Computer Organization and Networks

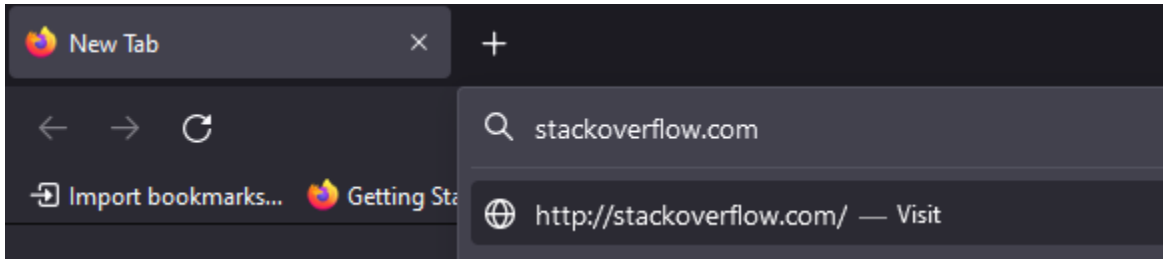
Chapter 8: Networking I

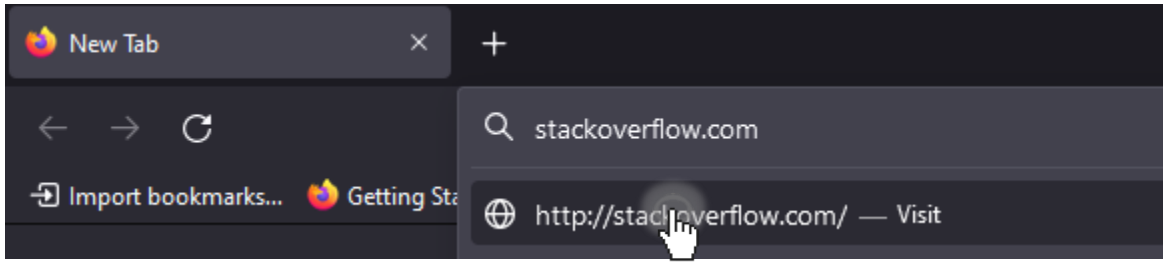
Winter 2021/2022

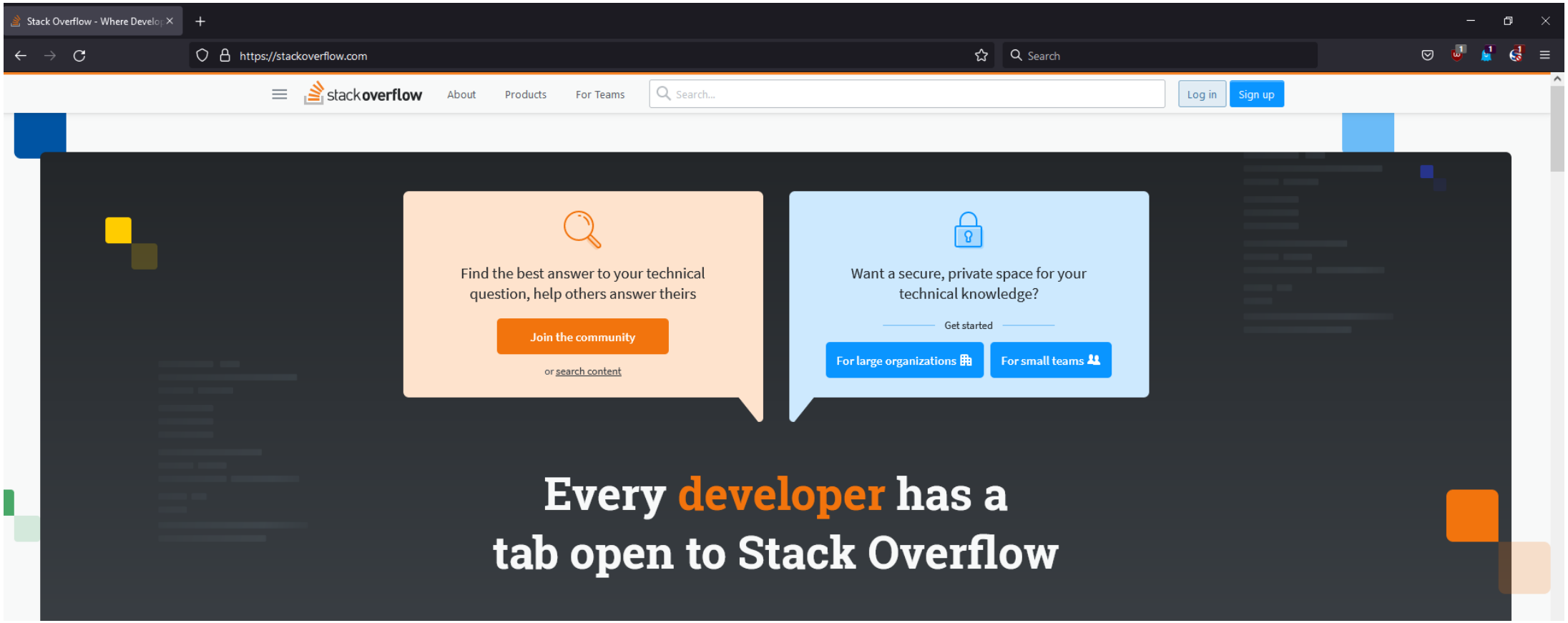
Jakob Heher, www.iaik.tugraz.at
he/his

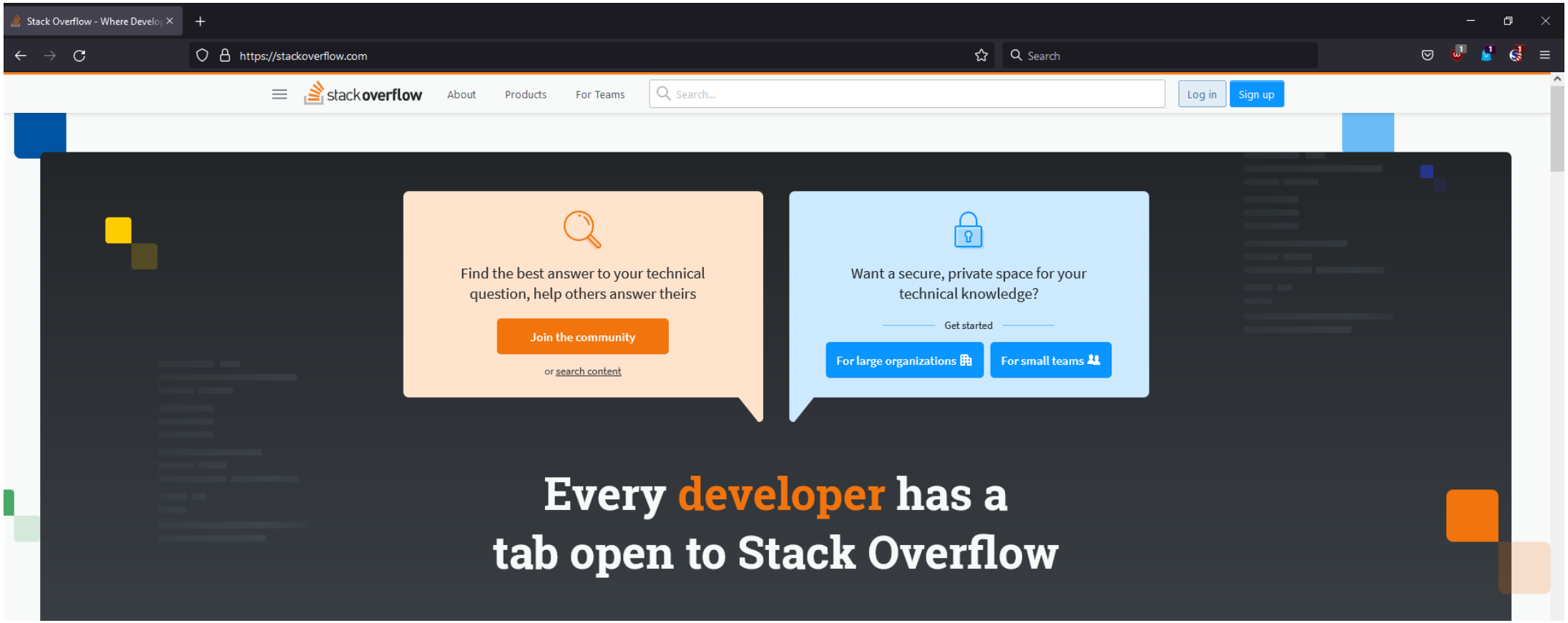
Motivation

- You’ve built a CPU
  - Now let’s make it talk to others

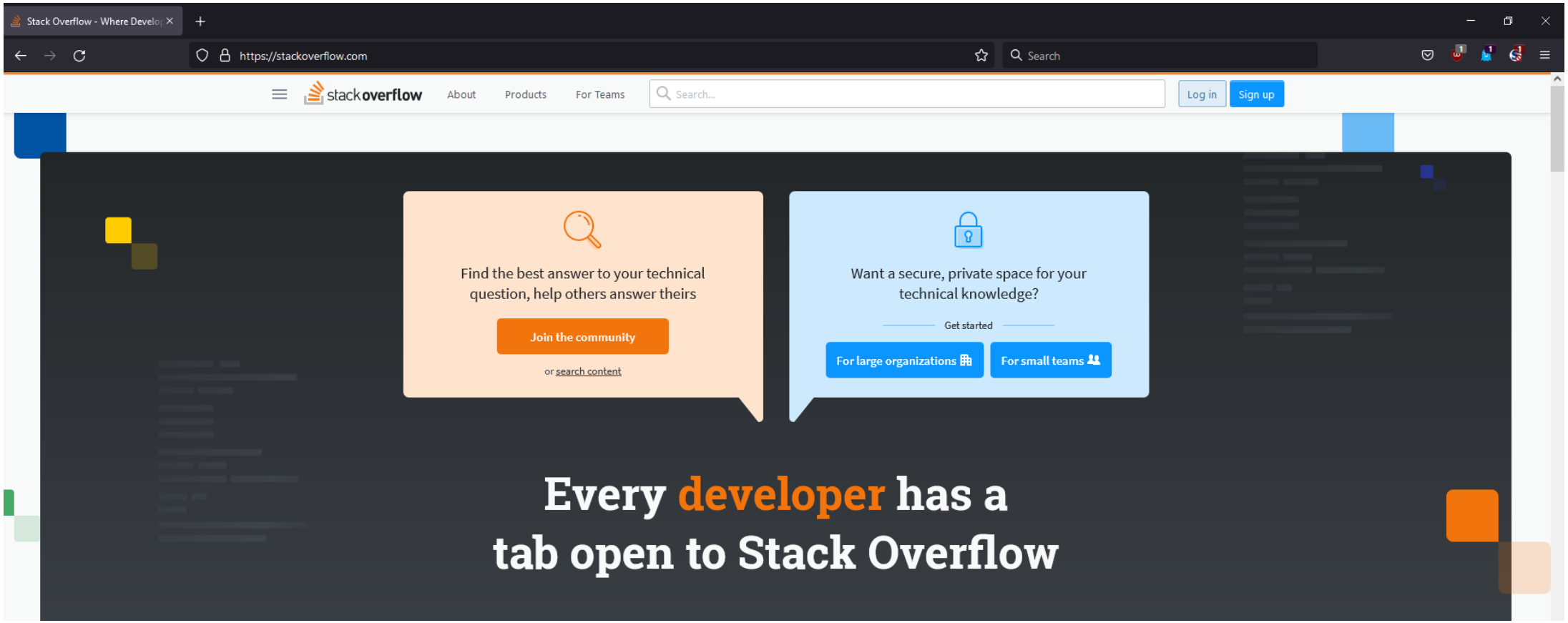

New Tab
stackoverflow.com
Import bookmarks...
Getting Sta
http://stackoverflow.com/ — Visit


New Tab
stackoverflow.com
Import bookmarks...
Getting Sta
http://stackoverflow.com/ — Visit

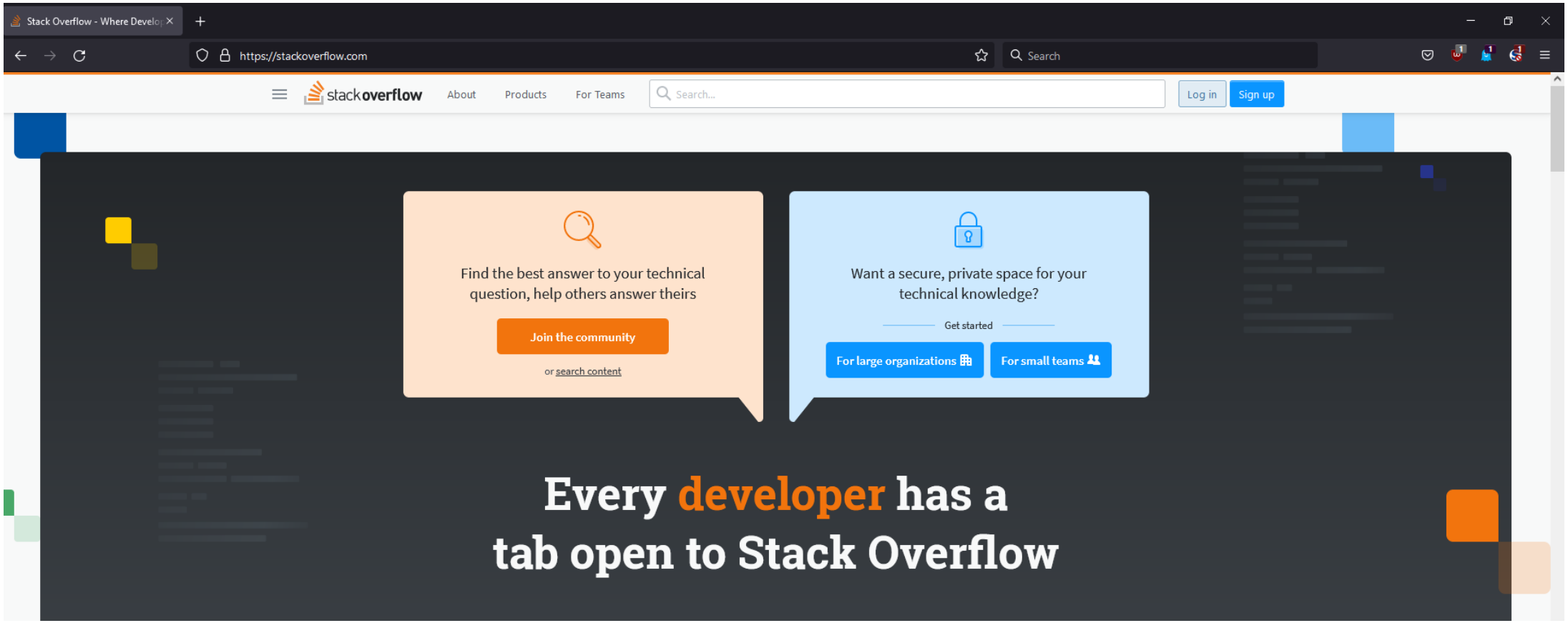

Stack Overflow - Where Develop
https://stackoverflow.com
Search
stackoverflow
About
Products
For Teams
Search...
Log in
Sign up
Find the best answer to your technical question, help others answer theirs
Join the community
or search content
Want a secure, private space for your technical knowledge?
Get started
For large organizations
For small teams
Every developer has a tab open to Stack Overflow



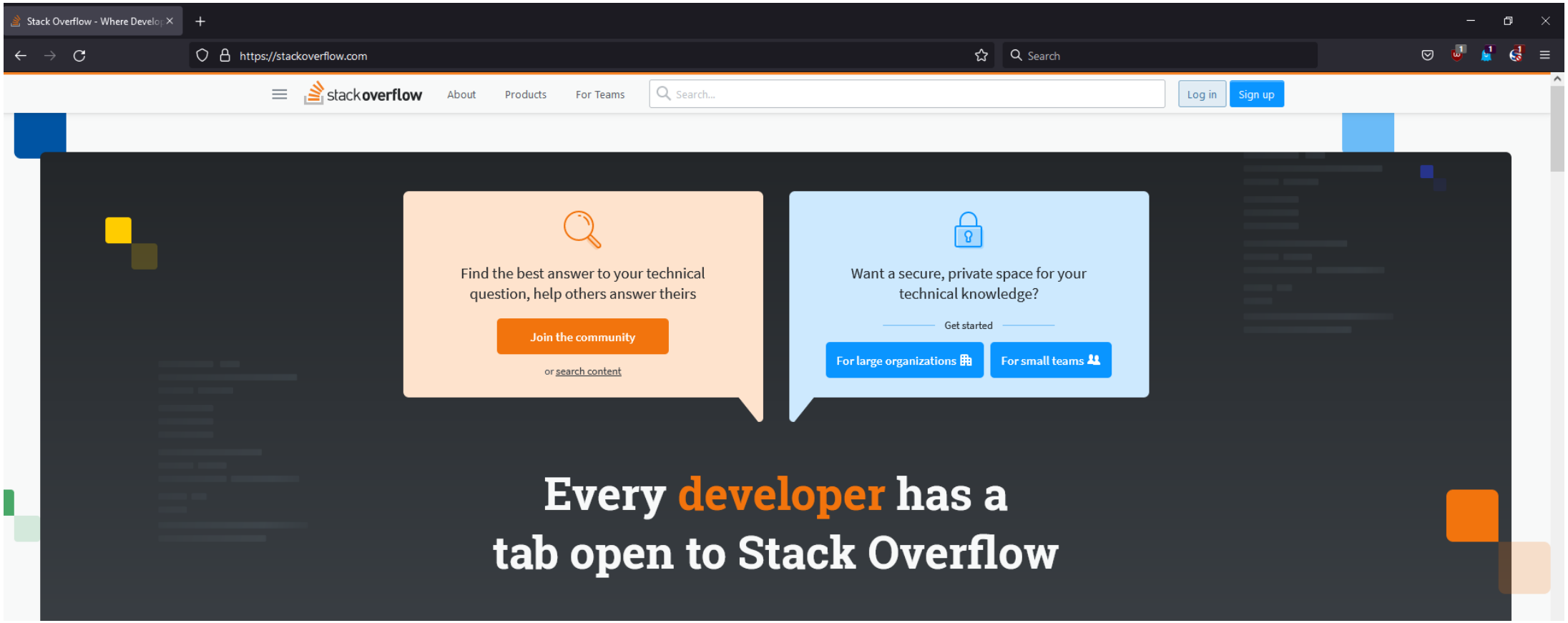
- OK, what did we just do?



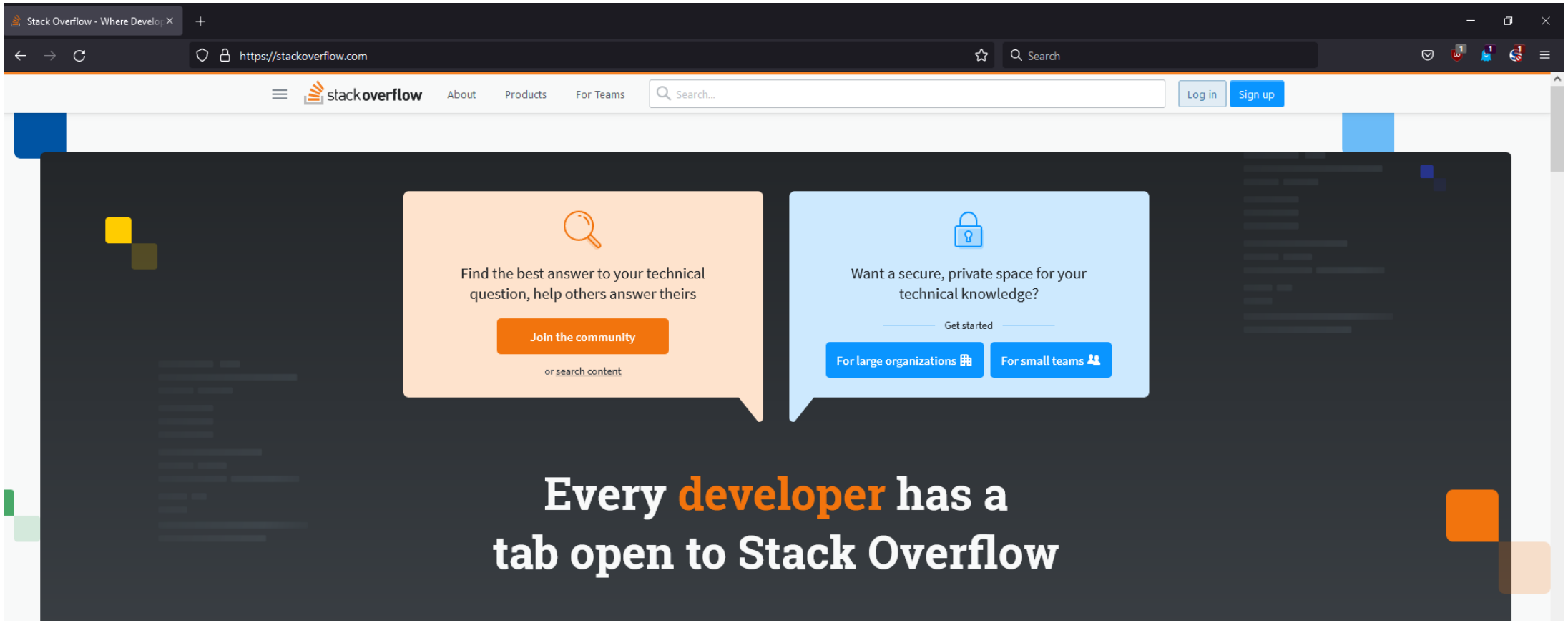
- OK, what did our browser just do?



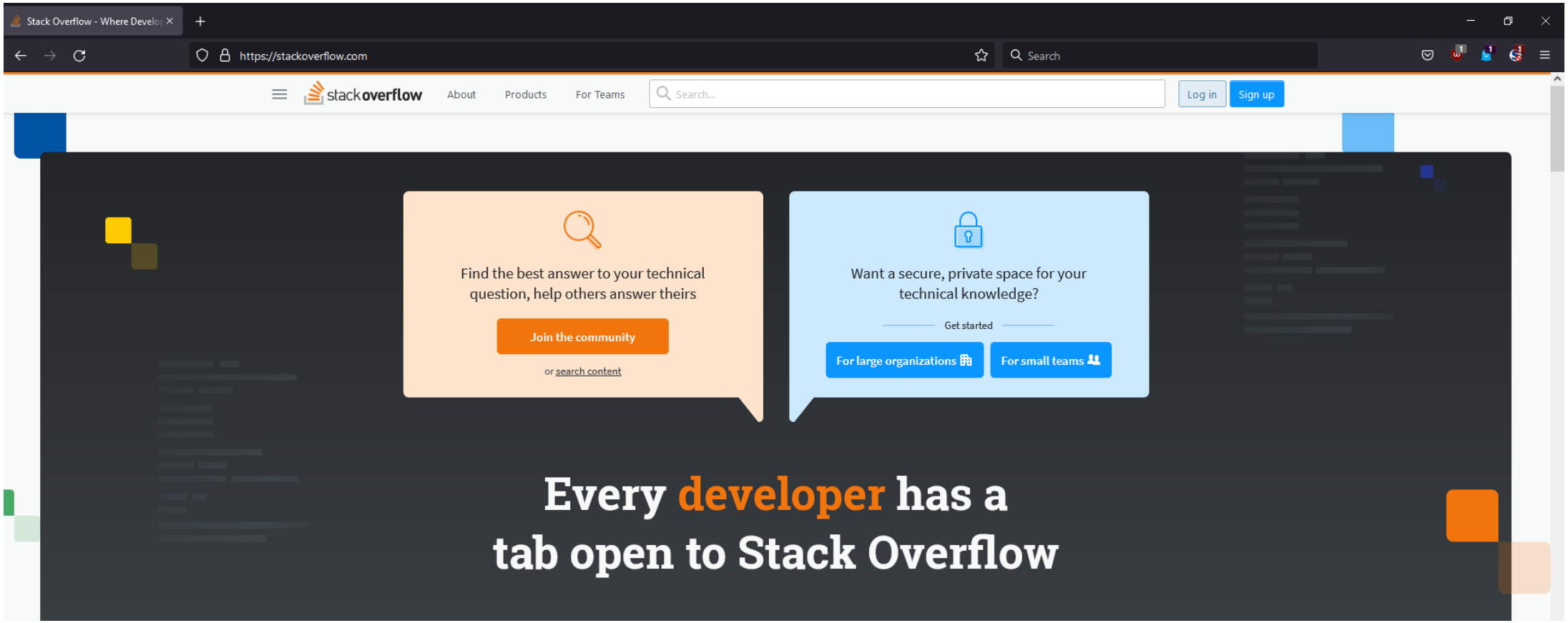
- OK, what did our OS just do?



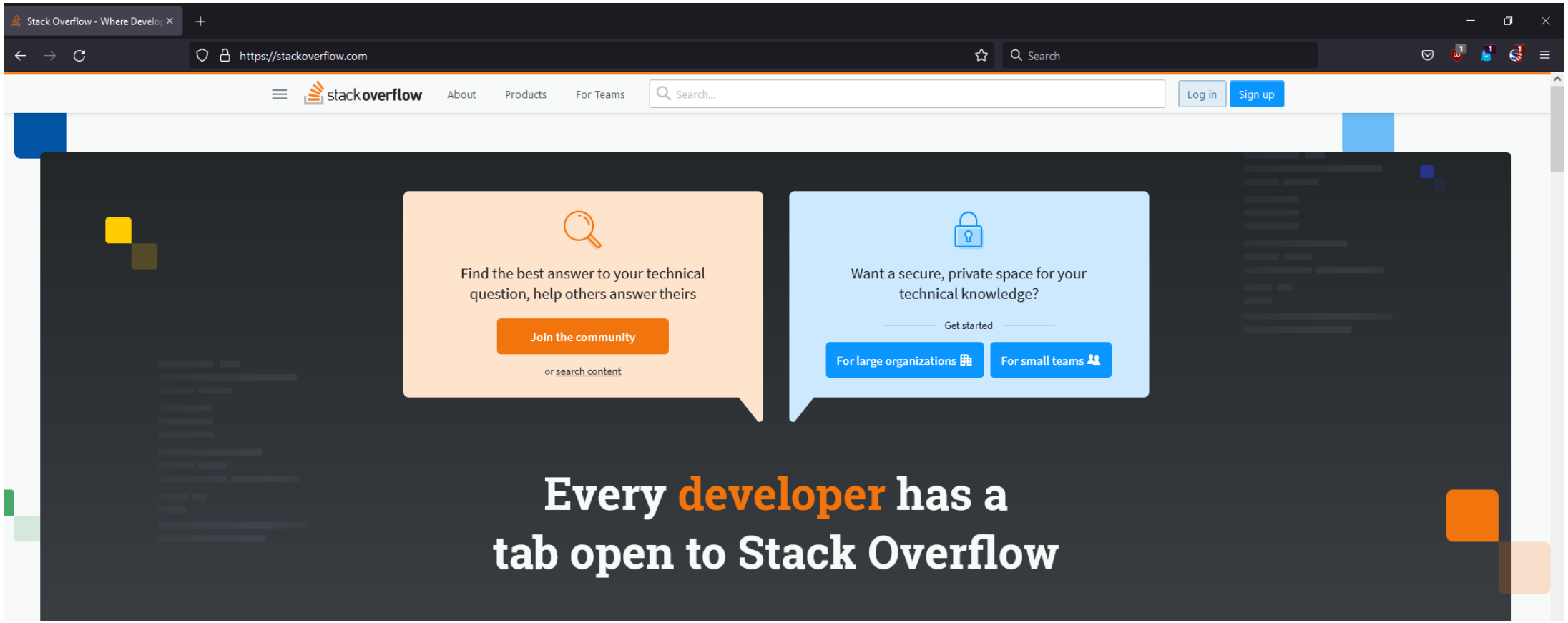
- OK, what did our network card just do?



- OK, what did our router just do?



- OK, what did the StackOverflow server just do?



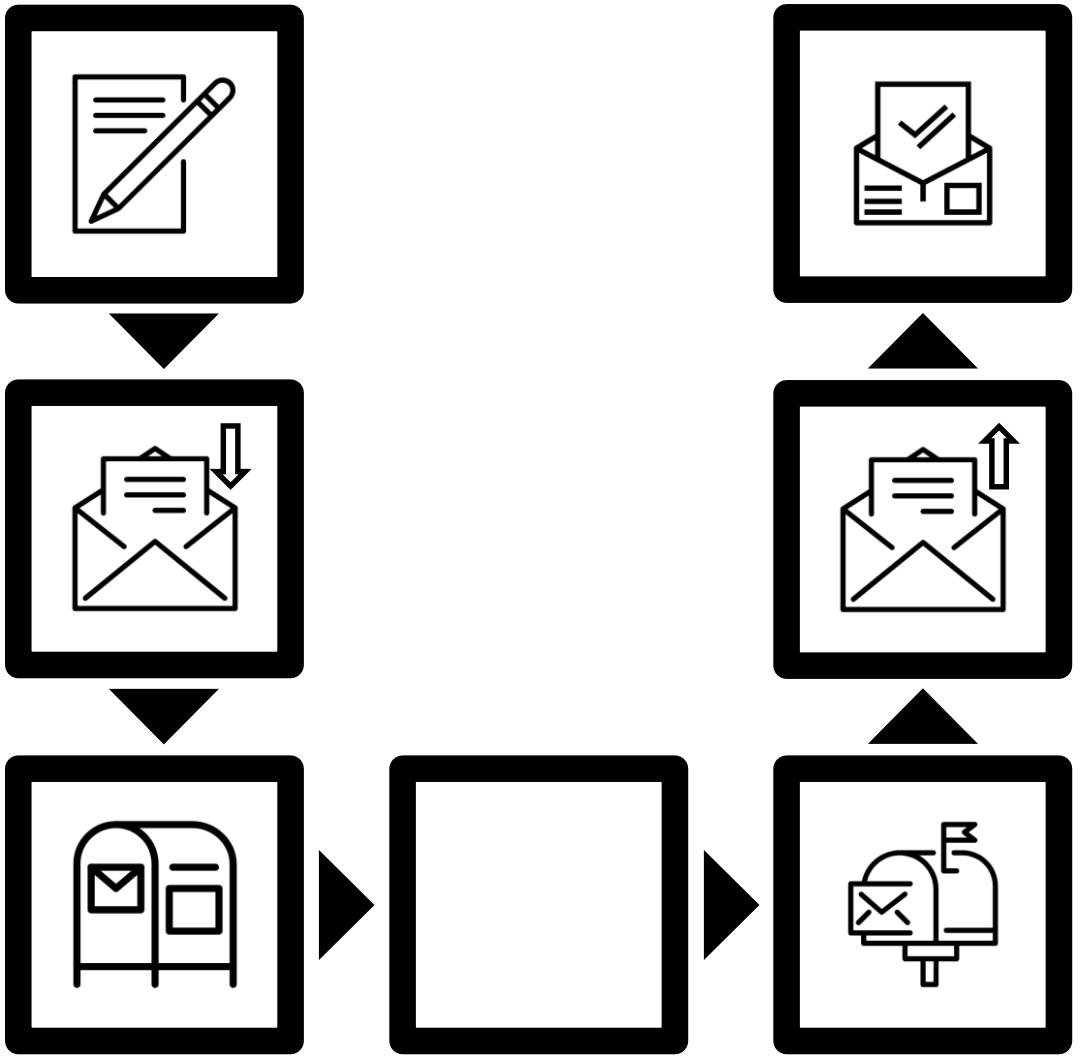
- By the time we're done here, you'll know!

Layers & Abstraction

- How do you send a postcard?

Layers & Abstraction

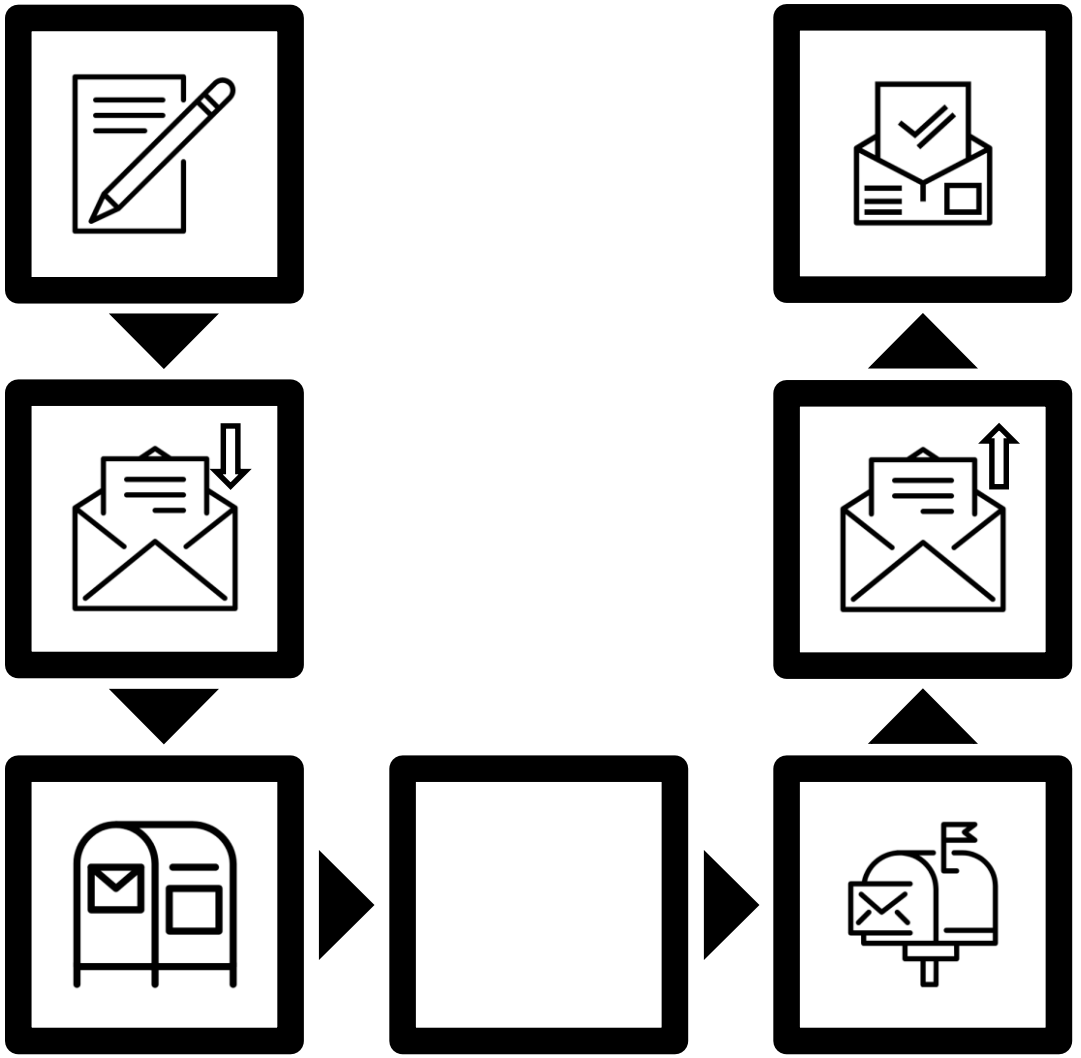
- How do you send a postcard?
  1. Write postcard
  2. Put postcard in envelope
  3. Mail envelope to recipient
  4. Recipient receives envelope
  5. Recipient opens envelope
  6. Recipient reads postcard



Layers & Abstraction

- How do you send a postcard?
  1. Write postcard
  2. Put postcard in envelope
  3. **Mail envelope to recipient**
  4. **Recipient receives envelope**
  5. Recipient opens envelope
  6. Recipient reads postcard

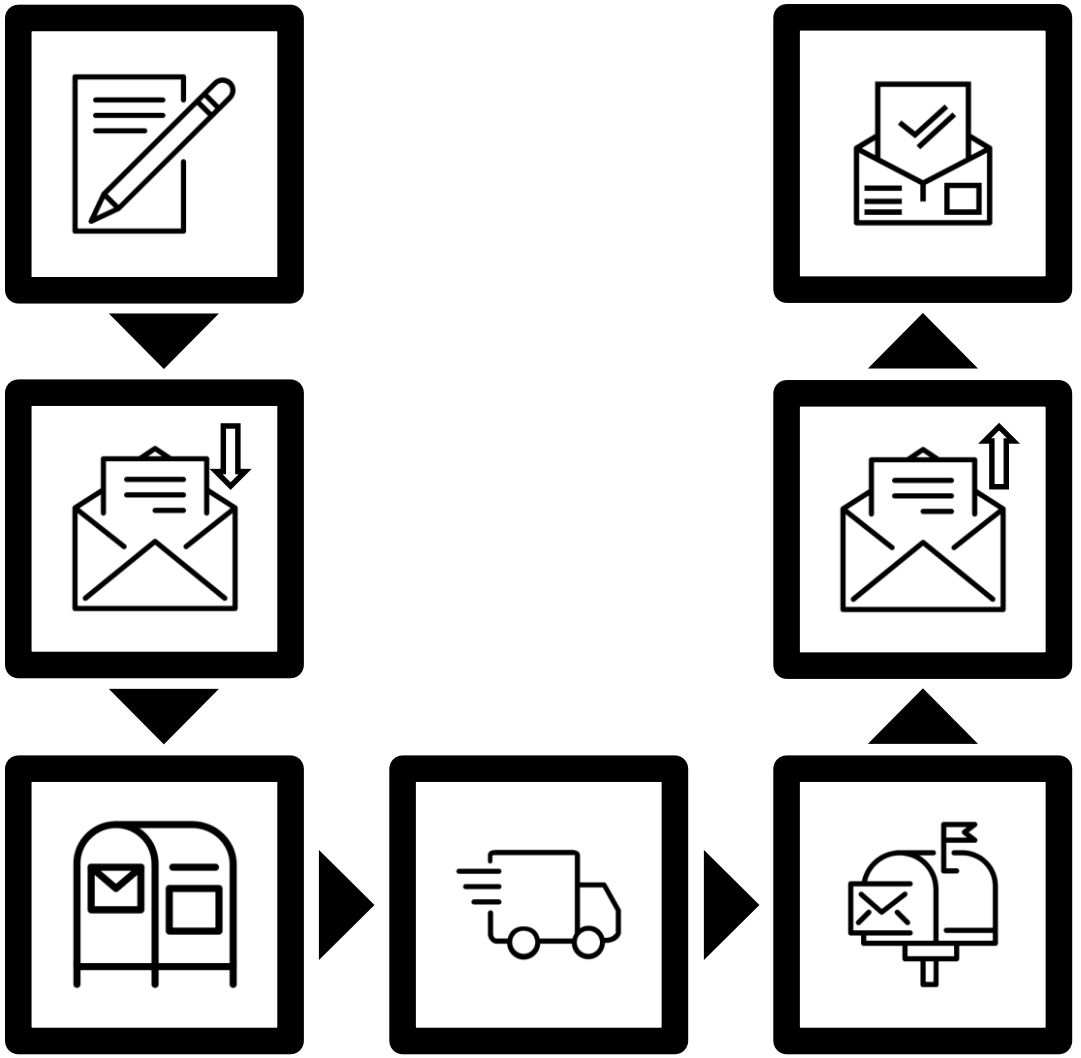
- How does the envelope get there?



Layers & Abstraction

- How do you send a postcard?
  1. Write postcard
  2. Put postcard in envelope
  3. **Mail envelope to recipient**
  4. **Recipient receives envelope**
  5. Recipient opens envelope
  6. Recipient reads postcard

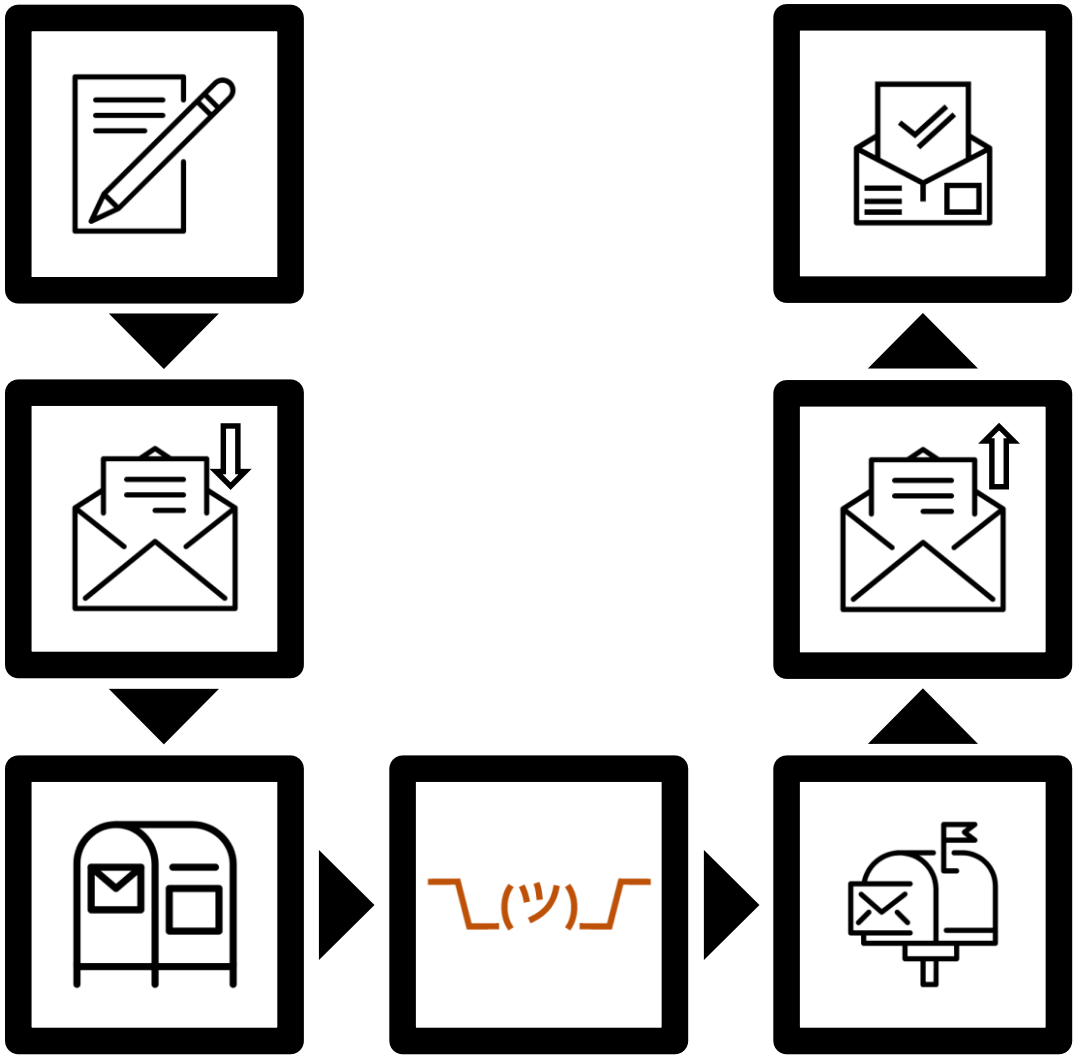
- How does the envelope get there?



Layers & Abstraction

- How do you send a postcard?
  1. Write postcard
  2. Put postcard in envelope
  3. **Mail envelope to recipient**
  4. **Recipient receives envelope**
  5. Recipient opens envelope
  6. Recipient reads postcard

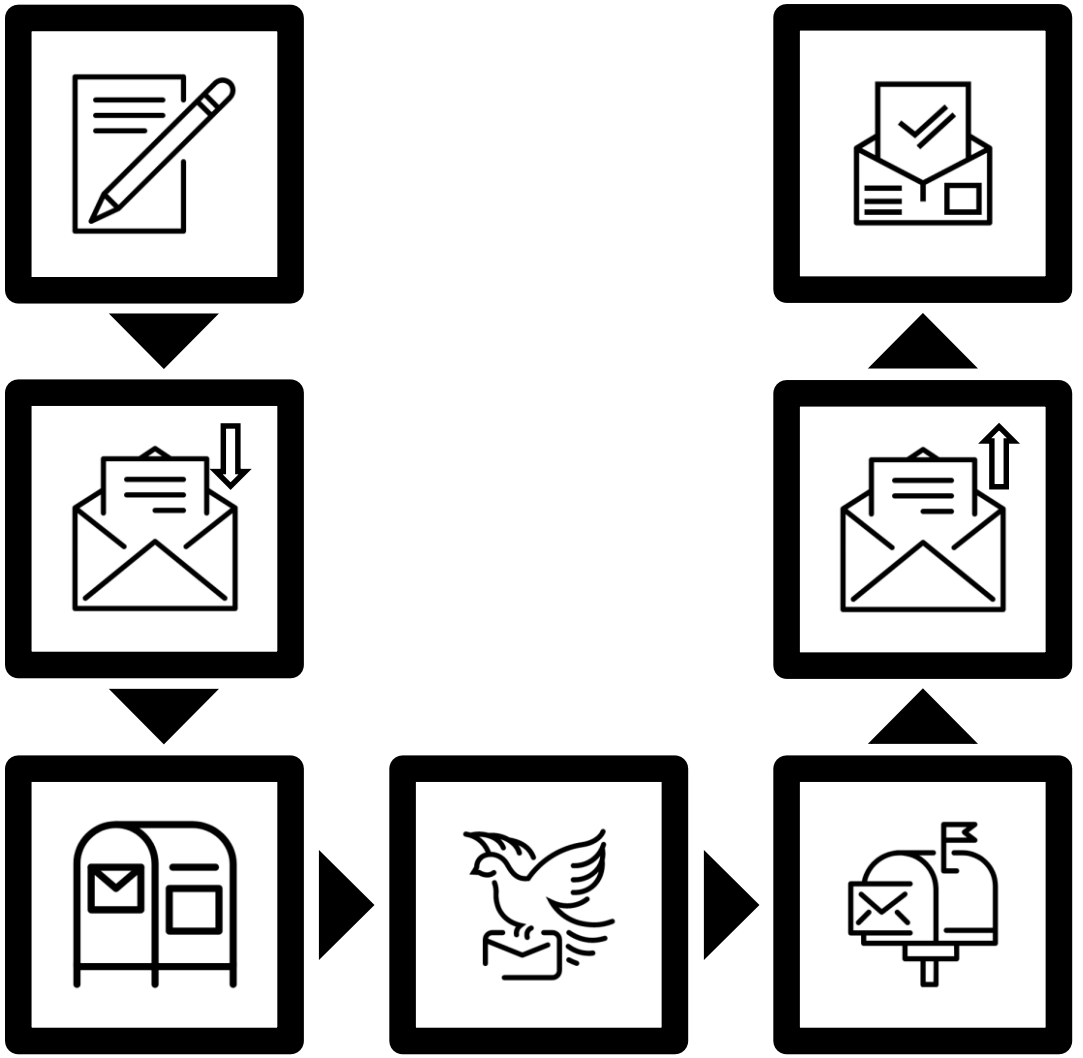
- How does the envelope get there?
  - We don’t care!



Layers & Abstraction

- How do you send a postcard?
  1. Write postcard
  2. Put postcard in envelope
  3. Mail envelope to recipient
  4. Recipient receives envelope
  5. Recipient opens envelope
  6. Recipient reads postcard

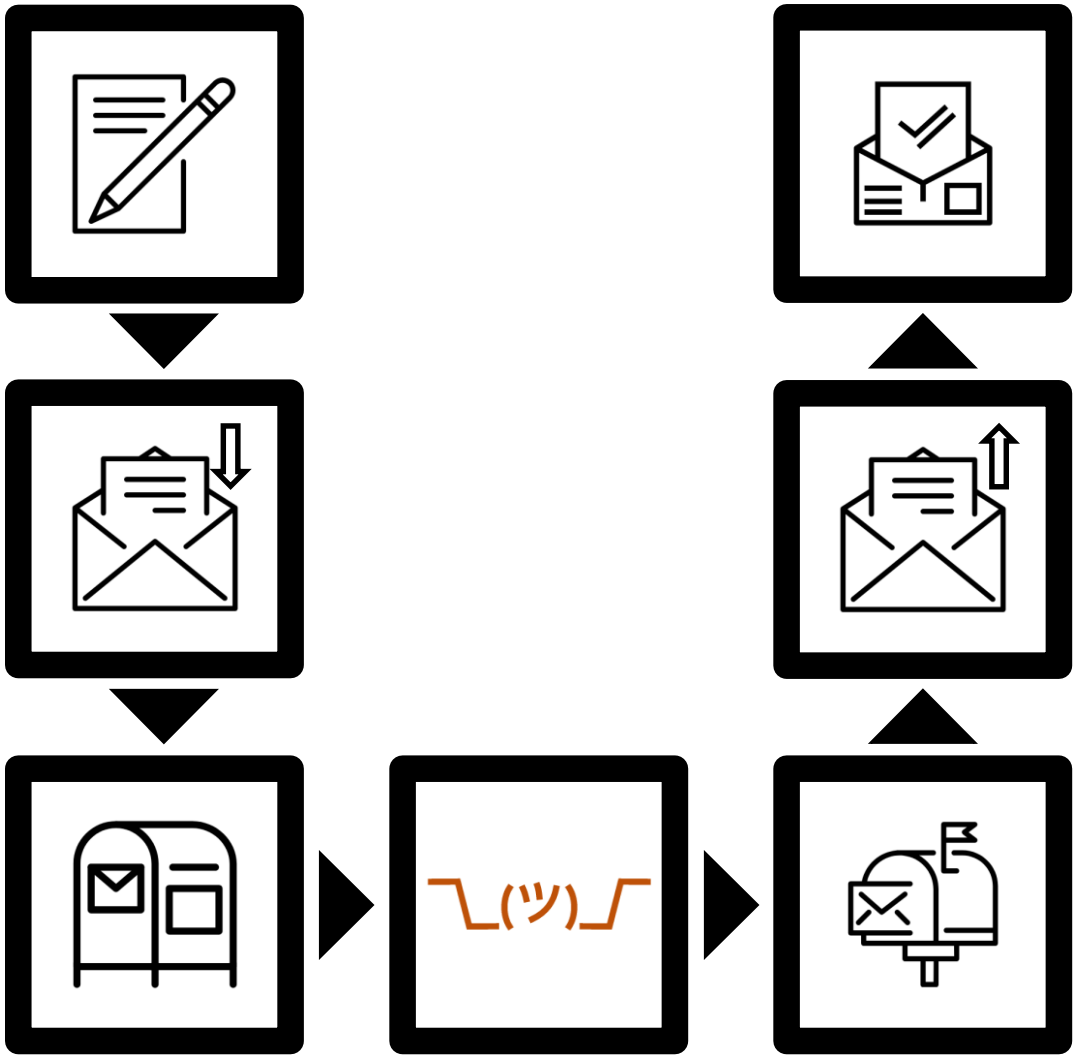
- Use a homing pigeon instead?



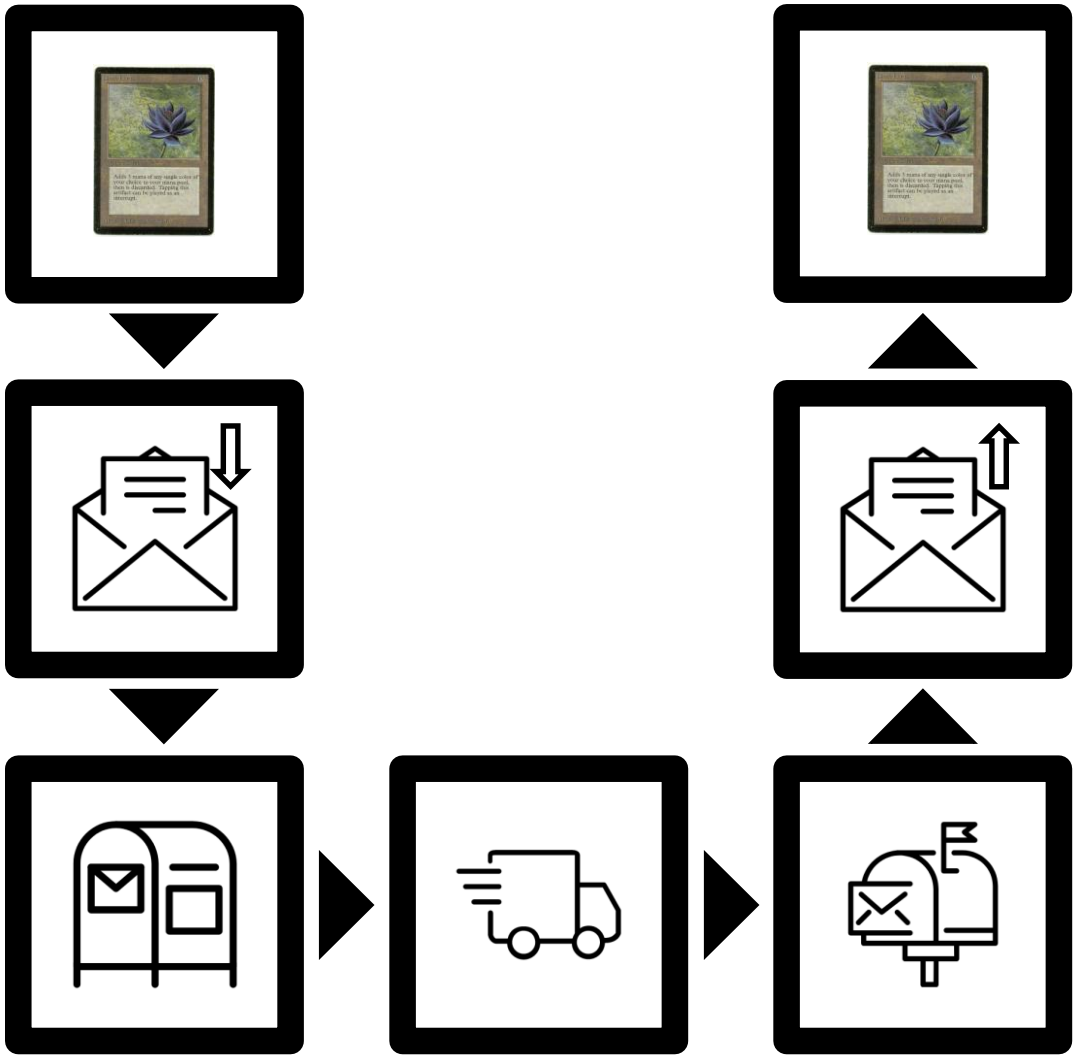
Layers & Abstraction

- How do you send a postcard?
  1. Write postcard
  2. Put postcard in envelope
  3. Mail envelope to recipient
  4. Recipient receives envelope
  5. Recipient opens envelope
  6. Recipient reads postcard

- Use a homing pigeon instead?
  - We don’t care!



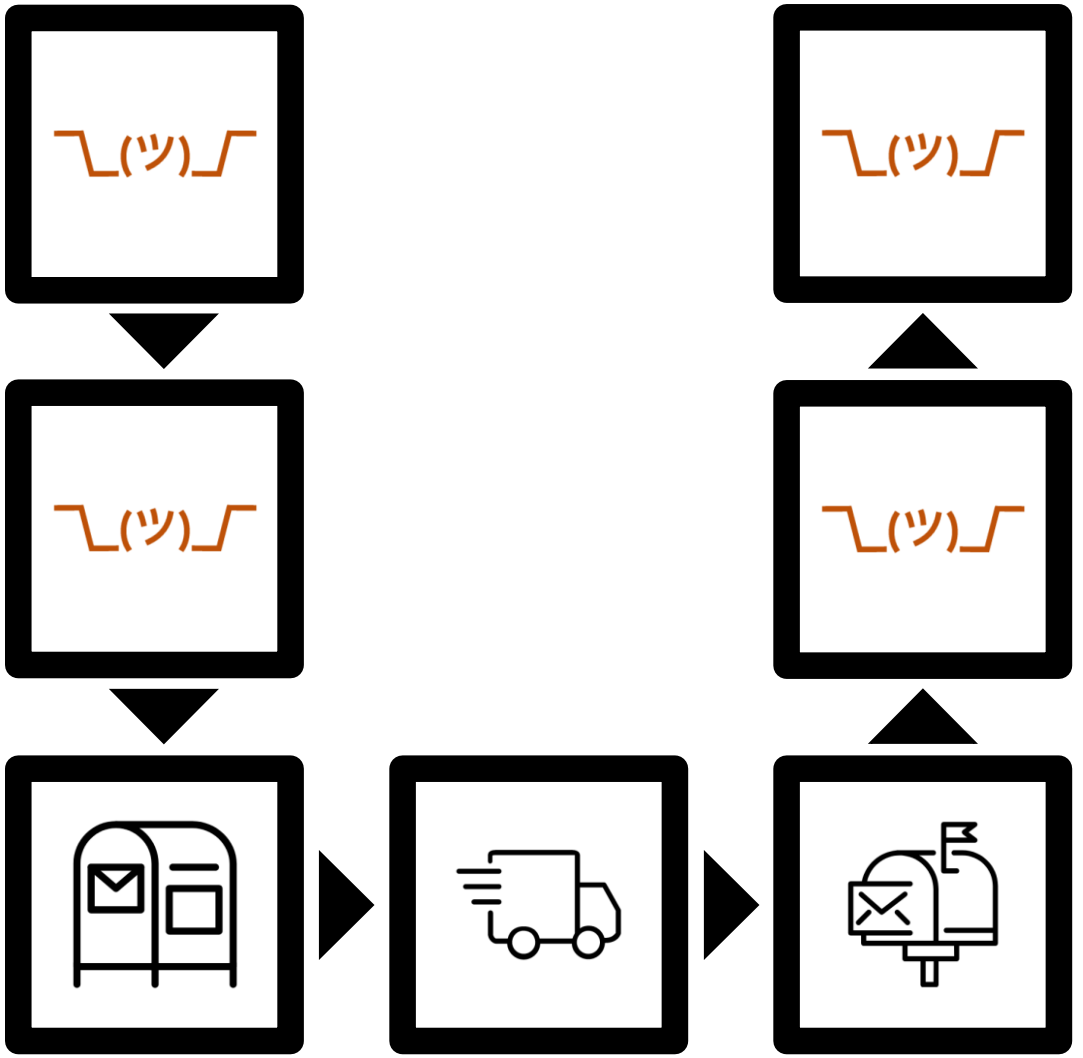
Layers & Abstraction



"Beta Black Lotus" photo © Sean Ellis, flickr.com, used under CC BY

- Mail a trading card instead?

Layers & Abstraction



- Mail a trading card instead?
  - The post office doesn’t care!

Layers & Abstraction

- Division of responsibility
  - I don’t need to care how my envelope gets there
    - Transporting it is the post office’s job
  - The post office only needs to care about envelopes
    - Securing the something *inside* an envelope is my job
- No need to constantly re-invent the wheel!

Layers & Abstraction

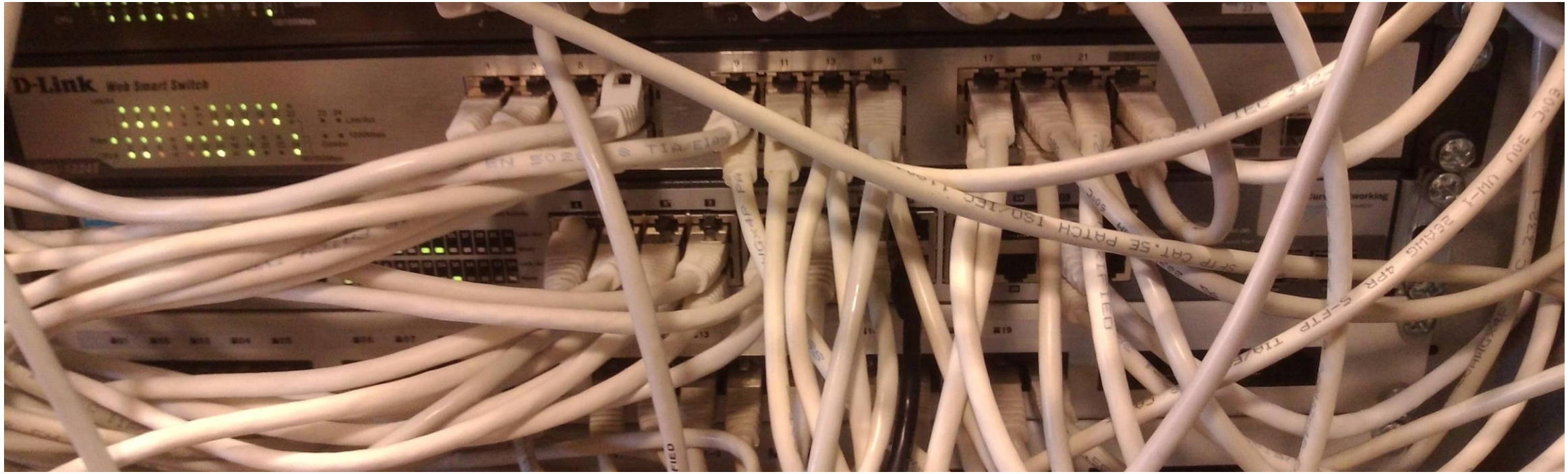
- Networking equivalent: **Layers**
- 1980/90s: competing models & protocol suites
  - TCP/IP, OSI, …
- Modern internet uses the TCP/IP model
  - So that's what we'll talk about!
  - Less powerful than OSI, but more flexible

The TCP/IP model

- Link layer
  - Send a chunk of data to a **directly connected** computer
- Internet layer
  - Route a chunk of data to a **remote** computer along a series of direct links
- Transport layer
  - Transmit a **structured** bit stream across the internet
- Application layer
  - **Offer services** without having to worry about details

Layers & Abstraction

- Abstraction:
  - Keeps complexity manageable
  - May introduce inefficiencies
  - Introduces rigidity
- Real-world protocols are not *fully* isolated from one another



The Link Layer

The Link Layer

- Computers A, B, C, etc. are all “connected” to each other
- **Goal:** Send data from A to C

- Properties of the medium:
  - Can you send and receive at the same time? (“half-duplex” vs “full-duplex”)
  - Can you send and listen at the same time? (collision detection)
  - If you speak, can “everyone” hear you? (shared medium)

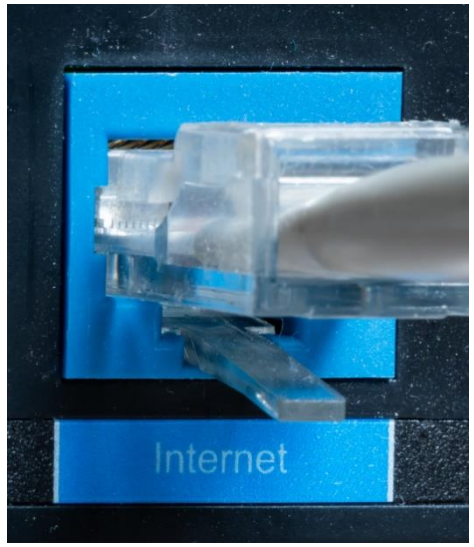
- Concerns:
  - Was the data distorted over the “wire”? (integrity)

Example: Wi-Fi (IEEE 802.11)

- Shared medium: Wireless radio
- Central access point
  - Nodes communicate via the AP
- Not full-duplex
  - If two nodes send at the same time, the signals are garbled
- No direct collision detection
  - If a node is sending, it cannot listen for transmissions at the same time
- Data is acknowledged
  - Collision -> no acknowledgment -> Data re-sent

Image: Public Domain

Example: Ethernet (IEEE 802.3)



- Star-shaped structure
  - Clients directly connected to one or more *switches*
  - Hardware failure only disconnects that client
- Full-duplex (in modern networks)
  - No collisions possible
- Switched medium (mostly, in modern networks)
  - We'll talk details in a bit

Addressing

An Ethernet frame, common on the modern Internet

- An *address* identifies a destination
  - In a shared medium, the recipient can recognize their data
  - In a switched medium, we know where to send the data
- **MAC address**: 48-bit identifier
  - Used in: Ethernet, Wi-Fi, Bluetooth, …
  - Should be locally unique
- **Broadcast address**: `FF:FF:FF:FF:FF:FF`
  - Will be sent to all connected hosts

Image: Public Domain

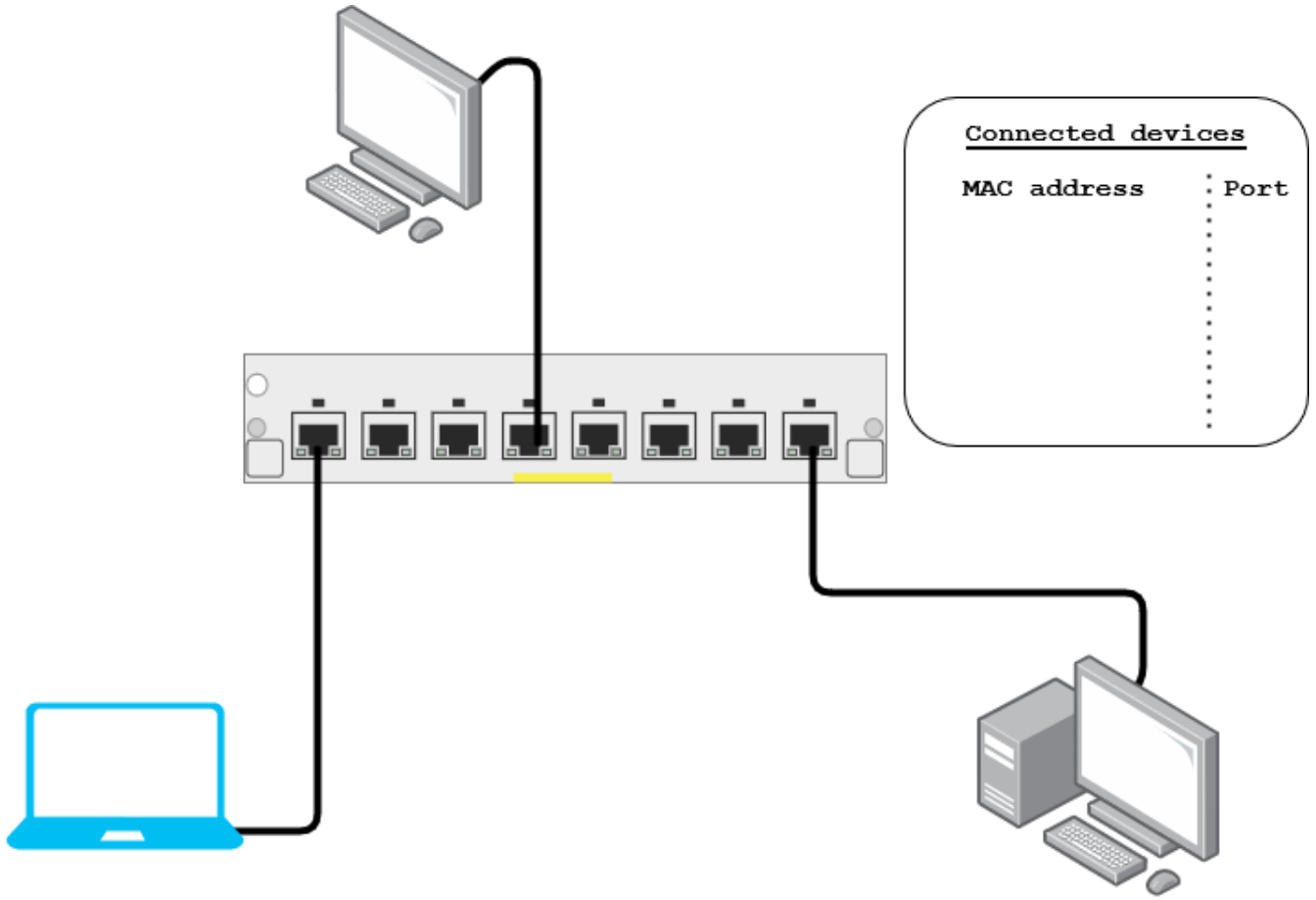
Ye Olde Ethernette

- Once upon a time, Ethernet was a shared medium…
  - At first, it used a single coaxial cable…
    - Physically connecting all the hosts!
  - Later, it used *Ethernet hubs* that emulated this…
    - Simply re-broadcast any received signal to all ports
- We interconnect hundreds of computers
  - Only one can talk at a time?

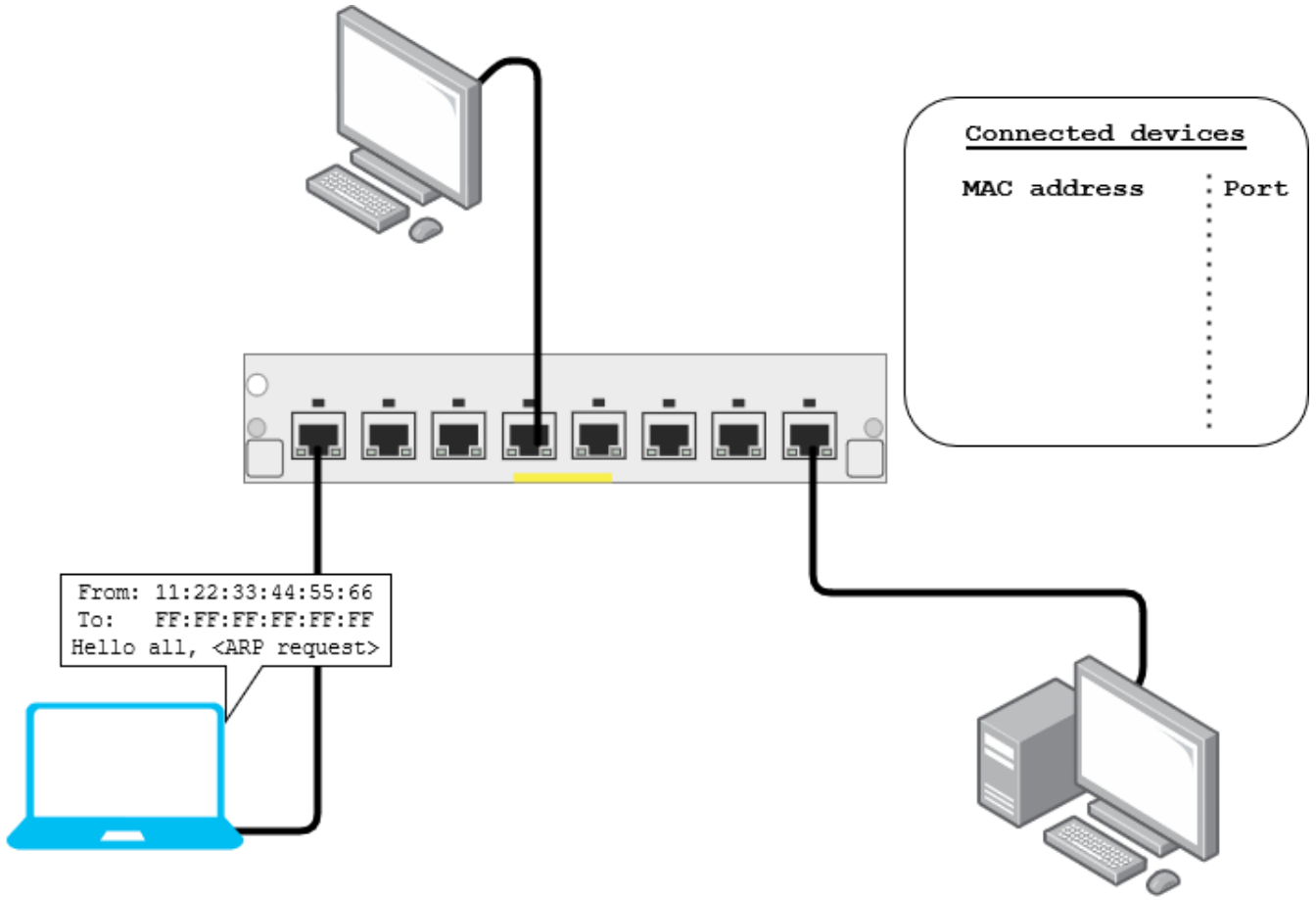
Ethernet: Switching

- Ethernet *switches* understand Link Layer data
  - Read source/destination MAC addresses
- Record source addresses to build map address <-> port
- Only forward packets to the appropriate port
  - Minimize wasted bandwidth
  - No collisions possible!

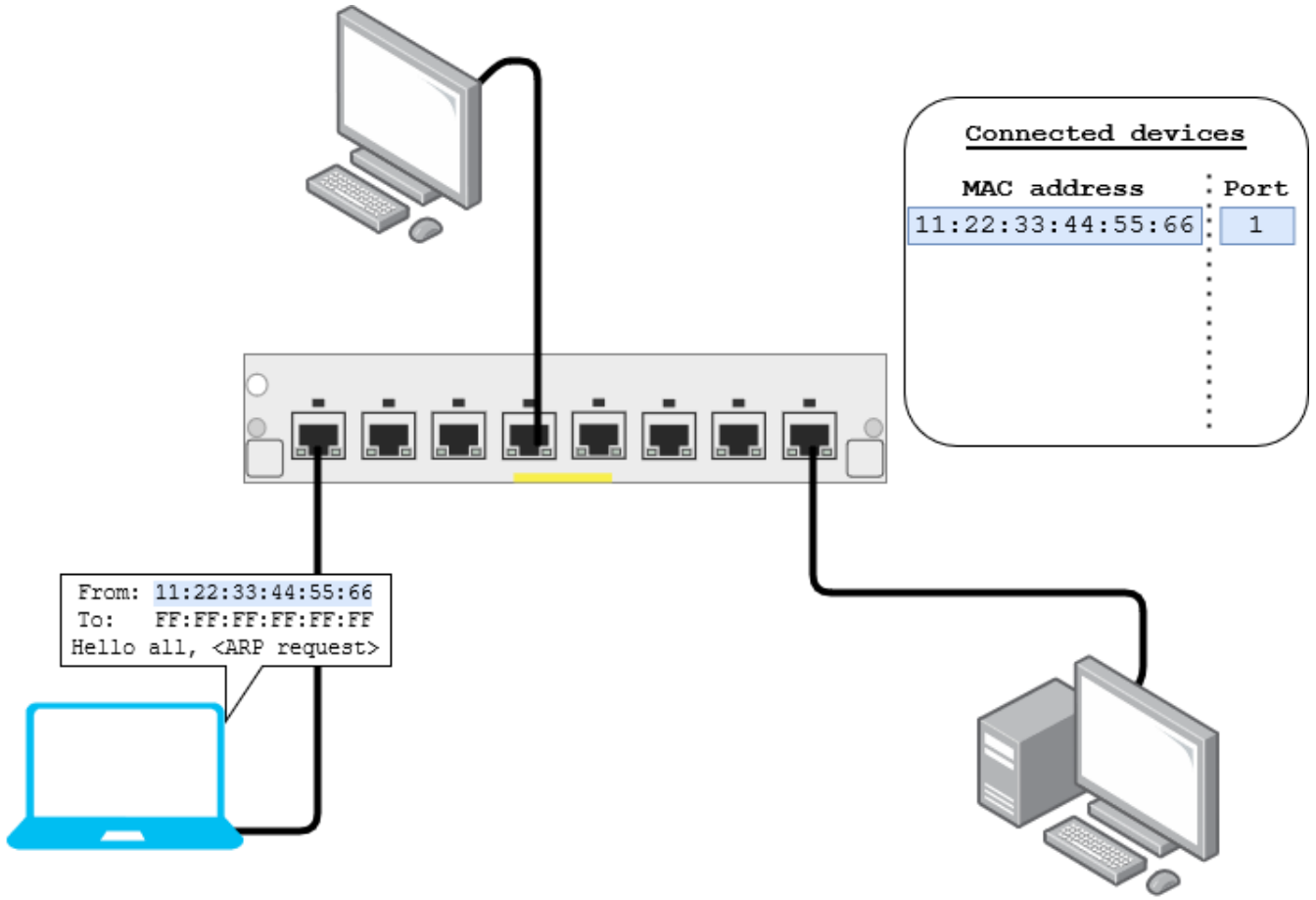
Ethernet: Switching



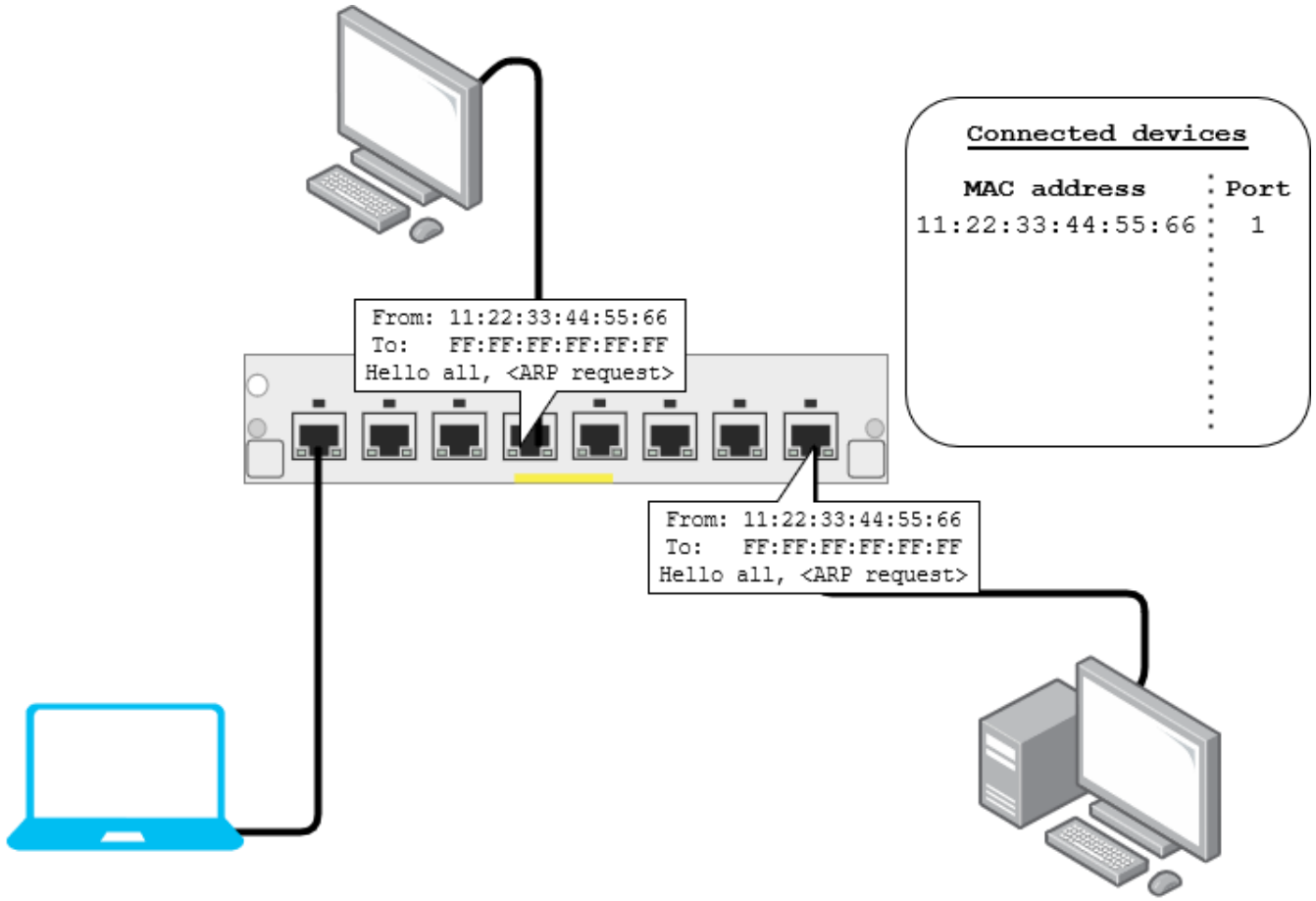
Ethernet: Switching



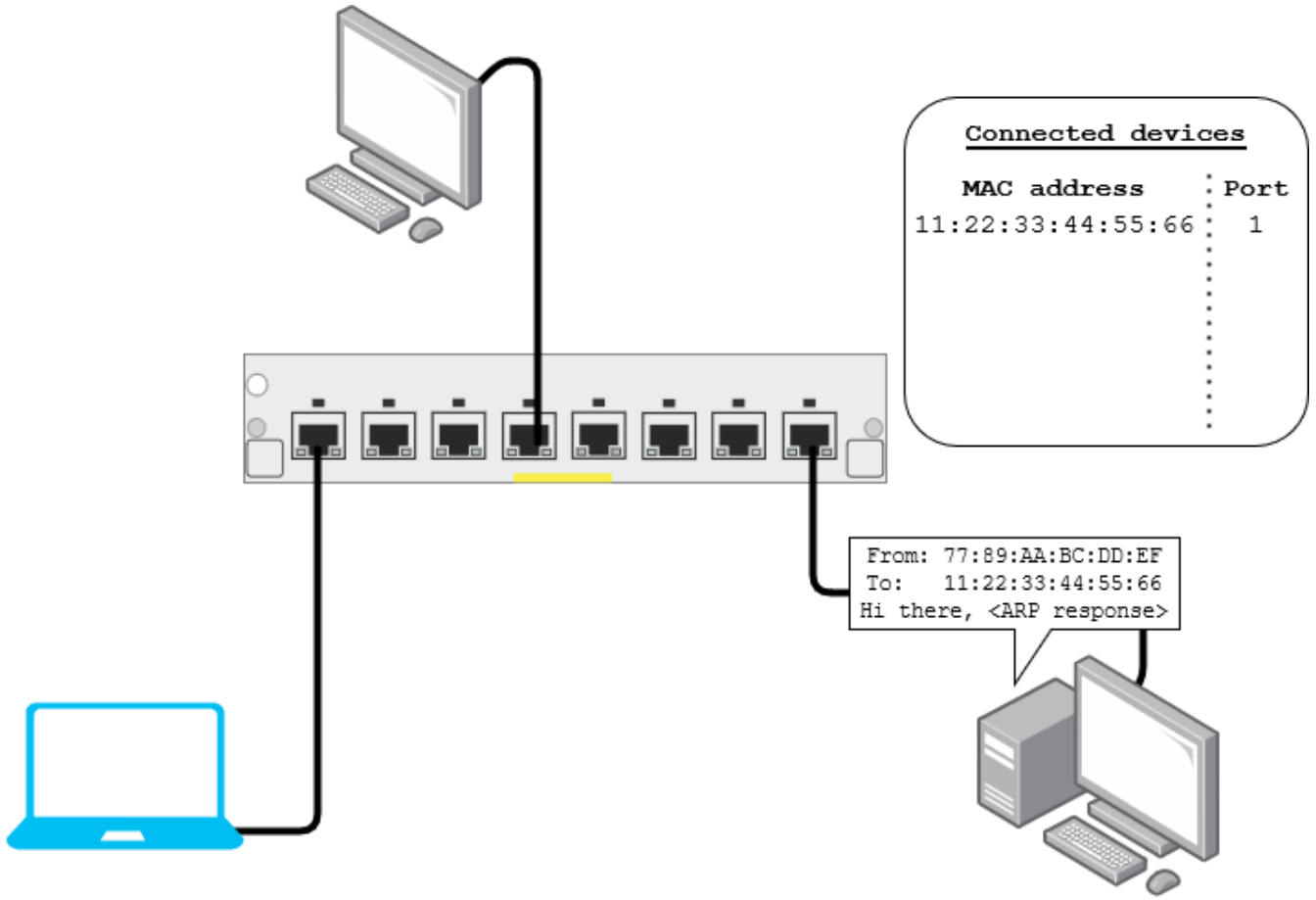
Ethernet: Switching



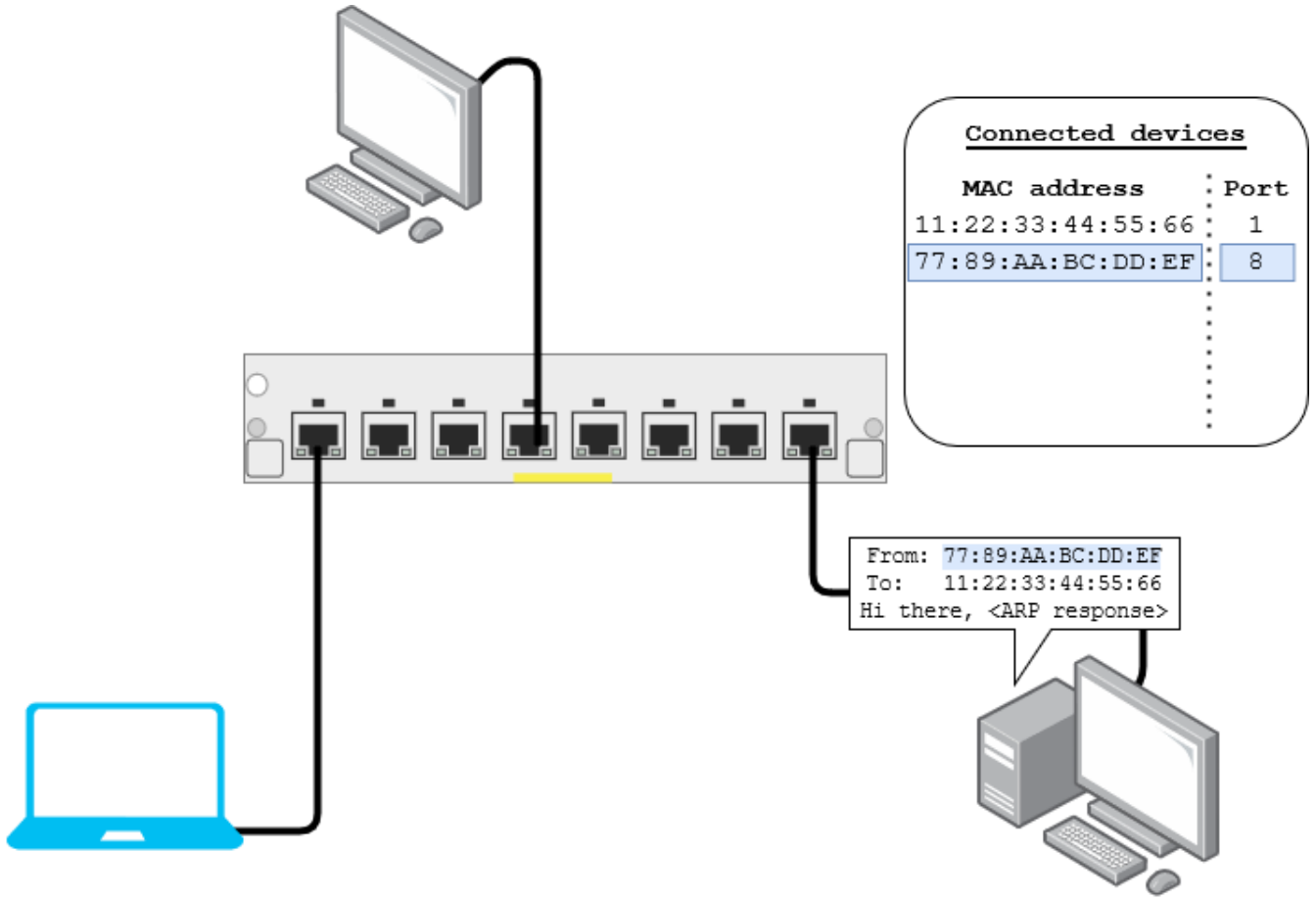
Ethernet: Switching



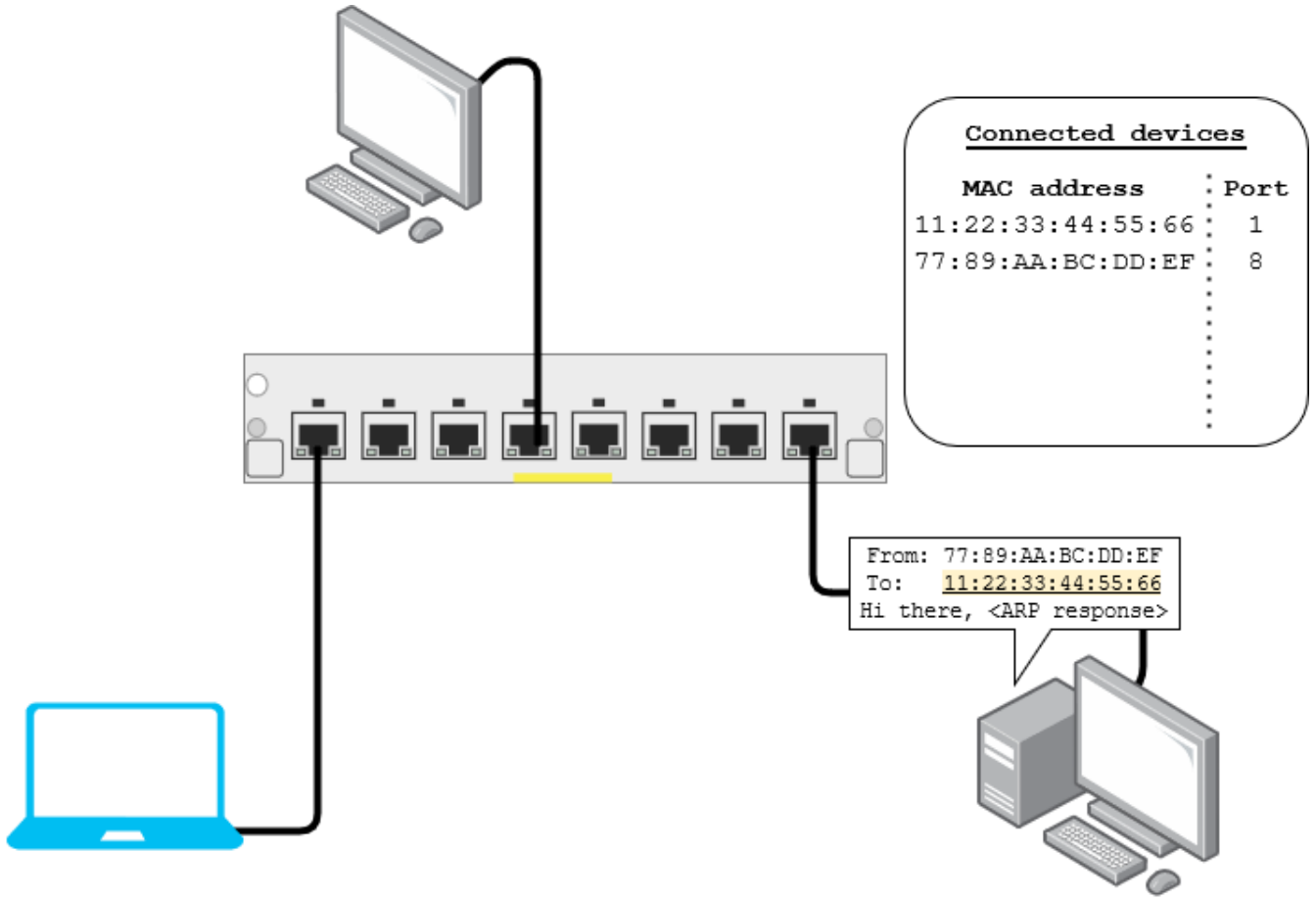
Ethernet: Switching



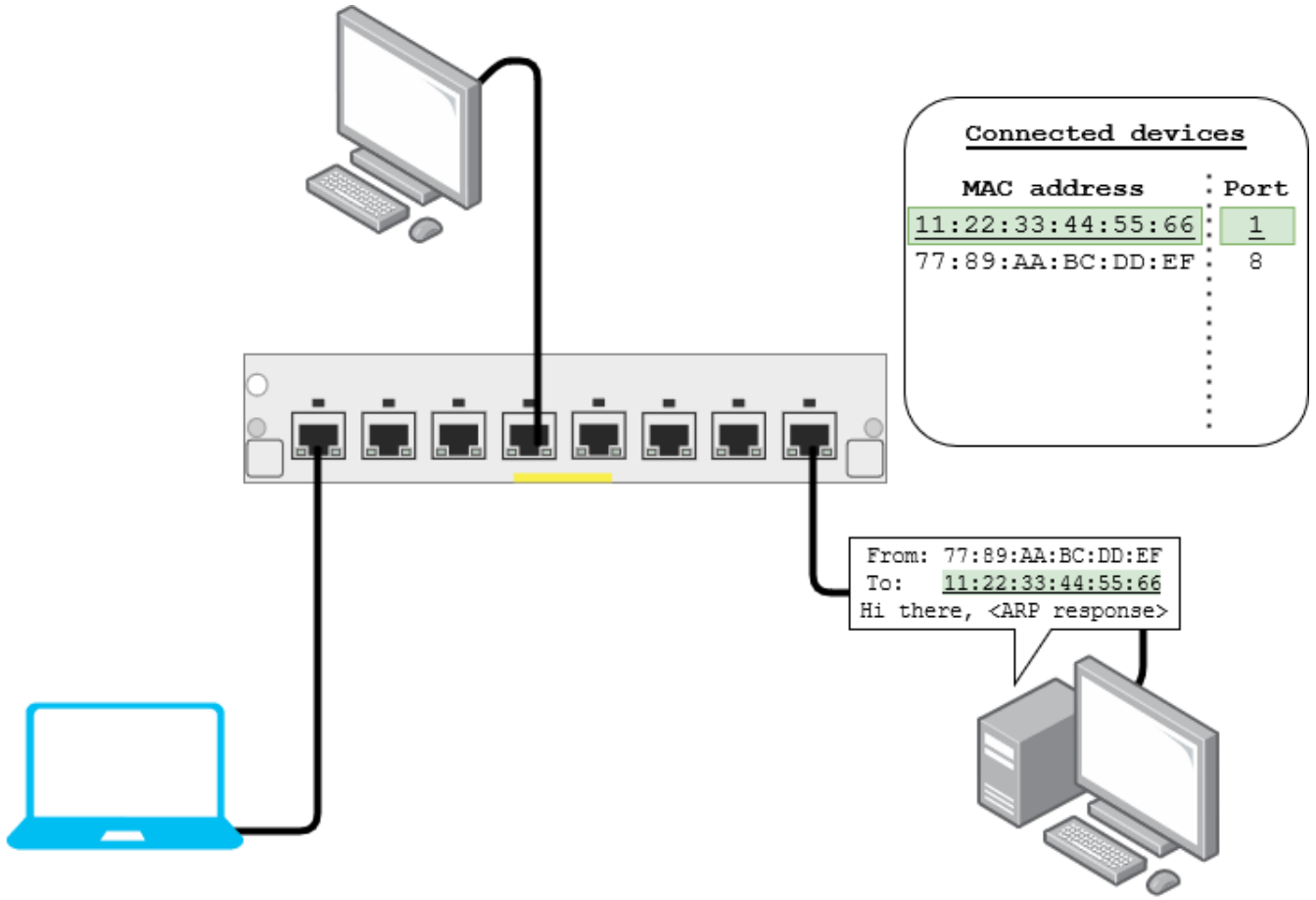
Ethernet: Switching



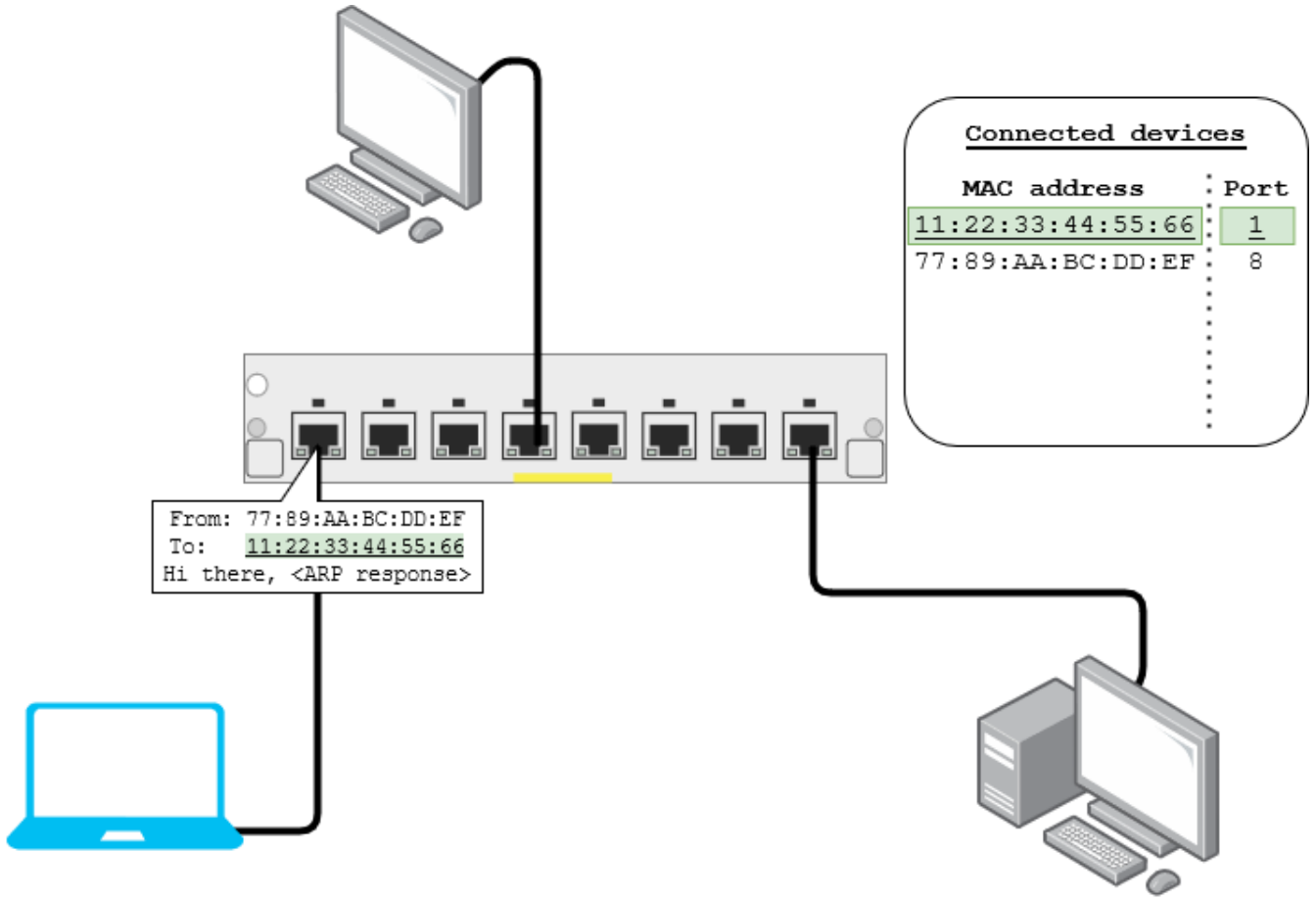
Ethernet: Switching



Ethernet: Switching



Ethernet: Switching

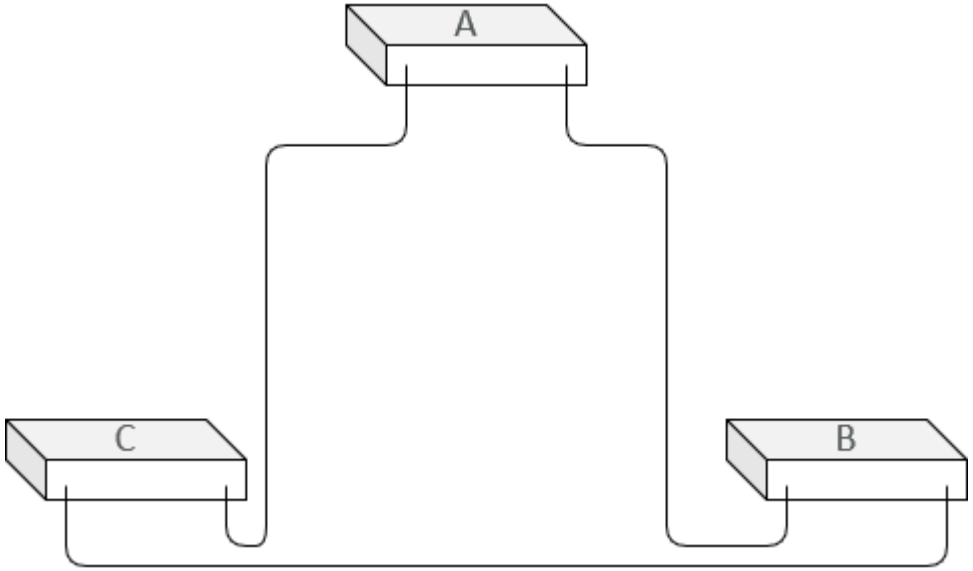


Ethernet: V-LAN

- **V**irtual **LAN**s
  - Partition switch ports into different logical networks
  - Devices on different networks cannot send packets to each other
  - Broadcast packets are only broadcast to the device's VLAN
- Benefits
  - Partitioned networks
  - No re-wiring required
  - Configured in software
- Downsides
  - Configured in software

Ethernet: Switching Loops

- Multiple switches can be interconnected to form one big network
- **Problem:** switching loops
  - Why is it a problem? Broadcasts!
  - If a broadcast frame reaches this topology, it will multiply endlessly
- **Solution:** don't build switching loops!
  - However, they are useful for redundancy
- **S**panning **T**ree **P**rotocol
  - Supported by professional switches
  - Automatically disables redundant links until needed



The Network Layer

The Network Layer

- Computers A and B are connected to different physical networks
- There is some way to get from A's network to B's network
- **Goal:** Send data from A to B

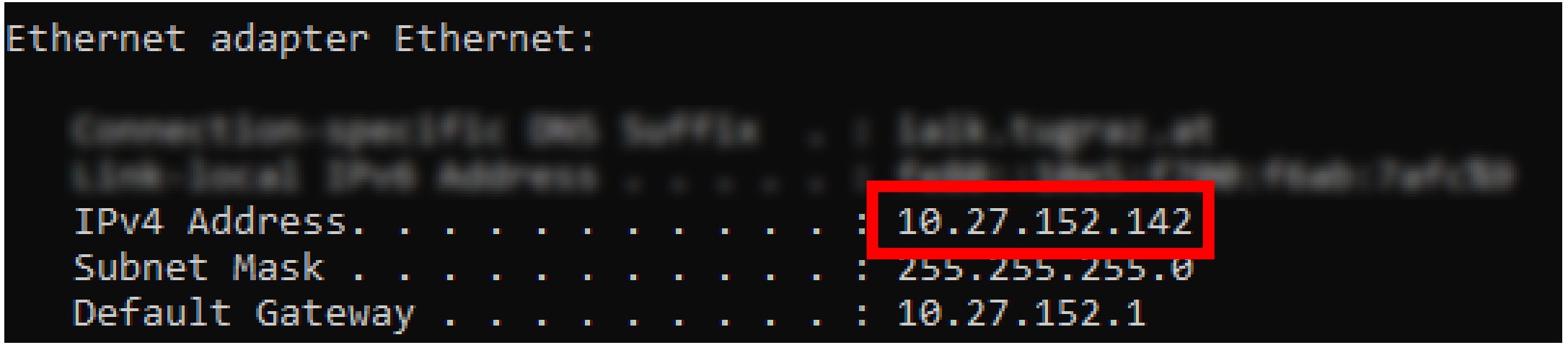
- Concerns:
  - How does the data get from A to B? (routing)
  - What if the data is too large for a certain path? (fragmentation)

IPv4

- **I**nternet **P**rotocol, **v**ersion **4**
- Foundation of today's internet

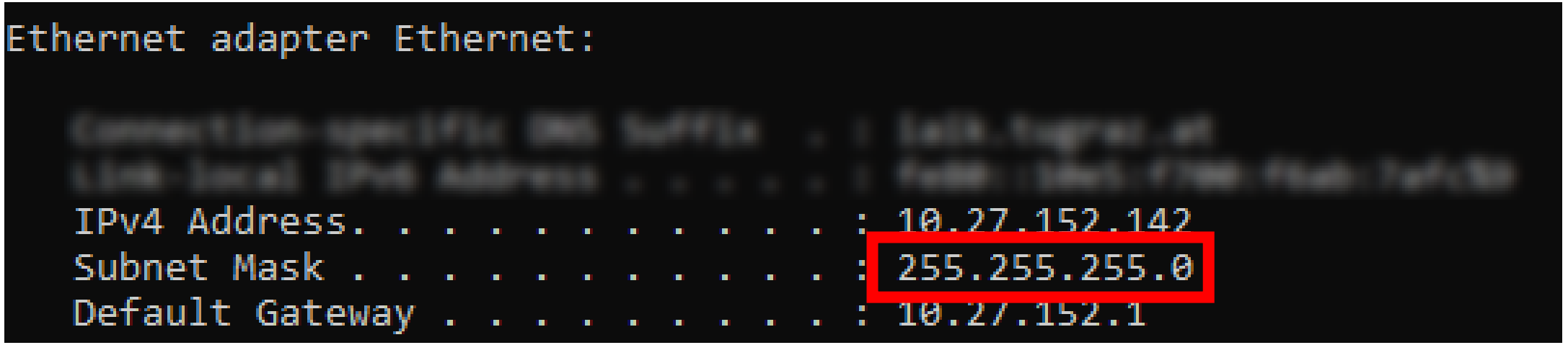
- Used in almost every network-enabled device

IPv4 addressing



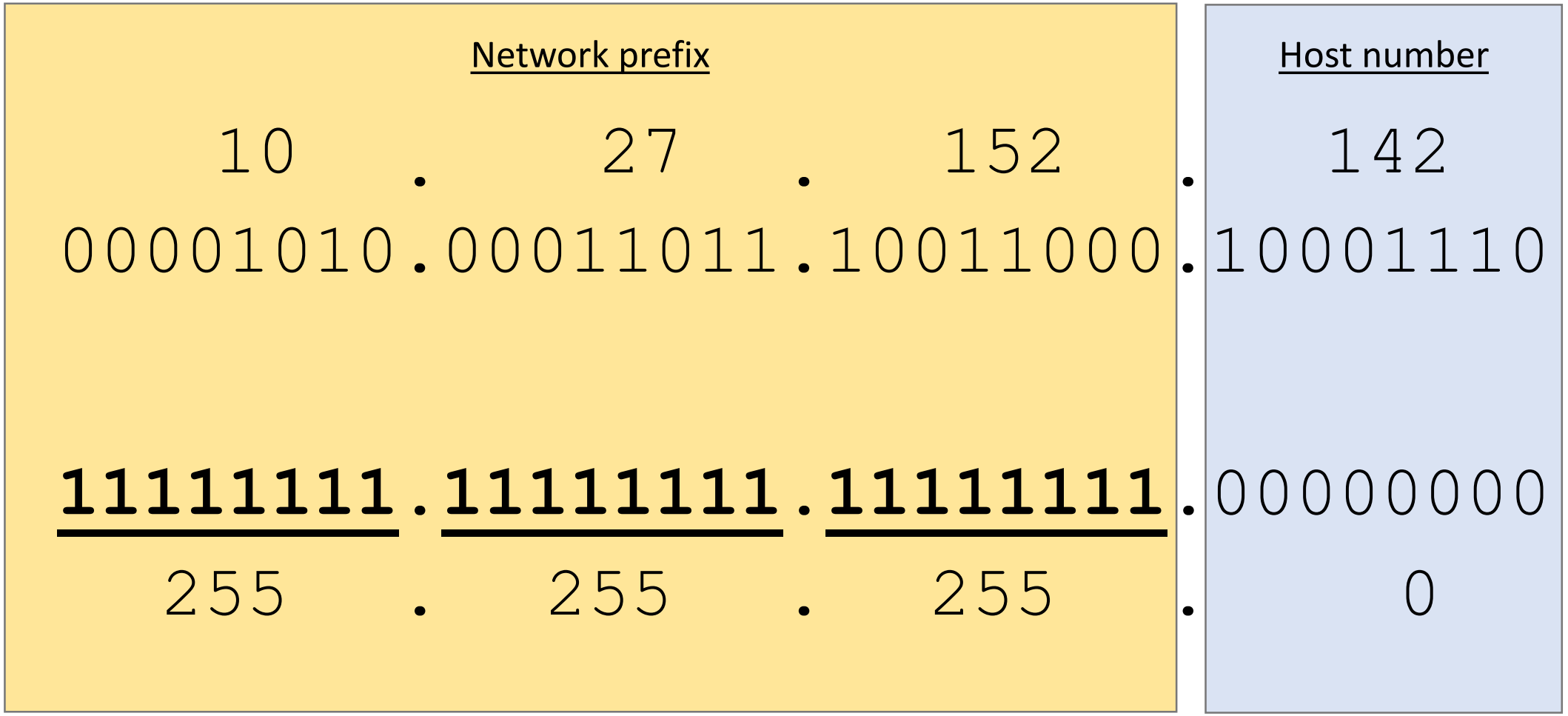
- 32-bit address
  - Notation: bytes' decimal value (0-255)
  - **`10.27.152.142`** is the same as **`0a 1b 98 8e`**
- Each participating network card has a single IPv4 address

IPv4 addressing



- 32-bit subnet mask
  - Splits address into *network prefix* and *host number*
  - All ones, followed by all zeros
  - Alternate notation: just specify number of ones
    - **`255.255.255.0`** is the same as **`/24`**

IPv4 addressing



IPv4 addressing

- All hosts with the same network prefix form a *subnet*
- Hosts within the same subnet can communicate directly
  - They’re in the same Link Layer network!

- Two addresses per subnet have special meaning
  - Host number all zeros ≙ network identifier
    - **`10.27.152.142/24`** is part of the **`10.27.152.0/24`** network
  - Host number all ones ≙ broadcast address
    - **`10.27.152.255/24`** is the broadcast address for the **`10.27.152.0/24`** network

IPv4 addressing

- Subnet masks do not need to be full bytes
  - **`255.255.255.240`** (28 bits network prefix, 4 bits host number ≙ **`/28`**)
  - **`192.168.13.80/28`** can have up to 14 host addresses
    - Network address: **`192.168.13.80`** (**`80`** ≙ **`01010000`**)
    - First host address: **`192.168.13.81`** (**`81`** ≙ **`01010001`**)
    - Last host address: **`192.168.13.94`** (**`94`** ≙ **`01011110`**)
- Not every broadcast address ends with **`.255`**!
  - What is the broadcast address for **`192.168.195.0/28`**?
- Not every address that ends with **`.255`** is a broadcast address!
  - **`10.5.0.255/16`** is the 255th host in the **`10.5.0.0/16`** subnet

IPv4 addressing

- Need addresses for your home?
  - Private address space that anyone can use:
    - **`10.0.0.0/8`** (i.e., **`10.0.0.0`** to **`10.255.255.255`**)
    - **`172.16.0.0/12`** (i.e., **`172.16.0.0`** to **`172.31.255.255`**)
    - **`192.168.0.0/16`** (i.e., **`192.168.0.0`** to **`192.168.255.255`**)
  - Not globally unique
    - Won't work over the internet!
- Never configured an IP address before?
  - Your ISP modem likely does this for you!
  - **D**ynamic **H**ost **C**onfiguration **P**rotocol
  - Enabled by default on modern devices

IPv4 addressing

- Destination address in my subnet?
  - Talk to it using Data Link Layer
- …talk to it using Data Link Layer?
  - We only have an IP address
  - At the Data Link Layer, we need a MAC address

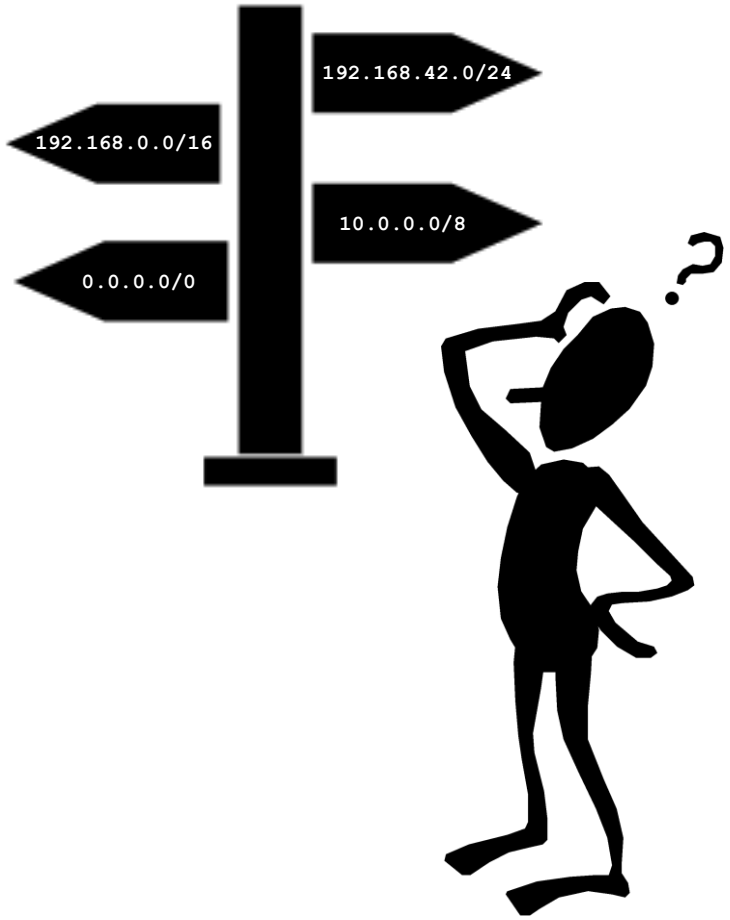
IPv4 addressing

- Destination address in my subnet?
  - Talk to it using Data Link Layer

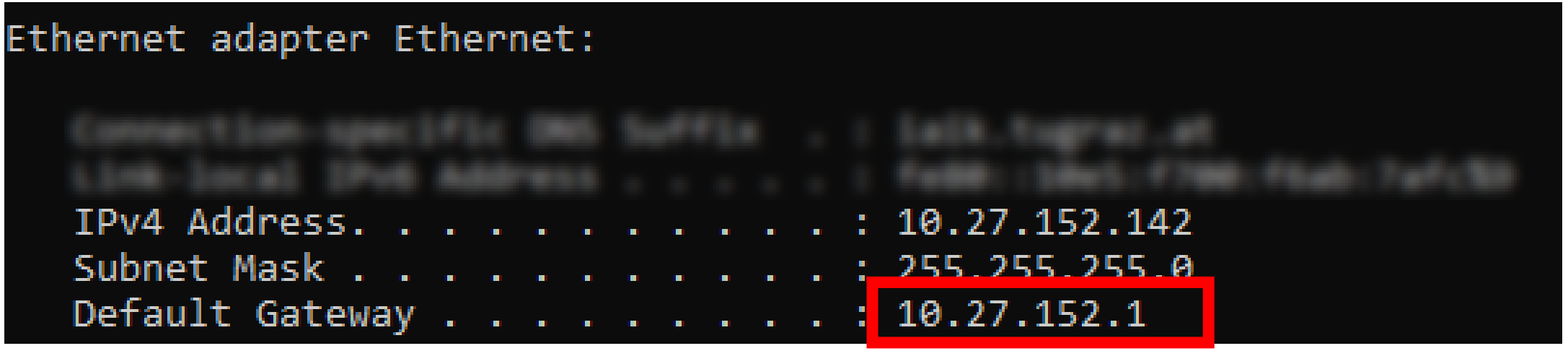
- **A**ddress **R**esolution **P**rotocol
  - Ethernet frames with type **0x0806**
  - Very simple stateless protocol
    - Request MAC for given IP (Ethernet broadcast)
    - Target responds (Ethernet unicast), now we know its MAC address
  - Heavily cached to avoid lots of broadcasting

IPv4 routing

- Destination address in my subnet?
  - Talk to it using Data Link Layer
- Destination address *not* in my subnet?
  - Check *routing table*
  - Maps destination address to *next hop*
    - Move packet in “the right direction”
  - Send packet to next hop using Data Link Layer
  - Eventually it gets there

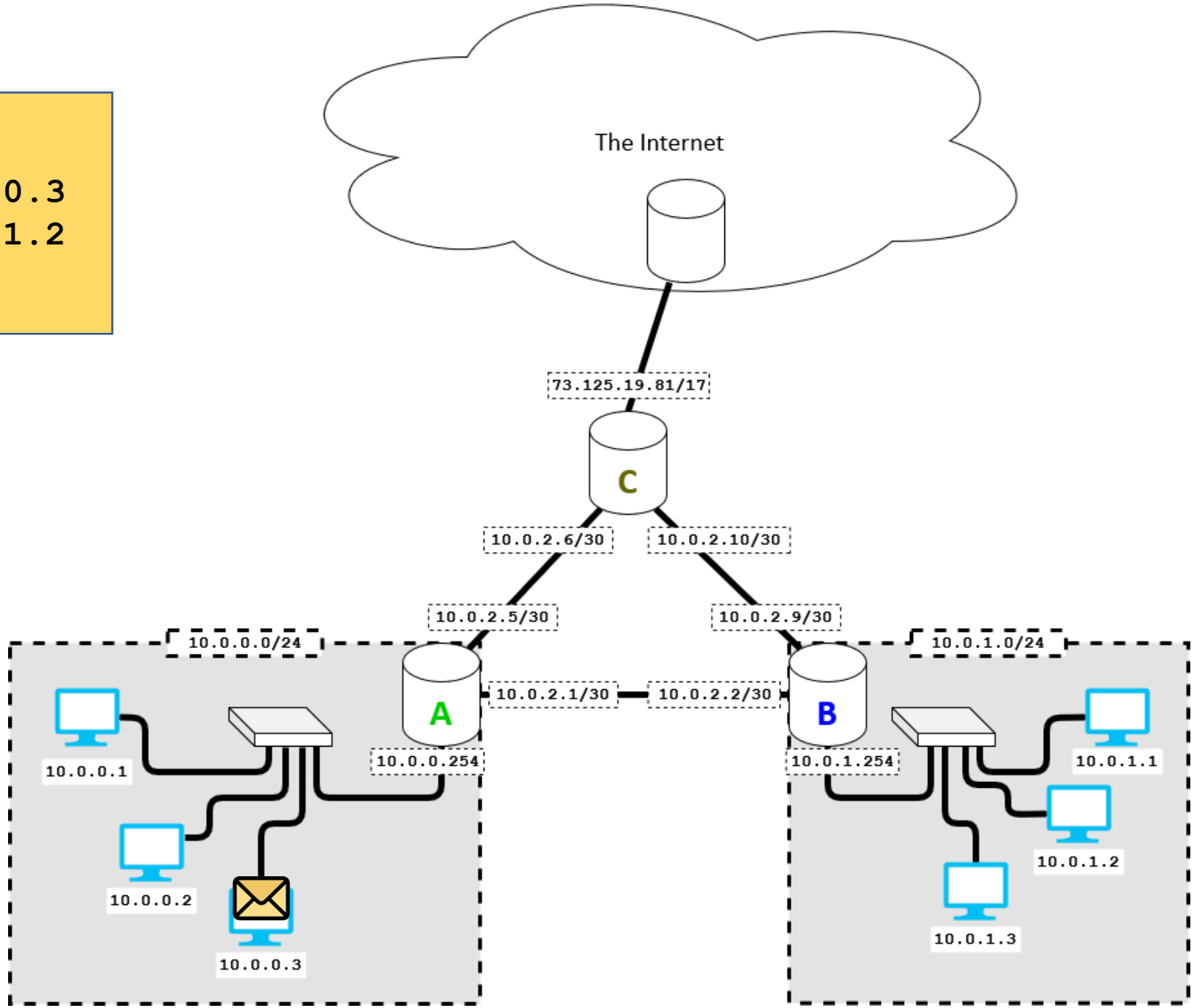


IPv4 routing

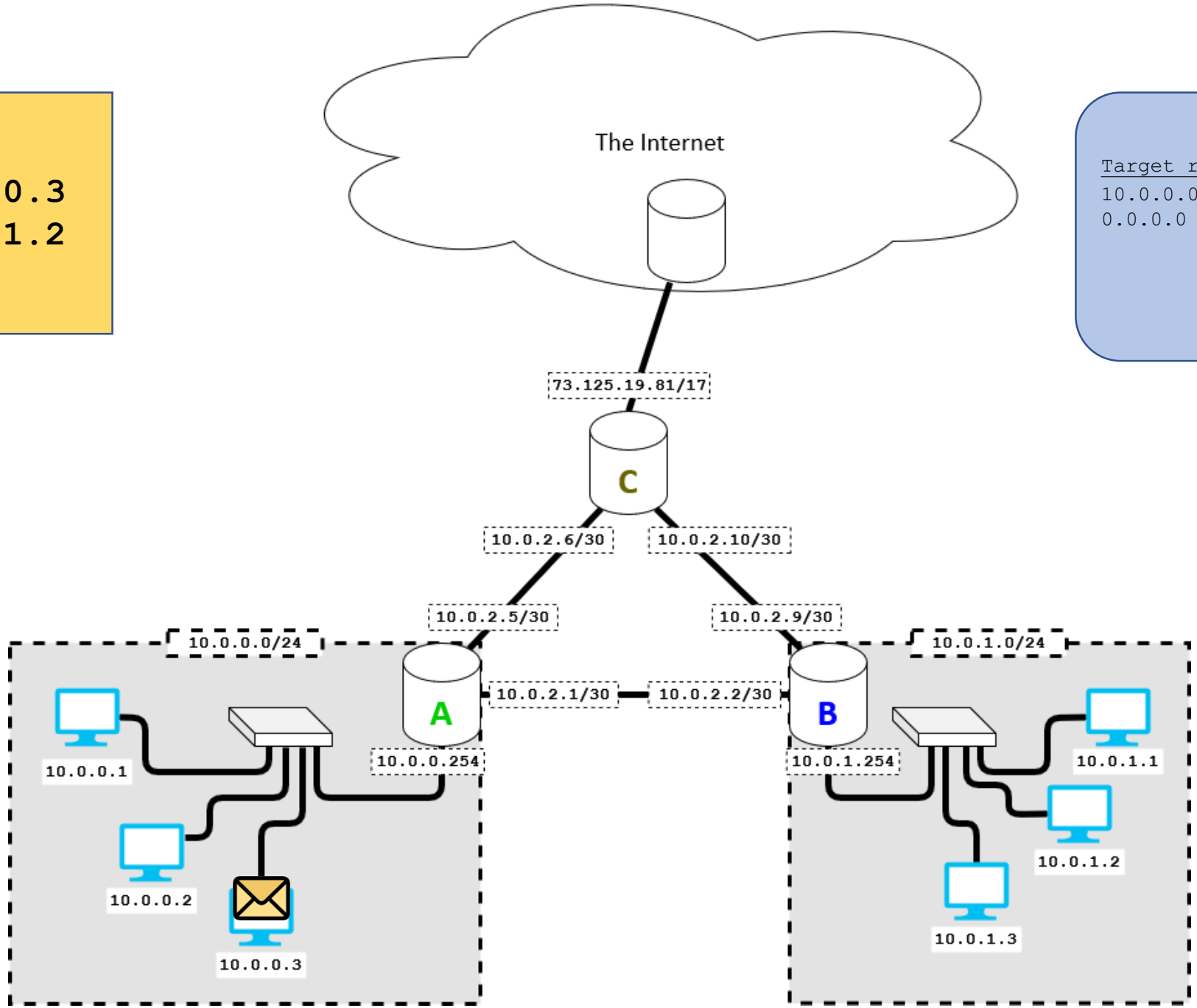


- Most host computers only have one entry in their routing table
  - Send any non-subnet data to this *router*
    - At home, this is usually your ISP modem!
  - The router will figure out where to pass the packet to

From: **10.0.0.3**
To: **10.0.1.2**



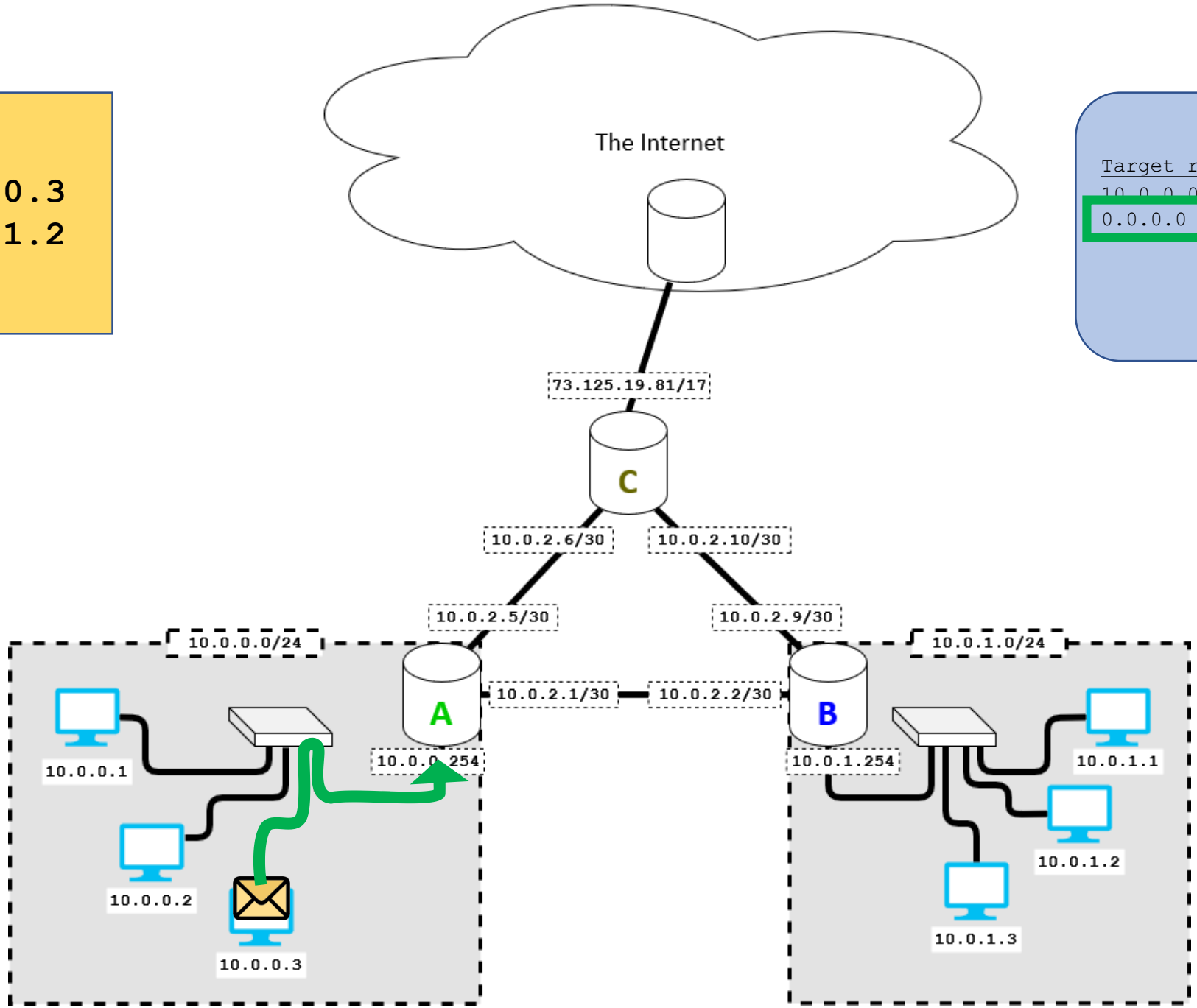
From: **10.0.0.3**
To: **10.0.1.2**



Routing Table

Target range		Next hop
10.0.0.0	/24	n/a
0.0.0.0	/ 0	10.0.0.254

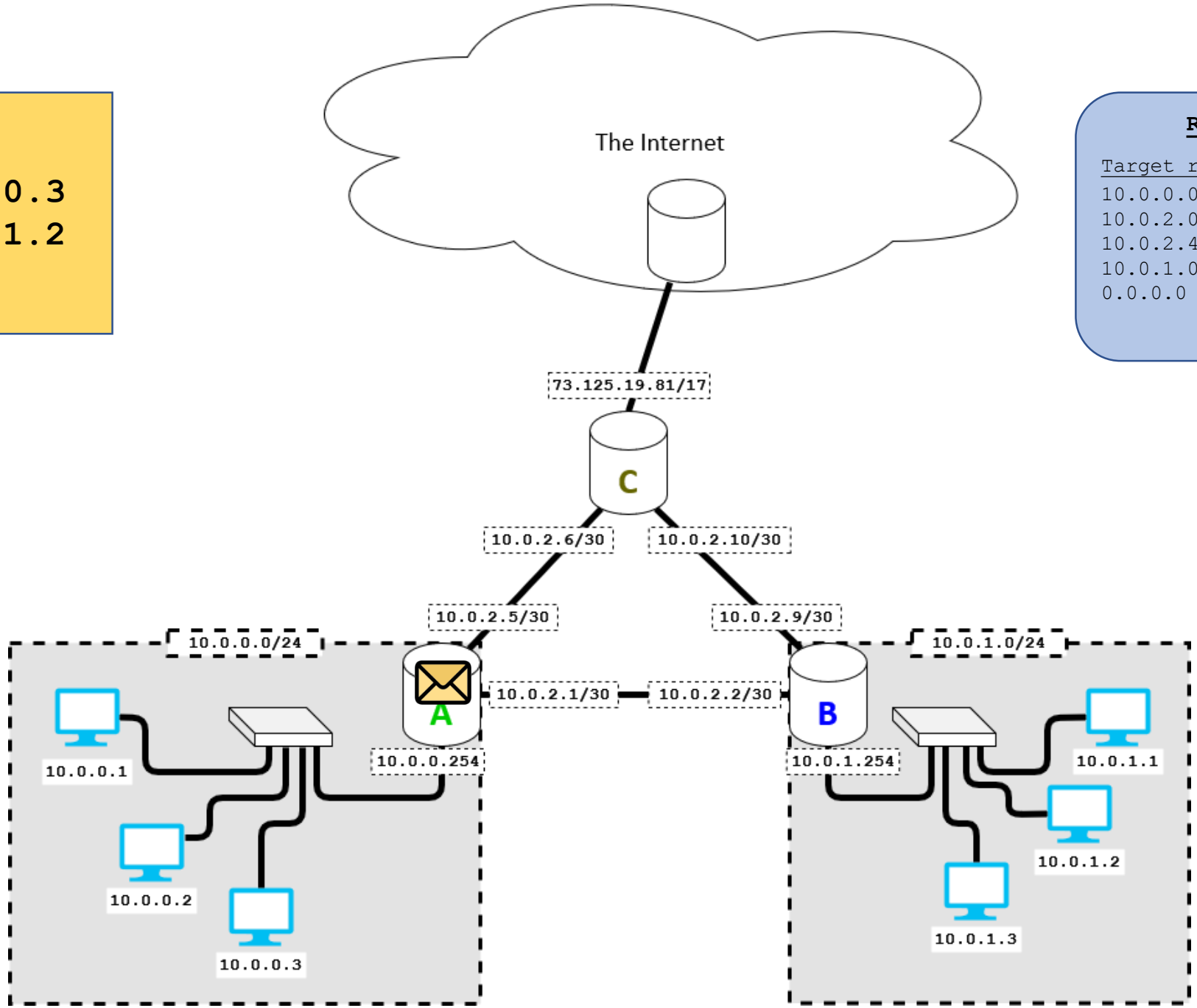
From: **10.0.0.3**
To: **10.0.1.2**



Routing Table

Target range		Next hop
10.0.0.0	/24	n/a
0.0.0.0	/ 0	10.0.0.254

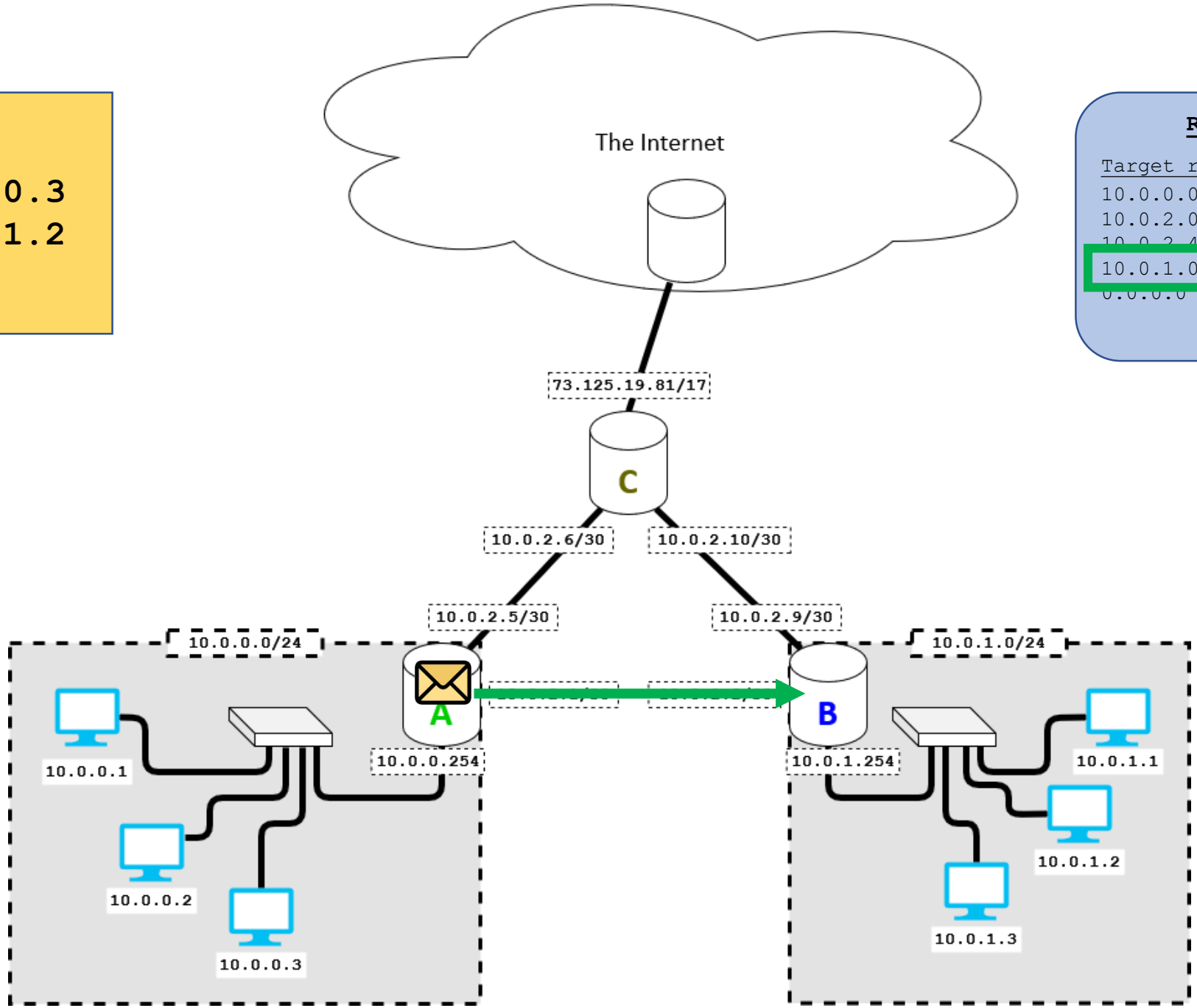
From: **10.0.0.3**
To: **10.0.1.2**



Routing Table on A

Target range		Next hop
10.0.0.0	/24	n/a
10.0.2.0	/30	n/a
10.0.2.4	/30	n/a
10.0.1.0	/24	10.0.2.2
0.0.0.0	/ 0	10.0.2.6

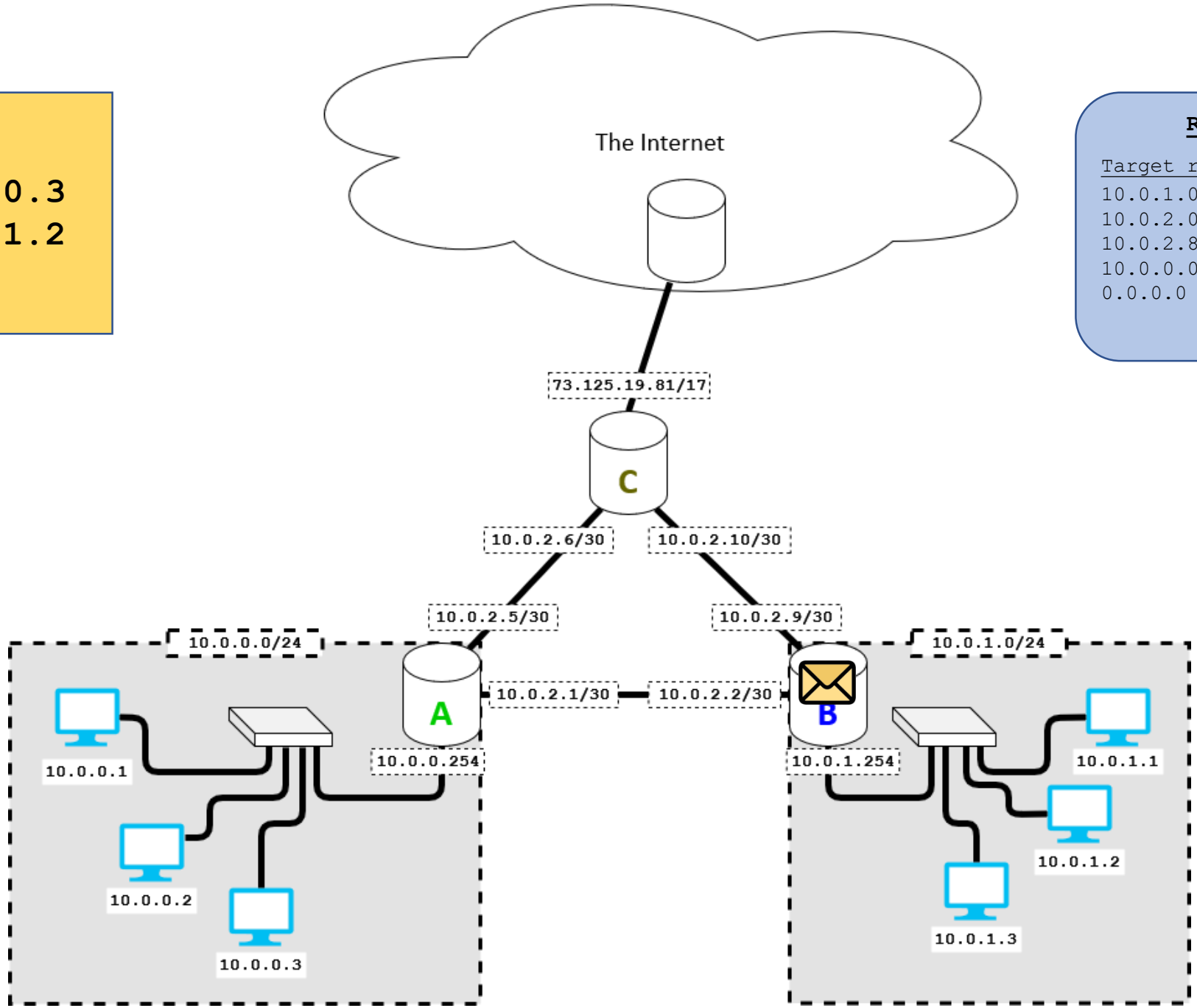
From: **10.0.0.3**
To: **10.0.1.2**



Routing Table on A

Target range		Next hop
10.0.0.0	/24	n/a
10.0.2.0	/30	n/a
10.0.2.4	/30	n/a
10.0.1.0	/24	10.0.2.2
0.0.0.0	/0	10.0.2.6

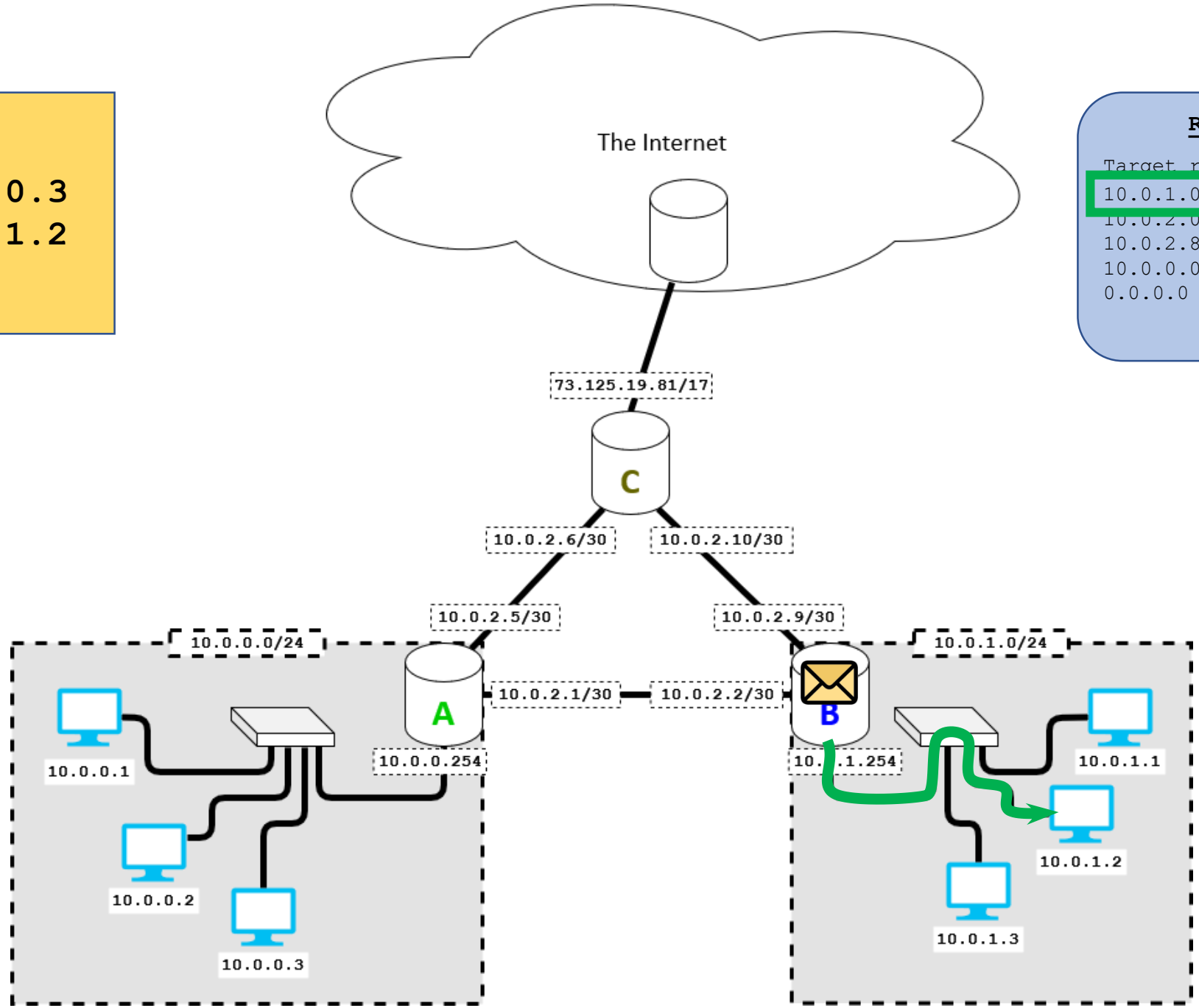
From: **10.0.0.3**
To: **10.0.1.2**



Routing Table on B

Target range		Next hop
10.0.1.0	/24	n/a
10.0.2.0	/30	n/a
10.0.2.8	/30	n/a
10.0.0.0	/24	10.0.2.1
0.0.0.0	/ 0	10.0.2.6

From: **10.0.0.3**
To: **10.0.1.2**



Routing Table on B

Target range		Next hop
10.0.1.0	/24	n/a
10.0.2.0	/30	n/a
10.0.2.8	/30	n/a
10.0.0.0	/24	10.0.2.1
0.0.0.0	/ 0	10.0.2.6

IPv4

- You can try this at home!
  - See your IP addresses:
    - **`ip addr`** or **`ifconfig`** (Linux, Mac), **`ipconfig`** (Windows)
  - See your routing table:
    - **`ip route`** or **`netstat -rn`** (Linux , Mac), **`route print`** (Windows)
  - Watch a packet over the internet:
    - **`traceroute`** (Linux , Mac), **`tracert`** (Windows)

```
Tracing route to stackoverflow.com [151.101.193.69]
over a maximum of 30 hops:

  1    <1 ms    <1 ms    <1 ms  10.27.152.1
  2    <1 ms    <1 ms    <1 ms  129.27.200.161
  3     *        *        *     Request timed out.
  4     1 ms     1 ms     1 ms  graz1.aco.net [193.171.21.41]
  5     5 ms     5 ms     5 ms  aconet-ias-aconet-gw.vie.at.geant.net [83.97.88.2]
  6     6 ms    11 ms     8 ms  aconet-ias-geant-gw.vie.at.geant.net [83.97.88.1]
  7     5 ms     5 ms     5 ms  193.203.0.65
  8     5 ms     5 ms     4 ms  151.101.193.69
```