CSCI 466: Networks

Subnets, NAT, IPv6, SDN

Reese Pearsall

Fall 2023

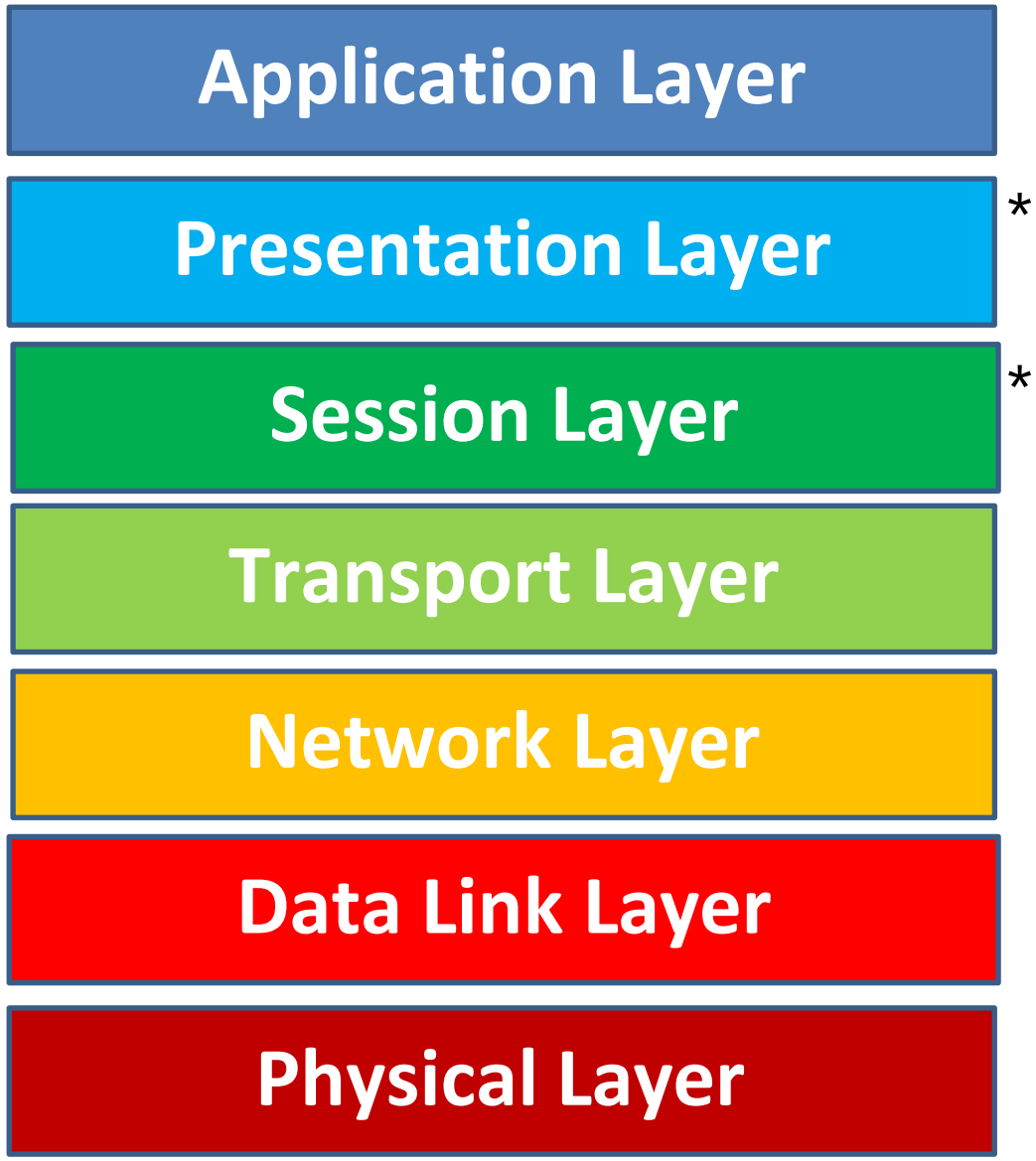
Announcements

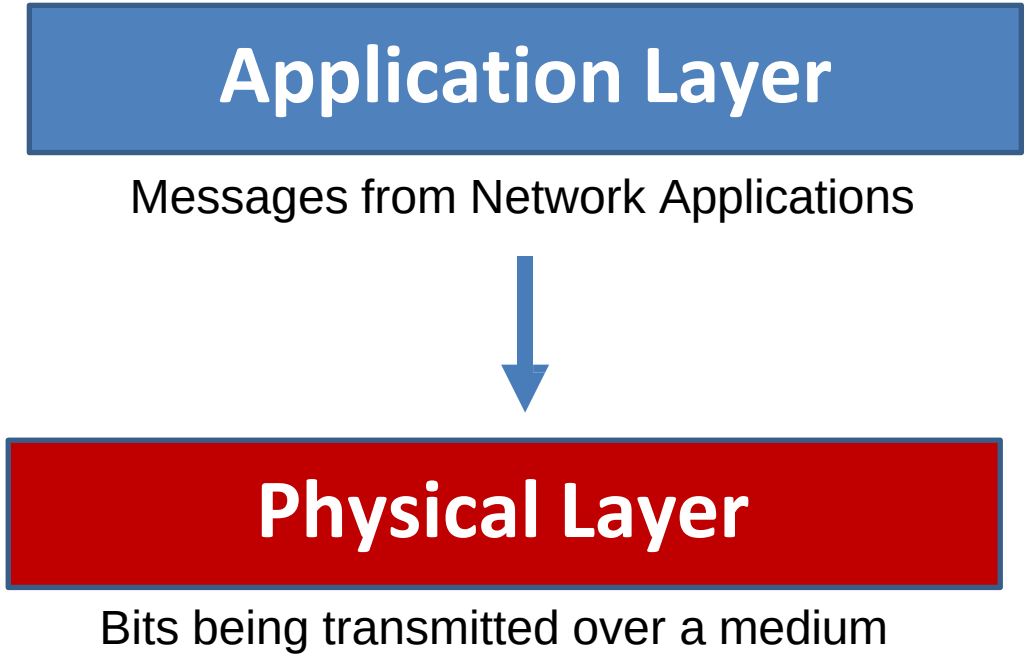
PA2 due **Friday** (10/20)

Final Ack cannot be corrupted

Friday will be a work day for PA2 (no lecture)

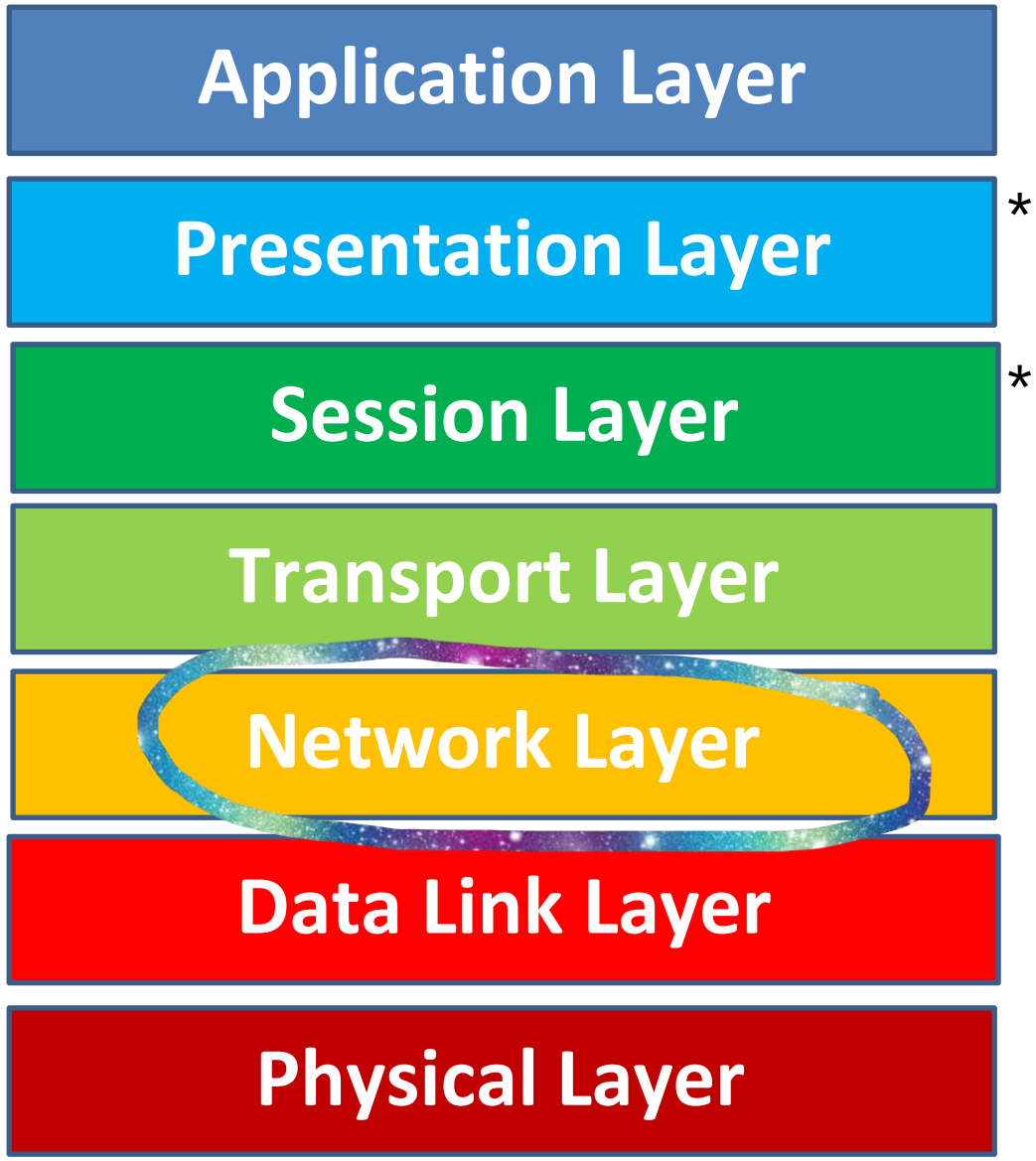
OSI Model





In the textbook, they condense it to a 5-layer model, but 7 layers is what is most used

OSI Model



Application Layer

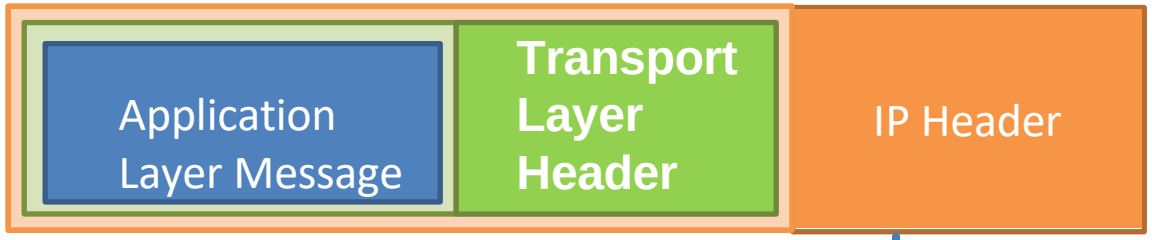
Messages from Network Applications

Physical Layer

Bits being transmitted over a medium

**In the textbook, they condense it to a 5-layer model, but 7 layers is what is most used*

Packets traversing through the network layer are referred to as a **datagram**. Each packet gets an IPv4/IPv6 header



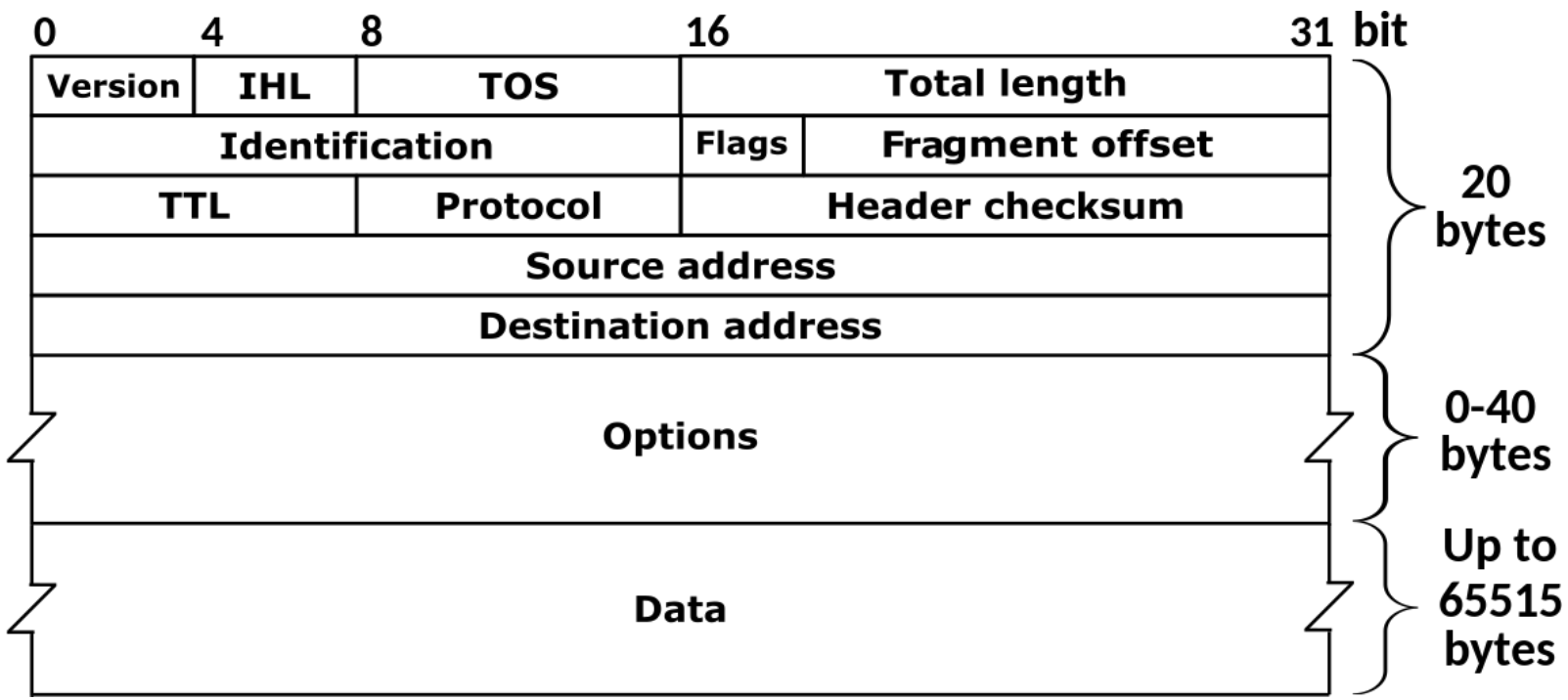
IPv4: 32-bit addresses (decimal)

192.149.252.76

IPv6: 64-bit addresses (hexademical)

3ffe:1900:fe21:4545::

Why do we need IPv6?



IPv4 Header

IP Address: Globally unique* 32 bit (4 byte) **dotted decimal** number assigned to interfaces on hosts and routers

Used to identify a host on a network

(1 byte = 8 bits)

193.32.216.9

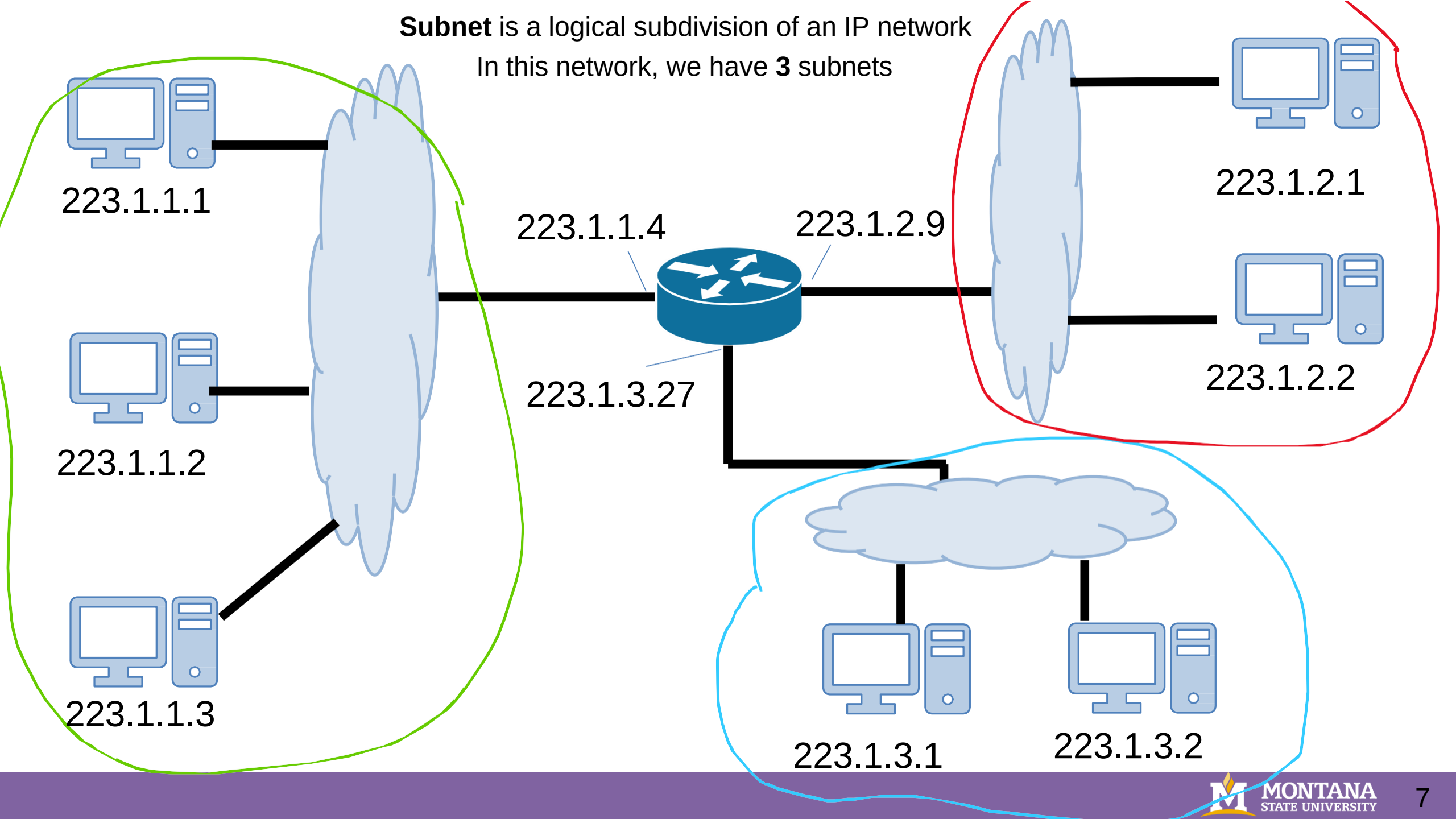
=

11000001 00100000 11011000 00001001

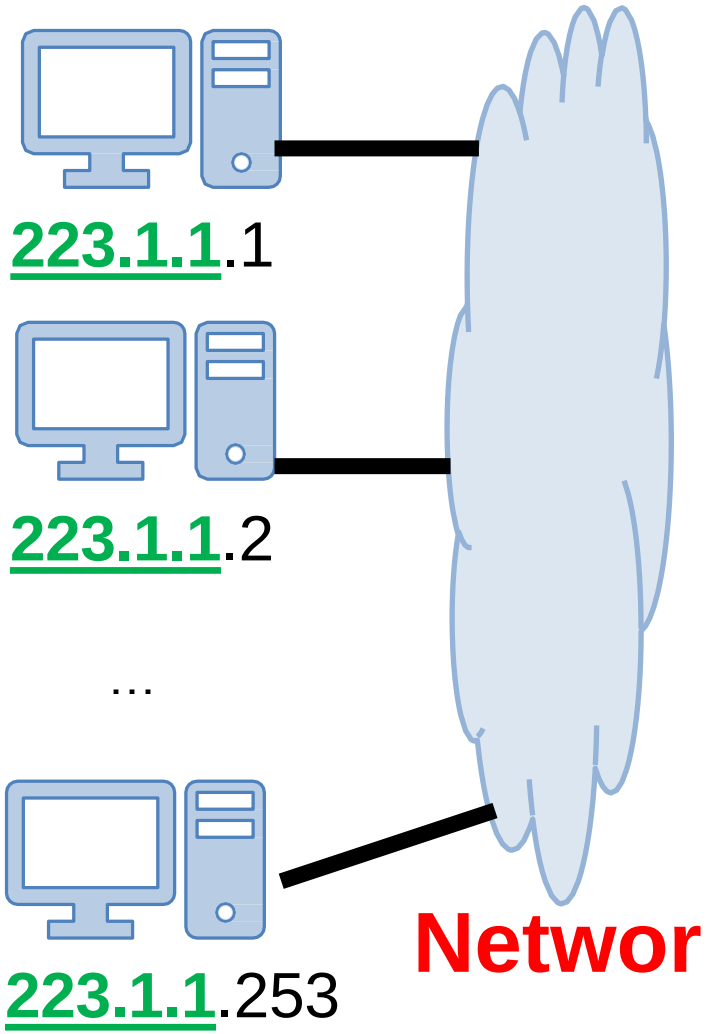
How many addresses are possible with a 32 bit number? **~4 billion possible IPv4 addresses**

Subnet is a logical subdivision of an IP network

In this network, we have **3** subnets



It is very common to have a **range** of IP addresses assigned to you (random assignment would be chaos)



Subnet mask

223.1.1.0/24

The leftmost 24 bits represent the prefix of the subnet

11111111 11111111 11111111 XXXXXXXX = **255.255.255**.0

223.1.1.67 ✓

223.2.1.67 ✗

223.2.1.255 ✗

Network bits **193.32.216.9** **Host bits**

11000001 00100000 11011000 00001001

A **subnet mask** can be used to reveal which bits are the host bits

A **subnet mask** can be used to reveal which bits are the host bits

172.16.1.0	10101100.00010000.00000001.00000000
255.255.0.0	11111111.11111111.00000000.00000000

A **subnet mask** can be used to reveal which bits are the host bits

172.16.1.0 10101100.00010000.00000001.00000000

255.255.0.0 11111111.11111111.00000000.00000000

A **subnet mask** can be used to reveal which bits are the host bits

172.16.1.0 10101100.00010000.**00000001.00000000**

255.255.0.0 11111111.11111111.00000000.00000000

The last two octets do not match 1s, so they are the **host bits**

A **subnet mask** can be used to reveal which bits are the host bits

172.16.1.0 10101100.00010000.**00000001.00000000**

255.255.0.0 11111111.11111111.00000000.00000000

The last two octets do not match 1s, so they are the **host bits**

This IP address would fall under the range of:

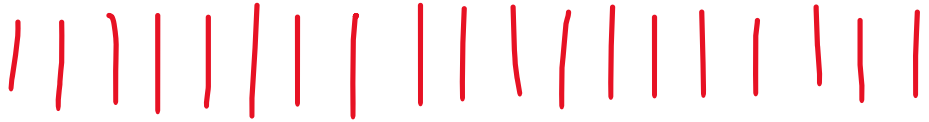
172.16.0.0 **/16**

A **subnet mask** can be used to reveal which bits are the host bits

172.16.1.0	10101100.00010000.00000001.00000000
255.255.224.0	11111111.11111111.11100000.00000000

A **subnet mask** can be used to reveal which bits are the host bits

172.16.1.0 10101100.00010000.000**00001.00000000**

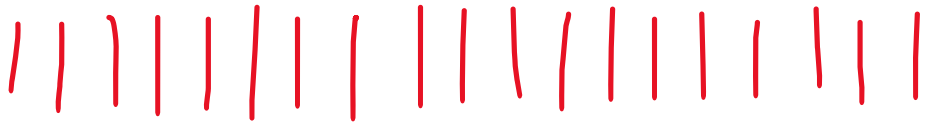


255.255.224.0 11111111.11111111.11100000.00000000

The last 13 bits are the host bits, the first 19 bits are the network bits

A **subnet mask** can be used to reveal which bits are the host bits

172.16.1.0 10101100.00010000.000**00001.00000000**



255.255.224.0 11111111.11111111.11100000.00000000

The last 13 bits are the host bits, the first 19 bits are the network bits

This IP address would fall under the range of

172.16.0.0/19

CIDR

A **subnet mask** can be used to reveal which bits are the host bits

172.16.1.0 10101100.00010000.000**00001.00000000**

255.255.224.0 11111111.11111111.11100000.00000000

The last 13 bits are the host bits, the first 19 bits are the network bits

This IP address would fall under the range of

172.16.0.0/19 CIDR

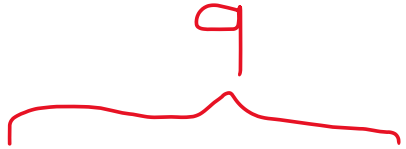
A **subnet mask** can be used to reveal which bits are the host bits

CIDR 172.0.0.0/9

Subnet Mask?

A **subnet mask** can be used to reveal which bits are the host bits

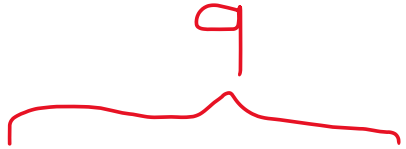
CIDR 172.0.0.0/9



Subnet Mask? 11111111.10000000.00000000.00000000

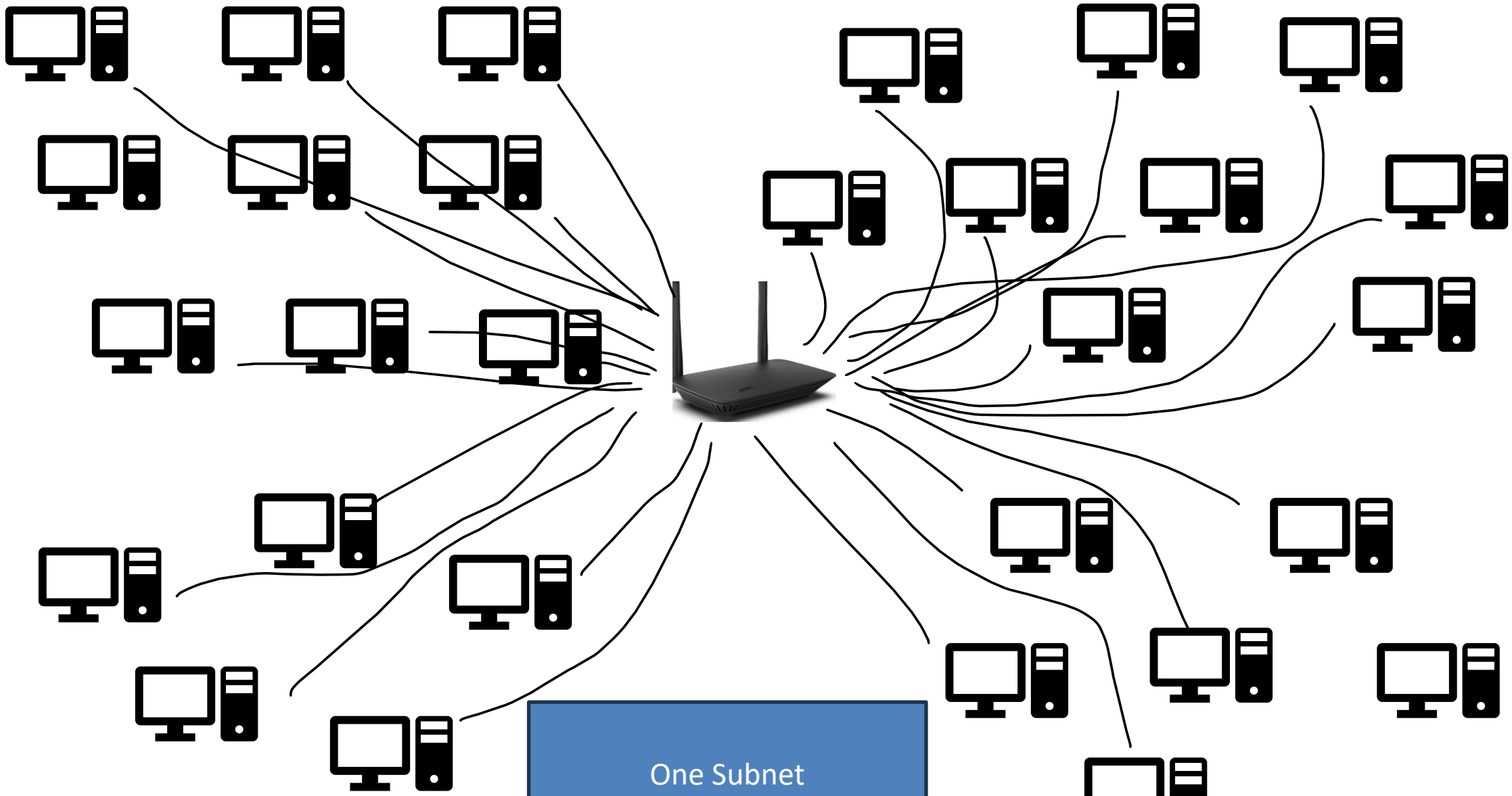
A **subnet mask** can be used to reveal which bits are the host bits

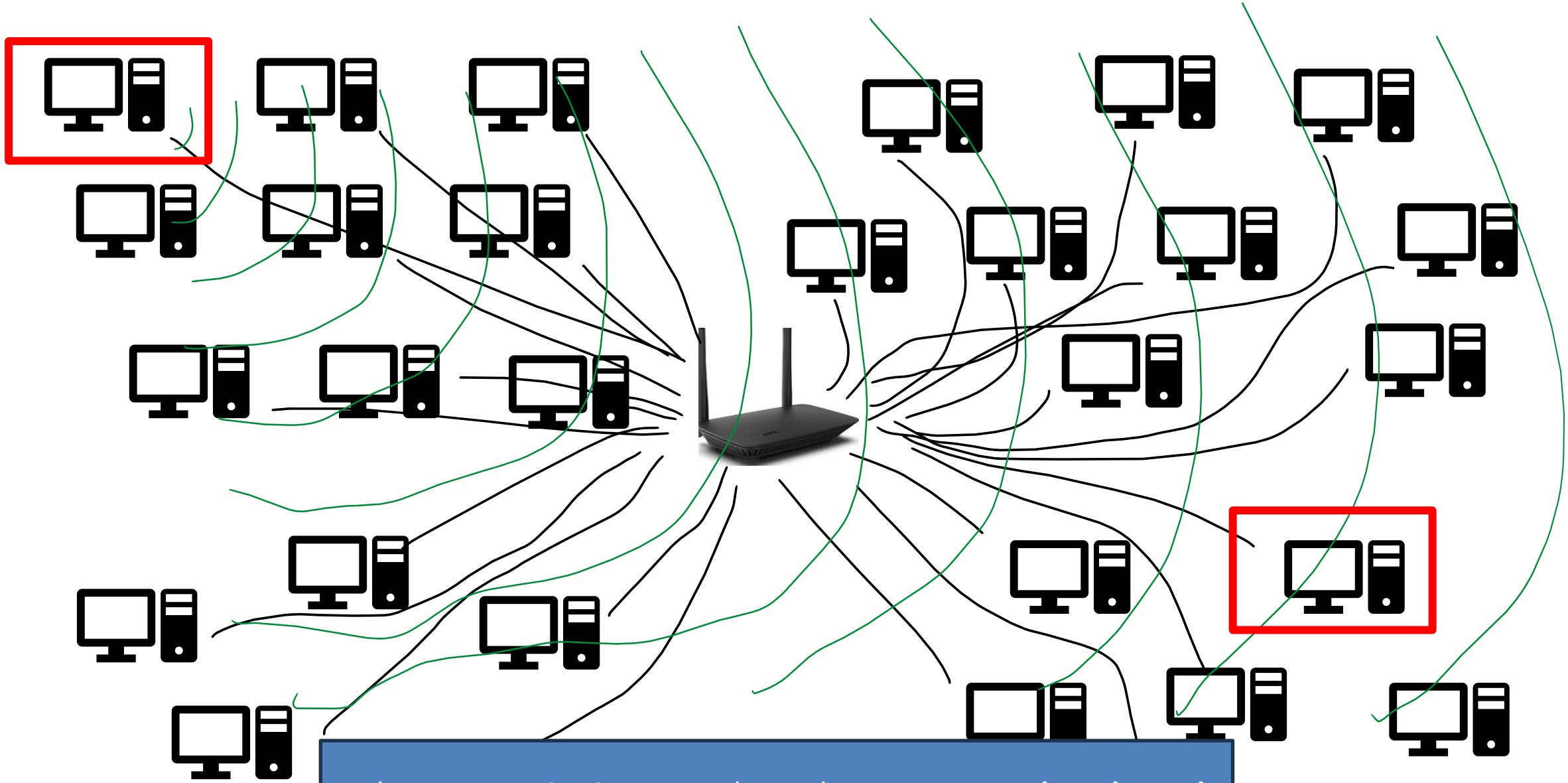
CIDR 172.0.0.0/9



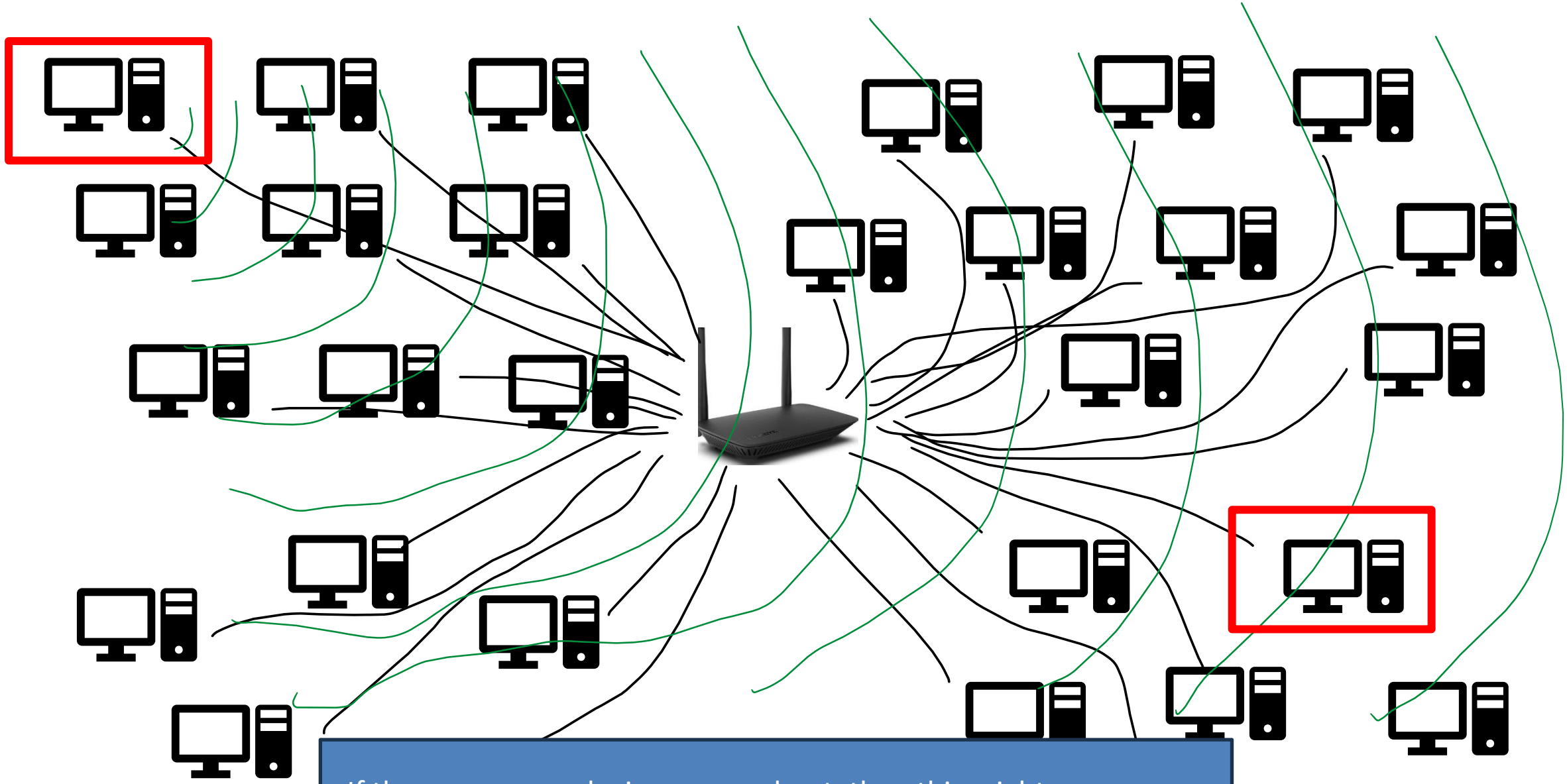
Subnet Mask? 11111111.10000000.00000000.00000000

255.128.0.0

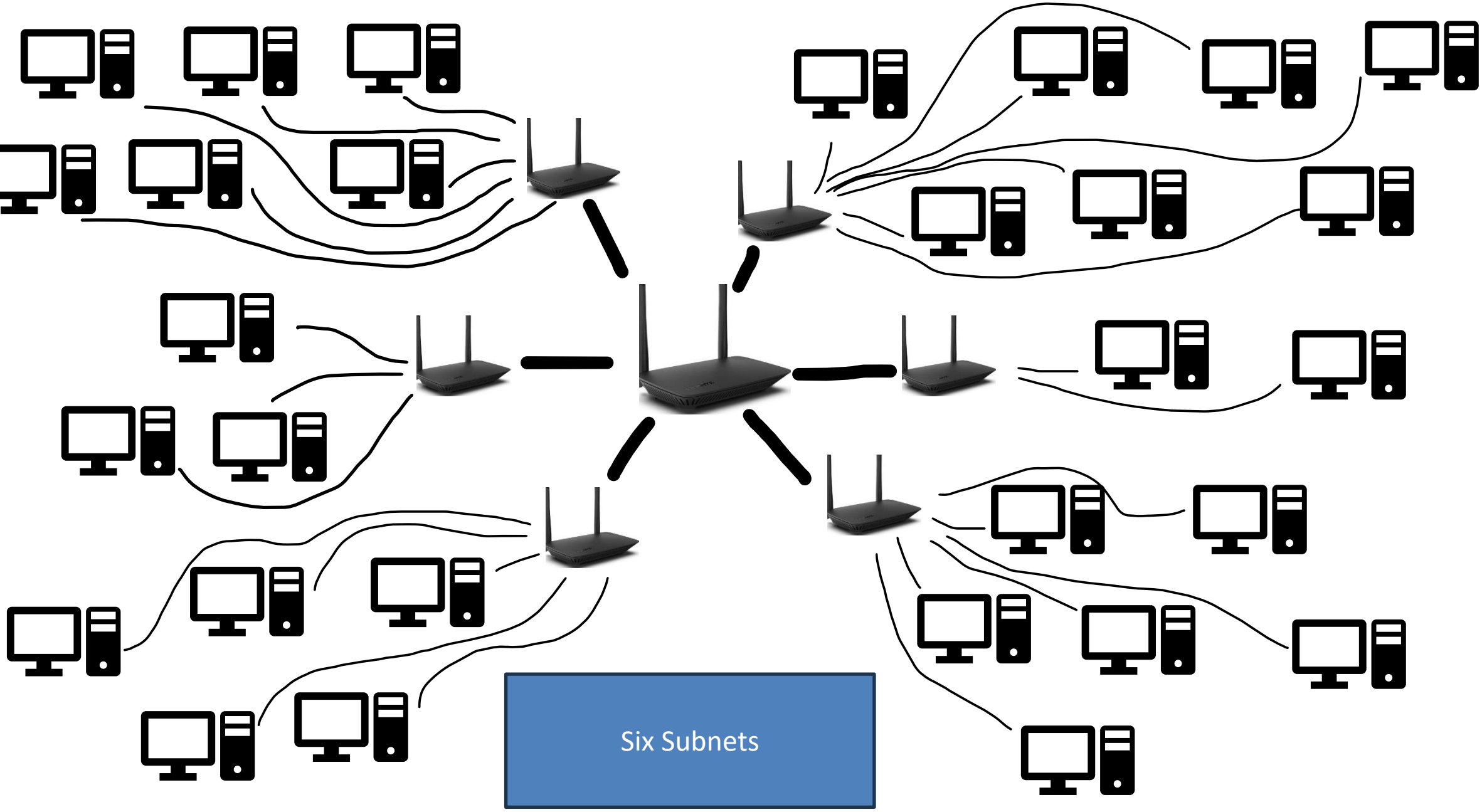

One Subnet

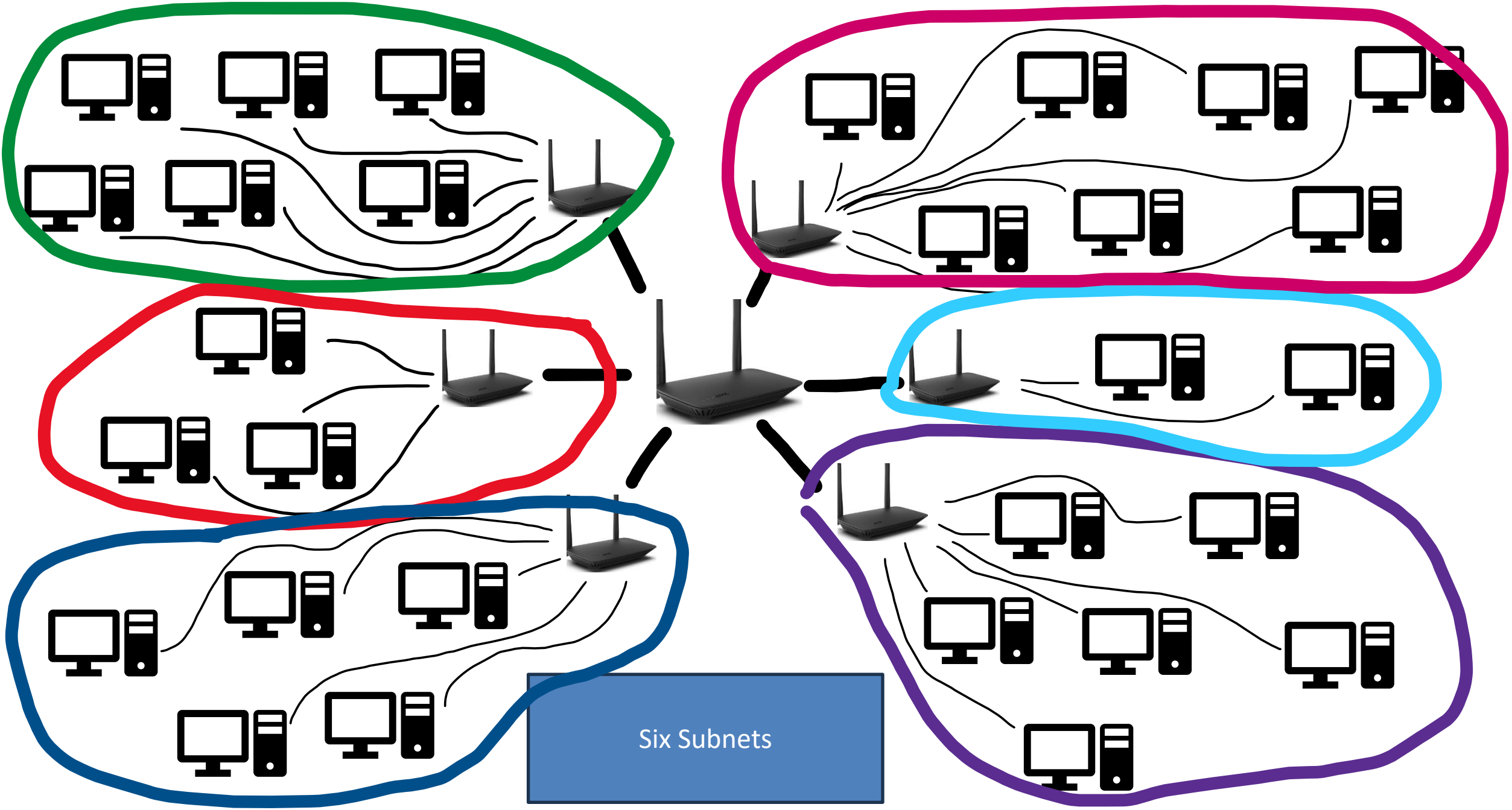


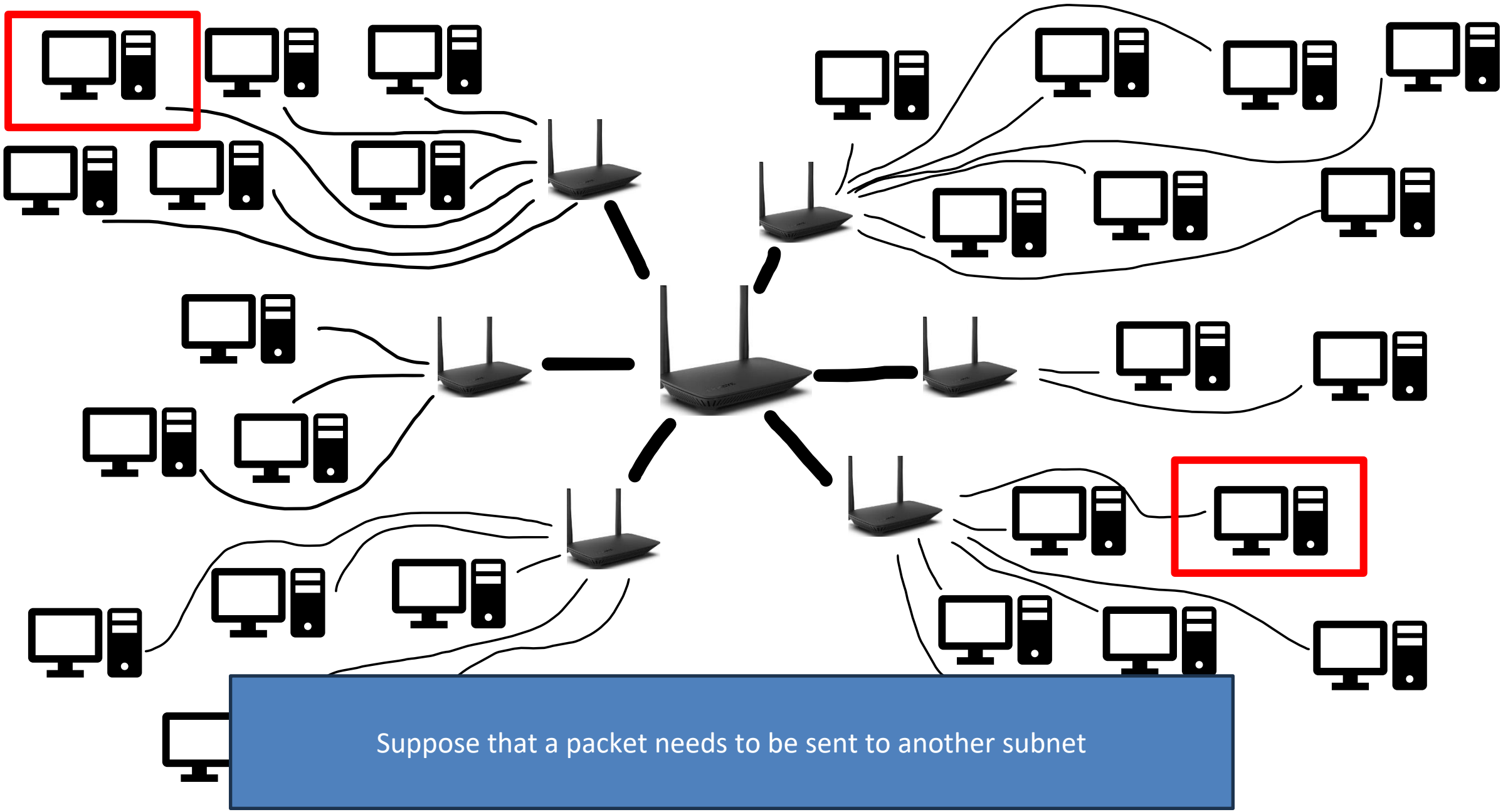
When communicating on a subnet, the message gets **broadcasted** to every device on the network

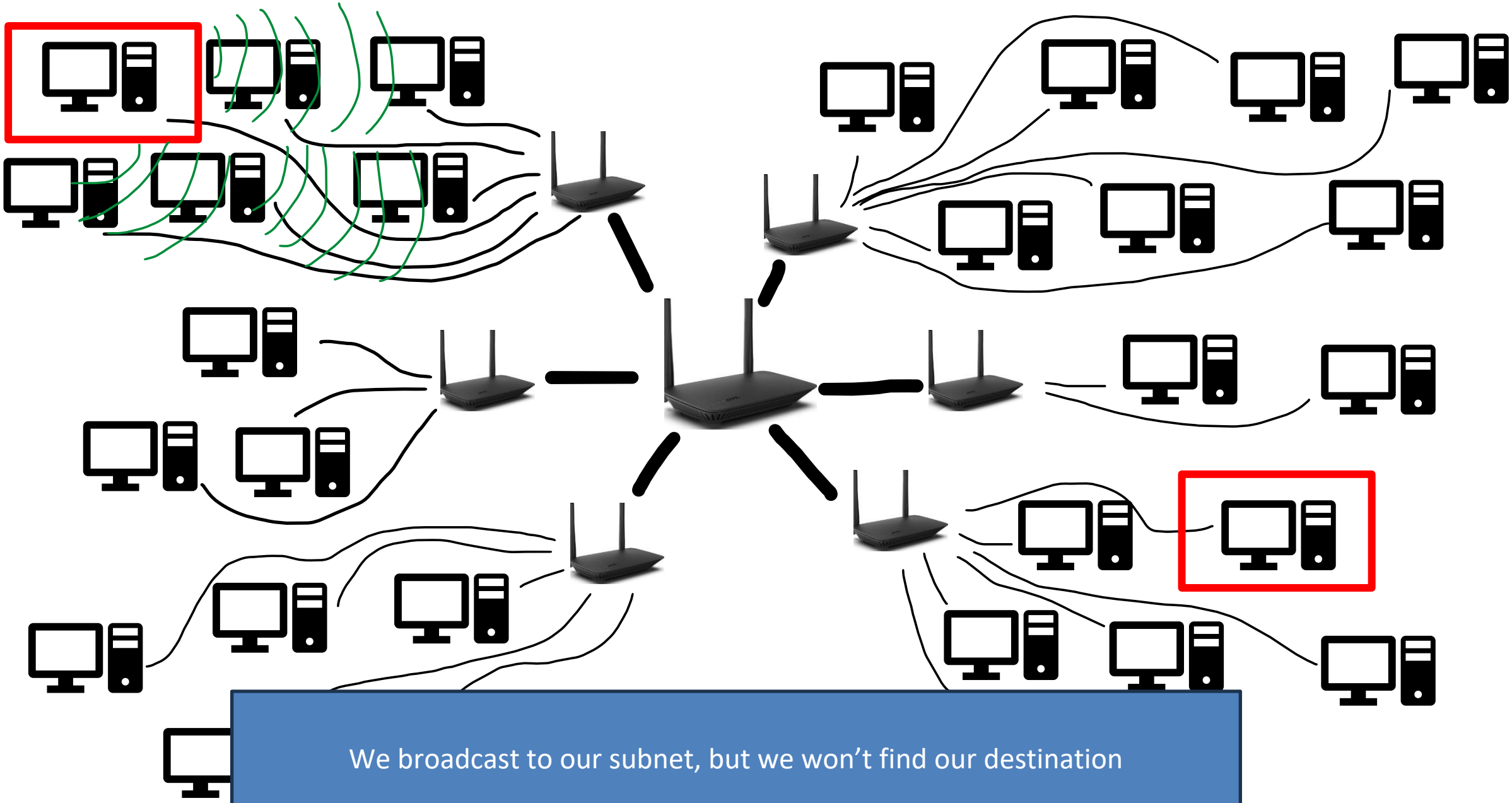


If there are many devices on a subnet, then this might cause some problems (congestion, lag, delay)

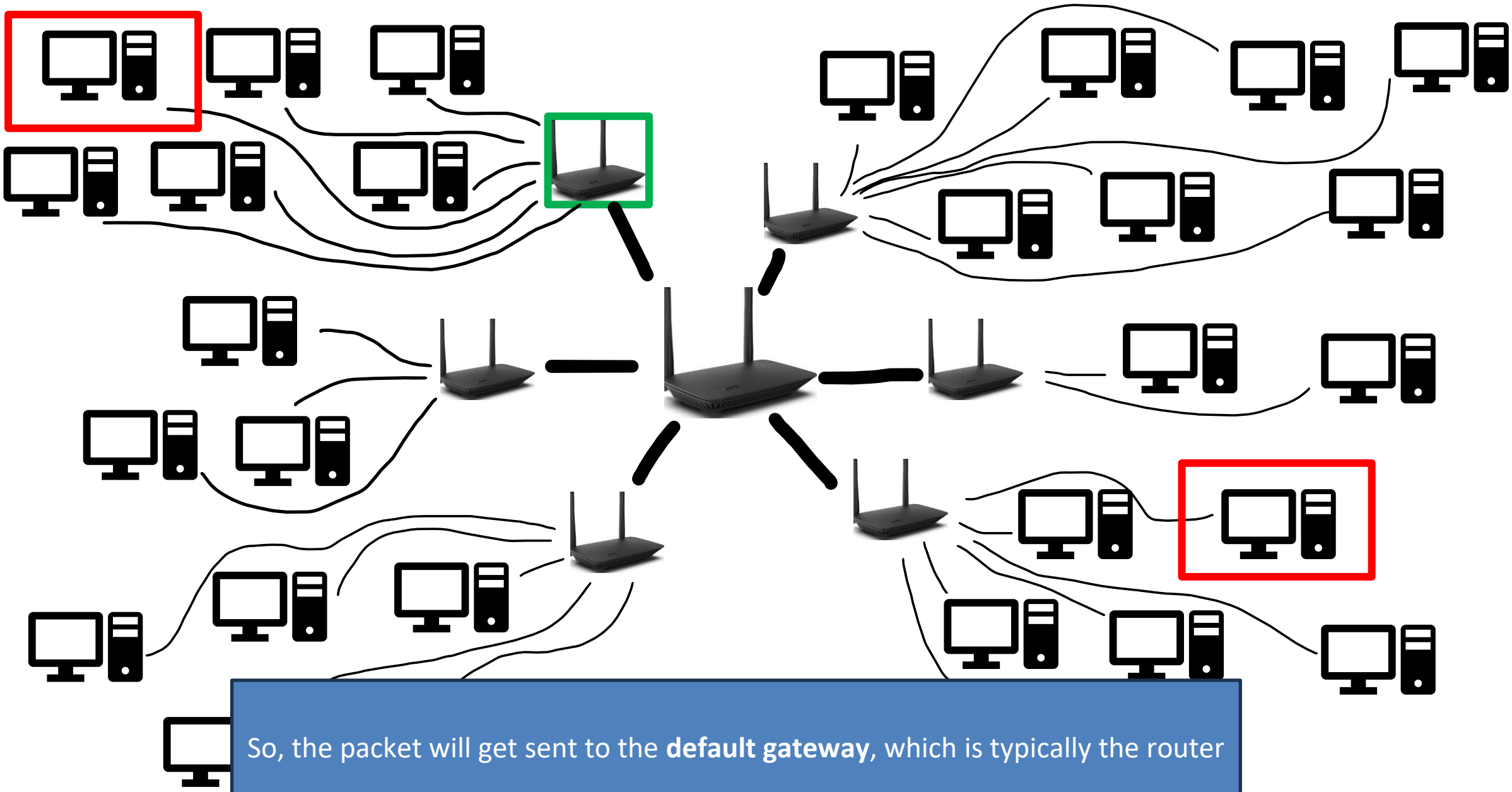

Six Subnets


Six Subnets

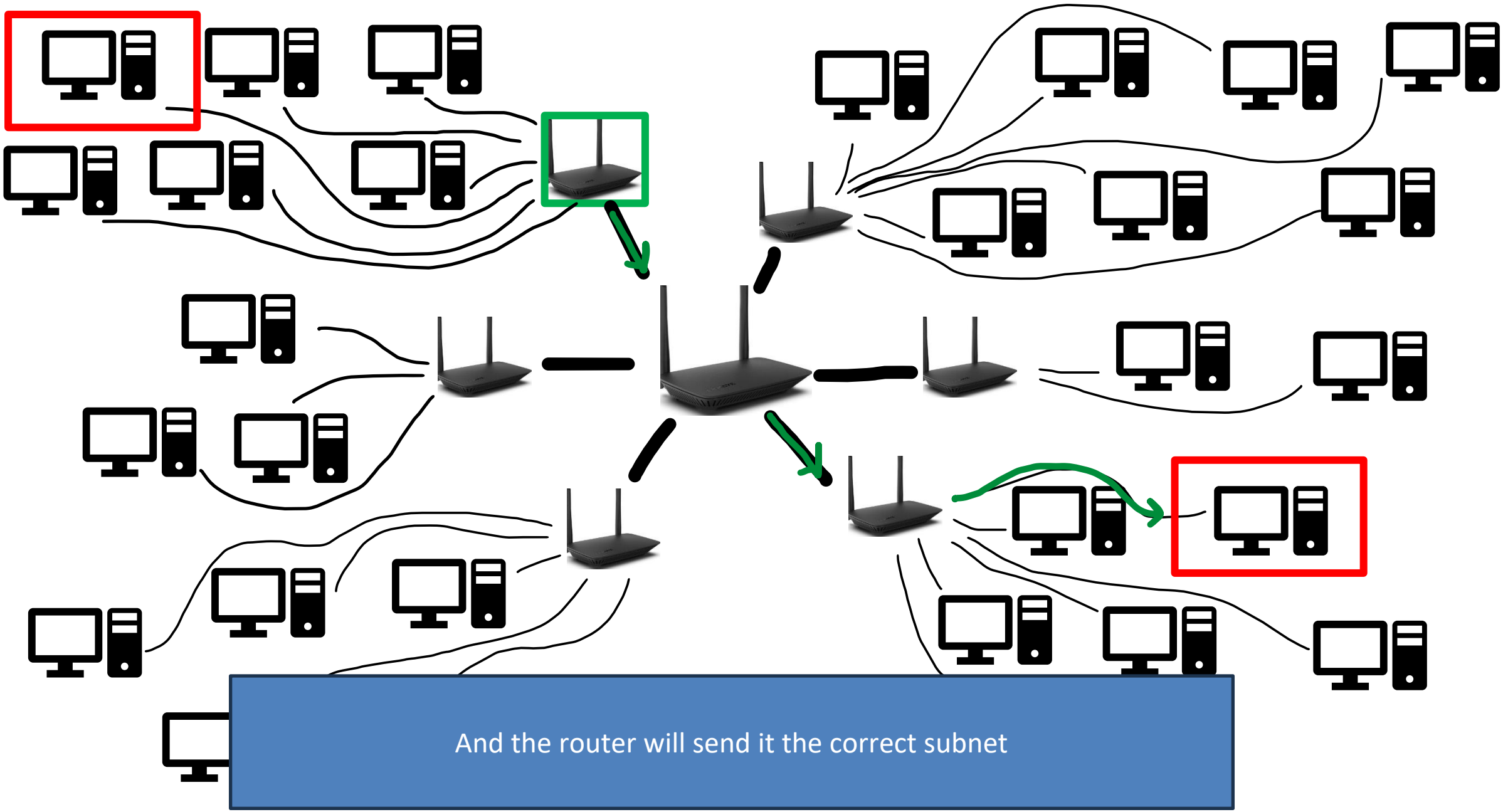

Suppose that a packet needs to be sent to another subnet



We broadcast to our subnet, but we won't find our destination



So, the packet will get sent to the **default gateway**, which is typically the router


And the router will send it the correct subnet

Subnets

Suppose we have a **class c** network

X.X.X.0/24 **Subnet Mask** 11111111.11111111.11111111.00000000

Subnets

Suppose we have a **class c** network

X.X.X.0/24 **Subnet Mask** 11111111.11111111.11111111.00000000

1 network with 253* hosts

What if we wanted to divide this network into multiple subnets?

Subnets

Suppose we have a **class c** network

X.X.X.0/24 **Subnet Mask** 11111111.11111111.11111111.00000000

1 network with 253* hosts

What if we wanted to divide this network into multiple subnets?

We sacrifice one of our host bits, and give it to the network bits

Subnets

Suppose we have a **class c** network

X.X.X.0/24 **Subnet Mask** 11111111.11111111.11111111.00000000

1 network with 253* hosts

What if we wanted to divide this network into multiple subnets?

We sacrifice one of our host bits, and give it to the network bits

Subnet 1: 11111111.11111111.11111111.00000000

Subnet 2: 11111111.11111111.11111111.10000000

2 Networks with 126 hosts

Subnets

Suppose we have a **class c** network

X.X.X.0/24 **Subnet Mask** 11111111.11111111.11111111.00000000

1 network with 253* hosts

What if we wanted to divide this network into multiple subnets?

We sacrifice one of our host bits, and give it to the network bits

Subnet 1: 11111111.11111111.11111111.00000000

Subnet 2: 11111111.11111111.11111111.10000000

Subnet 3: 11111111.11111111.11111111.01000000

Subnet 4: 11111111.11111111.11111111.11000000

4 Networks with 62 hosts

Subnets

Suppose we have a **class c** network

X.X.X.0/24 **Subnet Mask** 11111111.11111111.11111111.00000000

1 network with 253* hosts

*What if we wanted to divide this network into **four** subnets?*

We sacrifice one of our host bits, and give it to the network bits

Subnet 1: 11111111.11111111.11111111.00000000

Subnet 2: 11111111.11111111.11111111.10000000

Subnet 3: 11111111.11111111.11111111.01000000

Subnet 4: 11111111.11111111.11111111.11000000

4 Networks with 62 hosts

Subnets

Suppose we have a **class c** network

X.X.X.0/24 **Subnet Mask** 11111111.11111111.11111111.00000000

1 network with 253* hosts

*What if we wanted to divide this network into **four** subnets?*

We sacrifice one of our host bits, and give it to the network bits

New Subnet Mask 11111111.11111111.11111111.11000000

255.255.255.192

Issue: IPv4 can only support ~4 billion devices

If every device on the internet gets an IP address, we are going to run out of IP addresses to use

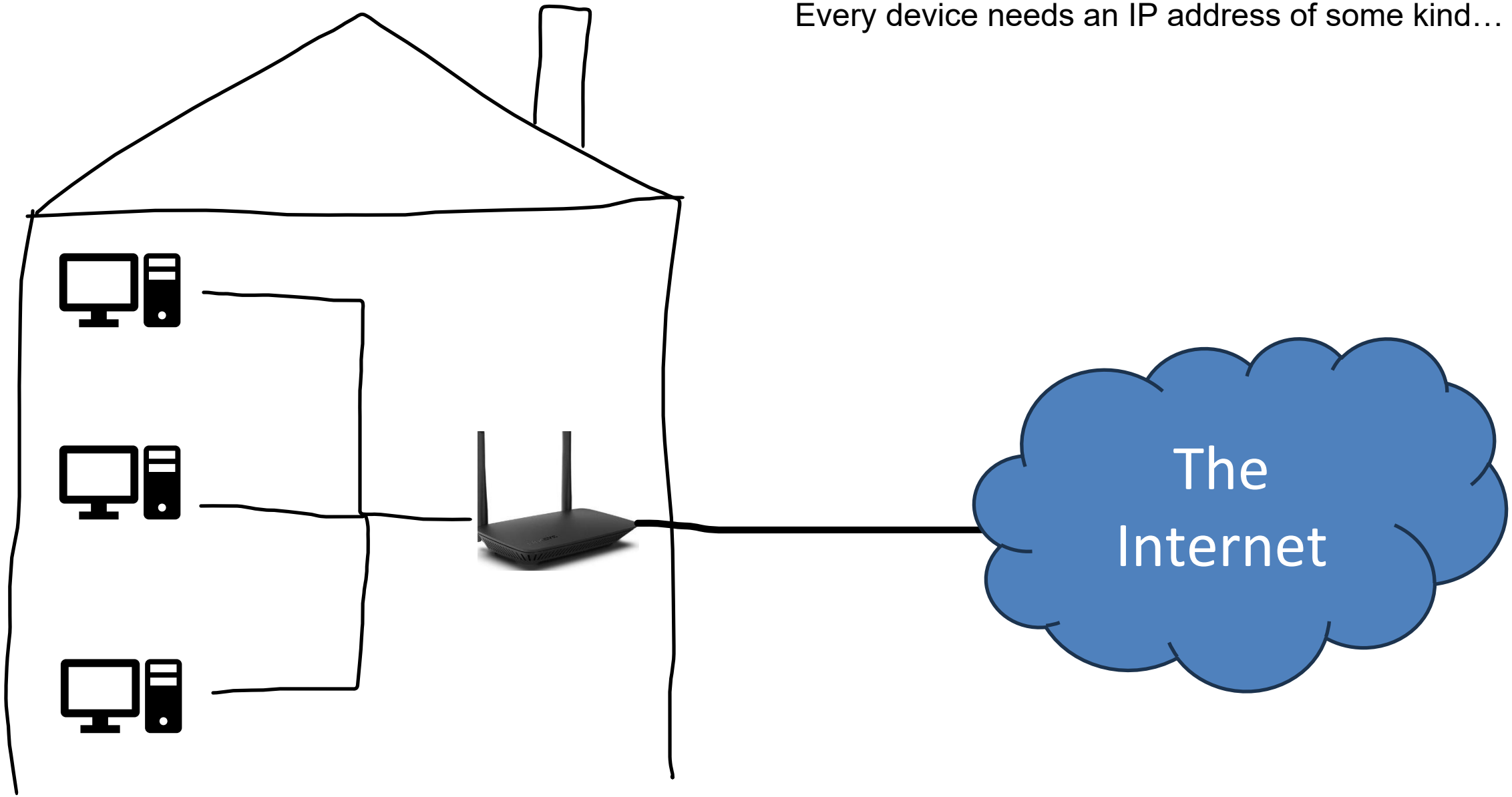
Issue: IPv4 can only support ~4 billion devices

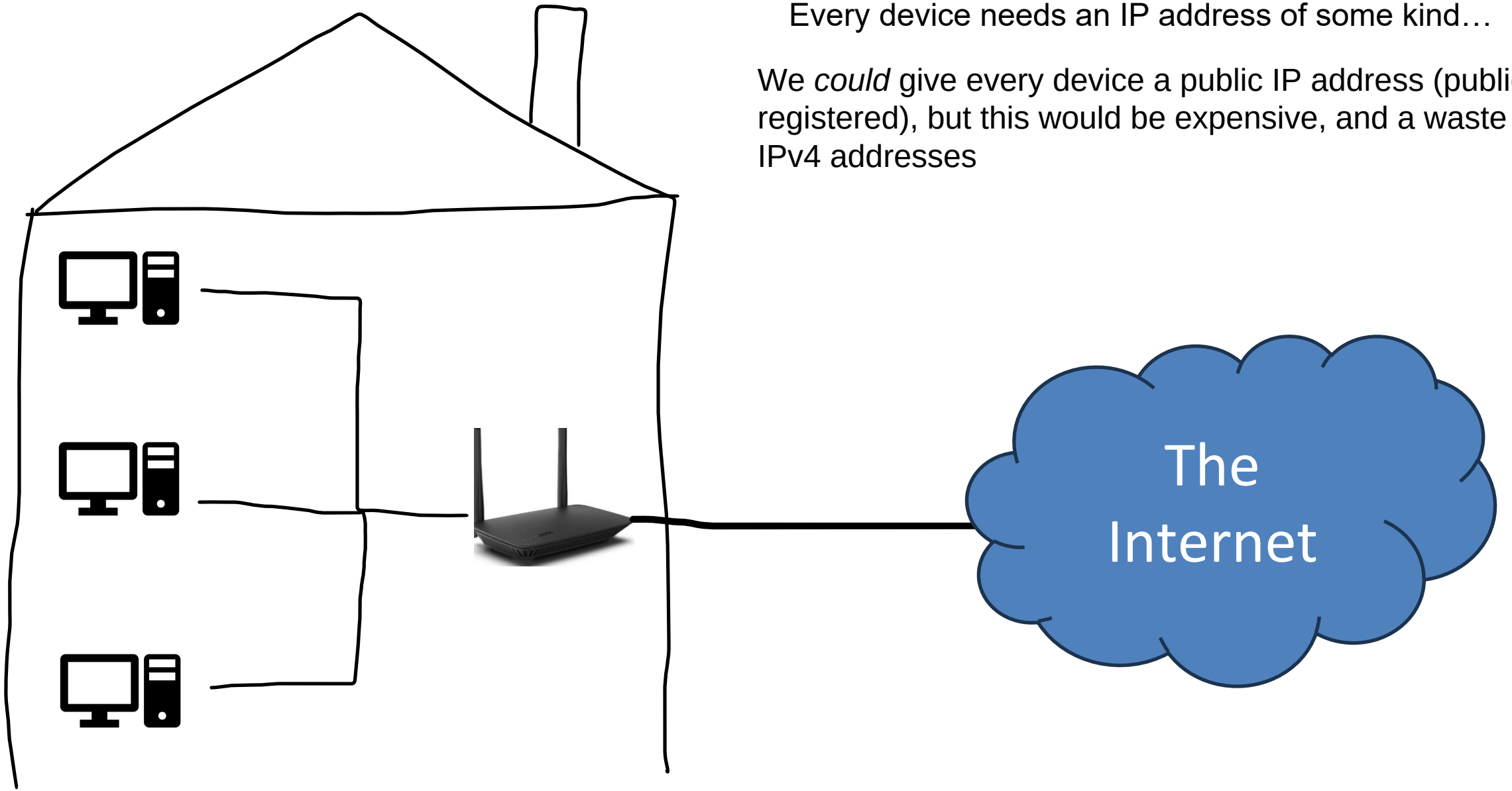
If every device on the internet gets an IP address, we are going to run out of IP addresses to use

Solution: IPv6 (This will take awhile to implement and switch to)

Temporary Solution: Public IP addresses, Private IP addresses, NAT

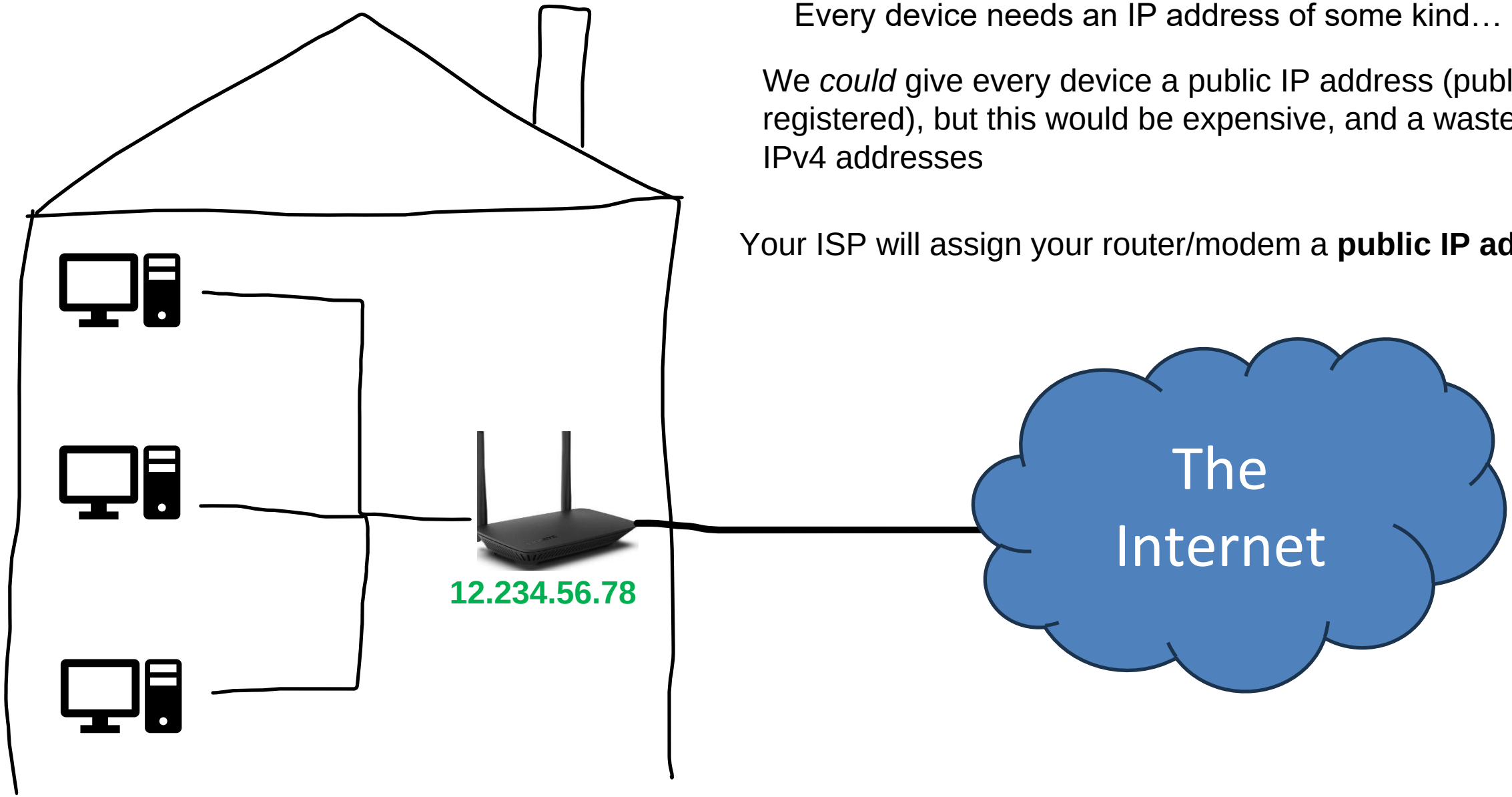
Every device needs an IP address of some kind…





Every device needs an IP address of some kind…

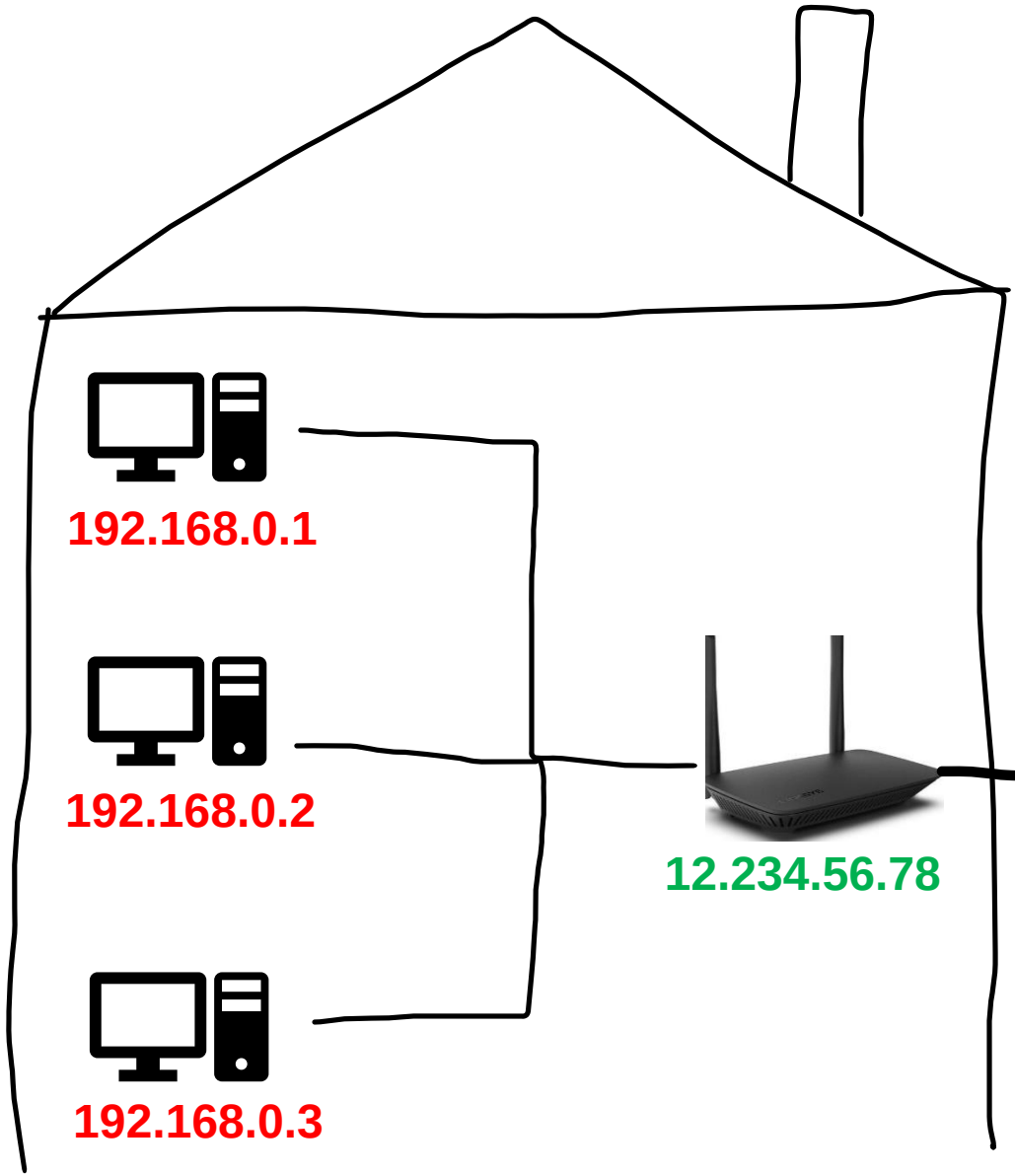
We *could* give every device a public IP address (public registered), but this would be expensive, and a waste of IPv4 addresses



Every device needs an IP address of some kind…

We *could* give every device a public IP address (public registered), but this would be expensive, and a waste of IPv4 addresses

Your ISP will assign your router/modem a **public IP address**



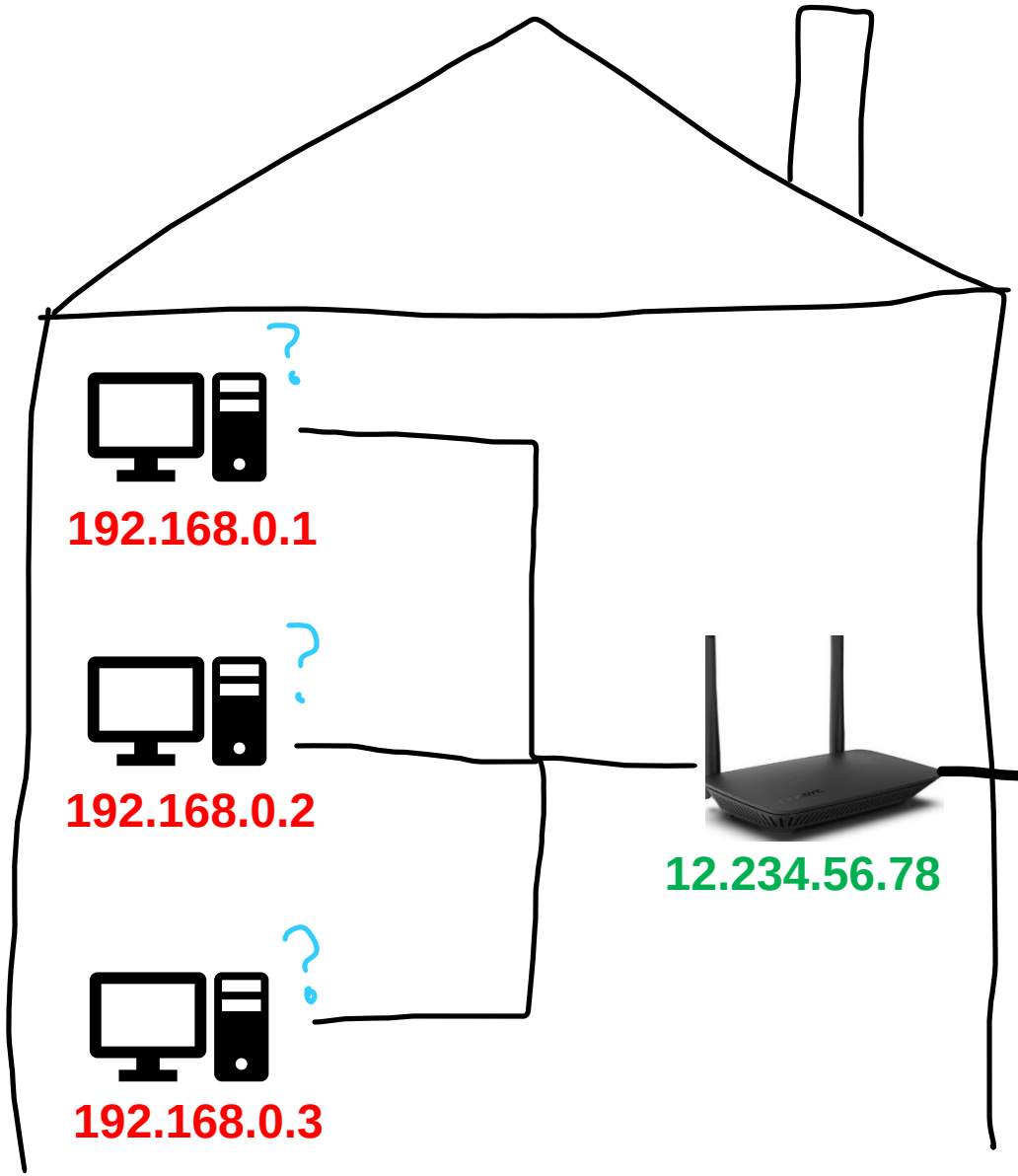
Your router will assign **private IP addresses** to each device via **DHCP**

Private IP addresses are only used internally, they are not seen by the “outside world”

Private IP addresses typically have a common prefix

The Internet

10.x.x.x
172.x.x.x
192.168.x.x

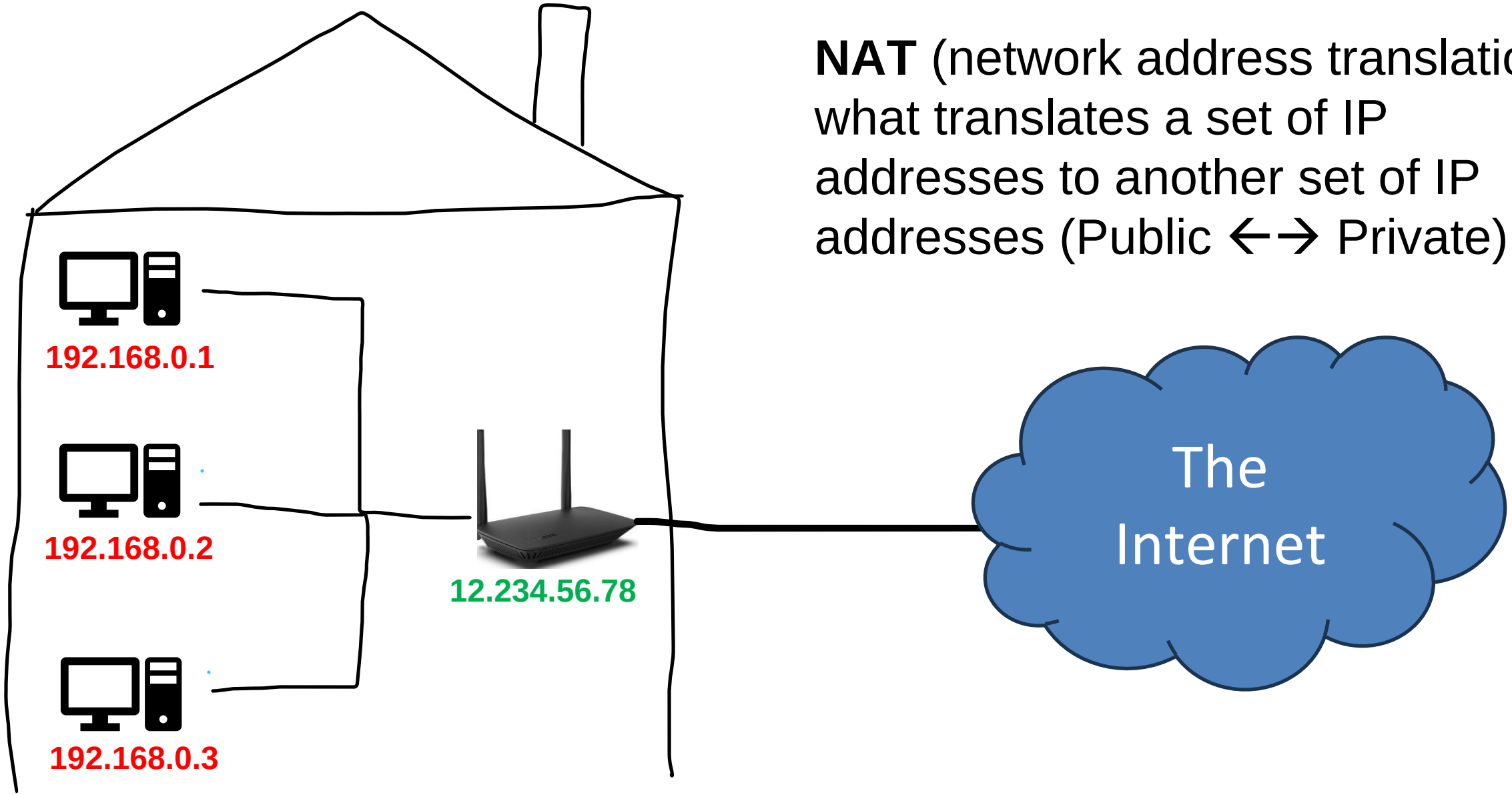


Issue: When receiving a packet with our public IP address, we need to convert to our private IP address to ensure it gets delivered to the correct host

(and vice versa)

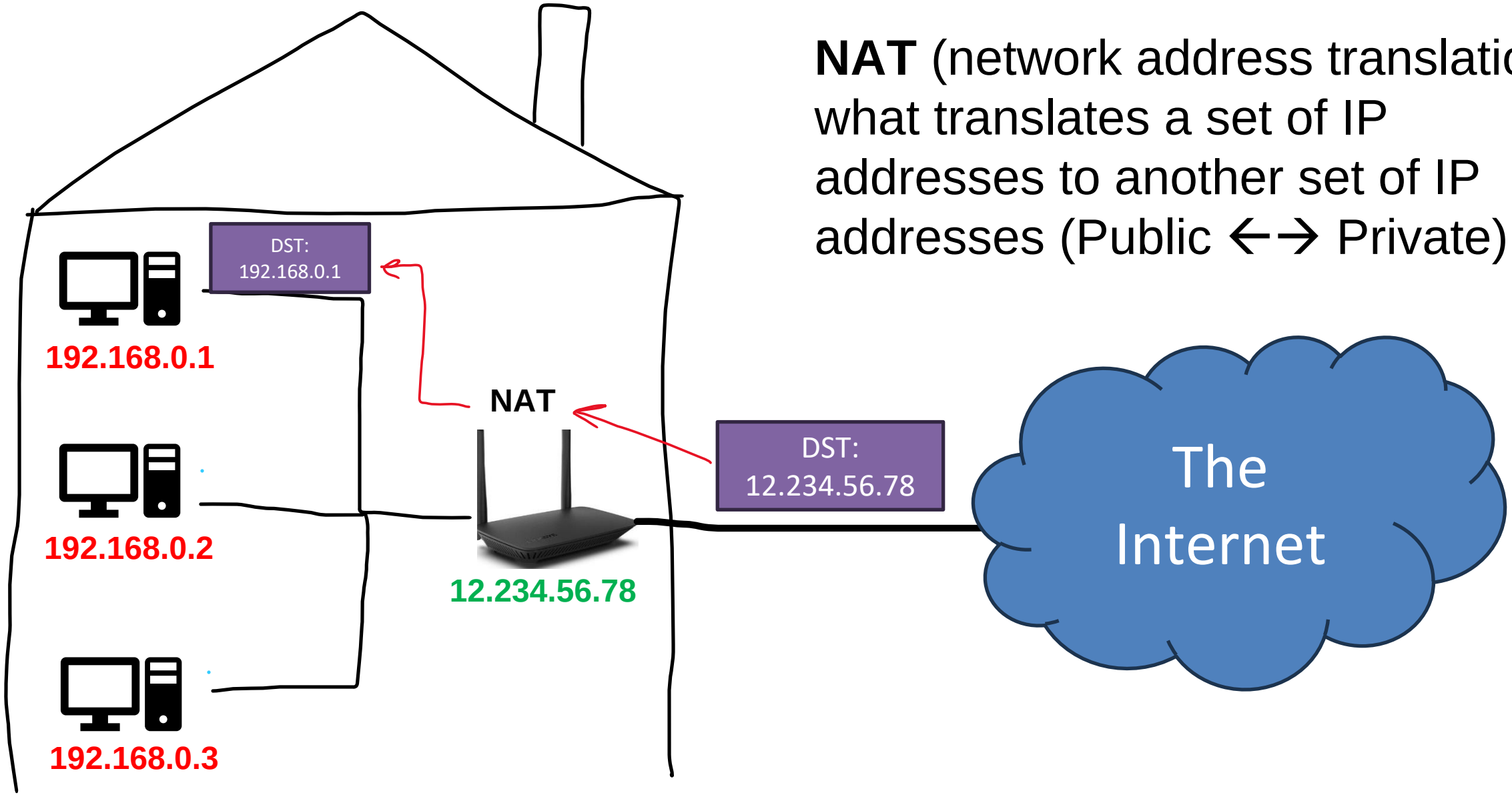
DST:
12.234.56.78

The Internet

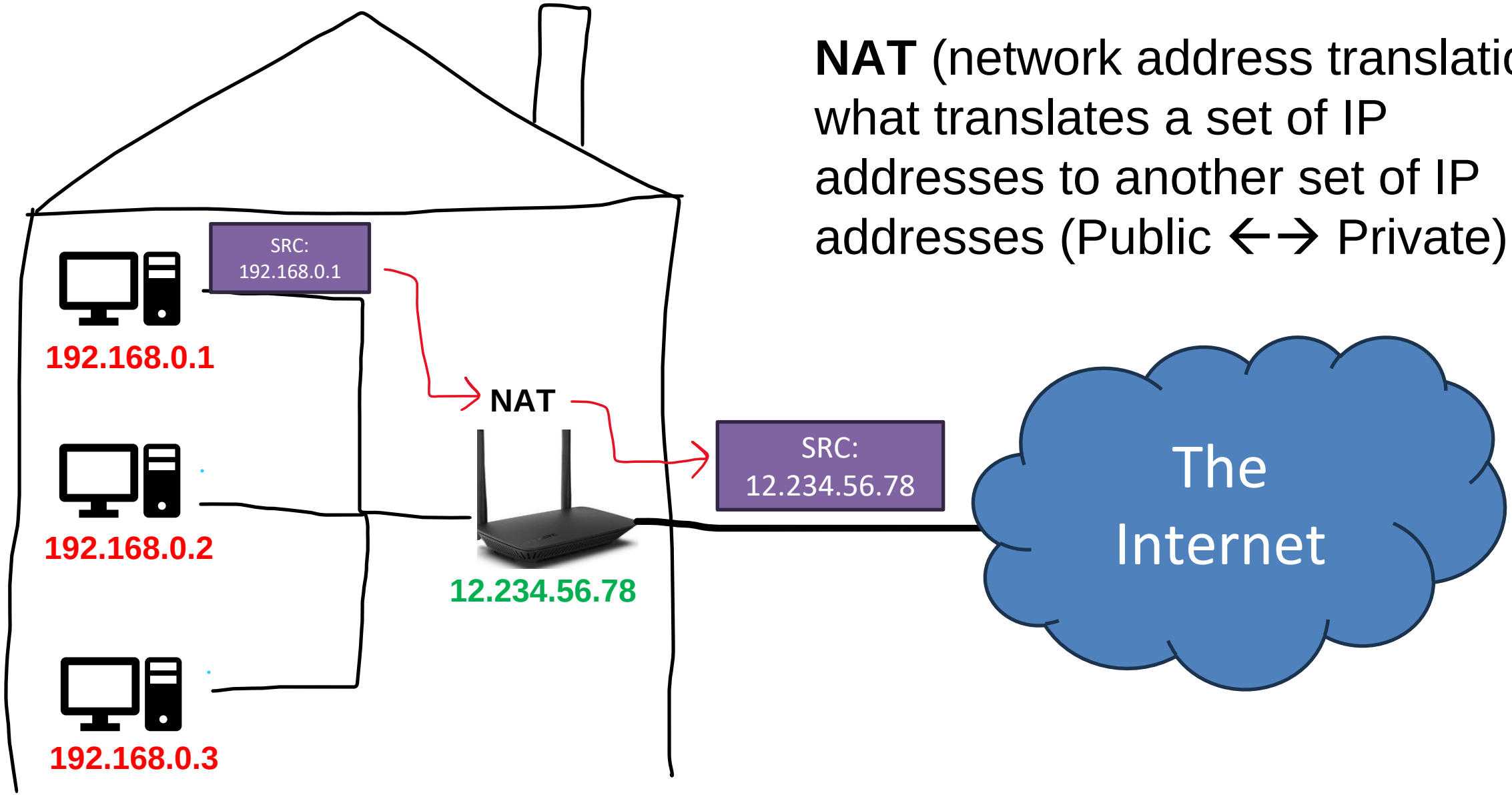


NAT (network address translation) is what translates a set of IP addresses to another set of IP addresses (Public ←→ Private)

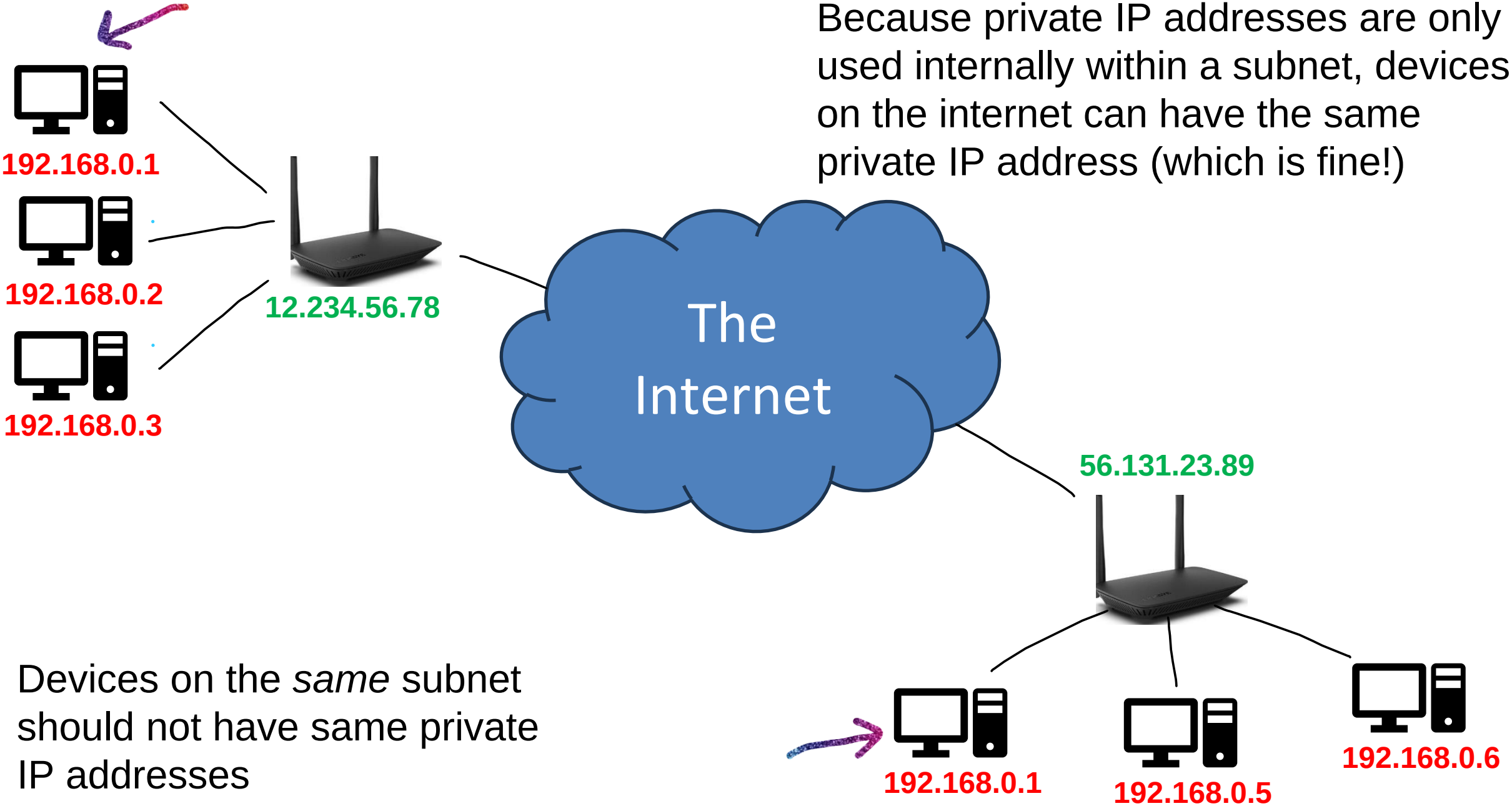
NAT (network address translation) is what translates a set of IP addresses to another set of IP addresses (Public ←→ Private)



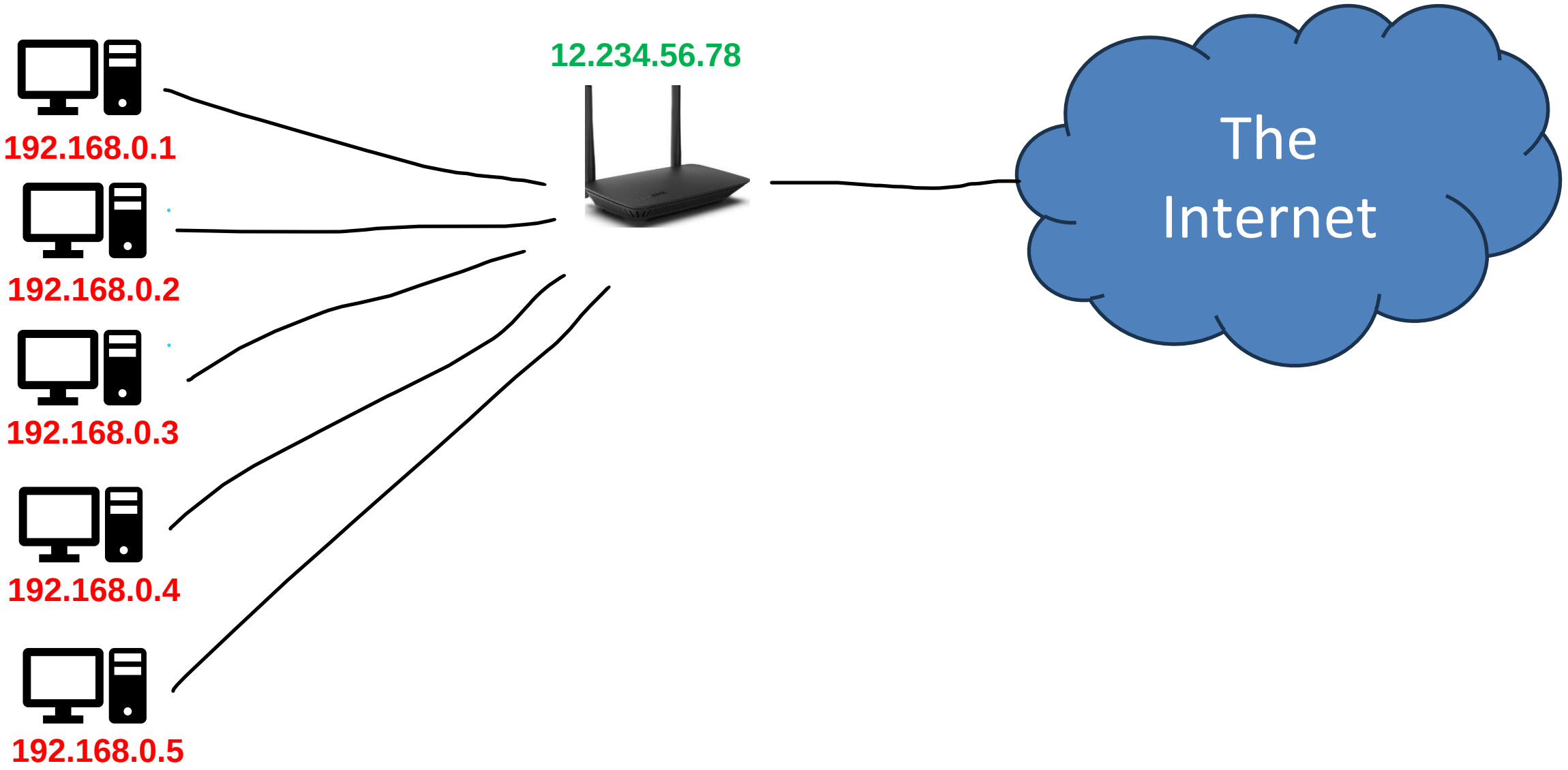
NAT (network address translation) is what translates a set of IP addresses to another set of IP addresses (Public ←→ Private)

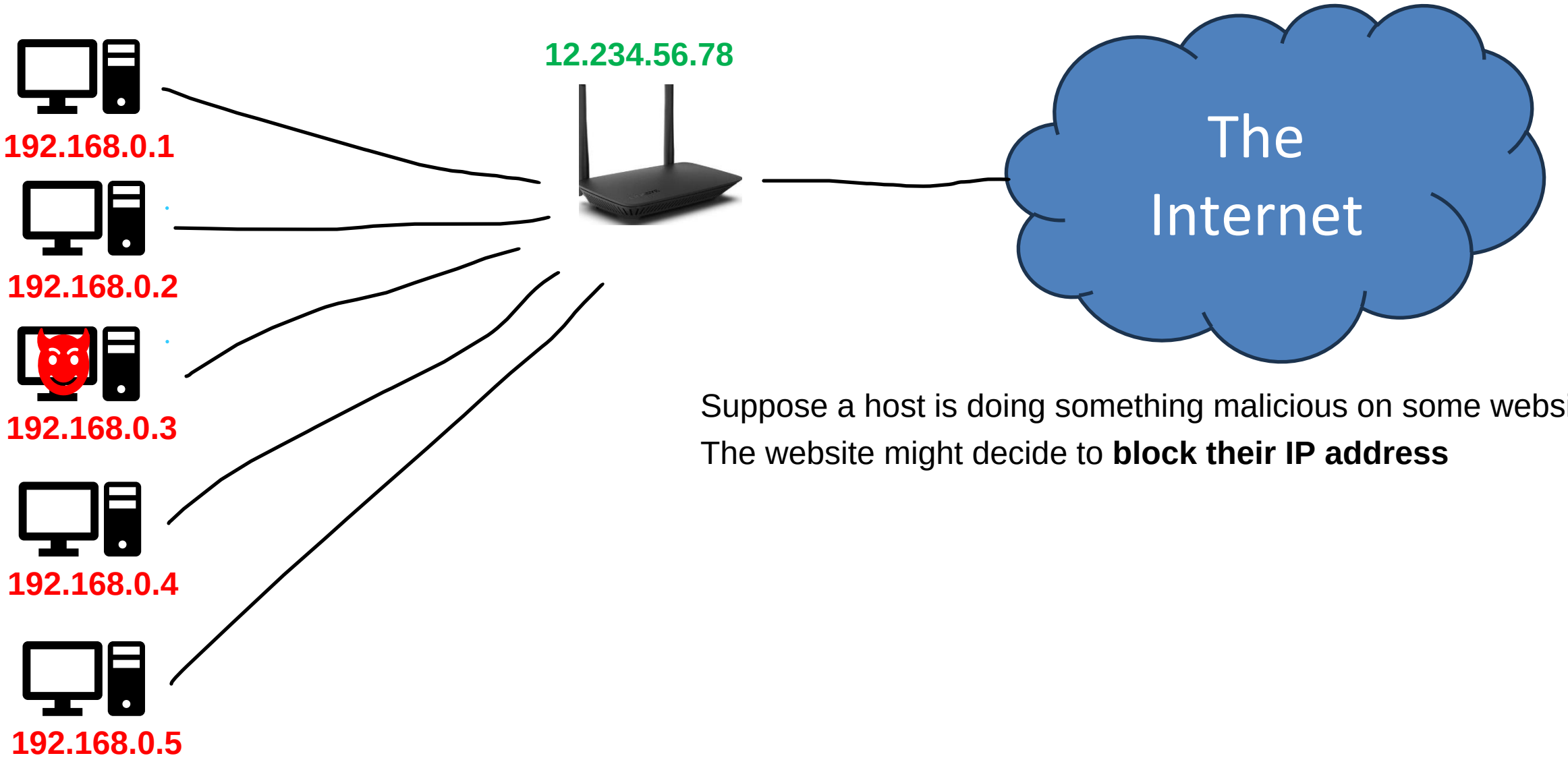


Because private IP addresses are only used internally within a subnet, devices on the internet can have the same private IP address (which is fine!)



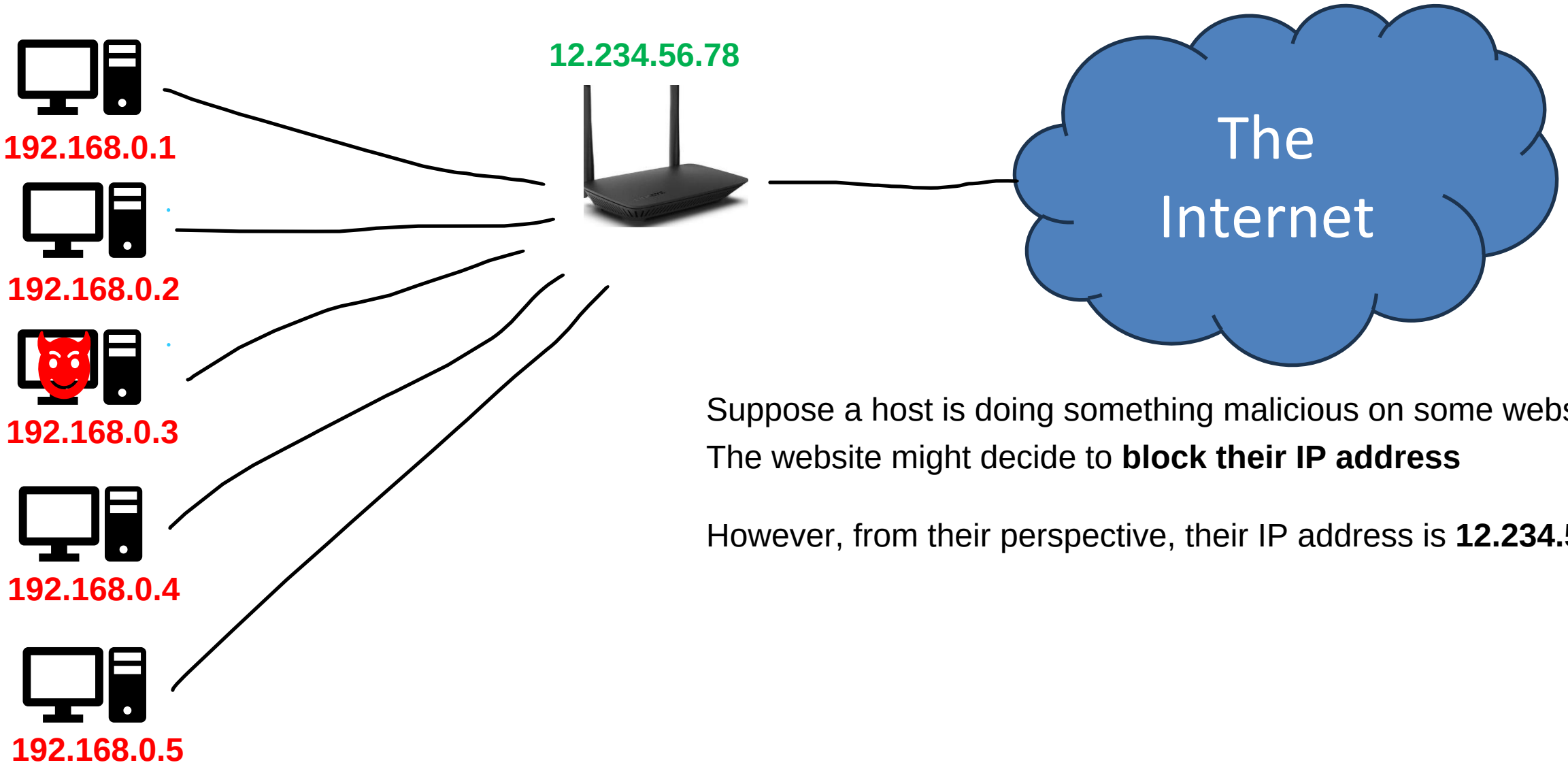
Devices on the *same* subnet should not have same private IP addresses


12.234.56.78
The Internet
192.168.0.1
192.168.0.2
192.168.0.3
192.168.0.4
192.168.0.5



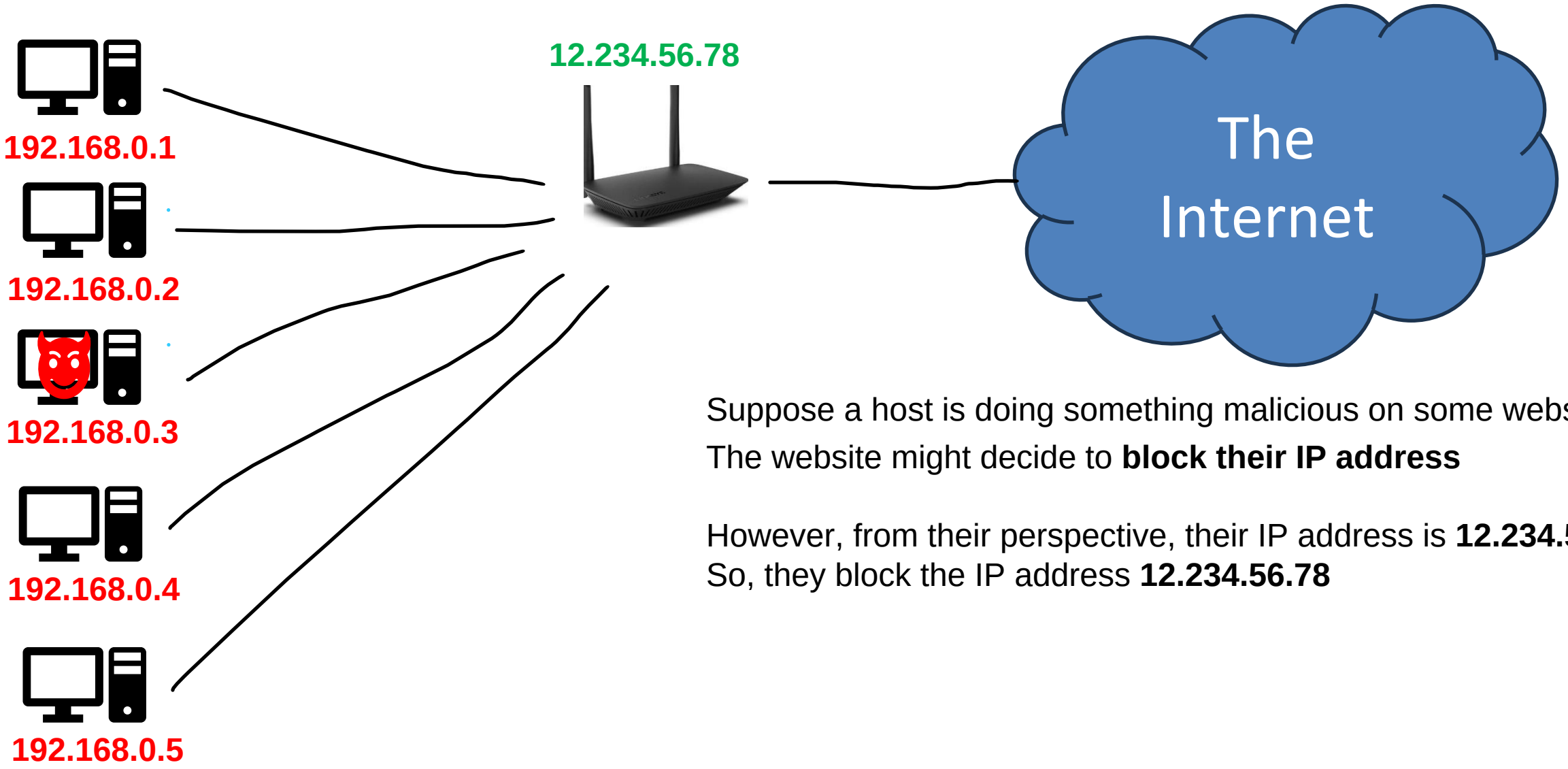
Suppose a host is doing something malicious on some website

The website might decide to **block their IP address**



Suppose a host is doing something malicious on some website
The website might decide to **block their IP address**

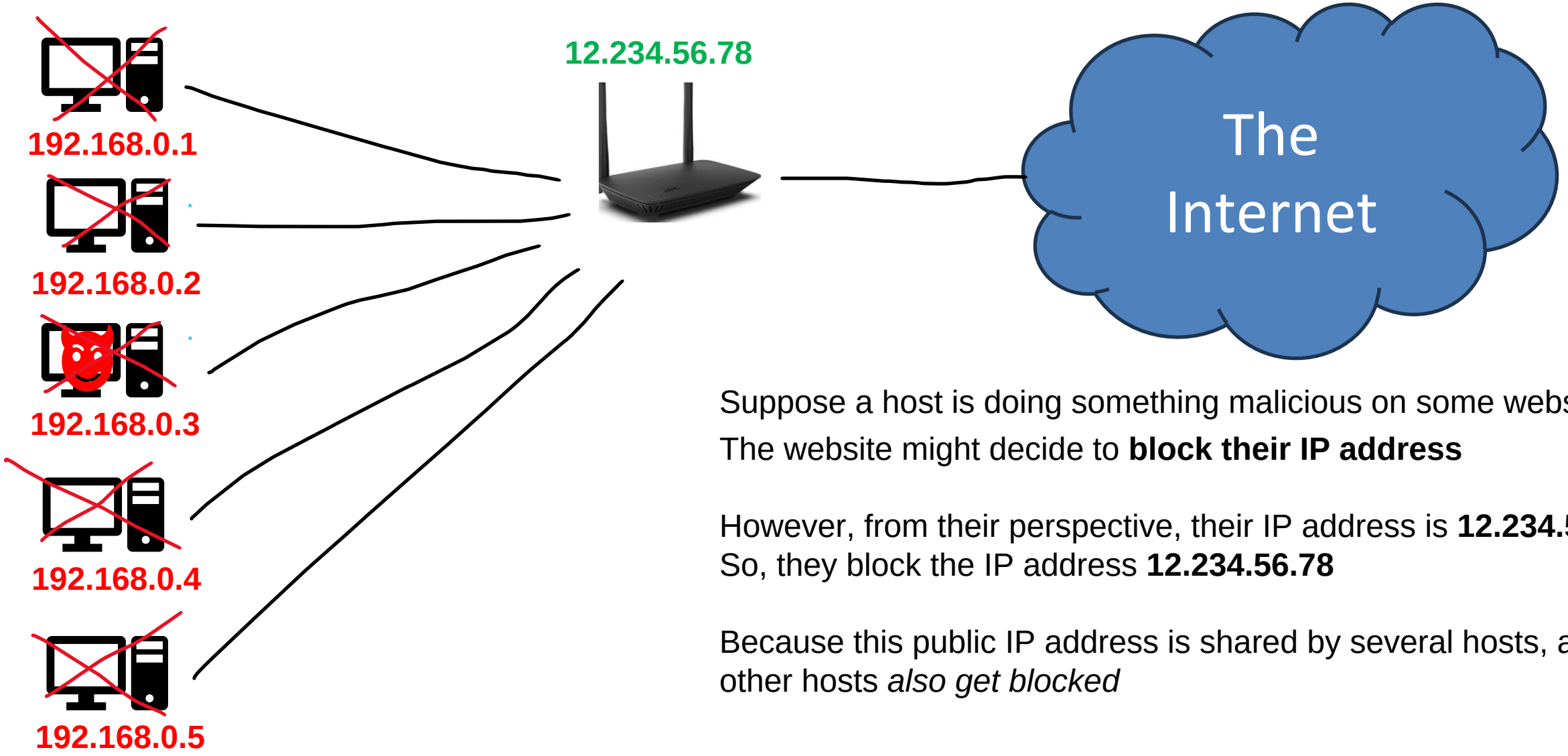
However, from their perspective, their IP address is **12.234.56.78**



Suppose a host is doing something malicious on some website

The website might decide to **block their IP address**

However, from their perspective, their IP address is **12.234.56.78**
So, they block the IP address **12.234.56.78**



Suppose a host is doing something malicious on some website
The website might decide to **block their IP address**

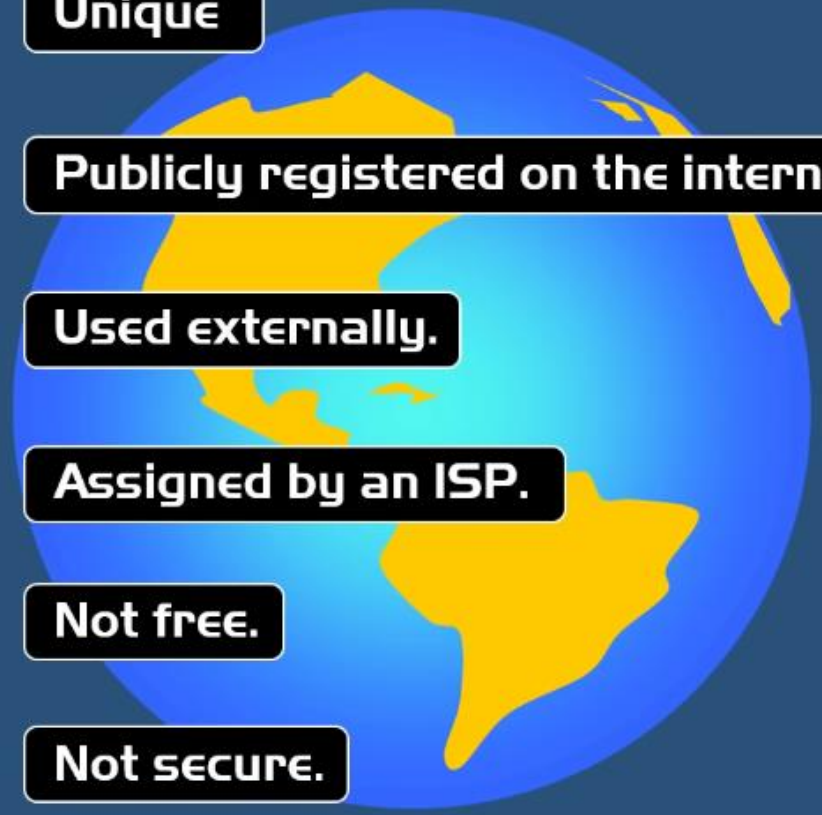
However, from their perspective, their IP address is **12.234.56.78**
So, they block the IP address **12.234.56.78**

Because this public IP address is shared by several hosts, all the other hosts *also get blocked*

How to see your private and public IP address

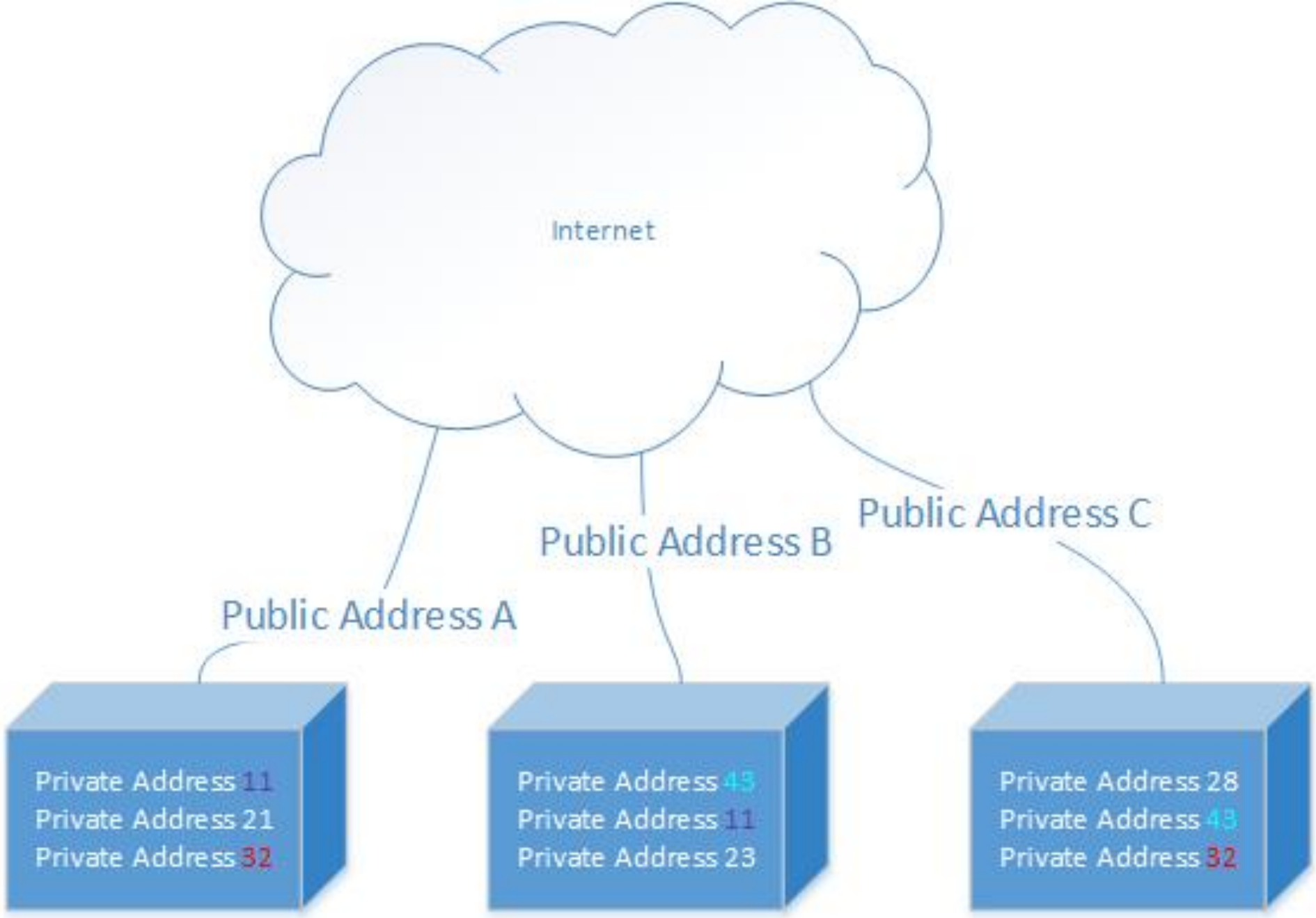
Public IP

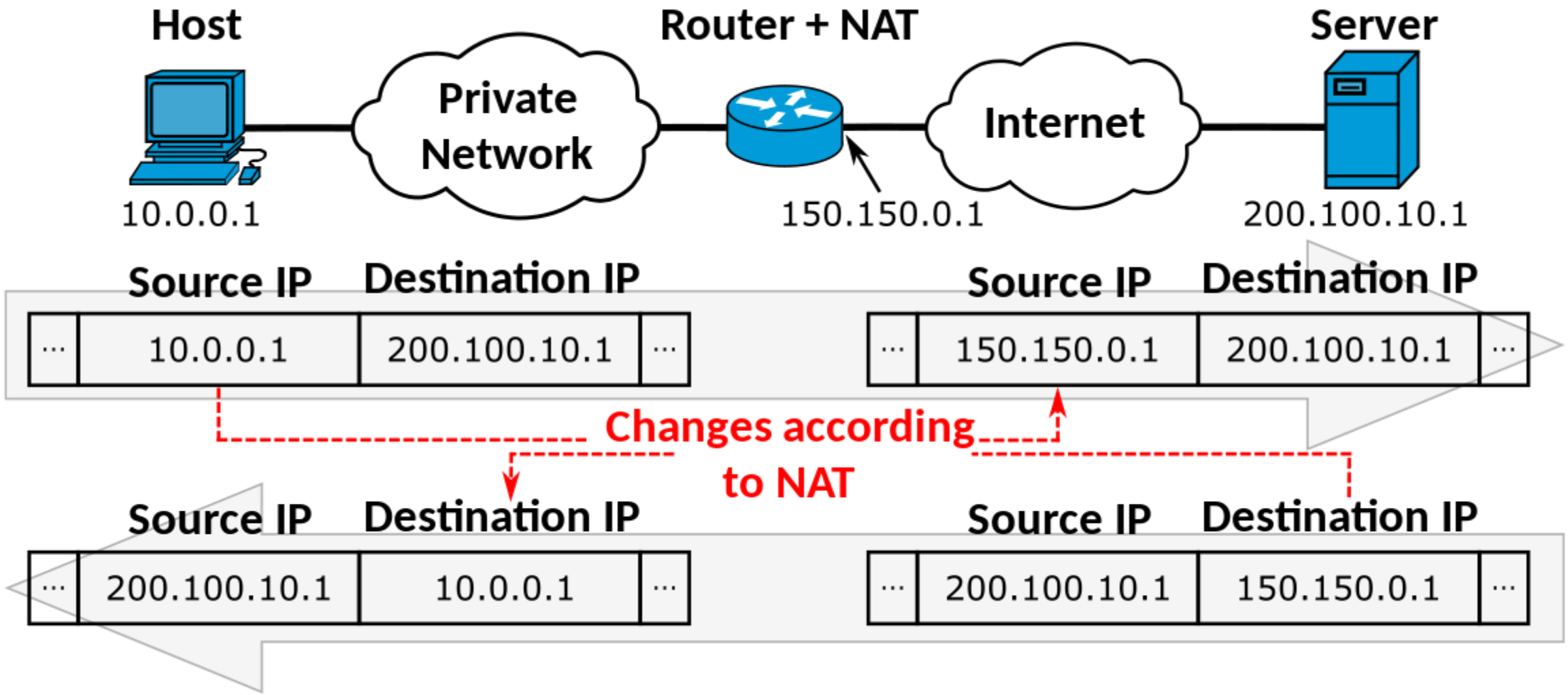
- Unique
- Publicly registered on the internet.
- Used externally.
- Assigned by an ISP.
- Not free.
- Not secure.



Private IP

- Non-unique. Can be used on other private networks.
- Not publicly registered.
- Used internally.
- Assigned by a router.
- Free.
- More secure.


Internet
Public Address A
Public Address B
Public Address C
Private Address 11
Private Address 21
Private Address 32
Private Address 43
Private Address 11
Private Address 23
Private Address 28
Private Address 43
Private Address 32

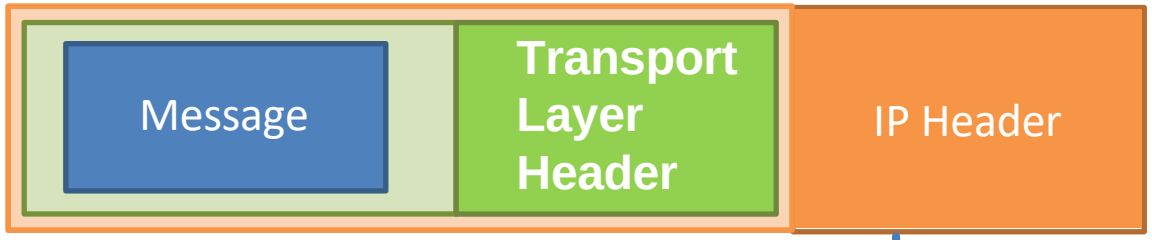

Host
Router + NAT
Server
Private Network
Internet
10.0.0.1
150.150.0.1
200.100.10.1
Source IP
Destination IP
10.0.0.1
200.100.10.1
Source IP
Destination IP
150.150.0.1
200.100.10.1
Changes according to NAT
Source IP
Destination IP
200.100.10.1
10.0.0.1
Source IP
Destination IP
200.100.10.1
150.150.0.1

NAT: network address translation

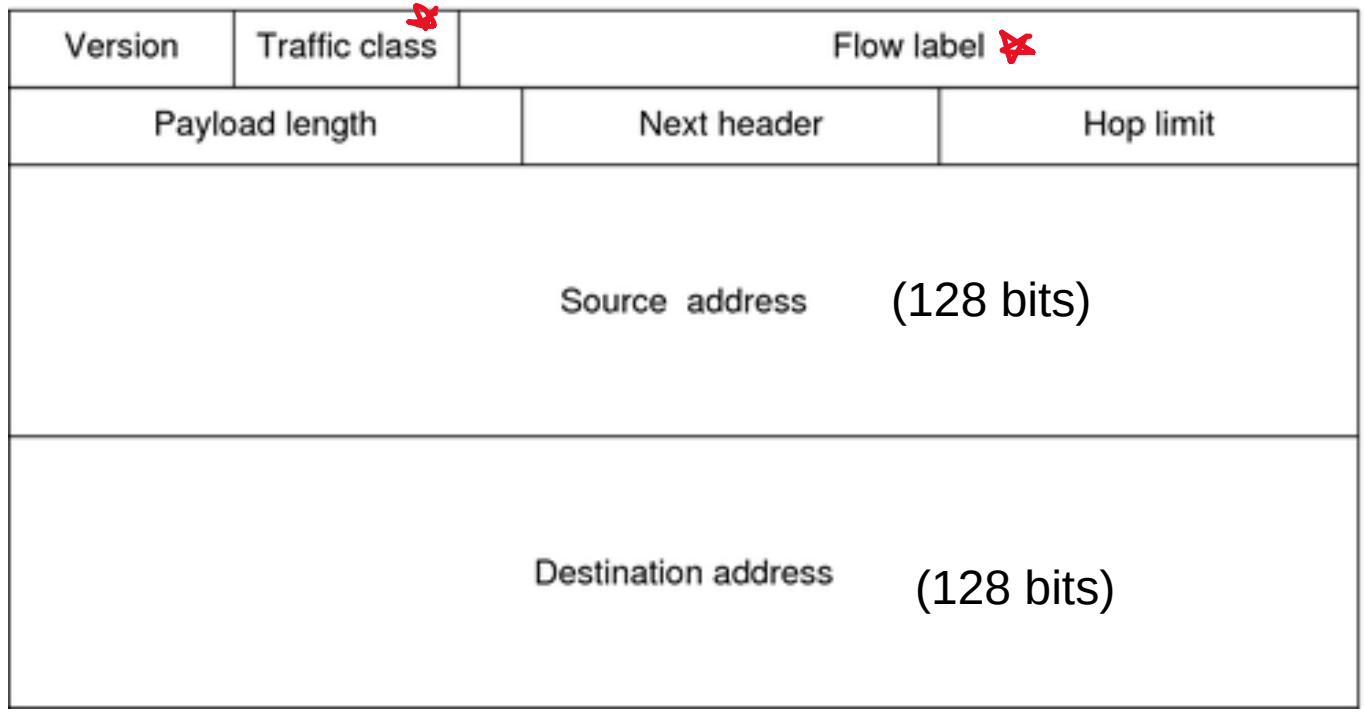
implementation: NAT router must (transparently):

- outgoing datagrams: replace (source IP address, port #) of every outgoing datagram to (NAT IP address, new port #)
- remember (in NAT translation table) every (source IP address, port #) to (NAT IP address, new port #) translation pair
- incoming datagrams: replace (NAT IP address, new port #) in destination fields of every incoming datagram with corresponding (source IP address, port #) stored in NAT table

Packets traversing through the network layer are referred to as a **datagram**. Each packet gets an IPv4/IPv6 header



TCP header = 20 bytes
IP header = 20 bytes



IPv6 = **128 bits**

IPv6 Header

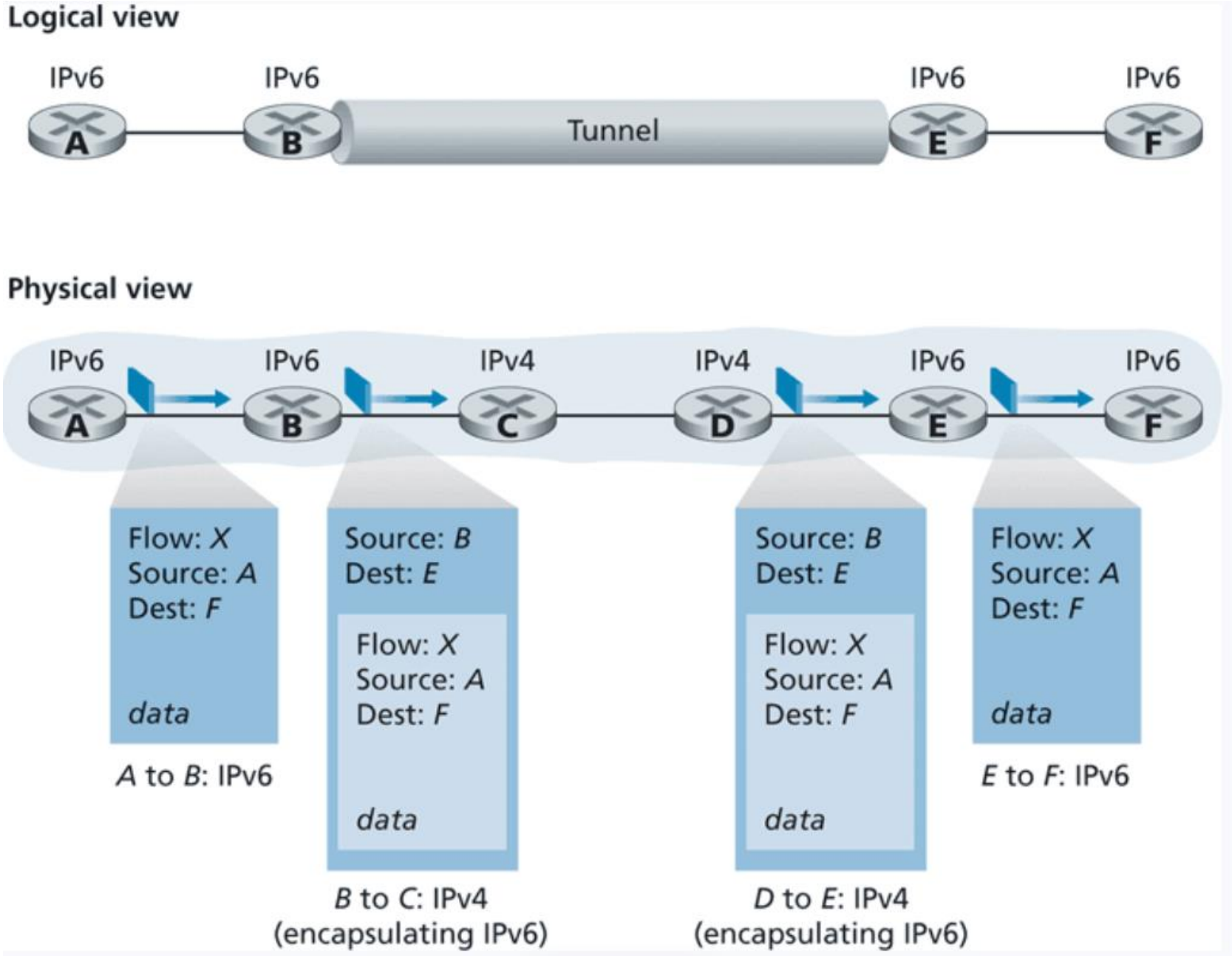
The goal is for our internet to be only using IPv6 in the near future? Issues?

Issue: some routers use IPv4, some use IPv6, some use both. How do we get all networking equipment to coexist with each other?

Solution:

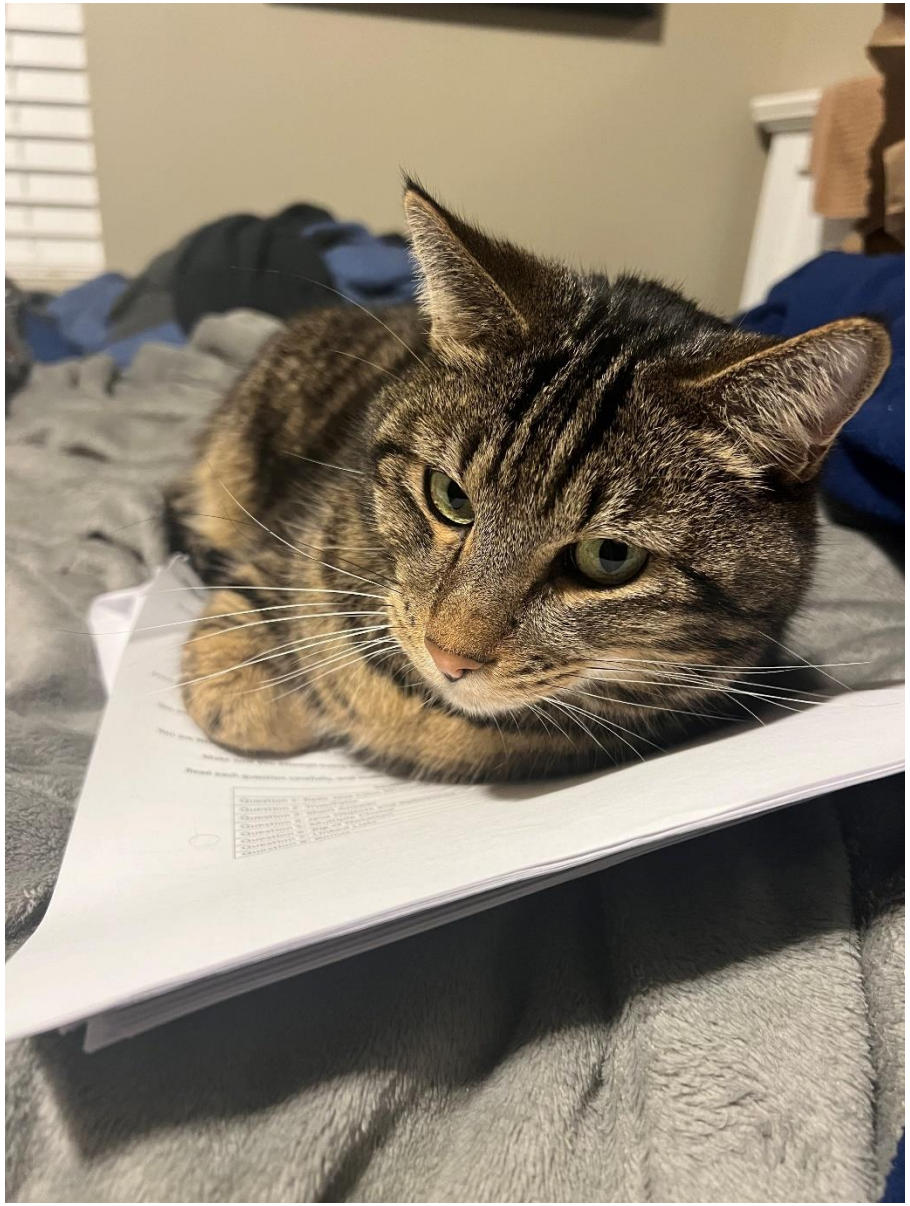
Issue: some routers use IPv4, some use IPv6, some use both. How do we get all networking equipment to coexist with each other?

Solution: **Tunneling**



tl:dr

We wrap our IPv6 datagram in an IPv4 header, and pass it to the IPv4 router!



So far, a router takes input from input port, and then outputs on some output port

What else might a router need to do?

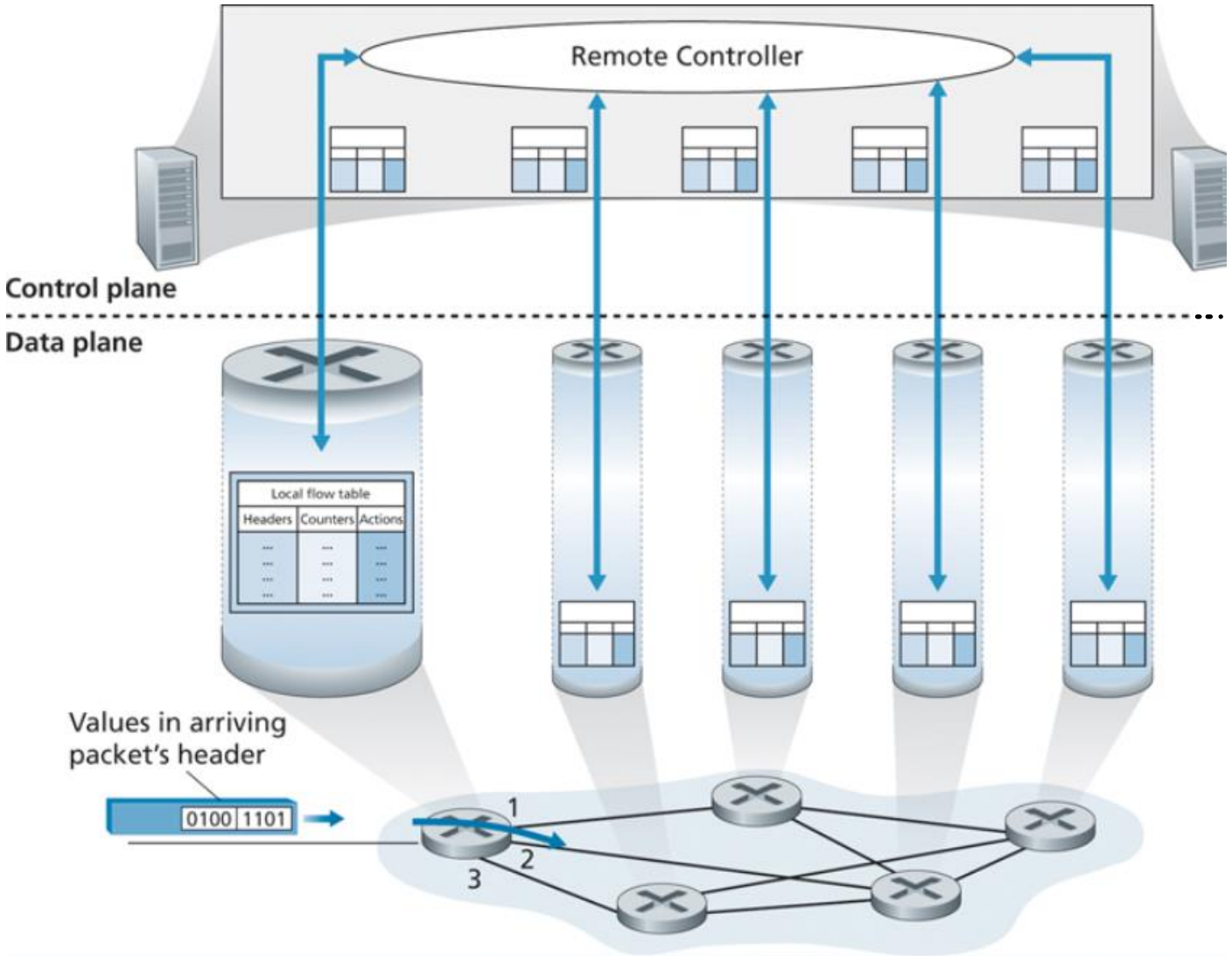
So far, a router takes input from input port, and then outputs on some output port

What else might a router need to do?

Forward, Drop stuff, Modify, Load balance

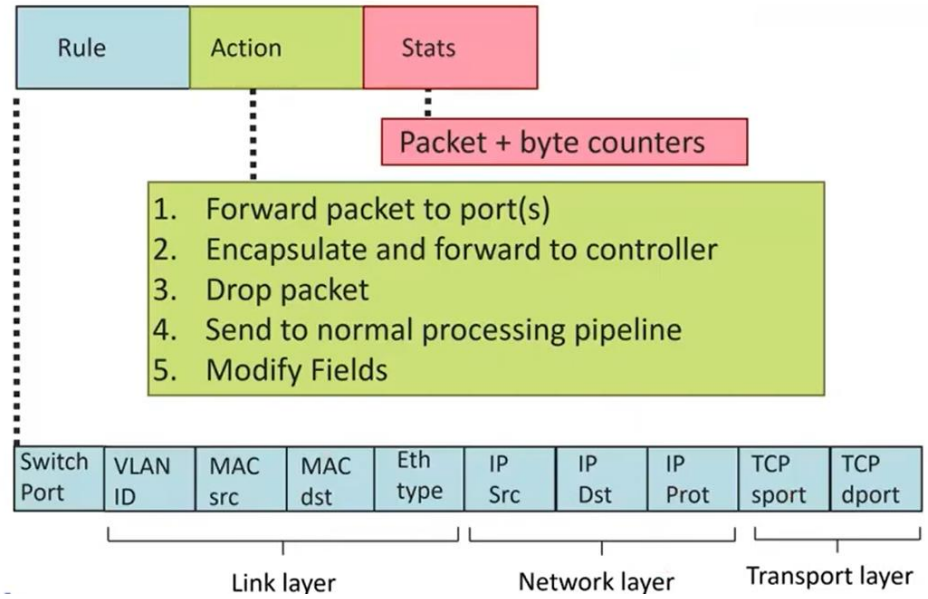
We need more flexibility and functionality with our forwarding!!

OpenFlow standard

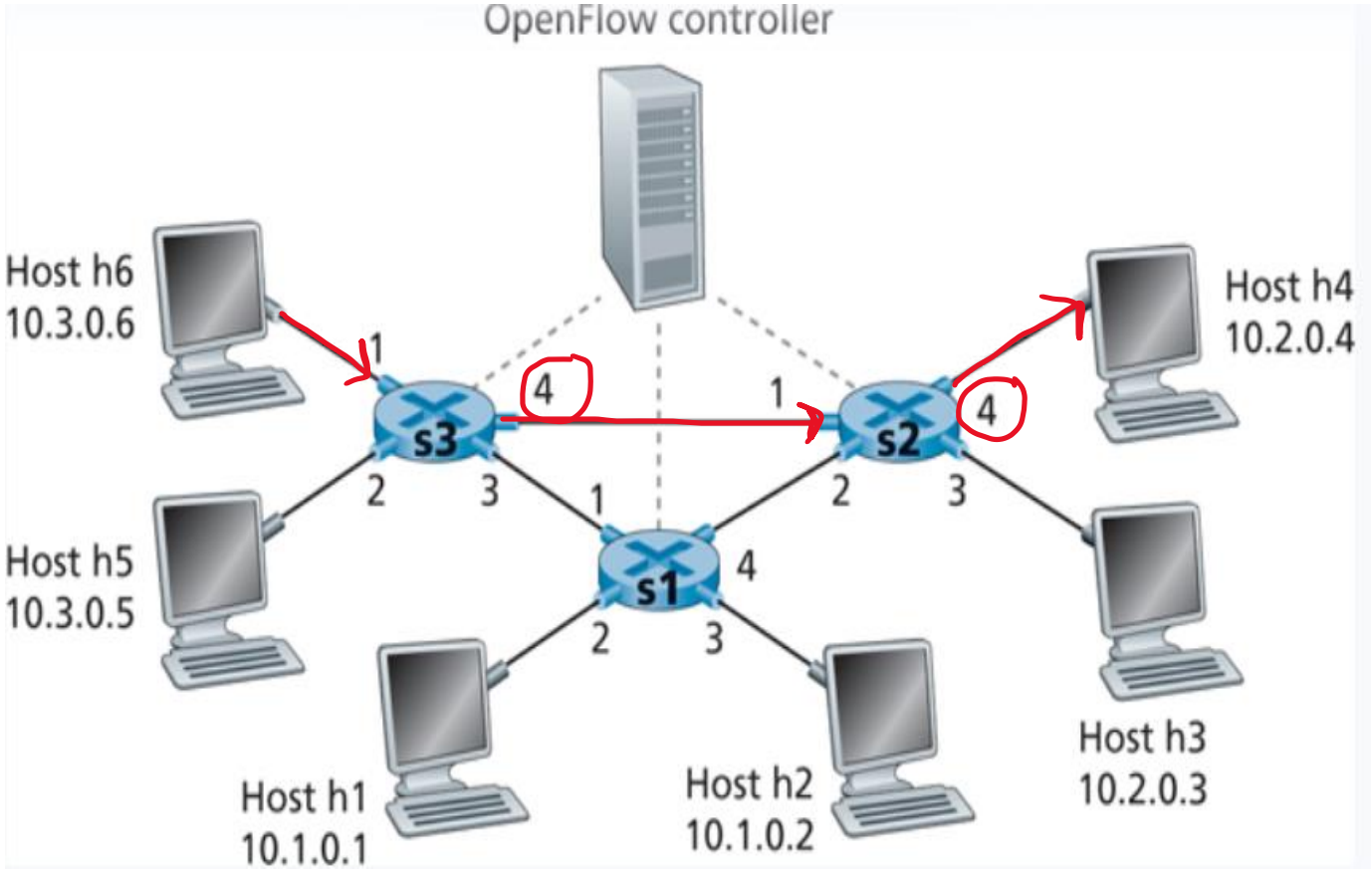


We need **headers/rules**, which are going to the values the remote controller is going to evaluate

We need **actions** to do based on some pattern match



Generalized Forwarding and Software Defines Network (SDN)

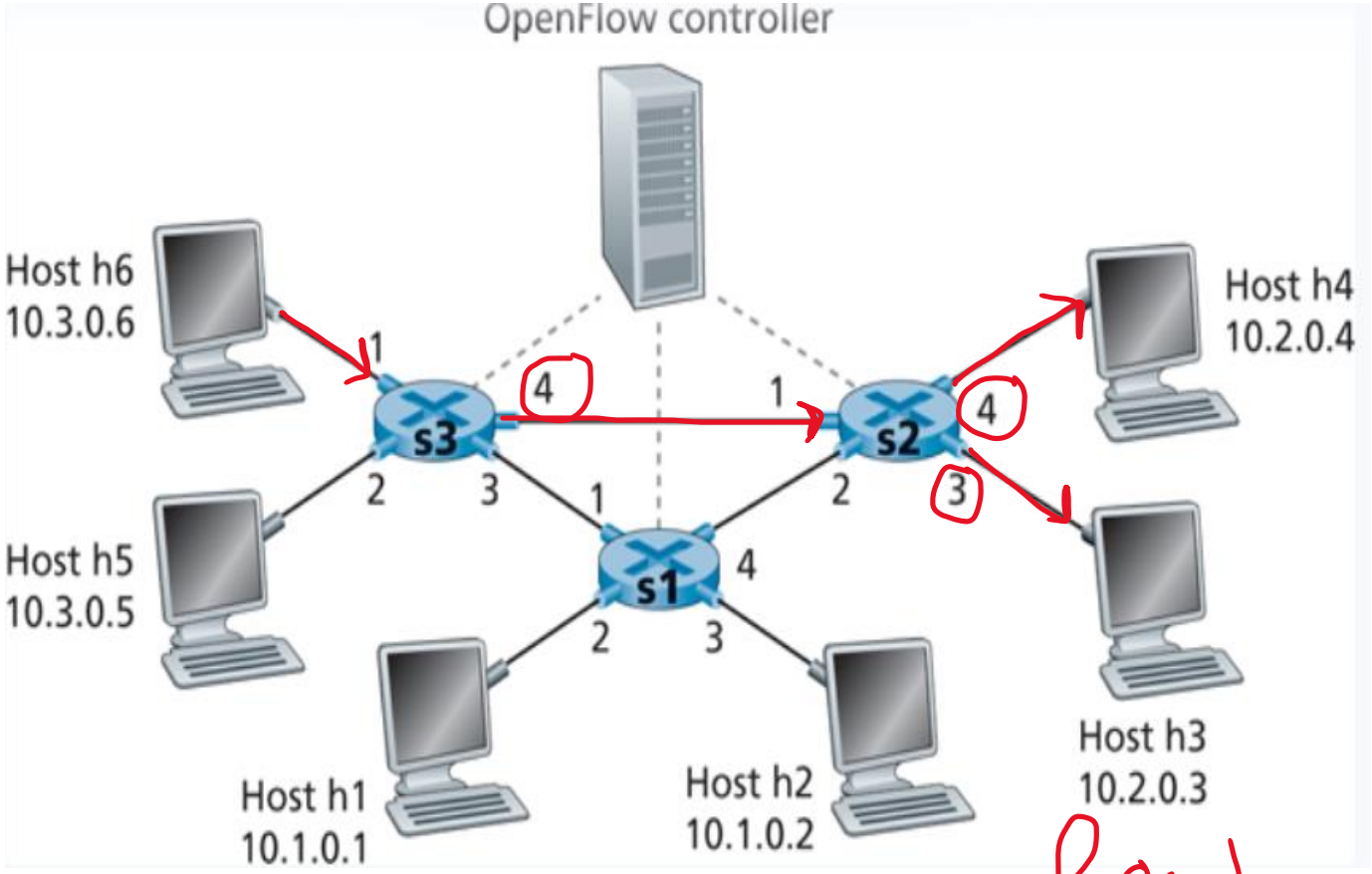


Match	Action
Ingress Port = 1 ; IP Src = 10.3.*.* ; IP Dst = 10.2.*.*	Forward(4)

Match	Action
IP Src = 10.3.*.* ; IP Dst = 10.2.*.*	Forward(3)

Match	Action
Ingress port = 2 ; IP Dst = 10.2.0.3	Forward(3)
Ingress port = 2 ; IP Dst = 10.2.0.4	Forward(4)

Generalized Forwarding and Software Defines Network (SDN)



Match	Action
Ingress Port = 1 ; IP Src = 10.3.*.* ; IP Dst = 10.2.*.*	Forward(4)

Match	Action
IP Src = 10.3.*.* ; IP Dst = 10.2.*.*	Forward(3)

Match	Action
Ingress port = 2 ; IP Dst = 10.2.0.3	Forward(3)
Ingress port = 2 ; IP Dst = 10.2.0.4	Forward(4)

Road Balancing

Destination-based forwarding:

Switch Port	MAC src	MAC dst	Eth type	VLAN ID	IP Src	IP Dst	IP Prot	TCP sport	TCP dport	Action
Pattern →	*	*	*	*	*	*	51.6.0.8	*	*	*	port6

← Action

IP datagrams destined to IP address 51.6.0.8 should be forwarded to router output port 6

Firewall:

Switch Port	MAC src	MAC dst	Eth type	VLAN ID	IP Src	IP Dst	IP Prot	TCP sport	TCP dport	Forward
*	*	*	*	*	*	*	*	*	22	drop

do not forward (block) all datagrams destined to TCP port 22

Destination-based layer 2 (switch) forwarding:

Switch Port	MAC src	MAC dst	Eth type	VLAN ID	IP Src	IP Dst	IP Prot	TCP sport	TCP dport	Action
*	22:A7:23:11:E1:02	*	*	*	*	*	*	*	*	port3

layer 2 frames from MAC address 22:A7:23:11:E1:02 should be forwarded to output port 6