Modified Flood

deactivated = Set()
on recv(pkt, ifin):
    if ifin in deactivated:
        return
    for i in interfaces:
        if (i != ifin and
            i not in deactivated):
            send(pkt, i)

Consider
alice $\xrightarrow{P}$ bob
bob $\xrightarrow{R}$ alice

What nodes process each packet?

Consider
alice $\xrightarrow{P}$ bob
bob $\xrightarrow{R}$ alice

What nodes need to process each packet?
This lecture: have fewer nodes process each packet.

Why?

- **Scalability**
  - Each link has a limit on the rate at which it can carry packets

Challenge?

How to get switches to deliver without flooding?

**Learning**

Alice → Bob

```
alice  P  bob
```

Bob → Alice

```
bob  R  alice
```

```
h4
h5
h3
h2
h1

A
B
C
D
E
```
table = dict()
on recv(p, iface):
    table[p.src] = iface
    if p.dst in table:
        send(p, table[p.dst])
    else:
        flood(p, iface)
How to handle hosts moving?

Now for something different.
Can we forward packets better?

Generalized Forwarding:

```
table = dict()
on recv(pkt, ifin):
    if pkt.dst in table:
        send(pkt, table[pkt.dst])
    else:
        # Drop the packet
```

Problem: How to populate `table`?
Q: What paths to choose? Shortest?

How to compute shortest paths?

Two options:

* Use some extension to the spanning tree protocol. - Lab 2, go over this later

* Get all nodes to learn the entire graph and use Dijkstra's. - Go over this first
Protocol: Learn the graph

- Learn neighbors

  - Learn neighbors

Packet:

on boot:

on recv: