CS 1680
IP Forwarding realities

Nick DeMarinis

Based partly on lecture notes by Rodrigo Fonseca, David Mazières, Phil Levis, John Jannotti

Administrivia

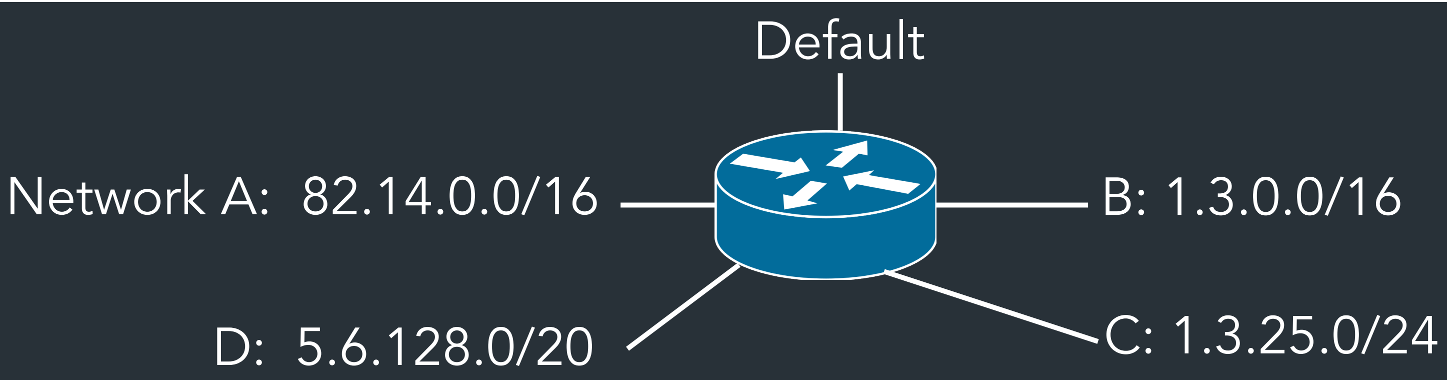
- Look for announcement to sign up for IP milestone meetings, preferably with your mentor TA, on or before Friday (Oct 4)
  - You don't need to show an implementation, but you are expected to talk about your design
- IP gearup II: Thursday 6-8pm in CIT368
  - Implementation and debugging tips
- HW1: Due Thursday (HW2 out Thursday)

Today

Odds and ends that make IP forwarding actually work

- Longest Prefix Match
- IP<->Link layer (ARP, DHCP)
- Network Address Translation (NAT)

After this: Routing



Prefix	IF/Next hop
82.14.0.0/16	(A)
1.3.0.0/16	(B)
1.3.4.0/24	(C)
5.6.128.0/20	(D)
0.0.0.0/0	(Default)

(X) is placeholder—could be an IP or an interface name

<u>Warmup</u>: based on the table, where would the router send packets destined for the following addresses:

1. 5.6.128.100 D
2. 1.3.1.1 B
3. 8.8.8.8 DEFAULT
4. 1.3.4.8 B OR C?

"Local delivery": what does it mean to send to IF1?

So far: "easy" to communicate with nodes on the same network. But how?

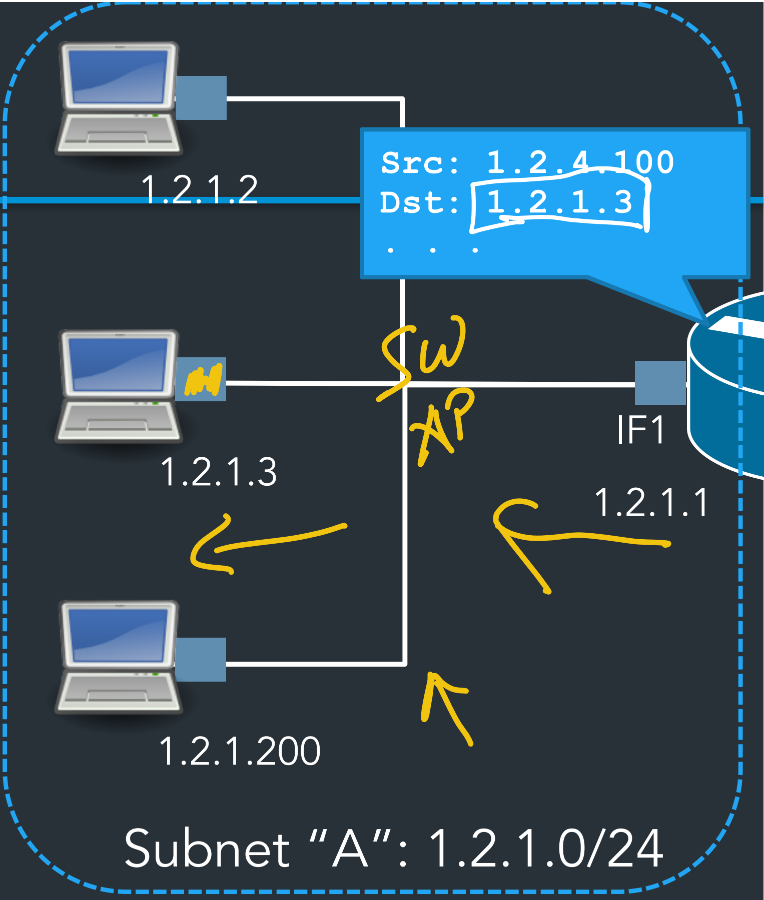
IN ORDER TO SEND ON LOCAL NET, NEED:
- DEST IP (L3)
- DEST MAC ADDRESS

	Src	Dest
Link	Know	???
IP	10.2.4.100	1.2.1.3

ETH/WIFI/...

HEADER INFO

NEED TO FIND THIS SOMEHOW.



Prefix	IF/Next hop
1.2.1.0/24	IF1
…	…

"Local delivery": what does it mean to send to IF1?

So far: "easy" to communicate with nodes on the same network. But how?

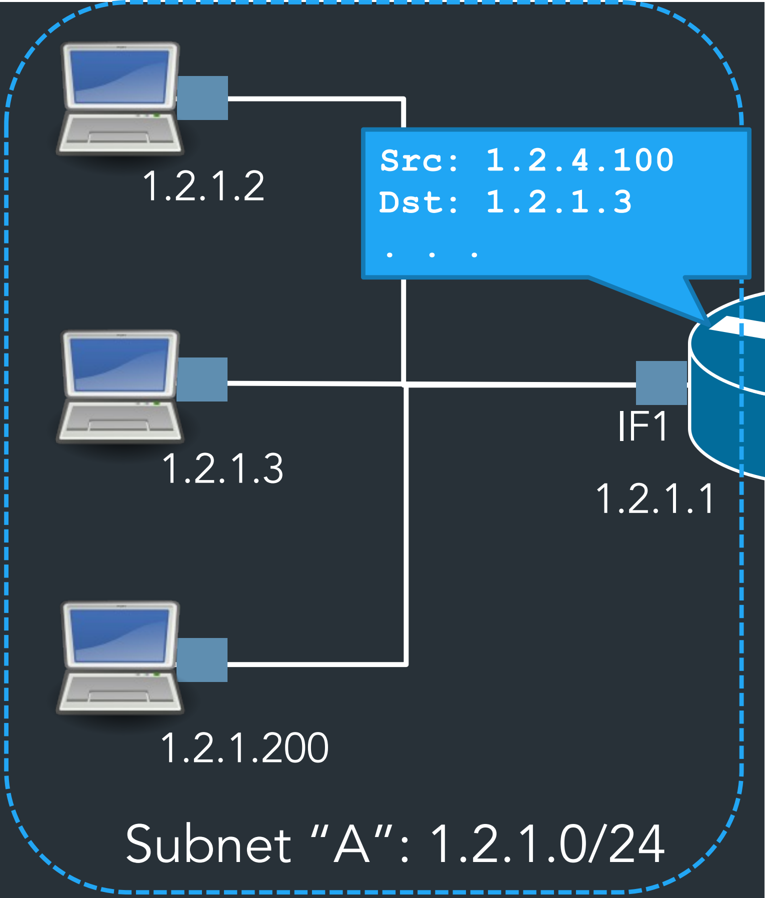
To send a packet on a local network, we need:

- Dest. IP (Network layer)
- Dest. MAC address (Link layer)

	Src	Dest
Link		???
IP	10.2.4.100	1.2.1.3

Assume: link layer can figure out the rest once we fill in this info

=> How do we get the MAC address?



Prefix	IF/Next hop
1.2.1.0/24	IF1
…	…

“Glue” between L2 and L3

Need a way to connect get link layer info (mac address) from network-layer info (IP address)

“What MAC address has IP 1.2.3.4?”

“Glue” between L2 and L3

Need a way to connect get link layer info (mac address) from network-layer info (IP address)

“What MAC address has IP 1.2.3.4?”

Solution: ask the network!
=> Address Resolution Protocol (ARP)

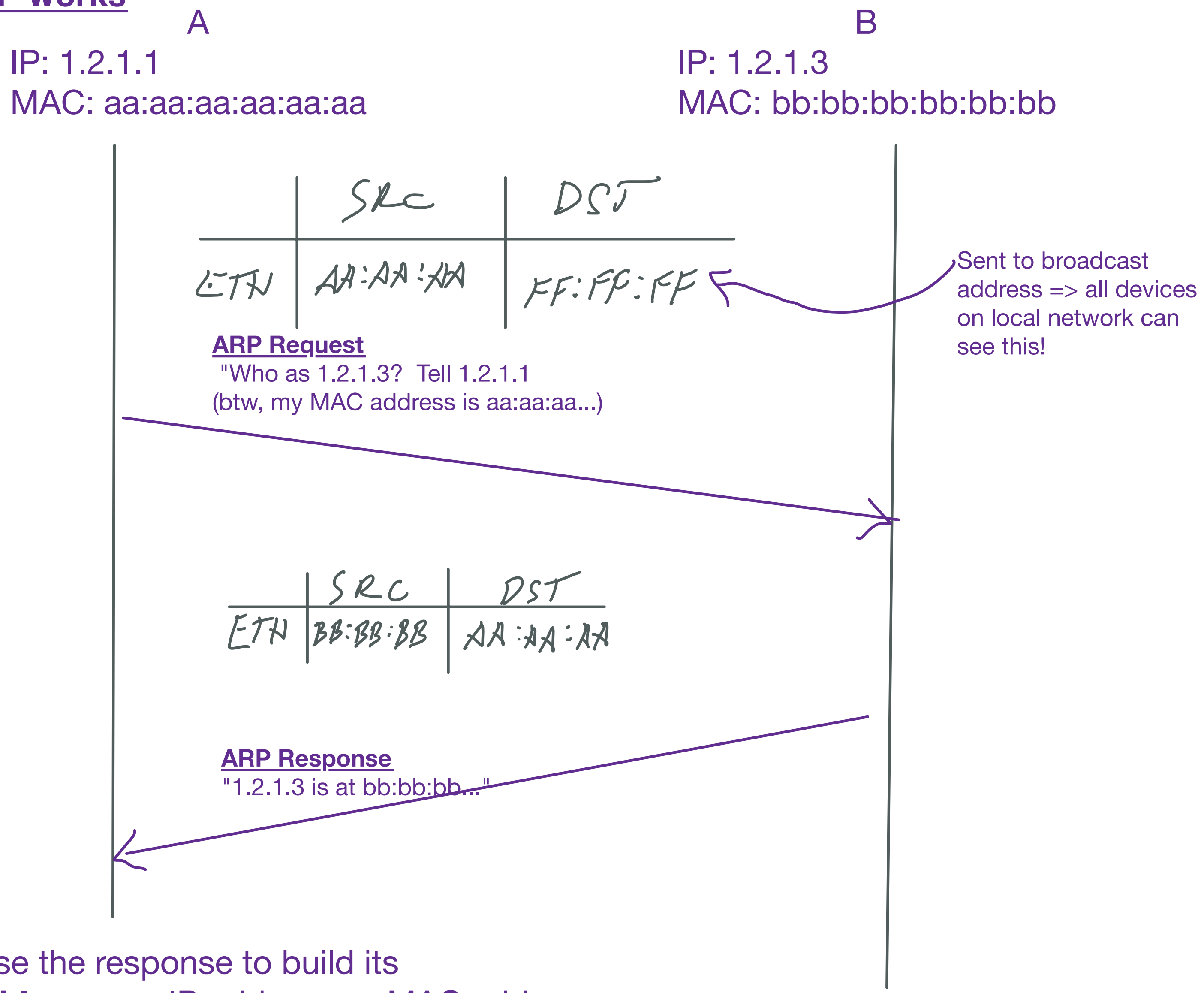
ARP: Address resolution protocol

Given an IP address, ask network for the MAC address

Request: “`Who has 1.2.3.4?`”

Response: “`aa:bb:cc:dd:ee:ff is at 1.2.3.4`”

How ARP works



A can use the response to build its
MAC table: maps IP address => MAC address

IP	MAC
1.2.1.3	BB:BB:BB

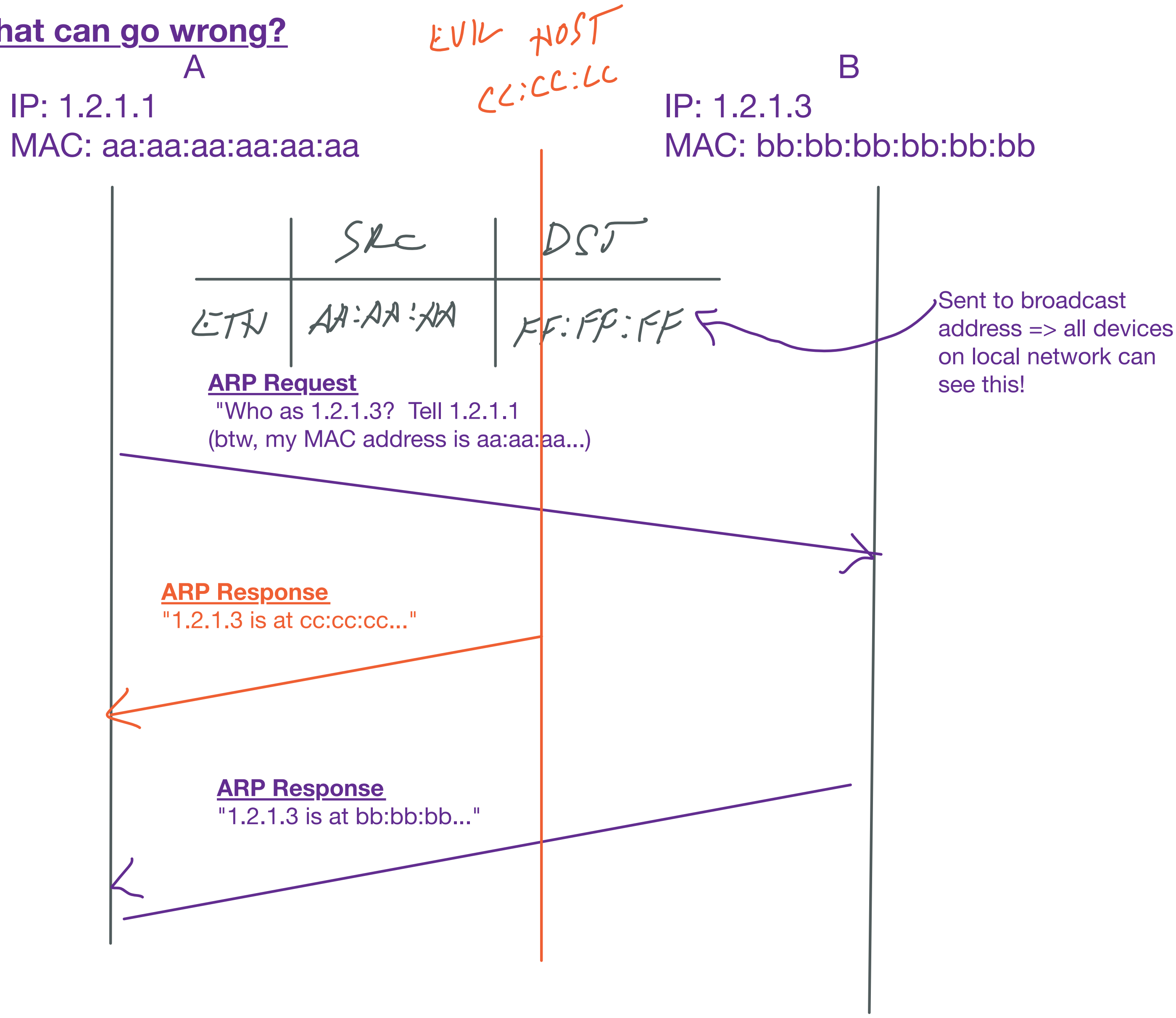
In a normal scenario:
- Request sent to broadcast address,
- B responds, A can add entry to its MAC table to know MAC address of B

(response might be sent broadcast too, in which case any host can update their table)

In general, when sending a packet to any IP on the local network, need to send an ARP first to look up MAC address => can cache results after first packet (some OS-specific timeout on how long entries last, etc.)

=> Problem: hosts can lie => ARP poisoning

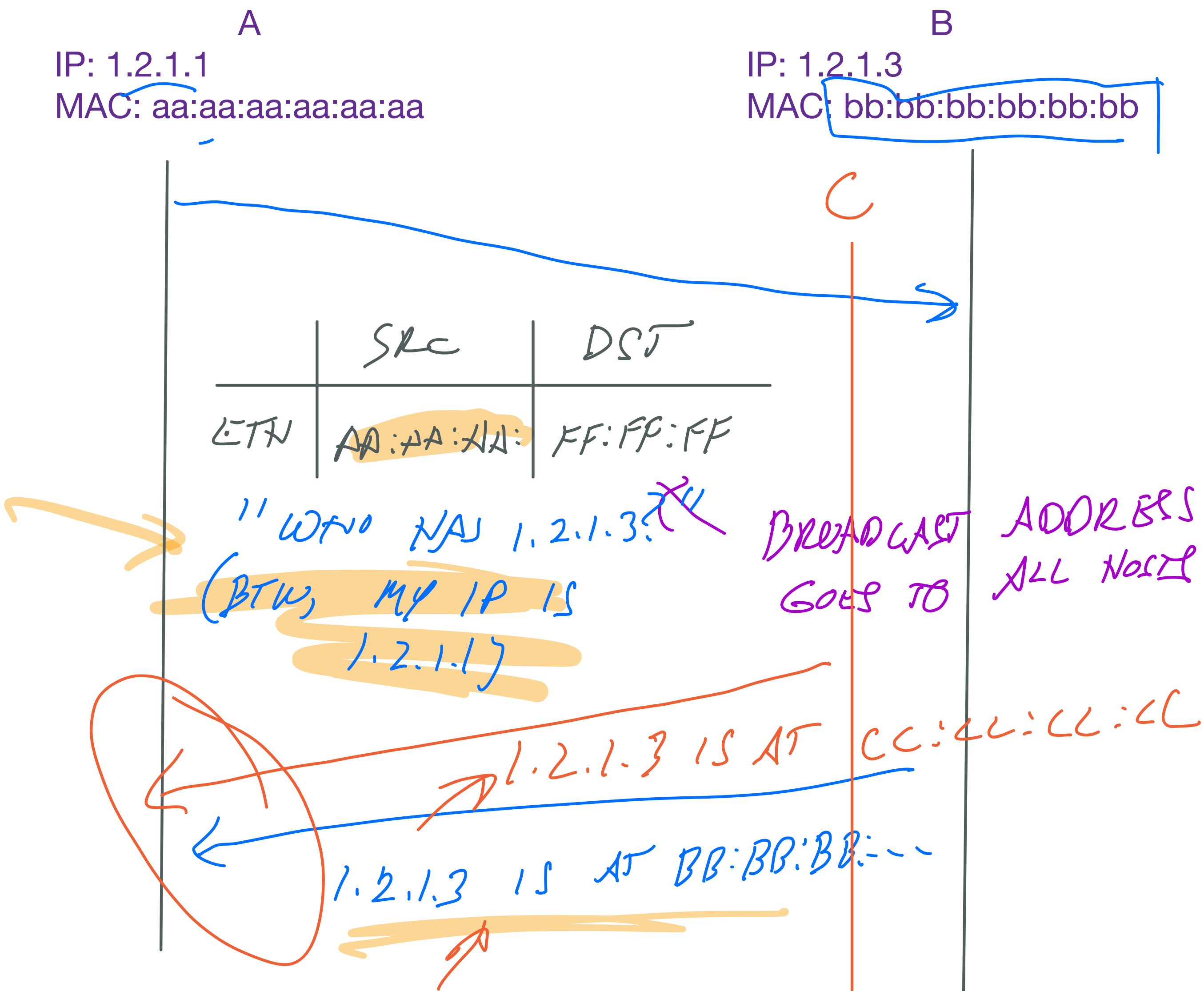
ARP: what can go wrong?



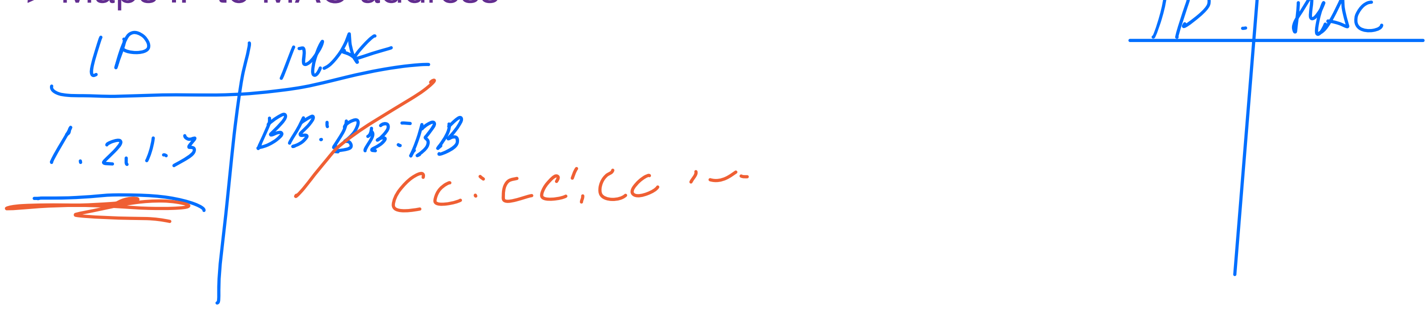
Problem: hosts could lie and send malicious ARP responses
=> Depending on timing, can overwrite/replace/win race and "poison" host's ARP table with invalid information => can use to intercept traffic!

=> Attacks like this are well known and have been possible for a long time. Modern networks can do some things to detect/prevent it in some cases.

Eg. - two ARP responses for different MACs in short time? sus.
- tons of ARP responses from one device in a short time? sus.



A can build a MAC address table
=> Maps IP to MAC address



In a normal scenario:

- Request sent to broadcast address,
- B responds, A can add entry to its MAC table to know MAC address of B
- Respones/requests sent broadcast, everyone in network can update their tables

=> Problem: hosts can lie => ARP poisonig

ARP: Address resolution protocol

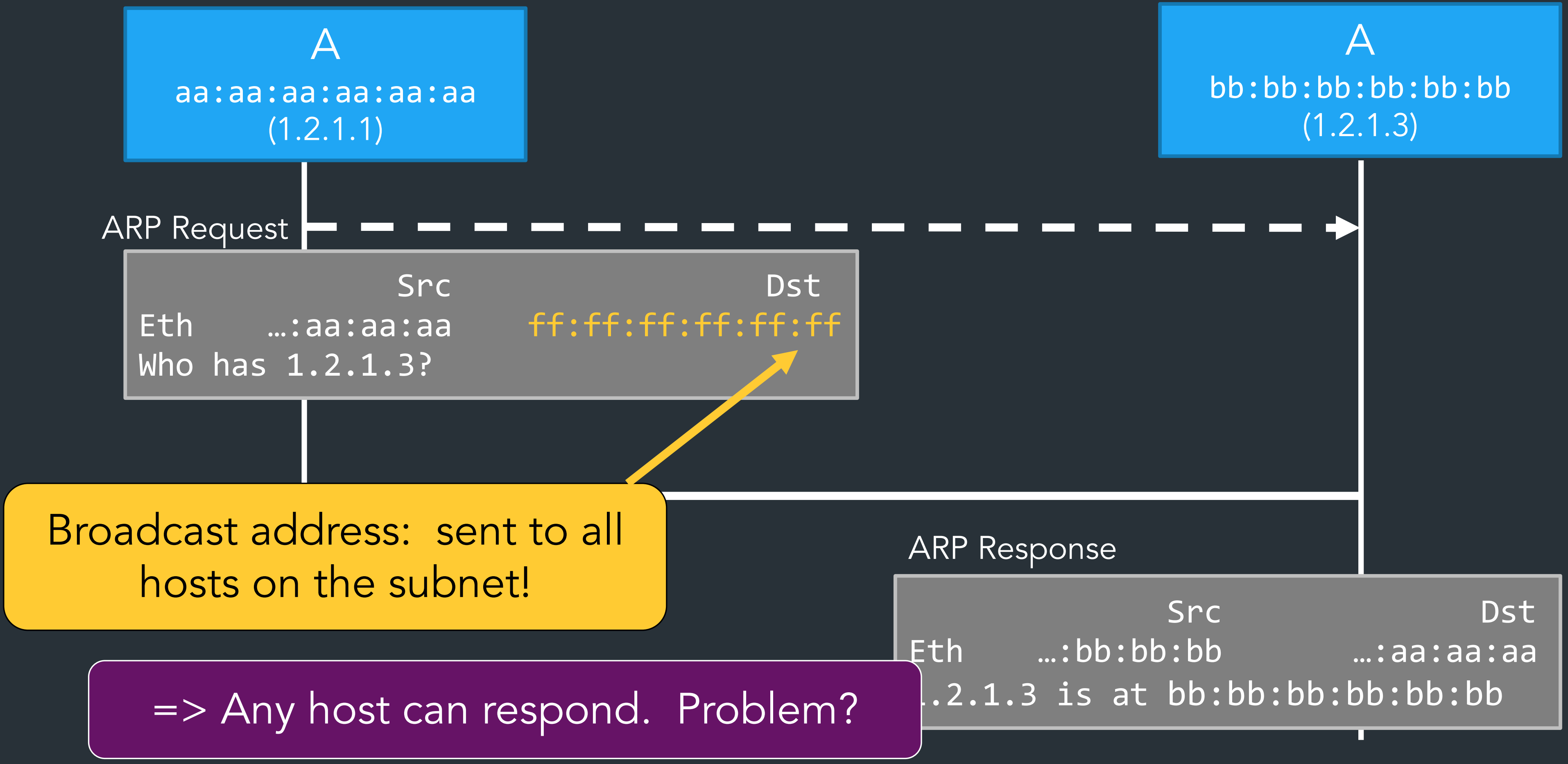
Given an IP address, ask network for the MAC address

Request: "`Who has 1.2.3.4?`"

Response: "`aa:bb:cc:dd:ee:ff is at 1.2.3.4`"

Key data structure: ARP table: map of IP -> MAC address

- All devices use ARP protocol to build their own table
- Requests send to *broadcast address*: `ff:ff:ff:ff:ff:ff`



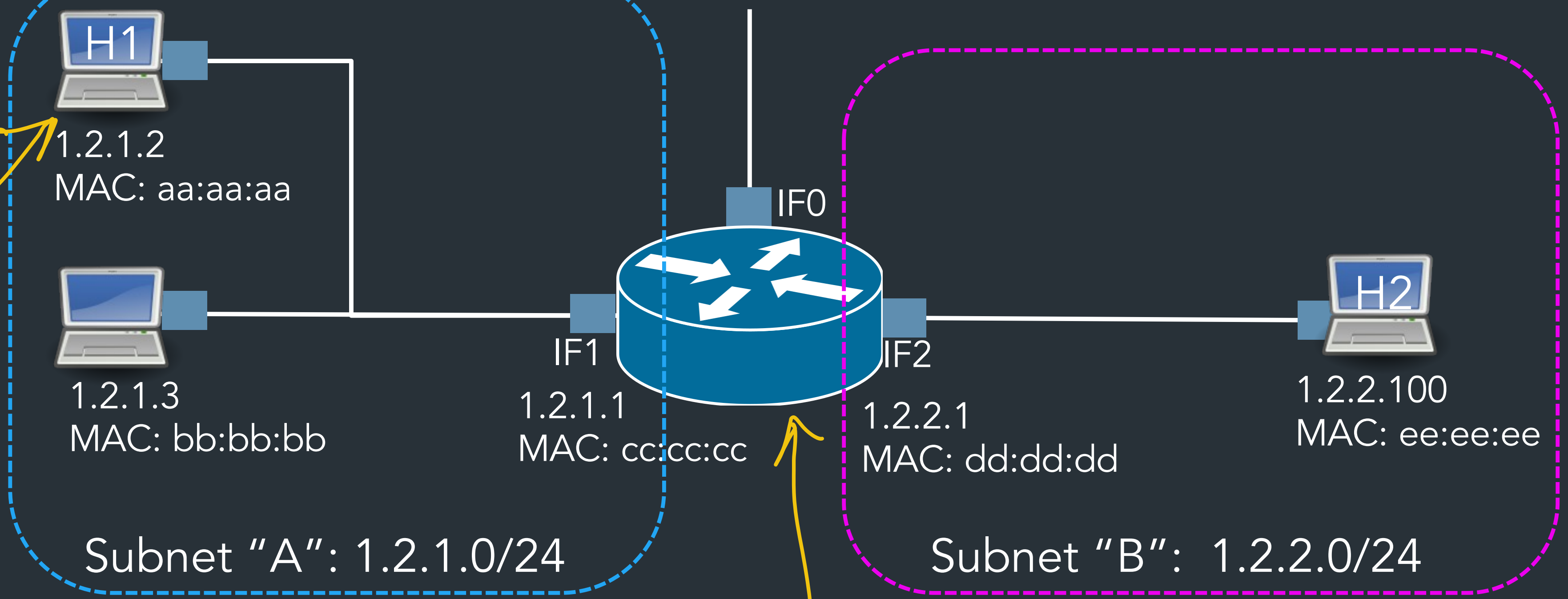
A
aa:aa:aa:aa:aa:aa
(1.2.1.1)
A
bb:bb:bb:bb:bb:bb
(1.2.1.3)
ARP Request
Src
Dst
Eth
…:aa:aa:aa
ff:ff:ff:ff:ff:ff
Who has 1.2.1.3?
Broadcast address: sent to all hosts on the subnet!
ARP Response
Src
Dst
Eth
…:bb:bb:bb
…:aa:aa:aa
..2.1.3 is at bb:bb:bb:bb:bb:bb
=> Any host can respond. Problem?

Example

```
# arp -n
Address                  HWtype  HWaddress           Flags Mask            Iface
172.17.44.1              ether   00:12:80:01:34:55   C                     eth0
172.17.44.25             ether   10:dd:b1:89:d5:f3   C                     eth0
172.17.44.6              ether   b8:27:eb:55:c3:45   C                     eth0
172.17.44.5              ether   00:1b:21:22:e0:22   C                     eth0
```

Putting it all together….

Example: a network with two subnets. Here's what we know:

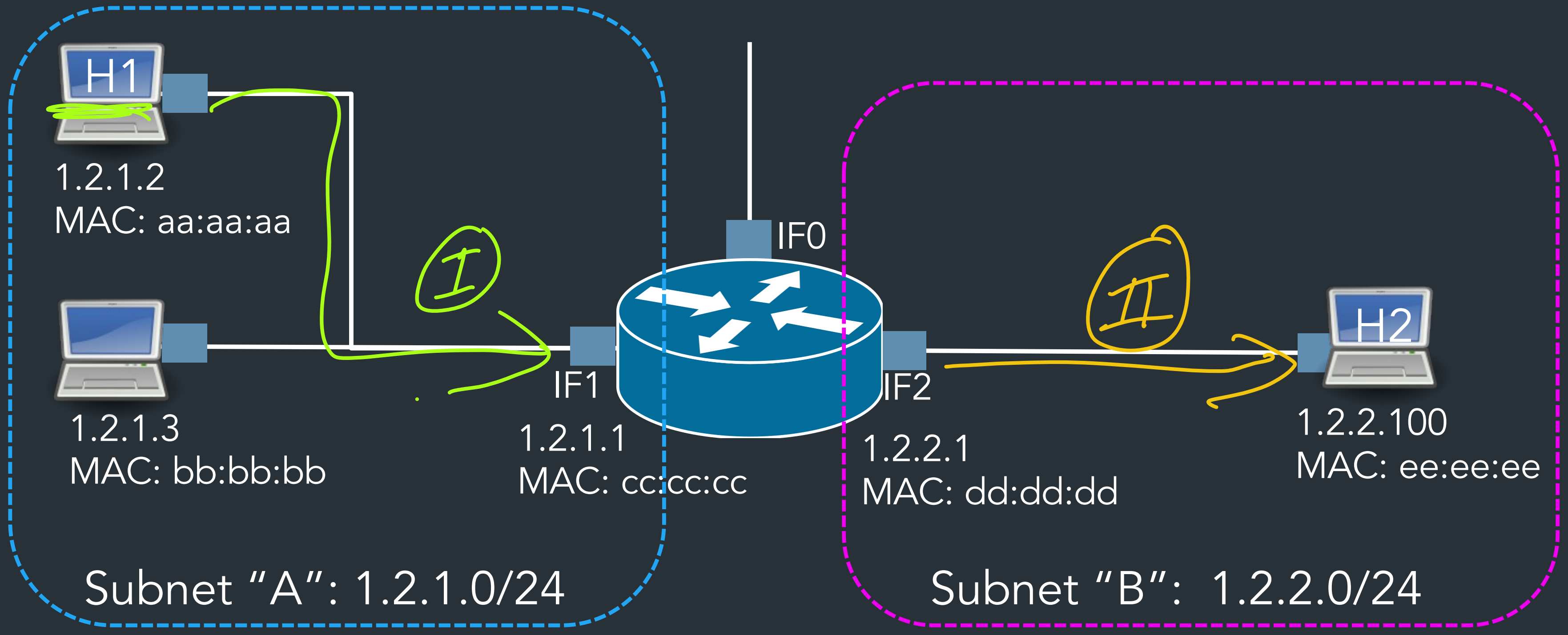


H1's forwarding table

Prefix	IF/Next hop
1.2.1.0/24	IF0
0.0.0.0/0	1.2.1.1

Router's forwarding table

Prefix	IF/Next hop
1.2.1.0/24	IF1
1.2.2.0/24	IF2



Suppose H1 wants to send a packet to H2.

Q: What would the headers look like when the packet leaves H1?

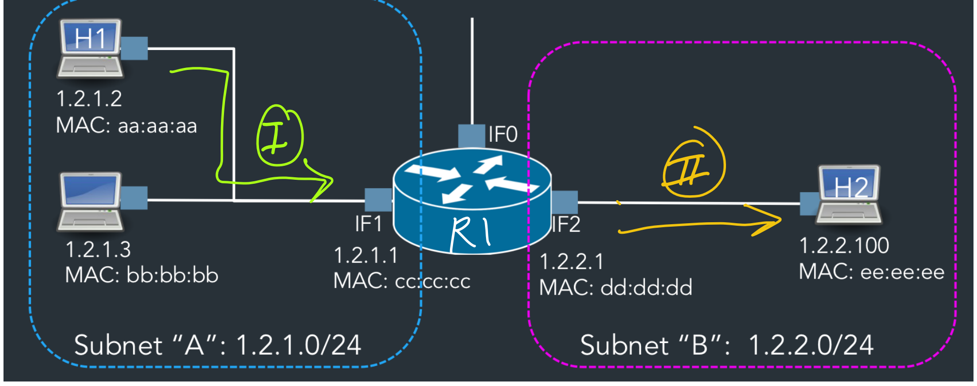
Q: Would it change after reaching R?

Based on what we know about forwarding, we know the packet needs to travel two hops. (I) (II)

=> But what will the headers look like?

Forwarding example: in detail

Goal: h1 wants to send a packet to h2.
For example: on h1, user types:
"ping 10.2.2.100"



How to READ: FOLLOW THE NUMBERS!

I. What happens on H1?

1. To start, it can begin filling in the IP src and dest fields for H1 (its own IP) and H2 (dest specified by user) (2B) (2A)

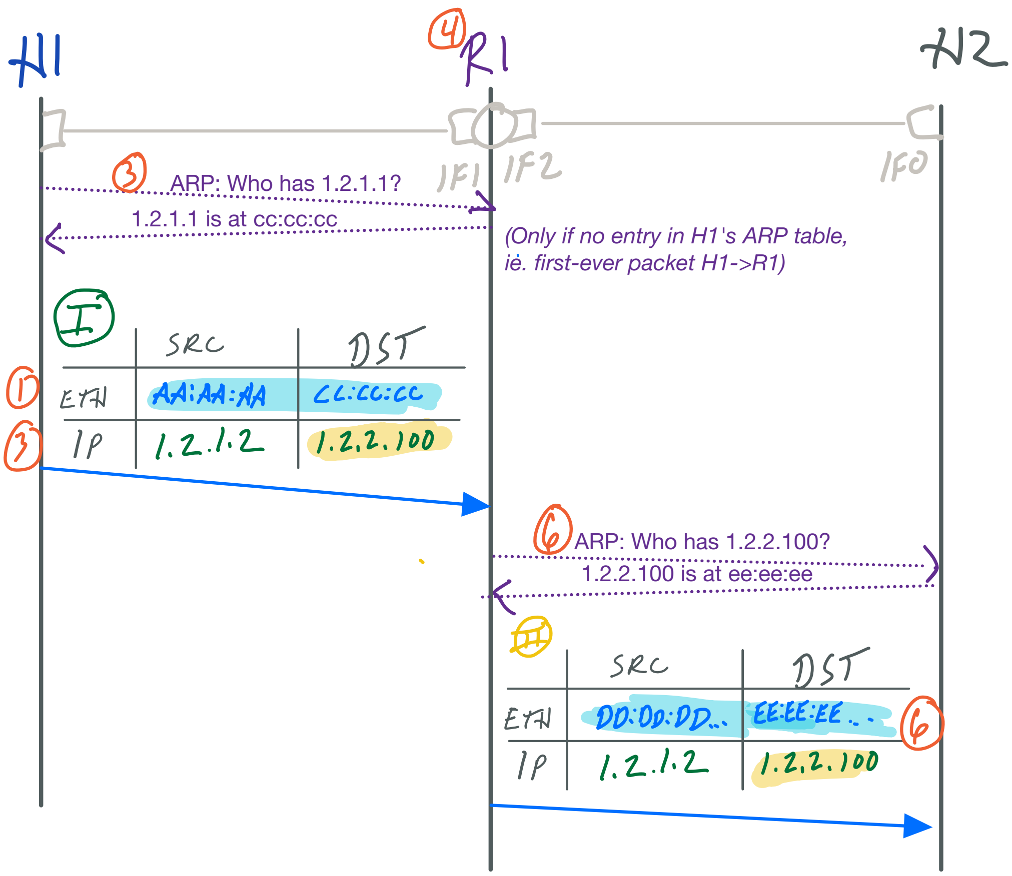
H1'S FORWARDING TABLE

Prefix	IF/Next hop
1.2.1.0/24	IF0
0.0.0.0/0	1.2.1.1

Q: How can there be an IF0 on both H1 and R1?
Interfaces are named on a per-device level by the OS
=> Can have repeats on different devices
In our examples, interface names start at "IF0" on each device

2. Next, H1 checks its forwarding table:
   a. **1.2.2.100 is NOT on H1's local network => match on default route**, which gives next hop IP of router (R1), 1.2.1.1
   b. **H1 then looks up 1.2.1.1 in its forwarding table => send on IF0**
   This means that the 1.2.2.1 (ie, R1) must be connected on the same subnet (ie, a neighbor) on H1's IF0.

3. **To send the packet on IF0, we need to know a MAC address we can use to reach R1.**
=> Consult its ARP table for an IP matching 1.2.1.1. If no entry, send an ARP request "who has 1.2.1.1?"
=> cc:cc:cc + update table for future



What happens on R1?

4. R1 checks the destination IP on the packet (1.2.2.100).
This doesn't match one of its IPs, so it needs to forward the packet.

5. R1 checks its forwarding table for 1.2.2.100 **=> match on IF2**

R1'S TABLE

Prefix	IF/Next hop
1.2.1.0/24	IF1
1.2.2.0/24	IF2

(5)

6. To send on IF2, R1 needs to fill in a new link-layer header for the packet:
- **Source MAC address: interface sending the packet (IF2, dd:dd:dd)**
- **Dest MAC address: MAC address of H2, ee:ee:ee**
  (if unknown, would need to send ARP to find it)

Finally, R1 can forward the packet! (Also needs to decrement TTL, recompute checksum, etc.)

What happens on H2?

H2 receives the packet, checks (dest IP == its own IP) => sends packet to part of OS that handles ping packets

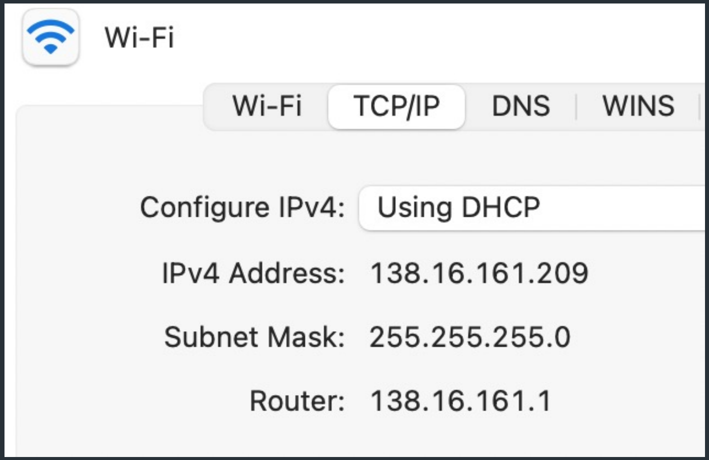
Putting it all together

Why do we need both MAC IP addresses?

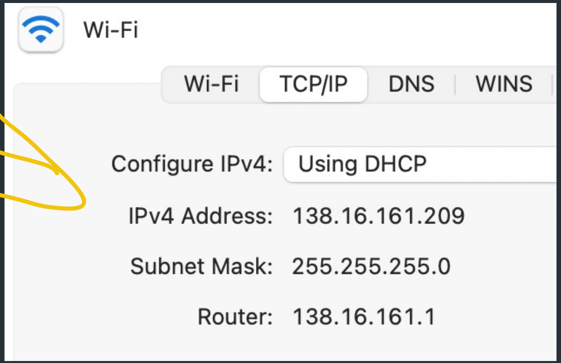
MAC/Link-layer address: info about **where packet is going on this link (next hop)**
- Changes every hop

IP address: info **packet's final destination**
- Persists across hops

How do you get an IP address?



Getting an IP



Two ways to configure an IP for a host:

- Static configuration: manually specify IP address, mask, gateway, … Only use this for devices that don't change often

- Automatic: ask the network for an IP when you connect!

  => More common for end hosts
  => DHCP: Dynamic Host Configuration Protocol (end hosts, home routers)

Getting an IP

Two ways to configure an IP for a host:

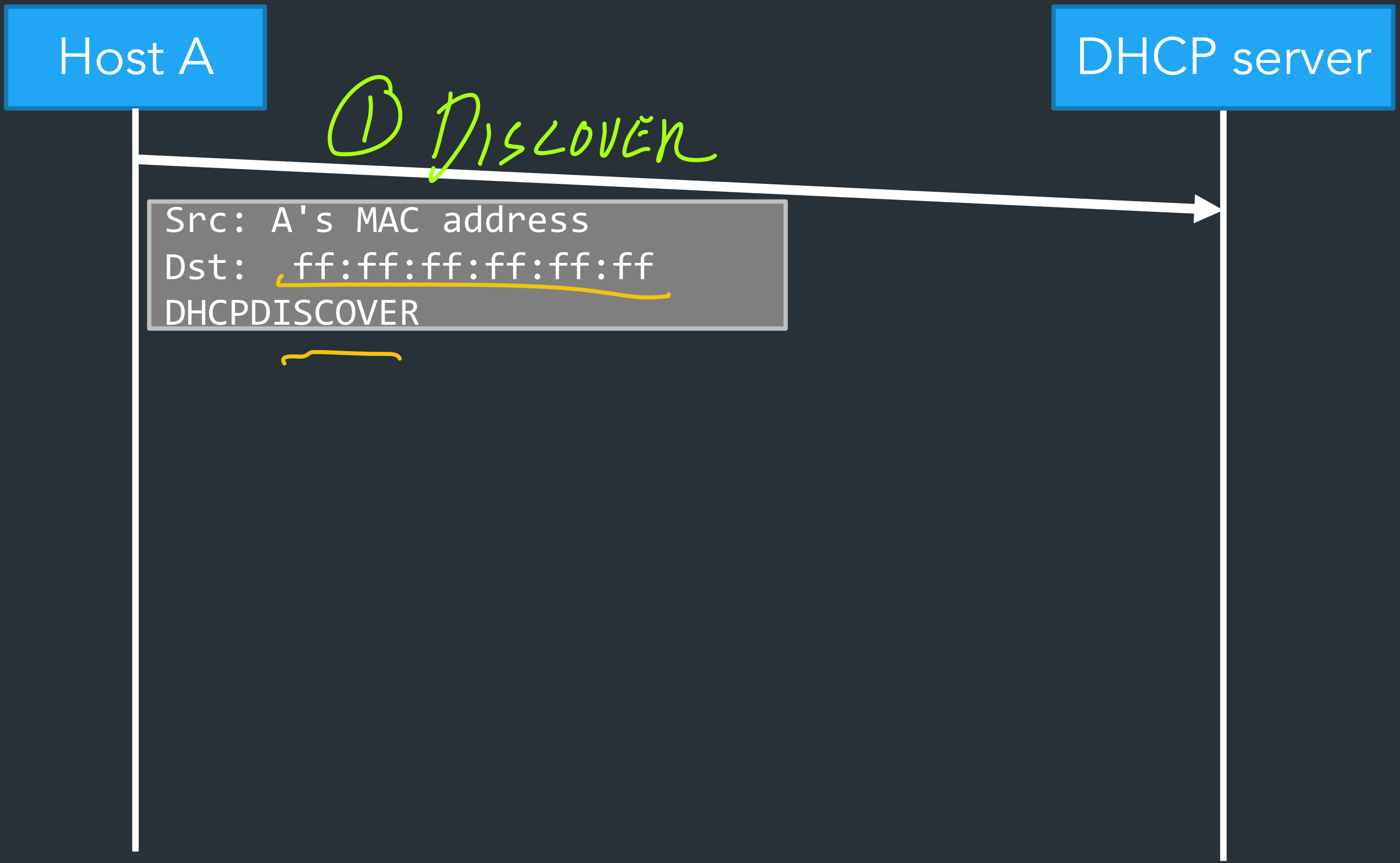
- Static configuration: manually specify IP address, mask, gateway, …

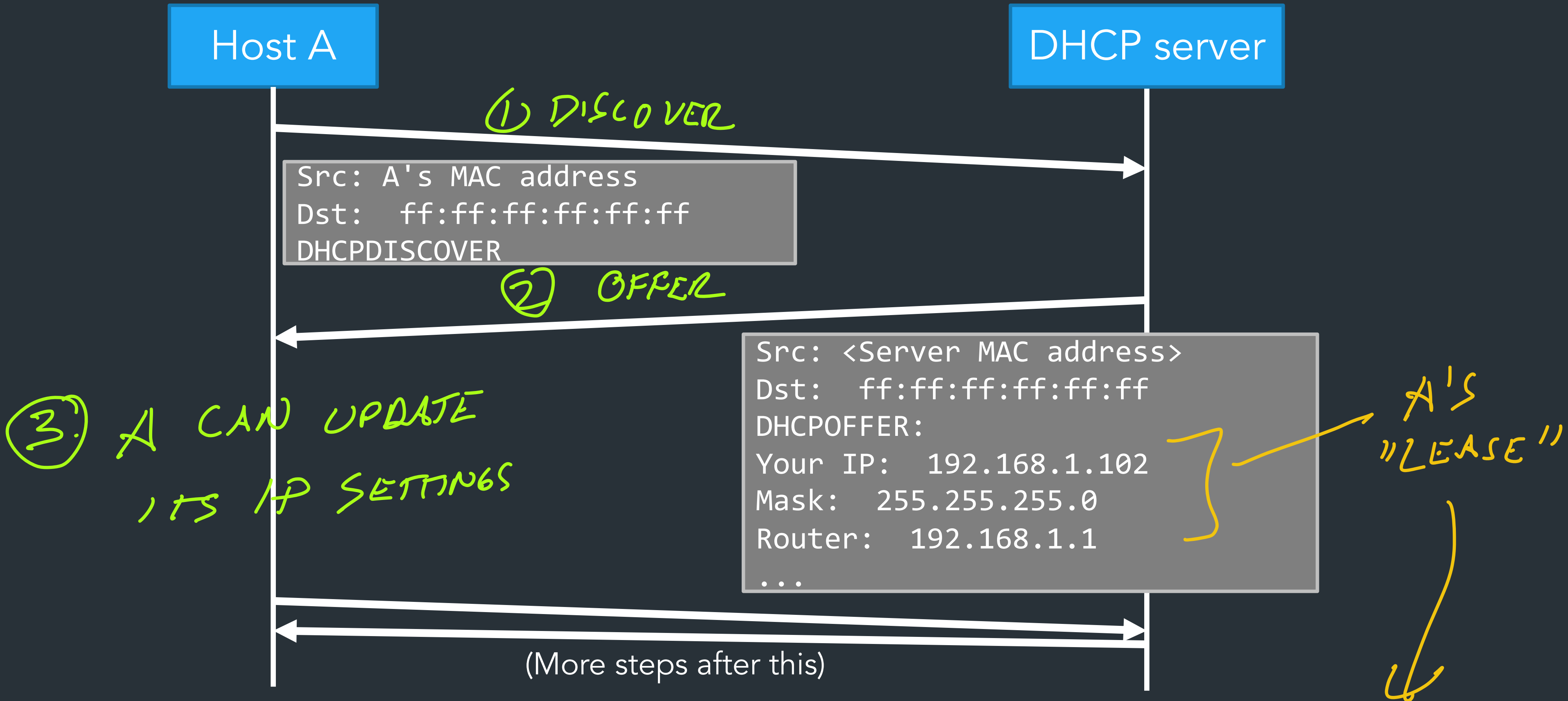
  => More common with network devices that don't change often

- Automatic: ask the network for an IP when you connect!

  => Most common for end hosts

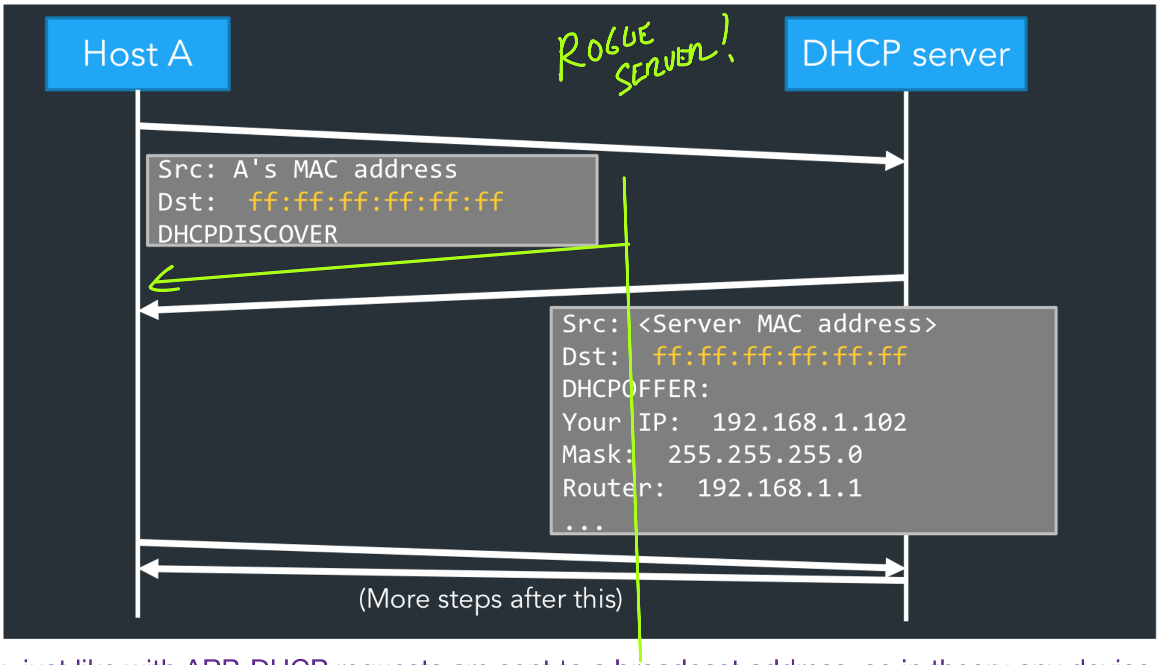
  => Dynamic Host Configuration Protocol (DHCP)


Host A
DHCP server
1 Discover
Src: A's MAC address
Dst: ff:ff:ff:ff:ff:ff
DHCPDISCOVER



The server's offer provides enough info for A to configure its IP settings (eg. address, mask, default gateway, + other info we'll discuss later
=> Known as "DHCP options"

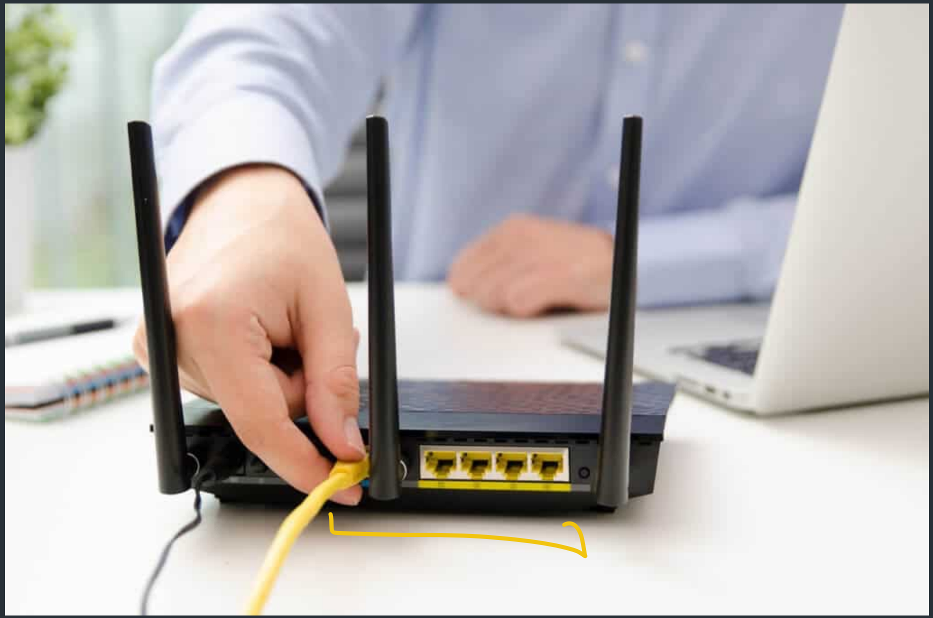
DHCP: What can go wrong?



Problem: just like with ARP, DHCP requests are sent to a broadcast address, so in theory any device can pretend to be the DHCP server and respond.

This would cause hosts to have incorrect settings, and could be a security risk (eg. the DHCP server could even configure the host's settings to intercept its traffic)!

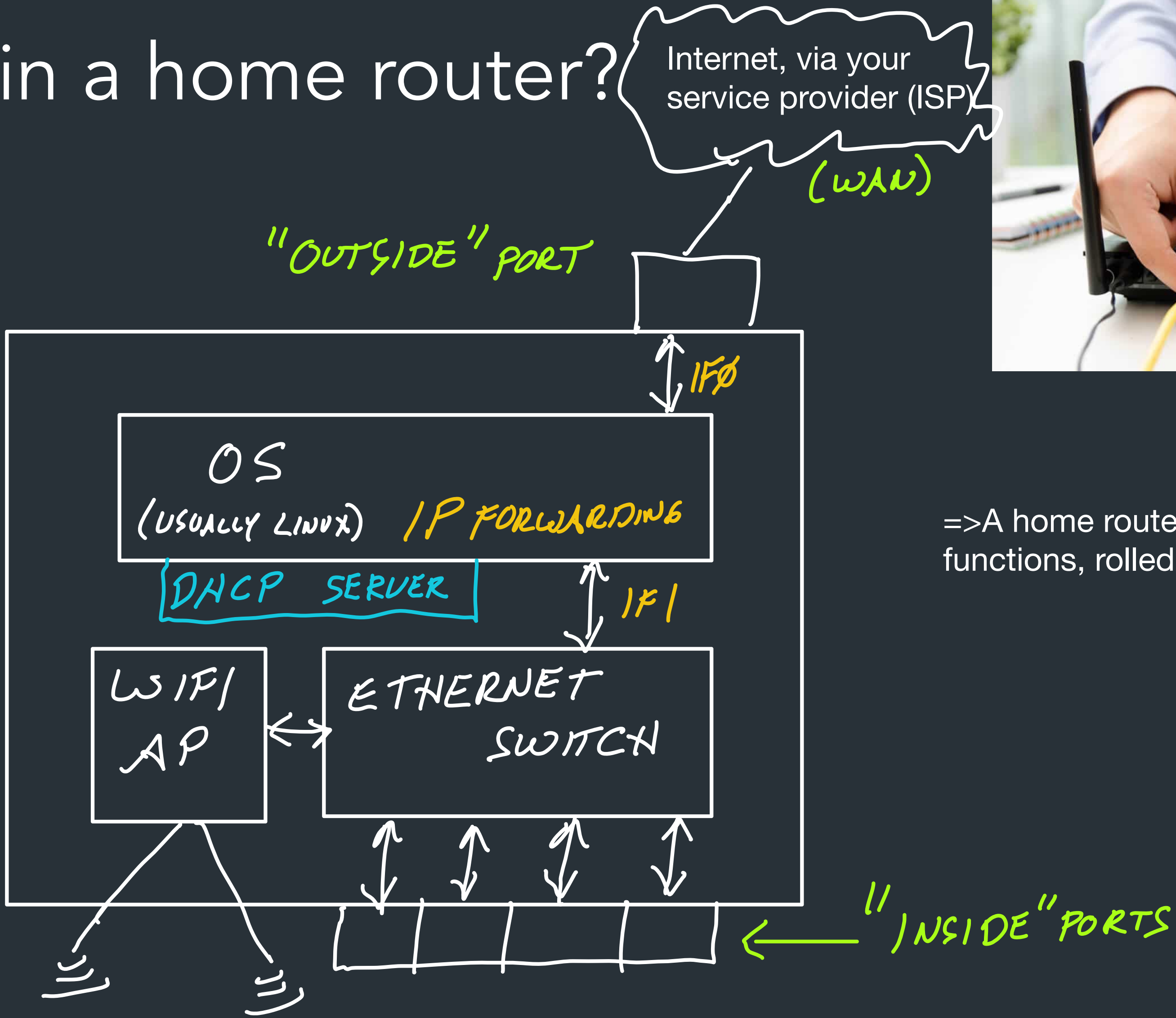
However, this problem is often detectable: network devices and other DHCP servers can monitor DHCP traffic, and could detect when a) a rogue server is detected handing out leases, b) someone is using an IP that wasn't given out as a lease from a valid DHCP server.

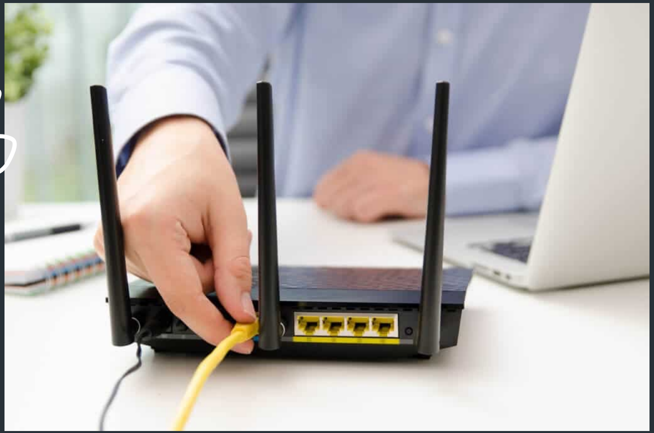


Home routers

The good, the bad, and the ugly…

What's in a home router?

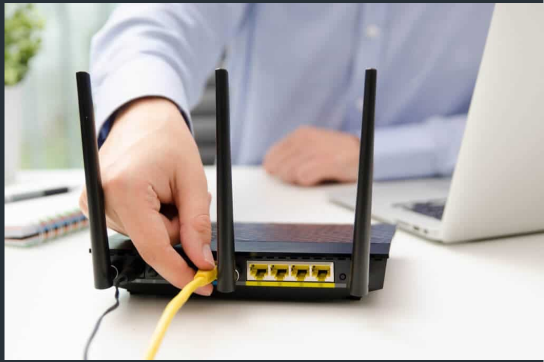




=>A home router performs all of these functions, rolled into one device!

Story time

Where it gets weird...



Try this at home!

1. Open up your IP settings and look at your IP address

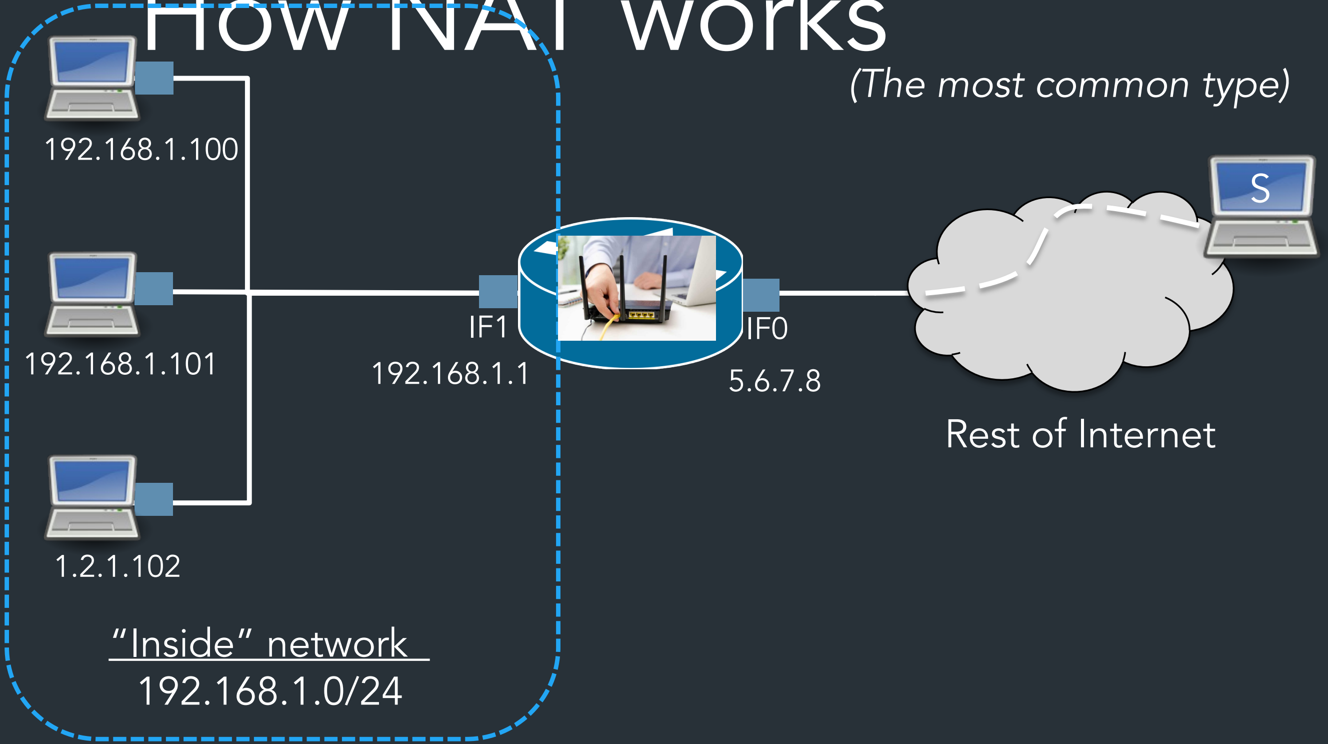
2. Open a browser and search "my IP address"

Are the results the same????? Nope!

WHY???? NEXT LECTURE!

How NAT works

(The most common type)



Goal: Share one IP among many hosts on a private network

Router *translates (modifies) packets from "inside" to use "outside" address*

=> Router needs to remember connection state

=> Router makes some (sketchy) assumptions about traffic