[544] Networking

Tyler Caraza-Harter

Learning Objectives

- explain how MAC addresses, IP addresses, and port numbers provide addressing, used to facilitate communicate between processes on different machines
- select IP addresses correctly for binding and using in a browser to achieve connection in the context of a server running behind a NAT
- identify the port number being used by a process
- select between transport methods (TCP and UDP) based on the functionality needed (on top of IP functionality)

Outline

Networks

Internets and "The Internet"

Transport Protocols

Network Interface Controllers and MAC Addresses

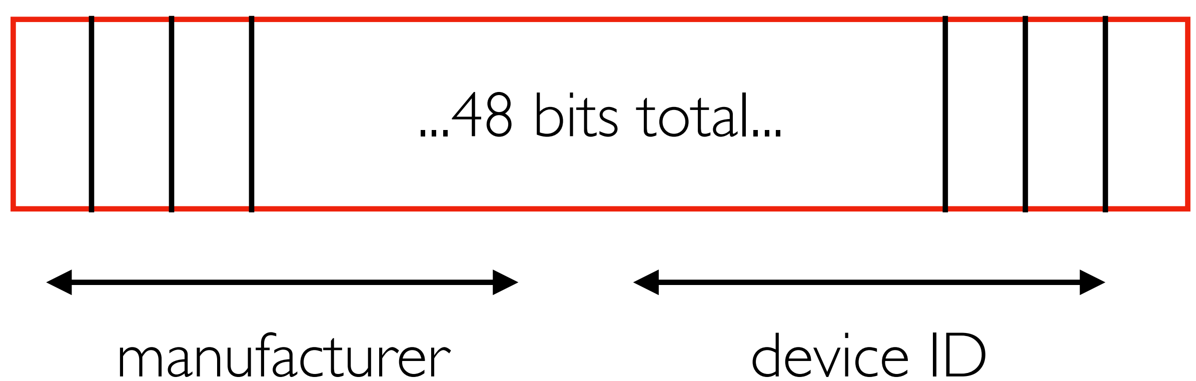
NICs can connect a computer to different physical mediums, such as:

- Ethernet (wired)
- Wi-Fi (wireless)

Every NIC in the world has a unique MAC (media access control) address

- 28 trillion possible addrs
- some devices randomly change their MAC addr for privacy

MAC address:



ip address

```
trh@instance-20240903-151711:~$ ip address
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group
default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host noprefixroute
       valid_lft forever preferred_lft forever
2: ens4: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1460 qdisc mq state UP group
default qlen 1000
    link/ether 42:01:0a:80:00:04 brd ff:ff:ff:ff:ff:ff
    altname enp0s4
    inet 10.128.0.4/32 metric 100 scope global dynamic ens4
       valid_lft 1870sec preferred_lft 1870sec
    inet6 fe80::4001:aff:fe80:4/64 scope link
       valid_lft forever preferred_lft forever
```

interface

MAC address

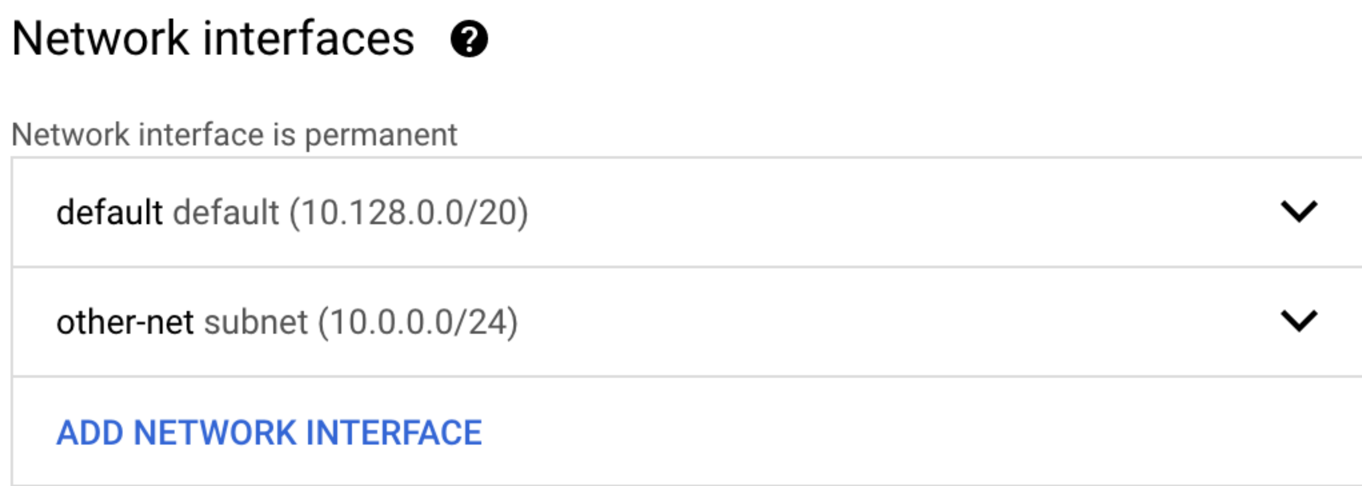
Virtual Interfaces

```
trh@instance-20240903-151711:~$ ip address
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group
default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host noprefixroute
       valid_lft forever preferred_lft forever
2: ens4: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1460 qdisc mq state UP group
default qlen 1000
    link/ether 42:01:0a:80:00:04 brd ff:ff:ff:ff:ff:ff
    altname enp0s4
    inet 10.128.0.4/32 metric 100 scope global dynamic ens4
       valid_lft 1870sec preferred_lft 1870sec
    inet6 fe80::4001:aff:fe80:4/64 scope link
       valid_lft forever preferred_lft forever
```

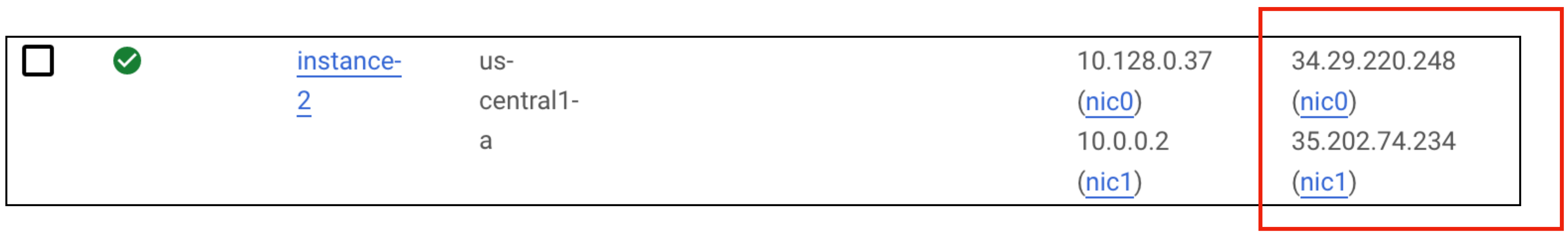
loopback (lo) device a virtual interface (not actual hardware)
connecting to a mini network containing just your computer

Google Console: Adding Interfaces (NICs)

Create Instance > Advanced Options > Networking



Virtual Machine Summary



Google Console: Adding Interfaces

Create Instance > Advanced Options > Networking

Network interfaces

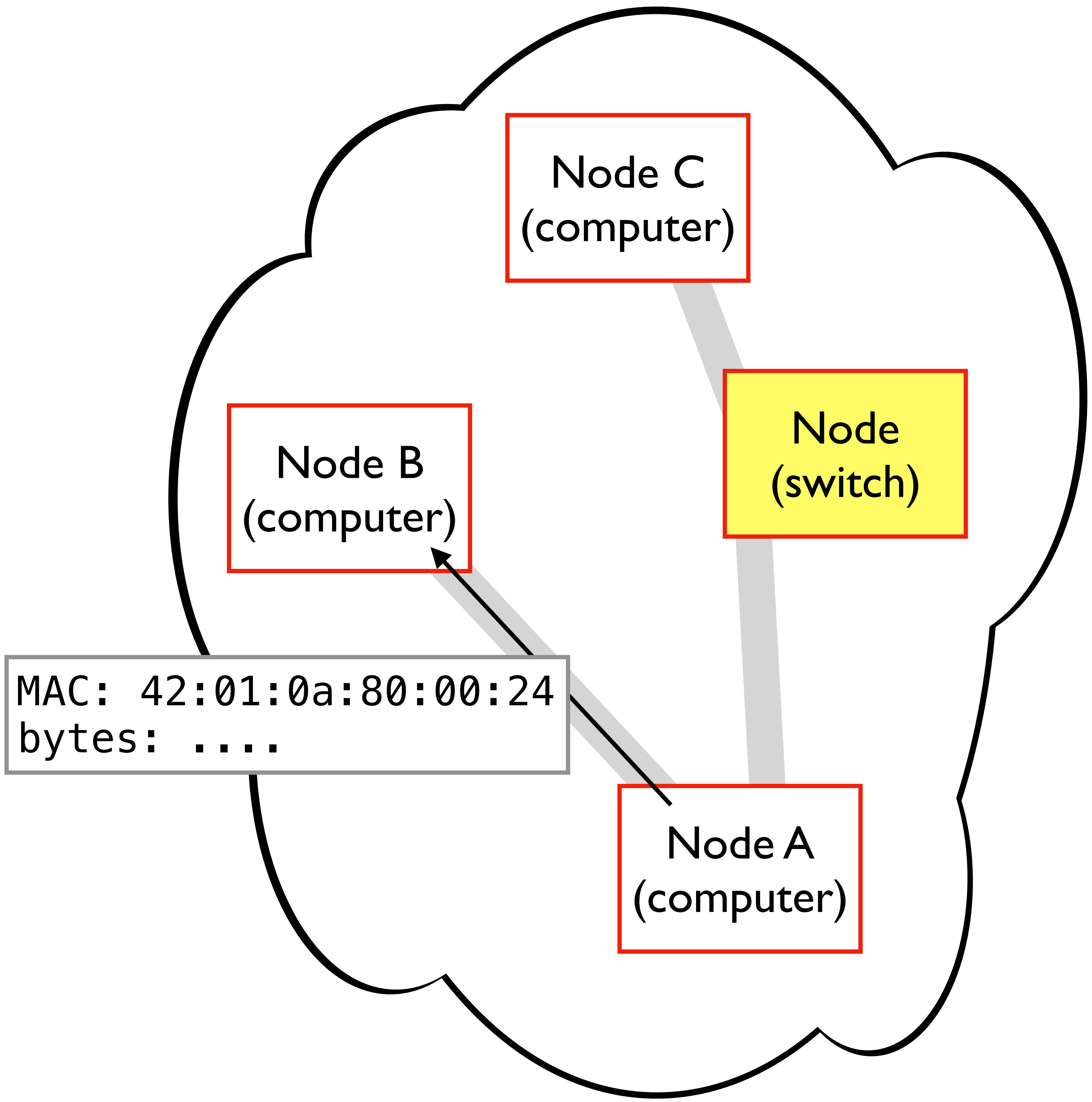
```
trh@instance-2:~$ ip address
ens4: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1460
        inet 10.128.0.37  netmask 255.255.255.255  broadcast 0.0.0.0
        inet6 fe80::4001:aff:fe80:25  prefixlen 64  scopeid 0x20<link>
        ether 42:01:0a:80:00:25  txqueuelen 1000  (Ethernet)
        RX packets 637  bytes 546000 (546.0 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 612  bytes 97265 (97.2 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

ens5: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1460
        inet 10.0.0.2  netmask 255.255.255.255  broadcast 0.0.0.0
        inet6 fe80::4001:aff:fe00:2  prefixlen 64  scopeid 0x20<link>
        ether 42:01:0a:00:00:02  txqueuelen 1000  (Ethernet)
        RX packets 51  bytes 9955 (9.9 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 61  bytes 6834 (6.8 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        inet6 ::1  prefixlen 128  scopeid 0x10<host>
        loop  txqueuelen 1000  (Local Loopback)
        RX packets 120  bytes 13534 (13.5 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 120  bytes 13534 (13.5 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

34.29.220.248 (nic0)
35.202.74.234 (nic1)

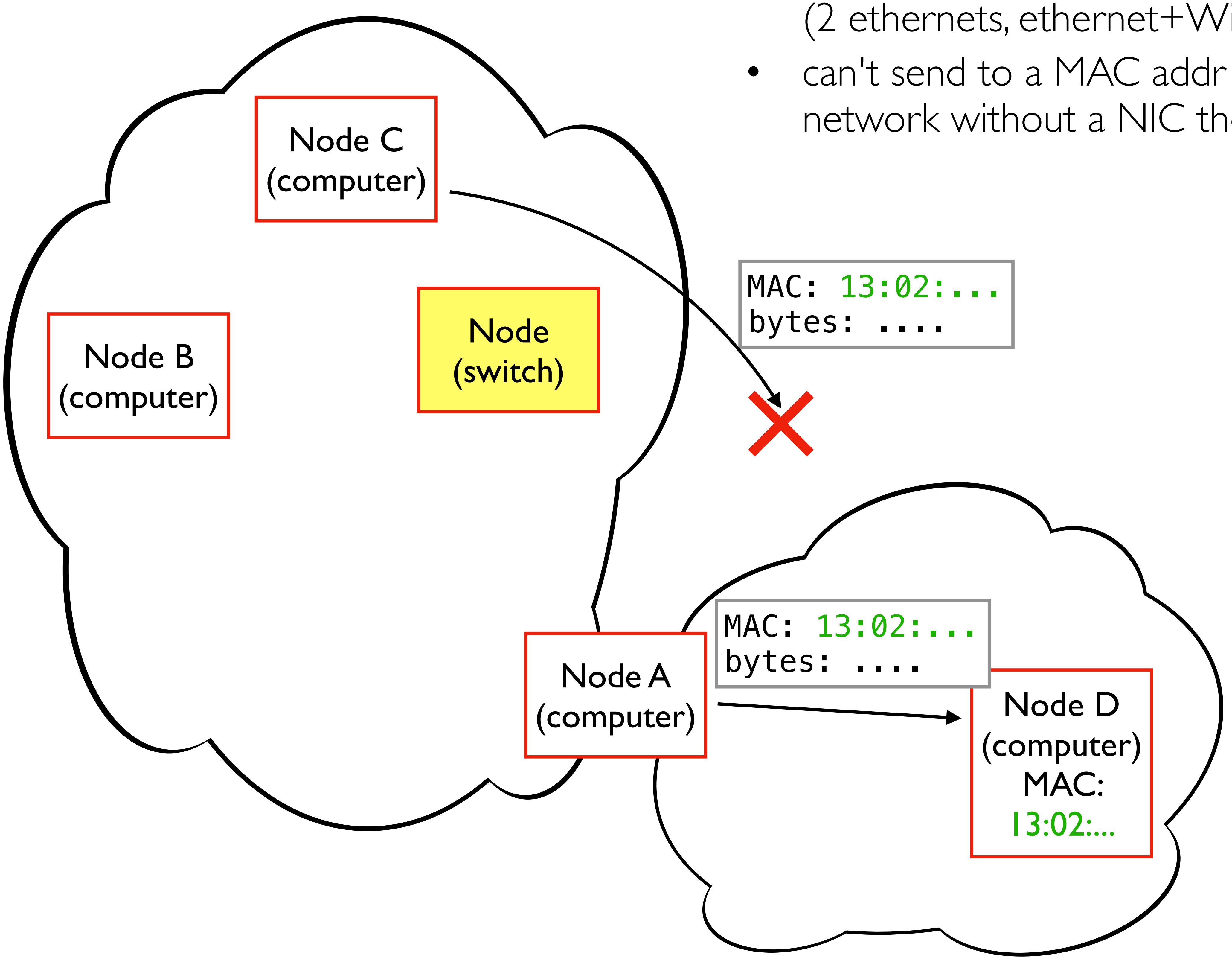
Networks



A network has nodes that send bytes to other nodes **by MAC address**

- **nodes**: computer, switch, etc
- direct, or **forwarded by switches**
- whole network uses same physical tech (Wi-Fi, Ethernet, etc)

Networks



Computers can have multiple NICs

- can be on multiple networks (2 ethernets, ethernet+Wi-FI, etc)
- can't send to a MAC addr in another network without a NIC there

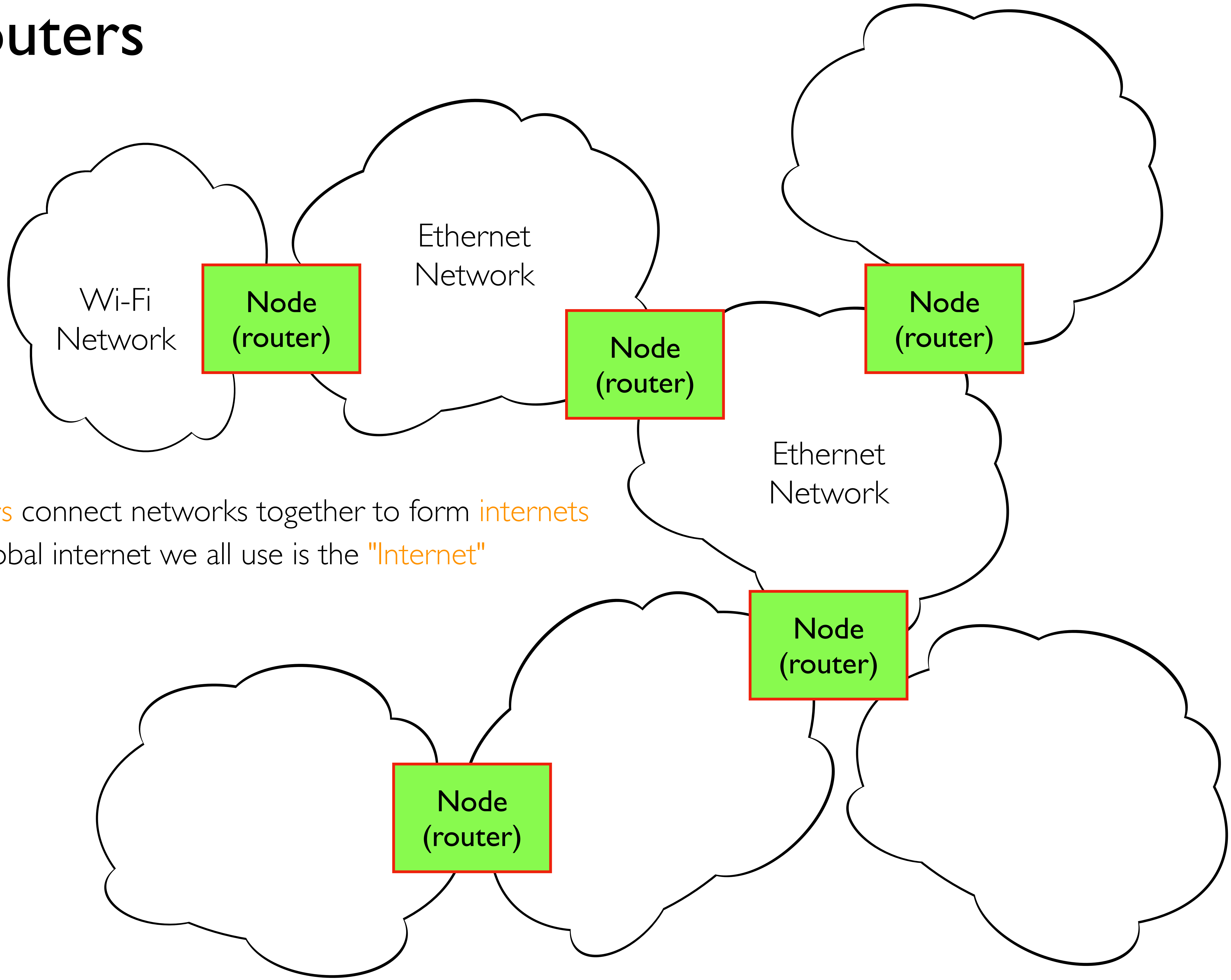
Outline

Networks

Internets and "The Internet"

Transport Protocols

Routers

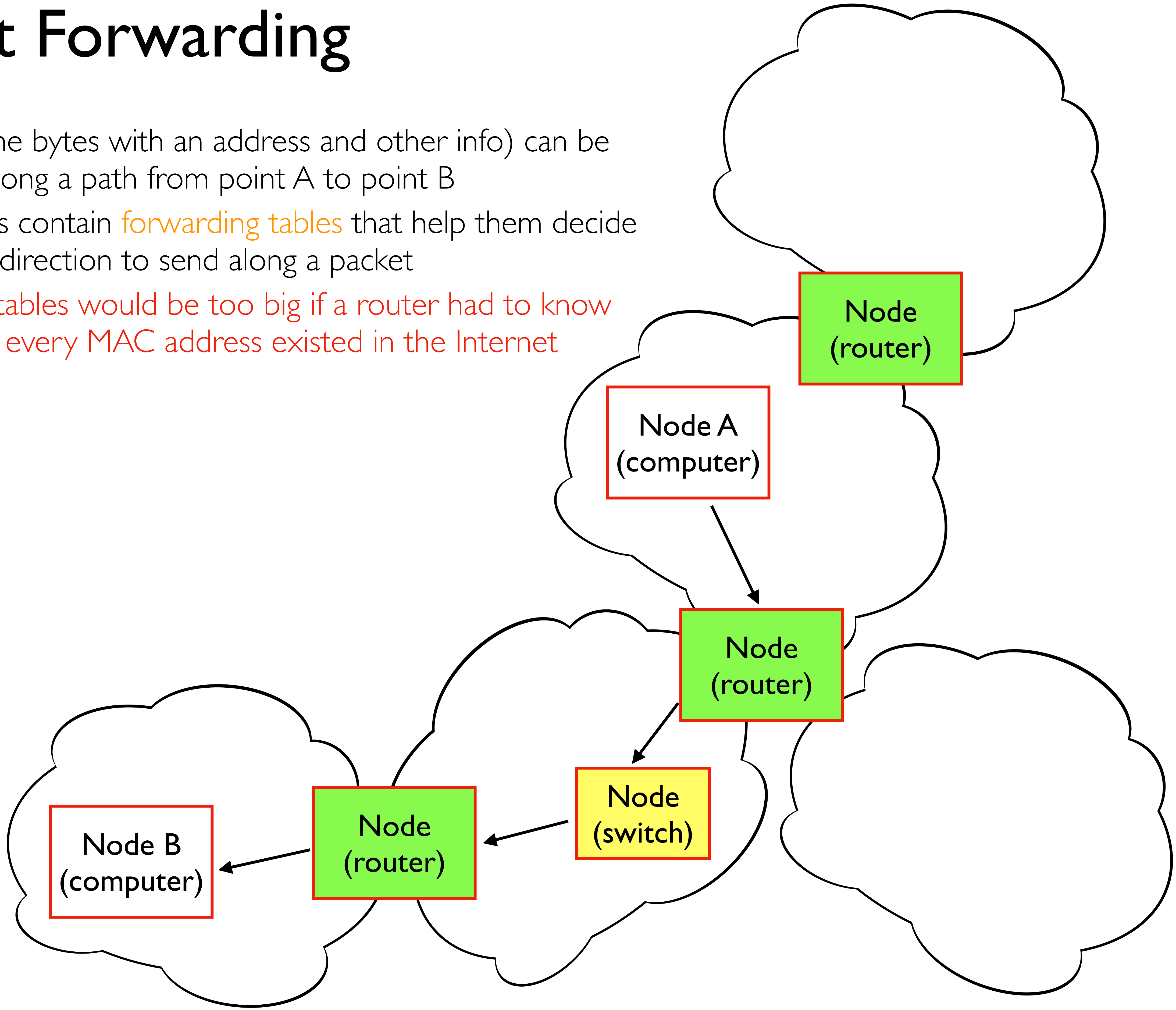


- routers connect networks together to form internets
- the global internet we all use is the "Internet"

Packet Forwarding

Packets (some bytes with an address and other info) can be forwarded along a path from point A to point B

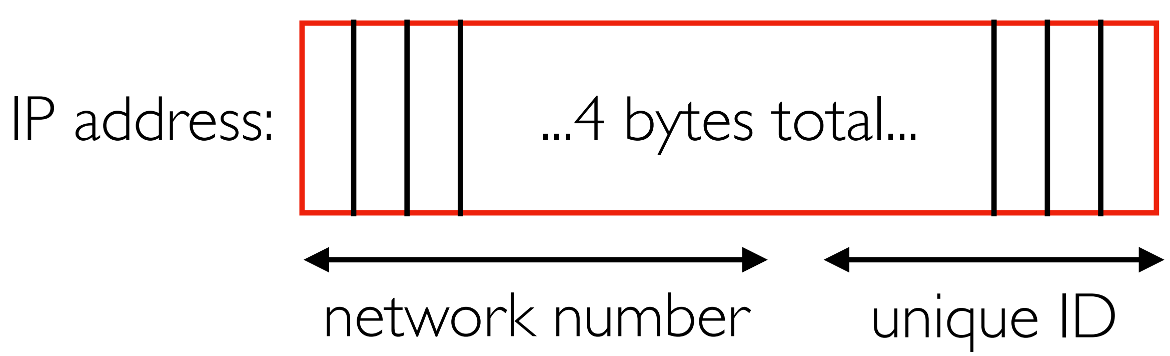
- routers contain forwarding tables that help them decide which direction to send along a packet
- those tables would be too big if a router had to know where every MAC address existed in the Internet

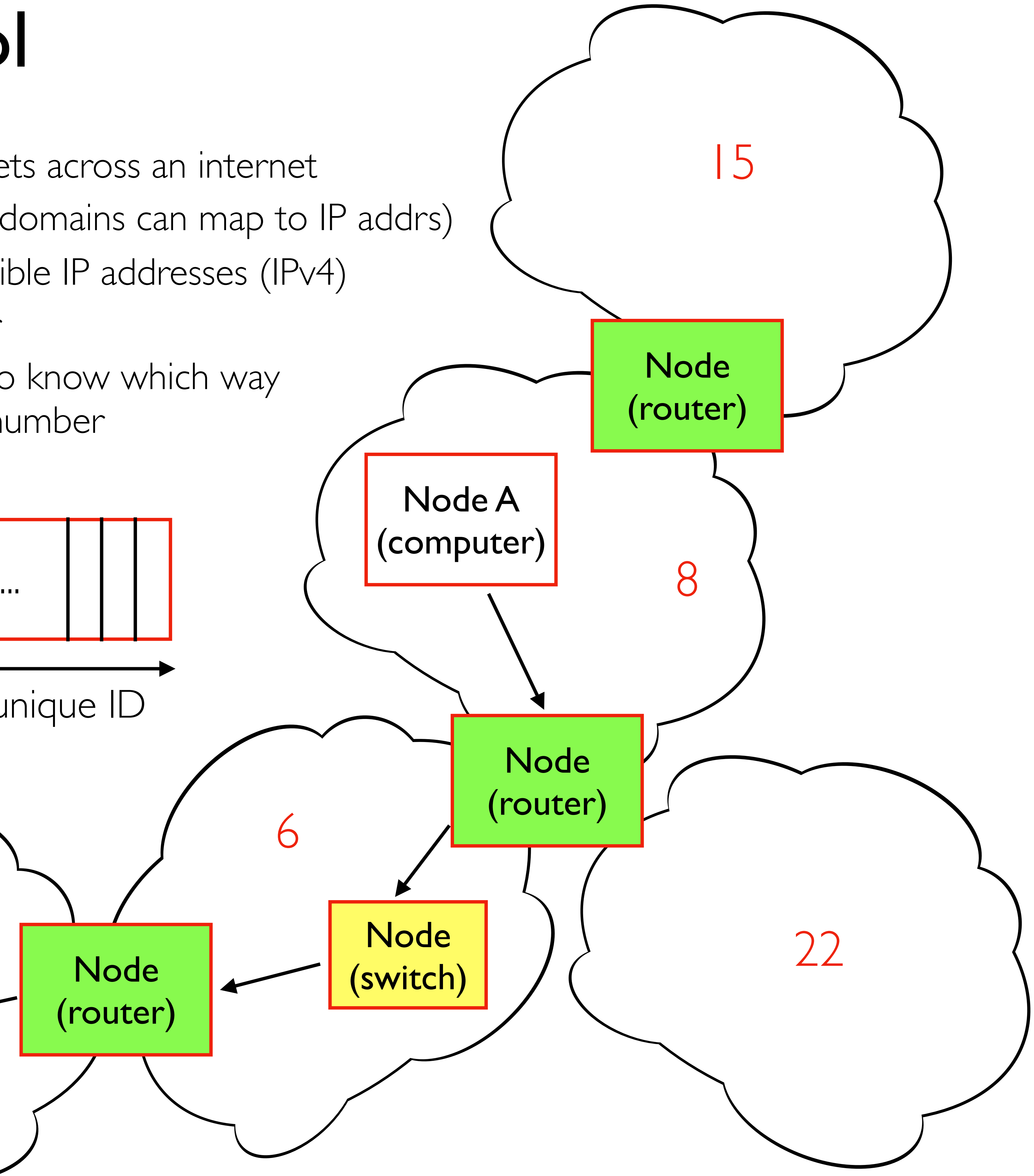


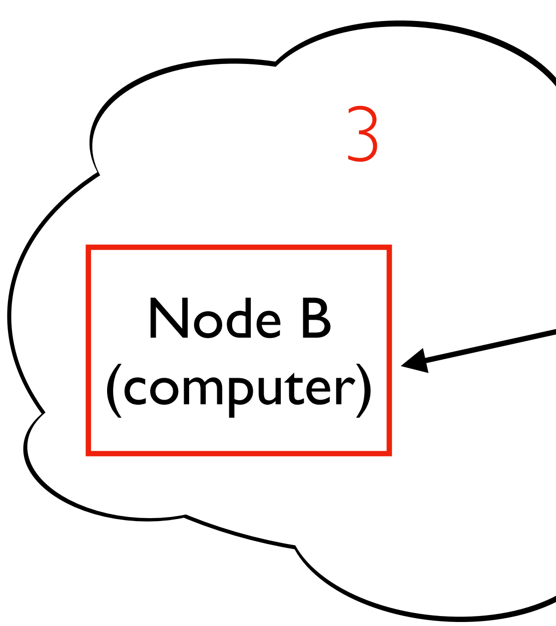
Internet Protocol

IP addresses are used to send packets across an internet

- example: `34.29.237.29` (domains can map to IP addrs)
- there are about 4 billion possible IP addresses (IPv4)
- IPv6 (less used) are 4x longer
- forwarding tables only need to know which way to send for a given network number







Listening on an Interface

```
trh@instance-2:~$ ip address
ens4: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1460
        inet 10.0.1.2  netmask 255.255.255.255  broadcast 0.0.0.0
        inet6 fe80
        python3 -m http.server --bind 10.0.1.2
        0<link>
        ether 42:01
        ...

ens5: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1430
        inet 10.0.3.2  netmask 255.255.255.255  broadcast 10.0.3.2
        inet6 fe80
        python3 -m http.server --bind 10.0.3.2
        0<link>
        ether 42:01
        ...

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        inet6 ::1
        python3 -m http.server --bind 127.0.0.1
        loop  txqu
        ...
```

all of them:

```
python3 -m http.server --bind 0.0.0.0
```

Private Networks

Challenges

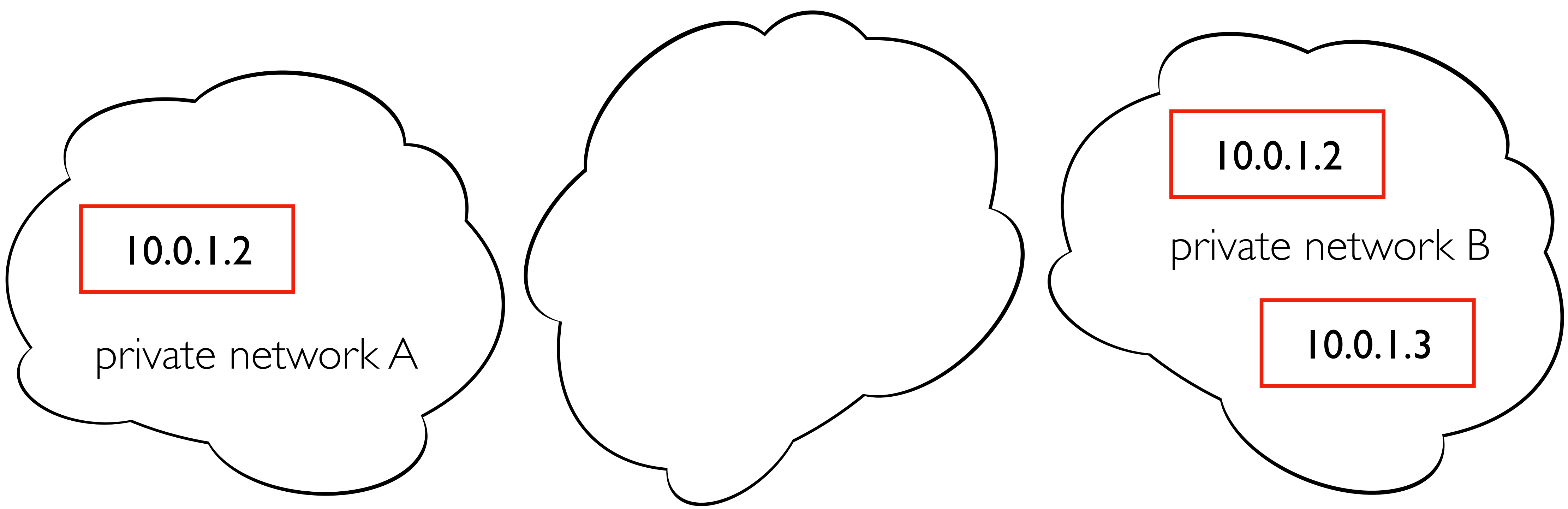
- we don't have enough IPv4 addresses
- we don't want every machine to be able to receive packets from anywhere

Private ranges:

- `192.168.0.0 to 192.168.255.255`
- `172.16.0.0 to 172.31.255.255`
- `10.0.0.0 to 10.255.255.255`

these can be divided into "sub networks" (subnets) to create different networks in a bigger org

Private networks allow duplicates and unreachable machines



Private Networks

```
trh@instance-2:~$ ip address
ens4: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1460
        inet 10.0.1.2
        inet6 fe80::40                                                  nk>
        ether 42:01:0a:00:01:02  txqueuelen 1000  (Ethernet)
        ...
```

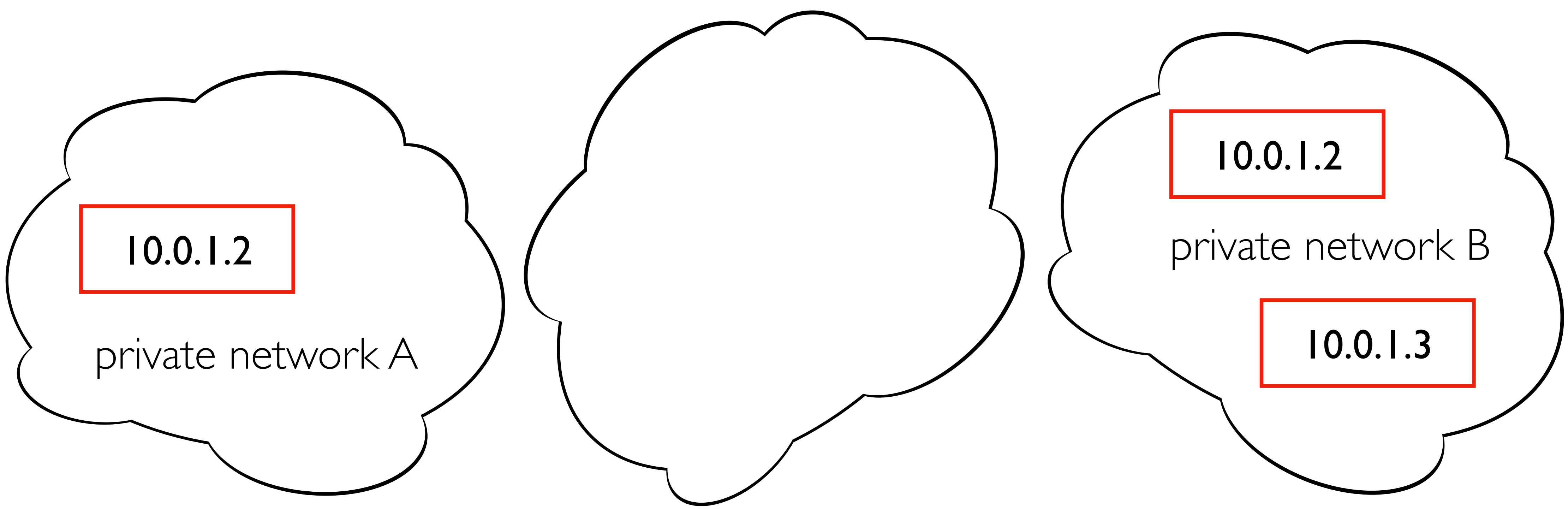
```
python3 -m http.server --bind 10.0.1.2
```

Private ranges:

- 192.168.0.0 to 192.168.255.255
- 172.16.0.0 to 172.31.255.255
- 10.0.0.0 to 10.255.255.255

http://10.0.1.2:...
won't work in web browser!

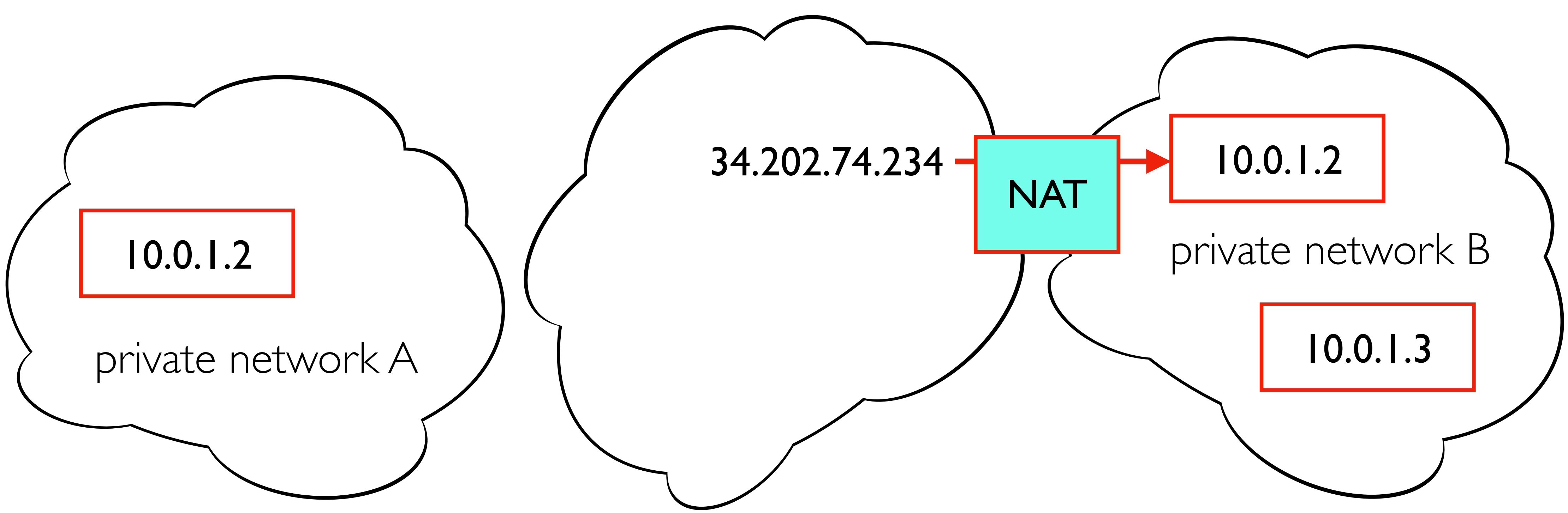
Private networks allow duplicates and unreachable machines



Network Address Translation

Google Console (view NAT config)

☐	Status	Name ↑	Internal IP	External IP
☐	✅	instance-1	10.128.0.36 (nic0)	34.29.237.29 (nic0)
☐	✅	instance-2	10.0.1.2 (nic0) 10.0.3.2 (nic1)	35.202.74.234 (nic0) 34.29.220.248 (nic1)

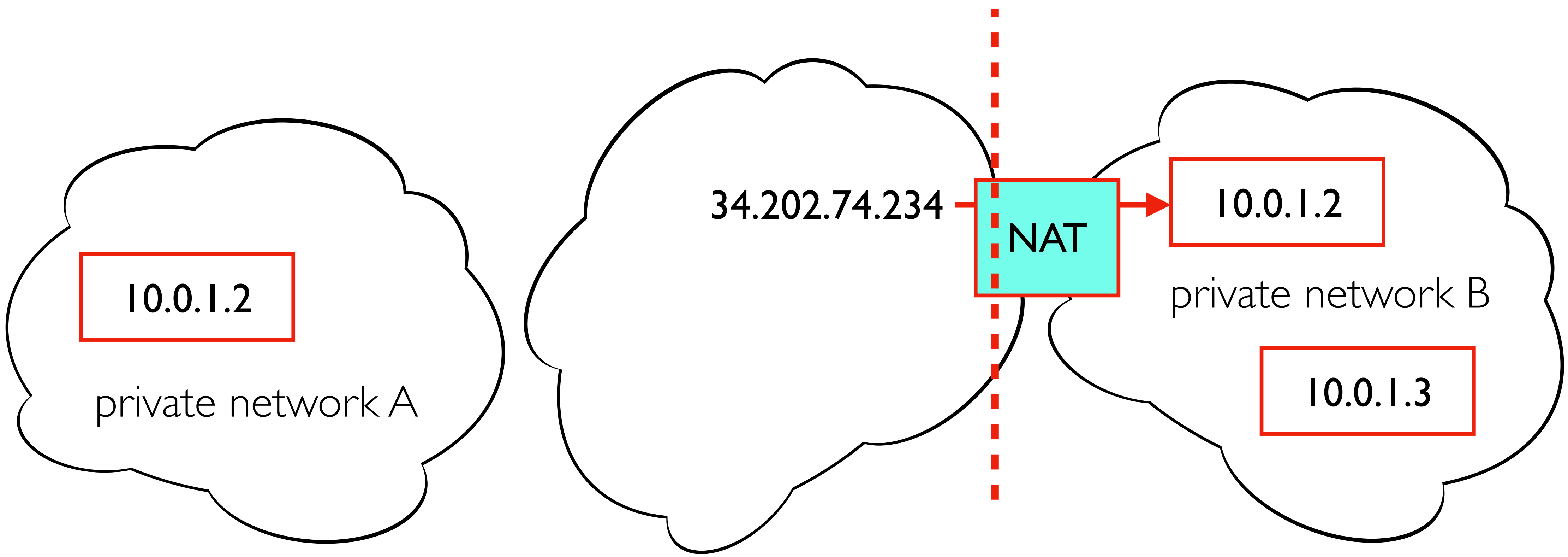


Network Address Translation

Google Console (view NAT config)

	Status	Name ↑	Internal IP	External IP
☐	✔	instance-1	10.128.0.36 (nic0)	34.29.237.29 (nic0)
☐	✔	instance-2	10.0.1.2 (nic0) 10.0.3.2 (nic1)	35.202.74.234 (nic0) 34.29.220.248 (nic1)

a firewall can limit what traffic is allowed

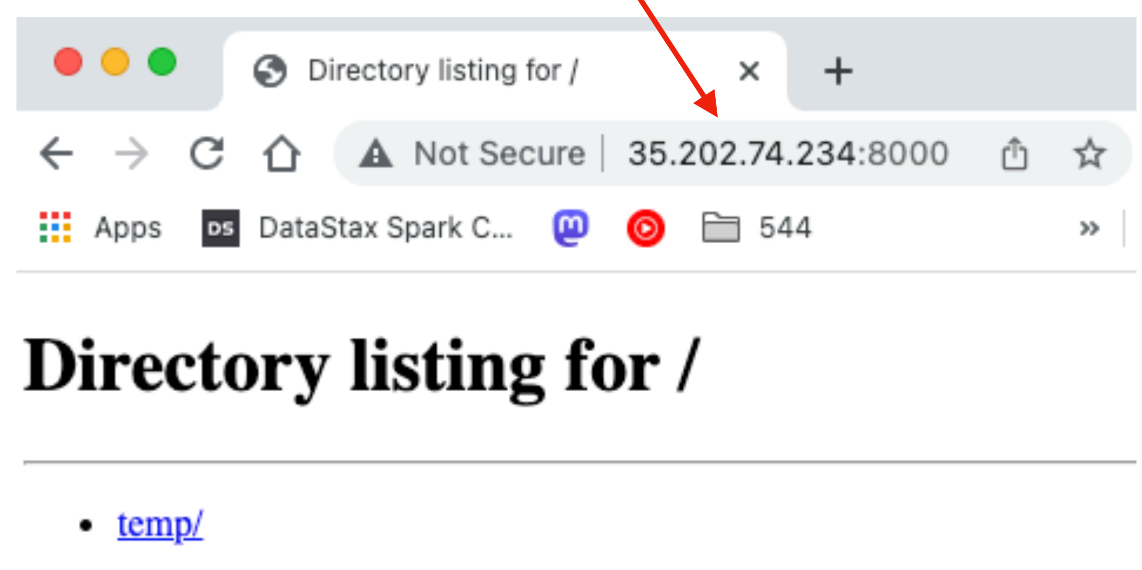


Network Address Translation

Google Console (view NAT config)

	Status	Name ↑	Internal IP	External IP
☐	✓	instance-1	10.128.0.36 (nic0)	34.29.237.29 (nic0)
☐	✓	instance-2	10.0.1.2 (nic0) 10.0.3.2 (nic1)	35.202.74.234 (nic0) 34.29.220.248 (nic1)

Browser



By default, the external IPs are "ephemeral" (change upon reboot). You can rent "static" IPs that don't change.

Server

```
trh@instance-2:~/temp$ python3 -m http.server --bind 10.0.1.2
Serving HTTP on 10.0.1.2 port 8000 (http://10.0.1.2:8000/) ...
72.33.0.184 - - [10/Feb/2023 21:12:53] "GET / HTTP/1.1" 200 -
...
```

Outline

Networks

Internets and "The Internet"

Transport Protocols

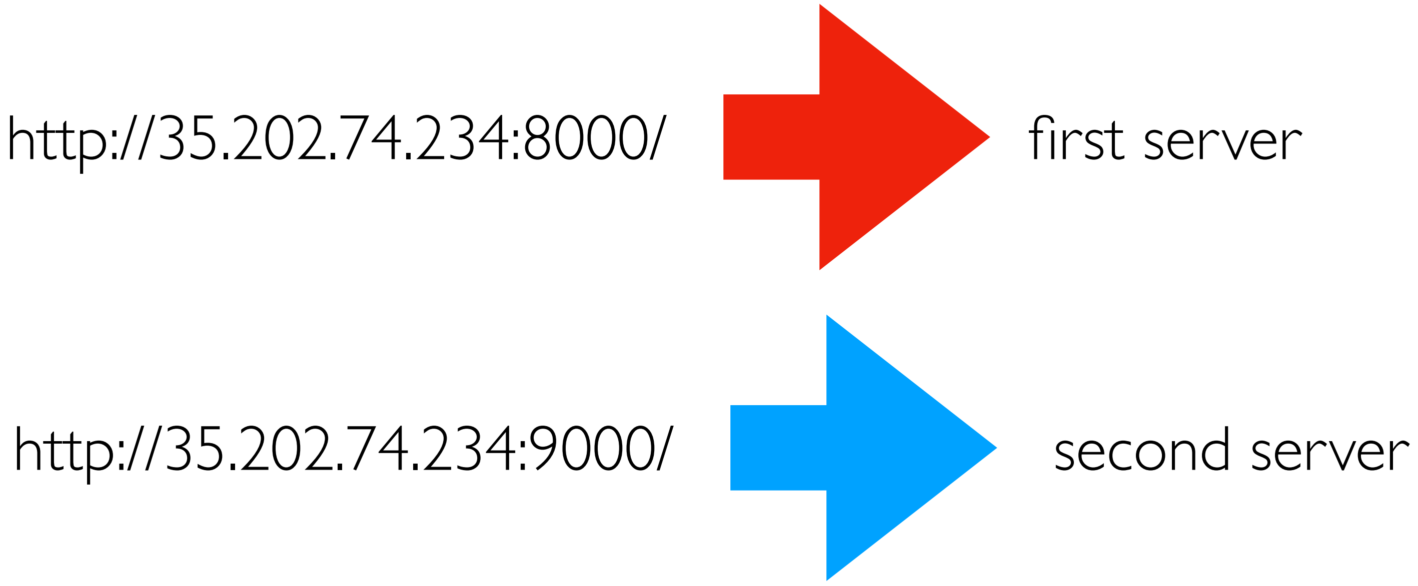
Port Numbers

Computers might be running multiple processes using the network

- IP address => which NIC?
- Port number => which process?

```
trh@instance-2:~$ python3 -m http.server --directory=A --bind 10.0.1.2 8000 &
[1] 13502
Serving HTTP on 10.0.1.2 port 8000 (http://10.0.1.2:8000/) ...

trh@instance-2:~$ python3 -m http.server --directory=B --bind 10.0.1.2 9000 &
[2] 13503
Serving HTTP on 10.0.1.2 port 9000 (http://10.0.1.2:9000/) ...
```



TopHat...

Transport Protocols

Most common

- UDP (User Datagram Protocol)
- TCP (Transmission Control Protocol)

BOTH build on IP networking and BOTH provide port numbers

-t: tcp, -u: udp

```
trh@instance-2:~/temp$ sudo ss -tlpn
State       Local Address:Port    Peer Address:Port     Process
LISTEN      10.128.0.4:8000       0.0.0.0:*             users:(("python3",...))
LISTEN      10.128.0.4:9000       0.0.0.0:*             users:(("python3",...))
LISTEN      *:22                  *:*
```

Reliability: UDP vs. TCP

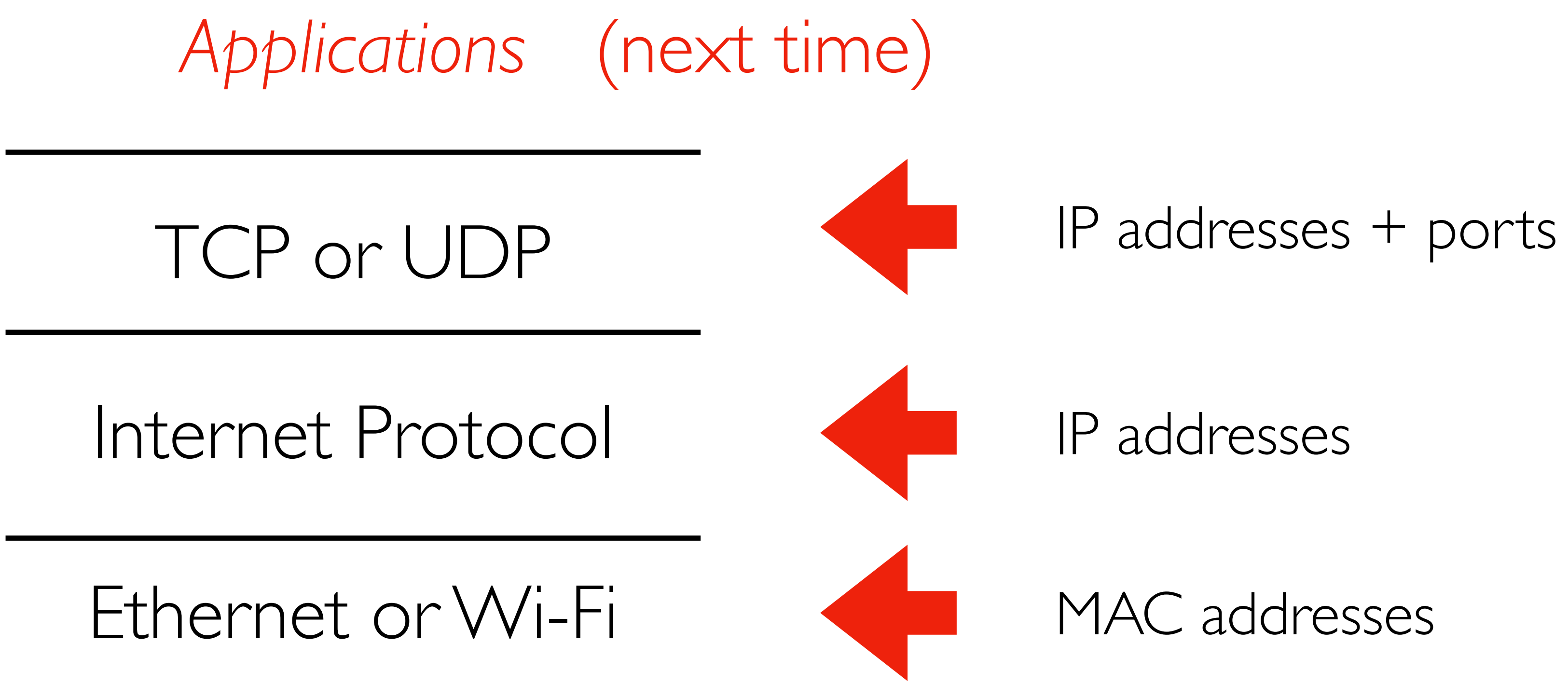
Packets may be

- dropped
- reordered
- split

TCP saves+reassembles packets in order to provide original message (when possible). For packet drops, it retries. We'll mostly use TCP.

UDP doesn't do this extra work. Why ever use UDP?

Network Stack: Common Implementations



Network applications (like most complex systems) are not built as one single system. Layers are built upon other layers to provide additional functionality.