MP2 - ARP Protocol Simulation Implementing Address Resolution Protocol (ARP) in Python and Packet Tracer

Diana Emal - 300210204
Marwa Alsalameh - 300185722
Nicholas Retel - 300196657
COMP 390: Data Communication
Rania Mohammed
Nov 17th, 2025

I. Abstract

This project implements a simulation of the Address Resolution Protocol (ARP), a fundamental network protocol that maps IP addresses to MAC addresses within a local area network. The implementation was achieved using Python socket programming for backend logic and Cisco Packet Tracer for visual simulation of the ARP request/reply process. The main outcome of the project is a functioning ARP resolver that demonstrates the communication between a client and a server to map network layer addresses to data link layer addresses.

II. Introduction

The Address Resolution Protocol (ARP) is a core process in network communication, responsible for resolving IP addresses into corresponding MAC (Media Access Control) addresses. This translation allows devices on the same network to communicate directly. Without ARP, local communication between devices would be impossible because network devices use MAC addresses at the data link layer to send frames. The goal of this mini-project is to implement ARP using Python sockets to simulate the request/reply process and visualize the same using Packet Tracer. The implementation highlights how ARP works at a conceptual level, demonstrating how a device requests the MAC address corresponding to a given IP address and updates its ARP table upon receiving a reply.

III. Related Work and Literature Review

To understand modern advancements and optimization in ARP implementations, three recent research papers were reviewed.

A. Enhancing ARP Security in Software Defined Networks

Buzura et al. [1] propose a modular software architecture to prevent ARP-spoofing attacks in the data plane of software defined networks (SDNs). Their system places configurable modules directly on switches to monitor ARP traffic, detect suspicious packets, enforce correct ARP–MAC mappings, and block or quarantine malicious hosts. The design is flexible, allowing adaptation to different network setups and attack scenarios, including distributed man-in-the-middle attempts. Tests in Mininet using Open vSwitch showed faster mitigation compared to controller only solutions, though response time grows with network size and traffic. The system maintains network throughput with moderate overhead, demonstrating that data plane interventions can improve ARP security, even though real world deployment and large scale performance require further study.

B. *Educational ARP Simulation and Visualization*

Putra *et al.* [2] investigate ARP-spoofing mitigation by implementing and simulating Dynamic ARP Inspection (DAI) within Cisco Packet Tracer to enhance network security. They construct a network with switches and end devices, configure DHCP Snooping to build binding tables, and enable DAI on untrusted ports to validate ARP packets. Simulations under normal and attack scenarios show that with DAI enabled, only legitimate ARP packets are forwarded, effectively preventing ARP-spoofing attacks and man-in-the-middle attempts. The study also reports that enabling DAI introduces minimal latency in packet forwarding while maintaining network connectivity, demonstrating that security enforcement does not significantly degrade performance. Furthermore, the results indicate that DAI combined with DHCP Snooping effectively blocks forged ARP replies from malicious hosts, highlighting the importance of correct port trust configuration. Overall, the paper confirms that DAI is a practical, low overhead security mechanism suitable for small to medium sized networks and provides a safe environment for testing configurations before real world deployment.

C. Controller Based ARP Spoofing Detection in SDNs

Hnamte and Hussain [3] developed a framework for mitigating ARP-spoofing attacks within software-defined networks (SDNs). Their model incorporates dynamic ARP table management and real time monitoring of network traffic to detect and isolate malicious hosts. The system uses adaptive flow rules orchestrated by the SDN controller to block suspicious ARP activity while maintaining efficient network performance. Evaluation results show high detection accuracy and low computational overhead, demonstrating scalability across networks of varying sizes. The main strength of this work lies in its real time, controller based approach to ARP security, though it requires SDN compatible infrastructure and has been tested primarily in simulated environments. Compared with previous studies, this research emphasizes adaptive mitigation and automated response rather than visualization or pedagogy, highlighting how ARP management can be optimized for modern programmable networks.

D. Critical Comparison:

Together, these studies illustrate complementary strategies for addressing ARP challenges in modern networks. Hnamte and Hussain [3] focus on enhancing ARP security in SDNs through controller based monitoring, adaptive flow rules, and real time detection, emphasizing automated mitigation and scalability. Putra et al. [2] present a simulation based implementation of Dynamic ARP Inspection in Cisco Packet Tracer, providing an educational and practical environment for understanding ARP vulnerabilities and mitigation mechanisms with low performance overhead. Buzura et al. [1] introduce a data-plane-level, modular architecture for ARP-spoofing mitigation that prioritizes performance optimization and extendability, demonstrating reduced latency and reliable cache management in SDN environments. Each study balances trade offs among complexity, infrastructure requirements, and applicability: Hnamte & Hussain require SDN controllers and DNN-based detection, Buzura et al. rely on SDN-compatible switches and data-plane modules, while Putra et al. focus on easily deployable simulations with pedagogical value. Collectively, these works highlight that ARP management remains a critical component for both network security and learning, with the current project aligning most closely with Putra et al. [2] by providing an interactive, educational tool for exploring ARP interactions before progressing to advanced SDN security and performance topics.

IV. Implementation

The implementation was divided into two main components:

- Python-based ARP Simulation (Client-Server Model)
- Packet Tracer Visualization

A. Tools Used

- Python 3.x for implementing socket-based communication
- Cisco Packet Tracer for ARP visualization
- Localhost (127.0.0.1) network simulation environment

B. Server Design

The arp_server.py script initializes a server socket on 127.0.0.1:12345 and maintains a predefined mapping (NETWORK dictionary) of IP addresses to MAC addresses. When a client sends an IP address, the server checks its dictionary and sends the corresponding MAC address as a response. If the IP is not found, it returns "Not Found".

Key steps:

a. Initialize server and listen for connections.
b. Wait for an ARP request (IP query).
c. Retrieve corresponding MAC from NETWORK.
d. Send the MAC back as an ARP reply.

C. Client Design

The arp_client.py script acts as the ARP requester. It prompts the user to input an IP address and sends the query to the server. The server's reply is then displayed and stored in an ARP table dictionary for future lookups.

Workflow:

a. User enters target IP.
b. Client connects to the server and sends IP.
c. Receives MAC and updates ARP table.
d. Prints updated ARP table after each query.

D. Packet Tracer Simulation

The ARP_simulator.pkt file represents a small LAN topology with PCs and a switch. The ARP request is visualized as a broadcast frame sent by one host to all others, with the destination MAC address FF:FF:FF:FF:FF:FF. The target device responds with its MAC address directly, completing the resolution process.

V. Results and Discussion

The system successfully demonstrates the ARP request and reply process.

- **Functionality:** IP addresses entered on the client side are resolved to corresponding MAC addresses stored on the server.
- **Scalability:** The program can easily be extended by adding more entries to the NETWORK dictionary.
- **Accuracy:** The mapping accurately reflects the behavior of ARP in real networks.
- **Visualization:** Packet Tracer simulation confirmed

that the ARP broadcast and reply packets behaved as expected.

VI. Conclusion

This project successfully implemented and demonstrated the Address Resolution Protocol using Python and Packet Tracer. The Python socket based simulation provides a simple yet effective way to understand the client-server mechanism behind ARP communication. Combined with Packet Tracer visualization, the project provides both theoretical and practical understanding of how ARP functions in real world networking.

Future Work: Future versions can include ARP cache expiration, error handling for spoofing prevention, and integration with real network interfaces to observe live traffic.

VII. References

[1] S. Buzura, M. Lehene, B. Iancu, and V. Dadarlat, “An extendable software architecture for mitigating ARP spoofing-based attacks in SDN data plane layer,” *Electronics*, vol. 11, no. 13, p. 1965, 2022. Doi: 10.3390/electronics11131965.

[2] F. P. E. Putra, A. B. Ubaidi, A. B. Tamam, and R. W. Efendi, “Implementation and simulation of Dynamic ARP Inspection in Cisco Packet Tracer for network security,” *Brilliance: Research of Artificial Intelligence*, vol. 4, no. 1, pp. 340–347, 2024. doi: 10.47709/brilliance.v4i1.4199.

[3] V. Hnamte and J. Hussain, “Enhancing security in software-defined networks: An approach to efficient ARP spoofing attacks detection and mitigation,” *Telematics and Informatics Reports*, vol. 14, p. 100129, 2024. doi: 10.1016/j.teler.2024.100129.