Introduction to Computer Networks

2025-09-11

Table of contents

1 Introduction

A network is a "bunch of connected entities". What constitutes an entity or the connection really depends on the context in which you are looking at the network.

In the context of network theory, the entities are typically referred to as **nodes** and the connections are referred to as **links**. There is a lot of overlap between networks and the concept of *Graphs* which you should have covered in *CSC 2203: Data Structures*

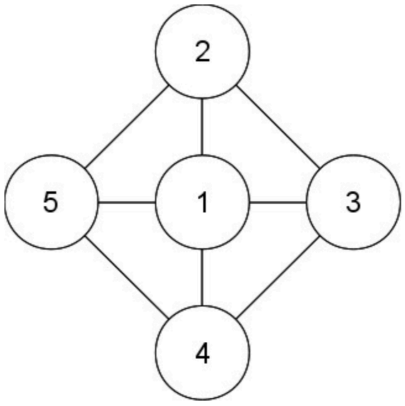


Figure 1: A basic network

In the context of the internet (or any computer network), the nodes are typically computers, and the links are the multiple ways that computers can be connected to each other e.g. wirelessly, ethernet cables, etc.

2 Routing

The main use of a computer network is to facilitate the transfer of messages between two nodes. The term *routing* is the process of sending a message along a path. We shall discuss a very basic example of what this process could look like in a more realistic network.

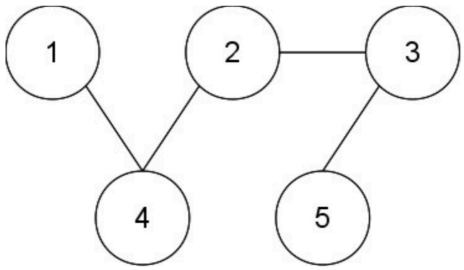


Figure 2: Routing a message through a network

Assume that I had to send a message from Node 1 to Node 5.

- Node 1 is connected only to Node 4. Therefore, the message goes to Node 4
- Node 4's routing table shows the best way to get a message to Node 5 is to send it to Node 2. Therefore, the message goes to Node 2

- Node 2's routing table shows the best way to get a message to Node 5 is to send it to Node 3. Therefore, the message goes to Node 3
- Node 3's routing table shows it is directly connected to the destination. Therefore, the message goes to Node 5.

Note that no single node in the network has knowledge of every other node in the network. All they have is a direct connection to another node that they believe will get the message closer to the target node.

3 IP Addresses

In the previous examples, we used numbers for the nodes. In reality, each node in a computer network has a unique number that can be used to identify it out of all the computers in the network.

3.1 IPv4

Figure 3: IPv4 address structure

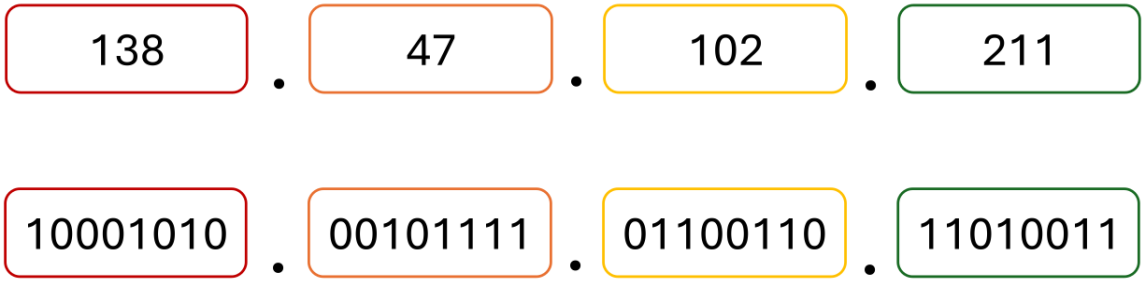


Figure 4: IPv4 address example

IPv4 addresses are made up of 4 **octets** i.e. 4 groups of 8 bits each. Each octet can take on any value in the range [0, 255] i.e. 256 unique values.

This means that there are a total of

$$256^4 = 4,294,967,296$$

unique addresses possible.

This is clearly not enough for every individual computer or device connected to the internet to have its own unique address. They dealt with this restriction in a few notable ways.

1. **Private IP Addresses**. Some ip address ranges e.g. 10.0.0.0/8, 172.16.0.0-172.31.0.0/12 and 192.168.0.0/16 can NOT be accessed from the internet. They are reserved for private networks and can be reused without any restriction. You might have noticed that your internal virtual machine internet address was from this range.
2. **Network Address Translation (NAT)** allows routers to represent multiple internal private IP addresses as a single public IP address when communicating with the internet. You might notice that your home router does something like this i.e. you and your roommate might have the same public facing ip address, but different internal addresses and the router knows who to send which data to.
3. **Dynmaic IP Assignment (DHCP)** allows Internet Service Providers to reuse addresses i.e. a computer cannot easily just have the same permanent public IP address. If you turn off your computer, your IP address could be assigned to someone else.
4. **Carrier-Grade NAT (CGNAT)** is similar to NAT but at a much larger scale i.e. tens of thousands of subscribers could be assigned one public facing IP address.
5. **Subnetting and Classless Inter-Domain Routing (CIDR)**. Back when 4 billion addresses seemed enough, ip addresses were split into rigid classes based on how many octets were for the network id i.e. class A used 1 octet for the network ID, leaving 256^3 addresses for the hosts, class B used two octets for the network ID, leaving 256^2 for the hosts, and class C used three octets for the network ID, leaving just 256 addresses for the hosts. Class A was of course more expensive to get. The problem, however, was that there was a lot of wastage. If a company had 500 hosts, they had to use a class B, which would leave $256^2 - 500$ unassigned addresses. These days, the differences are not so rigid, and the network portion of the address can be any number, not just full octets. As an example, as of right now, Louisiana Tech's network portion is 23 bits.

```
enp0s31f6: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 138.47.99.84  netmask 255.255.254.0  broadcast 138.47.99.255
        inet6 fe80::4ed7:17ff:fe94:e346  prefixlen 64  scopeid 0x20<link>
        ether 4c:d7:17:94:e3:46  txqueuelen 1000  (Ethernet)
        RX packets 98861907  bytes 28667473449 (28.6 GB)
        RX errors 0  dropped 536  overruns 0  frame 0
        TX packets 5362384  bytes 1549076829 (1.5 GB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
        device interrupt 19  memory 0x72380000-723a0000
```

Figure 5: Output of `ifconfig` on my system showing 23 bit netmask

3.2 IPv6

Figure 6: IPv6 address structure

Figure 7: IPv6 address example

IPv6 addresses are made up of 8 *hextets*, where a hextet is a group of 16 bits). Even if they were to permanently assign an ip address to a unique computer, IPv6 still has space for

$$65,536^8 \approx 3.4 \times 10^{38}$$

Just for context, the estimated number of grains of sand on planet earth is

$$7.5 \times 10^{19}$$

which means that each grain of sand on earth can have its own 4.6×10^{18} ip addresses.

4 Packet Structure

Messages that are sent over a computer network are split into smaller units called packets. Each packet can potentially take its own unique path from the origin to the destination.

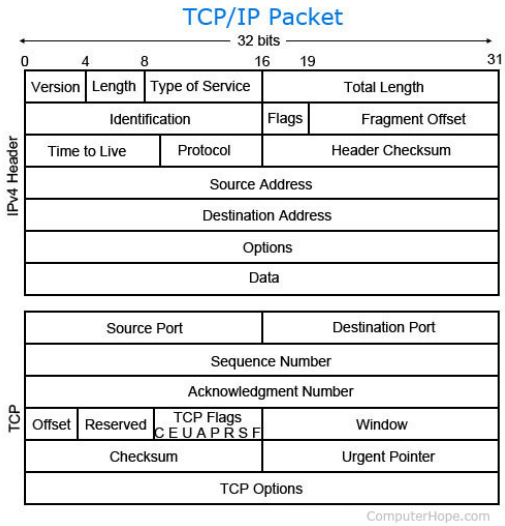


Figure 8: TCP/IP packets

Each packet will contain a lot of information to enable successful routing, and reassembly into the original message at the destination.

This information includes:

- A source IP address
- A destination IP address
- A sequence number
- A piece of the original message or data

The sequence number is used to reassemble the message from individual packets to a complete message while maintaining the order. As an example, the table below shows how a message "Meet me at midnight by the park." might be split into packets and sent from Node 1 to Node 5.

Source	Destination	Sequence Number	Data
1	5	1	`Meet`
1	5	2	`me`
1	5	3	`at m`
1	5	4	`idni`
1	5	5	`ght`
1	5	6	`by t`
1	5	7	`he p`
1	5	8	`ark.`

Figure 9: Message to packets

5 Netmasks

A *network mask* (or *netmask*) is a binary value that is used to divide an ip address into the *network id* and the *host id*.

A netmask can be represented as either an ip address i.e. the decimal equivalent of the binary value of each octet e.g. 255.255.255.0 **or** it can be represented as the number of 1s in the netmask e.g. /24

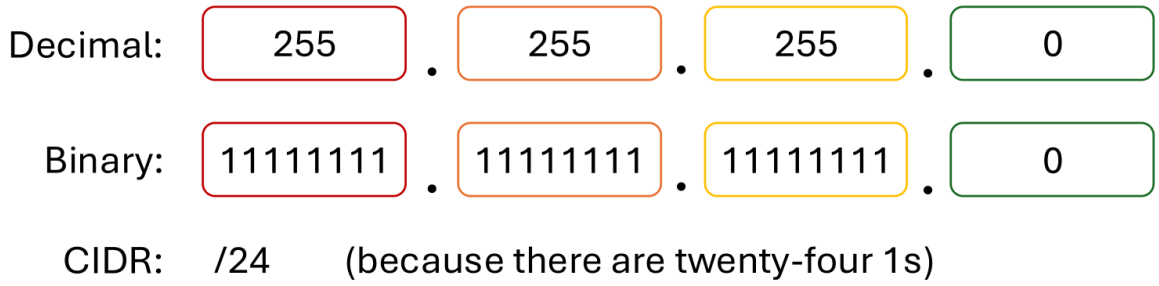
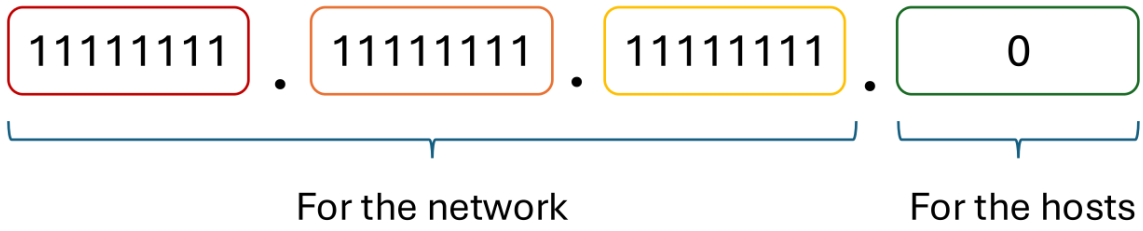


Figure 10: Ways of representing a Netmask

The netmask will always be made up of consecutive 1's and then consecutive 0's. The portion of the netmask that is made up of 1's is an indication of how much of the associated IP address is for the network. The portion of the netmastk made up of 0's is an indication of how much of the IP address is reserved for the individual/unique hosts on that network.



In the diagram/example above, each host (device on the network) will receive a unique value from 1 to 254 in the final octet.

If the router needs to send a *broadcast* message i.e. a message to all the hosts on the network, it will use the final .255 address in its message.

5.1 Calculating the Network ID

The netmask and ip address of any host can be used to calculate the network id.

Host IP Address	**192.168.1.100**
Netmask	**255.255.255.0**

Given the example host IP address and netmask above, the network id can be calculated from

```
network_id = ip_address & netmask
```

	11000000.10101000.00000001.01100100	Host IP Address
AND	11111111.11111111.11111111.00000000	Netmask
	11000000.10101000.00000001.00000000	Network ID

Network ID	**192.168.1.0**

> **Note**
>
> Recall that an *AND* between a 1 and any bit, is that bit. And an *AND* between a 0 and any bit is a 0. Therefore, *AND* between a netmask and any ip address maintains the first bits of the ip address, and makes the remaining bits 0.

5.2 Calculating the Host ID

To calculate the host ID from the netmask and ip address, we complement the netmask before putting it through the same operation as above.

```
host_id = ip_address & ~ netmask
```

	11000000.10101000.00000001.01100100	Host IP Address
AND	00000000.00000000.00000000.11111111	NOT Netmask
	00000000.00000000.00000000.01100100	Host ID

Host ID	**0.0.0.100**

5.3 Calculating the Broadcast address

The broadcast address can also be calculated from the ip address and netmask using the *OR* operation instead of the *AND* operation.

```
broadcast_addr = ip_address | ~ netmask
```

	11000000.10101000.00000001.01100100	Host IP Address
OR	00000000.00000000.00000000.11111111	NOT Netmask
	11000000.10101000.00000001.11111111	Broadcast Addr

Broadcast Addr	**192.168.1.255**

6 Conclusion

This section should have presented you with a very basic introduction to some of the terminology and basics in Computer Networks. This and more will be covered in **CSC 4503: Computer Networks**