Computer Networking

IP Address: The House Address of the Internet

- Every device on the internet has one
- Without it, devices can’t find each other
- **terminal command**:- ifconfig
- talk to yourself:- ping 127.0.0.1

What is an IP Address?

- Unique identifier for each device on a network
- Written in numbers (IPv4) or longer strings (IPv6)
- Example: `142.250.190.78`

Analogy: Like a house address or a phone number

Demo

- Command: `ping google.com`
- Shows the IP of Google’s server
- Demo screenshot on slide (insert terminal result)

Types of IP Addresses

- **IPv4**: 32-bit, dotted decimal (e.g. `192.168.1.1`)
- **IPv6**: 128-bit, hexadecimal (e.g. `2001:0db8:85a3:0000:0000:8a2e:0370:7334`)
- IPv6 needed because IPv4 pool is limited

IPv4 Classes Overview

IP addresses divided into **classes** (historical use, still important):

- **Class A**: `1.0.0.0 – 126.255.255.255`
- **Class B**: `128.0.0.0 – 191.255.255.255`
- **Class C**: `192.0.0.0 – 223.255.255.255`
- **Class D**: `224.0.0.0 – 239.255.255.255` (Multicast)
- **Class E**: `240.0.0.0 – 255.255.255.255` (Experimental)

Visual Table of Classes

Class	Starting IP	Ending IP	Use Case	Default Subnet Mask
A	1.0.0.0	126.255.255.255	Large networks	255.0.0.0
B	128.0.0.0	191.255.255.255	Medium networks	255.255.0.0
C	192.0.0.0	223.255.255.255	Small networks	255.255.255.0
D	224.0.0.0	239.255.255.255	Multicast	N/A
E	240.0.0.0	255.255.255.255	Research/Reserved	N/A

Classification (Functional)

- **Public IP** → Used on the internet, unique globally
- **Private IP** → Used inside local networks (e.g. 192.168.x.x)
- **Loopback** → 127.0.0.1, refers to your own computer
- **Reserved** → Special use, not assignable

IP Address Flow Diagram

Flow to explain:

1. User device gets IP (public or private)
2. Router/NAT maps internal private IPs to public IP
3. Data sent over internet to another public IP
4. Target device responds back

PC → Router → Internet → Server → Back)

Recap

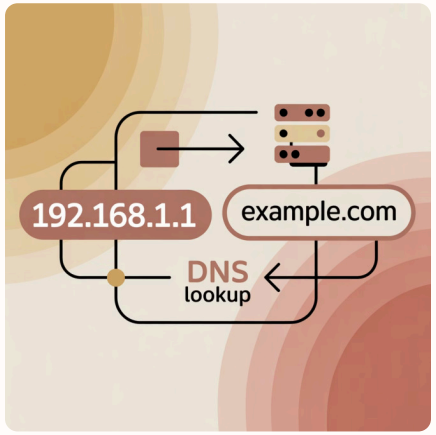
- IP = address of devices on the internet
- IPv4 vs IPv6
- IPv4 Classes (A–E)
- Public vs Private vs Loopback
- Essential for networking & web development

DNS

DNS: The Phonebook of the Internet

- Humans remember names, computers need numbers.
- DNS bridges that gap.

The Problem



- Computers talk in **IP addresses**: `142.250.190.78`
- But people prefer names: `google.com`
- Without DNS, we'd memorize numbers for every site

The Solution: DNS

- DNS = Domain Name System
- Like a **phonebook** that maps names to numbers
- Turns `google.com` → `142.250.190.78`

DNS Resolution Flow (Step by Step)

1. You type `google.com` in browser
2. Browser asks **Local DNS Resolver** (ISP / Google DNS / Cloudflare)
3. If not cached, resolver asks:
    - **Root Server** → directs to `.com` servers
    - **TLD Server (.com)** → directs to Google's authoritative server
    - **Authoritative Server (Google)** → returns the IP
4. Resolver caches result, gives IP back to browser
5. Browser connects to Google's server

DNS Hierarchy

- **Root servers** (.) – top of the tree
- **TLD servers** (.com, .org, .net, etc.)
- **Authoritative servers** – final source of truth for domain's IP

Analogy

- User → Librarian (DNS resolver)
- Librarian doesn’t know → checks catalogs (Root, TLD, Authoritative)
- Finds the book (IP address)
- Returns it to you

Recap

- DNS = Internet’s phonebook
- Converts domain names into IP addresses
- Resolution flow: Browser → Resolver → Root → TLD → Authoritative → IP
- Makes the internet usable for humans

HTTPS: Secure Communication on the Web

- From open postcards to locked boxes

Start with HTTP

- **HTTP (HyperText Transfer Protocol)** = how browsers talk to servers
- Sends **requests** and gets **responses**
- Problem: Data is plain text (anyone can read or alter it)

The Problem

- Login over HTTP → passwords visible to attackers
- Hackers can intercept (Man-in-the-Middle attack)
- No authenticity → you can’t be sure it’s really the site you think

HTTPS to the Rescue

- **HTTPS (HTTP Secure)** = HTTP + encryption
- Uses **SSL/TLS protocol**
- Ensures:
  - **Confidentiality** → data hidden from attackers
  - **Integrity** → data not tampered with
  - **Authentication** → you're talking to the real website

How HTTPS Works (Handshake)

1. Browser connects to server
2. Server sends its **SSL certificate**
3. Browser verifies certificate → trusted authority?
4. Browser & server agree on encryption keys (handshake)
5. From now → all data is encrypted

Demo Idea

- Visit `http://example.com` vs `https://example.com`
- Show **no lock** vs **lock symbol** in browser
- Click lock → view certificate details

Analogy

- **HTTP = Postcard** → everyone on the route can read your message
- **HTTPS = Locked Box** → only sender & receiver have the key

Recap

- HTTP = insecure, plain text
- HTTPS = HTTP + encryption (SSL/TLS)
- Protects privacy, prevents attacks, builds trust
- Every modern site must use HTTPS (browsers warn if not)

Ex:- Create a clean, minimal website with a light mode and dark mode toggle. The design should feel like Apple’s style — spacious, uniform, and consistent.

The website should include:

- A centered **hero section** with a short title ("Welcome to Our Project") and a subtitle ("Built step by step, deployed with care").
- A top navigation bar with three placeholder links: Home, About, Contact.
- A footer with small, clean text ("© 2025 Our Project. All rights reserved.").
- The light mode should use **white backgrounds, soft gray text, and subtle shadows**.
- The dark mode should use **#111 backgrounds, #EEE text, and smooth transitions**.
- Add a **toggle button** in the top-right to switch between modes, with a clean animation.

Netlify AND Vercel

Q&A / Thank You

Thank you for joining us on this journey into Collaborative Coding.

We look forward to seeing what you'll create your valuable projects with GitHub!