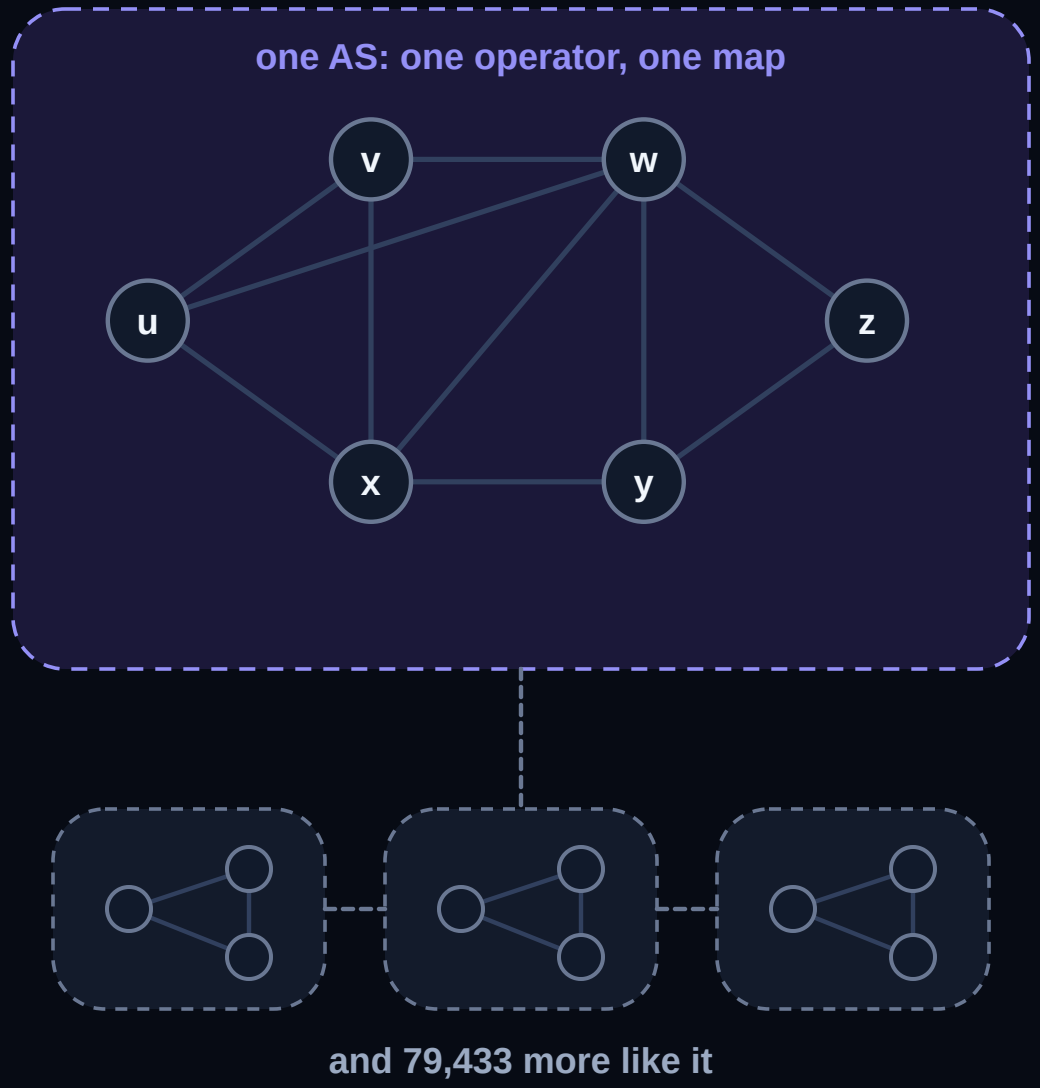
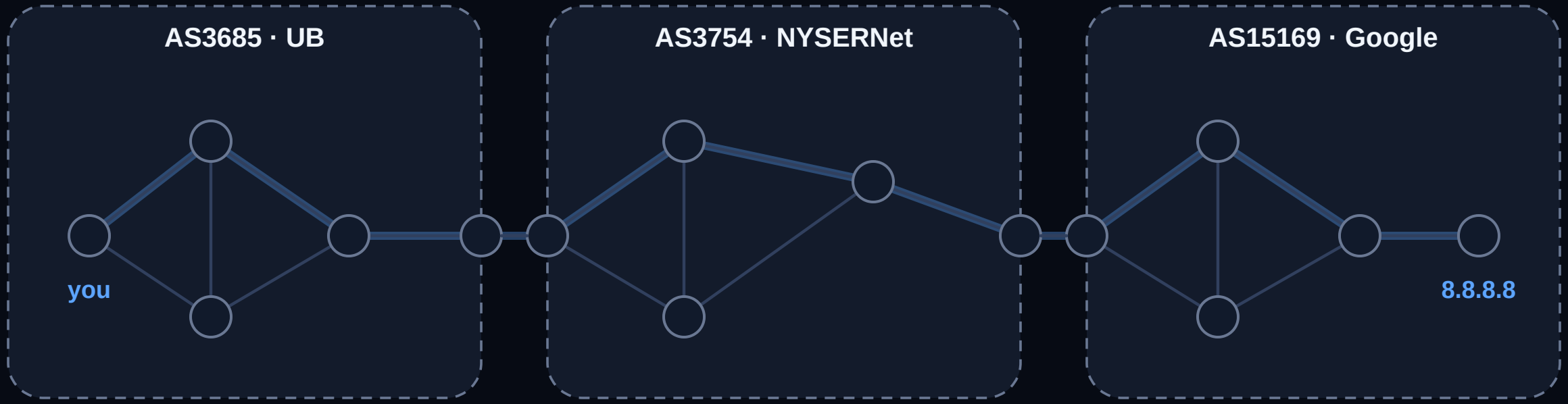
Routing inside one network

Link state and distance vector ran on six routers. The Internet is tens of thousands of networks, each run by its own operator. This lecture splits routing into two levels and follows link state into one real network: OSPF, its areas, and the link weights that decide where the traffic goes.

Yaxiong Xie



One packet from UB to Google crosses three networks



An illustrative path. Run `traceroute 8.8.8.8` from campus to see the real one.

Each network routes its own part, and none shows the others its map.

Four topics, each from its own problem

01



Autonomous systems

the Internet is
not one graph

02



OSPF

link state, run
for real in one AS

03



Areas

keep the flood
inside one part

04



Link weights

the costs decide
where traffic goes

Each one: why it is needed, how it works, what it costs.

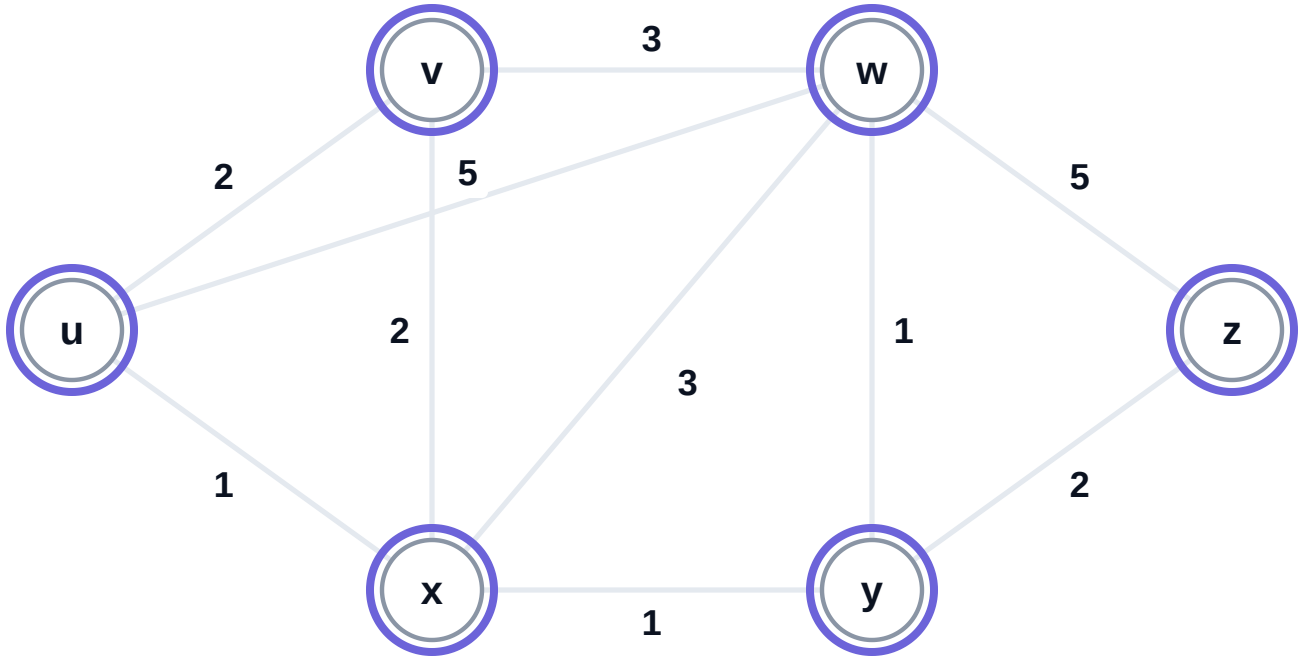
Can the whole Internet be one link-state graph?

01

Autonomous systems

Link state and distance vector assumed one graph that every router computes on. Try that with every router on the Internet.

Link state: every link reaches every router



every router ends with

x's LSA: its 4 links

+ the other five routers' LSAs

the same map at all six

The link-state lecture, point 02. An LSA (link-state advertisement) lists one router's own links and their costs.

Every link, and every change to it, reaches every router. Fine for six.

How big would one Internet-wide graph be?

79,433

networks

each run by its own operator

1,078,052

destination prefixes

one row each in a core router

if it were one graph

≈ 1,000,000

routers

≈ 2,000,000

links

Networks and prefixes: bgp.potaroo.net, 24 Sep 2026. Routers and links: an assumed order of magnitude, not a measurement.

About a million routers and two million links, if the Internet were one graph.

Could one router store and compute that map?

store the map

2,000,000 links × 12 bytes

= 24 MB

24 MB one router's memory: 8 GB

compute the paths

Dijkstra with a heap: $(n + |E|) \cdot \log_2 n$

$3{,}000{,}000 \times 20 = 6 \times 10^7$ steps

≈ 60 ms

at 1 ns per step

12 bytes per link: two router IDs and a cost. The heap version of Dijkstra is from the link-state lecture, point 03.

Yes. A 24 MB map and a 60 ms Dijkstra are both small.

How often would that map change?

assume each link goes down or comes back once a day

2,000,000 links ÷ 86,400 s ≈ 23 changes per second

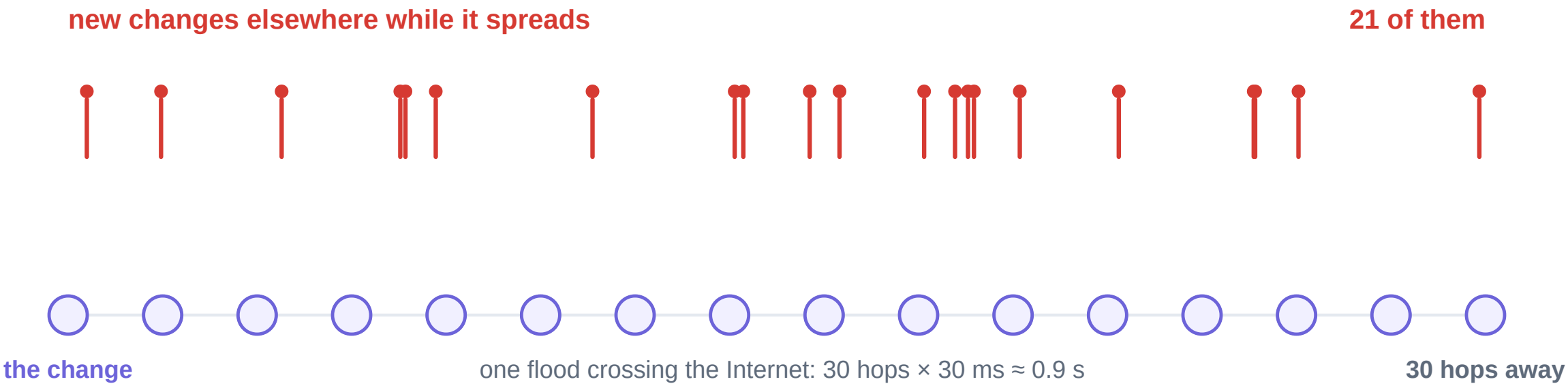


each one: a new LSA to a million routers, and a new Dijkstra at each

Once a day per link is an assumption. Make it once a week and it is still three changes every second.

About 23 changes every second, each one flooded to every router.

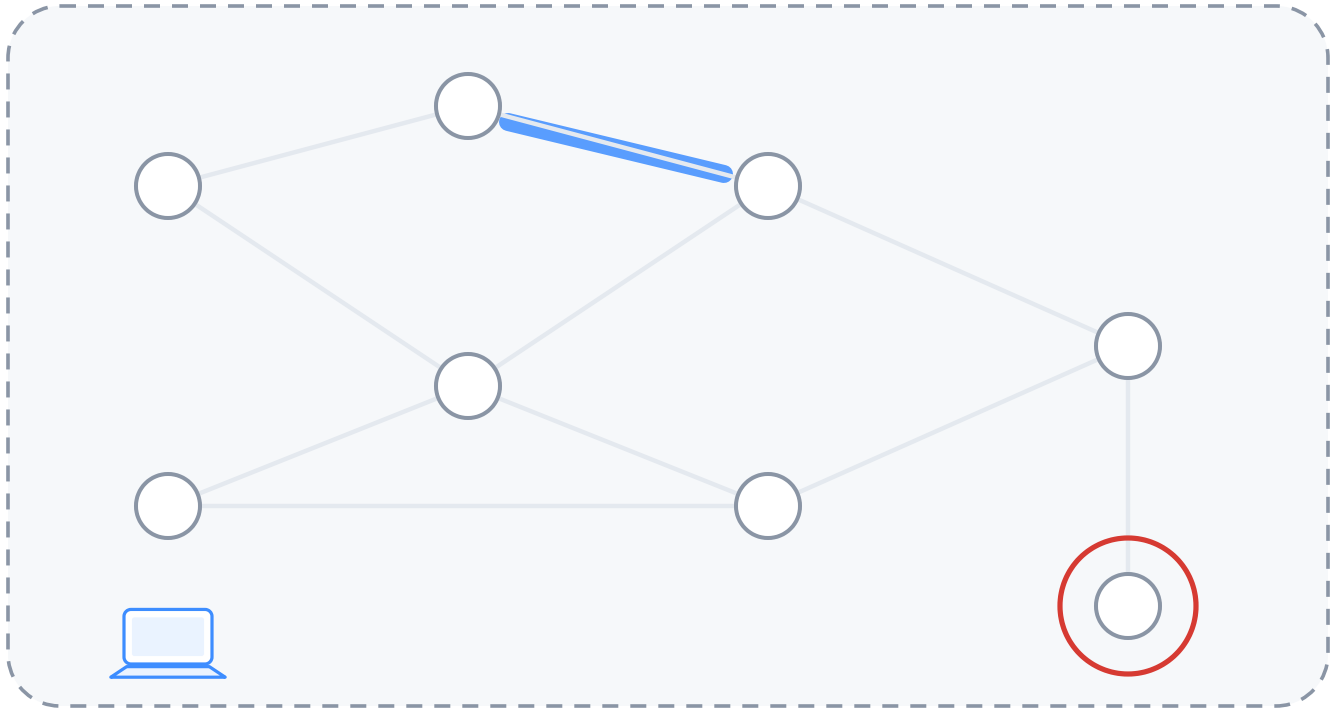
Would every router ever hold the same map?



Assumed: 30 hops across, 30 ms per hop. The link-state lecture: tables agree only once flooding has finished and nothing has changed since.

No. One flood takes about a second, and about 20 more changes start meanwhile.

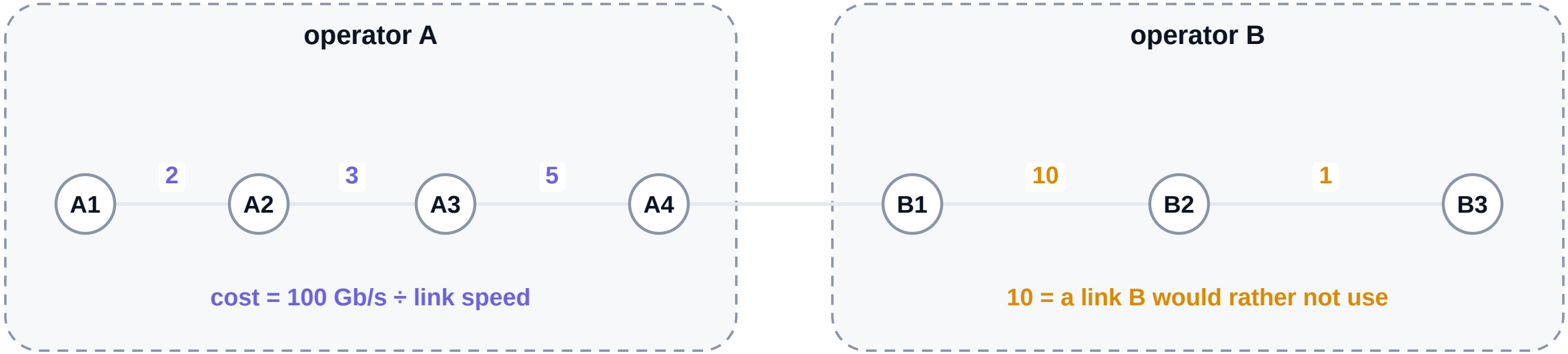
Would an operator publish its map?



- **where the capacity is**
  400 Gb/s here, 10 Gb/s there
- **who the customers are**
  and where they connect
- **where it breaks**
  one link holds up that corner

No. The map shows its capacity, its customers, and where it breaks.

Can two operators' costs be added?

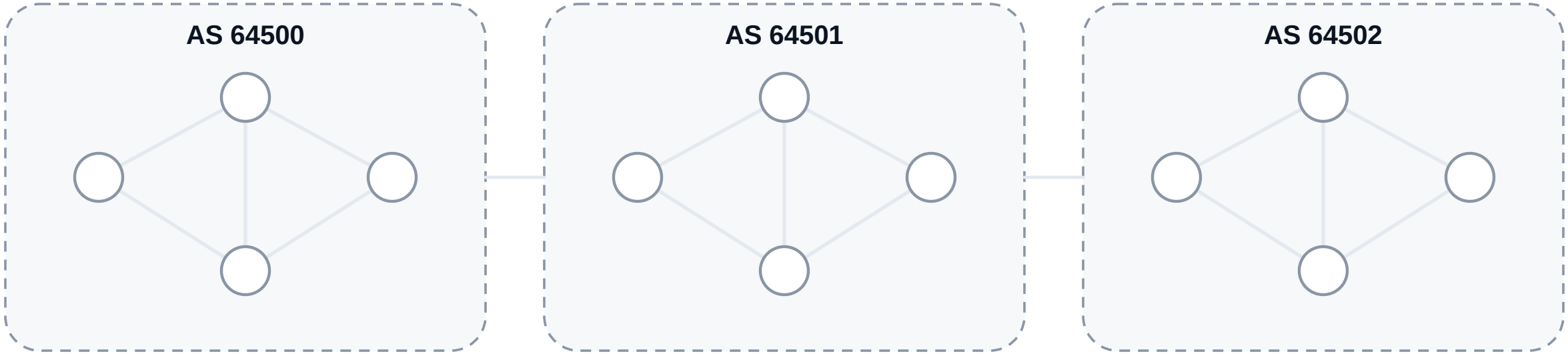


~~2 + 3 + 5 + 10 + 1 = 21~~

a sum of two different units

No. Each operator's costs mean what it wants them to mean.

Split the Internet by who runs the routers



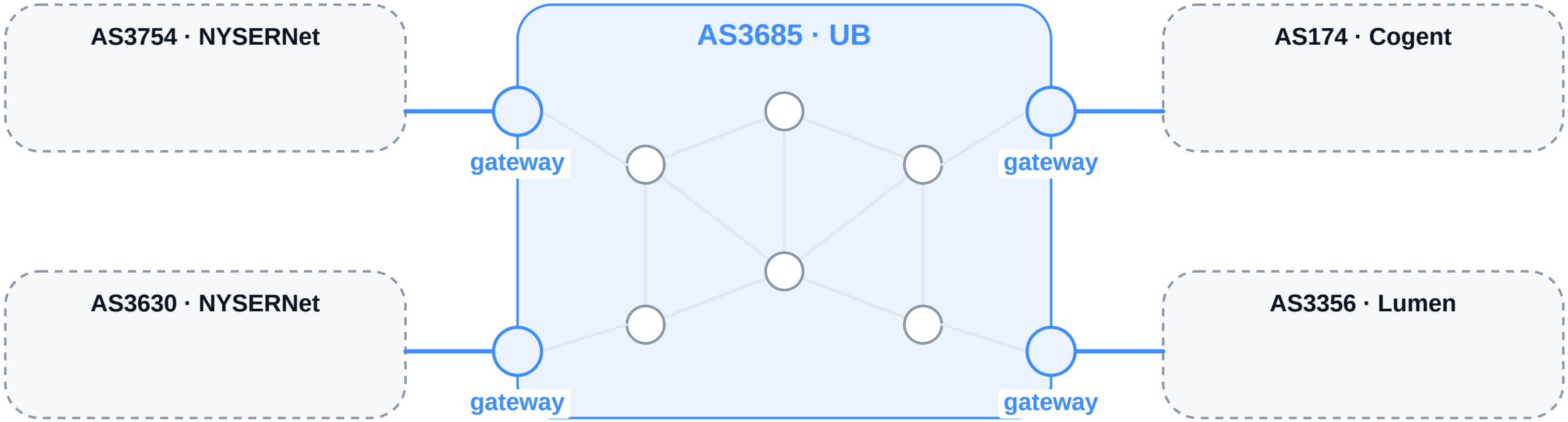
autonomous system (AS): the routers one operator runs, with one AS number

AS numbers 64500 to 64511 are reserved for examples (RFC 5398).

Each AS keeps its own map and its own costs. There are 79,433 of them.

1 Autonomous systems | 2 OSPF | 3 Areas | 4 Link weights

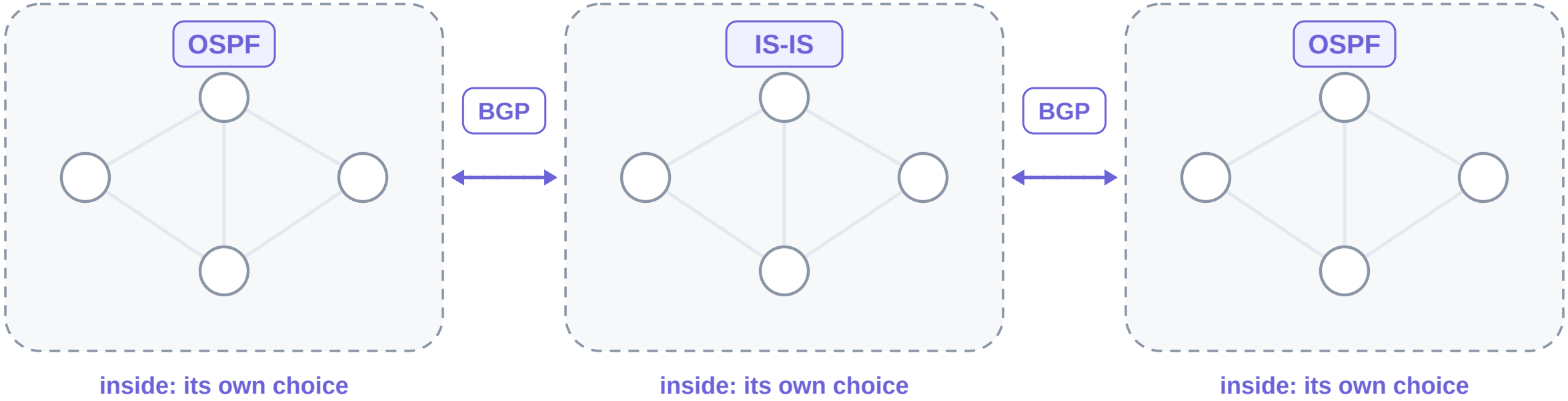
UB is one AS: AS3685



AS3685 is the number in ARIN’s record for 128.205.0.0/16 (the IP lecture); it announces 9 prefixes. Neighbours: bgp.he.net, 24 Sep 2026. Inside: illustrative.

UB runs its own routers. Its gateway routers link it to four other ASes.

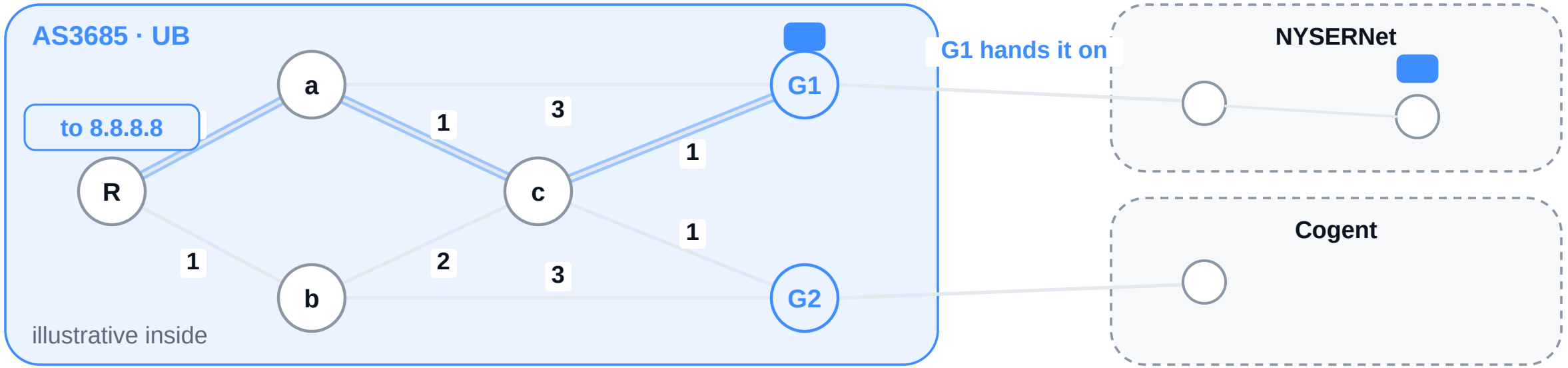
Two problems: inside an AS, and between ASes



Inside: OSPF and IS-IS are link state, RIP is distance vector. Between ASes, BGP is the next lecture.

Inside, each AS picks its protocol and costs. Between ASes, all speak BGP.

How does a UB router send to 8.8.8.8?



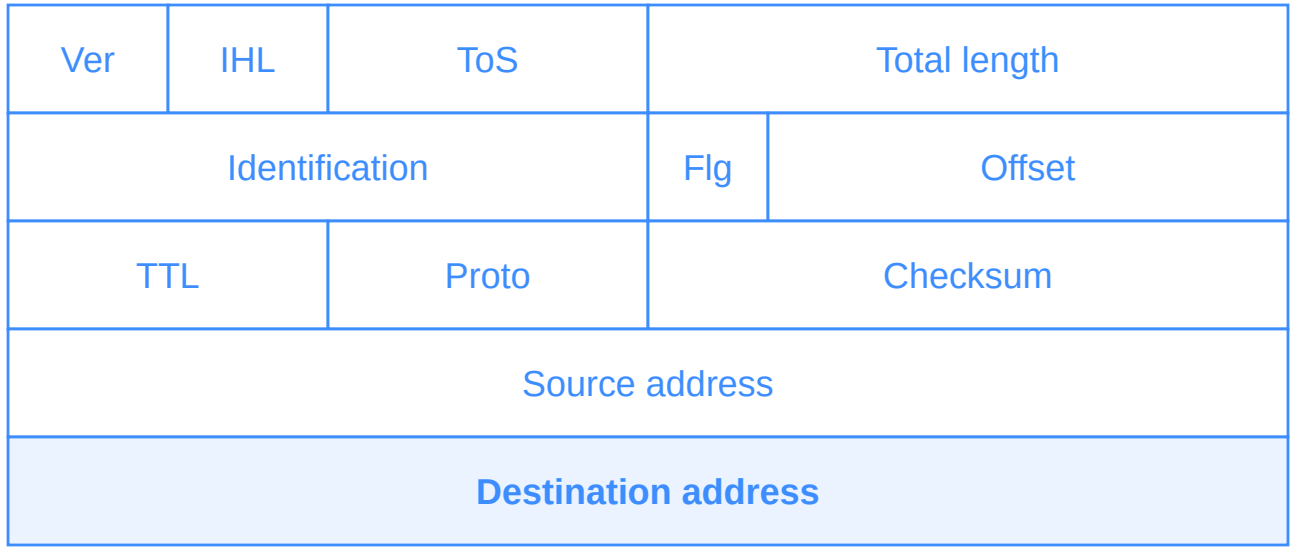
prefix	R sends it
128.205.36.0/22	link to b
0.0.0.0/0	toward G1

inside: from OSPF

outside: the way to a gateway

Inside, R only needs the way to a gateway. The gateway knows the way out.

Where is the AS number in the packet?



the IPv4 header: 20 bytes, every field

AS number?

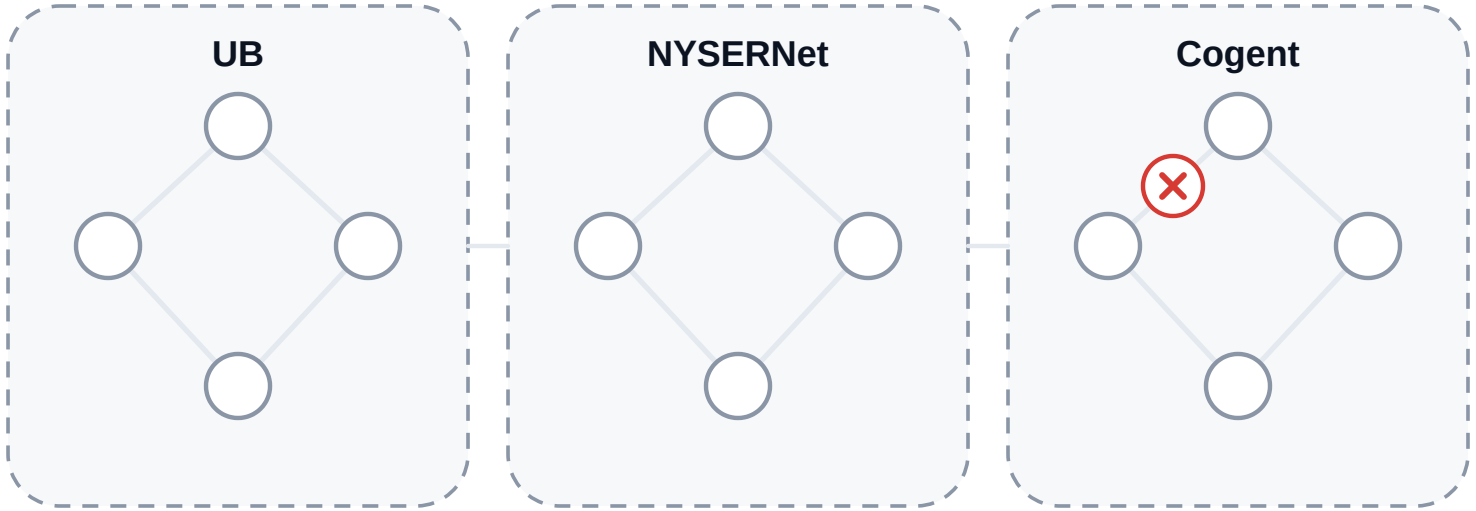
no field for it

routing messages

between ASes carry them

Nowhere. Forwarding still reads only the destination address.

What does splitting into ASes give?



a change stays in its AS

and each AS picks its own protocol and costs

still open

which gateway, and what lies beyond it: next lecture, BGP

A change inside one AS stays inside it. The rest of today stays inside one AS.

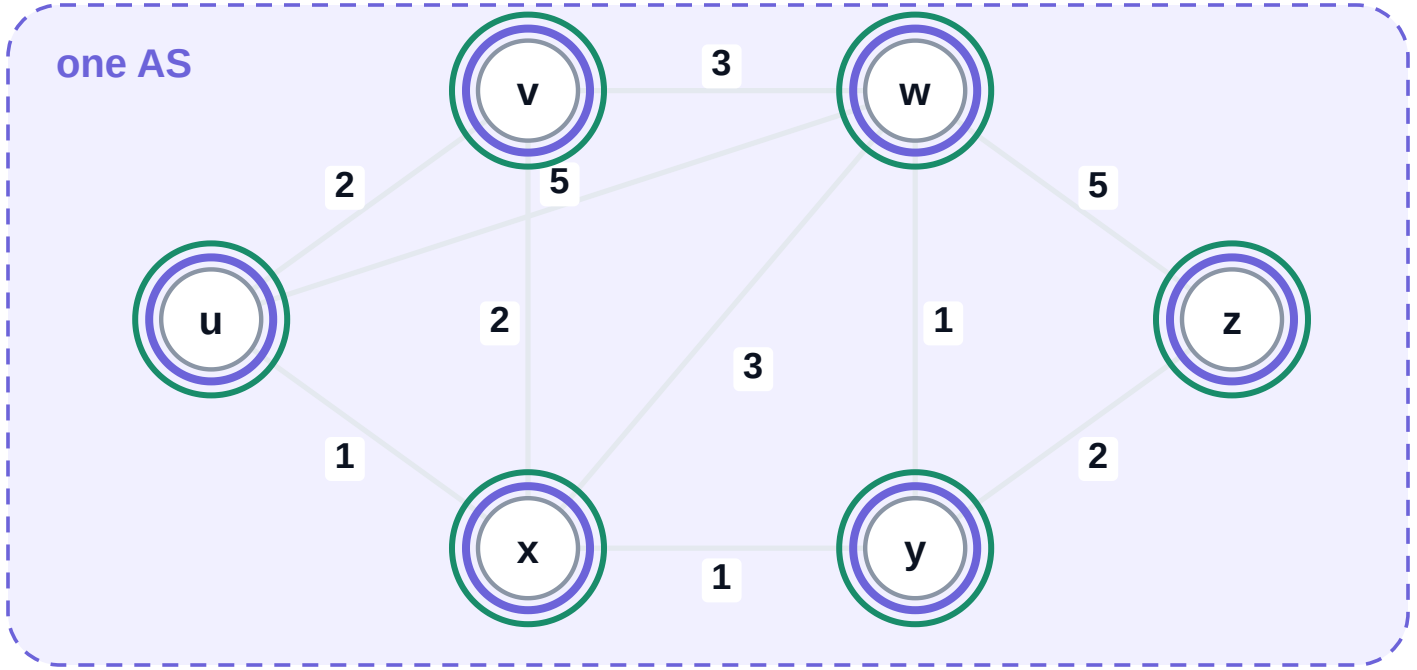
Inside one AS, link state fits. What does running it take?

02

OSPF

The link-state lecture assumed every router knows its own links, starts with the whole map, and tells the truth. A real network gives none of that for free.

OSPF: the link-state lecture, inside one AS



OSPF

each router floods its **own links (an LSA)**

every router holds the **same database**

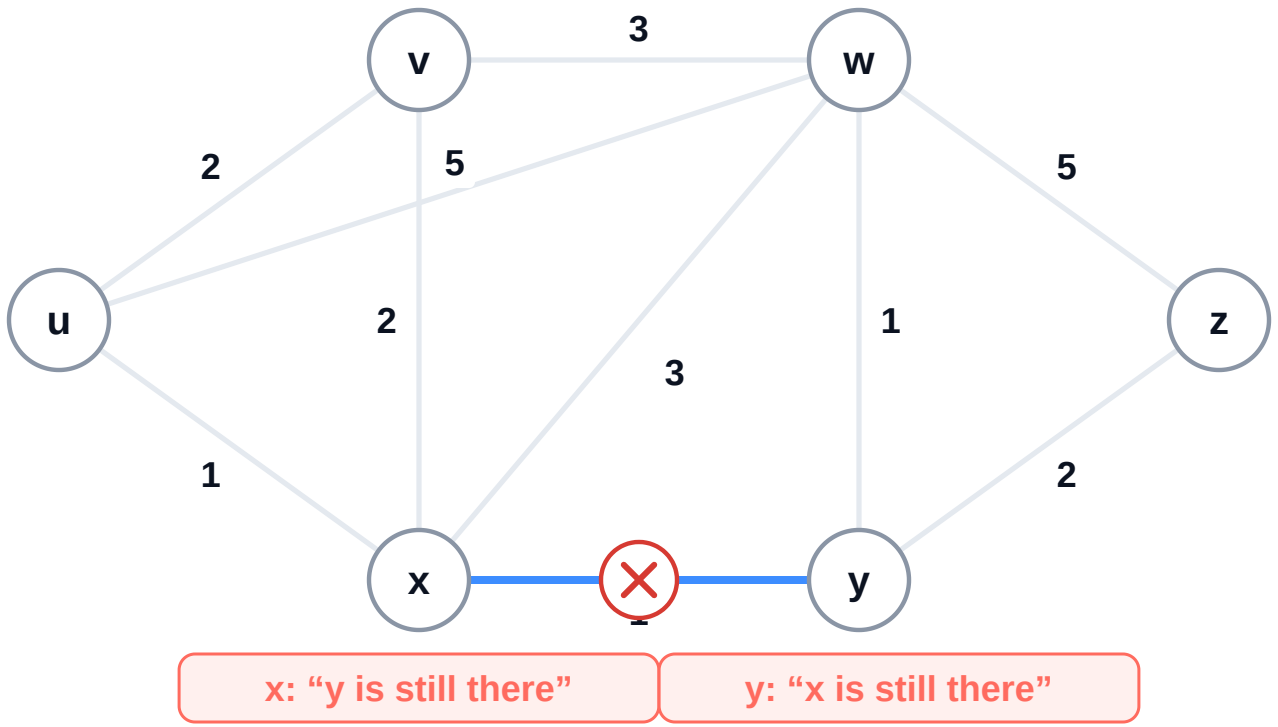
every router runs **Dijkstra itself**

OSPF version 2 is RFC 2328; its messages ride directly on IP (protocol 89). Many large ISPs run IS-IS instead: the same idea, other message formats.

Flood your own links, keep the same database, run Dijkstra. Nothing new yet.

1 Autonomous systems | 2 OSPF | 3 Areas | 4 Link weights

A fibre is cut. Who sends the news?



the fibre sends nothing

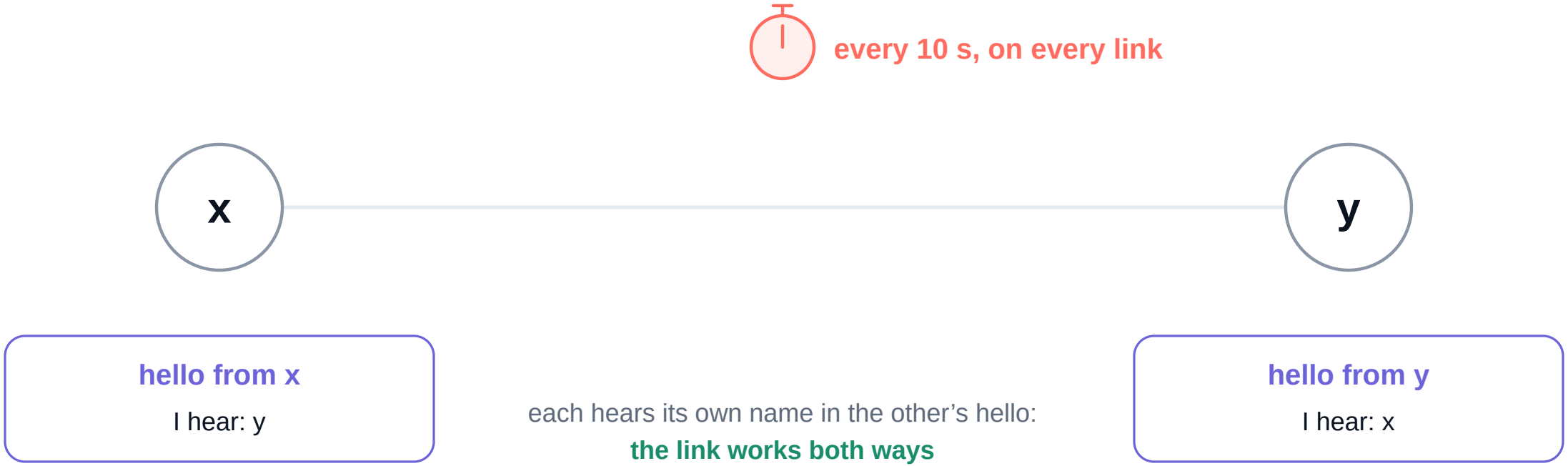
no LSA leaves x or y

x keeps sending to y

every packet for y and z
that x forwards is lost

Nobody. A cut fibre is silent, so x must notice the silence itself.

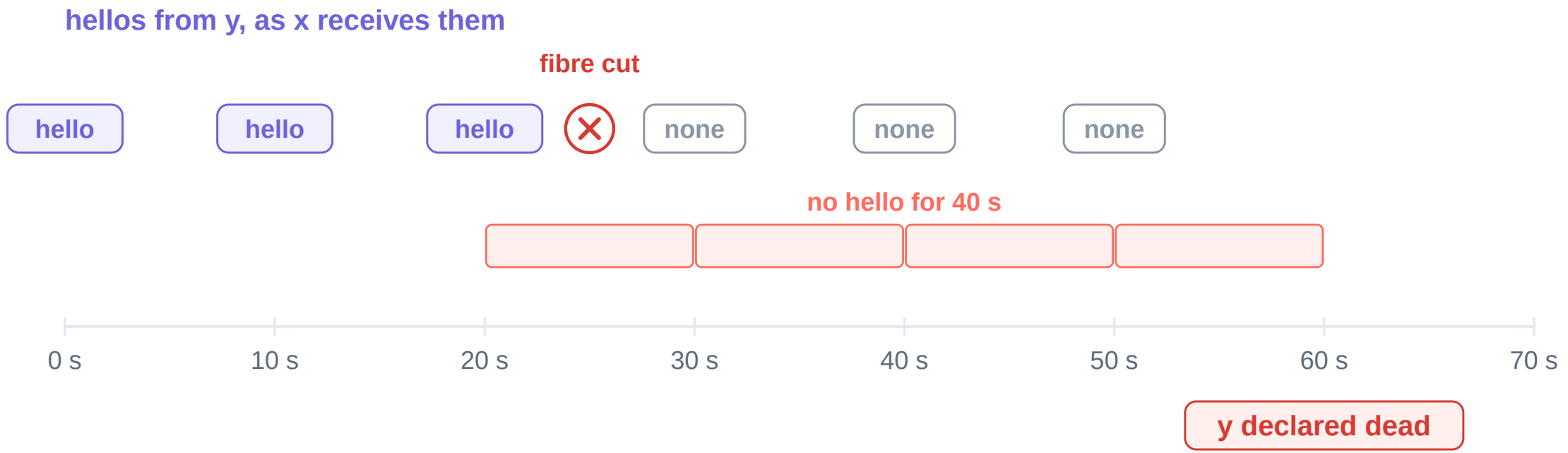
Say hello: every 10 s, on every link



Each router says hello on each link every 10 s, and lists who it hears.

1 Autonomous systems | 2 OSPF | 3 Areas | 4 Link weights

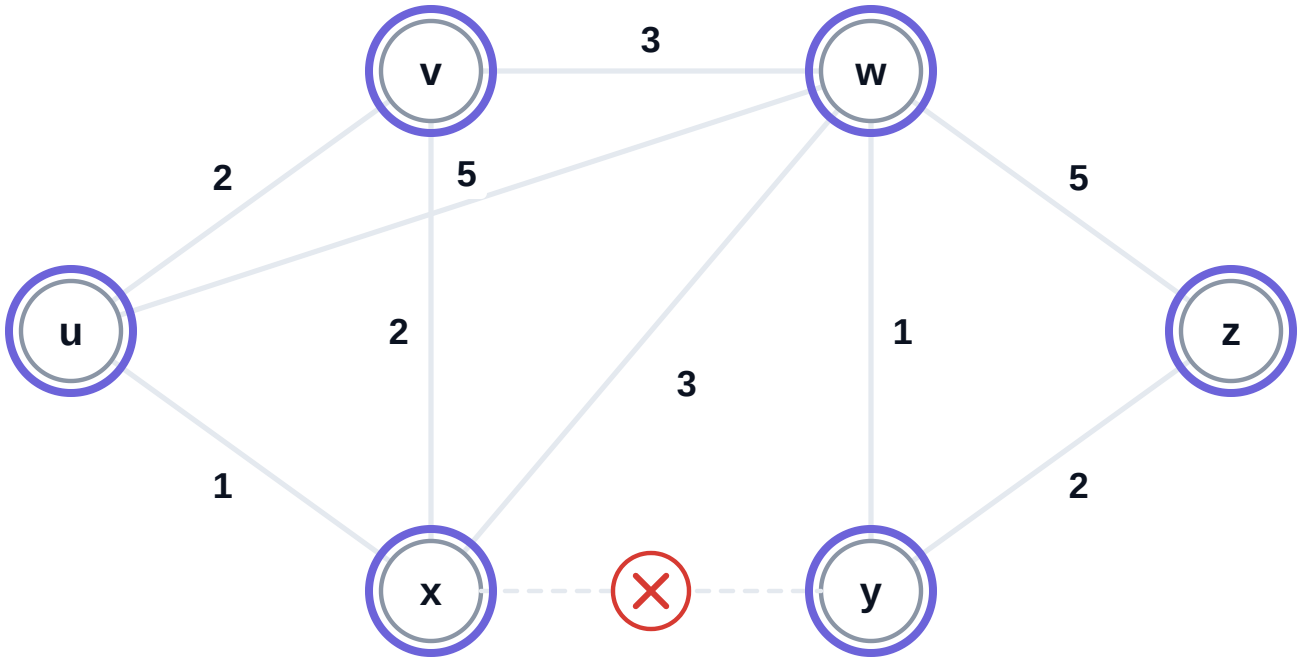
When is a silent neighbour declared dead?



40 s is OSPF's default dead interval: four hello intervals. Many networks add BFD, a faster hello that notices in well under a second.

After 40 s without a hello. One lost hello is normal; four in a row is not.

Once y is declared dead, what does x send?



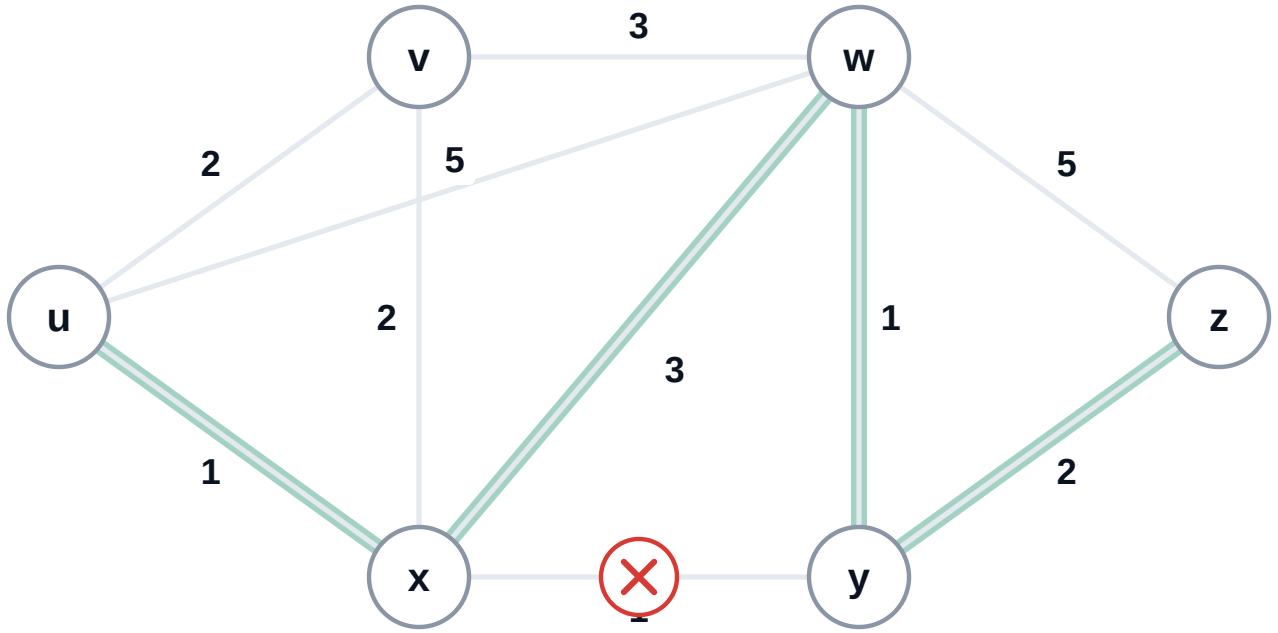
x's LSA
seq 12
(u, 1)
(v, 2)
(w, 3)
~~(y, 1)~~

x's new LSA
seq 13
(u, 1)
(v, 2)
(w, 3)

y does the same from its end

A new LSA without the dead link, with a higher sequence number, flooded.

Every router reruns Dijkstra. Where does u's path go?



before: u–x–y–z

cost 1 + 1 + 2 = 4

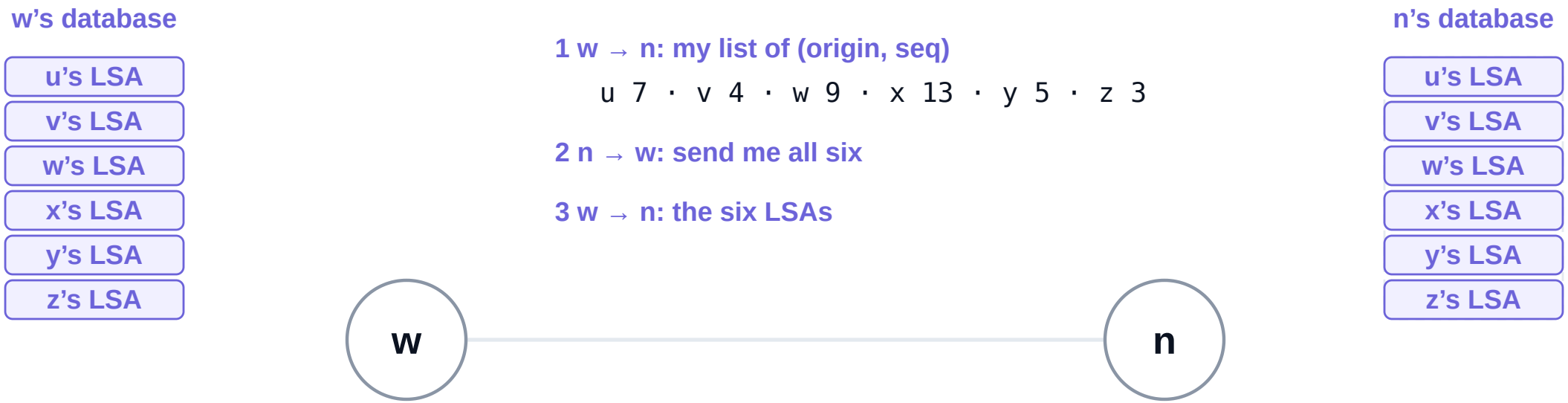
after: u–x–w–y–z

cost 1 + 3 + 1 + 2 = 7

Until u has x's new LSA and has rerun Dijkstra, its packets for z still head into the cut (the link-state lecture, point 04).

From u–x–y–z, cost 4, to u–x–w–y–z, cost 7.

A new router has an empty map. Must it wait 30 min?

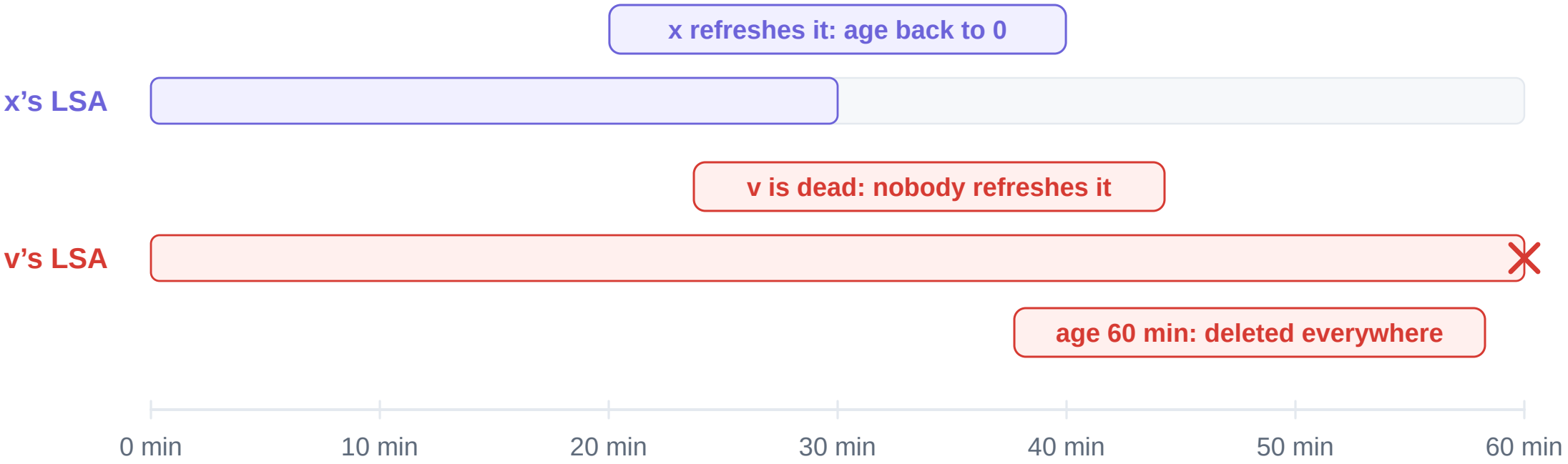


otherwise: the link-state lecture re-floods each LSA only every 30 min

OSPF calls this the database exchange. After it the two are adjacent, and only changes flow between them.

No. It swaps database lists with its neighbour once, and asks for what it lacks.

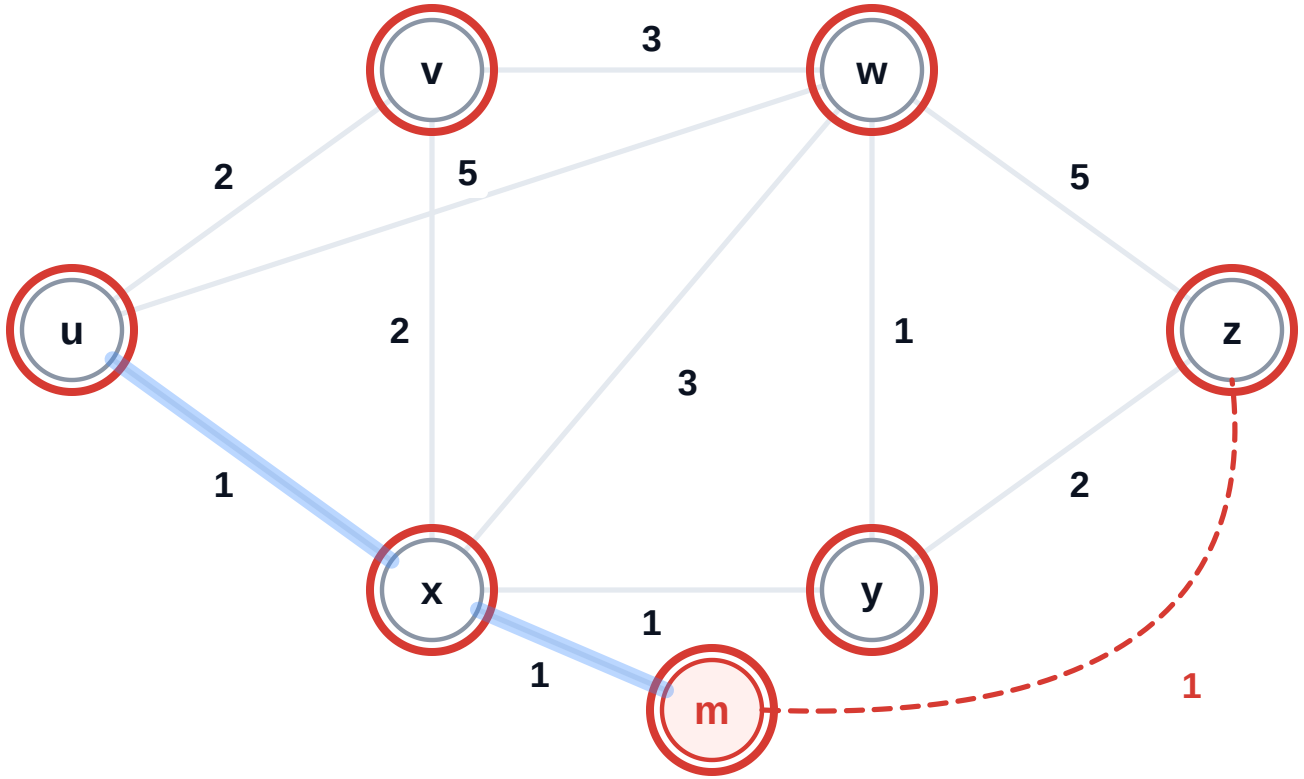
A router dies. Who deletes the LSA it sent?



Its neighbours drop their links to v after 40 s, so paths avoid v long before. Refresh every 30 min, maximum age 1 hour: RFC 2328.

Nobody has to. An LSA not refreshed for an hour ages out of every database.

Every router believes every LSA. What if one lies?



from m, seq 1

`(x, 1) (z, 1)`

m has no link to z

m: a laptop on a port where x speaks OSPF

u → z: u–x–m–z = 3

beats u–x–y–z = 4; m drops it

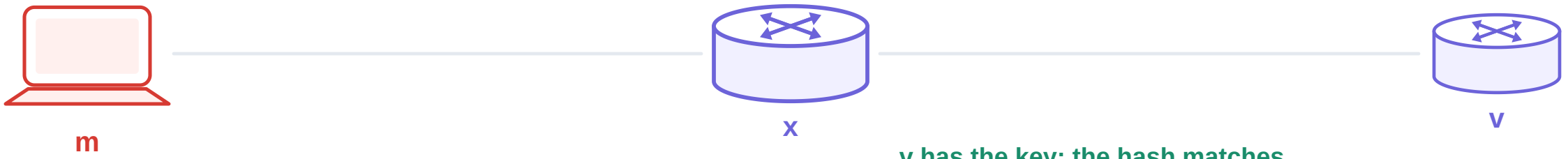
Every map now has the false link, and u's traffic for z runs into m.

1 Autonomous systems | 2 OSPF | 3 Areas | 4 Link weights

Authentication: a key per link, a hash on every message

`hash = HMAC-SHA( key, message )`

x recomputes it with the key it shares with each neighbour



m has no key: the hash is wrong, dropped

v has the key: the hash matches

OSPFv2 HMAC-SHA authentication: RFC 5709. A sequence number in each packet stops an old valid message from being replayed.

A message whose hash does not match is dropped before its LSA is stored.

What can a key not stop?

a key stops

an outsider with no key: its messages are dropped at the first router

a key cannot stop

a router that holds the key and lies: its hash is valid

fight-back

forge another router's LSA, and that router floods a newer true one

not enough

one compromised router can keep others' maps wrong (NDSS 2012)

Nakibly, Kirshon, Gonikman, Boneh, "Persistent OSPF Attacks", NDSS 2012.

A router that holds the key. The hash proves who sent it, not that it is true.

What OSPF adds to the link-state lecture



hello

find neighbours;
40 s of silence
= dead



join

a new router copies
the database in
one exchange



aging

an LSA nobody
refreshes leaves
after an hour



keys

a message without
the right hash
is dropped

Flooding and Dijkstra, plus hellos, a join, aging and keys.

1 Autonomous systems | 2 OSPF | 3 Areas | 4 Link weights

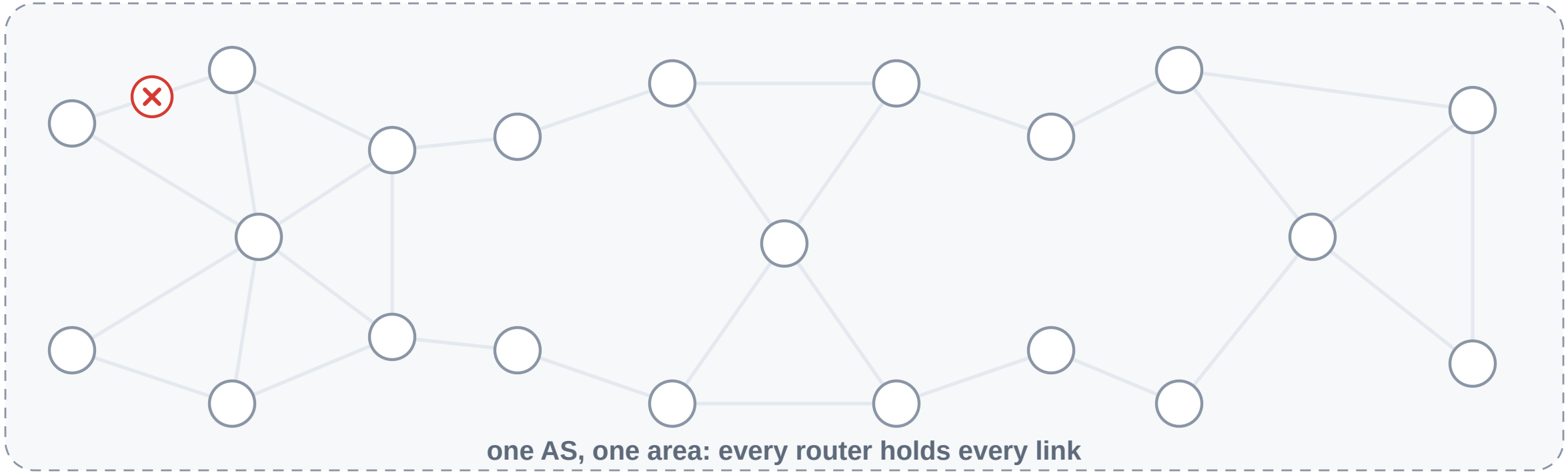
One link keeps failing at the edge of a big AS. Who pays?

03

Areas

Every LSA reaches every router in the AS. In a large AS, one flapping link at the edge keeps every router busy.

One flapping link makes every router recompute

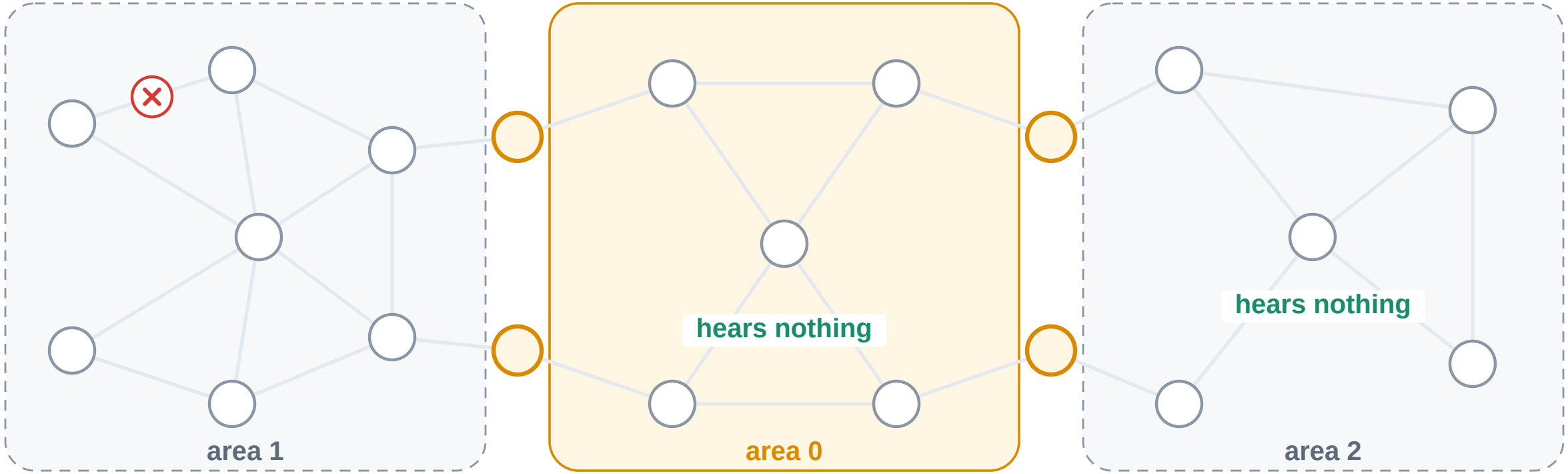


one AS, one area: every router holds every link

Every flap: a new LSA to every router, and a new Dijkstra at every router, over every link of the AS.

Every flap floods the whole AS, and every router reruns Dijkstra.

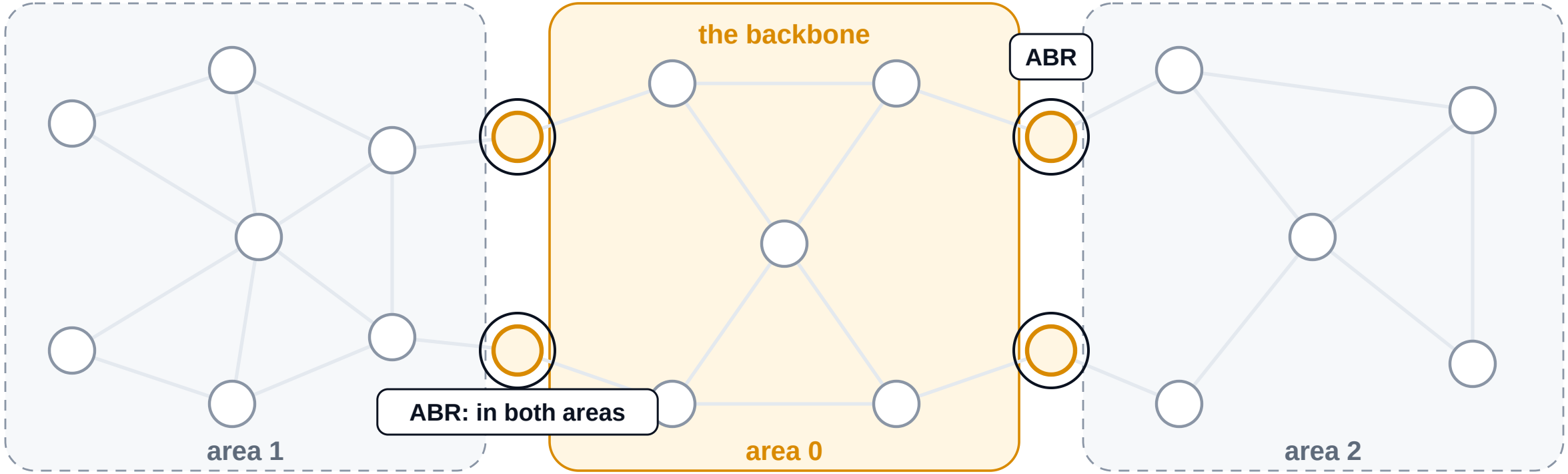
Cut the AS into areas. Flood only inside one



The same routers and links, split in three. The routers on the borders (filled) belong to two areas.

An LSA is flooded only inside its own area. The other areas never see the flap.

Area 0 in the middle, a border router on each edge

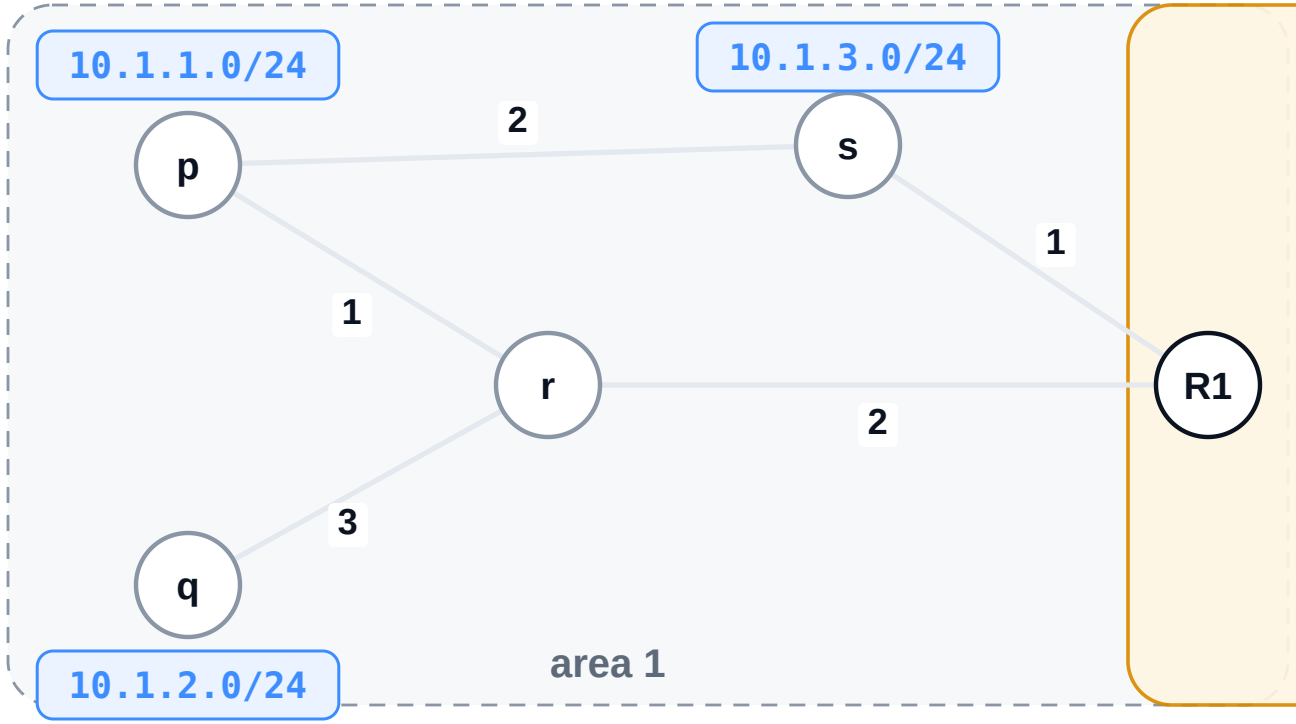


An area border router (ABR) runs link state in both of its areas: two databases, two Dijkstras.

Every area touches area 0, the backbone. Each ABR belongs to both sides.

1 Autonomous systems | 2 OSPF | 3 Areas | 4 Link weights

What does R1 tell area 0 about area 1?



what R1 sends into area 0

prefix	cost from R1
10.1.1.0/24	3
10.1.2.0/24	5
10.1.3.0/24	1

no links, no routers: three rows

area 0

R1 runs Dijkstra on area 1's map, then sends one row per prefix. OSPF calls it a summary LSA.

Not the links. One row per prefix: the prefix, and R1's cost to it.

1 Autonomous systems | 2 OSPF | 3 Areas | 4 Link weights

To area 2, area 1 is three numbers

area 1, seen from area 2

prefix	cost
10.1.1.0/24	3
10.1.2.0/24	5
10.1.3.0/24	1

no routers, no links



S's cost to 10.1.2.0/24

4 + 5 = 9, via R1

S's cost to R1: 4

the distance-vector equation:

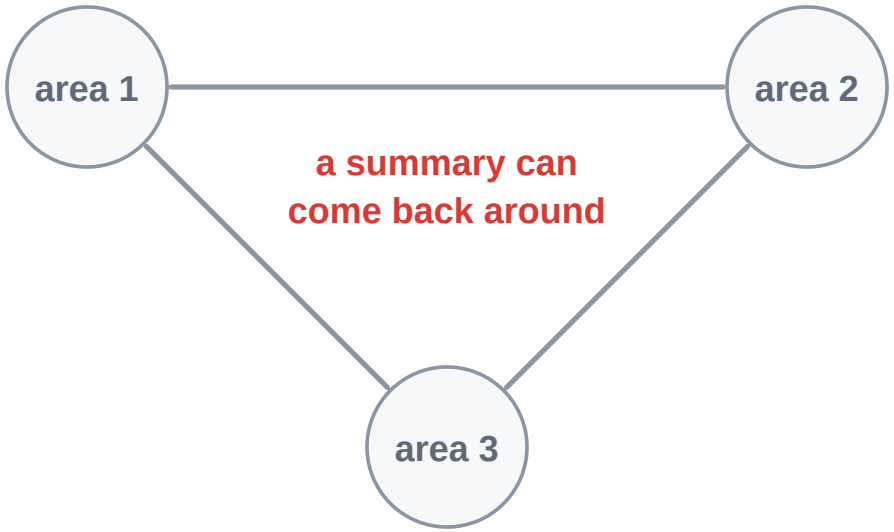
cost to a neighbour + what it reported

The distance-vector lecture, point 01: d(S, y) = min over v of c(S, v) + d(v, y).

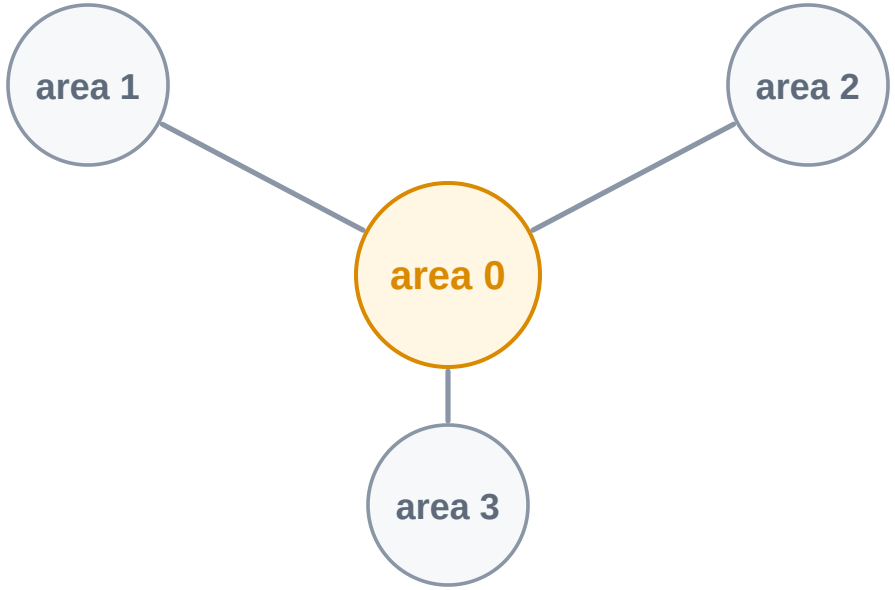
Link state inside an area, distance vector between areas.

Distance vectors can loop. What does OSPF do?

if areas linked up in a ring



OSPF: a star around area 0



RFC 2328: inter-area routing is “a star configuration … with the backbone as hub”. An ABR never passes area 2’s summaries back into another area on its own.

Areas form a star around area 0, so summaries cannot go around in a circle.

Summarise further: one row for the whole area

three rows

10.1.1.0/24	cost 3
10.1.2.0/24	cost 5
10.1.3.0/24	cost 1

one row

10.1.0.0/16	cost 5

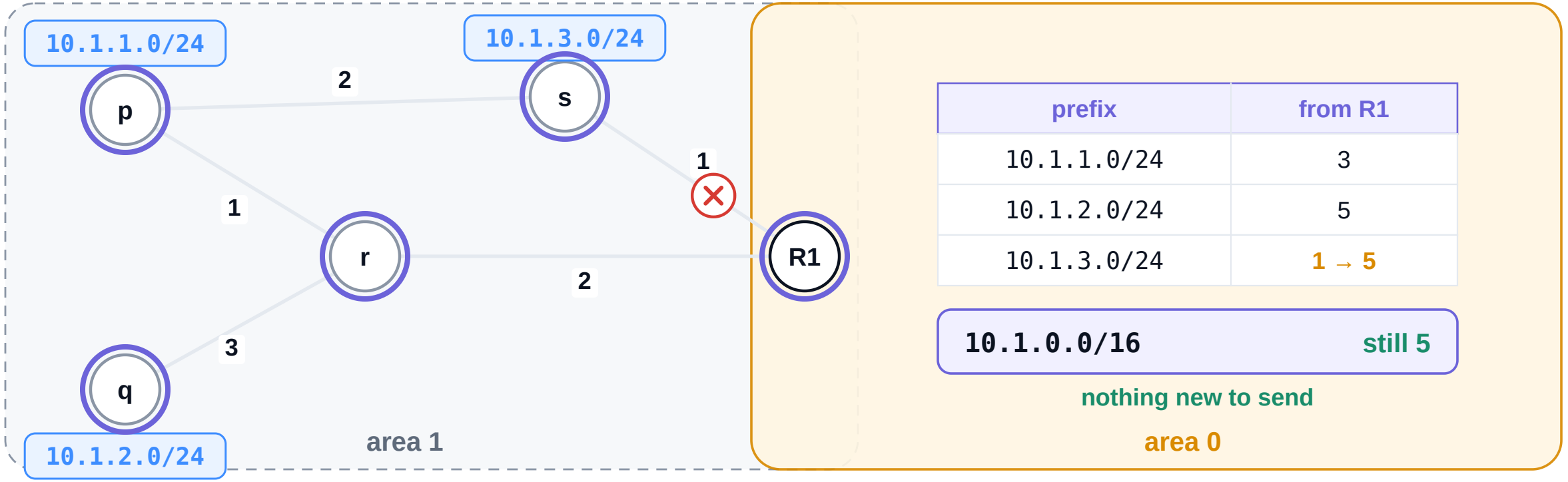
the cost: the largest inside the range

so no prefix in it is cheaper than claimed

Aggregation, as in the Forwarding lecture: one prefix covers all three. Range cost = the maximum of its parts (RFC 2328 §12.4.3).

One row for the area; its cost is the largest cost inside the range.

A link inside area 1 fails. Who hears about it?



prefix	from R1
10.1.1.0/24	3
10.1.2.0/24	5
10.1.3.0/24	1 → 5

10.1.0.0/16 still 5

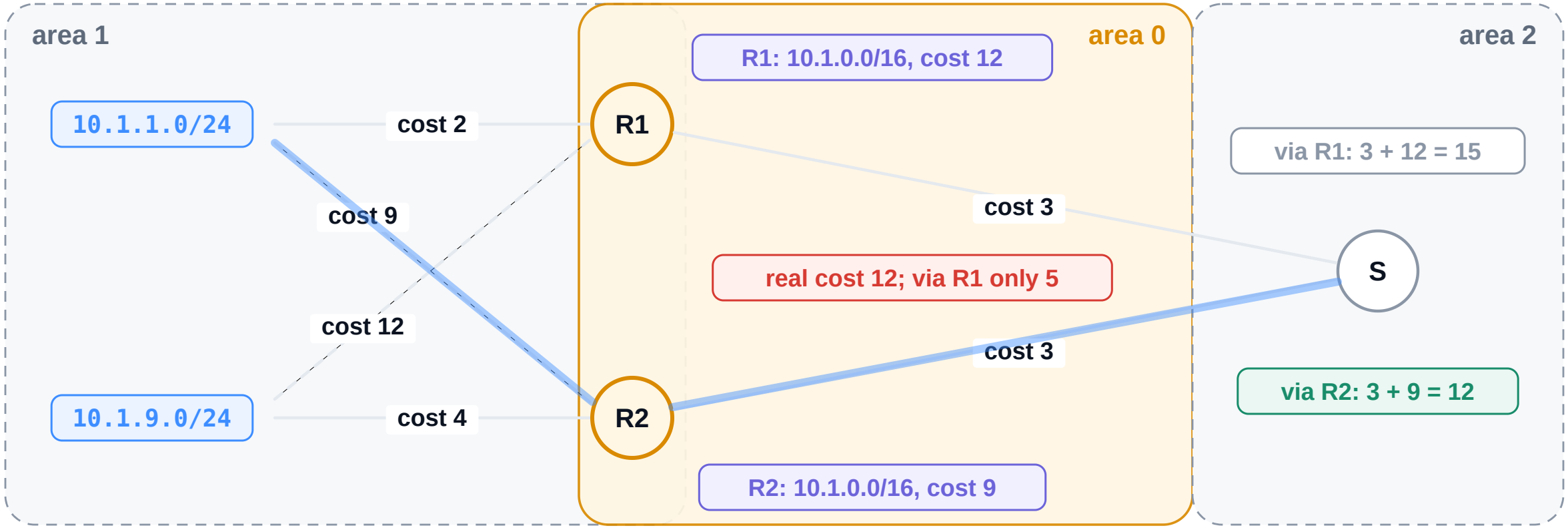
nothing new to send

R1 recomputes: 10.1.3.0/24 now costs 5 from R1, but the range cost was already 5.

Only area 1. The range still costs 5, so nothing crosses the border.

1 Autonomous systems | 2 OSPF | 3 Areas | 4 Link weights

What does the range hide?



S sends everything via R2: 10.1.1.0/24 costs 12, where 5 was possible.

1 Autonomous systems | 2 OSPF | 3 Areas | 4 Link weights

What do areas buy, and what do they cost?

what areas buy

a flap floods one area, not the AS
a map of one area plus summaries
Dijkstra over one area

what they cost

paths that are sometimes longer
area to area always through area 0
someone has to design the areas

inside an area: the whole map, exact shortest paths

between areas: summaries, a distance vector

Less flooding and smaller maps, paid for with paths that are sometimes longer.

1 Autonomous systems | 2 OSPF | 3 Areas | 4 Link weights

Same map everywhere. What decides where the traffic goes?

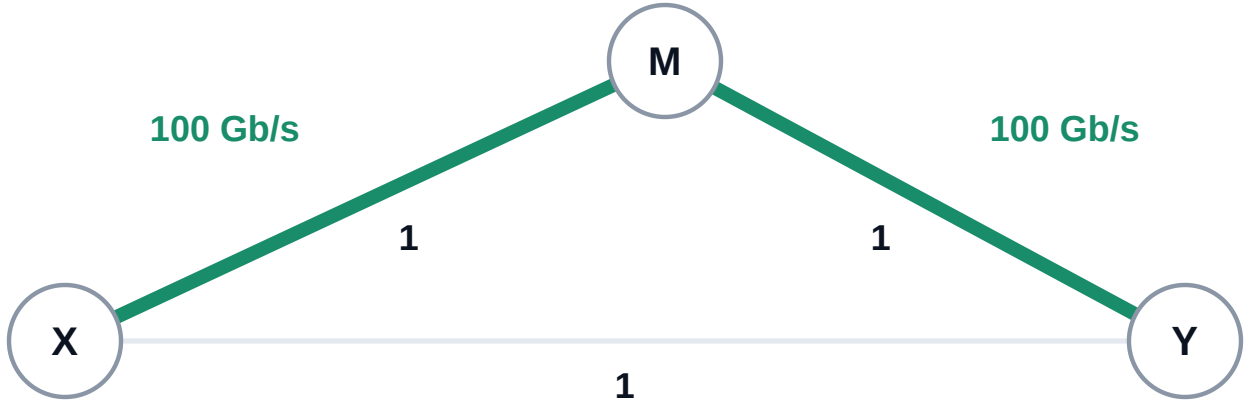
04

Link weights

Dijkstra follows the costs, and the operator writes the costs.
Change one number and traffic moves.

Who writes the costs? The operator

link speed	default cost
10 Mb/s	10
100 Mb/s	1
1 Gb/s	1
10 Gb/s	1
100 Gb/s	1



1 Gb/s, cost 1

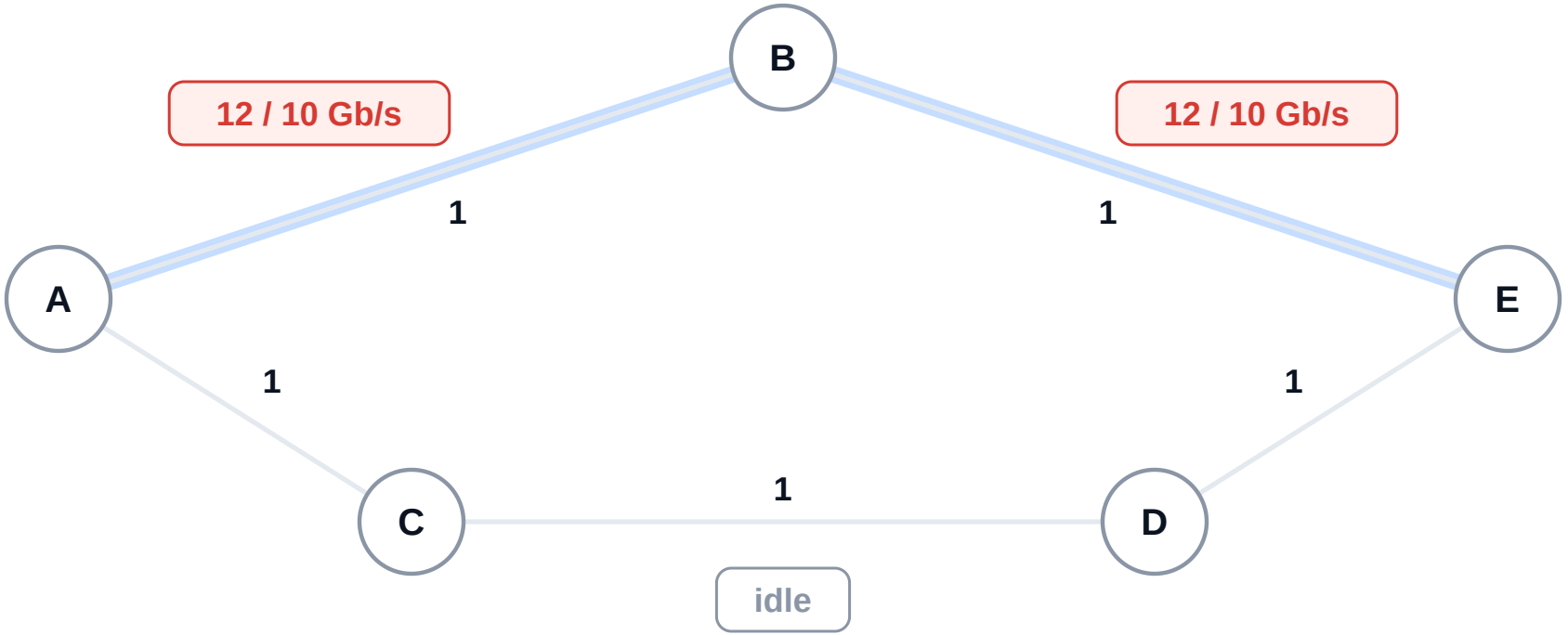
X → Y: 1 hop, cost 1, beats 2

cost = 100 Mb/s ÷ speed, at least 1

The default on Cisco routers: 100 Mb/s ÷ link speed, never below 1. Operators raise that reference, or set each cost by hand.

By default a 1 Gb/s and a 100 Gb/s link both cost 1. Operators set their own.

Three flows of 4 Gb/s go from A to E. Which way?



A → E

3 flows × 4 Gb/s

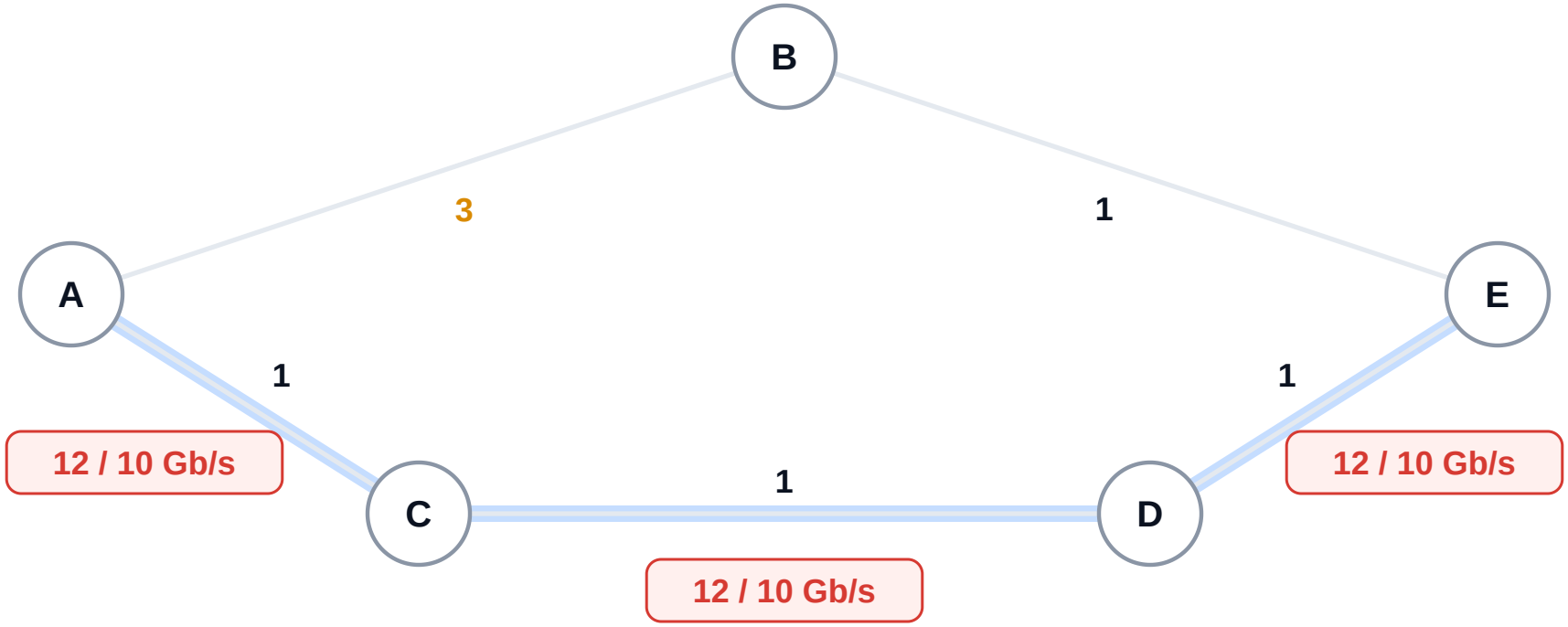
A–B–E: 2

A–C–D–E: 3

Every weight is 1, every link 10 Gb/s.

All 12 Gb/s take A–B–E. Two links overload while A–C–D–E sits idle.

Raise A–B from 1 to 3. Where does the traffic go?



A → E

A–B–E: 3 + 1 = 4

A–C–D–E: 3

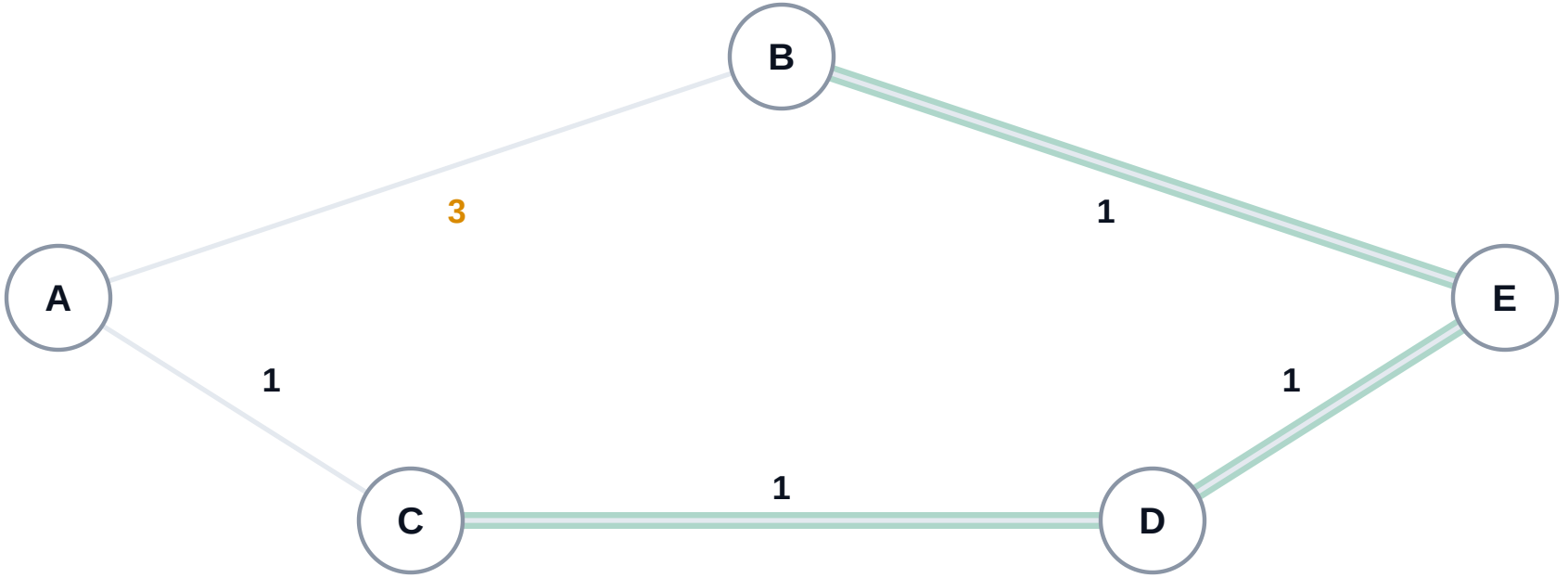
the lower path wins

A–B: 1 → 3

All of it moves to A–C–D–E, and the overload moves with it.

1 Autonomous systems | 2 OSPF | 3 Areas | 4 Link weights

Did the flow from C to B move too?



C → B

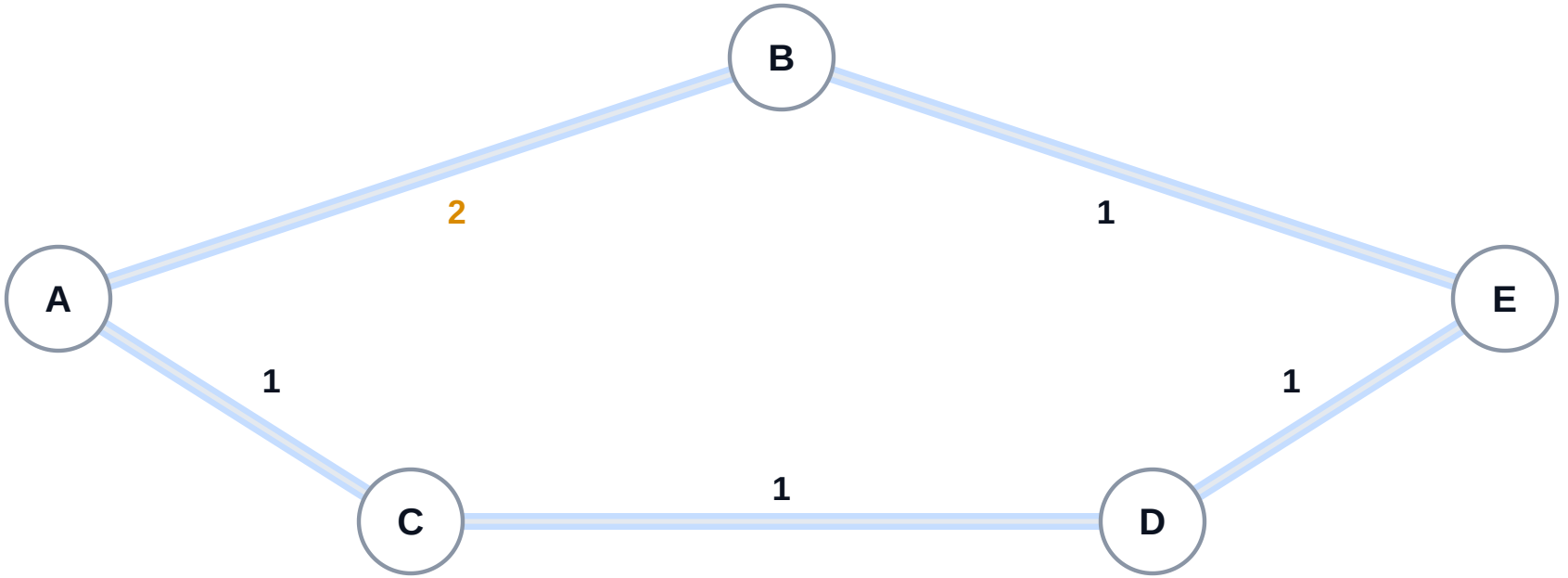
C–A–B: 1 + 3 = 4

C–D–E–B: 3

nobody asked it to move

Yes. One weight change moves every path that crossed that link.

Set A–B to 2. Now the two paths tie



A → E

A–B–E: 2 + 1 = 3

A–C–D–E: 3

A's table:

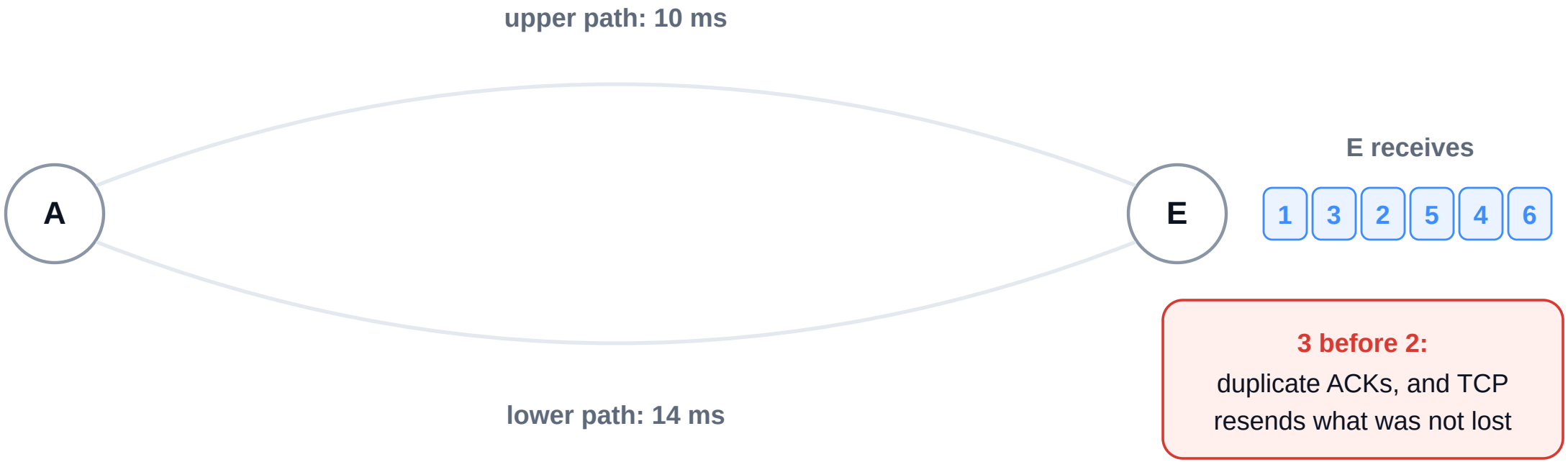
E → B or C

OSPF keeps every next hop that ties. This is equal-cost multipath: ECMP.

Two equal-cost paths, and OSPF keeps both next hops. How should A split?

1 Autonomous systems | 2 OSPF | 3 Areas | 4 Link weights

Split one TCP connection packet by packet?

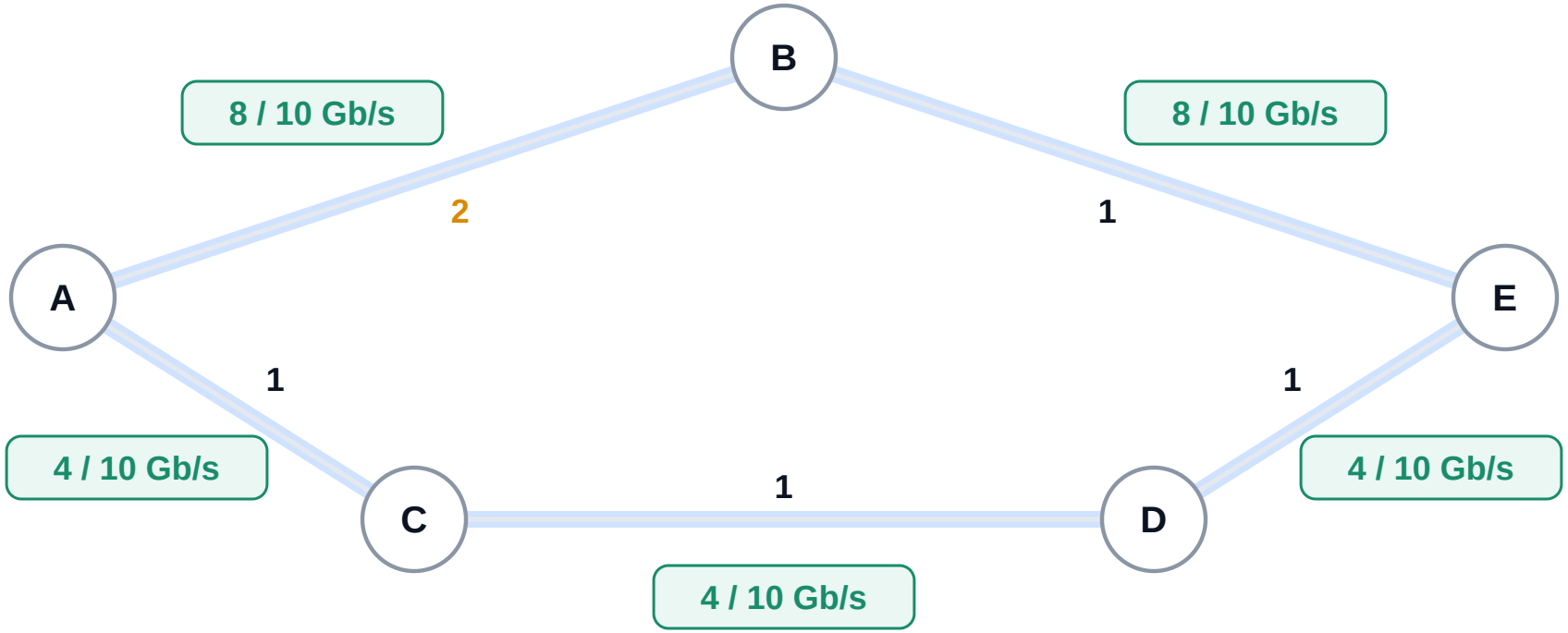


3 before 2:
duplicate ACKs, and TCP resends what was not lost

Odd packets on the upper path, even on the lower. Duplicate ACKs and fast retransmit: Chapter 3.

Packets overtake each other, and the sender slows down for nothing.

Split per flow: hash five header fields



A hashes

source address
destination address
protocol
source port
destination port

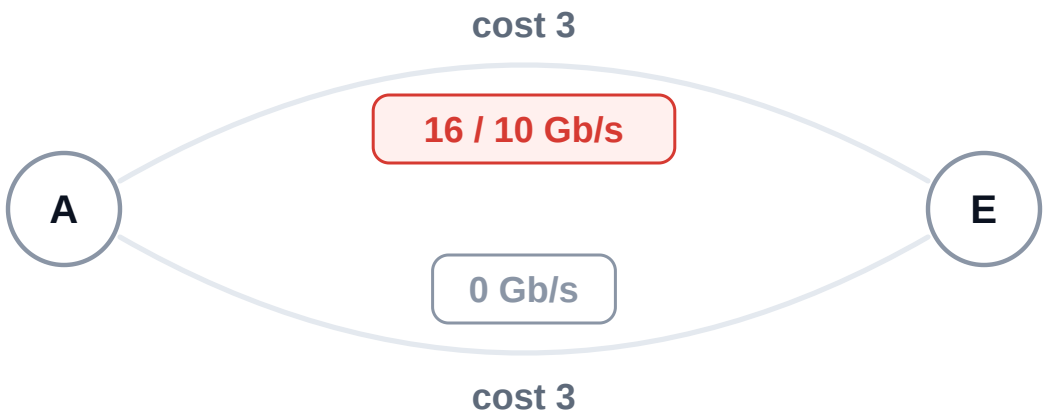
the same flow → the same path

Flows 1 and 3 hash to the upper path, flow 2 to the lower one.

Each flow stays on one path. Three flows split 2 to 1: 8 and 4 Gb/s, both fit.

1 Autonomous systems | 2 OSPF | 3 Areas | 4 Link weights

What can ECMP not do?



two big flows hash to the same path



costs 3 and 4 do not tie: no split

Where it shines: many parallel equal-cost links and many small flows. Data centre networks are built that way.

It balances flows, not bytes, and only among paths whose costs tie exactly.

1 Autonomous systems | 2 OSPF | 3 Areas | 4 Link weights

How would you pick every weight in a network?

the problem

given the traffic between every pair of routers, pick weights so no link is overloaded

hard

finding the best weights is NP-hard

but search works

change one weight at a time, keep the change if it helps

AT&T WorldNet backbone, 90 routers, 274 links: within a few percent

of the best any routing could do. The inverse-capacity default was clearly worse.

Fortz and Thorup, “Internet Traffic Engineering by Optimizing OSPF Weights”, INFOCOM 2000.

The best weights are NP-hard to find, but a search gets within a few percent.

What does steering with weights cost?

every change moves a lot

a new LSA, a new Dijkstra everywhere,
many paths shift at once

and briefly loops

while maps differ, packets can loop
(the link-state lecture, point 04)

cannot steer one flow

a weight belongs to a link, not to a flow. Pinning one flow to a path takes explicit paths (MPLS traffic engineering, segment routing): outside this course

A weight moves every path over its link. Steering one flow needs another tool.

Four topics, one line each

01



Autonomous systems

split by who runs the routers; inside and between

02



OSPF

link state plus hellos, a join, aging and keys

03



Areas

link state inside, distance vector between areas

04

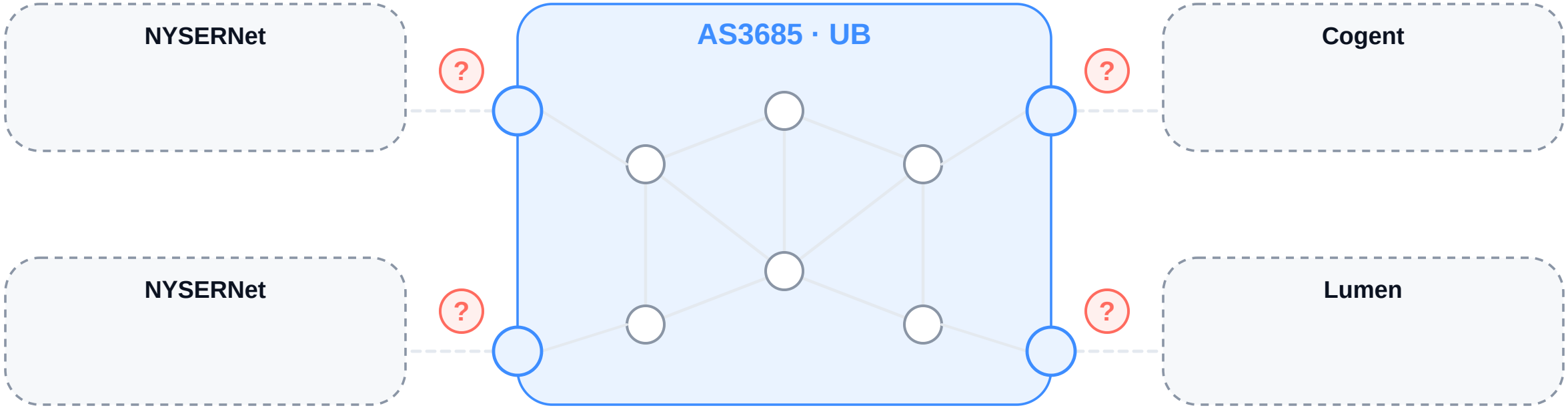


Link weights

costs pick the paths; ECMP splits flows on ties

Inside one AS: one operator, one map, shortest paths by the operator’s costs.

What is left: routing between ASes



which gateway? what lies beyond it? who will carry whose traffic?

Reading: Kurose and Ross, 5.3 (OSPF). Next lecture: BGP, 5.4.

Inside an AS the goal is the shortest path. Between ASes it is policy: BGP.