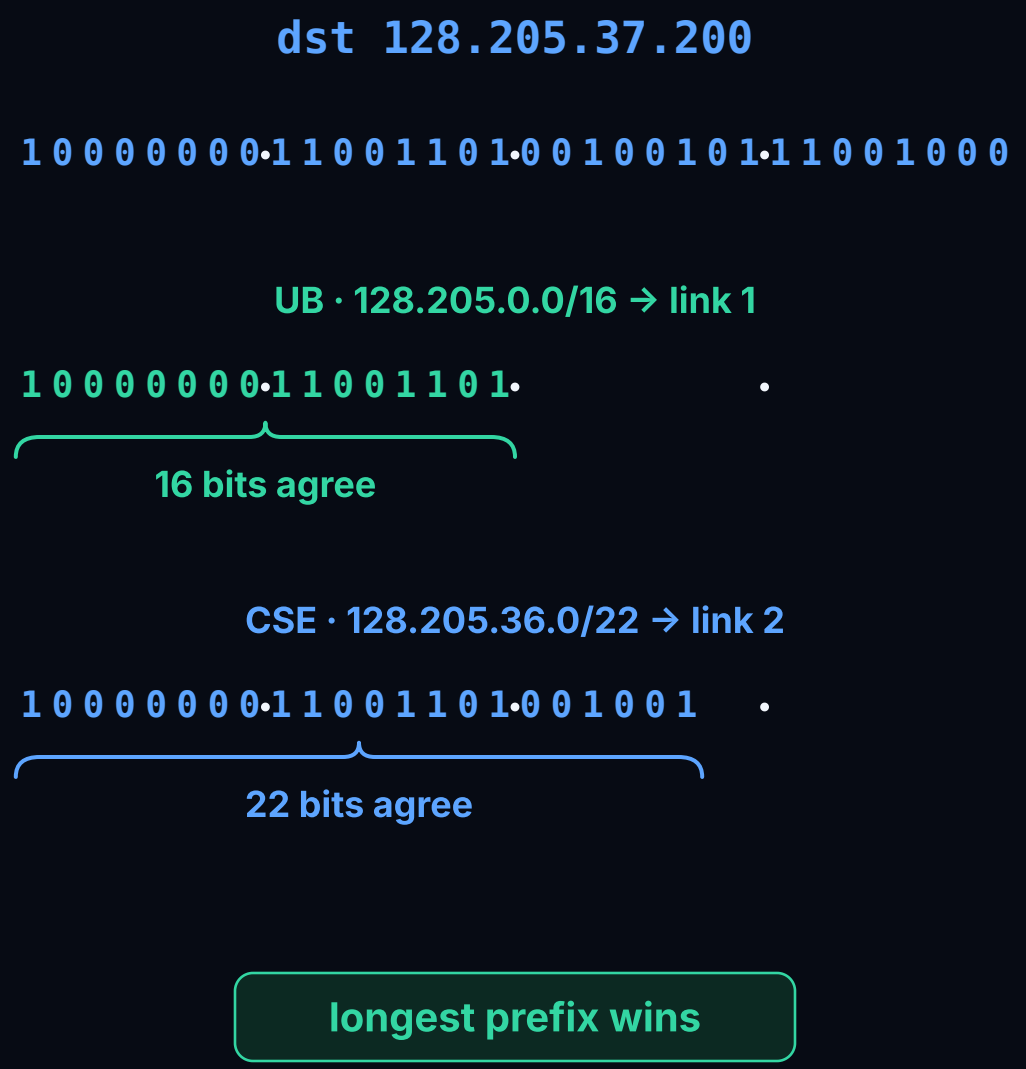
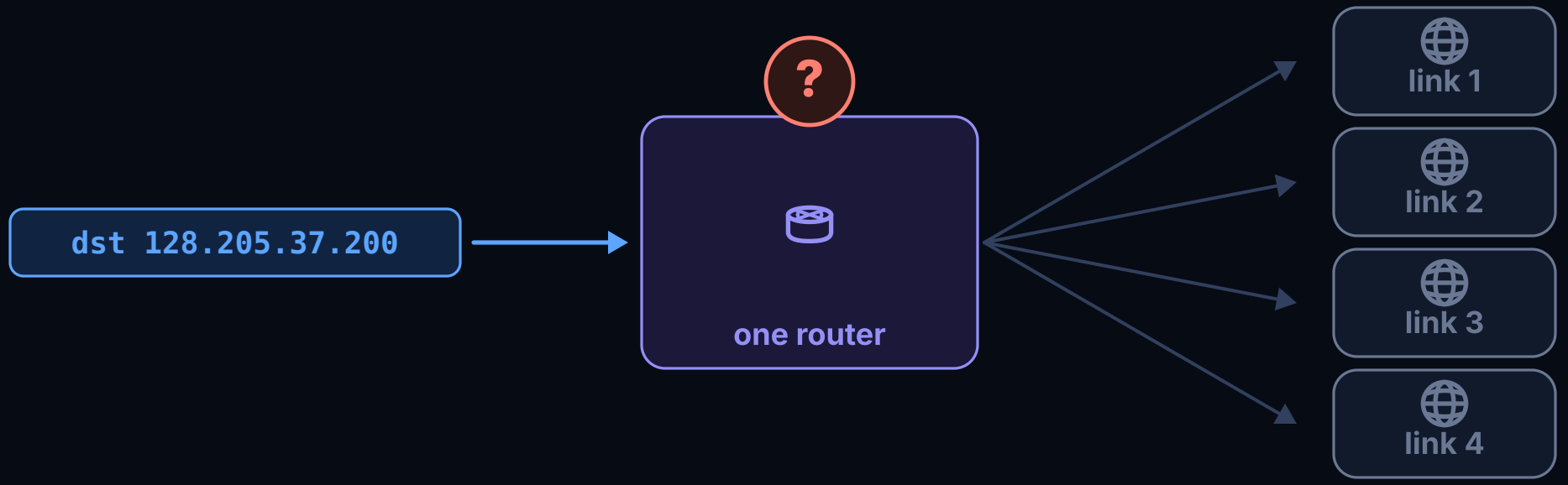
Forwarding
One packet, one table, one output link

A router reads one field of a packet and picks one output link. This lecture is that decision: how one row of a table stands for a whole block of addresses, and which row wins when several of them match.

Yaxiong Xie



One packet arrives. Four links leave. Which one?



Forwarding: pick one output link, for every packet.

What does the router know when it decides?



One field: the destination address. Nothing about the path.

So it needs a table



Forwarding = one lookup in one table, for every packet.

How many packets per second does a router forward today?

20,000,000,000

packets per second, through one router

Cisco 8223 · 64 × 800 GbE · 51.2 Tb/s · October 2025

One 800 GbE port alone can carry 1.19 billion 64-byte packets per second.

Twenty billion packets per second: twenty billion lookups per second.

The simplest table: one row per address

destination address	output link
128.205.37.199	1
128.205.37.200	1
128.205.37.201	1
128.205.37.202	3
…	…

4,294,967,296

rows

2^{32} addresses, 4.3 billion rows, in every router.

Can a table that size keep up?

EVERY SECOND

20×10^9

lookups

one lookup per packet

×

EVERY LOOKUP

4.3×10^9

rows to check

one row per address

=

8.6×10^{19} row checks, every second

20 billion lookups × 4.3 billion rows = 8.6×10^{19} row checks every second.

Is 8.6×10^{19} a number any machine can reach?

8.6×10^{19}

row checks per second, in one router

THE FASTEST SUPERCOMPUTER ON EARTH

about 2×10^{18} operations per second

THIS ROUTER, CHECKING EVERY ROW

8.6×10^{19} per second = 43 of them

El Capitan, the #1 system on the TOP500 list, does about 2×10^{18} operations per second.

No. That is 43 of the fastest computers on Earth, just to pick output links.

Two ways out



Fewer rows

less to store,
less to search per packet

this lecture



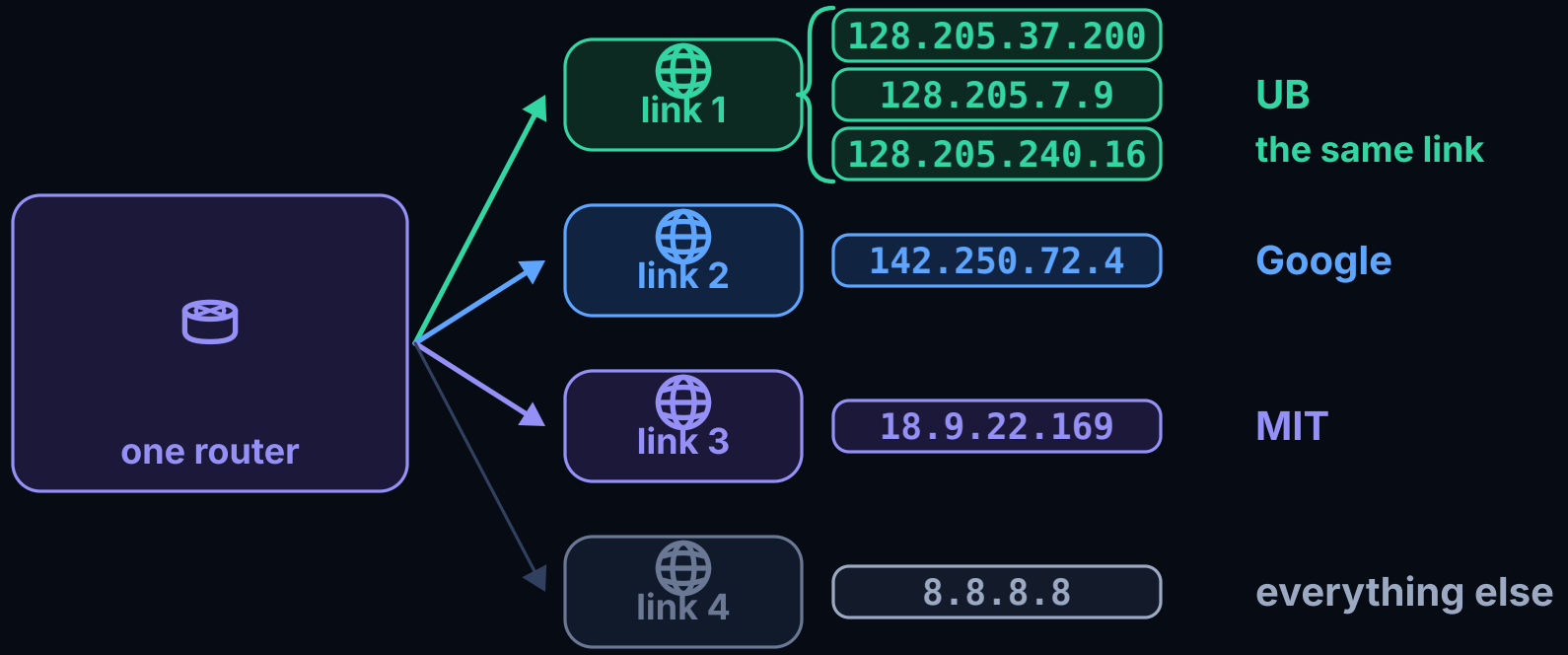
A faster search

search structures
and hardware

the router lecture

This lecture: make the table small.

Must every address have its own row?

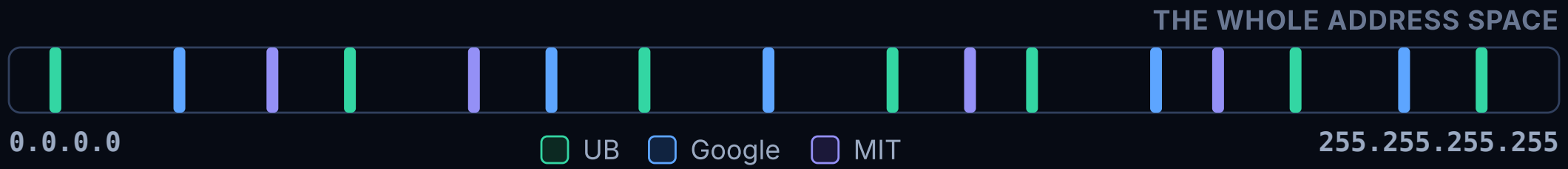


Three addresses, one answer.

Must each address have its own row, or could one row stand for all of UB?

Three addresses, one answer: could one row stand for all of UB?

What if UB's addresses were scattered?

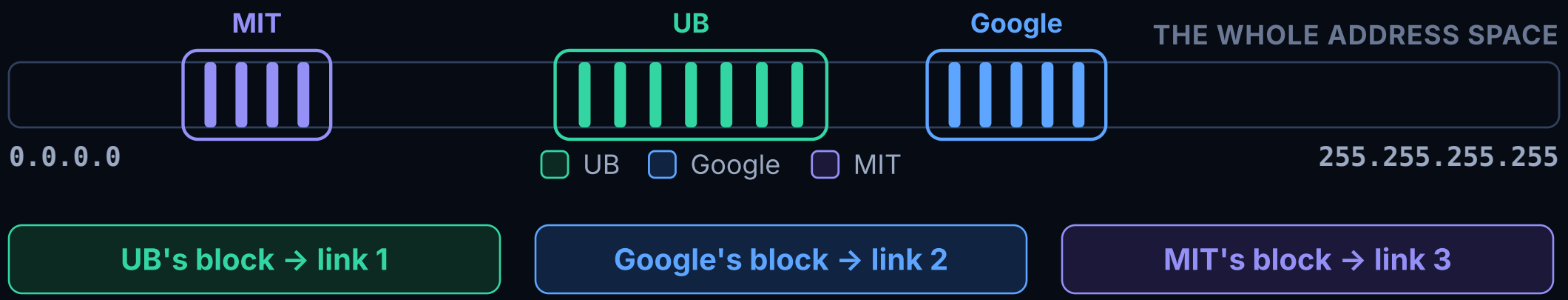


one row for UB would have to list every address

`128.205.37.200, 47.11.2.3, 200.4.9.1, 91.3.77.5, 12.0.6.8, …`

Then UB's set has no name but the list of its members. Nothing merges.

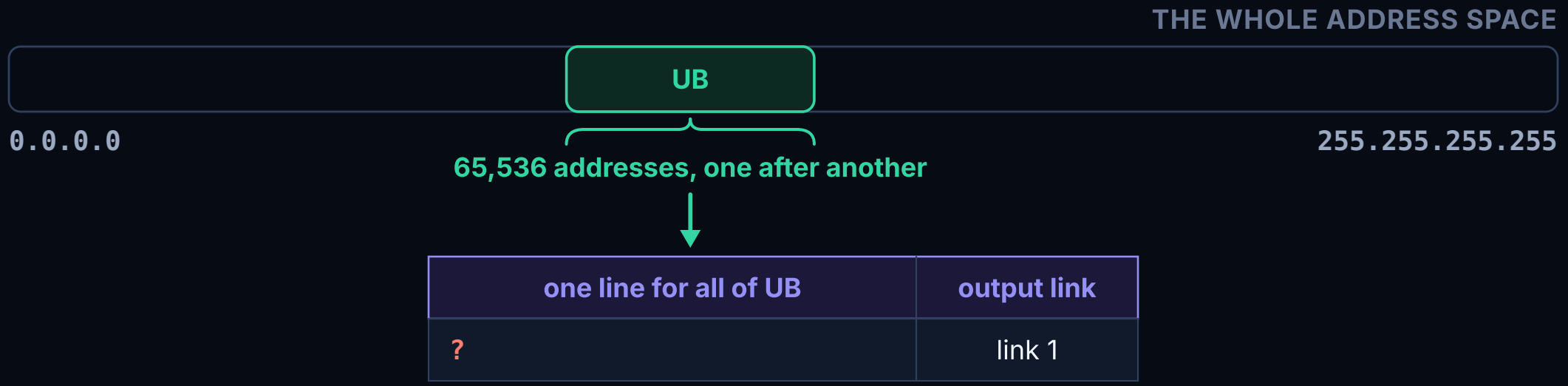
And if each organization's addresses sat together?



one block, one row

One block, one row. Is there a way to write a whole run of addresses as one thing?

Can one row stand for a whole range of addresses?



What goes in that cell?

One line that names all 65,536 addresses, and that a router can test any packet against.

Then: when two blocks nest, which row wins — and why anyone would want rows that overlap.

The first question: write a whole block of addresses in one row.

How do you write a set of addresses in one row?

01

A row is a block

One line has to name the whole set, and a router has to be able to test any address against it.

What must one row hold now?

BEFORE: ONE ROW, ONE ADDRESS

128.205.37.200	link 1

NOW: ONE ROW, A WHOLE SET

?	a set of addresses	link 1

 # short
one line, not a list of members

✓ testable
given 32 bits: in the set, or not?

A one-line name for a set, and a test that runs on the 32 bits of a packet.

An address is 32 bits

WHAT THE ROUTER SEES

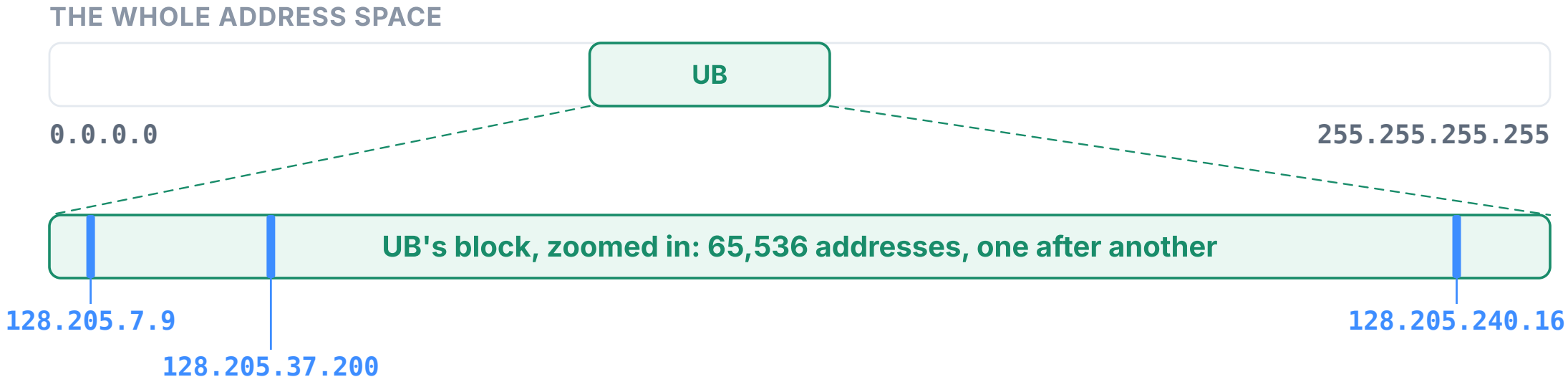
1	0	0	0	0	0	0	0	.	1	1	0	0	1	1	0	1	.	0	0	1	0	0	1	0	1	.	1	1	0	0	1	0	0	0
128									205									37									200								

WHAT WE WRITE

```
128 . 205 . 37 . 200
```

Four groups of eight bits, each written in decimal.

Which addresses does UB own?



128.205.0.0
the first address

128.205.255.255
the last address

From 128.205.0.0 to 128.205.255.255, every address belongs to UB.
One contiguous block: nothing in between belongs to anyone else, and nothing of UB's lies outside it.

One contiguous block: every address from 128.205.0.0 to 128.205.255.255 is UB's.

What do all of UB's addresses have in common?



Their first 16 bits. The set already carries a name: the bits its members share.

Write the block as a prefix



128.205.0.0/16

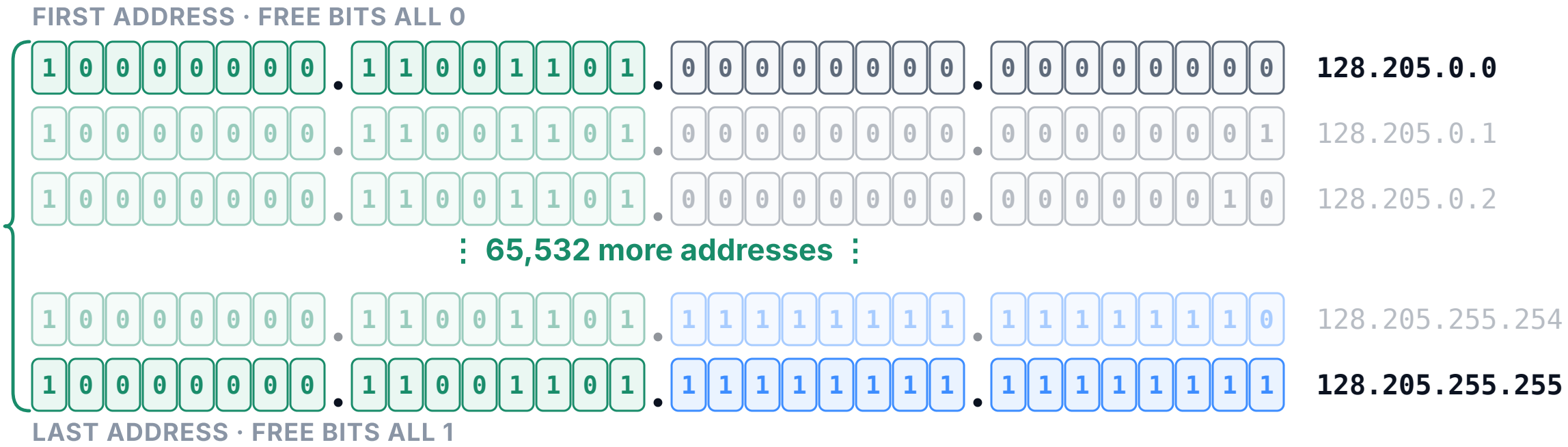
base address
free bits written as 0

prefix length
how many bits are fixed

CIDR notation: **a.b.c.d/x**.

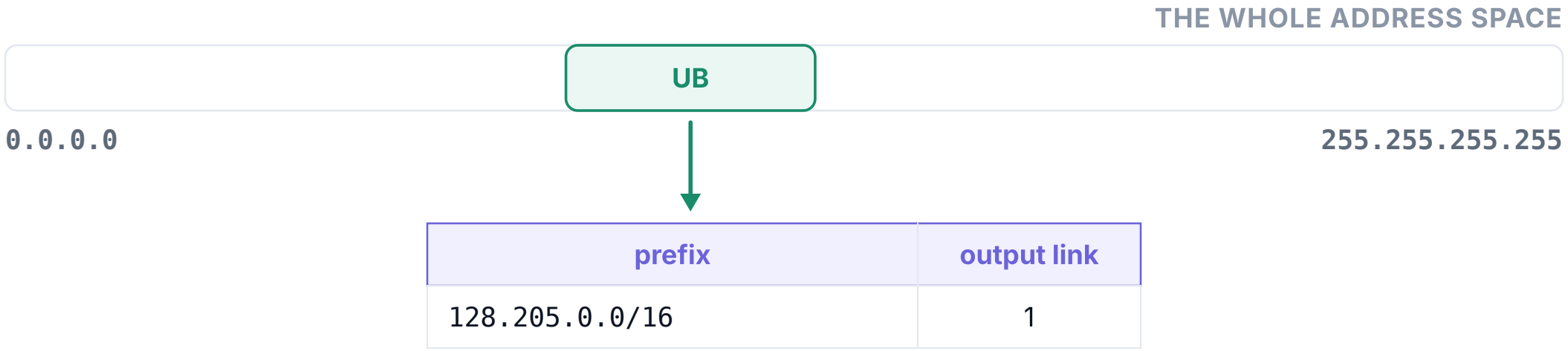
Base address + length: the first x bits fixed, the rest free.

How many addresses share those 16 bits?



From all zeros to all ones, first and last included: $2^{16} = 65{,}536$ addresses.

One row, one name, 65,536 addresses



prefix	output link
128.205.0.0/16	1

One row is a block. The first question is answered.

128.205.0.0/16 names all 65,536 addresses in one line: the bits they share, and how many.

One row is a block: the shared bits name the set, x says how many. Answered.

What exactly does a.b.c.d/x say?

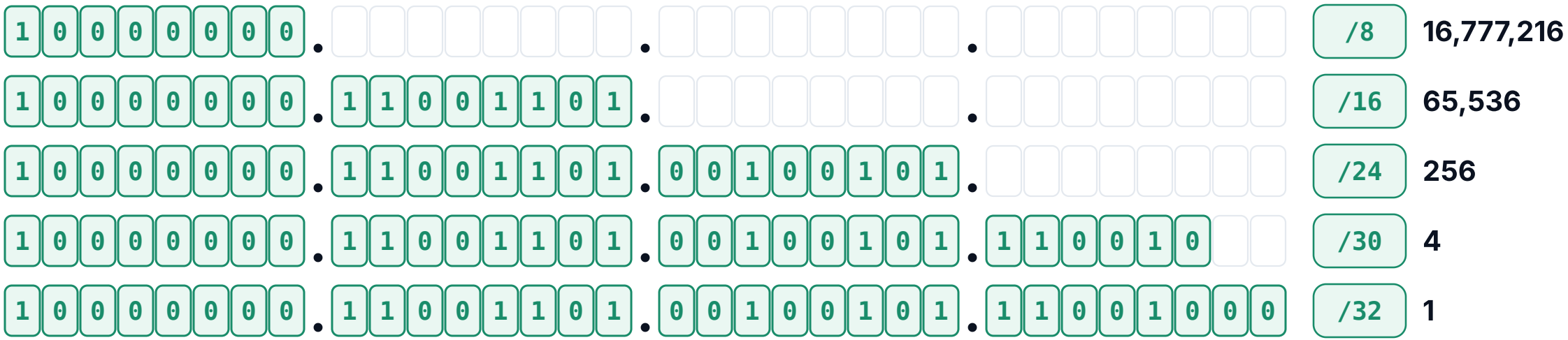
02

The prefix

A base address and a length. How many addresses a prefix covers, two special lengths, and where a block starts and ends.

How many addresses does a prefix cover?

a /x prefix covers 2^{32-x} addresses



Longer prefix: more bits fixed, fewer free, a smaller block. Each bit halves it.

Two special lengths

/32 · EVERY BIT FIXED



exactly one address

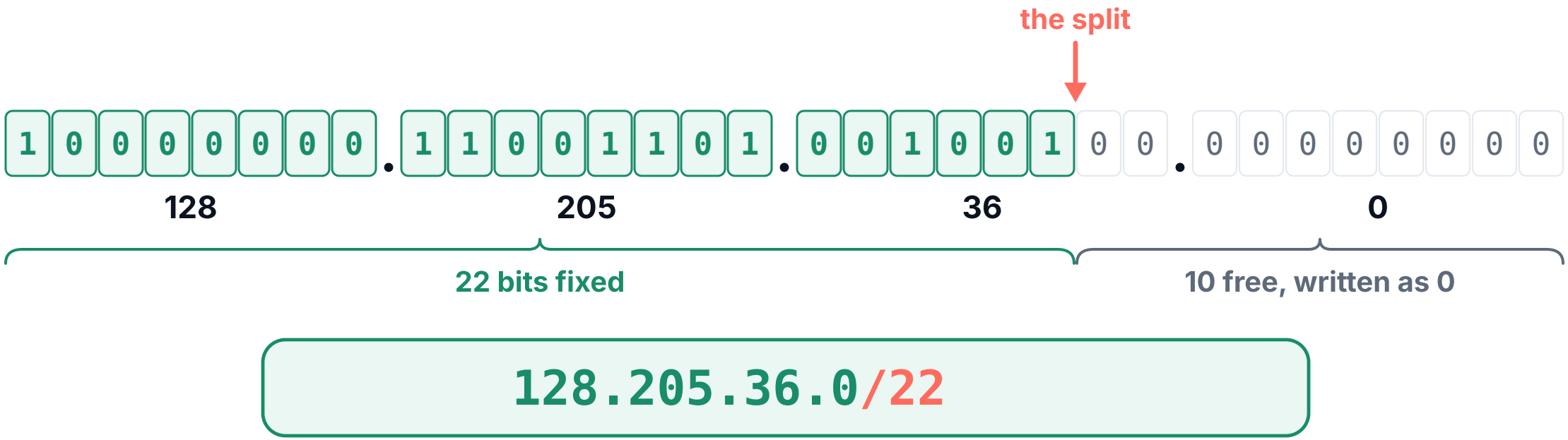
/0 · NO BIT FIXED



every address there is: 0.0.0.0/0

0.0.0.0/0 matches every address. It will matter in Part 4.

The split can be anywhere: /32 to /0



Slide the split left one bit: x drops by one, and the bit it passes is written as 0.

Where does 200.23.16.0/20 start and end?

FIRST ADDRESS · FREE BITS ALL 0



LAST ADDRESS · FREE BITS ALL 1

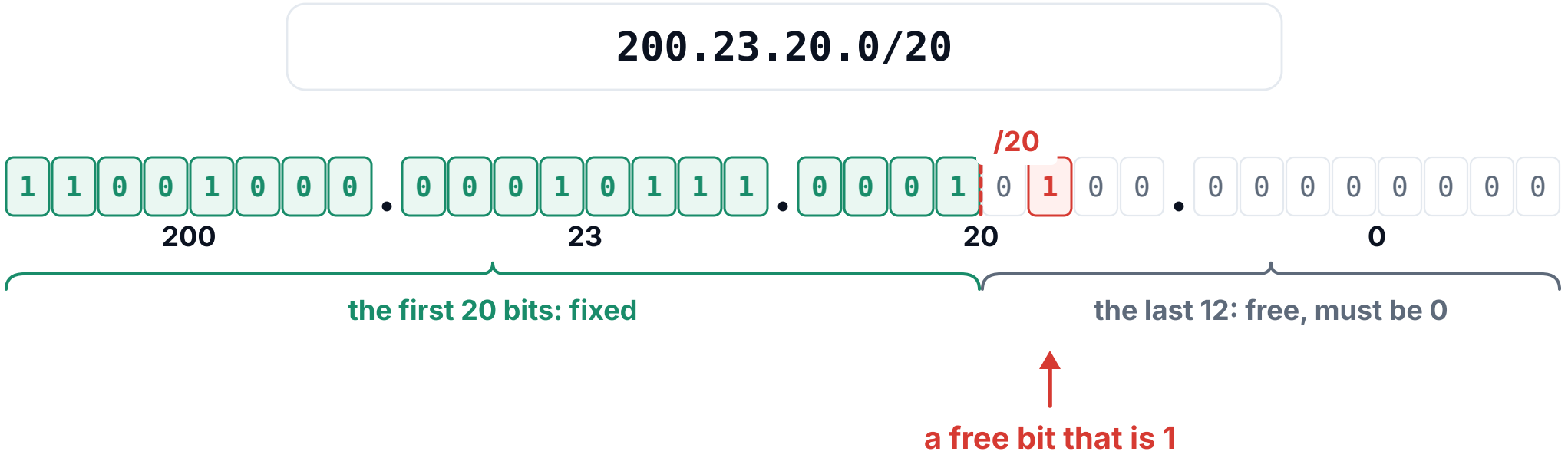


200.23.16.0 – 200.23.31.255

4,096 addresses

Work the split octet in binary; the other octets copy across.

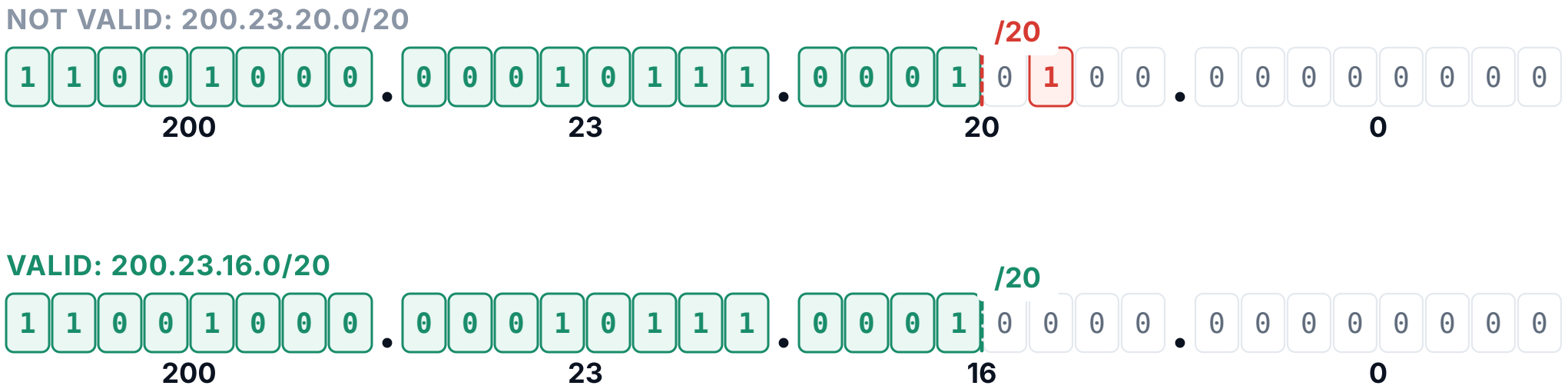
Is 200.23.20.0/20 a valid prefix?



✕ Not a valid prefix: a free bit is 1. The first 20 bits of 200.23.20.0 do not describe this address.

Write the bits, draw the /20 line: a bit right of the line is 1. Not valid.

The valid prefix: 200.23.16.0/20



The /20 that contains 200.23.20.0 is 200.23.16.0/20.

The same 20 fixed bits; the free bits written as 0. A prefix is written with its first address.

Same 20 fixed bits, every free bit 0: 200.23.16.0/20. That is the block.

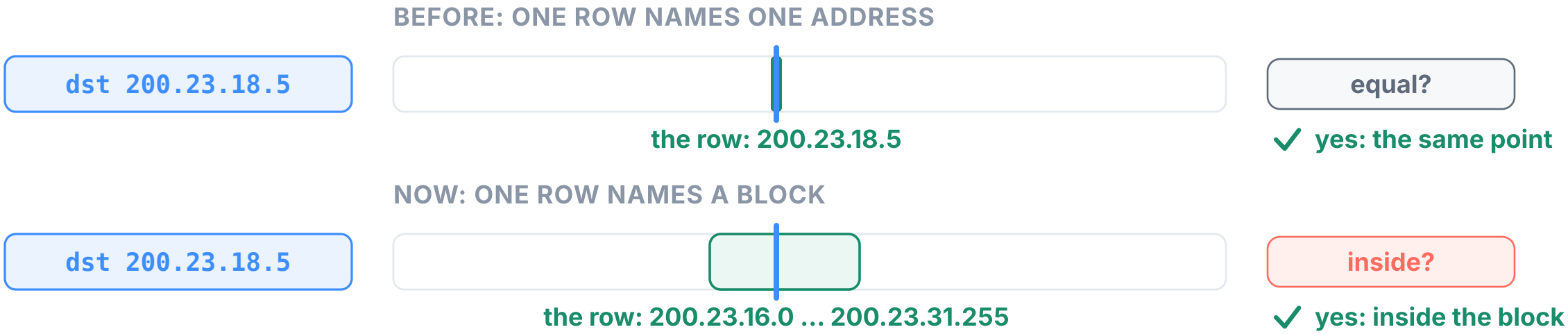
How does a router test an address against a prefix?

03

Prefix matching

Before, two addresses had to be equal, all 32 bits. Now one side is a block: is the address inside it? And the router must ask that the same way for every row.

The problem has changed: not "equal" but "inside"

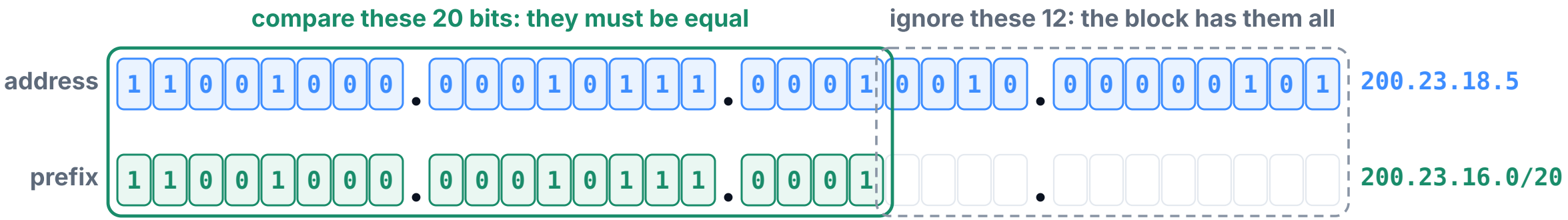


Before: are the 32 bits of the address equal to the 32 bits in the row?
Now: is the address one of the 4,096 addresses the row names?

Not "equal to the row's address?" but "inside the block the row names?"

1 A row is a block | 2 The prefix | 3 Prefix matching | 4 Longest match | 5 Overlap on purpose

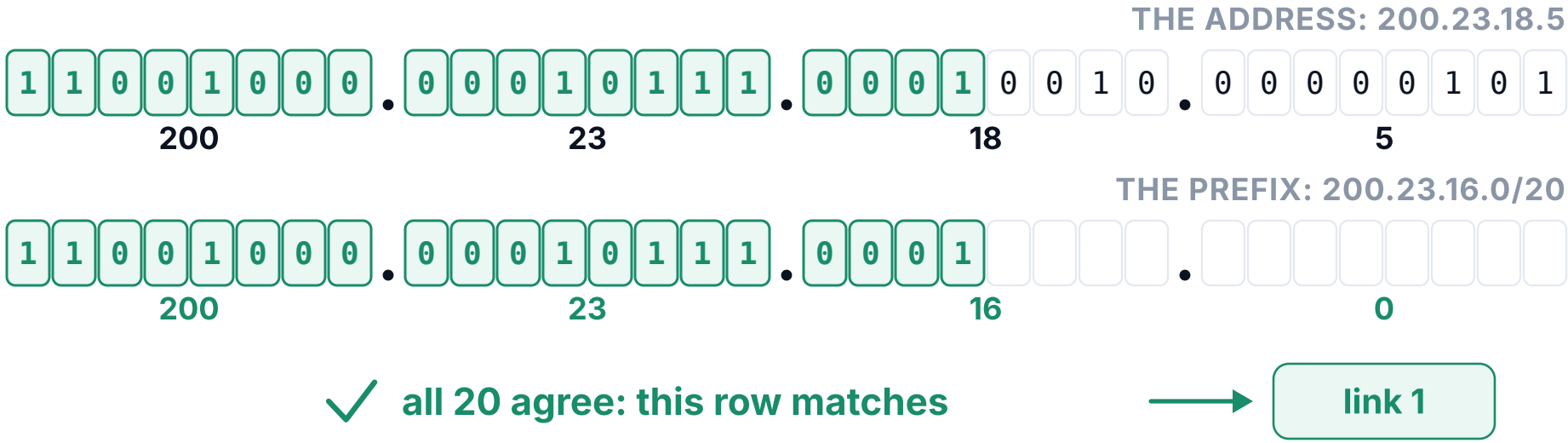
Prefix matching: only the first x bits must agree



Inside the block ⇔ the first 20 bits of the address equal the 20 bits of the prefix.
The other 12 bits can be anything: the block contains every value they can take.

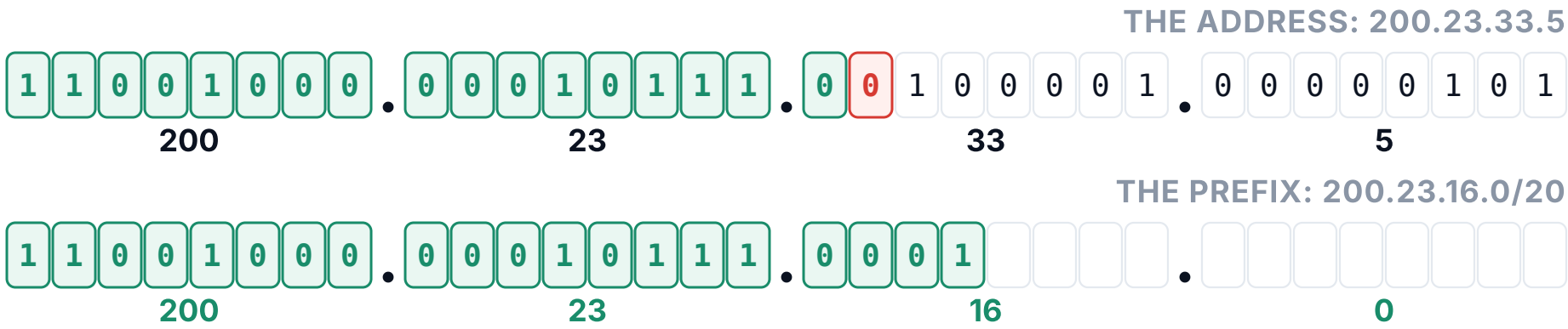
Inside the block = the first x bits equal the prefix. The other 32 − x bits are free.

Is 200.23.18.5 inside the range of 200.23.16.0/20?



From the left, one bit at a time. All 20 agree: match. The packet can go to link 1.

Is 200.23.33.5 inside the range of 200.23.16.0/20?

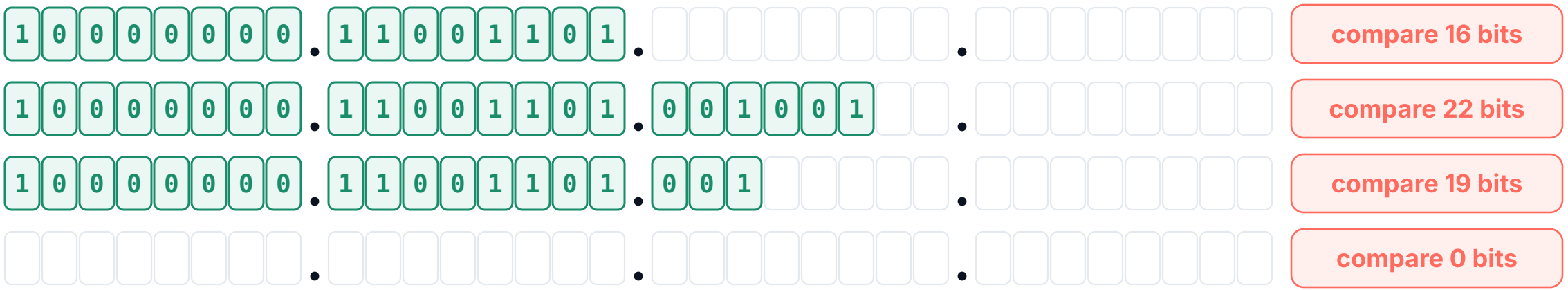


bit 18 differs: this row does not match

it cannot forward this packet

Bit 18 differs. No match: this row says nothing about where this packet goes.

A router is hardware: can every row ask the same task?

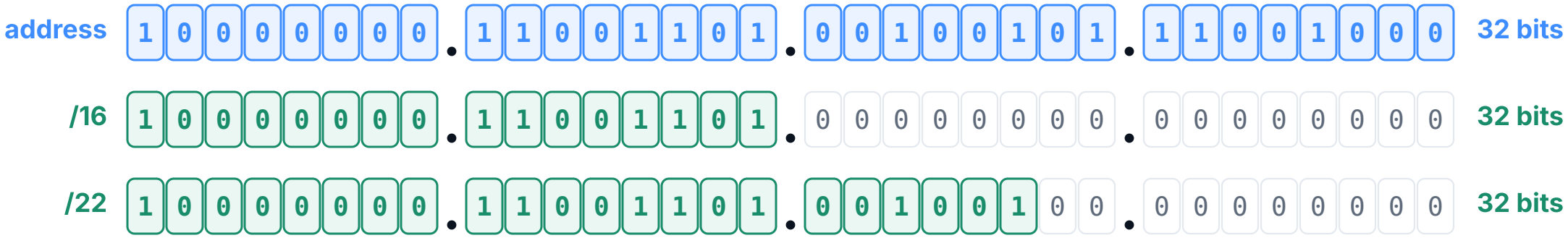


Four different comparisons. A circuit that does one fixed task is far simpler.

The idea: one fixed job, 32 bits against 32 bits

Are two bits the same? XOR them. **0 = the same, 1 = different**

`0 XOR 0 = 0` `0 XOR 1 = 1` `1 XOR 0 = 1` `1 XOR 1 = 0`

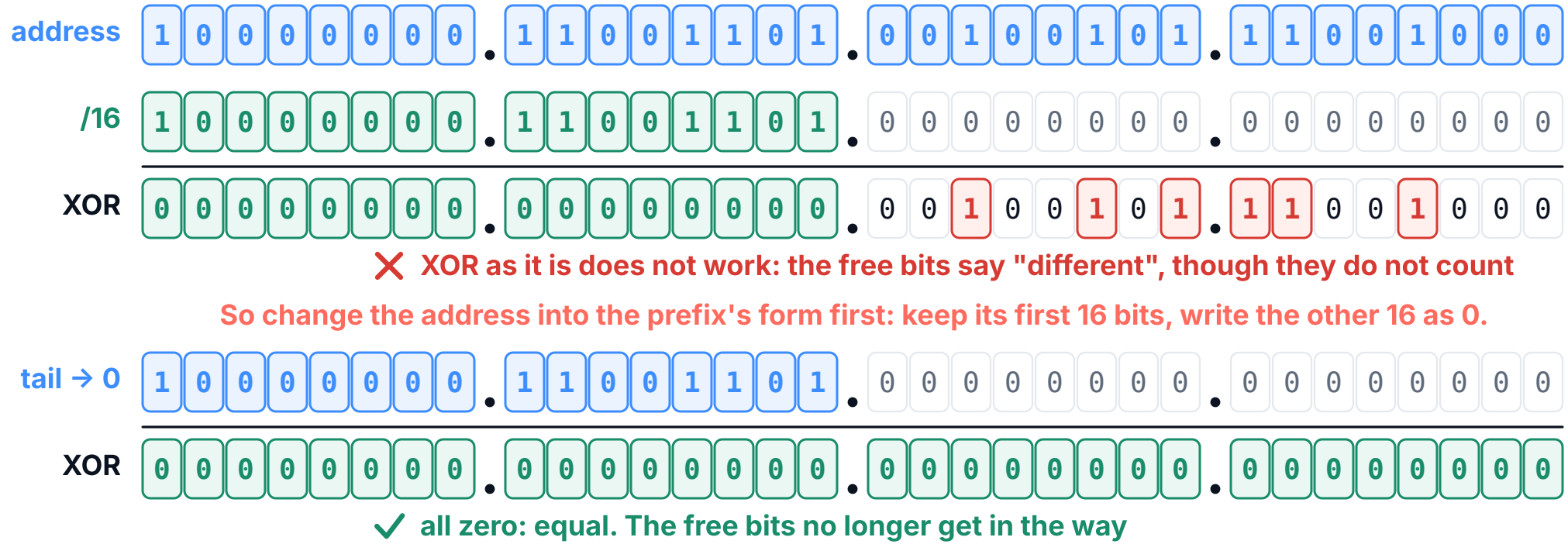


✓ **Long or short, both sides are 32 bits: one fixed job for every row.**

XOR the 32 bits of the address with the 32 bits of the prefix, then check for all zeros.

An address is 32 bits; a prefix of any length is 32 bits too. Compare 32 with 32.

One thing in the way: the address's free bits



Writing the free bits as 0 in one bit operation: AND with a mask (next page).

Zero the free bits first, so both sides have the prefix's form. A mask does that.

The mask: the length x, written as 32 bits

THE PREFIX: 200.23.16.0/20



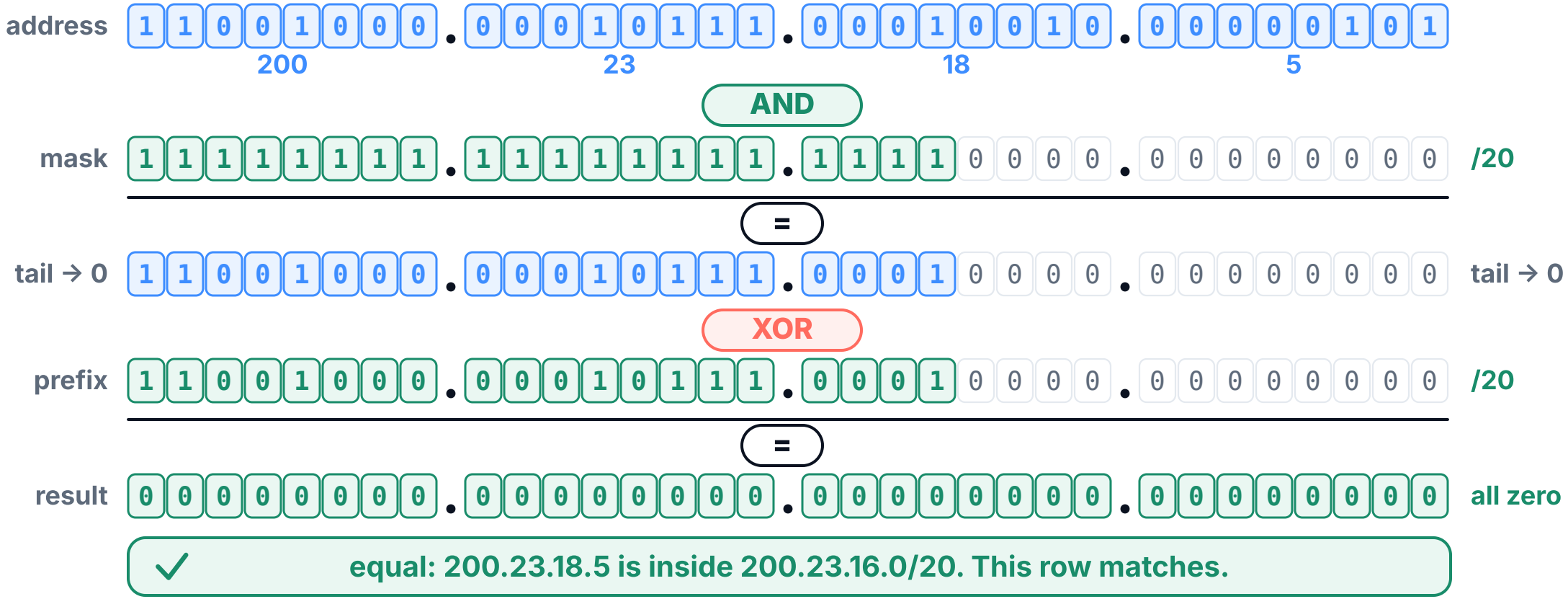
ITS MASK: /20 WRITTEN AS 32 BITS



✓ address AND mask keeps the first 20 bits of the address and writes the other 12 as 0.

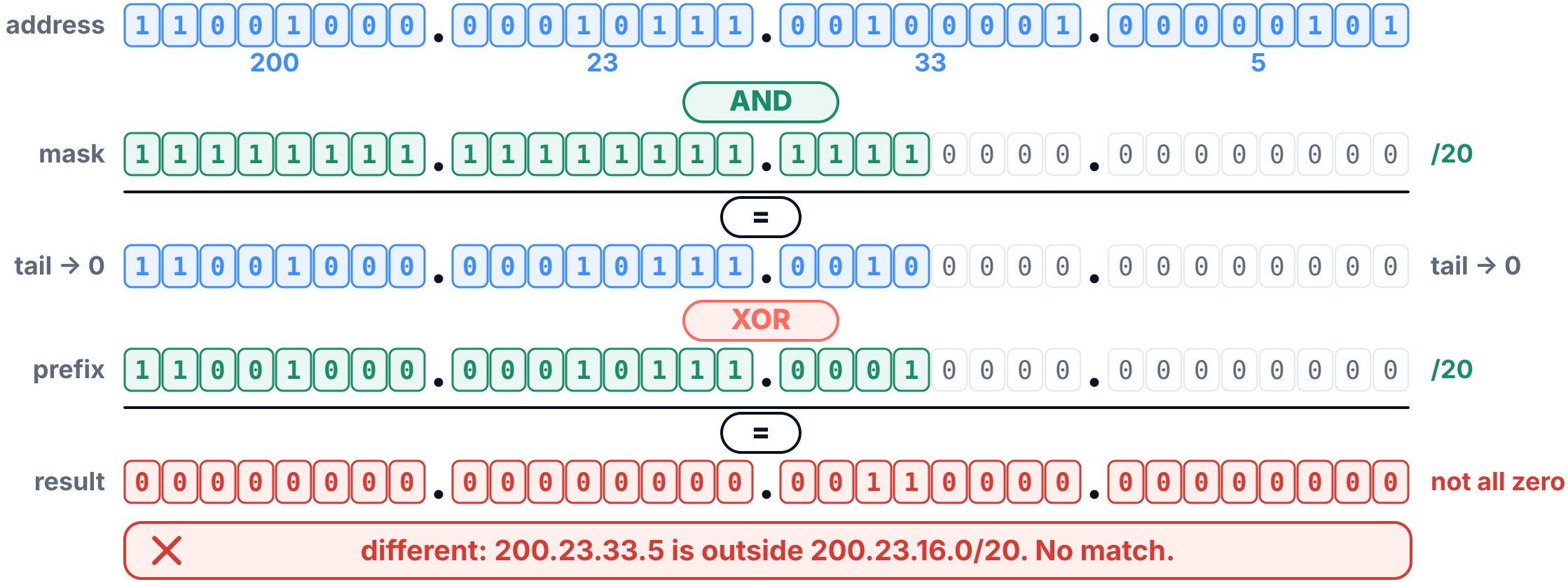
1 where the bit is kept, 0 where it is cleared. Always 32 bits, whatever x.

AND the mask, then XOR the prefix: the whole test



AND keeps 20 bits and zeroes 12; XOR with the prefix gives all zeros: a match.

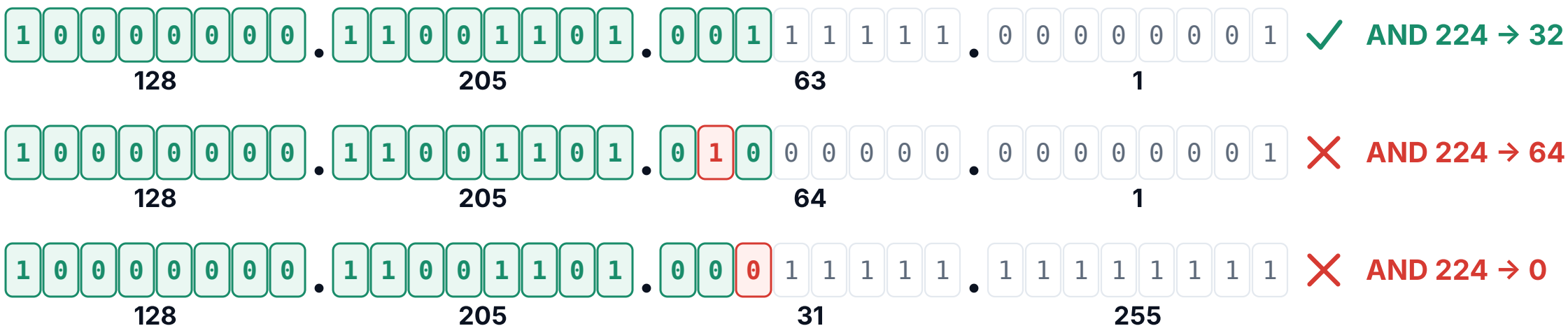
The same steps, another address



One kept bit differs: a 1 in the XOR result. Outside the block: no match.

Check: inside 128.205.32.0/19 or not?

Which of these are inside 128.205.32.0/19?



Mask the third octet with 224; the base is 32.

The same mask in dotted decimal



/20 = 255.255.240.0

Masks came first (RFC 950, 1985). /x is their shorthand (CIDR, 1993).

A prefix length and a mask say the same thing.

The masks you will meet

length	mask
/8	255.0.0.0
/16	255.255.0.0
/19	255.255.224.0
/20	255.255.240.0
/22	255.255.252.0
/24	255.255.255.0
/30	255.255.255.252

THE PARTIAL OCTET

bits	value
10000000	128
11000000	192
11100000	224
11110000	240
11111000	248
11111100	252
11111110	254
11111111	255

A partial octet is always 128, 192, 224, 240, 248, 252, 254 or 255.

With a mask, everything about a block is a bit operation

first address

```
address AND mask
```

last address

```
address OR (NOT mask)
```

size of the block

```
2 ^ (zeros in the mask)
```

longest prefix

```
the mask with the most ones
```

No arithmetic, no loops: AND, OR, NOT, and a count of ones.

How small does the table get?

ONE ROW PER ADDRESS

4,294,967,296

rows

÷ 4,000

THE REAL TABLE OF A CORE ROUTER TODAY

1,078,000

rows

every block announced on the whole Internet, one row each

IPv4 prefixes in the global BGP table, CIDR Report, September 2026. A campus router holds far fewer: most of the Internet is one default row for it.

A router that carries the whole Internet holds about a million rows, not four billion.

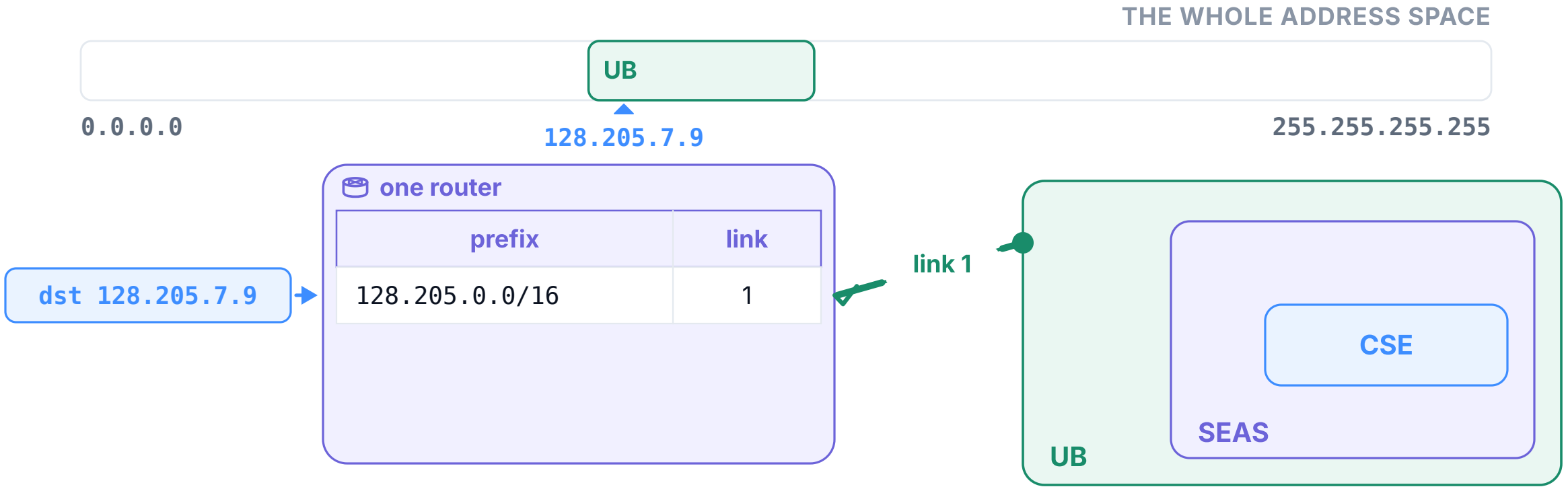
Two rows, one inside the other: which one wins?

04

Longest match

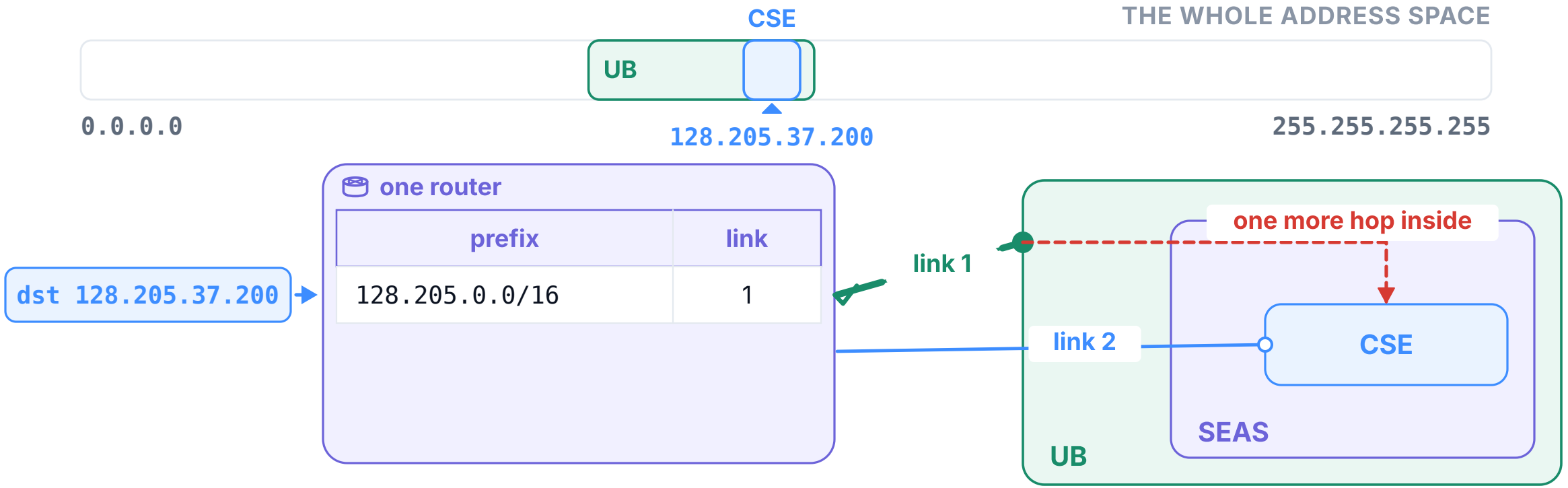
UB has one row. The CSE department wants its own. Both are allowed, and a packet to CSE matches both.

UB: one block, one row



Every address in 128.205.0.0/16 leaves this router by link 1, into the campus.

CSE has its own cable to the router

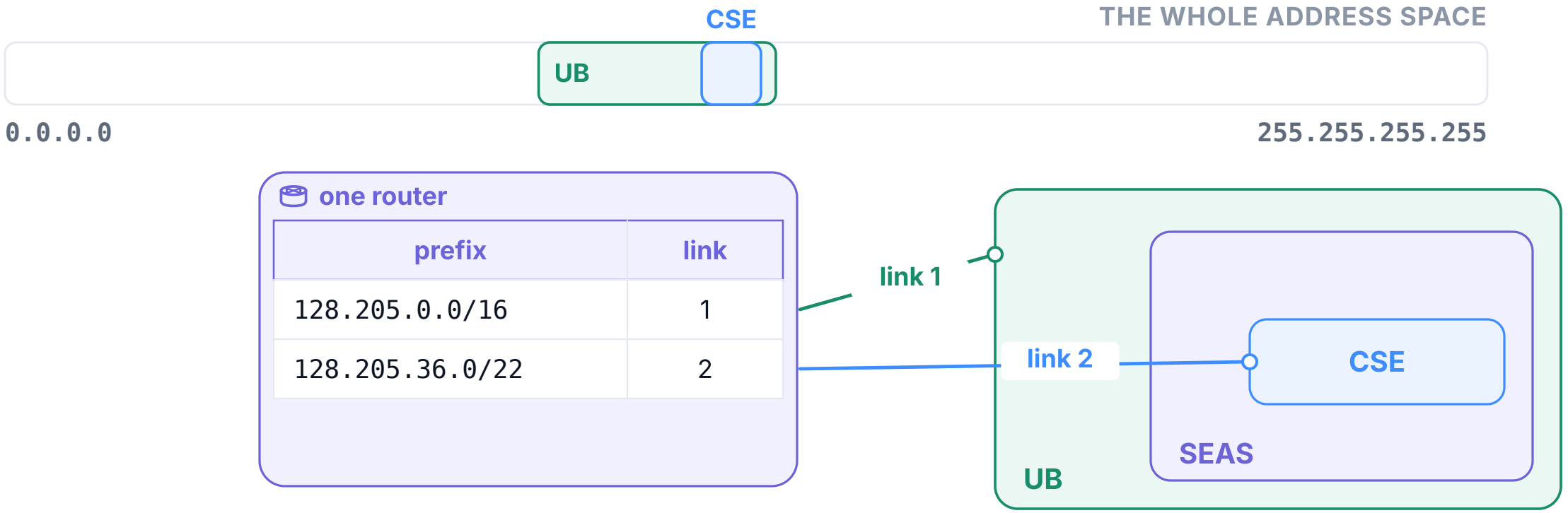


Link 2 leads straight to CSE and sits idle. Why not send the packet down link 2?

With one row, a packet to CSE goes into UB and is forwarded again. Link 2 is wasted.

1 A row is a block | 2 The prefix | 3 Prefix matching | 4 Longest match | 5 Overlap on purpose

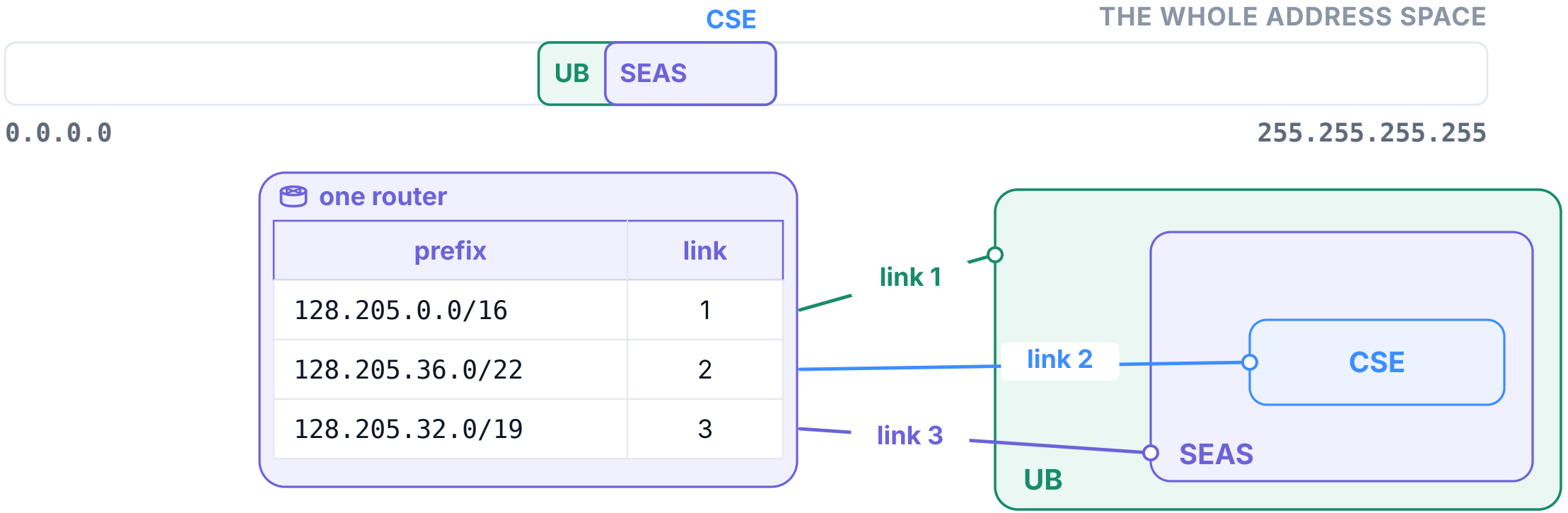
Add a row for CSE



✓ The intent: a packet for one of CSE's 1,024 addresses takes link 2, straight to CSE.

128.205.36.0/22 → link 2. The table now says where CSE's packets should go.

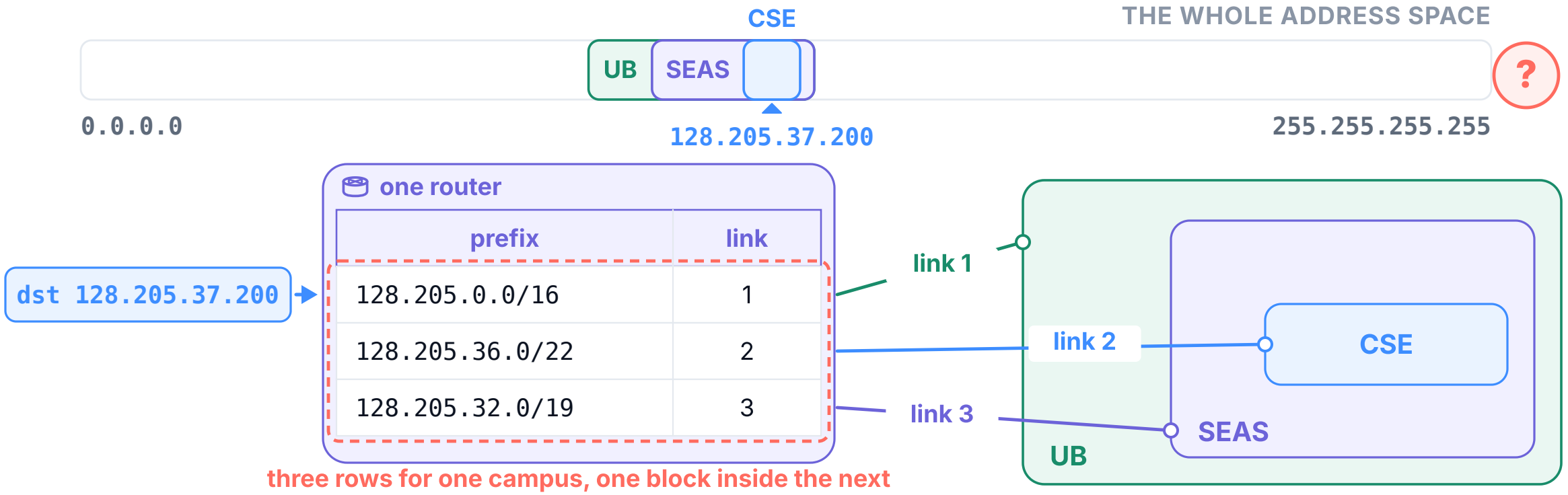
SEAS has its own cable too



✓ One more row: 128.205.32.0/19, SEAS's 8,192 addresses, takes link 3.

The same move again: one row for SEAS, 128.205.32.0/19 → link 3.

Three rows for one campus, and they overlap

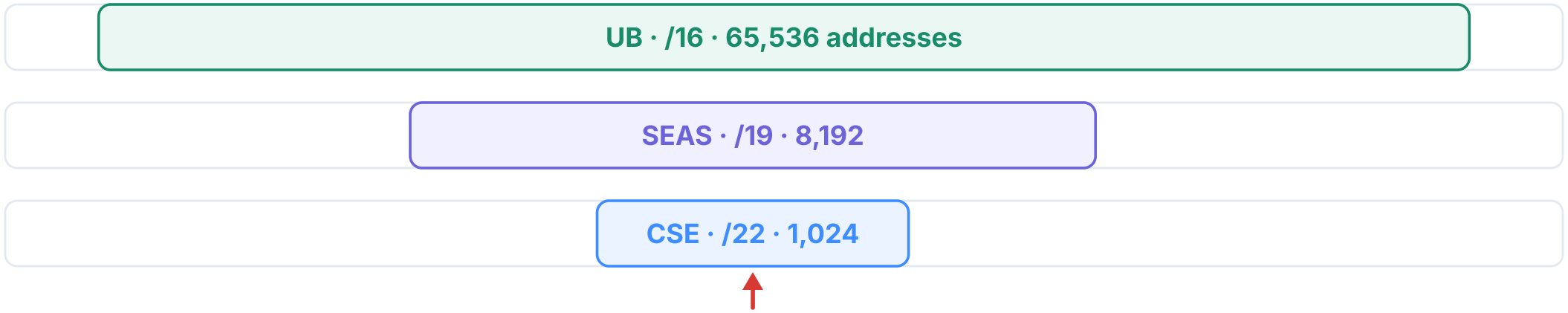


A packet to CSE arrives. Which row decides where it goes?

CSE's addresses are inside SEAS's, and SEAS's inside UB's. The rows overlap.

1 A row is a block | 2 The prefix | 3 Prefix matching | 4 Longest match | 5 Overlap on purpose

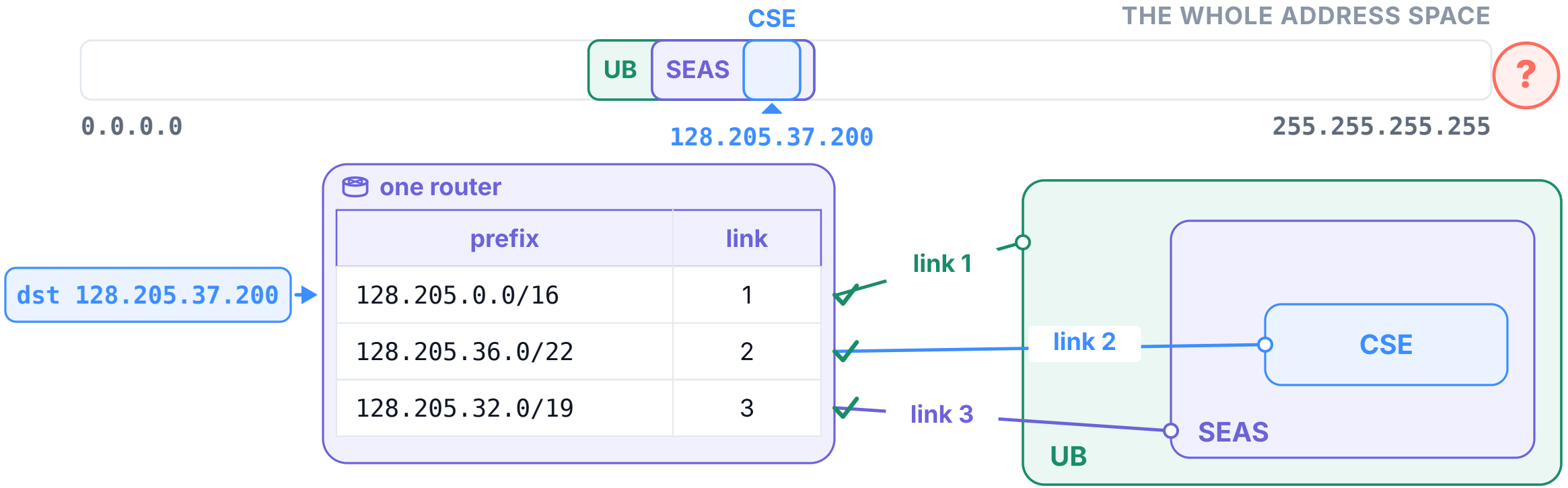
The bigger the range, the shorter the prefix



The bigger the range, the shorter the prefix: fewer bits are fixed.

UB /16, SEAS /19 inside it, CSE /22 inside that: the smaller the block, the longer the prefix.

A packet to CSE matches all three rows

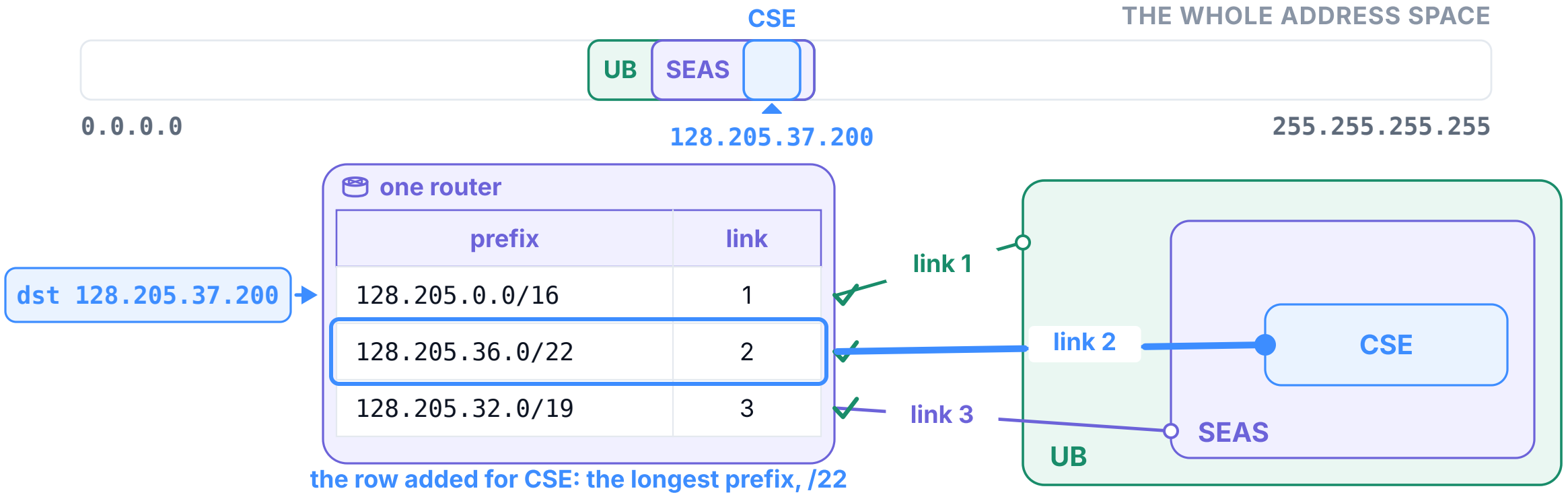


128.205.37.200 is a UB address, a SEAS address and a CSE address. All three rows match.

Inside CSE means inside SEAS and inside UB too. Three matches, one packet.

1 A row is a block | 2 The prefix | 3 Prefix matching | 4 Longest match | 5 Overlap on purpose

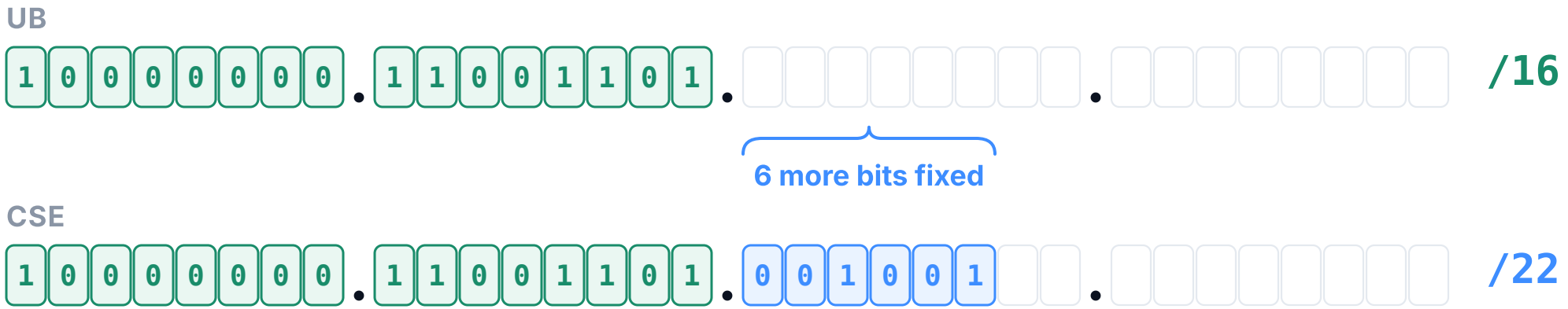
Which row should win? The one with the longest prefix



✓ Of the rows that match, take the one with the longest prefix. The packet goes straight down link 2.

Many rows match; the longest prefix wins. This is longest prefix matching.

What makes the CSE row more specific? A longer prefix



65,536 addresses → 1,024 addresses

More fixed bits: a smaller block, a narrower claim. Longer prefix = more specific.

The rule: longest prefix match

When more than one row matches the address,

use the row with the longest prefix.

128.205.0.0/16	128.205.32.0/19	128.205.36.0/22
UB: 16 fixed bits	SEAS: 19 fixed bits	**CSE: 22 fixed bits → wins**

This is why it is called longest prefix match: the row with the most fixed bits wins.

The same table, with a default row

prefix	output link
128.205.0.0/16	1
128.205.36.0/22	2
128.205.32.0/19	3
0.0.0.0/0	4

UB → 1, CSE → 2, SEAS → 3 as in the story; a fourth row for everything else → 4. The /19 and /22 are teaching subnets.

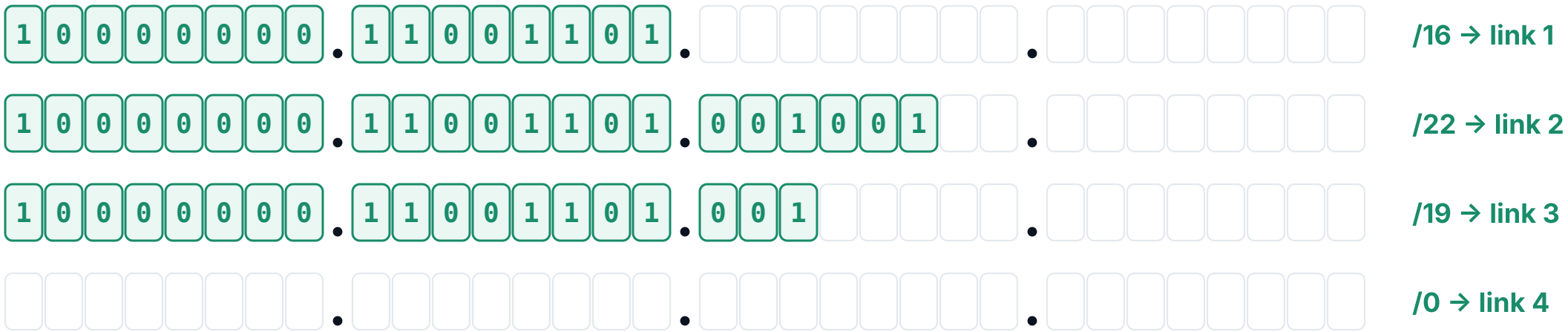
Four rows, three of them nested: CSE inside SEAS inside UB.

The same rows, with their masks

prefix	mask	output link
128.205.0.0/16	255.255.0.0	1
128.205.36.0/22	255.255.252.0	2
128.205.32.0/19	255.255.224.0	3
0.0.0.0/0	0.0.0.0	4

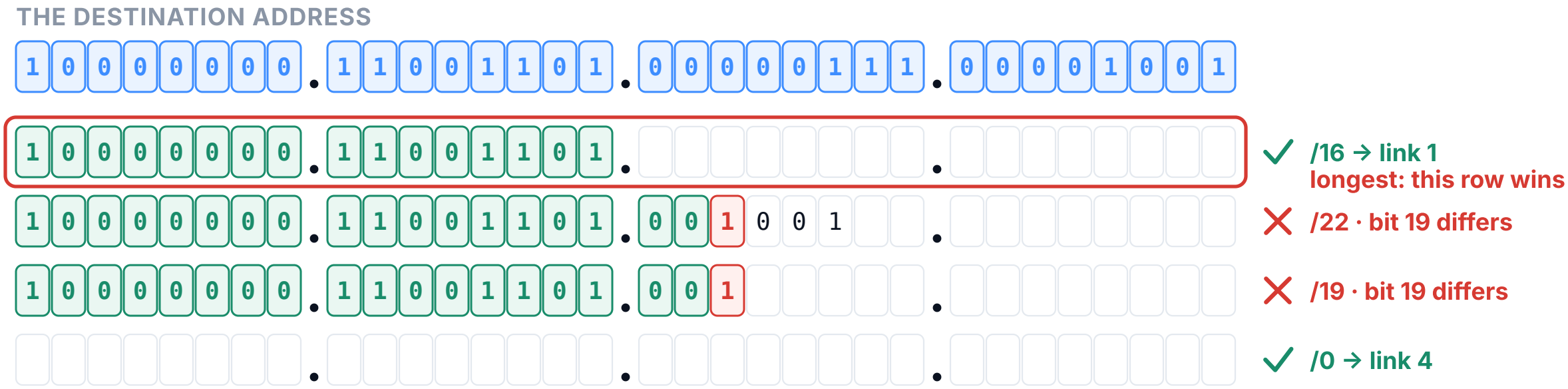
All three UB rows share the first 16 bits; the third octet separates them.

The same rows, as bits



Four runs of fixed bits: 16, 22, 19 and 0. Everything else is free.

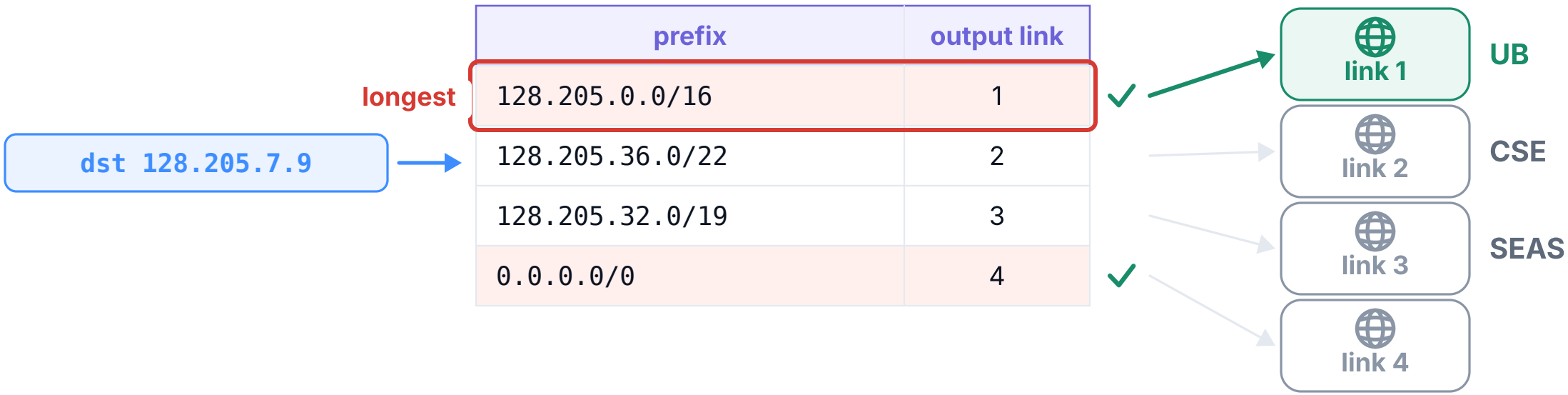
Address 128.205.7.9 against the four rows



Every row is checked. Every match gets a tick. The longest match wins.

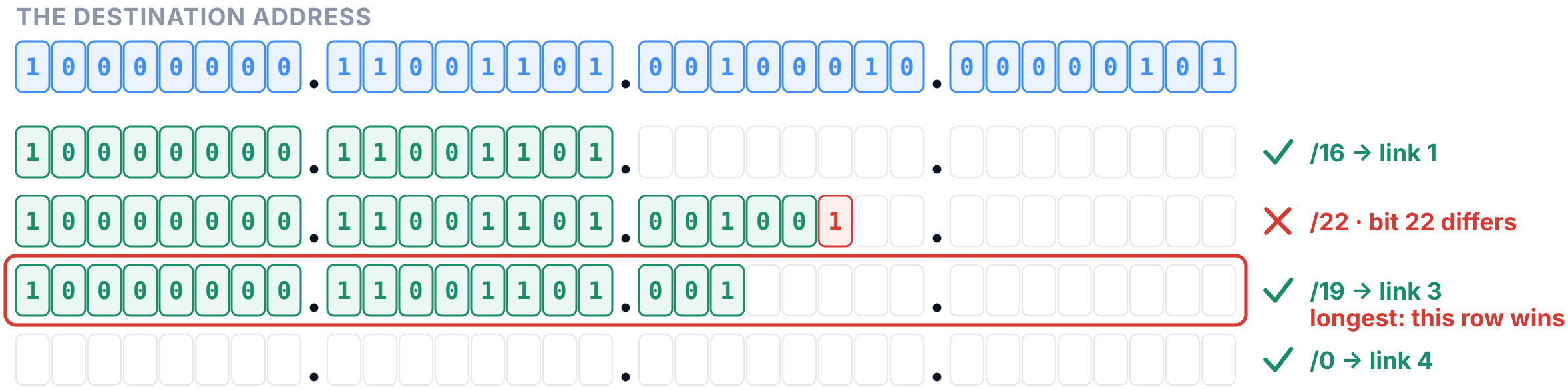
1 A row is a block | 2 The prefix | 3 Prefix matching | 4 Longest match | 5 Overlap on purpose

Address 128.205.7.9: the answer



Only /0 and /16 match. /16 is longer. Link 1.

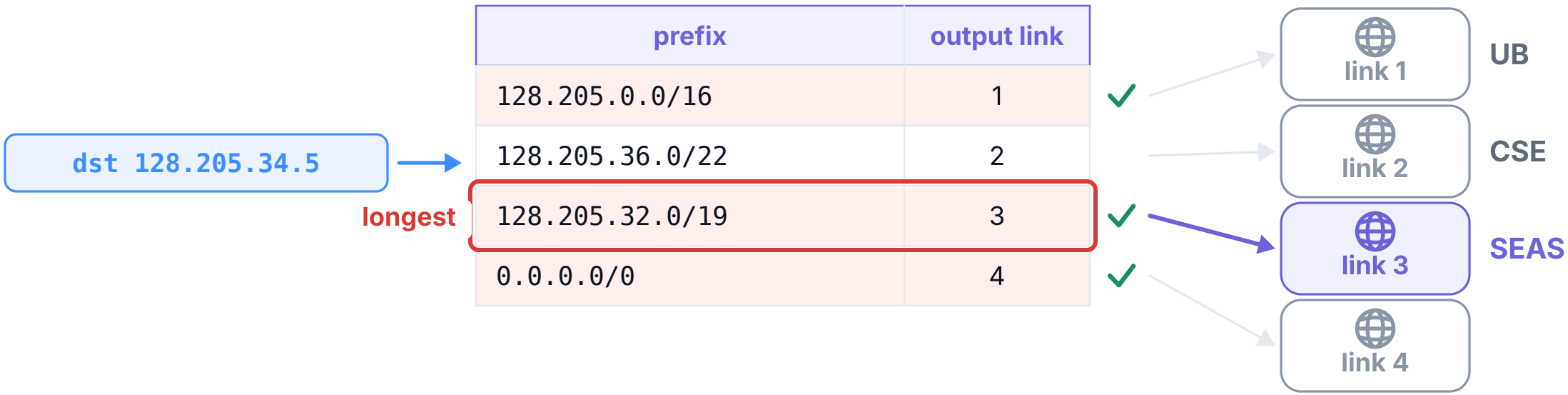
Address 128.205.34.5 against the four rows



Every row is checked. Every match gets a tick. The longest match wins.

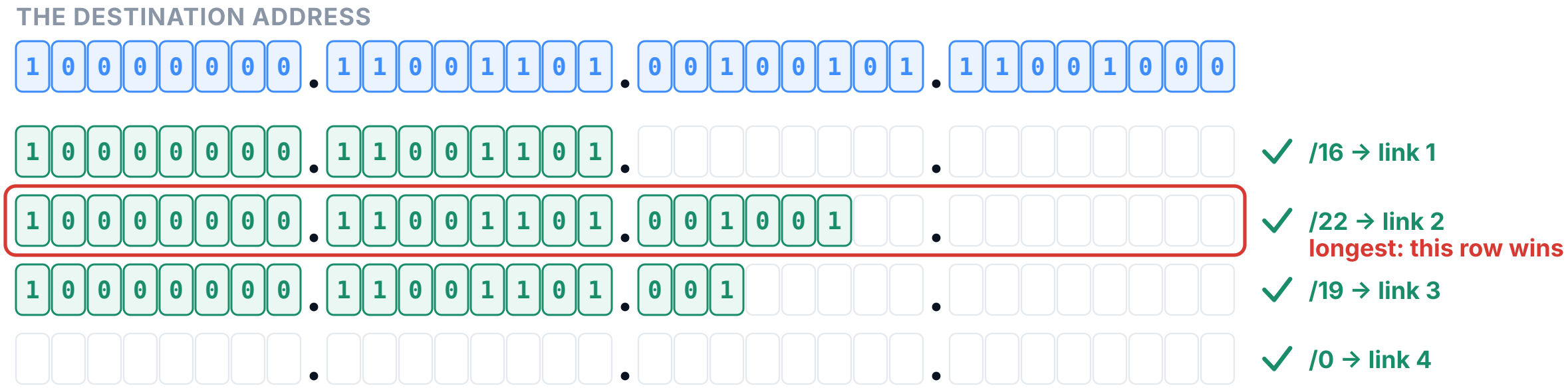
1 A row is a block | 2 The prefix | 3 Prefix matching | 4 Longest match | 5 Overlap on purpose

Address 128.205.34.5: the answer



/16 and /19 match. /19 is longer. Link 3.

Address 128.205.37.200 against the four rows



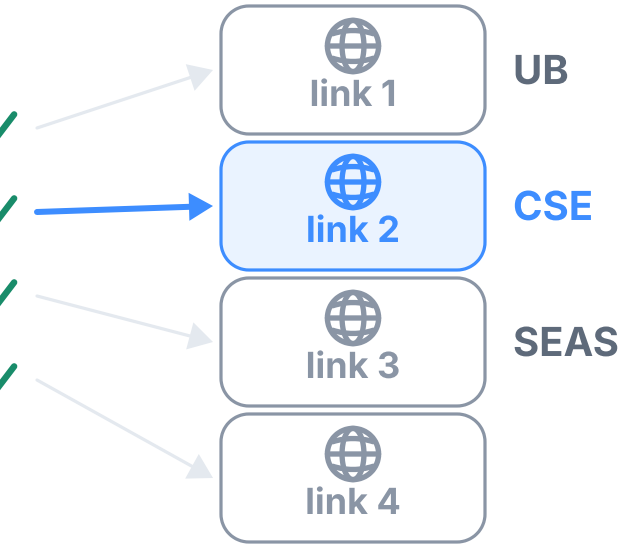
Every row is checked. Every match gets a tick. The longest match wins.

1 A row is a block | 2 The prefix | 3 Prefix matching | 4 Longest match | 5 Overlap on purpose

Address 128.205.37.200: the answer

`dst 128.205.37.200` longest

prefix	output link
128.205.0.0/16	1
128.205.36.0/22	2
128.205.32.0/19	3
0.0.0.0/0	4

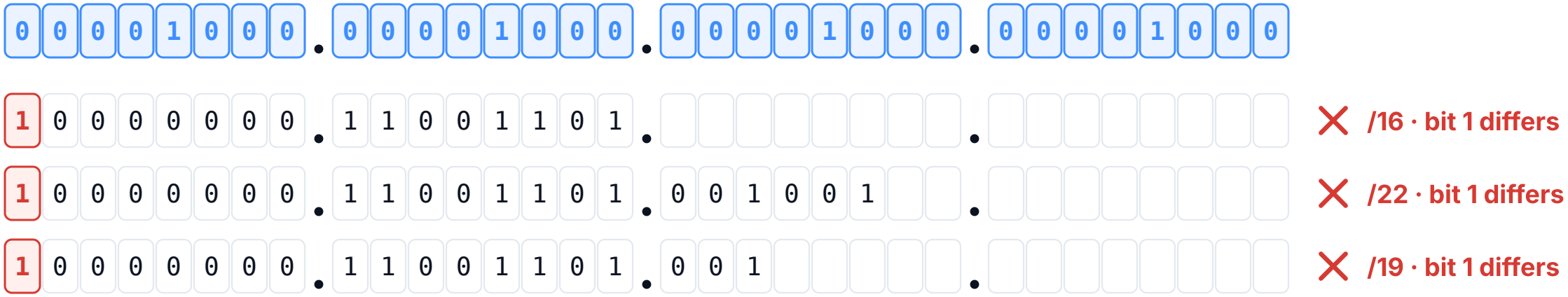


All four match. /22 is the longest. Link 2.

1 A row is a block | 2 The prefix | 3 Prefix matching | 4 Longest match | 5 Overlap on purpose

Address 8.8.8.8: nothing in UB matches

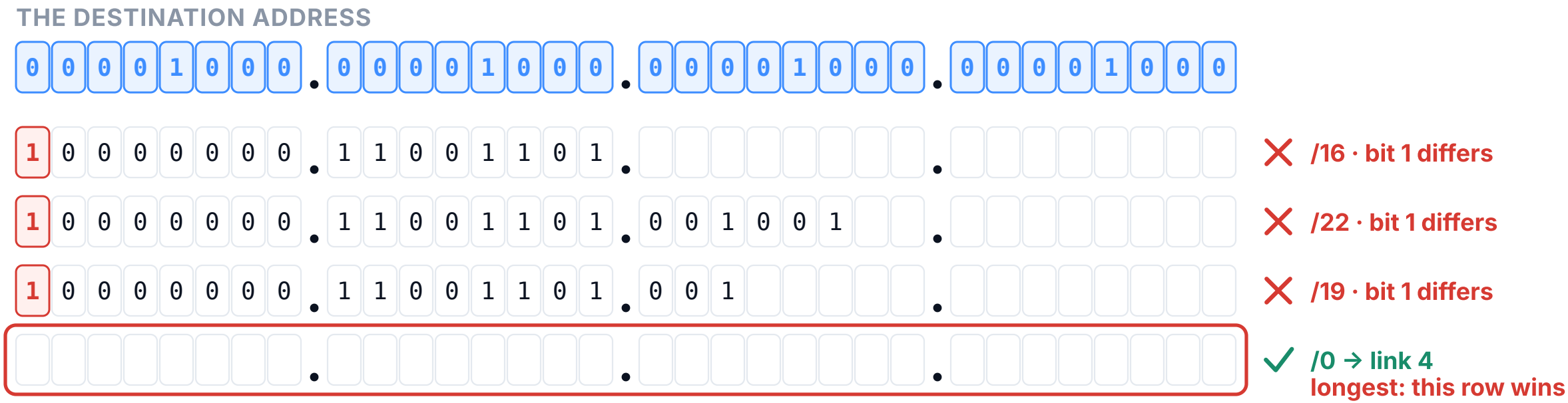
THE DESTINATION ADDRESS



Every UB row fails at the very first bit. Without another row, the packet is dropped.

1 A row is a block | 2 The prefix | 3 Prefix matching | 4 Longest match | 5 Overlap on purpose

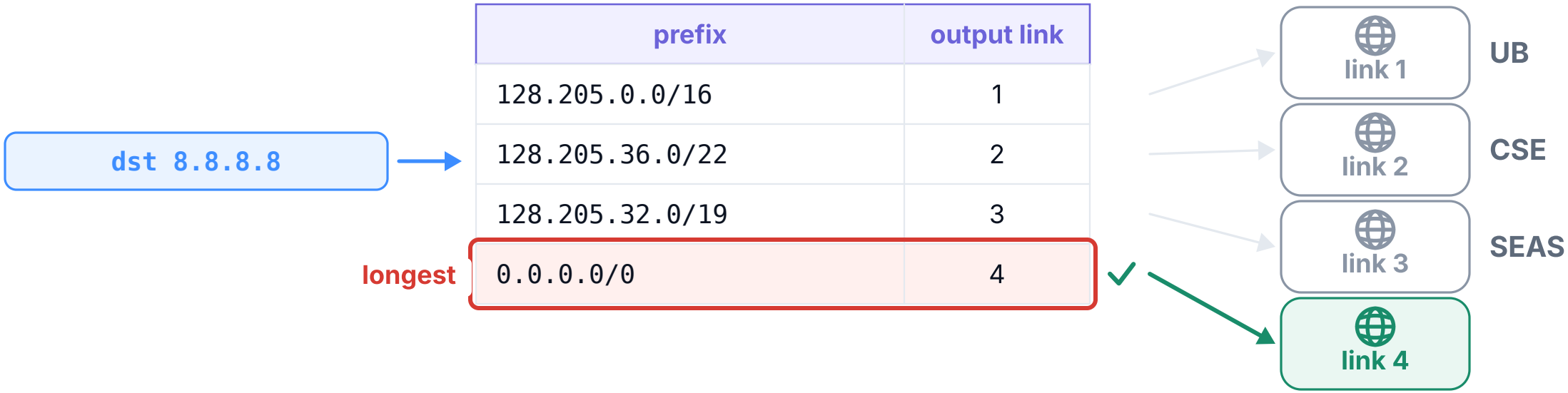
The default route: 0.0.0.0/0



Zero bits fixed: it matches every address, and loses to any other match.

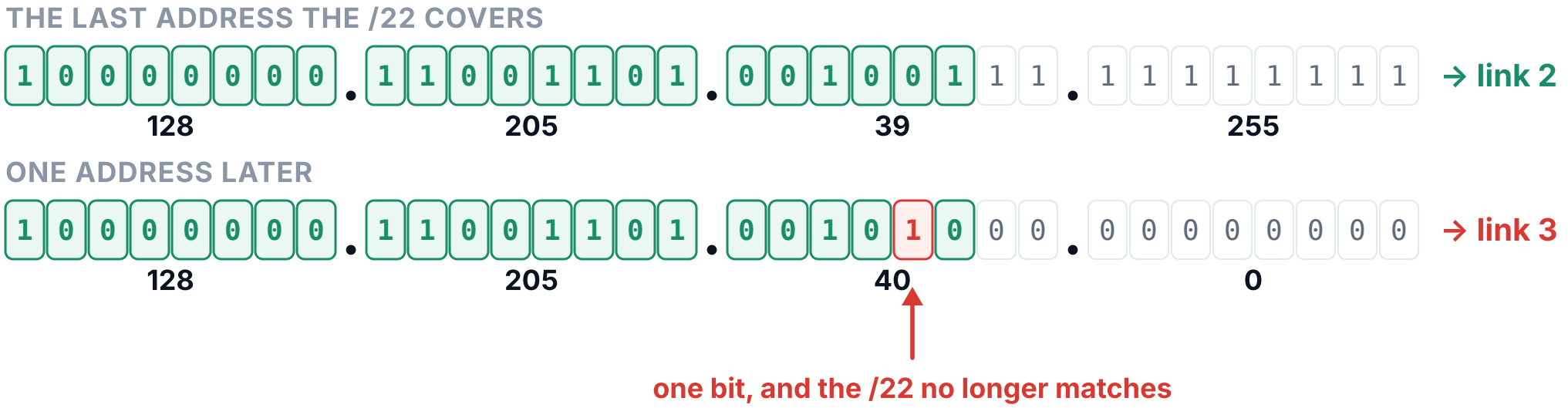
1 A row is a block | 2 The prefix | 3 Prefix matching | 4 Longest match | 5 Overlap on purpose

Address 8.8.8.8: the answer



One match: /0. Link 4.

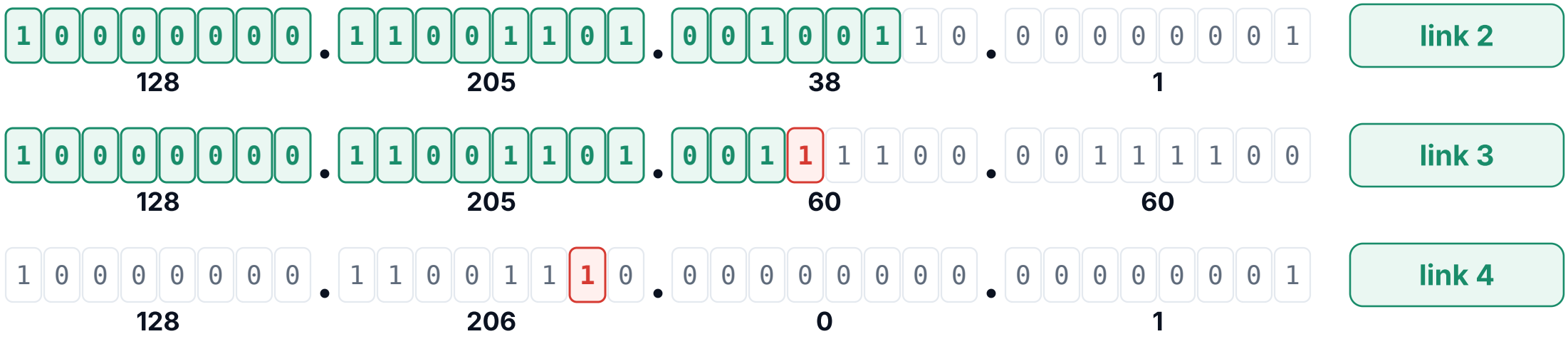
Where exactly does the /22 end?



At 128.205.39.255. One bit later, only the /19 (SEAS) still matches.

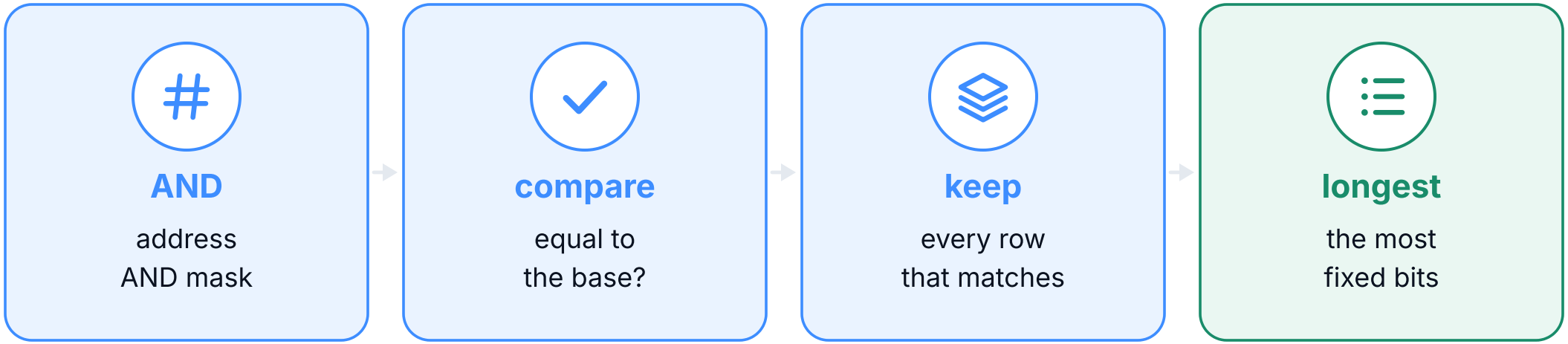
Check: three more addresses

Three more addresses: which link?



Compare from the left; the green run is the longest match.

The whole procedure



For each row: AND, compare, keep. Then take the longest.

The four lookups side by side

address	rows that match	longest
128.205.7.9	/0, /16	link 1
128.205.34.5	/0, /16, /19	link 3
128.205.37.200	/0, /16, /19, /22	link 2
8.8.8.8	/0	link 4

Four addresses, four lengths, one rule.

Why not "first match"?

THE SAME TWO ROWS, ORDER 1

prefix	output link
128.205.36.0/22	2
128.205.0.0/16	1

first match → link 2

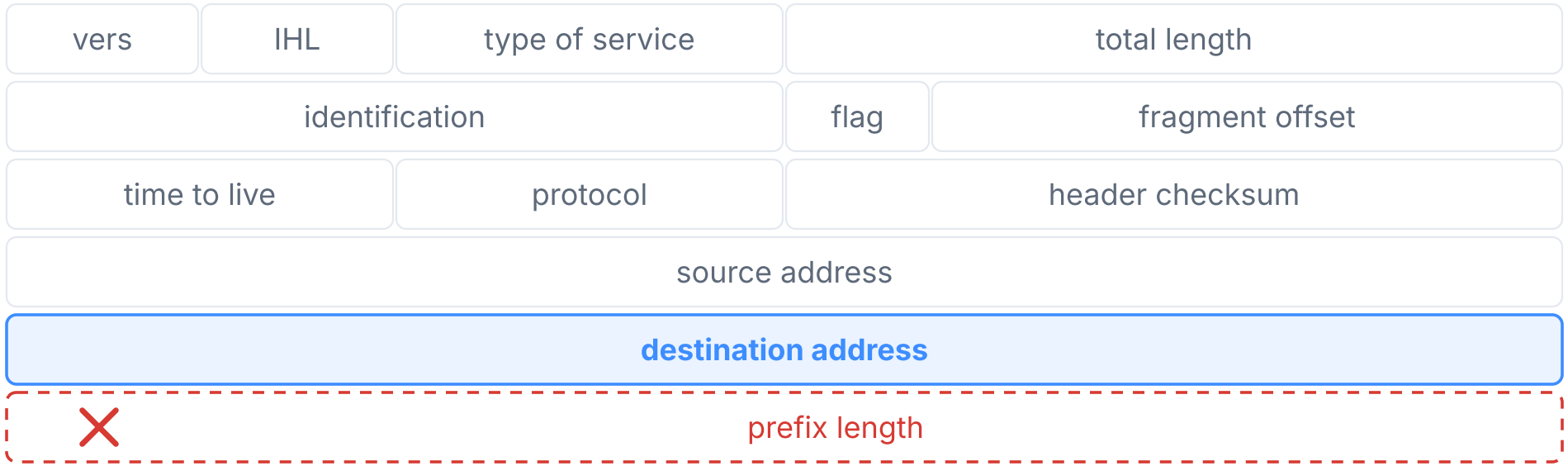
THE SAME TWO ROWS, ORDER 2

prefix	output link
128.205.0.0/16	1
128.205.36.0/22	2

first match → link 1

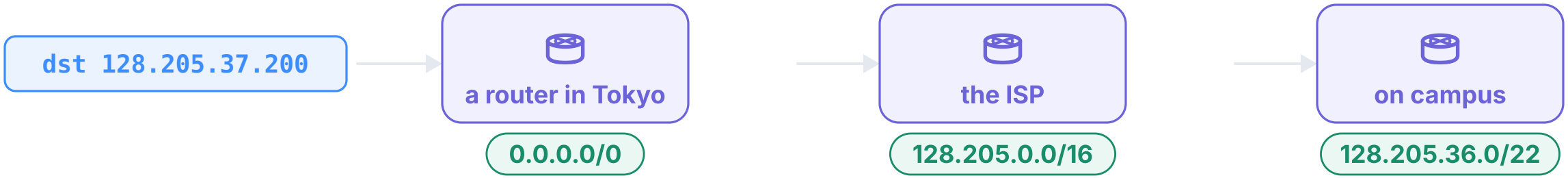
Then the order of the rows would be part of the answer.

Could the packet just say "I am CSE"?



There is no such field. The packet carries 32 bits and nothing else.

Why does the packet not carry a mask?

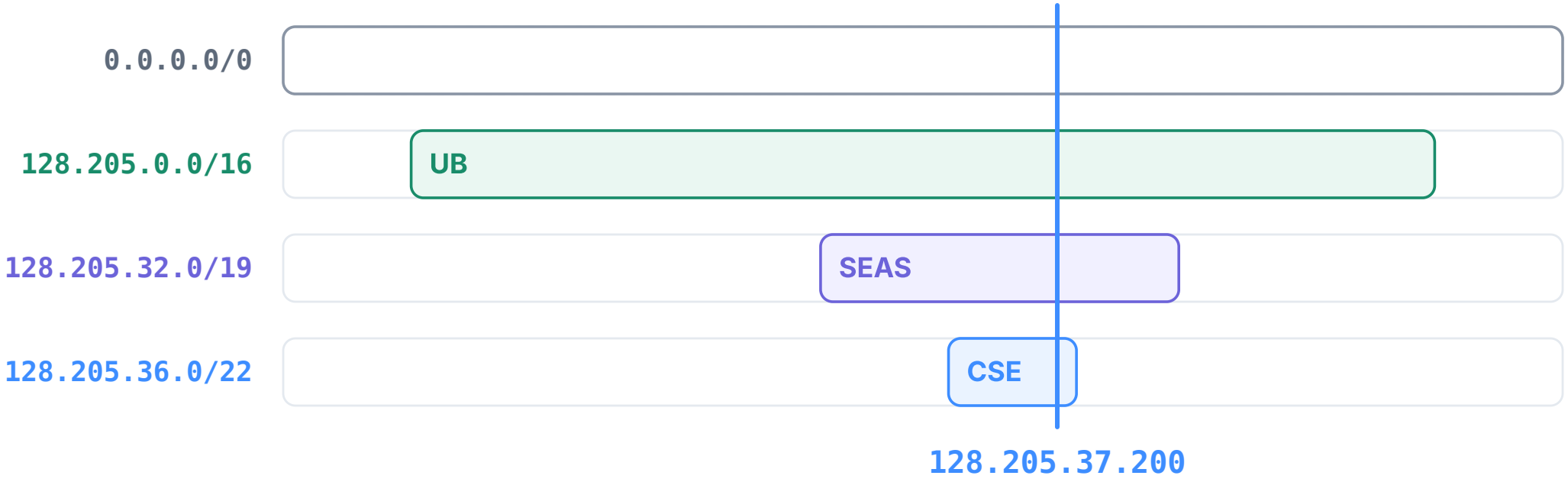


The same address, three tables, three different matching prefixes.

Which block matters depends on who is looking. The sender cannot know.

1 A row is a block | 2 The prefix | 3 Prefix matching | 4 Longest match | 5 Overlap on purpose

Which network does 128.205.37.200 belong to?



All of them. "Its network" is not a property of the address: the table decides.

1 A row is a block | 2 The prefix | 3 Prefix matching | 4 Longest match | 5 Overlap on purpose

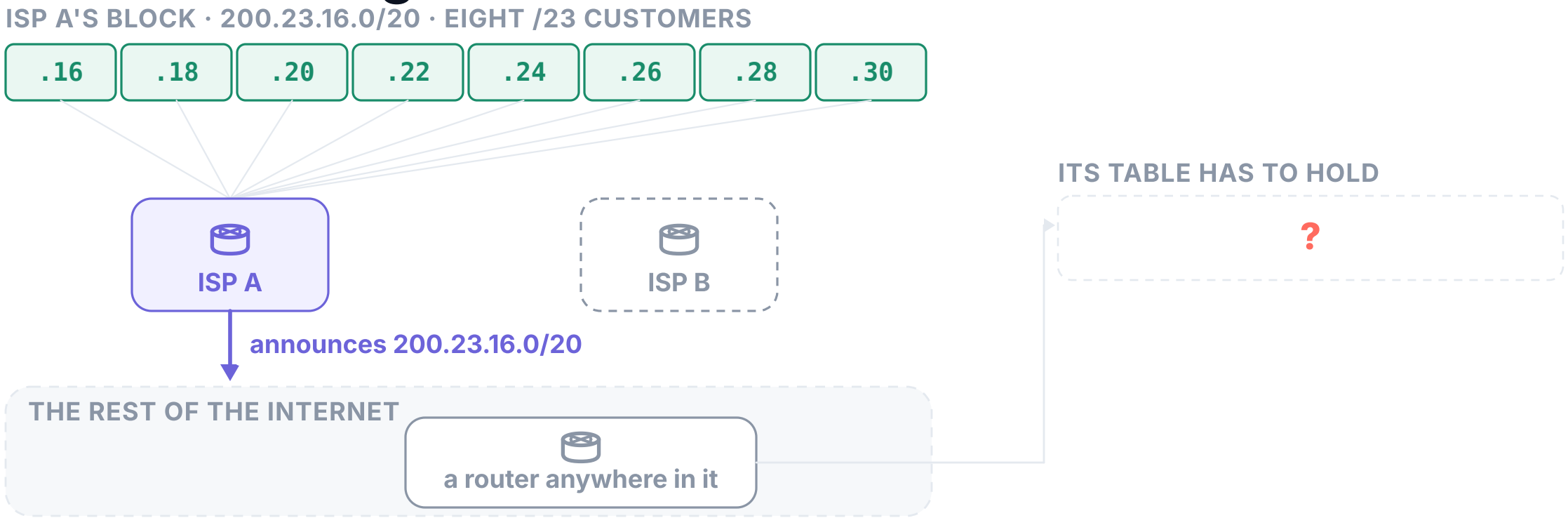
Why would anyone want rows that overlap?

05

Overlap on purpose

A short prefix keeps every table small. A long one inside it carries an exception. Longest match lets both live in one table.

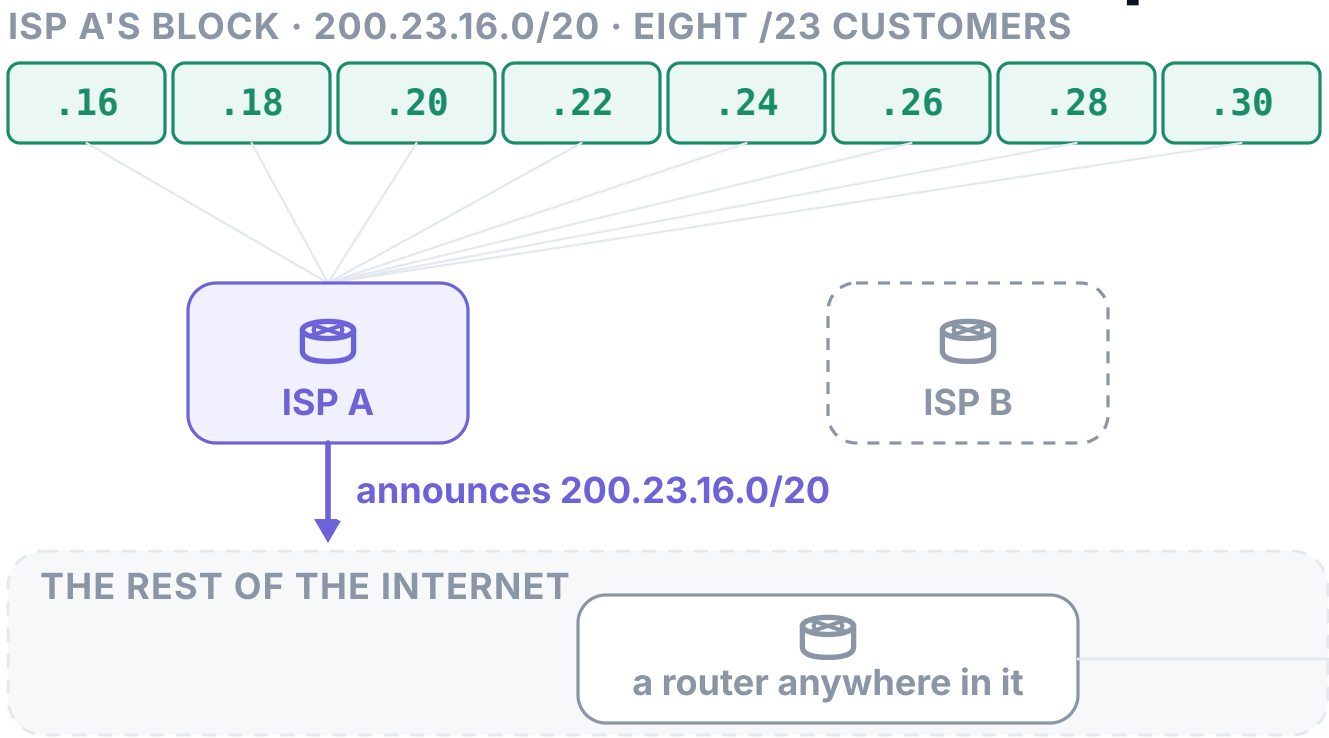
An ISP and its eight customers



Eight customers, one /23 each. What does a router elsewhere need to hold?

1 A row is a block | 2 The prefix | 3 Prefix matching | 4 Longest match | 5 Overlap on purpose

The far router could hold a row per customer

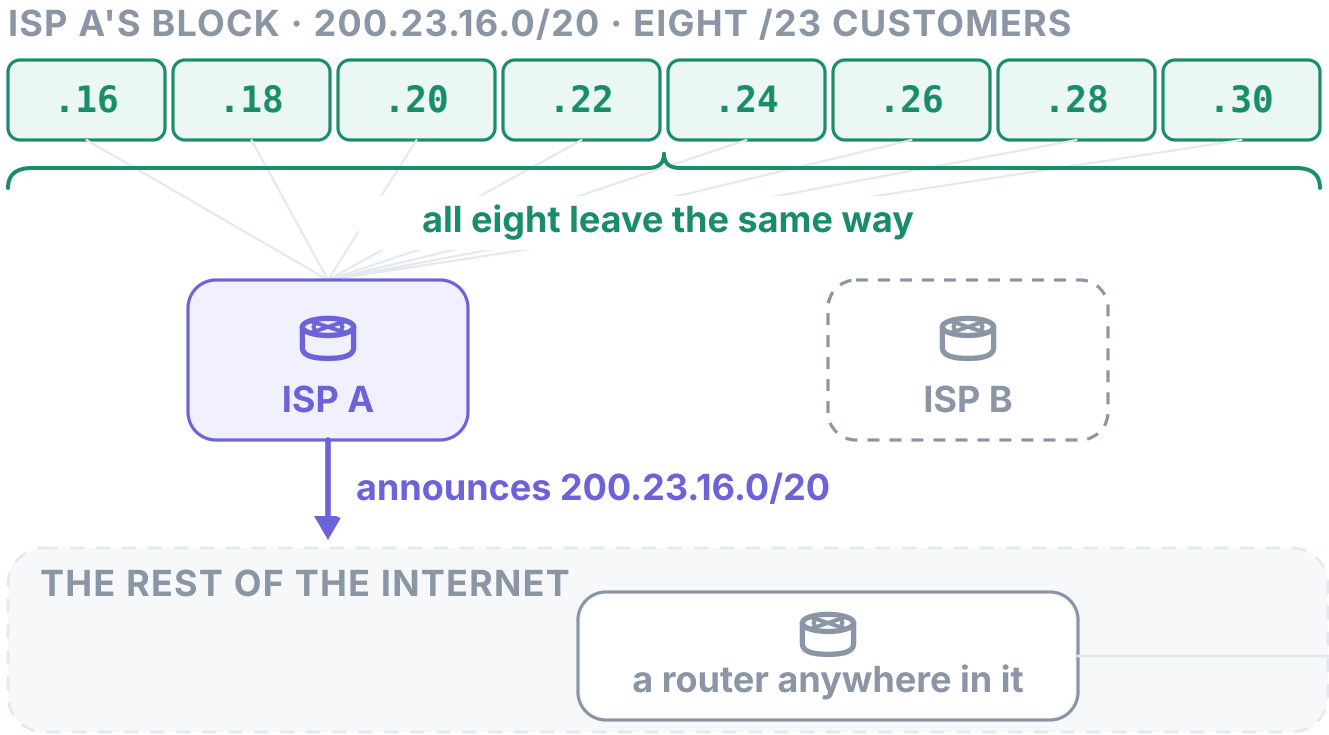


ITS TABLE HAS TO HOLD

prefix	ISP
200.23.16.0/23	A
200.23.18.0/23	A
200.23.20.0/23	A
200.23.22.0/23	A
200.23.24.0/23	A
200.23.26.0/23	A
200.23.28.0/23	A
200.23.30.0/23	A

Eight rows, in every router on the Internet, for one ISP.

Or one row: the ISP announces its whole block

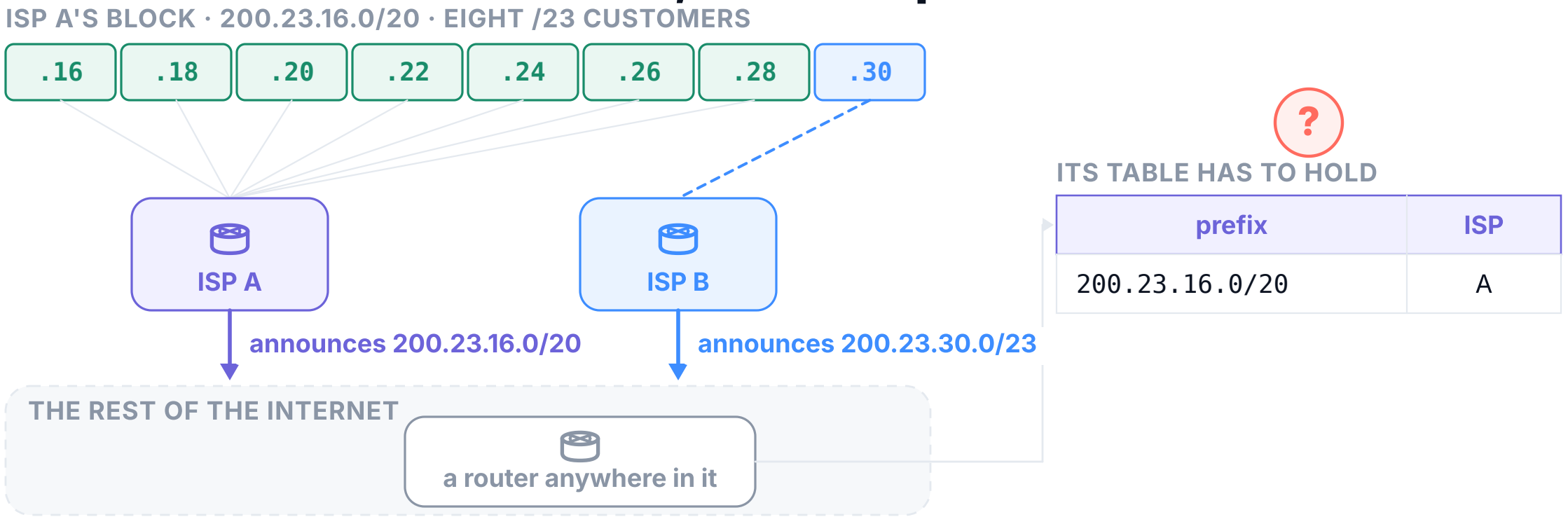


ITS TABLE HAS TO HOLD

prefix	ISP
200.23.16.0/20	A

One row for eight customers. That is what keeps the table near a million rows.

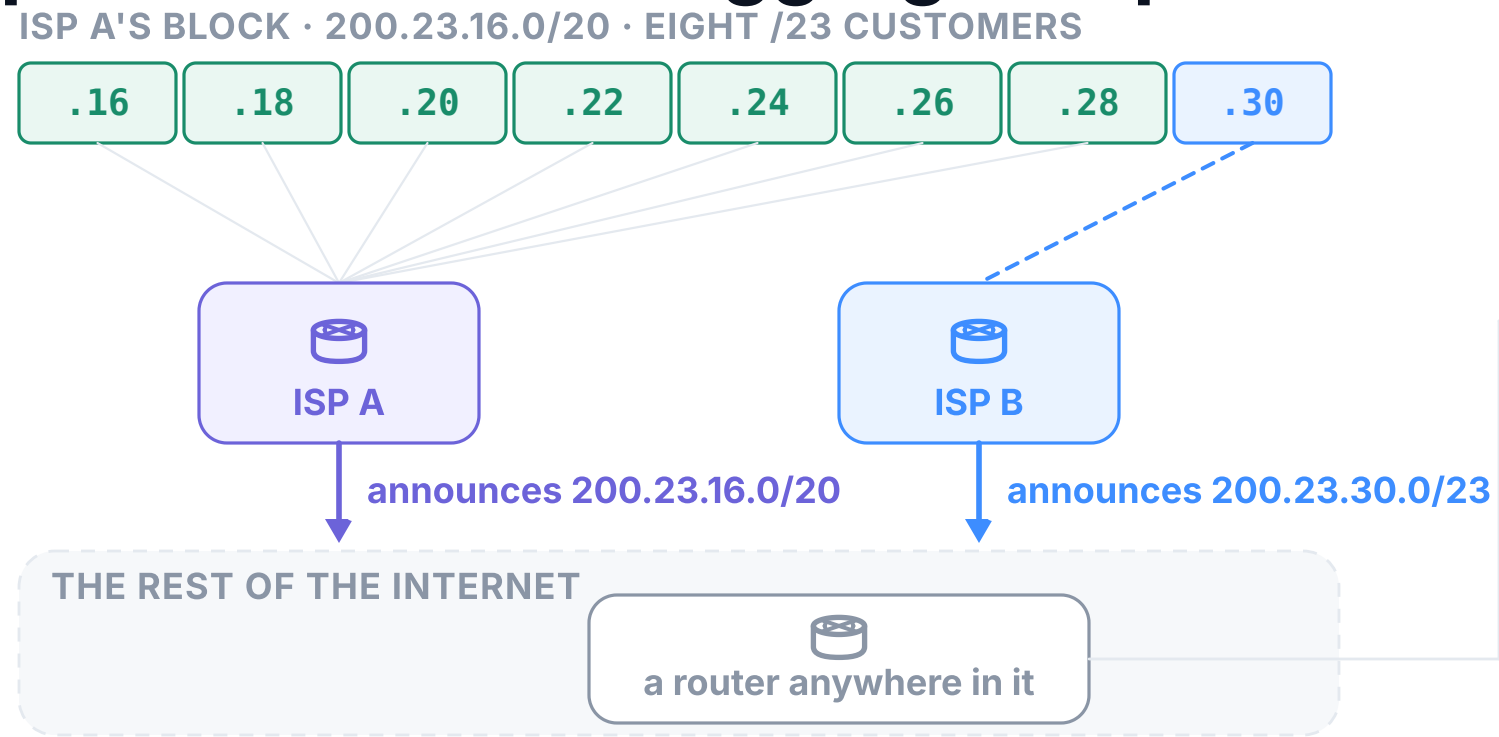
Customer 8 moves to ISP B, and keeps its addresses



The row still says A. Packets for 200.23.30.0/23 now go the wrong way.

1 A row is a block | 2 The prefix | 3 Prefix matching | 4 Longest match | 5 Overlap on purpose

Option 1: break the aggregate apart



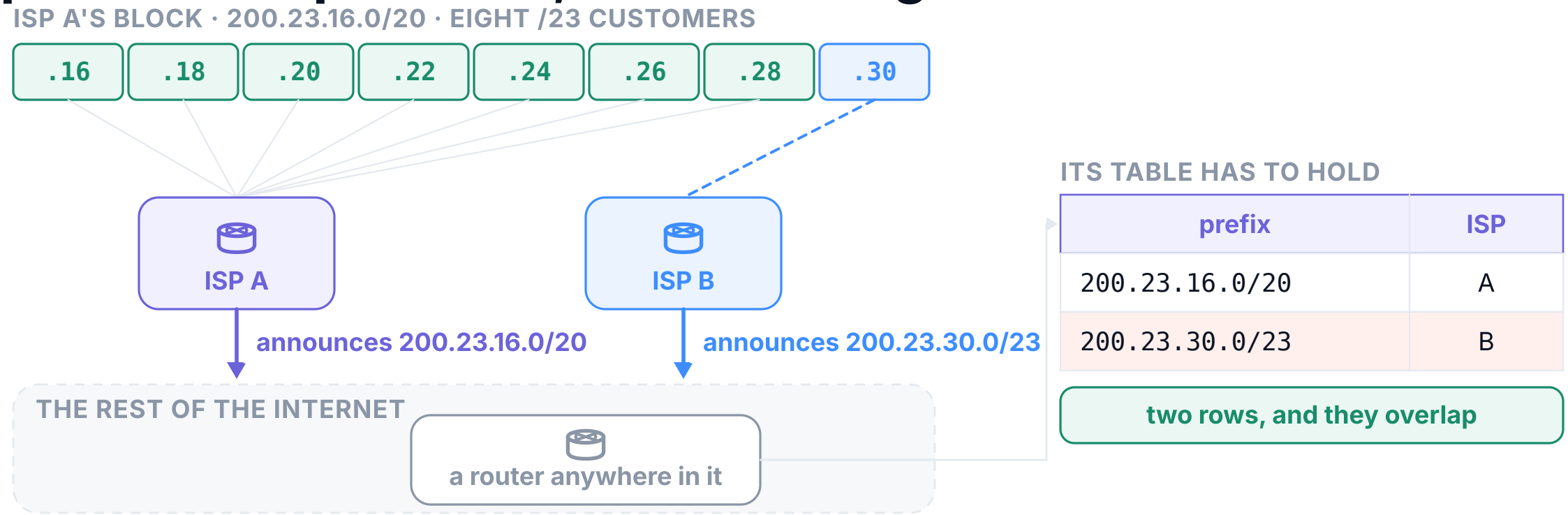
ITS TABLE HAS TO HOLD

prefix	ISP
200.23.16.0/23	A
200.23.18.0/23	A
200.23.20.0/23	A
200.23.22.0/23	A
200.23.24.0/23	A
200.23.26.0/23	A
200.23.28.0/23	A
200.23.30.0/23	B

eight rows again, everywhere

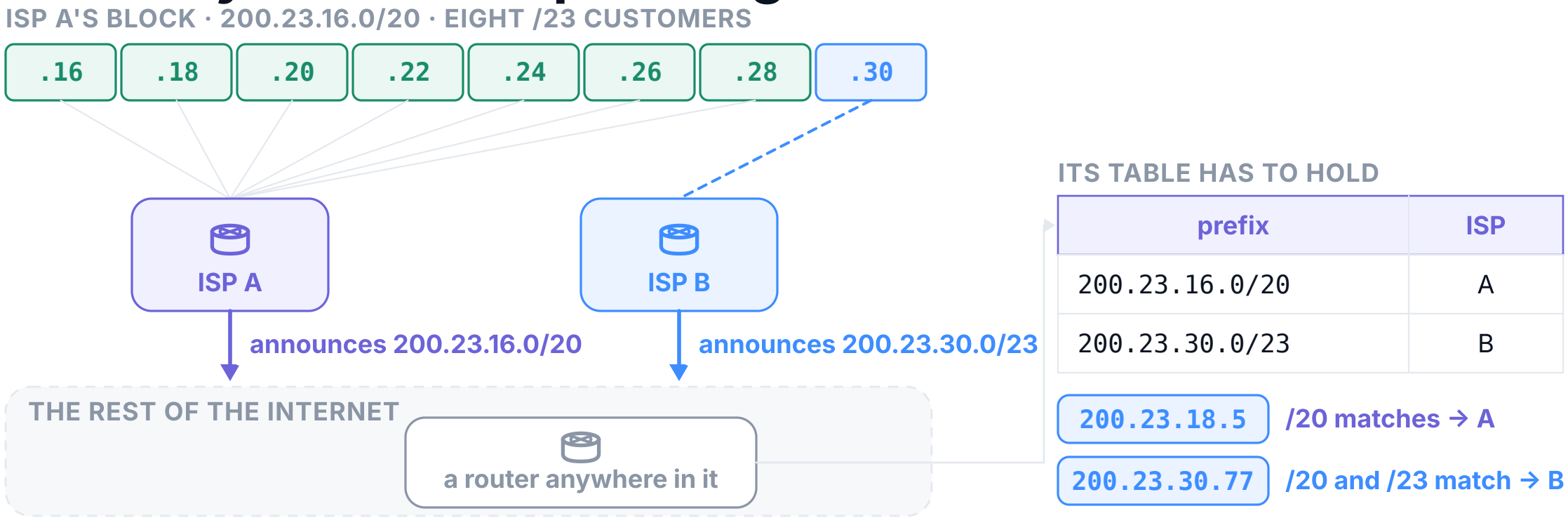
It works, and every router on the Internet grows by seven rows for one move.

Option 2: keep the /20, add one longer row



ISP A changes nothing. ISP B adds one row. Longest match sorts them out.

Which way does each packet go?



ITS TABLE HAS TO HOLD

prefix	ISP
200.23.16.0/20	A
200.23.30.0/23	B

`200.23.18.5` /20 matches → A

`200.23.30.77` /20 and /23 match → B

The /23 wins only for customer 8. Every other address in the /20 still goes to A.

1 A row is a block | 2 The prefix | 3 Prefix matching | 4 Longest match | 5 Overlap on purpose

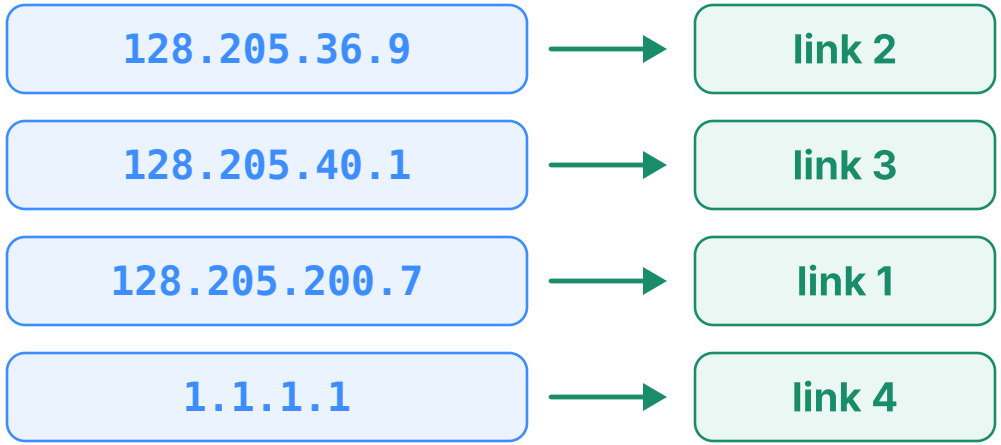
Two uses of one rule



Aggregation keeps the table small; a longer row carves out the exception.

The same table, four more packets

prefix	output link
128.205.0.0/16	1
128.205.36.0/22	2
128.205.32.0/19	3
0.0.0.0/0	4



You can now compute a real table's answer for any address.

What this lecture left open



Who writes the rows?

routing
Chapter 5



Where do blocks come from?

addressing, subnets,
DHCP, NAT



How fast can this run?

inside the router:
ports, fabric, queues

The table is the interface. The next lectures fill it and build the machine that reads it.