

CSCI-1680

Network Layer:
Intra-domain Routing

Nick DeMarinis

Administrivia

- IP milestone meetings: Should meet with staff on/before October 4 (tomorrow)
 - Sign up link via email
 - Can't find a time? Make a private post on Ed!
- IP Gearup II tonight (10/3) 6-8pm, CIT368
 - Implementation/debugging stuff; bring questions!
- HW1 due tonight; HW2 out after next class

Today

Two things

- NAT
- Intro to routing, RIP

Warmup

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

1)

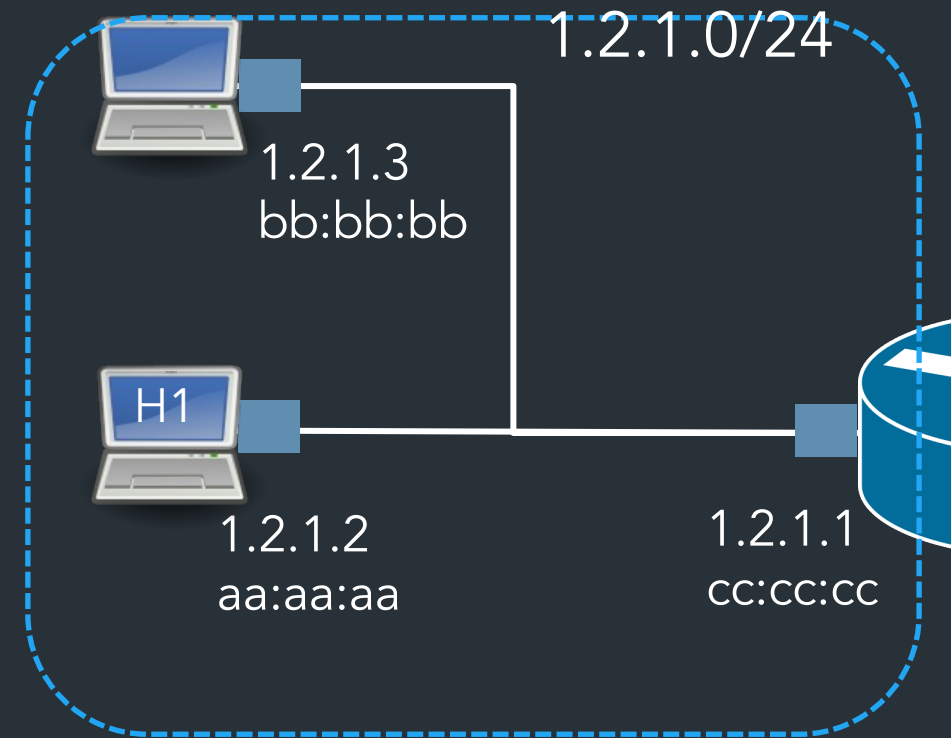
	Src	Dest
Link	aa:aa:aa	???
IP	1.2.1.2	1.2.1.1

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

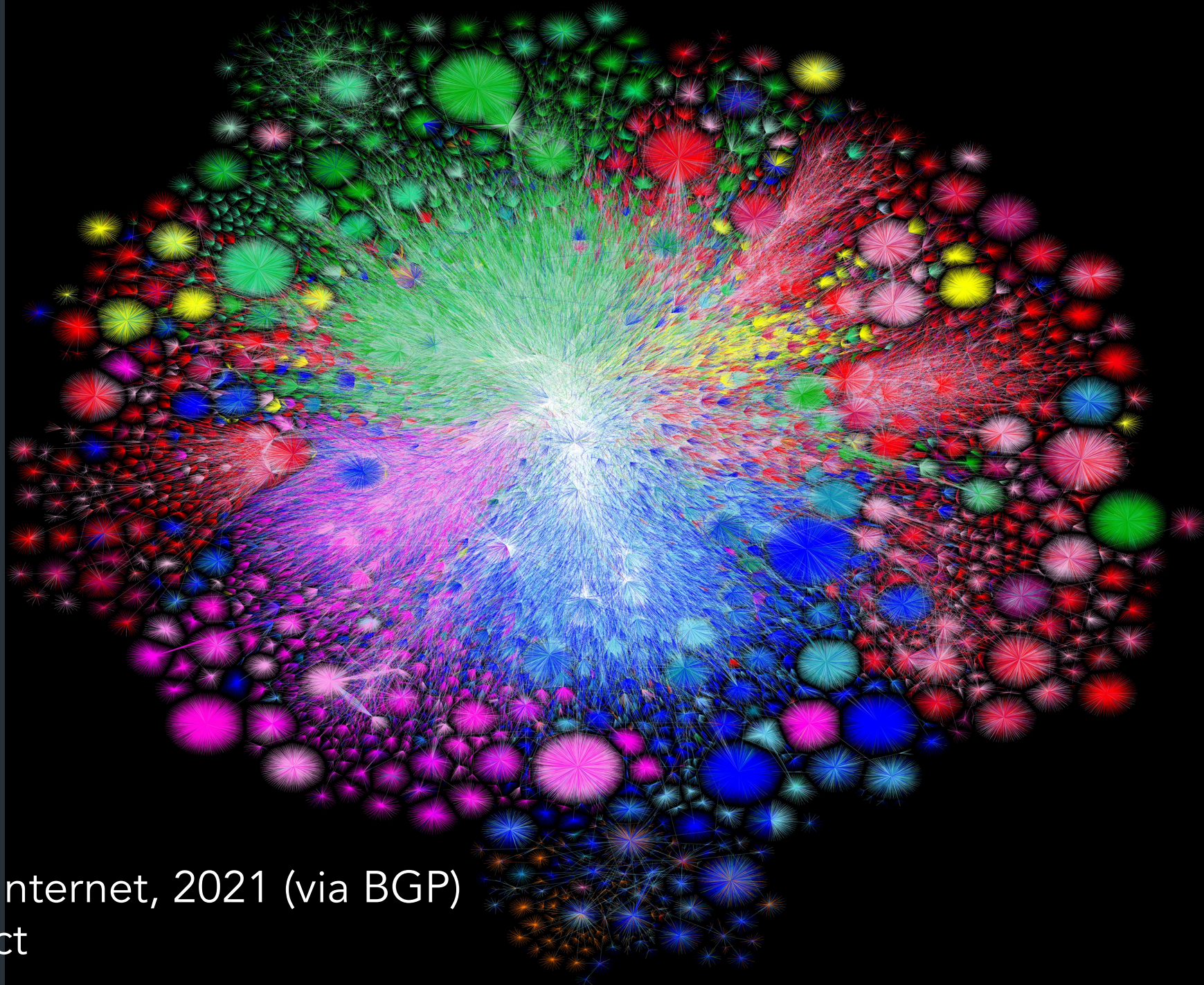
Link-layer header info (Ethernet/Wifi/etc)

- 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)

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

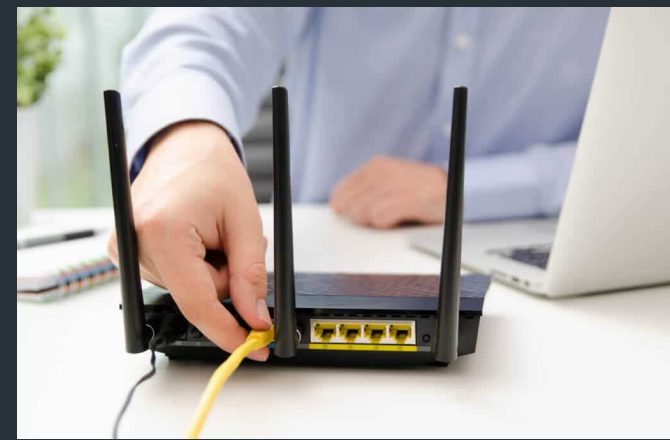


Map of the Internet, 2021 (via BGP)
OPTe project



... or does it?

Where it gets weird...



For many end hosts:

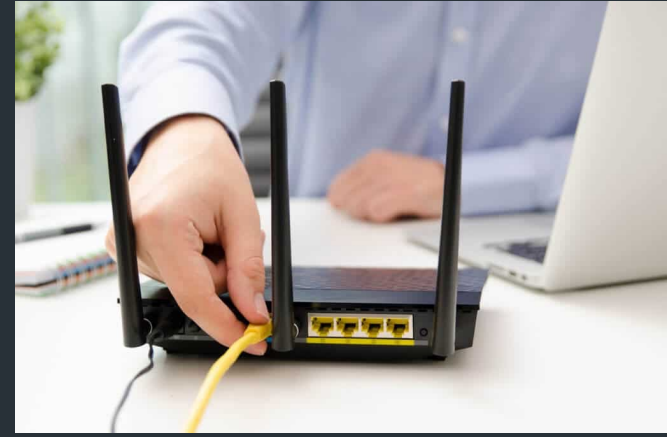
(IP assigned to your host) \neq (Your "public" IP)

(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!

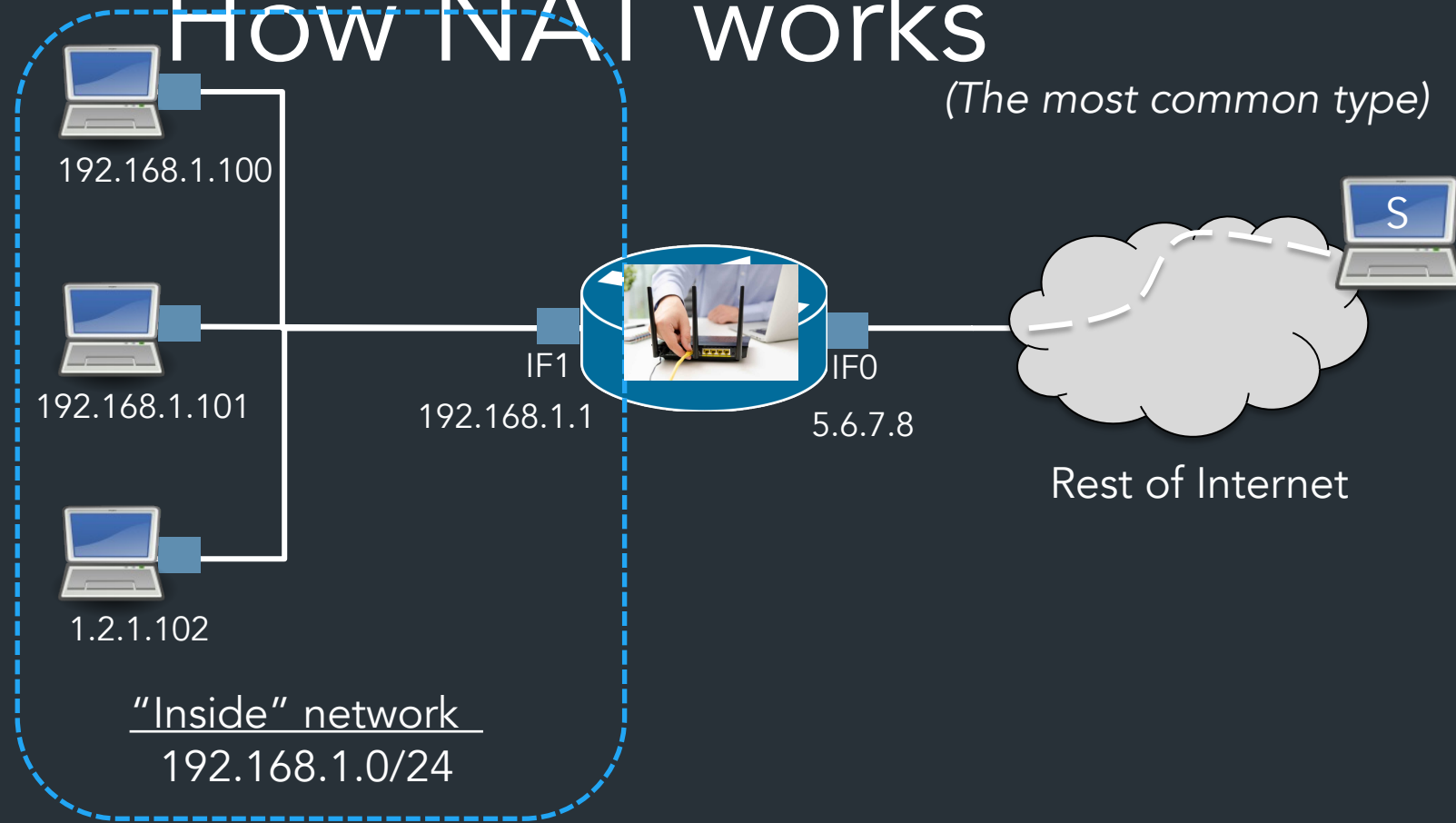


Solution: Create a “private” IP range used within local network

=> 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



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	
192.168.0.0/16	Reserved for private internal networks (RFC1918)
172.16.0.0/12	

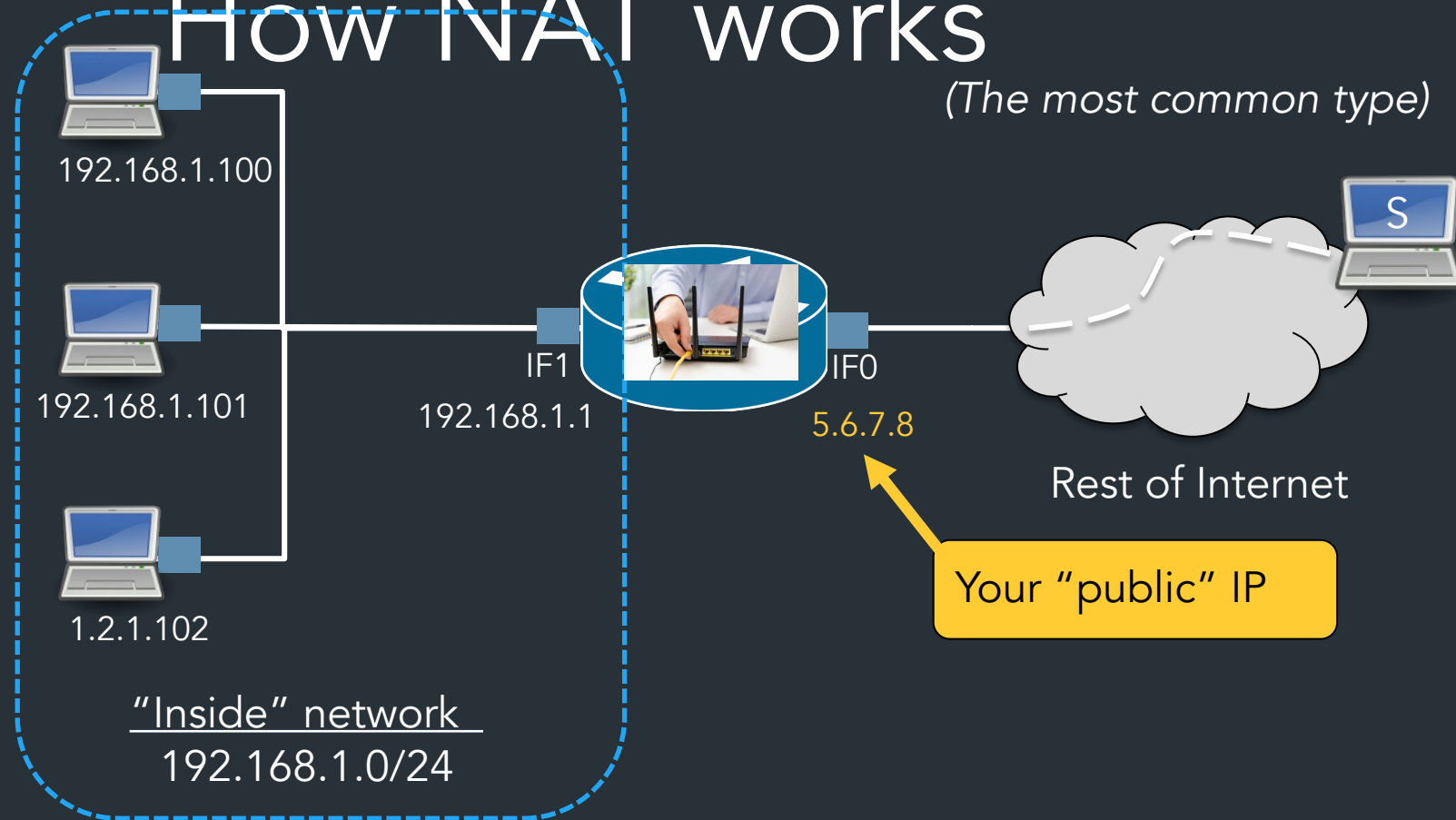
Private IPs (RFC1918)

IP ranges reserved for “private” networks:

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



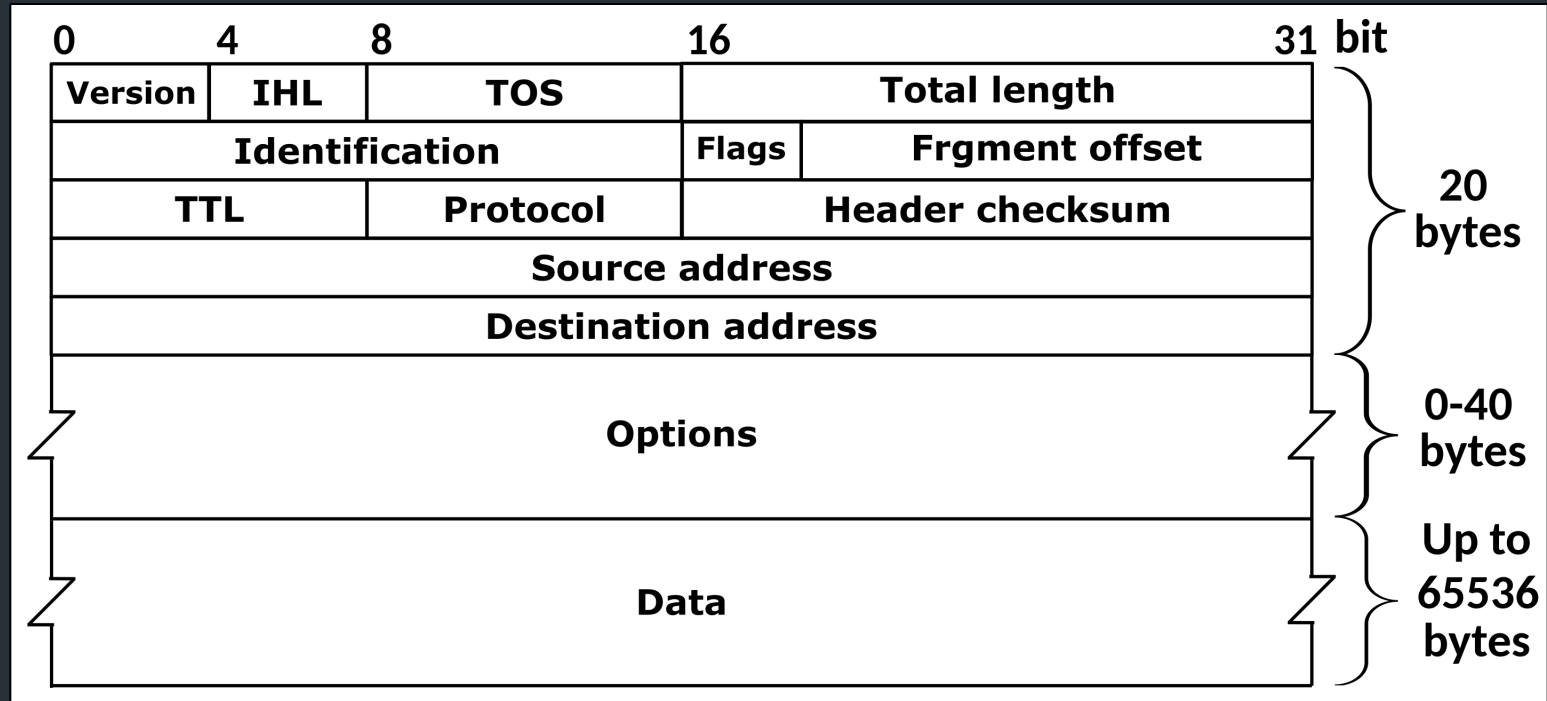
Goal: Share one IP among many hosts on a private network

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

=> Router needs to remember connection state

=> Router makes some (sketchy) assumptions about traffic

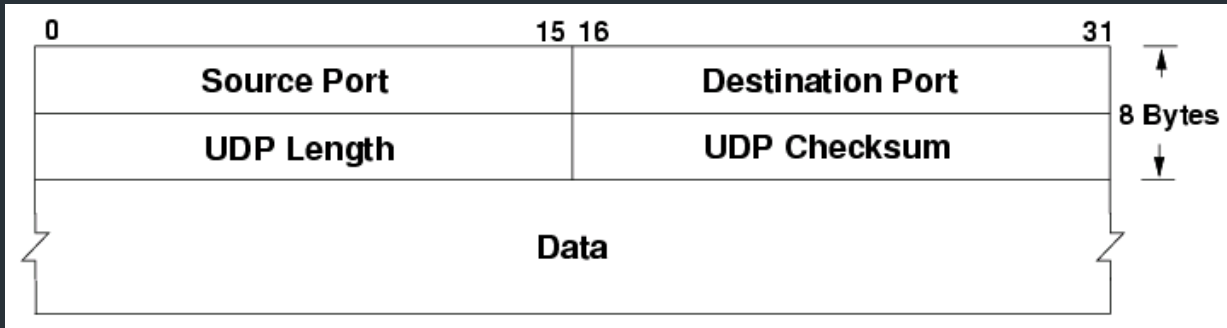
IP Header



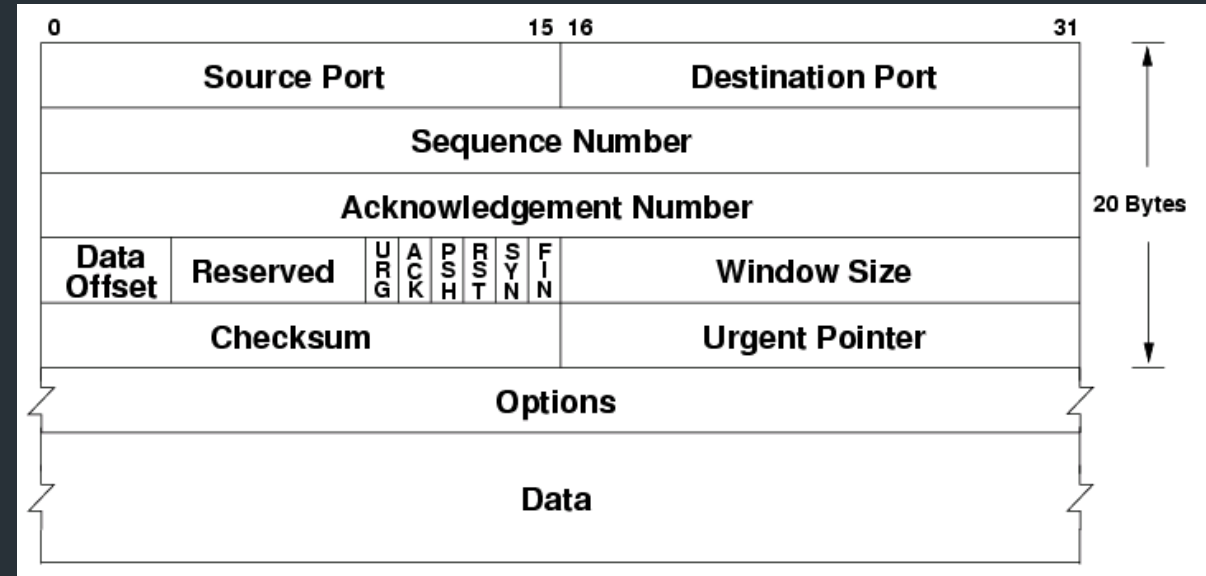
Where are the port numbers?????

... ports are actually part of the transport layer header!

UDP



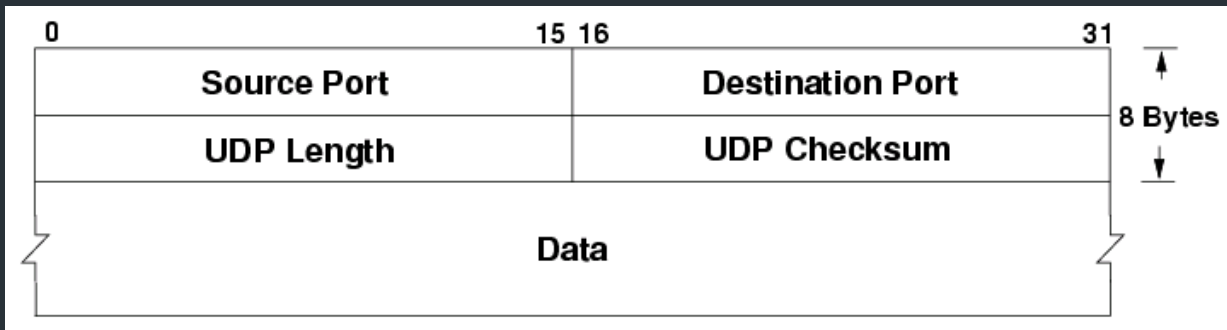
TCP



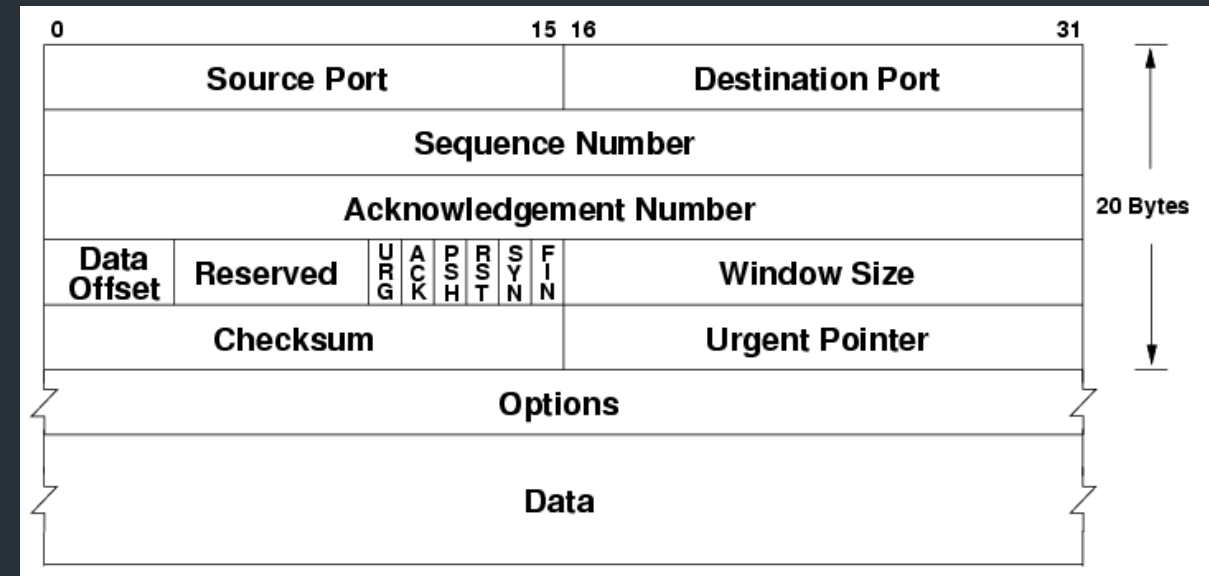
Problem???????

... ports are actually part of the transport layer header!

UDP



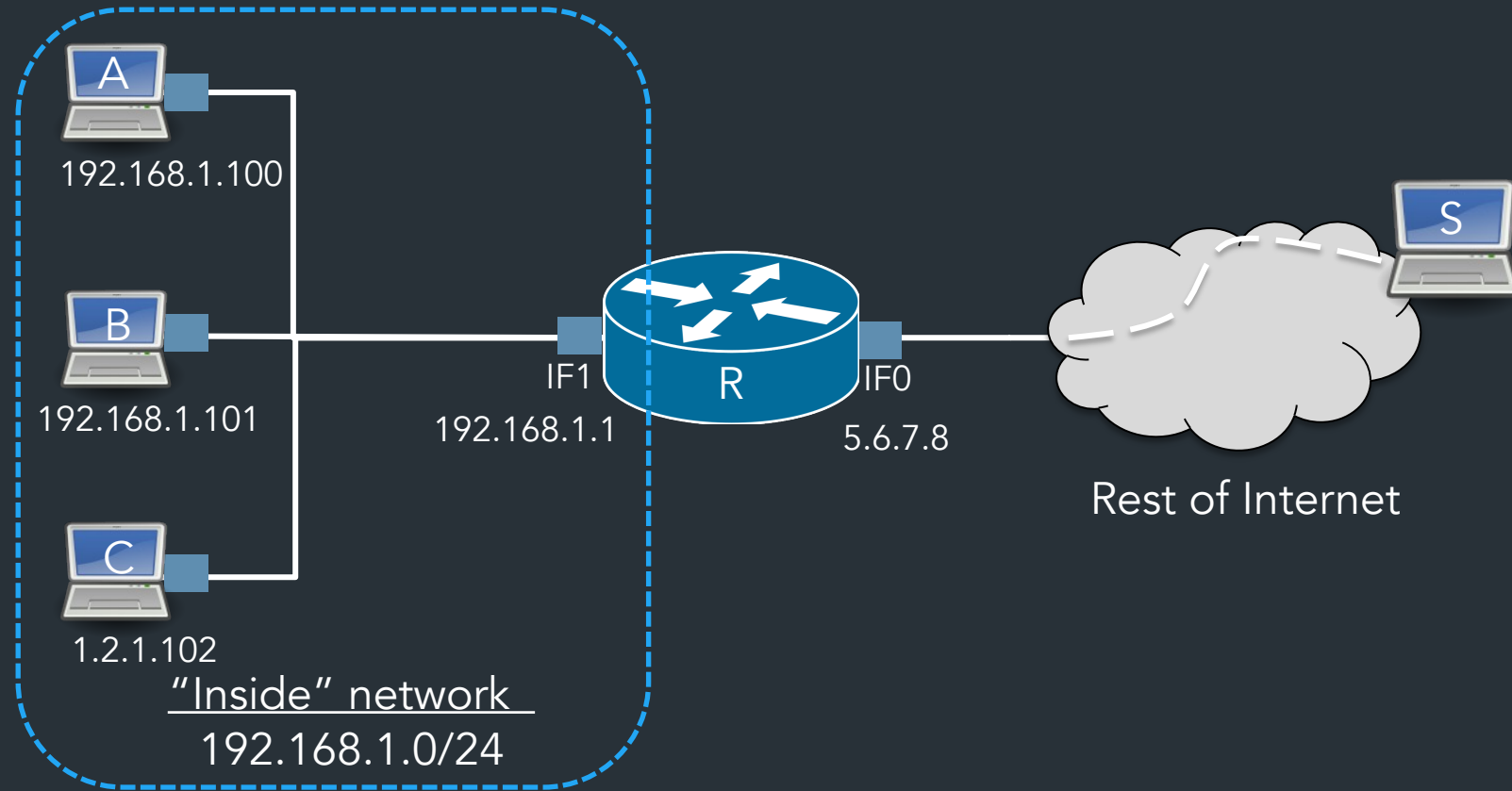
TCP



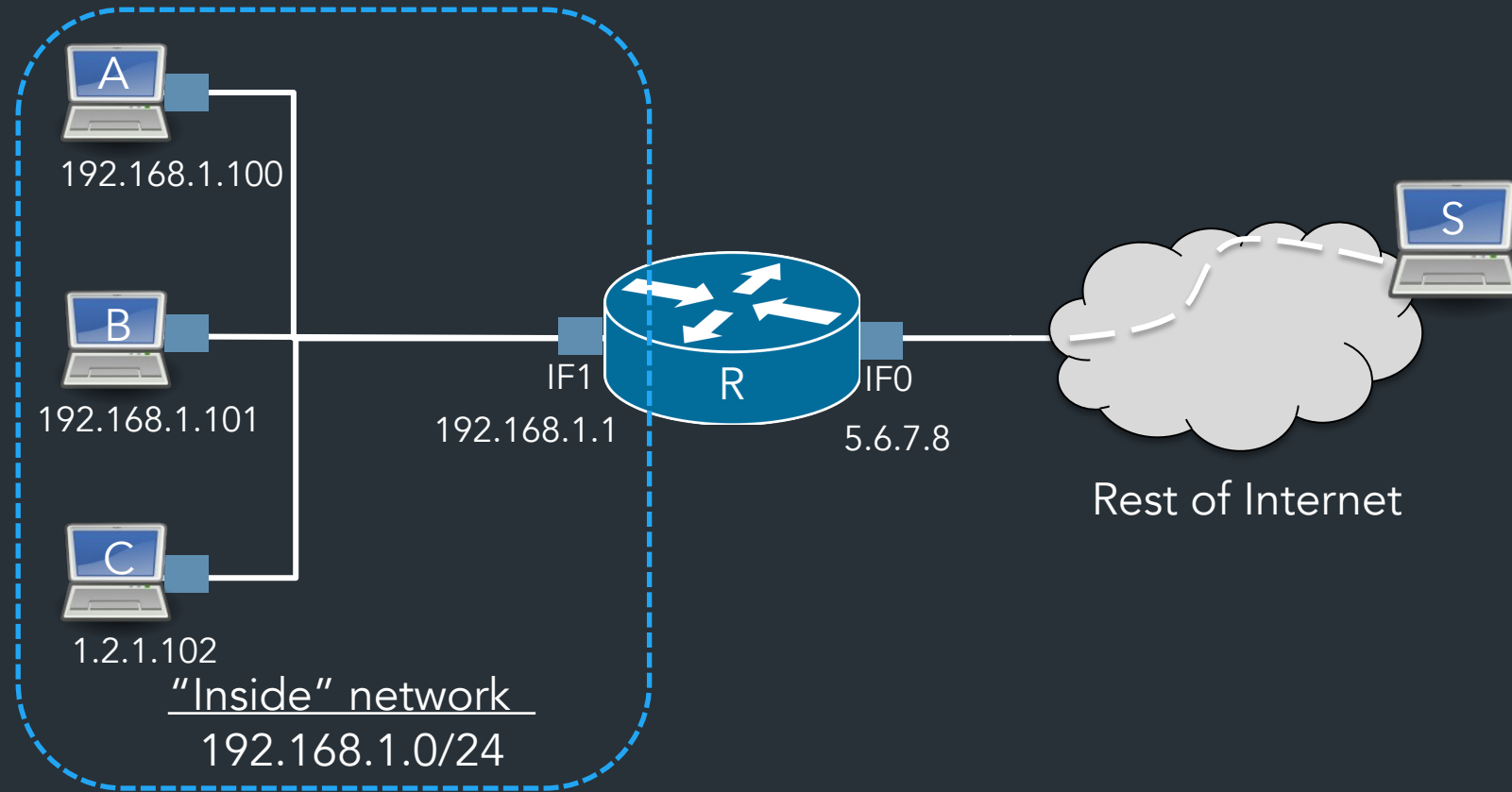
Problem?

- ⇒ 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?

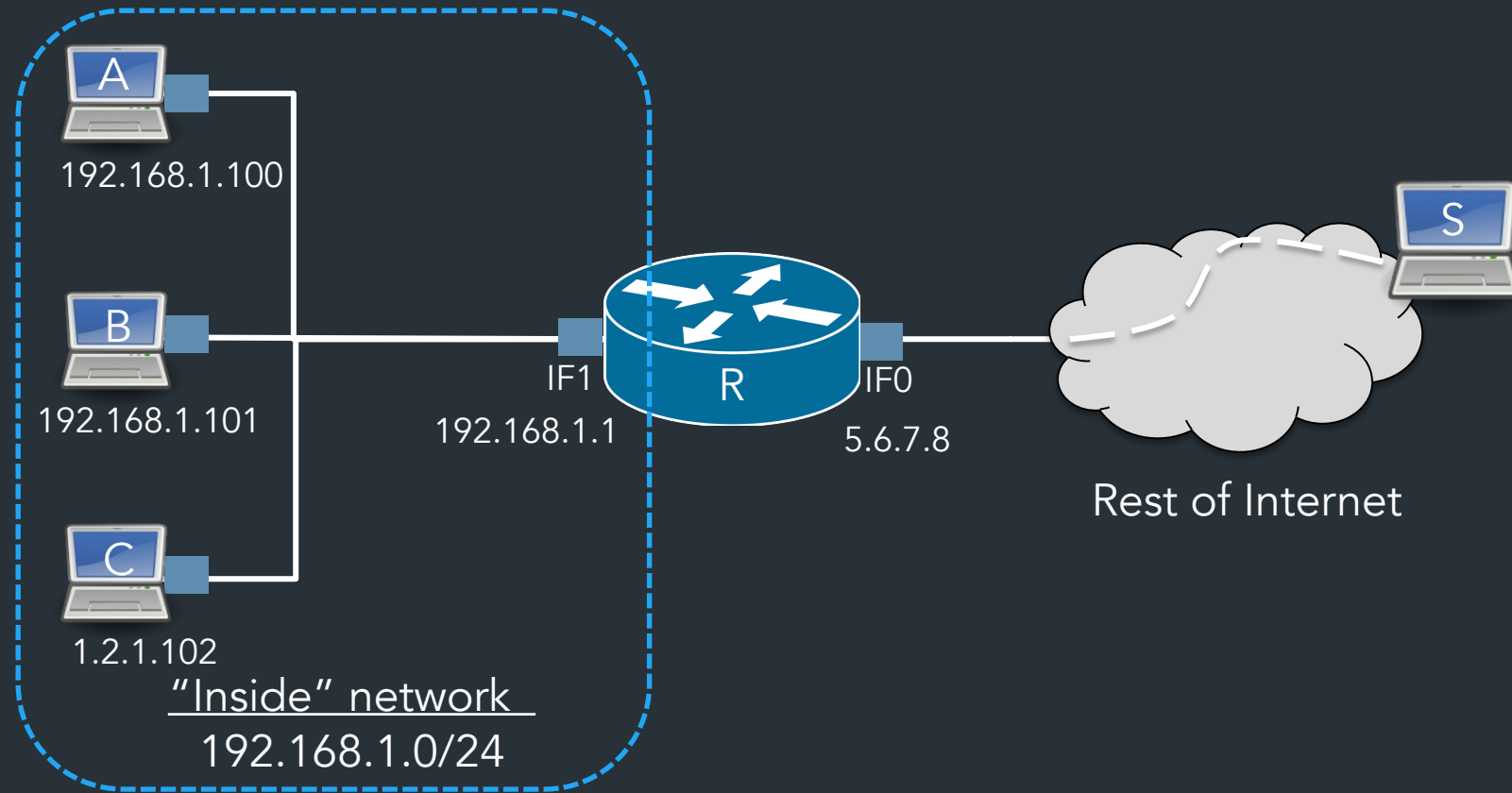


What happens when outside host S wants to connect to inside host A?

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

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

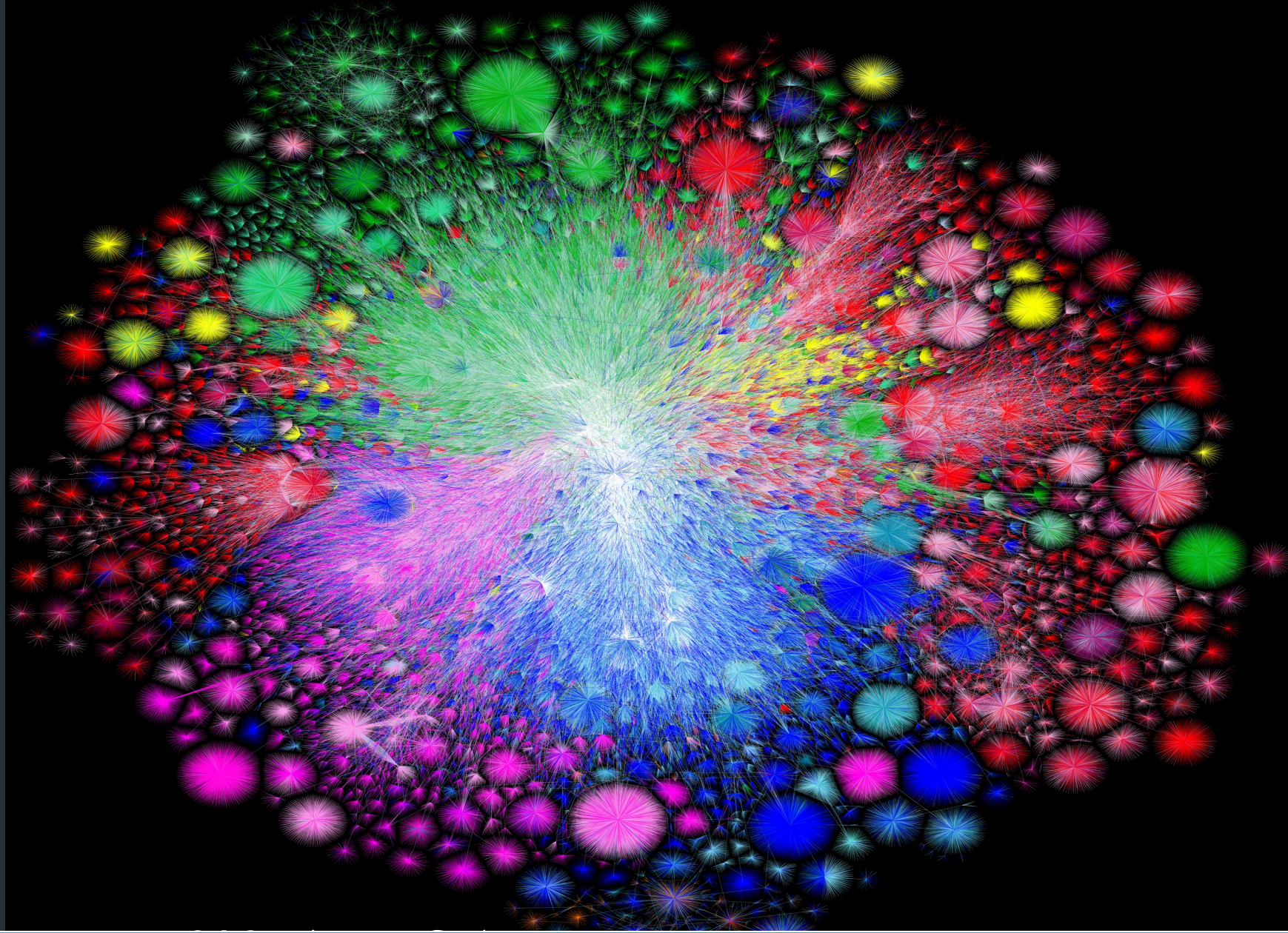
Routing is the process of updating forwarding tables

- 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 the
OPTe project

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
```

Thinking about the scale

At this stage, we think about **routing to *whole networks***, ie, some entity with some set of IP prefixes:

eg. Brown University @ 128.148.0.0/16, 138.16.0.0/16

Thinking about the scale

At this stage, we think about **routing to *whole networks***, ie, some entity with some set of IP prefixes:

eg. 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

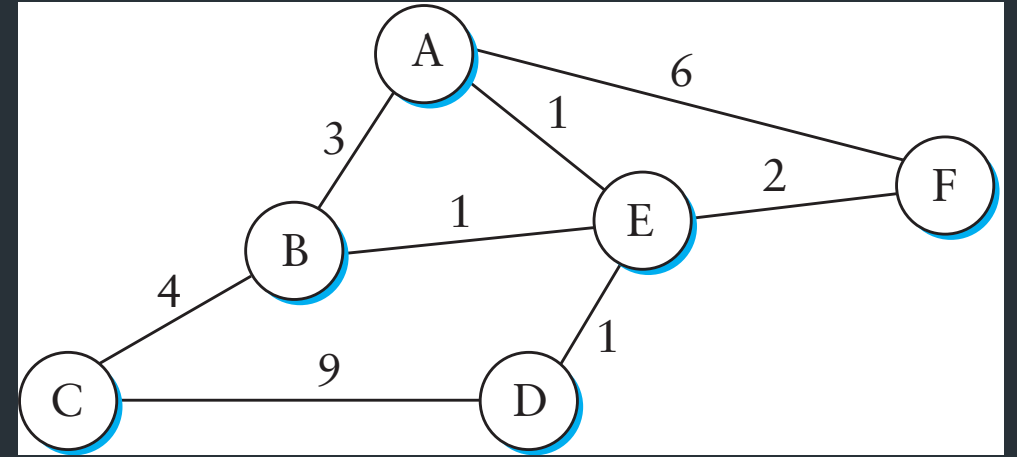
^ We are here today

- Inter-domain (**exterior**) routing: routing between ASes
 - => None of the above, decisions instead made by *policy* (later)

Intra-Domain (Interior) Routing

Typically, view network as a graph

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

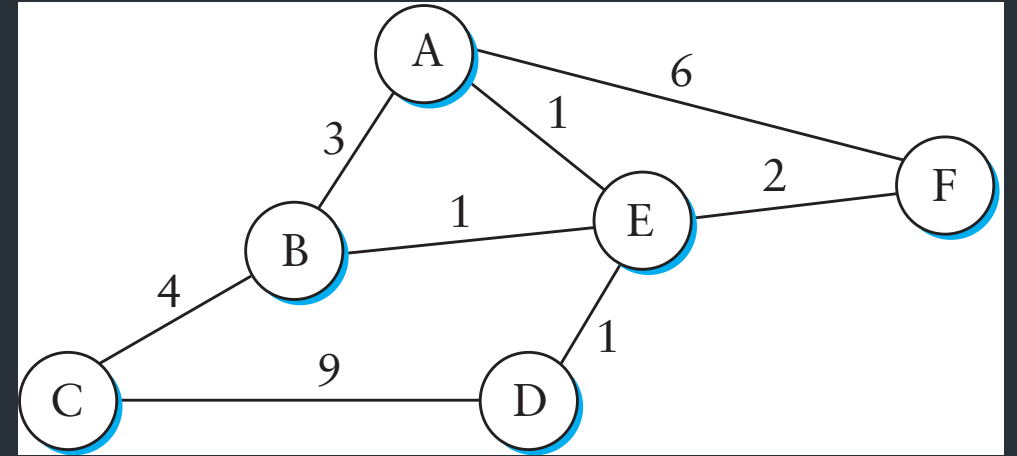


Goal: find lowest-cost path between nodes

- Each node individually computes routes

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 (Dijkstra/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 (Dijkstra/Prim shortest path algorithm)

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

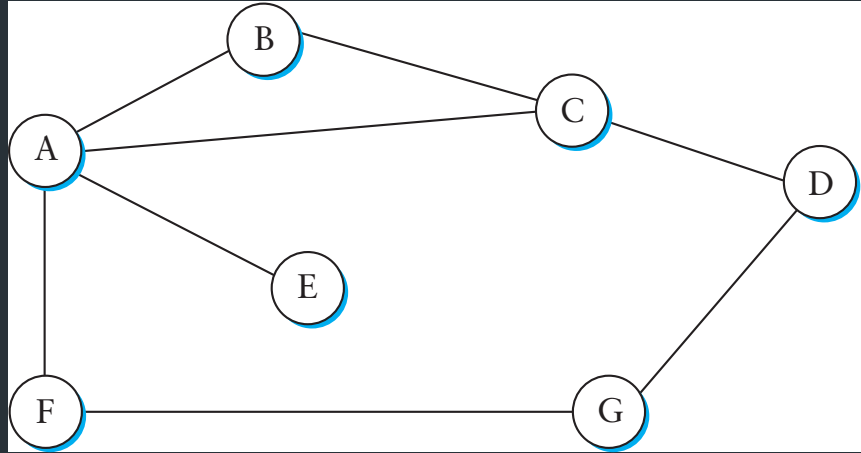
Say router R receives an update $\langle D, c_D \rangle$ from neighbor N at cost C_N

\Rightarrow Know: R can reach D via N with cost $c = c_D + C_N$

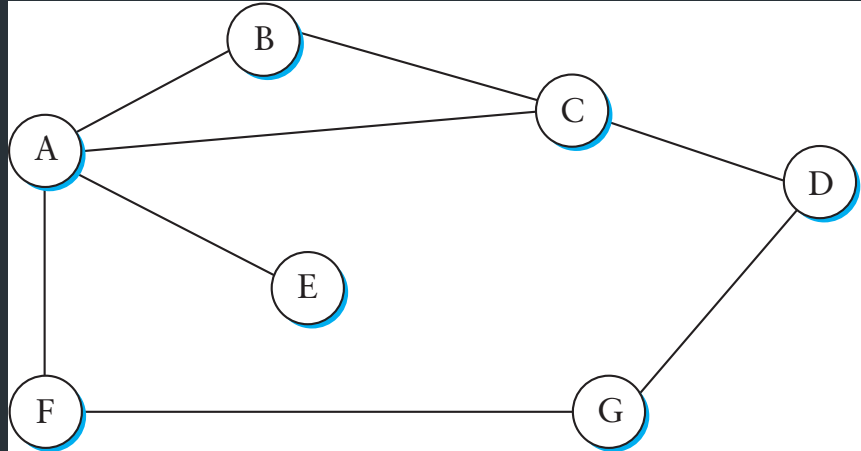
How to update table?

1. If D not in table, add $\langle D, c, N \rangle$ (New route!)
2. If table has entry $\langle D, M, c_{old} \rangle$:
 - if $c < c_{old}$: update table to $\langle D, c, M \rangle$. (Lower cost!)
 - if $c > c_{old}$ and $M == N$: update table to $\langle D, c, N \rangle$ (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

Warmup

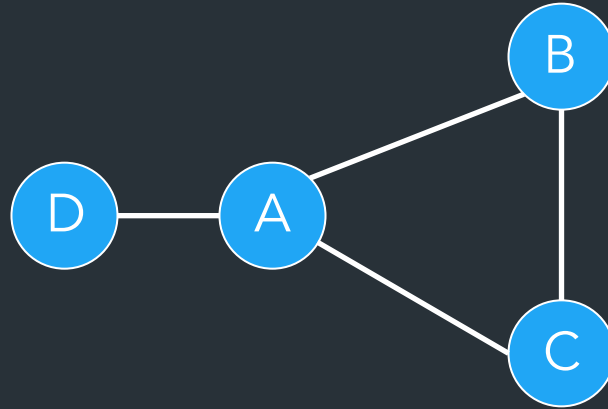
Suppose router R has the following table:

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

Dealing with Failures



- What happens when the D-A link fails?

=> "Count to Infinity" problem

How to avoid loops

- Does IP TTL help?
- Simple approach: consider a small cost n (e.g., 16) to be infinity
 - After n rounds decide node is unavailable
 - But rounds can be long, this takes time

Problem: distance vector based only on local information

One way: Split Horizon

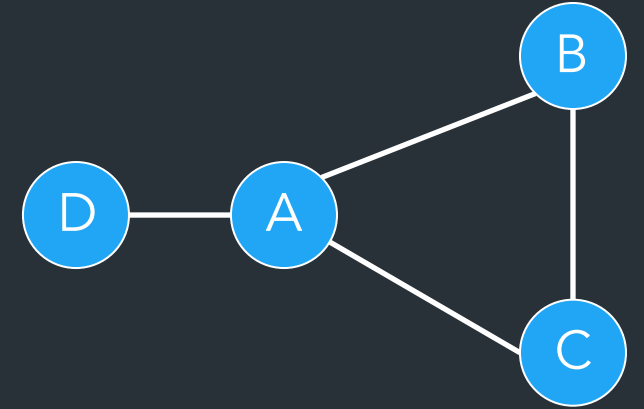
- When sending updates to node A, don't include routes you learned from A
- Prevents B and C from sending cost 2 to A

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

Distance-vector updates

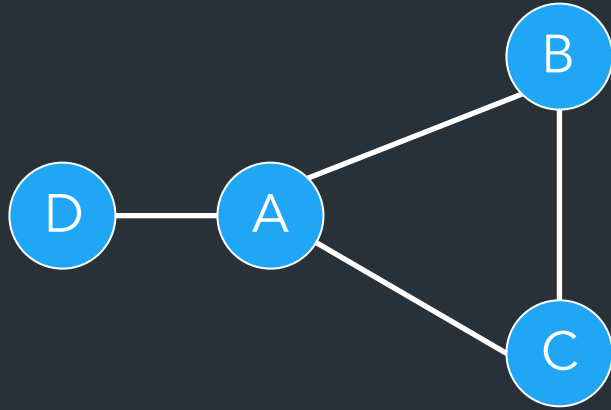
Even with split horizon + poison reverse,
can still create loops with >2 nodes



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



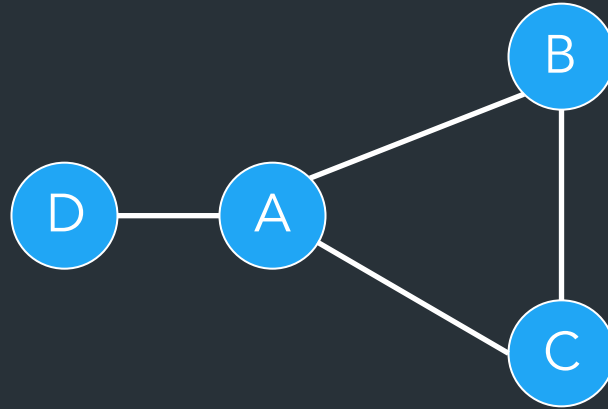
B's routing table

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?

Dealing with failures



- What happens when the D-A link fails?

Link State Routing

Link State Routing

- Strategy:
 - send to all nodes information about directly connected neighbors
- Link State Packet (LSP)
 - ID of the node that created the LSP
 - Cost of link to each directly connected neighbor
 - Sequence number (SEQNO)
 - TTL

Reliable Flooding

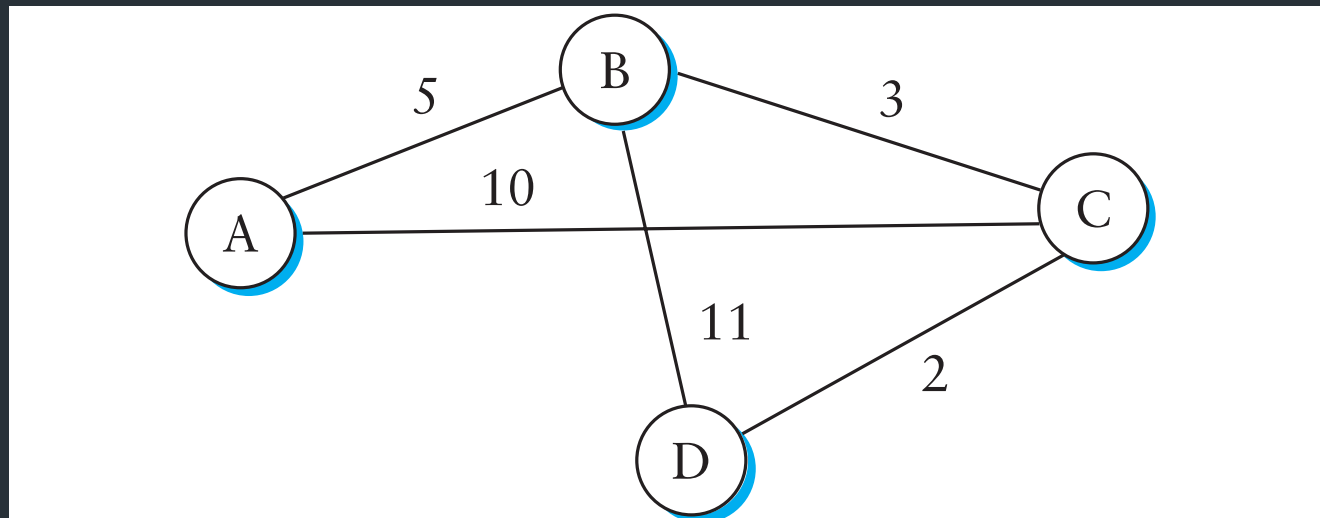
- Store most recent LSP from each node
 - Ignore earlier versions of the same LSP
- Forward LSP to all nodes but the one that sent it
- Generate new LSP periodically
 - Increment SEQNO
- Start at SEQNO=0 when reboot
 - If you hear your own packet with SEQNO=n, set your next SEQNO to n+1
- Decrement TTL of each stored LSP
 - Discard when TTL=0

Calculating best path

- Dijkstra's single-source shortest path algorithm
 - Each node computes shortest paths from itself
- Let:
 - N denote set of nodes in the graph
 - $l(i,j)$ denote the non-negative link between i,j
 - ∞ if there is no direct link between i and j
 - s denotes yourself (node computing paths)
 - $C(n)$ denote the cost of path from s to n
- Initialize variables
 - $M = \{s\}$ (set of nodes incorporated thus far)
 - For each n in $N - \{s\}$, $C(n) = l(s,n)$
 - $\text{Next}(n) = n$ if $l(s,n) < \infty$, – otherwise

Dijkstra's Algorithm

- While $N \neq M$
 - Let $w \in (N-M)$ be the node with lowest $C(w)$
 - $M = M \cup \{w\}$
 - Foreach $n \in (N-M)$, if $C(w) + l(w,n) < C(n)$
then $C(n) = C(w) + l(w,n)$, $\text{Next}(n) = \text{Next}(w)$
- Example: D: (D,0,-) (C,2,C) (B,5,C) (A,10,C)



Distance Vector vs. Link State

- # of messages (per node)
 - DV: $O(d)$, where d is degree of node
 - LS: $O(nd)$ for n nodes in system
- Computation
 - DV: convergence time varies (e.g., count-to-infinity)
 - LS: $O(n^2)$ with $O(nd)$ messages
- Robustness: what happens with malfunctioning router?
 - DV: Nodes can advertise incorrect *path* cost, which propagates through network
 - LS: Nodes can advertise incorrect *link* cost

Metrics

- Original ARPANET metric
 - measures number of packets enqueued in each link
 - neither latency nor bandwidth in consideration
- New ARPANET metric
 - Stamp arrival time (AT) and departure time (DT)
 - When link-level ACK arrives, compute
$$\text{Delay} = (\text{DT} - \text{AT}) + \text{Transmit} + \text{Latency}$$
 - If timeout, reset DT to departure time for retransmission
 - Link cost = average delay over some time period
- Fine Tuning
 - Compressed dynamic range
 - Replaced Delay with link utilization
- Today: commonly set manually to achieve specific goals

Examples

- RIPv2
 - Fairly simple implementation of DV
 - RFC 2453 (38 pages)
- OSPF (Open Shortest Path First)
 - More complex link-state protocol
 - Adds notion of *areas* for scalability
 - RFC 2328 (244 pages)
- ISIS (Intermediate System to Intermediate System)
 - OSI standard (210 pages)
 - Link-state protocol (similar to OSPF)
 - Does not depend on IP

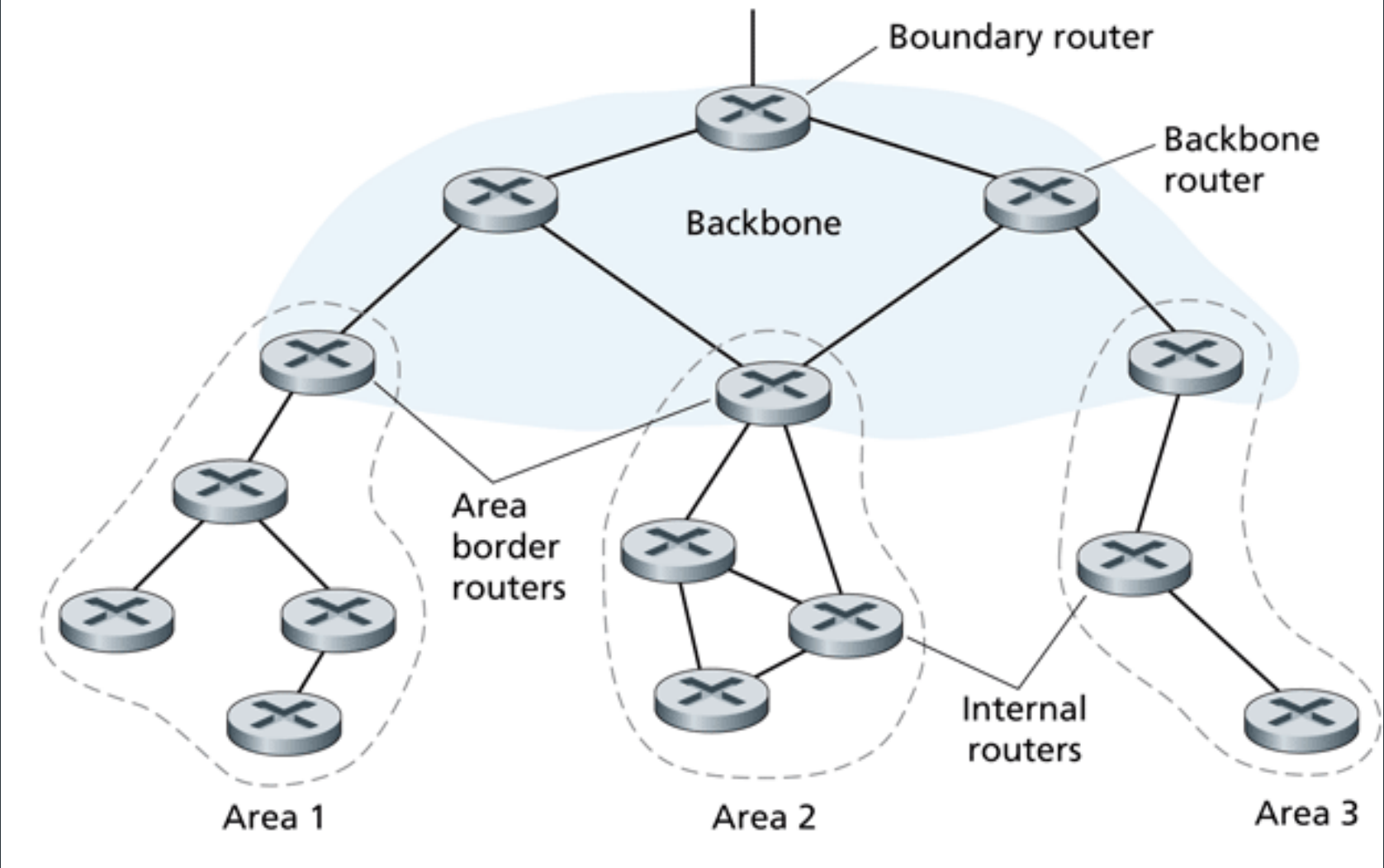
OSPFv2

- Link state protocol
- Runs directly over IP (protocol 89)
 - Must provide its own reliability
- All exchanges are authenticated
- Adds notion of *areas* for scalability

OSPF Areas

- Area 0 is “backbone” area (includes all boundary routers)
- Traffic between two areas must always go through area 0
- Only need to know how to route exactly within area
- Otherwise, just route to the appropriate area
- Tradeoff: scalability versus optimal routes

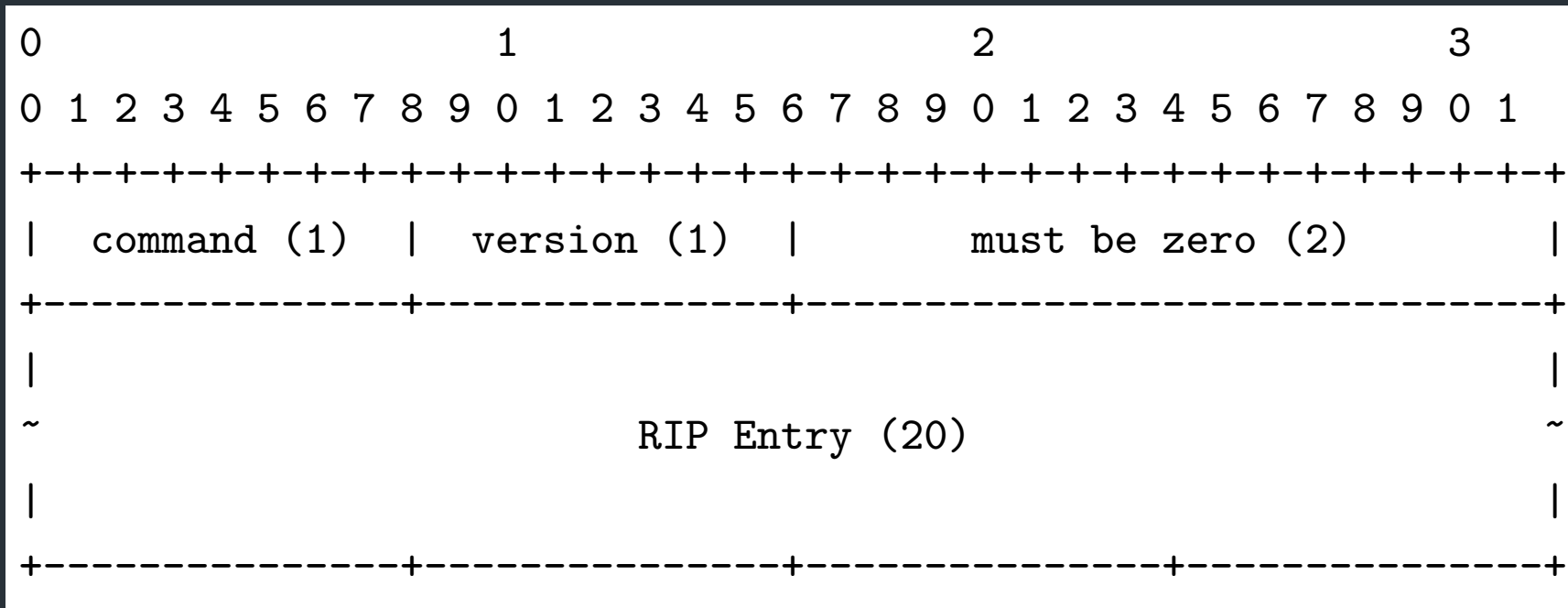
OSPF Areas



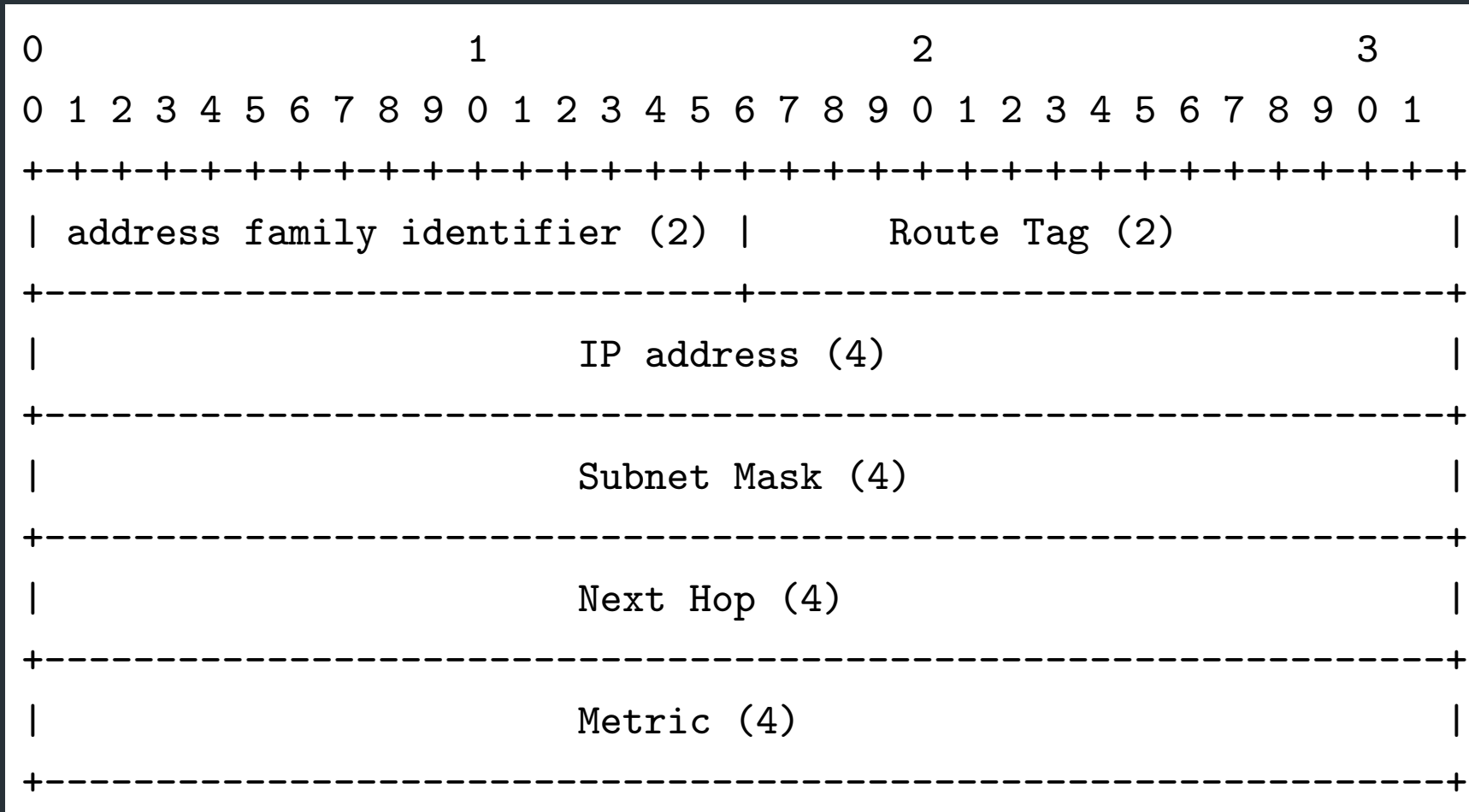
RIPv2

- Runs on UDP port 520
 - (IP assignment: directly in IP, protocol 200)
- Link cost = 1
- Periodic updates every 30s, plus triggered updates
- Relies on count-to-infinity to resolve loops
 - Maximum diameter 15 ($\infty = 16$)
 - Supports split horizon, poison reverse
- Deletion
 - If you receive an entry with metric = 16 from parent OR
 - If a route times out

Packet format



RIPv2 Entry

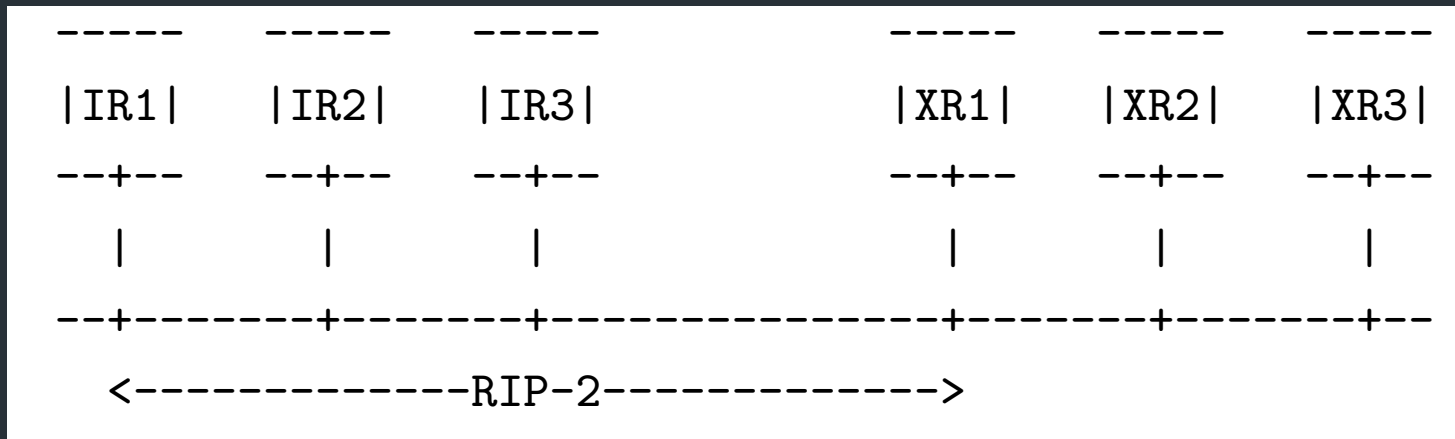


Route Tag field

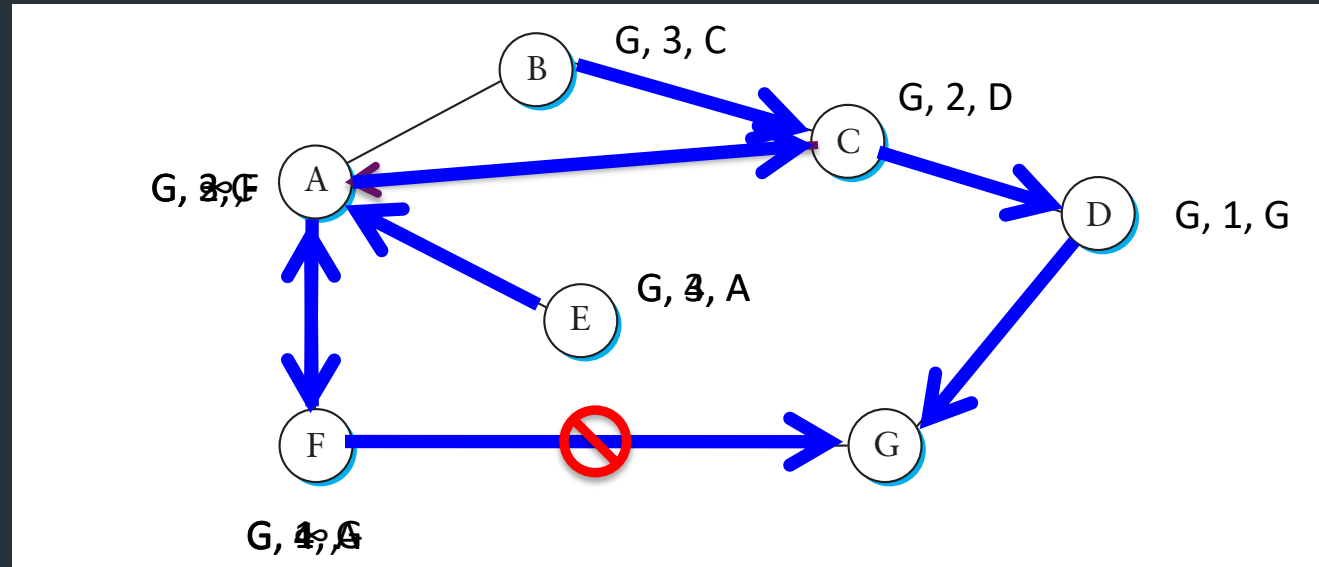
- Allows RIP nodes to distinguish internal and external routes
- Must persist across announcements
- E.g., encode AS

Next Hop field

- Allows one router to advertise routes for multiple routers on the same subnet
- Suppose only XR1 talks RIPv2:

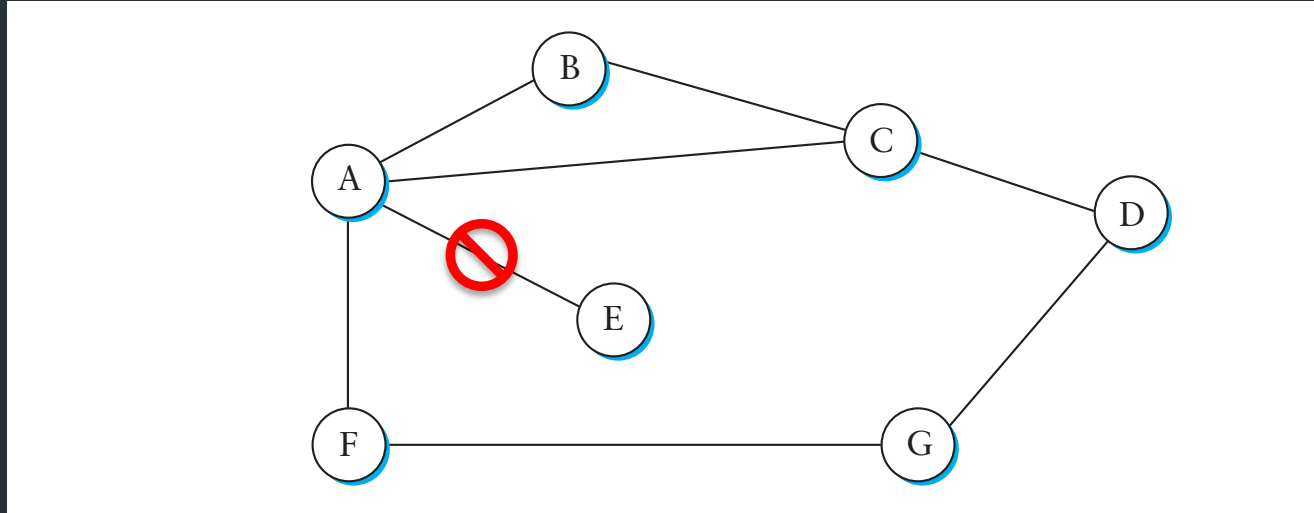


Adapting to Failures



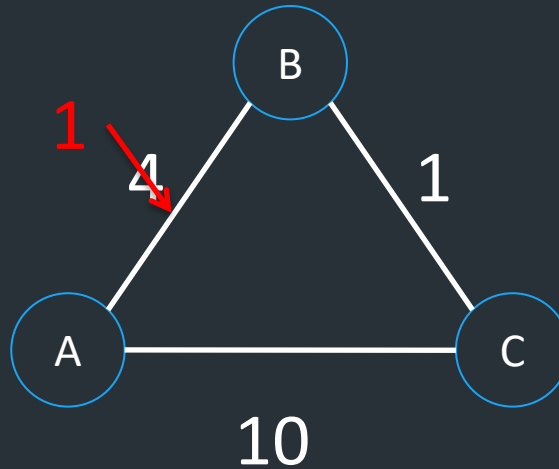
- F-G fails
- F sets distance to G to infinity, propagates
- A sets distance to G to infinity
- A receives periodic update from C with 2-hop path to G
- A sets distance to G to 3 and propagates
- F sets distance to G to 4, through A

Count-to-Infinity



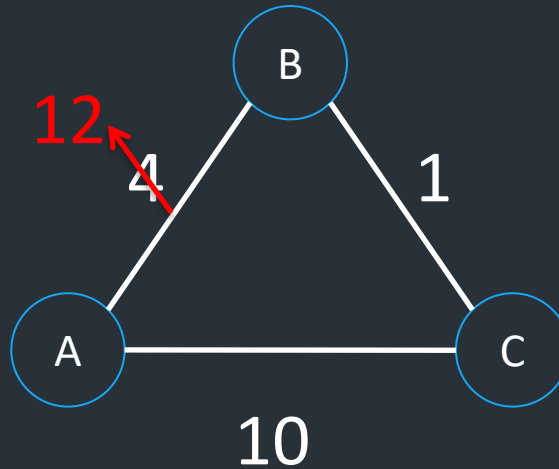
- Link from A to E fails
- A advertises distance of infinity to E
- B and C advertise a distance of 2 to E
- B decides it can reach E in 3 hops through C
- A decides it can reach E in 4 hops through B
- C decides it can reach E in 5 hops through A, ...
- **When does this stop?**

Good news travels fast



- A decrease in link cost must be fresh information
- Network converges at most in $O(\text{diameter})$ steps

Bad news travels slowly



- An increase in cost may cause confusion with old information, may form loops
- Consider routes to A
- Initially, B:A,4,A; C:A,5,B
- Then B:A,12,A, selects C as next hop -> B:A,6,C
- C -> A,7,B; B -> A,8,C; C -> A,9,B; B -> A,10,C;
- C finally chooses **C:A,10,A**, and B -> A,11,C!

Next Class

- Inter-domain routing: how scale routing to the entire Internet

IP Connectivity

For each destination address, a router must either:

- Have matching prefix in its forwarding table
 - Know a “smarter router”, ie default route for unknown prefixes
-
- Core routers know everything => no default route!
 - Manage using notion of *Autonomous System (AS)*

Scaling Issues

Problem: Every router must be able to forward based on *any* destination IP address

- Map destination address => next hop
- Could we have one entry per IP? No!

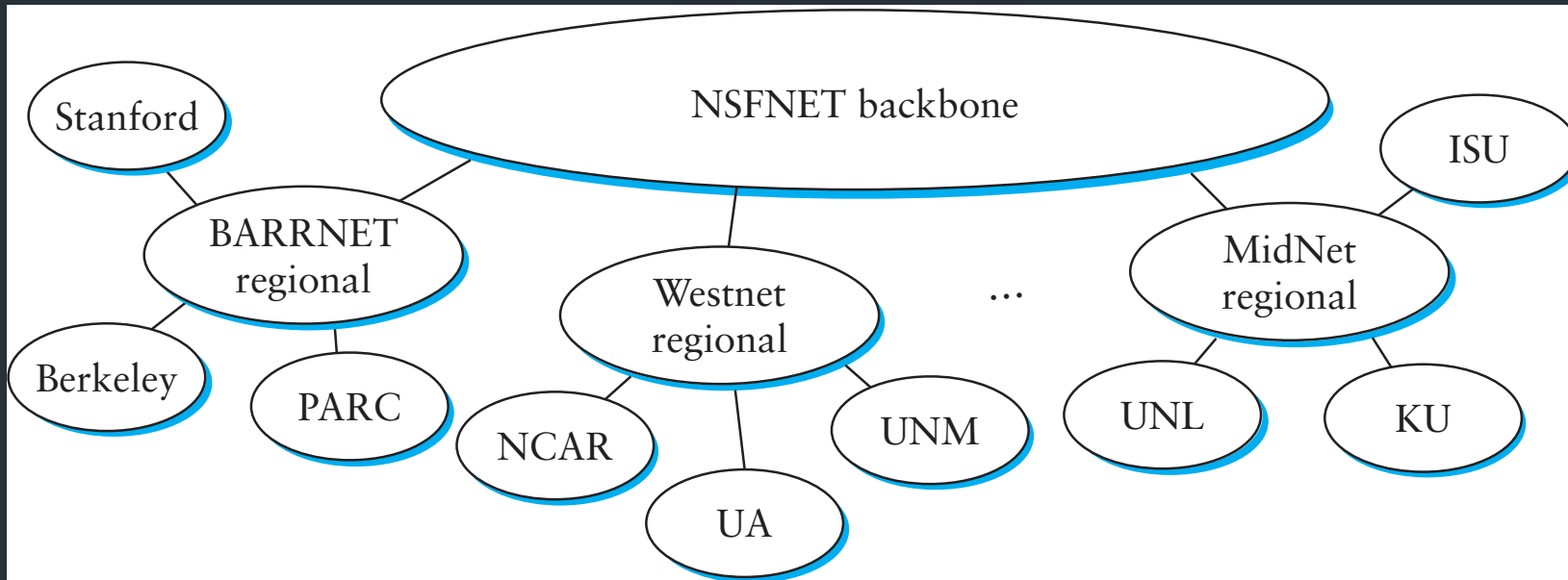
Solutions

- Leverage hierarchy in network topology
- Address aggregation
 - Address allocation is very important (should mirror topology)
- Default routes

Autonomous Systems (ASes)

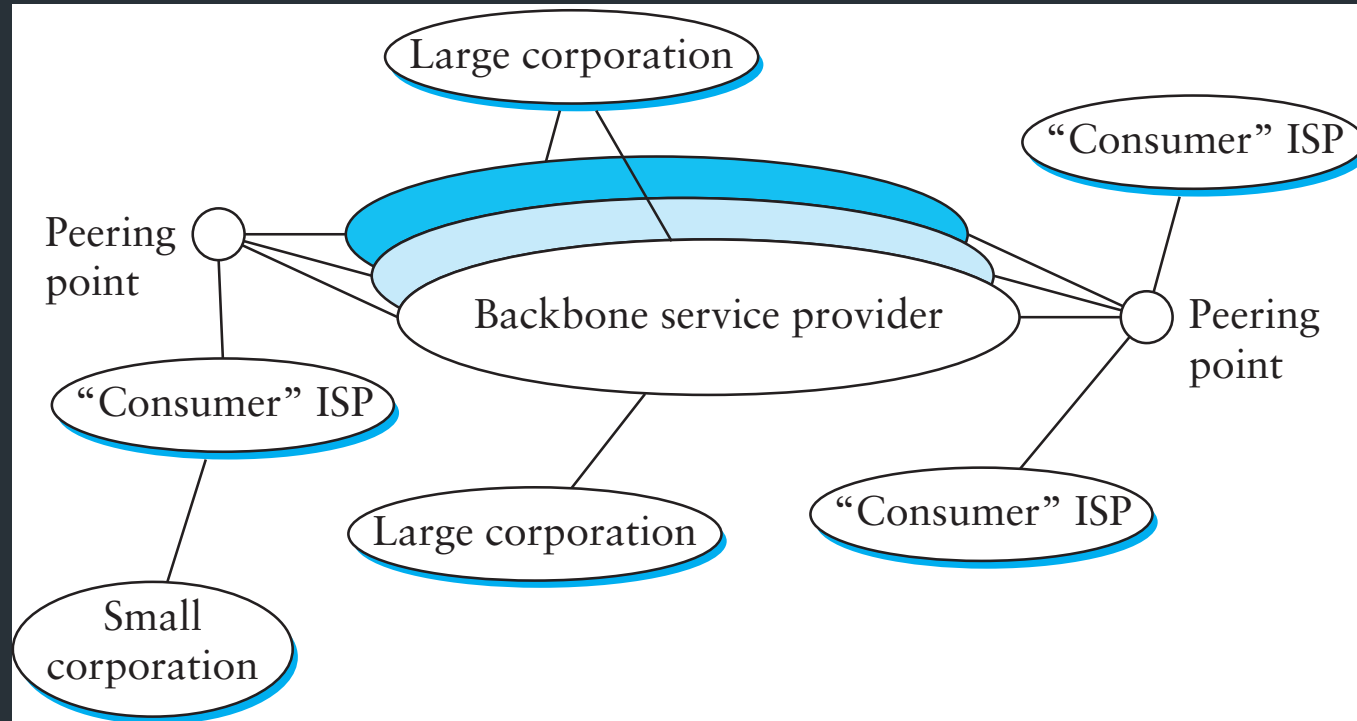
- Correspond to an administrative domain
 - AS's reflect organization of the Internet
 - E.g., Brown, large company, etc.
 - Identified by a 16-bit number (now 32)
- Goals
 - AS's choose their own local routing algorithm
 - AS's want to set policies about non-local routing
 - AS's need not reveal internal topology of their network

Internet structure, 1990



- Several independent organizations
- Hierarchical structure with single backbone

Internet structure, today



- Multiple backbones, more arbitrary structure