# CSCI1680
# Network Layer: IP & Forwarding II

Nick DeMarinis

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

# Administivia

- IP Project: out later today
  - Partner form: due TONIGHT by 11:59pm
  - You will get an email confirming your team tomorrow morning
- IP gearup: tonight 5-7pm, CIT368
- IP Milestone: meet with me/a TA on/before next Friday (October 6) to discuss your design
  - (No *working* code yet, just some serious plans/sketches)
-

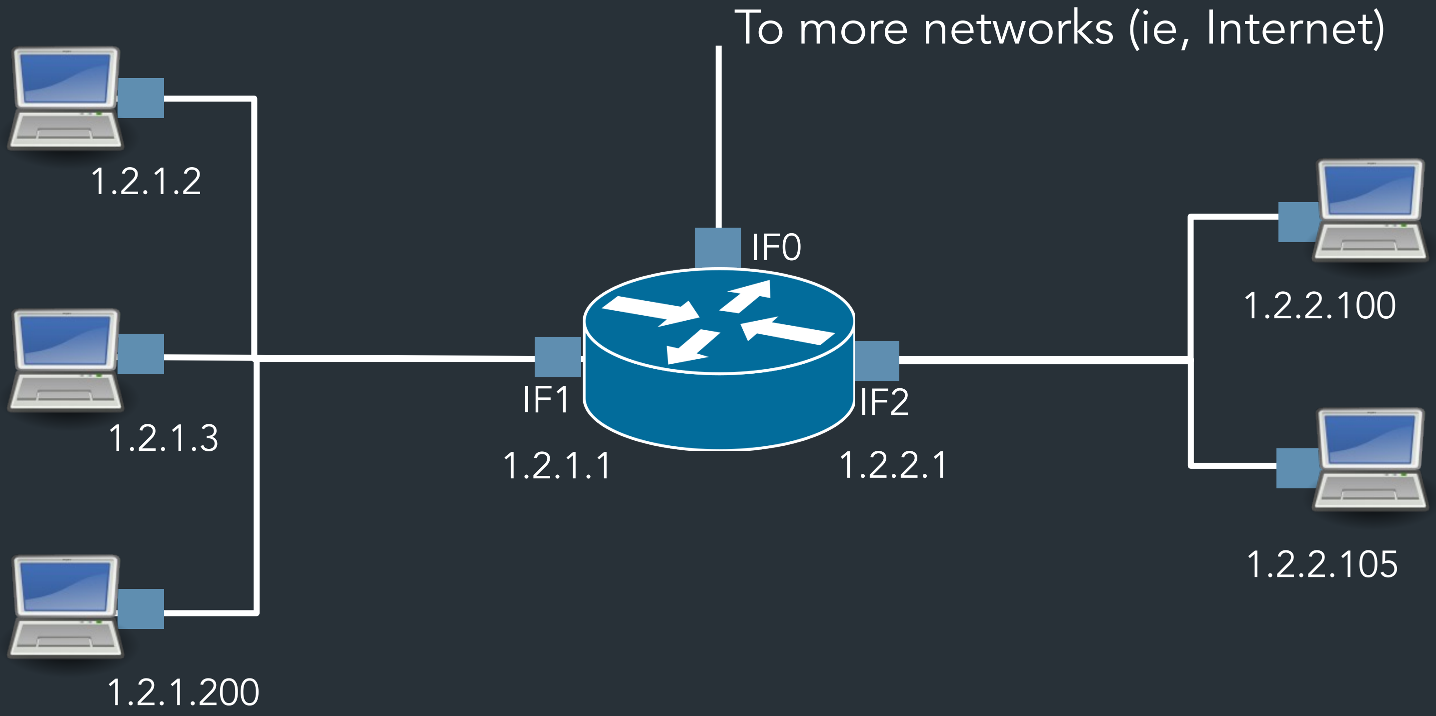
# Administivia

- IP Project: out later today
  - Partner form: due TONIGHT by 11:59pm
  - You will get an email confirming your team tomorrow morning
- IP gearup: tonight 5-7pm, CIT368
- IP Milestone: meet with me/a TA on/before next Friday (October 6) to discuss your design
  - (No *working* code yet, just some serious plans/sketches)
- HW1 (short): Due next Thursday

# Today

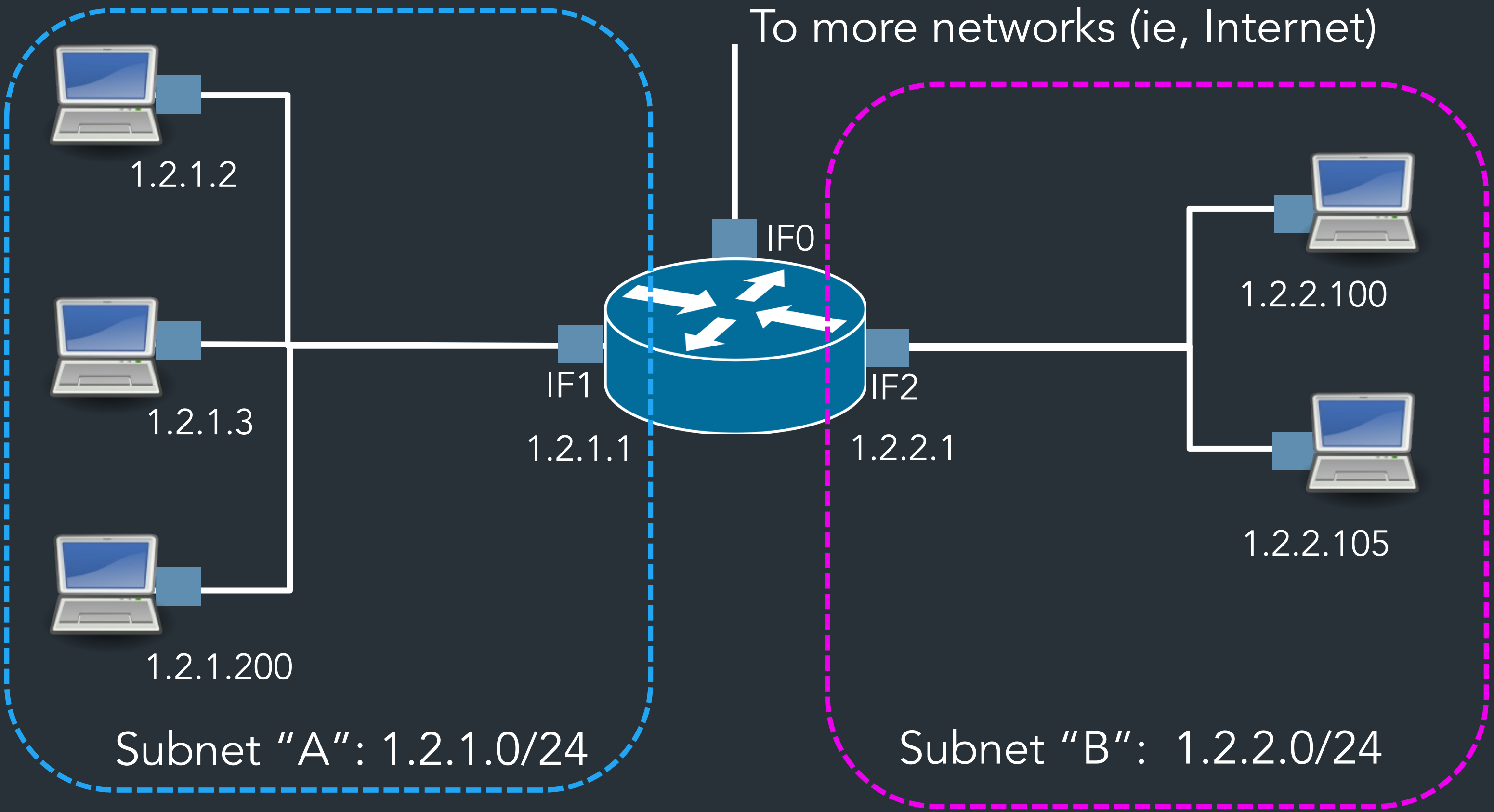
Continuing network layer

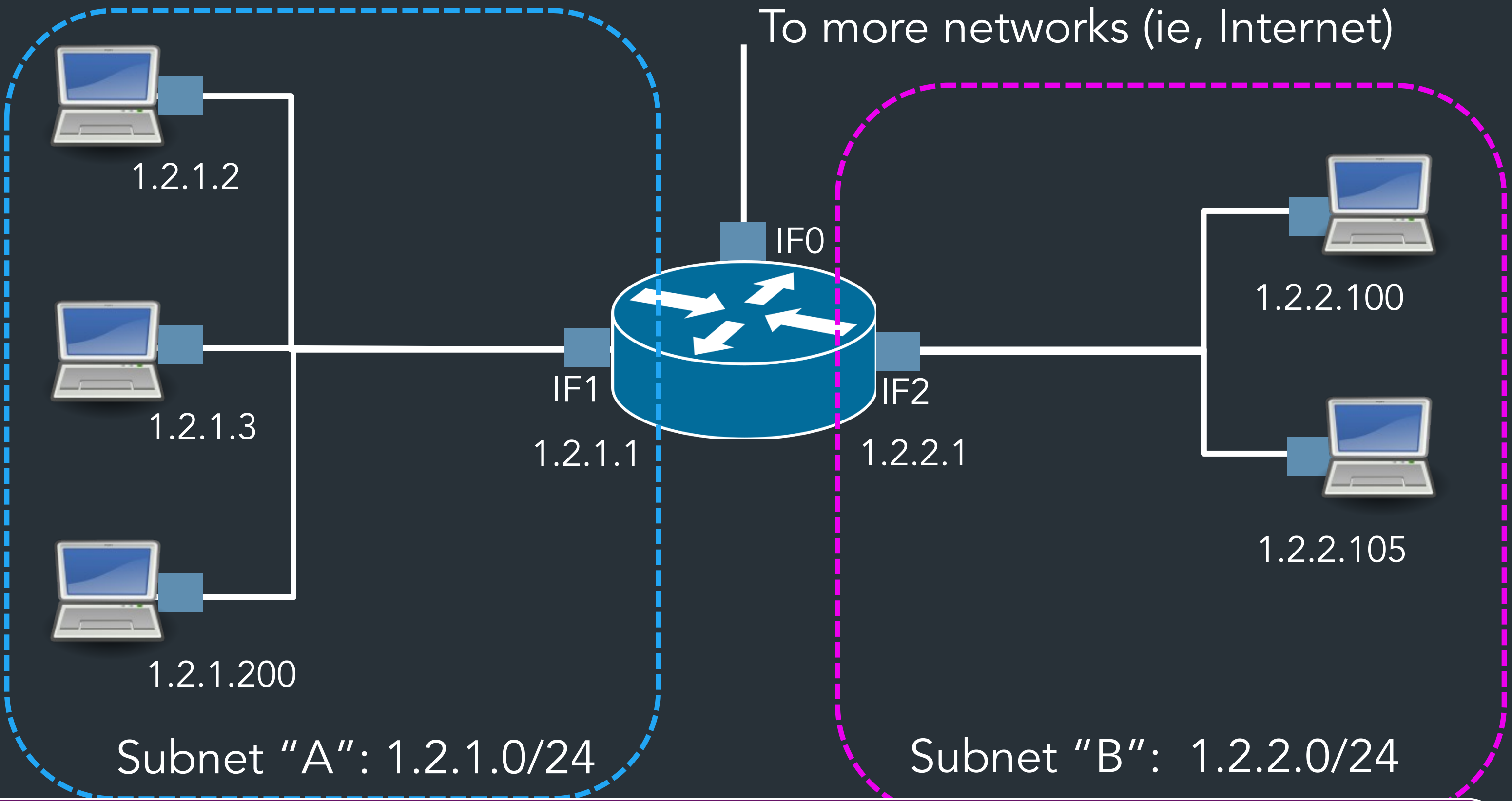
- IP forwarding mechanics
- About the IP project



Subnet “A”: 1.2.1.0/24

Subnet “B”: 1.2.2.0/24

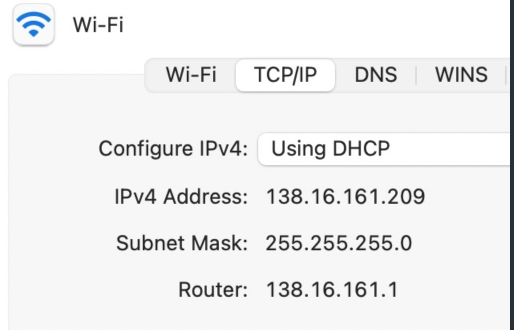

To more networks (ie, Internet)
IF0
IF1
1.2.1.1
IF2
1.2.2.1
1.2.1.2
1.2.1.3
1.2.1.200
1.2.2.100
1.2.2.105
Subnet "A": 1.2.1.0/24
Subnet "B": 1.2.2.0/24



The story so far:
=>Can "easily" communicate with nodes on the same network, but what about other networks?
=> Routers know about multiple networks, forward packets between them

*What does it mean to be on the same network?*

# All systems with an IP address have a configuration like this



Configure IPv4: Using DHCP
IPv4 Address: 138.16.161.209
Subnet Mask: 255.255.255.0
Router: 138.16.161.1

138.16.161.209

Addr:
138.16.161.209 10001010 00010000 10100001 11010001

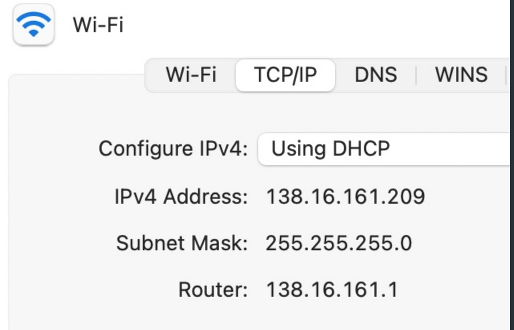
Mask:
255.255.255.0 & 11111111 11111111 11111111 00000000

$$\underbrace{10001010\ 00010000\ 10100001}_{\text{24 bits}}\ 00000000$$

138.16.161.0

Can also write as 138.16.161.209/24
"Prefix notation" or "CIDR notation"

# All systems with an IP address have a configuration like this


Wi-Fi

Configure IPv4: Using DHCP

IPv4 Address: 138.16.161.209

Subnet Mask: 255.255.255.0

Router: 138.16.161.1

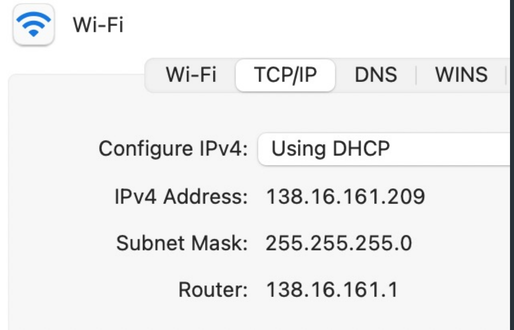
138.16.161.209

Addr:
138.16.161.209 `10001010 00010000 10100001 11010001`

Mask:
255.255.255.0 `11111111 11111111 11111111 00000000`

=> Bitmask used to “filter out” which part is for hosts on the same network

# All systems with an IP address have a configuration like this



Configure IPv4: Using DHCP

IPv4 Address: 138.16.161.209

Subnet Mask: 255.255.255.0

Router: 138.16.161.1

138.16.161.209

Addr:
138.16.161.209 10001010 00010000 10100001 11010001

Mask:
255.255.255.0 & 11111111 11111111 11111111 00000000

$\underbrace{10001010\ 00010000\ 10100001}_{\text{24 bits}}\ 00000000$

138.16.161.0

Can also write as 138.16.161.209/24
"Prefix notation" or "CIDR notation"

| | |
|---|---|
| 138.16.161.0/24 | `10001010 00010000 10100001 xxxxxxxx` |
| 138.16.161.204 | `10001010 00010000 10100001 10100001` |
| 1.2.3.4 | `00000001 00000010 00000011 00000100` |

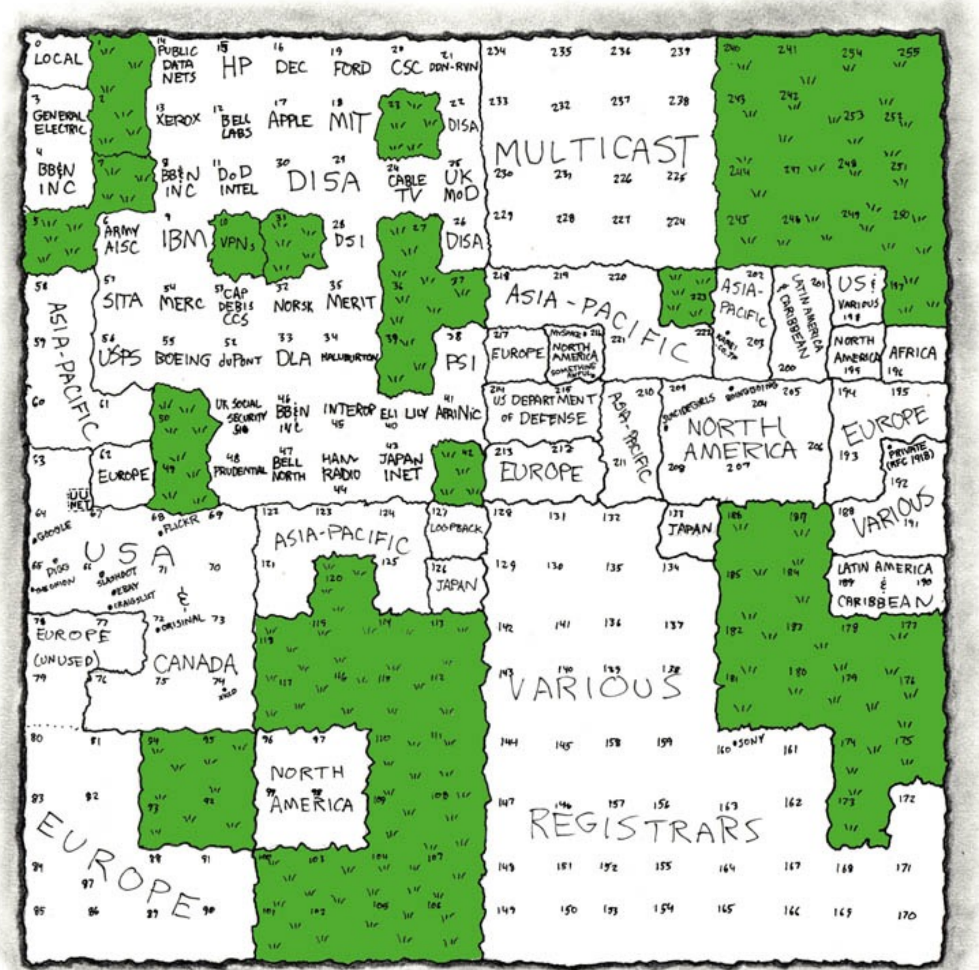
⇒ The mask can be any size 0-32
Not just checking the first three digits!

# Common prefixes

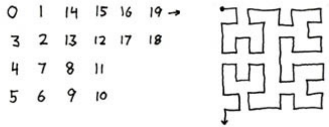
| | |
|---|---|
| 1.2.0.0/16 | `00000001 00000010 xxxxxxxx xxxxxxxx` |
| 8.0.0.0/8 | `00001000 xxxxxxxx xxxxxxxx xxxxxxxx` |
| 123.10.1.0/24 | `01111011 00001010 00000001 xxxxxxxx` |
| 201.112.10.200/30 | `11001001 01110000 00001010 110010xx` |

# MAP OF THE INTERNET

THE IPv4 SPACE, 2006



THIS CHART SHOWS THE IP ADDRESS SPACE ON A PLANE USING A FRACTAL MAPPING WHICH PRESERVES GROUPING -- ANY CONSECUTIVE STRING OF IPs WILL TRANSLATE TO A SINGLE COMPACT, CONTIGUOUS REGION ON THE MAP. EACH OF THE 256 NUMBERED BLOCKS REPRESENTS ONE /8 SUBNET (CONTAINING ALL IPs THAT START WITH THAT NUMBER). THE UPPER LEFT SECTION SHOWS THE BLOCKS SOLD DIRECTLY TO CORPORATIONS AND GOVERNMENTS IN THE 1990's BEFORE THE RIRs TOOK OVER ALLOCATION.



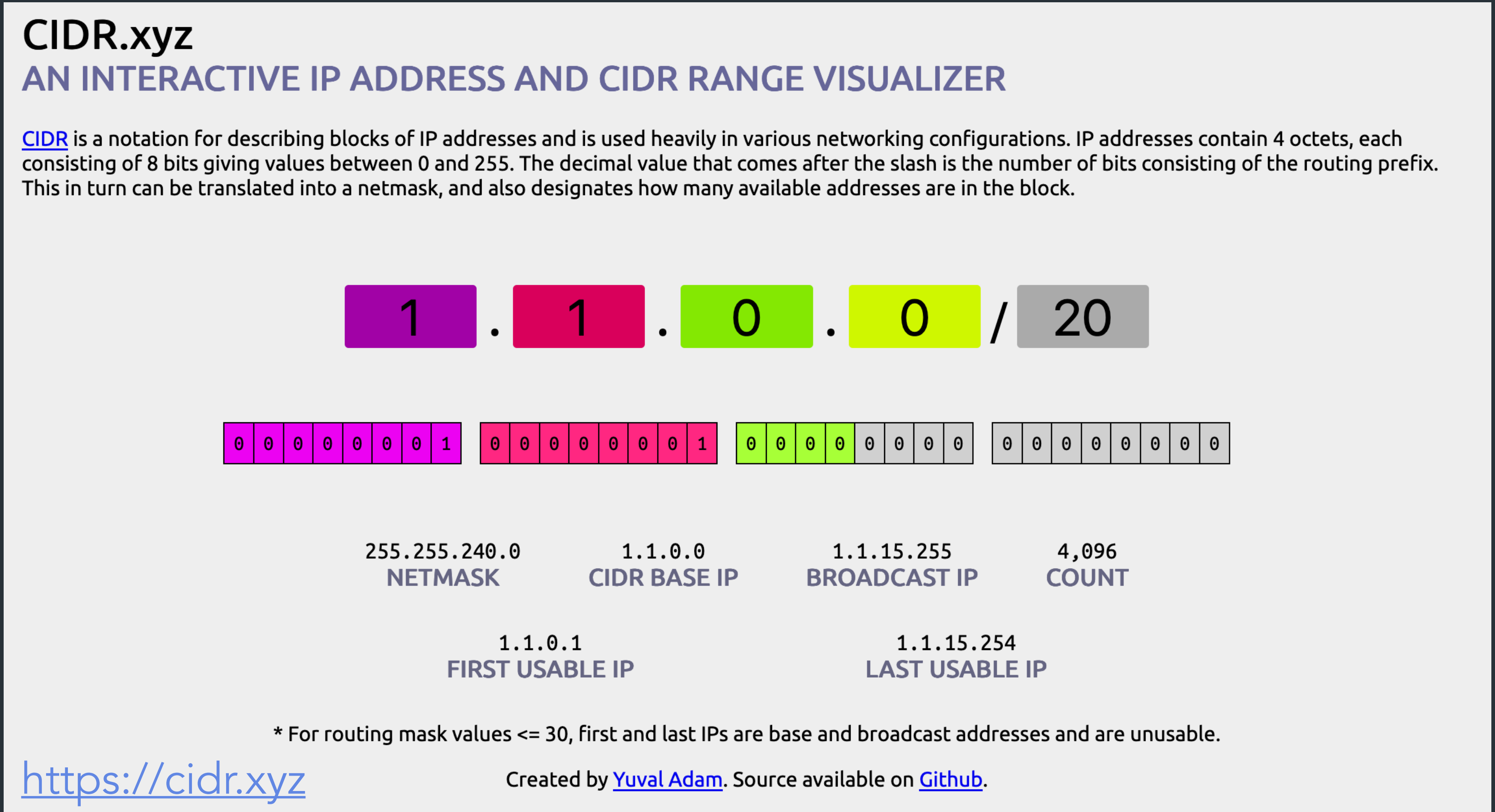
🟩 = UNALLOCATED BLOCK

https://xkcd.com/195/

# Example

How many addresses are in the network 1.1.0.0/20?

Is 1.1.16.1 in this prefix?



# CIDR.xyz

## AN INTERACTIVE IP ADDRESS AND CIDR RANGE VISUALIZER

CIDR is a notation for describing blocks of IP addresses and is used heavily in various networking configurations. IP addresses contain 4 octets, each consisting of 8 bits giving values between 0 and 255. The decimal value that comes after the slash is the number of bits consisting of the routing prefix. This in turn can be translated into a netmask, and also designates how many available addresses are in the block.

1 . 1 . 0 . 0 / 20

0 0 0 0 0 0 0 1   0 0 0 0 0 0 0 1   0 0 0 0 0 0 0 0   0 0 0 0 0 0 0 0

255.255.240.0
NETMASK

1.1.0.0
CIDR BASE IP

1.1.15.255
BROADCAST IP

4,096
COUNT

1.1.0.1
FIRST USABLE IP

1.1.15.254
LAST USABLE IP

* For routing mask values <= 30, first and last IPs are base and broadcast addresses and are unusable.

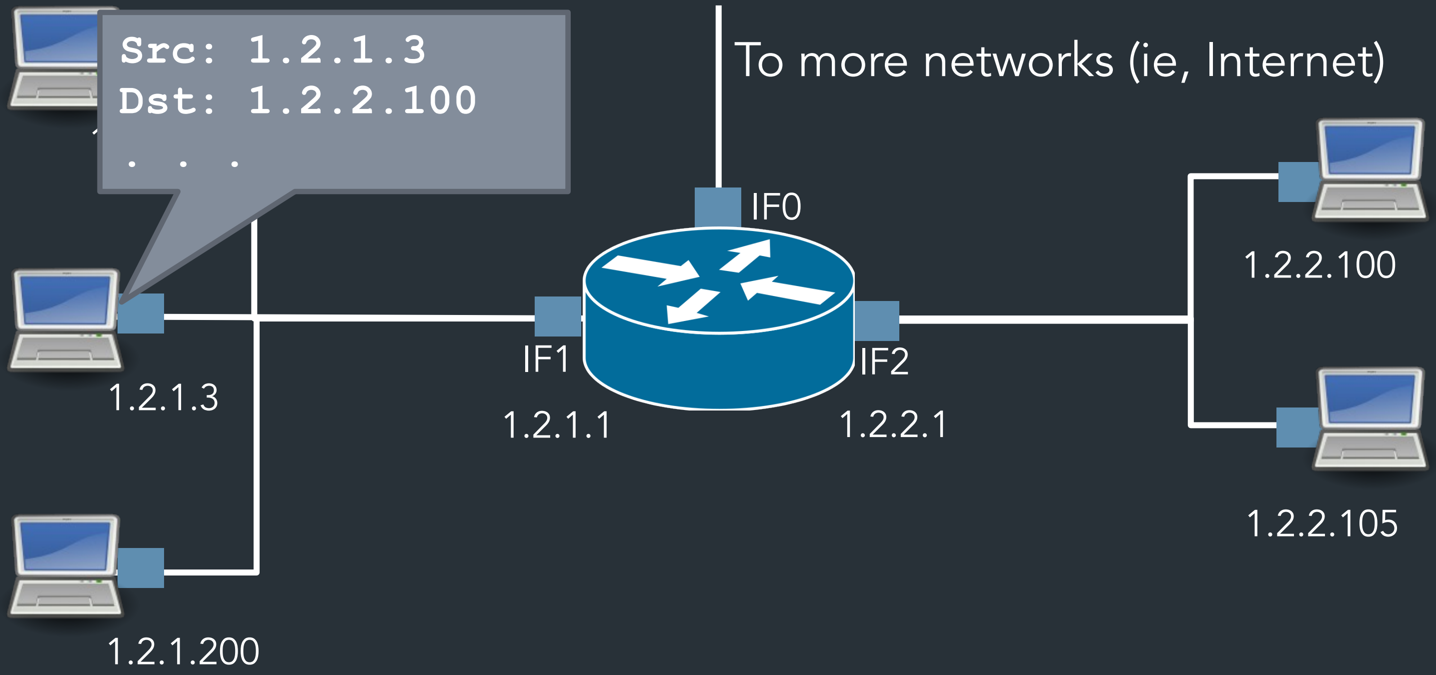
https://cidr.xyz

Created by Yuval Adam. Source available on Github.

Tools exist, use them!

*How do we move packets between networks?*

# Forwarding IP packets



```
Src: 1.2.1.3
Dst: 1.2.2.100
. . .
```

# IP forwarding

Decide where to send packets based on forwarding table

| Prefix | Interface/Next hop |
|---|---|
| | |
| | |
| | |

# IP forwarding

Decide where to send packets based on forwarding table

| Prefix | Interface/Next hop |
|---|---|
| | |
| | |
| | |

Key Type: An IP prefix (1.2.1.0/24)

Value type: Multiple forms

- Interface (IF0): “This is my neighbor (on local net), link-layer can figure it out”

- Next hop IP (eg. 1.1.1.1): send packet to this IP instead

# IP forwarding

Decide where to send packets based on forwarding table

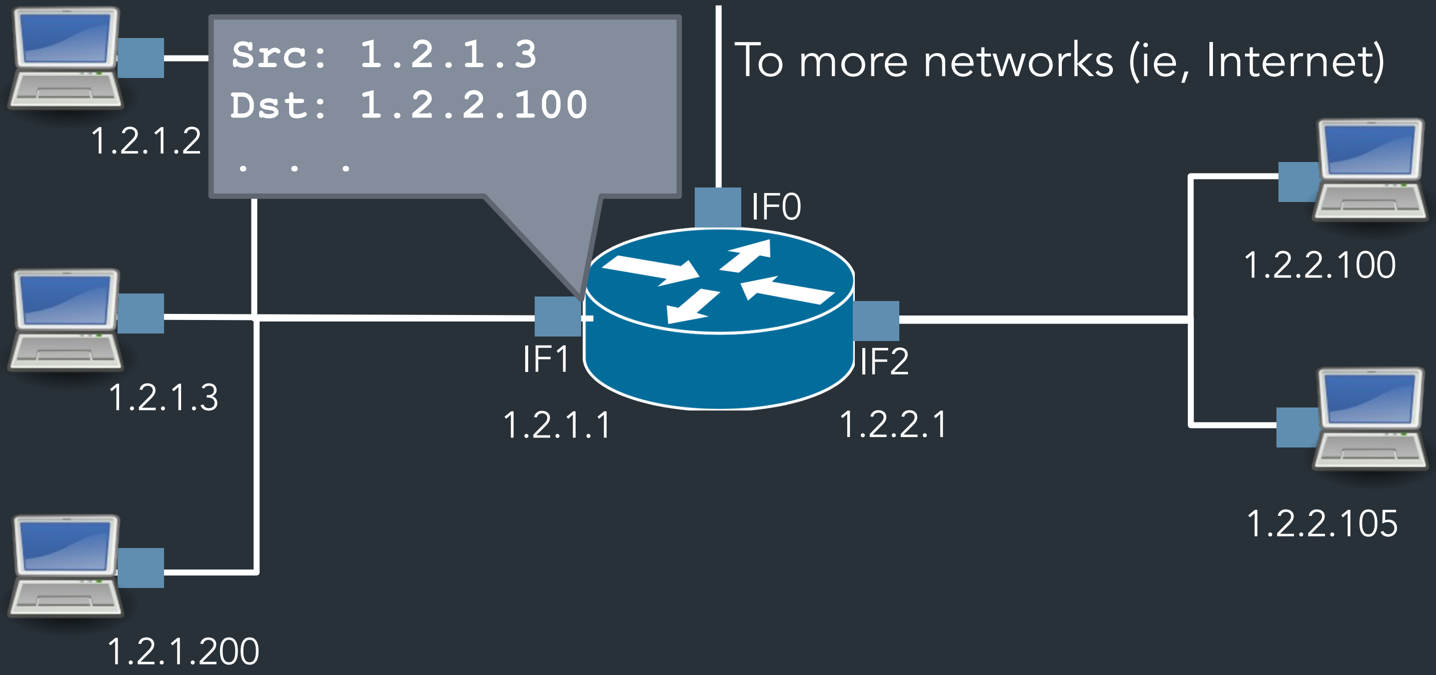
| Prefix | Interface/Next hop |
|---|---|
| | |
| | |
| | |

Key Type: An IP prefix (1.2.1.0/24)

Value type: Multiple forms

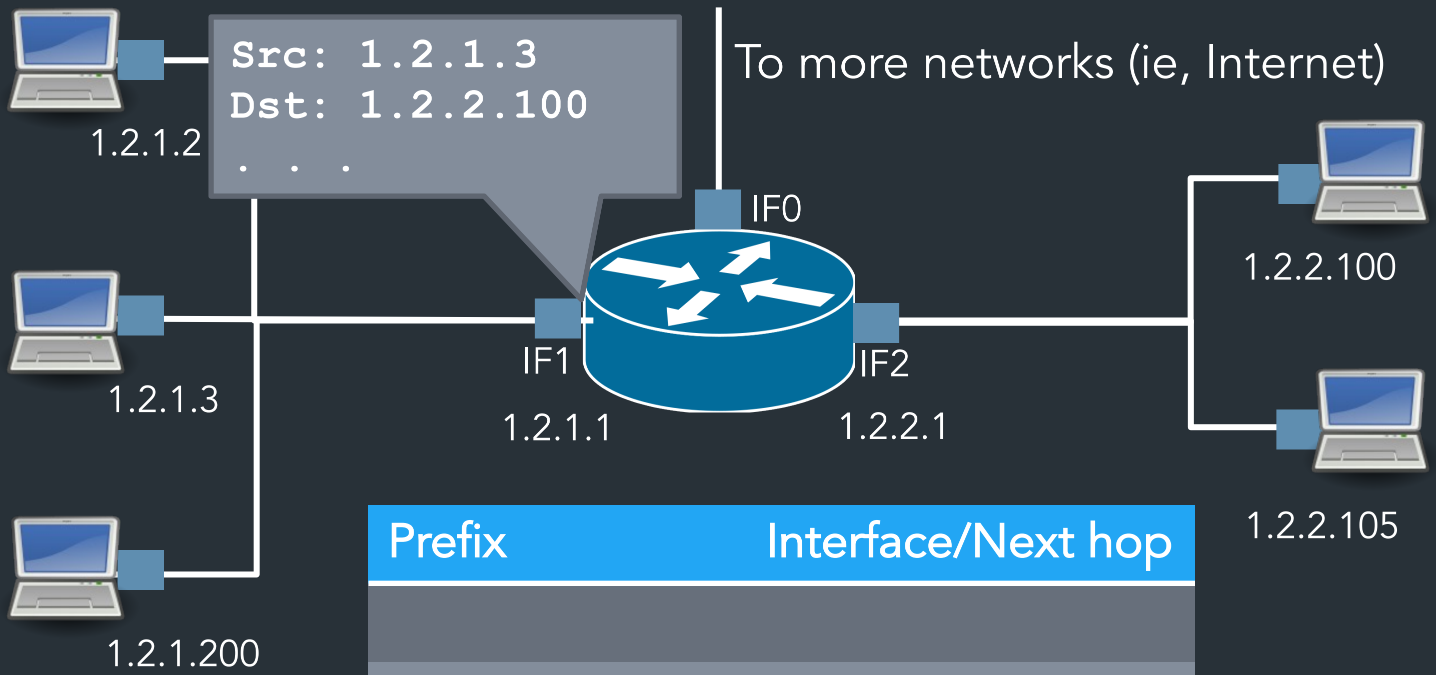
- Interface (IF0): "This is my neighbor (on local net), link-layer can figure it out"
  => "Local delivery"
- Next hop IP (eg. 1.1.1.1): send packet to this IP instead
  => Need to search for next hop in table!

# Forwarding IP packets



```
Src: 1.2.1.3
Dst: 1.2.2.100
. . .
```

# Forwarding IP packets

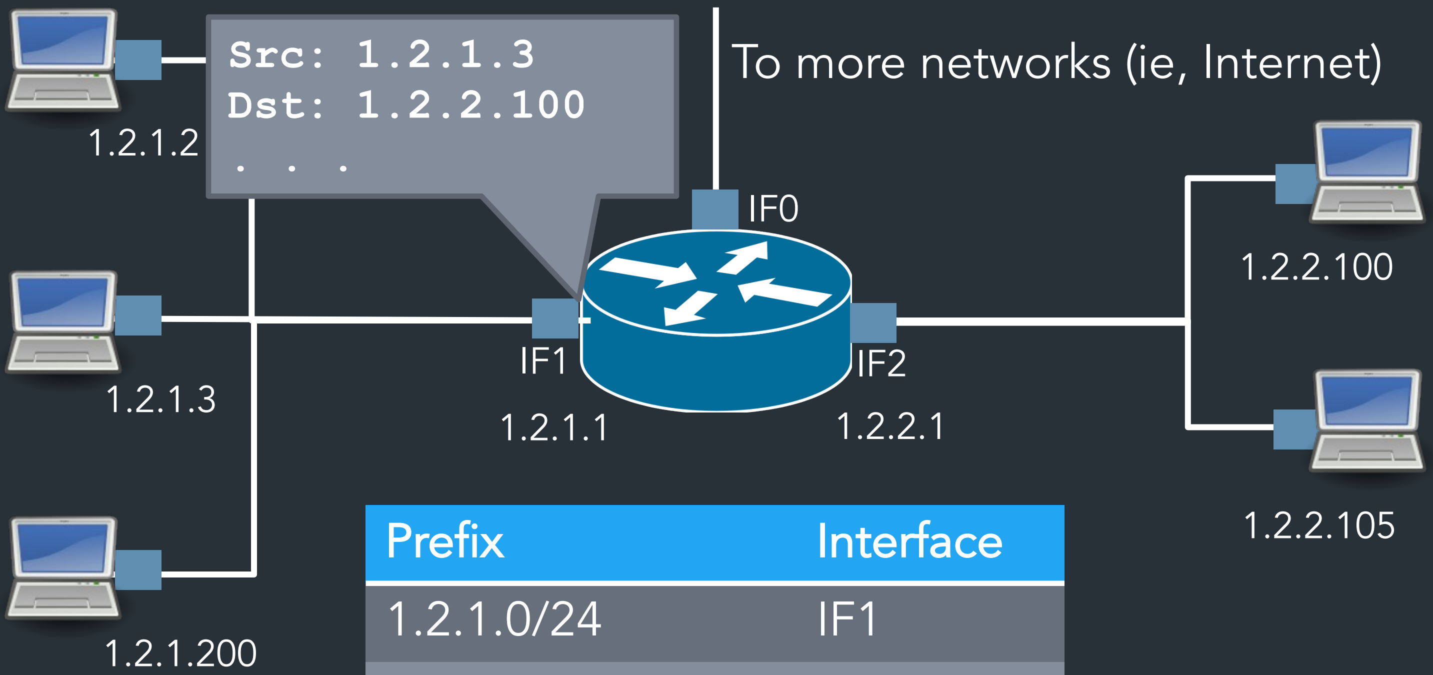


```
Src: 1.2.1.3
Dst: 1.2.2.100
. . .
```

To more networks (ie, Internet)

| Prefix | Interface/Next hop |
|---|---|
| | |
| | |
| | |

# Forwarding IP packets



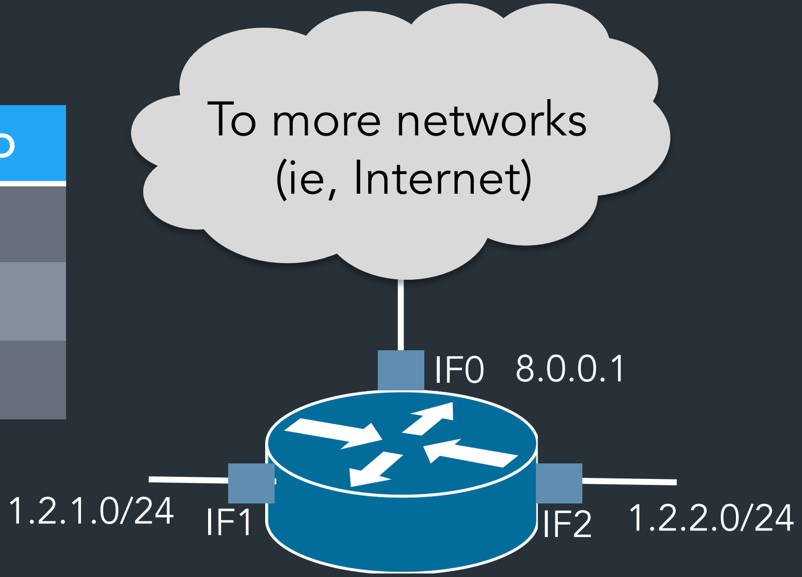
```
Src: 1.2.1.3
Dst: 1.2.2.100
. . .
```

| Prefix | Interface |
|---|---|
| 1.2.1.0/24 | IF1 |
| 1.2.2.0/24 | IF2 |
| … | … |

# What about the rest?

How to reach networks that aren't directly connected?

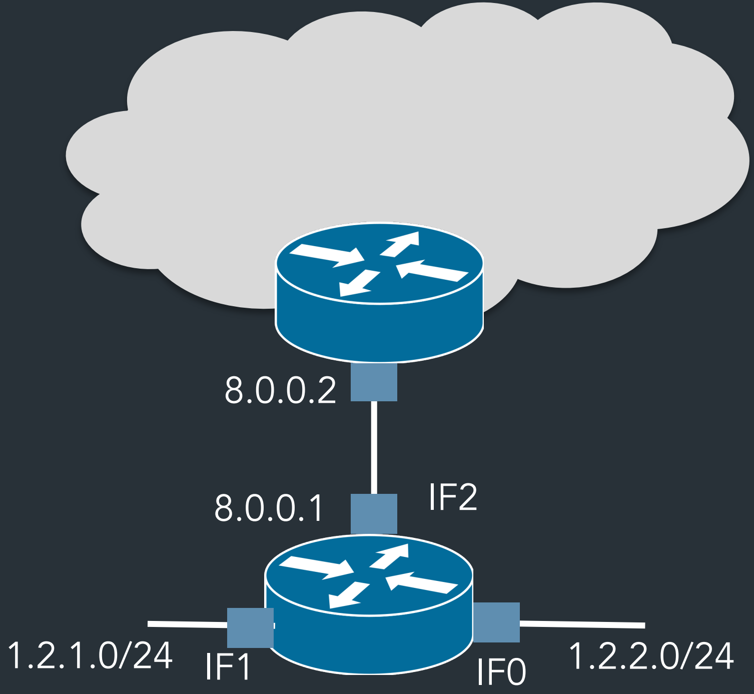
| Prefix | Interface/Next hop |
|---|---|
| 1.2.1.0/24 | IF1 |
| 1.2.2.0/24 | IF2 |
| <everything else> | |



Default gateway: where to send to reach anything not in the table

Gateway: device at the "edge" of a network (eg. Brown <-> Internet)

| Prefix | IF/Next hop |
|---|---|
| 1.2.1.0/24 | IF1 |
| 1.2.2.0/24 | IF2 |
| 8.0.0.0/30 | IF0 |
| 0.0.0.0/0 | 8.0.0.2 |



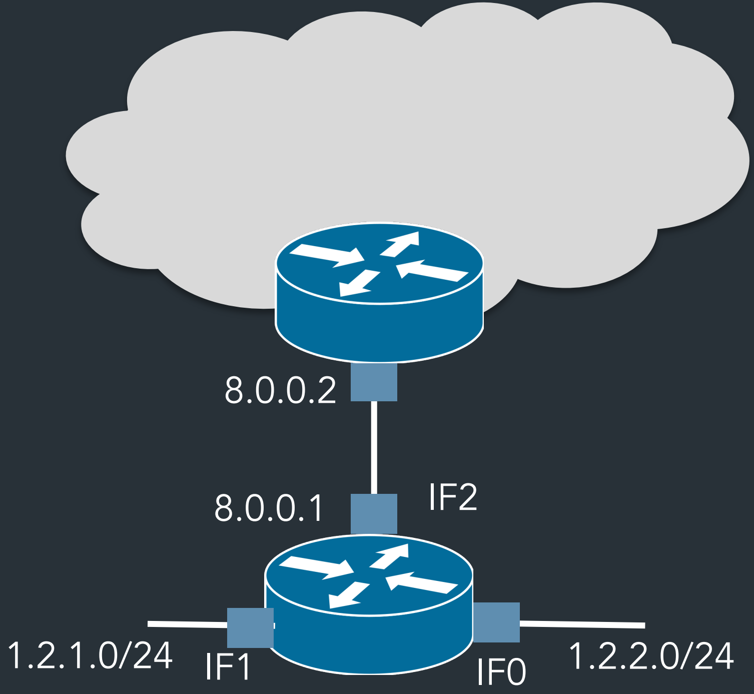
IF2

Default gateway: where to send to reach anything not in the table

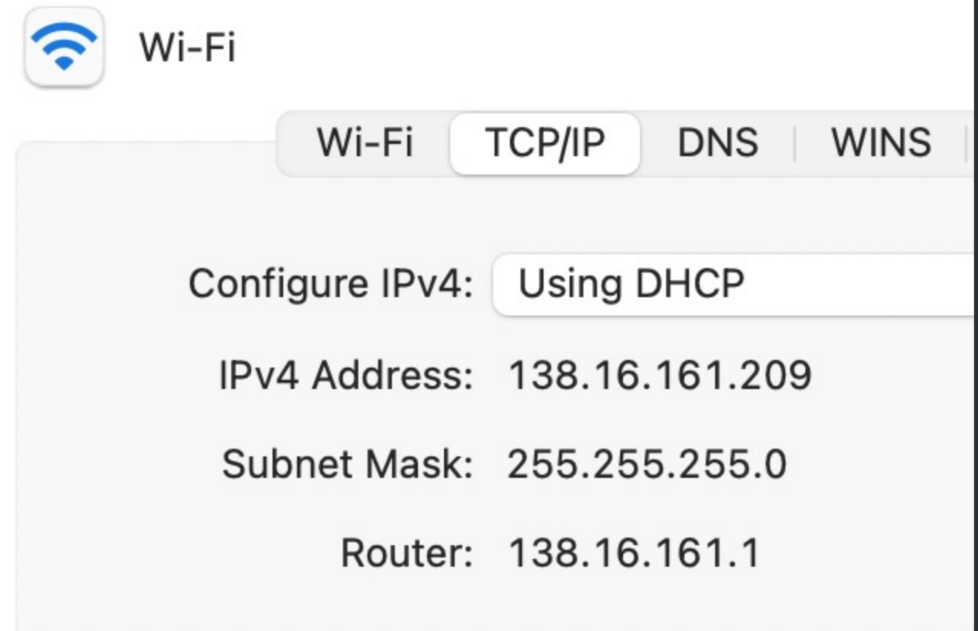
=> Use "Next hop IP" in table

Gateway: device at the "edge" of  a network (eg. Brown <-> Internet)

| Prefix | IF/Next hop |
|---|---|
| 1.2.1.0/24 | IF1 |
| 1.2.2.0/24 | IF2 |
| 8.0.0.0/30 | IF0 |
| 0.0.0.0/0 | 8.0.0.2 |



IF2



Configure IPv4: Using DHCP

IPv4 Address: 138.16.161.209

Subnet Mask: 255.255.255.0

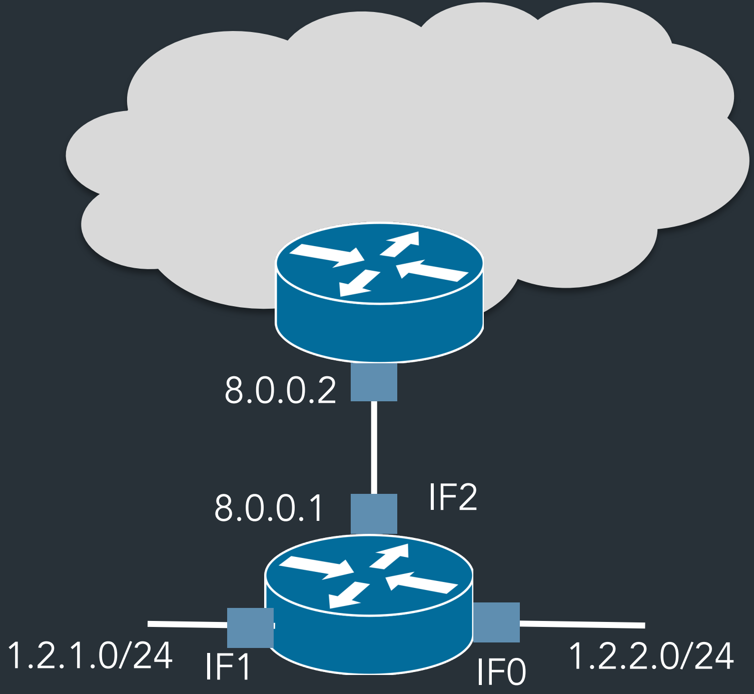
Router: 138.16.161.1

Default gateway: where to send to reach anything not in the table

=> Use "Next hop IP" in table

Gateway: device at the "edge" of a network (eg. Brown <-> Internet)

| Prefix | IF/Next hop |
|---|---|
| 1.2.1.0/24 | IF1 |
| 1.2.2.0/24 | IF2 |
| 8.0.0.0/30 | IF0 |
| 0.0.0.0/0 | 8.0.0.2 |



⇒ 0.0.0.0 matches on everything! Problem?
Can have multiple matches in table => use the most specific (longest) prefix
(more on this later)

# Does it scale?

| Prefix | IF/Next hop |
|---|---|
| 1.2.1.0/24 | IF1 |
| 1.2.2.0/24 | IF2 |
| 8.0.0.0/30 | IF0 |
| 0.0.0.0/0 | 8.0.0.2 |

# Does it scale?

| Prefix | IF/Next hop |
|---|---|
| 1.2.1.0/24 | IF1 |
| 1.2.2.0/24 | IF2 |
| 8.0.0.0/30 | IF0 |
| 0.0.0.0/0 | 8.0.0.2 |

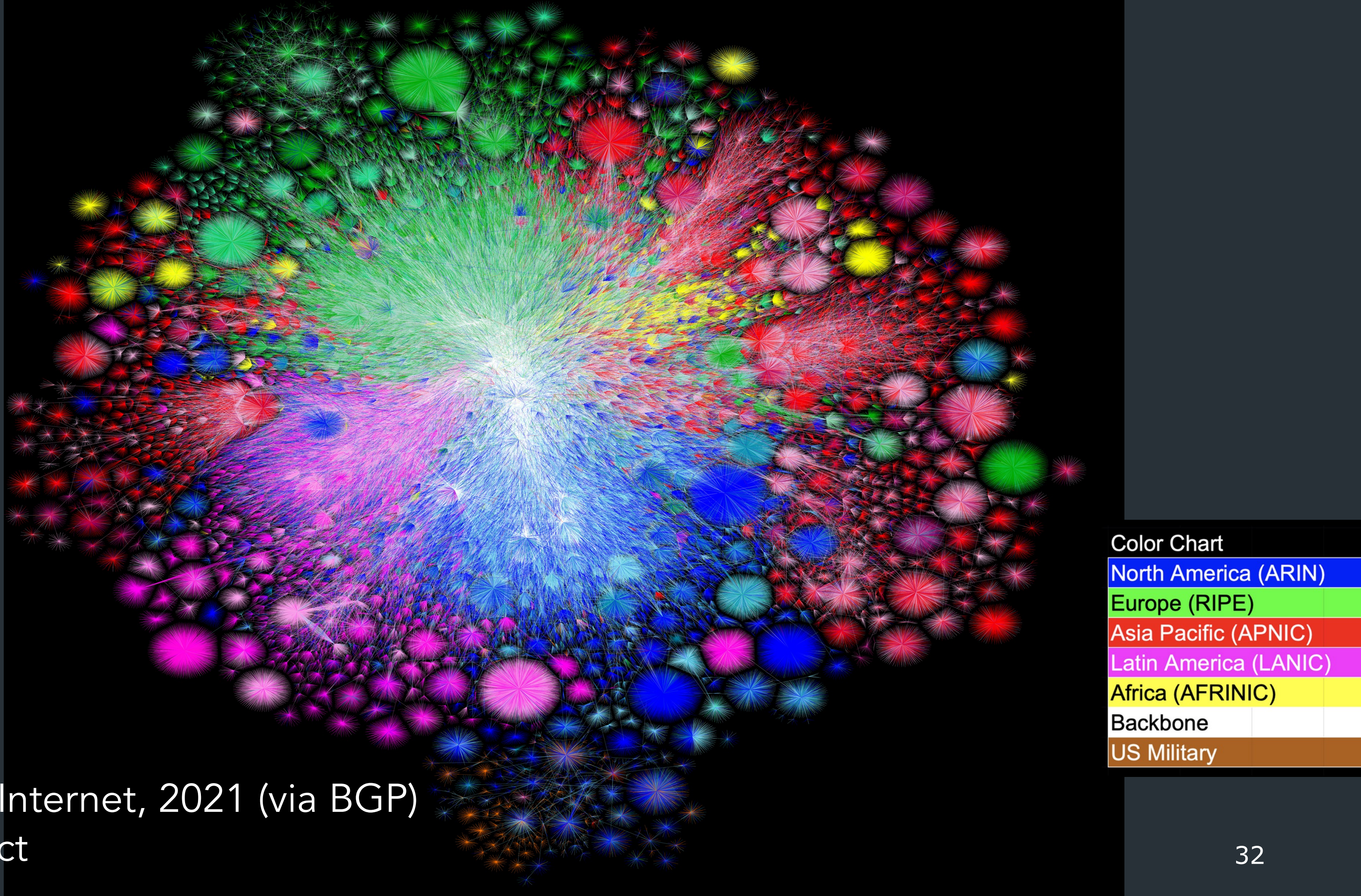
Yes! (At least enough to make the Internet as we know it…)

=> Forward packets based on IP prefixes

=> Don’t need to keep track of every single host

=> Routers at the “edges” of the network don’t need to know about every route

=> Larger, highly-connected routers (“core routers”) do need very large tables, specialized hardware, optimization tricks…

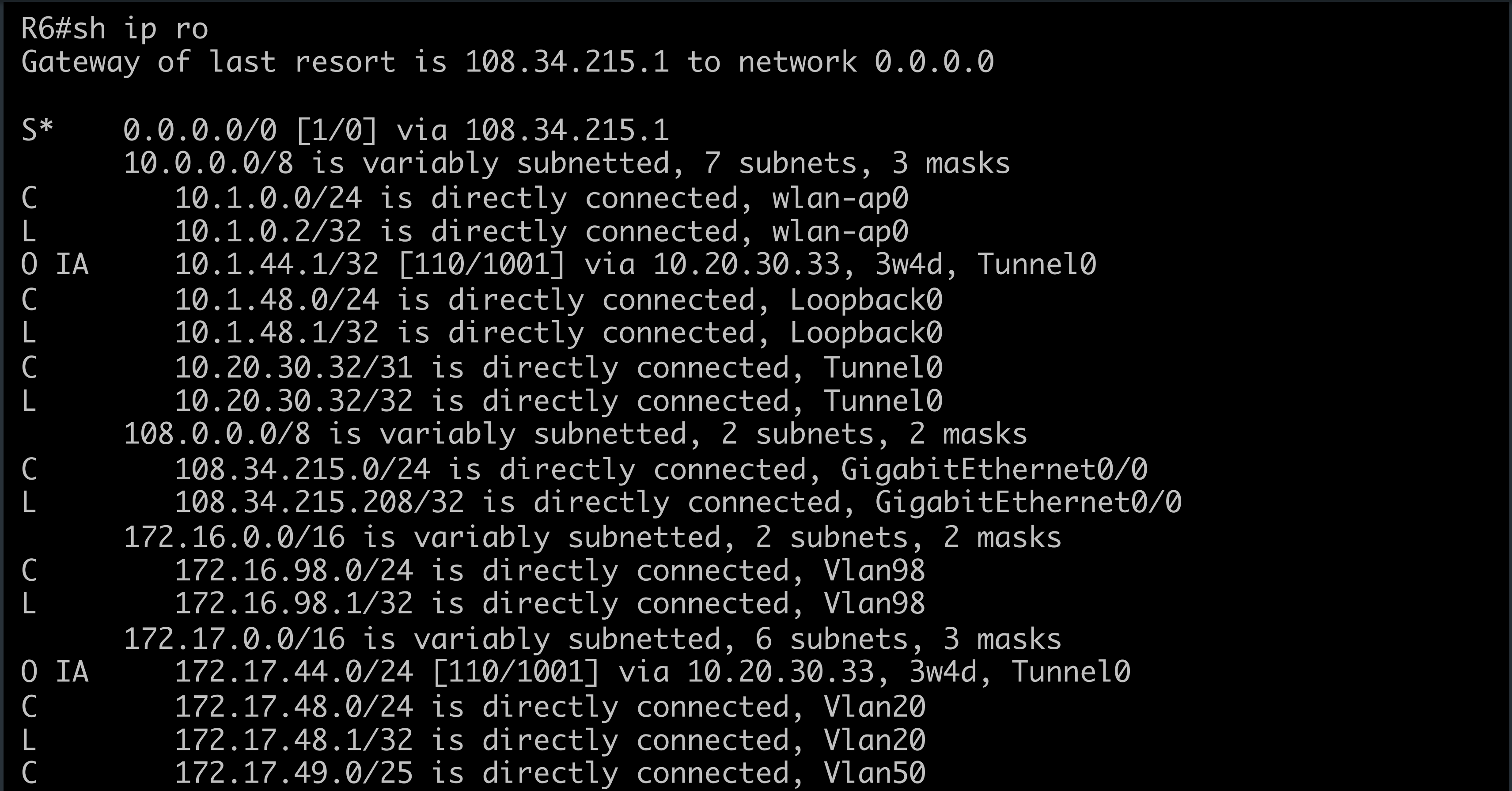


Map of the Internet, 2021 (via BGP)
OPTE project

# A forwarding table (my laptop)

```
deemer@ceres ~ % ip route
default via 10.3.128.1 dev wlp2s0
10.3.128.0/18 dev wlp2s0 proto dhcp scope link src 10.3.135.44 metric 3003
172.18.0.0/16 dev docker0 proto kernel scope link src 172.18.0.1
192.168.1.0/24 dev enp0s31f6 proto kernel scope link src 192.168.1.1
```

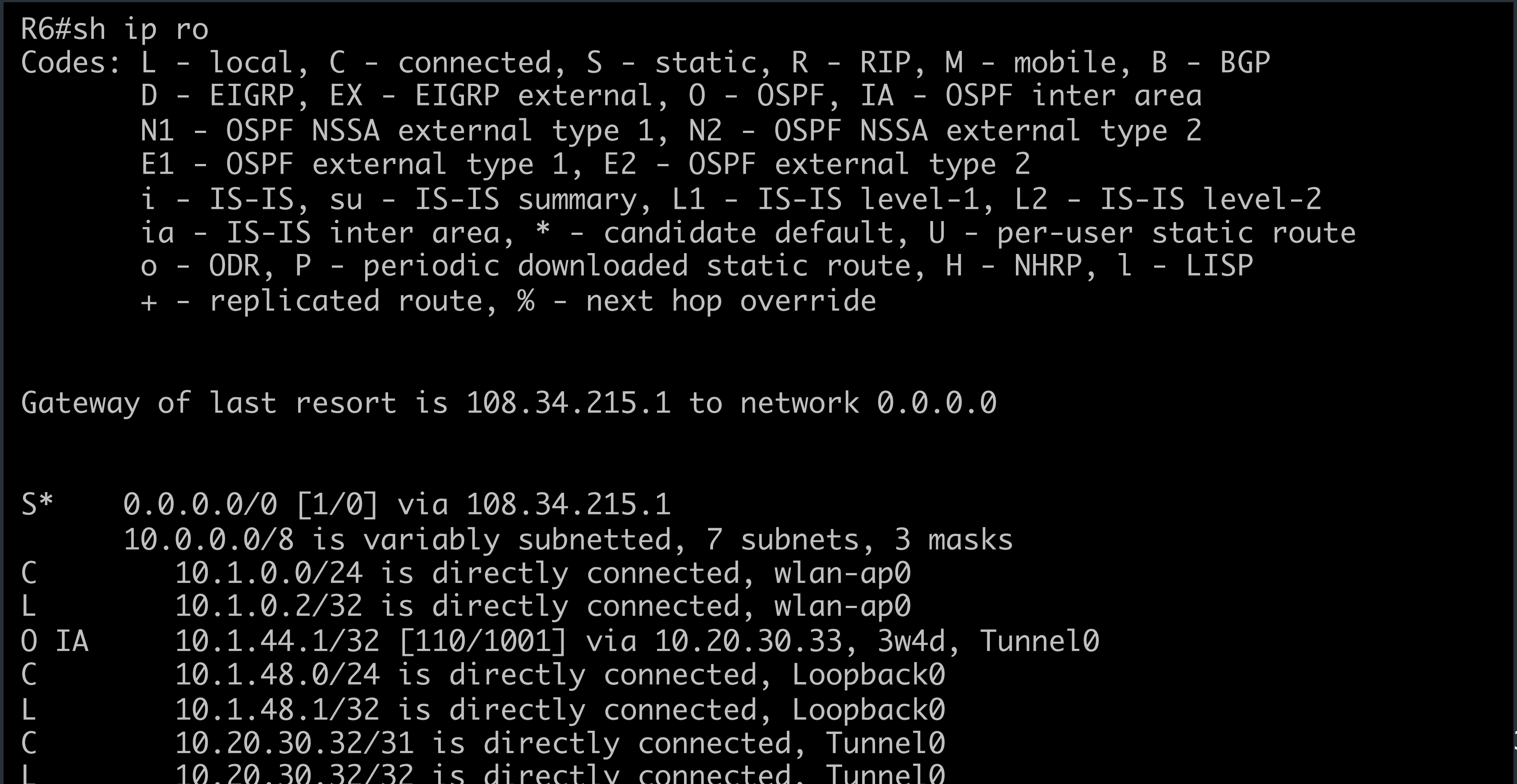
# A routing table



```
R6#sh ip ro
Gateway of last resort is 108.34.215.1 to network 0.0.0.0

S*    0.0.0.0/0 [1/0] via 108.34.215.1
      10.0.0.0/8 is variably subnetted, 7 subnets, 3 masks
C        10.1.0.0/24 is directly connected, wlan-ap0
L        10.1.0.2/32 is directly connected, wlan-ap0
O IA     10.1.44.1/32 [110/1001] via 10.20.30.33, 3w4d, Tunnel0
C        10.1.48.0/24 is directly connected, Loopback0
L        10.1.48.1/32 is directly connected, Loopback0
C        10.20.30.32/31 is directly connected, Tunnel0
L        10.20.30.32/32 is directly connected, Tunnel0
      108.0.0.0/8 is variably subnetted, 2 subnets, 2 masks
C        108.34.215.0/24 is directly connected, GigabitEthernet0/0
L        108.34.215.208/32 is directly connected, GigabitEthernet0/0
      172.16.0.0/16 is variably subnetted, 2 subnets, 2 masks
C        172.16.98.0/24 is directly connected, Vlan98
L        172.16.98.1/32 is directly connected, Vlan98
      172.17.0.0/16 is variably subnetted, 6 subnets, 3 masks
O IA     172.17.44.0/24 [110/1001] via 10.20.30.33, 3w4d, Tunnel0
C        172.17.48.0/24 is directly connected, Vlan20
L        172.17.48.1/32 is directly connected, Vlan20
C        172.17.49.0/25 is directly connected, Vlan50
```

# A routing table

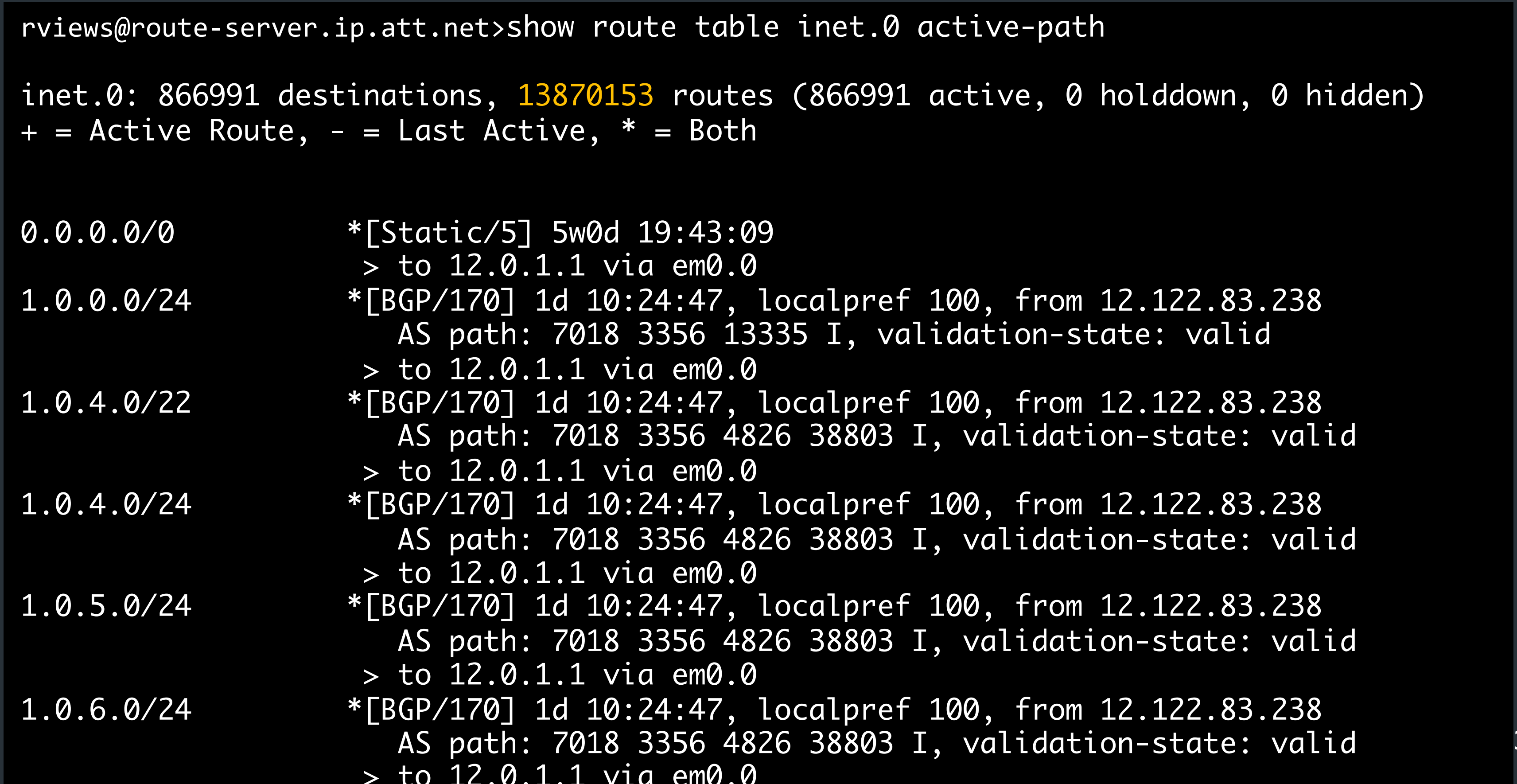


```
R6#sh ip ro
Codes: L - local, C - connected, S - static, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2
       i - IS-IS, su - IS-IS summary, L1 - IS-IS level-1, L2 - IS-IS level-2
       ia - IS-IS inter area, * - candidate default, U - per-user static route
       o - ODR, P - periodic downloaded static route, H - NHRP, l - LISP
       + - replicated route, % - next hop override


Gateway of last resort is 108.34.215.1 to network 0.0.0.0


S*    0.0.0.0/0 [1/0] via 108.34.215.1
      10.0.0.0/8 is variably subnetted, 7 subnets, 3 masks
C        10.1.0.0/24 is directly connected, wlan-ap0
L        10.1.0.2/32 is directly connected, wlan-ap0
O IA     10.1.44.1/32 [110/1001] via 10.20.30.33, 3w4d, Tunnel0
C        10.1.48.0/24 is directly connected, Loopback0
L        10.1.48.1/32 is directly connected, Loopback0
C        10.20.30.32/31 is directly connected, Tunnel0
L        10.20.30.32/32 is directly connected, Tunnel0
```

# A large table



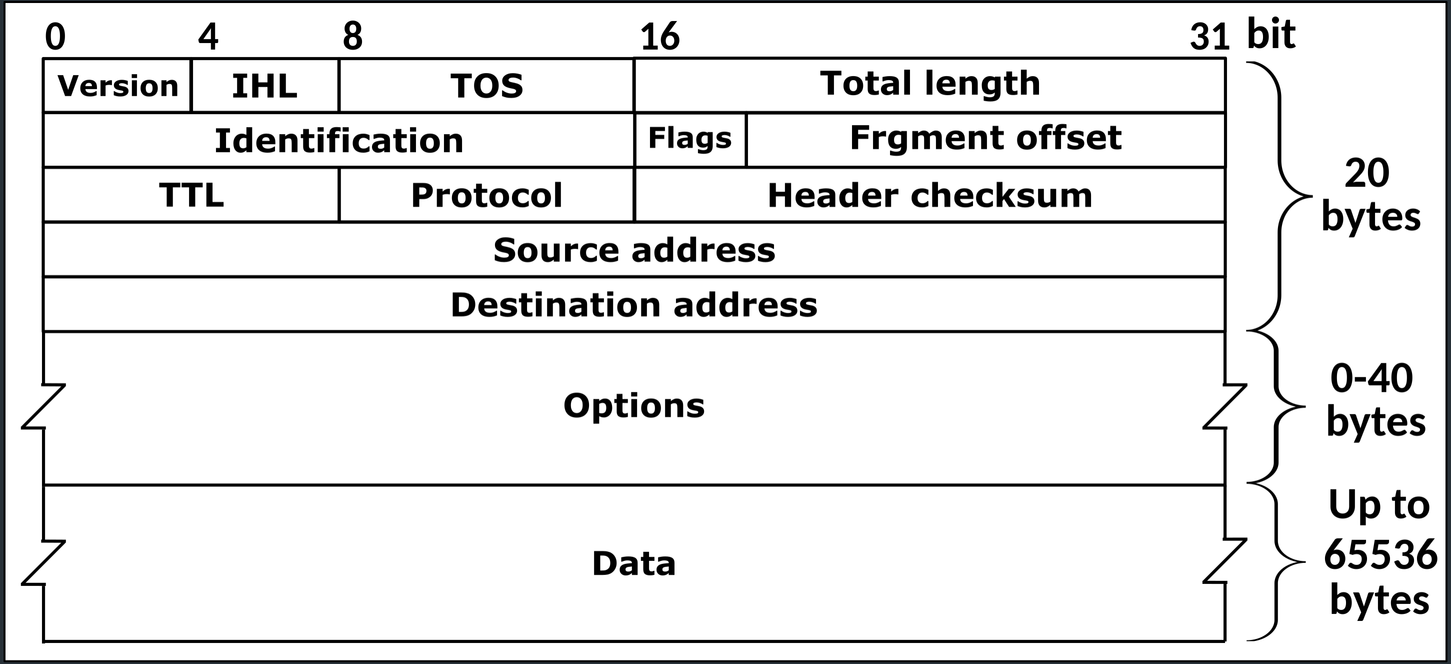
```
rviews@route-server.ip.att.net>show route table inet.0 active-path

inet.0: 866991 destinations, 13870153 routes (866991 active, 0 holddown, 0 hidden)
+ = Active Route, - = Last Active, * = Both


0.0.0.0/0          *[Static/5] 5w0d 19:43:09
                    > to 12.0.1.1 via em0.0
1.0.0.0/24         *[BGP/170] 1d 10:24:47, localpref 100, from 12.122.83.238
                      AS path: 7018 3356 13335 I, validation-state: valid
                    > to 12.0.1.1 via em0.0
1.0.4.0/22         *[BGP/170] 1d 10:24:47, localpref 100, from 12.122.83.238
                      AS path: 7018 3356 4826 38803 I, validation-state: valid
                    > to 12.0.1.1 via em0.0
1.0.4.0/24         *[BGP/170] 1d 10:24:47, localpref 100, from 12.122.83.238
                      AS path: 7018 3356 4826 38803 I, validation-state: valid
                    > to 12.0.1.1 via em0.0
1.0.5.0/24         *[BGP/170] 1d 10:24:47, localpref 100, from 12.122.83.238
                      AS path: 7018 3356 4826 38803 I, validation-state: valid
                    > to 12.0.1.1 via em0.0
1.0.6.0/24         *[BGP/170] 1d 10:24:47, localpref 100, from 12.122.83.238
                      AS path: 7018 3356 4826 38803 I, validation-state: valid
                    > to 12.0.1.1 via em0.0
```

# How does forwarding actually work?

# The IPv4 Header



Defined by RFC 791
RFC (Request for Comment): defines network standard

# Most Important fields

- Version: 4 for IPv4 packets, 6 for IPv6
- Source address: where the packet came from
- Destination address: where the packet is going

(continued...)

# More important fields

- TTL (time-to-live): decremented each hop
  - Can prevent forwarding loops (and do other stuff…)

- Checksum: computed over header (very weak!)

- Protocol identifier: describes what's in the packet
  - 6: TCP, 17: UDP, 1: ICMP, …
  - Defines the type of the payload

# Less important fields

- Header length:  in 32-bit units
  - >5 implies use of IP options
  - Almost all routers ignore IP options
- Fragmentation
  - Network can fragment a packet if next link requires a small frame
  - Most routers don’t fragment (or reassemble fragments)
- We won’t talk about…
  - Type of Service (TOS):  basic traffic classification
  - Identifier:  might have special meaning on some networks

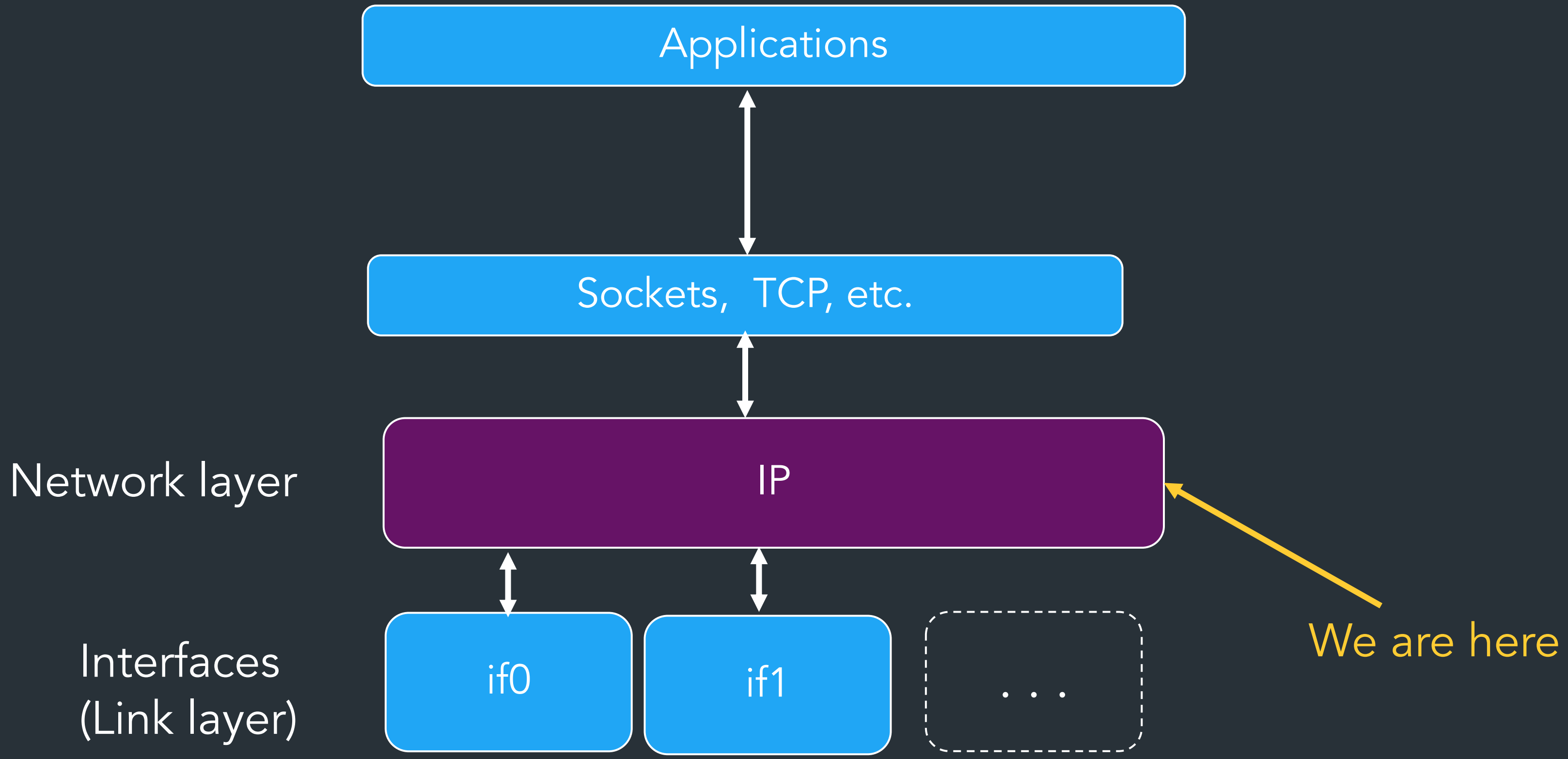
# Forwarding steps

What does a *device* do when it receives a packet?

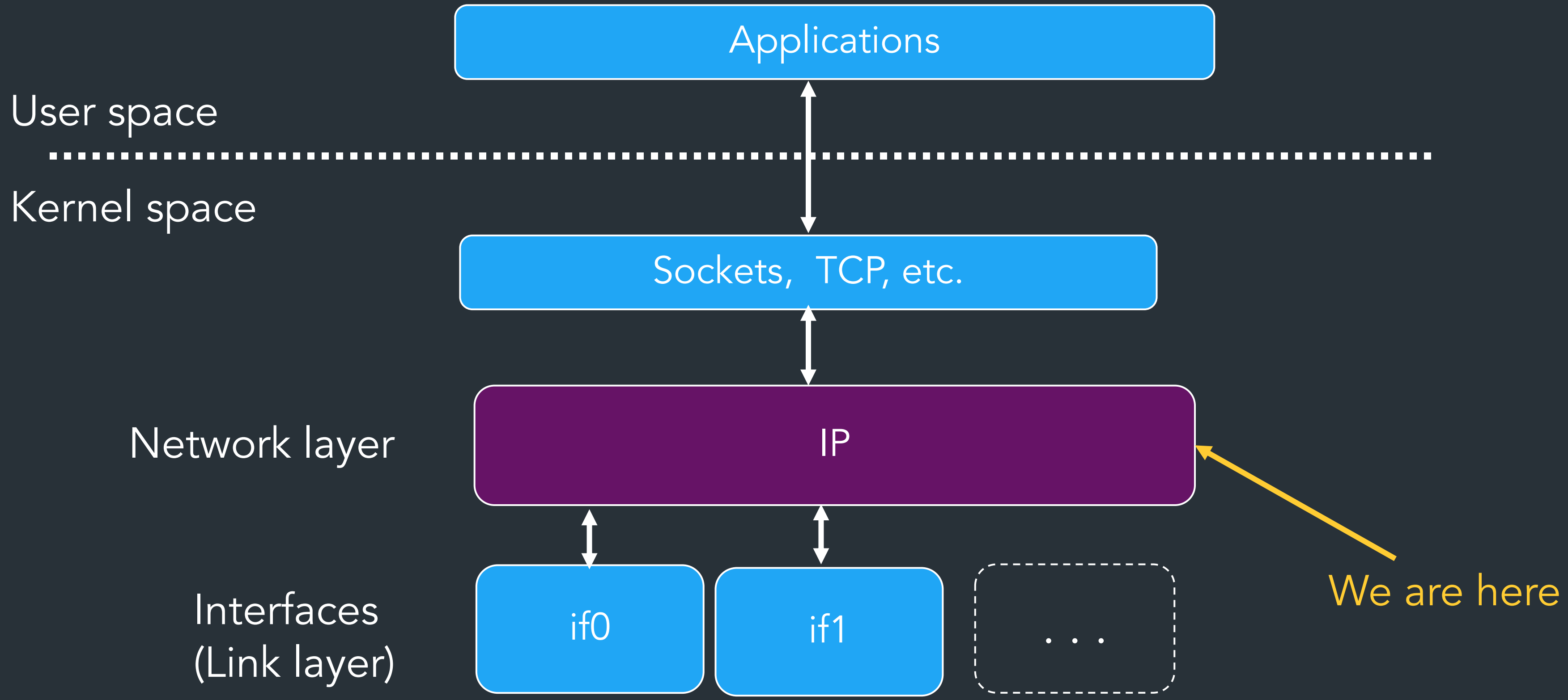
Device:  host, router, …

# A ”networking stack”

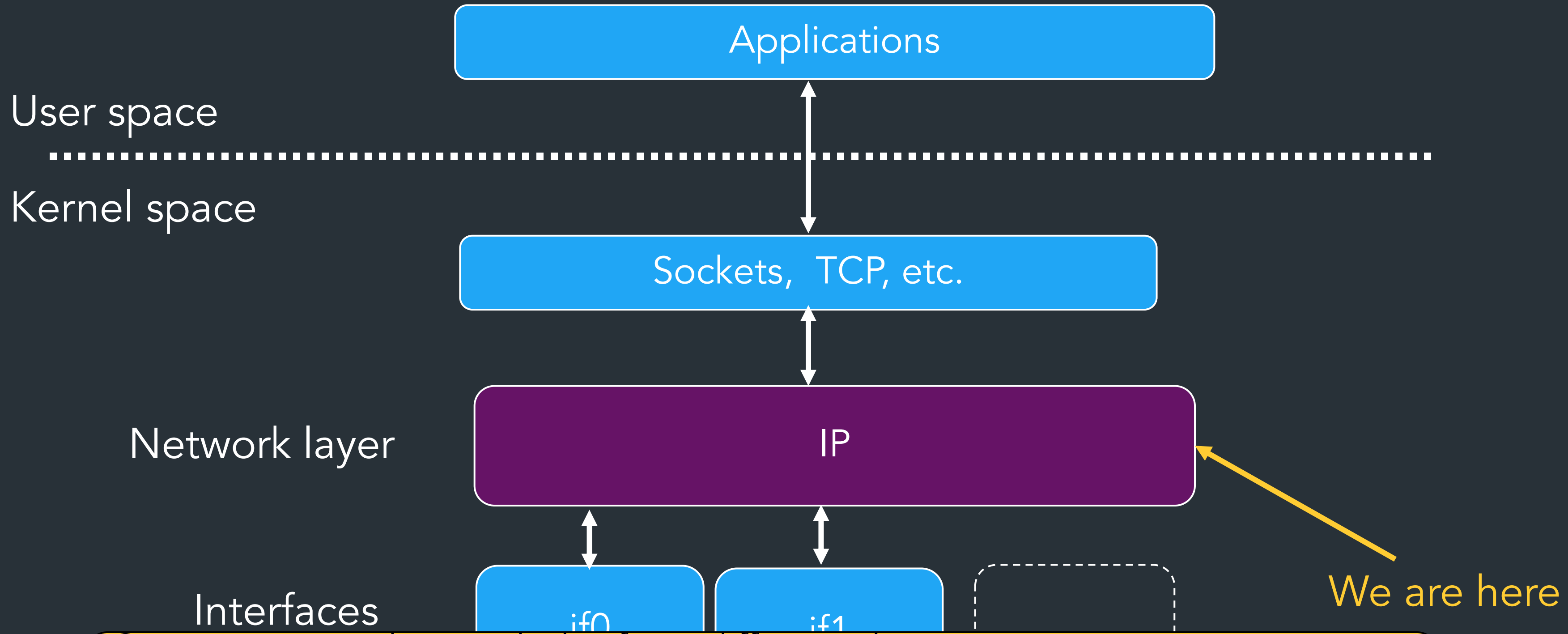
# A "networking stack"



# A "networking stack"



# A "networking stack"



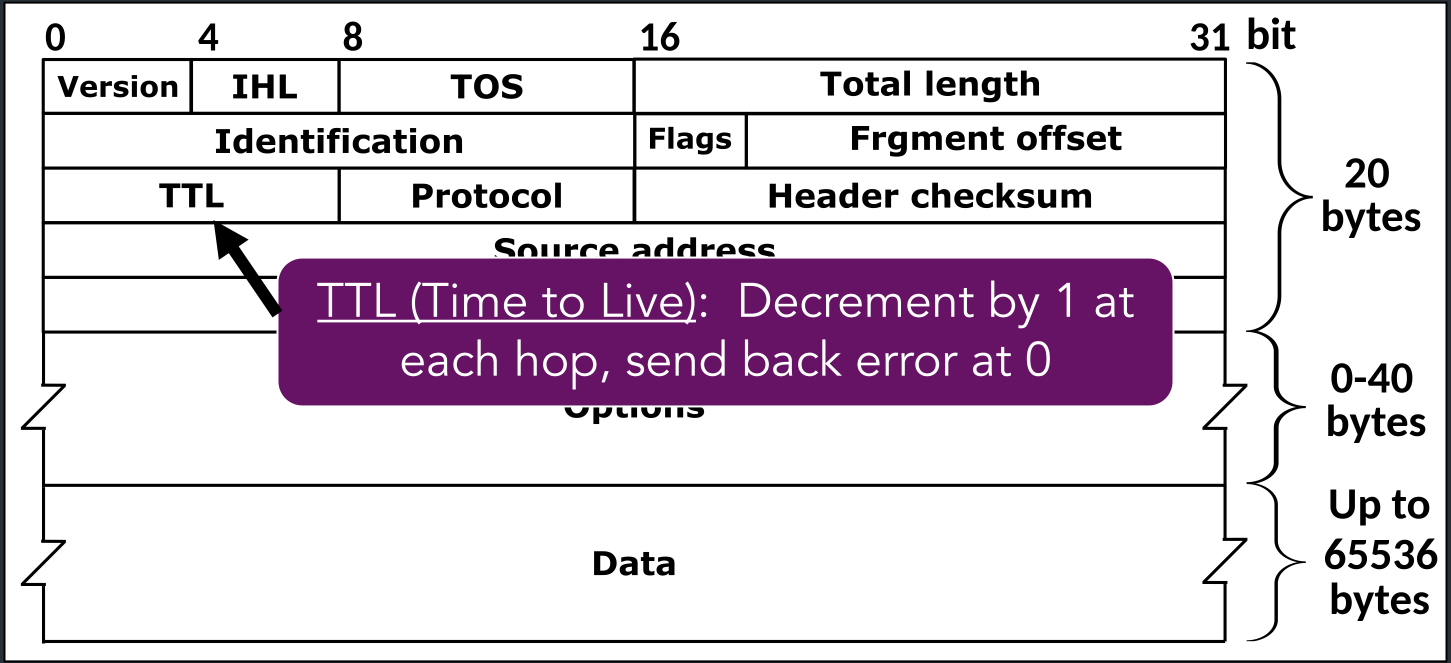
On a router: packet might be for a different destination
=> send out another interface
If it's for one for an IP assigned to this device, send to higher layer

# Forwarding mechanics

When an IP packet arrives at a host/router:

- Is it valid? Verify checksum over *header*
- **Is it for me?** If dest IP == your address, send to OS
- If not, where should it go?
  - Consult forwarding table => find next hop
  - Decrement TTL
  - Send packet to next hop

# How to avoid loops?



TTL (Time to Live): Decrement by 1 at each hop, send back error at 0

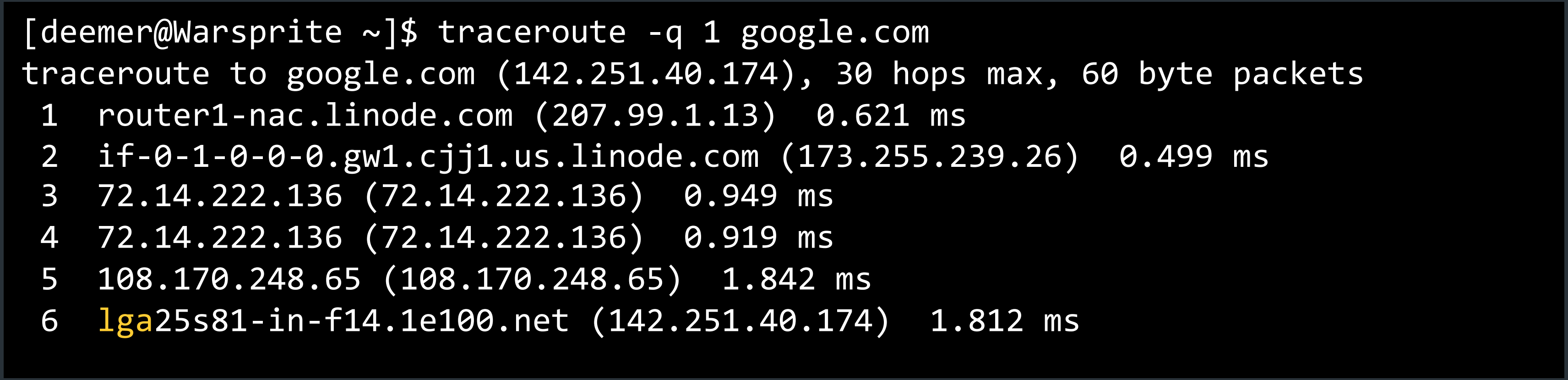
`traceroute`: tool to send packets with increasing TTLs
=> can learn about network paths!

# Traceroute

- When TTL reaches 0, router may send back an error
  - ICMP TTL exceeded

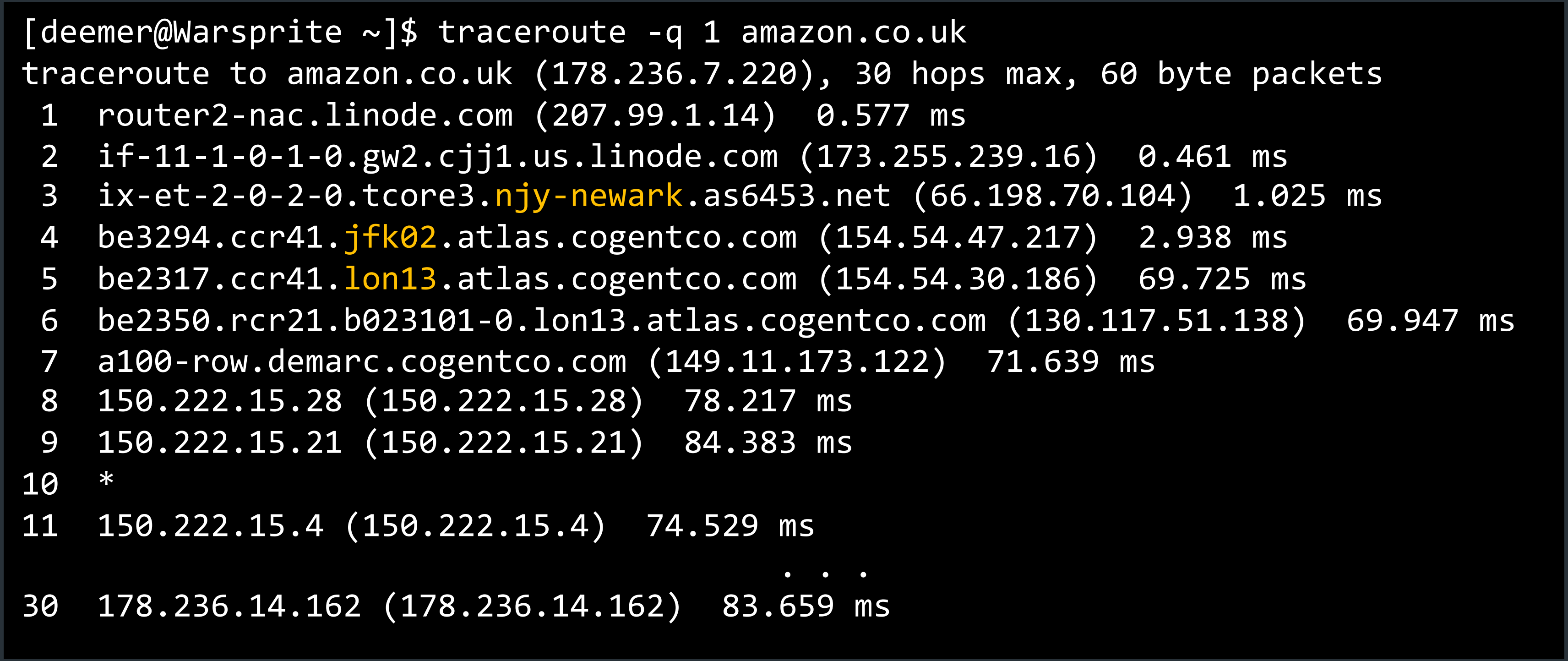
- If it does, we can identify a path used by a packet!

# Traceroute example



```
[deemer@Warsprite ~]$ traceroute -q 1 google.com
traceroute to google.com (142.251.40.174), 30 hops max, 60 byte packets
 1  router1-nac.linode.com (207.99.1.13)  0.621 ms
 2  if-0-1-0-0-0.gw1.cjj1.us.linode.com (173.255.239.26)  0.499 ms
 3  72.14.222.136 (72.14.222.136)  0.949 ms
 4  72.14.222.136 (72.14.222.136)  0.919 ms
 5  108.170.248.65 (108.170.248.65)  1.842 ms
 6  lga25s81-in-f14.1e100.net (142.251.40.174)  1.812 ms
```

# Traceroute example



```
[deemer@Warsprite ~]$ traceroute -q 1 amazon.co.uk
traceroute to amazon.co.uk (178.236.7.220), 30 hops max, 60 byte packets
 1  router2-nac.linode.com (207.99.1.14)  0.577 ms
 2  if-11-1-0-1-0.gw2.cjj1.us.linode.com (173.255.239.16)  0.461 ms
 3  ix-et-2-0-2-0.tcore3.njy-newark.as6453.net (66.198.70.104)  1.025 ms
 4  be3294.ccr41.jfk02.atlas.cogentco.com (154.54.47.217)  2.938 ms
 5  be2317.ccr41.lon13.atlas.cogentco.com (154.54.30.186)  69.725 ms
 6  be2350.rcr21.b023101-0.lon13.atlas.cogentco.com (130.117.51.138)  69.947 ms
 7  a100-row.demarc.cogentco.com (149.11.173.122)  71.639 ms
 8  150.222.15.28 (150.222.15.28)  78.217 ms
 9  150.222.15.21 (150.222.15.21)  84.383 ms
10  *
11  150.222.15.4 (150.222.15.4)  74.529 ms
                                        . . .
30  178.236.14.162 (178.236.14.162)  83.659 ms
```

# Putting it all together

# Demo:  IP project

# Coming up…

- ARP:  Mapping IPs to MAC addresses
- How are addresses assigned?
- NAT:  When it gets complicated
- Routing algorithms:  how to build forwarding tables

Fill out the group preference survey for the IP project (announcement soon) by
<u>tomorrow (Sep 30) by 11:59PM</u>

# Backup slides from last lecture

# Common prefixes

| | |
|---|---|
| 1.2.0.0/16 | `00000001 00000010 xxxxxxxx xxxxxxxx` |
| 8.0.0.0/8 | `00001000 xxxxxxxx xxxxxxxx xxxxxxxx` |
| 123.10.1.0/24 | `01111011 00001010 00000001 xxxxxxxx` |
| 201.112.10.200/30 | `11001001 01110000 00001010 110010xx` |

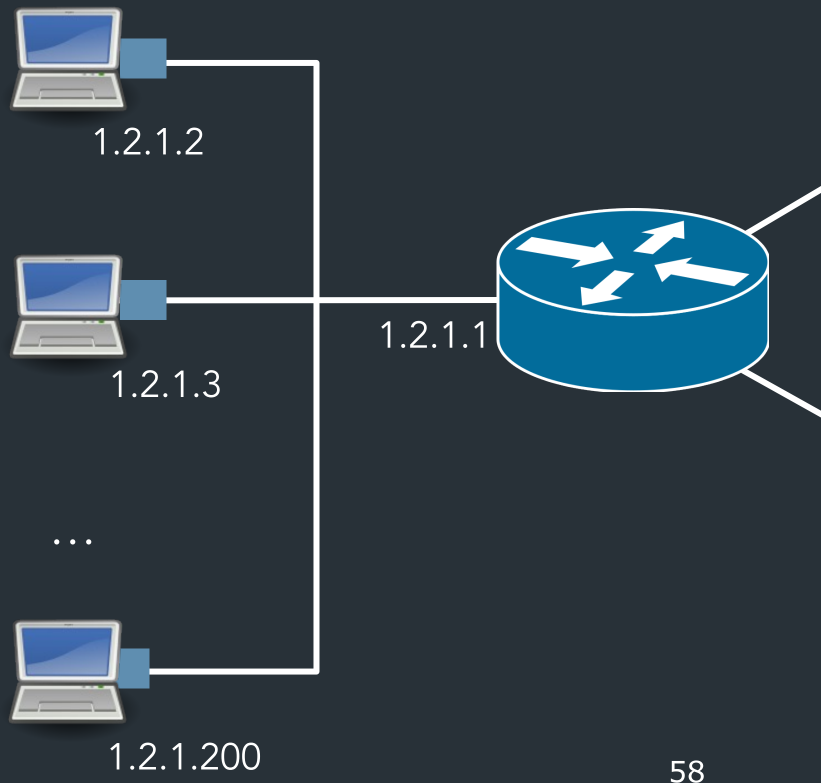
# How IP forwarding works

Assume:

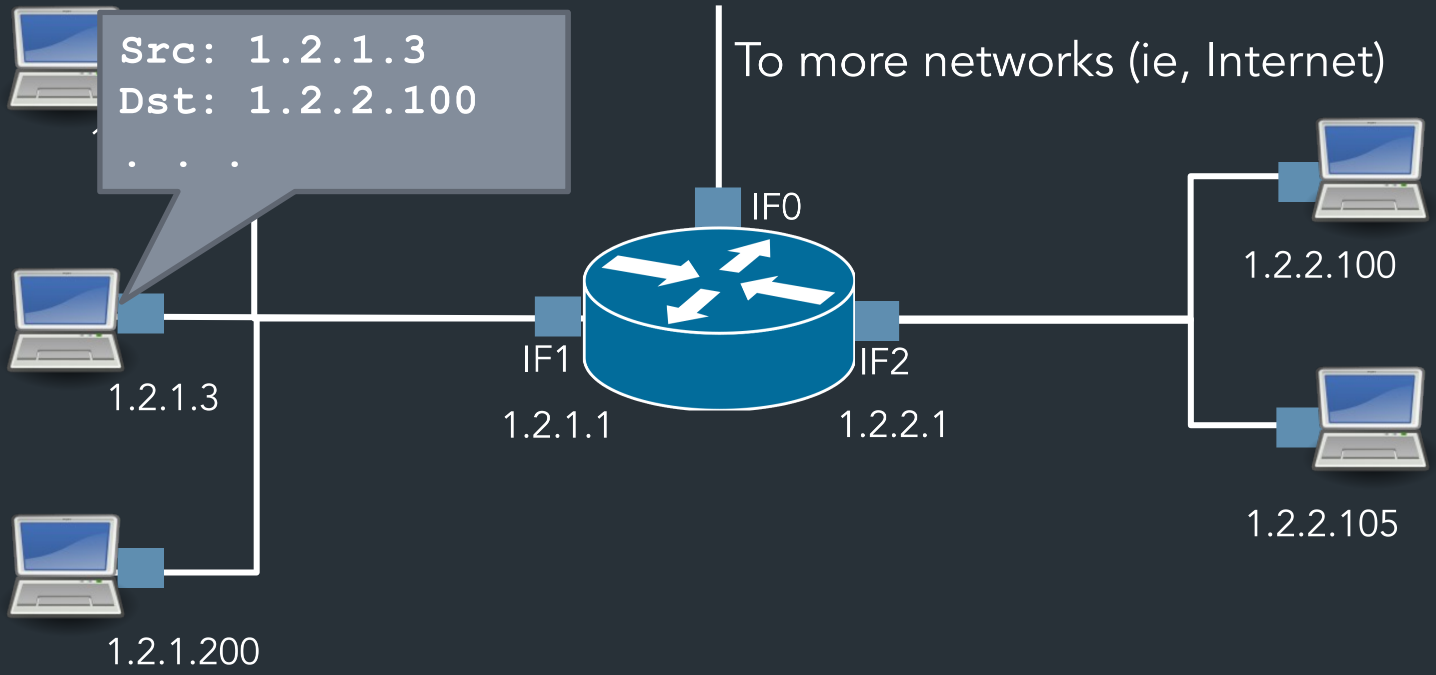
- Communicating on same network is easy—this is the link-layer's job!
- Can map IP addresses to MAC addresses (more on this later)

How to reach an address *outside* this network?

Send packets to a router, which forwards IP packets to other networks

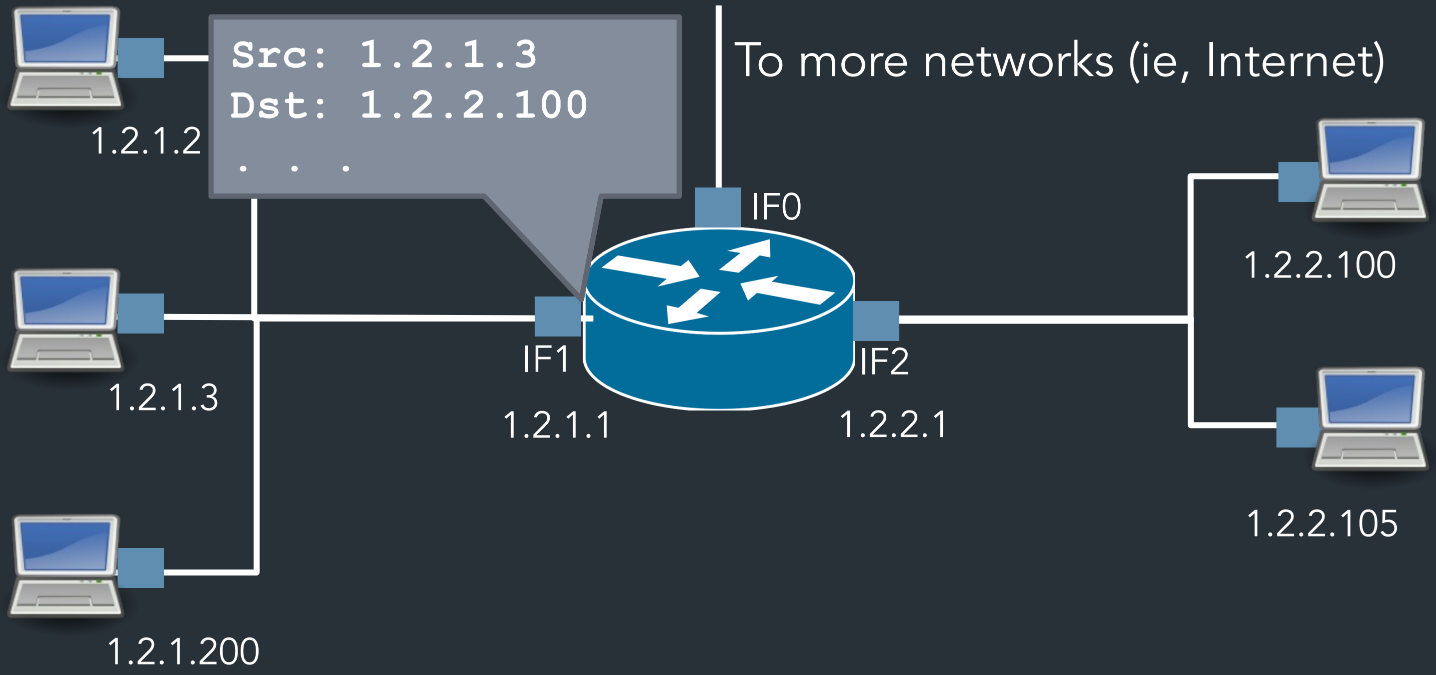


# Forwarding IP packets



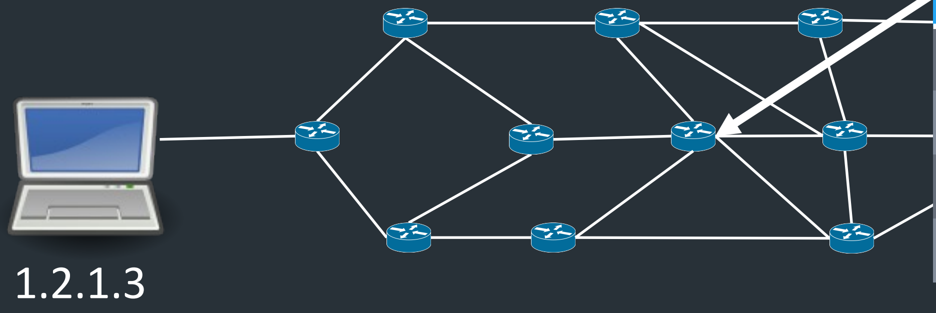
```
Src: 1.2.1.3
Dst: 1.2.2.100
. . .
```

# Forwarding IP packets



```
Src: 1.2.1.3
Dst: 1.2.2.100
. . .
```

# Putting it all together…



| Prefix | Interface |
|---|---|
| . . . | |
| 1.2.0.0/16 | 1.2.0.1 via IF42 |
| 142.0.0/8 | 5.0.0.5 via IF21 |
| . . . | |

- The more connected a router becomes, the more complex its forwarding table… and the more it may change!

- Routing algorithms: routers exchange path information to their forwarding tables (more on this later)

Goal: find the most specific (ie, longest) prefix matching the destination

| Prefix | Interface |
|---|---|
| 1.2.1.0/24 | IF1 |
| 1.2.2.0/24 | IF2 |
| 0.0.0.0/0 | IF0 |

How to reach 1.2.2.100?

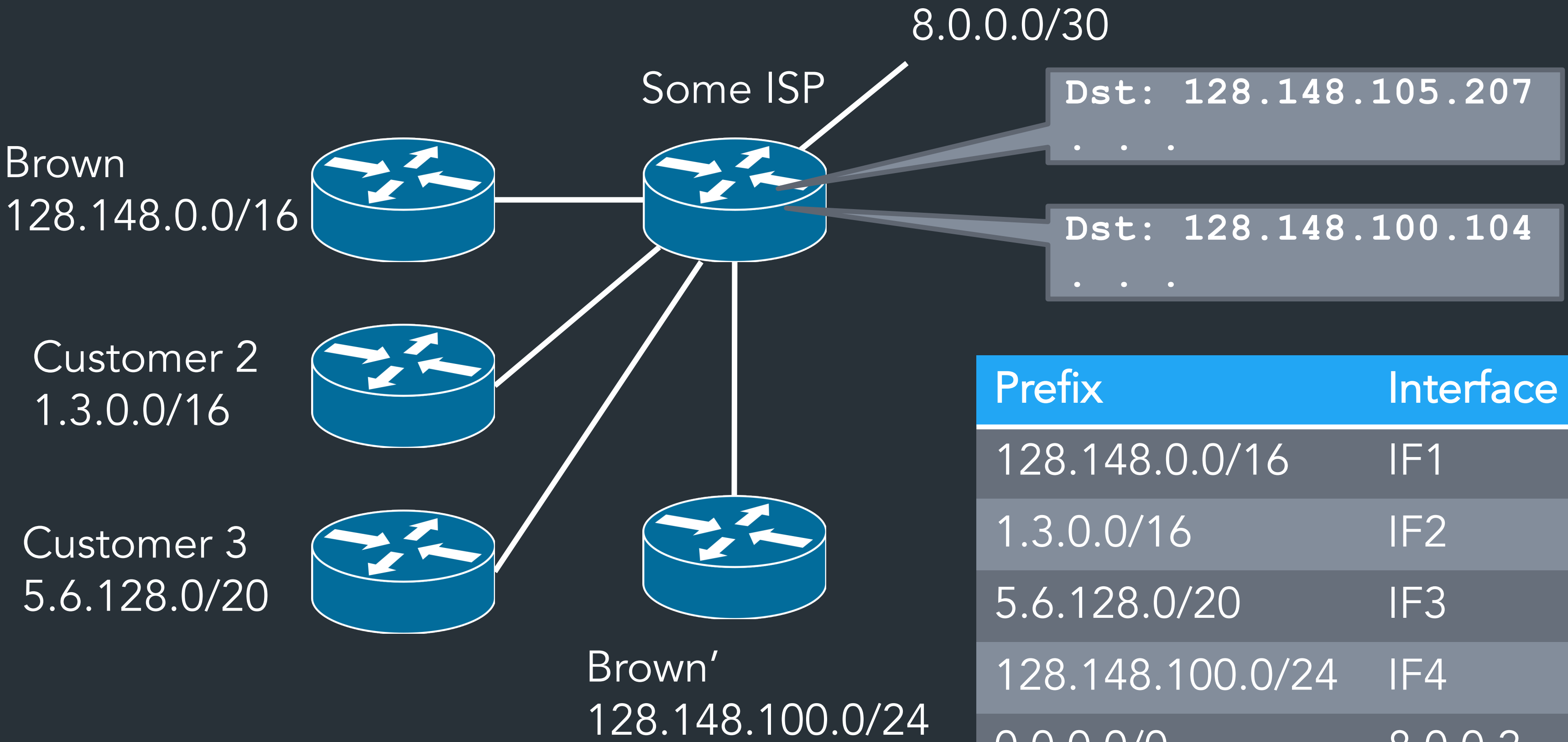
```
1.2.2.100        00000001.00000010.00000010.01100100
                                 ?=
1.2.1.0/24       00000001.00000010.00000001.xxxxxxxx

1.2.2.0/24       00000001.00000010.00000010.xxxxxxxx

0.0.0.0/0        xxxxxxxx.xxxxxxxx.xxxxxxxx.xxxxxxxx
```

Output: IF2

Longest Prefix Matching (LPM): can represent entire IP space in (small) table!



| Prefix | Interface |
|---|---|
| 128.148.0.0/16 | IF1 |
| 1.3.0.0/16 | IF2 |
| 5.6.128.0/20 | IF3 |
| 128.148.100.0/24 | IF4 |
| 0.0.0.0/0 | 8.0.0.2 |