**Chapter 05: IPv4 Addressing**

Table of Contents

[5.1 Introduction 3](#_Toc86833662)

[Hierarchical Addressing 3](#_Toc86833663)

[5.2 Classful Addressing 5](#_Toc86833664)

[Identifying Classes 6](#_Toc86833665)

[Subnetting 7](#_Toc86833666)

[Supernetting 14](#_Toc86833667)

[5.3 Classless Addressing 15](#_Toc86833668)

[Two-Level Addressing 15](#_Toc86833669)

[Slash Notation 15](#_Toc86833670)

[5.4 Special Addresses 16](#_Toc86833671)

[Class D 16](#_Toc86833672)

[Class E 16](#_Toc86833673)

[All-Zeroes Address 16](#_Toc86833674)

[All-Ones Address 17](#_Toc86833675)

[Loopback Address 17](#_Toc86833676)

[Private Addresses 17](#_Toc86833677)

[Multicast Addresses 17](#_Toc86833678)

[Network Address 18](#_Toc86833679)

[Direct Broadcast Address 18](#_Toc86833680)

[5.5 NAT 19](#_Toc86833681)

[Address Translation 19](#_Toc86833682)

[Translation Tables 20](#_Toc86833683)

## 5.1 Introduction

Every device connected to the internet has a unique address, called the **IP Address**. It is **4 bytes** or 32 bits long in IPv4 (128 bits in IPv6) and can be private or public. A **Private** IP Address can only be used within a local network, such as within an office or a building. A **Public** IP Address, also called a Routable or a Global IP Address, can be used to connect to the internet and is provided by the ISP.

The above is some basic information about IP addresses, but there are a few more things that are not common knowledge:

* IP Addresses are **hierarchical**, as opposed to flat
* They can be of two types, **Classful Addresses** or **Classless Addresses**
* It has **Classless Inter-Domain Routing** (CIDR)
* It has **Subnetting** and **Subnet Masking**.

### Hierarchical Addressing

We already have **MAC Addresses**, which are flat addresses. The MAC Address can be used to uniquely identify every device on the planet. This brings up the question of why we then need IP Addresses as well.

The problem with MAC Addresses is that it is not hierarchical. Specific parts of the address do not give us any information. This means that, although the MAC Address as a whole can be used to uniquely identify a device, if we wanted to find a device, we would literally end up searching through all the devices on the planet.

Since an IP address is hierarchical, it allows us to break the process of finding a specific device into parts.

The above is an example of an **IPv4 Address**. This notation is called a **dotted decimal** notation. Each of the numbers in the four sections are the decimal equivalents of 8-bit binary numbers. There are said to be four **octets**.

For a particular IP address, some of the octets identify the network, called the **network ID**, while the rest identify the host within the network, called the **host ID**. For example, the first three octets could be the network ID, while the last octet could be the host ID. This allows routers in between to just store network IDs and let the network sort out the host ID, thus dividing the work.

## 5.2 Classful Addressing

The number of octets used to identify the network allows us to divide IP addresses into **classes**. If the first octet is for the network while the other three are for the host, it is a **Class A** IP Address. If the first two octets are for the network while the last two are for the host, it is a **Class B** IP Address. If the first three octets are for the network while just the last octet is for the host, it is a **Class C** IP Address. This is what we mean by **Classful Addressing**.

The address is a **Class C** IP Address. We will look into identifying the classes soon, but for now, accept this. Since it is a class C address, the **first three octets** are used to identify the **network**. There are bits available in the first three octets, meaning a total of  **networks** can be identified. However, only the **last octet** is used to identify the **host**. Thus, there are only bits available to identify a host. This means, even though there can be different networks, each network can only have  **hosts**. This results in a huge number of **small networks**. Class C IP Addressing is what we commonly use (that’s why the address shown above looks familiar).

In **Class A**, on the other hand, there can only be  **networks**, but each network can have  **hosts**. This means there are very few networks available, but all of them can have a huge number of hosts.

### Identifying Classes

**Class A** addresses always **start with a**  (in binary) in the first octet. This means the first octet has bits left and can have networks. Thus, if the first octet has a number between and , it is a Class A address. However, of these are reserved, so in reality there can only be networks. These addresses are private and are not available to the general public. They cannot be accessed via the Internet.

**Class B** addresses **start with a**  in the first octet. This leaves bits in the first octet, so addresses between () and () are Class B addresses.

**Class C** addresses **start with a**  in the first octet. This leaves bits left in the first octet, so addresses between () and () are Class C addresses.

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Bits |  |  |  |  |  |  |  |  |  |  |  |
| Class A |  |  |  |  |  |  |  |  |  |  |  |
| N | | | | | | | | H | H | H |
|  | | | | | | | |  |  |  |
| Class B |  |  |  |  |  |  |  |  |  |  |  |
| N | | | | | | | | N | H | H |
|  | | | | | | | |  |  |  |
| Class C |  |  |  |  |  |  |  |  |  |  |  |
| N | | | | | | | | N | N | H |
|  | | | | | | | |  |  |  |

### Subnetting

#### Wasted Space

A huge issue with the classful addressing is the **wastage of addresses**. For example, if we want to buy a Class C network address, we will be forced to buy the complete network address, i.e. all the host addresses between and . Even if we needed just a few addresses, we would be forced to buy the complete ‘set’, also called a **block**. This wastage of addresses means we would quickly run out of addresses.

#### Reserved Addresses

Additionally, in every block, we need to reserve 2 addresses, one as a **broadcast address** and one as a **network address**. We already know what a broadcast address is. A network address is the address that is stored by routers to route packets to this network. For a Class C network, the network address would be the first host address in the block, i.e. as the host address, and the broadcast address would be the last host address in the block, i.e. as the host address.

The reservation of these two addresses means a Class A network has host addresses, a Class B network has host addresses and a Class C network has host addresses.

#### Creating Subnets

In order to deal with the wastage of space, networks can be divided into smaller portions. For a class C network for example, the host addresses that a block gives us can be divided into parts. Each of these parts is called a **subnet**.

Depending on the number of subnets we want to create, we can use a few bits from the **host address** to identify the subnet. For example, if we need to make 2 subnets, we can use 1 bit. If we need 4 subnets, we can use 2 bit and so on. The bits are taken from the left-hand side of the host address.

If we reserve 2 bits for subnet identification, then each subnet will have host addresses. However, just like we did previously, we need to **reserve two addresses** in each subnet for the network address and the broadcast address. Thus, each subnet would have host addresses. This means the total number of usable host addresses, the **block size**, has decreased from to , since we are having to reserve more addresses for network addresses and broadcast addresses.

#### Unusable Subnets

There is however, another issue. If we use two bits to create four subnets, then the subnet addresses would be , , and . However, we cannot use the first and the last subnet addresses, since they would be part of the network and broadcast addresses of the whole network.

Consider the address . This is the network address for the entire network, as identified by the host address, in binary. However, if we use as a subnet address, we would be unable to tell if identifies the network address of the entire network, or the network address of the subnet.

This single issue forces us to abandon the entire subnet, even though the other addresses in the subnet are perfectly usable. A similar scenario occurs for the subnet . This problem essentially halves the number of host addresses we have available to us. It also means that we cannot have just 2 subnets, borrowing a single bit from the host address section. The minimum number of bits we have to borrow is .

On the other hand, the maximum number of bits we can borrow from the host address is . If we borrow bits, this would mean every subnet would only have host addresses. Since we need to reserve host addresses in each subnet, this would leave no addresses left for use. Due to these limitations, it is preferable to use Class A networks for subnetting, since that gives us a larger range of possible subnets.

#### Example

Say we need to create 8 subnets. If we borrow bits from the host address, we can create subnets. However, as discussed before, we cannot use the first and last subnets. This forces us to borrow the bits from the host address, which would allow us to create subnets at most, but we will only be using the first subnets.

The first subnet will have a network address of , in decimal. The second subnet will have a network address of , in decimal. The third will have a network address of , in decimal. Notice how there is a pattern here. The block size is , so the network addresses differ by .

This pattern makes things very easy for us. If we are given an IP address where the host address is for example, we can easily identify that this belongs to the second subnet.

#### Subnet Masks

It is possible to hide the part of an IP address that holds the network identifier. This allows devices on the same network to communicate with each other without bothering to check the network identification, since it is of no use to them.

The first mask is the **default mask**. A Class A IP Address would have a default mask of , a Class B IP Address would have a default mask of and a Class C IP Address would have a default mask of . Essentially, we took the portion of the address that identifies the network and set all the bits to .

The second mask is the **subnet mask**. If we have subnets, then there are extra bits after the normal network identifier that is used to identify the subnet. For example, we saw above that we borrowed the first bits from the host address to identify the subnets. If we set these four bits to as well, for a Class C network’s subnet, the subnet mask would be , or . This is common for all the subnets.

However, notice that if we are given a subnet mask , we cannot say for certain by looking at just this that the subnet is for a Class C network. It could also be the case that it belongs to Class A or B and that the bits it has borrowed from the host identifier has caused it to appear this way. Similarly, we cannot take a default mask and immediately associate it with a particular class. It could be a default mask or a subnet mask depending on which class we associate it with.

The subnet mask can also be written as , since the first bits are set to . This is called the **CIDR Notation**. Similarly, would be and so one.

Examples

1. It is given that a network address is, and it has the subnet mask .

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Subnet Address |  |  |  |  | … |
| First Host Address |  |  |  |  | … |
| Last Host Address |  |  |  |  | … |
| Broadcast Address |  |  |  |  | … |

1. Given a host IP, and the subnet mask , find
   1. The subnet address.

The subnet address is the same as the host IP, except the last octet has a decimal value which is the largest multiple of smaller than , since bits are being used to identify subnets and . Thus, the subnet mask is .

* 1. The broadcast address for the subnet.

The broadcast address is .

1. A router has received a packet with the destination IP address and the subnet mask . What does the router do?

The given IP address is the broadcast address for the subnet . A broadcast can only take place within a network. This means that packet originated within the subnet, and due to how LANs work, every node within the subnet has already received that packet. As such, the router will simply drop the packet.

1. It is given that a network address is and it has a subnet mask .

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Subnet Address |  |  |  |  |
| First Host Address |  |  |  |  |
| Last Host Address |  |  |  |  |
| Broadcast Address |  |  |  |  |

1. It is given that a network address is and it has the subnet mask .

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Subnet Address |  |  | … |  |  |
| First Host Address |  |  | … |  |  |
| Last Host Address |  |  | … |  |  |
| Broadcast Address |  |  | … |  |  |

1. It is given that a network address is and it has a subnet mask .

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Subnet Address |  |  |  |  | … |
| First Host Address |  |  |  |  | … |
| Last Host Address |  |  |  |  | … |
| Broadcast Address |  |  |  |  | … |

1. It is given that a network address is and it has a subnet mask .

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Subnet Address |  |  |  |  | … |
| First Host Address |  |  |  |  | … |
| Last Host Address |  |  |  |  | … |
| Broadcast Address |  |  |  |  | … |

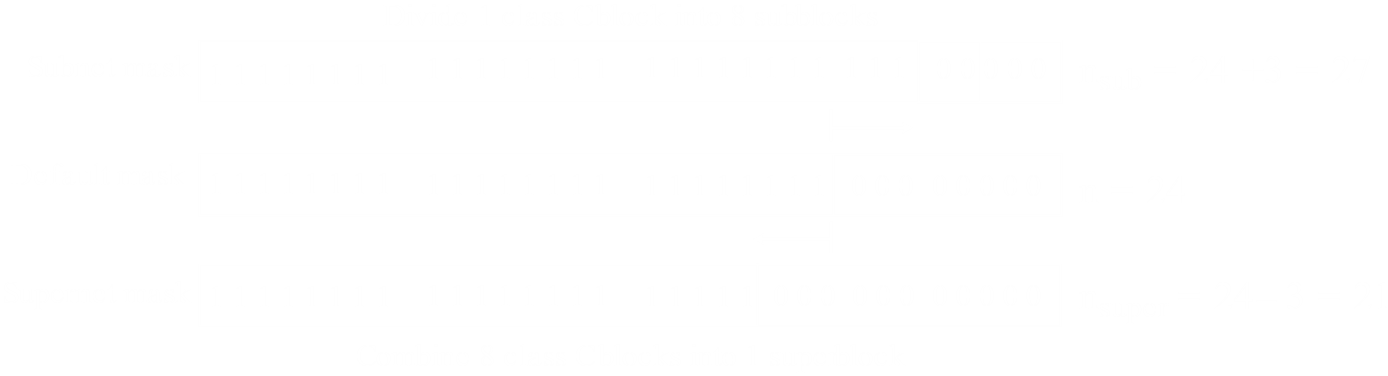
Notice that we get way **more subnets** for the same subnet mask in a **Class B** address than we do in a Class C address. For example, apply the subnet mask in a Class C address gives us subnets, while in Class B it gives us subnets. The number of hosts per subnet for the same subnet mask in different classes remains the same, for a subnet mask of .

### Supernetting

For some organizations, a Class C block is not large enough. However, they cannot get access to the higher classes either, since they have all been bought out. To resolve this issue, we use **supernetting**, where several Class C blocks are combined to form a **supernetwork**.

#### Supernet Masks

A **Supernet Mask** is the opposite of a subnet mask. Instead of taking bits from the host portion, we give bits to the host portion from the network portion.



However, since we can only decrease one bit at a time, the number of Class C blocks being combined needs to be a **power of** . Thus, we might be wasting some addresses. Additionally, it also complicates routing a lot.

## 5.3 Classless Addressing

As mentioned before, things are starting to get complicated. We clearly need more addresses, so a long-term solution, **IPv6 addressing** has been devised, to which we are slowly switching. However, this requires switching the formats for all IP packets, which might cause problems. So, in the short term, **classless addressing** has been created.

In classless addressing, **variable-length blocks** are used that belong to **no classes**. The blocks can be of 1 address, 2 addresses, 4 addresses and so on. The only restriction is that the blocks need to have a size of a **power of** .

### Two-Level Addressing

In Classful addressing, we had **network IDs** and **host IDs**. Similarly, we have **prefixes** and **suffixes** in classless addressing that respectively play the same roles. The difference is, network IDs in classful addressing depended on the **class**. They could either be 8 bits, or 16 bits or 24 bits. In classless addressing, the prefix length depends on the block size, so they can be anywhere from 1 to 32 bits.

### Slash Notation

Unlike classful addressing, we cannot find the **prefix length** just by looking at the address. To do this, we use **slash notation**, formally called the classless interdomain routing (CIDR) notation. For example, for the address , the first bits are for the prefix. Thus, the network mask is .

Once we have the CIDR notation, we can find the **network address**, the **range of host addresses** and the **broadcast address**.

## 5.4 Special Addresses

### Class D

**Class D** addresses have a value between and for the first octet. There is just one block of Class D addresses, with the default mask .

Class D was designed for **multicasting**. Each address is used to define one **group of hosts** on the Internet. When a group is assigned to an address in this class, every host in that group will have a **multicast address** alongside their normal, unicast address.

### Class E

There is just one block of **Class E** addresses. Class E addresses have a value between and for the first octet and the default mask . These addresses are **reserved**.

### All-Zeroes Address

The previous two special addresses were the only ones from the **classful addressing** scheme. The rest are all for the **classless addressing** scheme, starting with this one.

The block is called an **All-Zeroes Address**. This block has a single address. It is used by the host when it needs to send an IPv4 packet but does not know its own address. This could happen at boot time when the host needs to contact a bootstrap server. The host uses this address as the source and a limited broadcast address as the destination to find its own address.

### All-Ones Address

The block is called an **All-Ones Address** or a **Limited Broadcast Address**. A host that wants to send a message to every other host in the network can use this address. However, routers block this packet, which means the broadcast is limited to the local network.

### Loopback Address

The block is called a **Loopback Address**. A packet with this destination never leaves the machine. It is used to test the protocol software or when a client and server are on the same machine.

### Private Addresses

There are several blocks that are assigned for **private** use. They are not recognized globally. These addresses are used either in isolation or with NAT techniques.

### Multicast Addresses

Just like **Class D** addresses in classful addressing, is used for **multicast communication** in classless addressing.

### Network Address

A **network address** is the first address of a block, with all the bits of the **suffix** set to . This address defines the network itself and not any host within the network. For a subnetwork, this would be the **subnetwork address**.

### Direct Broadcast Address

A **direct broadcast address** is the last address of a block, with all the bits of the **suffix** set to . This address is used to send a packet to all the hosts in the network. All the hosts will accept this packet.

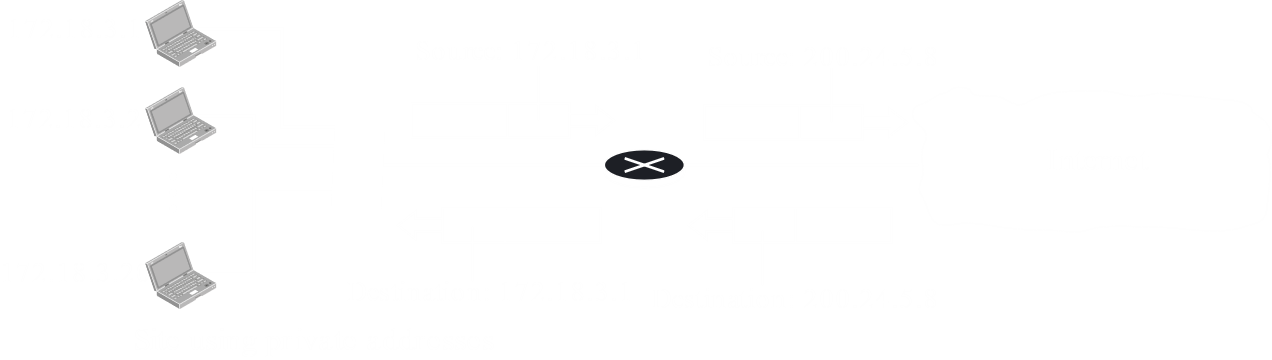
## 5.5 NAT

A huge problem with IPv4 addressing is that we simply do not have enough addresses for everyone. One possible solution to this is to use **NAT routers**.

Essentially, we have an organization, which is given a **small number of IP addresses**. Internally, that organization could have many more addresses that work within their **private network**. However, whenever a device from the internal network wants to connect to the Internet, they must go through the NAT router, which converts the private addresses to a **global address**. The rest of the internet will only see the global address.

NAT actually makes IPv4 perfectly usable, even today. The only reason we are switching to IPv6 is because it provides security features. Data packets can be **encrypted**.

### Address Translation



Consider that the organization is given the IP address and a device on the private network has an address of . The NAT router **changes the IP address** from to and connects to the Internet under that IP. For incoming addresses, the NAT router does the reverse, changing the destination from the global address to the required private address.

### Translation Tables

Changing the source for outgoing packets is easy enough, but changing the **destination** for **incoming packets** presents a problem. How does the NAT router know which private IP it needs to send an incoming packet to? To solve this problem, **translation tables** are used.

#### Using One IP Address

The simplest form of using a translation table is to just store the **private address** of the host **sending** a packet and the **destination** it is sending to. Any response that comes back from that destination are then simply sent to that host.

Here, we are assuming that there is a **single global address** which is being used by each host at a time. This is why we do not have issues like multiple hosts connecting to the same destination.

If we use this, then all communication must **start with the host**.

#### Using a Pool of IP Addresses

Instead of using just one global address, we could have a **pool of global addresses**. If we have addresses, hosts can connect to the Internet. However, no host can connect to multiple external servers simultaneously and multiple hosts cannot access the same external server simultaneously.

#### Using both IP Addresses and Port Addresses

To allow a **many-to-many** relationships between private network hosts and external servers, we need to extend the translation table.

Say two hosts inside the network have the addresses and respectively and both want to connect to the external host . If we use the **port addresses** of the source and destination along with the IP addresses, we can remove any ambiguity about where each packet needs to go, since each port address is **unique**.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Private Address | Private Port | External Address | External Port | Transport Protocol |
|  |  |  |  | TCP |
|  |  |  |  | TCP |
|  |  |  |  |  |

When a response comes back from the host, we look at both the **external address** and the **private port** to determine what the private address of the host is.