Network Layer

Pavlos Grigoriadis - pgrigor@csd.uoc.gr
Christina Papachristoudi - chrisp@csd.uoc.gr

Network Layer

Data Plane

How a router forwards the packets that arrive in its **incoming interfaces** to **outgoing interfaces**

Consists of:

- Header inspection
- Implemented mainly in hardware
- Follows the instructions given by the Control Plane
- Forwarding Table

Control Plane

How a packet is routed among the routers (**end-to-end** routing).

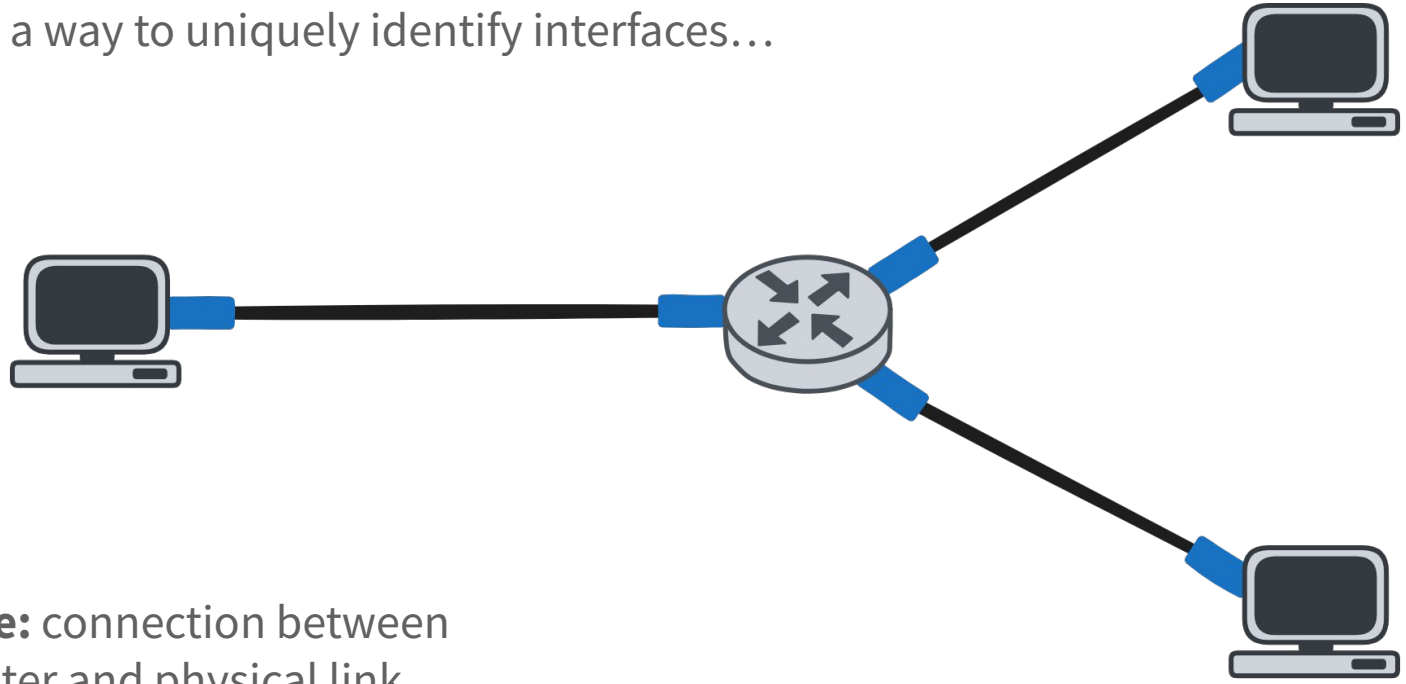
Consists of:

- Implemented mainly in software
- Routing Algorithms and Protocols
- Routing Table

Data Plane

Addressing

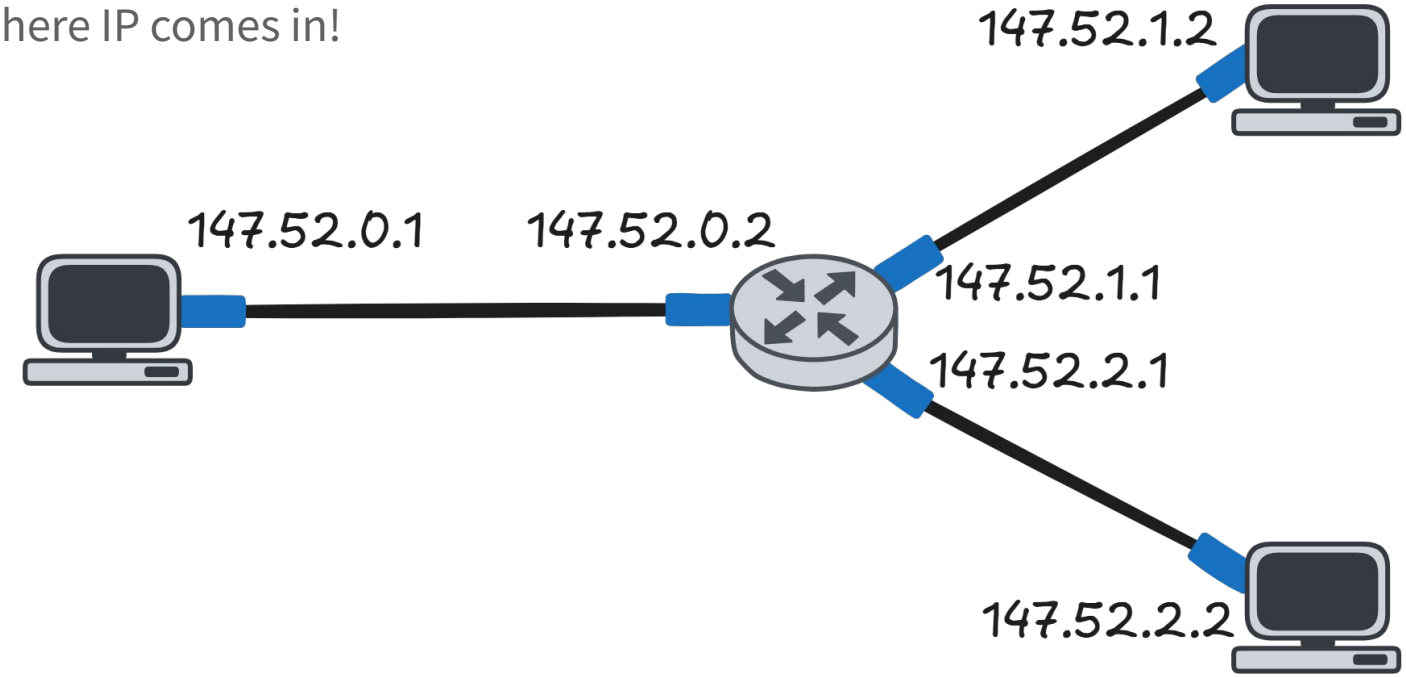
We need a way to uniquely identify interfaces…



interface: connection between host/router and physical link

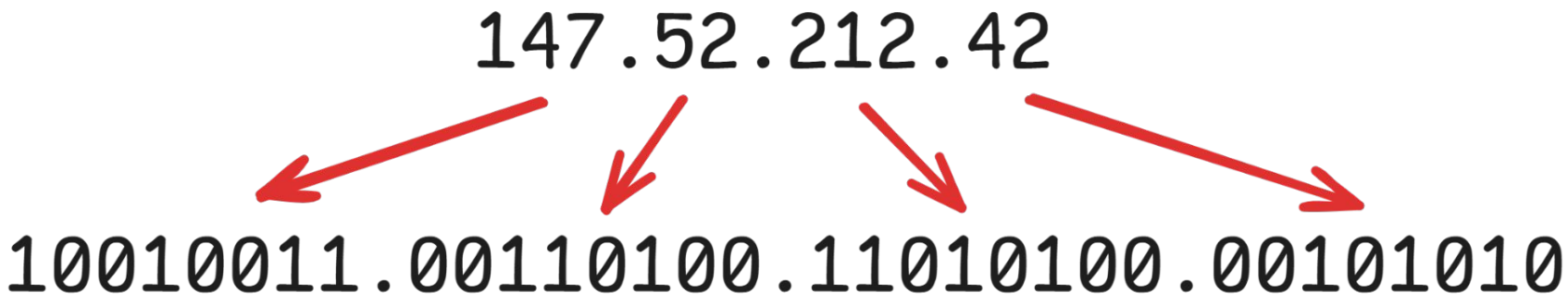
Addressing

That’s where IP comes in!



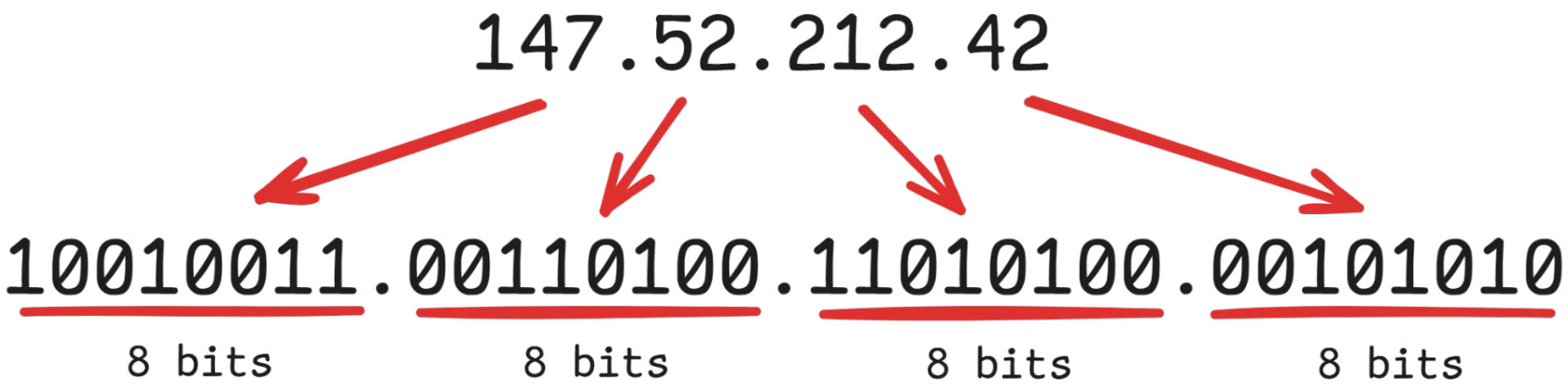
IPv4

- a unique identifier for each **interface**
- 32-bits, decimal representation **a.b.c.d**
- 4 octets of 8 bits



IPv4

- each octet is **8 bits** → can represent $2^8 = 256$ numbers
- so valid octets are values between **0 - 255**



Prefixes

- Each network has its own **prefix**
  - a range of IPs it can assign to its devices
- **a.b.c.d/x**
  - **x** is the subnet **mask:** defines how many bits the subnet part is
  - the first **x** bits are the **subnet part**: is the same for all IPs inside a subnet
  - the rest are the **host part**: is unique per host

`147.52.0.0/16`

Prefixes

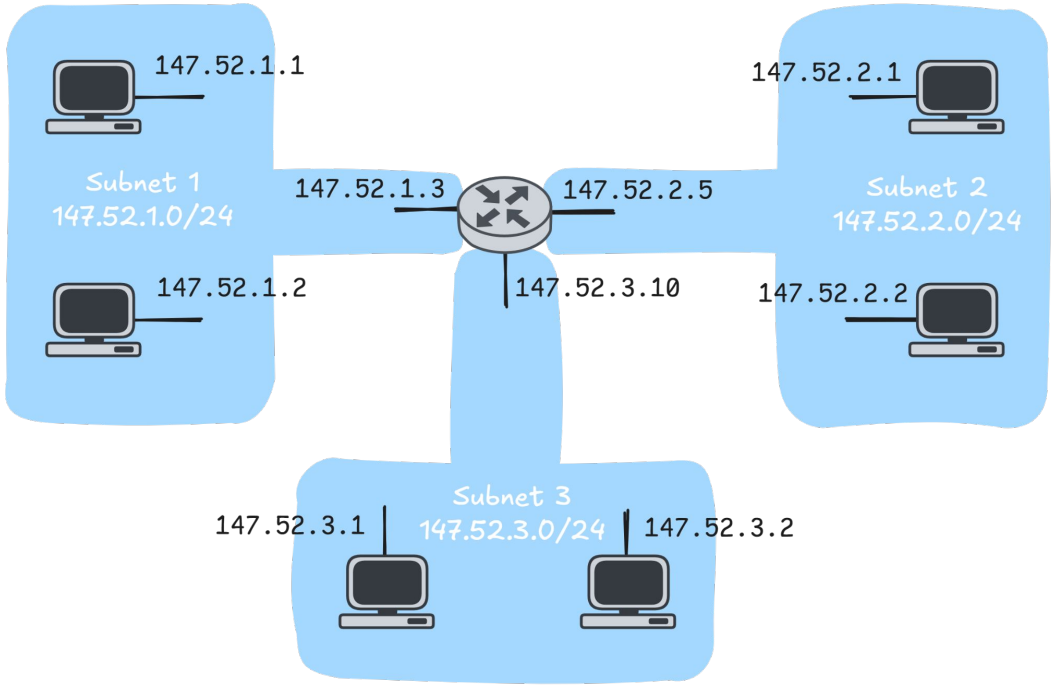
- Each network has its own **prefix**
  - a range of IPs it can assign to its devices
- **a.b.c.d/x**
  - **x** is the subnet **mask:** defines how many bits the subnet part is
  - the first **x** bits are the **subnet part**: is the same for all IPs inside a subnet
  - the rest are the **host part**: is unique per host

Subnet

Interfaces with **same subnet part** of IP address

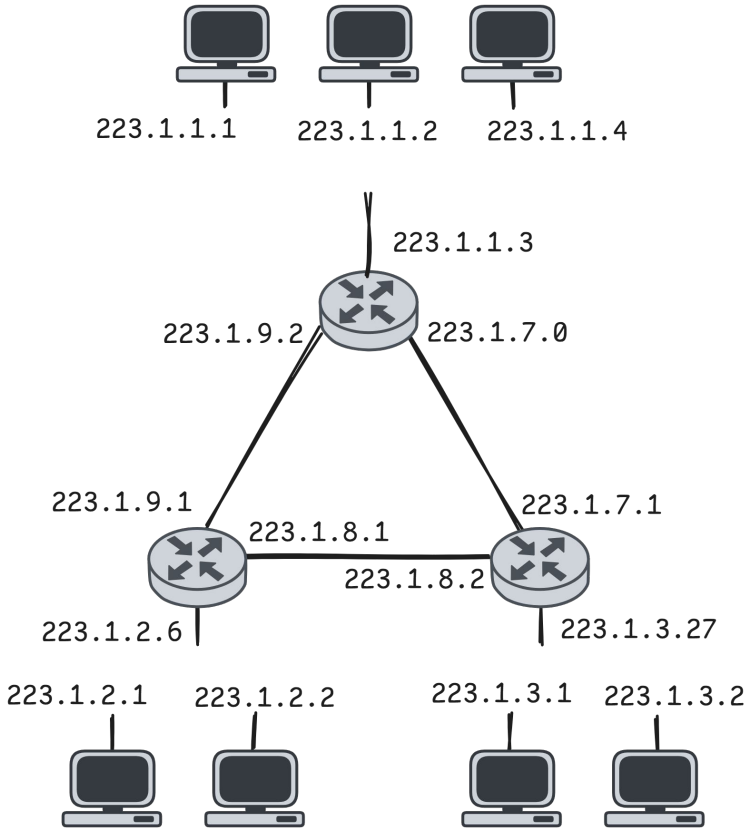
These interfaces can physically reach each other without passing through an intervening router

- inside the subnet communication happens via Link Layer



Subnets

How many subnets are here?

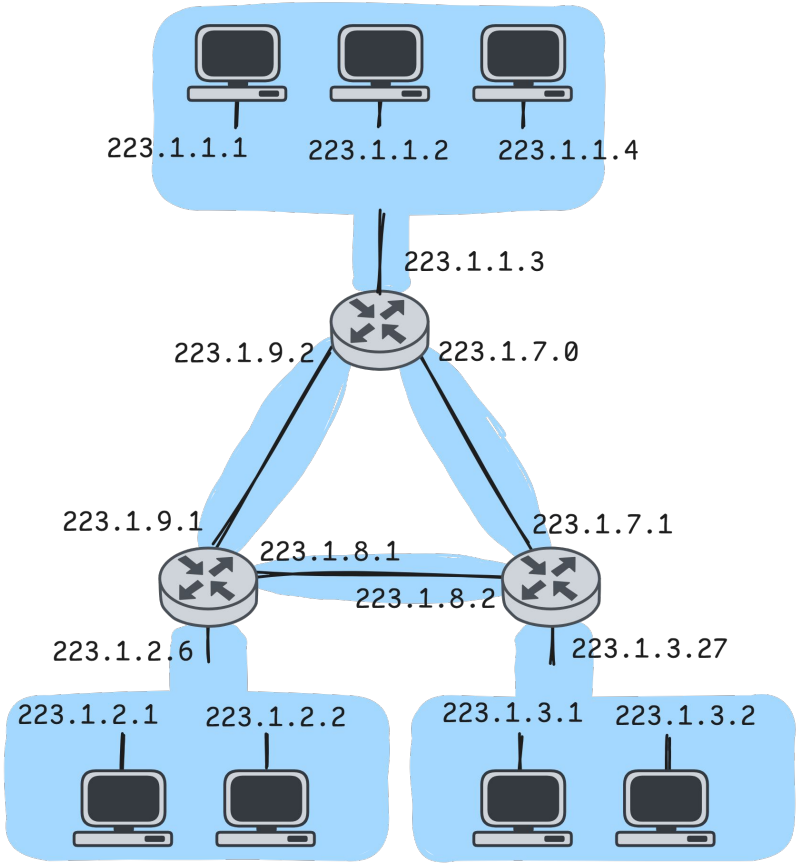


Subnets

How many subnets are here?

There are **6 subnets!**

Besides the 3 subnets that contain hosts, there are the subnets that consist of the interfaces of the links between 2 routers



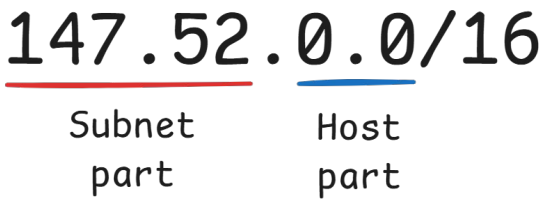
Prefixes (Continued)

There are some special IPs:

- **Network address:** `147.52.0.0`
  - the first IP of the network
  - all host part bits are **0**
  - identifies the network
- **Broadcast address:** `147.52.255.255`
  - the last IP of the network
  - all host part bits are **1**
  - sends the message to every device in the network

These IPs cannot be assigned to hosts

- Available IPs: $2^{32-16} - 2 = 2^{16} - 2$



Subnet Mask

There is a different way to present a subnet mask besides /x.

- A continuous **sequence of 1s** followed by 0s.
- The number of 1s defines the subnet part.

11111111.11111111.1100000.00000000

We have 18 sequential 1s so a **/18** mask.

In decimal format:

255.255.192.0

Other Examples:

- /16 == 255.255.0.0
- /24 == 255.255.255.0
- /23 == 255.255.254.0
- /18 == 255.255.192.0

Subnet Mask

Given the IP **`130.5.5.25`** and the mask **`/24`** we can find the prefix:

```
IP: 130.5.5.25          10000010.00000101.00000101.00011001
Mask: 255.255.255.0     11111111.11111111.11111111.00000000   AND
Prefix: 130.5.5.0       10000010.00000101.00000101.00000000
```

```
IP/Mask: 130.5.5.25/24  10000010.00000101.00000101.00011001
Prefix: 130.5.5.0       10000010.00000101.00000101.00000000
```

So we can see that the 2 notations have exactly the same meaning.

Subnetting

Many organizations divide their networks to smaller networks to:

- Facilitate the management
- Enhance the security by reducing the broadcast domains

Subnetting

- Network with an IP prefix `223.1.17.0/24`
- Separate the network into two subnets

Solution:

Decimal IP	223	1	17	0
Binary IP	11011111	00000001	0010001	00000000

Subnetting

- We **borrow a bit from the host part** and make it part of the subnet part
- With **n** borrowed bits we can create 2^n subnets

Original
223.1.17.0/**24**

Decimal IP	223	1	17	0
Binary IP	11011111	00000001	0010001	xxxxxxxx

Subnetting

- We **borrow a bit from the host part** and make it part of the subnet part
- With **n** borrowed bits we can create 2^n subnets

Original
223.1.17.0/**24**

Decimal IP	223	1	17	0
Binary IP	11011111	00000001	0010001	xxxxxxxx

Subnet 1
223.1.17.0/**25**

Decimal IP	223	1	17	0
Binary IP	11011111	00000001	0010001	0xxxxxxx

Subnet 2
223.1.17.128/**25**

Decimal IP	223	1	17	128
Binary IP	11011111	00000001	0010001	1xxxxxxx

Subnetting

Subnet 1

Prefix	`223.1.17.0/25`
Available IPs for hosts	2^{32-25} - 2 = 126
Network Address	`223.1.17.0`
Broadcast Address	`223.1.17.127`

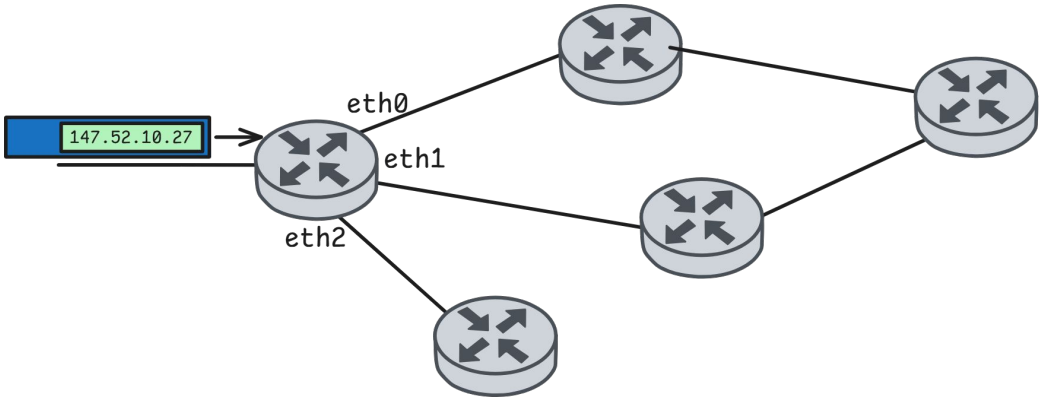
Subnet 2

Prefix	`223.1.17.128/25`
Available IPs for hosts	2^{32-25} - 2 = 126
Network Address	`223.1.17.128`
Broadcast Address	`223.1.17.255`

Forwarding Table

- Each router has a forwarding table
- The entries contain the **prefixes** with the corresponding **output interface**
- The table is filled by **routing algorithms**

Network Destination	Output Link
147.52.0.0/16	eth0
223.1.17.0/25	eth2
223.1.17.128/25	eth1
0.0.0.0/0	eth1



Forwarding Table

Assume this forwarding table of a router:

Network Destination	Output Link
200.23.16.0/21	eth0
200.23.24.0/24	eth1
200.23.24.0/21	eth2
0.0.0.0/0	eth2

Problem

From which interface will the router forward a packet with destination IP:

A. 200.23.17.154

B. 200.23.14.170

Forwarding Table

We convert the IP addresses in their binary form:

- `200.23.17.154` → `11001000 00010111 00010001 10011010`
- `200.23.24.170` → `11001000 00010111 00011000 10101010`

Network Destination	Output Link
`11001000 00010111 00010*** ********`	`eth0`
`11001000 00010111 00011000 ********`	`eth1`
`11001000 00010111 00011*** ********`	`eth2`
`******** ******** ******** ********`	`eth2`

Forwarding Table

- 200.23.17.154 → 11001000 00010111 00010001 10011010

Network Destination	Output Link
11001000 00010111 00010*** ********	eth0
11001000 00010111 00011000 ********	eth1
11001000 00010111 00011*** ********	eth2
******** ******** ******** ********	eth2

Forwarding Table

- 200.23.17.154 → 11001000 00010111 00010001 10011010
- So interface **eth0**

Network Destination	Output Link
11001000 00010111 00010*** ********	eth0
11001000 00010111 00011000 ********	eth1
11001000 00010111 00011*** ********	eth2
******** ******** ******** ********	eth2

Forwarding Table

- 200.23.24.170 → 11001000 00010111 00011000 10101010

Network Destination	Output Link
11001000 00010111 00010*** ********	eth0
11001000 00010111 00011000 ********	eth1
11001000 00010111 00011*** ********	eth2
******** ******** ******** ********	eth2

Forwarding Table

- 200.23.24.170 → 11001000 00010111 00011000 10101010
- IP matches to more than one prefix. What happens now? 🤔

Network Destination	Output Link
11001000 00010111 00010*** ********	eth0
11001000 00010111 00011000 ********	eth1
11001000 00010111 00011*** ********	eth2
******** ******** ******** ********	eth2

Forwarding Table

The forwarding table follows the rule of **Longest Prefix Match**

- When multiple prefixes match with the destination IP
- the one with the **most matching bits** will be chosen (the longest prefix)

Forwarding Table

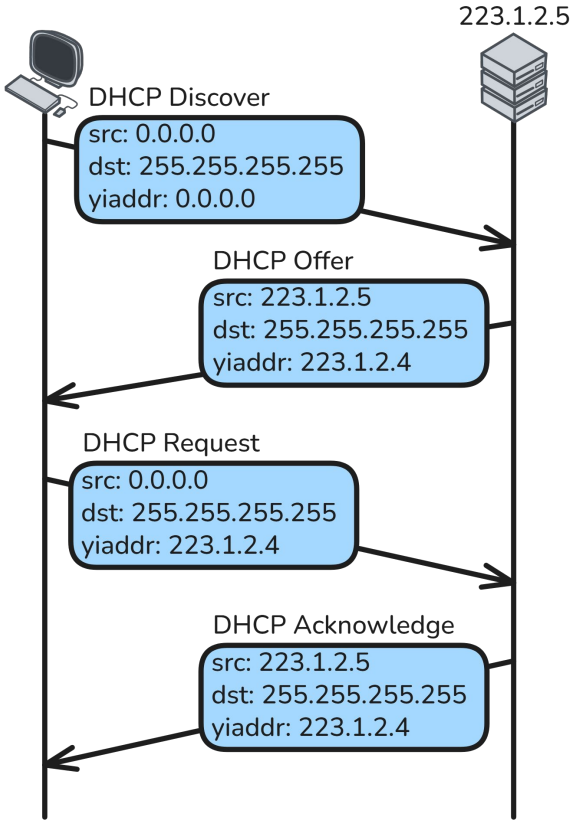
- `200.23.24.170` → `11001000 00010111 00011000 10101010`
- IP matches to more than one prefix. What happens now? 🤔
- **`eth1`** will be used because of **Longest Prefix Match**

Network Destination	Output Link
`11001000 00010111 00010*** ********`	`eth0`
`11001000 00010111 00011000 ********`	`eth1`
`11001000 00010111 00011*** ********`	`eth2`
`******** ******** ******** ********`	`eth2`

DHCP (Dynamic Host Configuration Protocol)

- Allows a host to **dynamically obtain an IP** address when it joins the network
- There is a **DHCP server** in each network
  - has an address pool
  - **<IP, lease time>** entries
  - is the network router in many cases
- The client requests an IP from the DHCP server

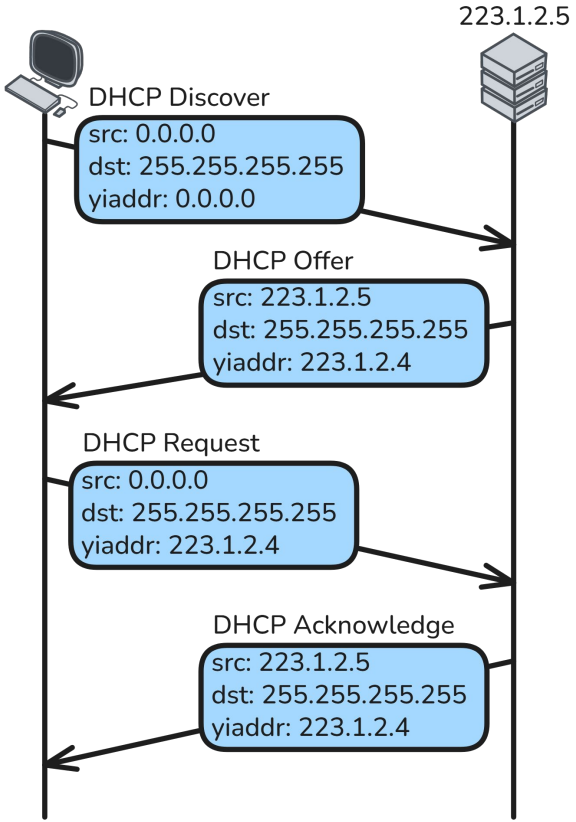
DHCP (Dynamic Host Configuration Protocol)



DHCP (Dynamic Host Configuration Protocol)

DORA:

- **D**iscover
- **O**ffer
- **R**equest
- **A**cknowledge



DHCP (Dynamic Host Configuration Protocol)

When the lease time expires the client:

- can renew the IP
- get a new IP

Besides the IP the DHCP provides info about:

- **Lease Time**
- **Subnet Mask**
- **Default Gateway**
  - The router’s IP that the client should send traffic to outside its local network
- **Local DNS Server**

NAT (Network Address Translation)

- As the Internet grew…
- …so did the demand for IPs
- IPv4s ~= 2^{32} = 4.3 billion addresses
- As you can imagine there are not enough

- Solution: NAT
  - Long time solution: IPv6 ~= 340,282,366,920,938,463,463,374,607,431,768,211,456 IPs

NAT (Network Address Translation)

Private IP ranges:

- `10.0.0.0/8`
- `172.16.0.0/12`
- `192.168.0.0/16`

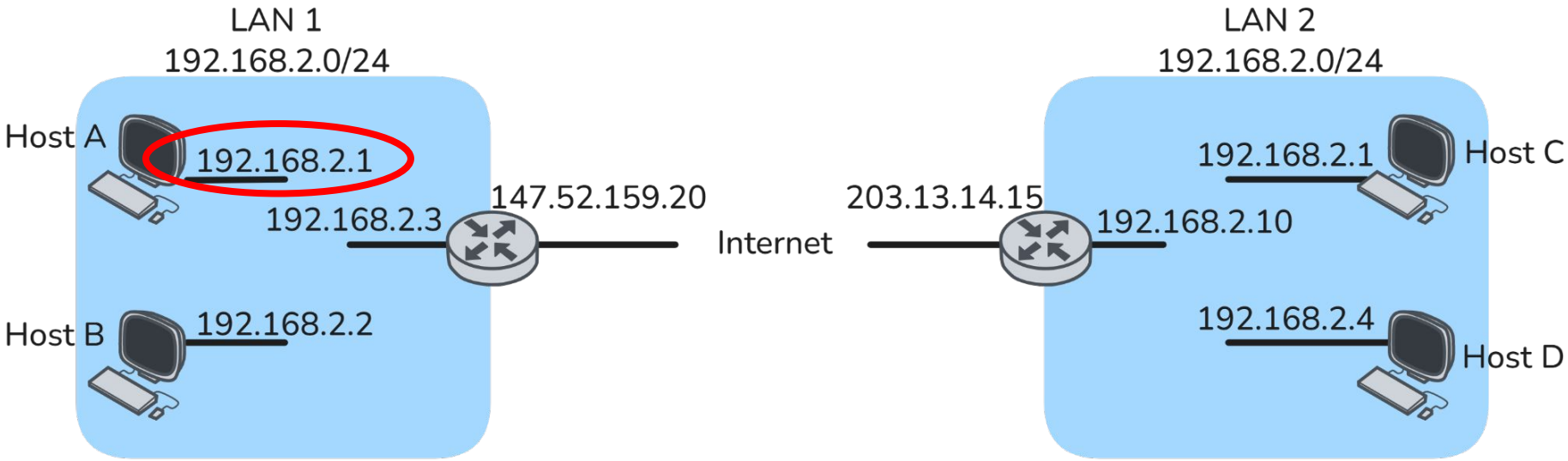
These IPs are not routable:

- Only for communication **inside** a network (private network)
- Routers will **not forward** them **outside the network** (to the Internet)

When we want to send traffic outside the network:

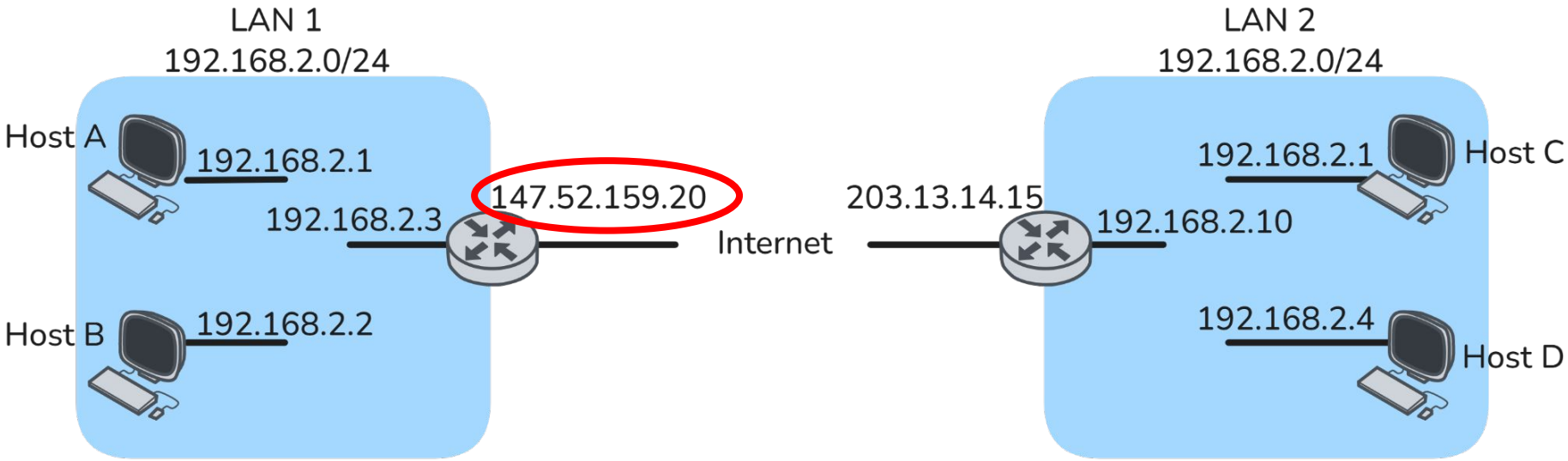
- NAT **translates** the **private IP** to a **public IP**

NAT (Network Address Translation)



- **Host A and B** will communicate using **private** addresses
  - Host B will see 192.168.2.1 as src IP for packet send from host A

NAT (Network Address Translation)



- **Host A and C** will communicate using **public** addresses
  - Host C will see 147.52.159.20 as src IP for packet send from host A

NAT (Network Address Translation)

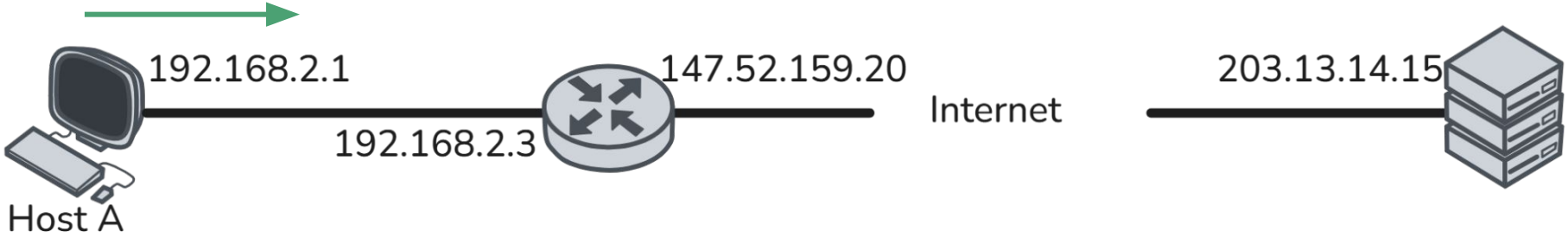
But how does this translation happen?

- NAT maps the source's **(private IP, port)** to **(public IP, new port)**
- Through the ports we can separate which host send which packet
- **Translation Table:**

Private IP, port	Public IP, port
192.168.2.1, 5800	147.52.159.20, 18000
192.168.2.2, 6000	147.52.159.20, 12036

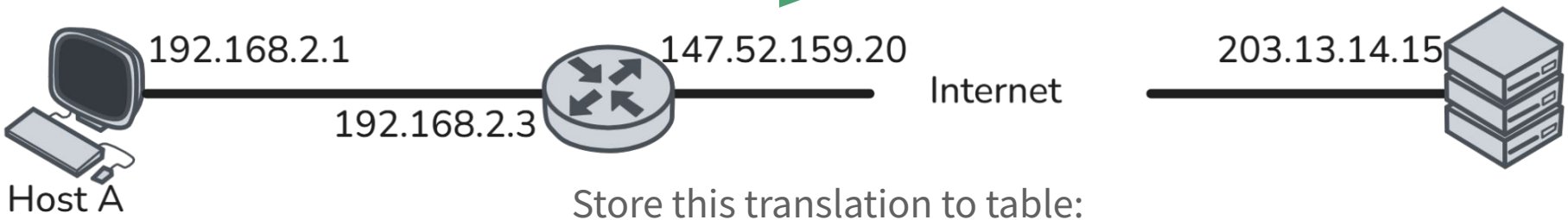
NAT (Network Address Translation)

src IP	192.168.2.1
src port	5800
dst IP	203.13.14.15
dst port	80



NAT (Network Address Translation)

src IP	147.52.159.20
src port	18000
dst IP	203.13.14.15
dst port	80

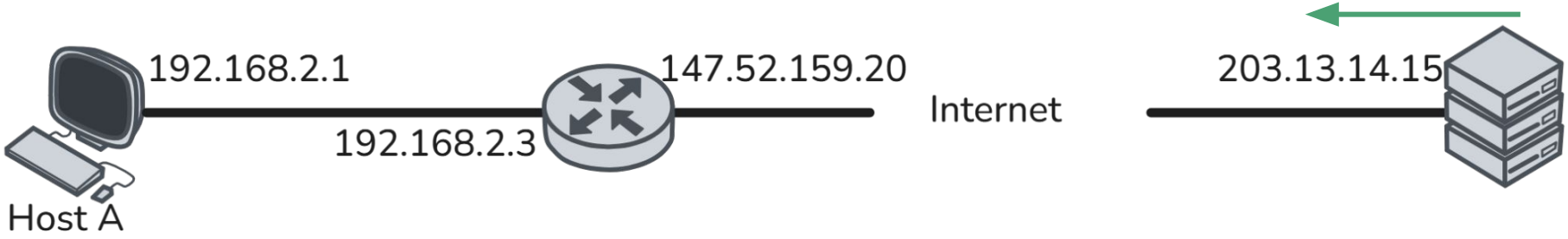


Store this translation to table:

(192.168.2.1, 5800) → (147.52.159.20, 18000)

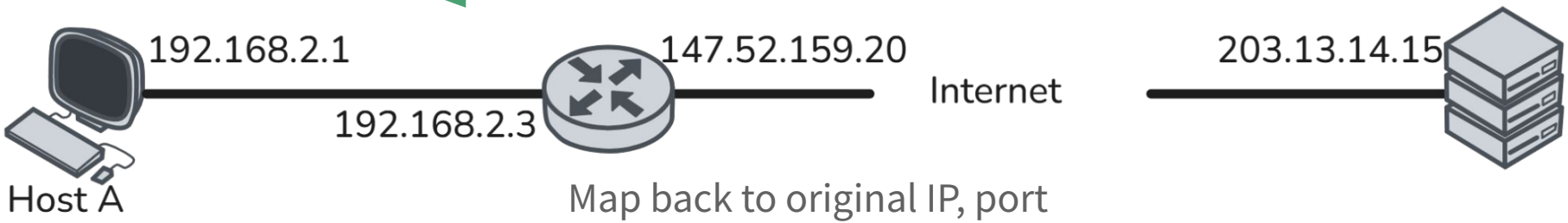
NAT (Network Address Translation)

src IP	203.13.14.15
src port	80
dst IP	147.52.159.20
dst port	18000



NAT (Network Address Translation)

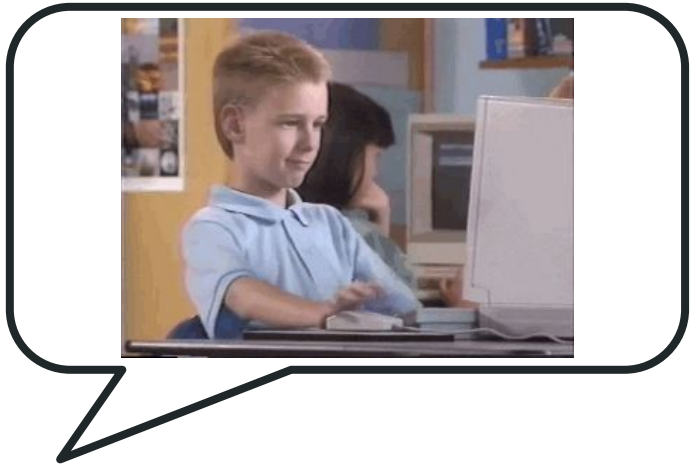
src IP	203.13.14.15
src port	80
dst IP	192.168.2.1
dst port	5800

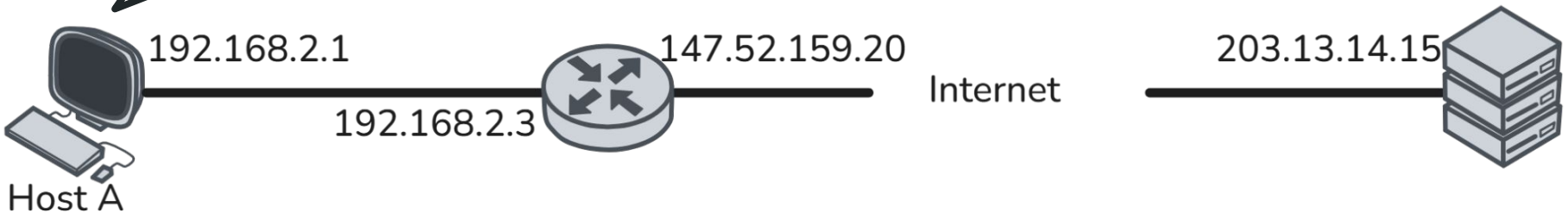


Map back to original IP, port

(147.52.159.20, 18000) → (192.168.2.1, 5800)

NAT (Network Address Translation)





NAT (Network Address Translation)

What do we gain?

1. **Conserves** public **IPv4 addresses**
    a. Multiple devices share one public IP.
    b. Multiple networks can use the same private ranges (two separate organizations can both use 192.168.x.x internally without conflict)
    c. This greatly reduces the number of public IPs a network needs.
2. Increases network **security** (basic privacy)
    a. Internal devices are not directly exposed to the Internet.
    b. Outside hosts cannot directly initiate connections to private IPs.
3. Allow easy network renumbering
    a. We can change internal private IP ranges without affecting the public-facing address.

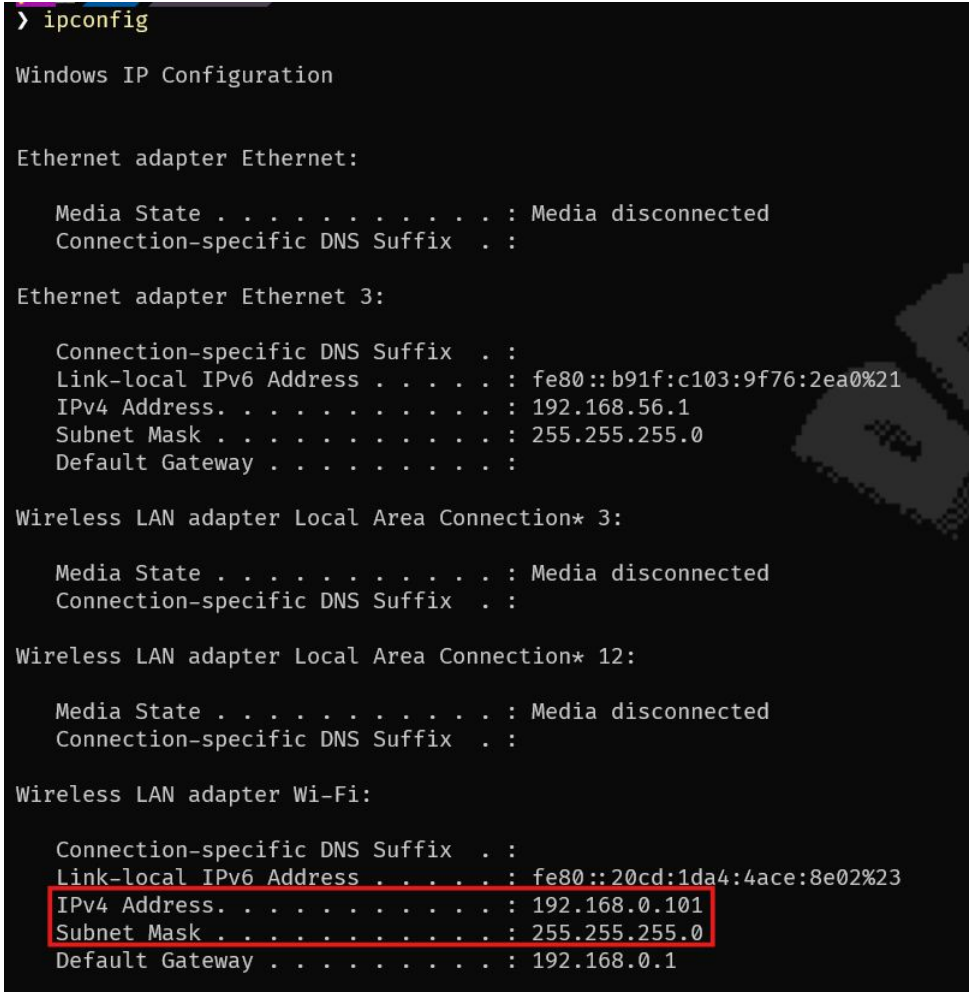
NAT (Network Address Translation)

What are the disadvantages?

1. **Breaks end-to-end** connectivity
    a. changes the IP
    b. another internet host will not be able to identify the specific host that sent the packet
2. Adds **processing overhead**
3. Complicates protocols that **embed IP** addresses
4. Makes **inbound connections** harder
    a. External clients cannot reach internal devices without extra configuration
5. Difficulties for **logging/filtering**
    a. track individual hosts
    b. apply IP-based access rules

Commands to show IP

- Windows: **`ipconfig`**
  - `ipconfig /all` for more info



Commands to show IP

- Linux/Mac: **`ifconfig`**