Lecture 5 (Routing 1)

Routing Principles

CS 168, Spring 2026 @ UC Berkeley

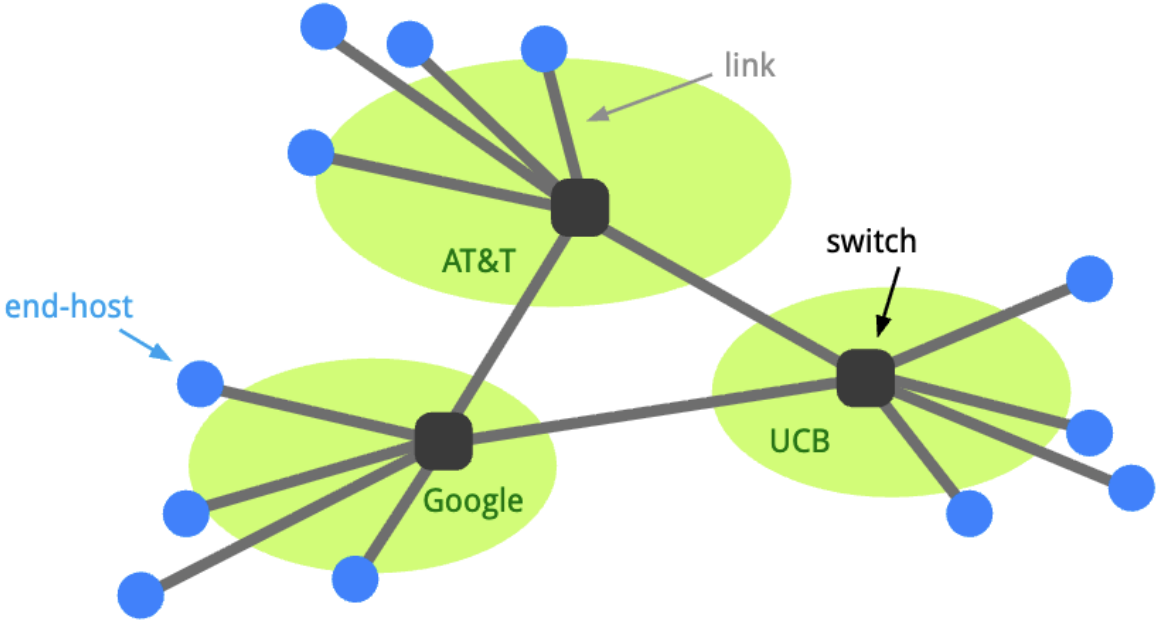
Slides credit: Sylvia Ratnasamy, Rob Shakir, Peyrin Kao, Iuniana Oprescu

Some Announcements

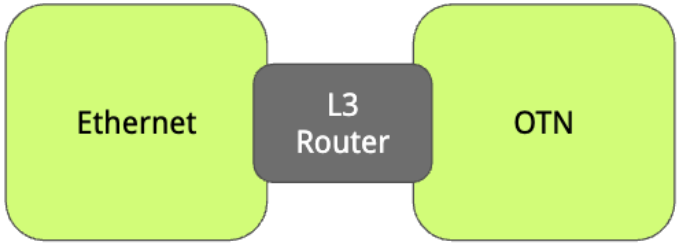
- Class expanded and everyone got off the waitlist
- Midterm is set to be Thursday, October 22 from 7:00-9:00PM
  - Alternate time midterm is on Thursday, October 22 from 9:00-11:00PM
- Sylvia traveling September 17-22

The Internet is a network of networks

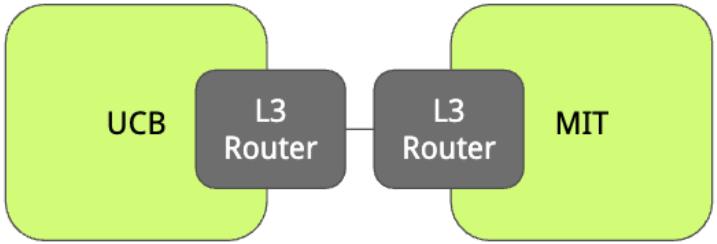
- The goal is to enable data transfer between any two end hosts
- Switches/routers are the intermediate nodes that connect end hosts



- Routing is the task of finding paths through a network
  - Finding a viable path between end hosts is necessary for data transfer
- Routing in the internet is done at the L3 layer, since the internet was focused on **global data delivery** (we already had local delivery).



To interconnect different L2 network technologies.



And to interconnect different administrative domains.

Why Do We Need Routing?

Lecture 5, CS 168, Spring 2026

Defining the Routing Problem

- **Why Do We Need Routing?**
- Modeling the Network
- What Makes Routing Hard?
- Types of Routing Protocols

Routing States

- Destination-Based Forwarding
- Routing State Validity
- Least-Cost Routing

Sneak Peek at Routing Algorithms

- Static routes
- Routing experiment

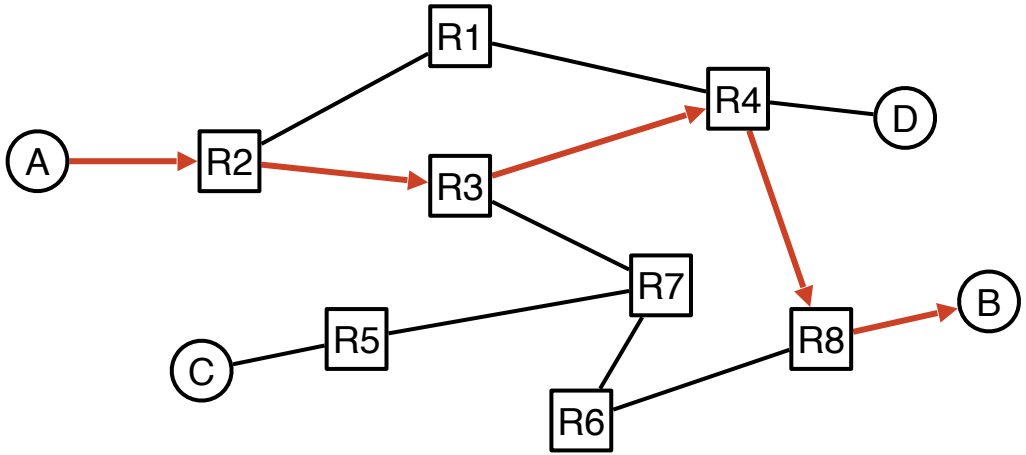
Today is the first of 6 lectures on routing.

We'll first formally define the problem.

- Why is it needed?
- Why is it a hard problem?
- What types of algorithms exist?

Then, we'll see how to assess a solution.

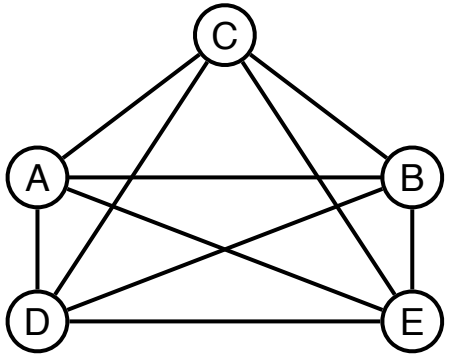
- What does a solution look like?
- What makes a solution valid?
- What makes a solution good?



If 2 machines want to communicate, we can add a link between them.

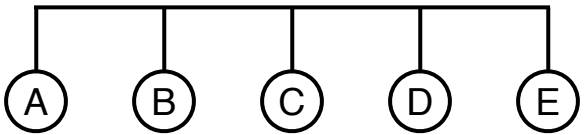
What if we had 5 machines?

- We could add a link between every pair of machines.
  - The result is called a **full-mesh** topology.
- Problem: This doesn't scale well. Imagine adding a new machine.
- Has benefits in limited settings.
  - Good if we need dedicated high bandwidth between every pair of machines.



Another approach: Use a single link (wire) to connect all 5 machines.

- Scales better than the full-mesh topology.
- Problem: Less bandwidth available. Everyone has to share the link.

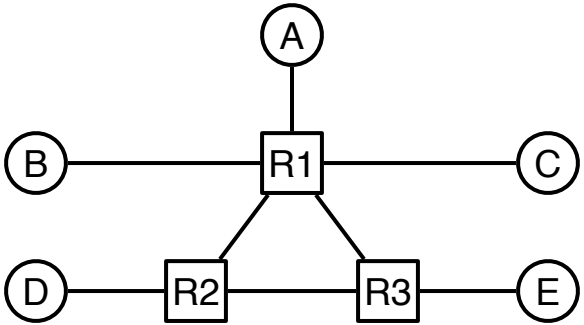


To use more sophisticated topologies, we need to introduce a **router**.

- Router: An intermediate machine that can forward data.

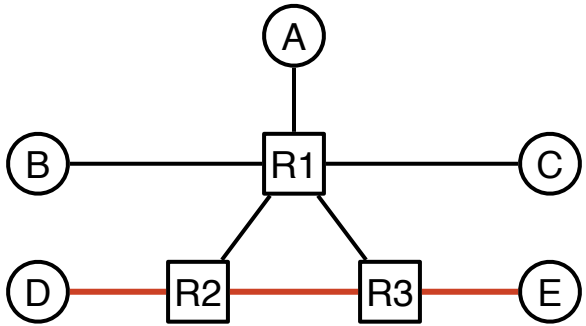
If we use routers to connect the machines:

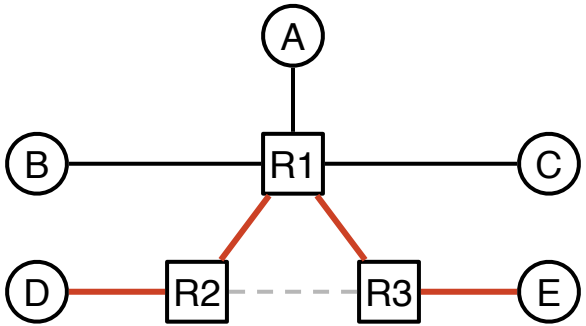
- We use fewer links than the full-mesh topology.
- We have more capacity than just a single link.



Another benefit: If a link goes down, traffic could use another path.

We now need some way to compute paths through the network: Routing protocols!





Link goes down!

Modeling the Network

Lecture 5, CS 168, Spring 2026

Defining the Routing Problem

- Why Do We Need Routing?
- **Modeling the Network**
- What Makes Routing Hard?
- Types of Routing Protocols

Routing States

- Destination-Based Forwarding
- Routing State Validity
- Least-Cost Routing

Sneak Peek at Routing Algorithms

- Static routes
- Routing experiment

We'll assume every machine on the network is one of two types.

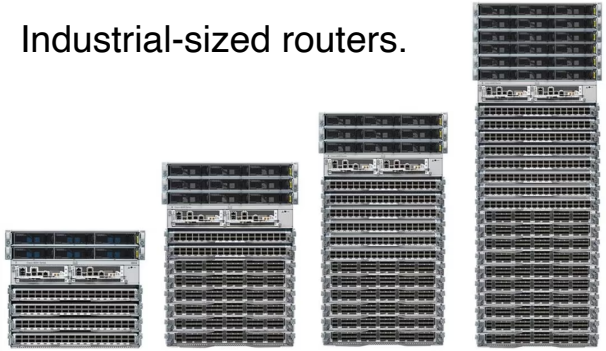
End hosts send and receive packets, to and from other end hosts.

- Example: Your personal computer.
- End hosts don't forward intermediate packets.

Routers forward intermediate packets.

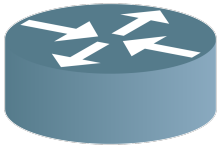
- Example: Your home router, heavy-duty router in a datacenter.
- For now, assume routers don't send and receive packets of their own.

Industrial-sized routers.



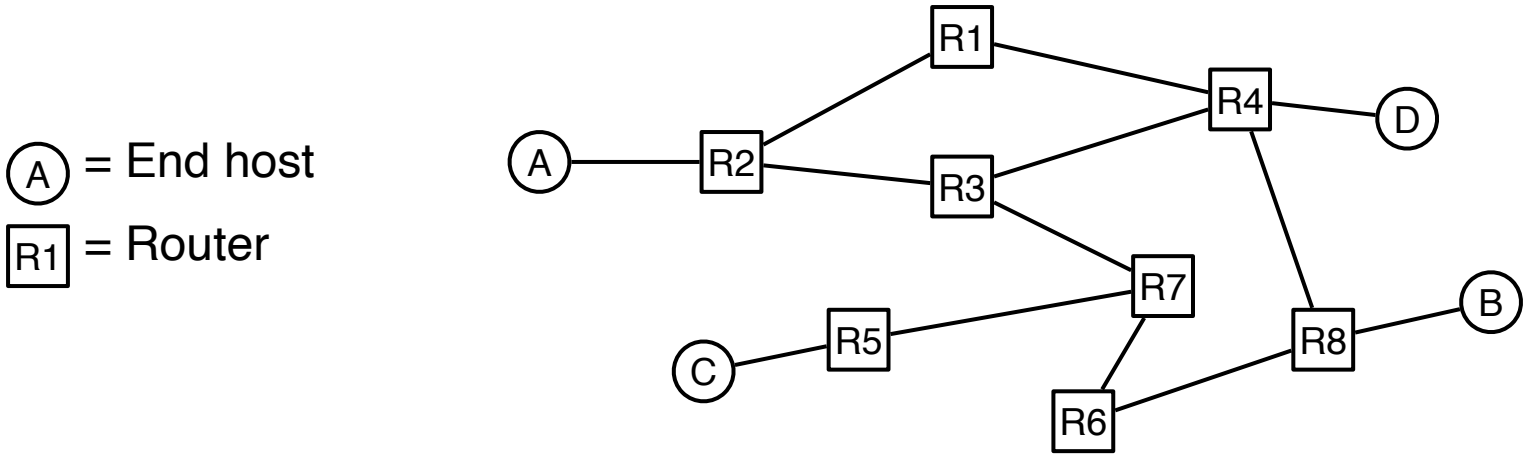
Home router.

Router symbol in diagrams.



We'll draw the network as a graph.

- Each edge represents a link. For now, assume a link connects exactly 2 machines.
- For now, assume each machine is identified by a unique label.
  - We'll think more about addressing later.



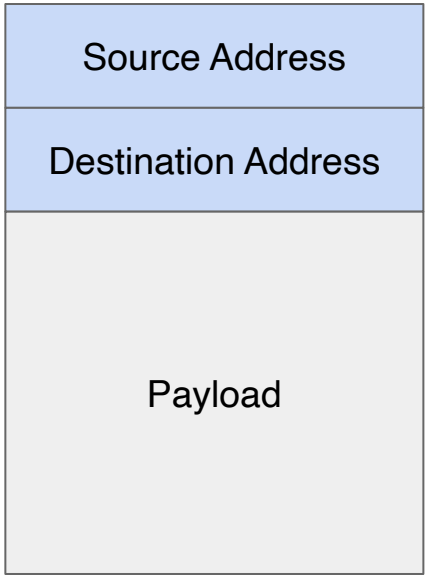
Packets are the basic unit of data sent across the network.

Packets have a header with metadata.

- For now, we only care about the source address and destination address fields.

Packets have a payload with the application data.

- Example: Website contents, image, etc.
- Routing isn't concerned about the payload. It's just some sequence of 1s and 0s we have to forward.



What Makes Routing Hard?

Lecture 5, CS 168, Spring 2026

Defining the Routing Problem

- Why Do We Need Routing?
- Modeling the Network
- **What Makes Routing Hard?**
- Types of Routing Protocols

Routing States

- Destination-Based Forwarding
- Routing State Validity
- Least-Cost Routing

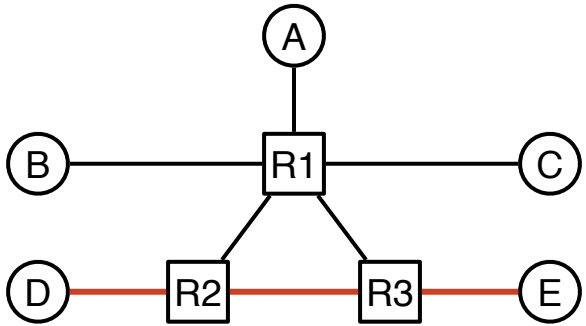
Sneak Peek at Routing Algorithms

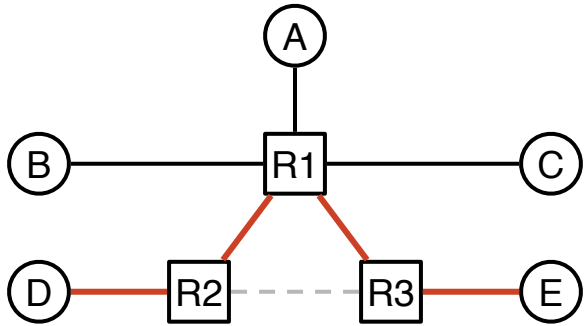
- Static routes
- Routing experiment

The network graph is constantly changing.

- Hosts join and leave the network.
- Links can fail, and new links can be added.

Our routing protocol needs to be robust to changes.





Link goes down!

Routers don't have a global, birds-eye view of the network.

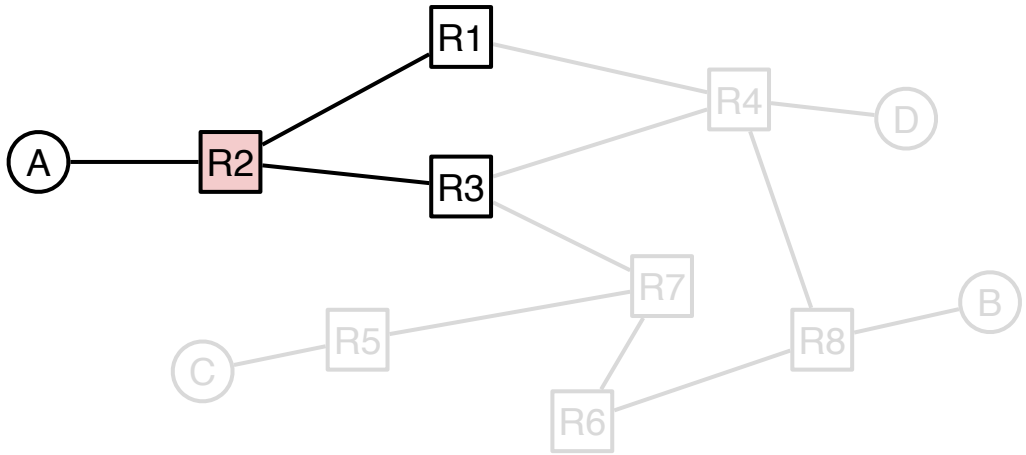
- If a link fails, routers don't automatically know about it.

Routing protocols have to be **distributed**.

- There's no central mastermind computing the answer.
- Each router computes its own part of the answer.
- Routers must coordinate with each other in the protocol.

R2 can't see the whole network.

R2 might only be able to know about its direct neighbors.



At Layer 3, links are best-effort. Packets could get dropped.

In summary, challenges of routing:

1. Topology changes over time.
2. Routers don't have a global view of the network.
3. Links are best-effort.

Types of Routing Protocols

Lecture 5, CS 168, Spring 2026

Defining the Routing Problem

- Why Do We Need Routing?
- Modeling the Network
- What Makes Routing Hard?
- **Types of Routing Protocols**

Routing States

- Destination-Based Forwarding
- Routing State Validity
- Least-Cost Routing

Sneak Peek at Routing Algorithms

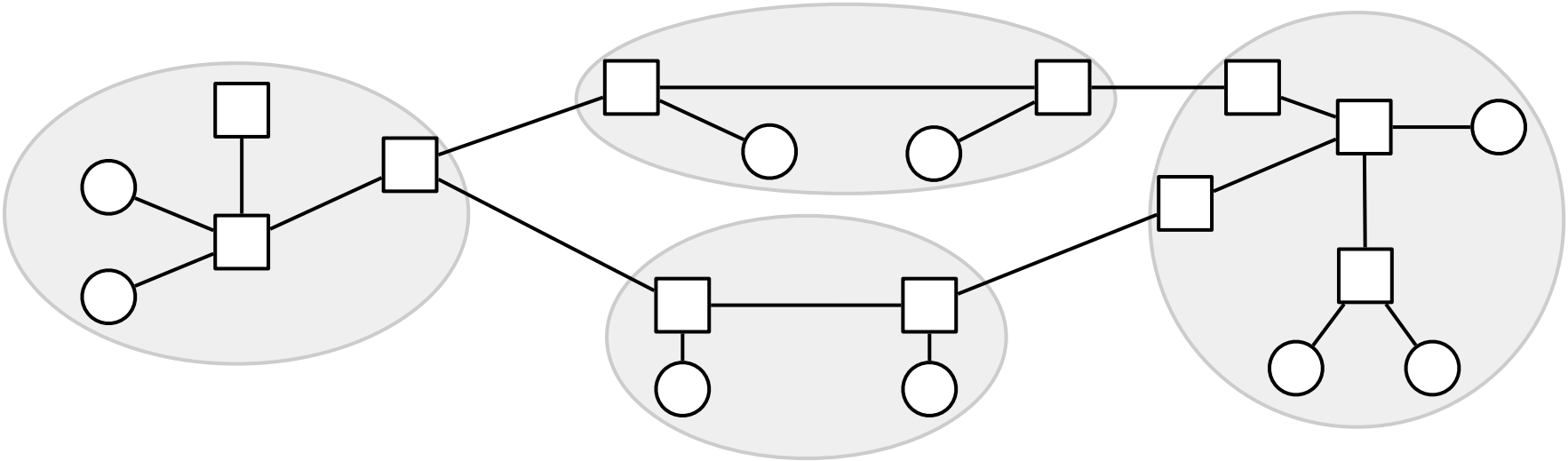
- Static routes
- Routing experiment

There are an endless number of possible routing protocols.

- We're going to talk about the "archetypal Internet."
- We'll see some alternative approaches later.

Recall: The Internet is a network of networks.

- The Internet does not have a single giant routing protocol.

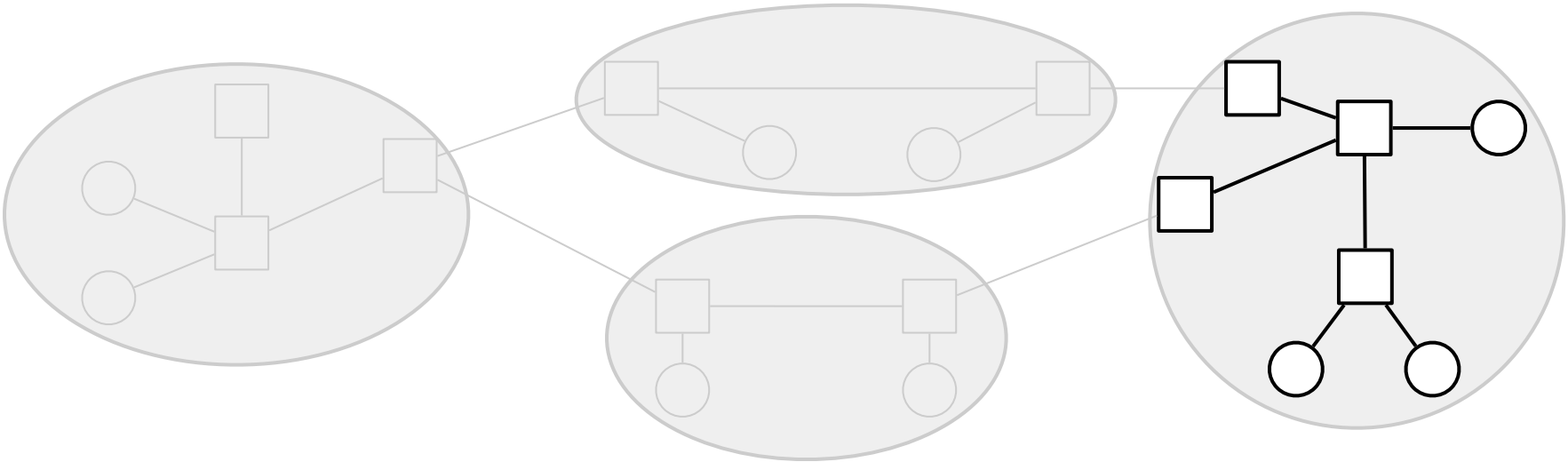


Intra-domain routing protocols compute routes within a single network.

- Sometimes called **Interior Gateway Protocols (IGPs)**.

Each network can choose their own intra-domain protocol based on their needs.

- Networks differ in size (geographic, number of hosts), capacity (bandwidth, latency), support (money, staff), etc.

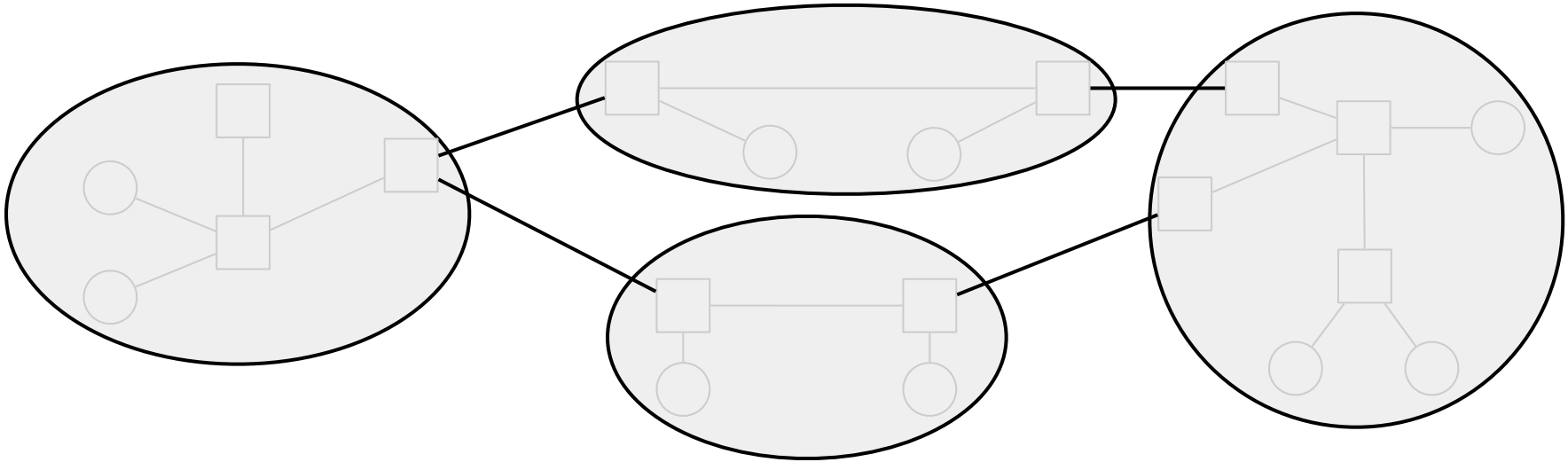


Inter-domain routing protocols compute routes between networks..

- Sometimes called **Exterior Gateway Protocols (EGPs)**.

Everybody has to agree on the same protocol.

- The Internet has used BGP since the 1990s.



Classifying routing protocols by *where* they operate:

- Let individual networks choose how to route *inside* their network. (Intra-domain.)
- Have all networks agree on how to route *between* each other. (Inter-domain.)

In practice, the lines between intra-domain and inter-domain routing are blurred.

- Example: BGP is sometimes used inside networks, as well as between networks.

We can also classify routing protocols by *how* they operate:

- Distance-vector protocols.
  - Path-vector protocols. (An extension to the distance-vector idea.)
- Link-state protocols.

Destination-Based Forwarding

Lecture 5, CS 168, Spring 2026

Defining the Routing Problem

- Why Do We Need Routing?
- Modeling the Network
- What Makes Routing Hard?
- Types of Routing Protocols

Routing States

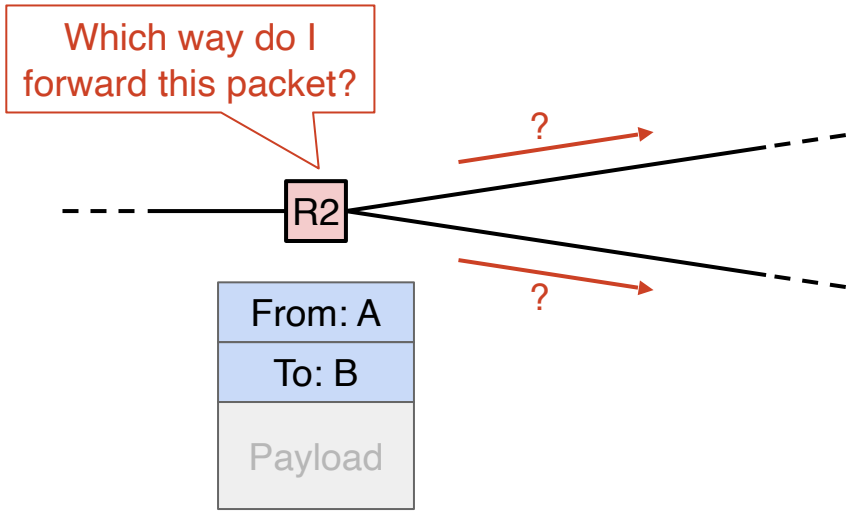
- **Destination-Based Forwarding**
- Routing State Validity
- Least-Cost Routing

Sneak Peek at Routing Algorithms

- Static routes
- Routing experiment

The basic challenge: When a packet arrives at a router, how does the router know where to send it next, such that it will eventually arrive at the desired destination?

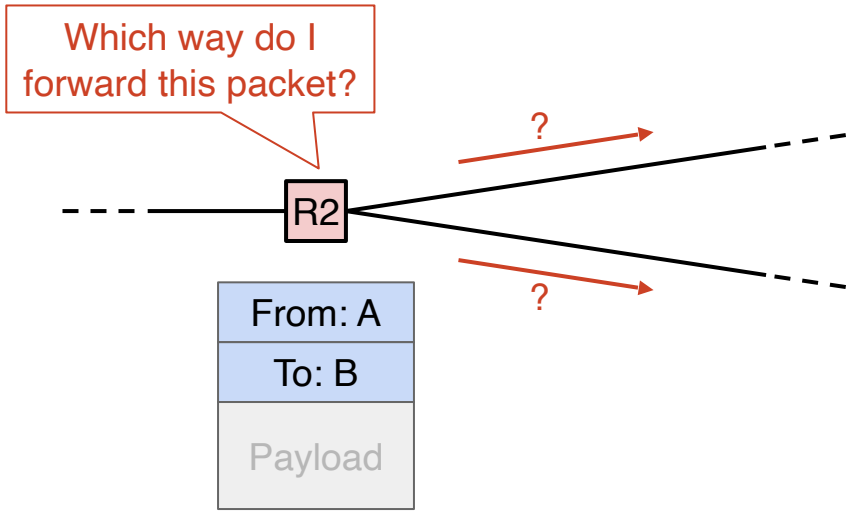
- The **next hop** is where to forward the packet next.



Some bad strategies:

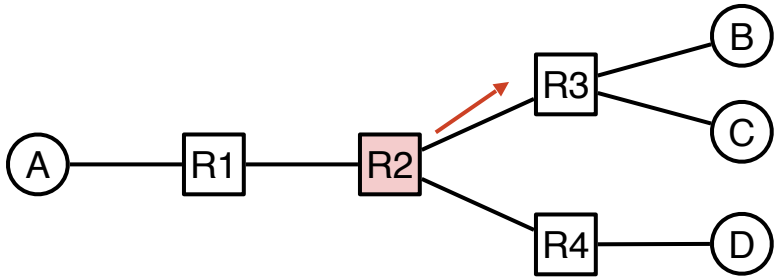
- Send along a random link – bad because packet might not reach destination.
- Send along every link – bad because wasting bandwidth.

Let's formalize what a solution looks like, and what makes it valid/good.

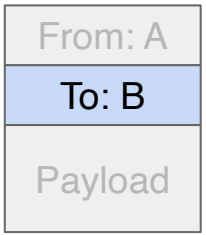


The Internet uses **destination-based forwarding**.

- Each router keeps a table, mapping destinations to next hops.
- The decision only depends on the destination field of the packet.



When a packet arrives, look up the destination...

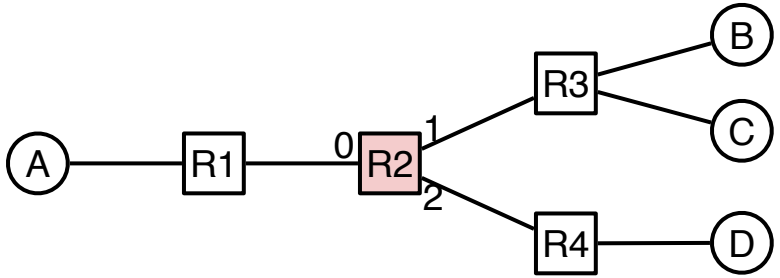


...and send along the corresponding next hop.

R2's Table	
Destination	Next Hop
A	R1
B	R3
C	R3
D	R4

In real life, the table often uses physical ports instead of next hops.

- Conceptual: "Send to next-hop of R3."
- Reality: "Send out of physical port 1."
- We'll use the conceptual picture for simplicity.



R2's Table (Conceptual)	
Destination	Next Hop
A	R1
B	R3
C	R3
D	R4

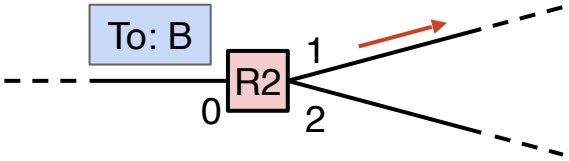
R2's Table (Reality)	
Destination	Port
A	0
B	1
C	1
D	2

Forwarding:

- Look up packet's destination in table, and send packet to neighbor.
- Inherently *local*. Depends only on arriving packet and local table.
- Occurs every time a packet arrives (nanoseconds).

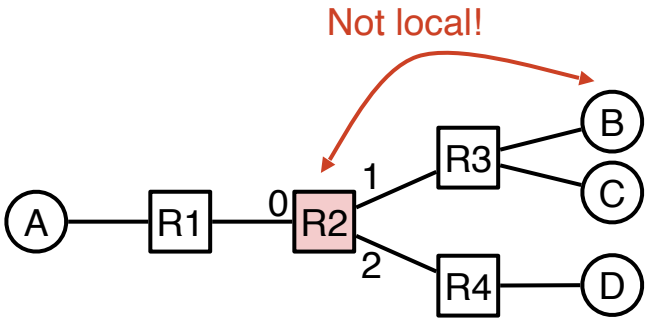
Routing:

- Communicates with other routers to determine how to populate tables.
- Inherently *global*. Must know about all destinations, not just local ones.
- Occurs every time the network changes (e.g. a link fails).



Forwarding is local.

R2's Table	
Destination	Port
A	0
B	1
C	1
D	2



Routing is global.

Routing State Validity

Lecture 5, CS 168, Spring 2026

Defining the Routing Problem

- Why Do We Need Routing?
- Modeling the Network
- What Makes Routing Hard?
- Types of Routing Protocols

Routing States

- Destination-Based Forwarding
- **Routing State Validity**
- Least-Cost Routing

Sneak Peek at Routing Algorithms

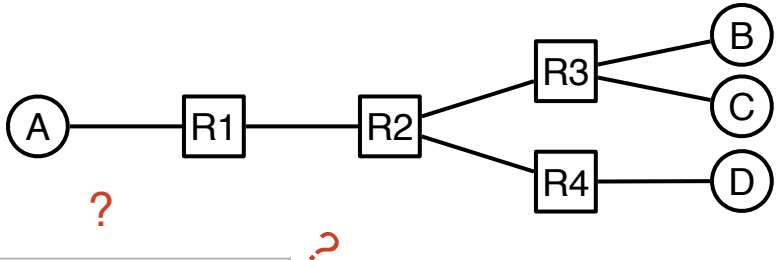
- Static routes
- Routing experiment

Routing State Validity is Global

A routing state is **valid** if packets actually reach their destinations.

A **local** routing state is a table in a single router.

- By itself, the state in a single router can't be evaluated for validity.



R2's Table	
Destination	Next Hop
A	R1
B	R3
C	R3
D	R4

Is this local state valid?

Will it get my packets to their destinations?

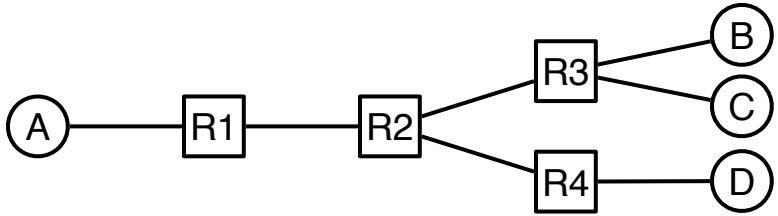
No way to tell from just this info!

The term "routing state validity" might only be used at Berkeley.

A **global** routing state is a collection of tables in all routers.

- Global state determines the paths a packet takes.
- Global state is valid if it produces forwarding decisions that deliver packets to their destinations.

Given a global state, how can you tell if it's valid?



R1's Table	
Destination	Next Hop
A	A
B	R2
C	R2
D	R2

R2's Table	
Destination	Next Hop
A	R1
B	R3
C	R3
D	R4

R3's Table	
Destination	Next Hop
A	R2
B	B
C	C
D	R2

R4's Table	
Destination	Next Hop
A	R2
B	R2
C	R2
D	D

A global routing state is valid *if and only if* there are no dead ends and no loops.

Dead end: A packet arrives at a router, but there is no next hop to forward it.

- Packet arriving at destination doesn't count as a dead end.

Uh oh. R3 didn't forward the packet.

Loop: A packet cycles around the same set of routers.

- If forwarding only depends on destination field, if a packet gets stuck in a loop, it can never escape.

Uh oh. Packet is stuck looping between R2 and R3.

A global routing state is valid (i.e. packets reach their destination)
if and only if there are no dead ends and no loops.

- To prove this, we to check both directions (necessary and sufficient).

Packets reach their destination *only if* there are no dead ends and loops. (Necessary.)

Proof:

- If you hit a dead end, you'll never reach the destination.
- If you get stuck in a loop, you'll never reach the destination.
  - In destination-based forwarding, no way to escape the loop.
    Forwarding decision is the same every time the packet arrives at a router.
  - Destination isn't part of the loop. It wouldn't have forwarded the packet!

If there are no dead ends and loops, packets will reach their destination. (Sufficient.)

Proof:

- Assume there are no loops and dead ends. Then:
  - Packet won't visit the same router twice. (We said no loops.)
  - Packet won't stop before hitting destination. (We said no dead ends.)
- Therefore:
  - Packet must keep wandering the network, visiting different routers.
  - There are only a finite number of unique routers to visit.
  - So the packet must eventually hit the destination.

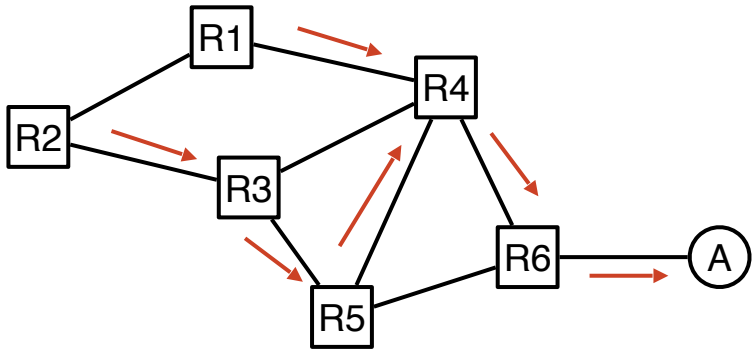
We've proven both directions!

A global routing state is valid *if and only if* there are no dead ends and no loops.

How do we apply this condition to check if a global state is valid?

- Strategy: Check each destination separately.
- For a given destination: Use the tables to see how each router forwards packets.
- The next-hop becomes an outgoing arrow.
- The result is a **directed delivery tree** for that destination.

R2's Table	
Destination	Next Hop
A	R3
...	...



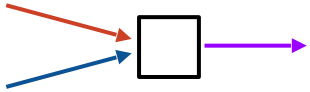
A **directed delivery tree** shows how packets get forwarded toward one destination.

Properties:

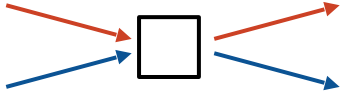
For now, ok to assume this.

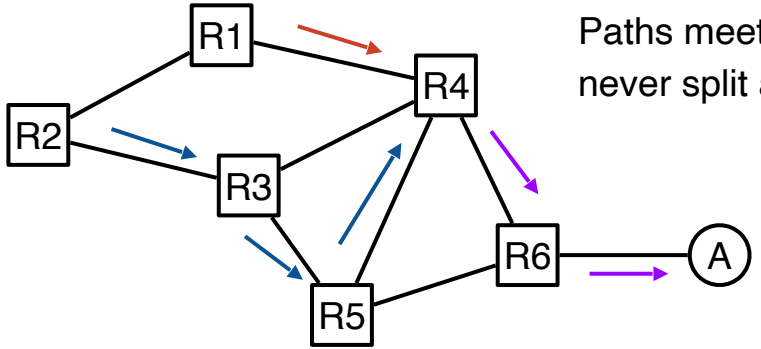
- At each router, 1 next-hop per destination. 1 outgoing arrow per node!
- Once paths meet, they never split. (Same destination = same next hop.)

Allowed:



Not allowed:



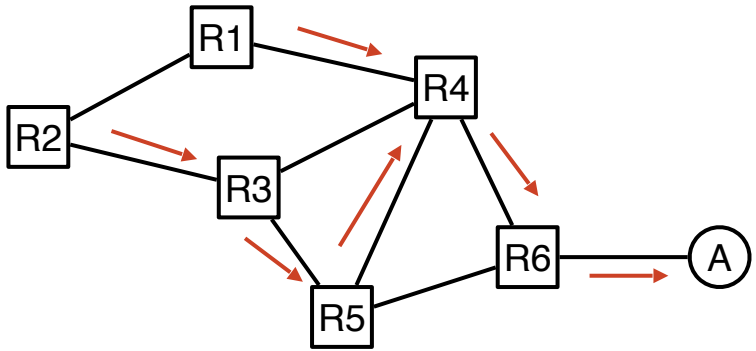


Paths meet at R4 and never split again.

A valid directed delivery tree is an **directed spanning tree** rooted at the destination.

- directed = Edges have arrows.
- Tree = No cycles, no disconnected components.
- Spanning = Tree touches every node, so every node can talk to the destination
- All edges point toward destination.

Starting at any node and following edges will reach the destination.



Verifying Routing State Validity – Steps

1. Pick a single destination.
2. For each router, draw an arrow to the next-hop.
   - Destination-based forwarding = there is only one outgoing arrow.
3. Delete all links with no arrows.
4. State is valid if and only if the remaining graph is a valid directed delivery tree.
   - No dead ends. (Node with no outgoing arrow.)
   - No loops. (Cycles in the graph.)
   - A directed spanning tree where all edges point toward the destination.

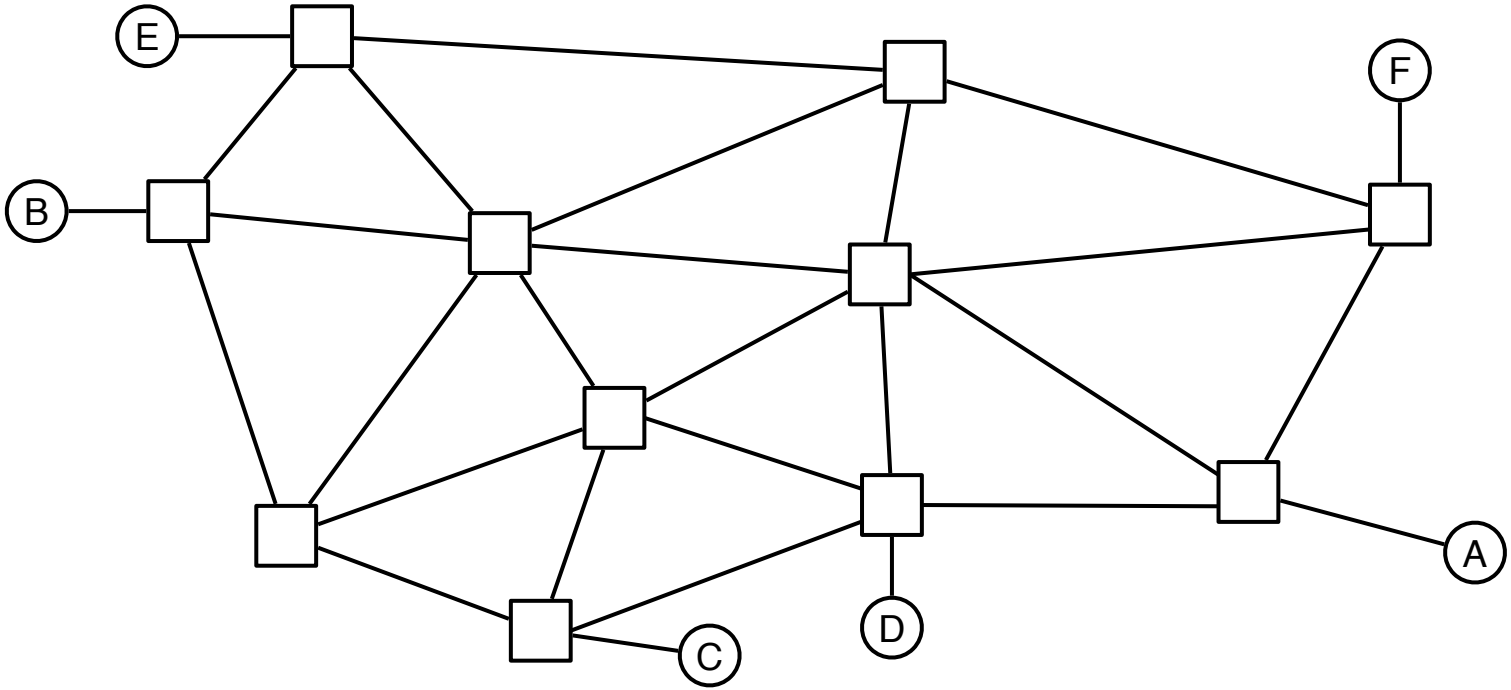
Touches every node.

No cycles. No disconnected nodes.

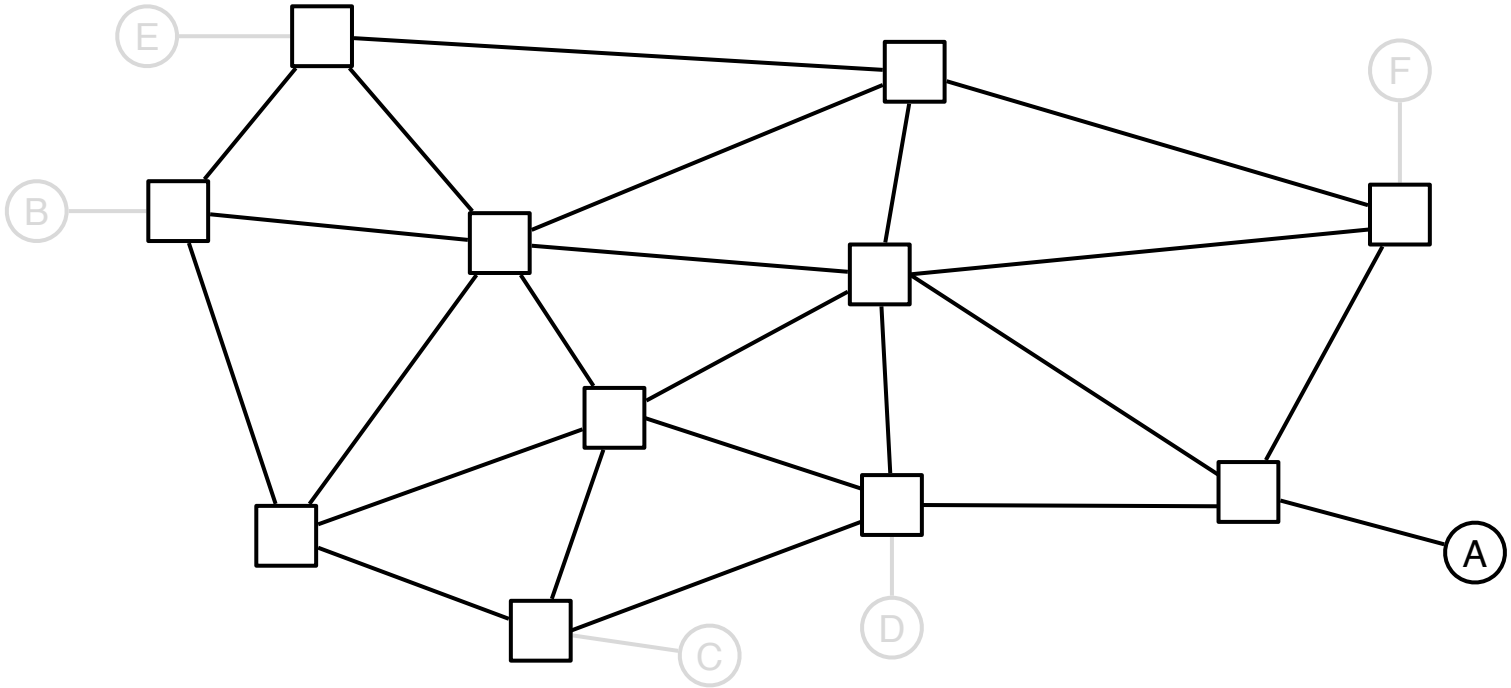
5. Repeat for every destination!

Is this global routing state valid?

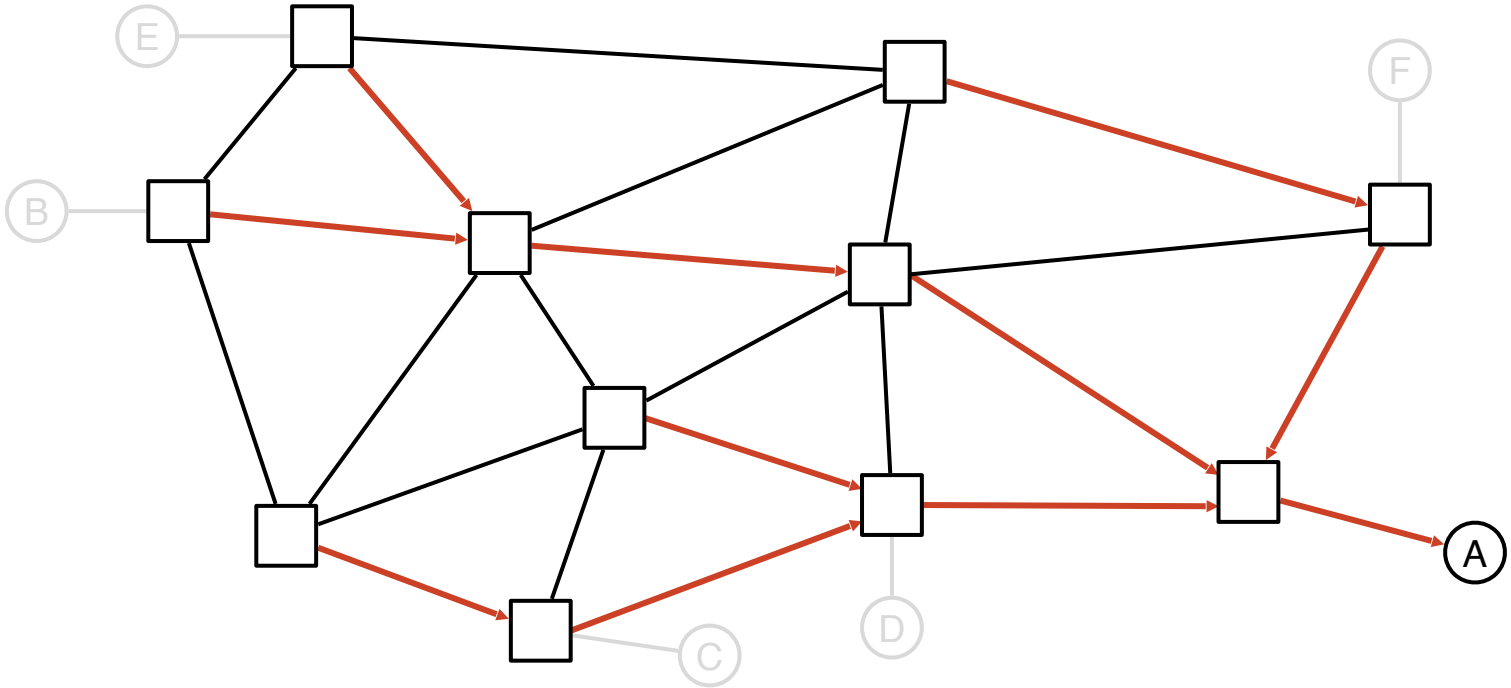
- Recall: Global routing state = what's in all the forwarding tables.



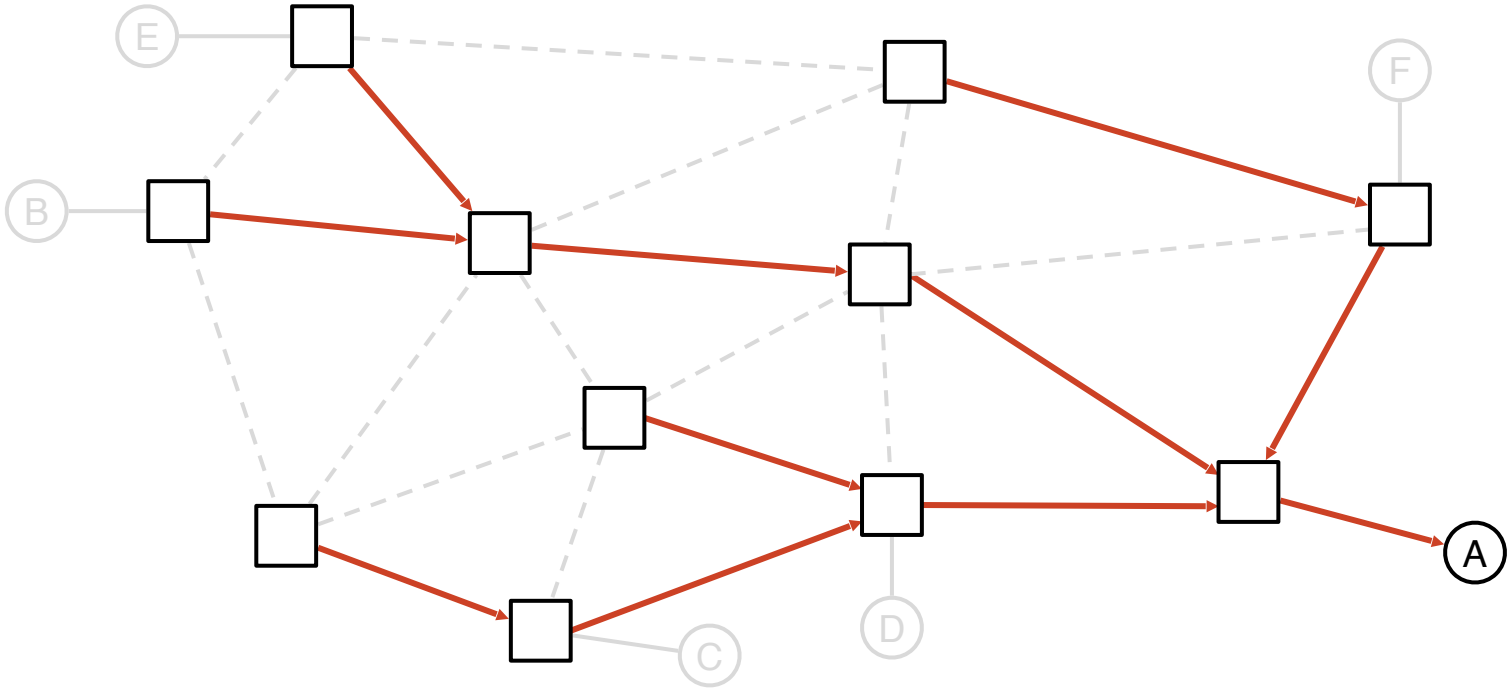
1. Pick a single destination. We'll pick A.



2. For each router, draw an arrow to the next-hop.

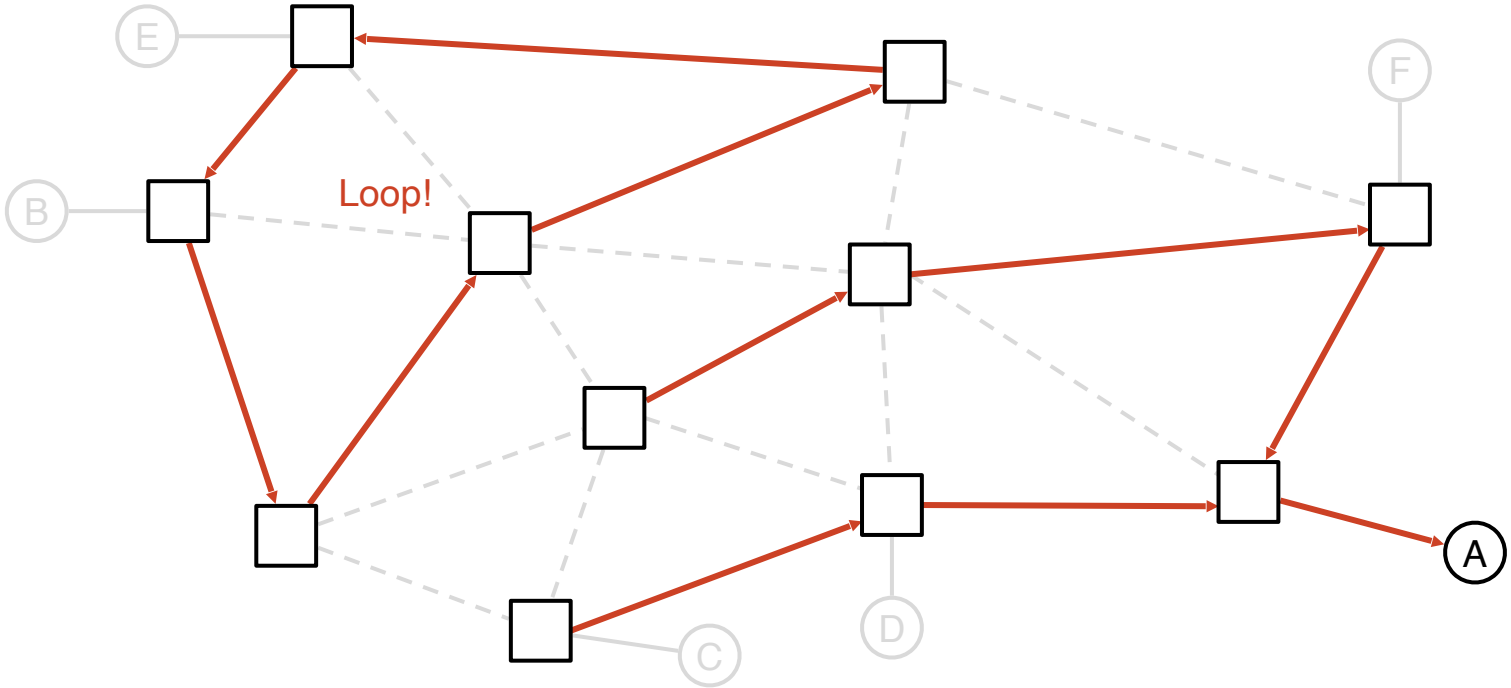


3. Delete all links with no arrows.
4. State is valid for A! Spanning tree where all edges point toward the destination.
5. Repeat for the other destinations.



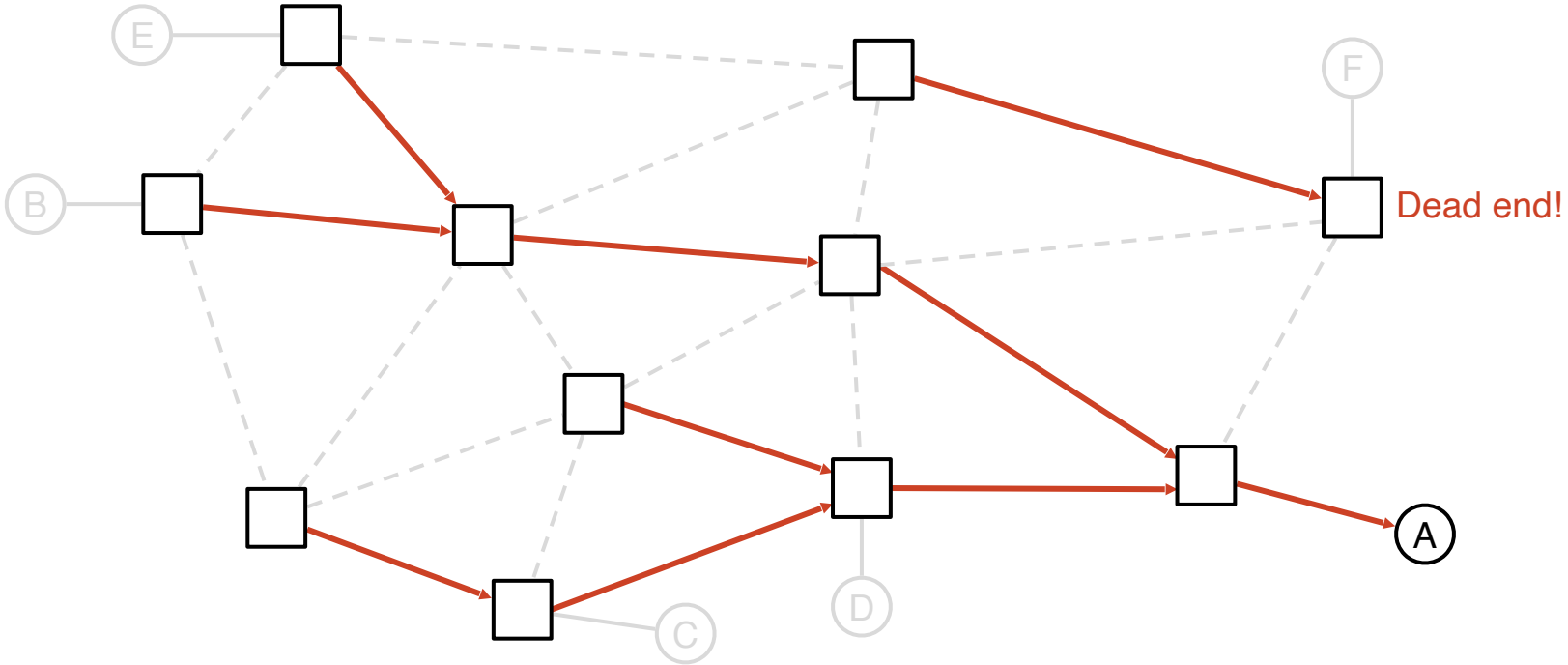
Is this valid?

No – Not a spanning tree, because there's a loop.



Is this valid?

No – Not a spanning tree, because there's a disconnected component (dead end).



Least-Cost Routing

Lecture 5, CS 168, Spring 2026

Defining the Routing Problem

- Why Do We Need Routing?
- Modeling the Network
- What Makes Routing Hard?
- Types of Routing Protocols

Routing States

- Destination-Based Forwarding
- Routing State Validity
- **Least-Cost Routing**

Sneak Peek at Routing Algorithms

- Static routes
- Routing experiment

We now know what a *valid* solution looks like (no loops, no dead ends).

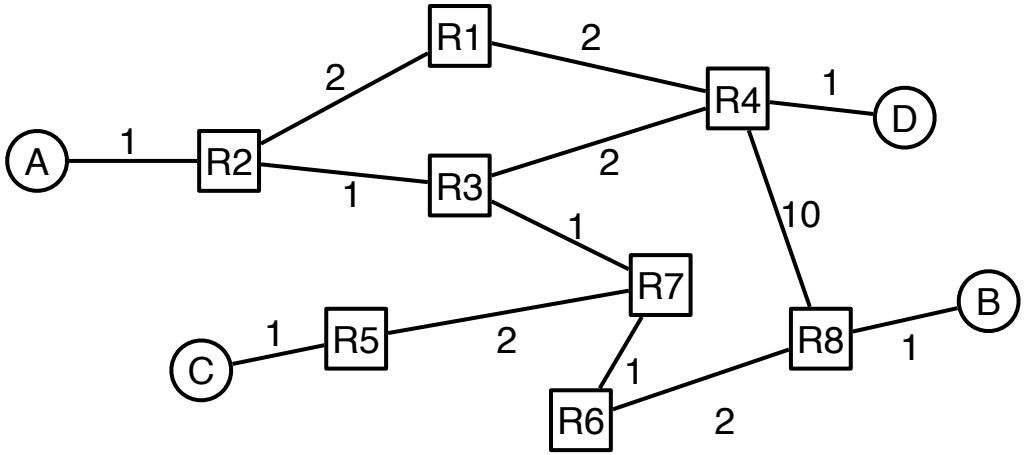
What makes a solution *good*?

Many different ways to define good, including:

- Price.
- Propagation delay.
- Distance.
- Unreliability.
- Bandwidth constraints.

Least-cost routing: Assign costs to every edge, and find paths with lowest cost.

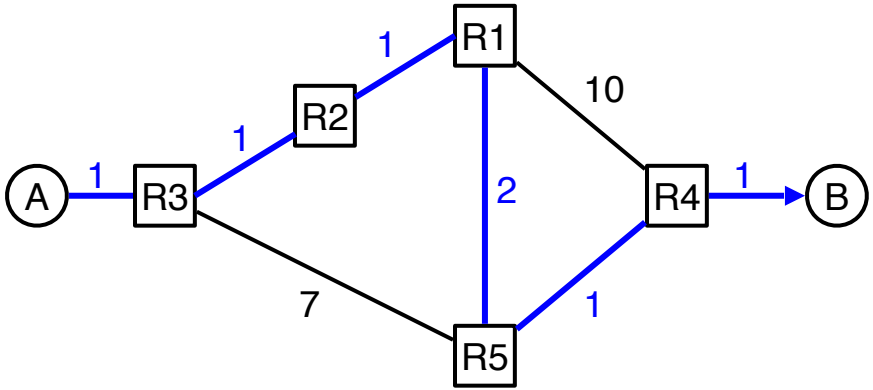
- Cost depends on the metric the operator wants to minimize.
- Costs can be arbitrary. Routing protocols don't care where the costs come from.



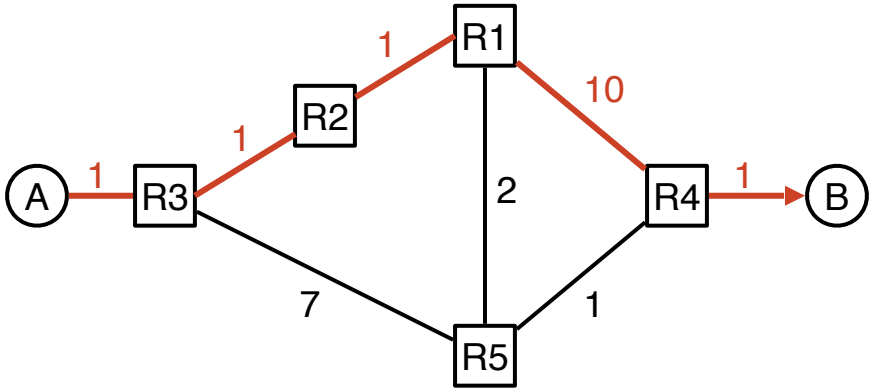
In least-cost routing, routers should forward packets such that they take the lowest-cost path to the destination.

Example: If all costs are 1, the protocol finds paths with the fewest hops.

- Usually, if edges are unlabeled, assume they have cost 1.



A → B Cost = 7



A → B Cost = 14

Where do costs come from?

- Costs are local to a router. Each router knows the cost of its own links.
- In practice, generally configured by an operator.
  - Some protocols allow for auto-configuration.

Properties of costs:

- Costs are always positive integers.
  - Can't traverse an edge and make a path cheaper.
  - Consistent with almost any practical metric you'd use.
- Costs are always symmetrical.
  - A → B costs the same as B → A.
  - Exceptions possible in theory (e.g. different upload/download bandwidth).
- These two assumptions will simplify our protocols.

Good paths (least-cost) also *valid* paths (no loops, no dead ends).

- Costs are positive, so there are no loops. They'd only increase the cost.
- Routes are still destination-based.
- Routes still form a spanning tree.

Static Routing

Lecture 5, CS 168, Spring 2026

Defining the Routing Problem

- Why Do We Need Routing?
- Modeling the Network
- What Makes Routing Hard?
- Types of Routing Protocols

Routing States

- Destination-Based Forwarding
- Routing State Validity
- Least-Cost Routing

Sneak Peek at Routing Algorithms

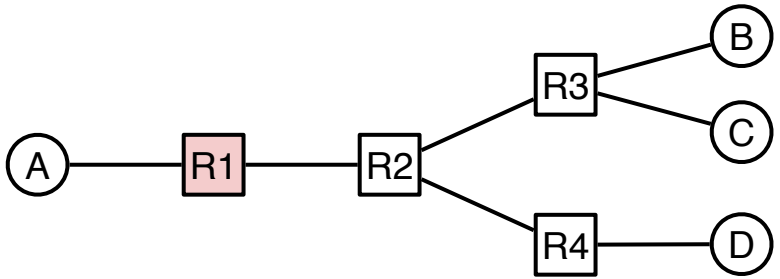
- **Static routes**
- Routing experiment

Recall: In a routing protocol, routers talk to each other to populate forwarding tables and learn paths to destinations.

Some table entries can be manually hard-coded.

- **Connected/Direct** routes let us forward to things we're connected to directly.
- These routes are created when the operator configures the router.
- No routing protocol needed for these entries.

R1's Table	
Destination	Next Hop
A	Direct
B	R2
C	R2
D	R2



Some table entries can be manually hard-coded.

- **Static** routes are hard-coded entries that we always want to be there.
- Router isn't necessarily directly connected to the destination.
- The route is static because:
  - It never changes.
  - No routing protocol used to discover it.
- Again, often manually created by an operator.

Routing game

Lecture 5, CS 168, Spring 2026

Defining the Routing Problem

- Why Do We Need Routing?
- Modeling the Network
- What Makes Routing Hard?
- Types of Routing Protocols

Routing States

- Destination-Based Forwarding
- Routing State Validity
- Least-Cost Routing

Sneak Peek at Routing Algorithms

- Static routes
- **Routing experiment**

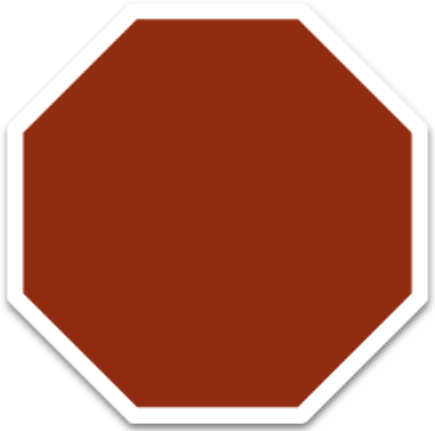
The Rules

- We're all going to work together
  - You're going to talk to your neighbors (people sitting to your left and right and in front and behind you)
  - Obviously, you may not have neighbors on all sides! (But hopefully at least one!)
- Everyone has a magic number
  - Your magic number is initially infinity
  - **You want to have as low of a magic number as possible**
- If your neighbor offers you a lower number…
  - Take it! It's now your magic number
  - Immediately offer your magic number plus one to all your neighbors
- At every point in time, try to remember who gave you your magic number
  - If someone offers same number or greater, ignore it

- Initialize:
  - Your number is infinity
  - Tell your neighbours your name and offer them your number + 1 (initially infinity)
- While true:
  - If a neighbour offers you a lower number
    - **Remember it! It is now your number.**
    - **Immediately offer your number + 1 to your neighbours.**

```
my_number = infinity
offer_to_neighbors(my_number + 1)

while True:
    offer = wait_for_offer_from_a_neighbor()
    if offer < my_number:
        my_number = offer  # I want lower
        offer_to_neighbors(my_number + 1)
```



Stop!

(But remember your number!)

Your Best Friend (for today)

- The person who gave you your current number is your best friend
  - Note that you cannot be your best friend's best friend. Sorry.

Your Best Friend (for today)

- The person who gave you your current number is your best friend
  - Note that you cannot be your best friend's best friend. Sorry.

- Did you forget who gave you your number?
  - Easy enough to figure out
  - You must have at least one neighbor whose number is yours - 1
    - Any of those could have given you your number
    - Pick any such person to be your best friend

The Test of Friendship

- If someone gives you a chocolate, give it to your best friend.
  - If your best friend gives it back to you, they must be confused.

- Let us see where the chocolates end up.

What was the point of this experiment?

- This was a distributed & asynchronous version of the Bellman-Ford algorithm
  - Can someone tell me why it worked?

What was the point of this experiment?

- This was a distributed & asynchronous version of the Bellman-Ford algorithm
  - Can someone tell me why it worked?

- What could/did go wrong?
  - Sitting too far apart (network is partitioned)
  - Forgot magic number (router failed/rebooted)
  - Mis-remembered or mis-updated magic number (implementation bug)
  - Neighbor didn't hear an update (packet drop)
  - Someone left after a neighbor accepted their offer (link failure)
  - Someone lied about their number (malicious actor)
  - Others?

We will study this algorithm and how it overcomes these issues in the next class

Routing allows us to find **paths** through the network.

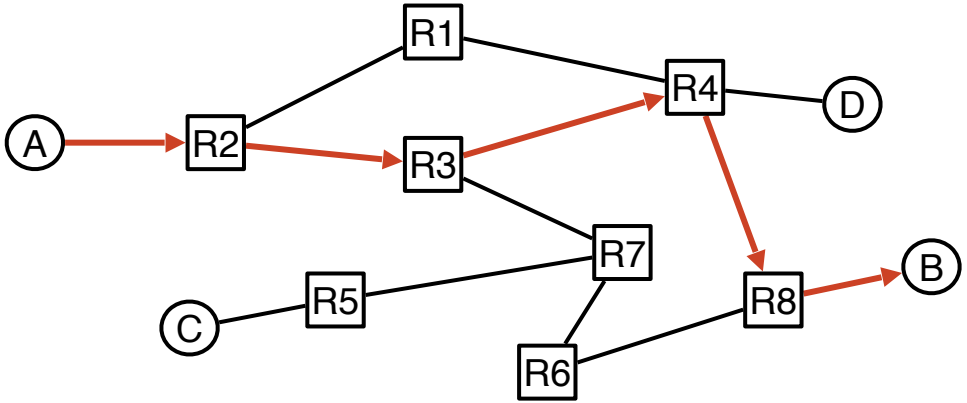
- Have to handle constant change, a non-global view, and best-effort links
- There are both intra-domain (choices!) and inter-domain routing protocols (BGP)

The Internet uses **destination-based forwarding**.

- Each router keeps a table, mapping destinations to next hops.

We need a valid directed delivery tree with **no dead ends and no loops**!

Static or direct routes can be hard coded!



R3's Table (Conceptual)	
Destination	Next Hop
A	R2
B	R4
C	R7
D	R4