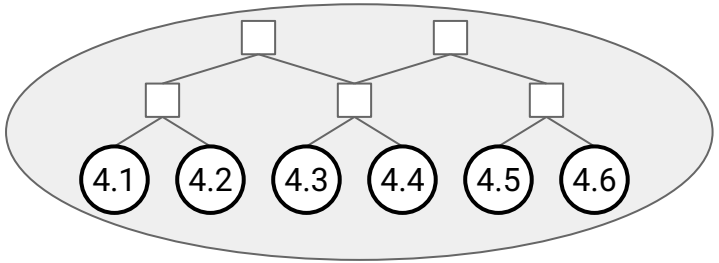


Lecture 6 (Routing 3)

Link-State Protocols, IP Addressing

CS 168, Spring 2026 @ UC Berkeley

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

Link-State Protocols

Lecture 5.3, Spring 2026

Link-State Protocols

- **Overview**
- Computing Paths
- Learning Graph Topology

IP Addressing

- Hierarchical Addressing
- Assigning Addresses
- Writing Addresses
- Aggregating Routes
- IPv6 Changes

Routing Protocols – Roadmap

Routing protocols can be classified by:

- *Where* they operate. (Intra-domain or inter-domain.)
- *How* they operate. (Distance-vector, link-state, or path-vector.)

Today, we'll look at **link-state** protocols.

- Very common as an Interior Gateway Protocol.
- Major examples:
  - IS-IS (Intermediate System to Intermediate System).
  - OSPF (Open Shortest Path First).

	Intra-domain (Interior Gateway Protocol)	Inter-domain (Exterior Gateway Protocol)
Distance-vector	RIP	–
Link-state	**IS-IS, OSPF**	–
Path-vector	–	BGP

Link-State Protocols: Definition

Link-state protocols:

1. Every router learns the full network graph.
2. Then, each router runs a shortest-path algorithm on the graph to populate the forwarding table.

How do routers learn this?

What algorithm do we run?

Distance-vector:

- **Local data**: Each node only knows about part of the network.
- **Global (distributed) computation**: Each node computes part of the solution, working with other nodes.

Link-state:

- **Global data**: Each node knows about the full network graph.
- **Local computation**: Each node computes the full solution by itself.

Global Data

Suppose R3 has learned the full network graph. How do routers learn this?

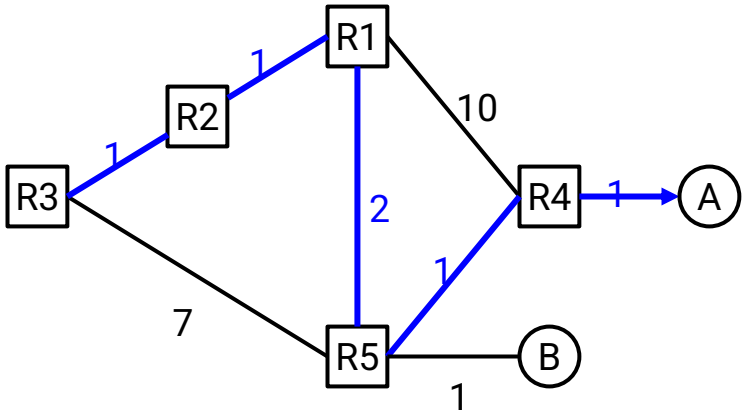
- We know all the links, their status (up or down?), and their costs.
- We know about all the destinations.

What algorithm do we run?

R3 can run a graph algorithm to compute paths.

- We can populate the forwarding table with the next-hop (in the computed path).

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



Computing Paths

Lecture 5.3, Spring 2026

Link-State Protocols

- Overview
- **Computing Paths**
- Learning Graph Topology

Link-State Algorithms

What graph algorithm do we run to compute paths?

Any single source shortest-path algorithm will work.

- Need to find shortest paths from a single source (the router) to every host.

Some possible choices:

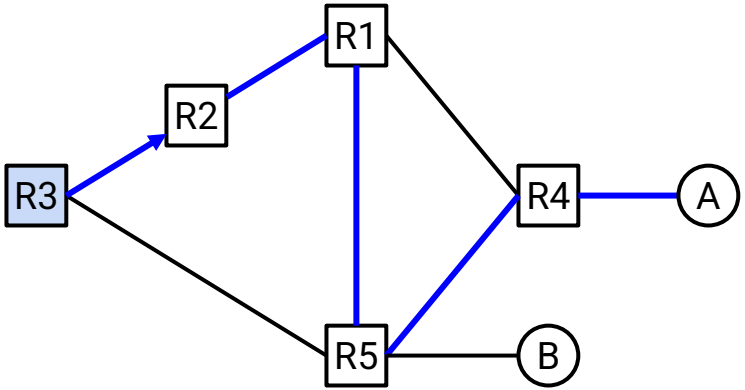
- Bellman-Ford (the original serial version).
- Dijkstra's algorithm.
- Breadth-first search.
- Dynamic shortest path.
- Approximate shortest path.
- Parallel single-source shortest path.

Ensuring Consistency

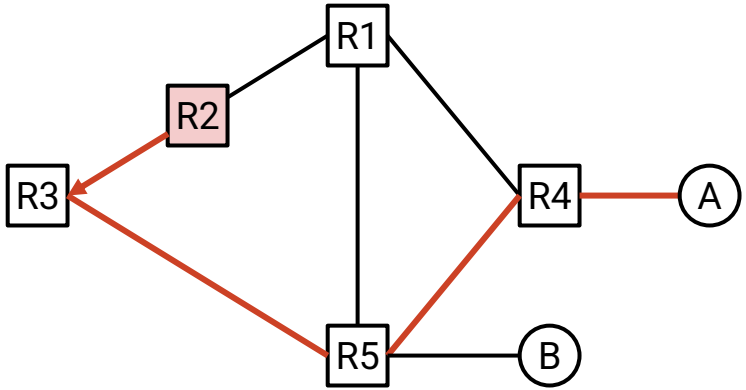
Each router can only influence its next hop.

No guarantee that routers compute the same paths!

We have to ensure that every router is using a "compatible" approach.



R3 forwards to R2.



R2 forwards to R3.

Ensuring Consistency

Requirements for routers to produce valid, compatible decisions:

1. Everyone agrees on the network topology.
2. Everyone is minimizing the same cost metric.
3. All costs are positive.
4. All routers use the same tie-breaking rules.

Routers don't necessarily need to use the same shortest-path algorithm, as long as they follow these rules.

- In practice, easier to have everyone use the same algorithm.

Learning Graph Topology

Lecture 5.3, Spring 2026

Link-State Protocols

- Overview
- Computing Paths
- **Learning Graph Topology**

IP Addressing

- Hierarchical Addressing
- Assigning Addresses
- Writing Addresses
- Aggregating Routes
- IPv6 Changes

Steps of Learning Network Graph

Link-state protocols:

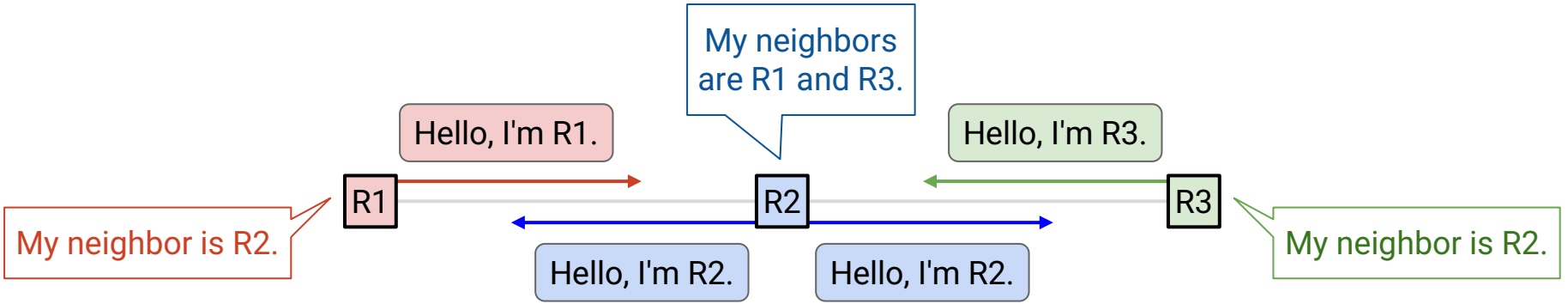
1. Every router learns the full network graph.
   - Step 1A: Discover my neighbors.
   - Step 1B: Tell everybody about my neighbors.
2. Then, each router runs a shortest-path algorithm on the graph to populate the forwarding table.

How do routers learn this?

Steps of Learning Network Graph (1/2) – Learn Neighbors

How do we discover who is adjacent to us and their identity? Say hello!

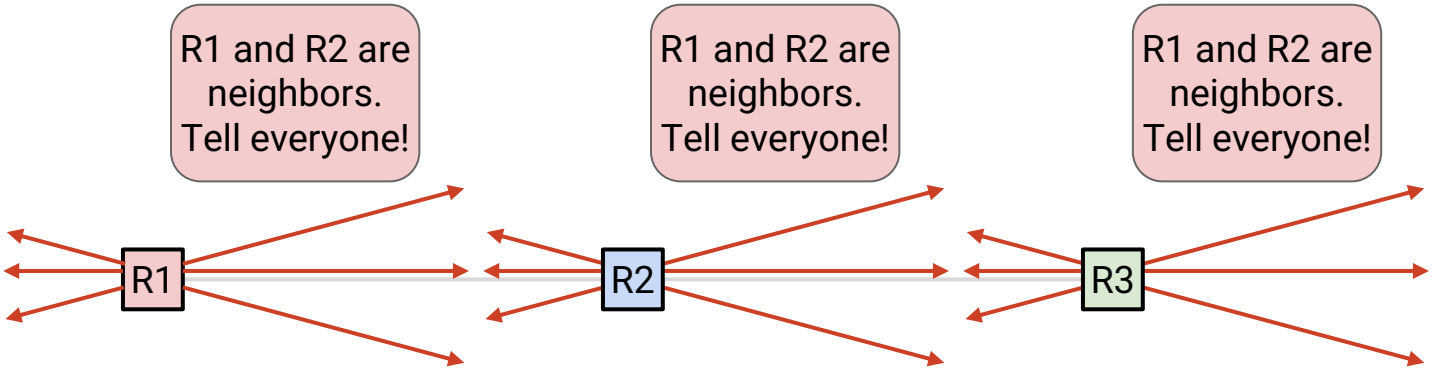
- Routers periodically send *hello* messages to their neighbors.
- If they stop saying hello, assume that they disappeared.
- This helps us learn about our direct neighbors, but not the whole network.
  - R1 does not know about the rest of the network (e.g. R3).



Steps of Learning Network Graph (2/2) – Propagate Neighbor Information

How do we learn about the rest of the network, beyond our neighbors?

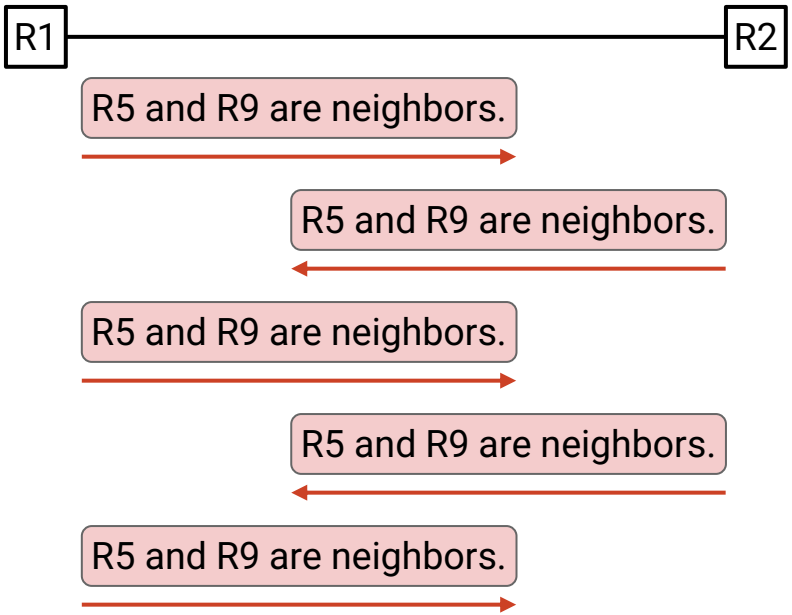
- Solution: **Flood** information across the network.
- When local information changes, send it to everyone.
- When you receive information from your neighbor, send it to everyone.



Avoiding Infinite Flooding

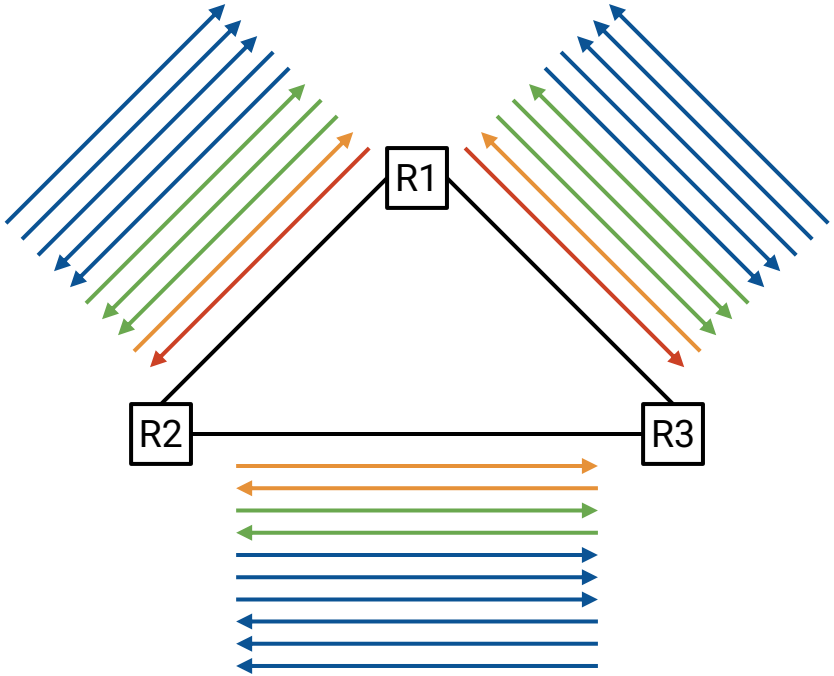
Flooding: When you receive an update, send it to everybody.

We have to be careful of the same update being sent repeatedly.



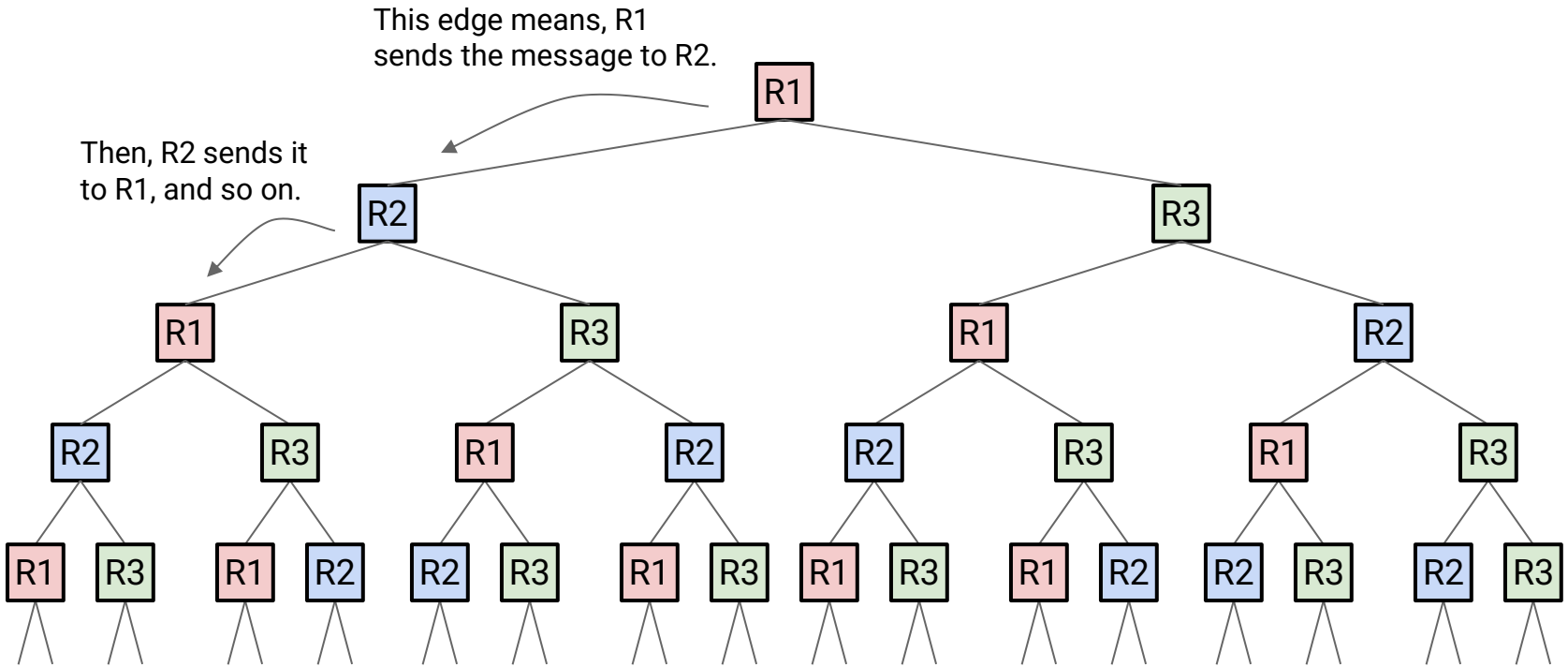
Avoiding Infinite Flooding

Amplification: When there's a loop, copies of the same message get multiplied.



Avoiding Infinite Flooding

Amplification: When there's a loop, copies of the same message get multiplied.



Avoiding Infinite Flooding

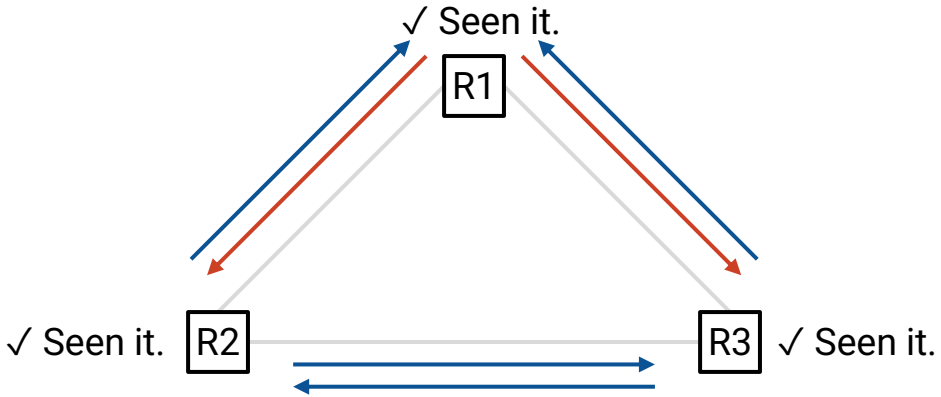
Problem: Naive solution (send to all neighbors) causes amplification.

Solution:

- When local information (about yourself) changes, send it to all neighbors.
- When you receive a packet from a neighbor, send it to all neighbors...
- ...unless you've already seen the packet before.

To identify packets you've seen before, add a timestamp.

- Or some other unique identifier, e.g. strictly increasing sequence numbers.



Ensuring Reliability

The network is still best-effort. Updates could get dropped.

- Recall: We need all routers to agree on topology, or else paths might be invalid.

Solution: Periodically re-send the update.

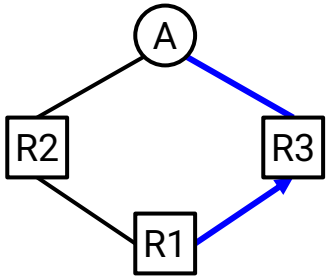
Ensuring Convergence

The network could change. What happens if a link goes down?

- Wait for routers to detect the failure.
- Wait to flood new information.
- Wait to recompute paths.

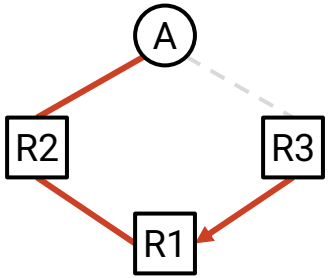
While waiting for convergence, the routing state might be invalid.

- Dead ends and loops. Packets using longer routes.



Link is down, but R1 doesn't know!

R1 forwards to R3.



R3 knows about the link failure!

R3 forwards to R1.

Link-State vs. Distance-Vector

Link-state is relatively simple. All the complexity is in the details!

- Everyone floods link/destination information.
- Everyone has a global map of the network.
- Everyone independently computes next-hops.

Why might we want to use link-state over distance-vector?

- Link-state gives routers more control over the path they choose.
  - Distance-vector: We have to trust what our neighbor says, and we don't know the path they're using.
- Link-state might be better at converging.
  - Distance-vector: Wait for neighbor to recompute and re-advertise path.
  - Link-state: Flood information before recomputing.

Real networks often use a combination of path/distance-vector and link-state.