COMP 7003
Assignment 2
Design

Parth Chaturvedi
A01256537
09/29/2025

1. Purpose

This program captures network packets using Scapy and manually parses protocol headers from hexa data. It supports some network protocols like ARP, IPv4, IPv6, ICMP, ICMPv6, TCP, UDP, DNS.

Key Operations:

- Capture packets from specified interface
- Convert raw packet data to hexadecimal strings
- Parse protocol headers manually from hex data
- Display parsed fields in both hex and decoded formats
- Route packets through appropriate protocol parsers

---

2. Data Structures

Global Variables

Variable	Type	Purpose
`packet_counter`	integer	Tracks number of captured packets
`stop_event`	Event	Signal to stop packet capture
`global_packet_limit`	integer	Maximum packets to capture

Command Line Arguments

Argument	Type	Description
`-i / --interface`	string	Network interface to capture on
`-c / --count`	integer	Number of packets to capture
`-f / --filter`	string	Filter expression (optional)

---

3. Functions

Main Module (`main.py`)

Function	Purpose
`packet_callback(packet)`	Process each captured packet
`interface_is_loopback(interface)`	Check if interface is loopback
`has_global_ip(interface)`	Check if interface has valid IP
`capture_packets(interface, filter)`	Capture packets on single interface
`capture_on_all_interfaces(filter, count)`	Capture on multiple interfaces

Parser Module (`packet_parsers.py`)

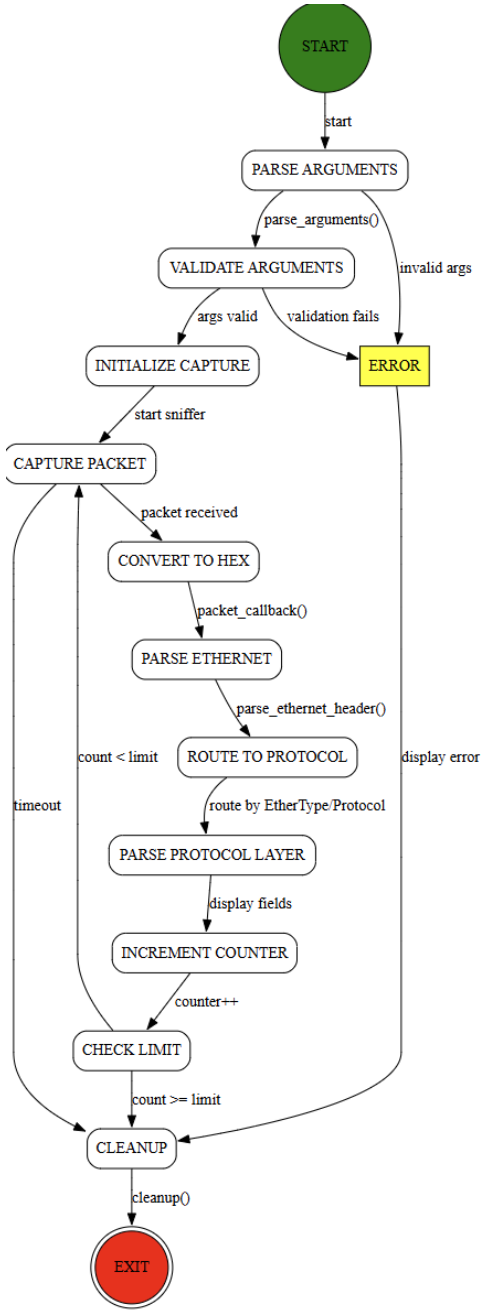
Function	Purpose
`parse_ethernet_header(hex_data)`	Parse Ethernet frame and route to next layer
`parse_arp_header(hex_data)`	Parse ARP protocol fields
`parse_ipv4_header(hex_data)`	Parse IPv4 header, route to transport layer
`parse_ipv6_header(hex_data)`	Parse IPv6 header, route to transport layer
`parse_icmp_header(hex_data)`	Parse ICMP fields
`parse_icmpv6_header(hex_data)`	Parse ICMPv6 fields
`parse_tcp_header(hex_data)`	Parse TCP fields
`parse_udp_header(hex_data)`	Parse UDP fields

`parse_dns_header(hex_data)`	Parse DNS fields

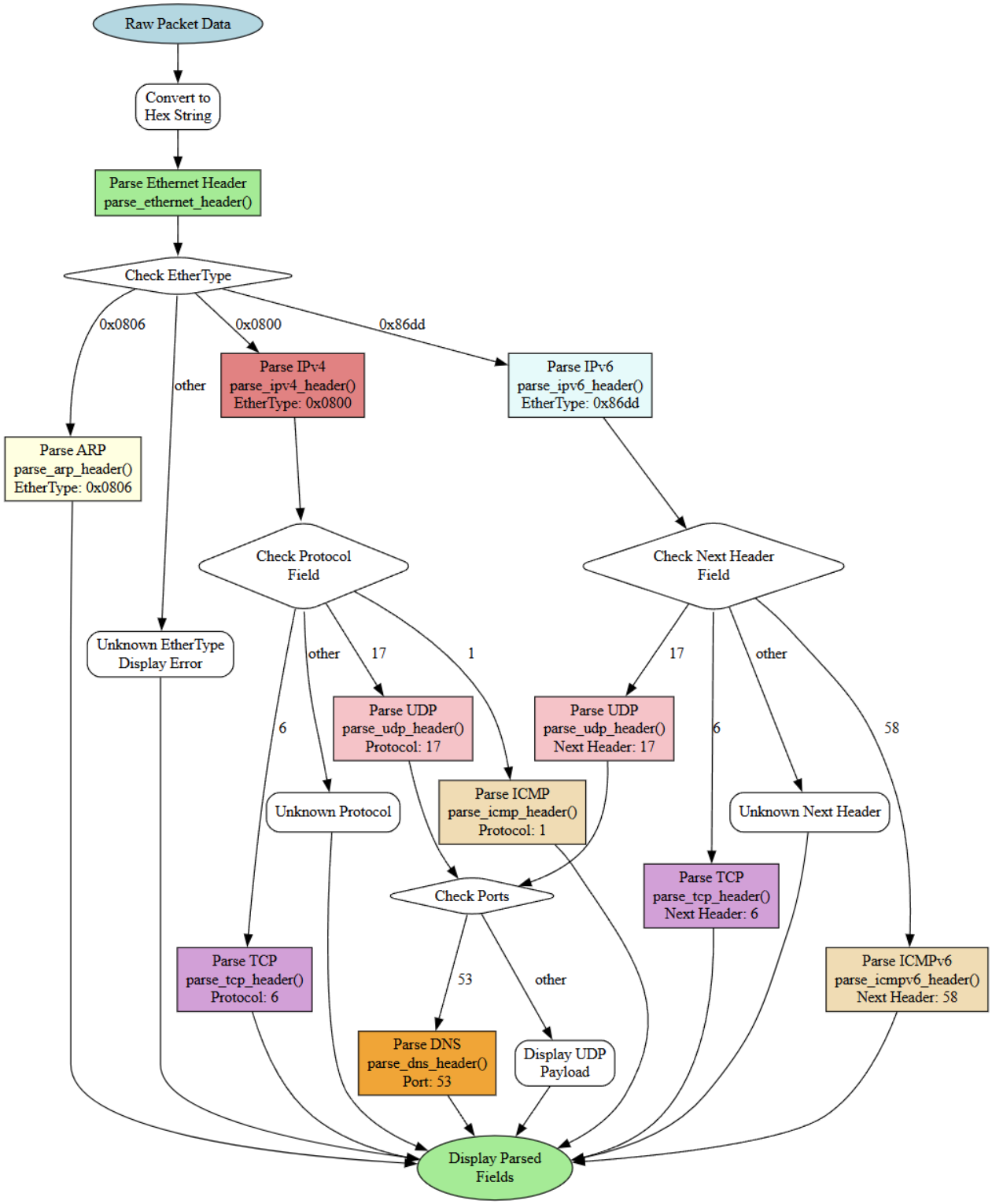
---

4. Program Flow

Main Execution Flow



Packet Parsing Flow



5. Protocol Parsing Logic

Ethernet Header Parsing

Input: Hex string with Ethernet frame
Process:

1. Extract destination MAC (bytes 0-5)
2. Extract source MAC (bytes 6-11)
3. Extract EtherType (bytes 12-13)
4. Route based on EtherType value

Output: Display MAC addresses and EtherType, route to next parser

IPv4 Header Parsing

Input: Hex string with IPv4 header
Process:

1. Extract version (4 bits)
2. Extract header length (4 bits) → multiply by 4 for bytes
3. Extract total length (2 bytes)
4. Extract flags and fragment offset (2 bytes)
5. Extract protocol field (1 byte)
6. Extract source IP (4 bytes)
7. Extract destination IP (4 bytes)
8. Calculate payload offset using header length
9. Route based on protocol value

Output: Display all IPv4 fields and route to the transport layer

TCP Header Parsing

Input: Hex string starting with TCP segment
Process:

1. Extract source/destination ports (2 bytes each)
2. Extract sequence/acknowledgment numbers (4 bytes each)
3. Extract data offset (4 bits) → multiply by 4
4. Extract reserved field (4 bits)
5. Extract flags byte (8 bits) → parse individual flags
6. Extract window size, checksum, urgent pointer
7. Calculate payload offset

Output: Display all TCP fields including individual flags

DNS Header Parsing

Input: Hex string starting with DNS header
Process:

1. Extract transaction ID (2 bytes)
2. Extract flags field (2 bytes)
3. Parse individual flag bits from flags field
4. Extract question/answer/authority/additional counts
5. Display header fields

Output: Display DNS header with decoded flags

---

6. Key Design Decisions

Hex String Approach

- All parsing done on hex strings, not binary data
- Each byte = 2 hex characters
- Enables consistent string slicing for field extraction

Protocol Routing

- Each parser responsible for routing to next layer
- Based on protocol-specific identifier fields
- Creates clean separation of concerns

Error Handling

- Checks for loopback and interfaces without IPs
- Graceful timeout if no packets match filter