Contents

Network Layer: Data Plane Overview

The network layer is responsible for moving data across different devices in a network. It acts like the middle manager of communication, ensuring messages from one device reach another correctly. It handles two main tasks: **forwarding** and **routing**.

Key Goals of the Network Layer

- Understand the principles of how network layer services work.
- Learn the difference between **forwarding** (local decision-making) and **routing** (global planning).
- Explore how a router works, including its architecture.
- Learn about addressing systems like IP.
- Understand advanced topics like generalized forwarding and Internet architecture.

Forwarding vs. Routing: Simple Analogy

Imagine taking a trip:

1. **Forwarding**: Think of it as navigating a single junction or interchange on a highway. You decide which exit to take based on signs.
2. **Routing**: Planning the entire journey before you even start driving, deciding the best route to your destination.

In networking:

- **Forwarding** moves packets (data) from the router's input port to the correct output port.
- **Routing** determines the overall path the packet will take from the source to the destination.

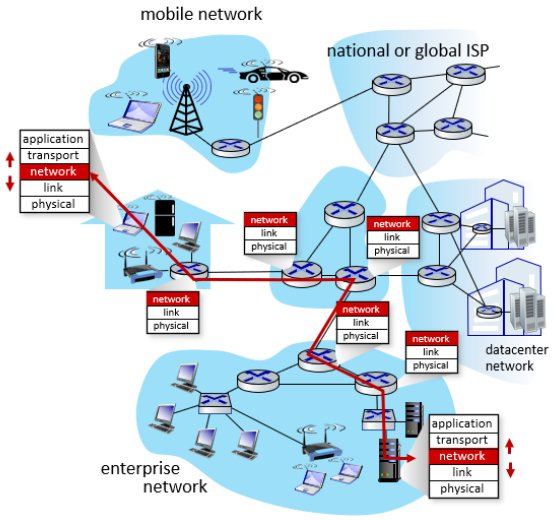
Network Layer Functionalities

The network layer has two parts:

1. **Data Plane**:
    - Works inside each router (local function).
    - It decides how a data packet is sent from the input port to the output port of the router.
2. **Control Plane**:
    - Uses a global view (network-wide logic).
    - It determines how packets are routed between routers on the way from the sender to the receiver.

Two types of control planes:

- **Traditional Routing**: Algorithms run on each router to determine the best path.
- **Software-Defined Networking (SDN)**: A centralized server calculates and controls routing decisions for the entire network.



Two Key Functions of the Network Layer

1. **Forwarding**: Ensures the data packet is delivered to the right place within the router. Example: If a letter arrives at a post office, forwarding is like sorting it to the correct delivery truck.
2. **Routing**: Plans the full path of a packet from sender to receiver. Example: A GPS system plans your entire trip from your home to a friend's house.

Example

Let's say a video file is sent over the Internet:

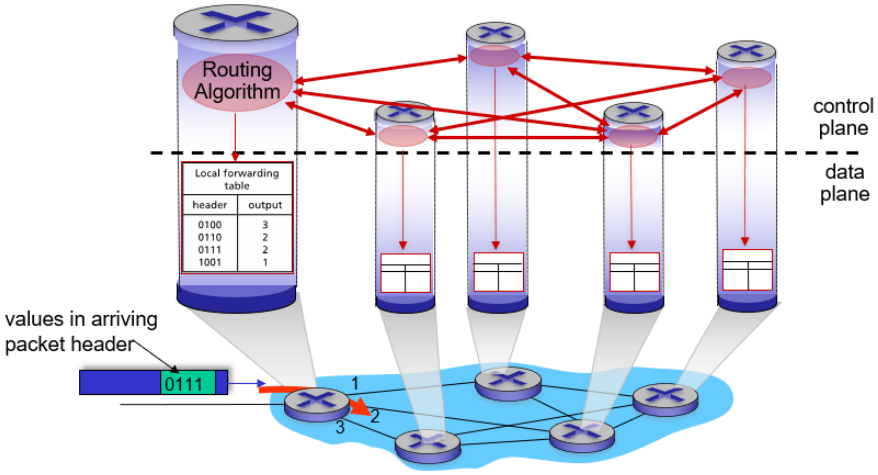
1. **Forwarding**: Ensures each packet of the video reaches the next router on its way.
2. **Routing**: Determines the best overall path for all packets to follow across cities or countries.

Data Plane vs. Control Plane: A Key Comparison

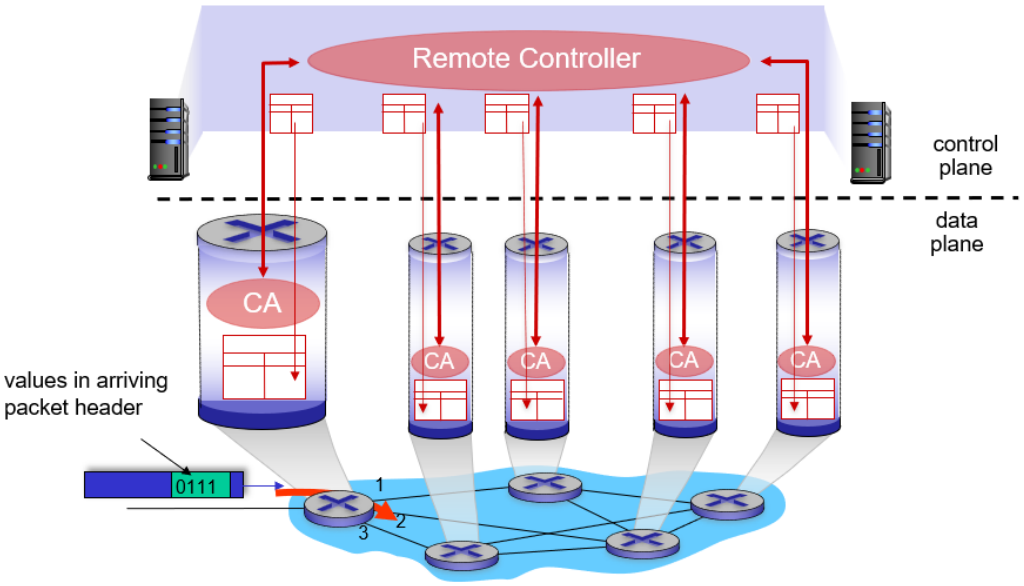
Aspect	Data Plane	Control Plane
Focus	Local router decisions	Global routing across the network
Implementation	Hardware-based (fast)	Software-based (flexible)
Timeframe	Operates in nanoseconds	Operates in milliseconds

Control Plane: Two Approaches

1. **Per-Router Control Plane**:
    - Each router independently calculates its routing decisions.
    - Example: A person deciding their path at each turn without guidance.



2. **SDN (Centralized Control)**:
    - A central server manages routing for all routers in the network.
    - Example: Following a travel itinerary planned by a tour guide.



Network Service Models

Different services can be provided for packets:

1. For **individual datagrams**:
    - **Guaranteed Delivery**: Ensures the packet reaches its destination.
    - **Guaranteed Delivery with Delay Constraint**: Delivery happens within a specific time (e.g., 40 milliseconds).
2. For a **flow of datagrams** (a series of packets):
    - **In-Order Delivery**: Packets arrive in the same order they were sent.
    - **Guaranteed Minimum Bandwidth**: Ensures enough speed for tasks like video calls.
    - **Inter-Packet Spacing**: Controls the time gaps between packets.

Example:

- Watching a live football match online: It requires in-order delivery and enough bandwidth.

Best-Effort Service Model
The Internet primarily uses a **best-effort model**, which means:

- No guarantees are made for delivery time or order.
- This model is simple and widely adopted because it works well with most applications.

Analogy: Sending a regular letter through postal mail without tracking or priority delivery. It usually works fine, but there's no guarantee when it will arrive.

Reflection on Best-Effort Model

- Simplicity makes it easy to deploy and maintain.
- Applications like video streaming or voice calls perform well because of sufficient bandwidth and distributed services (like data centers).

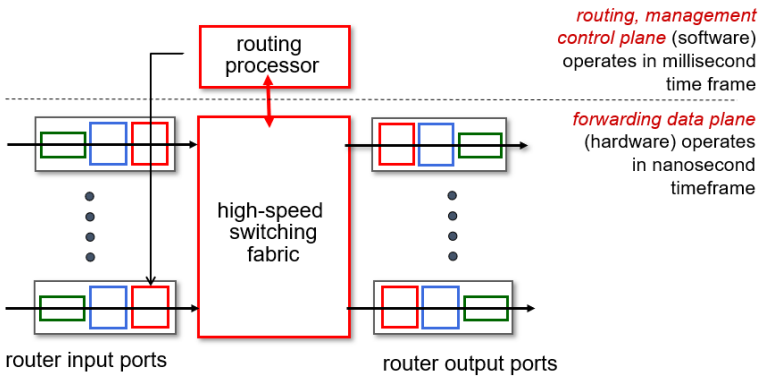
Router Architecture

Routers are crucial devices in a network. They connect different networks and decide how data packets are forwarded to their destination. Inside a router, various components work together to process and forward data efficiently.

High-Level View of a Router

A router consists of:

1. **Input Ports**: Where data packets enter the router.
2. **Switch Fabric**: Connects input ports to output ports, enabling the transfer of packets.
3. **Output Ports**: Where data packets leave the router.
4. **Routing Processor**: Handles high-level control tasks like running routing protocols and updating forwarding tables.



Two Planes in a Router

1. **Forwarding/Data Plane**:
   - Operates in hardware.
   - Processes packets quickly in nanoseconds.
   - Focuses on moving packets from input ports to output ports.
2. **Routing/Control Plane**:
   - Operates in software.
   - Handles decision-making like updating routing tables.
   - Works at a slower millisecond scale.

Analogy:

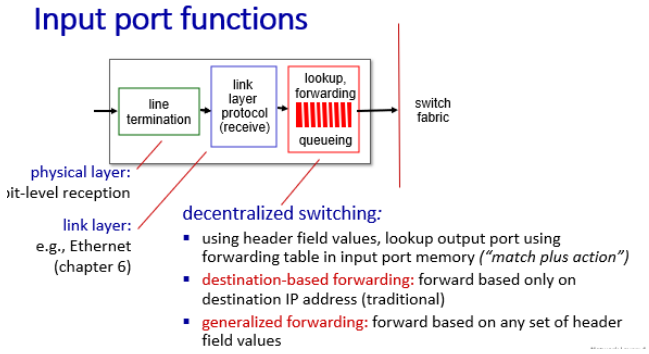
Think of a router as a busy train station:

- **Input ports** are the platforms where trains arrive.
- **Switch fabric** is the track system connecting platforms to different exits.
- **Output ports** are the tracks where trains leave.
- **Routing processor** is the station manager who plans routes and ensures smooth operation.

Input Port Functions

Each input port plays multiple roles:

1. **Line Termination**:
   - Handles signals from the physical link (e.g., Ethernet or fiber optic).
   - Converts them into a form the router can understand (bit-level reception).
2. **Data Link Processing**:
   - Verifies that packets are correctly received at the data link layer.
   - Removes link-layer headers before forwarding packets to the network layer.
3. **Lookup, Forwarding, and Queueing**:
   - Uses the **forwarding table** to decide the correct output port for a packet.
   - Employs "match+action" logic, which matches the packet's destination with a rule in the forwarding table and determines the next step.



- May temporarily queue packets if the switch fabric is busy.

Types of Forwarding

1. **Destination-Based Forwarding**:
   - In this traditional method, the router looks only at the **destination IP address** in the packet header. It then matches this address with an entry in its forwarding table to decide which output port to use.

   Example:

   - A packet is sent from a device in New York to a server in San Francisco.
   - The router at each step examines only the **destination address** (San Francisco).
   - It chooses the appropriate output port to move the packet closer to its destination without worrying about other fields in the packet.

   Analogy:

   - This is like sending a letter via the postal system. The postal workers check only the address written on the envelope (e.g., "San Francisco") to decide the next hop for delivery.

---

2. **Generalized Forwarding**:
   - Here, the router can look at **multiple fields** in the packet header, such as:
     - Destination IP address
     - Source IP address
     - Protocol type (e.g., TCP or UDP)
     - Port numbers
   - The router makes decisions based on flexible rules defined in the forwarding table.

   Example:

   - A company has a policy to prioritize packets carrying video streams over standard emails.
   - The router examines the packet's **protocol type** to identify video traffic and forwards it to a specific output port with higher bandwidth or lower delay.

   Analogy:

- Imagine a customs checkpoint at an airport. Officials examine not just your passport but also other details like your visa type and the purpose of your travel to decide how to process you. For example:
  - Tourists are directed to one line.
  - Business travelers go to another, faster line.

3. **Comparison Table**

Feature	Destination-Based Forwarding	Generalized Forwarding
Decision Basis	Only the destination IP address	Multiple fields (IP, port, protocol)
Flexibility	Fixed, simpler rules	Dynamic, complex rules
Use Case	General data forwarding on the Internet	Custom policies for QoS or security
Example	Packet delivery across routers in the Internet	Prioritizing video streams over emails

---

Longest Prefix Matching

The **Longest Prefix Matching** rule is a critical concept in routing. It determines how a router selects the most specific route for a packet when multiple entries in the forwarding table match the destination address.

Concept

- The router examines the **destination IP address** of a packet.
- It looks at the routing table to find the **longest prefix** (the most specific match) that matches the destination address.
- This ensures that the packet takes the most precise route to its destination.

How It Works

1. **IP Address and Prefix**:
   - An IP address (e.g., `192.168.1.0`) has a prefix that defines the network part of the address.
   - Prefixes are represented in **CIDR notation** (e.g., `192.168.1.0/24` means the first 24 bits define the network).
2. **Matching Process**:

- When a packet arrives, the router compares its destination IP address with all the prefixes in its forwarding table.
- The **longest matching prefix** is chosen for forwarding.

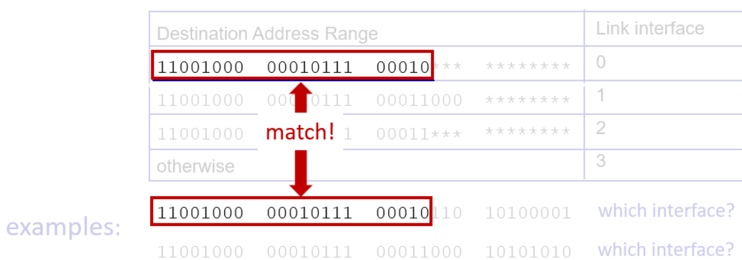
Example

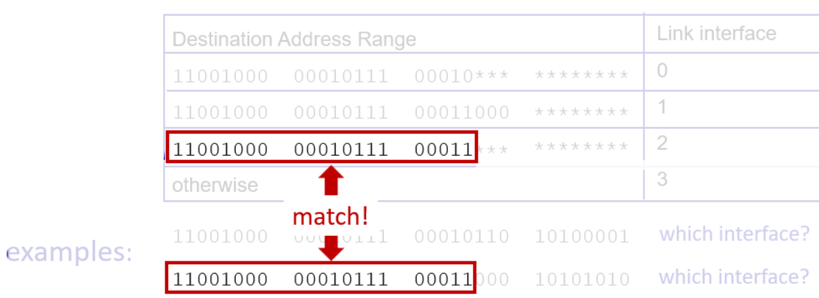
Destination Address Range	Link interface
11001000 00010111 00010*** ********	0
11001000 00010111 00011000 ********	1
11001000 00010111 00011*** ********	2
otherwise	3

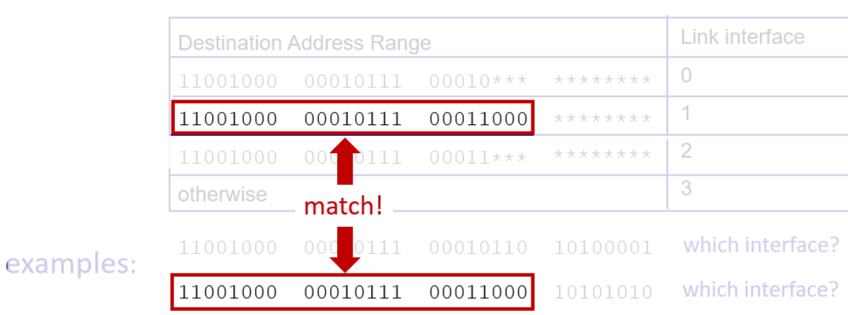
examples:

11001000 00010111 00010110 10100001 which interface?

11001000 00010111 00011000 10101010 which interface?







- longest prefix matching: often performed using ternary content addressable memories (TCAMs)

- *content addressable:* present address to TCAM: retrieve address in one clock cycle, regardless of table size
- Cisco Catalyst: ~1M routing table entries in TCAM

Why Longest Prefix Matching?

- **Efficiency**: Ensures the packet is forwarded as close to its destination as possible.
- **Scalability**: Helps routers handle a large number of IP addresses by using prefixes instead of individual entries.

Switch Fabric

The **switch fabric** is the heart of the router. It connects input ports to output ports and enables data to move efficiently through the router. There are three main types of switch fabrics:

1. **Switching via Memory**:
   - Oldest approach, where the CPU directly copies packets from input to output.
   - Limited by memory bandwidth and processing speed.
   - Example: Early routers acted like general-purpose computers.
2. **Switching via Bus**:
   - All input and output ports share a common bus (data transfer path).
   - Faster than memory switching but limited by bus bandwidth.
   - **Example**: Cisco 5600 routers with 32 Gbps bus speed, suitable for smaller networks.
3. **Switching via Interconnection Network**:
   - Modern routers use interconnection networks like **crossbars** or **Clos networks**.
   - Supports parallelism by dividing packets into smaller fixed-length cells, transferring them in parallel, and reassembling them at the output port.
   - Scalable to handle massive traffic in data centers or high-speed networks.

Switching Example

- Imagine a mail sorting facility:
  - **Switching via memory**: One worker sorts all the mail, slowing things down.
  - **Switching via bus**: A conveyor belt delivers mail to workers, but everyone shares the same belt.
  - **Switching via interconnection network**: Multiple sorting machines work in parallel, speeding up the process.

Input Port Queuing

When packets arrive faster than they can be processed or forwarded:

1. Packets are queued in the input buffer.
2. Delays and packet loss may occur if the queue becomes full.

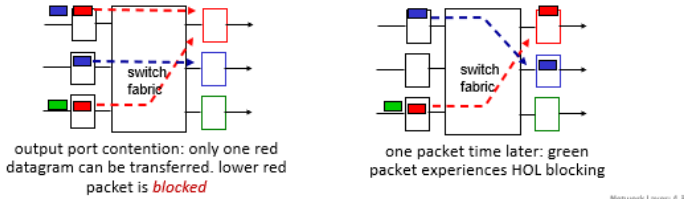
Key Problem: **Head-of-the-Line (HOL) Blocking**

- The first packet in the queue blocks others if it cannot be forwarded due to contention (e.g., two packets need the same output port).

Example: In a supermarket, if the first customer in the line cannot pay, it delays everyone else behind them, even if they are ready to check out.

Input port queuing

- If switch fabric slower than input ports combined -> queueing may occur at input queues
  - queueing delay and loss due to input buffer overflow!
- Head-of-the-Line (HOL) blocking: queued datagram at front of queue prevents others in queue from moving forward



output port contention: only one red datagram can be transferred. lower red packet is *blocked*

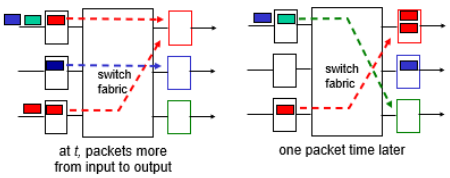
one packet time later: green packet experiences HOL blocking

Output Port Queuing

When packets are processed faster than the link can send them:

1. Packets are queued at the output buffer.
2. Buffers hold packets temporarily until they can be transmitted.

Output port queuing



at *t*, packets more from input to output

one packet time later

- buffering when arrival rate via switch exceeds output line speed
- *queueing (delay) and loss due to output port buffer overflow!*

How Much Buffering Is Needed?

Routers need buffers to store packets temporarily when there is congestion. However, deciding how much buffering is needed is crucial to avoid delays or packet loss.

Rule of Thumb (RFC 3439)

- The average buffering needed is proportional to the **round-trip time (RTT)** multiplied by the link's capacity:

$$\text{Buffering} = \text{RTT} \times \text{Capacity}$$

- Example:
  - For a 10 Gbps link with a typical RTT of 250 milliseconds:

$$\text{Buffer Size} = 10\ \text{Gbps} \times 0.25\ \text{s} = 2.5\ \text{Gbits}$$

- more recent recommendation: with *N* flows, buffering equal to

$$\frac{\text{RTT} \cdot C}{\sqrt{N}}$$

Challenges of Too Much Buffering

- While buffering helps reduce packet loss, **excessive buffering** can:
  - Increase delays, especially for real-time applications like video calls.
  - Degrade the performance of congestion-control mechanisms like TCP.
- **Ideal Goal**:
  - Keep the bottleneck link "just full enough" but not overloaded.

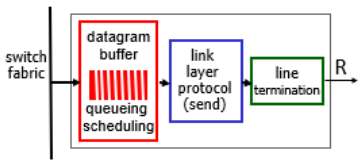
Analogy: Think of a water pipeline. If you use too many storage tanks (buffers), the water delivery gets delayed even though it doesn't spill.

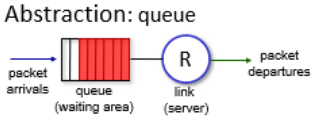
Buffer Management

When the buffer is full, routers must decide which packets to drop or prioritize. Buffer management policies help achieve this.

Key Policies

1. **Tail Drop**:
   - When the buffer is full, the router drops any newly arriving packet.
   - Simple to implement but can lead to bursty packet loss.
   - **Example**: A crowded elevator that refuses to let in any more people after reaching capacity.
2. **Priority-Based Dropping**:
   - Packets are prioritized based on their importance (e.g., real-time video traffic gets higher priority).
   - Low-priority packets are dropped first when the buffer is full.
3. **Marking Packets**:
   - Some routers use marking (e.g., Explicit Congestion Notification, ECN) to signal congestion instead of dropping packets.





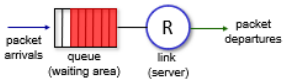
Packet Scheduling

When multiple packets are in the queue, the router decides the order in which they are sent. This process is called **packet scheduling**, and there are different scheduling policies.

packet scheduling: deciding which packet to send next on link
- first come, first served
- priority
- round robin
- weighted fair queueing

FCFS: packets transmitted in order of arrival to output port
- also known as: First-in-first-out (FIFO)
- real world examples?

Abstraction: queue



---

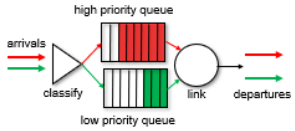
1. First Come, First Served (FCFS):

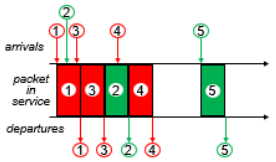
- Packets are transmitted in the same order they arrive.
- **Simple but inefficient** for prioritizing urgent traffic.

Analogy: A queue at a grocery store where everyone is served in order of arrival, regardless of what they are buying.

---

2. Priority Scheduling:



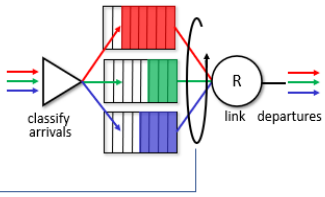


- Packets are classified into queues based on priority.
- The router transmits packets from the **highest-priority queue** first.
- **Use Case**: Prioritize voice or video traffic over regular emails.

Analogy: Emergency patients in a hospital are treated before others, regardless of their arrival time.

---

3. Round Robin (RR):

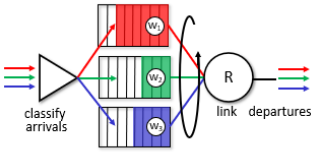


-
- The router cycles through multiple queues, transmitting one packet from each queue in turn (if available).
- **Fair but may not meet all traffic needs**.

Analogy: At a family dinner, each person is served food in a circular order, ensuring everyone gets a chance.

---

4. Weighted Fair Queueing (WFQ):



- A more advanced form of **round robin**.
- Each queue is assigned a weight (priority) that determines how many packets it can send per cycle. $\frac{w_i}{\sum_j w_j}$
- **Example**: A video call queue gets more weight than a web browsing queue.

Analogy: Imagine a buffet where some guests get bigger portions because they have a higher priority (e.g., VIP guests).

Network Neutrality

What Is Network Neutrality?

- A principle that ensures Internet Service Providers (ISPs) treat all traffic equally.
- ISPs cannot:
  - Block lawful content.
  - Slow down specific traffic (throttling).
  - Prioritize certain traffic for payment (paid prioritization).

Why Is It Important?

- **Technical Perspective**:
  - Network neutrality ensures fair packet scheduling and buffer management.
- **Social and Economic Perspective**:
  - Promotes free speech and competition.
  - Encourages innovation by treating small businesses the same as large corporations.

Real-World Example (US FCC 2015 Rules):

1. **No Blocking**:
   - ISPs cannot block access to lawful websites or applications.
2. **No Throttling**:
   - ISPs cannot intentionally slow down specific types of traffic.
3. **No Paid Prioritization**:
   - ISPs cannot create "fast lanes" for companies that pay extra.

Analogy: Think of a toll-free highway where all vehicles (traffic) can travel at the same speed, regardless of their size or purpose.

ISP Classification: Telecommunications or Information Service?

This debate arises from the **US Telecommunication Acts of 1934 and 1996**:

1. **Telecommunications Service (Title II)**:
   - ISPs act as "common carriers."
   - Regulated to ensure reasonable rates and non-discrimination.
   - **Example**: Traditional phone networks.
2. **Information Service (Title I)**:
   - ISPs are exempt from "common carrier" duties.
   - Less regulation but grants the FCC some oversight.

Why This Matters:

- The classification determines how ISPs are regulated and whether they must adhere to network neutrality principles.

Layered Model Integration
The network layer operates between:

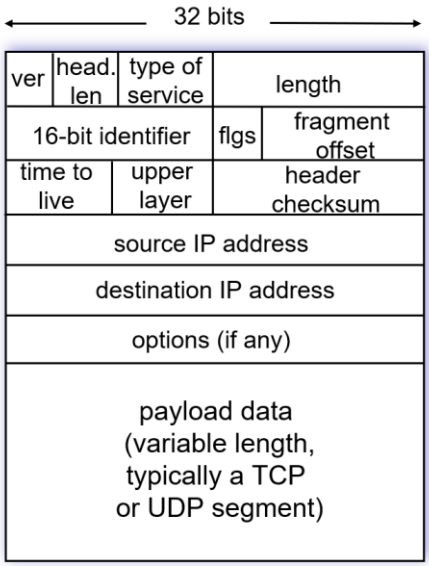
- **Transport Layer (e.g., TCP/UDP)**: Breaks down data into packets.
- **Link Layer (e.g., Ethernet)**: Transmits the packets over physical media.

Example:

- TCP ensures reliable delivery, and the network layer (IP) handles addressing and routing.

IP Datagram Format

The Internet Protocol (IP) organizes data into small units called **datagrams**. These datagrams are forwarded across networks to their destination.



Key Fields in an IP Datagram

1. **Version**:
    - Specifies the IP version (e.g., IPv4 or IPv6).
2. **Header Length**:
    - Indicates the size of the header in 32-bit words.
3. **Total Length**:
    - Defines the total size of the datagram, including header and payload.
4. **Identification, Flags, and Fragment Offset**:
    - Handle fragmentation when the datagram exceeds the maximum transmission unit (MTU) of a link.
5. **Time-to-Live (TTL)**:
    - Limits the datagram's lifespan to prevent it from looping indefinitely.
6. **Protocol**:
    - Specifies the upper-layer protocol (e.g., TCP, UDP).
7. **Source and Destination IP Addresses**:
    - Indicate the origin and destination devices.

Example of IP Datagram

- Source Address: `192.168.1.1`
- Destination Address: `10.0.0.2`
- TTL: `64` (decreases at each hop)
- Protocol: `TCP` (value 6 in the header)

Analogy: An IP datagram is like a parcel in a postal system:

- **Source and Destination Addresses**: Written on the package.
- **TTL**: A self-destruct timer to ensure parcels don’t circulate endlessly.

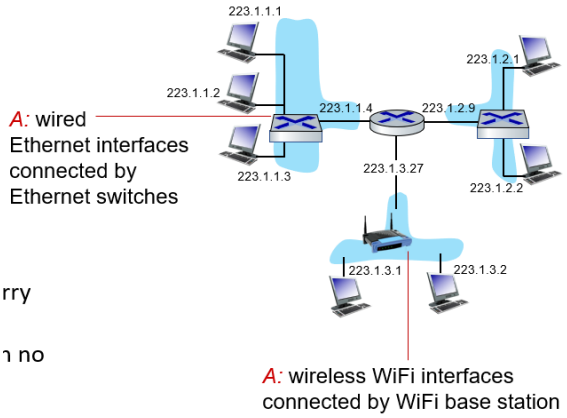
IP Addressing

IP addresses are unique 32-bit identifiers assigned to devices on a network. They enable devices to locate each other and communicate.

Key Characteristics:

1. **Interface-Based**:
    - Each device can have one or more interfaces (connections to a network).
    - Each interface has its own IP address.

2. **Dotted Decimal Notation**:
    - IP addresses are written in the form `A.B.C.D` (e.g., `192.168.1.1`).
3. **Two Parts**:
    - **Network Part**: Identifies the network.
    - **Host Part**: Identifies the specific device within the network.

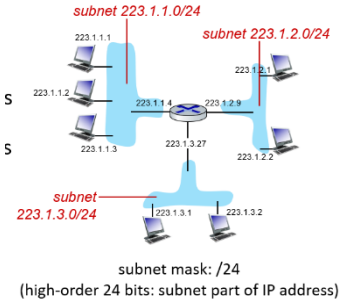


Subnets

- **Subnet**: A subdivision of an IP network.
- Devices within the same subnet can communicate directly without needing a router.

Subnet Mask:

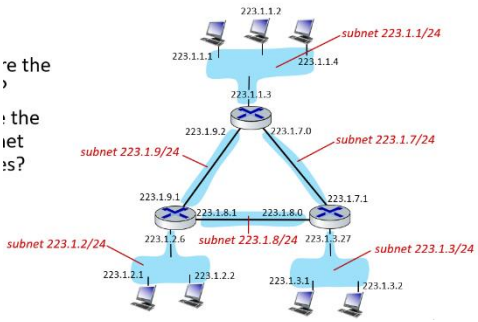
- Used to define the network and host parts of an IP address.
- Example: `/24` subnet mask means the first 24 bits represent the network.



Example of Subnets

1. Network: `192.168.1.0/24`
    - Subnet Range: `192.168.1.1` to `192.168.1.254`
    - Subnet Mask: `255.255.255.0`

Analogy: Think of an IP network as a neighborhood. Subnets are like individual blocks, and each house (device) has a unique number.



CIDR (Classless Inter-Domain Routing)

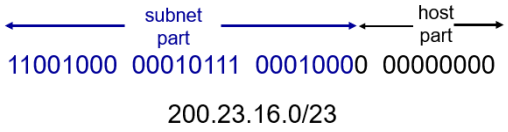
CIDR allows flexible allocation of IP addresses by using prefixes of arbitrary lengths, rather than fixed classes (A, B, C).

CIDR Notation:

- Written as `A.B.C.D/x`, where `x` is the number of bits in the network part.
- Example: `192.168.1.0/24`

Advantages:

- Efficient use of IP address space.
- Simplifies routing by aggregating multiple networks into a single route.



Example

- CIDR Block: `192.168.0.0/16`
  - Includes all addresses from `192.168.0.0` to `192.168.255.255`.
- Aggregated Route: Routers can handle all traffic for `192.168.x.x` as one route.

Analogy: CIDR is like creating a single delivery address for an entire apartment complex instead of assigning each unit its own delivery route.

IP Addresses: How to Get One?

When discussing how devices (hosts) or networks get IP addresses, we are essentially addressing two separate questions:

1. **How does a host get an IP address within its network?**
   - This refers to the assignment of the **host part** of the IP address, which uniquely identifies a device within a local network.
2. **How does a network get an IP address for itself?**
   - This refers to the allocation of the **network part** of the IP address, which represents the entire network in the global Internet.

How Does a Host Get an IP Address?

There are two primary ways for a host to obtain an IP address:

1. Hard-Coded by the System Administrator

- The IP address is manually configured in the device's operating system settings.
- For example:
  - On a Unix system, this could be done in a configuration file like `/etc/rc.config`.
- **Advantages**:
  - Provides full control over IP assignment.
  - Useful in environments where IP addresses rarely change.
- **Disadvantages**:
  - Time-consuming for large networks.
  - Not flexible for mobile devices or networks with frequent changes.

DHCP: Dynamic Host Configuration Protocol

How Networks Get IP Addresses

1. **From ISPs**:
   - ISPs receive IP address blocks from regional registries.
   - Example: `200.23.16.0/20` is allocated to an ISP, which divides it into smaller blocks for customers.
2. **From ICANN**:

- The Internet Corporation for Assigned Names and Numbers (ICANN) manages global IP address allocation.

Hierarchical Addressing

- Efficiently organizes and aggregates routes.
- Allows ISPs to announce a single prefix for multiple customers.

Example

- ISP announces `200.23.16.0/20`.
  - Customer 1: `200.23.16.0/23`
  - Customer 2: `200.23.18.0/23`

NAT (Network Address Translation)

Network Address Translation (NAT) is a mechanism used to map private IP addresses within a local network to a single public IP address when accessing the Internet. NAT is widely used in homes, offices, and organizations to conserve the limited IPv4 address space and provide additional security.

Key Concepts of NAT

1. **Why NAT?**
   - IPv4 has a limited number of unique public IP addresses (only ~4.3 billion). NAT allows multiple devices within a private network to share a single public IP address.
   - It provides a layer of security by hiding the internal structure of the network from external sources.
2. **Private IP Address Range**
   - NAT works with devices in private networks that use **private IP addresses**, which are not routable on the public Internet. These addresses fall into these ranges:
     - `10.0.0.0/8`
     - `172.16.0.0/12`
     - `192.168.0.0/16`
3. **Single Public IP for Local Network**
   - The NAT-enabled router uses one public IP address to represent all devices inside the private network when they communicate with external servers.

How NAT Works (USE SLIDES WHILE READING THIS)

1. **Outgoing Packets (Internal to External)**:
   - The router replaces the private **source IP address** and **port number** of the outgoing packet with the router's public IP address and a unique port number.
   - A mapping of the private (IP, port) to public (IP, port) is stored in a **NAT table**.
2. **Incoming Packets (External to Internal)**:
   - The router uses the NAT table to map the public IP and port back to the private IP and port, ensuring the packet reaches the correct device.

Example of NAT Translation

Private IP, Port	Public IP, Port
10.0.0.4:3345	138.76.29.7:5001
10.0.0.5:3346	138.76.29.7:5002

- Outgoing Packet:
  - Source: `10.0.0.4:3345` → Translated to `138.76.29.7:5001`.
- Incoming Reply:
  - Destination: `138.76.29.7:5001` → Translated back to `10.0.0.4:3345`.

Advantages of NAT

1. **Conserves IPv4 Addresses**:
   - Entire local networks can share a single public IP address.
2. **Security**:
   - Devices inside the private network are not directly addressable or visible to the external network, providing a natural firewall.
3. **Flexibility**:
   - Internal IP addresses can be changed without affecting external communications.

Disadvantages of NAT

1. **Violates End-to-End Connectivity**:
   - The end-to-end communication principle of the Internet is compromised as NAT modifies packet headers.
2. **NAT Traversal Issues**:
   - Applications like online gaming or VoIP (Voice over IP) may face problems as NAT interferes with peer-to-peer communication.
3. **Incompatibility with Certain Protocols**:
   - Protocols that embed IP addresses in the payload, such as FTP, may not work properly without additional configuration.

NAT Controversy

1. **Layer Violation**:
   - Routers should ideally operate only up to **Layer 3 (IP)** of the OSI model, but NAT manipulates information at the **transport layer (Layer 4)**, violating the traditional design.
2. **Alternative Solution**:
   - The IPv6 protocol eliminates the need for NAT by providing a much larger address space.

Example: NAT in Home Network

1. Devices in the Home Network:
   - Devices have private IP addresses like `192.168.1.2` or `192.168.1.3`.
   - The router has a single public IP address assigned by the ISP (e.g., `138.76.29.7`).
2. NAT Router Functionality:
   - When a laptop (`192.168.1.2`) accesses a website, NAT replaces the laptop's private IP with the public IP (`138.76.29.7`).
   - The external server only sees `138.76.29.7` as the source of the request.

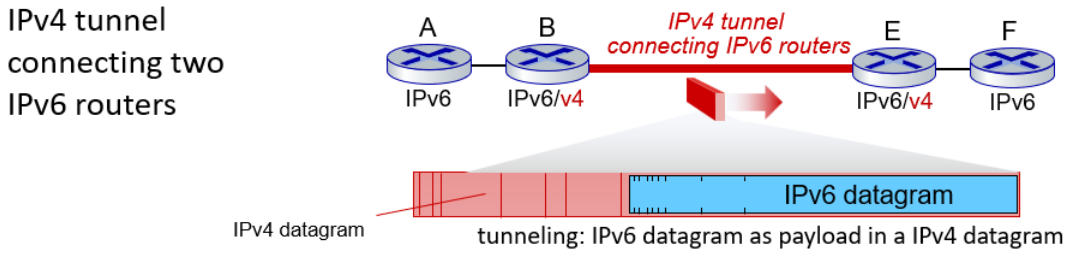
NAT and IPv6

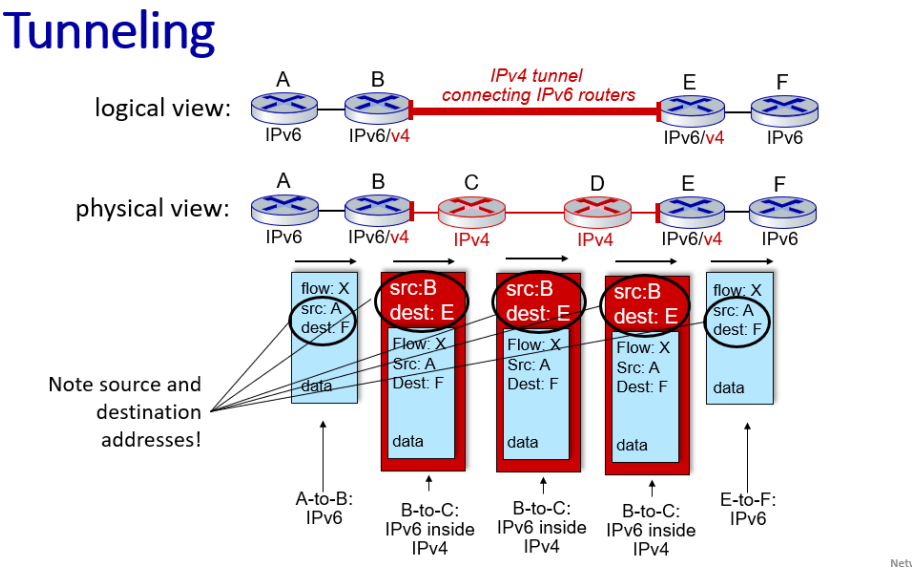
- **Why NAT Exists in IPv4**:
  - NAT was introduced to extend the lifespan of IPv4 as the number of available IPv4 addresses became insufficient.
- **How IPv6 Solves This**:
  - IPv6 provides a vastly larger address space (128-bit addresses), making NAT unnecessary.
  - Devices can have unique global IP addresses, restoring end-to-end connectivity.

Transition Challenges

1. **Mixed Networks**:

   During the transition from IPv4 to IPv6, networks with both IPv4 and IPv6 devices must use technologies like **tunneling** or **dual-stack operation** .





2. **Adoption Barriers**:

- NAT is deeply embedded in home and organizational networks, making its replacement challenging.

Key NAT Terms

- **NAT Table**: A mapping of private-to-public IP addresses and port numbers.
- **Port Address Translation (PAT)**: A common NAT implementation that maps multiple private IPs to a single public IP using different port numbers.

Flow Table Abstraction (USE SLIDES WHILE READING THIS)

Flow tables are used in **generalized forwarding** to define rules for how packets should be processed. Unlike traditional destination-based forwarding, flow tables allow more flexible and fine-grained control over packet handling.

Key Concepts:

1. **Flow**:
   - A flow is defined by a combination of packet header values (fields from link, network, and transport layers).
   - Example: Source IP, destination IP, port numbers, and protocol type.
2. **Match-Action Rules**:
   - Flow tables use rules where each rule consists of:
     - **Match**: Pattern values in packet header fields.
     - **Action**: Operation to be performed on matching packets (e.g., forward, drop, modify, log).
3. **Priority**:
   - If multiple rules match a packet, the rule with the **highest priority** is applied.
4. **Counters**:
   - Flow tables maintain statistics such as the number of bytes and packets matching a rule.

Example: Flow Table Entry

Match	Action	Counters
Destination IP = `192.168.1.1`	Forward to Port 3	Bytes: 5000, Packets: 100
TCP Port = `22` (SSH)	Drop	Bytes: 200, Packets: 10

- **Explanation**:
  - First rule: Packets destined to `192.168.1.1` are forwarded to port 3.
  - Second rule: Packets for TCP port 22 (SSH) are dropped.

OpenFlow: Flow Table Entries

OpenFlow is a protocol used in **Software-Defined Networking (SDN)** to manage flow tables in routers and switches. It enables centralized control over forwarding decisions.

Flow Table Structure:

- Each flow table entry contains:
    1. **Match Fields**: Header fields to identify packets (e.g., IP, MAC, port).
    2. **Action**: Operation to be applied (e.g., forward, drop, modify).
    3. **Statistics**: Counters to track packet and byte counts.

OpenFlow Examples:

1. **Forwarding Example**:
    - Rule: IP datagrams destined to `51.6.0.8` → Forward to output port `6`.
2. **Blocking Example**:
    - Rule: Block all datagrams destined to **TCP port 22** (SSH).
3. **Source Blocking Example**:
    - Rule: Block all packets originating from source `128.119.1.1`.

Generalized Forwarding

Generalized forwarding is an evolution beyond traditional forwarding, where decisions are no longer limited to the destination IP address. Instead, routers can analyze **any packet header fields** and apply actions based on programmable rules.

Key Features of Generalized Forwarding:

1. **Flexible Matching**:
    - Instead of just matching the destination IP address, routers can match fields at different layers:
        - Link layer: MAC address.
        - Network layer: IP address.
        - Transport layer: Port numbers, protocol type.
2. **Multiple Actions**:
    - Actions include:
        - **Forwarding**: Send the packet to a specific output port.

        - **Dropping**: Discard the packet.
        - **Modification**: Rewrite parts of the packet header.
        - **Logging**: Record the packet for analysis.
3. **Programmability**:
    - Generalized forwarding allows the control plane (e.g., SDN controllers) to **program network behavior** dynamically.

Comparison: Traditional vs. Generalized Forwarding

Aspect	Traditional Forwarding	Generalized Forwarding
Decision Basis	Destination IP address only	Any combination of header fields
Action	Forward to output port	Forward, drop, modify, or log
Flexibility	Fixed rules	Dynamic and programmable
Use Case	Standard IP routing	Advanced traffic management (e.g., QoS, firewalls)

Example of Generalized Forwarding

Match	Action
Source IP = `10.1.1.1` AND TCP port 80	Forward to Port 2
Destination IP = `192.168.1.5`	Drop
Protocol = ICMP (ping requests)	Forward to controller

- **Explanation**:
    - Rule 1: Traffic from `10.1.1.1` on port 80 (web traffic) is forwarded to port 2.
    - Rule 2: Traffic for destination `192.168.1.5` is dropped.
    - Rule 3: ICMP traffic (ping) is forwarded to the controller for inspection.

Why Generalized Forwarding?

1. **Traffic Control**:
    - Enables advanced Quality of Service (QoS) policies by identifying and prioritizing certain traffic types.
2. **Security**:
    - Allows dynamic blocking of malicious traffic based on specific patterns.
3. **Network Innovation**:
    - Supports programmable networks, where new rules can be applied without hardware changes.

Middleboxes

Middleboxes are network devices that perform functions **beyond the standard IP routing** of forwarding packets based on destination addresses. They act as intermediaries in the data path between source and destination.

What Are Middleboxes?

According to **RFC 3234**, a middlebox is:
"Any intermediary device that performs functions apart from the normal, standard functions of an IP router on the data path."

Examples of Middleboxes

1. **NAT (Network Address Translation)**:
   - Translates private IP addresses to public IPs for communication over the Internet.
2. **Firewalls**:
   - Monitors and controls incoming and outgoing traffic based on security rules.
3. **Load Balancers**:
   - Distributes traffic across multiple servers to optimize resource utilization.
4. **Intrusion Detection/Prevention Systems (IDS/IPS)**:
   - Monitors traffic for malicious patterns and prevents attacks.
5. **Web Caches**:
   - Temporarily stores frequently accessed content closer to users.
6. **Application-Specific Middleboxes**:
   - Devices optimized for specific applications, such as video streaming or content delivery.

Where Are Middleboxes Used?

Middleboxes are deployed across various networks:

1. **Home Networks**:
   - NAT for sharing a single public IP.
2. **Enterprise Networks**:
   - Firewalls, load balancers, and security systems.
3. **Data Centers**:
   - Load balancing, caching, and firewalls for handling large-scale traffic.
4. **Mobile and ISP Networks**:
   - NAT for IPv4 address conservation and content optimization.

Middleboxes: Benefits and Challenges

Benefits	Challenges
Enhanced security (firewalls, IDS/IPS).	Break the end-to-end communication model.
Efficient resource use (load balancers).	Adds complexity to network management.
Better performance (caching and optimization).	Harder to troubleshoot or debug networks.

Generalized Forwarding and Middleboxes

- Middleboxes complement **generalized forwarding** by matching various packet header fields and applying flexible actions like dropping, modifying, or forwarding packets.

- **SDN (Software-Defined Networking)** can unify and simplify middlebox operations through programmable rules.

Architectural Principles of the Internet

The Internet's architecture has evolved based on a few fundamental principles:

1. **Simple Connectivity**:
   - The network's primary goal is to enable devices to communicate with one another.
2. **The Internet Protocol (IP)**:
   - IP serves as the "narrow waist" of the network, ensuring compatibility between layers.
3. **End-to-End Principle**:
   - Intelligence and complexity should be placed at the edges of the network, not in the core.

End-to-End Principle Example

- Applications like **error correction** and **encryption** are implemented at the endpoints (source and destination), not in intermediate routers.

IP Fragmentation and Reassembly

When a large IP datagram exceeds the **Maximum Transmission Unit (MTU)** of a link, it must be **fragmented** into smaller pieces to be transmitted. The fragments are then **reassembled** at the destination.

Key Terms:

1. **MTU (Maximum Transmission Unit)**:
   - The largest size of a packet that a network link can carry.
   - Example: Ethernet has an MTU of **1500 bytes**.
2. **IP Fragmentation**:
   - A single large datagram is broken into multiple smaller datagrams (fragments) for transmission.
   - Each fragment has:
     - A **header**: Contains information to identify and reassemble fragments.
     - A **data payload**: Part of the original packet.
3. **Reassembly**:
   - Fragments are combined back into the original datagram **only at the destination**.

IP Header Fields Used in Fragmentation:

1. **Identification**:
   - Identifies all fragments belonging to the same original datagram.
2. **Fragment Offset**:
   - Specifies the position of a fragment within the original datagram.
   - Measured in **8-byte units**.
3. **Flags**:
   - **More Fragment (MF) Bit**:
     - `1` means more fragments follow.
     - `0` means this is the last fragment.

    - **Don’t Fragment (DF) Bit**:
        - `1` means the packet must not be fragmented. If it exceeds the MTU, it is dropped.

Example of Fragmentation

- **Original Datagram**: 4000 bytes
- **MTU**: 1500 bytes

Fragmentation Result:

1. Fragment 1: 1480 bytes (Data) + 20 bytes (Header), Offset = 0
2. Fragment 2: 1480 bytes (Data) + 20 bytes (Header), Offset = 185 (1480 / 8)
3. Fragment 3: Remaining 1040 bytes (Data) + 20 bytes (Header), Offset = 370

Each fragment has its own header but shares the same **Identification** field.

Problems with Fragmentation

1. **Reassembly Overhead**:
    - Reassembling fragments increases processing load at the destination.
2. **Loss of Fragments**:
    - If one fragment is lost, the entire datagram is discarded.
3. **Performance**:
    - Fragmentation increases delay and reduces network efficiency.

Solution: Modern protocols like IPv6 eliminate fragmentation within the network and rely on **Path MTU Discovery**.

Example 1: IP Fragmentation

Given:

- **Original Datagram Size**: 4000 bytes (including 20 bytes for the IP header)
- **MTU (Maximum Transmission Unit)**: 1500 bytes
- **Header Size**: 20 bytes (remains the same for all fragments)

Objective: Break the datagram into fragments that can fit into the MTU.

Step-by-Step Solution

1. **Calculate Data Size Per Fragment**:
   - The MTU is 1500 bytes, but 20 bytes are reserved for the header.
   - Therefore, the maximum **data** per fragment = 1500 - 20 = **1480 bytes**.
2. **Number of Fragments**:
   - Total data size in the original datagram = 4000 - 20 = **3980 bytes**.
   - Each fragment can carry 1480 bytes of data.
   - Number of fragments needed:

$$\text{Number of Fragments} = \frac{\text{Total Data Size}}{\text{Data Per Fragment}} = \frac{3980}{1480} = 2 \text{ full fragments} + 1 \text{ smaller fragment}$$

   - Breakdown of fragments:
     - Fragment 1: 1480 bytes of data
     - Fragment 2: 1480 bytes of data
     - Fragment 3: Remaining 1020 bytes of data
3. **Fragment Offset Calculation**:
   - **Fragment Offset** is measured in units of **8 bytes**.
   - For each fragment:
     - **Fragment 1**:
       - Offset = **0** (first fragment)
     - **Fragment 2**:
       - Offset = $\frac{\text{Data in Fragment 1}}{8} = \frac{1480}{8} = 185$
     - **Fragment 3**:
       - Offset = $\frac{\text{Data in Fragment 1} + \text{Fragment 2}}{8} = \frac{1480+1480}{8} = 370$
4. **Fragment Headers**:
   Each fragment has its own IP header with the following details:

Fragment	Data Size	Header Size	Total Size	Fragment Offset	More Fragment (MF) Bit
1	1480 bytes	20 bytes	1500 bytes	0	1
2	1480 bytes	20 bytes	1500 bytes	185	1
3	1020 bytes	20 bytes	1040 bytes	370	0

Example 2: Reassembly at the Destination

Given:

- Three fragments arrive at the destination with the following details:

Fragment	Fragment Offset	Data Size	More Fragment (MF) Bit
1	0	1480 bytes	1
2	185	1480 bytes	1
3	370	1020 bytes	0

Objective: Reassemble the original datagram.

Step-by-Step Solution

1. **Order Fragments Using the Offset**:
   - Fragment 1 (Offset = 0) → Data: 1480 bytes
   - Fragment 2 (Offset = 185) → Data: 1480 bytes
   - Fragment 3 (Offset = 370) → Data: 1020 bytes
2. **Calculate Total Data Size**:
   Add the data sizes of all fragments:

   Total Data=1480+1480+1020=3980 bytes

3. **Add the Header Size**:
   - Original header size = 20 bytes
   - Total size of the original datagram = 3980 + 20 = **4000 bytes**
4. **Reassembled Datagram**:
   The destination combines the fragments in order of their offsets:
   - **Fragment 1** → 0–1479 bytes
   - **Fragment 2** → 1480–2959 bytes
   - **Fragment 3** → 2960–3980 bytes

The **MF bit** of Fragment 3 is `0`, confirming it is the last fragment.

Example 3: Fragment Loss

Scenario:

- Suppose Fragment 2 (Offset = 185) is lost.
- The remaining fragments (Offset 0 and 370) arrive successfully.

Effect:

1. The destination cannot reassemble the datagram because **one fragment is missing**.
2. The entire datagram is **discarded**, and the sender may need to retransmit it.