rem;direction:rtl">
29        <p style="margin-bottom:1rem">
30          اكتب <strong>اسم العلان بالخط العريض</strong>
31          <em>mosi_layers.png</em> أو <em>intro.mp4</em> (فقط <b>يمكن</b>).
32          إذا لم يعثر على الملف مكتوباً، سعيد الخامد توجهك إلى
33          بحث مورة أو فيديو في &nbsp;Google.
34        </p>
35
36        <form action="/getImage" method="get" class="member-list" style="font-size:1rem">
37          <label for="imageName">قم &nbsp;بالملف</label><br>
38          <input id="imageName" name="imageName" type="text" required
39            style="direction:ltr;padding:.5rem;border:1px solid #ccc; border-radius:4px; margin:.5rem 0 1rem">
40          <button type="submit"
41            style="padding:.5rem 1.25rem; border:none; border-radius:4px; background:var(--accent); color:#fff; cursor:pointer">
42            جلب
43          </button>
44        </form>
45      </div>
46    </section>
47
48  </main>
49
50  <footer class="page-footer" style="text-align:center">
51    <p>
52      <a href="main_ar.html">العودة إلى الصفحة الرئيسية</a>
53    </p>
54  </footer>
55
56 </body>
57
58 </html>
```

Figure 2.31: HTML code – Arabic local request page (mySite_1232951_ar.html)

Rendered Pages:

English

مرحبا بك في ENCS3320 - خادم شبكات الحاسوب

اعضاء الفريق

الاسم: عبد الله
الايميل: 112232951_ar@stu.edu.sa
المجال: تطوير برمجي، تطوير بلغة البرمجة ابستانت لدوى الاتصالات
الايميل: 112232951_ar@stu.edu.sa
الهوايات: الذكاء الصناعي، Home Assistant، مشاريع كهربائية، الميكانيك، الالكترونيات

الاسم: سارة
الايميل: 1232951_ar@stu.edu.sa
المجال: تطوير برمجي، ملء نسخة
المهارات: HTML, CSS, JavaScript, React, Node.js, MongoDB
الهوايات: اذكياء اصطناعي، مطبخ الشيف

تقسيم التردد (FDM) مقابل تقسيم الزمن (TDM)

(FDM)

- تقسيم التردد هو تقسيم التردد على نطاقات متحدة.
- يحصل كل اتصال على نطاق متسلق ويسكنه الارسال بأقصى معدل لذلك النطاق.

(TDM)

- تقسيم الزمن إلى ثبات و زمنية.
- يحصل كل اتصال حادة دورية ويسمه استخدام كامل النطاق فقط جلبه.

طبقات بروتوكول الانترنت

طبقات بروتوكول الانترنت

Application .1	HTTP, SMTP, IMAP
Transport .2	TCP, UDP, ICMP
Network .3	روابط بروتوكولات IP
Link .4	PPP, Wi-Fi, Ethernet
Physical .5	الDevices على الشبك

مصدر: مراجع: موقع الكتاب | ملخص دروس | حصص ملخص [mySite_1232951_ar.html]

Figure 2.32: Rendered view of Arabic main page (`main_ar.html`)

English

طلب صورة / فيديو

أكتب اسم الملف بالمعنى التالي (ex: layer.png) أو
أدخل اسم الملف بالمعنى التالي (intro.mp4)
سيتم إنشاء وتحديث الملف بمحتوى الملف المدخل
في Google Drive

اسم الملف:

إضافة

العودة إلى الصفحة الرئيسية

Figure 2.33: Rendered view of Arabic local page (`mySite_1232951_ar.html`)

CSS Styling

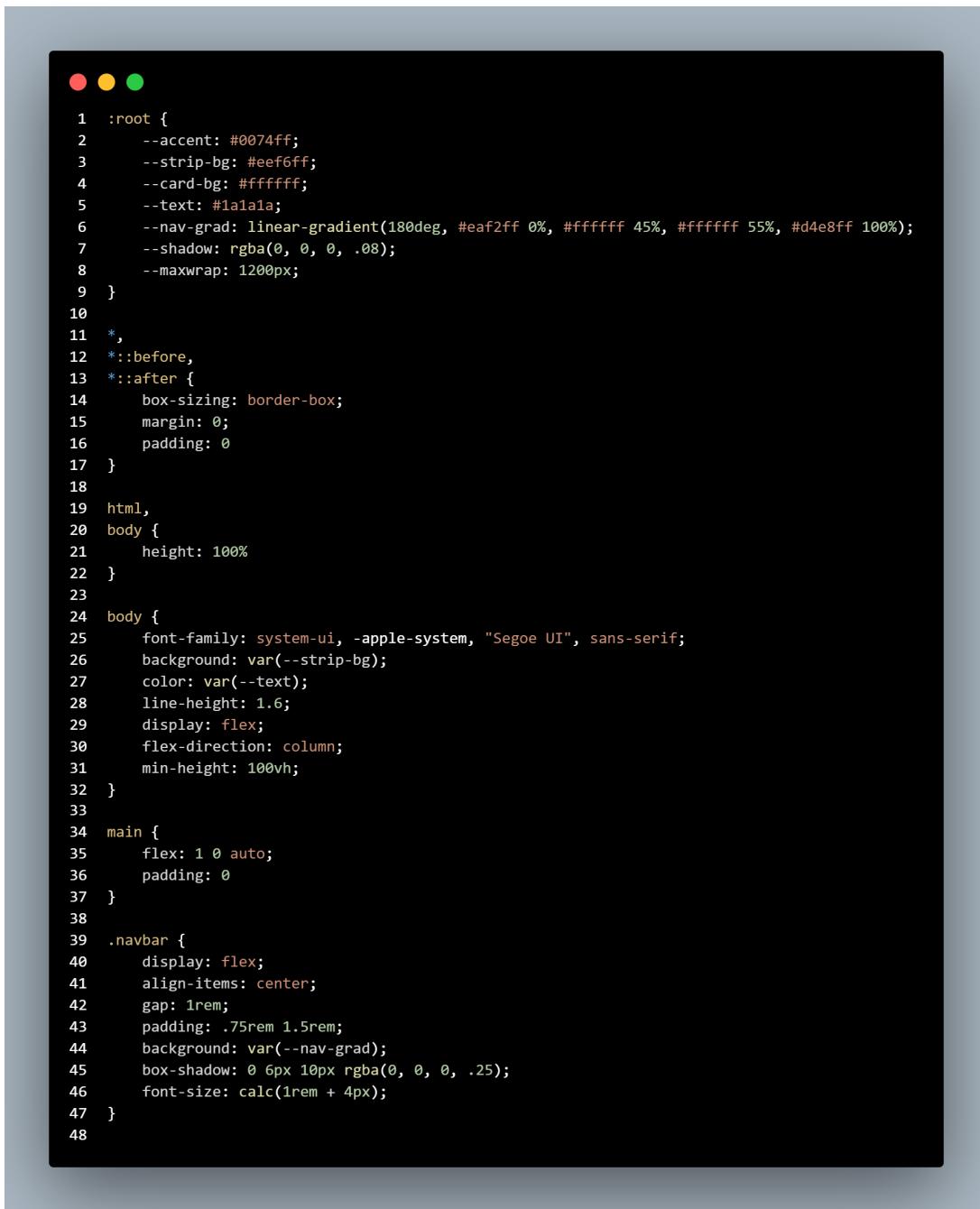
To improve the appearance and readability of both English and Arabic pages, external CSS files were created. These files defined consistent fonts, colours, spacing, shadows, and layout rules.

Two stylesheets were used:

- `Style.css` – For general layout and design.
- `rtl.css` – For Arabic RTL (right-to-left) adjustments.

The CSS styling included:

- Custom variables using `:root` to control colours, shadows, max width, and spacing.
- Flexbox and CSS Grid used for layout control.
- Responsive design using `@media` queries.
- Separate styles applied to navbar, footer, team cards, topic sections, and image containers.
- RTL-specific CSS ensured Arabic text alignment and directionality were properly displayed.



```
1  :root {
2    --accent: #0074ff;
3    --strip-bg: #eef6ff;
4    --card-bg: #ffffff;
5    --text: #1a1a1a;
6    --nav-grad: linear-gradient(180deg, #eaf2ff 0%, #ffffff 45%, #ffffff 55%, #d4e8ff 100%);
7    --shadow: rgba(0, 0, 0, .08);
8    --maxwrap: 1200px;
9  }
10 *
11 *::before,
12 *::after {
13   box-sizing: border-box;
14   margin: 0;
15   padding: 0
16 }
17
18 html,
19 body {
20   height: 100%
21 }
22
23 body {
24   font-family: system-ui, -apple-system, "Segoe UI", sans-serif;
25   background: var(--strip-bg);
26   color: var(--text);
27   line-height: 1.6;
28   display: flex;
29   flex-direction: column;
30   min-height: 100vh;
31 }
32
33 main {
34   flex: 1 0 auto;
35   padding: 0
36 }
37
38 .navbar {
39   display: flex;
40   align-items: center;
41   gap: 1rem;
42   padding: .75rem 1.5rem;
43   background: var(--nav-grad);
44   box-shadow: 0 6px 10px rgba(0, 0, 0, .25);
45   font-size: calc(1rem + 4px);
46 }
47
48
```

Figure 2.34: CSS base variables and body styling (`Style.css`)

```
1  @media(min-width:768px) {  
2      .navbar {  
3          padding: .75rem 3rem  
4      }  
5  }  
6  
7  .lang-switch {  
8      color: var(--accent);  
9      font-weight: 600;  
10     text-decoration: none  
11  }  
12  
13 .lang-switch:hover {  
14     color: #3398ff  
15  }  
16  
17 .nav-title {  
18     flex: 1;  
19     text-align: center;  
20     font-weight: 700;  
21     color: var(--accent)  
22  }  
23  
24 .team-section,  
25 .topic-section {  
26     background: var(--strip-bg);  
27     padding: 1rem 0 1.5rem;  
28     margin: 0;  
29  }  
30  
31 .section-heading {  
32     text-align: center;  
33     margin-bottom: 1rem;  
34     font-size: clamp(1.5rem, 3vw + .5rem, 2.4rem);  
35     color: var(--accent);  
36  }  
37  
38 .team-grid {  
39     display: grid;  
40     gap: 2rem;  
41     grid-template-columns: repeat(auto-fit, minmax(260px, 1fr));  
42     max-width: var(--maxwrap);  
43     margin: 0 auto;  
44  }  
45  
46
```

Figure 2.35: Navigation bar, team sections, and headings

```
● ○ ● ●  
1 .team-card {  
2     position: relative;  
3     text-align: center;  
4     background: var(--card-bg);  
5     padding: 3.5rem 1rem 2rem;  
6     border-radius: 8px;  
7     box-shadow: 0 4px 12px var(--shadow);  
8 }  
10  
11 .avatar-wrapper {  
12     position: absolute;  
13     top: -45px;  
14     left: 50%;  
15     transform: translateX(-50%);  
16     width: 90px;  
17     height: 90px;  
18     border-radius: 50%;  
19     overflow: hidden;  
20     border: 4px solid var(--card-bg);  
21     box-shadow: 0 0 0 2px var(--accent);  
22 }  
23  
24 .avatar-wrapper img {  
25     width: 100%;  
26     height: 100%;  
27     object-fit: cover  
28 }  
29  
30 .member-list {  
31     list-style: none;  
32     font-size: 1.25rem;  
33     text-align: left;  
34     margin-top: .5rem  
35 }  
36  
37 .topic-grid {  
38     display: grid;  
39     gap: 2rem;  
40     grid-template-columns: 4fr 3fr;  
41     align-items: stretch;  
42     max-width: var(--maxwrap);  
43     margin: 0 auto;  
44 }  
45  
46 @media(max-width:700px) {  
47     .topic-grid {  
48         grid-template-columns: 1fr  
49     }  
50 }  
51  
52 .topic-card {  
53     background: var(--card-bg);  
54     padding: 1.5rem;  
55     border-radius: 8px;  
56     box-shadow: 0 4px 12px var(--shadow);  
57     position: relative;  
58 }  
59
```

Figure 2.36: Team card design, avatar styling, and responsive layout

```
1 .topic-card h3 {  
2     color: var(--accent);  
3     margin: .5rem 0  
4 }  
5  
6 .topic-card ul,  
7 .topic-card ol {  
8     margin-left: 1rem;  
9     margin-bottom: .5rem  
10 }  
11  
12 .topic-card li {  
13     margin: .25rem 0  
14 }  
15  
16 .topic-card:not(.topic-image) {  
17     font-size: calc(1rem + 4px)  
18 }  
19  
20 .topic-image img {  
21     width: 100%;  
22     display: block;  
23     border-radius: 4px  
24 }  
25  
26 .topic-image::before,  
27 .topic-image::after {  
28     content: '';  
29     position: absolute;  
30     left: 0;  
31     width: 100%;  
32     height: 14px;  
33     background: var(--strip-bg);  
34     pointer-events: none;  
35 }  
36  
37 .topic-image::before {  
38     top: 0  
39 }  
40  
41 .topic-image::after {  
42     bottom: 0  
43 }  
44
```

Figure 2.37: Topic section layout and styling

```
1 .page-footer {  
2     background: var(--nav-grad);  
3     box-shadow: 0 -6px 10px rgba(0, 0, 0, .25);  
4     padding: 1rem 1.5rem;  
5     font-size: calc(1rem + 4px);  
6     text-align: center;  
7     color: var(--accent);  
8     flex-shrink: 0;  
9 }  
10  
11 .page-footer a {  
12     color: var(--accent);  
13     text-decoration: none  
14 }  
15  
16 .page-footer a:hover {  
17     color: #3398ff  
18 }  
19  
20 html,  
21 body,  
22 main,  
23 section {  
24     margin: 0  
25 }  
26  
27 @media (max-width:480px) {  
28  
29     .nav-title {  
30         font-size: 1.1rem;  
31         line-height: 1.3  
32     }  
33  
34     .section-heading {  
35         font-size: 1.25rem  
36     }  
37 }
```

Figure 2.38: Topic content spacing, image layout, and hover styles

```
1      .avatar-wrapper {  
2          top: -35px;  
3          width: 70px;  
4          height: 70px  
5      }  
6  
7      .team-card {  
8          padding: 3rem .75rem 1.5rem  
9      }  
10  
11     .section-heading {  
12         padding: 2rem;  
13     }  
14 }
```

Figure 2.39: Footer styles and small screen adjustments

```
1  input[type="text"]{  
2    width:100%;padding:.6rem;  
3    border:1px solid #ccc;border-radius:4px;  
4    margin:.5rem 0 1.25rem;font-size:1rem;  
5  }  
6  button{  
7    padding:.6rem 1.5rem;border:none;border-radius:4px;  
8    background:var(--accent);color:#fff;  
9    font-size:1rem;font-weight:600;cursor:pointer;  
10 }  
11 button:hover{background:#3398ff}  
12  
13 /* footer */  
14 .page-footer{  
15   background:var(--nav-grad);  
16   box-shadow:0 -6px 10px rgba(0,0,0,.25);  
17   padding:1rem 1.5rem;  
18   font-size:1.1rem;  
19   text-align:center;  
20   color:var(--accent);  
21 }  
22 .page-footer a{color:var(--accent);text-decoration:none}  
23 .page-footer a:hover{color:#3398ff}  
24  
25 /* small-screen tweak */  
26 @media(max-width:480px){  
27   .form-card{padding:1.5rem 1rem}  
28 }  
29
```

Figure 2.40: General layout and team member section CSS



```
1  :root{  
2    --accent:#0074ff;  
3    --strip-bg:#eef6ff;  
4    --card-bg:#ffffff;  
5    --text:#1a1a1a;  
6    --shadow:rgba(0,0,0,.08);  
7    --nav-grad:linear-gradient(180deg,#eaf2ff 0%,#ffffff 45%,#ffffff 55%,#d4e8ff 100%);  
8  }  
9  
10 /* reset + base */  
11 *,*::before,*::after{box-sizing:border-box;margin:0;padding:0}  
12 body{  
13   font-family:system-ui,-apple-system,"Segoe UI",sans-serif;  
14   background:var(--strip-bg);  
15   color:var(--text);  
16   line-height:1.6;  
17   display:flex;flex-direction:column;min-height:100vh;  
18 }  
19  
20 /* navbar */  
21 .navbar{  
22   display:flex;align-items:center;gap:1rem;  
23   padding:.75rem 1.5rem;  
24   background:var(--nav-grad);  
25   box-shadow:0 6px 10px rgba(0,0,0,.25);  
26   font-size:1.25rem;  
27 }  
28 .nav-title{flex:1;text-align:center;font-weight:700;color:var(--accent)}  
29 .lang-switch{color:var(--accent);font-weight:600;text-decoration:none}  
30 .lang-switch:hover{color:#3398ff}  
31  
32 /* main wrapper */  
33 main{  
34   flex:1 0 auto;  
35   display:flex;  
36   align-items:center;  
37   justify-content:center;  
38   padding:2rem 1rem;  
39 }  
40  
41 /* form card */  
42 .form-card{  
43   background:var(--card-bg);  
44   max-width:420px;width:100%;  
45   padding:2rem 1.5rem;  
46   border-radius:8px;  
47   box-shadow:0 4px 12px var(--shadow);  
48 }
```

Figure 2.41: Form design and padding for mobile screens



```
1 html[dir="rtl"] body {
2     direction: rtl;
3     text-align: right
4 }
5
6
7 html[dir="rtl"] .navbar {
8     direction: rtl
9 }
10
11 html[dir="rtl"] .lang-switch {
12     direction: rtl
13 }
14
15 html[dir="rtl"] .nav-title {
16     text-align: center
17 }
18
19 html[dir="rtl"] .team-grid {
20     direction: rtl
21 }
22
23 html[dir="rtl"] .member-list {
24     direction: rtl;
25     padding-right: 1rem;
26 }
27
28
29 html[dir="rtl"] ul,
30 html[dir="rtl"] ol {
31     direction: rtl;
32     list-style-position: inside;
33     padding-right: 1.2rem;
34     margin-right: 0;
35 }
36
37 html[dir="rtl"] li {
38     text-align: right
39 }
40
41
42 html[dir="rtl"] .topic-grid {
43     direction: ltr
44 }
45
46
47 html[dir="rtl"] .topic-card {
48     direction: rtl
49 }
50
51 html[dir="rtl"] .topic-card:not(.topic-image) {
52     text-align: right
53 }
54
55 html[dir="rtl"] .page-footer {
56     text-align: center
57 }
```

Figure 2.42: Right-to-left (RTL) layout rules for Arabic pages (`rtl.css`)

2.3 Task 2 (d) – Server Functionality

The server was implemented in Python using the `socket` module, listening on port 9991 (derived from student ID). Upon receiving an HTTP request, it performs the following steps:

- Parses and logs the full HTTP request (method, path, headers) on the terminal.
- Identifies the file extension to set the appropriate `Content-Type` (e.g., `text/html`, `image/png`, `text/css`, `video/mp4`).
- Returns a valid HTTP response using status codes such as `200 OK`, `307 Temporary Redirect`, or `404 Not Found`.
- Logs the response code and both the client's and server's socket addresses.

Figure 2.43 shows the part of the code responsible for setting up the server and handling 404 responses. **Figure 2.44** illustrates the request parsing, default routing, and 307 redirection to external sites. **Figure 2.45** displays image/video redirect handling and MIME type assignment.

```

● ○ ●

1  from socket import gethostbyname, socket, AF_INET, SOCK_STREAM, gethostname
2
3  mainAddress = (gethostbyname(gethostname()), 9991)
4
5  serverSocket = socket(AF_INET, SOCK_STREAM)
6  serverSocket.bind(mainAddress)
7  serverSocket.listen()
8
9
10 print("The server is up!")
11 print(f"Listening to http://{mainAddress[0]}:{mainAddress[1]}")
12
13 def notFoundHtmlText(file, addr):
14     return f"""
15         <html>
16             <head><title>Error 404</title></head>
17             <body>
18                 <h1 style='color: red;'>The file "{file}" is not found</h1>
19                 <p><b>1192316 - Jamal Shehadeh<br />1232951 - Abdallah Ababsi</b></p>
20                 <p>Client IP: {addr[0]}</p>
21                 <p>Client Port: {addr[1]}</p>
22             </body>
23         </html>
24     """
25
26 def HttpResponse(code, status, type, body):
27     return (f"HTTP/1.1 {code} {status}\r\n"
28            f"Content-Type: {type}; charset=utf-8\r\n"
29            f"Content-Length: {len(body)}\r\n\r\n").encode() + body
30
31 def readFile(filePath, connectionSocket, addr):
32     try:
33         file = open(f"static/{filePath}", "rb")
34         return file.read()
35     except OSError:
36         ext = filePath.lower().split('.')[-1]
37         if ext in ('png', 'jpg', 'jpeg', 'gif'):
38             url = f"https://www.google.com/search?q={filePath}&tbo=isch"
39             connectionSocket.send(
40                 f"HTTP/1.1 307 Temporary Redirect\r\nLocation: {url}\r\n\r\n".encode())
41         )
42         print("→ 307 Temporary Redirect")
43     elif ext in ('mp4', 'webm'):
44         url = f"https://www.google.com/search?q={filePath}&tbo=vid"
45         connectionSocket.send(
46             f"HTTP/1.1 307 Temporary Redirect\r\nLocation: {url}\r\n\r\n".encode())
47         )
48         print("→ 307 Temporary Redirect")
49     else:
50

```

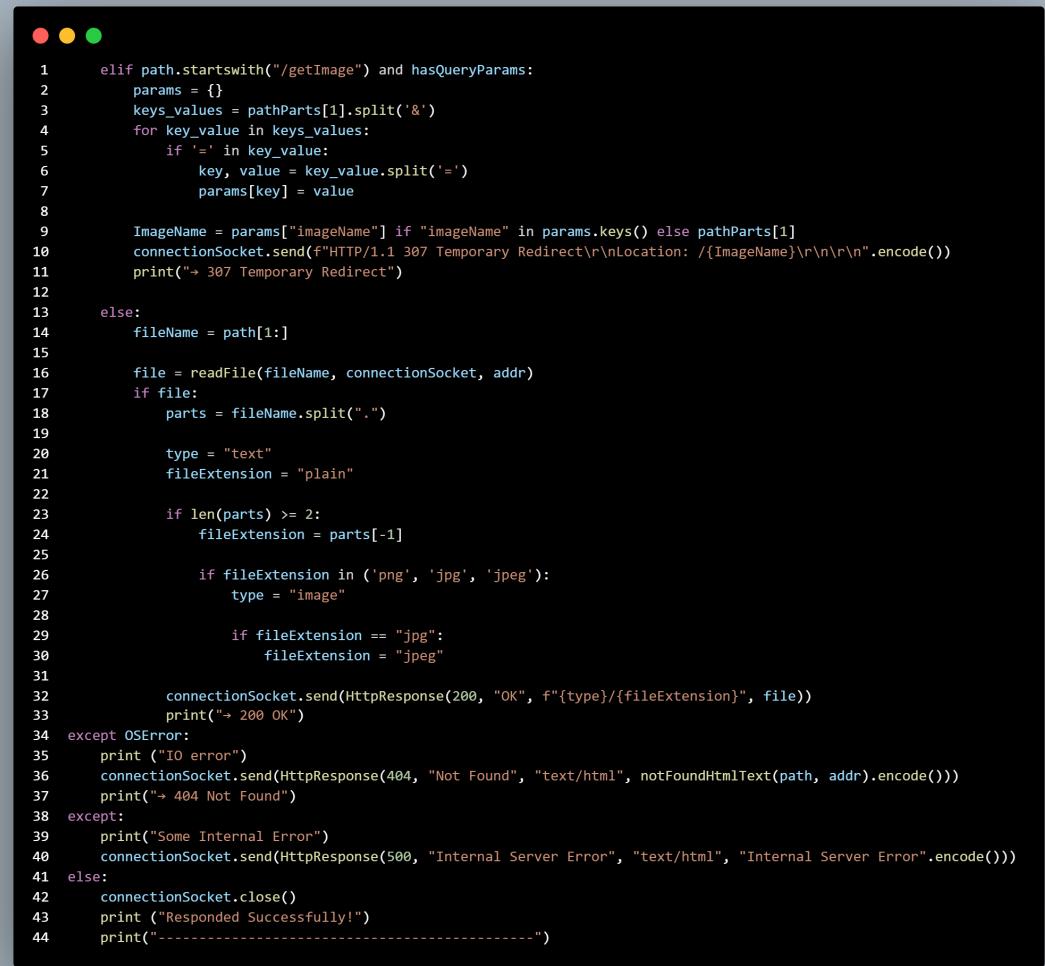
Figure 2.43: Server setup and 404 response generation

```

1             HttpResponse(404, "Not Found", "text/html",
2                 notFoundHtmlText(filePath, addr).encode())
3         )
4         print("→ 404 Not Found")
5     return
6
7 while True:
8     try:
9         connectionSocket, addr = serverSocket.accept()
10    request = connectionSocket.recv(2048).decode()
11
12    if not request or len(request) == 0:
13        print("Heartbeating....")
14        continue
15
16    print("-----")
17    print("Request From " + str(addr))
18    print(request)
19
20    lines = request.split("\r\n")
21
22    requestLine = lines[0].strip()
23    method, fullPath, version = requestLine.split(" ")
24
25    pathParts = fullPath.split("?")
26    hasQueryParams = len(pathParts) != 1
27    path = pathParts[0] if hasQueryParams else fullPath
28
29    path = path[:-1] if path.endswith("/") and len(path) != 1 else path
30
31    if path == '/' or path.startswith('/index.html', '/en'):
32        htmlText = readFile("main_en.html", connectionSocket, addr)
33        if htmlText:
34            connectionSocket.send(HttpResponse(200, "OK", "text/html", htmlText))
35            print("→ 200 OK")
36
37    elif path.startswith('/ar'):
38        htmlText = readFile("main_ar.html", connectionSocket, addr)
39        if htmlText:
40            connectionSocket.send(HttpResponse(200, "OK", "text/html", htmlText))
41            print("→ 200 OK")
42
43    elif path.startswith('/so'):
44        connectionSocket.send("HTTP/1.1 307 Temporary Redirect\r\nLocation: https://stackoverflow.com\r\n\r\n".encode())
45        print("→ 307 Temporary Redirect")
46
47    elif path.startswith('/itc'):
48        connectionSocket.send("HTTP/1.1 307 Temporary Redirect\r\nLocation: https://itc.birzeit.edu\r\n\r\n".encode())
49        print("→ 307 Temporary Redirect")
50
51

```

Figure 2.44: Routing logic and redirection handling



```

1     elif path.startswith("/getImage") and hasQueryParams:
2         params = {}
3         keys_values = pathParts[1].split('&')
4         for key_value in keys_values:
5             if '=' in key_value:
6                 key, value = key_value.split('=')
7                 params[key] = value
8
9         ImageName = params["imageName"] if "imageName" in params.keys() else pathParts[1]
10        connectionSocket.send(f"HTTP/1.1 307 Temporary Redirect\r\nLocation: /{ImageName}\r\n\r\n".encode())
11        print("→ 307 Temporary Redirect")
12
13    else:
14        fileName = path[1:]
15
16        file = readFile(fileName, connectionSocket, addr)
17        if file:
18            parts = fileName.split(".")
19
20            type = "text"
21            fileExtension = "plain"
22
23            if len(parts) >= 2:
24                fileExtension = parts[-1]
25
26            if fileExtension in ('png', 'jpg', 'jpeg'):
27                type = "image"
28
29            if fileExtension == "jpg":
30                fileExtension = "jpeg"
31
32            connectionSocket.send(HttpResponse(200, "OK", f"{type}/{fileExtension}", file))
33            print("→ 200 OK")
34    except OSError:
35        print("IO error")
36        connectionSocket.send(HttpResponse(404, "Not Found", "text/html", notFoundHtmlText(path, addr).encode()))
37        print("→ 404 Not Found")
38    except:
39        print("Some Internal Error")
40        connectionSocket.send(HttpResponse(500, "Internal Server Error", "text/html", "Internal Server Error".encode()))
41    else:
42        connectionSocket.close()
43        print ("Responded Successfully!")
44        print("-----")

```

Figure 2.45: Image/video redirect logic and MIME detection

2.4 Task 2 (e) – Default and Error Responses

The server responds appropriately based on the requested path:

- For requests to `/`, `/en`, `/index.html`, or `/main_en.html`, the server responds with `main_en.html`.
- For requests to `/ar` or `/main_ar.html`, the response is `main_ar.html`.
- For invalid URLs or non-existent files, the server generates a custom 404 page with:
 - Status line: `HTTP/1.1 404 Not Found`
 - Browser tab title: `Error 404`
 - Red body text: “The file is not found”
 - Client IP and port

This behaviour was verified through browser testing and command-line logs

2.5 Task 2 – Experimental Cases

To verify the proper functionality of the web server, we tested different types of requests and confirmed that the expected responses were generated according to MIME type logic and redirect/error handling.

2.5.1 Case 1: Successful Image Request

An image file named `test1.png` was requested from the server. The server responded with a 200 OK message and correctly returned the image.

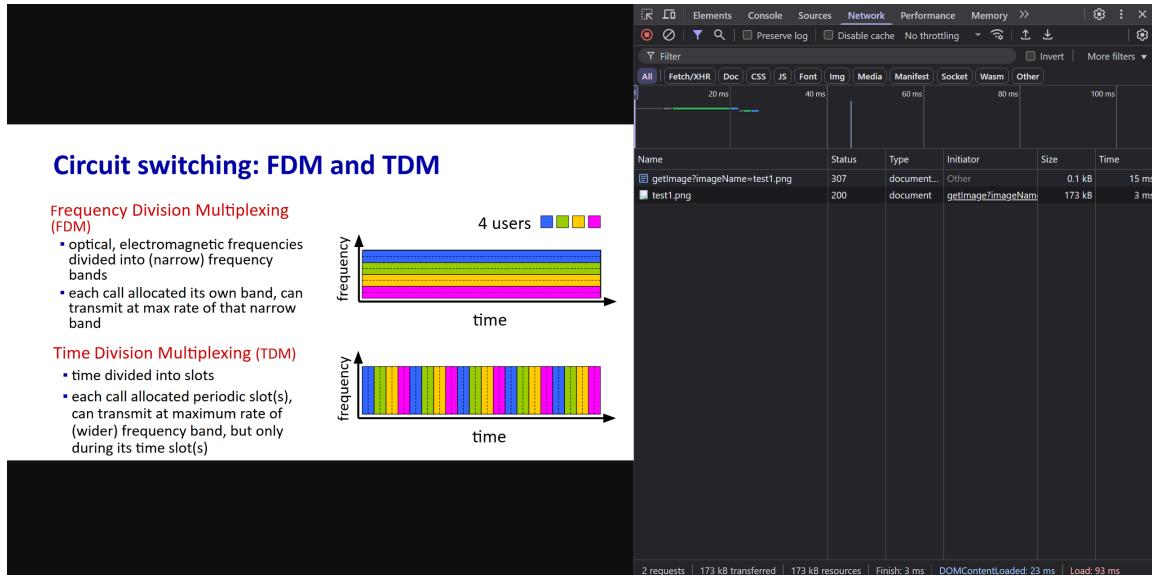


Figure 2.46: Image successfully retrieved (`test1.png`)

```
Request From ('192.168.56.1', 63462)
GET /test1.png HTTP/1.1
Host: 192.168.56.1:9991
Connection: keep-alive
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/136.0.0.0 Safari/537.36
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8
Sec-GPC: 1
Accept-Language: en-US,en;q=0.6
Referer: http://192.168.56.1:9991/mySite_1232951_en.html
Accept-Encoding: gzip, deflate

→ 200 OK
Responded Successfully!
```

Figure 2.47: Terminal log showing successful image request and 200 OK response

2.5.2 Case 2: Successful Google Image Redirect (PNG)

The file `abdallah.png` was requested. The server could not locate the file locally and responded with a 307 Temporary Redirect to Google Image Search.

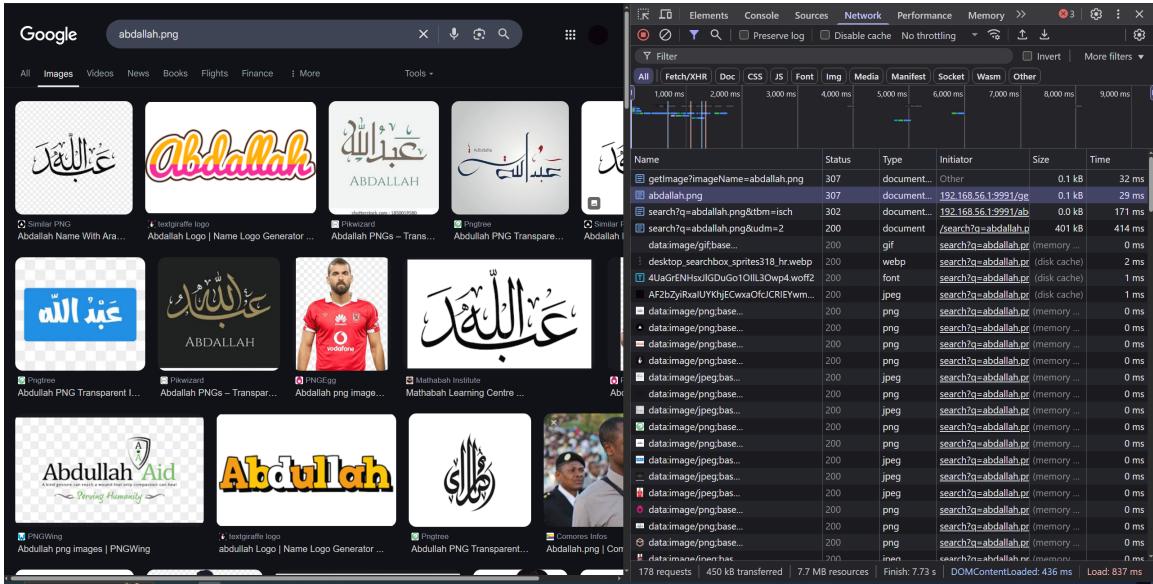


Figure 2.48: Image redirection example for abdallah.png (status 307)

2.5.3 Case 3: Successful Google Video Redirect (MP4)

A media file named `jamal.mp4` was requested. The file does not exist in the server directory. The server identified the `.mp4` MIME type and redirected the client to a YouTube search using a 307 response.

```

-----  

Request From ('192.168.56.1', 63190)  

GET /getImage?imageName=jamal.mp4 HTTP/1.1  

Host: 192.168.56.1:9991  

Connection: keep-alive  

Upgrade-Insecure-Requests: 1  

User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/136.0.0.0 Safari/537.36  

Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8  

Sec-GPC: 1  

Accept-Language: en-US,en;q=0.6  

Referer: http://192.168.56.1:9991/mySite_1232951_en.html  

Accept-Encoding: gzip, deflate  

→ 307 Temporary Redirect  

Responded Successfully!
-----
```

Figure 2.49: Redirected search result for `jamal.mp4` to YouTube (307 Temporary Redirect)

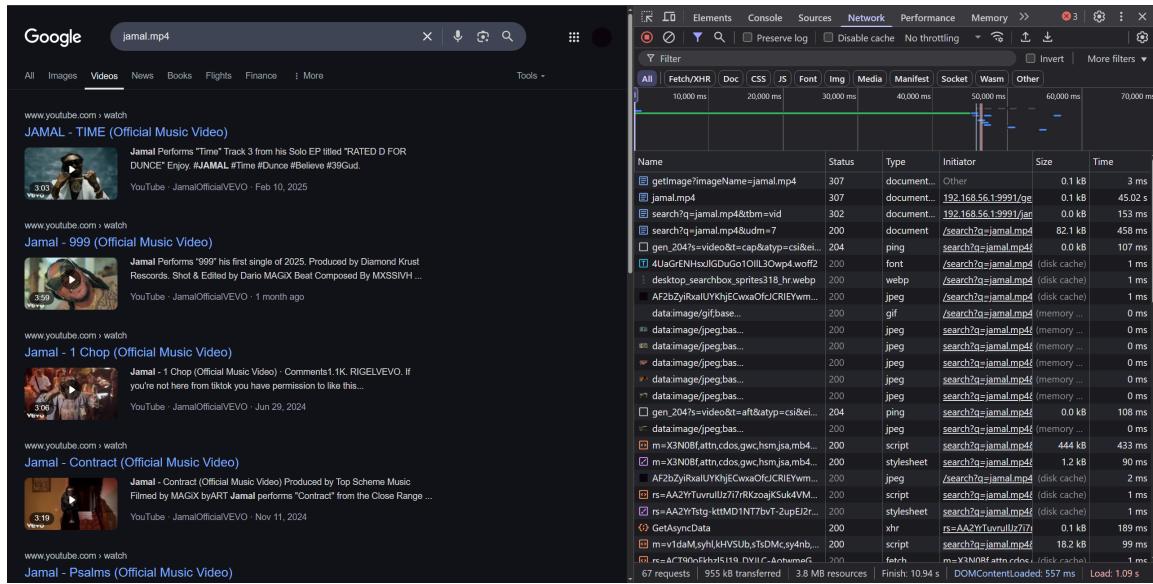


Figure 2.50: Redirect behaviour for video file (jamal.mp4) shown in browser and developer tools

2.5.4 Case 4: File Not Found (404 Error)

A text file named `jamal.txt` was requested. Since the file does not exist and doesn't match any redirect logic, the server returned a 404 error page with client details.

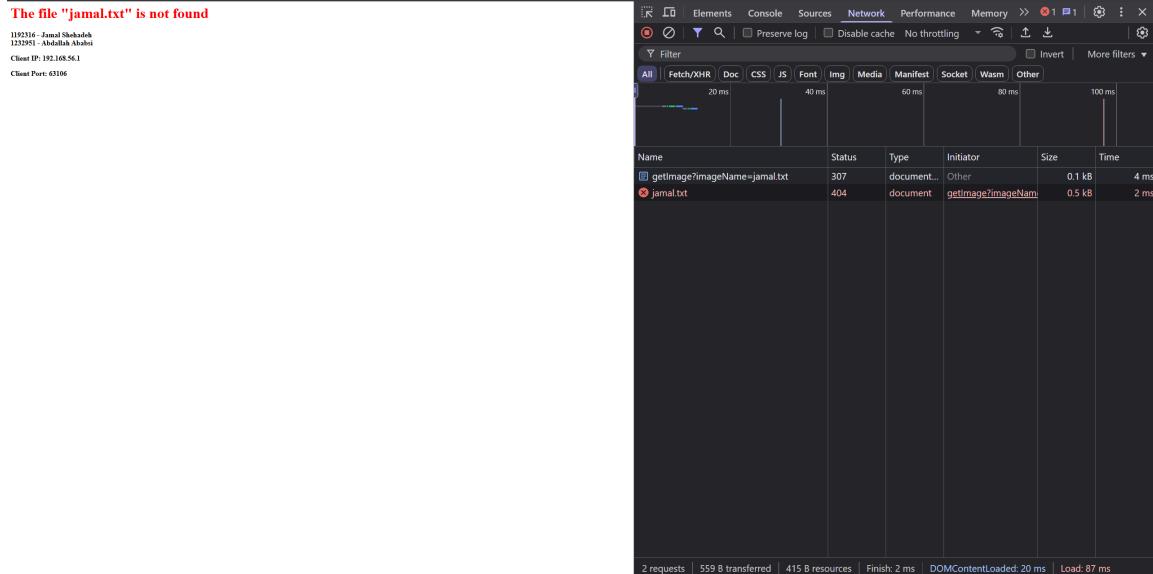


Figure 2.51: Client-side display of 404 Not Found error for jamal.txt

```
-----  
Request From ('192.168.56.1', 63106)  
GET /jamal.txt HTTP/1.1  
Host: 192.168.56.1:9991  
Connection: keep-alive  
Upgrade-Insecure-Requests: 1  
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/136.0.0.0 Safari/537.36  
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8  
Sec-GPC: 1  
Accept-Language: en-US,en;q=0.6  
Referer: http://192.168.56.1:9991/mySite_1232951_en.html  
Accept-Encoding: gzip, deflate  
  
→ 404 Not Found  
Responded Successfully!  
-----
```

Figure 2.52: Server-side log output for 404 Not Found (jamal.txt)

2.6 Task 3 – TCP/UDP Hybrid Client-Server Game Using Socket Programming

2.6.1 a) Game Features

This is a multiplayer number-guessing game where clients compete to guess a secret number generated by the server. The game uses a hybrid network architecture:

- **TCP** is used for player registration, control messages, and results.
- **UDP** is used for fast real-time guessing to reduce latency.

Features include:

- Support for 2 to 4 simultaneous players.
- Unique username registration with validation.
- Game rounds with real-time feedback: "Higher", "Lower", "Correct".
- Result announcement and round restart via TCP.

b) Components and Protocol Design

TCP Phase:

- Clients connect to the server (port 6000) and send `JOIN <username>`.
- Server checks uniqueness and accepts or rejects the username.
- When enough players have joined, the game begins.

UDP Phase:

- Server selects a random number within a defined range (e.g., 1–100).
- Players send guesses to the server on UDP port 6001.
- Server replies to each player via UDP with feedback.
- The first player to guess correctly wins; results are broadcast to all clients via TCP.

2.6.2 c) Game Workflow

Server Side:

- Initializes TCP and UDP sockets.
- Waits for player registration.
- Starts game round and generates secret number.

- Collects UDP guesses, checks correctness, and announces winner.

Client Side:

- Connects via TCP, submits username.
- Sends guesses via UDP.
- Listens for game updates and final results.

Code Screenshots:



```

1
2 # generate random secret number by server (done)
3 # register players with TCP (done)
4 # players attempt to guess using UDP (done)
5 # server responds to each guess with a message using UDP (e.g., "too high", "too low", "correct") (done)
6 # if player wins, send them a message using TCP to all players (done)
7
8 import socket, random, threading, time, collections
9
10
11
12 tcp_port = 6000
13 udp_port = 6001
14 server_name = "localhost"
15 min_players = 2
16 max_players = 4
17 to_enter_your_guess = 10.0
18 game_duration = 60.0
19 player_reg_timeout = 120.0
20 player_kick_timeout = 5.0
21
22 player_list = []
23 tcp_connections = {}
24
25 udp_player_addresses = {}
26
27 server_secret_number = 0 # initialized to 0
28
29 tcp_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
30 udp_socket = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
31
32 player_guesses = {}
33
34
35
36
37
38
39 def udp_pingpong(player_name: str):
40     try:
41         udp_socket.sendto(b"ping", udp_player_addresses[player_name])
42         udp_socket.settimeout(5)
43         if udp_socket.recvfrom(1024)[0] != b"pong":
44             raise Exception("No pong received")
45     except Exception as e:
46         udp_player_addresses[player_name].close()
47         return False
48     return True
49
50
51

```

Figure 2.53: Client-side socket setup

```
● ○ ●
1
2
3 def generate_secret_number() -> int:
4     return random.randint(1, 100)
5
6
7 def register_udp_players():
8     timeout = 5
9     start_time = time.time()
10    while True:
11        elapsed_time = time.time() - start_time
12        if elapsed_time > timeout:
13            break
14        try:
15            udp_socket.settimeout(10)
16            message, client_address = udp_socket.recvfrom(1024)
17            player_name = message.decode()
18            udp_player_addresses[player_name] = udp_player_addresses.get(player_name, client_address)
19
20        except socket.timeout:
21            continue
22        except Exception as e:
23            continue
24
25    def verify_udp_players():
26        for player_name in player_list:
27            if player_name not in udp_player_addresses:
28                remove_player(player_name)
29
30
31
32    def check_round_over(start_time: float) -> bool:
33        for player_name in player_list:
34            if player_guesses[player_name] == server_secret_number:
35                return True
36        elapsed_time = time.time() - start_time
37        if elapsed_time > game_duration:
38            return True
39        return False
40
41
42    def udp_remove_player(player_name: str):
43        if player_name in udp_player_addresses:
44            del udp_player_addresses[player_name]
45        if player_name in player_guesses:
46            del player_guesses[player_name]
47        if player_name in player_list:
48            player_list.remove(player_name)
49        if player_name in tcp_connections:
50
51
```

Figure 2.54: UDP game loop and message handling

```

● ● ●

1     tcp_connections[player_name].close()
2     del tcp_connections[player_name]
3
4
5 def udp_player_turn(player_name: str, start_time):
6     while True:
7         try:
8             if check_round_over(start_time):
9                 return
10
11         if not udp_pingpong(player_name):
12             print(f"Player {player_name} disconnected.")
13             udp_remove_player(player_name)
14             return
15
16
17
18     udp_socket.sendto("Enter your guess (1-100): ".encode(), udp_player_addresses[player_name])
19     message, client_address = udp_socket.recvfrom(1024)
20     raw_guess = message.decode()
21     try:
22         guess = int(raw_guess)
23         if guess < 1 or guess > 100:
24             udp_socket.sendto("Warning: Out of range, miss your chance.".encode(), client_address)
25             time.sleep(to_enter_your_guess)
26             continue
27     except ValueError:
28         udp_socket.sendto("Warning: invalid input. miss your chance.".encode(), client_address)
29         time.sleep(to_enter_your_guess)
30         continue
31
32     player_guesses[player_name] = guess
33
34     if guess == server_secret_number:
35         udp_socket.sendto("Feedback: CORRECT".encode(), client_address)
36         broadcast_tcp_message(f"{player_name} guessed the number {server_secret_number} correctly!")
37         return
38     elif guess < server_secret_number:
39         udp_socket.sendto("Feedback: Higher".encode(), client_address)
40         time.sleep(to_enter_your_guess)
41     else:
42         udp_socket.sendto("Feedback: Lower".encode(), client_address)
43         time.sleep(to_enter_your_guess)
44     except Exception as e:
45         continue
46
47
48 def game_round():
49     start_time = time.time()
50

```

Figure 2.55: Server TCP ping-pong and client monitoring

```
● ○ ●
1     threading.Thread(target=udp_player_turn, args=(player_name,start_time)).start()
2
3     while not check_round_over(start_time):
4         time.sleep(1)
5
6
7
8
9
10    def broadcast_udp_message(message: str):
11        for player_name in udp_player_addresses:
12            try:
13                udp_socket.sendto(message.encode(), udp_player_addresses[player_name])
14            except Exception as e:
15                continue
16
17
18
19 ##### TCP #####
20 #          TCP          #
21 #####
22
23 # returns true/false if the game can start
24 def check_start_game() -> bool:
25     if len(player_list) < min_players:
26         return False
27     return True
28
29
30 def broadcast_tcp_message(message: str):
31     for player_name in tcp_connections:
32         connection = tcp_connections[player_name]
33         try:
34             connection.send(message.encode())
35         except Exception as e:
36             continue
37     return True
38
39
40 # attempt to register a player, returns true if successful, false otherwise
41 def register_player(player_name: str) -> tuple[str, bool]:
42     if player_name in player_list:
43         return "player already registered", False
44     if len(player_list) > max_players:
45         return "too many players", False
46     if len(player_name) < 3:
47         return "name too short", False
48     if len(player_name) > 100:
49         return "name too long", False
50     player_list.append(player_name)
51
```

Figure 2.56: UDP player management and gameplay round

```

1  waiting_room = '\n'.join(player_list)
2  return f"\nConnected as {player_name}\nWaiting room:\n{waiting_room}", True
3
4
5
6  def remove_player(player_name: str):
7      if player_name in player_list:
8          player_list.remove(player_name)
9
10
11
12 def check_player_connections():
13     for player_name in player_list:
14         try:
15             tcp_connections[player_name].settimeout(player_kick_timeout)
16             tcp_connections[player_name].send(b"ping")
17             if tcp_connections[player_name].recv(1024) != b"pong":
18                 raise Exception("No pong received")
19         except socket.timeout:
20             print(f"Player {player_name} timed out. Removing from game.")
21             tcp_connections[player_name].close()
22             del tcp_connections[player_name]
23             remove_player(player_name)
24         except Exception as e:
25             print(f"Error checking connection for player {player_name}: {e}")
26             tcp_connections[player_name].close()
27             del tcp_connections[player_name]
28             remove_player(player_name)
29
30 def close_tcp_connections():
31     for con in tcp_connections:
32         try:
33             connection = tcp_connections[con]
34             connection.close()
35         except Exception as e:
36             print(f"Error closing connection: {e}")
37             continue
38     tcp_connections.clear()
39
40 def init_sockets():
41     tcp_socket.bind((server_name, tcp_port))
42     tcp_socket.listen(5)
43     tcp_socket.setblocking(False)
44     print(f"TCP socket listening on port {tcp_port}...")
45
46
47     udp_socket.bind((server_name, udp_port))
48     udp_socket.setblocking(False)
49     print(f"UDP socket listening on port {udp_port}...")
50
51

```

Figure 2.57: UDP guess handling and feedback logic

```
1 def pingpong(connection_socket: socket.socket):
2     try:
3         connection_socket.send(b"ping")
4         connection_socket.settimeout(5)
5         if connection_socket.recv(1024) != b"pong":
6             raise Exception("No pong received")
7     except socket.timeout:
8         connection_socket.close()
9     return False
10 except Exception as e:
11     connection_socket.close()
12     return False
13
14 return True
15
16 def handle_client(connection_socket: socket.socket, addr):
17     try:
18         # send the welcome message
19         connection_socket.send(f"Welcome to the game server! Please enter your name: ".encode())
20         connection_socket.settimeout(player_reg_timeout)
21
22         # receive the player's name
23         player_name = connection_socket.recv(1024).decode()
24         print(f"Received player name: {player_name}")
25
26         if not player_name:
27             print("No player name received. Closing connection.")
28             connection_socket.close()
29             return
30         response_message, response_status = register_player(player_name)
31
32
33         while not response_status:
34             connection_socket.send(f"{response_message}".encode())
35             player_name = connection_socket.recv(1024).decode()
36             response_message, response_status = register_player(player_name)
37
38             connection_socket.send(f"{response_message}".encode())
39
40         connection_socket.settimeout(None)
41         tcp_connections[player_name] = connection_socket
42
43     return
44
45 except TimeoutError:
46     print("Player registration timed out. Closing connection.")
47     connection_socket.close()
48     return
49
50 def accept_tcp_connections_threaded():
51
```

Figure 2.58: Round broadcasting and result distribution

```

1      while True:
2          # accept a new connection
3          try:
4              connection_socket, addr = tcp_socket.accept()
5          except BlockingIOError:
6              time.sleep(2)
7              check_player_connections()
8              if check_start_game():
9                  print("Game can start...")
10             return
11         continue
12     except Exception as e:
13         print(f"Error accepting connection: {e}")
14         continue
15
16     # start a new thread for each connection
17     threading.Thread(target=handle_client, args=(connection_socket,addr)).start()
18
19
20 def ask_users_to_play():
21     for player_name in player_list:
22         try:
23             tcp_connections[player_name].send(b"Do you want to play? (yes/no): ")
24             tcp_connections[player_name].settimeout(5)
25             response = tcp_connections[player_name].recv(1024).decode()
26             if response.lower() == "no":
27                 remove_player(player_name)
28                 continue
29         except socket.timeout:
30             print(f"Player {player_name} timed out. Removing from game.")
31             tcp_connections[player_name].close()
32             del tcp_connections[player_name]
33             remove_player(player_name)
34         except Exception as e:
35             print(f"Error checking connection for player {player_name}: {e}")
36             tcp_connections[player_name].close()
37             del tcp_connections[player_name]
38             remove_player(player_name)
39
40
41
42
43 ##### MAIN #####
44 #
45 ##### MAIN #####
46
47 def main():
48     init_sockets()
49
50     # print start message
51

```

Figure 2.59: TCP handling thread logic

```

● ● ●

1     # print start message
2     print(f"Server started on {server_name}: TCP {tcp_port}, UDP {udp_port}")
3
4     accept_tcp_connections_threaded()
5     still_playing = True
6
7     while still_playing:
8         for player_name in player_list:
9             player_guesses[player_name] = 0
10
11    print(f"Game started with {len(player_list)} players...")
12    broadcast_tcp_message(f"Game started with players: {' '.join(player_list)}")
13
14    # initialize game logic
15    server_secret_number = generate_secret_number()
16
17    register_udp_players()
18    time.sleep(5)
19    verify_udp_players()
20
21    broadcast_udp_message("You have 60 seconds to guess the number (1-100)!")
22
23    print("round begin")
24    game_round()
25
26    for player_name in player_list:
27        if player_guesses[player_name] == server_secret_number:
28            message = """==GAME RESULTS==
29            Target number: {server_secret_number}
30            Winner: {player_name}
31            """
32            broadcast_tcp_message(f"{player_name} guessed the number {server_secret_number} correctly!")
33        else:
34            message = f"==GAME RESULTS==\nTarget number: {server_secret_number}\nWinner: None"
35            broadcast_tcp_message(message)
36
37        ask_users_to_play()
38        still_playing = check_start_game()
39
40    close_tcp_connections()
41
42
43 main()
44
45

```

Figure 2.60: Final game loop and result tracking

d) Test Cases and Expected Output

EXP – Two Players Join

```

Administrator: Command Prompt
Microsoft Windows [Version 10.0.19045.5737]
(c) Microsoft Corporation. All rights reserved.

C:\Windows\system32>cd C:\Users\meh62\Desktop\T051_Project1\Task 3
C:\Users\meh62\Desktop\T051_Project1\Task 3>python server.py
TCP socket listening on port 6000...
UDP socket listening on port 6001...
Server started on localhost: TCP 6000, UDP 6001
Received player name: abdallah
Game can start...
Game started with 2 players...
round begin
Player jamal timed out. Removing from game.

Administrator: Command Prompt
(c) Microsoft Corporation. All rights reserved.

C:\Windows\system32>color 9
C:\Windows\system32>cd C:\Users\meh62\Desktop\T051_Project1\Task 3
C:\Users\meh62\Desktop\T051_Project1\Task 3>python client.py
The filename, directory name, or volume label syntax is incorrect.

Administrator: Command Prompt
(c) Microsoft Corporation. All rights reserved.

C:\Windows\system32>cd C:\Users\meh62\Desktop\T051_Project1\Task 3
C:\Users\meh62\Desktop\T051_Project1\Task 3>python client.py
Welcome to the game server! Please enter your name: abdallah

Connected as abdallah
Waiting room:
jamal
abdallah
UDP connection established
Game started with players: jamal, abdallah
UDP message sent to server: jamal
You have 60 seconds to guess the number (1-100)!40
UDP message sent to server: 40
Error receiving message: [WinError 10054] An existing connection was forcibly closed by the
remote host

C:\Users\meh62\Desktop\T051_Project1\Task 3>

```

Figure 2.61: Two players join and the game starts. A bug happens.

```
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
```

Figure 2.62: Client inputs a guess outside valid range and receives a warning.

```

● ● ●

1     # print start message
2     print(f"Server started on {server_name}: TCP {tcp_port}, UDP {udp_port}")
3
4     accept_tcp_connections_threaded()
5     still_playing = True
6
7     while still_playing:
8         for player_name in player_list:
9             player_guesses[player_name] = 0
10
11    print(f"Game started with {len(player_list)} players...")
12    broadcast_tcp_message(f"Game started with players: {' '.join(player_list)}")
13
14    # initialize game logic
15    server_secret_number = generate_secret_number()
16
17    register_udp_players()
18    time.sleep(5)
19    verify_udp_players()
20
21    broadcast_udp_message("You have 60 seconds to guess the number (1-100)!")
22
23    print("round begin")
24    game_round()
25
26    for player_name in player_list:
27        if player_guesses[player_name] == server_secret_number:
28            message = """==GAME RESULTS==
29            Target number: {server_secret_number}
30            Winner: {player_name}
31            """
32            broadcast_tcp_message(f"{player_name} guessed the number {server_secret_number} correctly!")
33        else:
34            message = f"==GAME RESULTS==\nTarget number: {server_secret_number}\nWinner: None"
35            broadcast_tcp_message(message)
36
37        ask_users_to_play()
38        still_playing = check_start_game()
39
40    close_tcp_connections()
41
42
43 main()
44
45

```

Figure 2.63: Server detects disconnected player and handles it gracefully.

e) Initial Settings

- `tcp_port = 6000`
- `udp_port = 6001`
- `server_name = 'localhost'`
- `min_players = 2, max_players = 4`
- `guess_timeout = 10` seconds
- `game_duration = 60` seconds
- Player Names: `Player_1 = Abdallah_Abbasi, Player_2 = Jamal_Shehadeh`
- Libraries used: `socket, random, threading, time, collections`

Code Overview and Architecture

The game is implemented using:

- `server.py` – Sets up TCP/UDP sockets, manages game rounds, handles registration, guessing, and result broadcasting.
- `client.py` – Handles TCP registration and UDP-based guessing; receives feedback and results from server.

Server Highlights (`server.py`)

- Player registration is handled through dedicated TCP threads.
- UDP guesses are processed in real-time using a listener thread.
- A separate monitoring thread checks TCP connection health using ping messages.
- Results are broadcast to all clients upon a correct guess.

Client Highlights (`client.py`)

- Connects to TCP server and submits unique username.
- Sends guesses via UDP and listens for game status updates.
- Replies with "pong" when receiving TCP ping to prevent disconnection.

Technical Challenge

Despite proper hybrid implementation, a critical issue persists:

- The server periodically sends "ping" over TCP to verify clients.
- The client doesn't respond quickly enough with "pong", triggering `WinError 10054`.
- The server then closes the TCP connection and ends the round, even though UDP communication continues.

Attempts were made to fix this by implementing a TCP pong reply handler on the client side. However, due to timing or threading limitations, the issue remains. Future work should ensure a persistent TCP listener on the client side to catch and reply to pings immediately.

Chapter 3

Conclusion

Conclusion

A complete and simple network system was created and put into use for this project. A simple web server, HTML pages in Arabic and English, and a multiplayer guessing game that uses both TCP and UDP were all part of the project.

Every feature that was needed was created successfully. The web server managed redirects and errors while returning the appropriate pages. Players could register, guess a number in real time, and get feedback while playing the game.

The main logic and gameplay were finished correctly, even though a small issue in the TCP connection during the game. This project helped in understanding of socket programming and client-server communication management.