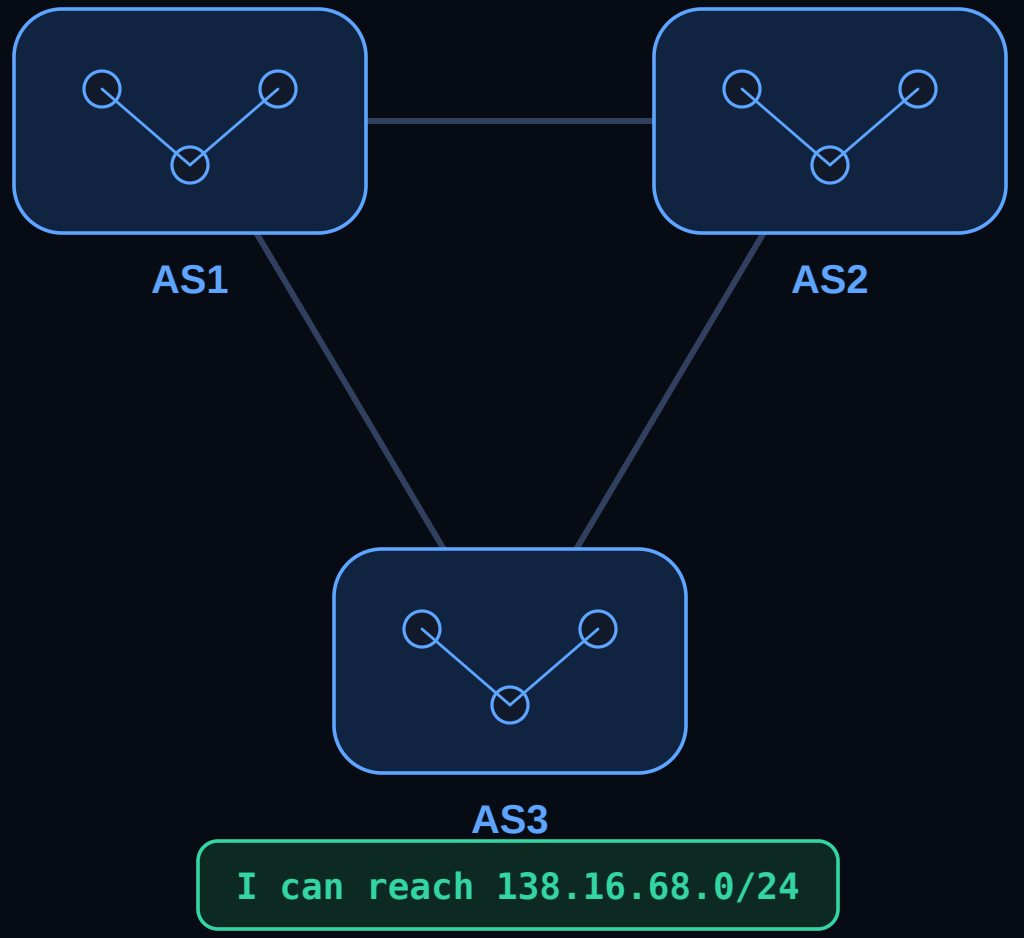
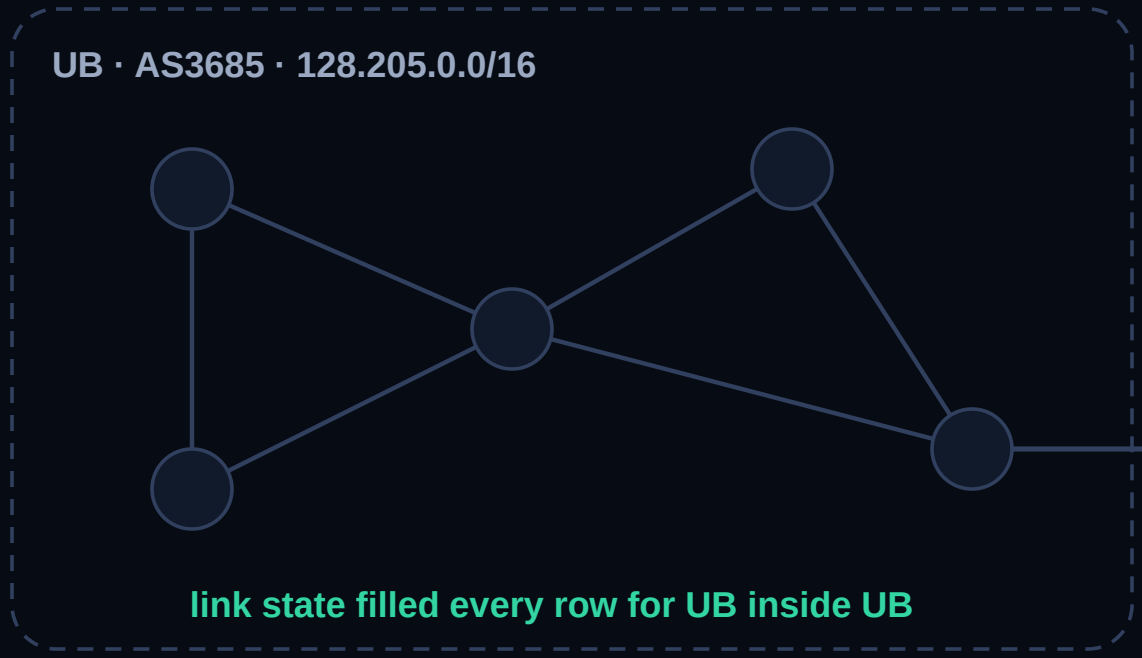
# BGP
# Routing between autonomous systems

Every table so far was filled inside one network. This lecture is about the other rows: how some eighty thousand independently run networks tell each other which addresses they can reach, which routes they choose, who pays for the traffic, and what happens when one of them lies.

**Yaxiong Xie**



# Every row so far was computed inside one network




| prefix | port |
|---|---|
| 128.205.32.0/19 | 2 |
| 128.205.36.0/22 | 3 |
| 128.205.64.0/18 | 1 |
| 8.8.8.8 | ? |

no row matches

A default route only hands the question to someone else. Somewhere, routers must hold a row for every prefix on the internet.

**Where do the rows for the rest of the internet come from?**

# Eight topics

**01**
**Reachability**
what a network announces

**02**
**AS-PATH**
how a route avoids loops

**03**
**The session**
how routes are carried

**04**
**iBGP**
spreading a route inside

**05**
**NEXT-HOP**
from route to table row

**06**
**Choosing**
one row, many routes

**07**
**Who pays whom**
which routes are offered

**08**
**Hijacks**
when a network lies

**Each from its own problem: why it is needed, how it works, what it costs.**

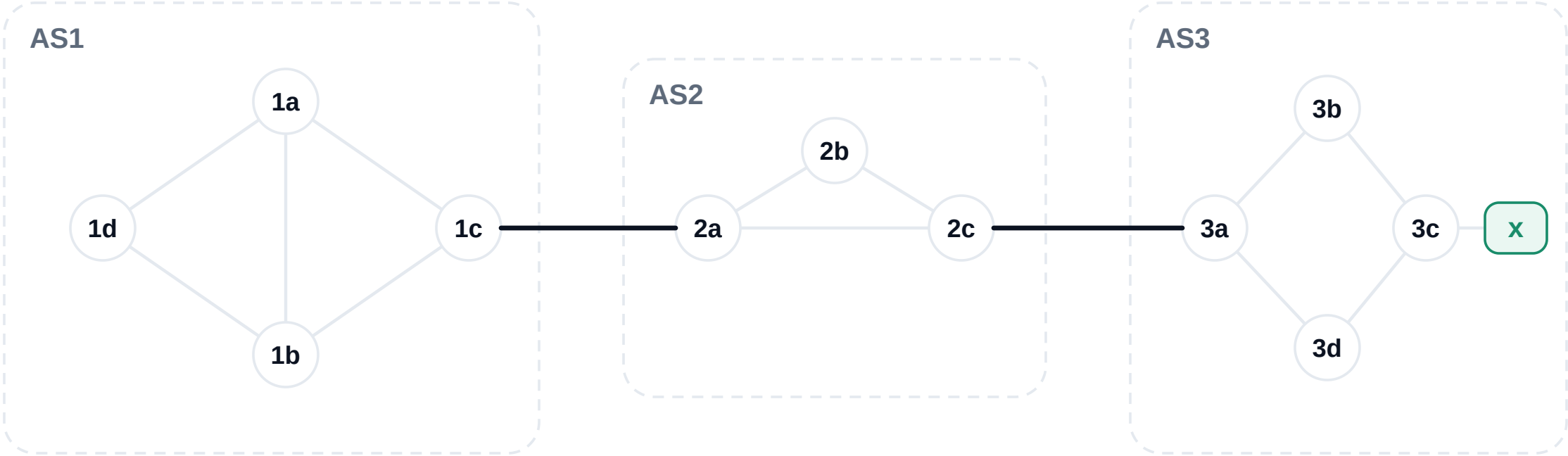
# 1d has no row for x, and x is in another network

01

## Reachability, not topology

Link-state flooding stopped at the edge of 1d’s network. Nothing inside it knows what lies beyond.

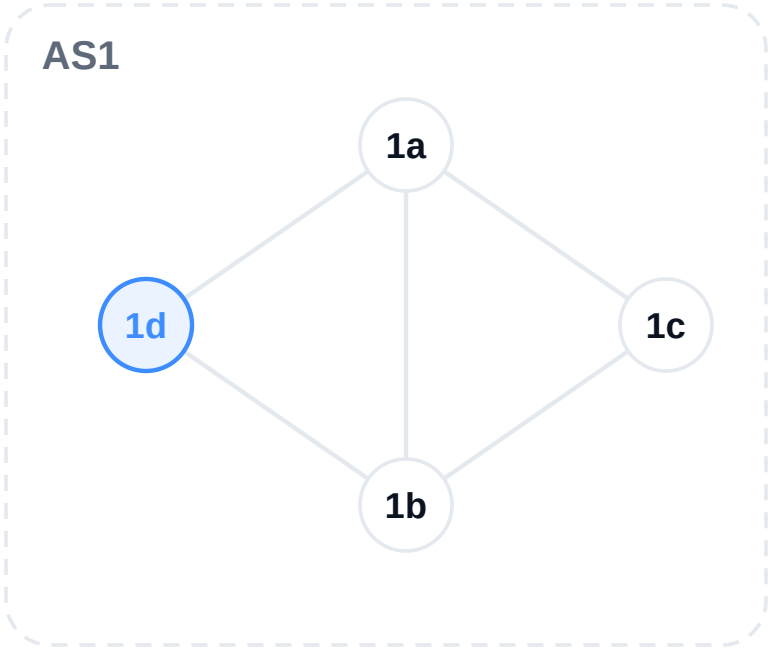
# Three networks, each run by its own operator




Press → for each network, then for the links between them. UB is one: AS3685, the number on its ARIN record.

**Each is an autonomous system (AS): one operator, its own routing inside.**

# Why is there no row for x at 1d?




**1D'S FORWARDING TABLE**

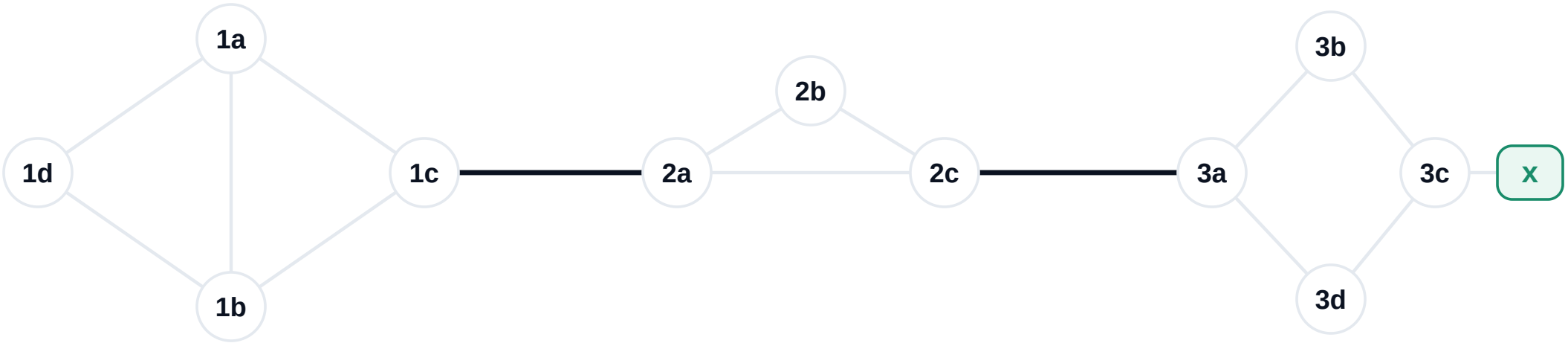
| prefix | next hop |
|---|---|
| 10.1.1.0/24 | 1a |
| 10.1.2.0/24 | 1b |
| 10.1.3.0/24 | 1a |
| 10.1.4.0/24 | 1b |
| 138.16.68.0/24 | ? |

a packet for 138.16.68.4 is waiting at 1d

**Link state saw only AS1's links. x is not in AS1's graph.**

# Could the three ASes run one link-state protocol together?

one graph, one flood, every router




**Then every router would hold every link and weight of every AS.**

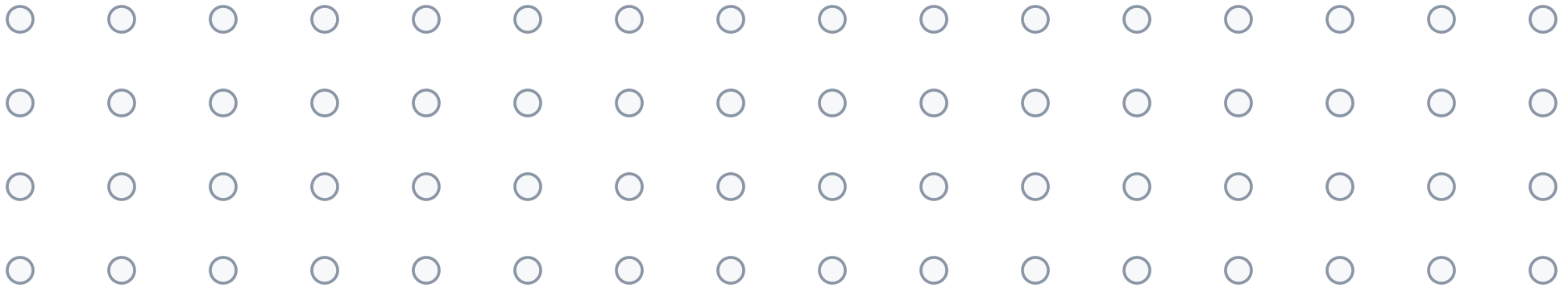
# How big would that one graph be?

**79,467**

ASes in the routing system

**1,077,688**

IPv4 prefixes they announce



**every change anywhere, flooded to every router**

cidr-report.org, 22 September 2026. Links fail, recover and change cost somewhere every second.

**Too big for every router, and it would never stop changing.**

# Would AS2 publish its links and weights?




**No. Its topology is its operator's business, and its neighbours are its competitors.**

# Do AS1's link weights mean the same as AS2's?



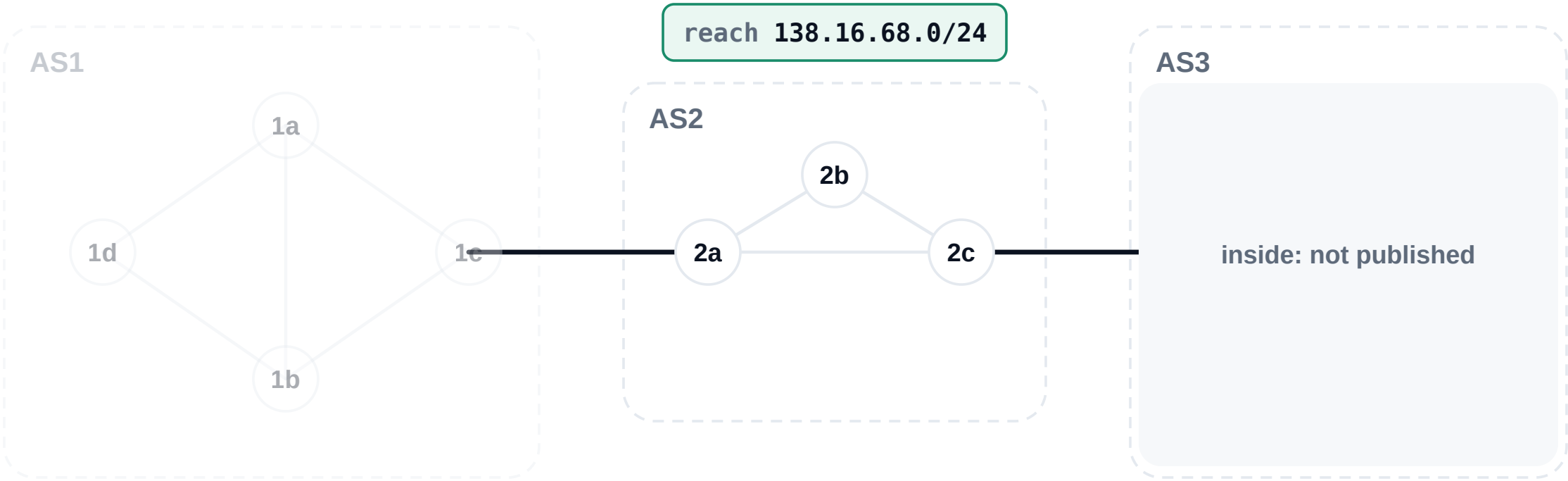

AS1 weights links by delay

AS2 by what a link costs it

`5 ms + $90 = ?`

**No. There is no shared cost to add across ASes, so no single shortest path.**

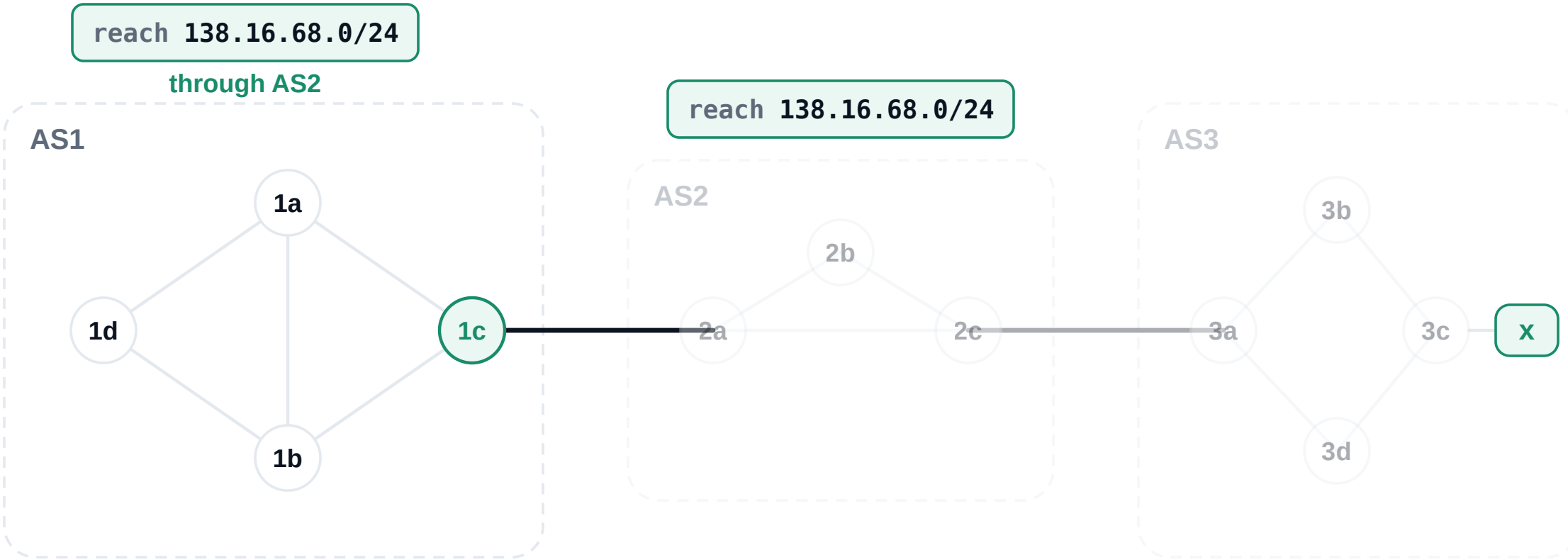
# What can AS3 tell AS2 instead?




Press → to send it.

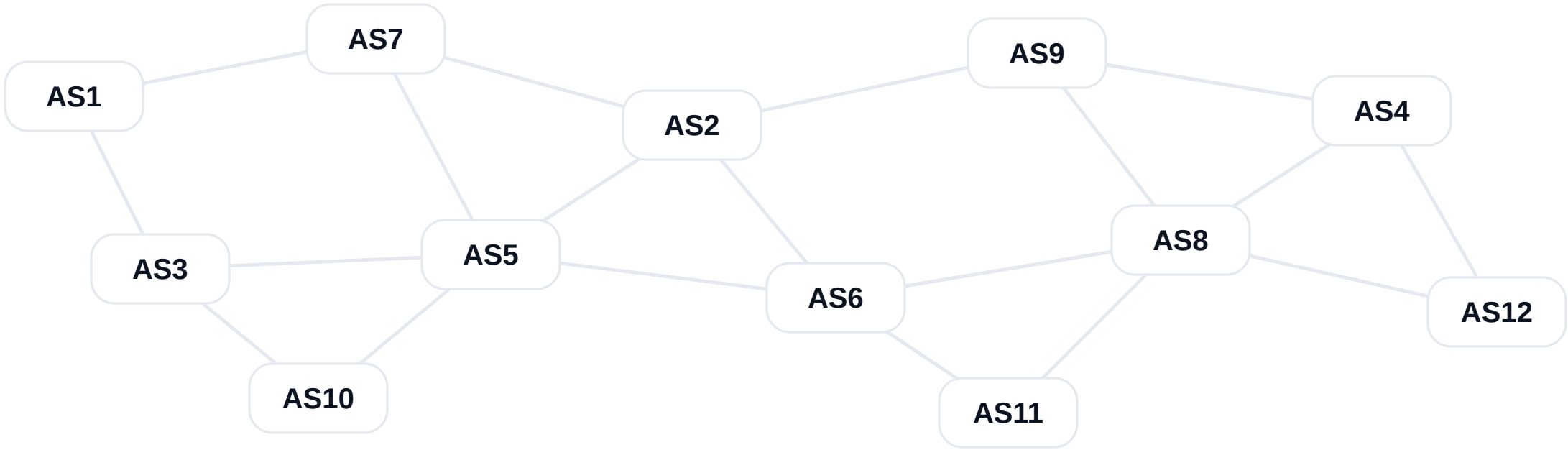
**Only the prefixes it can reach, nothing inside. That message is an advertisement.**

# AS2 passes it on, and AS1 learns a way to x




**An AS that uses a route may pass it on. AS1 now reaches x through AS2.**

# One protocol does this at every AS border




What it gives: reachability without publishing insides. What it costs: an AS sees only what its neighbours choose to tell it.

**BGP, the Border Gateway Protocol (BGP-4, RFC 4271), runs between all ASes.**

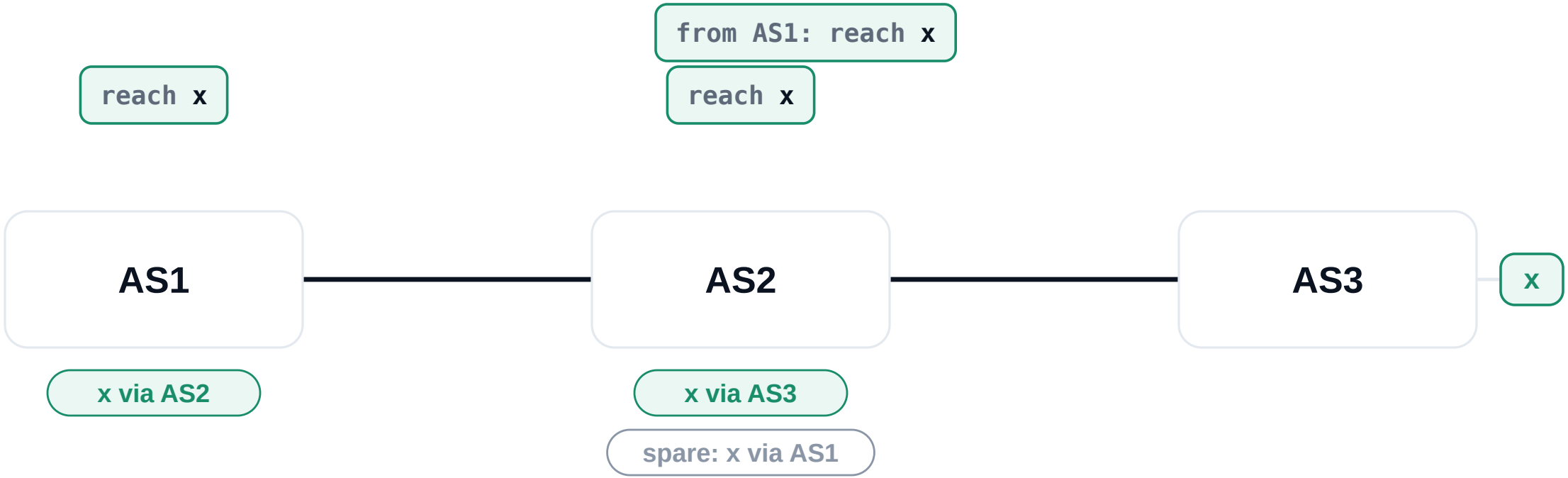
# “I can reach x” can come back to the AS that said it

02

## AS-PATH

A claim that carries no path cannot show the listener that the route runs through the listener itself.

# AS1 tells every neighbour it can reach x, AS2 included




Press → three times: AS3 to AS2, AS2 to AS1, and AS1 to all its neighbours.

**AS1 cannot tell that its own route to x runs through AS2.**

# The link between AS2 and AS3 fails




Press →: the link fails, AS2 falls back on AS1's claim. Press again: send a packet.

**AS2 sends x's packets to AS1, and AS1 sends them back: a loop.**

# Where have we seen this before?

distance vector lecture

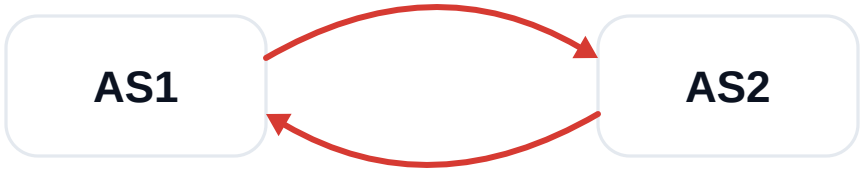
**But z's route to x goes through y**




**z told y "to x: 5", with no path**

this lecture

**AS1's route to x goes through AS2**




**AS1 told AS2 "reach x", with no path**

**The same failure as counting to infinity: a claim that hides its path.**

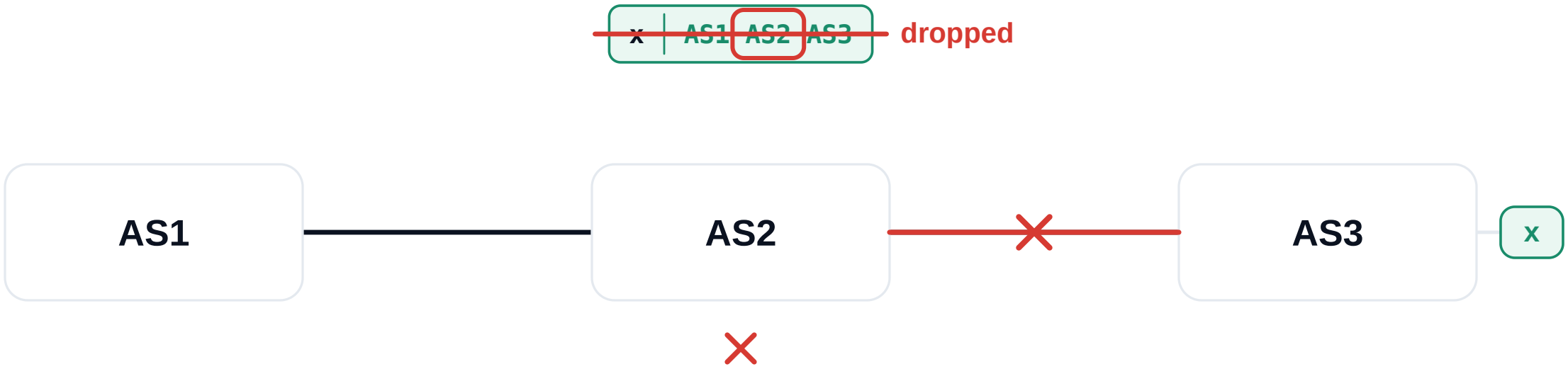
# Carry the path in the advertisement




Press → for each hop. The number just added is amber.

**Each AS puts its own number in front before passing it on: the AS-PATH.**

# AS2 receives “AS1 AS2 AS3”




Press →: AS1 passes the route back. Press again: the link to AS3 fails.

**An AS drops any advertisement that already holds its own number. No loop forms.**

# A number and a path, side by side

**distance vector**

`to x: 5`

how far, not through whom

**BGP**

| x | AS1 AS2 AS3 |
|---|---|

AS1 AS2 AS3

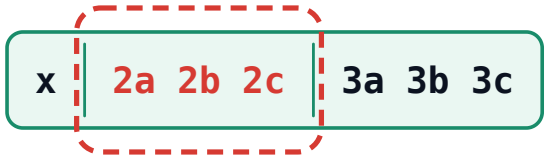
every AS on the way, by name

**A distance hides who is on the way. A path names every AS: BGP is path-vector.**

# Why AS numbers, and not router addresses?




two numbers, nothing about the inside




every router: long, and it exposes the inside

AS2 will not publish these

**AS numbers keep the path short and every inside hidden.**

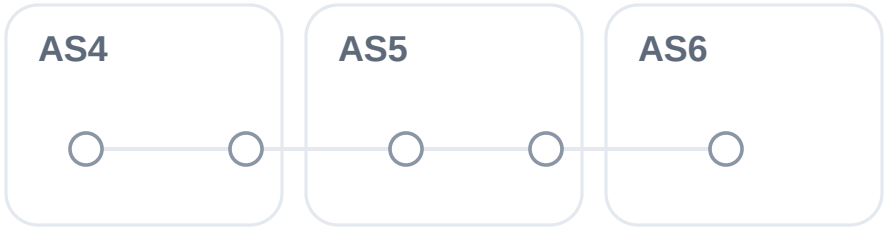
# Is a shorter AS-PATH a shorter route?

**AS-PATH: AS2 AS3 (TWO ASES)**




**14 routers, two continents**

**AS-PATH: AS4 AS5 AS6 (THREE ASES)**




**5 routers, one city**

**No. One AS on the path can be one router or a thousand.**

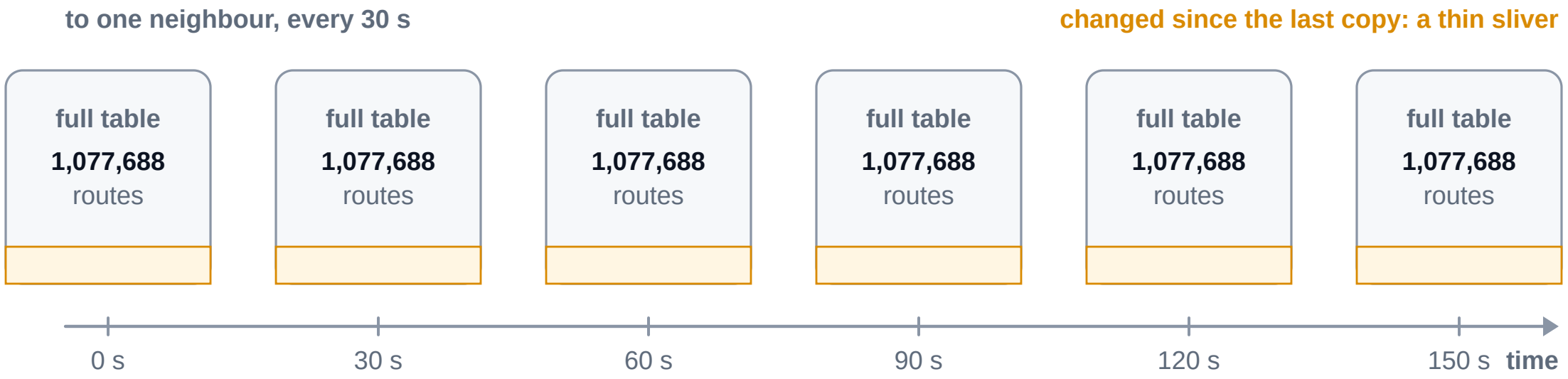
# A million routes cannot be resent every 30 seconds

03

## The session

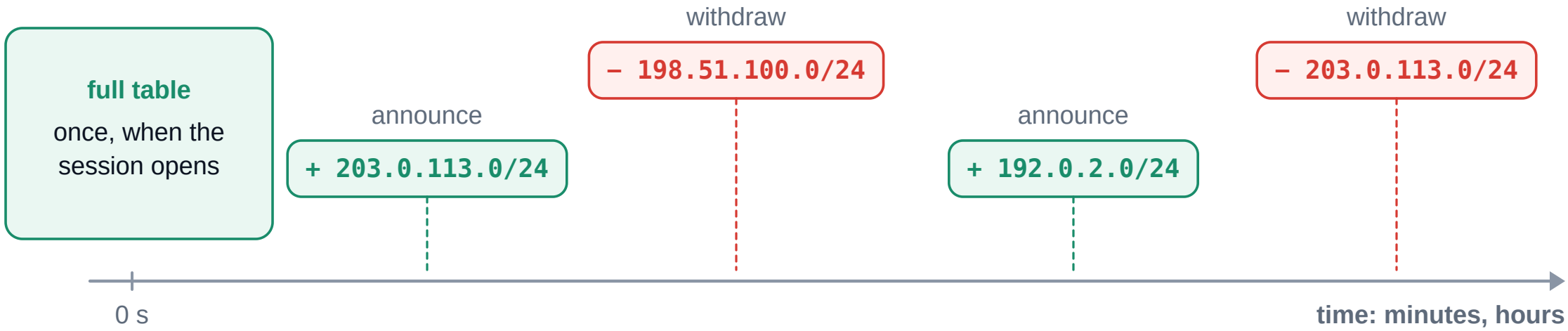
Distance vector kept routers in step by resending the whole vector on a timer. For BGP, that vector is the whole routing table.

# What would resending everything cost?




**Almost every route sent would repeat what the neighbour already has.**

# Send the table once, then only the changes




Press → for each change. Between changes, nothing is sent.

**A route stays valid until it is withdrawn. Nothing is ever repeated.**

# What if one of those messages is lost?



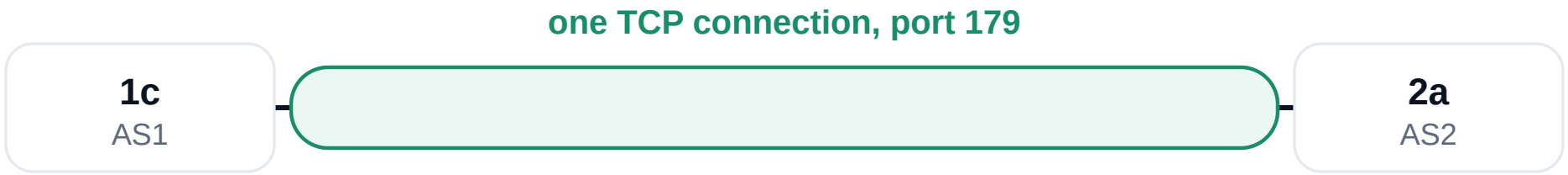

**1C'S BGP TABLE**

| prefix | learned from |
|---|---|
| 198.51.100.0/24 | 2a |
| 203.0.113.0/24 | 2a |
| 138.16.68.0/24 | 2a |

still here an hour later

**Sent once means it must arrive. BGP runs over TCP.**

# A BGP session is one TCP connection between two routers

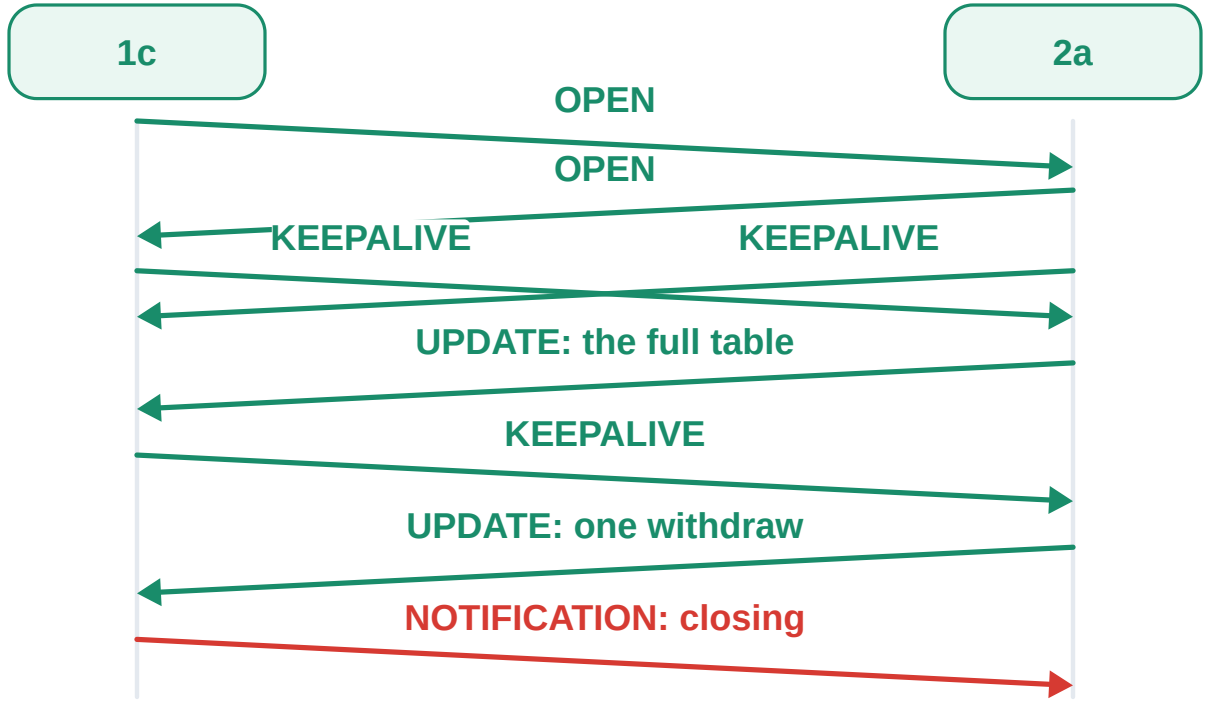



opened once, kept open for days or months

**one session per pair of routers that talk BGP, not one per AS**

**The routers at a border hold one long-lived TCP connection: a BGP session.**

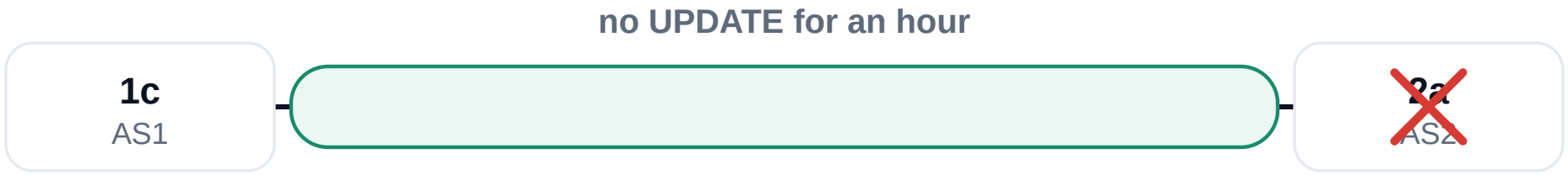
# One session, message by message

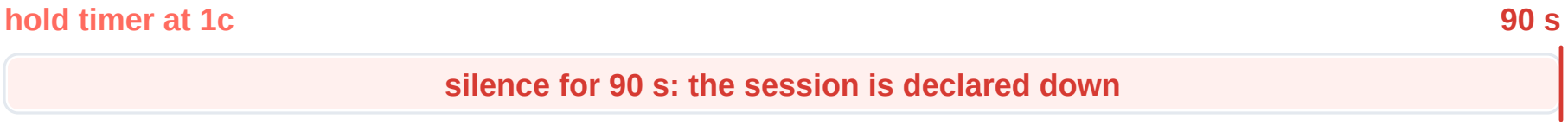



Press → for each message.

**Four messages: OPEN, UPDATE, KEEPALIVE, and NOTIFICATION to report an error.**

# Nothing changes for an hour. Is the neighbour still there?



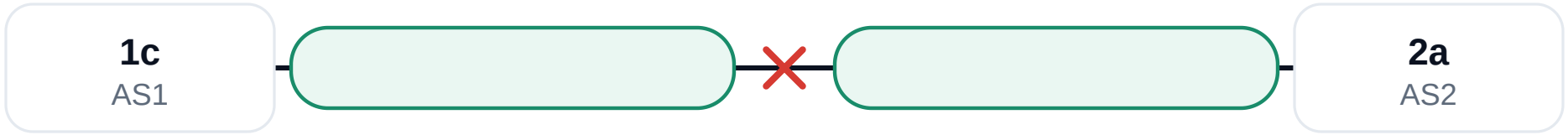




a KEEPALIVE every 30 s resets it

RFC 4271 suggests a 90 s hold time and a KEEPALIVE every third of it. Press → for a neighbour that died.

**KEEPALIVEs tell silence from death. No message for a hold time: the session is down.**

# The session drops




**1C'S BGP TABLE**

| prefix | learned from |
|---|---|
| ~~198.51.100.0/24~~ | ~~2a~~ |
| ~~203.0.113.0/24~~ | ~~2a~~ |
| ~~138.16.68.0/24~~ | ~~2a~~ |

**and so on, for every route learned over this session**

Gives: messages only on change. Costs: one failure removes everything learned over it.

**Every route learned over the session is withdrawn at once.**

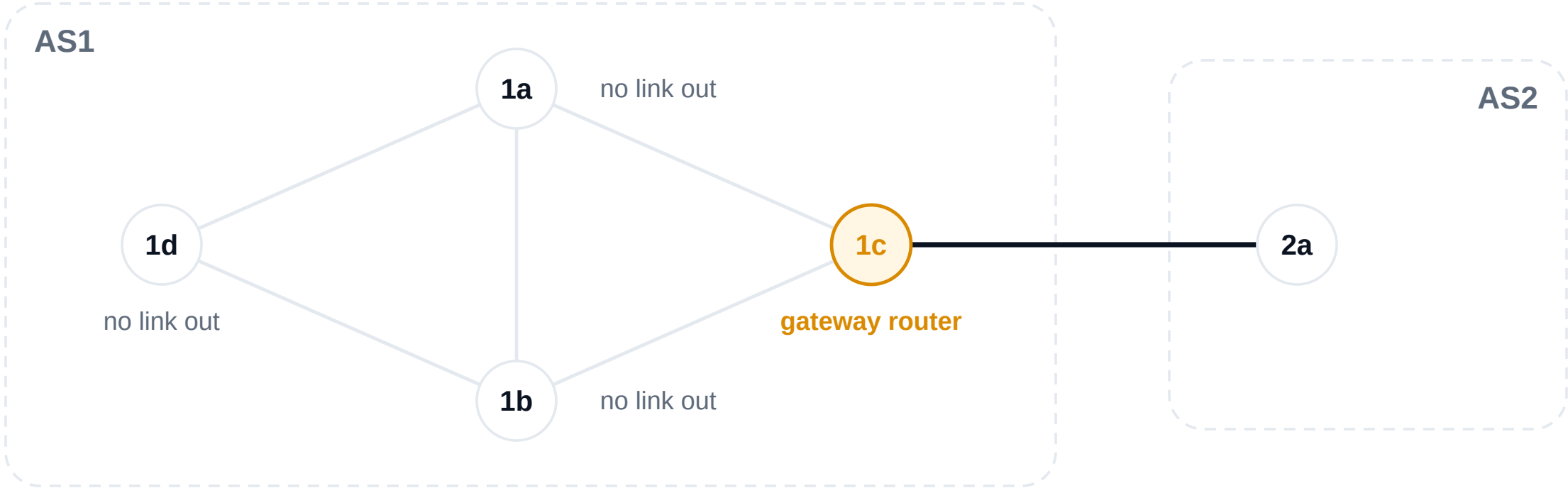
# Only 1c talks to AS2, but 1d forwards packets for x too

04

## iBGP

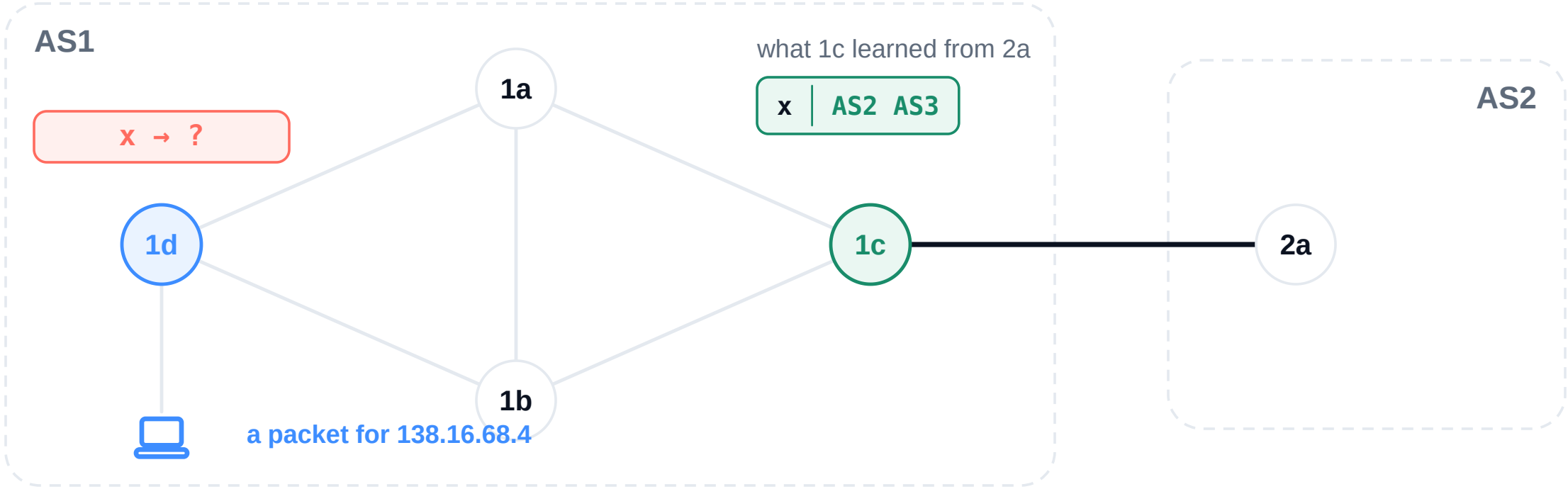
The route has arrived at the edge of AS1. Every router inside needs it.

# Which routers in AS1 have a link to another AS?



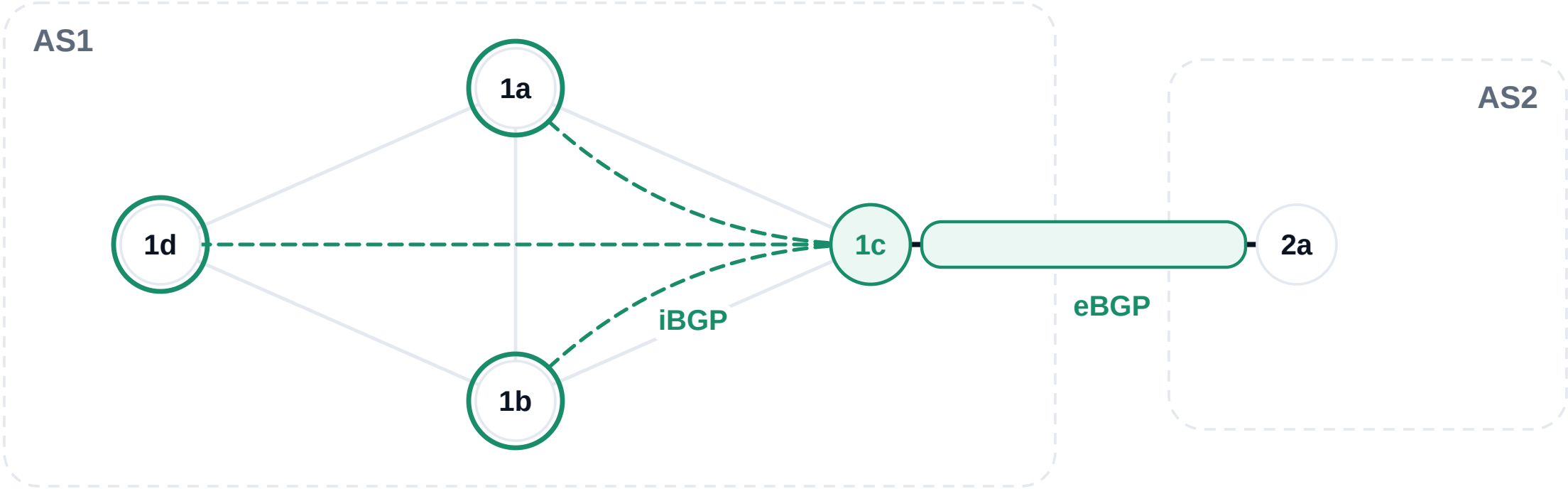

**Only 1c, the gateway router. 1a, 1b and 1d have no link out of AS1.**

# Does 1d need the route to x?




**Yes. 1d must pick a link for every packet to x, and it has never heard of x.**

# 1c opens a BGP session to every router in AS1




Press → for each session, then to name the two kinds.

**Inside an AS, BGP runs over iBGP sessions; between ASes, over eBGP sessions.**

# What changes on each kind of session?

### eBGP: 2a in AS2 → 1c in AS1




SENT

| x | AS3 |
|---|---|

RECEIVED

| x | AS2 AS3 |
|---|---|

AS2 added

### iBGP: 1c → 1d, both in AS1




SENT

| x | AS2 AS3 |
|---|---|

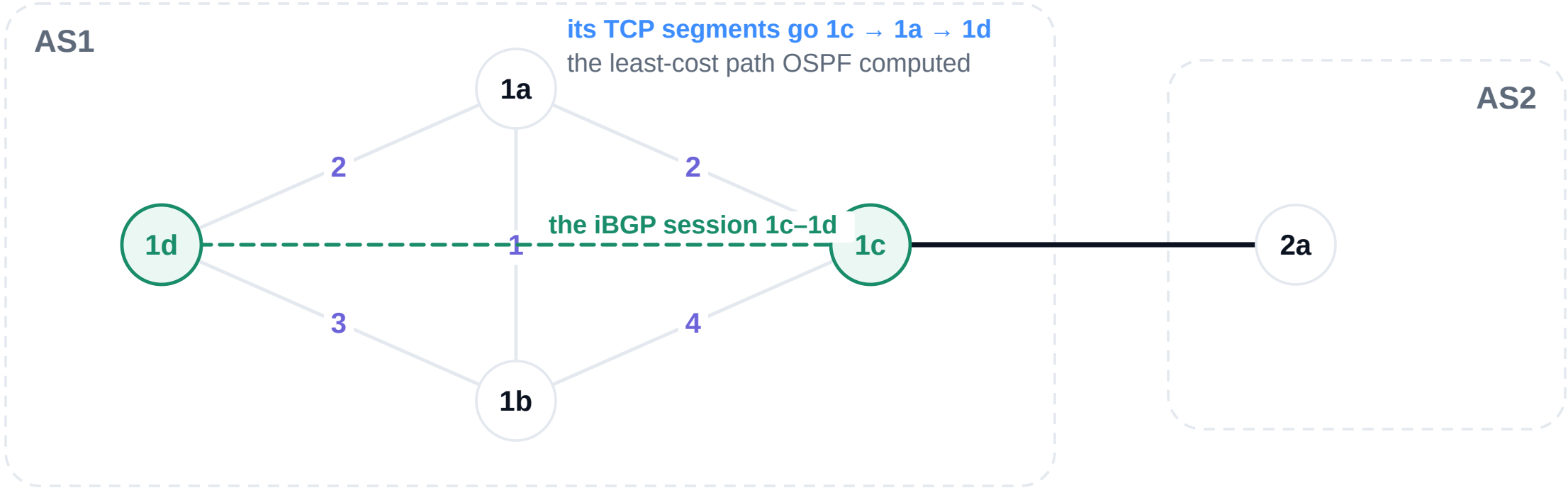
RECEIVED

| x | AS2 AS3 |
|---|---|

unchanged

**An AS adds its number only on eBGP. Over iBGP a route arrives unchanged.**

# 1c and 1d have no direct link

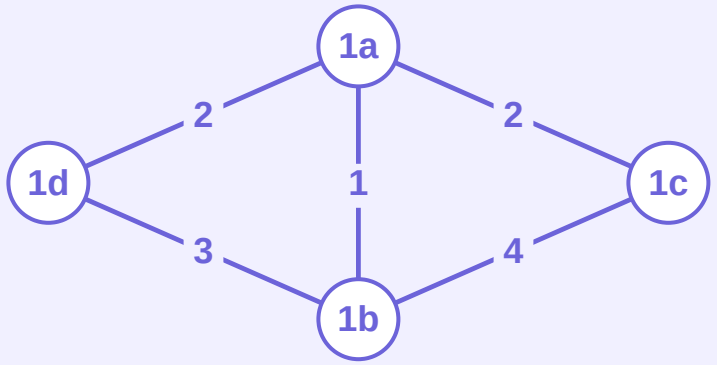



**An iBGP session is a TCP connection carried over the paths OSPF computed.**

# Is iBGP AS1's routing protocol?

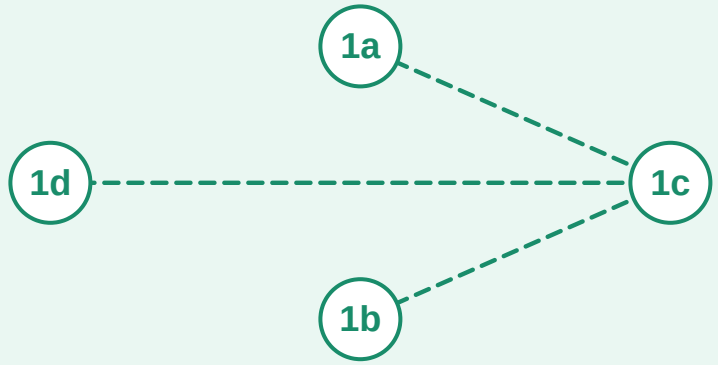
~~"iBGP computes the paths inside AS1"~~

**OSPF: how to reach any address inside**




a packet to 1c: 1d → 1a → 1c

**iBGP: which outside prefixes exist**

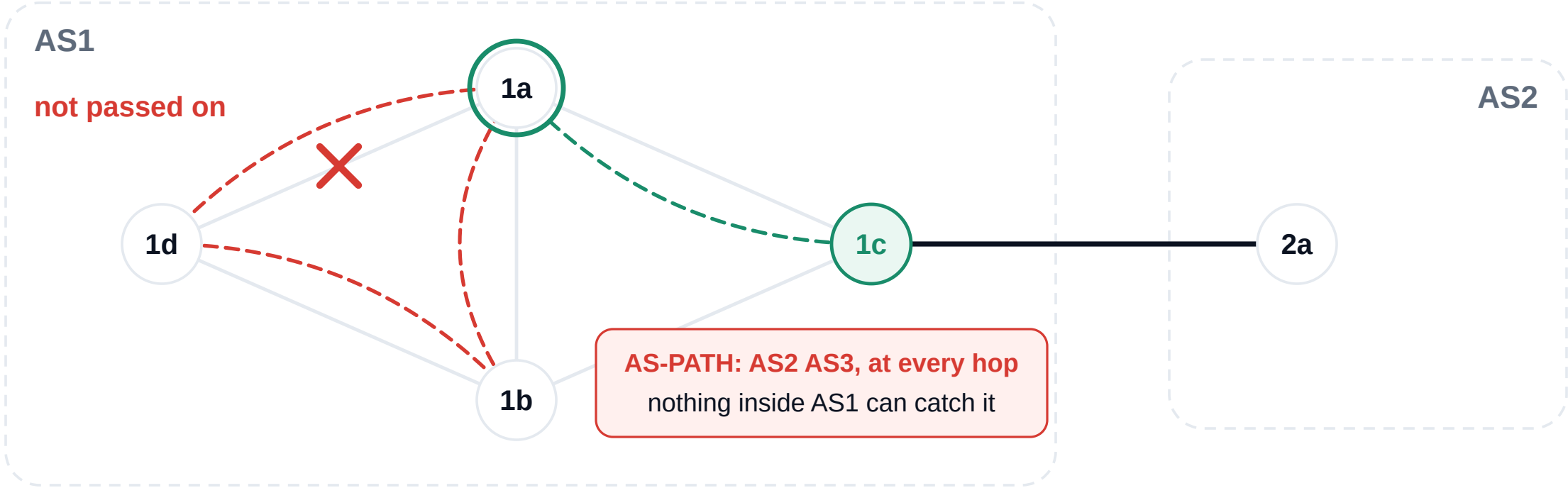



x: reach it through 1c

Press → for the usual mistake.

**No. OSPF computes every path inside AS1. iBGP only says which outside prefixes exist.**

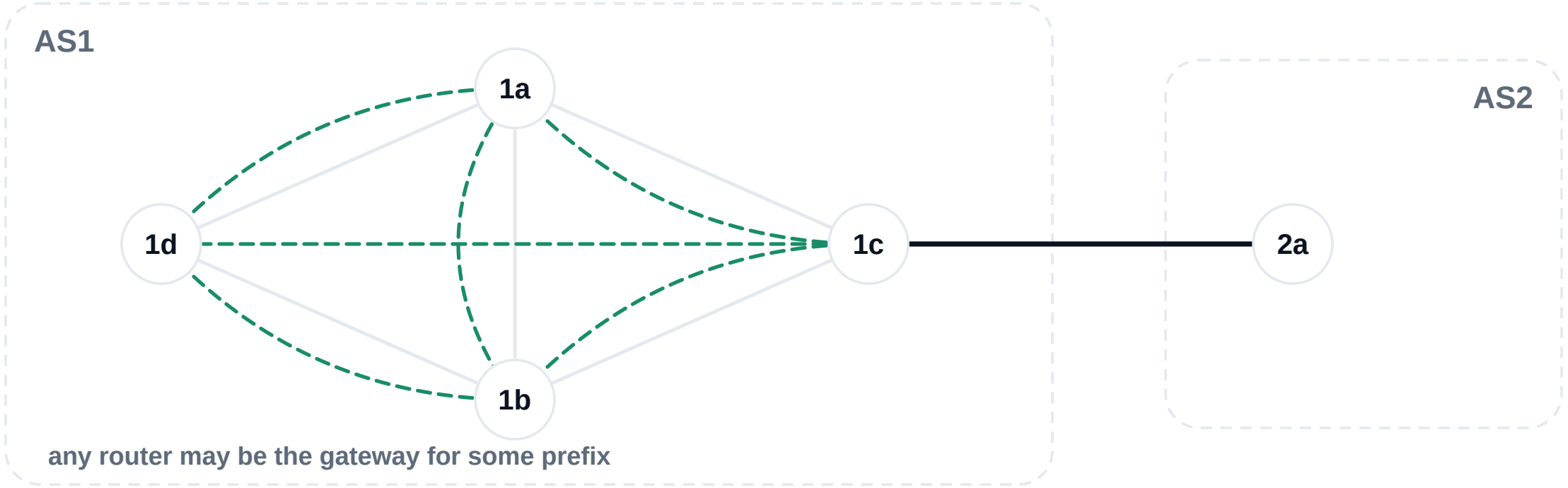
# Could 1a pass on to 1d what it heard from 1c?




Press →: 1c tells 1a. Again: suppose routers passed it on. Again: the rule.

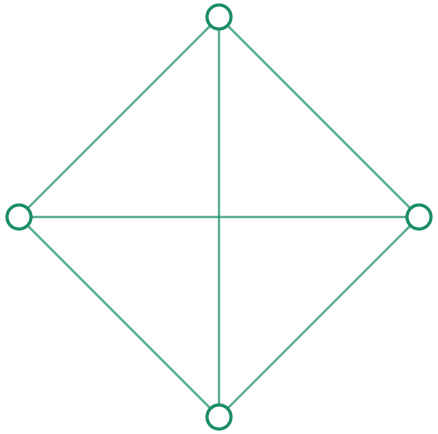
**No. A route learned over iBGP is never passed on over iBGP.**

# So every router must hear each route from its gateway



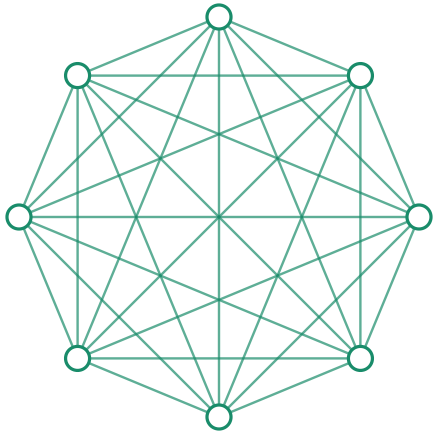

**Every pair of routers in the AS holds an iBGP session: a full mesh.**

# How many sessions is a full mesh?



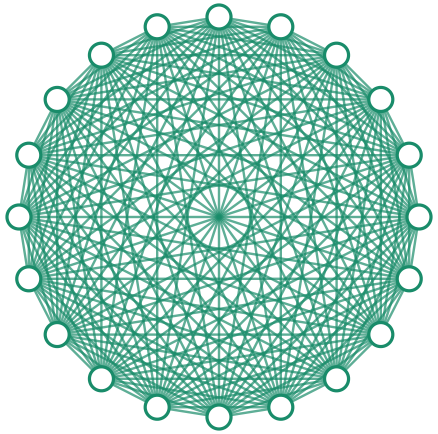
4 routers

6 sessions



8 routers

28 sessions



20 routers

190 sessions

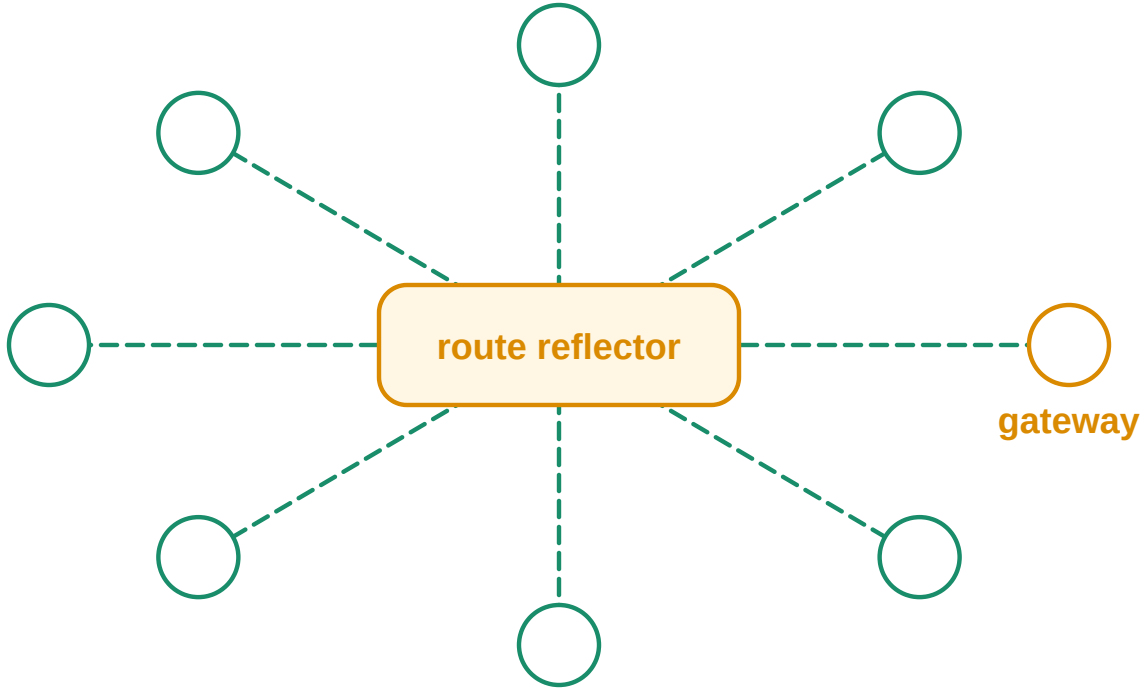
500 routers

124,750

sessions

**n(n − 1)/2: the sessions grow with the square of the number of routers.**

# What operators do instead: route reflectors




A reflector is allowed to pass on what it learned over iBGP. Large ASes use a few, for redundancy.

**Each router keeps one session, to the reflector, instead of one to every router.**

# 1d has a route to x, but a forwarding row needs a port

05

## NEXT-HOP

The AS-PATH says which ASes carry the packet. It does not say which of 1d’s links to use.

# What does 1d hold, and what does its table need?

**1D'S BGP ROUTE FOR X**

| 138.16.68.0/24 | AS2 AS3 |
|---|---|

**1D'S FORWARDING TABLE**

| prefix | port |
|---|---|
| 10.1.1.0/24 | 1 |
| 10.1.2.0/24 | 2 |
| 138.16.68.0/24 | ? |




**A port. The route must name something 1d can turn into one of its ports.**

# What could the route name?

**the next AS**

`AS2`

✗

not something to look up

**a port**

`port 3`

✗

local to one router

**a router**

`2a`

✗

a name, not an address

**an IP address**

`203.0.113.2`

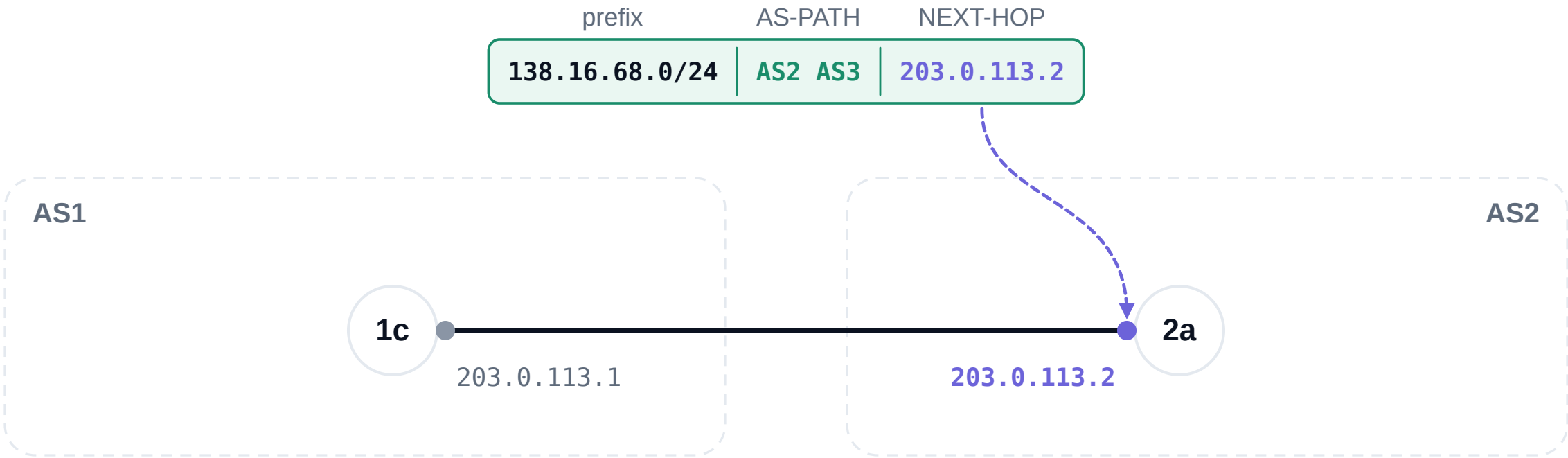
✓

any router can look it up

Press → for each candidate.

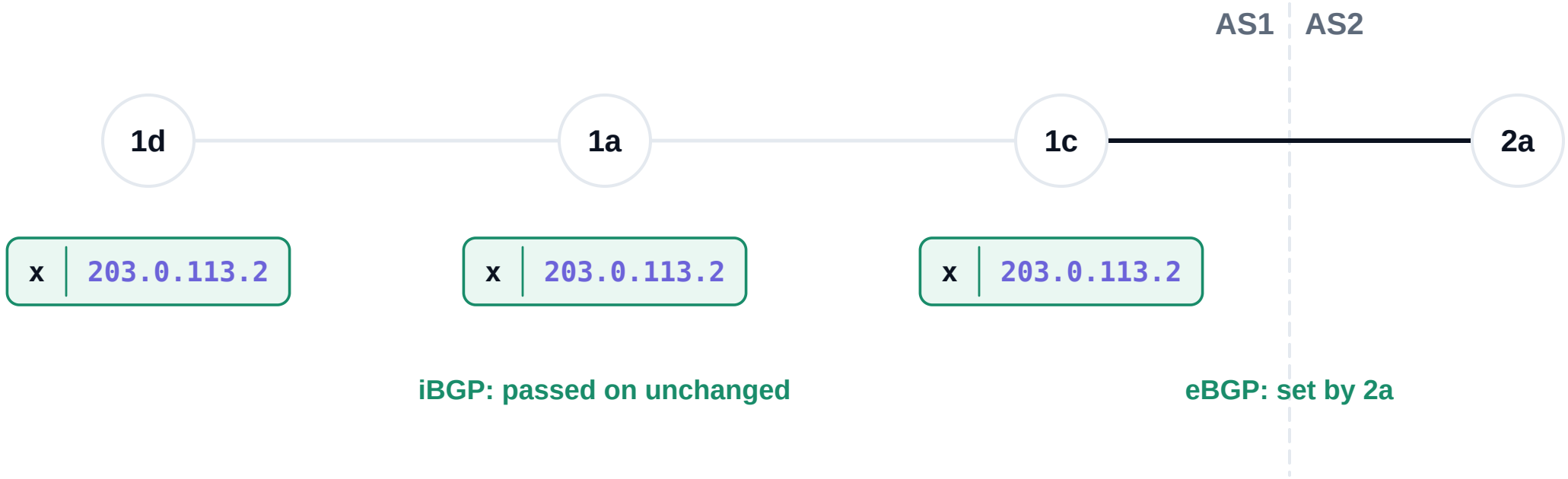
**Not an AS, not a port, not a name: an address every router in AS1 can look up.**

# NEXT-HOP: the address on the far side of the border




**Over eBGP the sender puts its own address on the shared link in the NEXT-HOP.**

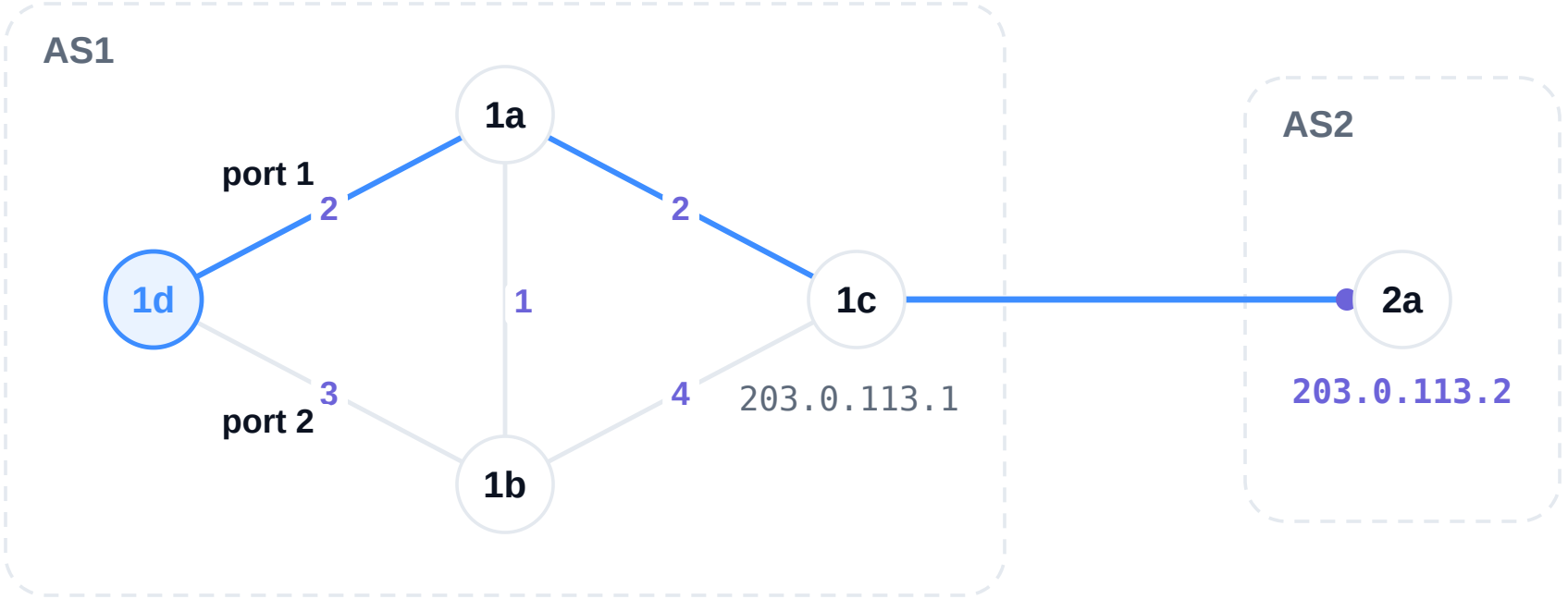
# Inside AS1, the NEXT-HOP does not change




Press →: eBGP from 2a to 1c, then iBGP from 1c to 1a and to 1d.

**Every router in AS1 holds the same address: the entrance to AS2.**

# Step 1: how does 1d reach 203.0.113.2?




**1D'S OSPF TABLE**

| destination | port |
|---|---|
| 10.1.1.0/24 | 1 |
| 10.1.2.0/24 | 2 |
| 203.0.113.0/30 | 1 |

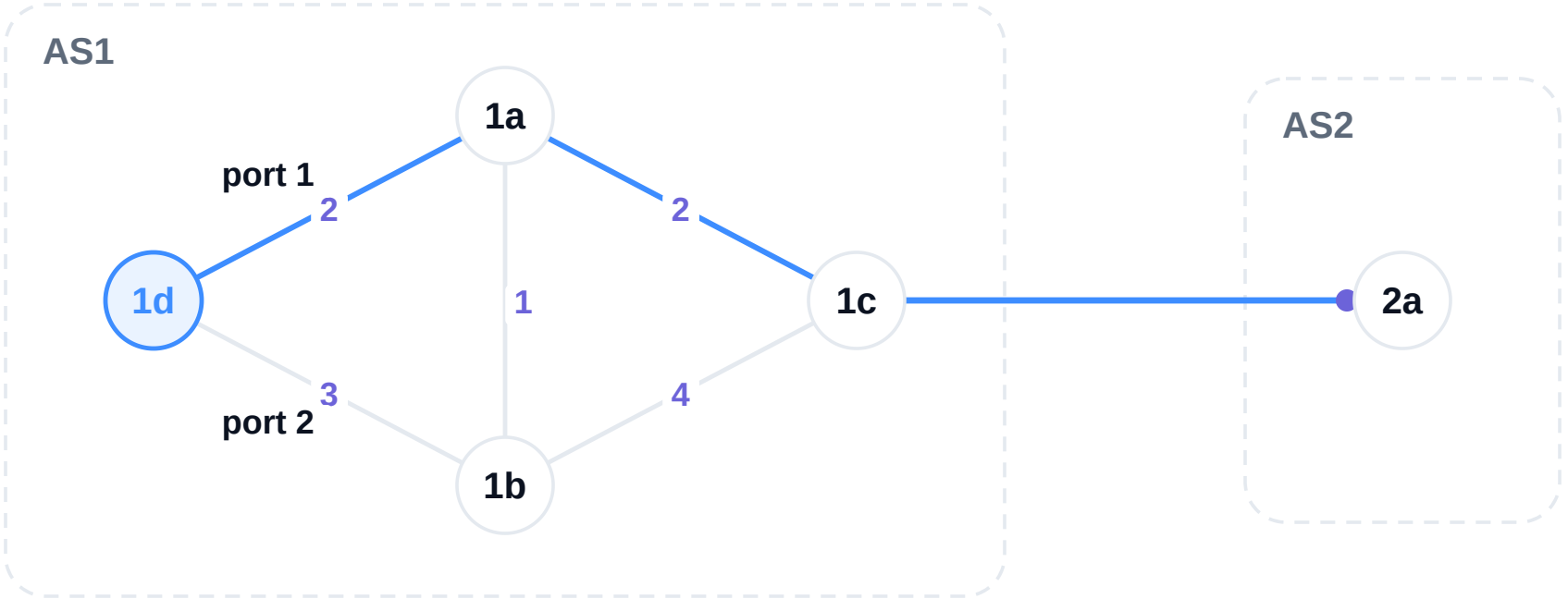
**cost 4, first link: port 1**

1c announces its border link's subnet into OSPF, so every router in AS1 has a row for it.

**OSPF already knows: port 1, the first link of the least-cost path.**

1 Reach | 2 AS-PATH | 3 Session | 4 iBGP | 5 NEXT-HOP | 6 Choosing | 7 Who pays | 8 Hijacks

# Step 2: write the row for x




**1D'S OSPF TABLE**

| destination | port |
|---|---|
| 10.1.1.0/24 | 1 |
| 10.1.2.0/24 | 2 |
| 203.0.113.0/30 | 1 |

**1D'S FORWARDING TABLE**

| prefix | port |
|---|---|
| 10.1.1.0/24 | 1 |
| 138.16.68.0/24 | 1 |

Press → to write the row.

**The port for x is the port toward its NEXT-HOP.**

# Two tables, one row

**BGP: WHICH EXIT**

| prefix | NEXT-HOP |
|---|---|
| 138.16.68.0/24 | 203.0.113.2 |

**OSPF: HOW TO GET THERE**

| destination | port |
|---|---|
| 203.0.113.0/30 | 1 |

**FORWARDING**

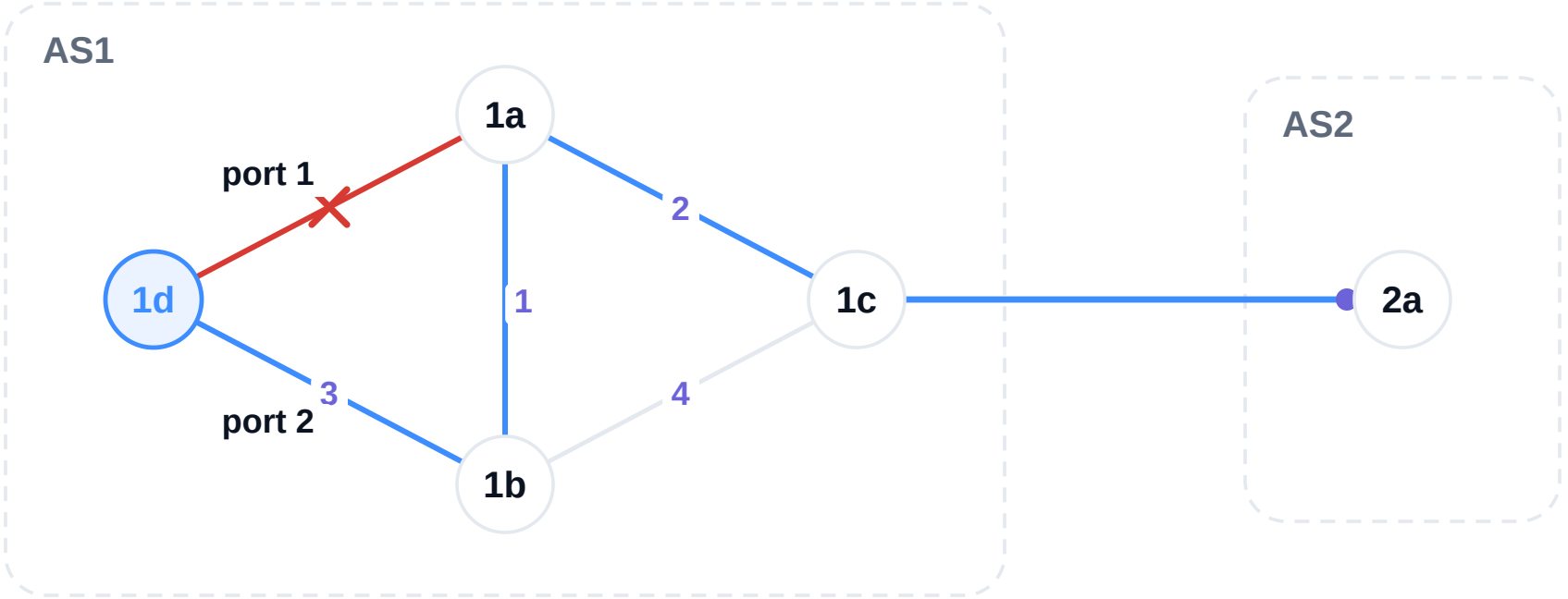
| prefix | port |
|---|---|
| 138.16.68.0/24 | 1 |

**x → NEXT-HOP → port**

Press →: look up the NEXT-HOP, then carry the port across.

**BGP says which exit. OSPF says which port leads to it.**

# A link inside AS1 fails. Does BGP need to say anything?




**1D'S OSPF TABLE**

| destination | port |
|---|---|
| 10.1.1.0/24 | 1 |
| 10.1.2.0/24 | 2 |
| 203.0.113.0/30 | 2 |

**FORWARDING ROW FOR X**

| | |
|---|---|
| 138.16.68.0/24 | 2 |

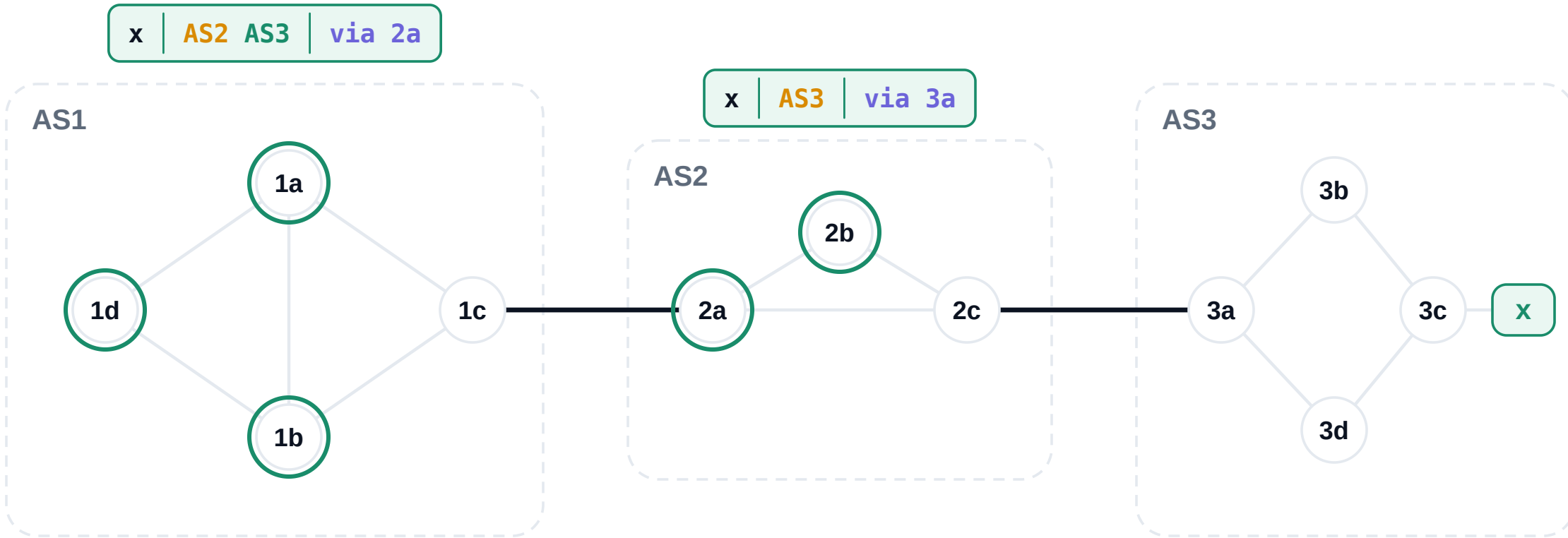
**BGP ROUTE FOR X**

unchanged, no message

Press → to break the link 1d–1a. OSPF finds 1d → 1b → 1a → 1c, cost 6.

**No. The row for x follows OSPF. BGP speaks only when something outside changes.**

# From AS3 to 1d, the whole journey




**eBGP carries a route between ASes, iBGP spreads it inside, OSPF finds the exit.**

1 Reach | 2 AS-PATH | 3 Session | 4 iBGP | 5 NEXT-HOP | 6 Choosing | 7 Who pays | 8 Hijacks

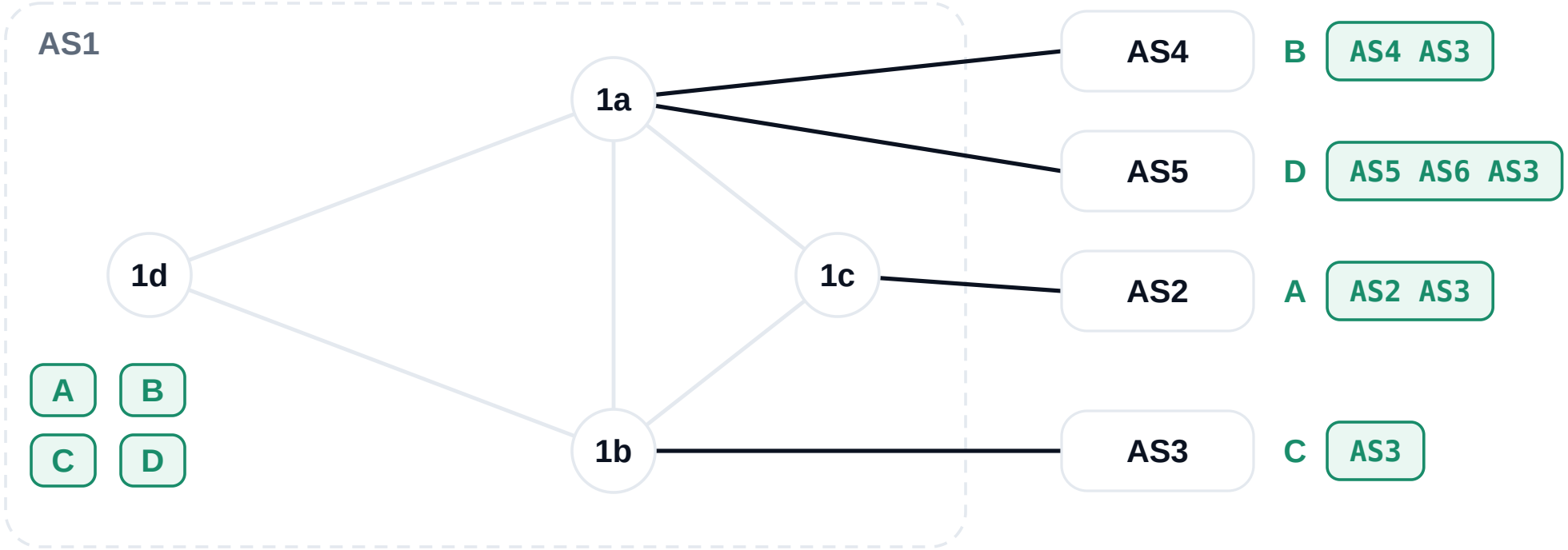
# 1d hears four routes to x, and its table holds one row

06

## Choosing one route

AS1 has more than one neighbour. Each may offer its own route to the same prefix.

# Four routes for x arrive at 1d




Press → for each neighbour's advertisement, carried to 1d over iBGP.

**One prefix, four routes. The forwarding table holds one row for x.**

# Is the shortest AS-PATH the winner?

| route | AS-PATH | NEXT-HOP |
|---|---|---|
| A | AS2 AS3 | 2a |
| B | AS4 AS3 | 4a |
| C | AS3 | 3d |
| D | AS5 AS6 AS3 | 5a |

**shortest: one AS**

**AS1's operator:**

AS3 bills AS1 for every gigabyte on that link. Send as little as possible that way.

Press → for the shortest route, then for what the operator knows about it.

**Not first. Before length, the AS applies its own preference.**

# Rule 1: the highest LOCAL-PREF

| route | AS-PATH | NEXT-HOP | LOCAL-PREF |
|---|---|---|---|
| A | AS2 AS3 | 2a | 100 |
| B | AS4 AS3 | 4a | 100 |
| C | AS3 | 3d | 50 |
| D | AS5 AS6 AS3 | 5a | 100 |

**LOCAL-PREF**

a number the operator sets
where the route enters AS1,
carried to every router
over iBGP, never to another AS
**highest wins**

Press → to show the numbers, then to apply the rule.

**The operator’s number decides first. C is out, though its path is the shortest.**

# Rule 2: the shortest AS-PATH among the rest

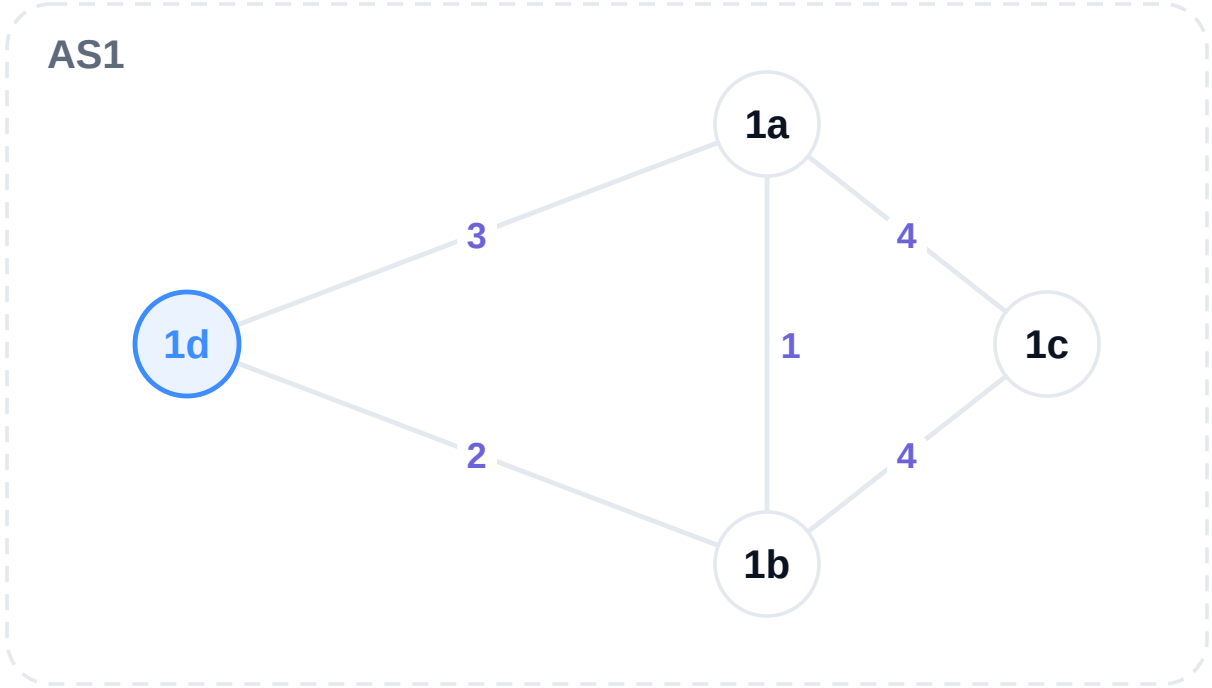
| route | AS-PATH | NEXT-HOP | LOCAL-PREF | length |
|---|---|---|---|---|
| A | AS2 AS3 | 2a | 100 | 2 |
| B | AS4 AS3 | 4a | 100 | 2 |
| C | AS3 | 3d | 50 | 1 |
| D | AS5 AS6 AS3 | 5a | 100 | 3 |

**A and B tie at two ASes**

Press → to apply the rule.

**D is out: three ASes. A and B tie at two.**

# Rule 3: the nearest NEXT-HOP



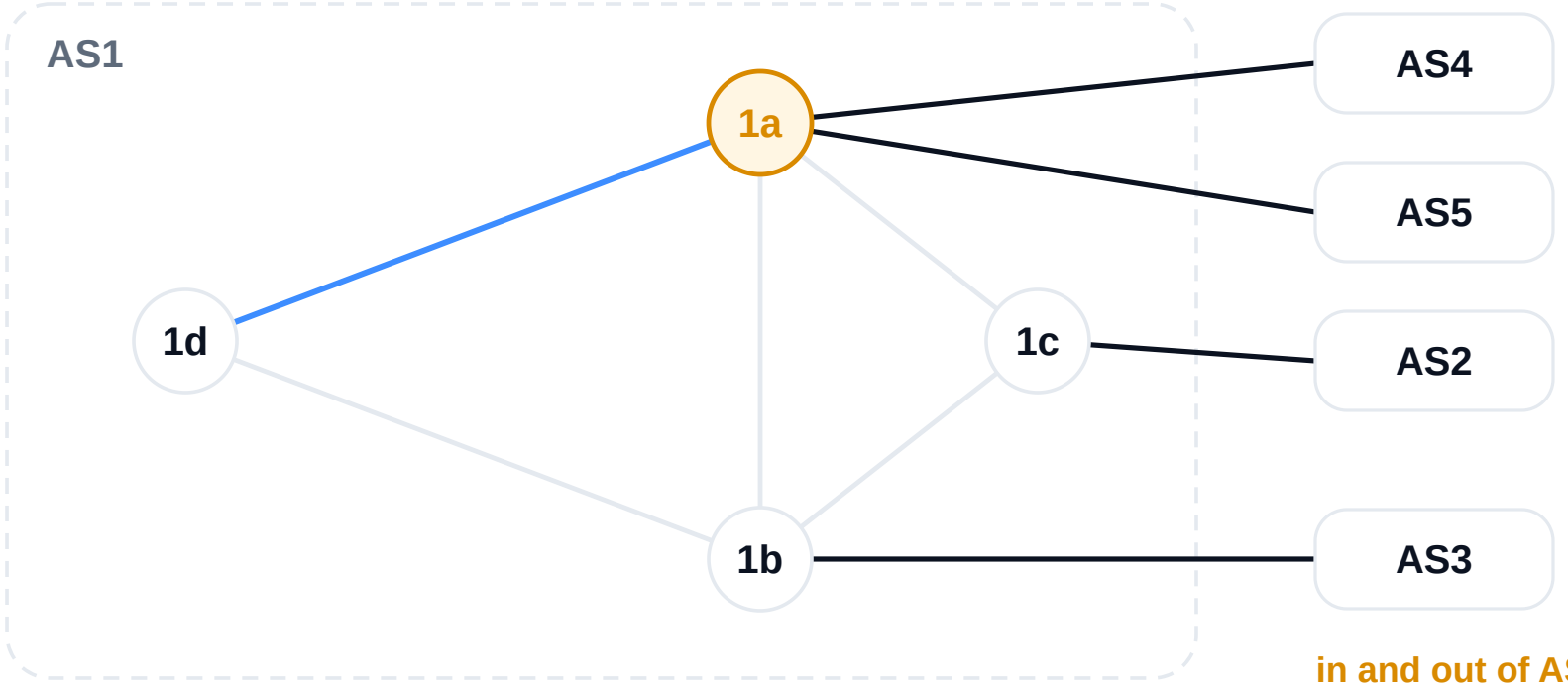

| route | exit | cost inside AS1 |
|---|---|---|
| A | 1c | 2 + 4 = 6 |
| B | 1a | 3 |

**B wins: its exit is nearer**

Costs are AS1's own OSPF costs from 1d to each exit. Press → to apply the rule.

**B: its exit is 3 away inside AS1; A's is 6.**

# Why leave by the nearest exit?

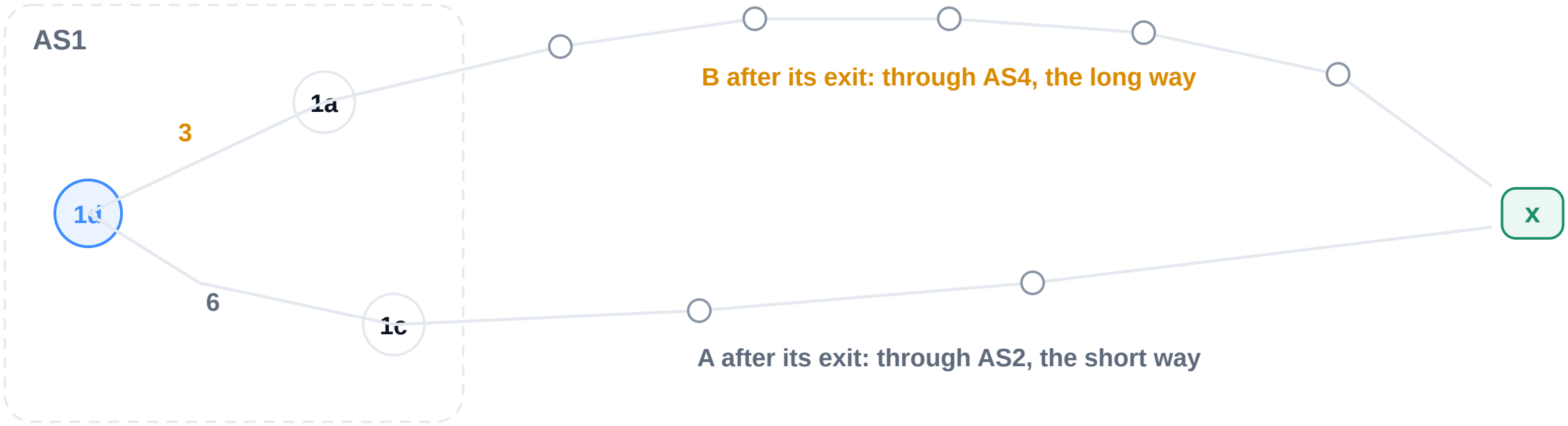



**in and out of AS1 over as few links as possible**

The name: hot-potato routing. Hand the packet on before it burns your hands.

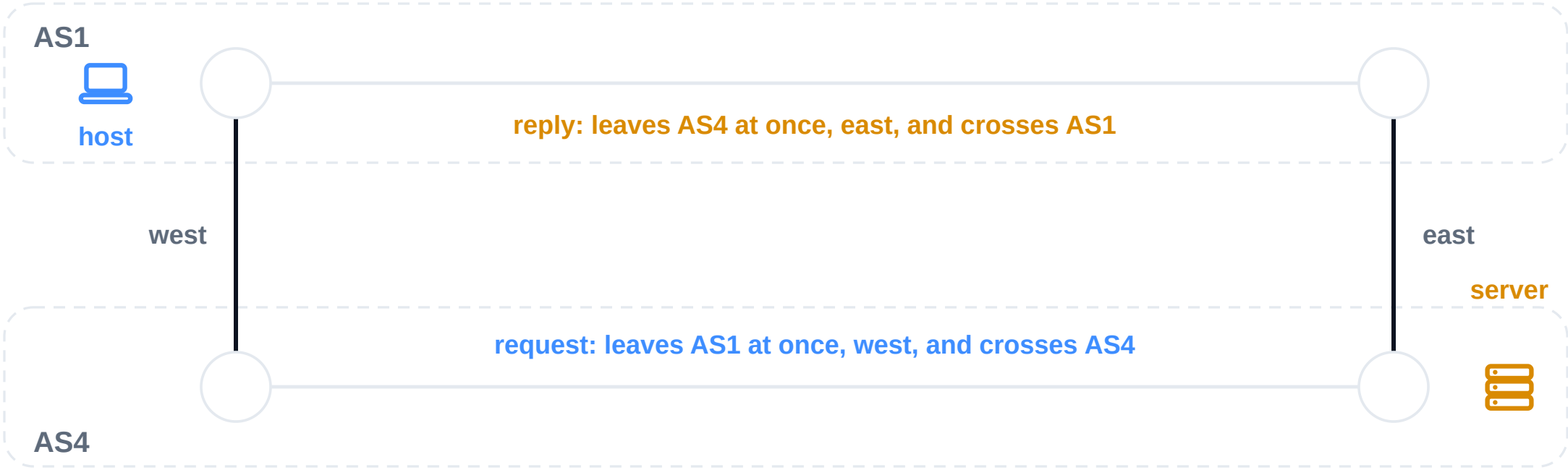
**It spends the least of AS1’s own links on traffic that is not AS1’s.**

# What does the nearest exit ignore?




**Everything after the exit. Rule 3 keeps AS1's cost low, not the whole path's.**

# The reply comes back a different way




A traceroute shows only the way out. The way back is invisible to it.

**Each AS hands off at its own nearest exit, so the two directions differ.**

# Rule 4: if all else ties, the lowest router ID

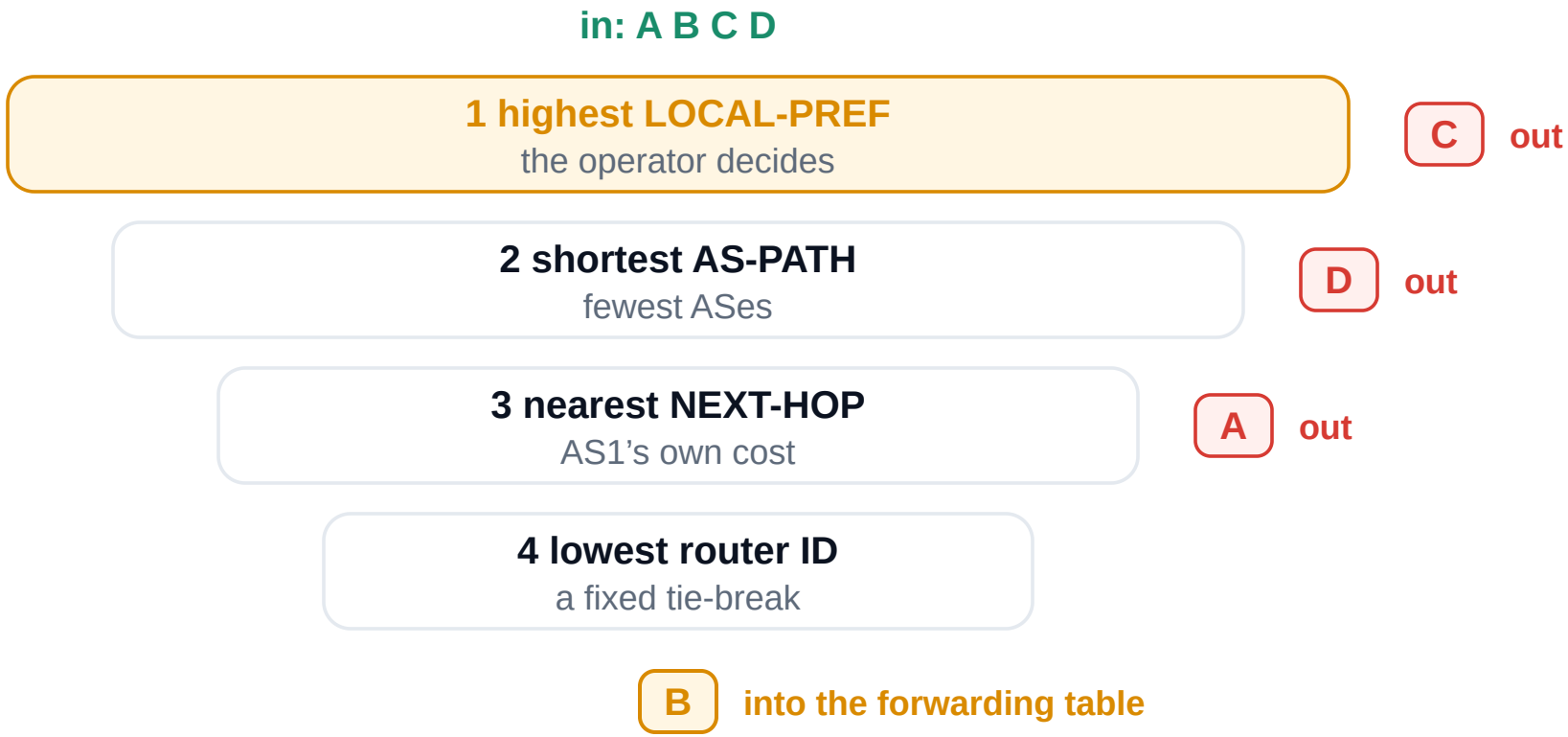
| route | LOCAL-PREF | length | exit cost | neighbour's BGP ID |
|---|---|---|---|---|
| E | 100 | 2 | 3 | 10.0.0.7 |
| F | 100 | 2 | 3 | 10.0.0.1 |

**F: the lower identifier**

Real routers insert a few more steps (origin, MED, eBGP before iBGP). The first three are what matter here.

**A fixed tie-break, so a router always makes the same choice.**

# What the order says




**The operator's preference comes before anything the protocol measures.**

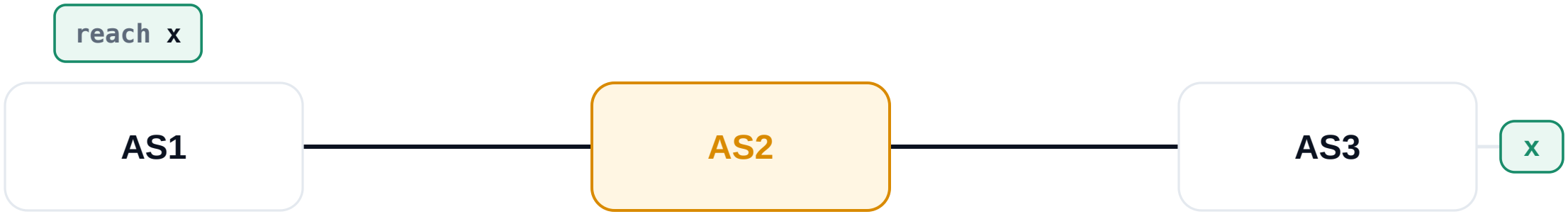
# AS2 can reach x. Will it tell AS1?

07

## Who pays whom

Nothing in BGP makes an AS advertise a route it has.

# Suppose AS2 tells AS1



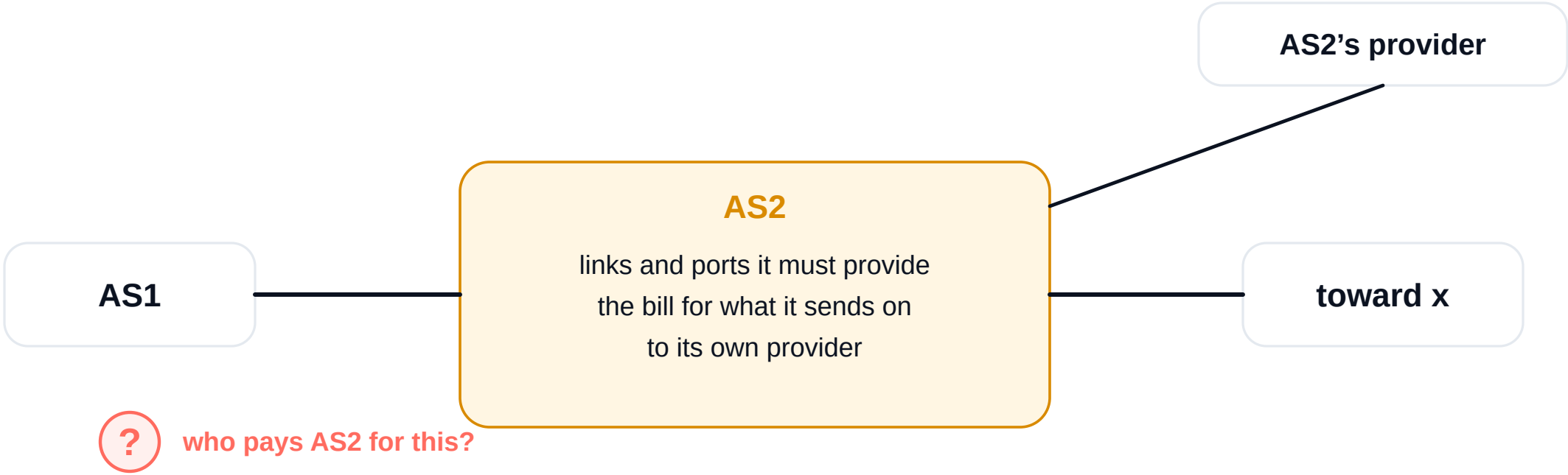

every packet AS1 sends to x now crosses AS2

Press → to advertise, then again for the traffic that follows.

**Advertising a route means accepting the traffic that follows it.**

1 Reach | 2 AS-PATH | 3 Session | 4 iBGP | 5 NEXT-HOP | 6 Choosing | 7 Who pays | 8 Hijacks

# What does carrying it cost AS2?



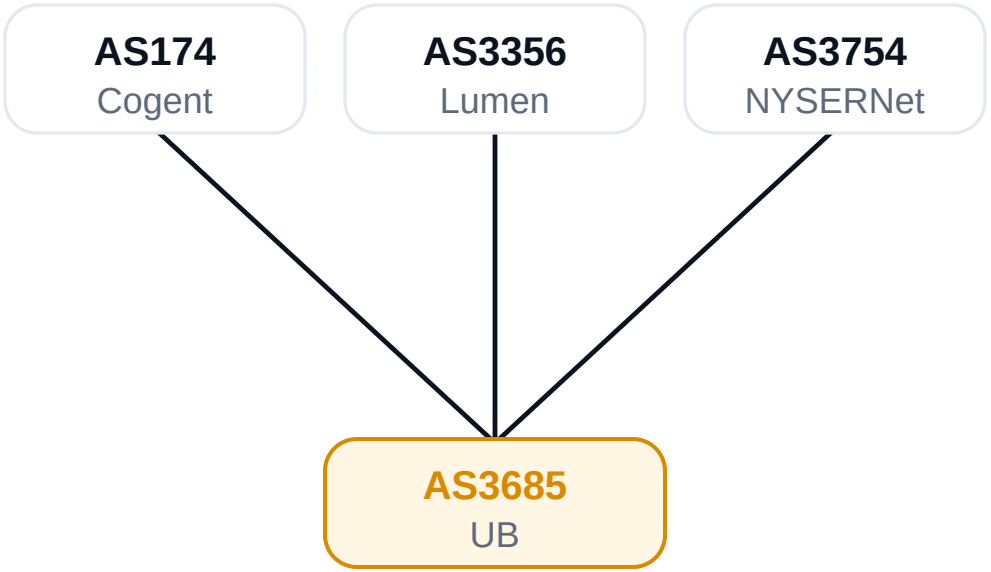

**Capacity, and money: AS2 pays its own provider for the traffic it passes on.**

# A customer pays its provider




carries its traffic to and from everyone

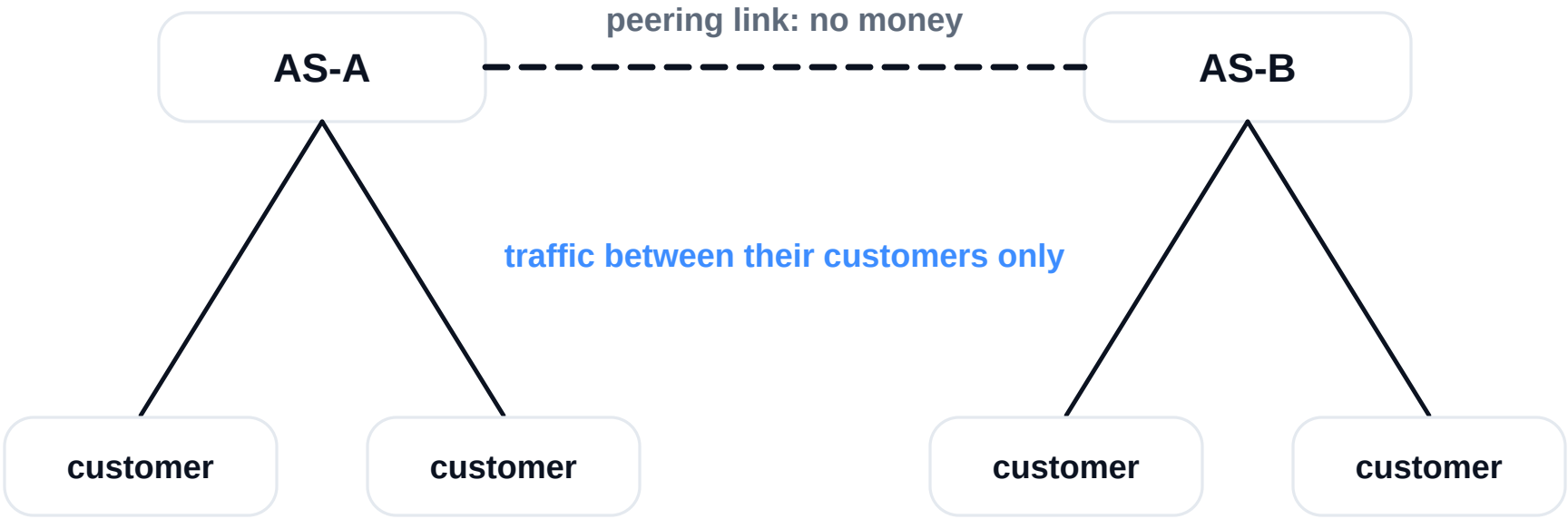
UB'S UPSTREAM NEIGHBOURS IN BGP




AS3685's upstream neighbours as seen in BGP data (RIPEstat, 23 September 2026).

**The customer pays; the provider carries its traffic to and from the whole internet.**

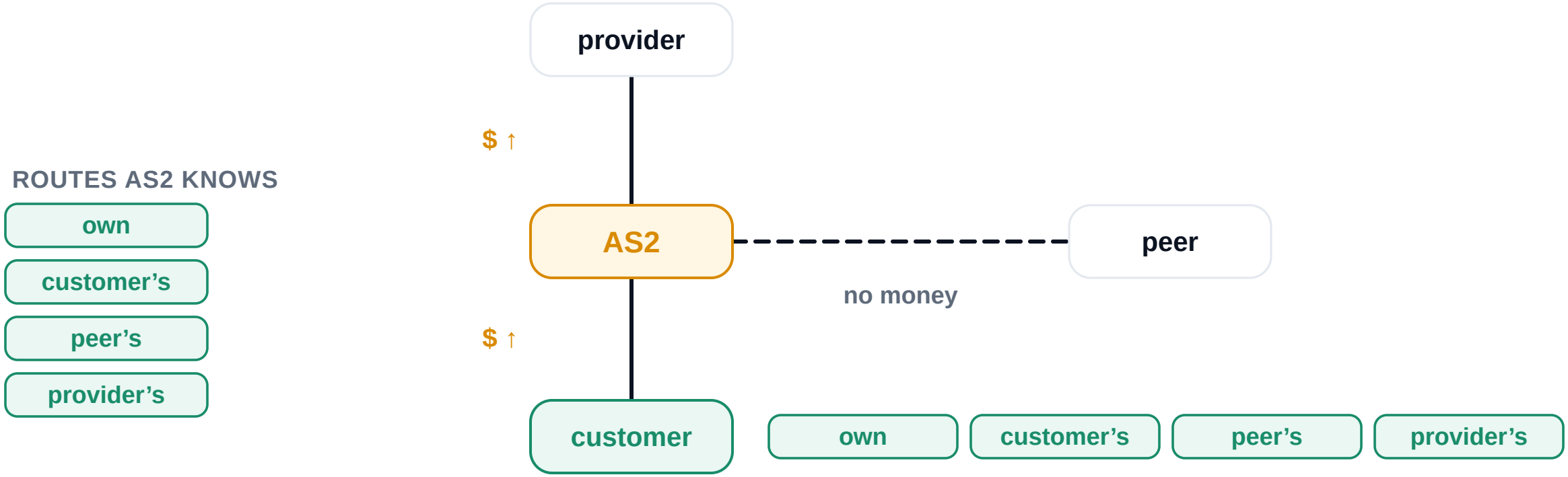
# Two ASes of similar size connect, and neither pays




Each saves what its provider would have charged for this traffic.

**Peering: each carries traffic between its own customers and the other's, free.**

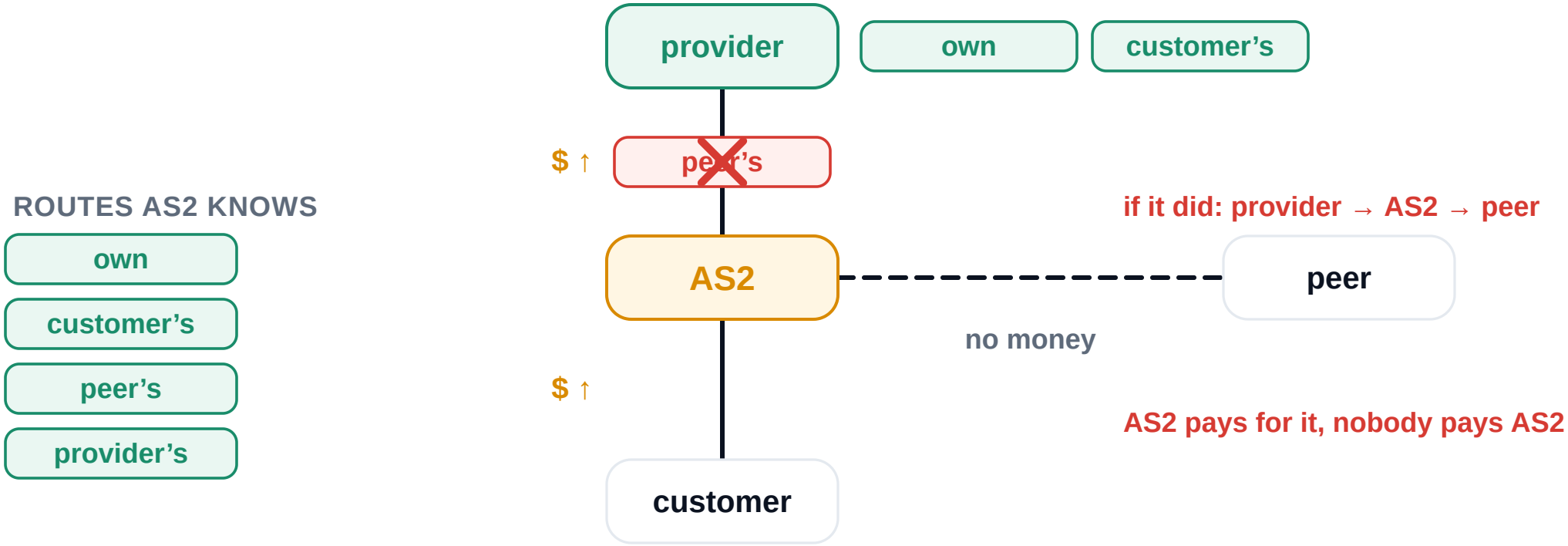
# What should AS2 tell a customer?




**Everything it knows. The customer pays to reach all of it.**

1 Reach | 2 AS-PATH | 3 Session | 4 iBGP | 5 NEXT-HOP | 6 Choosing | 7 Who pays | 8 Hijacks

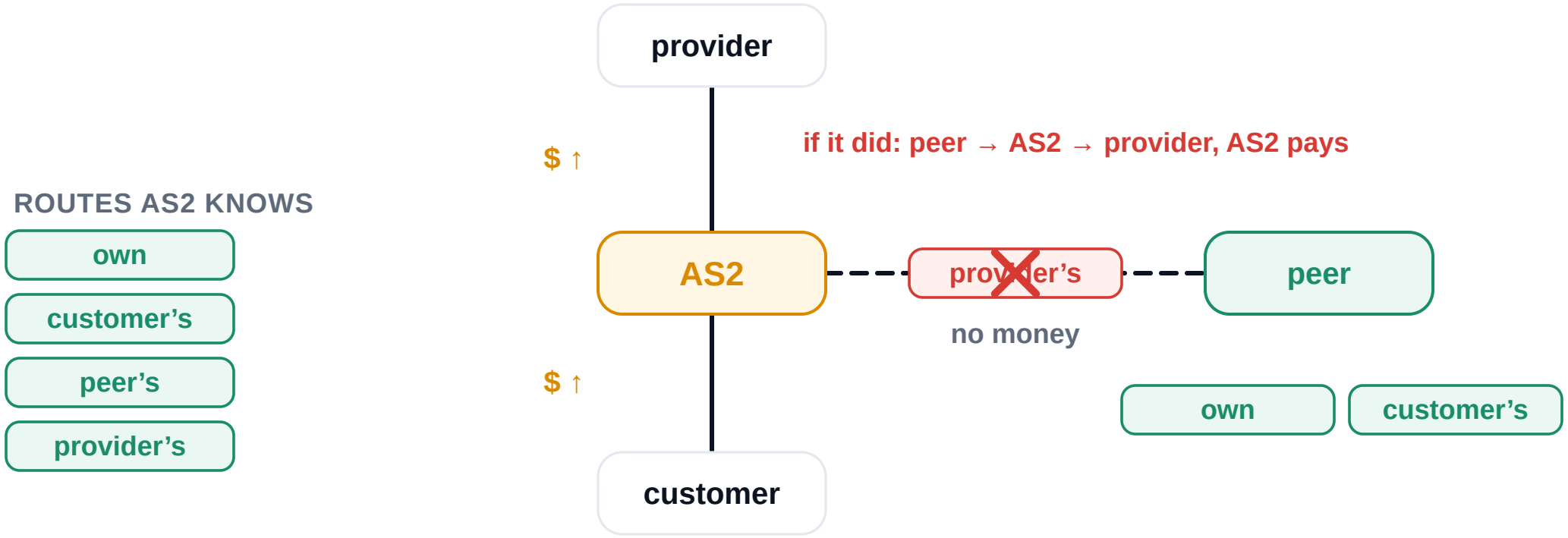
# What should AS2 tell its provider?




Press → for what would happen if AS2 passed on its peer's routes.

**Its own and its customers' routes only. Anything else is traffic nobody pays for.**

# What should AS2 tell its peer?




Press → for what would happen if AS2 passed on its provider's routes.

**The same: its own and its customers' routes. Never a provider's or another peer's.**

# The three rules on one picture

| learned from | tell a customer | tell a peer | tell a provider |
|---|---|---|---|
| a customer | ✓ | ✓ | ✓ |
| a peer | ✓ | ✗ | ✗ |
| a provider | ✓ | ✗ | ✗ |

Press → for each row.

**Customer routes go to everyone. Peer and provider routes go only to customers.**

# Which route to the same prefix does AS2 prefer?

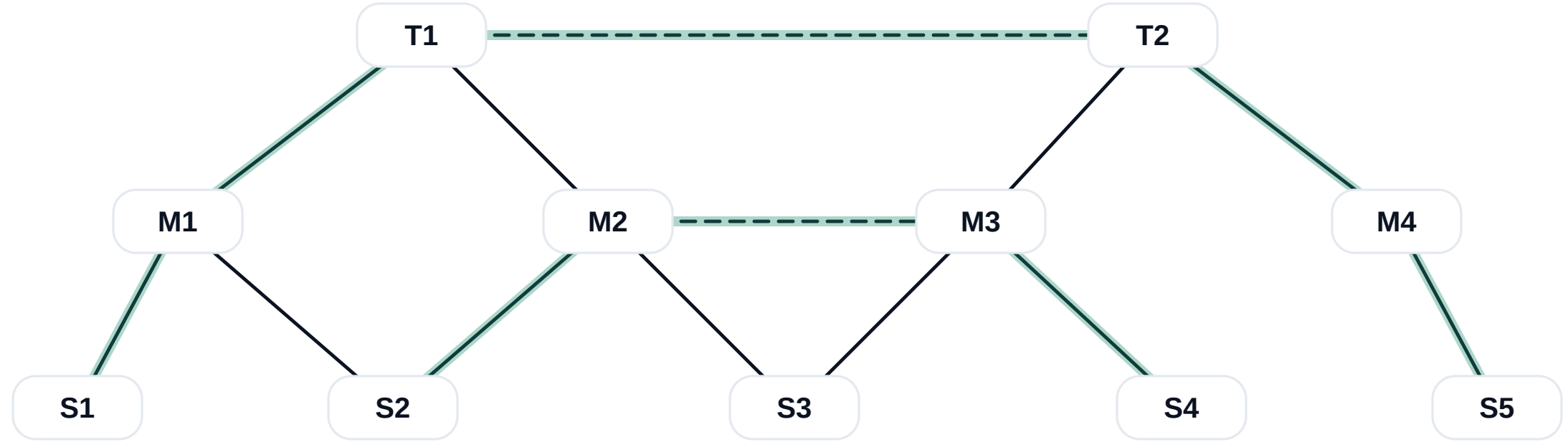
| route to p learned from | money | LOCAL-PREF |
|---|---|---|
| a customer | the customer pays AS2 | 200 |
| a peer | free | 150 |
| a provider | AS2 pays the provider | 100 |

**rule 1 of route selection picks the customer's route**

Press → to apply rule 1 from Choosing one route.

**Customer first, then peer, then provider: LOCAL-PREF follows the money.**

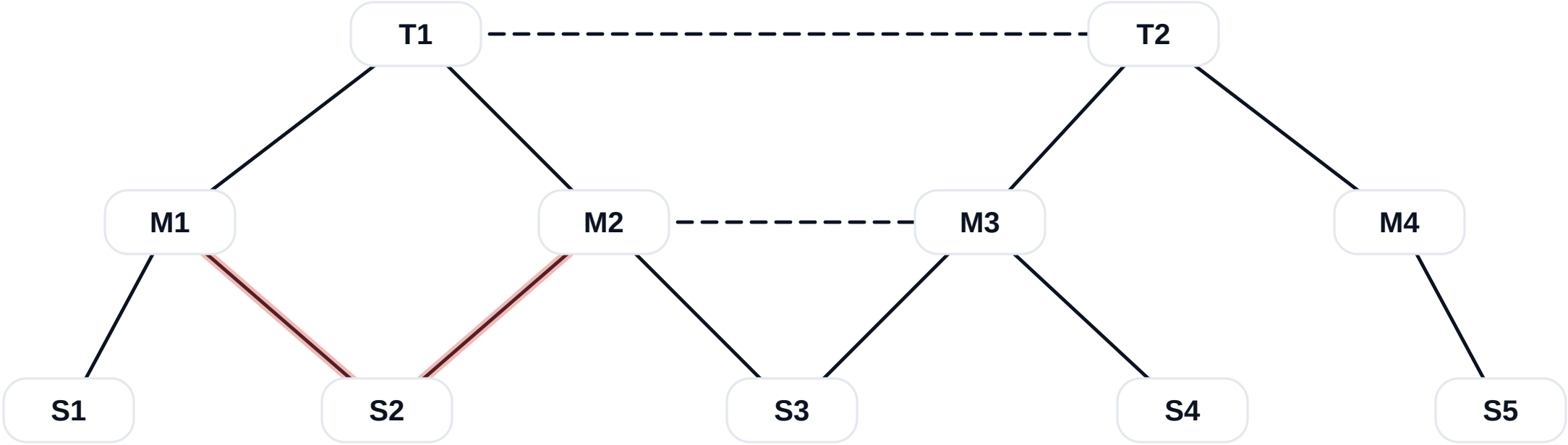
# Which paths can exist?




Press → for a second path, across a lower peering link.

**Every path climbs, crosses at most one peering link, then descends.**

# Why is there never a valley?




**S2 would pay both M1 and M2 to carry their traffic**

Nothing in BGP enforces these rules. They come from contracts, configured at every AS.

**The AS at the bottom would pay both its providers, so it never offers that route.**

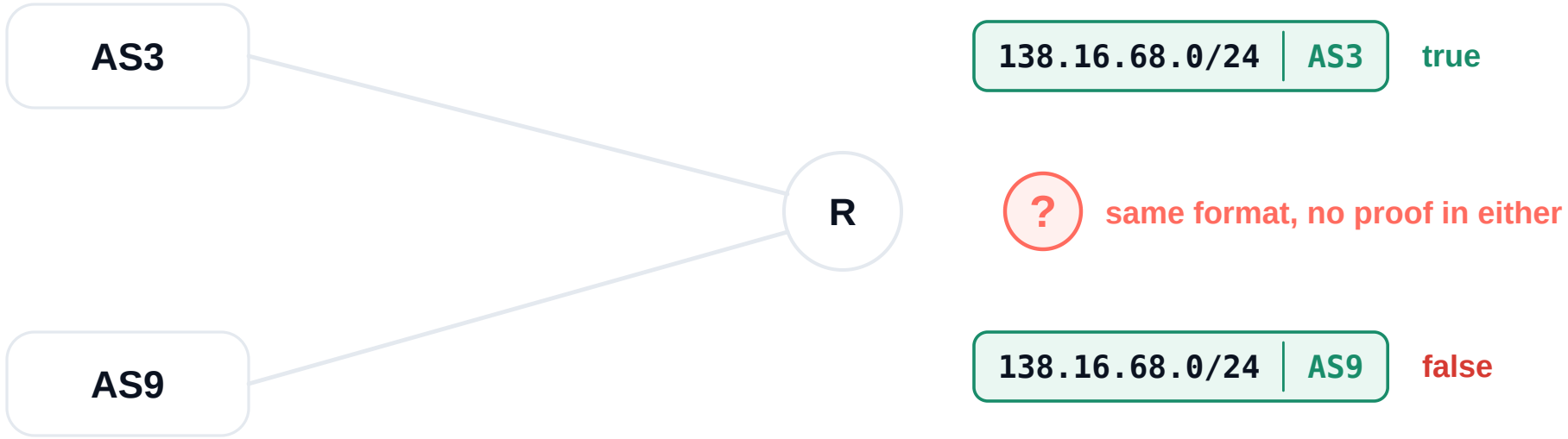
# A router believes whatever its neighbours advertise

08

## Hijacks and RPKI

Nothing in an advertisement proves that the origin AS holds the prefix.

# Who checks that AS3 really holds x?

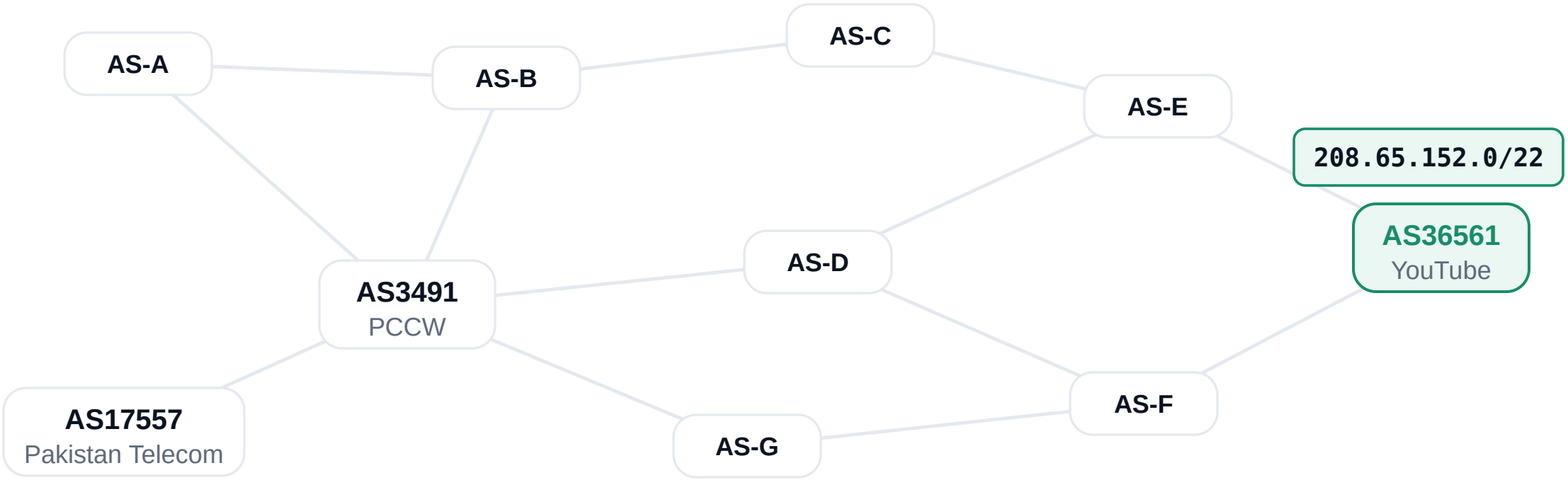



The same weakness as distance vector (What happens when one router gets it wrong?).

**Nobody. Any AS can originate any prefix, and its neighbours pass it on.**

1 Reach | 2 AS-PATH | 3 Session | 4 iBGP | 5 NEXT-HOP | 6 Choosing | 7 Who pays | 8 Hijacks

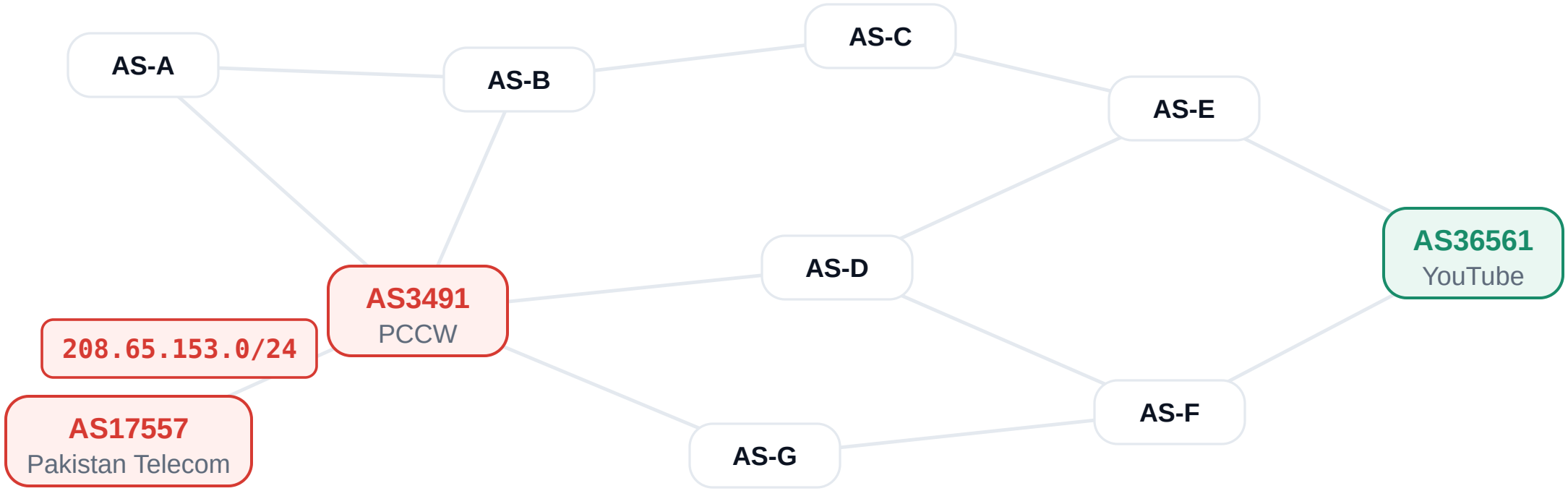
# YouTube: 208.65.152.0/22, from AS36561




**Every router on the internet had a row for YouTube's /22.**

1 Reach | 2 AS-PATH | 3 Session | 4 iBGP | 5 NEXT-HOP | 6 Choosing | 7 Who pays | 8 Hijacks

# 24 Feb 2008, 18:47 UTC: AS17557 announces 208.65.153.0/24




Meant to block YouTube inside Pakistan. It went to its provider PCCW (AS3491), which passed it on. Press →.

**A second route, for a smaller block inside YouTube's, spread worldwide in minutes.**

1 Reach | 2 AS-PATH | 3 Session | 4 iBGP | 5 NEXT-HOP | 6 Choosing | 7 Who pays | 8 Hijacks

# A packet for 208.65.153.7: which row matches?

a packet for 208.65.153.7 at any router that heard both

| prefix | toward | matches 208.65.153.7? |
|---|---|---|
| 208.65.152.0/22 | AS36561, YouTube | ✓ 22 bits |
| 208.65.153.0/24 | AS17557, Pakistan | ✓ 24 bits |

**longest prefix match: the /24 wins**

as in the Forwarding lecture: the longest matching row decides

Press → to test each row, then to pick.

**Both rows match. Longest prefix match picks the /24, at every router that heard it.**

1 Reach | 2 AS-PATH | 3 Session | 4 iBGP | 5 NEXT-HOP | 6 Choosing | 7 Who pays | 8 Hijacks

# Why did route selection not protect YouTube?

**what route selection compares**

| 208.65.152.0/22 | AS36561 |
|---|---|

| 208.65.152.0/22 | AS9 AS36561 |
|---|---|

**two routes for the same prefix**

one of them becomes the row

**what happened**

| 208.65.152.0/22 | AS36561 |
|---|---|

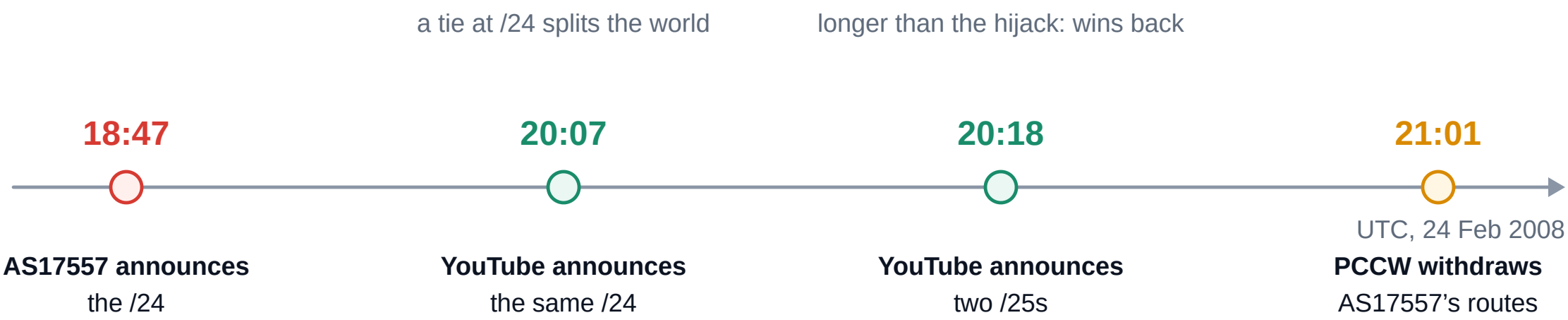
| 208.65.153.0/24 | AS3491 AS17557 |
|---|---|

**two different prefixes: nothing to compare**

both become rows; each packet takes the longer

**The /24 never competed with the /22. Both went in; every packet took the longer match.**

# How it ended

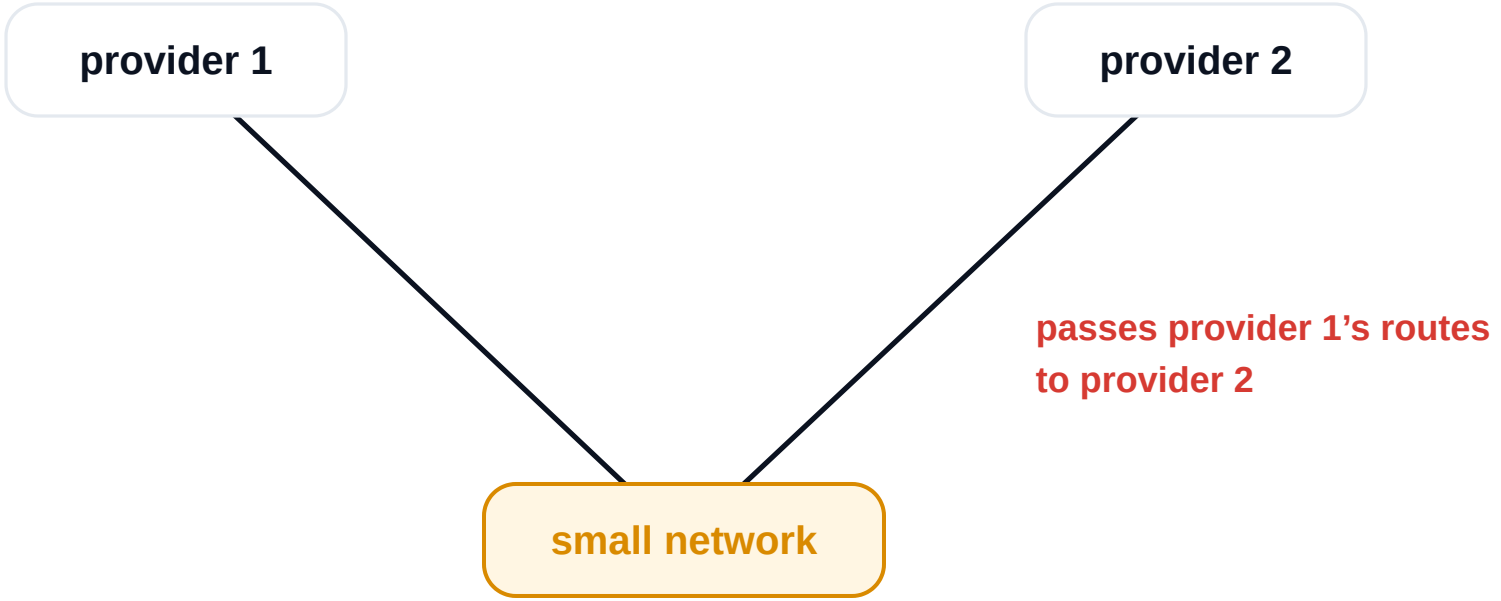



RIPE NCC RIS case study. Press → for each event.

**About two hours, ended by hand: YouTube out-matched the /24, then PCCW cut it off.**

1 Reach | 2 AS-PATH | 3 Session | 4 iBGP | 5 NEXT-HOP | 6 Choosing | 7 Who pays | 8 Hijacks

# Not every bad route is an attack: a route leak

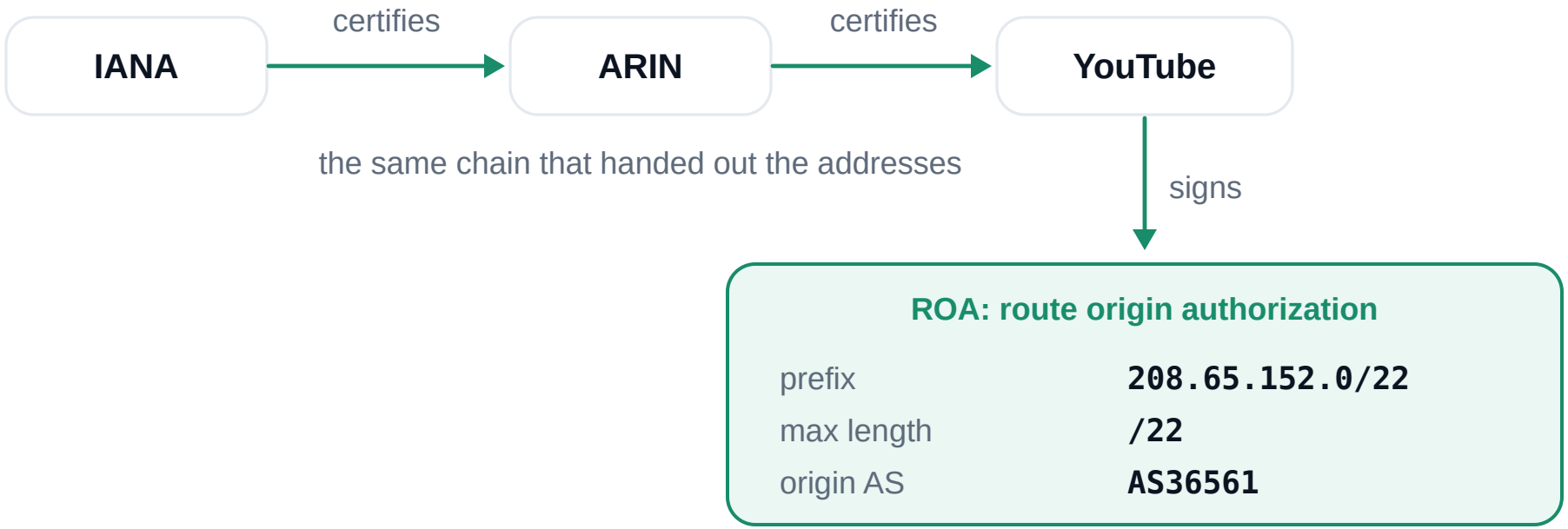



24 June 2019: routes leaked this way were accepted by Verizon (AS701); traffic to Cloudflare and others was pulled through a small network.

**A leak breaks the export rules of Who pays whom. Routers believe it just the same.**

1 Reach | 2 AS-PATH | 3 Session | 4 iBGP | 5 NEXT-HOP | 6 Choosing | 7 Who pays | 8 Hijacks

# Can a router be given proof of who holds a prefix?




Press → for the ROA. A hypothetical one: YouTube had none in 2008.

**RPKI: registries certify who holds each block; the holder names its origin AS in a ROA.**

# A router checks each route’s origin against the ROAs

**208.65.152.0/22 | AS36561** → **valid** — **accept**

a ROA covers it; origin and length match

**208.65.153.0/24 | AS17557** → **invalid** — **drop**

a ROA covers it; wrong origin, too long

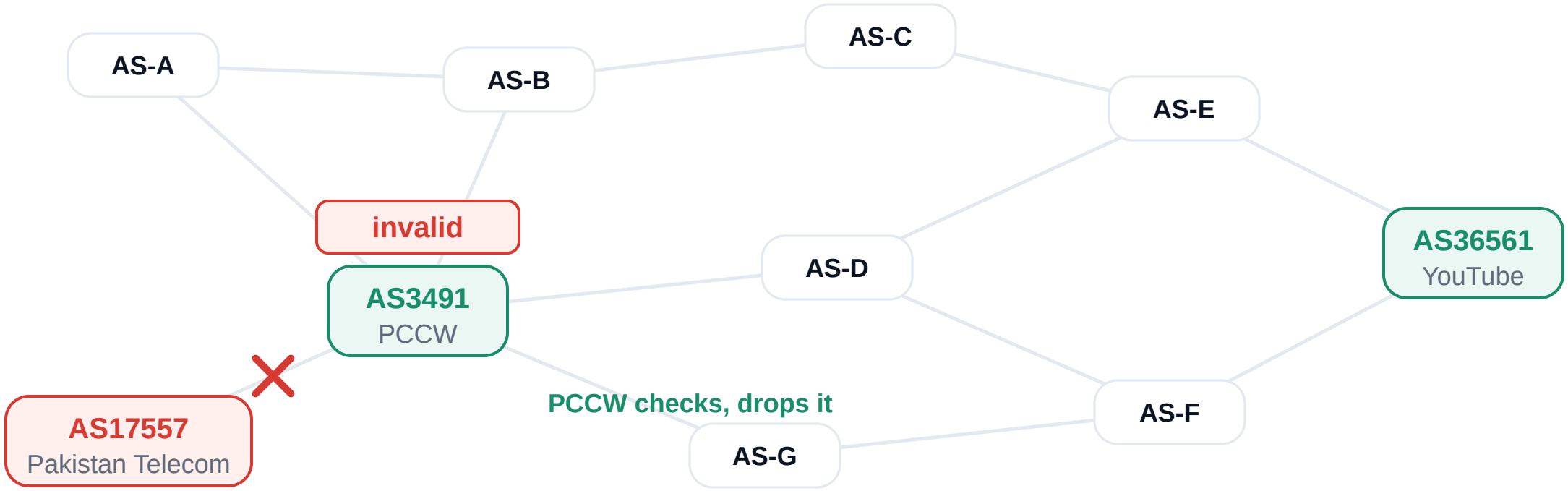
**128.205.0.0/16 | AS3685** → **not found** — **accept**

no ROA covers it (UB, 24 Sep 2026)

Press → for each route.

**Valid, invalid or not found. Routers that enforce it drop the invalid ones.**

# 2008 replayed, with a ROA




Press → to send the hijack again.

**The /24 is invalid. Every router that enforces RPKI drops it; YouTube stays reachable.**

1 Reach | 2 AS-PATH | 3 Session | 4 iBGP | 5 NEXT-HOP | 6 Choosing | 7 Who pays | 8 Hijacks

# What RPKI does not check

a forged advertisement, sent by AS17557







Press → for the real topology. Checking the path (ASPA, BGPsec) is still being deployed.

**Only the origin. A forger who appends the real origin AS still passes.**

1 Reach | 2 AS-PATH | 3 Session | 4 iBGP | 5 NEXT-HOP | 6 Choosing | 7 Who pays | 8 Hijacks

# How much of the internet is protected today?

**about 2/3**

**of routed IPv4 prefixes have a ROA**

Hurricane Electric RPKI report, July 2026

**and only**

**routers that enforce it**

**drop the invalid ones**



**read today's number live before class**

rpki-monitor.antd.nist.gov and radar.cloudflare.com/routing show it live; isbgpsafeyet.com lists who enforces.

**Protection needs both: a ROA from the holder, and routers that drop invalid routes.**

1 Reach | 2 AS-PATH | 3 Session | 4 iBGP | 5 NEXT-HOP | 6 Choosing | 7 Who pays | 8 Hijacks

# Eight topics, one line each

**01**

### Reachability

an AS says which prefixes it reaches, not its inside

**02**

### AS-PATH

each AS adds its number; your own number: drop it

**03**

### The session

TCP; the table once, then only the changes

**04**

### iBGP

spreads a route inside; full mesh or reflectors

**05**

### NEXT-HOP

a border address; OSPF turns it into a port

**06**

### Choosing

LOCAL-PREF, length, nearest exit, router ID

**07**

### Who pays whom

customers hear all; peers and providers much less

**08**

### Hijacks

longest match lets a /24 win; RPKI checks origin

**Each point from its own problem: why it is needed, how it works, what it costs.**

# Two goals, one table

**inside an AS**

**the shortest path**

by the AS's own costs

**OSPF**

**one forwarding table**

every row explained

**between ASes**

**a path someone is paid to carry**

**BGP**

**the data plane reads it, twenty billion times a second**

**Inside an AS: the shortest path. Between ASes: a path someone is paid to carry.**