**5.2: Classful Addressing**

**🔍 Classful Addressing আসলে কী?**

**Classful Addressing** হচ্ছে একটি পুরনো IP ঠিকানা ভাগ করার নিয়ম, যেটা IPv4 ঠিকানাগুলোকে ৫টি ভিন্ন শ্রেণিতে (Class) ভাগ করতো — **A, B, C, D, E**।

এখন এই পদ্ধতি আর বেশি ব্যবহার হয় না, কারণ **CIDR (Classless Inter-Domain Routing)** অনেক বেশি flexible। তবে networking-এর basic concept বুঝতে classful addressing জানা দরকার।

**🧠 Classful Addressing-এর মূল ধারণা:**

* IPv4 address হয় ৩২-বিটের।
* এই ৩২-বিট কে ৪টি ভাগে ভাগ করা হয়, যেগুলোকে বলে **Octet** (৮-বিট করে)।
* উদাহরণ: 192.168.1.1 (প্রতিটা অংশই একটি Octet)
* প্রথম কিছু বিট দেখে বোঝা যায়, IP address টা কোন **class**-এর মধ্যে পড়ে।

**📊 ভিন্ন ভিন্ন ক্লাসের বিবরণ:**

| **ক্লাস** | **শুরু বিট** | **প্রথম Octet-এর রেঞ্জ** | **নেটওয়ার্ক/হোস্ট বিভাজন** | **ব্যবহারের ক্ষেত্র** |
| --- | --- | --- | --- | --- |
| **A** | 0 | 1 - 126 | ৮ বিট নেটওয়ার্ক, ২৪ বিট হোস্ট | বড় বড় নেটওয়ার্কের জন্য |
| **B** | 10 | 128 - 191 | ১৬ বিট নেটওয়ার্ক, ১৬ বিট হোস্ট | মাঝারি নেটওয়ার্ক |
| **C** | 110 | 192 - 223 | ২৪ বিট নেটওয়ার্ক, ৮ বিট হোস্ট | ছোট নেটওয়ার্ক |
| **D** | 1110 | 224 - 239 | Multicast এর জন্য ব্যবহৃত |  |
| **E** | 1111 | 240 - 255 | Experimental (গবেষণার জন্য) |  |

**📝 উদাহরণ:**

* 10.0.0.1 → Class A
* 172.16.0.1 → Class B
* 192.168.1.1 → Class C

**❗ অতিরিক্ত তথ্য:**

* **Class D** ব্যবহার হয় Multicast-এর জন্য (একসাথে অনেককে ডেটা পাঠাতে)।
* **Class E** ব্যবহার হয় গবেষণার উদ্দেশ্যে।
* 127.0.0.1 → Loopback address (নিজের কম্পিউটারে নিজের সাথে যোগাযোগের জন্য)

**📌 উপসংহার:**

Classful Addressing ছিল একটি প্রাথমিক পদ্ধতি, যেটা এখন অনেকটাই পুরনো হয়ে গেছে। আধুনিক নেটওয়ার্কগুলোতে আমরা **CIDR (Classless)** ব্যবহার করি, কারণ এটা আরও বেশি flexible ও efficient।

তুমি চাইলে আমি **CIDR vs Classful** comparison-ও বাংলায় বুঝিয়ে দিতে পারি।

খুব ভালো প্রশ্ন! চল আমরা দেখি কিভাবে বুঝবে একটা IP Address কোন **Class** এর মধ্যে পড়ে।

## ✅ **কোন IP কোন Class তা কিভাবে বুঝবে?**

তুমি শুধু **IP address এর প্রথম অংশটা (1st Octet)** দেখলেই চলবে।

IP address: A.B.C.D (এখানে A = 1st Octet)

### 🎯 ****রুলস (প্রথম Octet দেখে):****

| **Class** | **প্রথম Octet-এর রেঞ্জ (A)** | **উদাহরণ** |
| --- | --- | --- |
| **A** | 1 – 126 | 10.0.0.1 → A |
| **B** | 128 – 191 | 172.16.0.1 → B |
| **C** | 192 – 223 | 192.168.1.1 → C |
| **D** | 224 – 239 | 224.0.0.1 → D (Multicast) |
| **E** | 240 – 255 | 245.1.1.1 → E (Experimental) |

🔴 **127.0.0.1** → এটা কোন ক্লাস না, এটা **loopback address** (নিজের কম্পিউটারকে টেস্ট করার জন্য ব্যবহৃত হয়)।

### 🔍 চল উদাহরণগুলো ব্যাখ্যা করি:

#### 🟢 10.0.0.1

* প্রথম অংশ = 10
* 1 থেকে 126 এর মধ্যে → **Class A**

#### 🟡 172.16.0.1

* প্রথম অংশ = 172
* 128 থেকে 191 এর মধ্যে → **Class B**

#### 🔵 192.168.1.1

* প্রথম অংশ = 192
* 192 থেকে 223 এর মধ্যে → **Class C**

### 📌 মনে রাখার ট্রিক:

* **1–126** → A (বড় নেটওয়ার্ক)
* **128–191** → B (মাঝারি)
* **192–223** → C (ছোট)
* **224–239** → D (Multicast)
* **240–255** → E (Experimental)

যদি চাও, আমি একটা ছোট Python code-o dite pari ja diye IP input dile Class ber kore dibe ✅  
বললে দিয়ে দিচ্ছি।

Great! Let’s understand:

## 🧩 **What is NetID and HostID in IP Addressing?**

Every **IPv4 address** (like 192.168.1.1) is divided into two logical parts:

### 🔹 ****1. NetID (Network ID):****

* এটি নির্ধারণ করে **কোন নেটওয়ার্কে** IP address টি আছে।
* এটা হল সেই অংশ যেটা সব device-এর জন্য **একই থাকে** যেগুলো একই network-এ আছে।

### 🔸 ****2. HostID (Host ID):****

* এটি নির্ধারণ করে **network-এর ভিতরে কোন specific device (host)** কে নির্দেশ করে।
* একই network-এর মধ্যে প্রত্যেক host-এর HostID আলাদা হয়।

### ✅ উদাহরণ দিয়ে বুঝি:

#### IP Address: 192.168.1.10

Class C Address → Network = 24 bits, Host = 8 bits

📌 Binary view:

192.168.1.10 → 11000000.10101000.00000001.00001010

| NetID | HostID |

* **NetID:** 192.168.1 → এটি বলে device টি কোন network-এ আছে
* **HostID:** 10 → এটি বলে সেই network-এর মধ্যে device টি কারা

### 🔍 ক্লাসভিত্তিক NetID ও HostID ভাগ:

| **Class** | **NetID Bits** | **HostID Bits** | **উদাহরণ** |
| --- | --- | --- | --- |
| A | 8 | 24 | 10.0.0.1 → NetID: 10, HostID: 0.0.1 |
| B | 16 | 16 | 172.16.5.4 → NetID: 172.16, HostID: 5.4 |
| C | 24 | 8 | 192.168.1.10 → NetID: 192.168.1, HostID: 10 |

### 💡 মনে রাখার কৌশল:

* **NetID = Network চিনে**
* **HostID = Network-এর ভিতরের “লোক” চিনে**

চাও চাইলে আমি চিত্রসহ বা Python code দিয়ে বুঝিয়ে দিতে পারি কীভাবে NetID, HostID আলাদা করা যায় কোনো IP address থেকে। বললেই পেয়ে যাবে! 😊

একদম সুন্দর প্রশ্ন! চল এখন সহজভাবে বুঝি:

## 🧩 **Classes এবং Blocks কী?**

IP Addressing এ **"Class"** এবং **"Block"** শব্দ দুটি প্রায়ই ব্যবহৃত হয়, কিন্তু ওদের কাজ ও উদ্দেশ্য আলাদা।

## 🔷 1. **Class (ক্লাস) কী?**

IP Address গুলোকে একসময় **Classful addressing system** অনুযায়ী ভাগ করা হতো। এতে IP address গুলো ৫টি ভাগে ভাগ করা হয়:

| **Class** | **IP Range (প্রথম Octet)** | **ব্যবহার** |
| --- | --- | --- |
| A | 1 – 126 | বিশাল network |
| B | 128 – 191 | মাঝারি network |
| C | 192 – 223 | ছোট network |
| D | 224 – 239 | Multicast |
| E | 240 – 255 | Research/Experimental |

প্রতিটি ক্লাসে নির্দিষ্ট সংখ্যক **NetID** ও **HostID** থাকে, যা Network এবং Device চিহ্নিত করে।

## 🔶 2. **Block (ব্লক) কী?**

**Block** মানে হলো — একটি নির্দিষ্ট **range** বা **group** of IP addresses যা কোনো network বা organization এর জন্য বরাদ্দ দেওয়া হয়।

যখন **Classless Addressing (CIDR)** শুরু হলো, তখন IP address গুলিকে **"block"** আকারে ভাগ করা শুরু হয়।

### উদাহরণ:

192.168.1.0/24 → একটি **block** যার মধ্যে আছে:

* 256 IP addresses (from 192.168.1.0 to 192.168.1.255)

এখানে /24 মানে হচ্ছে **24-bit NetID**, এবং বাকি 8-bit HostID

## 🔁 **Class vs Block তুলনা:**

| **বিষয়** | **Class** | **Block** |
| --- | --- | --- |
| পুরনো না আধুনিক | পুরনো (Classful) | আধুনিক (CIDR) |
| Address range নির্ধারণ | Fixed ভাগ (A, B, C) | Flexible যেকোনো size (CIDR) |
| Efficiency | কম | বেশি efficient |
| উদাহরণ | Class C → 192.168.1.0 | Block → 192.168.1.0/26 |

## 🎯 Summary:

* **Class:** পুরনো ধারা, যেখানে IP গুলোকে বড়, মাঝারি, ছোট network অনুযায়ী ভাগ করা হতো।
* **Block:** আধুনিক ধারা (CIDR), যেখানে IP গুলো নির্দিষ্ট subnet size অনুযায়ী ভাগ করা হয়।

যদি তুমি চাও আমি চিত্র বা table আকারে subnet blocks বা class visualization করে দিতেও পারি। বললেই করে দেব! ✅

### 🧩 What is Two-Level Addressing?

**Two-level addressing** হল একটি আইডিয়া যেখানে একটি আইপি ঠিকানাকে **দুইটি অংশে ভাগ** করা হয়:  
👉 **Network ID (NetID)**  
👉 **Host ID (HostID)**

এটি IP addressing-এর মূল ভিত্তি — যেটা বলছে, একটা IP Address শুধুমাত্র একটা "কম্পিউটার" চিহ্নিত করে না, বরং প্রথমে বলে **"কোন নেটওয়ার্কে"** আর তারপর বলে **"নেটওয়ার্কের ভিতরে কোন হোস্ট/ডিভাইস"**।

## 🧠 Two-Level Addressing Explained:

### 🔹 ****Level 1: Network Address (NetID)****

* এটি আইপি ঠিকানার সেই অংশ যা বলে, IP টি **কোন নেটওয়ার্কের** মধ্যে পড়ে।
* রাউটার বা ইন্টারনেট এটিকে দেখে বুঝে যে ডেটা **কোথায় পাঠাতে হবে**।

### 🔸 ****Level 2: Host Address (HostID)****

* এটি নেটওয়ার্কের ভিতরের নির্দিষ্ট **কম্পিউটার বা ডিভাইস** কে চিহ্নিত করে।
* একবার নেটওয়ার্কে পৌঁছে গেলে, HostID ব্যবহার করে **target device খুঁজে বের করা হয়**।

## 📦 উদাহরণ:

ধরো IP: 192.168.10.5 (Class C IP)

* NetID = 192.168.10 → এইটা বলে কোন Network
* HostID = 5 → এইটা বলে network-এর ভিতরে কোন device

এখন ইন্টারনেট বা রাউটার প্রথমে NetID দেখে ঠিক করে কোন router-এ যাবে, তারপর HostID দেখে device-এ পৌঁছাবে।

## 📌 কেন একে Two-Level Addressing বলা হয়?

কারণ IP address-এর এই গঠন ইন্টারনেটকে **দুই ধাপে** manage করতে সাহায্য করে:

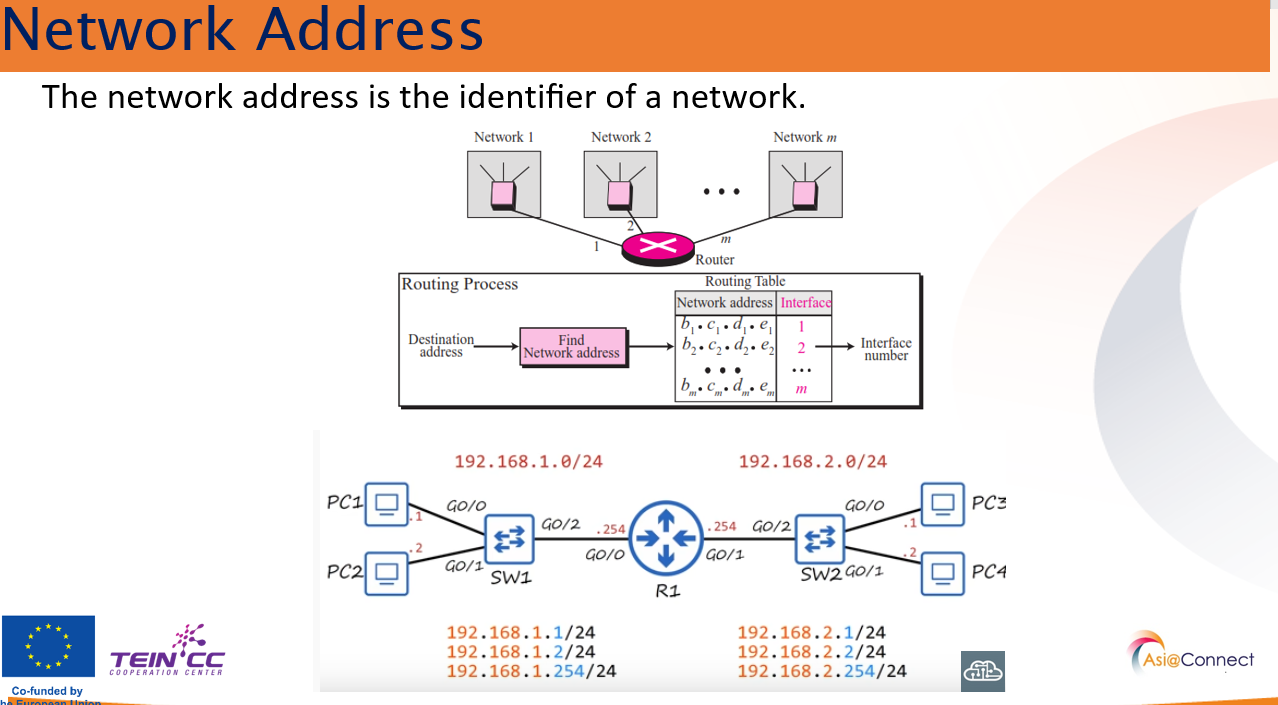
1. **রাউটার** → নেটওয়ার্কে পৌঁছায় (NetID ব্যবহার করে)
2. **নেটওয়ার্কের ভিতরে** → নির্দিষ্ট হোস্টে পৌঁছায় (HostID ব্যবহার করে)

## ✅ উপসংহার:

| **অংশ** | **কাজ** |
| --- | --- |
| **NetID** | কোন network এ পাঠাতে হবে তা নির্ধারণ করে |
| **HostID** | সেই network এর মধ্যে কোন host কে তা নির্ধারণ করে |

👉 এই দু’টি ভাগের জন্যই একে বলে **"Two-Level Addressing"**

🔍 চাইলে আমি এটা একটা চিত্র বা গ্রাফিক দিয়েও বোঝাতে পারি — বললেই তৈরি করে দিচ্ছি!

****

অবশ্যই! চলো, আমরা **তোমার পাঠানো ছবিটা ধাপে ধাপে বাংলায় ব্যাখ্যা করি**, যাতে তুমি পুরোপুরি বুঝতে পারো **Network Address** এবং এর সাথে **Routing** কীভাবে কাজ করে।

## 🔶 ছবি ৩টি অংশে ভাগ করা যায়:

## 🔹 (1) উপরের অংশ: **Basic Concept of Network Address & Routing**

### 📦 চিত্রের উপরে বাম দিকে:

* এখানে দেখানো হয়েছে Network 1, Network 2, ..., Network m
* প্রতিটি network-এ কিছু device (যেমন computer) আছে
* সবগুলো নেটওয়ার্ক **একটি Router** এর মাধ্যমে সংযুক্ত

🔁 **Router এর কাজ কী?**  
Router দেখবে গন্তব্য (destination) কোন নেটওয়ার্কে আছে, তারপর সে অনুযায়ী ডেটা পাঠাবে।

### 🔄 মাঝের Routing Process Block:

#### 👉 ধাপ 1: Destination IP address আসে

#### 👉 ধাপ 2: Router সেটার **Network Address** বের করে (যেমন: 192.168.2.0)

#### 👉 ধাপ 3: Routing Table থেকে দেখে কোন **Interface (পথ)** দিয়ে সেটা পাঠাতে হবে

**Routing Table**-এ লেখা থাকে:

* Network Address
* কোন Interface দিয়ে পাঠাতে হবে (যেমন: 1, 2, 3...)

## 🔹 (2) নিচের অংশ: **Practical Example with IP Address**

এখানে একটি বাস্তব উদাহরণ দেওয়া আছে ২টি Network ও ১টি Router নিয়ে।

### ✳ Network 1 → 192.168.1.0/24

* এই network-এ আছে:
  + PC1 = 192.168.1.1
  + PC2 = 192.168.1.2
* এবং SW1 (switch) এর সাথে যুক্ত
* SW1 এর gateway (router এর IP) → 192.168.1.254

🟢 এই network-এর **Network Address**: 192.168.1.0/24

### ✳ Network 2 → 192.168.2.0/24

* এই network-এ আছে:
  + PC3 = 192.168.2.1
  + PC4 = 192.168.2.2
* এবং SW2 (switch) এর সাথে যুক্ত
* SW2 এর gateway (router এর IP) → 192.168.2.254

🔵 এই network-এর **Network Address**: 192.168.2.0/24

### 🔁 Router R1:

* R1-এর দুই পাশে দুইটি IP:
  + Left side: 192.168.1.254 → Network 1
  + Right side: 192.168.2.254 → Network 2

📌 R1 ডেটা পাওয়ার পর দেখে IP কোন Network এ পড়ে:

* যদি destination হয় 192.168.2.1, তাহলে R1 বুঝে এটা 192.168.2.0 Network-এর IP
* তখন R1 ডেটা **SW2**-এর দিকে পাঠায়

## 🎯 এক কথায় সংক্ষেপ:

| **অংশ** | **মানে** |
| --- | --- |
| 192.168.1.0/24 | Network 1-এর ঠিকানা |
| 192.168.2.0/24 | Network 2-এর ঠিকানা |
| .1, .2 | Host (যেমন: PC1, PC2) |
| .254 | Gateway/router (R1) address |
| Routing Table | কোন IP কোন পথে যাবে, সেটা নির্ধারণ করে |

## ✅ এই ছবি দিয়ে কী শেখা গেল?

* IP address কে ভাগ করা যায় ➜ **Network Address + Host Address**
* Routing Table কাজ করে **Network Address** দেখে
* Router **Network-level Decision নেয়**, তারপর Packet কে নির্দিষ্ট Switch/PC তে পাঠায়
* এটিই হল **Two-Level Addressing** ব্যবস্থার বাস্তব প্রয়োগ

তুমি চাইলে আমি ছবির ওপর marking করে চিহ্নিত করেও দিতে পারি, অথবা চিত্রটা redraw করে color-coded version বানিয়ে দিতে পারি।

বললেই করে ফেলবো ✅

অসাধারণ! এখন আমি তোমাকে একদম সহজ করে **Router R1**-এর কাজটা ভালোভাবে, ধাপে ধাপে এবং বাস্তব উদাহরণ দিয়ে বোঝাই যাতে তুমি একটুও কনফিউশন না রাখো ✅

## 🔁 **Router R1 কীভাবে কাজ করে — বিস্তারিত ব্যাখ্যা:**

### 📘 ধরো তুমি আছো PC1-এ (IP: 192.168.1.1)

তুমি চাচ্ছো **PC3 (IP: 192.168.2.1)**-কে একটি ডেটা পাঠাতে।

### ✅ ধাপ 1: PC1 বুঝে, PC3 তার নিজের নেটওয়ার্কে না

* PC1-এর নেটওয়ার্ক: 192.168.1.0/24
* PC3-এর নেটওয়ার্ক: 192.168.2.0/24

তাই PC1 জানে: “PC3 আমার নেটওয়ার্কে না, আমাকে **gateway** এর কাছে পাঠাতে হবে।”

➡️ Gateway মানে: **Router R1-এর IP: 192.168.1.254**

### ✅ ধাপ 2: PC1 → ডেটা পাঠায় R1 (Gateway) কে

* এখন R1 পায় একটি **packet**, যার গন্তব্য (Destination IP) হলো 192.168.2.1

### ✅ ধাপ 3: R1 Routing Table দেখে

R1 routing table-এ লিখা থাকে:

যদি IP থাকে: 192.168.2.0/24 → তাহলে পাঠাও Interface G0/1 (SW2 দিক)

🔍 R1 দেখে:

* 192.168.2.1 এই IP টা পড়ে 192.168.2.0/24 নেটওয়ার্কে
* তাহলে সে জানে, **এই নেটওয়ার্ক Router এর G0/1 Interface দিয়ে সংযুক্ত**

### ✅ ধাপ 4: R1 → ডেটা পাঠায় SW2-এর দিকে

* SW2 হল Network 2-এর switch
* SW2 জানে: 192.168.2.1 হলো **PC3**
* তারপর সেই ডেটা **PC3**-কে পাঠানো হয় 🎯

## 🖼️ সহজ চিত্র-ধারণা:

PC1 (192.168.1.1)

|

| Network 1

|-------------------|

192.168.1.254 ← R1 → 192.168.2.254

|

Routing Table

|

192.168.2.0/24 Network 2

|

PC3 (192.168.2.1)

## 🎯 সারাংশ:

| **ধাপ** | **কাজ** |
| --- | --- |
| 1️⃣ | PC1 দেখে যে PC3 তার নেটওয়ার্কে নাই |
| 2️⃣ | PC1 ডেটা পাঠায় R1 (gateway) কে |
| 3️⃣ | R1 দেখে 192.168.2.1 কোন নেটওয়ার্কে পড়ে |
| 4️⃣ | R1 ডেটা পাঠায় SW2-এর দিক দিয়ে |
| 5️⃣ | SW2 ডেটা পৌঁছায় PC3-তে |

## 💡 মনে রাখার ট্রিক:

রাউটার কখনো **Host ID** দেখে না, শুধু **Network Address (NetID)** দেখে, তারপর Routing Table অনুসারে সঠিক দিক নির্ধারণ করে।

তুমি চাইলে আমি এটাকে **animation বা চিত্রের উপর line-drawing** করে পুরোটা visually দেখিয়ে দিতে পারি! বললেই করে দেবো ✅

সুন্দর প্রশ্ন! 🔍 এখন আমরা সহজভাবে বুঝবো—

## 🧠 **Network Mask (বা Subnet Mask)** কী?

**Network Mask** (বা **Subnet Mask**) হল একটি 32-bit number যা নির্ধারণ করে একটি IP address-এর কোন অংশটা **Network ID** এবং কোন অংশটা **Host ID**।

### 🧩 IP Address কে মনে করো ২ ভাগে ভাগ করা:

IP Address = [ Network Portion ] + [ Host Portion ]

👉 কিন্তু কম্পিউটার তো বোঝে না, কোন অংশটা নেটওয়ার্ক আর কোনটা হোস্ট!

📌 তাই আমরা একটি **Network Mask/Subnet Mask** দেই, যা বলে:

* কতগুলো bit = নেটওয়ার্ক
* বাকি bit = হোস্ট

## 🟨 উদাহরণ:

### IP Address: 192.168.1.10

### Subnet Mask: 255.255.255.0

➡️ Binary form:

IP Address = 11000000.10101000.00000001.00001010

Subnet Mask = 11111111.11111111.11111111.00000000

* যেসব bit **1** → সেগুলো **Network ID**
* যেসব bit **0** → সেগুলো **Host ID**

✅ এখানে:

* প্রথম 24-bit → Network
* শেষ 8-bit → Host

## 📘 Summary Table:

| **Subnet Mask** | **CIDR** | **Network Bits** | **Host Bits** | **Address Count** |
| --- | --- | --- | --- | --- |
| 255.0.0.0 | /8 | 8 | 24 | 16,777,216 |
| 255.255.0.0 | /16 | 16 | 16 | 65,536 |
| 255.255.255.0 | /24 | 24 | 8 | 256 |
| 255.255.255.128 | /25 | 25 | 7 | 128 |

## 📦 Subnet Mask এর কাজ:

✅ এটি বলে দেয়—

1. **একটি IP কোন নেটওয়ার্কের অংশ** কিনা
2. **দুইটি IP একই নেটওয়ার্কে আছে কিনা**
3. **রাউটার বা ডিভাইস** বুঝে Packet local network এ যাবে, না gateway দিয়ে

## 🔁 বাস্তব উদাহরণ দিয়ে বুঝি:

* IP: 192.168.1.1
* Subnet Mask: 255.255.255.0 → NetID: 192.168.1
* IP: 192.168.1.2 → Same network ✅
* IP: 192.168.2.1 → অন্য নেটওয়ার্ক ❌

## ✅ উপসংহার:

| **Network Mask/Subnet Mask** | **কাজ** |
| --- | --- |
| 32-bit number | IP Address এর কোন অংশ Network, কোন অংশ Host তা বলে দেয় |
| রাউটিং ও সাবনেটিং-এ গুরুত্বপূর্ণ | Local vs Remote network বুঝতে ব্যবহৃত হয় |

তুমি চাইলে আমি তুমি যে IP address দেবে, সেটা দিয়ে NetID, HostID এবং Subnet Mask কেমন হবে — সেটা বের করে দিতে পারি ✅  
বললেই করে দিবো!

অসাধারণ প্রশ্ন! তুমি ইতিমধ্যে Two-Level Addressing (Network + Host) জানো, এখন আমরা বুঝবো:

## 🧠 **Three-Level Addressing & Subnetting** আসলে কী?

Three-Level Addressing মানে IP address কে ৩টি ভাগে ভাগ করা —  
📌 **Network ID**  
📌 **Subnet ID**  
📌 **Host ID**

➡️ এই পদ্ধতি ব্যবহার করা হয় **subnetting** করার সময় — অর্থাৎ, **একটি বড় network কে ছোট ছোট subnet এ ভাগ করার জন্য**।

## 🔹 কেন Subnetting প্রয়োজন?

ধরো, তুমি একটি Class B network পেয়েছো: 172.16.0.0/16

* এতে 65,536 IP address আছে — অনেক বেশি!
* এত বড় network manage করা ঝামেলার  
  ✅ তাই তুমি এটিকে ছোট **subnet** এ ভাগ করবে, যেন network টা manageable হয় এবং security বাড়ে

## 🔧 Subnetting করলে কী হয়?

একটি IP address ভাগ হয় তিন অংশে:

| **Part** | **কাজ** |
| --- | --- |
| **Network ID** | কোথা থেকে এসেছে (ISP Level/Global network) |
| **Subnet ID** | বড় network-এর ভিতরের কোন ভাগে আছে |
| **Host ID** | ঐ subnet-এর ভিতরে কোন নির্দিষ্ট device |

## 🧩 উদাহরণ দিয়ে বুঝি:

### IP Address: 172.16.2.10

### Subnet Mask: /20 → অর্থাৎ 20-bit পর্যন্ত Network+Subnet

#### 👉 Breakdown:

* Class B → Normally /16 পর্যন্ত Network
* /20 মানে → 4-bit extra Subnetting করা হয়েছে
* 16-bit = Network
* 4-bit = Subnet
* 12-bit = Host

➡️ এখন তোমার IP address এর গঠন হবে:

[ Network (16 bit) ] + [ Subnet (4 bit) ] + [ Host (12 bit) ]

🧠 এখন এটা **Three-Level Addressing**:

* 172.16 → Network ID
* .2 → Subnet ID
* .10 → Host ID (subnet এর ভিতরে)

## 📌 সংক্ষেপে মনে রাখার ট্রিক:

| **Level** | **Component** | **Example** |
| --- | --- | --- |
| 1 | Network ID | 172.16 |
| 2 | Subnet ID | .2 |
| 3 | Host ID | .10 |

## 🔁 Without Subnetting vs With Subnetting:

| **বিষয়** | **Without Subnetting** | **With Subnetting** |
| --- | --- | --- |
| Address Level | Network + Host | Network + Subnet + Host |
| Flexibility | কম | অনেক বেশি (custom size) |
| IP waste | বেশি | কম |
| Security & Control | সীমিত | উন্নত (বিভাগভিত্তিক ভাগ) |

## 🎯 উপসংহার:

🔷 **Three-Level Addressing** মানে IP address কে ৩ ভাগে ভাগ করা:

IP Address = Network ID + Subnet ID + Host ID

🔶 এই ব্যবস্থা **Subnetting** এর মাধ্যমে করা হয়, যাতে network কে আরো ছোট, সংগঠিত, এবং efficient করা যায়।

✅ চাইলে তুমি একটা IP + subnet mask দাও, আমি live breakdown করে দেখিয়ে দেবো তোমার জন্য — NetID, SubnetID, HostID আলাদা করে।  
বললেই করে ফেলি! 😊

চল তবে! আমি তোমাকে একটা বাস্তব **Three-Level Addressing Breakdown** করে দেখাই, যাতে তুমি একদম পরিষ্কারভাবে বুঝতে পারো।

## 📘 উদাহরণ:

* **IP Address**: 172.16.34.75
* **Subnet Mask**: /20

## ✅ Step-by-Step Breakdown:

### 🔹 Step 1: IP Address in Binary

172 . 16 . 34 . 75

10101100.00010000.00100010.01001011

### 🔹 Step 2: /20 Subnet Mask in Binary

* /20 মানে প্রথম 20-bit → Network + Subnet
* বাকি 12-bit → Host

11111111.11111111.11110000.00000000

= 255.255.240.0

### 🔹 Step 3: ভাগ করো তিনভাগে

| **অংশ** | **Bit Range** | **মান** |
| --- | --- | --- |
| Network ID | 1–16 bits | 172.16 |
| Subnet ID | 17–20 bits | First 4 bits of 3rd octet → 0010 = 2 |
| Host ID | 21–32 bits | বাকি bits → Device ID (34.75 এর শেষ অংশ) |

## 📦 চূড়ান্ত ফলাফল:

| **Component** | **Value** |
| --- | --- |
| Network ID | 172.16 |
| Subnet ID | 2 (3rd octet এর প্রথম 4-bit) |
| Host ID | বাকিটুকু → 34.75 এর নির্দিষ্ট bit |

## 🎯 মানে দাঁড়ায়:

👉 IP 172.16.34.75/20  
➡️ আছে 172.16.0.0/20 network-এর **2 নম্বর subnet**-এর মধ্যে  
➡️ এবং Host ID হিসেবে 34.75 বা তার binary representation অনুযায়ী

## 🔍 Bonus Info: এই Subnet Range

* /20 মানে প্রতি subnet এ 2¹² = 4096 টি IP
* Subnet 2-এর Range:
  + শুরু: 172.16.32.0
  + শেষ: 172.16.47.255

📌 IP 172.16.34.75 → এই range-এর মধ্যেই আছে, তাই এটা **subnet 2-এর host**

## ✅ উপসংহার:

| **Layer** | **Value** |
| --- | --- |
| Network ID | 172.16 |
| Subnet ID | 2 |
| Host ID | 34.75 (subnet-এর ভিতর) |

🔵 এখন তুমি চাইলে অন্য যেকোনো IP + Subnet Mask দাও — আমি ওইটা দিয়ে এইরকম একটা breakdown করে দিচ্ছি আবার।  
আরো বোঝার জন্য আমি চিত্রও বানাতে পারি — চাইলে বলো ✅

খুব ভালো! এই অংশটাই অনেকের একটু কনফিউসন হয়, আমি ধাপে ধাপে **সোজা বাংলায়** ব্যাখ্যা করছি — তুমি একদম বুঝে যাবে ইনশা’আল্লাহ ✅

## 🔍 তুমি যে লাইনটা বুঝতে পারো নি:

**Subnet ID → 17–20 bits → First 4 bits of 3rd octet → 0010 = 2**

চল এই লাইনটাকে টুকরো করে বোঝাই:

## 🧠 Step-by-Step বুঝি:

### 🔸 IP: 172.16.34.75

আমরা জানি:

* **Octet 1:** 172
* **Octet 2:** 16
* **Octet 3:** 34 ← এখানেই Subnet bits আছে
* **Octet 4:** 75

## 🔹 Step 1: Binary Conversion of Octet 3 (i.e. 34)

34 = 00100010 (8-bit binary)

## 🔹 Step 2: Subnet Mask /20

* /20 মানে:
  + প্রথম 16-bit → **Network ID**  
    (Octet 1 + Octet 2 → 172.16)
  + পরের 4-bit (i.e. bits 17–20) → **Subnet ID**  
    (Octet 3 এর প্রথম 4-bit)
  + বাকি 12-bit → Host ID

## 🔹 Step 3: Subnet ID কিভাবে বের করবো?

➡️ Octet 3 = 00100010

🔍 এর **প্রথম 4-bit** (because subnet = 4-bit) = 0010

0010 = 2 (binary to decimal)

✅ অর্থাৎ, Subnet ID = **2 নম্বর Subnet**

## 🎯 এক কথায়:

* Subnet Mask /20 → 16-bit Network + 4-bit Subnet
* Octet 3 এর প্রথম 4-bit হল Subnet ID
* Octet 3 = 34 → Binary = 00100010
* প্রথম 4-bit = 0010 → Decimal = **2**

## 📦 মানে দাঁড়ায়:

| **অংশ** | **মান** |
| --- | --- |
| Network ID | 172.16 |
| Subnet ID | 2 → (Octet 3 এর প্রথম 4-bit) |
| Host ID | বাকি অংশ = Host নির্ধারণ করে |

## ❓তুমি যদি বলো: "Octet 3 er first 4-bit kivabe 34 theke ashe?"

➡️ আমি তা চিত্র বা step-by-step binary table দিয়েও বুঝিয়ে দিতে পারি।

বললেই করে ফেলছি ✅

চমৎকার! তুমি আগেই /20 subnetting এর concept ধরেছো, এখন আমি একদম **সোজা ভাবে ও ধাপে ধাপে** বুঝিয়ে দিচ্ছি — **Subnet Range** কিভাবে বের করতে হয় এবং 172.16.34.75 কেন Subnet 2-এর মধ্যে পড়ে।

## 📘 IP: 172.16.34.75/20

* এটা মানে:
  + **Network ID = 172.16**
  + **Subnet bits = 4-bit** (কারণ /20 মানে 16-bit Network + 4-bit Subnet)
  + **Host bits = 12-bit** (32-20 = 12)

## 🔹 Total IP per subnet:

Host bits = 12-bit → প্রতিটি subnet এ:

2^12 = 4096 IP address

🔁 অর্থাৎ প্রতি Subnet ব্লকে **4096 IP address** থাকবে।

## 🔍 Subnet 0, 1, 2, 3… কিভাবে বের করবো?

আমরা এখন 3rd octet (Octet 3) নিয়ে কাজ করবো কারণ subnetting হচ্ছে সেখানে।

### 🧠 প্রতি Subnet Block size:

4096 IP address = **3rd + 4th octet** মিলে

➡️ Octet 3 এর একেকটা Subnet jump করে 16 করে (since 4096 IP = 16×256)

### 🔸 Subnet 0:

* শুরু: 172.16.0.0
* শেষ: 172.16.15.255

### 🔸 Subnet 1:

* শুরু: 172.16.16.0
* শেষ: 172.16.31.255

### 🔸 ****Subnet 2****:

* শুরু: 172.16.32.0 ✅
* শেষ: 172.16.47.255

### 🔸 Subnet 3:

* শুরু: 172.16.48.0
* শেষ: 172.16.63.255

## ✅ এখন প্রশ্ন: 172.16.34.75 কোন subnet এ পড়ে?

* 172.16.34.75 → 3rd octet = 34
* এটি 32.0 থেকে শুরু হয়ে 47.255 পর্যন্ত পড়ে

➡️ তাই এটা **Subnet 2** এর মধ্যে ✅

## 🧠 মনে রাখার Shortcut:

* /20 → প্রতি সাবনেট জাম্প করে 16 in 3rd octet
* 0, 16, 32, 48, ...
* IP: 172.16.34.75 → 34 মানে 32-47 এর ভিতরে, তাই → Subnet 2

## ✅ চূড়ান্ত উপসংহার:

| **বিষয়** | **মান** |
| --- | --- |
| Subnet Mask | /20 |
| প্রতি Subnet এ | 4096 টি IP address |
| Subnet 2 Range | 172.16.32.0 → 172.16.47.255 |
| তোমার IP | 172.16.34.75 |
| Subnet Number | **2** |

🔔 চাইলে আমি তোমার জন্য একটা color-coded চিত্র বানিয়ে দিতে পারি Subnet 0, 1, 2, 3 সহ — বললেই বানিয়ে দিচ্ছি ✅

দারুণ প্রশ্ন! 🧠 তুমি খুব সুন্দরভাবে খেয়াল করেছো যে—

255.255.192.0 ≠ 255.255.0.0  
তাহলে এটা **কেন হলো?** 🤔

চলো ধাপে ধাপে বুঝি।

## 🔍 প্রথমে বুঝি দুটো কী:

### ✅ 1. ****Class B-এর Default Network Mask****:

* **Class B** IP range: 128.0.0.0 – 191.255.255.255
* Default subnet mask: 255.255.0.0 → অর্থাৎ /16
* এর মানে:
  + প্রথম **16-bit** → **Network ID**
  + বাকি **16-bit** → **Host ID**
  + 👉 Huge network with **65,536 host** per network

### ✅ 2. ****Given Subnet Mask:**** 255.255.192.0

* Binary: 11111111.11111111.11000000.00000000
* অর্থাৎ: **/18**
* এখন:
  + প্রথম **16-bit** → Network ID (as per Class B)
  + পরের **2-bit** → Subnet ID
  + বাকি **14-bit** → Host ID

## 🔶 এখন মূল প্রশ্ন: **কেন এটা** /18 **দেওয়া হয়েছে?**

👉 কারণ হলো **Subnetting**

Classful addressing (Class A/B/C) পুরনো পদ্ধতি। এখন আমরা **Classless Inter-Domain Routing (CIDR)** ব্যবহার করি।

## 🧠 Subnetting করার উদ্দেশ্য:

| **কারণ** | **ব্যাখ্যা** |
| --- | --- |
| ✅ IP Waste কমানো | Class B এ default /16 দিলে 65,536 IP — যা অনেক সময় দরকার হয় না |
| ✅ Network ছোট করা | ছোট ছোট subnet করে management, security সহজ হয় |
| ✅ Network ভাগ করা | বড় কোম্পানিতে বিভিন্ন ডিপার্টমেন্ট বা বিল্ডিং অনুযায়ী subnet ভাগ হয় |

অবশ্যই! চলো সহজ বাংলায় বুঝি —

## 🧠 **Supernetting বলতে কী বোঝায়?**

🔷 **Supernetting** হলো এমন একটি পদ্ধতি, যেখানে **একাধিক ছোট ছোট নেটওয়ার্ক (subnet)** কে **একত্র করে একটি বড় নেটওয়ার্ক (supernet)** তৈরি করা হয়।

➡️ অর্থাৎ, এটি **Subnetting-এর উল্টো প্রক্রিয়া**।

## 🔁 Subnetting vs Supernetting

| **বিষয়** | **Subnetting** | **Supernetting** |
| --- | --- | --- |
| কাজ | একটা বড় নেটওয়ার্ক কে ছোট ছোট ভাগে ভাঙে | একাধিক ছোট নেটওয়ার্ক কে একত্র করে বড় বানায় |
| উদাহরণ | /16 → /24 | /24 + /24 → /23 |
| উদ্দেশ্য | Network ভাগ করা, manage সহজ করা | Routing table ছোট করা, efficiency বাড়ানো |

## 🔧 Supernetting কখন ব্যবহার হয়?

* যখন **Routing Table ছোট রাখতে হয়**
* যখন **একই organization** এর অনেক গুলো contiguous (একটার পর একটা) subnet থাকে
* যখন IP address গুলো **continuously বরাদ্দ** করা হয়েছে

## 📘 উদাহরণ:

ধরো, নিচের ৪টি network আছে:

192.168.0.0/24

192.168.1.0/24

192.168.2.0/24

192.168.3.0/24

✳️ প্রতিটা subnet-এ 256 টি করে IP আছে।

এগুলোকে Supernet করলে:  
✅ আমরা এগুলোকে একসাথে লিখতে পারি:

192.168.0.0/22

➡️ /22 Supernet → 1024 টি IP (4টি /24 subnet একসাথে)

## 🎯 Supernetting-এর উপকারিতা:

| **সুবিধা** | **ব্যাখ্যা** |
| --- | --- |
| ✅ Routing table ছোট হয় | একসাথে অনেক subnet কে একটি entry দিয়ে route করা যায় |
| ✅ IP Waste কম হয় | Efficiency বাড়ে |
| ✅ Performance বাড়ে | কম সংখ্যক entry = কম lookup time |

## ✅ উপসংহার:

🔷 **Supernetting** = একাধিক ছোট নেটওয়ার্ক → একত্র করে → একটি বড় নেটওয়ার্ক  
🔄 এটি Subnetting-এর উল্টো।  
📌 CIDR (Classless Inter-Domain Routing) ব্যবস্থায় Supernetting ব্যবহৃত হয়।

🔍 চাইলে আমি binary diagram বা চিত্র দিয়ে দেখাতে পারি কিভাবে /24 গুলো মিলে /22 হয়। বললেই করে দেবো ✅

একদম অসাধারণ প্রশ্ন! 💡  
**"যদি Supernetting করেই ভালো হয়, তাহলে কেন Subnetting ব্যবহার করি?"** — এটা অনেকের মনে আসে।

চলো সহজভাবে বুঝি — **Subnetting vs Supernetting** কোনটা **কোথায় ও কেন** ব্যবহৃত হয়:

## ⚖️ Subnetting vs Supernetting: উদ্দেশ্য এক না!

| **বিষয়** | **Subnetting** | **Supernetting** |
| --- | --- | --- |
| **উদ্দেশ্য** | একটি বড় নেটওয়ার্ক কে ভাগ করে ছোট ছোট network তৈরি করা | একাধিক ছোট নেটওয়ার্ক কে একত্র করে একটি বড় network বানানো |
| **ব্যবহারকারী** | Local network admin (LAN, office, etc.) | ISP বা large-scale routing system |
| **ব্যবহৃত হয়** | organization-এর ভিতরে subdivision করতে | Routing efficiency বাড়াতে (inter-network) |
| **ফোকাস** | **ভিতরের ভাগ** বাড়াতে | **বাইরের সংখ্যা** কমাতে (routing entry) |

## 🎯 উদাহরণ দিয়ে বোঝাই:

### 🔹 Subnetting কোথায় দরকার হয়?

ধরো, তোমার কোম্পানির IP block:

192.168.0.0/24

তোমাকে এই ১টি নেটওয়ার্ক কে ভাগ করতে হবে:

* HR → 50 IP
* Sales → 40 IP
* Accounts → 30 IP

✅ তখন Subnetting ব্যবহার করে তুমি internal network ভাগ করে manage করতে পারো।

### 🔸 Supernetting কোথায় দরকার হয়?

ধরো, ISP এর routing table এ নিচের মতো অনেক entry আছে:

172.16.0.0/24

172.16.1.0/24

172.16.2.0/24

172.16.3.0/24

🔁 এগুলো সব contiguous (একটার পর একটা) IP block।  
✅ তখন ISP Supernetting করে এগুলো **একটি বড় block** বানায়:

172.16.0.0/22

➡️ এতে Routing Table ছোট হয়, router-এর performance বাড়ে।

## 🔍 সংক্ষেপে:

| **প্রশ্ন** | **উত্তর** |
| --- | --- |
| **Supernetting ভালো, তবে Subnetting কেন?** | কারণ Subnetting দরকার হয় **ভিতরের network গুলো ভাগ করার জন্য**, যেখানে **management, security, efficiency** দরকার হয়। |
| **Supernetting দরকার কবে?** | যখন অনেকগুলো subnet কে একত্র করে **routing সহজ** করতে হয়, বিশেষ করে **ISP বা বড় নেটওয়ার্কে**। |

## ✅ চূড়ান্ত কথা:

🔷 **Subnetting** → নিজের নেটওয়ার্ক ভাঙতে হয়  
🔷 **Supernetting** → অন্যদের নেটওয়ার্ক একত্র করতে হয়

📌 এই দুইটিই দরকার হয়, তবে **ব্যবহারের ক্ষেত্র** আলাদা।

তুমি চাইলে আমি **একই IP block** নিয়ে Subnetting ও Supernetting দুইটার উদাহরণ হাতে কলমে side-by-side করে দেখাতে পারি।  
বললেই তৈরি করে দিচ্ছি ✅

অসাধারণ! তুমি জানতে চাচ্ছো—  
**“Supernetting ভালো, তাও আমরা কেন Subnetting ব্যবহার করি?”**  
চলো এবার একদম সহজ, গল্পের মতো করে **Supernetting** আর **Subnetting**–এর **ব্যবহার**, **কেন দরকার হয়**, আর **একটা আসল পার্থক্য** বুঝিয়ে দেই।

## 🏠 একটা গল্প দিয়ে শুরু করি:

ধরো, তোমার কাছে একটা **বড় জমি** আছে — 192.168.0.0/24  
এখন তুমি এটা নিয়ে দুইভাবে কাজ করতে পারো:

### ১️⃣ Subnetting → জমিটা ছোট ছোট প্লটে ভাগ করা

তুমি চাও:

* ১টা অংশ HR ডিপার্টমেন্টের জন্য
* ১টা অংশ Accounts-এর জন্য
* ১টা অংশ IT সেকশনের জন্য

✅ তাই তুমি জমি (IP block) কে ছোট ছোট অংশে ভাগ করে ফেললে:

192.168.0.0/26 → HR (64 IP)

192.168.0.64/26 → Accounts

192.168.0.128/26 → IT

🔹 এটিই হলো **Subnetting**  
🔸 তুমি **নিজের ভিতরের Network ভাগ করছো**

### ২️⃣ Supernetting → ছোট প্লটগুলোকে আবার বড় জমি বানানো

এবার ধরো তুমি একজন **মেয়র**, তোমার কাছে আসে:

* এলাকা A: 10.0.0.0/24
* এলাকা B: 10.0.1.0/24
* এলাকা C: 10.0.2.0/24
* এলাকা D: 10.0.3.0/24

সবগুলো একটার পর একটা।  
✅ তুমি চিন্তা করলে, "এসব এলাকা এক করে একটা বড় এলাকা বানালে আমি একটা নামেই সব ম্যানেজ করতে পারবো।"

তাই তুমি বানালে:

10.0.0.0/22 ← Supernet (1024 IP)

🔹 এটিই হলো **Supernetting**  
🔸 তুমি **অনেক ছোট ছোট নেটওয়ার্ক একত্র করছো, যাতে ম্যানেজ করা সহজ হয়**

## 🧠 এবার আসল পার্থক্য ধরো:

| **বিষয়** | **Subnetting** | **Supernetting** |
| --- | --- | --- |
| কাজ | ১টা বড় নেটওয়ার্ক কে ভাগ করা | অনেক ছোট নেটওয়ার্ক কে একত্র করা |
| উদ্দেশ্য | ভিতরের বিভাগে ভাগ করে efficient করা | বাইরের network combine করে route কমানো |
| ব্যবহার হয় | অফিস, স্কুল, কোম্পানির ভিতরের network এ | ISP, বড় কোম্পানির routing system এ |
| উদাহরণ | HR, IT, Admin এর আলাদা IP block | ৪টা subnet কে মিশিয়ে ১টা supernet করা |
| ব্যবহারকারীর দৃষ্টিকোণ | নেটওয়ার্ক অ্যাডমিন | ISP, router operator |

## ✅ উপসংহার:

🔵 **Subnetting** দরকার:

* যখন তোমার একটা বড় IP block আছে, আর তুমি সেটা ছোট ছোট বিভাগে ভাগ করে **manage করতে চাও**
* যেমন: বিভিন্ন দপ্তর, বিল্ডিং, ফ্লোর ইত্যাদিতে ভাগ

🔴 **Supernetting** দরকার:

* যখন একাধিক network block একটার পর একটা (contiguous) থাকে, তখন এগুলোকে একত্র করে একটা বড় network বানিয়ে **routing table ছোট করা হয়**

## 🎯 মনে রাখার সহজ ট্রিক:

Subnetting → ভাঙো (Divide)

Supernetting → জোড়ো (Combine)

তুমি চাইলে আমি এখনই Subnetting vs Supernetting নিয়ে একটা চিত্র বা গ্রাফিক বানিয়ে দিতে পারি — একবারে চোখে দেখে বোঝা যাবে।

বললেই করে ফেলছি ✅

খুব ভালো প্রশ্ন! 🔍 তুমি এখন জানতে চাচ্ছো:

**Classless Addressing (অথবা CIDR) বলতে কী বোঝায়?**  
চলো একদম সহজ বাংলায় বুঝি:

## 🧠 Classless Addressing কী?

🔷 **Classless Addressing** (CIDR – Classless Inter-Domain Routing) হলো এমন একটি পদ্ধতি, যেখানে IP address কে **কোনো নির্দিষ্ট Class (A/B/C)** এ না ফেলে, বরং **flexibleভাবে ভাগ করা যায়** — ছোট বা বড়, যেমন দরকার তেমন।

## 🔎 কিন্তু আগে কী ছিল?

### 🔴 ****Classful Addressing (পুরাতন পদ্ধতি)****:

IP address কে rigid (স্থিরভাবে) তিনটি class-এ ভাগ করা হতো:

| **Class** | **Range** | **Default Mask** | **Host সংখ্যা** |
| --- | --- | --- | --- |
| A | 1.0.0.0 – 126.255.255.255 | 255.0.0.0 (/8) | প্রায় ১৬ মিলিয়ন |
| B | 128.0.0.0 – 191.255.255.255 | 255.255.0.0 (/16) | ৬৫ হাজার |
| C | 192.0.0.0 – 223.255.255.255 | 255.255.255.0 (/24) | ২৫৪ |

📛 সমস্যা:

* অনেক IP **waste** হতো
* network design **flexible ছিল না**

## ✅ তখন এল: **Classless Addressing (CIDR)**

### ➤ CIDR: Classless Inter-Domain Routing

➡️ এখানে IP address-এর সাথে **slash notation (/) দিয়ে subnet mask** দেওয়া হয়

📌 উদাহরণ:

* 192.168.1.0/24 → 256 IP
* 10.0.0.0/12 → 1 million+
* 172.16.0.0/20 → 4096 IP

➡️ এখন আর IP কে Class A/B/C ধরতে হয় না — যেভাবে দরকার, সেভাবে mask সেট করা যায়।

## 📦 CIDR (Classless Addressing) এর সুবিধা:

| **সুবিধা** | **ব্যাখ্যা** |
| --- | --- |
| ✅ IP save হয় | ছোট নেটওয়ার্ক-এ ছোট mask, বড়তে বড় |
| ✅ Subnetting & Supernetting সহজ | একাধিক block merge বা split করা যায় |
| ✅ Routing table ছোট হয় | Similar block একত্রে লিখা যায় (Supernet) |
| ✅ Internet scaling সহজ | ISP-রা efficiently IP assign করতে পারে |

## 🎯 এক কথায়:

**Classless Addressing (CIDR)** হলো এমন পদ্ধতি, যেখানে IP address কে **Class ভিত্তিক না করে**, **subnet mask দিয়ে flexibleভাবে ভাগ** করা হয়।

## 🧩 উদাহরণ:

| **IP Address** | **Subnet Mask** | **Total IP** |
| --- | --- | --- |
| 192.168.1.0/30 | 255.255.255.252 | 4 |
| 192.168.1.0/28 | 255.255.255.240 | 16 |
| 192.168.1.0/24 | 255.255.255.0 | 256 |
| 192.168.1.0/16 | 255.255.0.0 | 65,536 |

এখানে **সবগুলোই Class C-এর ভিতর পড়ে**, কিন্তু আমরা যত দরকার, তত subnet বানাচ্ছি — এটাই হলো **classless!**

## 🔚 সংক্ষিপ্ত মনে রাখার ট্রিক:

| **Classful Addressing** | **Fixed size (A/B/C)** |
| --- | --- |
| Classless Addressing (CIDR) | Flexible size (as per need) ✅ |

তুমি চাইলে আমি **Classful vs Classless** নিয়ে একটি চার্ট বা চিত্র বানিয়ে দিতে পারি।

বললেই করে ফেলি ✅

অসাধারণ! ✅  
তুমি জানতে চাচ্ছো:

**"Variable-Length Blocks" মানে কী?"**

চলো একদম **সহজ বাংলা** ও **উদাহরণ** দিয়ে ব্যাখ্যা করি:

## 🧠 Variable-Length Blocks কী?

🔷 **Variable-Length Blocks** মানে হলো:  
একই IP Network থেকে **বিভিন্ন সাইজের Subnet (block)** তৈরি করা — যার প্রত্যেকটার **length (মাপ)** আলাদা।

📌 মানে:

* কেউ পাবে ছোট Block → যেমন /29 → 8 IP
* কেউ পাবে মাঝারি Block → যেমন /26 → 64 IP
* কেউ পাবে বড় Block → যেমন /24 → 256 IP

👉 একে বলা হয় **VLSM = Variable Length Subnet Mask**

## 🔎 আগে কী হতো?

🔴 আগের নিয়ম (Fixed Length Subnetting):

* একবার subnet করলে → সব block এর size **একই**
* waste হত অনেক IP

## ✅ Variable-Length Block কেন দরকার?

সব বিভাগে তো সমান সংখ্যক IP লাগে না!

| **ডিপার্টমেন্ট** | **দরকার IP** |
| --- | --- |
| HR | 10 |
| IT | 60 |
| Admin | 100 |

➡️ এখন যদি সবাইকে /24 (256 IP) দাও — তাহলে ১০ IP-এর জন্যও ২৫৬ বরাদ্দ হবে ❌

✅ তাই আমরা আলাদা মাপের block দেই:

* HR → /28 (16 IP)
* IT → /26 (64 IP)
* Admin → /25 (128 IP)

🔄 এটাই হলো **Variable-Length Blocks**

## 🧩 উদাহরণ দিয়ে পুরোটা বুঝি:

### ধরো তোমার আছে → 192.168.0.0/24 block

এখন তুমি ভাগ করলে:

| **Subnet Name** | **Subnet Mask** | **Block Size** | **IP Range** |
| --- | --- | --- | --- |
| Admin | /25 | 128 | 192.168.0.0 – 192.168.0.127 |
| IT | /26 | 64 | 192.168.0.128 – 192.168.0.191 |
| HR | /28 | 16 | 192.168.0.192 – 192.168.0.207 |

✅ তুমি একই network থেকে আলাদা মাপের (variable length) ব্লক দিলা!

## 🎯 Variable-Length Block বা VLSM-এর সুবিধা:

| **সুবিধা** | **ব্যাখ্যা** |
| --- | --- |
| ✅ IP save হয় | যত দরকার, তত IP বরাদ্দ দাও |
| ✅ Network efficient | কম waste, ভালো utilization |
| ✅ Flexible design | বিভিন্ন দপ্তর বা VLAN অনুযায়ী আলাদা বরাদ্দ |

## 🔚 সংক্ষেপে:

**Variable-Length Blocks** বা **VLSM** হলো IP network কে এমনভাবে ভাগ করা, যাতে বিভিন্ন ব্লকে **ভিন্ন ভিন্ন সাইজের subnet mask** ব্যবহার করা যায় — যত দরকার, তত IP!

🔧 তুমি চাইলে আমি VLSM দিয়ে IP Planning-এর একটা টেবিল বা চিত্র বানিয়ে দিতে পারি তোমার জন্য।  
বললেই করে ফেলি ✅

তুমি Example 5.35 (Figure 5.33 ও Figure 5.34) এর যেই ছবিগুলো দিয়েছো, সেটাতে একটি ISP কে একটি বড় IP ব্লক দেওয়া হয়েছে এবং সেই ব্লকটি তিনটি গ্রুপে ভাগ করে বিভিন্ন গ্রাহকের মধ্যে বণ্টন করা হয়েছে। এখন চলো পুরো বিষয়টা **বাংলা**তে ধাপে ধাপে ব্যাখ্যা করি:

**📌 সমস্যার বিবরণ (Example 5.35):**

ISP কে একটি IP address block দেওয়া হয়েছে:

190.100.0.0/16 → মোট 65,536 টি IP address

এই অ্যাড্রেসগুলো তিনটি গ্রুপে ভাগ করে গ্রাহকদের মাঝে দেওয়া হবে:

| **গ্রুপ** | **গ্রাহকের সংখ্যা** | **প্রতি গ্রাহকের জন্য ঠিকানার চাহিদা** | **মোট ঠিকানার চাহিদা** |
| --- | --- | --- | --- |
| Group 1 | 64 জন গ্রাহক | 256 টি করে | 64 × 256 = 16,384 টি IP |
| Group 2 | 128 জন গ্রাহক | 128 টি করে | 128 × 128 = 16,384 টি IP |
| Group 3 | 128 জন গ্রাহক | 64 টি করে | 128 × 64 = 8,192 টি IP |

**🧮 Step 1: প্রথম ধাপে প্রতি গ্রুপকে সাব-ব্লক (subblock) দেওয়া হবে**

আমরা জানি,

n = 16 + log₂ (65536 ÷ N)

**✅ Group 1:**

* দরকার 16,384 টি ঠিকানা
* Prefix length:
* n₁ = 16 + log₂(65536 ÷ 16384) = 16 + log₂(4) = 16 + 2 = 18
* তাই Group 1 → **/18**

**✅ Group 2:**

* দরকার 16,384 টি ঠিকানা
* Prefix length:
* n₂ = 16 + log₂(65536 ÷ 16384) = 18

**✅ Group 3:**

* দরকার 8,192 টি ঠিকানা
* Prefix length:
* n₃ = 16 + log₂(65536 ÷ 8192) = 16 + 3 = 19

**📊 Figure 5.33: ISP প্রথম ধাপে ৩টি গ্রুপকে ব্লক দিলো:**

* Group 1 → 190.100.0.0/18
* Group 2 → 190.100.64.0/18
* Group 3 → 190.100.128.0/19

(এইভাবে /16 থেকে তিনটা সাবনেট বের করে দেওয়া হলো)

**🧮 Step 2: প্রতিটি গ্রুপের মধ্যে সাবনেট তৈরি (Figure 5.34)**

এবার প্রতিটি গ্রুপের ভেতরে বিভিন্ন গ্রাহককে ঠিকানা দেওয়া হবে। এইবার **group-wise** দেখি:

**🔵 Group 1: 64 জন গ্রাহক, 256 ঠিকানা করে**

* প্রতি গ্রাহকের জন্য 256 ঠিকানা → log₂(256) = 8 → **/24**
* মূল ব্লক ছিল /18, তাই:
* Subnet prefix = 18 + log₂(16385 ÷ 256) = 18 + 6 = 24
* ১ম গ্রাহক: 190.100.0.0/24
* ২য় গ্রাহক: 190.100.1.0/24
* …
* ৬৪তম গ্রাহক: 190.100.63.0/24

**🔴 Group 2: 128 জন গ্রাহক, 128 ঠিকানা করে**

* প্রতি গ্রাহকের জন্য 128 ঠিকানা → log₂(128) = 7 → **/25**
* মূল ব্লক /18 →
* Subnet prefix = 18 + log₂(16385 ÷ 128) = 18 + 7 = 25
* ১ম গ্রাহক: 190.100.64.0/25
* ২য় গ্রাহক: 190.100.64.128/25
* …
* ১২৮তম গ্রাহক: 190.100.127.128/25

**🟢 Group 3: 128 জন গ্রাহক, 64 ঠিকানা করে**

* প্রতি গ্রাহকের জন্য 64 ঠিকানা → log₂(64) = 6 → **/26**
* মূল ব্লক /19 →
* Subnet prefix = 19 + log₂(8192 ÷ 64) = 19 + 6 = 25
* ১ম গ্রাহক: 190.100.128.0/26
* ২য় গ্রাহক: 190.100.128.64/26
* …
* ১২৮তম গ্রাহক: 190.100.159.192/26

**✅ সারাংশ:**

| **Group** | **মোট ঠিকানা** | **Prefix (/n)** | **প্রতি গ্রাহককে দেওয়া সাবনেট** |
| --- | --- | --- | --- |
| 1 | 16,384 | /18 | /24 (256 ঠিকানা) |
| 2 | 16,384 | /18 | /25 (128 ঠিকানা) |
| 3 | 8,192 | /19 | /26 (64 ঠিকানা) |

**🔚 এখন চাও যদি আমি এর জন্য subnetting সহ একটা ছক বা চিত্র তৈরি করে দেই, সেটাও করতে পারি!**

তোমার প্রশ্নটা খুব ভালো — “**Group 1 → 190.100.0.0/18**, Group 2 → 190.100.64.0/18, Group 3 → 190.100.128.0/19\*\*” — এখানে 0, 64, 128 কীভাবে এল?

চলো এটা ধাপে ধাপে এবং **গাণিতিক ও বাস্তবভিত্তিক উদাহরণে** ব্যাখ্যা করি:

**🔍 প্রথমে বুঝি /18 এবং /19 কী বোঝায়**

* IP address: 32-bit
* /18 মানে: প্রথম 18 bit হচ্ছে **network part**, বাকি **14 bit** host part → মোট address সংখ্যা = 2¹⁴ = **16,384টি**
* /19 মানে: প্রথম 19 bit হচ্ছে network → 2¹³ = **8,192টি address**

**✅ এখন বরাদ্দ কীভাবে হচ্ছে?**

তোমাকে দেওয়া হয়েছিল একটি মূল ব্লক:

190.100.0.0/16 → এতে মোট 2¹⁶ = 65,536 টি address

এখন, Group 1 এবং 2 কে /18 সাইজের ব্লক দেওয়া হয়েছে। /18 তে আছে 16,384টি address, তাই এগুলোকে যখন ভাগ করা হবে, তখন **address গ্যাপে 16,384 করে লাফ** দিয়ে পরবর্তী ব্লক শুরু হবে।

**🧠 Group 1 → 190.100.0.0/18**

এখানে শুরু হচ্ছে 190.100.0.0 থেকে  
এই ব্লকে আছে **16,384 address**, তাই এর পরবর্তী ব্লক শুরু হবে:

190.100.0.0 + 16,384 address = 190.100.64.0

**কেন 64?**  
কারণ, প্রতি /18 ব্লকে আছে:

* 2¹⁴ = 16,384 টি address
* একেকটি /18 ব্লকের range হয়:

190.100.0.0 → 190.100.63.255

এখানে 0-63 মোট 64 ব্লক (class C size)  
তারপর পরবর্তী address → 190.100.64.0

**🧠 Group 2 → 190.100.64.0/18**

* এই ব্লকও /18, তাই এতে আবার 16,384 টি address থাকবে
* Range হবে:

190.100.64.0 → 190.100.127.255

তাহলে পরবর্তী address হবে:

190.100.128.0

**🧠 Group 3 → 190.100.128.0/19**

* এইবার Group 3 কে /19 দেওয়া হয়েছে (8192 address)
* শুরু হল 190.100.128.0 থেকে
* কারণ Group 2 এর শেষ address ছিল 190.100.127.255

**✅ এক কথায়:**

| **Group** | **Starting IP** | **ব্লক সাইজ** | **Address সংখ্যা** | **কারণ** |
| --- | --- | --- | --- | --- |
| Group 1 | 190.100.0.0 | /18 | 16,384 | শুরু থেকেই |
| Group 2 | 190.100.64.0 | /18 | 16,384 | আগের ব্লকের ঠিক পরেই |
| Group 3 | 190.100.128.0 | /19 | 8,192 | আগের ২ ব্লকের পর |

**🔢 Math Shortcut (Block Jump)**

তুমি চাইলে Address gap হিসাবেও এটা বুঝতে পারো:

একটি /18 → 2^(32-18) = 2^14 = 16384 address

একটি /19 → 2^(32-19) = 2^13 = 8192 address

CIDR block-এর শুরু address হতে হবে:

Block size দিয়ে divisible → মানে 16384 বা 8192 দিয়ে ভাগ যায় এমন address

**🔚 উপসংহার:**

তাই,

* 190.100.0.0/18 → প্রথম ব্লক (0 থেকে 63 পর্যন্ত)
* 190.100.64.0/18 → পরের ব্লক (64 থেকে 127 পর্যন্ত)
* 190.100.128.0/19 → তার পরের ব্লক (128 থেকে…)

🧠 **"0", "64", "128" এসেছে কারণ প্রতি গ্রুপের block size অনুযায়ী address range লাফ দিয়ে বেড়েছে** — যেটা subnetting এ খুবই গুরুত্বপূর্ণ বিষয়।

চাও চাইলে আমি এটা নিয়ে একটা table বা block diagram বানিয়ে দিতে পারি — আরও ভালোভাবে ধরতে পারবে।

অবশ্যই! আমি তোমাকে উপরের বিষয়গুলো সহজ ভাষায় ও পরিষ্কারভাবে বুঝিয়ে দিচ্ছি।

**IPv4 Special Addresses — সহজ ব্যাখ্যা**

IP ঠিকানাগুলোকে আমরা সাধারণত দুই প্রকারে ভাগ করি:

* **Classful addressing** (পুরনো পদ্ধতি)
* **Classless addressing** (আধুনিক পদ্ধতি, CIDR)

Classless addressing এ অনেক কিছু Classful addressing থেকে নেওয়া হয়েছে, বিশেষ করে কিছু বিশেষ ঠিকানা (special addresses)। নিচে তোমার দেওয়া বিষয়গুলো সহজ করে বুঝিয়ে দিলাম —

**১. All-Zeros Address (০.০.০.০ / 0.0.0.0)**

* ঠিকানা: **0.0.0.0** (একটা মাত্র ঠিকানা)
* ব্যবহার: যখন কোনো কম্পিউটার বা ডিভাইস তার নিজের IP ঠিকানা **ঠিকঠাক জানে না**, তখন সে এই ঠিকানা ব্যবহার করে।
* উদাহরণ: নতুন কোনো ডিভাইস যখন প্রথমবার ইন্টারনেটে যুক্ত হয়, তখন DHCP সার্ভার থেকে IP নেয়ার জন্য সে এই ঠিকানা ব্যবহার করে।
* Source IP হয় 0.0.0.0 এবং Destination হয় Limited Broadcast Address (যেমন 255.255.255.255)।

**২. All-Ones Address (Limited Broadcast Address) - 255.255.255.255**

* ঠিকানা: **255.255.255.255**
* ব্যবহার: যখন কোনো ডিভাইস একই নেটওয়ার্কের সব ডিভাইসকে মেসেজ পাঠাতে চায়, তখন এটি Destination ঠিকানা হিসেবে ব্যবহার হয়।
* কিন্তু এটা শুধু লোকাল নেটওয়ার্কের জন্য; রাউটার এগুলো বাইরের নেটওয়ার্কে পৌঁছে দিতে দেয় না।
* এর মাধ্যমে নেটওয়ার্কের সব ডিভাইস ঐ মেসেজ পায় এবং প্রসেস করে।

**৩. Loopback Address (127.0.0.0/8)**

* ঠিকানা ব্লক: **127.0.0.0 থেকে 127.255.255.255** (সাধারণত 127.0.0.1 সবচেয়ে পরিচিত)
* ব্যবহার: নিজের ডিভাইসের মধ্যে নিজের প্রোগ্রাম বা সিস্টেম টেস্ট করার জন্য।
* এই ঠিকানায় পাঠানো প্যাকেট ডিভাইস থেকে বের হয় না, বরং আবার ওই একই ডিভাইসে ফিরে আসে।
* যেমন, ping 127.0.0.1 দিয়ে তুমি নিজের কম্পিউটারের IP স্ট্যাক কাজ করছে কিনা পরীক্ষা করতে পারো।

**৪. Private Addresses (প্রাইভেট আইপি)**

* কিছু ঠিকানা ব্লক শুধুমাত্র **লোকাল নেটওয়ার্ক** এর মধ্যে ব্যবহারের জন্য রাখা হয়েছে, বাইরের ইন্টারনেট থেকে সেগুলো দেখা যায় না।
* এইগুলো মূলত LAN (Local Area Network) বা WiFi নেটওয়ার্কে ব্যবহার হয়।
* সবচেয়ে পরিচিত প্রাইভেট ব্লকগুলো:
  + 10.0.0.0/8 (মোট ১৬ মিলিয়নের বেশি ঠিকানা)
  + 172.16.0.0/12 (প্রায় ১ মিলিয়ন ঠিকানা)
  + 192.168.0.0/16 (৬৫ হাজারের বেশি ঠিকানা)
* এছাড়া 169.254.0.0/16 (APIPA) ব্লক আছে যা স্বয়ংক্রিয় ঠিকানা পেতে ব্যর্থ হলে স্বয়ংক্রিয় ঠিকানা হিসেবে দেয়া হয়।
* প্রাইভেট ঠিকানাগুলোকে সাধারণত **NAT (Network Address Translation)** এর মাধ্যমে ইন্টারনেট অ্যাক্সেস দেয়া হয়।

**৫. Multicast Addresses (224.0.0.0/4)**

* ঠিকানা ব্লক: **224.0.0.0 থেকে 239.255.255.255**
* ব্যবহার: একাধিক ডিভাইসকে একই সাথে ডেটা পাঠানোর জন্য (যেমন ভিডিও কনফারেন্স, লাইভ স্ট্রিমিং)।
* এই ঠিকানাগুলো multicast communication এর জন্য সংরক্ষিত।

**৬. Network Address (নেটওয়ার্ক ঠিকানা)**

* কোনো IP ব্লকের **প্রথম ঠিকানা** (যেখানে হোস্ট অংশের সব বিট 0) হয় নেটওয়ার্ক ঠিকানা।
* এটি কোনো ডিভাইসের ঠিকানা নয়, বরং পুরো নেটওয়ার্ক বা সাবনেটকে নির্দেশ করে।
* উদাহরণ: 192.168.1.0/24 - এখানে 192.168.1.0 হলো নেটওয়ার্ক ঠিকানা।

**৭. Direct Broadcast Address (ডাইরেক্ট ব্রডকাস্ট ঠিকানা)**

* কোনো IP ব্লকের **শেষ ঠিকানা** (হোস্ট অংশের সব বিট 1) কে direct broadcast address বলা হয়।
* এটা ব্যবহার করে একটি রাউটার ঐ নেটওয়ার্কের সব হোস্টকে একসাথে প্যাকেট পাঠাতে পারে।
* এই ঠিকানাটিও শুধু Destination ঠিকানায় ব্যবহার হয়, হোস্ট ঠিকানায় নয়।
* উদাহরণ: 192.168.1.255/24 - এই ঠিকানায় পাঠানো প্যাকেট নেটওয়ার্কের সব ডিভাইসে পৌঁছে যাবে।

**সহজ উদাহরণ দিয়ে সারাংশ:**

| **ঠিকানা** | **ব্যবহার** | **লক্ষ্য** |
| --- | --- | --- |
| 0.0.0.0 | নিজ ঠিকানা জানা না থাকলে | উৎস (source) IP |
| 255.255.255.255 | লোকাল নেটওয়ার্কে ব্রডকাস্ট মেসেজ | গন্তব্য (destination) IP |
| 127.0.0.1 | নিজের ডিভাইসে টেস্টের জন্য | গন্তব্য (destination) IP |
| 10.x.x.x, 172.16.x.x, 192.168.x.x | প্রাইভেট নেটওয়ার্কের জন্য | - |
| 224.x.x.x - 239.x.x.x | মাল্টিকাস্ট গ্রুপ কমিউনিকেশন | - |
| নেটওয়ার্ক ঠিকানা | প্রথম ঠিকানা, নেটওয়ার্ক নির্দেশ করে | - |
| ডাইরেক্ট ব্রডকাস্ট | শেষ ঠিকানা, নেটওয়ার্কের সব হোস্টকে প্যাকেট পাঠায় | - |

**তোমার প্রশ্ন থাকলে বলো, আরও সহজ করে বা উদাহরণ দিয়ে বুঝিয়ে দিবো!**

নিশ্চিতভাবে! NAT (Network Address Translation) সম্পর্কিত তোমার দেওয়া টেক্সটটি সহজভাবে ও সংক্ষেপে বুঝিয়ে দিচ্ছি —

## NAT (Network Address Translation) কি? কেন দরকার?

### সমস্যা:

একজন ISP (ইন্টারনেট সার্ভিস প্রোভাইডার) যখন ছোট একটি নেটওয়ার্ক বা হাউসহোল্ডকে ঠিকানার ছোট একটি রেঞ্জ দেয়, তখন পরবর্তীতে যদি ওই ব্যবসা বা পরিবার বড় হয়, বড় রেঞ্জ পাওয়া কঠিন হয়ে যায়। কারণ পাশের ঠিকানাগুলো অন্য নেটওয়ার্কে দেওয়া হয়ে থাকতে পারে।

### বাস্তব পরিস্থিতি:

* অনেক সময় নেটওয়ার্কে থাকা সব কম্পিউটার একসঙ্গে ইন্টারনেট ব্যবহার করে না।
* যেমন, ২০টি কম্পিউটার থাকলেও একসঙ্গে মাত্র ৫টি ইন্টারনেটে থাকতে পারে।
* তাই সব কম্পিউটারের জন্য পাবলিক IP দেওয়ার প্রয়োজন হয় না।

## NAT কীভাবে কাজ করে?

### মূল ধারণা:

* নেটওয়ার্কে থাকা কম্পিউটারগুলো **প্রাইভেট IP** ঠিকানা ব্যবহার করবে (যেমন 172.18.3.x)।
* বাইরের ইন্টারনেটের সাথে যোগাযোগ করার জন্য শুধু একটা বা কয়েকটা **পাবলিক IP** ঠিকানা ব্যবহার করা হবে (যেমন 200.24.5.8)।
* একটি NAT সক্ষম রাউটার দুই রকম ঠিকানা ম্যাপিং করে:
  + **প্রাইভেট ঠিকানা ↔ পাবলিক ঠিকানা**
* যখন একটি কম্পিউটার ইন্টারনেটে কিছু রিকোয়েস্ট পাঠায়, NAT রাউটার প্রাইভেট ঠিকানাটি পাবলিক ঠিকানায় পরিবর্তন করে প্যাকেট পাঠায়।
* রেসপন্স প্যাকেট আসলে পাবলিক ঠিকানায় আসে, NAT রাউটার তা আবার সঠিক প্রাইভেট ঠিকানায় রূপান্তর করে ভিতরের কম্পিউটারে পৌঁছে দেয়।

### Address Translation (ঠিকানা অনুবাদ)

* **Outgoing (বাইরের দিকে যাওয়া প্যাকেট):**
  + উৎস ঠিকানা (Source IP) প্রাইভেট থেকে পাবলিকে বদলে যায়।
* **Incoming (অন্তর্ভুক্ত প্যাকেট):**
  + গন্তব্য ঠিকানা (Destination IP) পাবলিক থেকে প্রাইভেটে বদলে যায়।

### NAT রাউটারের কাজ - Translation Table

NAT রাউটার একটি টেবিল রাখে যাতে থাকে —

* প্রাইভেট IP
* পাবলিক IP
* (কখনো কখনো) Source Port, Destination IP, Destination Port, Protocol ইত্যাদি

এর মাধ্যমে রাউটার বুঝতে পারে কোন ইন্টারনেট থেকে আসা প্যাকেট কোন প্রাইভেট হোস্টকে পাঠাতে হবে।

### একাধিক ডিভাইস একসাথে ইন্টারনেট ব্যবহার করতে চাইলে?

* যদি শুধু একটা পাবলিক IP থাকে, তাহলে একসময় শুধু একটাই কম্পিউটার ইন্টারনেট ব্যবহার করতে পারে।
* সমস্যার সমাধান হল, **পাবলিক IP এর একটা পুল (pool)** ব্যবহার করা (উদাহরণ: 200.24.5.8 থেকে 200.24.5.11 পর্যন্ত চারটি IP)।
* এতে একসঙ্গে চারটি ডিভাইস একই বাইরের সার্ভারের সাথে কথা বলতে পারে।
* তবে এতে কিছু সীমাবদ্ধতা থাকে, যেমন একই সার্ভারে একই সময় একাধিক সার্ভিসে প্রবেশ করা কঠিন।

### Port-Based NAT (PAT) — অনেক কম্পিউটার, একটি পাবলিক IP ব্যবহার

* NAT রাউটার **Port Number** ব্যবহার করে ভেতরের অনেক কম্পিউটারকে বাইরের সাথে একসাথে কানেক্ট করতে দেয়।
* উদাহরণ:
  + 172.18.3.1:1400 (কম্পিউটার ও পোর্ট) ↔ 200.24.5.8:50000 (পাবলিক IP ও পোর্ট)
  + 172.18.3.2:1401 ↔ 200.24.5.8:50001
* এই পদ্ধতিতে **Source IP ও Source Port** দুইটি ব্যবহার করে অনন্য সেশন (connection) তৈরি হয়।
* NAT টেবিলে এই তথ্য রেখে রাউটার ঠিক বুঝতে পারে কোন রেসপন্স কোন কম্পিউটারে যাবে।

### NAT এর সীমাবদ্ধতা ও ব্যবহার

* NAT ব্যবহার করলে প্রাইভেট নেটওয়ার্ক থেকে ইন্টারনেট-ইনিশিয়েটেড (Initiated) কমিউনিকেশন হয় ঠিকঠাক।
* কিন্তু বাইরে থেকে প্রাইভেট নেটওয়ার্কে সরাসরি সার্ভার রান করানো বা ডিরেক্ট এক্সেস দেওয়া কঠিন।
* অধিকাংশ হোম বা ছোট ব্যবসার নেটওয়ার্কে NAT খুবই দরকারি কারণ তারা একাধিক ডিভাইসকে একটা বা কিছু পাবলিক আইপি দিয়ে ইন্টারনেট ব্যবহার করতে দেয়।

## সংক্ষেপে:

| **বিষয়** | **ব্যাখ্যা** |
| --- | --- |
| সমস্যা | পাবলিক IP ঠিকানা সীমিত, বড় নেটওয়ার্ককে দেয়া কঠিন |
| সমাধান | NAT দিয়ে প্রাইভেট IP ও পাবলিক IP ম্যাপ করা হয় |
| NAT রাউটার কাজ | প্রাইভেট ঠিকানাকে পাবলিক ঠিকানায় ও ফিরতি ঠিকানায় অনুবাদ করে |
| Translation Table | NAT রাউটার কোন প্রাইভেট IP পাবলিক IP এর সাথে যুক্ত তা রাখে |
| একাধিক ডিভাইস | Port Number ব্যবহার করে একাধিক কম্পিউটার একই পাবলিক IP ব্যবহার করতে পারে |
| সীমাবদ্ধতা | বাইরের সার্ভার থেকে প্রাইভেট নেটওয়ার্কে সরাসরি প্রবেশ কঠিন |

**তুমি চাইলে আরও ছোট্ট উদাহরণ বা ডায়াগ্রামসহ বুঝাতে পারি।** তোমার জন্য কেমন হলো?

অবশ্যই, বাংলায় সহজ করে বলছি—

**NAT এর আসল কাজ হলো:**

একটা ছোট বা বড় প্রাইভেট নেটওয়ার্কে যতগুলো কম্পিউটার বা ডিভাইস আছে, সবাইকে আলাদা আলাদা পাবলিক (ইন্টারনেট) আইপি ঠিকানা না দিয়ে, একটা বা কিছু পাবলিক আইপি ঠিকানা ব্যবহার করে ইন্টারনেটে যোগাযোগ করার সুযোগ দেওয়া।

অর্থাৎ, NAT একটা রাউটার বা ডিভাইসের মাধ্যমে:

* **প্রাইভেট (লোকাল) আইপি ঠিকানাগুলোকে পাবলিক আইপি ঠিকানায় রূপান্তর করে।**
* যাতে অনেকগুলো ডিভাইস একই সাথে ইন্টারনেট ব্যবহার করতে পারে, শুধু কয়েকটি পাবলিক আইপি ঠিকানা দিয়ে।
* এতে IP ঠিকানার অভাব (shortage) থেকে রক্ষা পাওয়া যায়।
* বাইরের ইন্টারনেট থেকে প্রাইভেট নেটওয়ার্কের ঠিকানা লুকানো থাকে, নিরাপত্তাও বাড়ে।

সারাংশ:  
**NAT এমন একটা পদ্ধতি যা প্রাইভেট নেটওয়ার্কের ডিভাইসগুলোকে পাবলিক ইন্টারনেট ঠিকানা ছাড়াই ইন্টারনেটে যুক্ত হতে সাহায্য করে।**

আরো সহজে:

NAT হচ্ছে এমন একটা মাধ্যম যা একাধিক কম্পিউটারের আইপি ঠিকানাকে একটা বা কিছু আইপি ঠিকানায় পরিবর্তন করে ইন্টারনেট ব্যবহারের সুযোগ দেয়।

বুঝতে পারছো?