Computer Networks (BCSCN501)

Open Ended Experiment

On
"Implementation of address resolution protocol"

Submitted By:

Yashwanth S	**4VV22CS189**
Venkatesha Y K	**4VV22CS181**
Yesh Gowda	**4VV22CS190**

Submitted To:
Prof. Nitish A
Department of
Computer Science and Engineering
VVCE

Abstract :

Address Resolution Protocol (ARP) is an integral component of the TCP/IP protocol suite, bridging the gap between the network and data link layers. By enabling seamless translation of logical IP addresses to physical MAC addresses, ARP ensures efficient data routing within local area networks. The protocol's functionality is vital in scenarios where devices need to communicate across different layers of the OSI model. This report delves into the operational flow of ARP, its message format, and practical implementation. Key considerations, such as ARP cache optimization and handling broadcast storms, are also addressed. Furthermore, we evaluate ARP's performance and discuss its role in enhancing network reliability and scalability. The report concludes by exploring real-world applications and potential improvements to the protocol.

Important ARP Terms

- **ARP Cache :-** After receiving the MAC address, ARP passes it to the sender where it is stored in a table for future reference. And this is called ARP Cache which is later used to obtain the MAC address.

- **ARP Cache Timeout :-** This is the time in which the MAC address can remain in the ARP Cache.

- **ARP request :-** Broadcasting a packet over the network to verify whether we have arrived at the destination MAC address.

- **ARP response/reply :-** It is a MAC address response that the sender receives from the receiver which helps in further communication of data.

Types of ARP

There are four types of ARP protocol they are as follows:-

1. Proxy ARP
2. Gratuitous ARP
3. Reverse ARP
4. Inverse ARP

ARP Protocol Working:

a. ARP request is broadcast

b. ARP reply is unicast
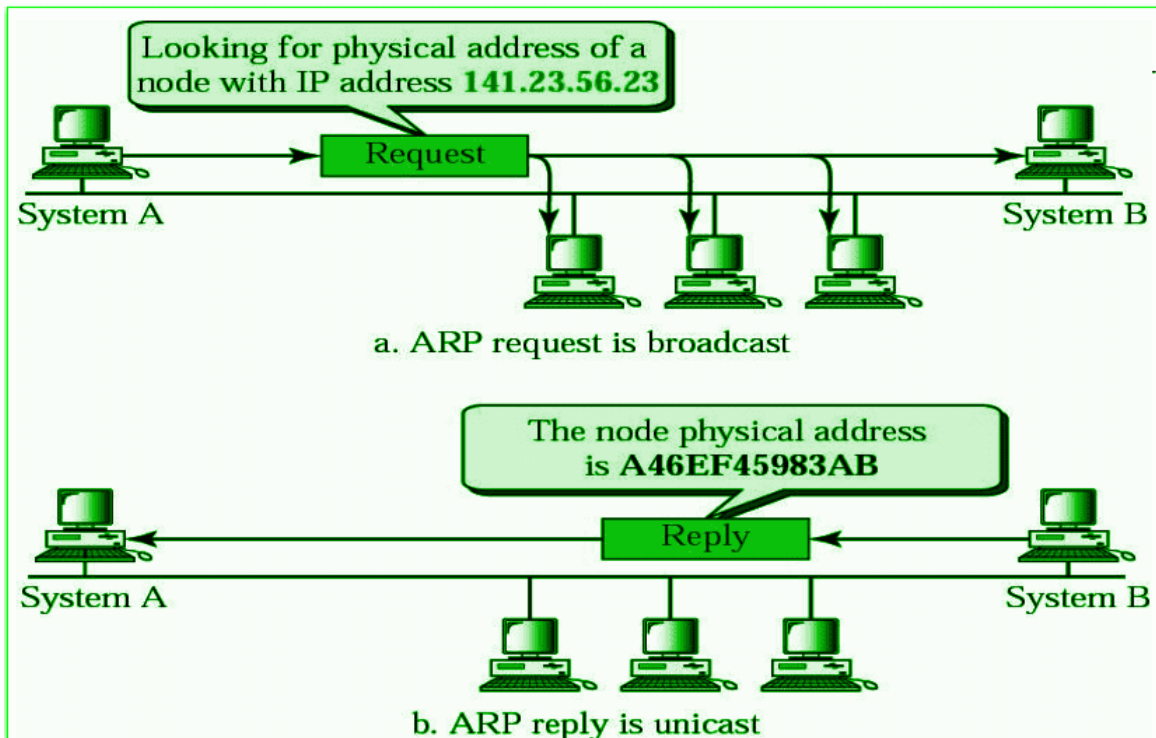
Below is the working of address resolution protocol is being explained in some steps :-

- When a sender wants to communicate with a receiver, the sender first checks its ARP cache. Sender checks whether the receiver's MAC address is already present in the ARP cache or not?

- If the receiver's MAC address is already present in the ARP cache, the sender will communicate with the receiver using that MAC address.

- If the MAC address of the receiver device is not already present in the ARP cache, then in such a situation an ARP request message is prepared by the sender device. This message contains the MAC address of the sender, IP address of the sender and IP address of the receiver. The field containing the MAC address of the receiver is left blank because it is being searched.

- Sender device broadcasts this ARP request message in the LAN. Because this is a broadcast message, every device connected to the LAN receives this message.

- All devices match the receiver IP address of this request message with their own IP address. Devices whose IP address does not match drop this request message.

- The device whose IP address matches the receiver IP address of this request message receives this message and prepares an ARP reply message. This is a unicast message which is sent only to the sender.

- In ARP reply message, the sender's IP address and MAC address is used to send the reply message. Besides, in this message the receiver also sends its IP address and MAC address.

- As soon as the sender device receives this ARP reply message, it updates its ARP cache with the new information (Receiver's MAC address). Now the MAC address of the receiver is present in the ARP cache of the sender. The sender can send and receive data without any problem.

Code:

```python
class ARP:

  def __init__(self):

    self.arp_table = {}  # Dictionary to simulate ARP cache


  def add_entry(self, ip_address, mac_address):

    """Add an IP-MAC mapping to the ARP table."""

    self.arp_table[ip_address] = mac_address


  def resolve(self, ip_address):

    """Resolve an IP to MAC address."""

    if ip_address in self.arp_table:
```

```
        return f"MAC Address for {ip_address}: {self.arp_table[ip_address]}"

    else:

        return f"No MAC Address found for {ip_address}. Broadcasting ARP request."


# Example usage

arp = ARP()

arp.add_entry("192.168.1.1", "00:1A:2B:3C:4D:5E")

arp.add_entry("192.168.1.2", "00:1A:2B:3C:4D:5F")

print(arp.resolve("192.168.1.1"))  # Found in ARP table

print(arp.resolve("192.168.1.3"))  # Not found, broadcast ARP request
```

Output:

MAC Address for 192.168.1.1: 00:1A:2B:3C:4D:5E

No MAC Address found for 192.168.1.3. Broadcasting ARP request.

Time Complexity Analysis with Graph:

Time Complexity Analysis

- **Insertion:** Adding an entry to the ARP table is **O(1)** since it uses a dictionary (hash map).

- **Lookup:** Resolving an IP address to a MAC address is **O(1)** due to the dictionary-based structure.

- **Broadcast:** In worst-case scenarios (IP not found), the protocol broadcasts a request, making it **O(n)**, where nnn is the number of devices in the LAN.

```python
import matplotlib.pyplot as plt

devices = list(range(1, 101))  # Number of devices

lookup_time = [1 for _ in devices]  # Constant O(1)

broadcast_time = [n for n in devices]  # Linear O(n)


plt.plot(devices, lookup_time, label="Lookup (O(1))")

plt.plot(devices, broadcast_time, label="Broadcast (O(n))")

plt.xlabel("Number of Devices")

plt.ylabel("Time Complexity")

plt.title("ARP Time Complexity Analysis")

plt.legend()

plt.show()
```

Output:
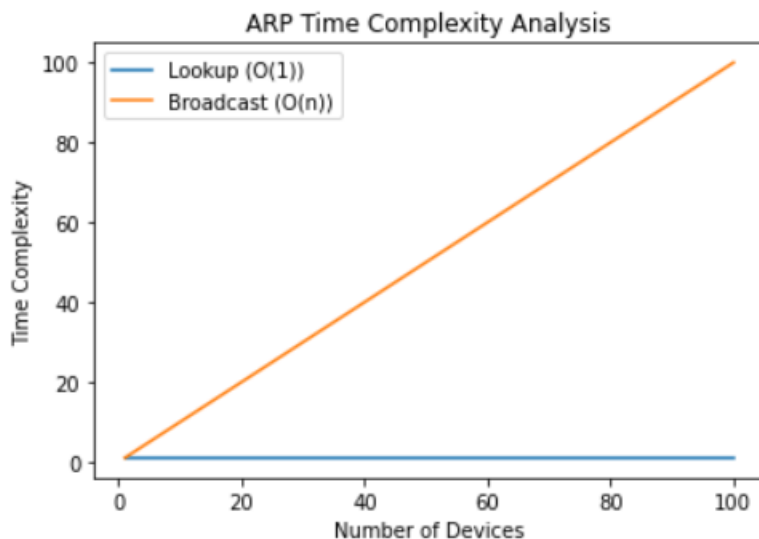
Conclusion with Applications:

Conclusion

The implementation demonstrates how ARP efficiently resolves IP addresses to MAC addresses in a network. While hash tables ensure fast lookups, broadcast operations can become bottlenecks in larger networks. Optimizing ARP, especially for high-density environments, is crucial to maintaining network performance.

Applications

1. **Local Area Networks (LAN):**
   ARP is essential for device communication within LANs.

2. **Dynamic Host Configuration Protocol (DHCP):**
   ARP is used to verify IP address uniqueness during lease assignments.

3. **Networking Tools:**
   Network diagnostic tools, such as ARP scanners, rely on ARP for troubleshooting.

4. **Switching and Routing:**
   ARP enables layer-3 devices to locate devices within the same subnet.