Networks

By Sheldon Chong

View: 🌐 Networks for the best reading experience

My github: https://github.com/Sheldon-Chong

▼ **If you're a student at 42, read this first!:**

It will cover all necessary topics and theories and calculation methods to solve netpractice, but also contains several other topics necessary for understanding the full picture for networking.

The distinction is below

> ❓ **Topics covered**
>
> **Outsid netpractice:**
>
> 1. ARP requests and MAC addresses
> 2. DNS
> 3. DHCP
> 4. supports private and public IPs
>
> **Part of netpractice:**
>
> 1. Overlapping IP's
> 2. CIDR
> 3. Routing table

> 4. Switches
> 5. Routers
> 6. Interfaces
> 7. reserved IP addresses
> 8. subnet masks
> 9. IP addresses

> This guide aims to give a comprehensive understanding of networking, and paint the full picture of how various networking systems link together to facilitate communication.

▼ ❓ **What makes this guide unique**

✅ Concepts are taught from scratch

✅ Gradually builds your understanding using an ongoing narrative.

✅ Hand-on segments and step-by step instructions to see networking happen on your own device (currently these examples only run on windows)

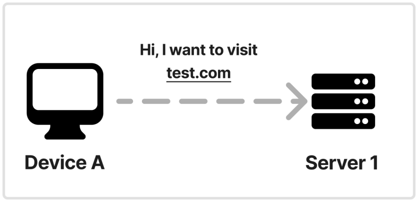
✅ Packed with many illustrations and graphics to simplify and visualize networking

▼ 📖 **Contents:**

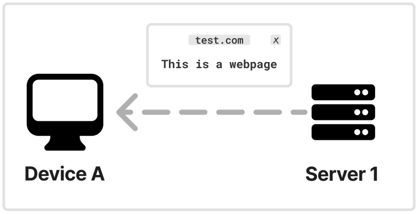
d. Routing table

Chapter 1: Networks and communication

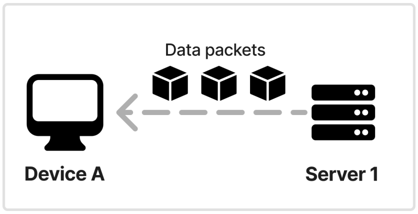
Let’s set the stage. Let’s say there Device A wants to access a webpage that’s stored in Server 1. To do so, it will need to send a request (specifically, a HTTP request) to the server to fetch the required data from the server



The server responds with the HTML, CSS, JavaScript, images, and other resources needed to render the page.



Webpages can be pretty small, or very large. Nevertheless, any data sent to another device first needs to be broken down into data packets, small units of data that have information about how to travel to its end destination.



This process of breaking-down data into packets is typically done in a process called TCP

▼ 💡 **TCP (Transmission Control Protocol)**

> A protocol that breaks data into data packets, and performs ordered, and error-checked delivery of data over a network

1. TCP establishes a connection between the sender and receiver before data transmission (called a "three-way handshake")
2. Ensures accurate data delivery by:
    - Error detection, re-transmitting lost or corrupted packets, and ensuring the receiver acknowledges each successfully received packet
    - Preventing a sender from overwhelming a receiver by controlling the rate of data transmission based on the receiver’s capacity, and adjusting data transition rate if congestion is detected.

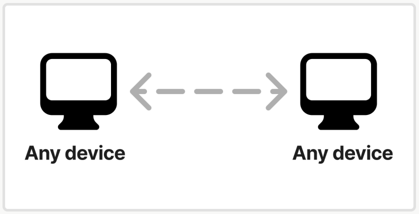
- Commonly used in situations where data needs to be loaded correctly, like in **web-browsing** (HTTP/HTTPS), **sending files** (FTP) **or emails** (SMTP, IMAP, POP3), or connecting remotely (SSH)
- Less commonly used for **streaming**, **online gaming** of **voice calls**, where speed and low-latency are prioritized over guaranteed delivery, as packet loss significantly affect the overall experience

For device A and Server 1 to be able to connect and communicate with each other, they have to be on the same network.

▼ 💡 **networks:**

> Refers to group of two or more devices *(e.g. computers, smartphones, servers etc)* that are connected together to share resources and communicate with each other.

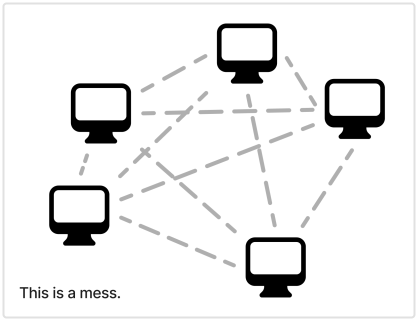
- Devices on the same network share the same system of rules, known as **communications protocols**, to transmit information physically or wirelessly.
- Each device on the network is known as a **host**



- In this example, two devices connect directly to each other to interact and share resources. This is known as a peer-to-peer (P2P) configuration.

The example we've just discussed is a simple network, consisting of only two hosts. But what if we were in a room with multiple hosts, all of which wish to be able to send information to each other?

A p2p approach scales poorly, as you would need to create a mesh-like network where each device connects to every other device with a direct link, so that they are able to send resources to each other. In reality, this becomes impractical and complex as the number of devices increases.

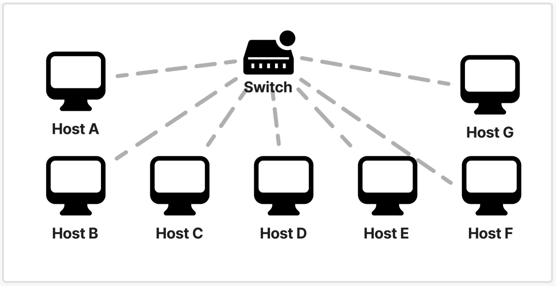


Several devices exists to handle this issue. One such device is the switch

▼ 💡 **switch**

> A switch is a physical device that connects networked devices that are part of the same network. Each device is connected to the switch via an Ethernet cable or wirelessly.

- With such an approach, all devices will only need to send resources to the switch, rather than coordinating data transfer between each device.



All devices you see here are part of the same network

- A switch receives data packets from a sender, and forwards data to the destination device based on the information that is given on the data packet

With many device present In a network, each device needs to have some way of identifying themselves. “Host A” and “Host B” are not good names to help data packets know where to travel to. This is where IP addresses come in:

▼ 💡 **IP address**

> A unique numerical label assigned to each device connected to a network.

- The two major types of IP addresses that exist today are
  - IPv4 (Internet Protocol version 4)
  - IPv6 (Internet Protocol version 6)
  - Onwards from here, IPv4 addresses will be the main focus to simplify concepts
- IP’s are assigned to devices automatically by organizations such as your Internet Service Provider, through devices such as the DHCP, or manually by individuals like network administrators.

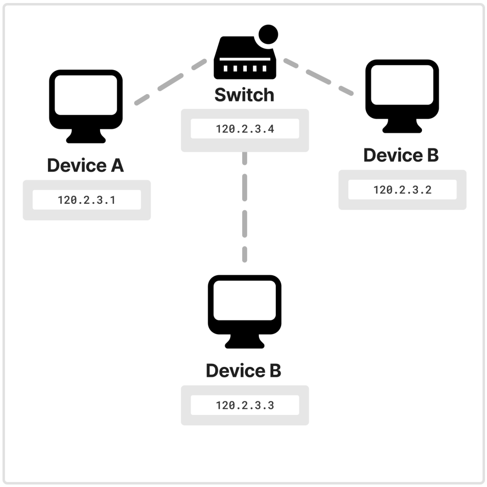
An IPv4 address’s structure consist of 4 octets (4 bytes), delimited by dots, like so:

```
192.168.1.1
```

Each octet stores a number between 0-255, and four of such octets in an IP allow for 4.3 billion unique address combinations (However part of these addresses are reserved for specific purposes, reducing the number available for public use).

❓ Despite the large pool of addresses that can be formed from IPv4 (4.3 billion combinations), it has faced exhaustion, prompting the need for IPv6, which offers a greater number of addresses.

Every device on the same network should have a unique IP address. This is crucial as data packets will be unable to distinguish between two devices of the same IP, even if they are different.



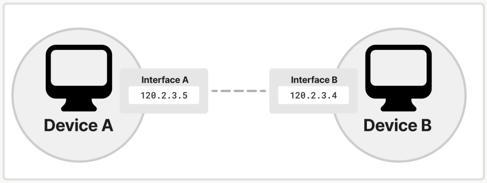
Where's the subnet mask :0???
Don't worry, we'll get to that ;)

We've learned that data packets are sent from one host, and received by another. However, that isn't the full picture. Packets need points to travel from, and to arrive to. These points are known as interfaces.

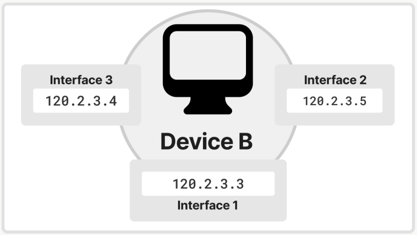
▼ 💡 **interface**

> a point of connection where a device interacts with a network, which allows devices to send and receive data.

- Interfaces serve serves as the entry and exit point for data packets on a device.



  - When a device sends data, the data packet leaves through an interface.
  - When a device receives data, the packet arrives at one of its interfaces.
- A device may have multiple interfaces



Device B has 3 interfaces as seen here

- A network interface can be physical

(e.g., a Network Interface Card, NIC)
or virtual
(e.g., a loopback interface or a virtual network adapter)

💻 use `ipconfig` to view interfaces on your device

Now that we know about interfaces, it's time to reveal the big secret: devices DON'T actually have IP addresses. Yes, a device itself doesn't have a dedicated IP to identify itself. Devices have interface(s), and each interface has an IP address. This is designed as such because it allows a device to communicate with multiple other devices simultaneously .

As established data travels from one device to another on a network, and requires an IP address of the end-destination's device.

However, it is slightly more complicated than that. For a data packet to be sent to the destination interface, it actually needs to know the interface's MAC address.

▼ 💡 **Mac address**

> A 48-bit hexadecimal number, often displayed as six groups of two hexadecimal digits separated by colons or hyphens (e.g., `00:1A:2B:3C:4D:5E`).

- A mac address, just like an IP is an address that helps to identify devices in a LOCAL network

💻 CMD: use `getmac` to display the physical address of your network adapters

Q: Wait! but didn't we just learn that IP address are used to identify a device in a network? What's the point of IP's then?

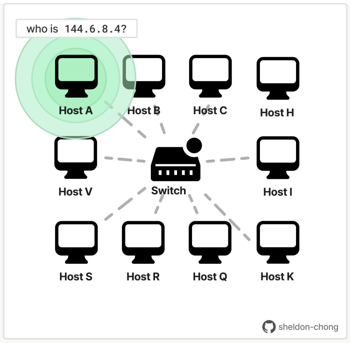
Don't worry however, IP's are not useless. IP's play an important role in obtaining the mac address, and in inter-network communication.

When a device (like a computer or router) wants to communicate with another device on the same local network but only knows its IP address, it sends out a ARP request. This request is broadcast to all devices on the local network.

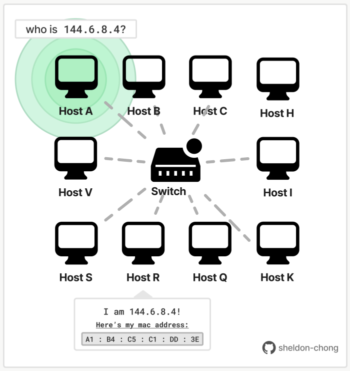
▼ 💡 **ARP request**

> A broadcast to all devices on a network to search and retrieve the MAC address associated with a given IP address

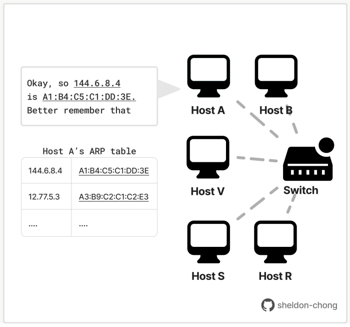
1. The ARP request packet contains the IP address of the destination device and asks, "Who has this IP address? Please send me your MAC address."



2. The device with the specified IP address responds with an ARP reply, which includes its MAC address. This reply is sent directly to the requesting device.



3. The requesting device receives the ARP reply and updates its ARP table (or cache) with the MAC address corresponding to the IP address. This allows the device to send data directly to the correct MAC address in future communications.


Okay, so 144.6.8.4 is A1:B4:C5:C1:DD:3E. Better remember that
Host A
Host B
Host V
Switch
Host S
Host R
sheldon-chong

Host A's ARP table

144.6.8.4	A1:B4:C5:C1:DD:3E
12.77.5.3	A3:B9:C2:C1:C2:E3
....	....

The main reason data packets use by MAC addresses instead of IP's to reach their destination is that it is more efficient and faster to do so.

▼ 🔄 **Summary of everything we've learnt so far:**

1. Two or more devices that are connected to each other are considered part of the same network.
2. For more connecting more than two devices in the same network, a switch comes in handy

3. Files that are sent across a network must be broken down into small units of data known as data packets.
4. Data packets are sent from and received by interfaces, one or more of which can be found on a device
5. Data packets use a MAC address to know which destination interface it should be sent to
6. To get the MAC address of the destination interface, the current device or switch will send an ARP request to all devices on the network, and receives the MAC address of the matching IP.

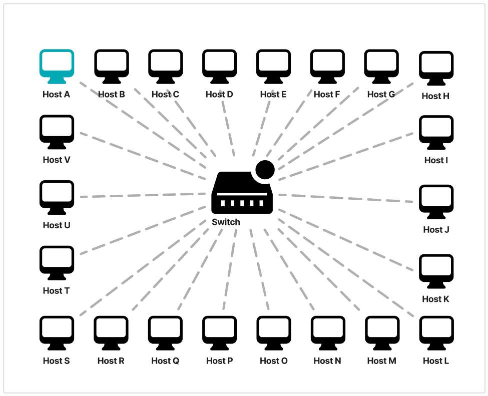
So far, we've learned about devices communicating within the same network.

- But what if a device wants to communicate with another device on other networks?
- What does it mean for devices to be on the same network?
- How do we know what network a device is on?

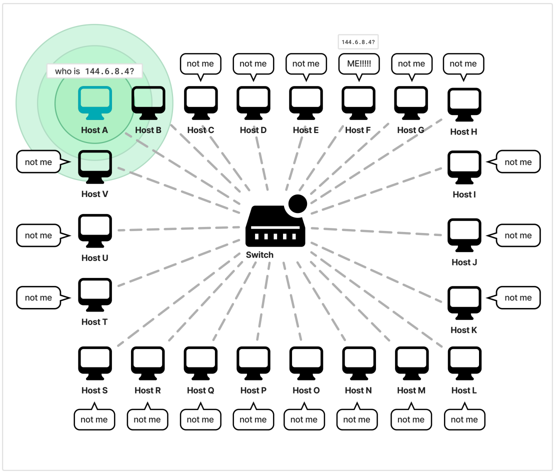
These questions will be discussed in the next section

Chapter 2: Multiple networks

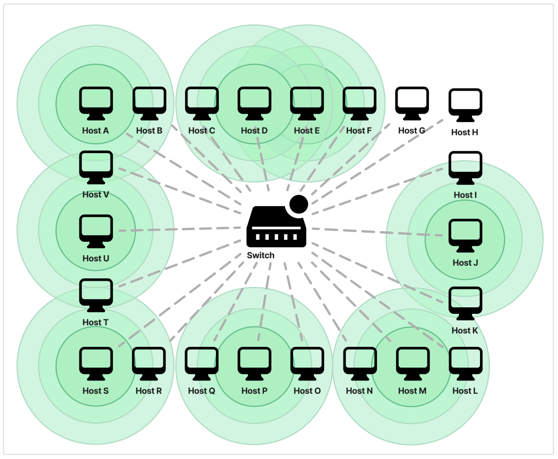
1. Let's set the stage. Device A wants to communicate with Device F in a large network of devices. All devices in the network have IP addresses, but Device A doesn't know the MAC address for Device F.



1. To get the MAC address, Device A sends an ARP (Address Resolution Protocol) broadcast request to all devices on the network. All devices receive this broadcast, but only Device F responds, revealing its MAC address so that Device A can communicate directly with it.



3. There's one tiny problem with this scenario: there are simply too many devices within the same network. Sending a broadcast to all devices would be inefficient. Furthermore, if many devices wish to communicate concurrently, each device would send its own broadcast to identify others, causing a flood of broadcast traffic. This is chaotic, messy, and inefficient, especially since not all devices need to communicate with each other.



4. This scenario is a result of all devices being on the **same network**, leading to excessive broadcast traffic. By dividing a large network into smaller, more manageable subnets, we can reduce the size of each broadcast domain, improving the overall network performance and organization. This is known as subnetting

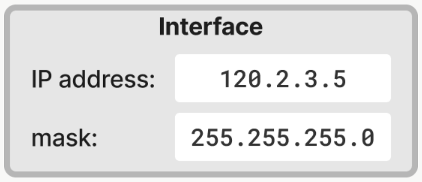
So, how do we separate a large network of devices such as this? This is where IP addresses become very important. These addresses will need to not only need to identify the host, but also what network the device belongs to. An IP has to convey these two aspects within the confines of 4 octets.

To solve this problem, IP addresses come with a component known as a subnet mask, which help to distinguish between the IP's network address and host address.

▼ 💡 **Subnet mask**

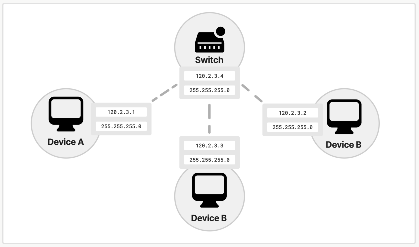
> A 32-bit address that is applied to an IP address to distinguish between the host and network portion of the address. It is in the same structure as an IP address, consisting of 4 octets delimited by dots.

- All IP's and interfaces come with a subnet mask



Interface

IP address:	120.2.3.5
mask:	255.255.255.0



A setup with IP addresses and their respective masks

💻 On CMD

- use `arp -a` to view the ARP cache stored in your current device

▼ ✏️ **Simple subnet mask calculation**

How does a subnet mask help convey the network and host address of an IP address? Let's take a look:

1. the region of 1's in the subnet mask indicate the network portion. The remaining 0s indicate the host number.

```
IP   : 11000000.01110010.01111001.00001111
mask : 11111111.11111111.11111111.00000000
                          network part | host part
```

2. Let's use a mask of `255.255.255.0`:

```
IP   : 192.114.121.15
mask : 255.255.255.0

IN BINARY:
IP   : 11000000.01110010.01111001.00001111
mask : 11111111.11111111.11111111.00000000
```

3. Let's say we have the IP `192.114.121.15` . We want to establish which portion of this address is the network and which portion is the host by specifying its subnet mask configuration.
4. To find the network address, the `AND` operation is performed on the IP address and mask.

```
  11000000.01110010.01111001.00001111
AND --------------------------------
  11111111.11111111.11111111.00000000
```

5. By getting the network portion, subsequently we also know the remaining bits of the IP address are for the host portion.

```
host address    = 00000000.00000000.00000000.00001111
network address = 11000000.01110010.01111001.00000000

host address    = 0.0.0.15
network address = 192.114.121.0
```

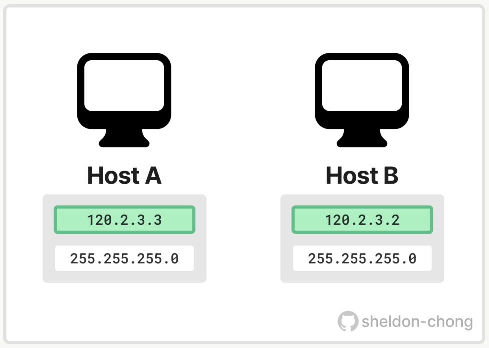
That's it. That's how you obtain the network and host portion of an IP address using a subnet.

5. The result from the operation is

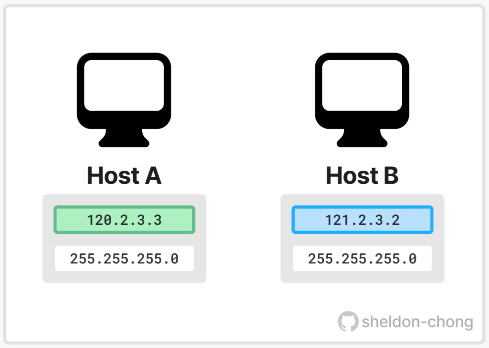
```
  11000000.01110010.01111001.00001111
AND --------------------------------
  11111111.11111111.11111111.00000000
  ↓↓↓↓↓↓↓↓ ↓↓↓↓↓↓↓↓ ↓↓↓↓↓↓↓↓ ↓↓↓↓↓↓↓↓
= 11000000.01110010.01111001.00000000 (we are left with the network address)
```

With some understanding of host and network addresses, let’s observe the rules for networks and communications:

▼ Understanding the network address

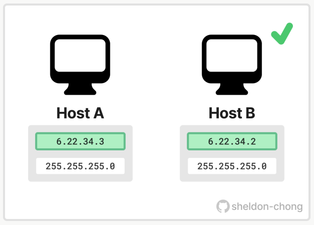


In this example, host A and Host B are part of the same network — `120.2.3.0` , as their first 3 octets are the same. This means they are allowed to connect with each other

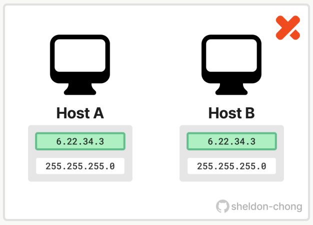


The first 3 octets of Host A differ from Host B, hence they are considered on “separate networks”. These devices require another intermediary (such as a router) to communicate with each other

Understanding host address

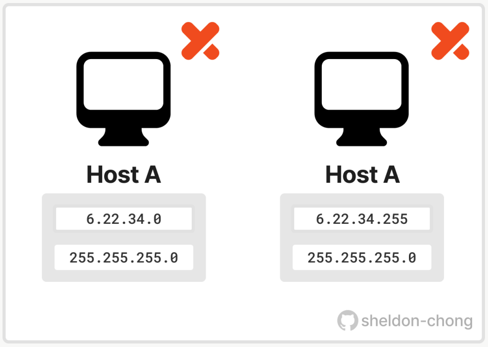


Both devices are part of the same network, with a unique host address. This configuration is valid



Both devices are part of the same network, but share the same address. This configuration will cause an error

Take note!:



Despite 0 and 255 being the extent of the host address, hosts can have only have an address between `1-254`.

- `0` is reserved for the network address. This number represents the entire network rather than any specific device on the network.
- `255` is reserved for the broadcast address within the network.

So far, we have discussed the usage of the subnet mask `255.255.255.0`, which is a configuration that specifies that the first 3 octets should be the network portion, and the last octet be the host portion.

```
IP   : 193.144.221.52
mask : 255.255.255.0
        network part | host part
```

However, you allocate more 0's to the host portion, and by doing so reduce the amount of available addresses for the network portion

```
mask   : 255.255.255.0
       network part | host part

mask   : 255.255.0.0
    network part | host part

mask   : 255.0.0.0
network part | host part
```

Summary

Decimal	Binary	Available Network addresses	Available host addresses per network
`255.255.255` `.` `0`	`00000000.00000000.00000000` `.` `11111111`	16,777,216 addresses range: `0.0.0` - `255.255.255`	256 addresses range: `0` - `255`
`255.255` `.` `0.0`	`00000000.00000000` `.` `11111111.11111111`	65,536 addresses range: `0.0` - `255.255`	65,536 addresses range: `0.0` - `255.255`
`255` `.` `0.0.0`	`00000000` `.` `11111111.11111111.11111111`	256 addresses range: `0` - `255`	16,777,216 addresses range: `0.0.0` - `255.255.255`

These are the 3 basic subnet mask configurations. Depending on how many networks you need, or devices you need per network, you can pick and choose from these masks.

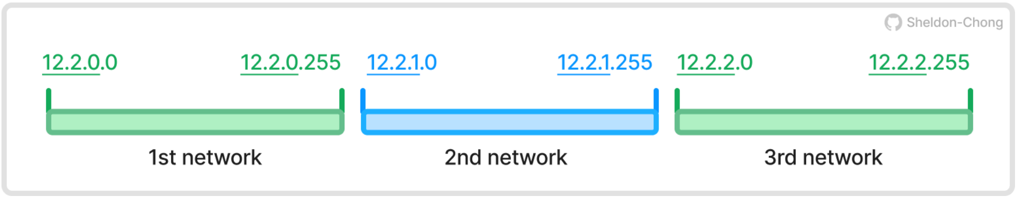
As you've guessed however, these masks are generally quite rigid, and the jump from 16,777,216 available 65,536 network addresses when switching the mask from `255.255.255.0` to `255.255.0.0` can be too drastic, or unsuited for one's needs. This is where we have to start configuring subnet masks to mask with different precisions.

In subnet masks, octets can have other numbers too, such as `224` or `192`, which in binary represent octets that are partially occupied by 1's.

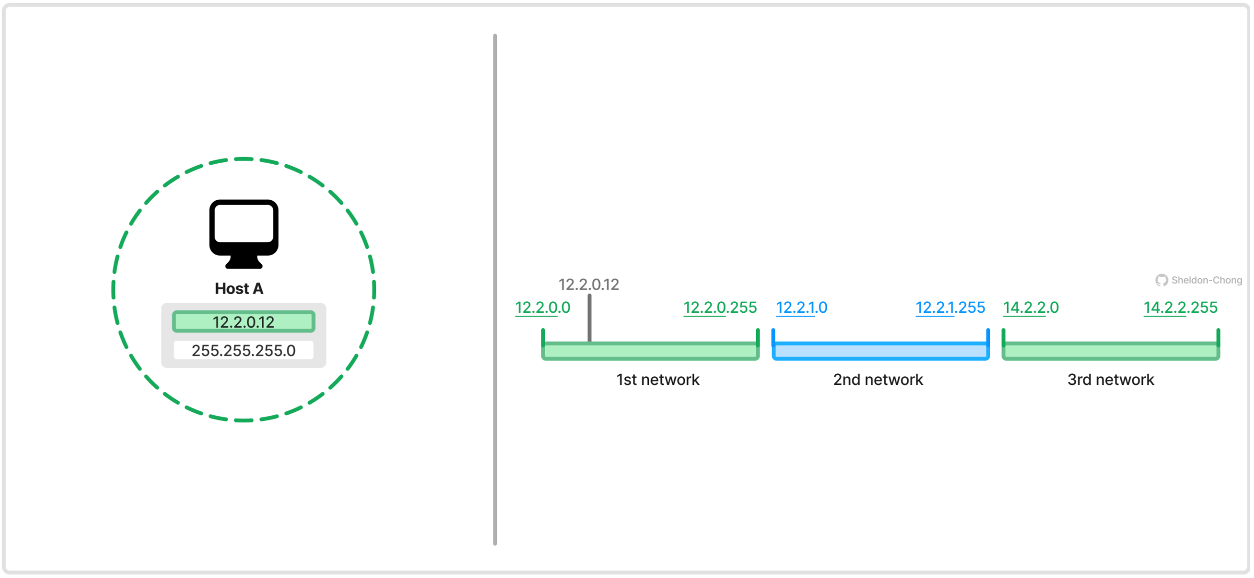
```
mask 255.255.255.224 = 11111111.11111111.11111111.11100000
mask 255.255.255.192 = 11111111.11111111.11111111.11000000
```

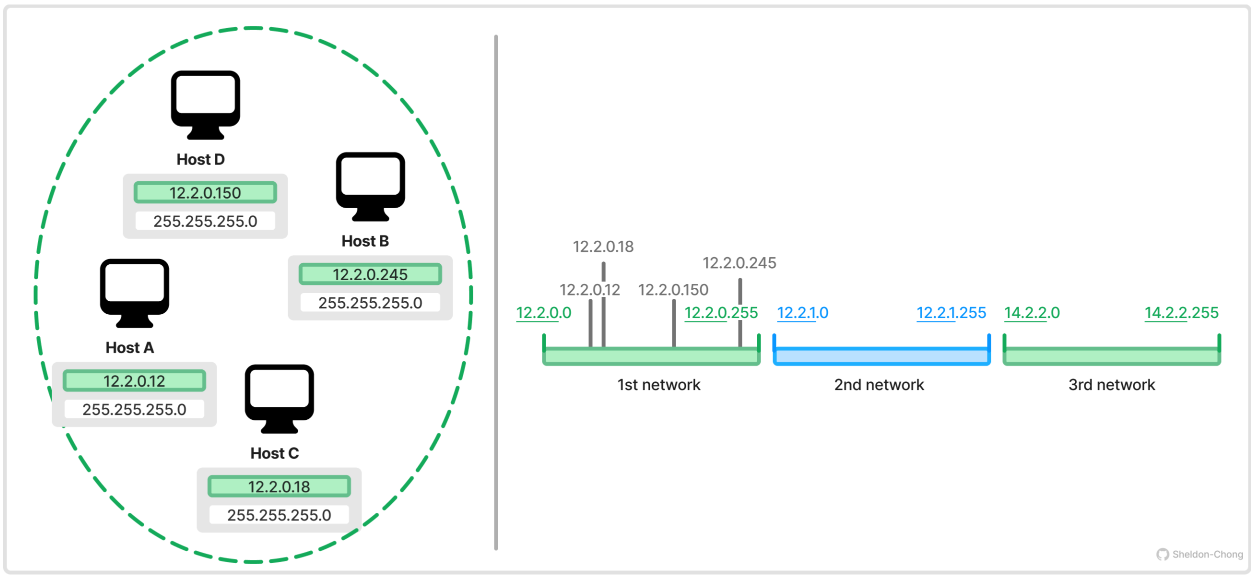
However, translating this into decimal is where it gets a little confusing, because it raises a question of how the ranges for host and network portions are defined. I personally call these masks "unaligned subnet masks".

To better understand how to deal with unaligned subnet masks, let's recontextualize what we've just learnt, by visualizing it in another form. This diagram visualizes the possible IP's as a range/scale starting from `12.2.0.0`, using a subnet mask of `255.255.255.0`

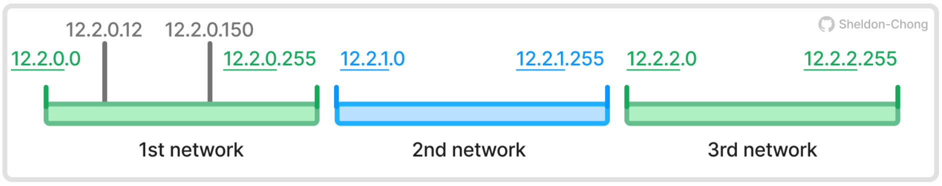


If one host belongs to the 1st network's range, all host in the same network must have a host address that is within that network's range





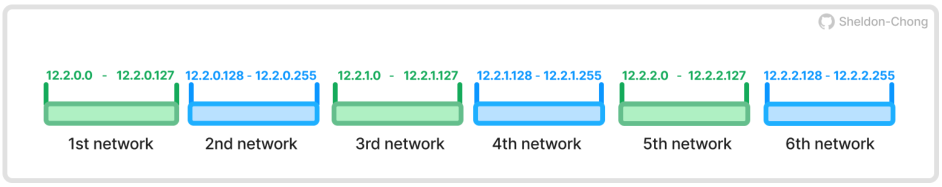
If we observe the IP addresses: `13.2.0.12` and `13.2.0.150` , both share the same host address, hence they both fall under network 1:



Now, let's let's visualize the scale of another subnet mask like `255.255.255.128`

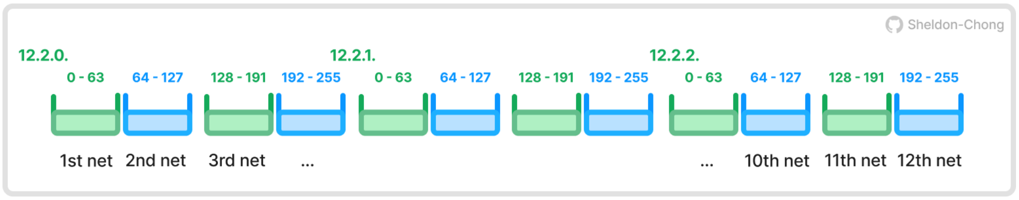
```
decimal format  | binary format
255.255.255.128 | 11111111.11111111.11111111.10000000
```

With this mask, we have allocated 1 extra bit to the network portion, while shrinking the the host portion by 1 bit. This means that we have more network address combinations, but less unique host combinations per network. Specifically; we have doubled the amount of network addresses, but halved the hosts per network. The scale gets subdivided



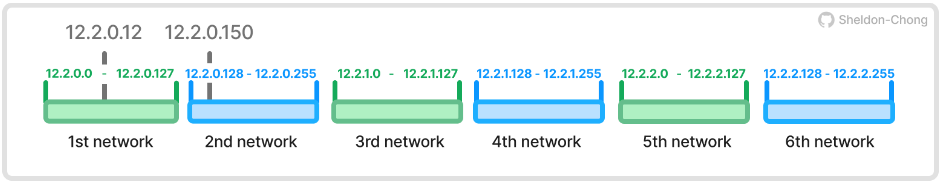
`0-127` are the available hosts addresses of the FIRST network, and `128-255` are the available hosts of the ANOTHER network.

Increase the bit by 1, and every range has been further subdivided.

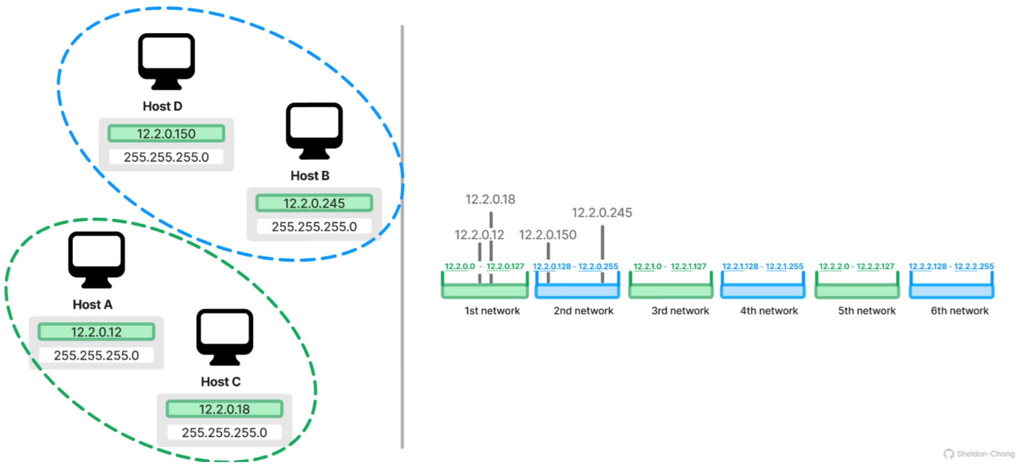


You know how the story goes from here...

Using the same IP addresses as before (`13.2.0.12` and `13.2.0.150`), with the mask `255.255.255.128`, notice that now both IP addresses are land in different networks. They no longer belong to the same network, despite sharing the same 3 octets, because the last octet has been subdivided by the subnet mask. This is the power of the subnet mask



Which means the four devices we had earlier are now considered part of separate networks

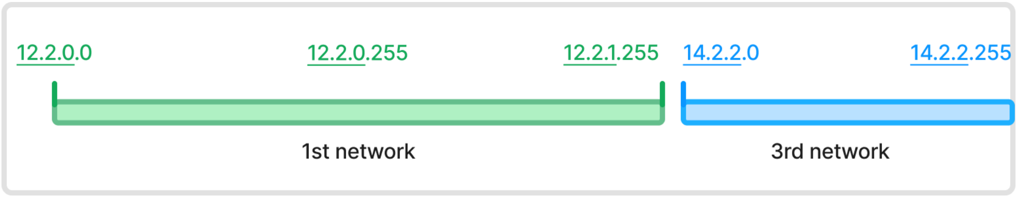


By subnetting, we have created artificial boundaries within a single octet. If an host address exceeds or succeeds the current network range it is in, it will be deemed part of another network.

Similarly, we can remove bits from the network portion, allocating more space for network addresses.

```
mask 255.255.254.0 = 11111111.11111111.11111110.00000000
mask 255.255.252.0 = 11111111.11111111.11111100.00000000
```

In the scale, octets are combined together, doubling the extent of every range.



This means that for every two times the octet reaches 255, it would register as another network.

Summary: As we increase the number of bits in the subnet mask, **the number of networks doubles** while **the number of available host addresses within each network is halved**

As we decrease the number of bits in the subnet mask, **the number of networks halve** while **the number of available host addresses within each network is doubled**

This phenomenon happens as each additional bit shifts the boundary between the network portion and the host portion of the IP address.

- With a mask of `11111111.11111111.11111111.00000000` (`/24`), the available host addresses range from 0 to 255 (254 usable hosts).
- With a mask of `11111111.11111111.11111111.10000000` (`/25`), the host addresses range from 0 to 127 (126 usable hosts).
- With a mask of `11111111.11111111.11111111.11000000` (`/26`), the host addresses range from 0 to 63 (62 usable hosts).
- With a mask of `11111111.11111111.11111111.11100000` (`/27`), the host addresses range from 0 to 31 (30 usable hosts).
- And so on…

By the way, the notation `/24` , `/25` , `/26` etc indicates how many bits in the IP address are used for the network portion. This is notation is known as CIDR:

▼ 💡 **CIDR (Classless Inter-Domain Routing)**

> A method for allocating IP addresses. CIDR allows variable-length subnet masking (VLSM), meaning you can allocate IP addresses based on need, rather than being restricted to fixed blocks like Class A, B, or C.

- Facilitates efficient and flexible division and use of IP address space
- CIDR Notation: The format used in CIDR is
  - *E.g. we have a CIDR notation of:* `192.168.0.0` `/24`
    - The **IP address** is to the left (before the slash)
    - The prefix length (number after the slash) indicates how many bits are used for the network portion of the address. This represents the subnet mask
      - `/24` = `11111111.11111111.11111111.00000000` (24 bits)
      - `/25` = `11111111.11111111.11111111.10000000` (25 bits)
      - `/23` = `11111111.11111111.11111110.00000000` (23 bits)

Use the cheat sheet below to get a quick reference of CIDR numbers and the respective host the networks available

Cheat Sheet

Subnet Mask	CIDR	Binary Notation	Available Addresses
255.255.255.255	/32	11111111.11111111.11111111.11111111	1
255.255.255.254	/31	11111111.11111111.11111111.11111110	2
255.255.255.252	/30	11111111.11111111.11111111.11111100	4
255.255.255.248	/29	11111111.11111111.11111111.11111000	8
255.255.255.240	/28	11111111.11111111.11111111.11110000	16
255.255.255.224	/27	11111111.11111111.11111111.11100000	32
255.255.255.192	/26	11111111.11111111.11111111.11000000	64
255.255.255.128	/25	11111111.11111111.11111111.10000000	128
255.255.255.0	/24	11111111.11111111.11111111.00000000	256
255.255.254.0	/23	11111111.11111111.11111110.00000000	512
255.255.252.0	/22	11111111.11111111.11111100.00000000	1024
255.255.248.0	/21	11111111.11111111.11111000.00000000	2048
255.255.240.0	/20	11111111.11111111.11110000.00000000	4096
255.255.224.0	/19	11111111.11111111.11100000.00000000	8192
255.255.192.0	/18	11111111.11111111.11000000.00000000	16384
255.255.128.0	/17	11111111.11111111.10000000.00000000	32768
255.255.0.0	/16	11111111.11111111.00000000.00000000	65536
255.254.0.0	/15	11111111.11111110.00000000.00000000	131072
255.252.0.0	/14	11111111.11111100.00000000.00000000	262144
255.248.0.0	/13	11111111.11111000.00000000.00000000	524288
255.240.0.0	/12	11111111.11110000.00000000.00000000	1048576
255.224.0.0	/11	11111111.11100000.00000000.00000000	2097152
255.192.0.0	/10	11111111.11000000.00000000.00000000	4194304
255.128.0.0	/9	11111111.10000000.00000000.00000000	8388608
255.0.0.0	/8	11111111.00000000.00000000.00000000	16777216

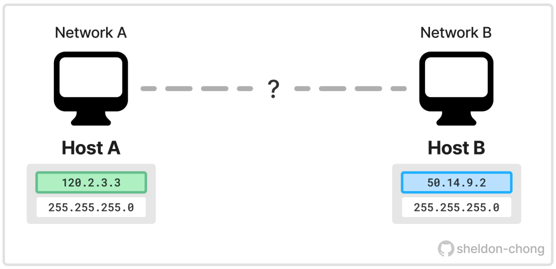
🔄 Recap

1. Classes and CIDR are ways in which we specify the configurations of subnet masks.
2. When we increase the bits of the mask, the number of network addresses are doubled, while the host addresses available per network are halved.
3. Subnet masks serve as a way to highlight the network and host number of a device.
4. Allowing devices to communicate with each other from separate networks first requires that data packets are able to distinguish between networks and host, which is where subnet masks come in
5. Separating large networks into smaller ones help with traffic and efficiency of communication

Chapter 3: Connecting networks together

Now that we know how to read the host and network addresses, let's move onto how we connect devices of different networks together.

Let's say host A wants to send resources to host B, but they are on separate networks

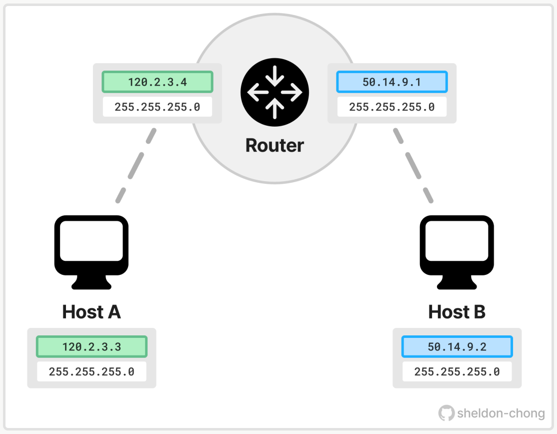


For the two to communicate with each other, we need an intermediary, which is where the router comes in

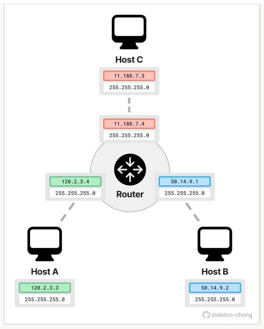
▼ 💡 **Router**

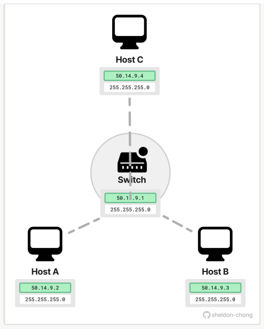
> A physical device that connects devices from different networks together, allowing them to communicate with each other.

- The diagram below showcases a router connecting Host A and Host B together, essentially functioning as a bridge between the two devices:



- Unlike a switch, a router has an interface for every network.





Furthermore, switches only connect devices of the same network together, while routers connect devices from different networks together

- Routers often act as **Dynamic Host Configuration Protocol (DHCP) servers**, assigning **IP addresses** to devices within a network.
- Routers provide security features such as **firewalls** and **access control lists (ACLs)** to filter traffic and prevent unauthorized access to networks.
- They also help translate private IP addresses (used inside a network) into public IP addresses for internet communication, a process called **Network Address Translation (NAT)**.

Many routers function as a Dynamic Host Configuration Protocol (DHCP), assigning IP addresses to devices within a network. Routers assign IP's from Private IP Address ranges:

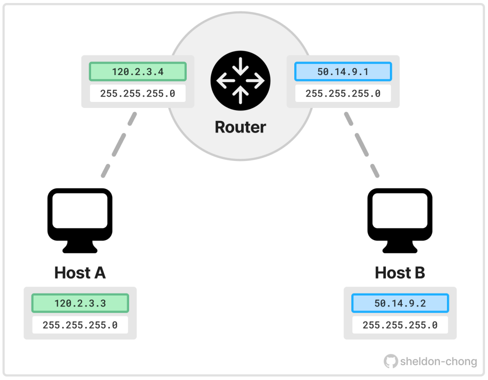
▼ 💡 **Private IP addresses**

> A private IP address is used within a private network. It is used for communication within a local area network (LAN).

Class	IP Address Range	Default Subnet Mask	Network Bits	Host Bits	Number of Networks	Hosts per Network
A	`0.0.0.0` to `127.255.255.255`	`255.0.0.0` (or `/8`)	8	24	128 (from `0` to `127`)	16,777,214 (2^24 - 2)
B	`128.0.0.0` to `191.255.255.255`	`255.255.0.0` (or `/16`)	16	16	16,384 (from `128.0` to `191.255`)	65,534 (2^16 - 2)
C	`192.0.0.0` to `223.255.255.255`	`255.255.255.0` (or `/24`)	24	8	2,097,152 (from `192.0.0` to `223.255.255`)	254 (2^8 - 2)

An important thing to note, ARP requests don't get past the router. This design may appear to be a flaw, but it is beneficial to ensure that broadcast are contained within their own network and don't spill over, reducing traffic congestion.

But if broadcasts are local, how does Host A obtain the MAC address of host B which happens to be in another network?



There are a few ways in which host A can obtain the MAC address of host B:

▼ 📜 **Obtaining the IP address of a device on another network**

1. **Manually configure IP addresses (Static IP Addressing):** In some cases, Device A may already know Device B's IP address because it was manually configured by an administrator, and won't be automatically assigned or changed
2. **DNS (Domain Name System)**: In most networks, **Device A** doesn't need to know Device B's IP directly but knows its **hostname** or **domain name** *(e.g., `deviceB.local` or `example.com`)*. It finds the corresponding **IP address** for that hostname, using a process called DNS Resolution

   ▼ **DNS Resolution**

   🌐 DNS Resolution is the process of converting a website name (hostname) into an IP address so your device can connect to it.

   - **DNS Query Initiation**: When a user types a domain name (like `example.com`) into their web browser, the device first checks its **DNS cache** to see if it has recently looked up the domain. If it's in the cache, the IP address is used immediately.
   - If the IP address is not cached, the device sends a **DNS query** to its configured **DNS resolver**. This is usually the DNS server provided by the Internet Service Provider (ISP), but it could also be a public DNS service like Google's **8.8.8.8**.
   - The DNS resolver begins the recursive query process by contacting a series of servers in search of the domain name
   - The authoritative DNS server responds to the resolver with the IP address corresponding to the domain name (e.g., `203.0.113.5` for `www.example.com`).
   - The resolver sends the IP address back to the client device, which can now use the IP address to establish a connection to the target server.

3. **DHCP Communication**: If Device B has received its IP address through DHCP, the DHCP server often maintains a **list of active devices and their IP addresses.** Devices may be able to query the DHCP server for other devices' IP addresses, or an **admin interface** may provide this information.

Navigating networks can become quite complex. Many networked devices come with routing tables, especially routers. These tables explicitly tell data packets how they should travel to their end destination

▼ 💡 **Routing table**

> A set of instructions stored on a network device (E.g. a router) that guides data packets on how to travel to their destination. It provides the necessary information for the router to decide where to forward incoming packets based on their destination IP address.

- **Routing tables look something like this:**

Destination Network	Next Hop	Interface	Metric
192.168.1.0/24	192.168.1.1	eth0	1
192.168.1.128/25	192.168.1.129	eth1	2
192.168.2.0/24	192.168.2.1	eth2	1
10.0.0.0/8	10.0.0.1	eth3	5

1. **Destination Network**: Contains the network or host that the data packet is intended for. The router checks the destination IP address of an incoming packet and compares it to this list of networks in the table to determine the best match. The network listed here is usually represented in CIDR notation (e.g., `192.168.1.0/24`).
    a. If no specific match is found, the router may use a **default route** (typically `0.0.0.0/0`) for all other destinations.

       E.g. packet is headed for `192.168.1.0`. It arrives at this router. The router checks whether this address exists in the destination network column. If yes, it will utilize the instructions there such as specified subnet mask configuration, next hop, interface to send the packet

2. **Next Hop**: The IP address of the **next router** or network device to which the packet should be forwarded. The router sends the packet to this "next hop," which is typically another router that’s a step-closer to the destination.

- **Interface**: The router’s local network interface (like `eth0`, `eth1`, etc.) through which the packet should be sent. It tells the router which physical or logical port to use.
- **Metric:** A value that represents the "cost" of the route, usually based on the number of hops, bandwidth, or other criteria. Lower metrics are preferred, meaning the route with the lowest cost will be chosen if there are multiple routes to the same destination.

💻 use `route print` to display the routing table of your device

We have talked about private IP addresses, and how they are assigned by the router. However, when a device wants to communicate with the internet, it needs to use a public IP addresses.

▼ 💡 **Public IP address**

> An address that is assigned to a device directly connected to the internet. It is globally unique and can be accessed from anywhere on the internet.

- Public IPs are used by routers and servers to communicate with other devices on the internet. Websites and online services use public IPs to send data to and receive data from users.

📎 Find your router’s public IP address with the following URL: https://www.whatismyip.com/

📜 Connecting with the internet

1. **Your Device**: Router assigns a private IP address (e.g., `192.168.1.100`)
2. **Your Router**: Has a public IP (e.g., `203.0.113.5`) assigned by your ISP.
3. your router uses **Network Address Translation (NAT)** to translate the private IP to a public IP for external communication. The internet can simply send data packets to this public IP address, and they shall be directed to their respective devices.
4. **Internet**: Once the public IP is assigned, your device can communicate with websites and other internet services. Data sent from the internet to devices on a local network, and vice versa are facilitated by these routers, essentially acting as an intermediary.

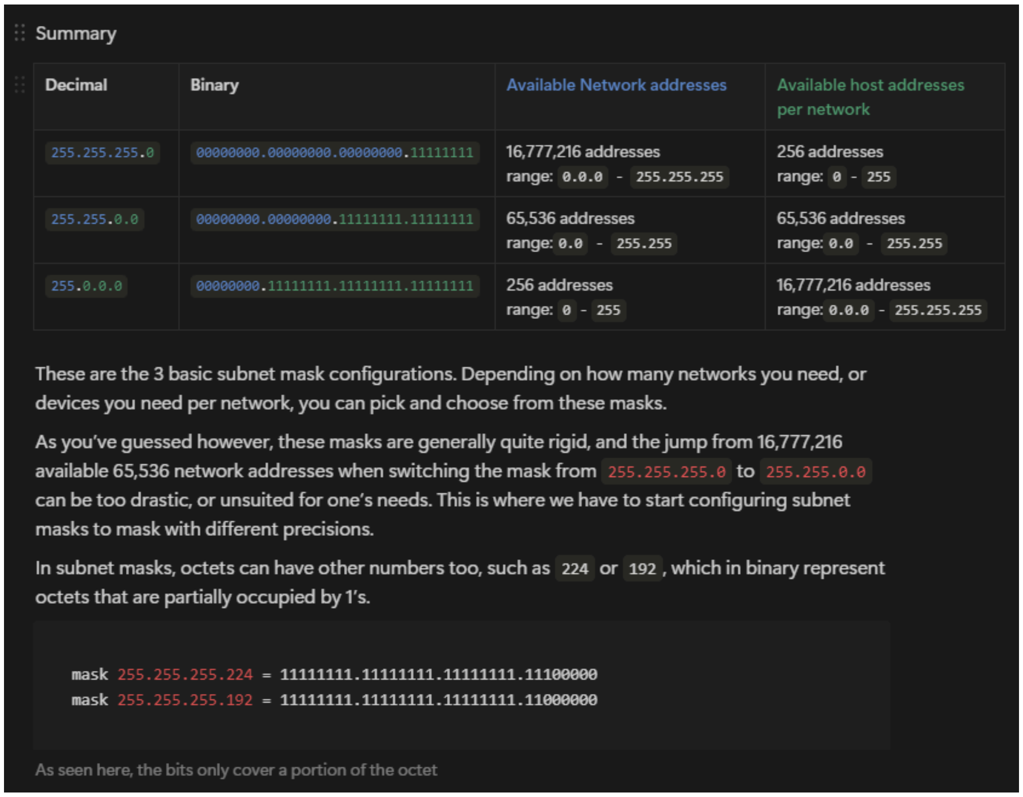
> **💻 On CMD:**
>
> - **Use `ping` to test connectivity with another device on the internet**
>   - `ping 8.8.8.8`
>   - `ping google.com`
>   - `ping nonexistent.non`
> - **Use `nslookup` to resolve domain names into IP addresses or vice versa**
>   - `nslookup google.com`
> - **CMD: use `tracert` to view that path packets take to reach their destination**
>   - `tracert google.com`

🔄 Summary

1. Routers connect multiple networks together
2. Routers assign private IP addresses to devices on a local network, and public addresses to IP’s that wish to communicate with the internet
3. Devices can obtain the MAC address of other devices on separate networks using static IP addresses, Domain Name System resolution and communicating with the DHCP (which is typically in the router)
4. Routing tables are found on many networked devices, which help data packets to know how to travel to their end-destination by specifying a general route

End of guide :)

More coming in the future



Summary

Decimal	Binary	Available Network addresses	Available host addresses per network
`255.255.255.0`	`00000000.00000000.00000000.11111111`	16,777,216 addresses range: `0.0.0` - `255.255.255`	256 addresses range: `0` - `255`
`255.255.0.0`	`00000000.00000000.11111111.11111111`	65,536 addresses range: `0.0` - `255.255`	65,536 addresses range: `0.0` - `255.255`
`255.0.0.0`	`00000000.11111111.11111111.11111111`	256 addresses range: `0` - `255`	16,777,216 addresses range: `0.0.0` - `255.255.255`

These are the 3 basic subnet mask configurations. Depending on how many networks you need, or devices you need per network, you can pick and choose from these masks.

As you've guessed however, these masks are generally quite rigid, and the jump from 16,777,216 available 65,536 network addresses when switching the mask from `255.255.255.0` to `255.255.0.0` can be too drastic, or unsuited for one's needs. This is where we have to start configuring subnet masks to mask with different precisions.

In subnet masks, octets can have other numbers too, such as `224` or `192`, which in binary represent octets that are partially occupied by 1's.

```
mask 255.255.255.224 = 11111111.11111111.11111111.11100000
mask 255.255.255.192 = 11111111.11111111.11111111.11000000
```

As seen here, the bits only cover a portion of the octet