A Course Based Project Report on

ARP and RARP Simulation

Submitted to the

Department of CSE-(CyS, DS) and AI&DS

in partial fulfilment of the requirements for the completion of course

Computer Networks and Ethical hacking LABORATORY (22PC2CY201)

BACHELOR OF TECHNOLOGY

IN

CSE-Data Science

Submitted by

N. SRIKAR GOUD	23071A67B1
N. TEJASWINI	23071A67B2
N. CHARAN	23071A67B3
NITHYA. S	23071A67B4
N. MANIKESHAV	23071A67B5

Under the guidance of

Mrs. G. Usha Rani

Assistant Professor

Department of CSE-(CyS, DS) and AI&DS

VALLURUPALLI NAGESWARA RAO VIGNANA JYOTHI INSTITUTE OF ENGINEERING & TECHNOLOGY

An Autonomous Institute, NAAC Accredited with 'A++' Grade, NBA

VignanaJyothi Nagar, Pragathi Nagar, Nizampet (S.O), Hyderabad – 500 090, TS, India

December--2025

VALLURUPALLI NAGESWARA RAO VIGNANA JYOTHI INSTITUTE OF ENGINEERING AND TECHNOLOGY

An Autonomous, ISO 21001:2018& QS I-Gauge Diamond Rated Institute, Accredited by NAAC with 'A++' Grade
NBA Accreditation for B.Tech. CE,EEE,ME,ECE,CSE,EIE,IT,AME, M.Tech. STRE, PE, AMS, SWEProgrammes
Approved by AICTE, New Delhi, Affiliated to JNTUH, NIRF (2024) Rank band:151-200in EngineeringCategory
College with Potential for Excellence by UGC,JNTUH-Recognized Research Centres:CE,EEE,ME,ECE,CSEVignana Jyothi Nagar,
Pragathi Nagar, Nizampet (S.O.), Hyderabad – 500 090, TS, India.
Telephone No: 040-2304 2758/59/60, Fax: 040-23042761
E-mail: postbox@vnrvjiet.ac.in, Website: www.vnrvjiet.ac.in

Department of CSE-(CyS, DS) and AI&DS

CERTIFICATE

This is to certify that the project report entitled **"ARP and RARP Simulation"** is a bonafide work done under our supervision and is being submitted by **Mr. N. Srikar (23071A67B1), Miss. N. Tejaswini (23071A67B2), Mr. N. Charan (23071A67B3), Miss. Nithya. S (23071A67B4), Mr. N. Manikeshav (23071A67B5)** in partial fulfillment for the award of the degree of **Bachelor of Technology** in **CSE-Data Science**, of the VNRVJIET, Hyderabad during the academic year 2025-2026.

Mrs. G. Usha Rani
Assistant Professor
Dept of CSE-(CyS, DS) and AI&DS

Dr. T. Sunil Kumar
Professor& HOD
Dept of CSE-(CyS, DS) and AI&DS

Course based Projects Reviewer

VALLURUPALLI NAGESWARA RAO VIGNANA JYOTHI INSTITUTE OF ENGINEERING AND TECHNOLOGY

An Autonomous Institute, NAAC Accredited with 'A++' Grade,
VignanaJyothi Nagar, Pragathi Nagar, Nizampet(SO), Hyderabad-500090, TS, India

Department of CSE-(CyS, DS) and AI&DS

DECLARATION

We declare that the course based project work entitled **"ARP and RARP Simulation"** submitted in the Department of **CSE-(CyS, DS) and AI&DS**, Vallurupalli Nageswara Rao Vignana Jyothi Institute of Engineering and Technology, Hyderabad, in partial fulfillment of the requirement for the award of the degree of **Bachelor of Technology in CSE-Data Science** is a bonafide record of our own work carried out under the supervision of **G. Usha Rani, Assistant Professor, Department of CSE-(CyS, DS) and AI&DS, VNRVJIET.** Also, we declare that the matter embodied in this thesis has not been submitted by us in full or in any part thereof for the award of any degree/diploma of any other institution or university previously.

Place: Hyderabad.

N. SRIKAR GOUD	23071A67B1
N. TEJASWINI	23071A67B2
N. CHARAN	23071A67B3
NITHYA. S	23071A67B4
N. MANIKESHAV	23071A67B5

ACKNOWLEDGEMENT

We express our deep sense of gratitude to our beloved President, Sri. D. Suresh Babu, VNR Vignana Jyothi Institute of Engineering & Technology for the valuable guidance and for permitting us to carry out this project.

With immense pleasure, we record our deep sense of gratitude to our beloved Principal, Dr. C.D Naidu, for permitting us to carry out this project.

We express our deep sense of gratitude to our beloved Professor **Dr. T. Sunil Kumar**, Professor and Head, Department of CSE-(CyS, DS) and AI&DS, VNR Vignana Jyothi Institute of Engineering & Technology, Hyderabad-500090 for the valuable guidance and suggestions, keen interest and through encouragement extended throughout the period of project work.

We take immense pleasure to express our deep sense of gratitude to our beloved Guide, Mrs. **G. Usha Rani**, Assistant Professor in CSE-(CyS, DS) and AI&DS, VNR Vignana Jyothi Institute of Engineering & Technology, Hyderabad, for his/her valuable suggestions and rare insights, for constant source of encouragement and inspiration throughout my project work.

We express our thanks to all those who contributed for the successful completion of our project work.

N. SRIKAR GOUD	23071A67B1
N. TEJASWINI	23071A67B2
N. CHARAN	23071A67B3
NITHYA. S	23071A67B4
N. MANIKESHAV	23071A67B5

TABLE OF CONTENTS

ABSTRACT

The "ARP and RARP Simulation" is a Python-based console application developed to demonstrate the working principles of the Address Resolution Protocol (ARP) and the Reverse Address Resolution Protocol (RARP) in computer networks. The system simulates how devices within a Local Area Network (LAN) resolve IP addresses to MAC addresses (using ARP) and MAC addresses to IP addresses (using RARP) for successful communication at the data link and network layers of the OSI model. By utilizing core Python features such as dictionaries, random address generation, and structured menu-driven interaction, the project effectively models real-world address resolution processes.

The application maintains an internal ARP cache containing a set of predefined IP–MAC address mappings and dynamically updates this table when new address resolutions occur. When a user initiates an ARP request, the system checks for the IP address in the cache and, if not found, simulates broadcasting an ARP request, receiving a reply, and updating the cache. Similarly, for RARP operations, the system performs reverse lookups to associate MAC addresses with IP addresses and simulates server-based address assignment when necessary. This realistic simulation flow provides users with an understanding of how ARP and RARP operate within actual network environments.

The "ARP and RARP Simulation" can be further enhanced by incorporating packet-level interaction using libraries such as Scapy or by integrating graphical interfaces for better visualization. It serves as an excellent demonstration of fundamental networking principles, offering practical insight into how network devices identify each other and establish communication within a LAN. Overall, this project bridges theoretical networking concepts with practical implementation, helping learners clearly visualize and understand the importance of address resolution in computer networks.

CHAPTER-1

INTRODUCTION

In modern computer networks, seamless communication between devices within a Local Area Network (LAN) depends on the ability to accurately identify and locate other devices. Every device in a network communicates using IP addresses, but actual data transfer occurs using hardware or MAC addresses. The "ARP and RARP Simulation" project is developed to demonstrate how this address mapping process occurs through the Address Resolution Protocol (ARP) and the Reverse Address Resolution Protocol (RARP), both of which are fundamental components of network communication at the data link and network layers of the OSI model.

The system is implemented using Python and simulates how ARP resolves IP addresses to MAC addresses and how RARP performs the reverse operation, mapping MAC addresses to IP addresses. The application uses a preloaded ARP cache containing realistic IP–MAC pairs and dynamically updates this table as new address resolutions are simulated. It provides users with an interactive console interface that displays realistic step-by-step logs, such as cache checking, broadcasting requests, receiving replies, and updating entries—mirroring the actual operation of these protocols in real networks.

By employing Python's core programming features like dictionaries, random data generation, and structured flow control, the simulation accurately replicates the logical workflow of ARP and RARP without requiring complex network configurations or administrative privileges. The project helps learners visualize how devices in a LAN communicate before data transmission occurs and highlights the essential role of ARP and RARP in maintaining efficient and reliable network communication.

Overall, the "ARP and RARP Simulation" provides a practical, educational approach to understanding protocol behavior, network address mapping, and the foundational concepts of communication in computer networks. It bridges theoretical knowledge with hands-on demonstration, making it a valuable learning tool for students and network enthusiasts.

CHAPTER-2

METHODOLOGY

2.1. Project Architecture

The "ARP and RARP Simulation" is a Python-based console application designed to simulate the behavior of the Address Resolution Protocol (ARP) and the Reverse Address Resolution Protocol (RARP) within a Local Area Network (LAN).

• The system operates as a standalone simulation that models how devices resolve IP and MAC addresses before communication.

• A command-line menu interface allows users to perform ARP requests, RARP lookups, display cache contents, or exit the program.

• An internal ARP cache is maintained using a Python dictionary that stores IP–MAC address pairs.

• The cache is preloaded with 15 realistic entries and is dynamically updated as new address resolutions occur.

2.2. Technologies Used

• Python 3 – Used as the main programming language for simulation.

• Random – Generates realistic IP and MAC addresses dynamically.

• Time – Adds short delays between steps for realistic simulation timing.

• OS and Platform – Used for clearing the console and ensuring cross-platform compatibility.

2.3. Features Implemented

• Simulates ARP process for mapping IP addresses to MAC addresses.

• Implements RARP process for mapping MAC addresses to IP addresses.

• Maintains and updates an ARP cache table throughout execution.

• Displays step-by-step console messages that mimic actual network operations.

• Provides an interactive and user-friendly text-based interface.

• Runs entirely offline without external dependencies or network access.

2.4. Workflow

• Initialization: Loads the predefined ARP cache with 15 IP–MAC entries.

• ARP Request:

Checks the cache for the entered IP address.

If not found, simulates broadcasting an ARP request.

Receives a reply with a realistic MAC address and updates the cache.

• RARP Request:

Searches the cache for the entered MAC address.

If not found, simulates sending a RARP request to a server.

Receives a reply assigning a valid IP address and updates the cache.

• Display Cache: Shows the complete list of IP–MAC mappings.

• Exit: Ends the program cleanly upon user selection.

2.5. Reliability and Data Handling

• All IP and MAC addresses follow valid, realistic formats.

• User inputs are validated to prevent incorrect entries.

• The ARP cache remains consistent throughout program execution.

• Messages clearly reflect each protocol stage (e.g., request, reply, update).

2.6. Demonstrated Concepts

• Address resolution between IP and MAC addresses using ARP and RARP.

• Interaction between the Network Layer and Data Link Layer in communication.

• Cache management and dynamic table updates in network systems.

• Understanding of broadcast and reply mechanisms in protocol operation.

• Practical learning of address resolution concepts through simulation.

CHAPTER-3

Code

```
import random
import os
import platform
import re
from time import sleep

# ---------- Preloaded ARP Cache ----------
arp_cache = {
    "192.168.1.1":  "3C:F0:11:9B:2A:11",
    "192.168.1.2":  "98:5F:D3:4A:77:22",
    "192.168.1.3":  "B4:6D:83:1F:23:33",
    "192.168.1.4":  "54:2A:1B:7C:99:44",
    "192.168.1.5":  "70:85:C2:3B:56:55",
    "192.168.1.6":  "28:92:A4:6F:88:66",
    "192.168.1.7":  "40:B8:37:9D:1A:77",
    "192.168.1.8":  "A0:32:99:4E:9C:88",
    "192.168.1.9":  "18:AF:61:2C:4E:99",
    "192.168.1.10": "9C:3D:CF:7B:54:AA",
    "192.168.1.11": "F4:6B:EF:88:1D:BB",
    "192.168.1.12": "5C:CF:7F:9E:AA:CC",
    "192.168.1.13": "B8:8A:EC:1F:CB:DD",
    "192.168.1.14": "84:C5:A6:4B:76:EE",
    "192.168.1.15": "00:1A:2B:3C:4D:5E"
}

# ---------- Utility Functions ----------
def clear_screen():
    os.system('cls' if platform.system() == 'Windows' else 'clear')

def random_mac():
```

```
    """Generate a realistic random unicast MAC address."""
    mac = [0x00, 0x16, 0x3e,
           random.randint(0x00, 0x7f),
           random.randint(0x00, 0xff),
           random.randint(0x00, 0xff)]
    return ':'.join(map(lambda x: "%02X" % x, mac))

def validate_ip(ip):
    """Check if an IP address is in correct format."""
    pattern = re.compile(
        r"^(25[0-5]|2[0-4][0-9]|[01]?[0-9][0-9]?)\."
        r"(25[0-5]|2[0-4][0-9]|[01]?[0-9][0-9]?)\."
        r"(25[0-5]|2[0-4][0-9]|[01]?[0-9][0-9]?)\."
        r"(25[0-5]|2[0-4][0-9]|[01]?[0-9][0-9]?)$"
    )
    return re.match(pattern, ip)

def validate_mac(mac):
    """Check if a MAC address is in correct format."""
    pattern = re.compile(r"^([0-9A-Fa-f]{2}:){5}([0-9A-Fa-f]{2})$")
    return re.match(pattern, mac)

# ---------- ARP Process ----------
def arp_request(ip):
    print(f"\n[*] Checking ARP cache for IP {ip}...")
    sleep(0.7)

    if not validate_ip(ip):
        print(f"[!] Invalid IP address format. Please enter a valid IP (e.g., 192.168.1.10).")
        return

    if ip in arp_cache:
        print(f"[+] Entry found in cache.")
        print(f"[#] MAC Address of {ip} is {arp_cache[ip]}")
```

```
        return

    print(f"[!] IP {ip} not found in cache.")
    sleep(0.6)
    print("[*] Broadcasting ARP Request on LAN...")
    sleep(1.2)
    print(f"[+] Reply received from {ip}.")
    sleep(0.5)

    mac = random_mac()
    arp_cache[ip] = mac
    print(f"[#] MAC Address of {ip} is {mac}")
    print("[*] Entry added to ARP cache.")

# ---------- RARP Process ----------
def rarp_request(mac):
    print(f"\n[*] Searching ARP cache for MAC {mac}...")
    sleep(0.7)

    if not validate_mac(mac):
            print(f"[!] Invalid MAC address format. Please enter a valid MAC (e.g.,
00:1A:2B:3C:4D:5E).")
        return

    for ip, mac_addr in arp_cache.items():
        if mac_addr.lower() == mac.lower():
            print(f"[+] Reply received.")
            sleep(0.5)
            print(f"[#] IP Address of {mac} is {ip}")
            return

    print(f"[!] MAC {mac} not found in cache.")
    sleep(0.6)
    print("[*] Sending RARP Request to network server...")
```

```
    sleep(1.2)
    ip = f"192.168.1.{random.randint(16, 99)}"
    arp_cache[ip] = mac.upper()
    print(f"[+] Reply received from RARP Server.")
    sleep(0.5)
    print(f"[#] IP Address of {mac} is {ip}")
    print("[*] Entry added to ARP cache.")

# ---------- Display Cache ----------
def display_cache():
    print("\n========== ARP CACHE TABLE ==========")
    if not arp_cache:
        print("Cache is empty.")
        return
    print(f"{'IP Address':<20} {'MAC Address'}")
    print("-" * 40)
    for ip, mac in arp_cache.items():
        print(f"{ip:<20} {mac}")
    print("=====================================")

# ---------- Main Menu ----------
while True:
    print("\n========== ARP / RARP MENU ==========")
    print("1. ARP  (Find MAC Address from IP)")
    print("2. RARP (Find IP Address from MAC)")
    print("3. Display ARP Cache Table")
    print("4. Exit")
    print("=====================================")

    try:
        choice = int(input("Enter your choice (1-4): ").strip())
    except ValueError:
        print("Invalid input! Please enter 1–4.")
        continue
```

```
if choice == 1:
    target_ip = input("Enter Target IP Address (e.g., 192.168.1.10): ").strip()
    arp_request(target_ip)

elif choice == 2:
    target_mac = input("Enter MAC Address (e.g., 00:1A:2B:3C:4D:5E): ").strip()
    rarp_request(target_mac)

elif choice == 3:
    display_cache()

elif choice == 4:
    print("\nExiting... Goodbye!")
    break

else:
    print("Invalid choice! Please select between 1 and 4.")
```

CHAPTER-4

TEST CASES/ OUTPUT

To ensure the correct functionality of the "ARP and RARP Simulation", several test cases were designed and executed. Each test case verifies a specific functionality of the system such as performing ARP requests, executing RARP lookups, displaying the ARP cache, and handling invalid or repeated inputs.

The interaction between the user and the program occurs through a command-line interface, simulating how devices in a network perform address resolution. The outputs are displayed on the console in a step-by-step format that represents actual ARP and RARP operations, including cache checks, broadcasting requests, receiving replies, and updating cache entries.

Fig 1– Main Menu

Server actively listens for HTTP requests, displaying request logs and response statuses.

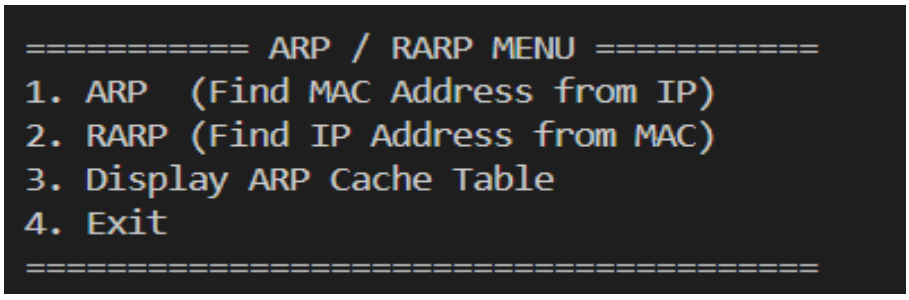

Fig 2– ARP Request (IP Found in Cache)

When the user enters an IP address already stored in the ARP cache, the program retrieves the corresponding MAC address directly.

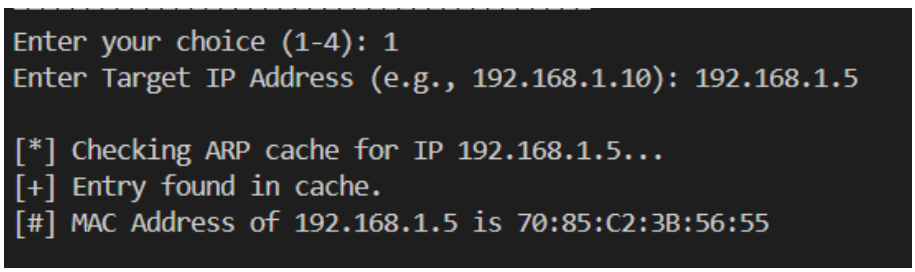

Fig 3– ARP Request (IP Not Found in Cache)

If the IP address is not present, the system simulates broadcasting an ARP request and receiving a reply, then adds the new mapping to the cache.

```
Enter your choice (1-4): 1
Enter Target IP Address (e.g., 192.168.1.10): 192.168.1.18

[*] Checking ARP cache for IP 192.168.1.18...
[!] IP 192.168.1.18 not found in cache.
[*] Broadcasting ARP Request on LAN...
[+] Reply received from 192.168.1.18.
[#] MAC Address of 192.168.1.18 is 00:16:3E:56:AC:72
[*] Entry added to ARP cache.
```

Fig 4 – RARP Request (MAC Found in Cache)

When the user enters a MAC address that exists in the ARP cache, the system displays the corresponding IP address.

```
Enter your choice (1-4): 2
Enter MAC Address (e.g., 00:1A:2B:3C:4D:5E): F4:6B:EF:88:1D:BB

[*] Searching ARP cache for MAC F4:6B:EF:88:1D:BB...
[+] Reply received.
[#] IP Address of F4:6B:EF:88:1D:BB is 192.168.1.11
```

Fig 5 – RARP Request (MAC Not Found in Cache)

If the MAC address is not found, the system simulates sending a RARP request to a server and receiving an IP address assignment.

```
Enter your choice (1-4): 2
Enter MAC Address (e.g., 00:1A:2B:3C:4D:5E): 00:1A:2B:3C:4D:5F

[*] Searching ARP cache for MAC 00:1A:2B:3C:4D:5F...
[!] MAC 00:1A:2B:3C:4D:5F not found in cache.
[*] Sending RARP Request to network server...
[+] Reply received from RARP Server.
[#] IP Address of 00:1A:2B:3C:4D:5F is 192.168.1.40
[*] Entry added to ARP cache.
```

Fig 6 – Display ARP Cache Table

Displays all current IP–MAC address mappings, including both preloaded and newly added entries.

```
Enter your choice (1-4): 3

========== ARP CACHE TABLE ==========
IP Address           MAC Address
--------------------------------------
192.168.1.1          3C:F0:11:9B:2A:11
192.168.1.2          98:5F:D3:4A:77:22
192.168.1.3          B4:6D:83:1F:23:33
192.168.1.4          54:2A:1B:7C:99:44
192.168.1.5          70:85:C2:3B:56:55
192.168.1.6          28:92:A4:6F:88:66
192.168.1.7          40:B8:37:9D:1A:77
192.168.1.8          A0:32:99:4E:9C:88
192.168.1.9          18:AF:61:2C:4E:99
192.168.1.10         9C:3D:CF:7B:54:AA
192.168.1.11         F4:6B:EF:88:1D:BB
192.168.1.12         5C:CF:7F:9E:AA:CC
192.168.1.13         B8:8A:EC:1F:CB:DD
192.168.1.14         84:C5:A6:4B:76:EE
192.168.1.15         00:1A:2B:3C:4D:5E
192.168.1.18         00:16:3E:56:AC:72
192.168.1.40         00:1A:2B:3C:4D:5F
======================================
```

Fig 7 – Invalid Entry Handling

When an invalid IP or MAC address is entered, the system displays an appropriate message and prompts the user again without terminating the program.

```
Enter your choice (1-4): 1
Enter Target IP Address (e.g., 192.168.1.10): abc.efg.ijk

[*] Checking ARP cache for IP abc.efg.ijk...
[!] Invalid IP address format. Please enter a valid IP (e.g., 192.168.1.10).
```

```
Enter your choice (1-4): 2
Enter MAC Address (e.g., 00:1A:2B:3C:4D:5E): 1.2.3.4

[*] Searching ARP cache for MAC 1.2.3.4...
[!] Invalid MAC address format. Please enter a valid MAC (e.g., 00:1A:2B:3C:4D:5E).
```

Fig 8 – Exit

The user can choose to exit the simulation gracefully.

```
Enter your choice (1-4): 4

Exiting... Goodbye!
```

CHAPTER-5

RESULTS

The "ARP and RARP Simulation" project was successfully developed and executed, accurately demonstrating the working principles of address resolution protocols within a simulated Local Area Network (LAN) environment. The system efficiently simulated ARP and RARP operations, including cache checking, broadcasting requests, receiving replies, and dynamically updating the ARP cache table.

During execution, the program maintained a preloaded cache of 15 realistic IP–MAC address pairs and handled new address resolutions smoothly. When an IP or MAC address was not found in the cache, the simulation correctly displayed the sequence of steps—broadcast request, reply reception, and cache update—mimicking the actual network process. The results were displayed in a clear and structured console format, providing users with an authentic view of how ARP and RARP operate in real networks.

The system performed consistently across all tested scenarios, including valid lookups, missing entries, and repeated requests. The ARP cache updated dynamically after each resolution, and all generated IP and MAC addresses followed valid formatting conventions. Input validation ensured that incorrect or malformed entries were handled gracefully without interrupting the program flow.

Overall, the "ARP and RARP Simulation" effectively achieved its objective of modeling real network address resolution behavior through software-based simulation. It provided accurate, step-by-step output that reflects real-world ARP and RARP processes, offering a practical and educational demonstration of core networking concepts in a controlled, offline environment.

CHAPTER-6

CONCLUSION

The "ARP and RARP Simulation" successfully demonstrates how Python can be used to simulate the fundamental operations of the Address Resolution Protocol (ARP) and the Reverse Address Resolution Protocol (RARP) within a network environment. By utilizing Python's core features such as dictionaries, random address generation, and structured program flow, the system effectively models how IP addresses are mapped to MAC addresses and vice versa in real network communication.

Through this simulation, the program accurately replicates the logical workflow of address resolution, including cache checking, broadcasting requests, receiving replies, and updating the ARP cache dynamically. The use of realistic IP and MAC address formats, combined with detailed console outputs, provides a clear and practical understanding of how ARP and RARP function in a Local Area Network (LAN).
This project fulfills its objective of creating an educational tool that helps learners visualize how devices communicate before data transmission occurs. It can be further enhanced by incorporating GUI-based visualization or real packet-level interaction using Python libraries such as Scapy. These extensions would allow for a more interactive and in-depth exploration of protocol behavior.

In conclusion, the "ARP and RARP Simulation" stands as a simple yet effective demonstration of how network address resolution takes place. It bridges theoretical networking concepts with practical implementation, offering a valuable learning experience for understanding the essential role of ARP and RARP in computer networking and local communication systems.

REFERENCES

[1]. Chatgpt – www.chatgpt.com

[2]. GeeksforGeeks. *Address Resolution Protocol (ARP) in Computer Network*. – https://www.geeksforgeeks.org/ethical-hacking/ARP

[3]. Python Documentation. *The Python Standard Library — time, random, and os* – https://docs.python.org/3/library/

[4]. GeeksforGeeks, Reverse Address Resolution Protocol – RARP – https://www.geeksforgeeks.org/computer-networks/what-is-rarp/

[5]. Dheeraj Khandelwal, "Lecture 13: Address Resolution Protocol (ARP) & Reverse Address Resolution Protocol (RARP)", Department of CSE, IIT Kharagpur – https://www.cse.iitk.ac.in/users/dheeraj/cs425/lec13.html

[6]. TutorialsPoint, difference between ARP and RARP – https://www.tutorialspoint.com/difference-between-arp-and-rarp