# Wireshark Project: ARP, TCP Handshake, and DNS Analysis

## Executive Summary

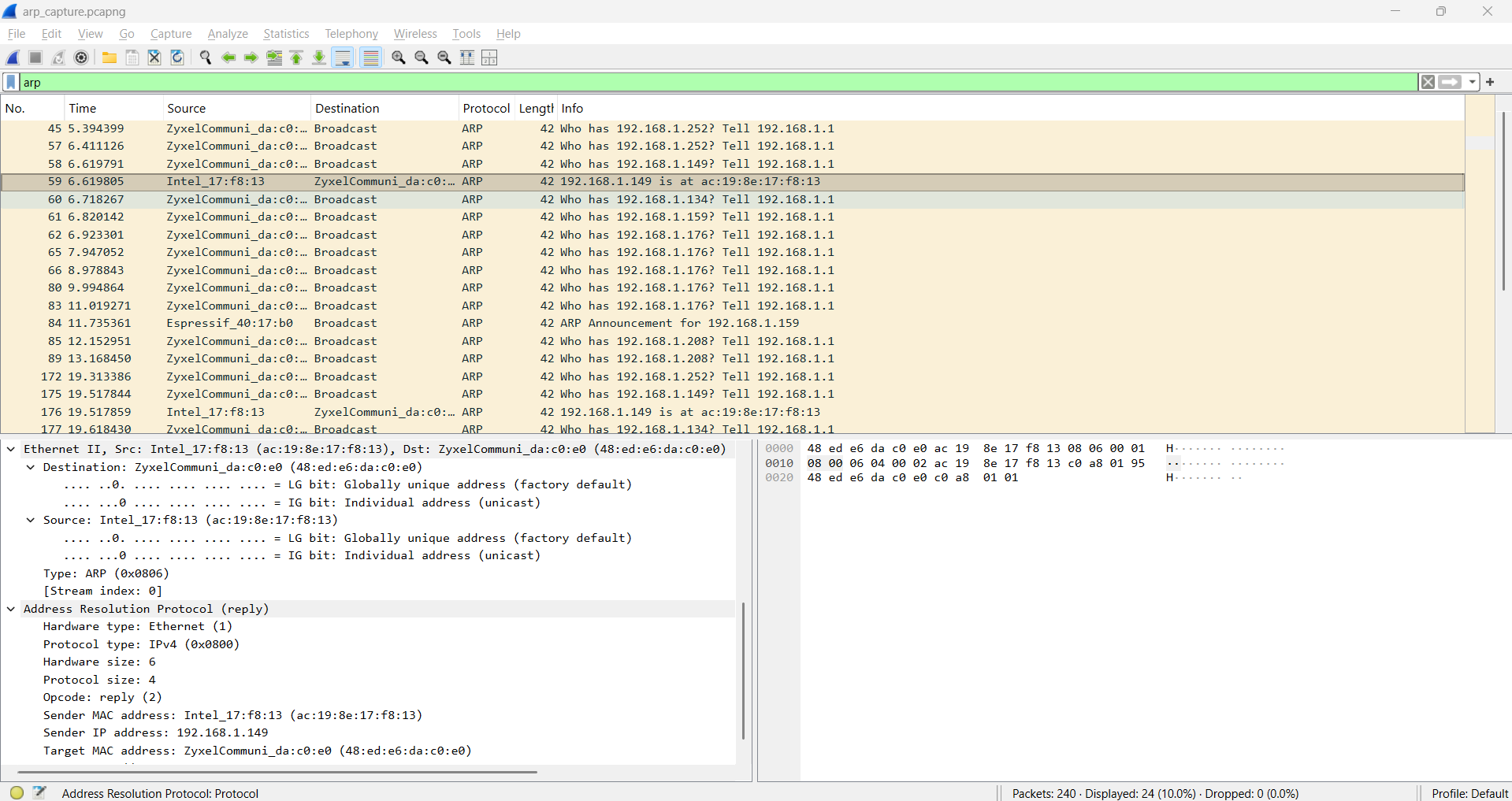
In this project, Wireshark was used to capture and analyze three core networking processes: Address Resolution Protocol (ARP), the TCP 3-Way Handshake, and DNS queries/responses. Each section demonstrates packet-level communication at different layers of the OSI model, providing insights into how local addressing, reliable session establishment, and domain name resolution function in practice.

## Methodology

1. Identified the active interface (Wi-Fi) using 'ncpa.cpl' on Windows.  
2. Used capture/display filters in Wireshark to isolate ARP, TCP, and DNS traffic.  
3. Generated network traffic with system commands:  
 - ARP: 'arp -d \*' followed by 'ping 192.168.1.1'.  
 - TCP Handshake: Accessed 'http://neverssl.com'.  
 - DNS: Ran 'nslookup neverssl.com 8.8.8.8'.  
4. Stopped and saved captures for analysis.  
5. Annotated and interpreted packet fields including MAC addresses, sequence/ack numbers, and DNS records.

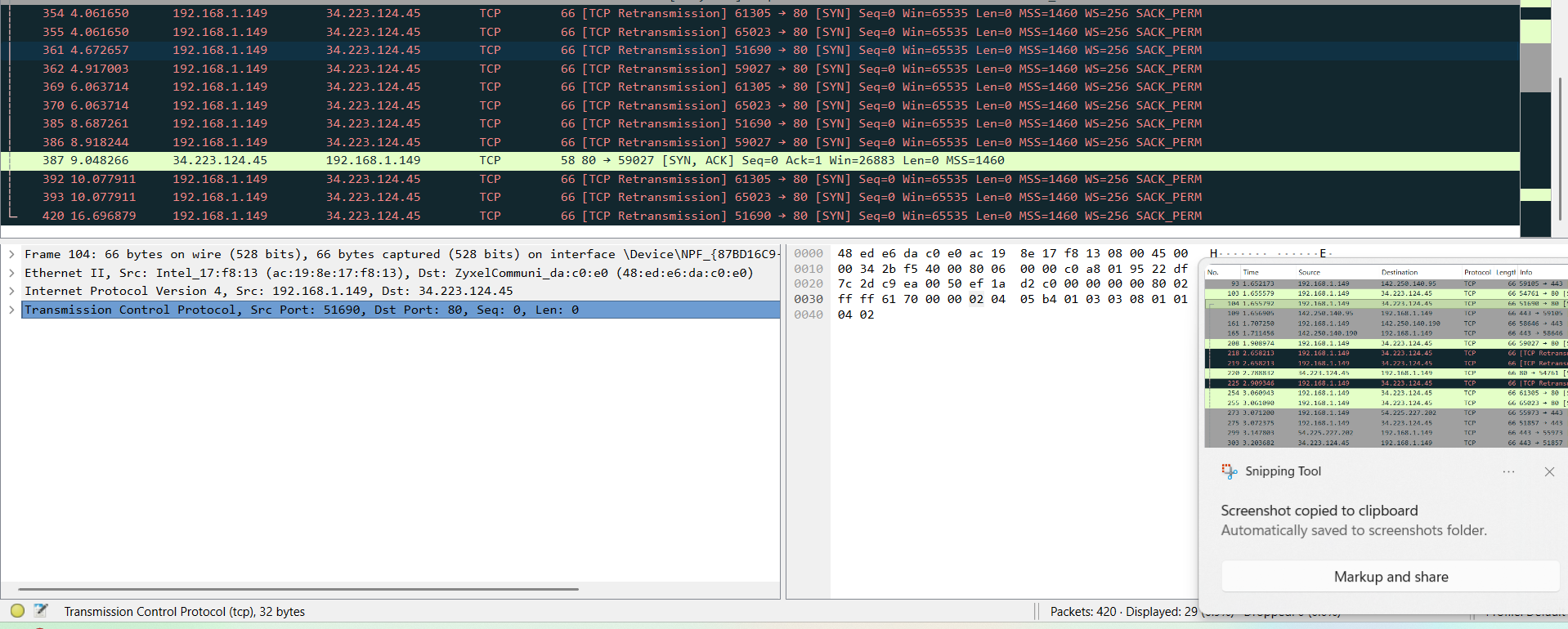
## ARP Capture & Analysis

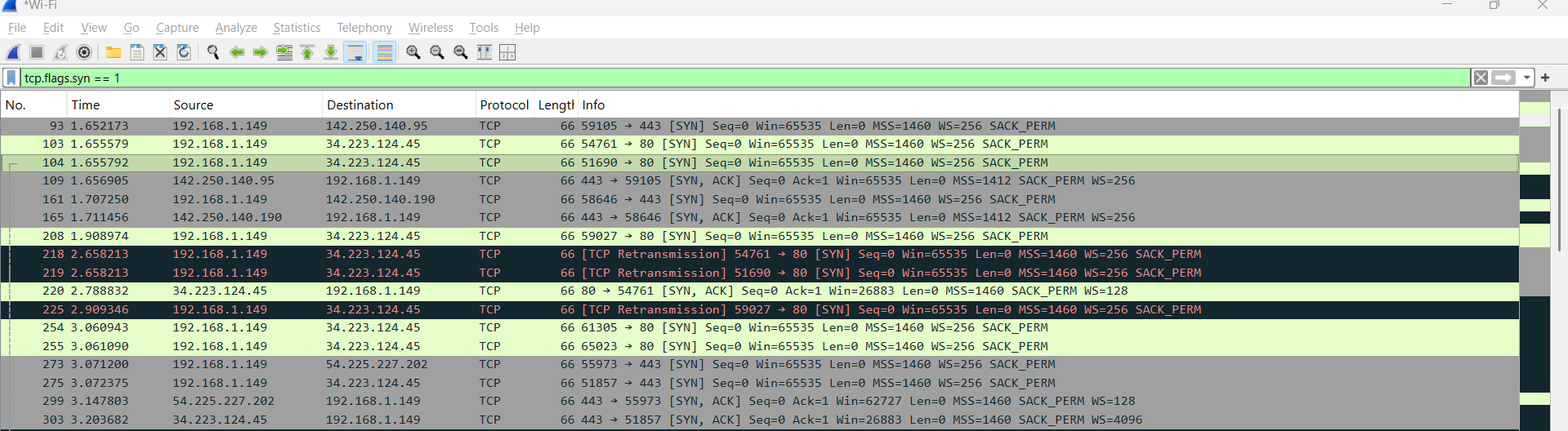
To capture ARP traffic, the ARP cache was cleared using 'arp -d \*' and the gateway was pinged (192.168.1.1). This forced the host to resolve the gateway's MAC address.  
  
In the ARP captures, host 192.168.1.1 broadcast a request asking “Who has 192.168.1.149?”. This request was sent to the broadcast address ff:ff:ff:ff:ff:ff. Host 192.168.1.149 (the local machine) replied with “192.168.1.149 is at ac:19:8e:17:f8:13”, providing its MAC address to the router. This reply was unicast to the router’s MAC.  
  
This sequence demonstrates how ARP resolves an IP address to a hardware address on a local network. Each ARP request and reply clearly shows the sender and target IP/MAC addresses.



## TCP 3-Way Handshake

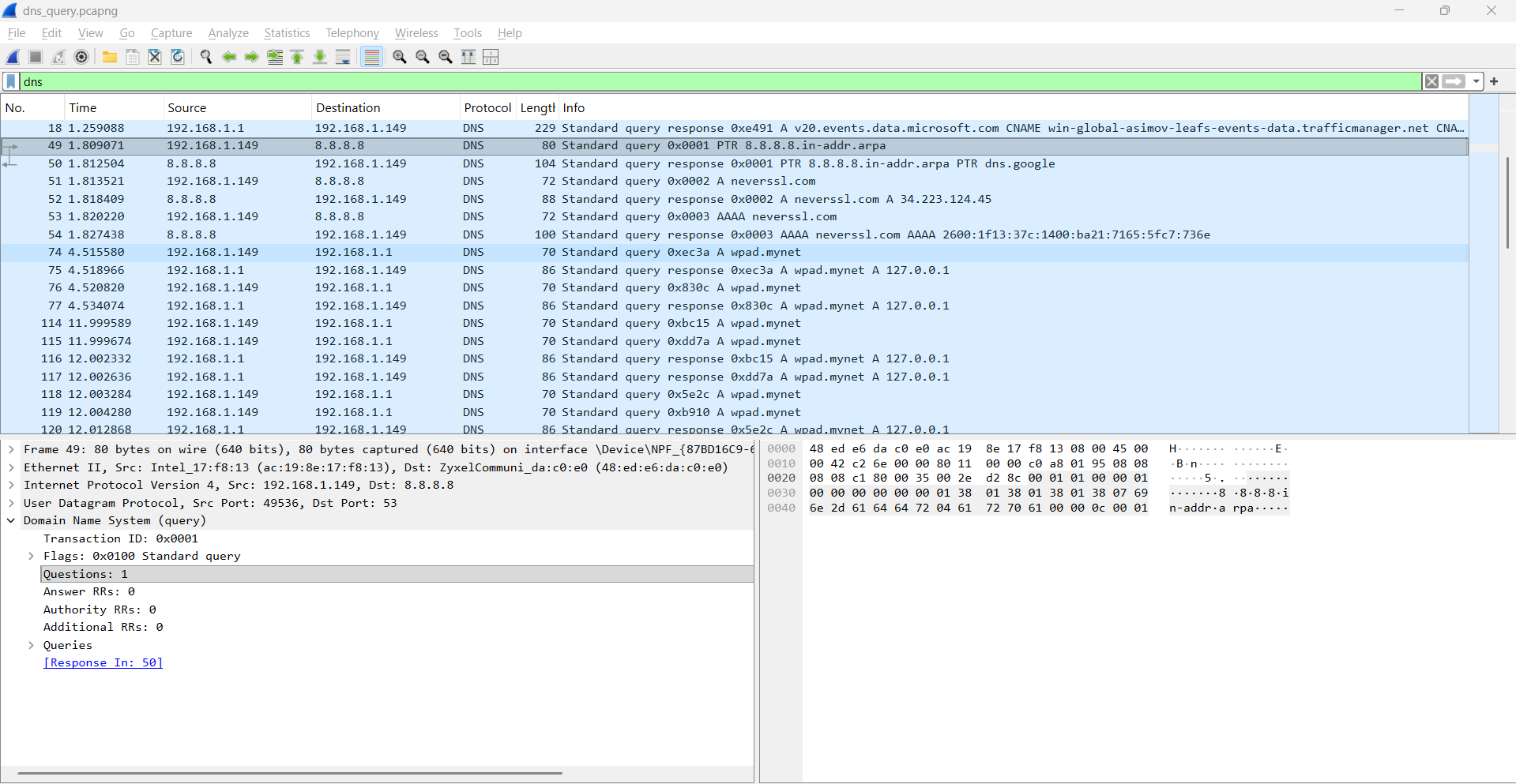
The TCP handshake was captured by applying the filter 'tcp.flags.syn==1' and accessing 'http://neverssl.com'.  
  
To initiate a connection to neverssl.com (34.223.124.45), the host (192.168.1.149) sent a TCP SYN to port 80 with an initial sequence number of 0. The packet included options such as MSS=1460, Window Scale=256, and SACK permitted.  
  
The server replied with a SYN-ACK, acknowledging the client ISN+1 and providing its own ISN. Wireshark also showed that this packet acknowledged the previous SYN.  
  
Finally, the host sent an ACK acknowledging the server’s ISN+1, completing the 3-way handshake. Several retransmissions occurred before the handshake succeeded, likely due to wireless packet loss.  
  
This sequence demonstrates the reliable session establishment of TCP.





## DNS Query & Response

DNS traffic was captured by applying the 'dns' filter and using the 'nslookup' command: 'nslookup neverssl.com 8.8.8.8'.  
  
Frame 51 (DNS Query)  
- Source: 192.168.1.149 → 8.8.8.8  
- Transaction ID: 0x0001  
- Question: A neverssl.com  
  
Frame 52 (DNS Response)  
- Source: 8.8.8.8 → 192.168.1.149  
- Same Transaction ID (0x0001, matches query)  
- Answer: neverssl.com A 34.223.124.45  
- Also contains an AAAA record: 2600:1f13:37c:1400:ba21:7165:5fc7:736e (IPv6)  
  
The DNS resolver responded with both IPv4 and IPv6 addresses. The TTL field indicated caching duration, and the RA (Recursion Available) flag showed the resolver performed recursion. This highlights DNS’s role in translating domain names into usable IP addresses.



## Conclusion

This project demonstrated packet-level analysis of ARP, TCP handshakes, and DNS. At Layer 2, ARP resolved IP-to-MAC mappings for communication within the LAN. At Layer 4, TCP’s 3-way handshake reliably established a session with negotiated options. At Layer 7, DNS translated human-readable domain names into IP addresses. Together these examples illustrate the interaction of multiple OSI layers in real-world networking.