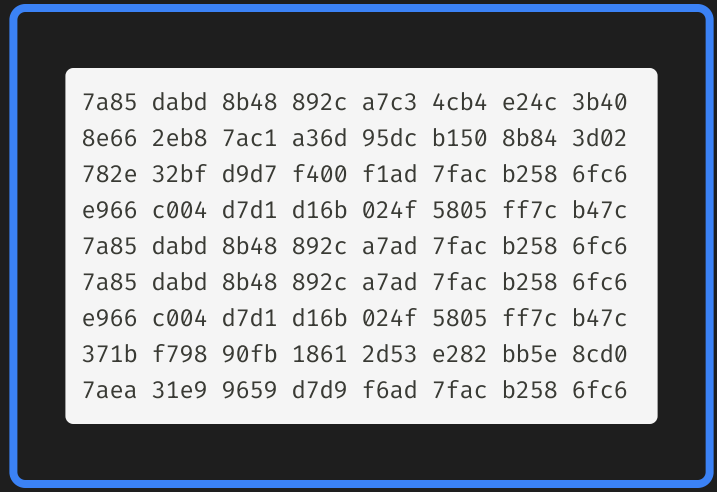
Network Protocols

Protocols, handshakes, and packet analysis

Start Week 5 →



Quiz

- Available on Elms
- Mostly multiple choice
- Work on your own
- No internet or notes are allowed
- Good luck

Open Systems Interconnection (OSI) Model

How do I transmit unstructured data?

The **physical layer** is responsible for the transmission of raw, unstructured data bits over a physical medium, defining the electrical, mechanical, and functional specifications for devices to connect to that medium.

- Ethernet cables
- Fiber optics
- Wireless transmitters
- Signal repeaters

How do I transfer data between two nodes?

The **data link layer** provides the functional and procedural means to transfer data between network entities and may also provide the means to detect and possibly correct errors that can occur in the physical layer.

- Ethernet Protocol
  - Hosts may be connected with ethernet cables
  - Hosts are identified by 48-bit MAC-address addresses

How do I route data across networks?

The **network layer** is responsible for packet forwarding including routing through intermediate routers, enabling any-to-any connectivity between hosts on different networks.

- IP Protocol
  - Devices are identified by a 32-bit (IPv4) or 128-bit (IPv6) address

How do I ensure data delivery?

The **transport layer** provides communication services for applications within a layered architecture of network components and protocols, ensuring reliable data transfer.

- Popular transport layer protocols use port numbers
- Different applications may have one or more open ports
- TCP Protocol
  - Connection oriented with sequencing and acknowledgements
- UDP Protocol
  - Stateless, sends data in packets without maintaining session information

How to I serve user applictions?

- The **session layer** provides the mechanism for opening, closing and managing a session between end-user application processes, including authentication and authorization.
- The **presentation layer** is responsible for the delivery and formatting of information to the application layer for further processing or display, including encryption and compression.
- The **application layer** is the OSI layer closest to the end user, which means both the OSI application layer and the user interact directly with the software application.

Packets

The fundamental transmission units of many network protocols are referred to as packets. These generally consist of two elements: a header and a payload.

- Headers specify protocol specific parameters (like packet number, destination address, payload type, etc.)
- The payload may contain data or the header for another protocol
- This convention allows for layers of encapsulation

Ethernet Frame

The Ethernet frame is the fundamental unit of data transmission in Ethernet networks.

Ethernet Frame Structure

☐ Labels ☐ Colors

```
0000     ff ff ff ff ff ff 00 50 56 c0 00 08 08 00                 .......PV.....
0010     45 00 00 3c 1a 2b 40 00 12 34 56 78                       E..<.+@..4Vx
```

Network Packet

The Internet Protocol (IP) packet provides network-layer addressing and routing.

IPv4 packet

IPv4 Packet Structure ☐ Labels ☐ Colors

```
0000     45 00 00 3c 00 00 40 00 40 01 00 00 c0 a8 01 01    E..<..@.@.......
0010     c0 a8 01 64                                        ...d
```

IPv6 Packet

IPv6 Packet Structure ☐ Labels ☐ Colors

```
0000     60 00 00 00 00 00 00 00 00 20 3a                   `........ :
```

Transport Segment

Transport layer protocols provide end-to-end communication services.

The Transmission Control Protocol (TCP) provides reliable, connection-oriented communication.

TCP Segment Structure ☐ Labels ☐ Colors

```
0000     08 00 00 50 00 00 00 01 00 00 00 00 50 02 00 00     ...P........P...
0010     00 00 00 00                                         ....
```

The User Datagram Protocol (UDP) provides connectionless, unreliable communication.

UDP Datagram Structure ☐ Labels ☐ Colors

```
0000     08 00 00 35 00 1c 00 00 12 34 56 78 9a bc de f0     ...5.....4Vx....
```

HTTP Request

The Hypertext Transfer Protocol (HTTP) request is the application-layer protocol for web communication.

HTTP Request Structure

☐ Labels ☐ Colors

```
0000     47 45 54 20 2f 20 48 54 54 50 2f 31 2e 31 0d 0a    GET / HTTP/1.1..
0010     48 6f 73 74 3a 20 65 78 61 6d 70 6c                Host: exampl
```

Ethernet, IP, TCP, and HTTP

A complete network packet showing all layers from Ethernet frame to HTTP request.

Complete Network Packet

☐ Labels ☐ Colors

```
0000     ff ff ff ff ff ff 00 50 56 c0 00 08 08 00 45 00     .......PV.....E.
0010     00 3c 00 50 00 00 40 00 40 06 00 00 c0 a8 01 01     .<.P..@.@.......
0020     c0 a8 01 64 08 00 00 50 00 00 00 01 00 00 00 00     ...d...P........
0030     50 02 00 00 00 00 00 00 47 45 54 20 2f 20           P.......GET /
0040     48 54 54 50 2f 31 2e 31 0d 0a 48 6f 73 74           HTTP/1.1..Host
0050     3a 20 65 78 61 6d 70 6c                             : exampl
```

OS Network Stack Implementation

The operating system implements the network stack across different layers, with each layer handled by specific components.

Layer	Handled By	Examples	OS Support
Application (7)	User applications	HTTP, FTP, DNS	System libraries
Transport (4)	Kernel TCP/IP stack	TCP, UDP	Network subsystem
Network (3)	Kernel IP stack	IP routing, ICMP	Routing tables
Data Link (2)	Network drivers	Ethernet, WiFi, PPP	Interface drivers
Physical (1)	Hardware/drivers	NICs, wireless adapters	Device drivers

World Wide Web

- Web servers usually serve content over HTTP/HTTPS
- Servers can be identified by IP addresses, but are more easily accessed by domain names
  - For example, `example.com`
  - The mapping from domain names to ip addresses is provided by the Domain Name System (DNS) protocol
  - Can be queried with `dig <domain name>`

Wireshark

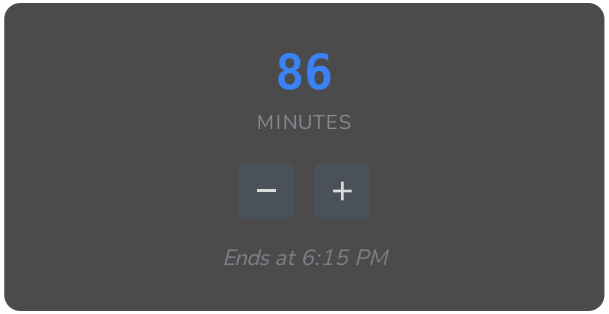
Popular tool for packet introspection.

- Walkthrough?

Lab 1

Packet analysis.

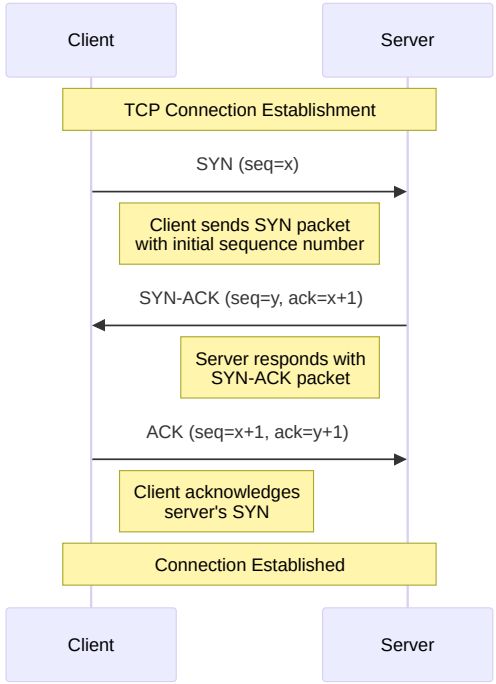
https://hacs408e.dev/labs/week-05/lab-1/



Handshakes

The TCP connection establishment process involves a three-way handshake between client and server.

- **SYN**: Client initiates connection with sequence number
- **SYN-ACK**: Server responds with its sequence
- **ACK**: Client acknowledges server's SYN to complete handshake



TCP Connection in C

Creating a TCP socket connection to a server using C system calls.

```c
#include <sys/socket.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#include <unistd.h>
#include <stdio.h>

int main() {
    // Create socket
    int sockfd = socket(AF_INET, SOCK_STREAM, 0);

    // Configure server address
    struct sockaddr_in server_addr;
    server_addr.sin_family = AF_INET;
    server_addr.sin_port = htons(80);  // HTTP port
    inet_pton(AF_INET, "93.184.216.34", &server_addr.sin_addr);  // example.com

    // Connect to server
    if (connect(sockfd, (struct sockaddr*)&server_addr, sizeof(server_addr)) < 0) {
        perror("Connection failed");
        return 1;
    }
```

HTTP GET Request in Python

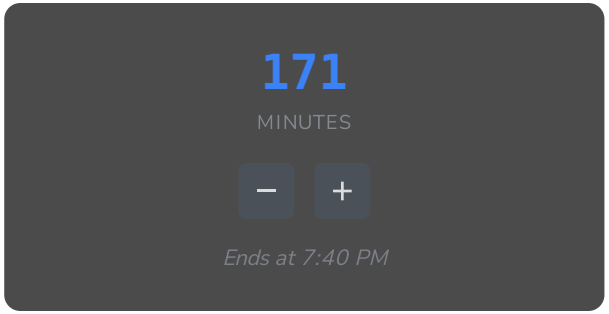
Making an HTTP GET request using Python's requests library.

```python
import requests
import socket

# Method 1: Using requests library (high-level)
def http_get_requests():
    try:
        response = requests.get('http://example.com')
        print(f"Status Code: {response.status_code}")
        print(f"Headers: {response.headers}")
        print(f"Content: {response.text[:200]}...")
    except requests.exceptions.RequestException as e:
        print(f"Request failed: {e}")

# Method 2: Using socket library (low-level)
def http_get_socket():
    # Create socket
    sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)

    try:
        # Connect to server
        sock.connect(('93.184.216.34', 80))  # example.com

        # Send HTTP request
```

Lab 2

More packet analysis.

https://hacs408e.dev/schedule/week-05/lab-2/



Homework

Homework 3 is due in two week.

Quiz 2 is two weeks away.