CS 1680
Intra-domain Routing

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

Administrivia

- IP milestone meetings: Should meet with staff today
- IP: Next steps
  - Gearup II: see recording/notes
  - See Implementation Start Guide, other resources
  - Do not leave this project until the last minute

- HW1 due tonight

Warmup

What is the destination MAC address when H1 is sending the following packets?

1)

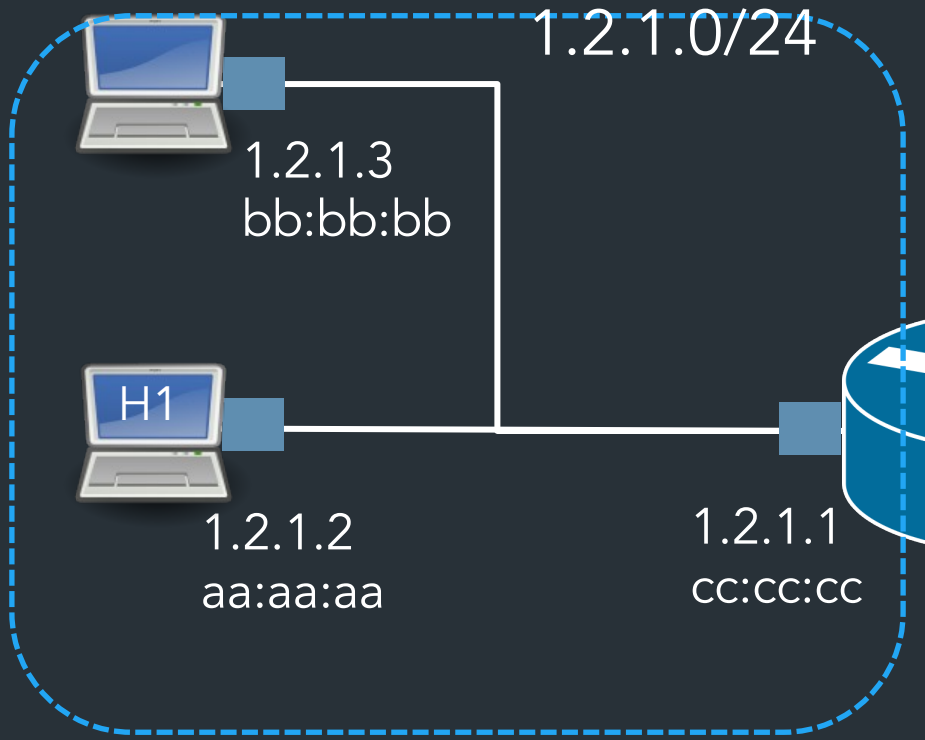
	Src	Dest
Link	aa:aa:aa	???
IP	1.2.1.2	1.2.1.1

2)

	Src	Dest
Link	aa:aa:aa	???
IP	1.2.1.2	1.2.1.3

3)

	Src	Dest
Link	aa:aa:aa	???
IP	1.2.1.2	8.8.8.8 (Google)



H1's forwarding table:

Prefix	IF/Next hop
1.2.1.0/24	IF1
0.0.0.0	1.2.1.1

Recap: IP vs. Link-layer address

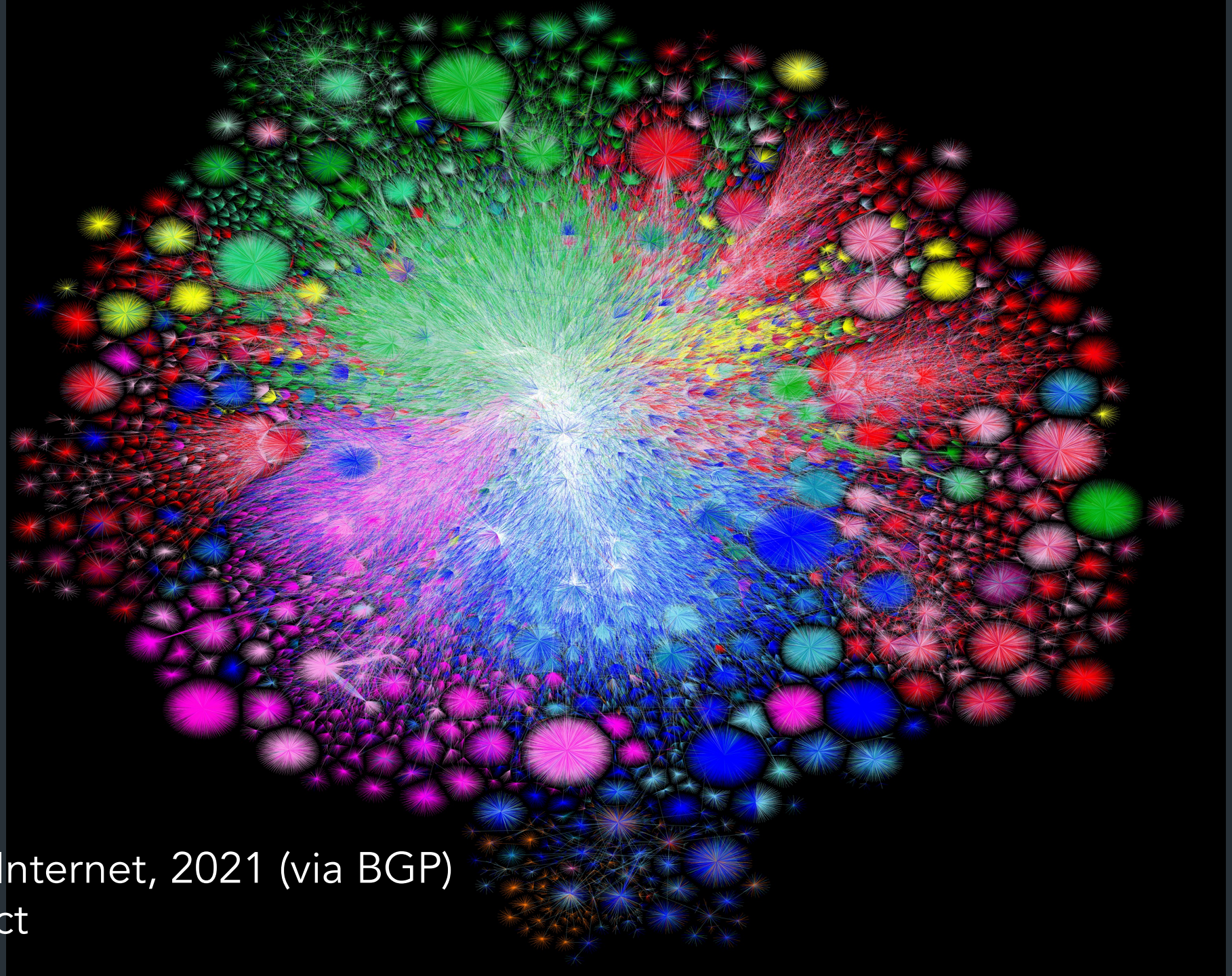
	Src	Dest
Link	aa:aa:aa	cc:cc:cc
IP	1.2.1.2	8.8.8.8 (Google)

Link-layer header info (Ethernet/Wifi/etc)

- Destination MAC address is link-layer addr for packet's next hop
- Changes every hop
- Each hop could use a different link-layer protocol!

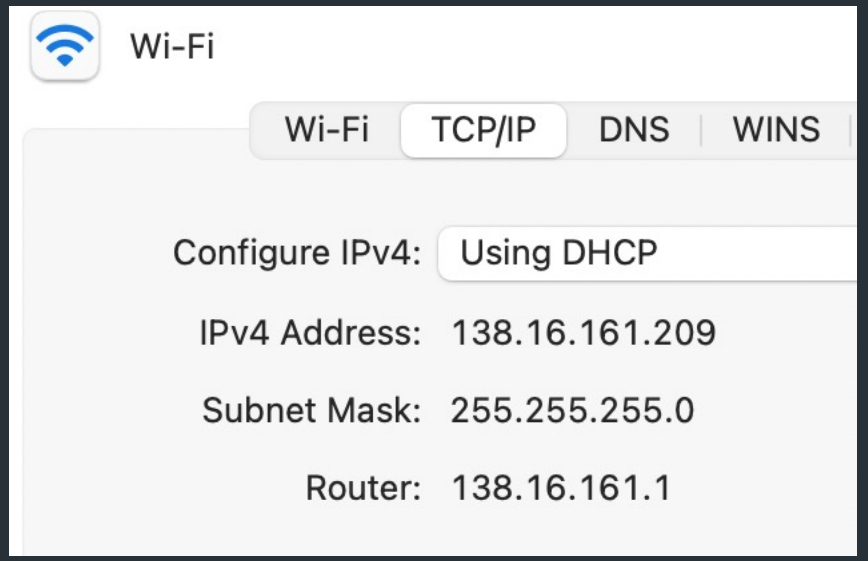
IP header info

- Destination IP is IP address of packet's final destination
- Routers look at destination IP to figure out where packet goes next (and which MAC address goes on packet next)

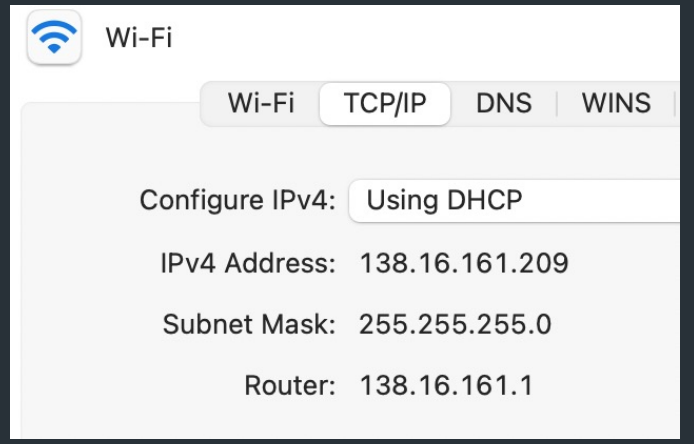


Map of the Internet, 2021 (via BGP)
OPTE project

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, …

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

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)

DHCP: The idea

Dynamic Host Configuration Protocol

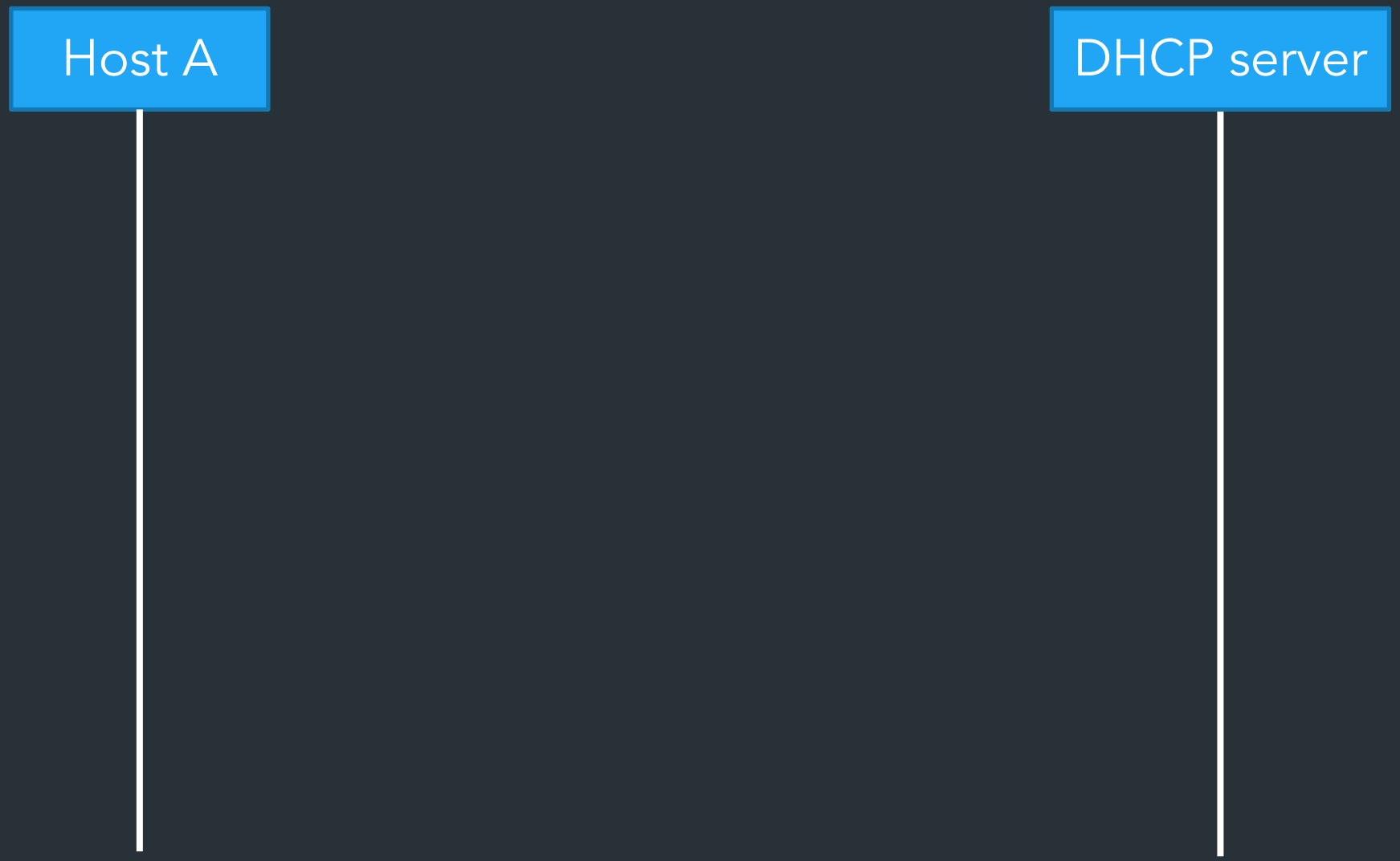
DHCP: The idea

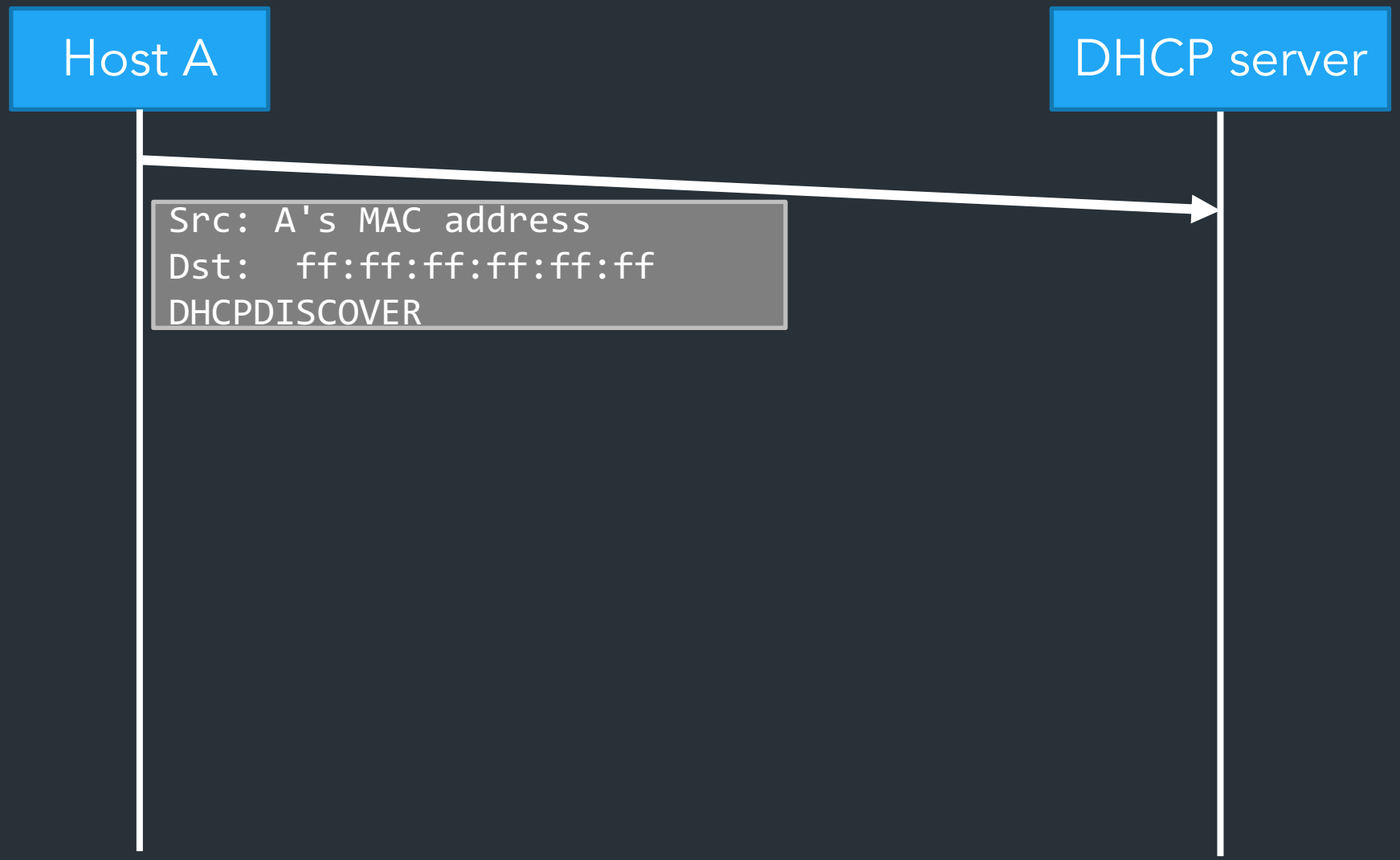
Dynamic Host Configuration Protocol

- Every network has a "pool" of IPs it can assign to hosts
  Some subset of its prefix (eg. 192.168.1.0/24)

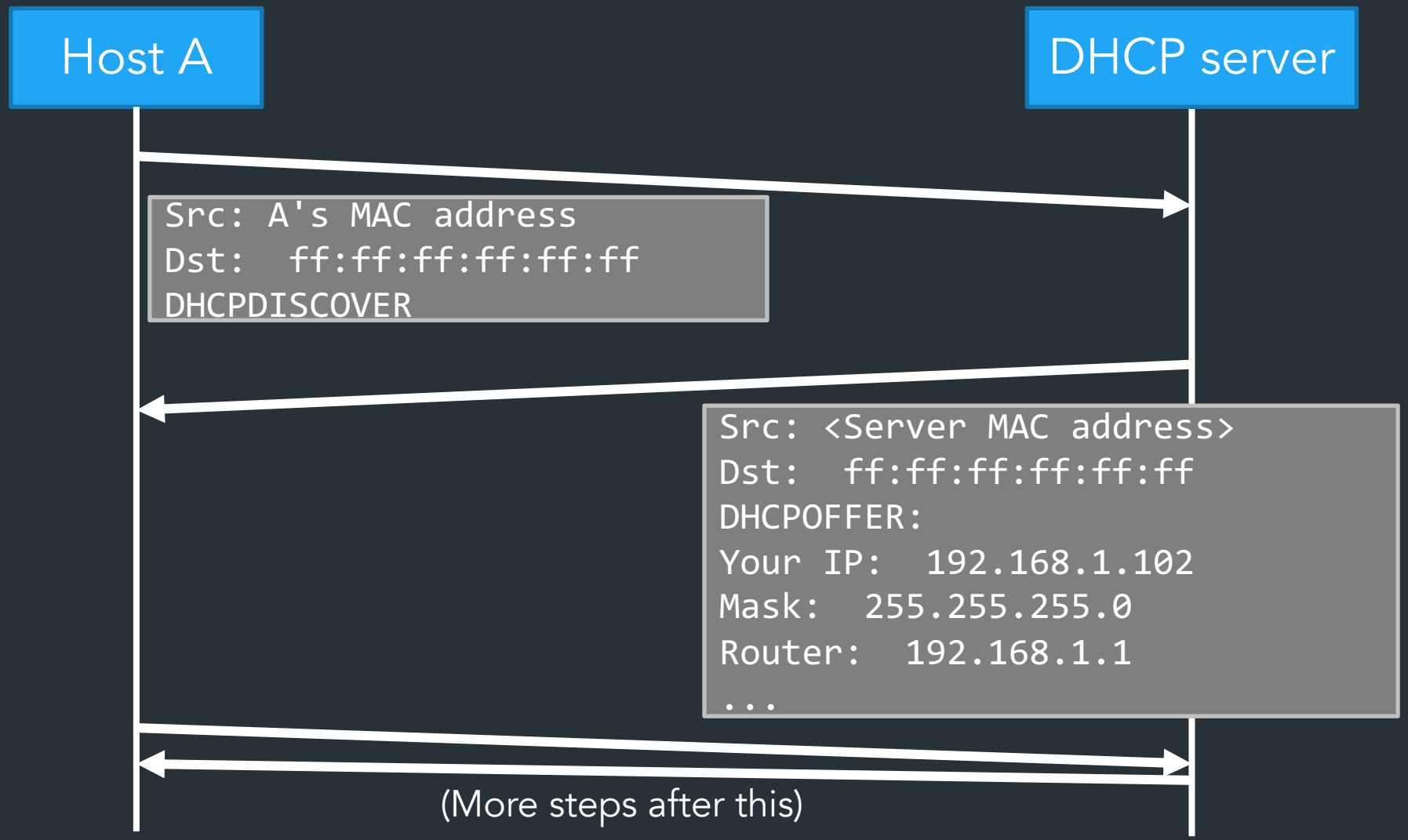
- When a host connects, it asks a DHCP server for an address from the pool

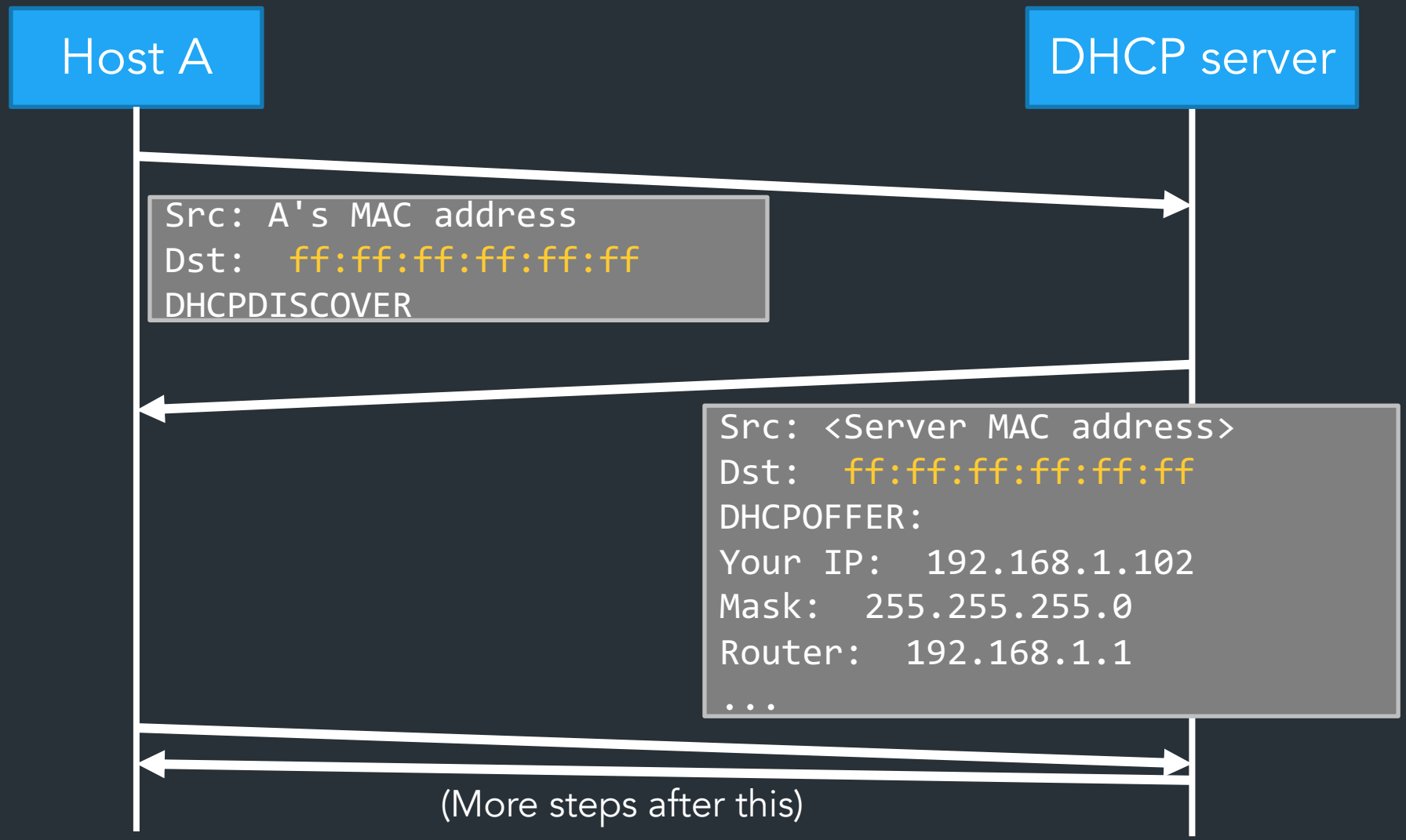
- DHCP server(s) act like allocators: give "leases" to IPs, provide other config info


Host A
DHCP server

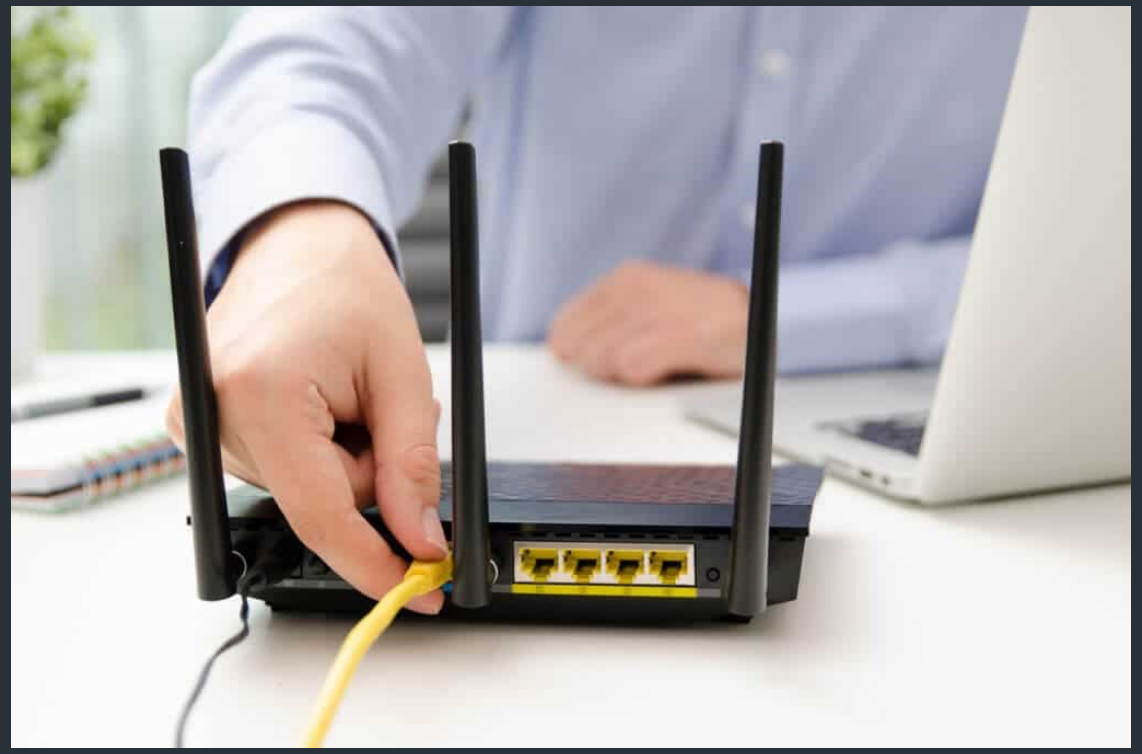


=> Again, host needs to use broadcast address. Why?
=> Problem?


Host A
DHCP server
Src: A's MAC address
Dst: ff:ff:ff:ff:ff:ff
DHCPDISCOVER
Src: <Server MAC address>
Dst: ff:ff:ff:ff:ff:ff
DHCPOFFER:
Your IP: 192.168.1.102
Mask: 255.255.255.0
Router: 192.168.1.1
...
(More steps after this)



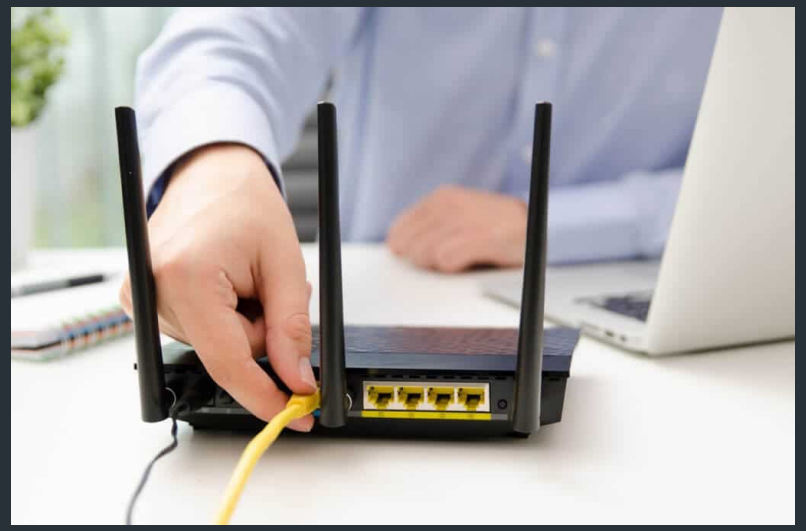
=> Again, host needs to use broadcast address. Why?
=> Problem?



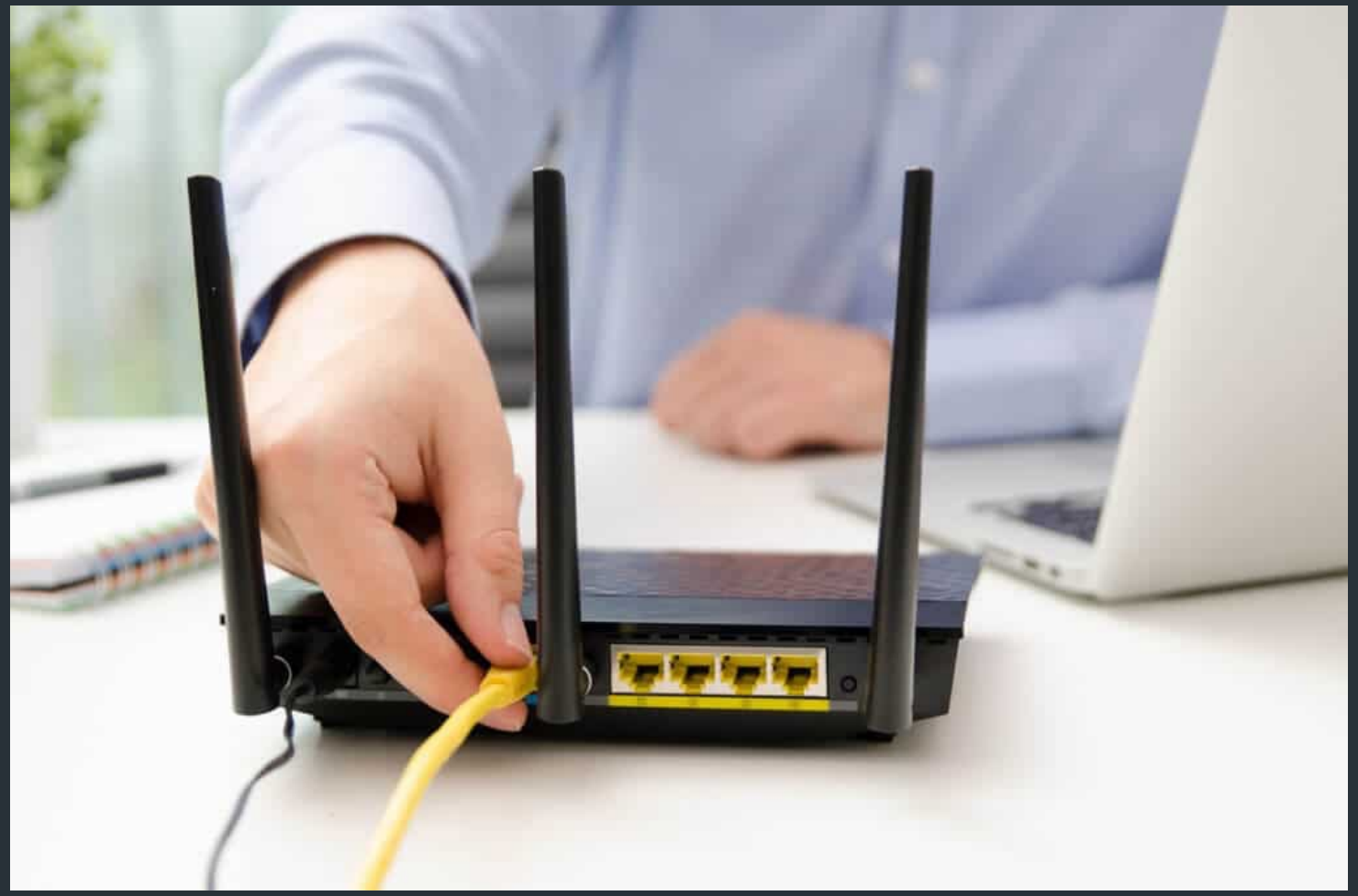
Home routers

The good, the bad, and the ugly…

What’s in a home router?



Story time



Routing

Challenges in moving packets

- Forwarding:

- Routing:

Challenges in moving packets

- Forwarding: given a packet, decide which interface to send the packet (based on IP destination)

- Routing: network-wide process of determining a packet's path through the network

  => How each router builds its forwarding table

Routing is the process of updating forwarding tables

- Routers exchange messages about networks they can reach

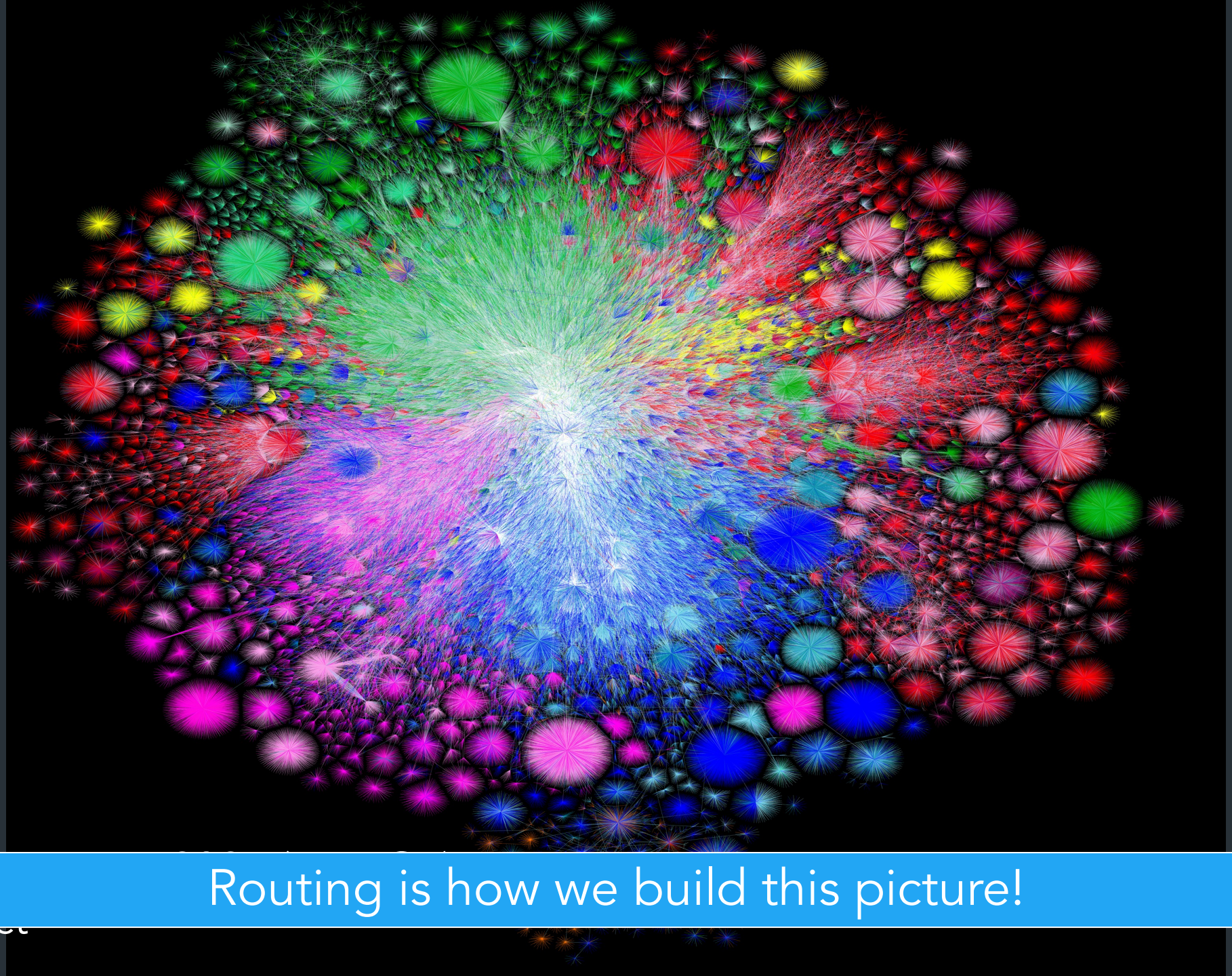
Routing is the process of updating forwarding tables

- Routers exchange messages about networks they can reach

Goal: find optimal route (or *any* route…) for every other destination

This is a hard problem

- Decentralized
- Topology always changing
- Scale!



Map of th
OPTE proj

Routing is how we build this picture!

How do we connect *everything*?

Relies on hierarchical nature of IP addressing

- Smaller routers don't need to know everything, just another router that knows more

  ⇒ Has default route

- Core routers know everything => no default!

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 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
```

At this scale, we think about routing to *whole networks*, i.e., some entity with some set of IP prefixes:

e.g. Brown University @ 128.148.0.0/16, 138.16.0.0/16

At this scale, we think about routing to *whole networks*, i.e., some entity with some set of IP prefixes:

e.g. Brown University @ 128.148.0.0/16, 138.16.0.0/16

We call each entity an Autonomous System (AS):
a single administrative domain that lives on the Internet

Routing is organized in two levels:

- Intra-domain (interior) routing: routing within an AS

- Inter-domain (exterior) routing: routing between ASes

Routing is organized in two levels:

- Intra-domain (interior) routing: routing within an AS

  => Full knowledge of the network inside the AS
  => One administrator, routing policy
  => Strive for optimal paths

  ^ We are here today

- Inter-domain (exterior) routing: routing between ASes

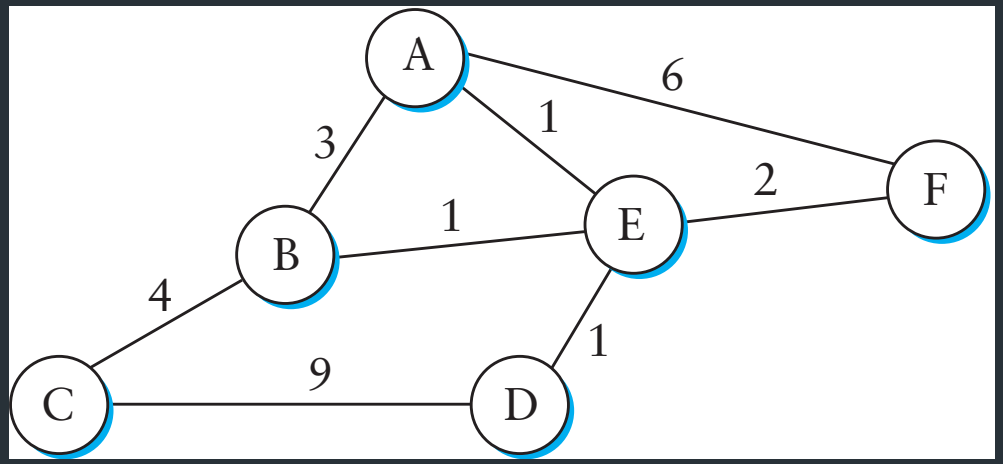
  => None of the above, decisions instead made by *policy* (later)

Intra-Domain (Interior) Routing

Network == Graph

Assign cost to edges

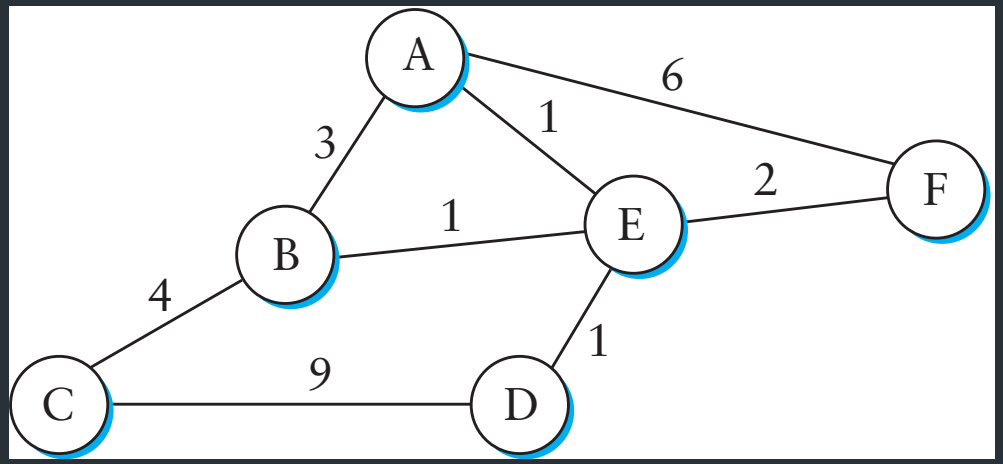
=> latency, bandwidth, ….



Network == Graph

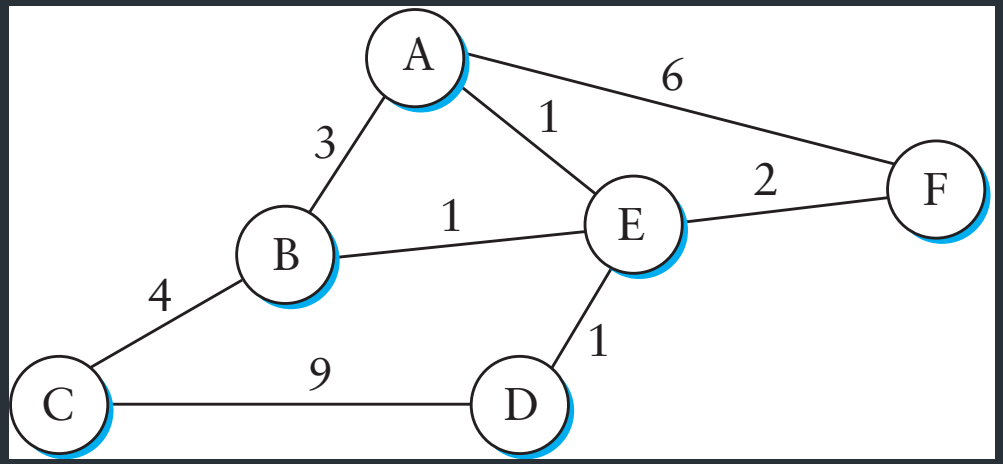
Assign cost to edges

=> latency, bandwidth, ….



Typically, view network as a graph

- Nodes are routers
- Assign some *cost* to each edge
  - latency, b/w, queue length, …



Goal: find lowest-cost path between nodes

- Each node individually computes routes

Collect routes into a *routing table*, used to generate the forwarding table based on lowest-cost path

Generally: routing algorithms are *decentralized*

Generally: routing algorithms are *decentralized*

=>In general, no one entity telling routers what routes to use

=> Even for “interior” routing, where there is one admin, routers independently compute how to update their tables based on latest info from other routers

Two classes of intra-domain routing algorithms

Distance Vector (Bellman-Ford shortest path algorithm)

Link State (Djikstra/Prim shortest path algorithm)

Two classes of intra-domain routing algorithms

Distance Vector (Bellman-Ford shortest path algorithm)

- Idea: routers get updates from their neighbors

Link State (Djikstra/Prim shortest path algorithm)

First type: Distance Vector Routing

Distance Vector Routing

- Each node maintains a *routing table*

- Exchange *updates* with neighbors about node's links:
  => List of (Destination, Cost) pairs

Distance Vector Routing

- Each node maintains a routing table
- Exchange *updates* with neighbors about node's links:
  => List of (Destination, Cost) pairs
- When to send updates?
  - Periodically (seconds to minutes)
  - Whenever table changes (*triggered* update)
  - Time out an entry if no updates within some time interval

Distance Vector Routing

- Each node maintains a routing table
- Exchange *updates* with neighbors about node's links:
  => List of (Destination, Cost) pairs
- When to send updates?
  - Periodically (seconds to minutes)
  - Whenever table changes (*triggered* update)
  - Time out an entry if no updates within some time interval

Dest.	Cost	Next Hop
A	3	S
B	4	T
C	5	S
D	6	U

Distance Vector: Update rules

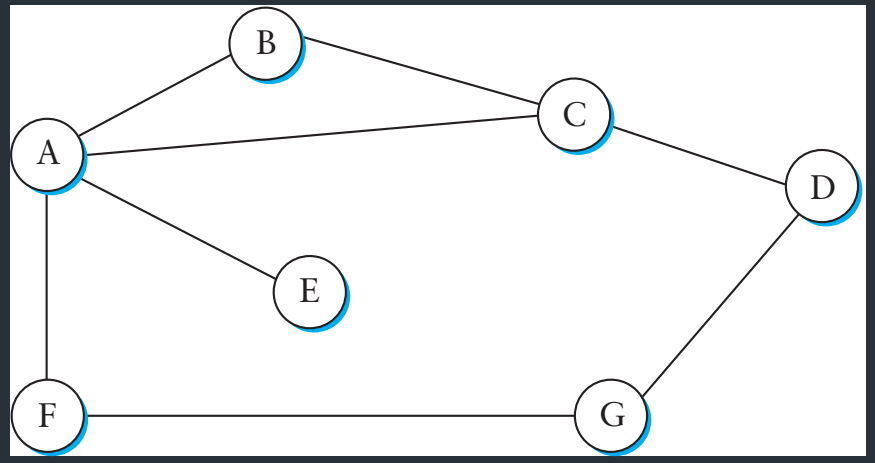
Say router R receives an update (D, c_D) from neighbor N at cost C_N

=> Know: R can reach D via N with cost c = c_D + c_N

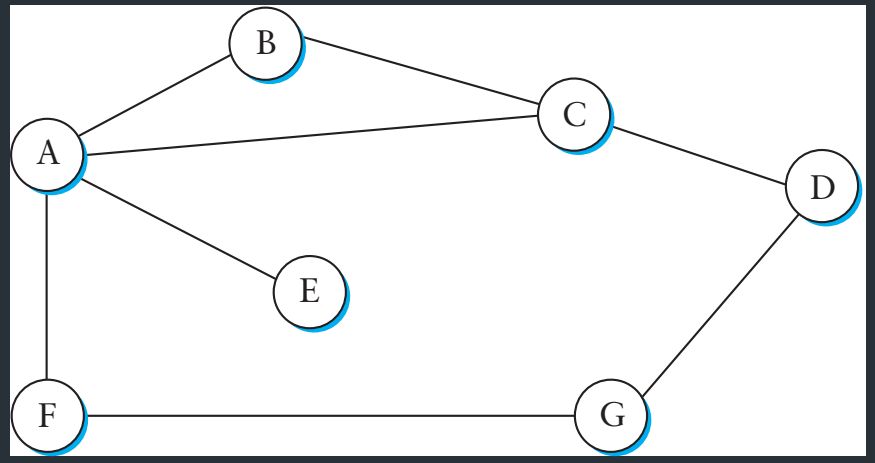
How to update table?

1. If D not in table, add (D, c, N) (New route!)
2. If table has entry (D, M, c_{old}):
   - if c < c_{old}: update table to (D, c, M). (Lower cost!)
   - if c > c_{old} and M == N: update table to (D, c, N) (Cost increased!)
   - if c > c_{old} and M != N: ignore (N is better)
   - if c == c_{old} and M ==N: no change (No new info)
     (Just refresh timeout)

DV Example



DV Example



B's routing table

Dest.	Cost	Next Hop
(B)	(0)	(B)
A	1	A
C	1	C
D	2	C
E	2	A
F	2	A
G	3	A

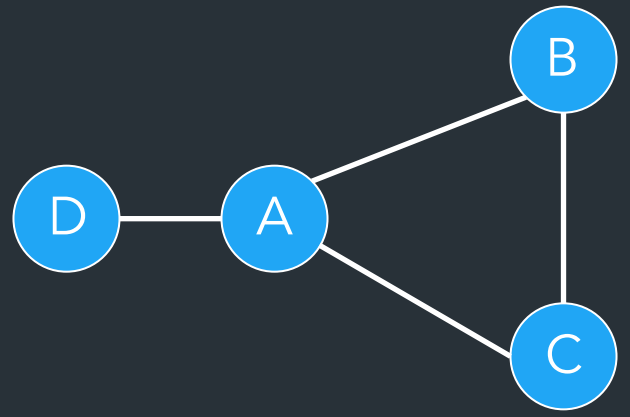
Suppose router R has the following table:

Dest.	Cost	Next Hop
A	3	S
B	4	T
C	5	S
D	6	U

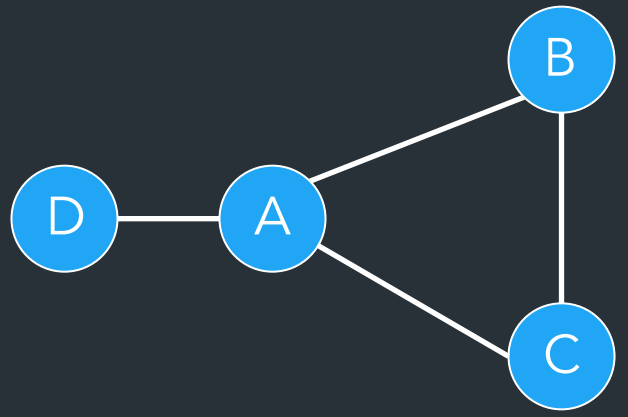
What happens when it gets this update from router S?

Dest.	Cost
A	2
B	3
C	5
D	4
E	2

What happens when the D-A link fails?

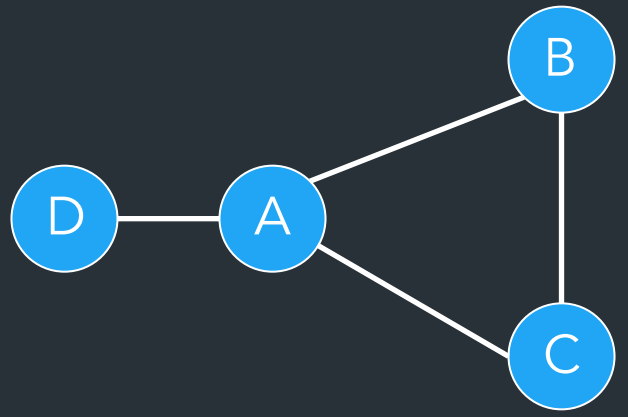


What happens when the D-A link fails?



=> “Count to Infinity” problem

What happens when the D-A link fails?



Updates occur in a loop with increasing cost until cost reaches infinity (16)!
=> Count to infinity => long time to converge when links fail

Can we avoid loops?

- Does IP TTL help? Nope.
- Simple approach: consider a small cost *n* (e.g., 16) to be infinity

Can we avoid loops?

- Does IP TTL help? Nope.
- Simple approach: consider a small cost *n* (e.g., 16) to be infinity

Fundamental problem: distance vector only based on local information!
=> Not enough info to resolve loops, race conditions, count-to-infinity, but there are some tricks…

RFC1058 (1988): The original RIP standard*

supply the information that is needed to do routing.

1.1. Limitations of the protocol

This protocol does not solve every possible routing problem. As mentioned above, it is primary intended for use as an IGP, in reasonably homogeneous networks of moderate size. In addition, the following specific limitations should be mentioned:

*: Obsoleted by RFC2453 (don't use RFC 1058 for the project, Use RFC 2453 instead)

One strategy: Split Horizon

- When sending updates to node A, don’t include routes you learned from A
- Prevents B and C from sending cost 2 to A

Split Horizon + Poison reverse

- Rather than not advertising routes learned from A, explicitly include cost of ∞.
- Faster to break out of loops, but increases advertisement sizes

Split Horizon + Poison reverse

- Rather than not advertising routes learned from A, explicitly include cost of ∞.
- Faster to break out of loops, but increases advertisement sizes

=> Does it help?

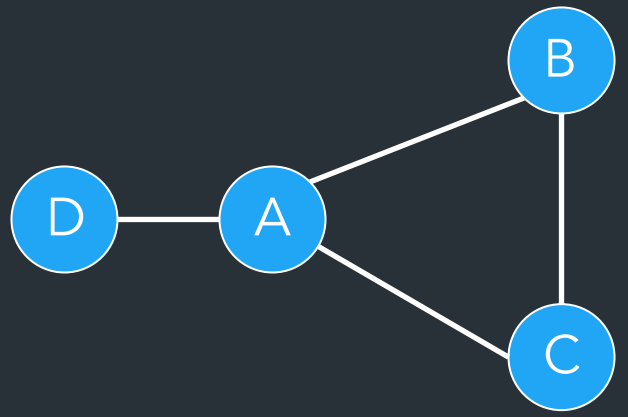
Split Horizon + Poison reverse

- Rather than not advertising routes learned from A, explicitly include cost of ∞.
- Faster to break out of loops, but increases advertisement sizes

⇒ Does it help? Not completely.

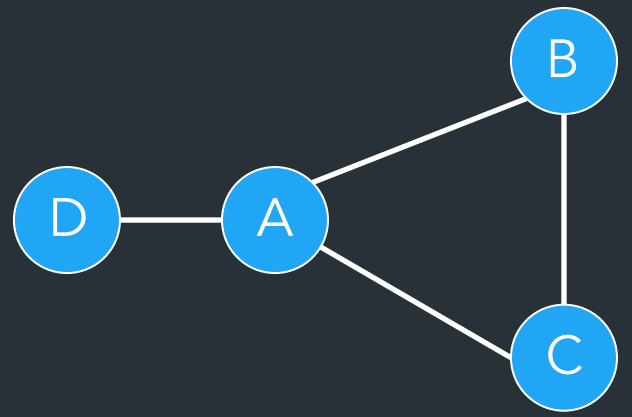
=> A common convention, might reduce time to converge, but overall hard to see effect vs. split horizon

Even with split horizon + poison reverse,
can still create loops with >2 nodes!

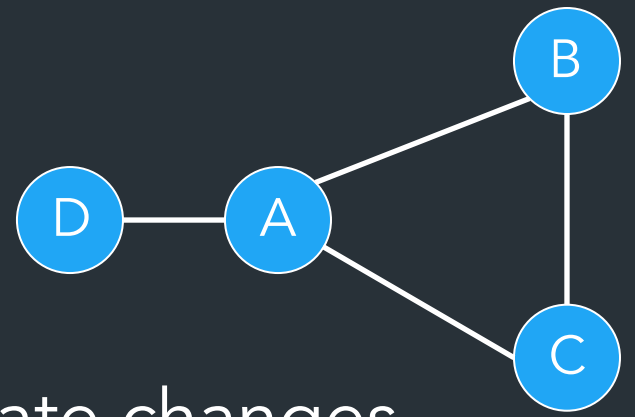


Even with split horizon + poison reverse,
can still create loops with >2 nodes!

What else can we do?



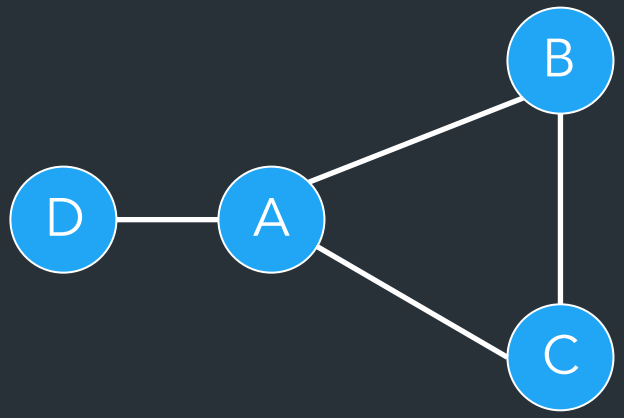
Even with split horizon + poison reverse,
can still create loops with >2 nodes!



What else can we do?

- Triggered updates: send update as soon as link state changes
- Hold down: delay using new routes for certain time, affects convergence time

Practice



B's routing table

Dest.	Cost	Next Hop
A	1	A
C	1	C
D	2	A

Routers A,B,C,D use RIP. When B sends a periodic update to A, what does it send…

- When using standard RIP?
- When using split horizon + poison reverse?

From RFC2453, RIP v2 (1998):

3.2 Limitations of the Protocol

This protocol does not solve every possible routing problem. As mentioned above, it is primary intended for use as an IGP in networks of moderate size. In addition, the following specific limitations are be mentioned:

- The protocol is limited to networks whose longest path (the network's diameter) is 15 hops. The designers believe that the basic protocol design is inappropriate for larger networks. Note that this statement of the limit assumes that a cost of 1 is used for each network. This is the way RIP is normally configured. If the system administrator chooses to use larger costs, the upper bound of 15 can easily become a problem.

- The protocol depends upon "counting to infinity" to resolve certain unusual situations. (This will be explained in the next section.) If the system of networks has several hundred networks, and a routing loop was formed involving all of them, the resolution of the loop would require either much time (if the frequency of routing updates were limited) or bandwidth (if updates were sent whenever changes were detected). Such a loop would consume a large

Link State Routing

Link State Routing: The Alternative

Example: OSPF

Strategy: each router sends information about its neighbors to *all nodes*

Link State Routing: The Alternative

Example: OSPF

Strategy: each router sends information about its neighbors to *all nodes*

- Nodes build the full graph, not just neighbor info

- Updates have more state info

Link State Routing: The Alternative

Strategy: each router sends information about its neighbors to *all nodes*

- Nodes build the full graph, not just neighbor info

  => Can define "areas" to scale this in large networks

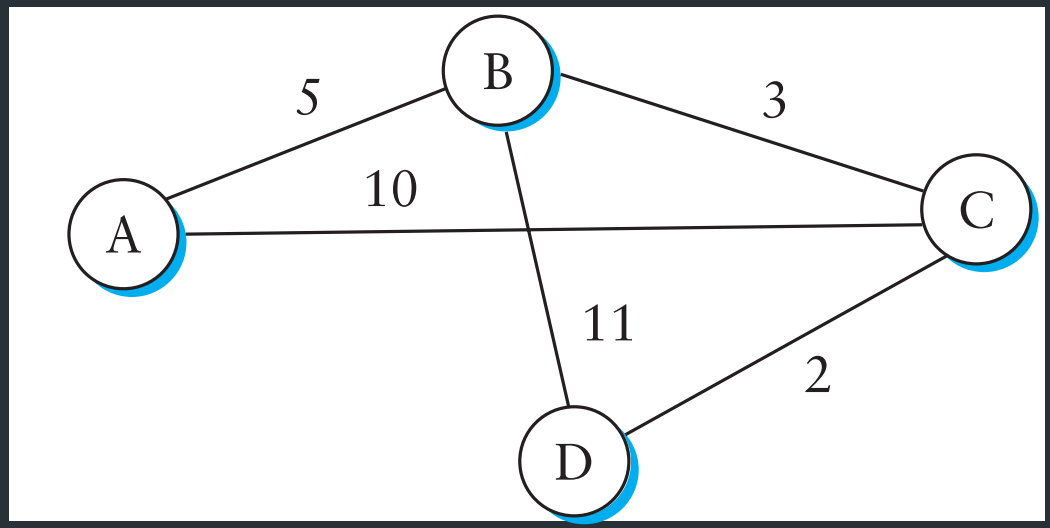
- Updates have more state info
  - Node IDs, version info (sequence number, TTL), …

    => Can be used to detect loops, stale info

⇒ Focuses on building a consistent view of network state

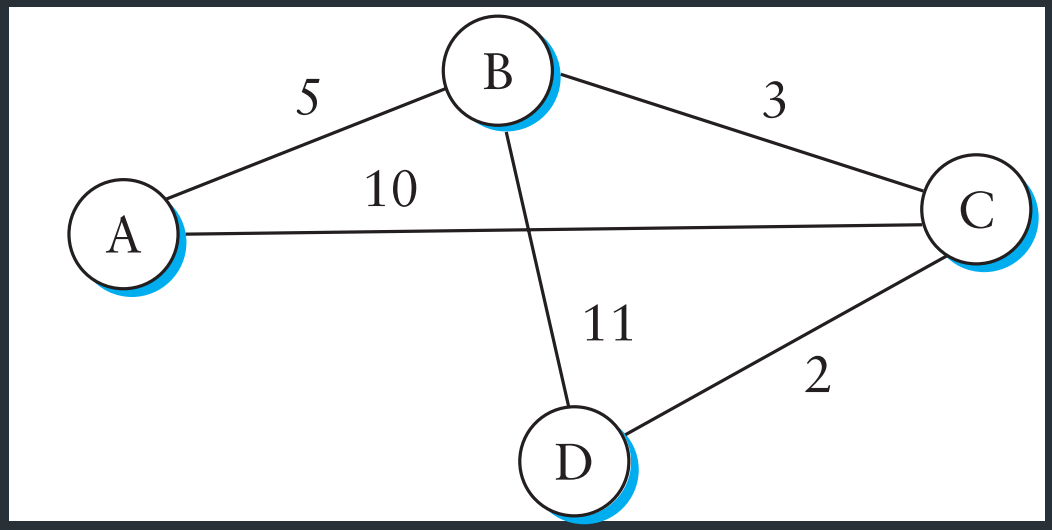
Link State Routing: how it works

- Each node computes shortest paths from itself
- How? Dijkstra's algorithm
  - Given: full graph of nodes
  - Find best next hop to each other node



Link State Routing: how it works

- Each node computes shortest paths from itself
- How? Dijkstra's algorithm
  - Given: full graph of nodes
  - Find best next hop to each other node



Tradeoffs?

Tradeoffs: Link State (LS) vs. Distance Vector (DV)

- LS sends more messages vs. DV
- LS requires more computation vs. DV
- Convergence time
  - DV: Varies (count-to-infinity)
  - LS: Reacts to updates better
- Robustness
  - DV: Bad updates can affect whole network
  - LS: Bad updates affect a single node's update

=> RIP isn't used in production environments anymore…

Examples (and complexity)

Algorithm	Method	Pages in RFC
RIP v2 (RFC 2453)	Distance Vector	38

Examples (and complexity)

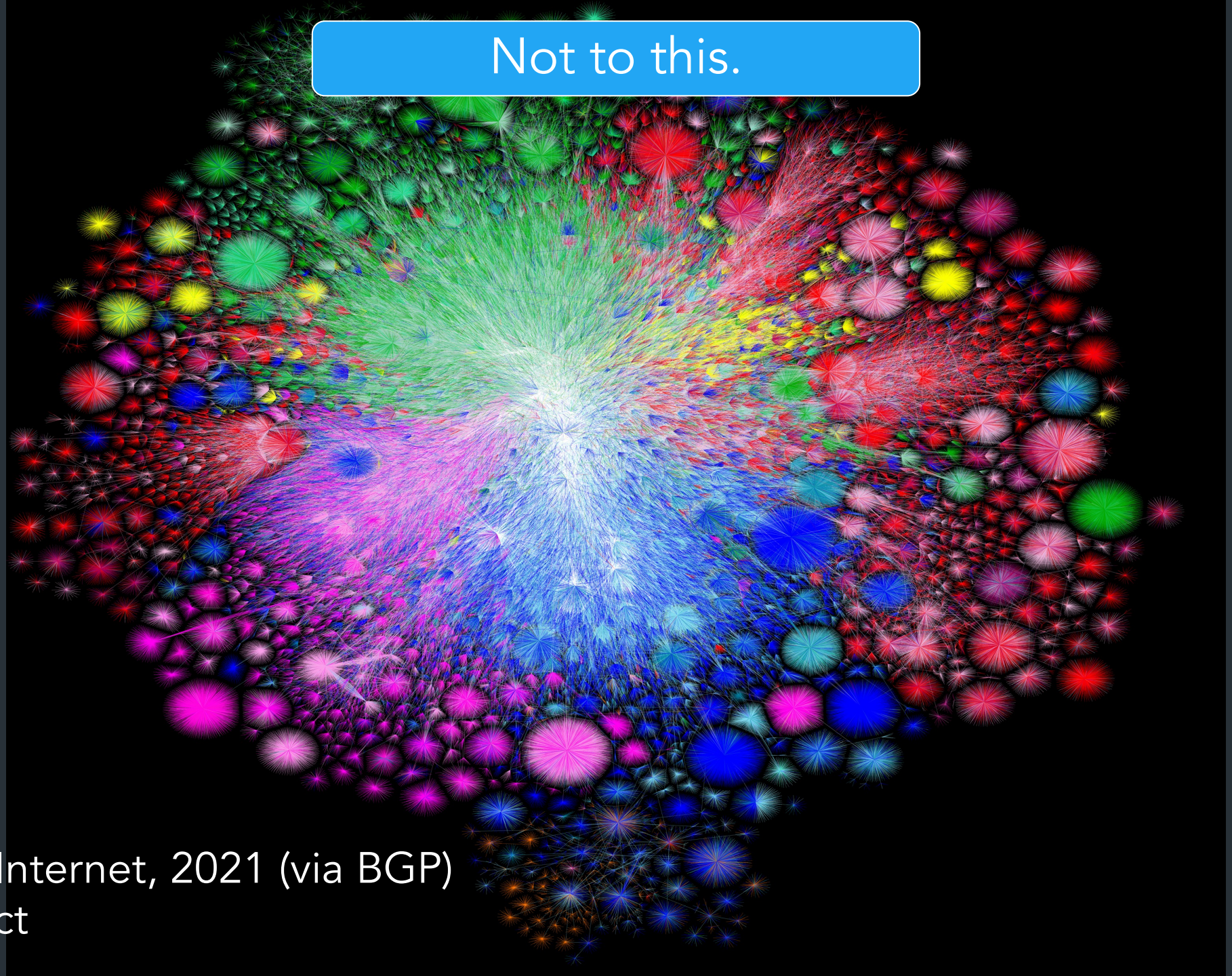
Algorithm	Method	Pages in RFC
RIP v2 (RFC 2453)	Distance Vector	38
OSPF (RFC 2328)	Link-State	244
IS-IS (OSI)	Link-State	210

OSPF: Open Shortest Path First

IS-IS: OSI standard similar to OSPF, doesn't depend on IP

So why not just use OSPF everywhere?

Does it scale?



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

Why not?

⇒ Can’t build a full routing graph with the whole Internet

⇒ More a policy problem than a technical problem

Why not?

⇒ Can't build a full routing graph with the whole Internet

⇒ More a policy problem than a technical problem

- No unified way to represent cost
- No single administrator
- Networks (ASes) have different policies on what "best" routes to choose

Need a different routing mechanism for exterior routing => BGP

With BGP: we talk about routing to Autonomous Systems (ASes)

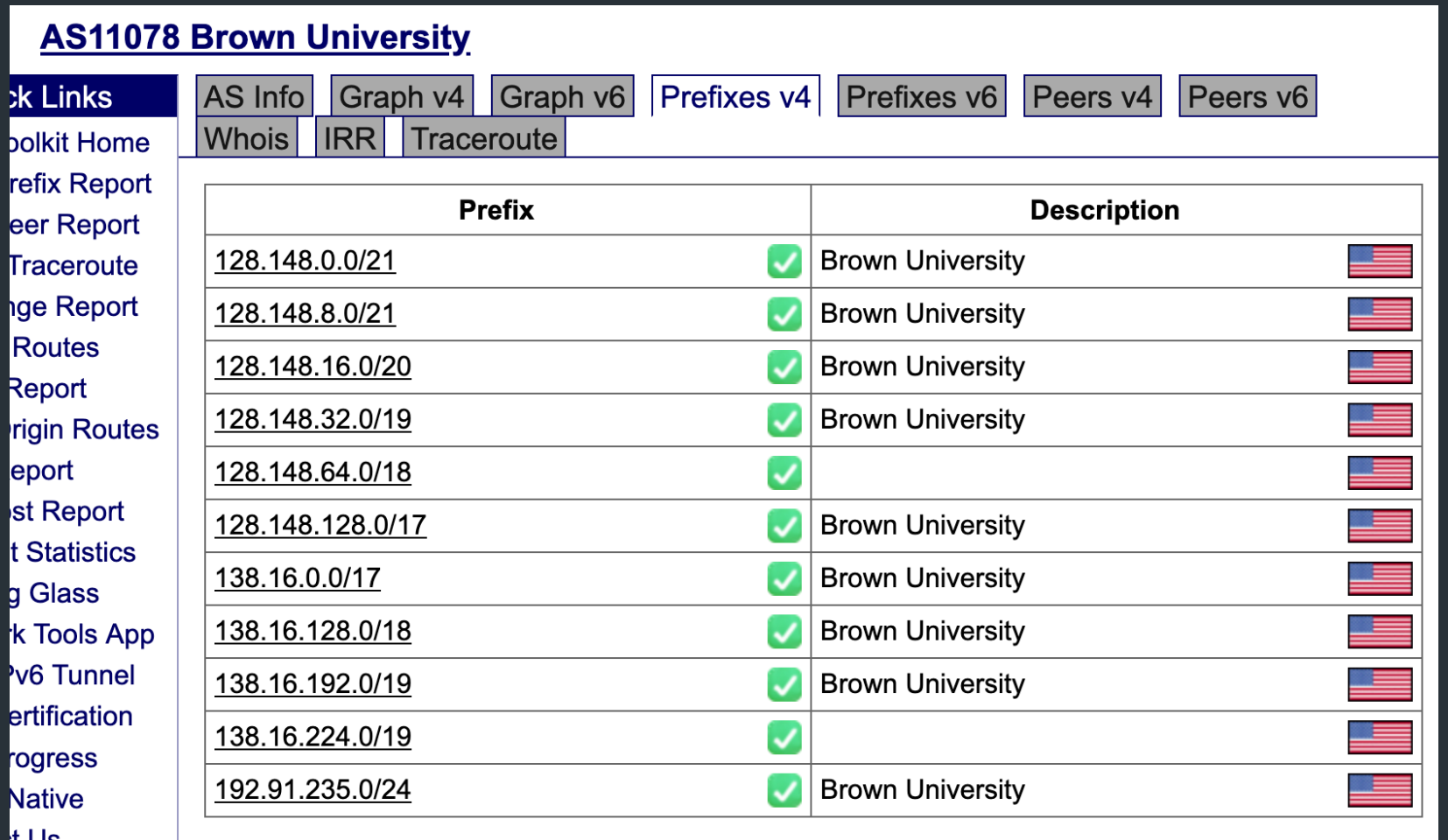
= > Generally, large networks advertise some set of IP prefixes to the Internet

=> Each AS has its own *policy* for how it does routing

With BGP: we talk about routing to Autonomous Systems (ASes)

= > Generally, large networks advertise some set of IP prefixes to the Internet

=> Each AS has its own *policy* for how it does routing



AS11078 Brown University

ck Links
oolkit Home
refix Report
eer Report
Traceroute
nge Report
Routes
Report
rigin Routes
eport
st Report
t Statistics
g Glass
k Tools App
v6 Tunnel
ertification
rogress
Native

AS Info | Graph v4 | Graph v6 | Prefixes v4 | Prefixes v6 | Peers v4 | Peers v6 | Whois | IRR | Traceroute

Prefix	Description
128.148.0.0/21	Brown University
128.148.8.0/21	Brown University
128.148.16.0/20	Brown University
128.148.32.0/19	Brown University
128.148.64.0/18	
128.148.128.0/17	Brown University
138.16.0.0/17	Brown University
138.16.128.0/18	Brown University
138.16.192.0/19	Brown University
138.16.224.0/19	
192.91.235.0/24	Brown University

BGP: A Path Vector Protocol

Distance vector + extra information

eg. "I can reach prefix 128.148.0.0/16 through ASes 44444 3356 14325 11078"

BGP: A Path Vector Protocol

Distance vector + extra information

eg. "I can reach prefix 128.148.0.0/16 through ASes 44444 3356 14325 11078"

- For each route, router store the complete path (ASs)
- No extra computation, just extra storage (and traffic)
- BGP gets to decide what path to *advertise* to neighbors

Fun fact: loops are easy to avoid…

eg. *“I can reach prefix 128.148.0.0/16 through ASes 44444 3356 14325 11078”*

What would a loop look like?

BGP: A Path Vector Protocol

Distance vector + extra information

eg. "I can reach prefix 128.148.0.0/16 through ASes 44444 3356 14325 11078"

- For each route, router store the complete path (ASs)
- No extra computation, just extra storage (and traffic)
- BGP gets to decide what path to *advertise* to neighbors

⇒ BGP routers look at path to decide how to "propagate" route, based on policy

⇒ Can easily avoid loops!

BGP Implications

- No loops!
- Not all ASs know all paths
- Reachability not guaranteed
  - Decentralized combination of policies

- Scaling
  - 74K ASs
  - 959K+ prefixes
  - ASs with one prefix: 25K
  - Most prefixes by one AS: 10008 (Uninet S.A. de C.V., MX)

Source: cidr-report 18Oct2022

Why study BGP?

BGP is what makes the Internet run.

Lots of problems…

Explainer

Facebook outage: what went wrong and why did it take so long to fix after social platform went down?

RYAN SINGEL SECURITY FEB 25, 2008 10:37 AM

Pakistan's Accidental YouTube Re-Routing Exposes Trust Flaw in Net

TECHNOLOGY

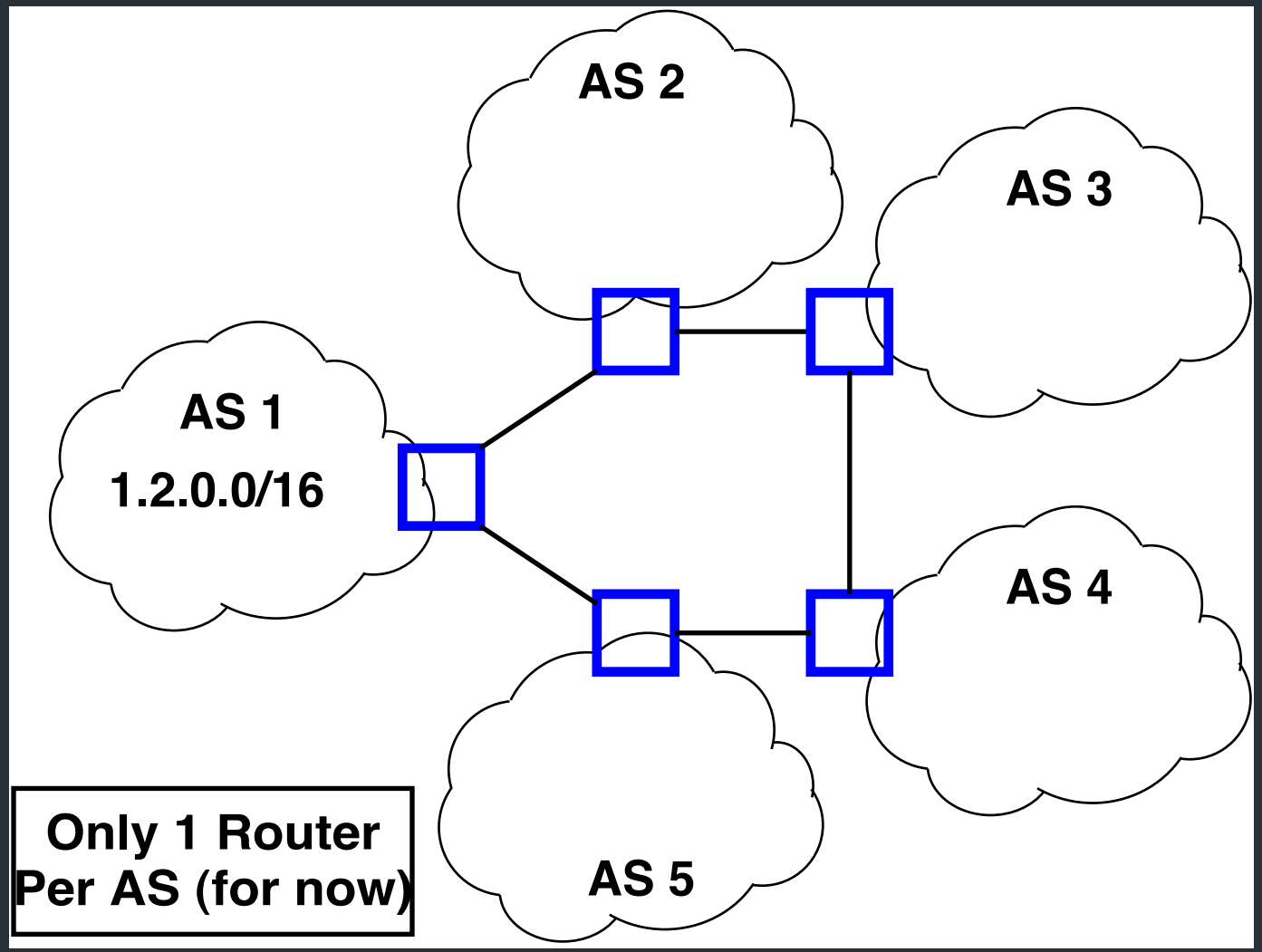
How Was Egypt's Internet Access Shut Off?

How Russia Took Over Ukraine's Internet in Occupied Territories

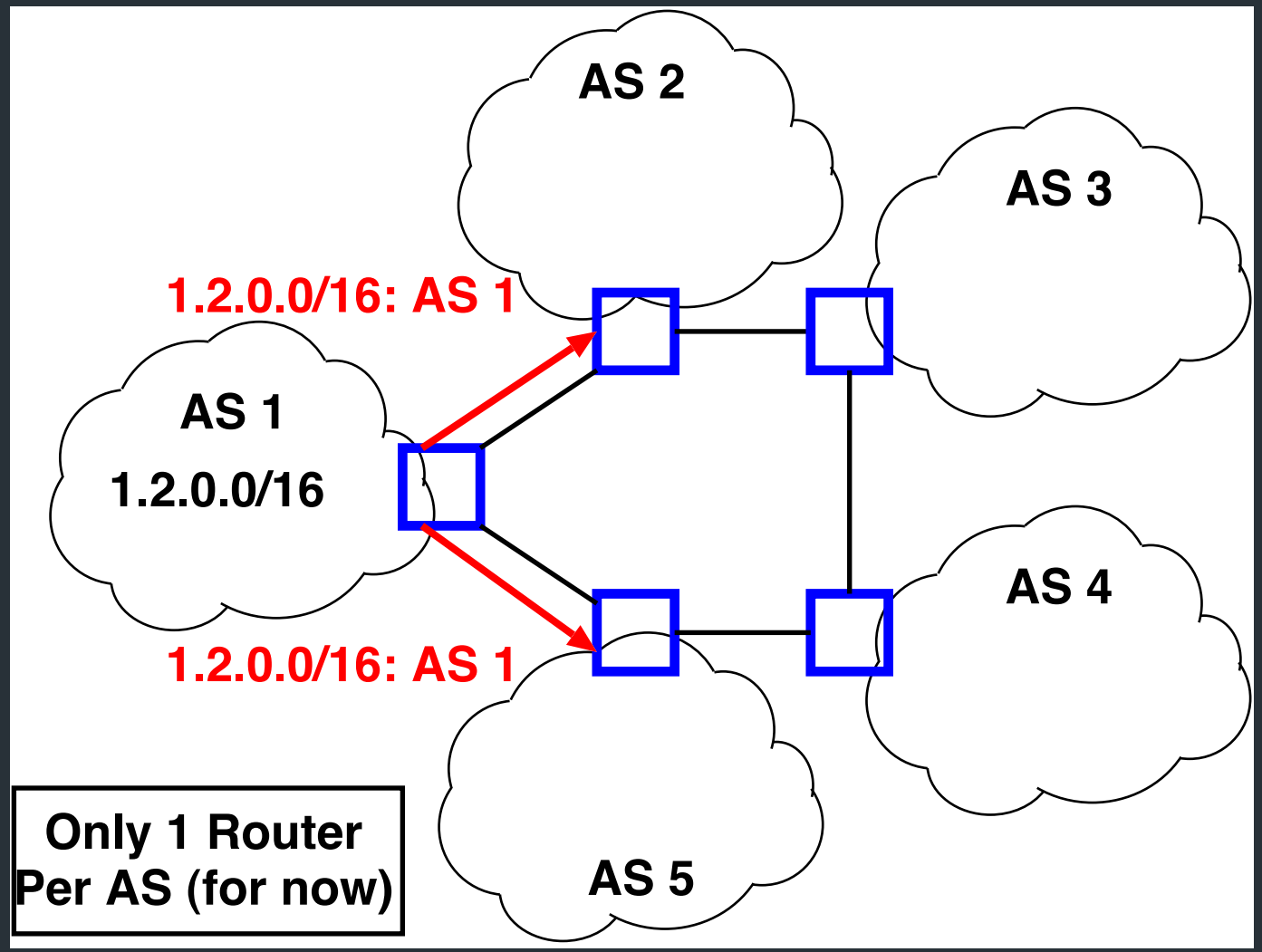
By Adam Satariano and
Graphics by Scott Reinhard
Aug. 9, 2022

A Network Operations Center (NOC)

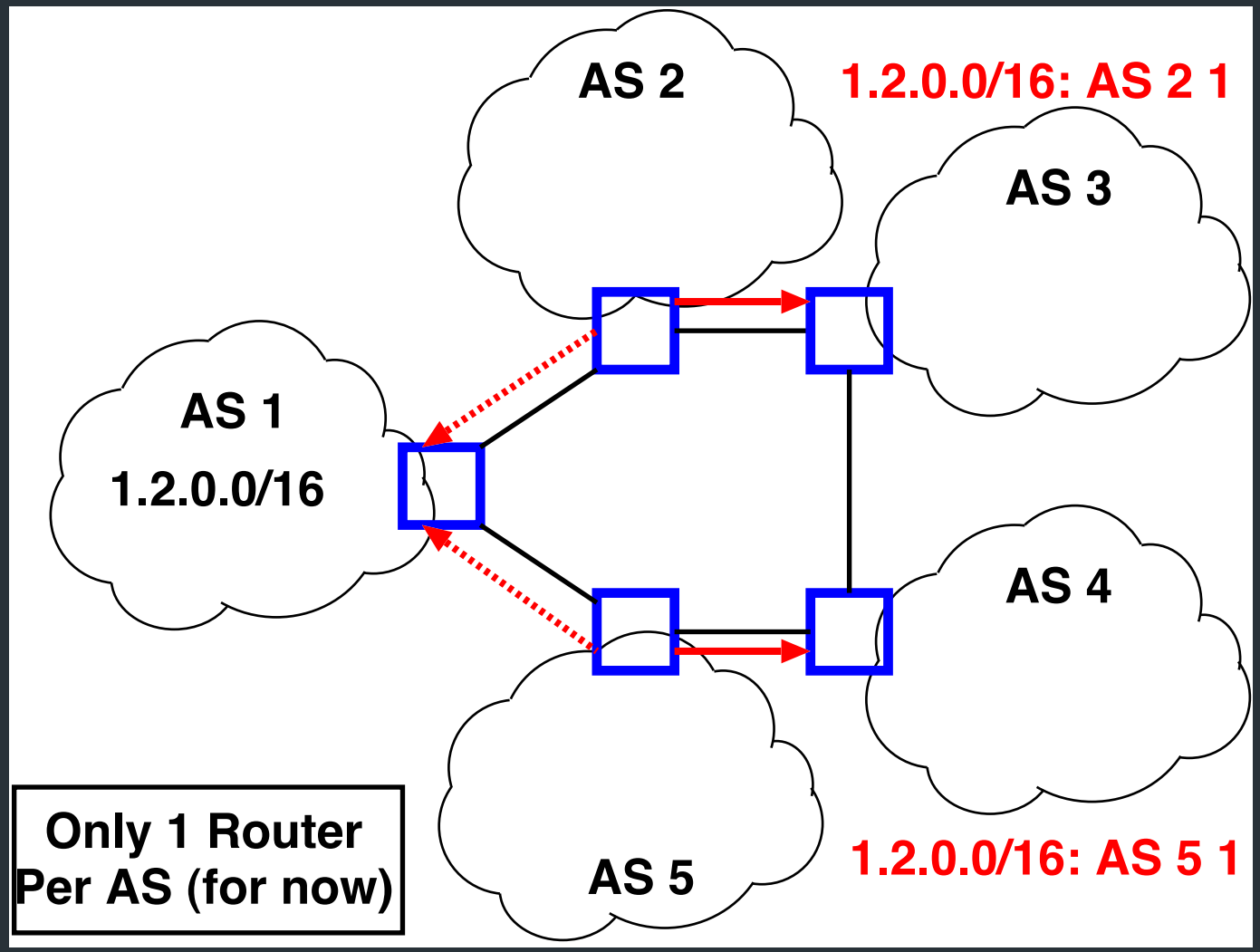
BGP Example



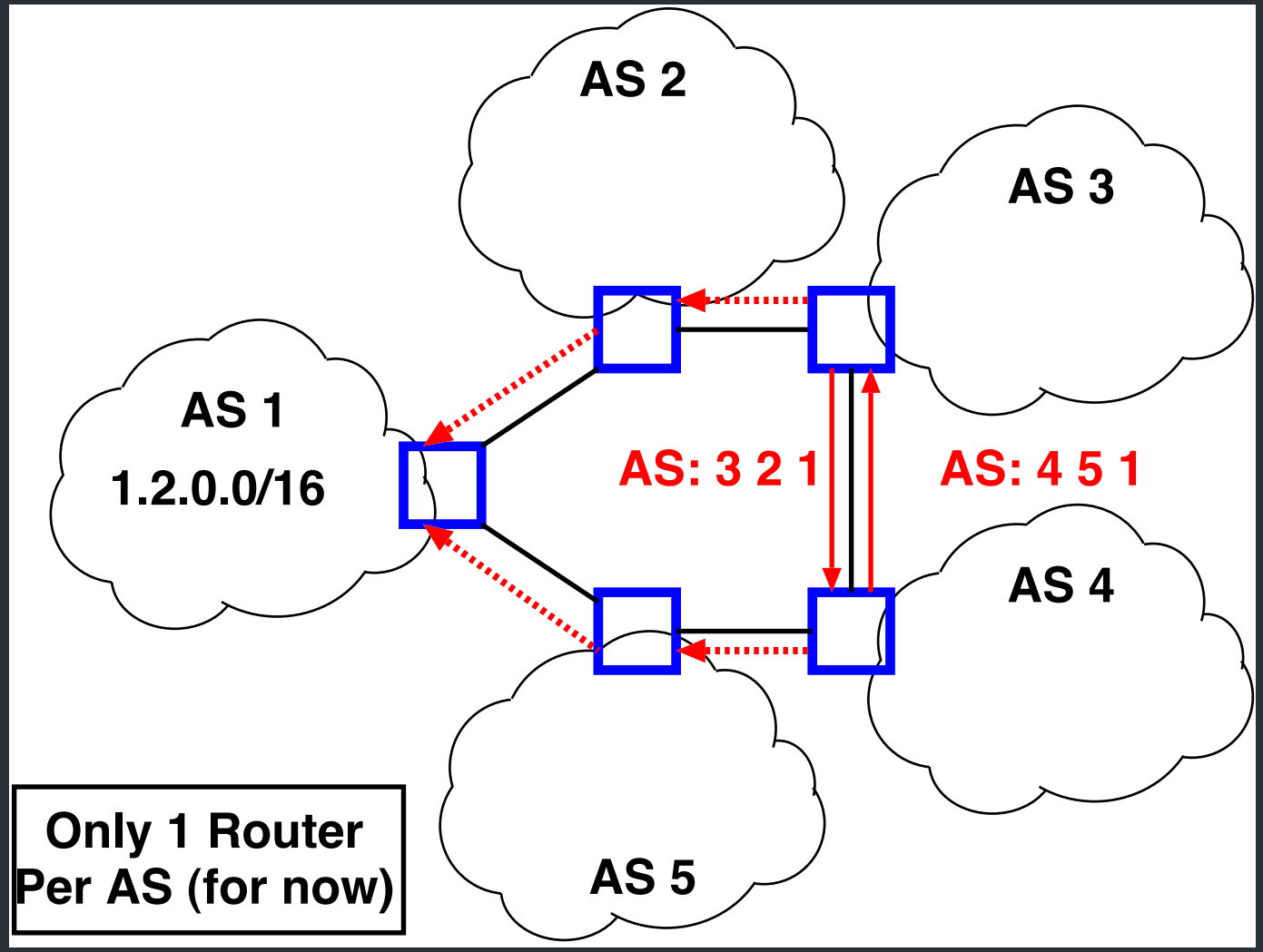
BGP Example


AS 2
AS 3
1.2.0.0/16: AS 1
AS 1
1.2.0.0/16
AS 4
1.2.0.0/16: AS 1
Only 1 Router
Per AS (for now)
AS 5

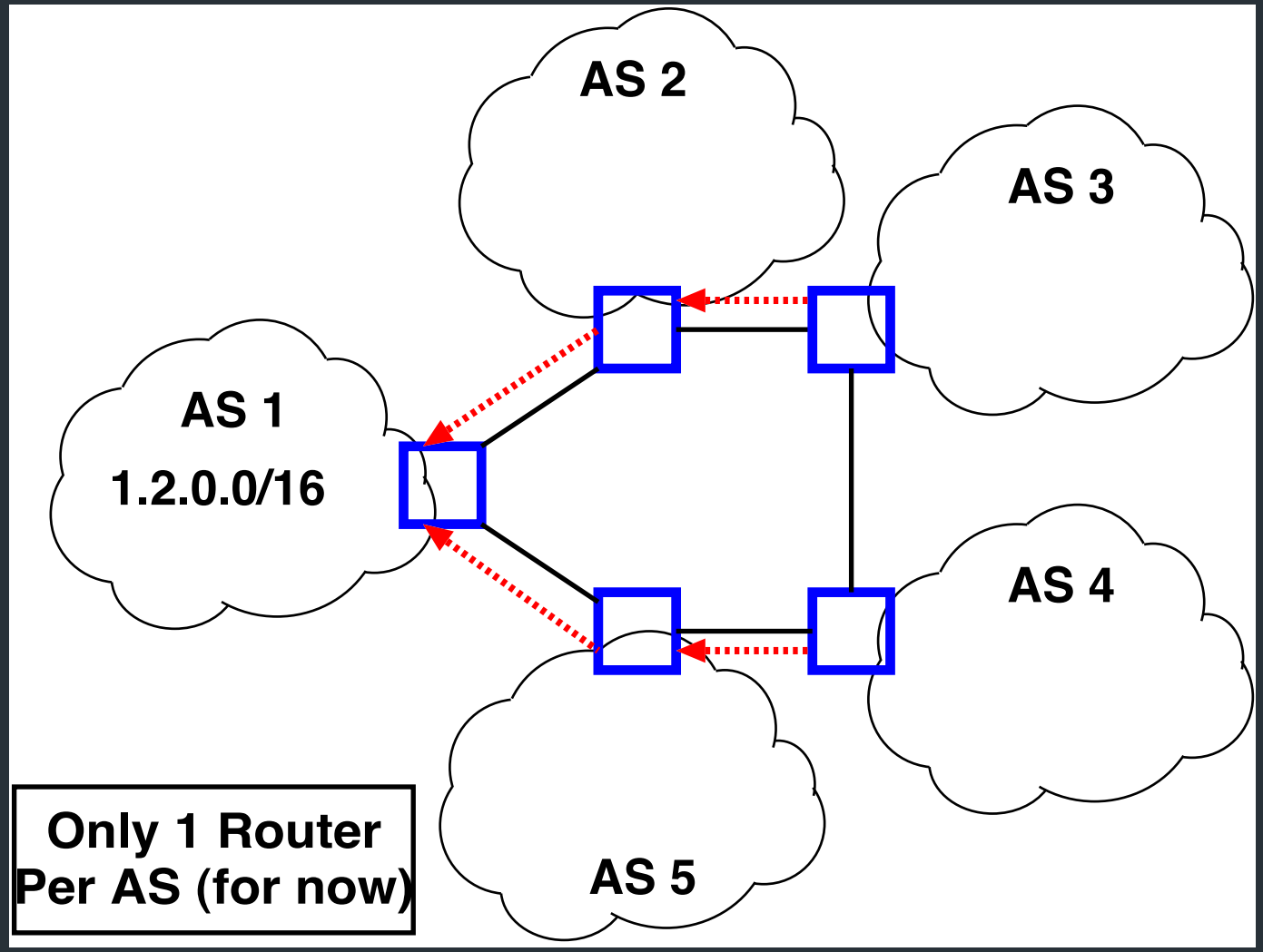
BGP Example


AS 2
1.2.0.0/16: AS 2 1
AS 3
AS 1
1.2.0.0/16
AS 4
AS 5
1.2.0.0/16: AS 5 1
Only 1 Router
Per AS (for now)

BGP Example

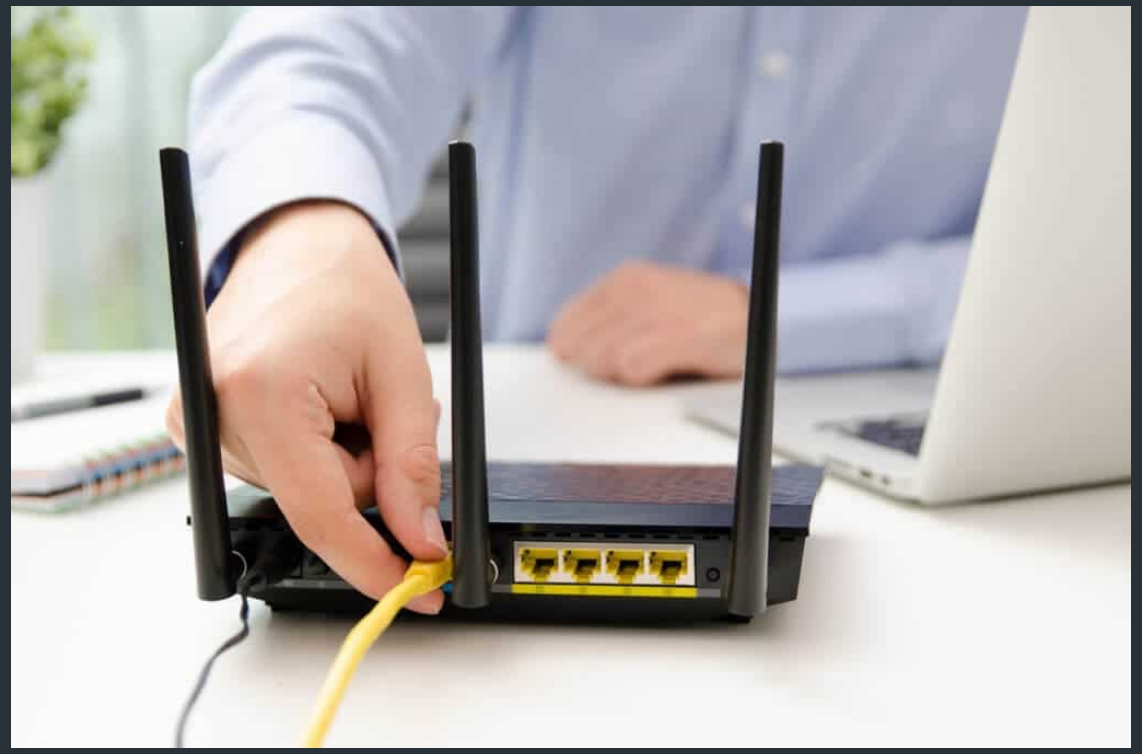

AS 2
AS 3
AS 1
1.2.0.0/16
AS: 3 2 1
AS: 4 5 1
AS 4
AS 5
Only 1 Router
Per AS (for now)

BGP Example



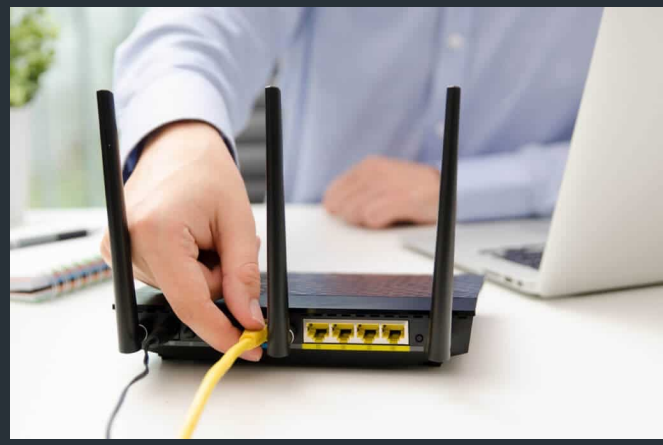
Demo: AS11078

Extra content



… or does it?

Where it gets weird…

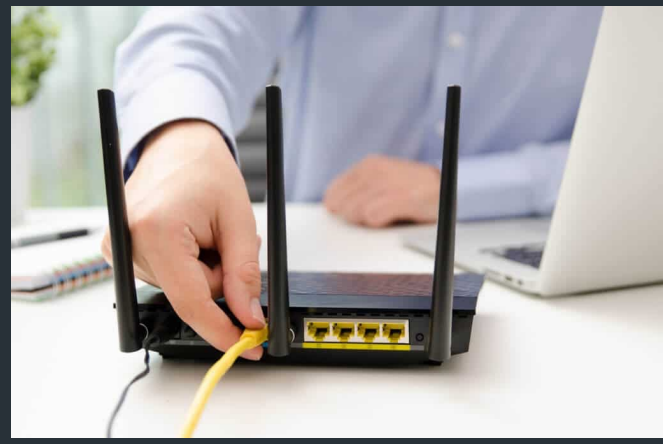


For many end hosts:

(IP assigned to your host) != (Your "public" IP)

(IP seen by other systems on the Internet)

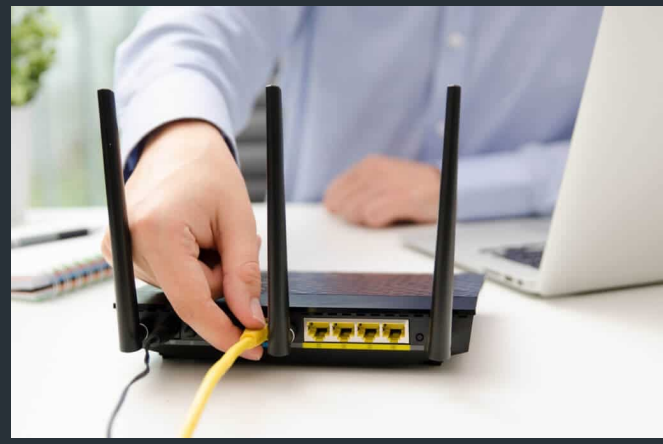
Where it gets weird…



You get just one IP from your ISP…

=> Need to share IP among many devices on the same network!

Where it gets weird…



You get just one IP from your ISP…

=> Need to share IP among many devices on the same network!

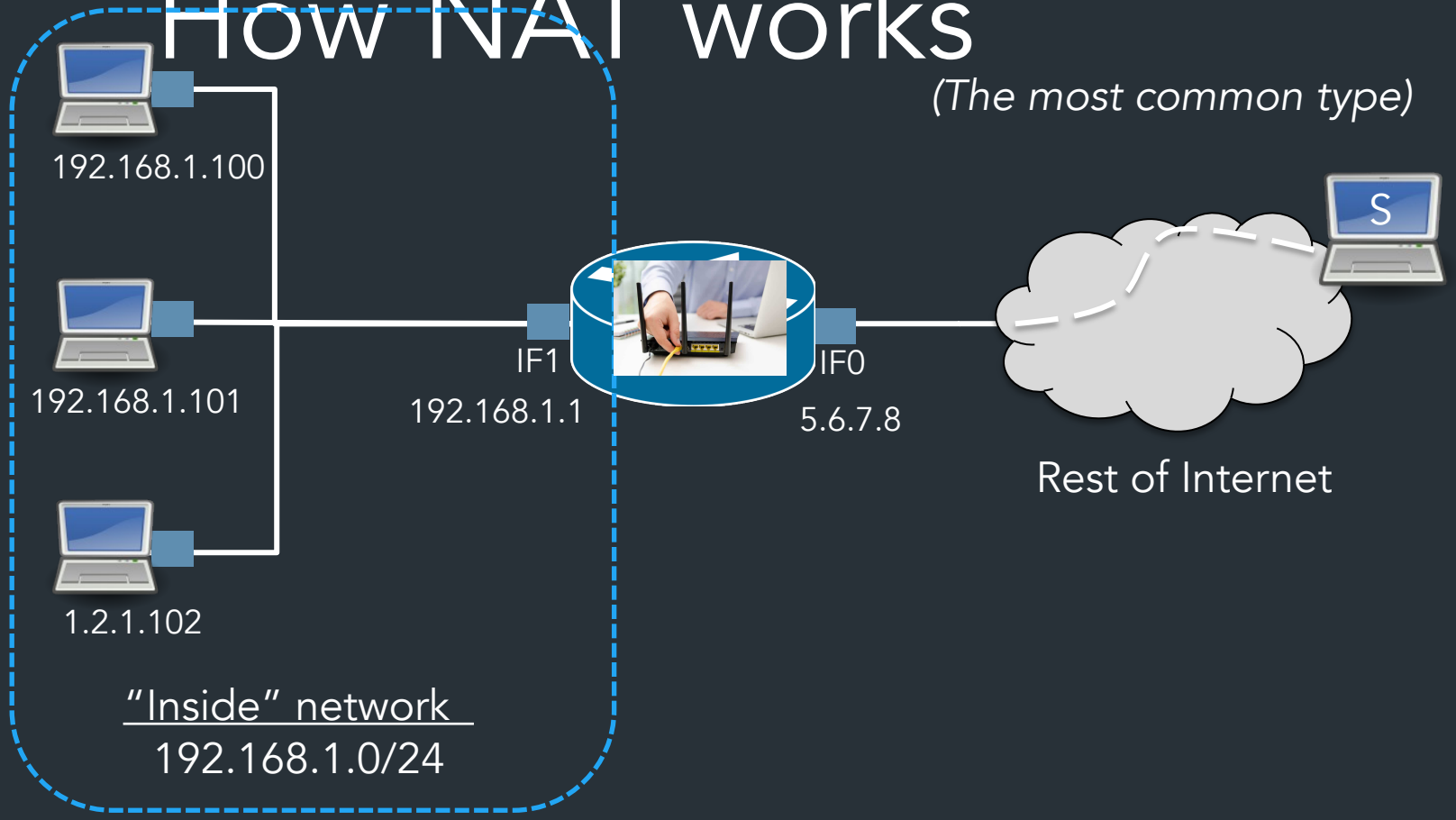
Solution: Create a "private" IP range used within local network

=> Routers need to do extra work to share public IP among many private IPs

=> Network Address Translation (NAT)
(A form of connection multiplexing)

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*

Private IPs (RFC1918)

IP ranges reserved for “private” networks:

Prefix	Use
127.0.0.0/8	“Loopback” address—always for current host
10.0.0.0/8	Reserved for private internal networks (RFC1918)
192.168.0.0/16	
172.16.0.0/12	

Private IPs (RFC1918)

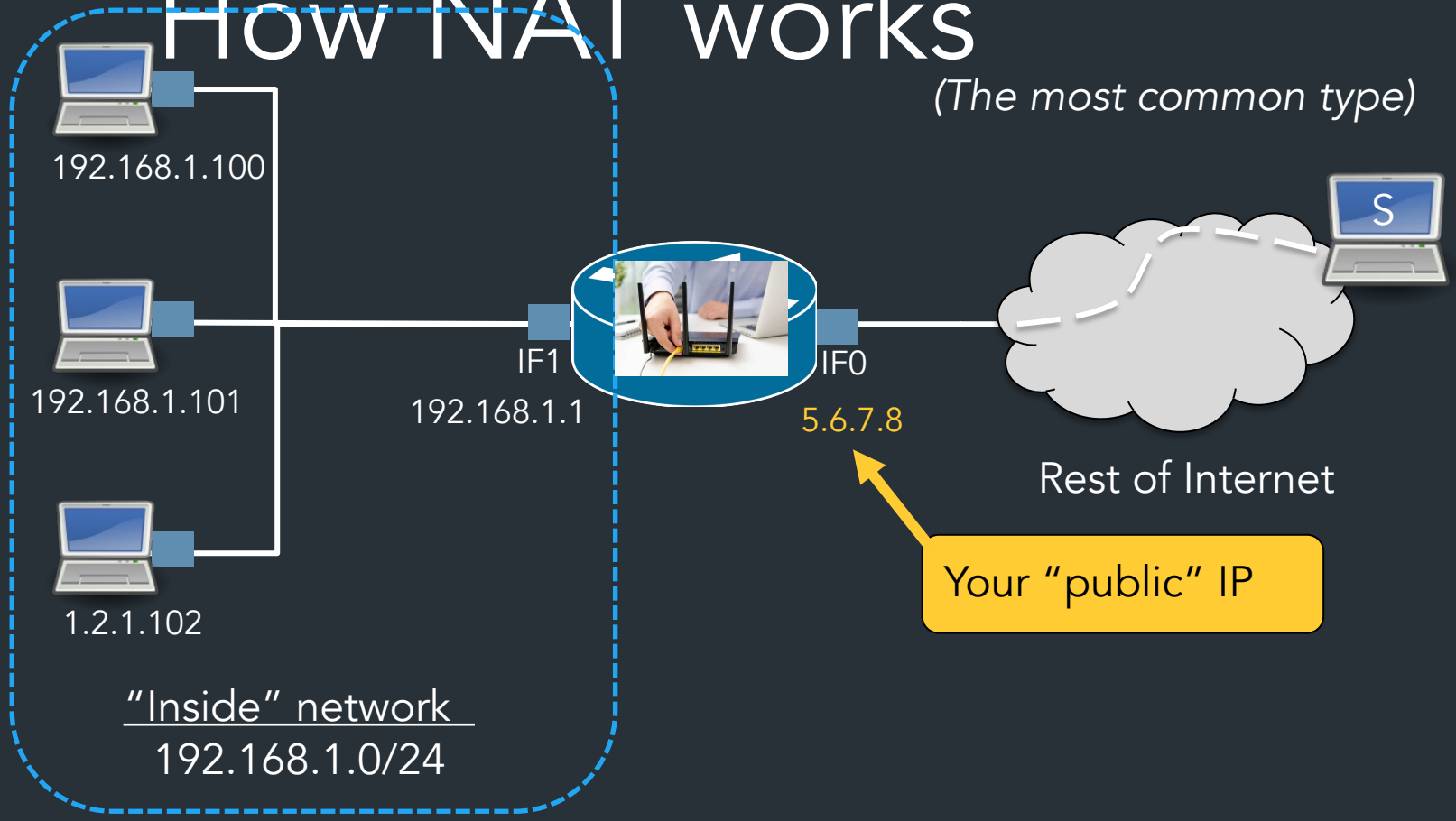
IP ranges reserved for "private" networks:

Prefix	Use
127.0.0.0/8	"Loopback" address—always for current host
10.0.0.0/8	Reserved for private internal networks (RFC1918)
192.168.0.0/16	
172.16.0.0/12	

- Many networks will use these blocks internally
- These IPs should never be routed over the Internet!
  - What would happen if they were?

How NAT works

(The most common type)



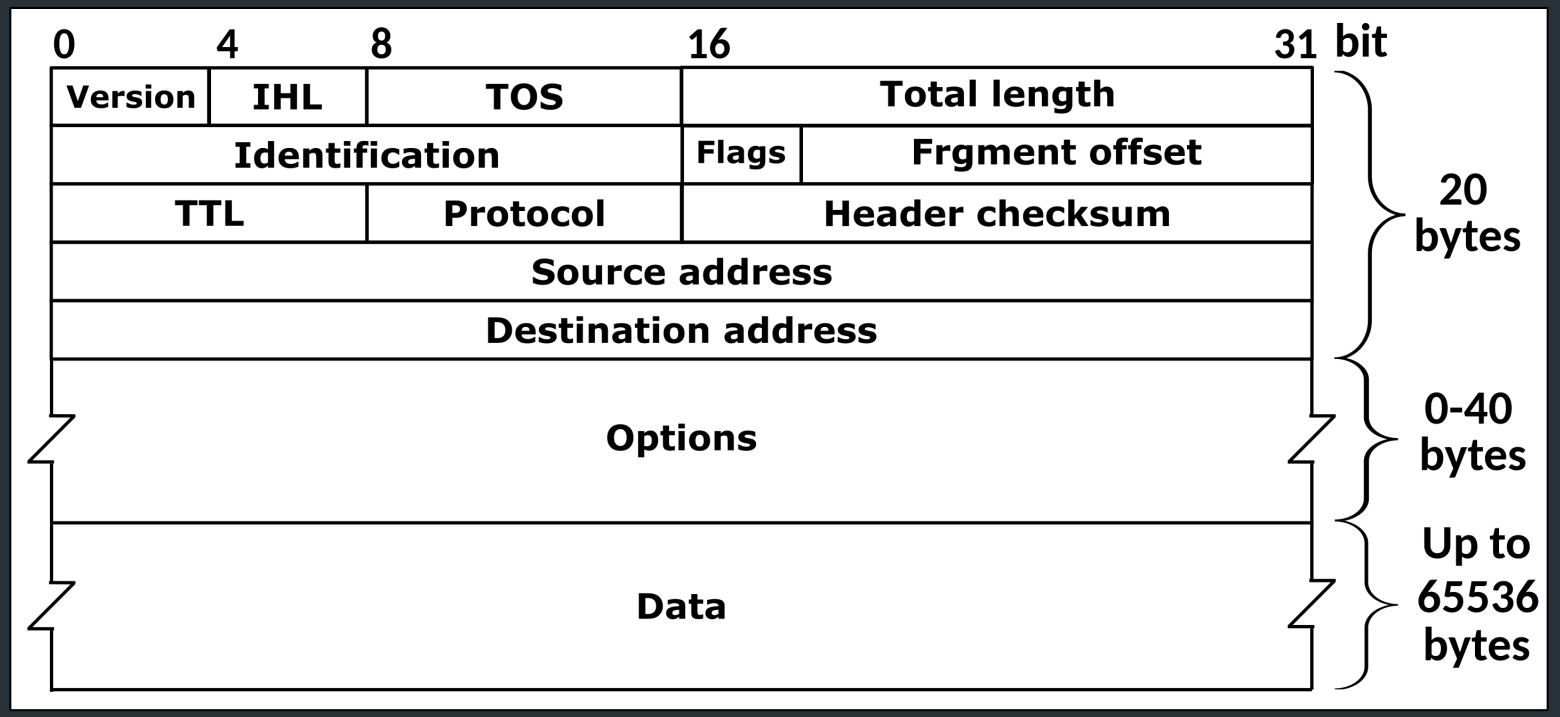
Goal: Share one IP among many hosts on a private network

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

=> Router needs to remember connection state

=> Router makes some (sketchy) assumptions about traffic

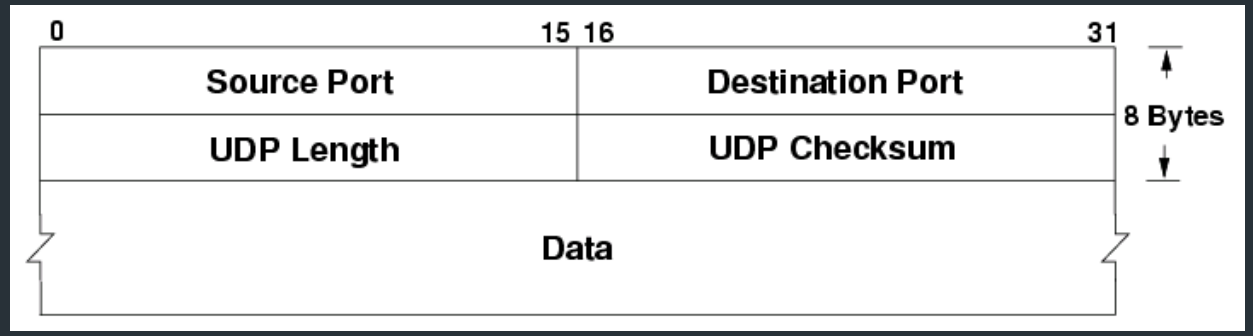
IP Header



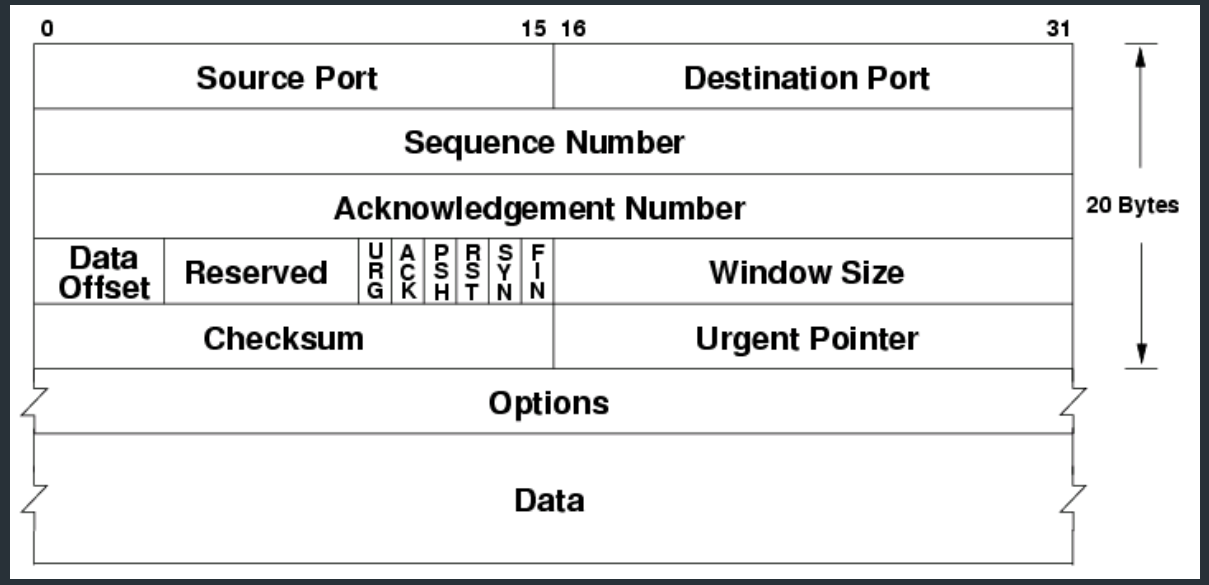
Where are the port numbers?????

… ports are actually part of the transport layer header!

UDP



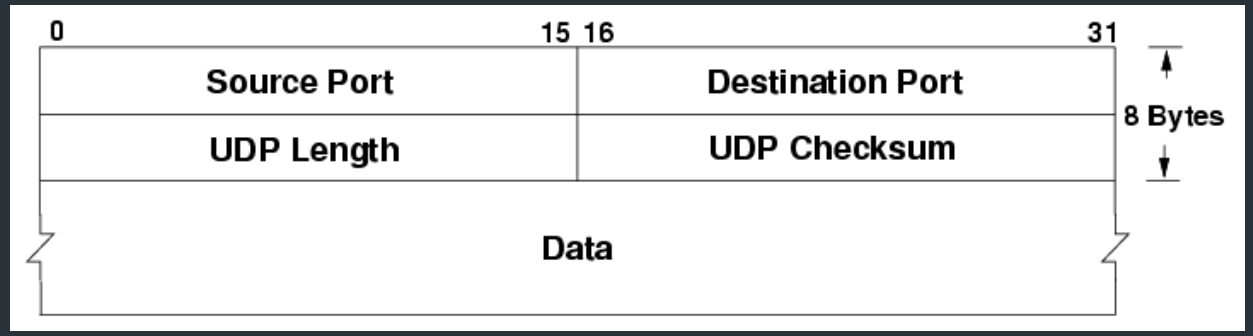
TCP



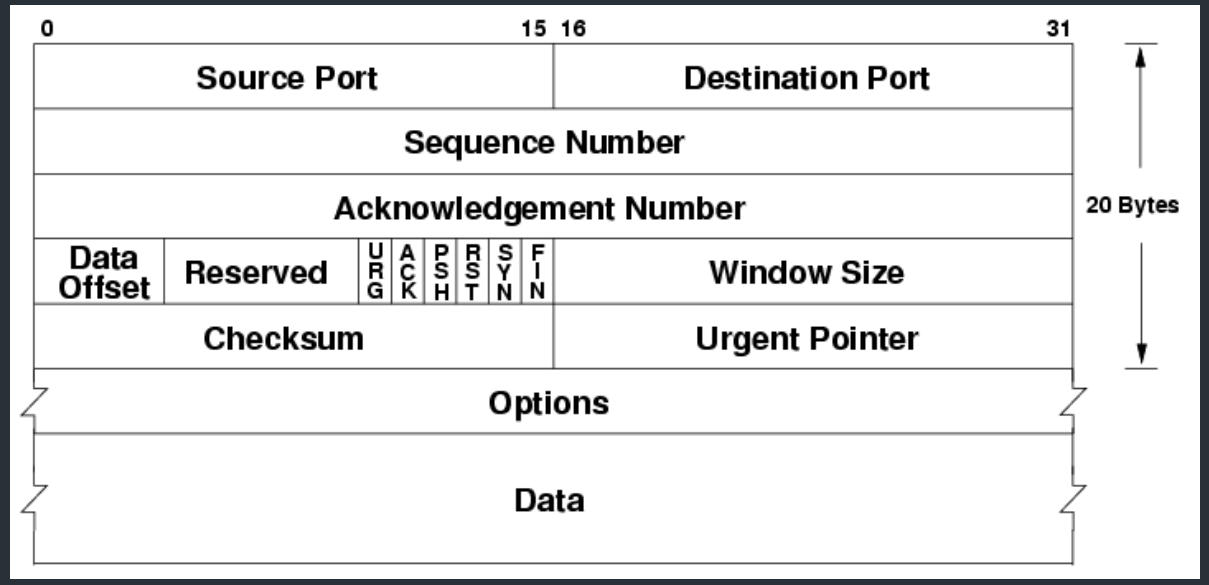
Problem???????

… ports are actually part of the transport layer header!

UDP



TCP



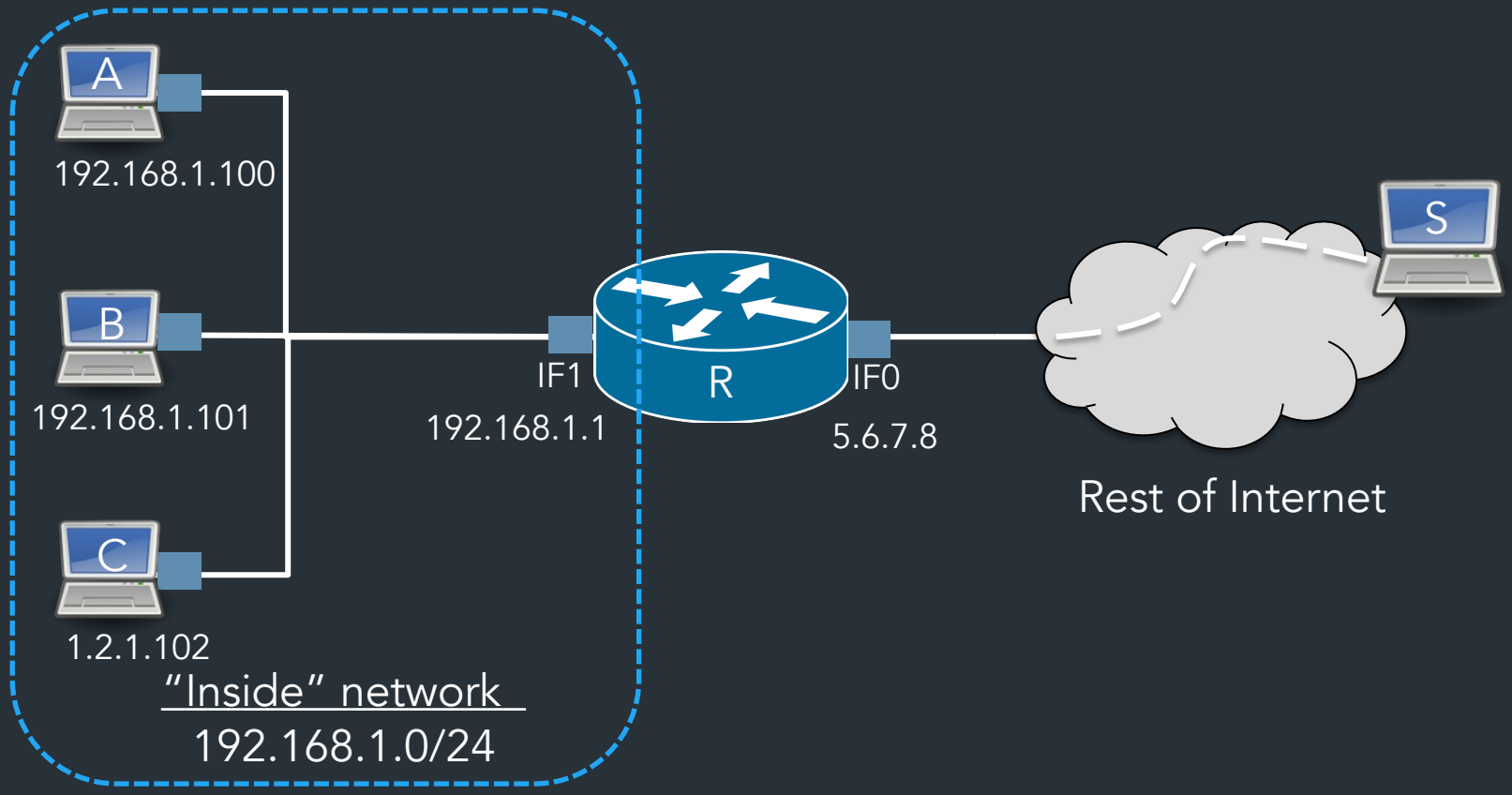
Problem?

⇒ Technically a violation of layering! Network layer shouldn't care about port numbers, but here it matters

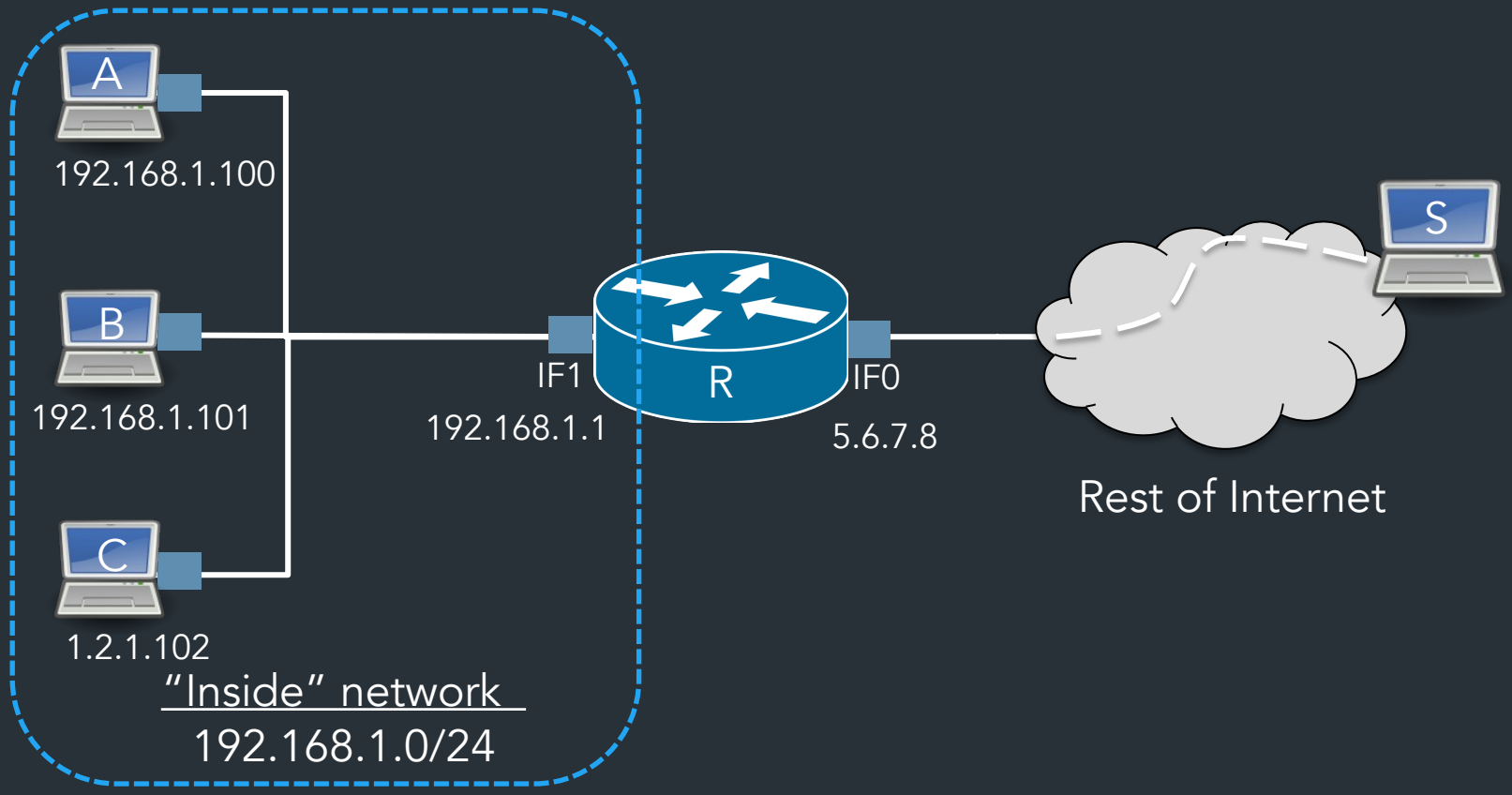
⇒ NAT needs to know semantics of TCP/UDP (how connections start/end…

…but wait there's more…

NAT vs. Snowcast



What happens when outside host S wants to connect to inside host A?

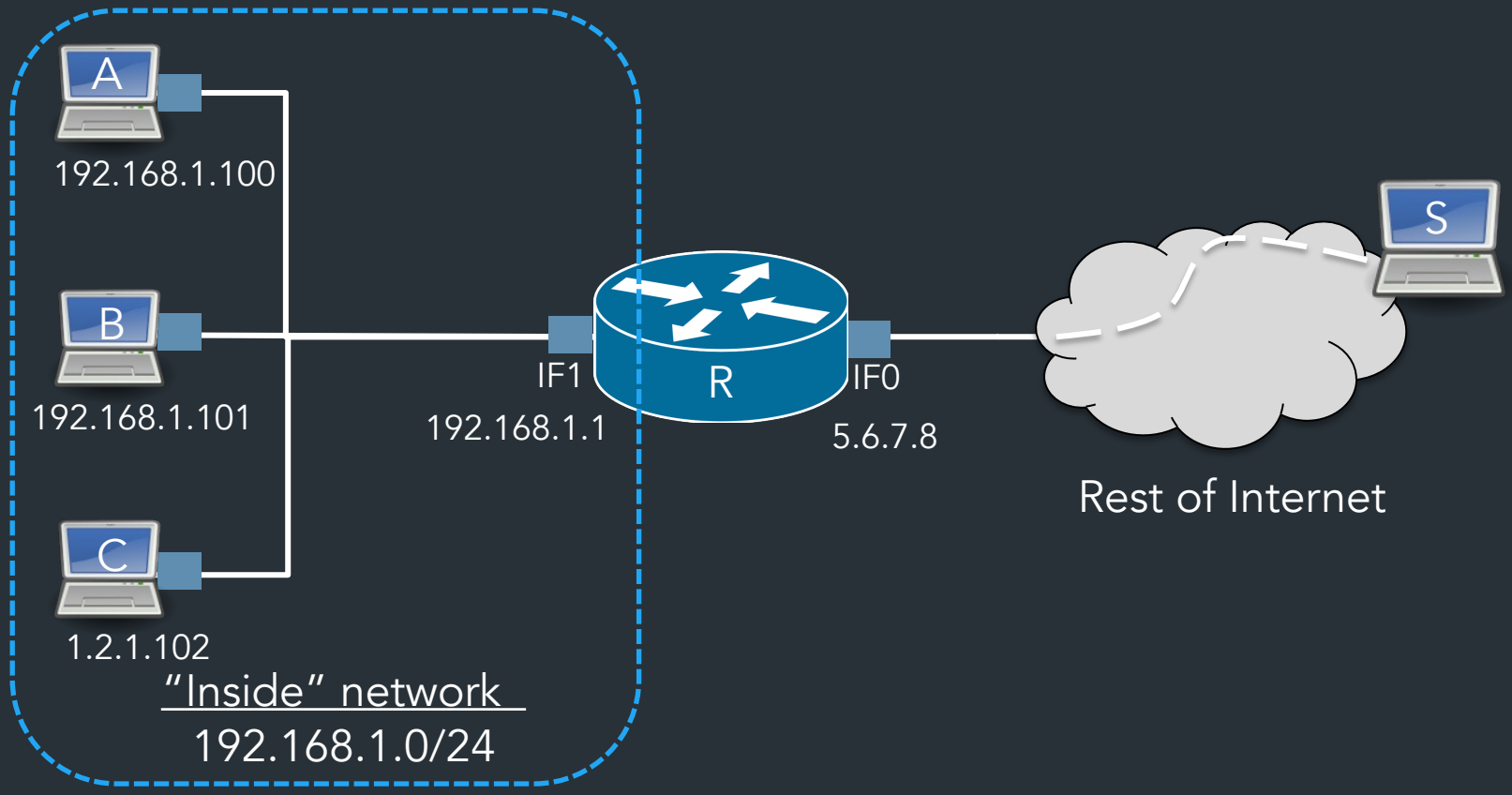


What happens when outside host S wants to connect to inside host A?

Can't do it (at least without special setup)!

⇒ By default, R only knows how to translate packets for connections originating from INSIDE the network

⇒ Breaks end to end connectivity!!!



What happens when outside host S wants to connect to inside host A?

Can't do it (at least without special setup)!

⇒ By default, R only knows how to translate packets for connections originating from INSIDE the network

⇒ Breaks end to end connectivity!!!

End to end connectivity, you say?

Why is this bad?

NAT is used in just about every consumer network

- Generally: can’t connect directly to an end host unless it connects to you first

Why is this bad?

NAT is used in just about every consumer network

- Generally: can't connect directly to an end host unless it connects to you first

- Need extra work for any protocols that need a direct connection between hosts

=> When do we need this?

Why is this bad?

NAT is used in just about every consumer network

- Generally: can't connect directly to an end host unless it connects to you first

- Need extra work for any protocols that need a direct connection between hosts

⇒ Protocols that aren't strictly client-server
⇒ Latency critical applications: voice/video calls, games

NAT Traversal

Various methods, depending on the type of NAT

Examples:

- Manual method: port forwarding
- ICE: Interactive Connectivity Establishment (RFC8445)
- STUN: Session Traversal Utilities for NAT (RFC5389)

One idea: connect to external server via UDP, it tells you the address/port