IP
The header, the address, and running out

The forwarding lecture read one field of the packet and picked a link. This lecture is the IP protocol itself: the header around that field, a few of the fields in it, and above all the address: where it comes from and what the internet did when the supply ran out.

Yaxiong Xie

128.205.13.7

one address, 32 bits

10000000 11001101 00001101 00000111

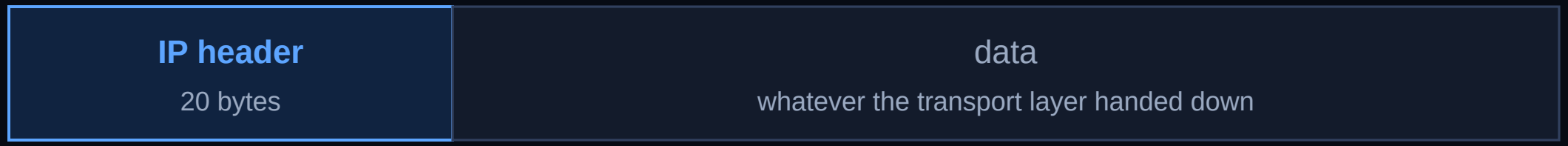
four numbers, 0 to 255 each, eight bits apiece

- **a 20-byte header on every packet**
- **every address is handed down a chain**
- **4.29 billion of them, and they ran out**

five topics, each its own story

This lecture: the IP protocol

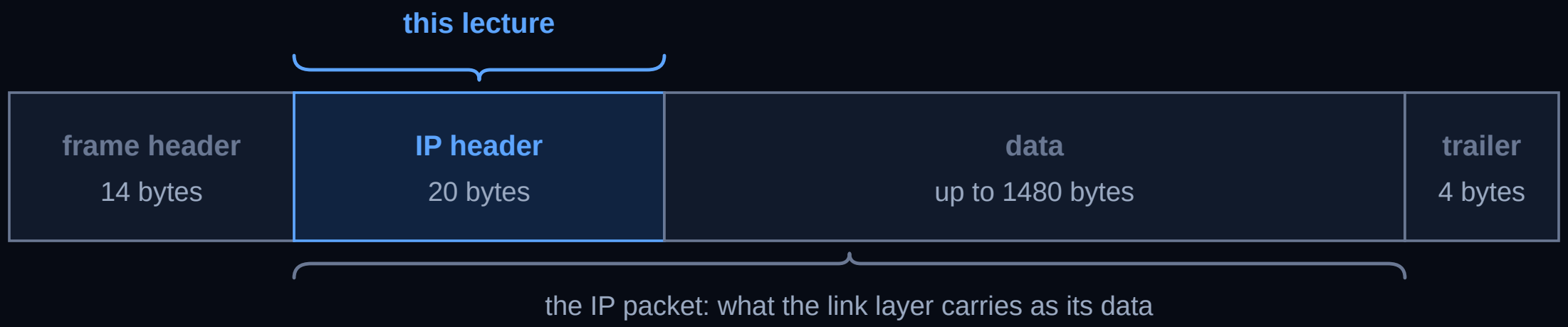
the IP protocol is this header, and the rules for it



The forwarding lecture read one field of it. This lecture is the rest.

Every packet on the internet is 20 bytes of IP header in front of the data.

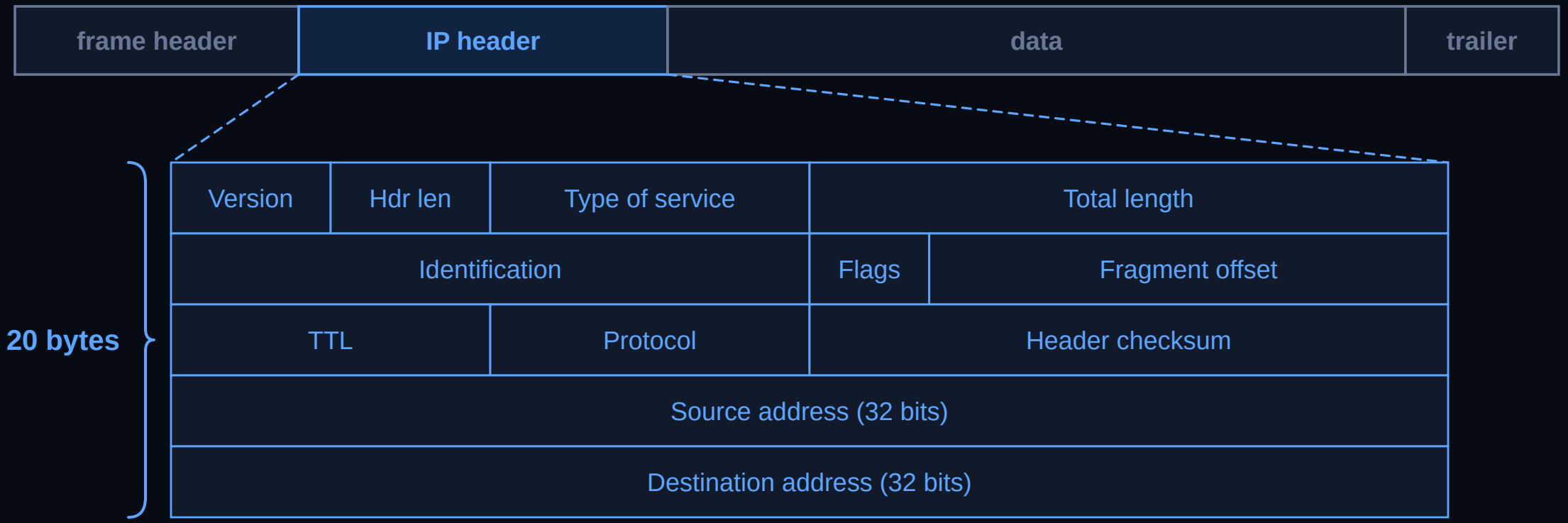
Where does the IP header sit?



Every layer puts its own header in front of what it was given.
The frame carries the whole IP packet as its data; to IP, everything behind its own header is data.

Inside the frame, at the front of the packet. Everything behind it is data to IP.

What is inside the IP header?



thirteen fields; the two addresses alone are 8 of the 20 bytes

Thirteen fields in 20 bytes. Open it up: the two addresses take almost half.

What does IP promise about delivery?

IN THE HEADER



where to deliver it
and who sent it

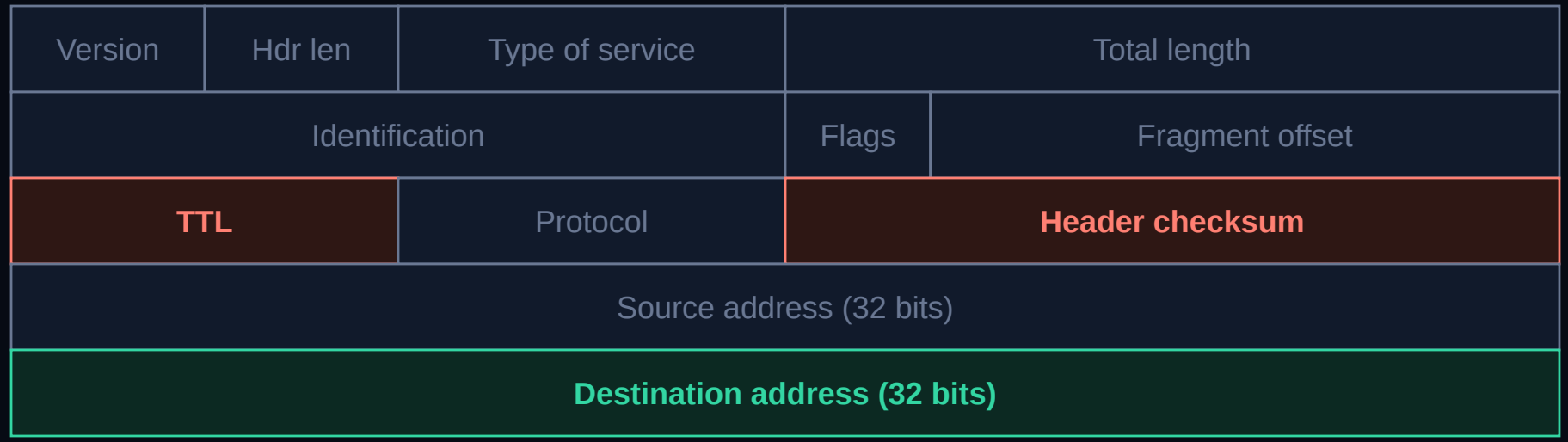
so IP promises to try to deliver it

NOT IN THE HEADER

- ✗ **a sequence number** — so no ordering
- ✗ **an acknowledgement** — so no reliability
- ✗ **a retry count** — so no retransmission
- ✗ **a deadline** — so no delay bound

Best effort: it will try. Order, reliability and timing are the transport layer’s job.

Does a router need all thirteen fields?



grey: the two ends' business, carried but never read by a router

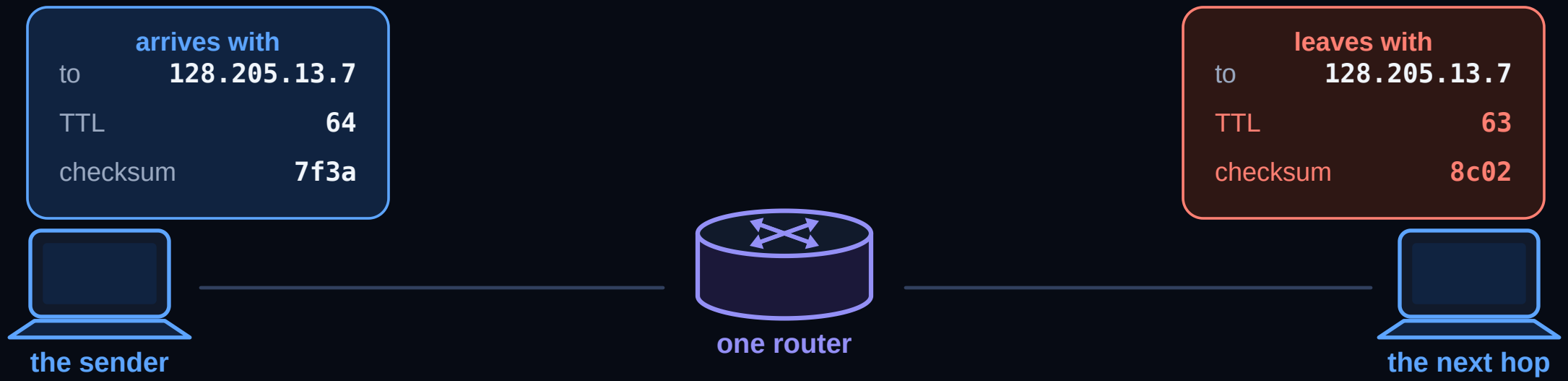
the source address, for one, is for the server: it says where to send the reply

read: where is it going

read: still intact, and still allowed on

No. Three: where it goes, whether the header is intact, how far it may still travel.

Does the router change anything in the header?

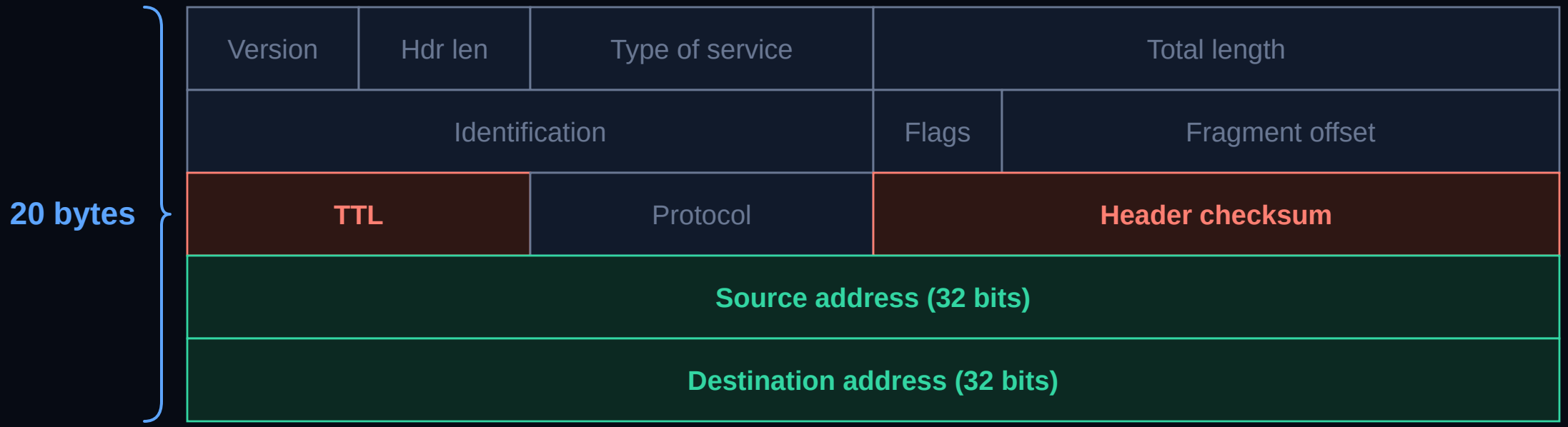


Two fields are rewritten at every router; the destination is only read.
The checksum values are made up; the real one is computed from the header bytes.

Yes, two fields: TTL goes down by one, so the checksum must be redone. At every hop.

So which fields will this lecture explain?

the two a router rewrites: Parts 01, 02 — the address: Parts 03, 04, 05 — named today, not explained



Four of thirteen: the two fields a router rewrites, and the two addresses.

Five topics, each from its own problem

01



TTL

a packet that never dies

02



Checksum

a bit flips on the way

03



IP allocation

who hands out addresses

04



NAT

when addresses run out

05



IPv6

the fix, and why so slow

Two header fields, then the address: where it comes from, running out, the fix.

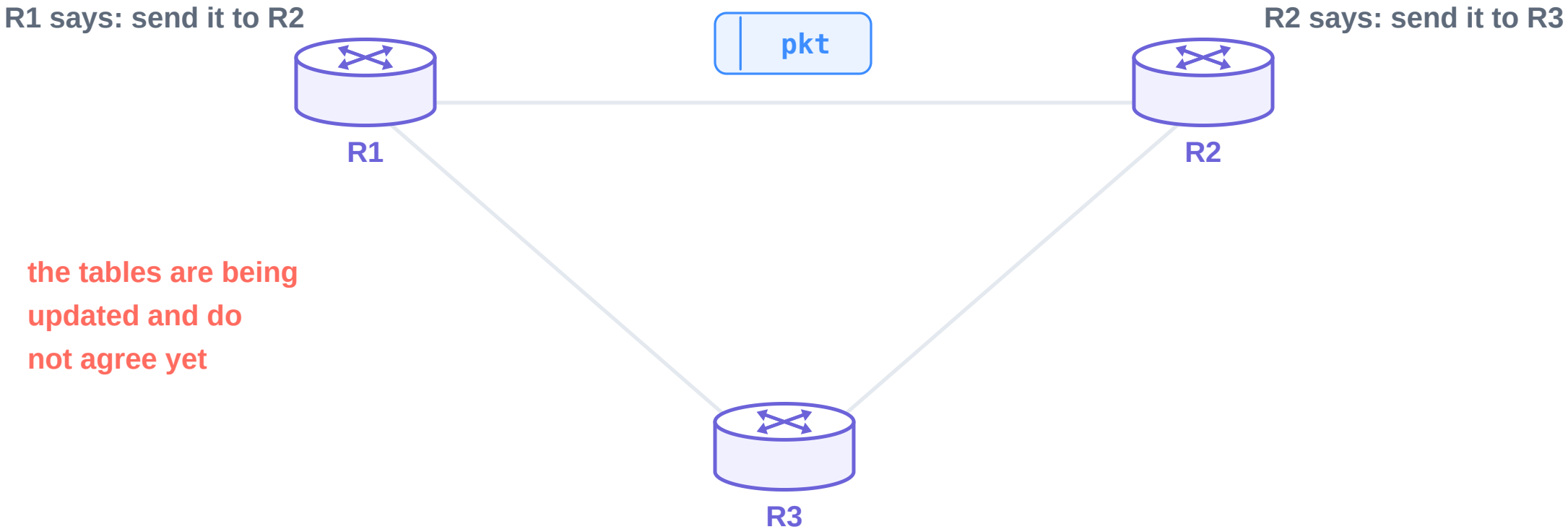
Why does a packet need a lifetime?

01

TTL

Routers do not always agree, and a packet caught between two that disagree would circle forever. One 8-bit field ends that, and gives us traceroute for free.

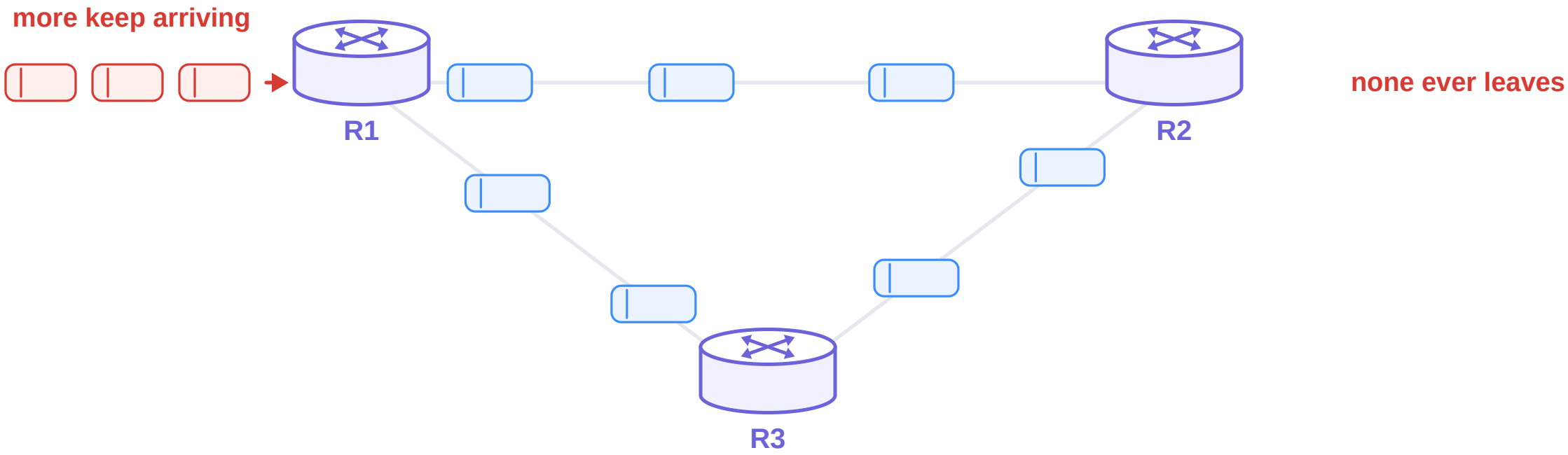
What if three routers do not agree with each other?



the tables are being updated and do not agree yet

While routing converges the tables disagree, and a packet can go in a circle.

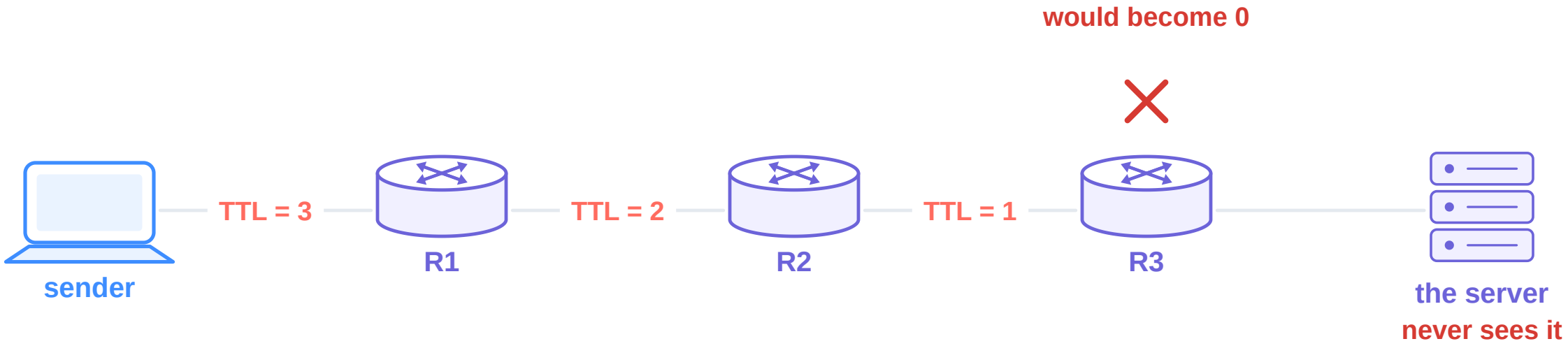
Does a looping packet go away on its own?



Every router on the path is forwarding correctly.
Nothing in the loop removes a packet, so they pile up until the links carry nothing else.

No. Whatever enters the loop stays in it, more keep coming, and the links fill up.

How does a packet die on its own?

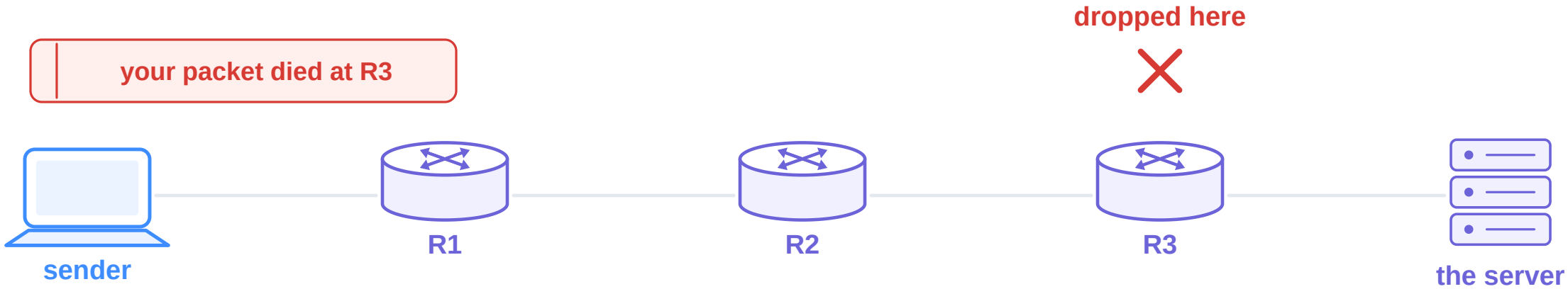


Every router subtracts one; the router that would reach zero drops it.
Sent with 3 here so the drop fits on the slide. Real packets start at 64 (Linux, macOS) or 128 (Windows).

A counter every router lowers by one. The router that takes it to zero drops the packet.

1 TTL | 2 Checksum | 3 IP allocation | 4 NAT | 5 IPv6

What comes back when a packet is dropped?



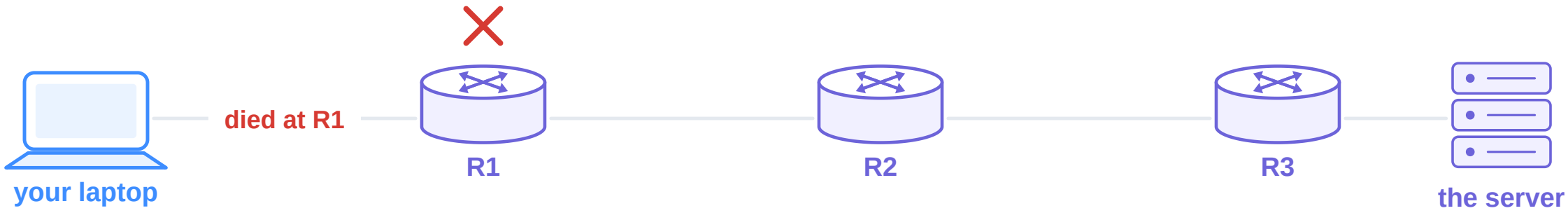
the message the router sends back

from	**R3's own address**
to	**the sender**
says	**your packet ran out of hops here**

The sender now knows the address of the router where the packet died.

A short message from the router that dropped it, carrying that router's own address.

Can we use that on purpose?



what the laptop learns from that one reply

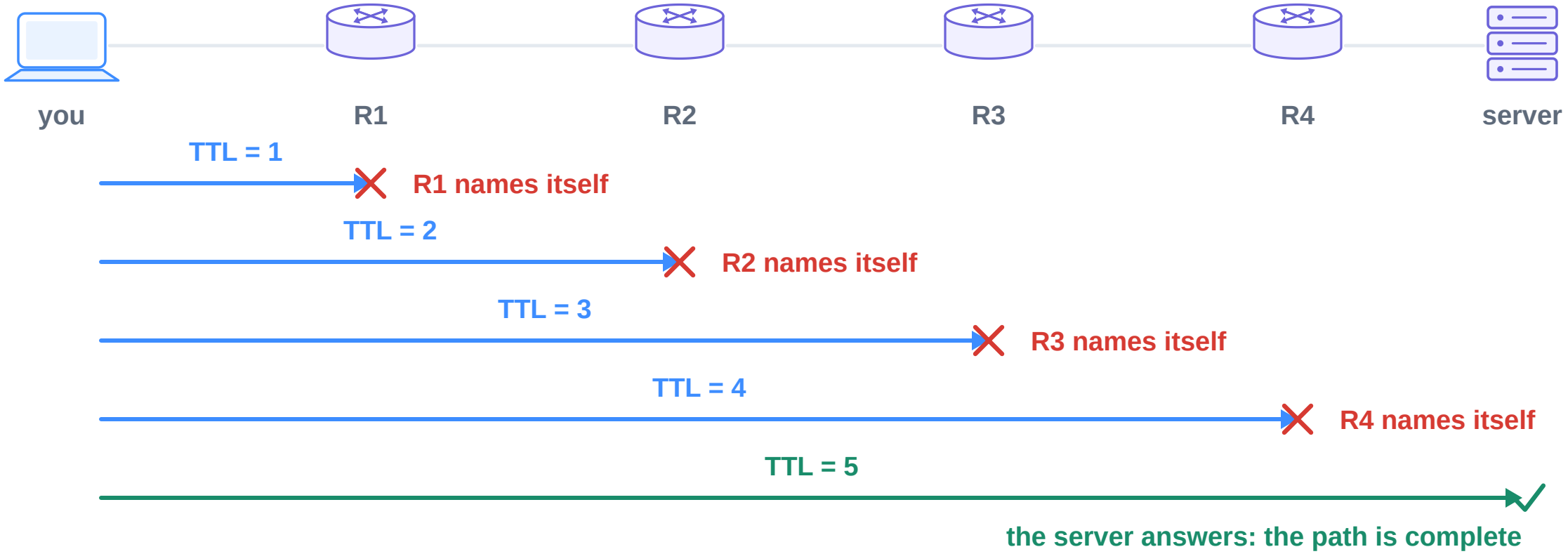
the address of the first router — it is the source of the message

the time to it and back — the laptop looked at its clock, twice

R1 had to name itself in order to report the drop, and that name is what we wanted.

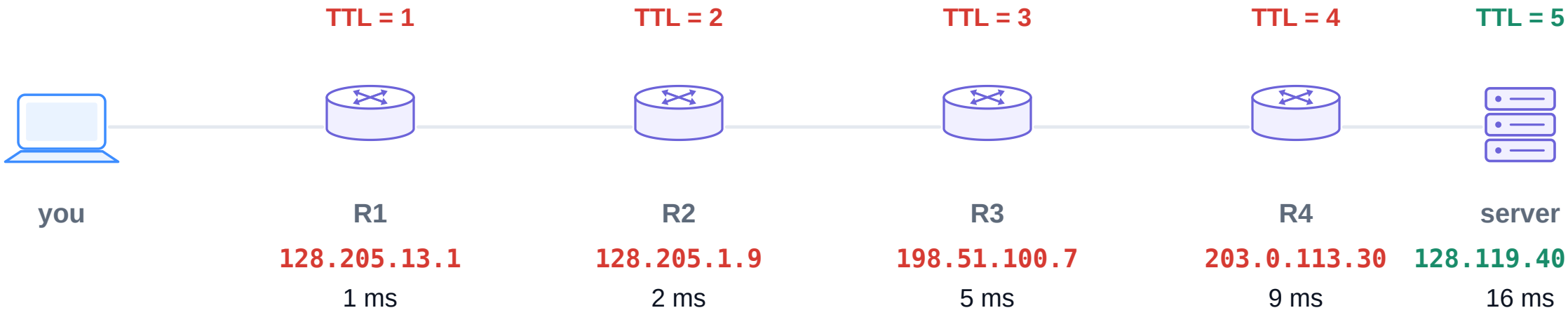
Send TTL = 1 on purpose. The reply names the first router and times the trip to it.

How do you find the whole path?



Send TTL = 1, 2, 3, ... Each router in turn is forced to name itself, until the server answers.

What do we know at the end?



Five packets, five replies: the whole route, one router per TTL value.
Traceroute is exactly this: TTL = 1, 2, 3, ... and the replies. Addresses and times are made up.

The route, hop by hop: every router’s address and the time to reach it.

What does traceroute print?

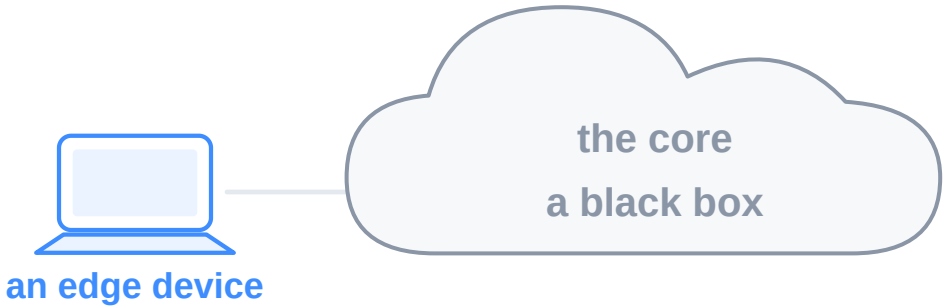
TTL	who answered	time
1	gateway.home	3 ms
3	border.campus.example	3 ms
5	core.campus.example	4 ms
6	* * *	no reply
7	edge1.regional.example	6 ms
8	core3.backbone.example	8 ms
14	server.example.com	16 ms

Names and times here are made up to show the shape of the output. Run it yourself and read the real lines in class.

Each line is one TTL value, and a star means that router chose not to answer.

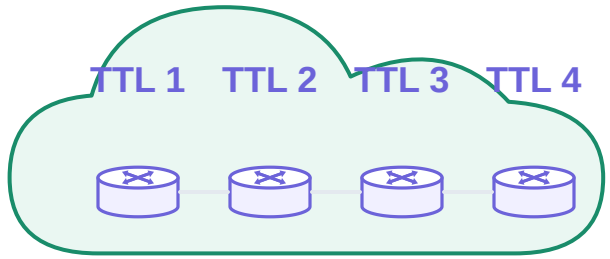
What did we just gain?

BEFORE



it sees its first router and nothing past it

WITH TTL



every router inside can be made to name itself

An edge device can now probe the core, one hop at a time.
Nothing in the core was changed; the TTL field and the message back were already there.

The core was a black box. With TTL, an edge device makes every router inside name itself.

1 TTL | 2 Checksum | 3 IP allocation | 4 NAT | 5 IPv6

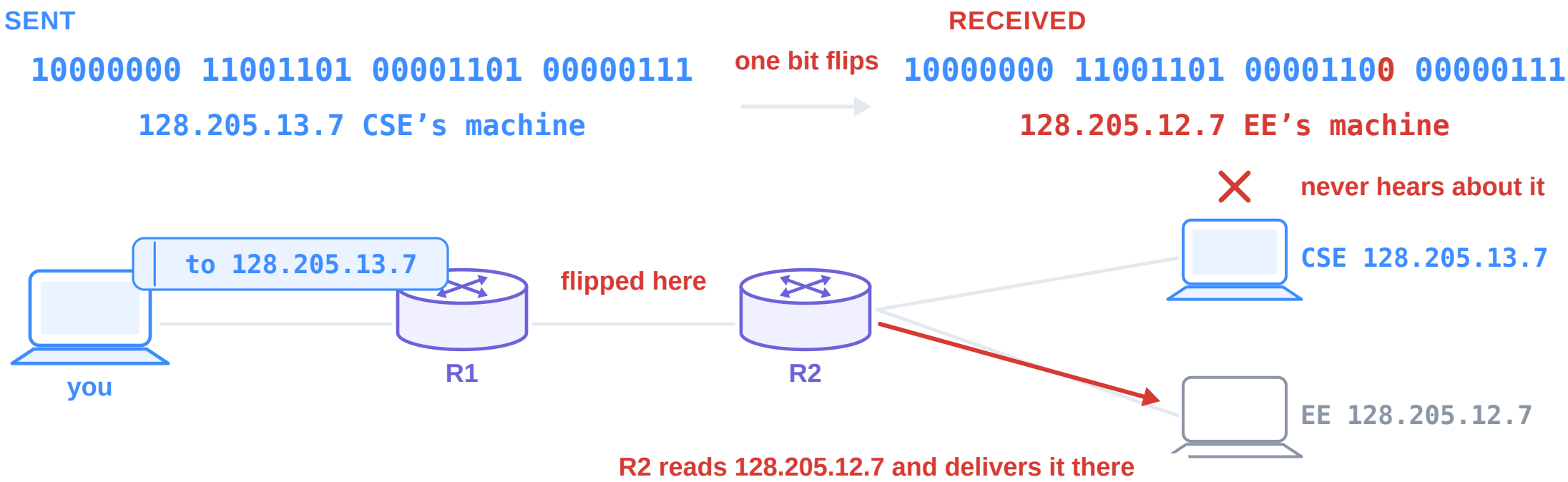
What if a bit of the header flips on the way?

02

The checksum

A corrupted destination address is worse than a lost packet: it arrives at the wrong machine. One 16-bit field catches it, at a price paid at every hop.

What if one bit of the destination address flips on the way?

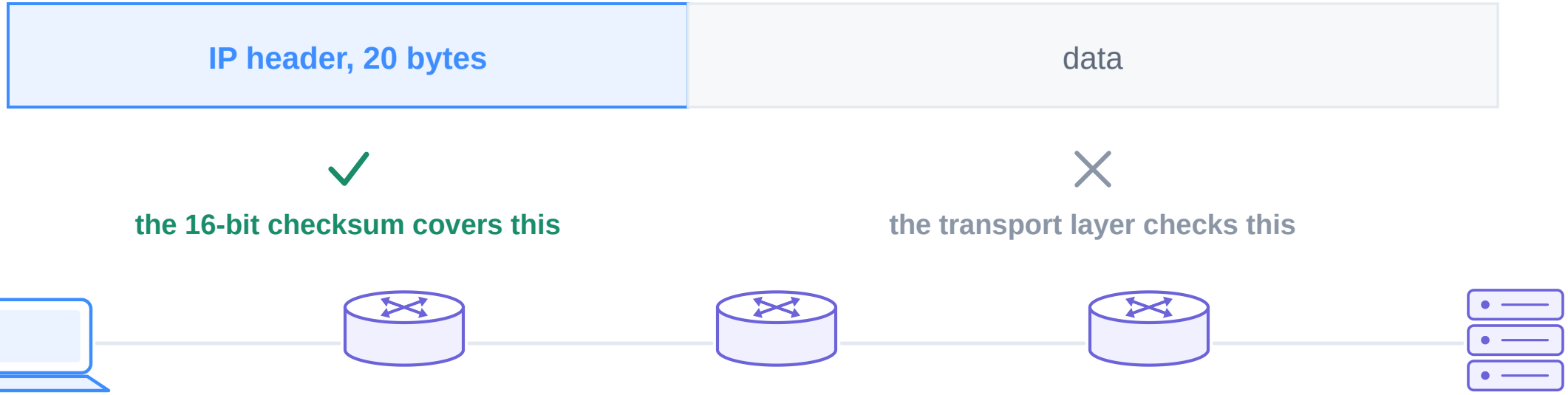


Worse than a lost packet: a lost packet is missed and sent again.
This one arrives, at a machine that never asked, and the machine that asked never learns that anything was sent.

It is delivered to the wrong machine, EE's instead of CSE's, and nobody notices.

1 TTL | 2 Checksum | 3 IP allocation | 4 NAT | 5 IPv6

How is a corrupted header caught?

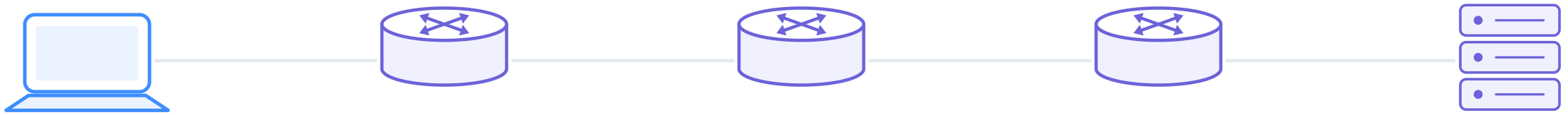


Header only: the data is already checked end to end by TCP or UDP, and the header is what a router acts on.

A checksum over the header only, verified by every router. A bad header is dropped.

1 TTL | 2 Checksum | 3 IP allocation | 4 NAT | 5 IPv6

What does the checksum cost?



TTL − 1
checksum again

TTL − 1
checksum again

TTL − 1
checksum again

TTL changes at every hop (Part 01), so the checksum changes with it.

Verified and recomputed, on every packet, at every router.
A checksum over a header that changes at every hop has to be redone at every hop. That is the price of Part 01.

One recomputation per hop, on every packet, because one header field changes per hop.

1 TTL | 2 Checksum | 3 IP allocation | 4 NAT | 5 IPv6

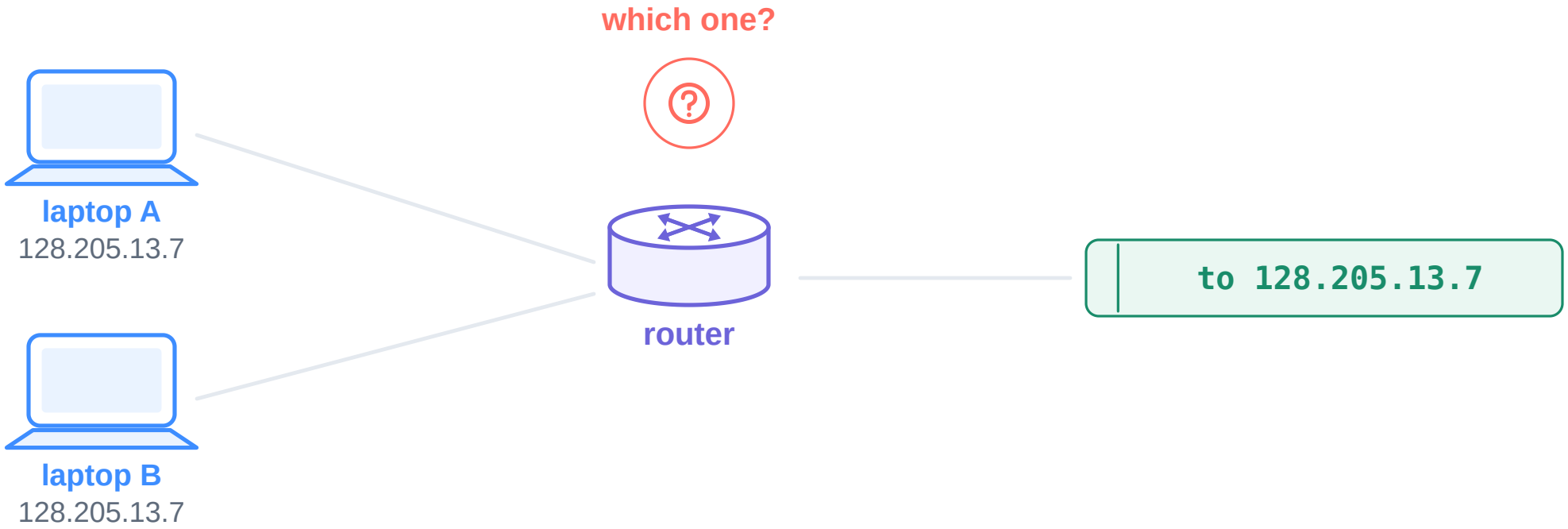
Who decides which addresses you may use?

03

IP allocation

Nobody picks their own. Every address comes down a chain of cuts, from the whole supply to a region, to UB, to a school, to a department; and on each subnet a server hands one address to each machine that asks.

What if every machine picked its own address?



So it is not allowed. Every address is handed out; nobody picks their own.
An address has to be unique across the whole internet, so the supply is strictly divided up, and each machine is given one.

Two would pick the same one, and no router could tell them apart. So nobody may.

1 TTL | 2 Checksum | 3 IP allocation | 4 NAT | 5 IPv6

Where do all the addresses start?

every IPv4 address there is: 0.0.0.0 to 255.255.255.255, 4,294,967,296 of them



one registry per region; each hands out its own piece

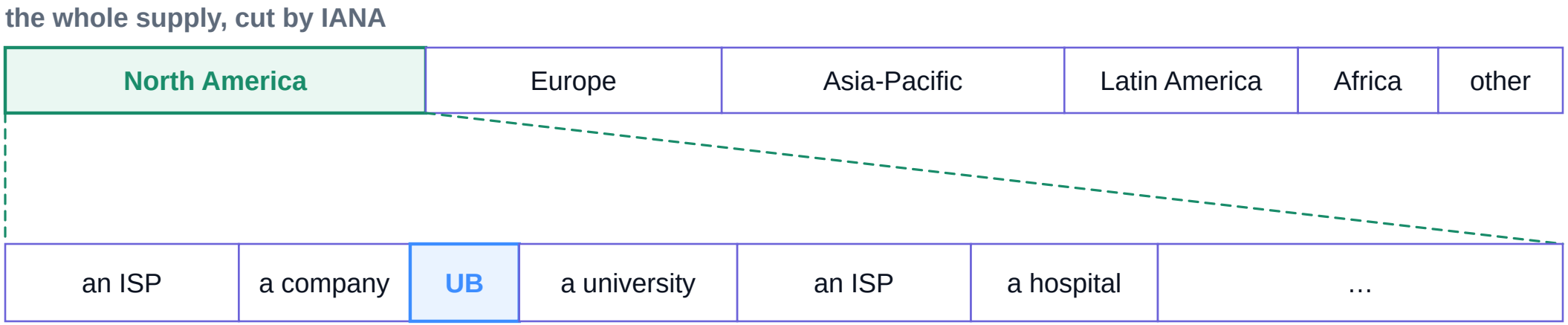
IANA cuts the supply into big pieces, one per region of the world.
The real pieces are many and uneven, handed out over thirty years; the drawing shows the idea. Every region has one registry.

Press → to cut. Piece sizes are only indicative.

In one place, IANA, which cut the supply into pieces for the regions of the world.

1 TTL | 2 Checksum | 3 IP allocation | 4 NAT | 5 IPv6

Zoom in: what does a region do with its piece?



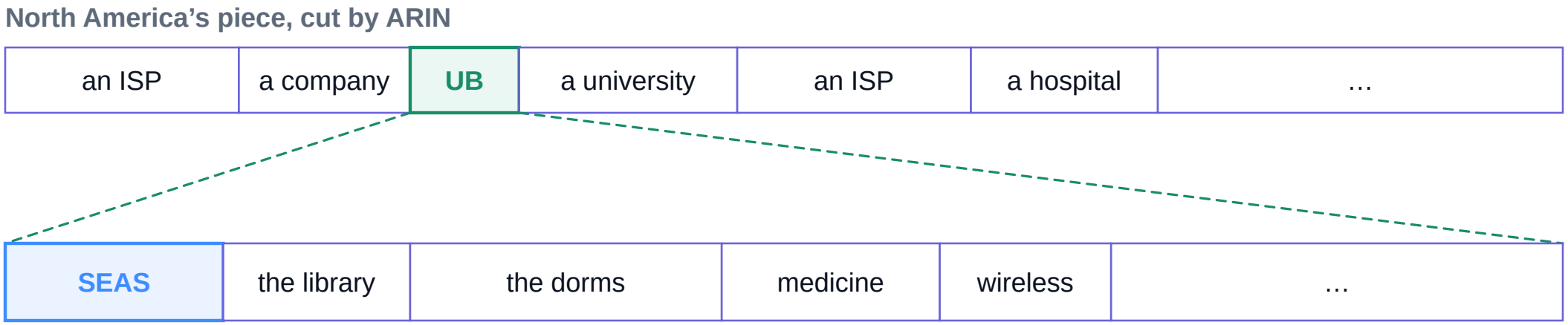
North America's piece, handed out by ARIN to whoever asks and can justify it

UB was given 128.205.0.0 to 128.205.255.255: 65,536 addresses in a row

A registry cuts its piece into one block per organisation.
UB's block is real (ARIN record NET-128-205-0-0-1); the neighbours and the piece sizes are illustrative.

The registry cuts it into one block per organisation. UB got 65,536 addresses.

Zoom in: what does UB do with 65,536 addresses?



UB’s block: 128.205.0.0 to 128.205.255.255, cut by UB itself

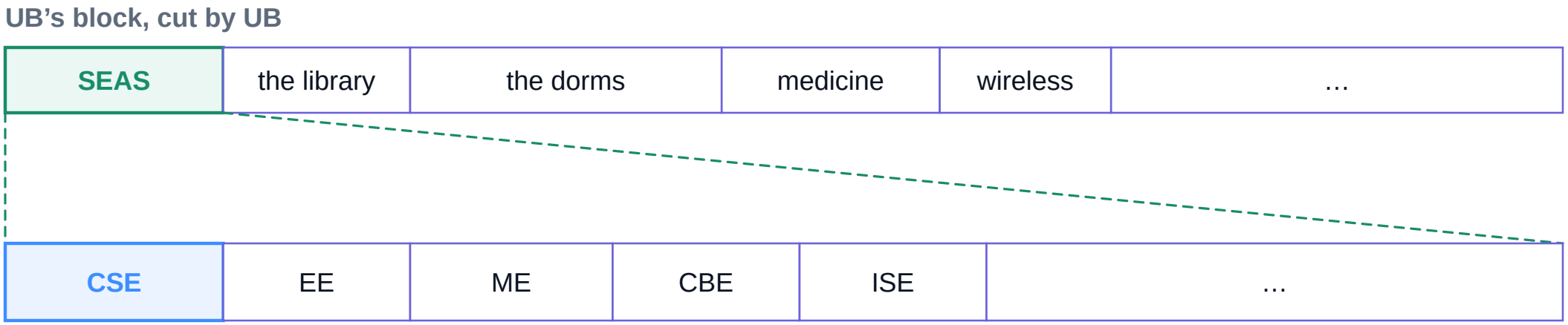
SEAS gets 128.205.8.0 to 128.205.15.255: 2,048 addresses

UB does to its block what ARIN did to its piece: cuts it into smaller blocks.
One block per school or building. The cuts inside UB are illustrative.

UB cuts its block the same way: one piece per school or building. SEAS gets 2,048.

1 TTL | 2 Checksum | 3 IP allocation | 4 NAT | 5 IPv6

Zoom in: what does the school do with its piece?



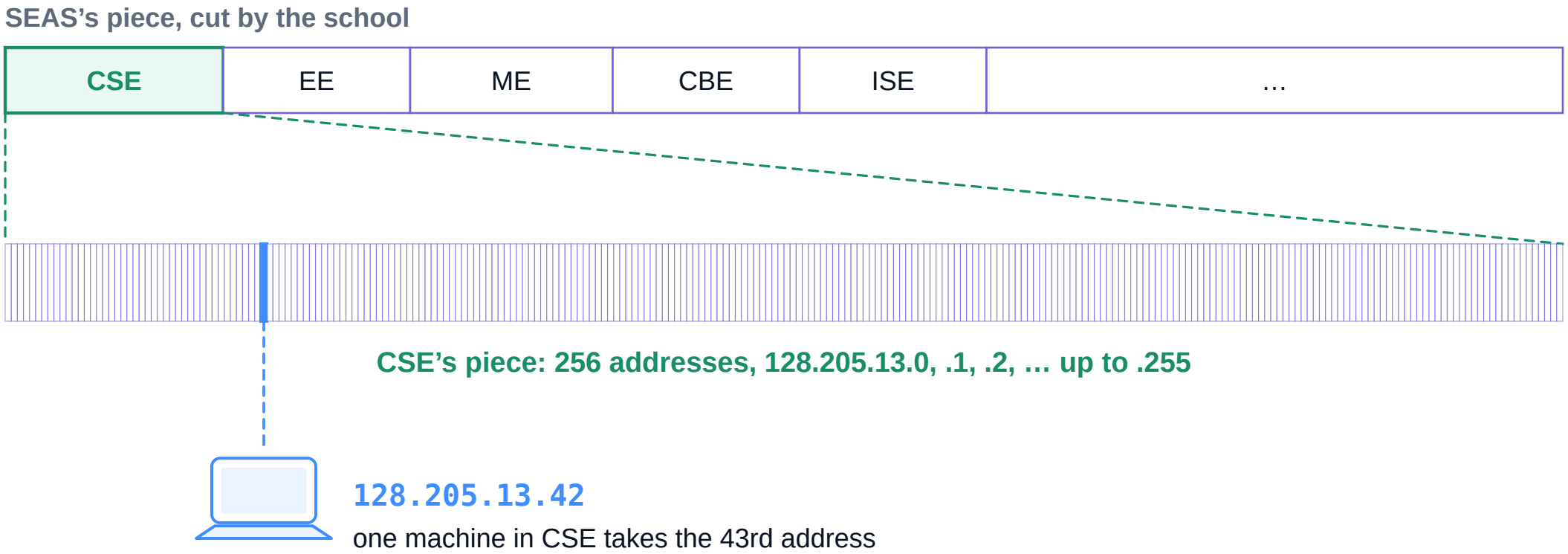
SEAS's piece: 128.205.8.0 to 128.205.15.255, cut by the school

CSE gets 128.205.13.0 to 128.205.13.255: 256 addresses

And the school cuts again: one piece per department.
Every address in CSE's piece starts with 128.205.13; that is what a piece is.

And SEAS cuts again: one piece per department. CSE gets 256 addresses.

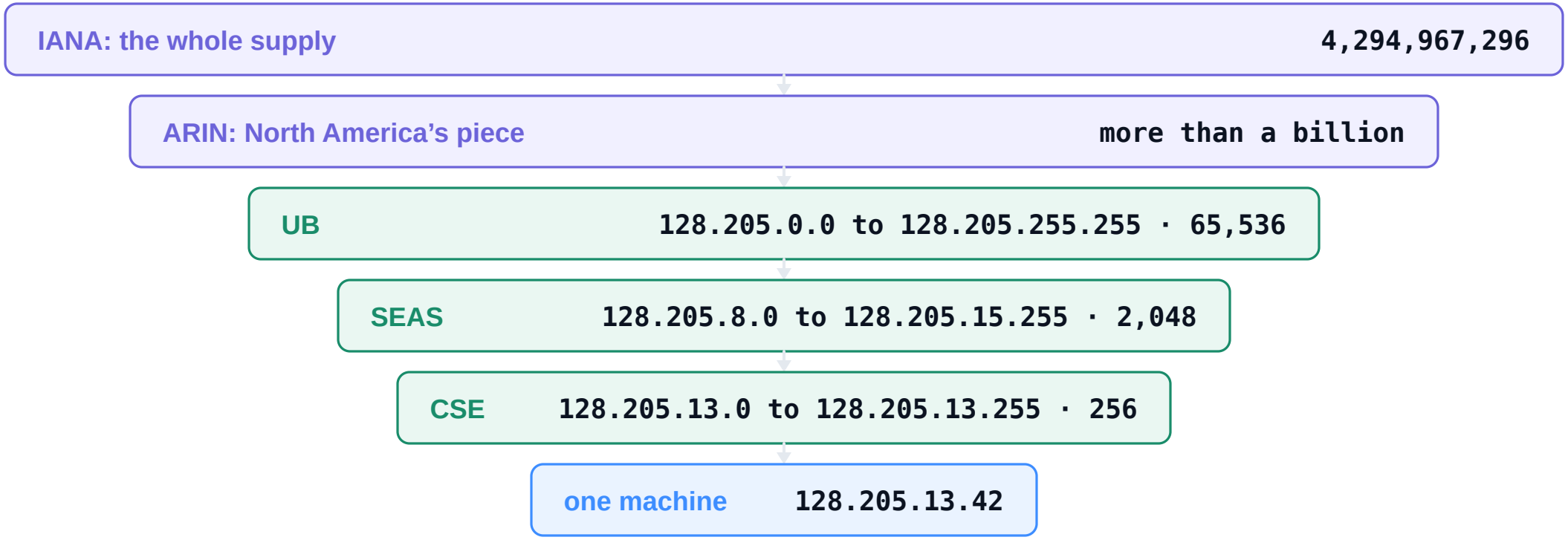
The last step: one machine in CSE



A machine in CSE gets one address out of CSE’s piece: 128.205.13.42.

1 TTL | 2 Checksum | 3 IP allocation | 4 NAT | 5 IPv6

The whole chain at a glance

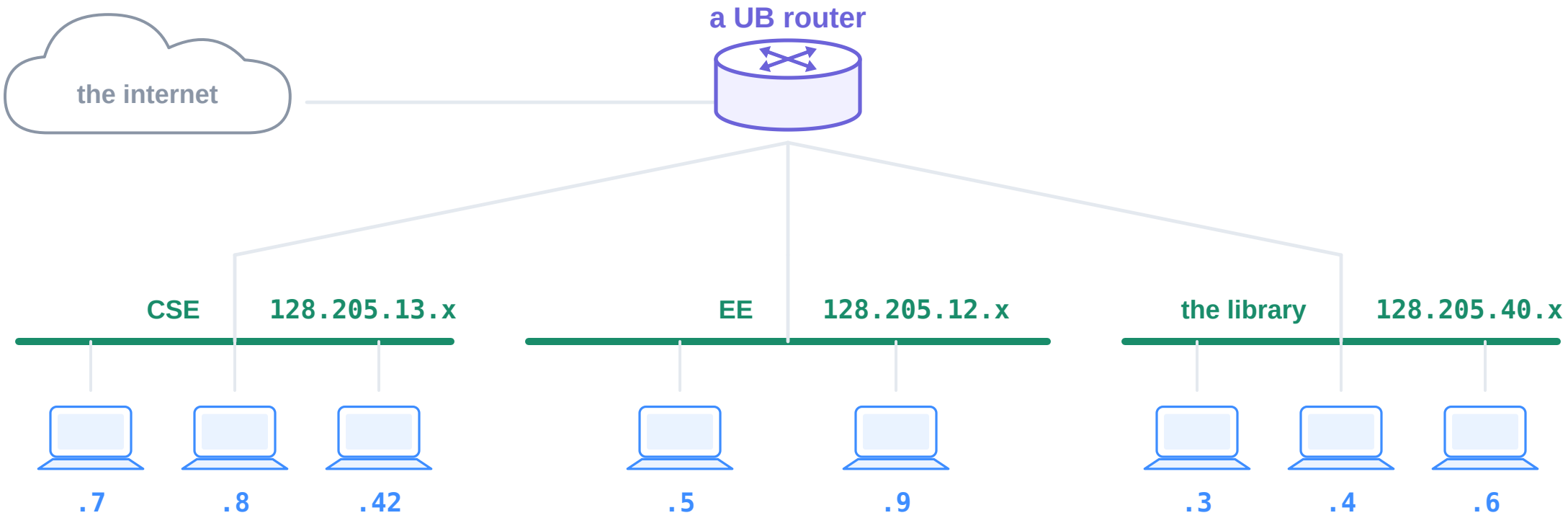


The cuts inside UB are illustrative; the rest is real.

Every address on the internet came down a chain like this. Nobody picks their own.

1 TTL | 2 Checksum | 3 IP allocation | 4 NAT | 5 IPv6

What does UB look like from the inside?

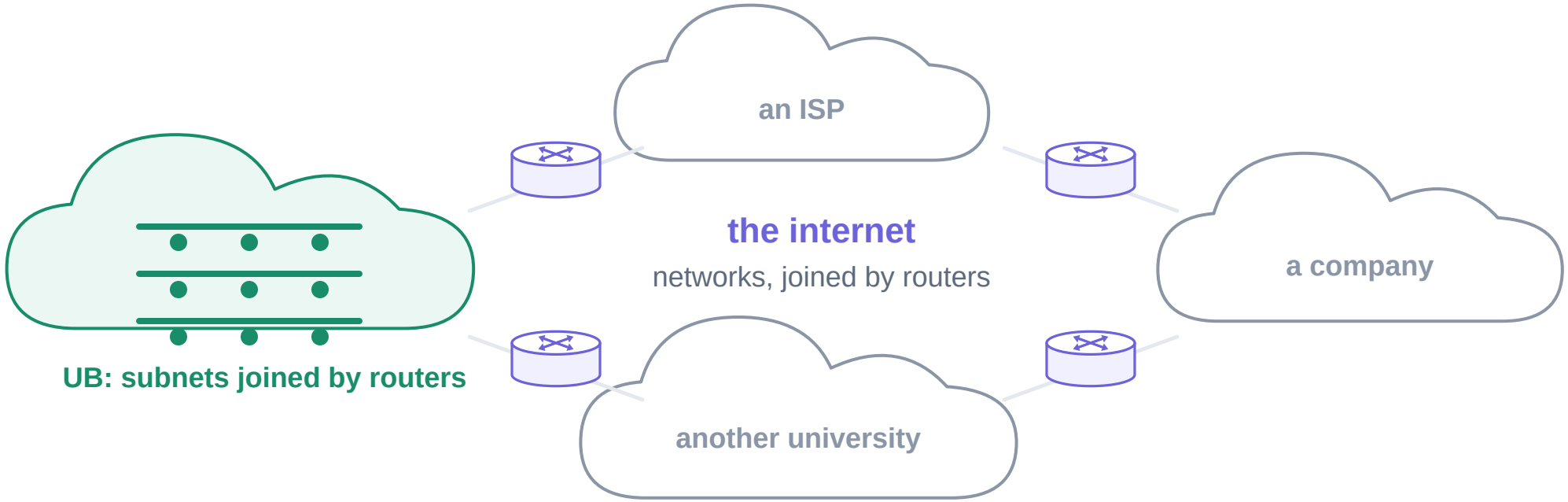


Three of UB's subnets on one of its routers; a real campus has hundreds of each. Addresses are illustrative except CSE's piece.

Many small networks, one per building or lab, joined by routers. Each is a subnet.

1 TTL | 2 Checksum | 3 IP allocation | 4 NAT | 5 IPv6

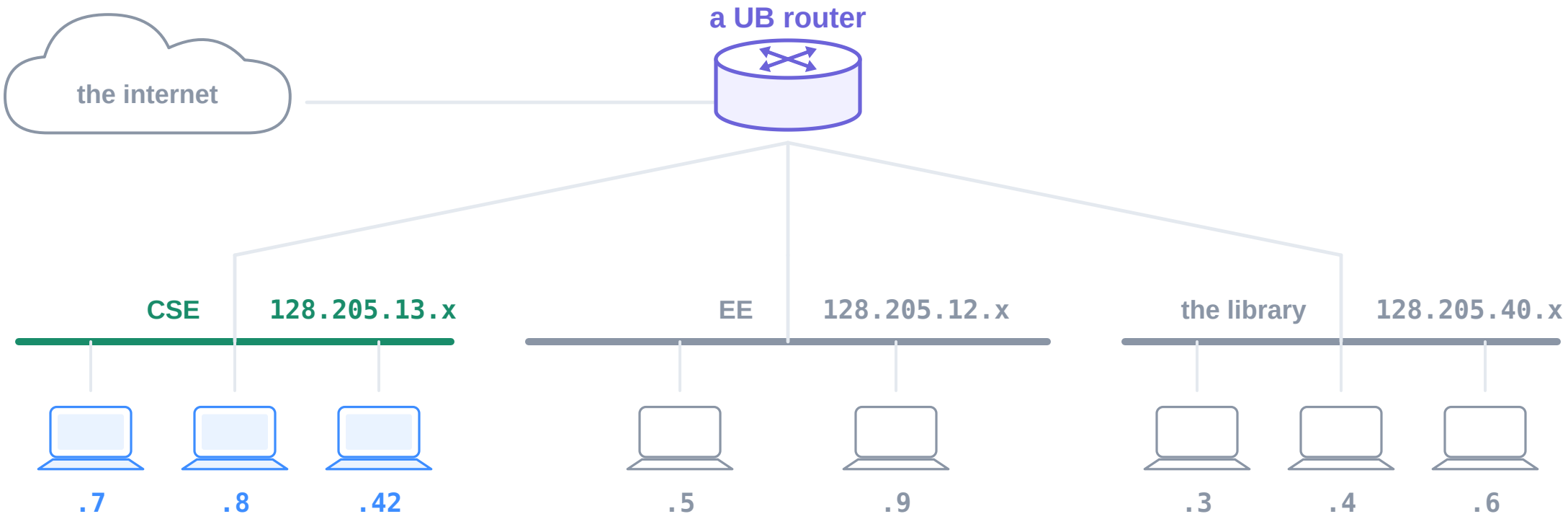
And what is the internet, then?



Routers join subnets into a campus, and campuses, ISPs and companies into the internet.

1 TTL | 2 Checksum | 3 IP allocation | 4 NAT | 5 IPv6

What is a subnet?



Two machines on the same subnet reach each other directly over the link; anything else goes to the router first.

The machines on one link, one router as their way out, every address 128.205.13.x.

1 TTL | 2 Checksum | 3 IP allocation | 4 NAT | 5 IPv6

How does one machine get one address out of its subnet?

by hand

typed in

someone opens the settings
and fills in four fields

servers and routers

DHCP

asked for

the machine asks on its subnet
and is told what to use

laptops, phones, everything else

Almost every machine asks, and a server on the same subnet answers.

1 TTL | 2 Checksum | 3 IP allocation | 4 NAT | 5 IPv6

What does a laptop know when it joins the subnet?

a laptop just plugged into 128.205.13.x: what it has to find out

an address
one out of this subnet
?

my subnet
which addresses are next door
?

the default gateway
the router, my way out
?

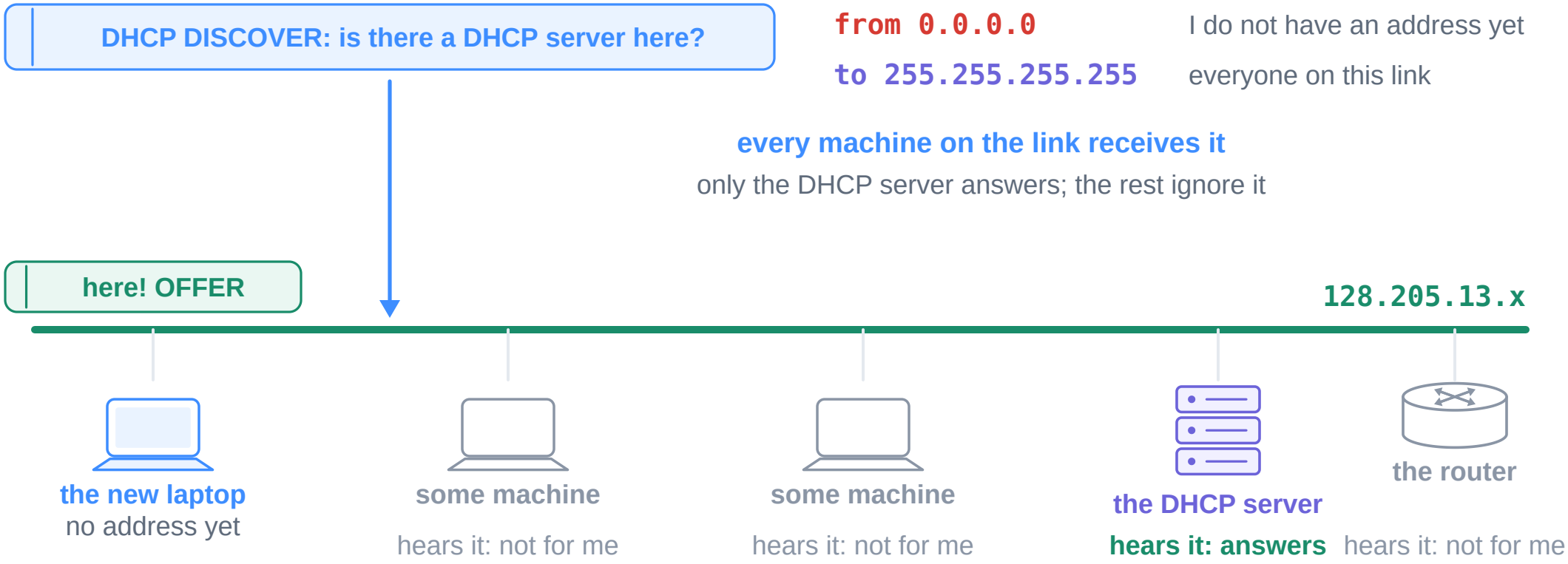
a DNS server
who resolves names
?

It has a link and nothing else.
It cannot even ask a question by name, because it has no DNS server to ask.

Nothing. Four blanks, and asking has to fill all four.

1 TTL | 2 Checksum | 3 IP allocation | 4 NAT | 5 IPv6

How can it ask when it has no address?



It shouts from 0.0.0.0 to everyone on the link; the DHCP server hears it and answers.

1 TTL | 2 Checksum | 3 IP allocation | 4 NAT | 5 IPv6

What does the server answer?

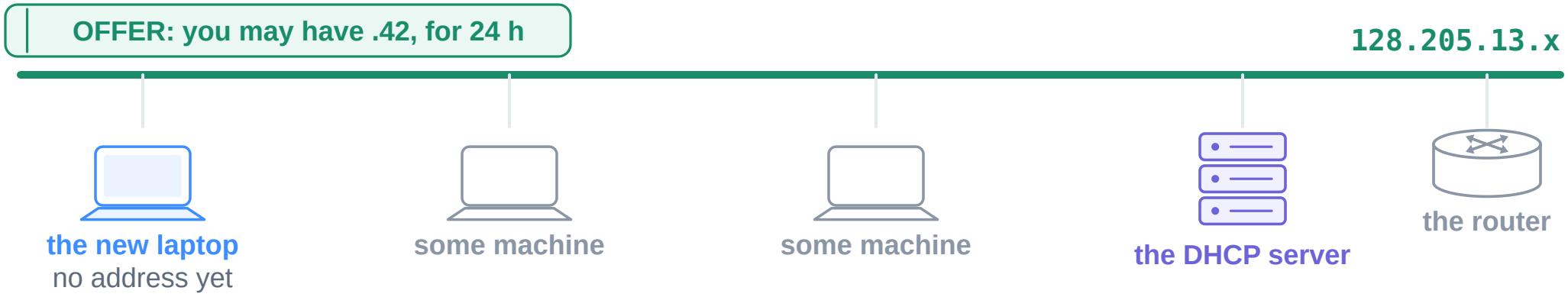
the server heard the shout,
looks in its pool for a free address,
picks .42,
and offers it: for 24 hours

the laptop: offered .42, not mine yet

the server's pool for 128.205.13.x

.2	.3	.4	.5	.6	.7	.8	.9	.40
.41	**.42**	.43	.44	.45	.46	.47	.48	.49

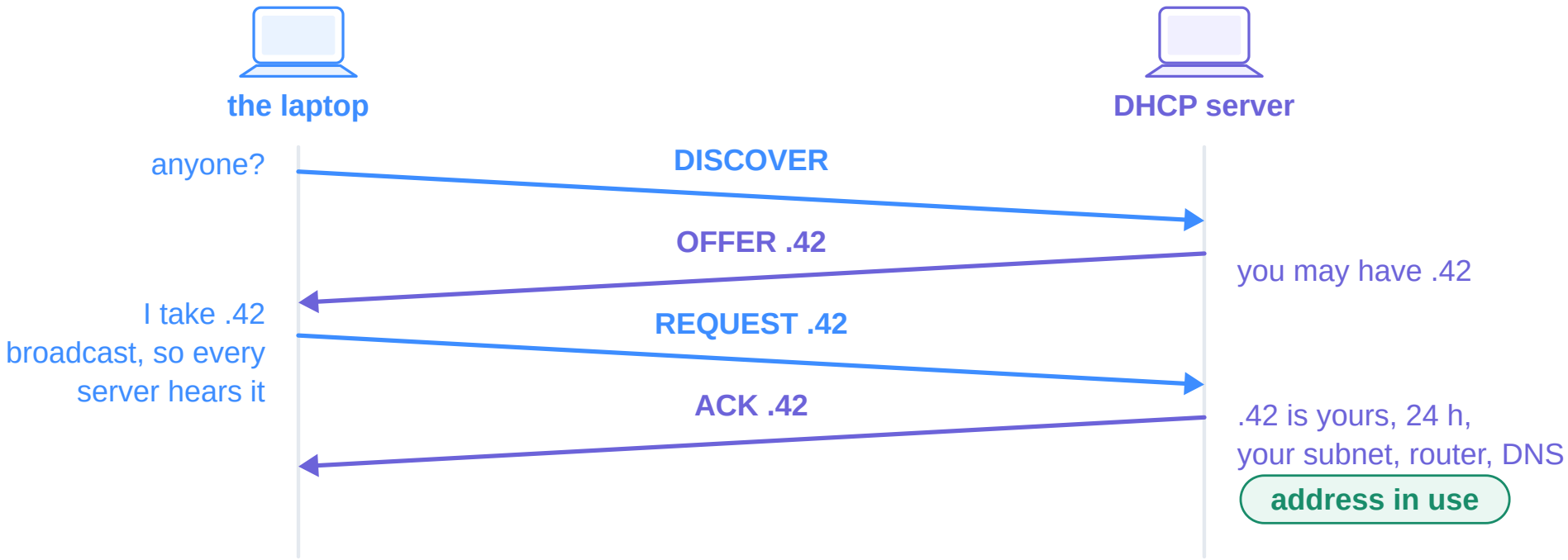
grey: already lent to somebody



OFFER: a free address out of its pool for this subnet, and for how long.

1 TTL | 2 Checksum | 3 IP allocation | 4 NAT | 5 IPv6

Why does the laptop have to answer, and what settles it?



REQUEST is broadcast because a second server may have offered a different address; it hears the request and puts its offer back in the pool.

REQUEST says which offer it takes; ACK confirms it and adds the other three answers.

1 TTL | 2 Checksum | 3 IP allocation | 4 NAT | 5 IPv6

What does the ACK carry?

the four answers in the ACK, for a laptop on 128.205.13.x



your address

`128.205.13.42`

who I am



your subnet

`128.205.13.0 to .255`

who is next door



default gateway

`128.205.13.1`

the router, my way out



DNS server

`128.205.7.4`

who resolves names for me

Four answers in one message. That is why plugging in just works.

The gateway and the DNS server get a page each next.

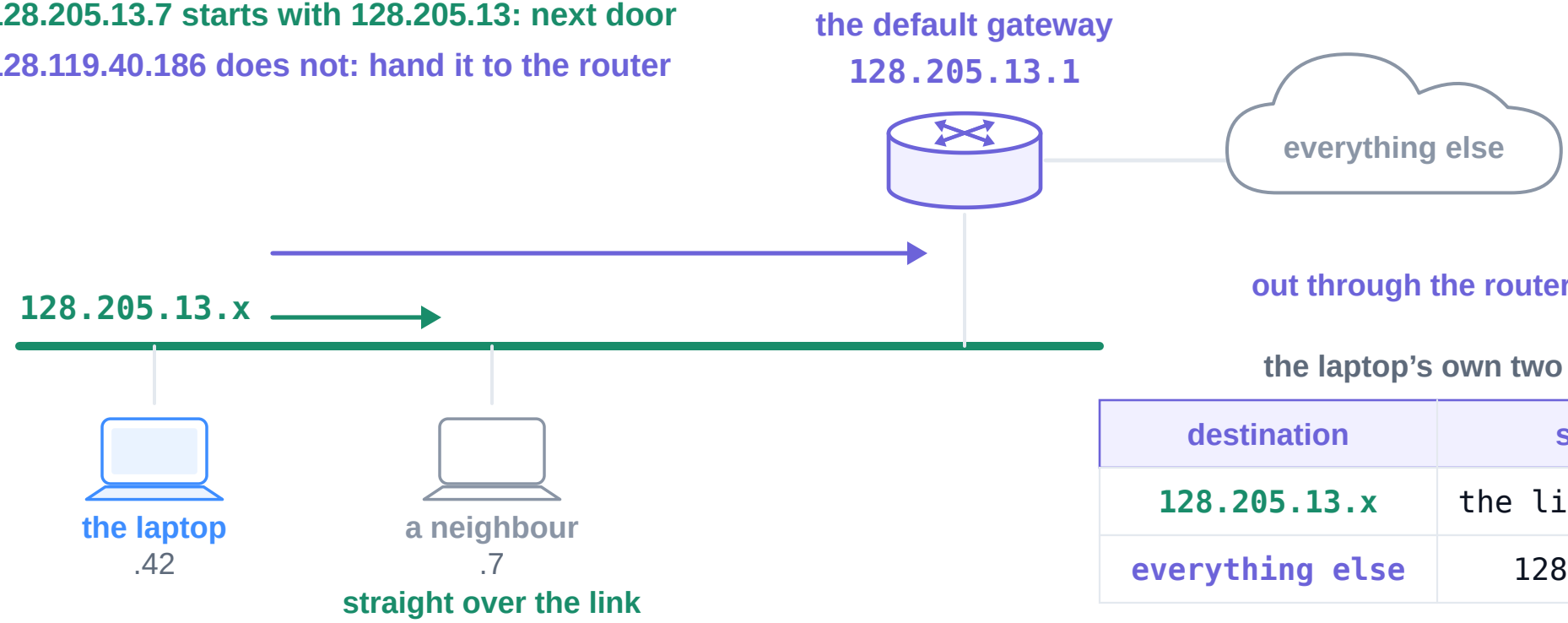
Address, subnet, default gateway and DNS server, all four in the fourth message.

1 TTL | 2 Checksum | 3 IP allocation | 4 NAT | 5 IPv6

The default gateway: the way out of the subnet

128.205.13.7 starts with 128.205.13: next door

128.119.40.186 does not: hand it to the router



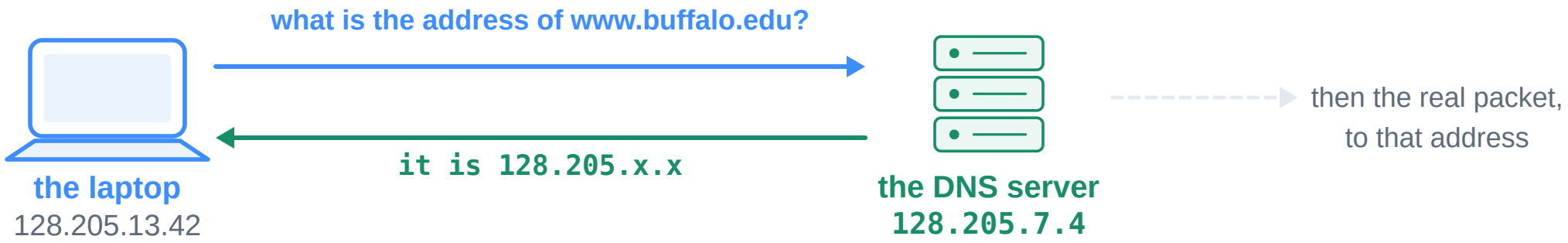
the laptop's own two rules

destination	send it to
128.205.13.x	the link, directly
everything else	128.205.13.1

The router's address on this link. Anything not next door is handed to it first.

1 TTL | 2 Checksum | 3 IP allocation | 4 NAT | 5 IPv6

The DNS server: names into addresses



People type names; packets need addresses.
Every connection starts with this question, so the laptop must know one machine to ask (the DNS lecture).

The DNS server's address here is illustrative; www.buffalo.edu is the real name, and its address is one of UB's.

The machine to ask when a name must become an address. Without it, names do not work.

What breaks if one of the four is missing?

missing	what happens
the address	the laptop cannot send anything at all
the subnet	it cannot tell which addresses are next door
the gateway	it can reach its own subnet and nothing else
the DNS server	addresses work, names do not resolve

Each of the four answers one question, and each has its own failure when it is wrong.

Four different symptoms, one for each answer the ACK carries.

1 TTL | 2 Checksum | 3 IP allocation | 4 NAT | 5 IPv6

Is the address given to the laptop, or lent?

0 h

the ACK

.42 is yours
for 24 hours

→

12 h

still here

the laptop asks again
and gets another 24 h

→

the laptop leaves

no renewal

nobody asks to keep .42;
the clock runs out

→

24 h later

back in the pool

.42 is free again,
for the next laptop

The address is lent, not given.
A machine that leaves stops renewing, and its address goes back into the pool for the next one.

Lent, for a fixed time. Addresses have to come back, because the pool is small.

1 TTL | 2 Checksum | 3 IP allocation | 4 NAT | 5 IPv6

What happens when the addresses run out?

04

NAT

There are 4.29 billion IPv4 addresses and the last block was handed out in 2011. The idea that kept the internet growing: many devices behind one address, with a router translating at the edge.

How many IPv4 addresses are there in total?

4,294,967,296

every address a 32-bit field can hold

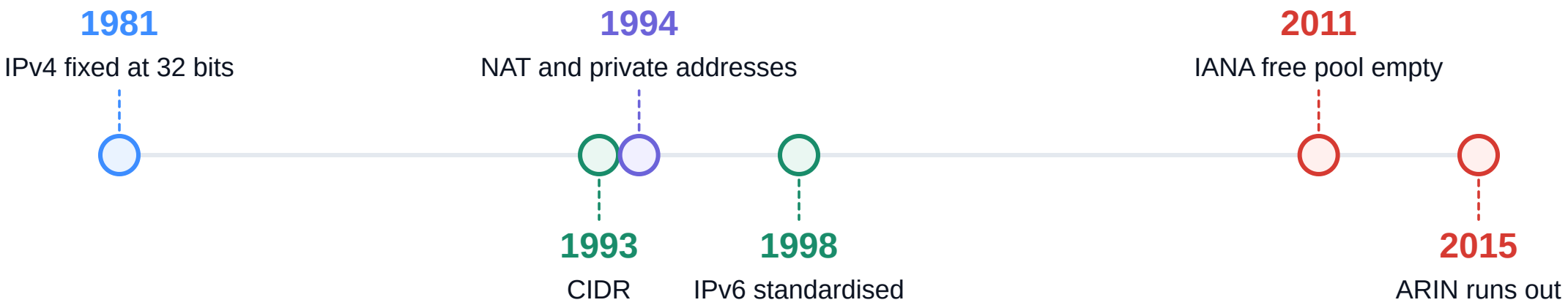
2^{32}

In 1981 that looked like infinity.

4,294,967,296, and there are far more connected devices than that.

1 TTL | 2 Checksum | 3 IP allocation | 4 NAT | 5 IPv6

When did they run out?

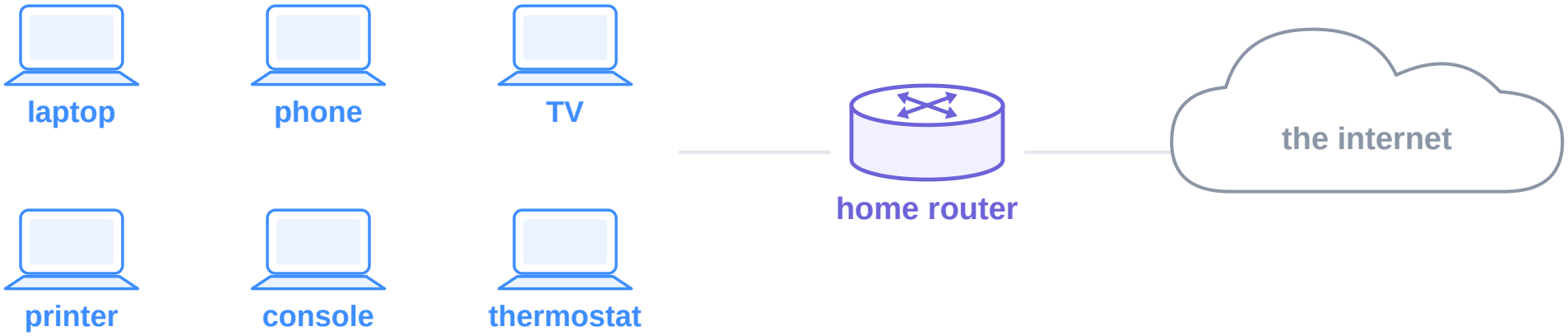


On 3 February 2011 the free pool was empty.
Since then a new block has to be bought from somebody who already has one.

IANA handed out its last blocks in February 2011; the registries followed.

1 TTL | 2 Checksum | 3 IP allocation | 4 NAT | 5 IPv6

How many addresses does a home get?



the ISP gave this house one address

six devices, one address: who gets it?

One, from the ISP’s block. Six devices cannot share one address the way the header works.

1 TTL | 2 Checksum | 3 IP allocation | 4 NAT | 5 IPv6

The idea: two kinds of address

A PUBLIC ADDRESS

`128.205.13.42`

unique in the whole world
handed down the chain from IANA
any router on the internet can reach it

A PRIVATE ADDRESS

`192.168.1.5`

used only inside one network
from a block nobody owns
no router on the internet carries it

the three private blocks; anyone may use them, inside their own network

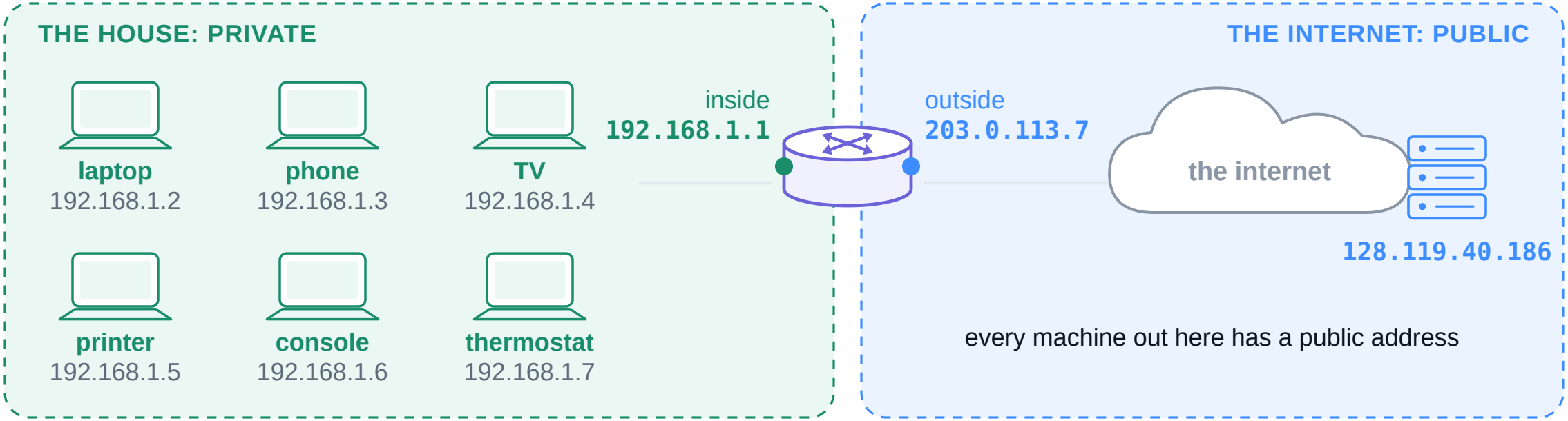
`10.x.x.x` `172.16.x.x to 172.31.x.x` `192.168.x.x`

UB's wireless hands out 10.83.x.x, from the first block: that is where the check page's address came from. Most homes use 192.168.x.x.

Private addresses inside, one per device; one public address for the whole network.

1 TTL | 2 Checksum | 3 IP allocation | 4 NAT | 5 IPv6

Where is which kind of address used?

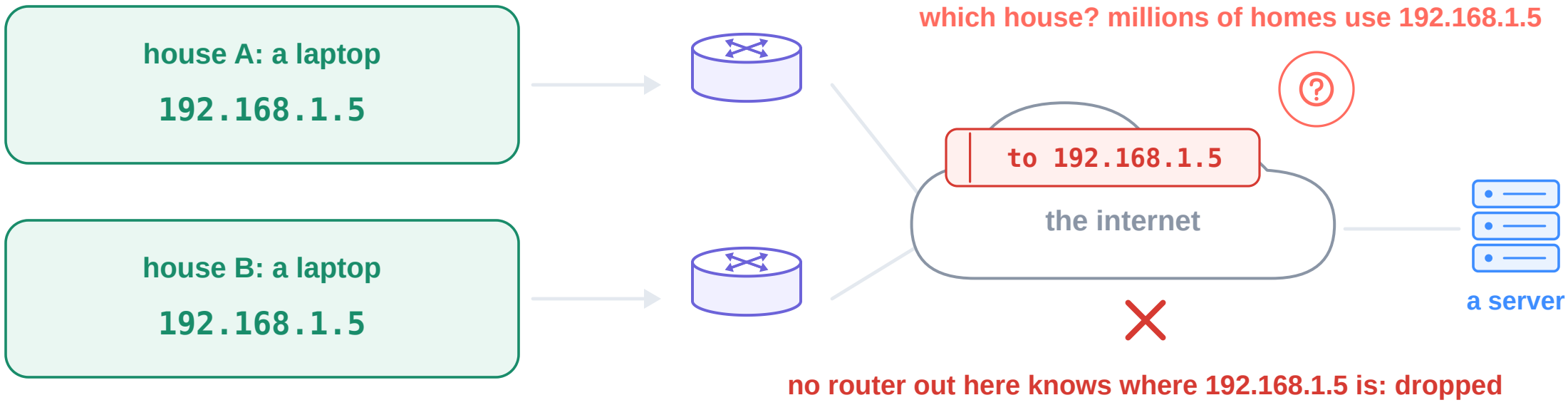


The router is the border: a private address on the house side, the one public address on the other.
Six devices, six private addresses; one public address for the whole house, and it sits on the router.

Private inside, one per device. The one public address sits on the router's outside.

1 TTL | 2 Checksum | 3 IP allocation | 4 NAT | 5 IPv6

Why can a private address not be used out there?

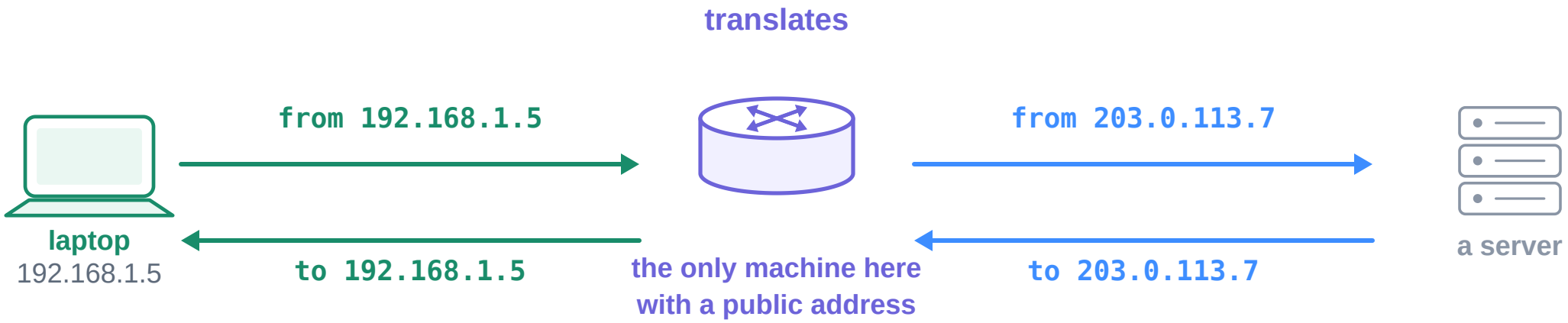


Only a public address is unique, so only a public address can be delivered to.
A private address is in use in millions of homes at once; no router outside can tell which one you mean.

The internet delivers only to public addresses, because only those are unique.

1 TTL | 2 Checksum | 3 IP allocation | 4 NAT | 5 IPv6

So how can the laptop talk to a server at all?

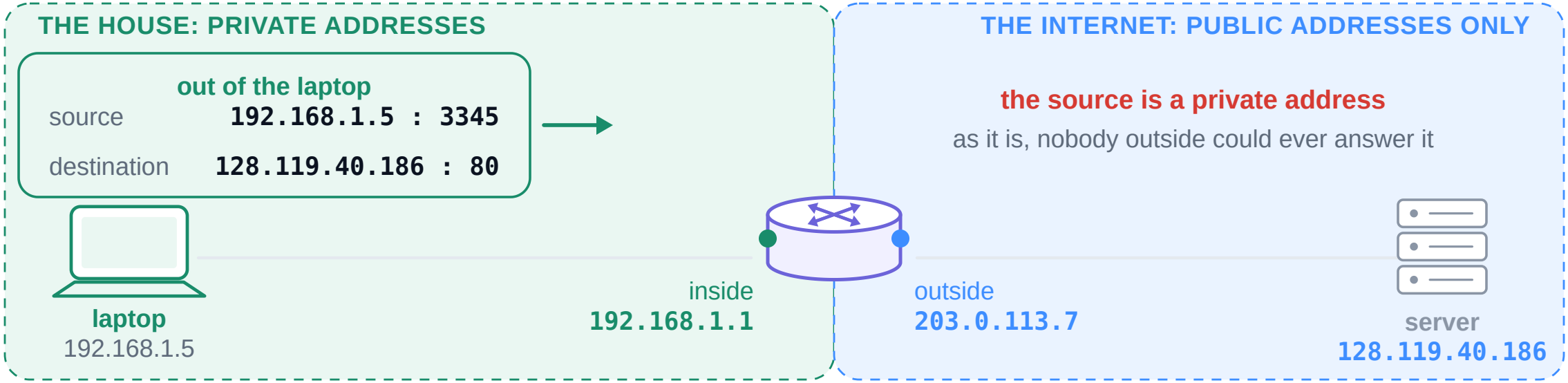


Network Address Translation: the router swaps addresses at the border.
Out: the private source becomes public. Back: the public destination becomes private again.

The router stands in for it: private address out, public address in, swapped at the border.

1 TTL | 2 Checksum | 3 IP allocation | 4 NAT | 5 IPv6

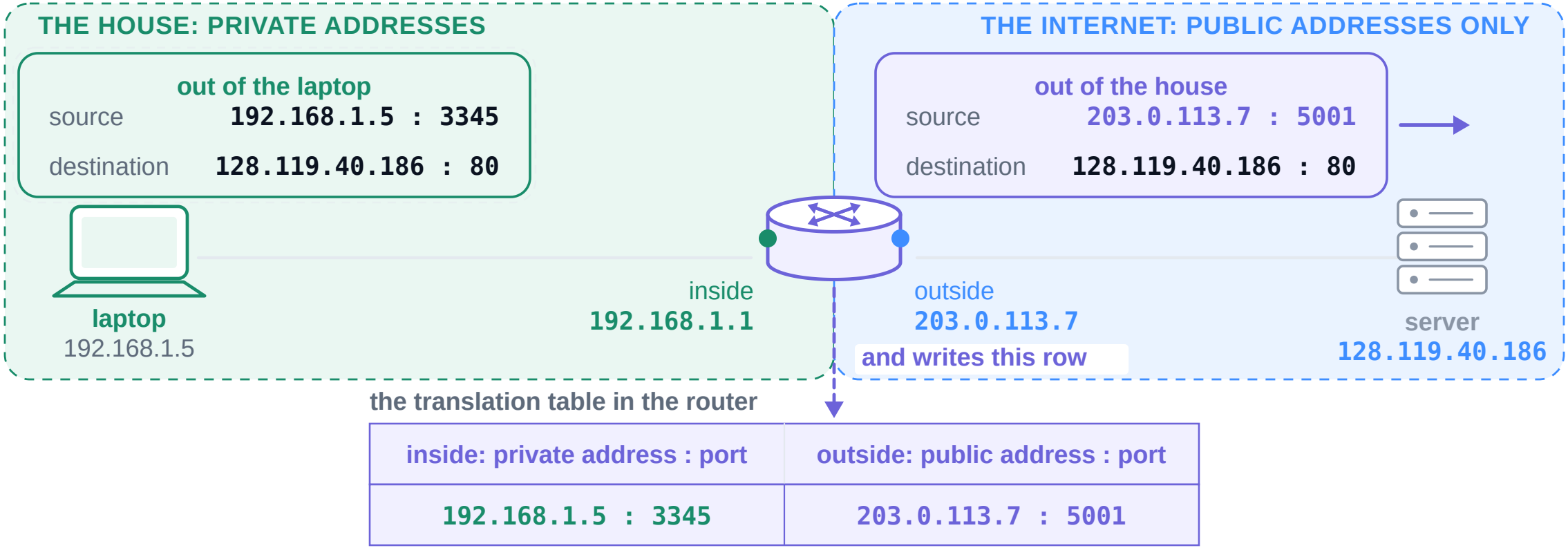
Step 1: the packet leaves the laptop



The source is 192.168.1.5, port 3345. As it is, nobody outside could ever answer it.

1 TTL | 2 Checksum | 3 IP allocation | 4 NAT | 5 IPv6

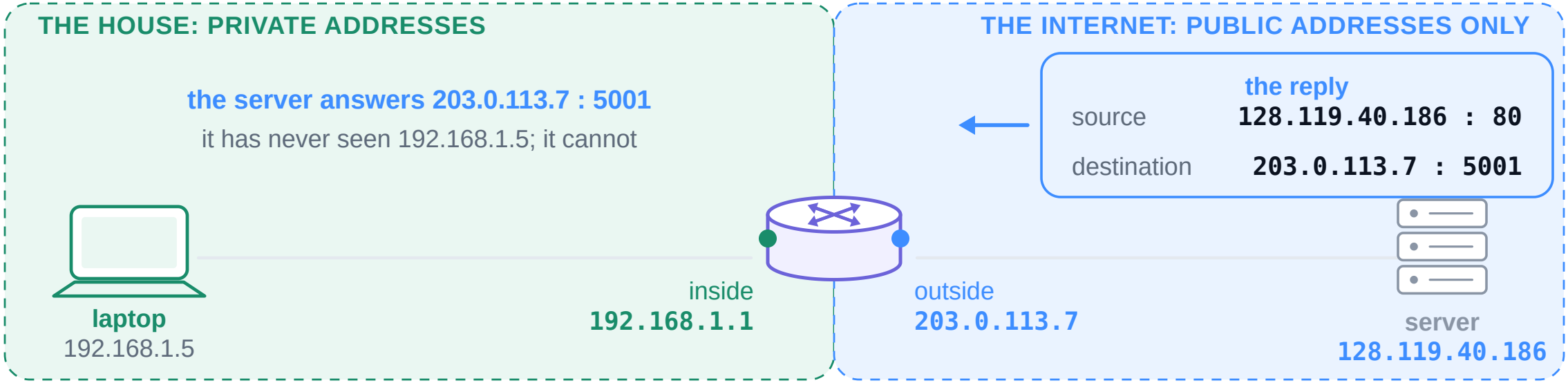
Step 2: the router rewrites the source, and remembers



Source becomes 203.0.113.7, port 5001; the pair it replaced goes into one row of a table.

1 TTL | 2 Checksum | 3 IP allocation | 4 NAT | 5 IPv6

Step 3: the server replies to the public address



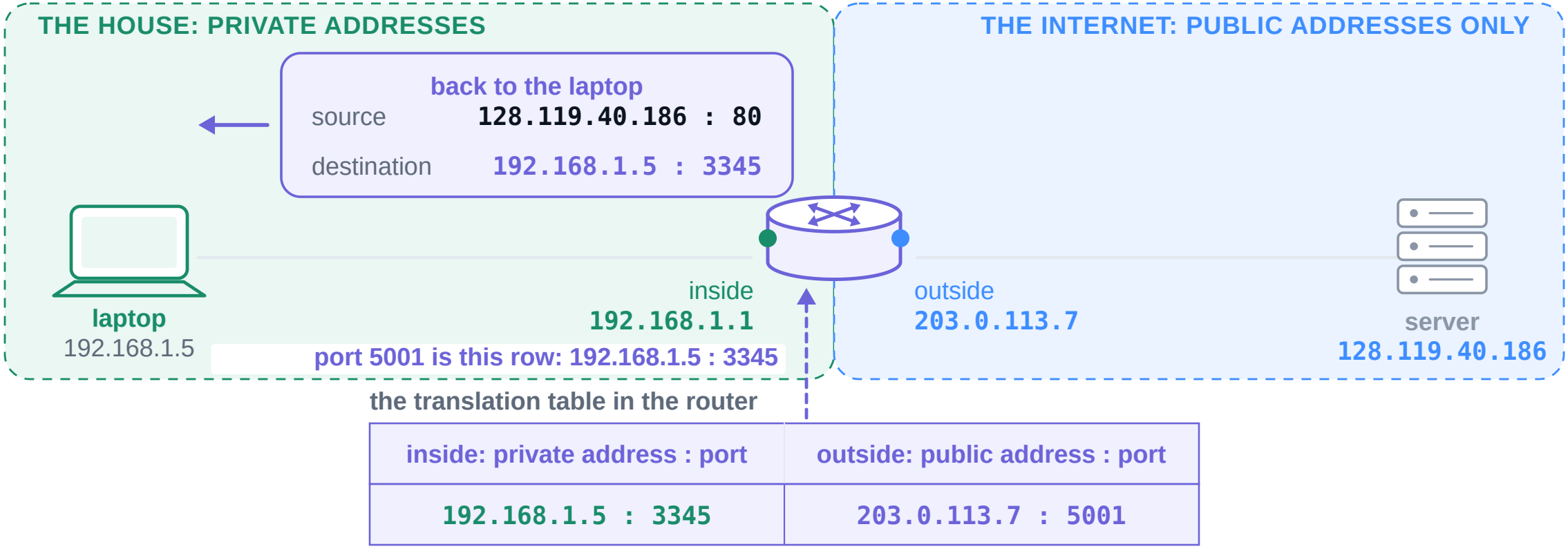
the translation table in the router

inside: private address : port	outside: public address : port
192.168.1.5 : 3345	203.0.113.7 : 5001

The reply is addressed to 203.0.113.7 port 5001: that is the router, not the laptop.

1 TTL | 2 Checksum | 3 IP allocation | 4 NAT | 5 IPv6

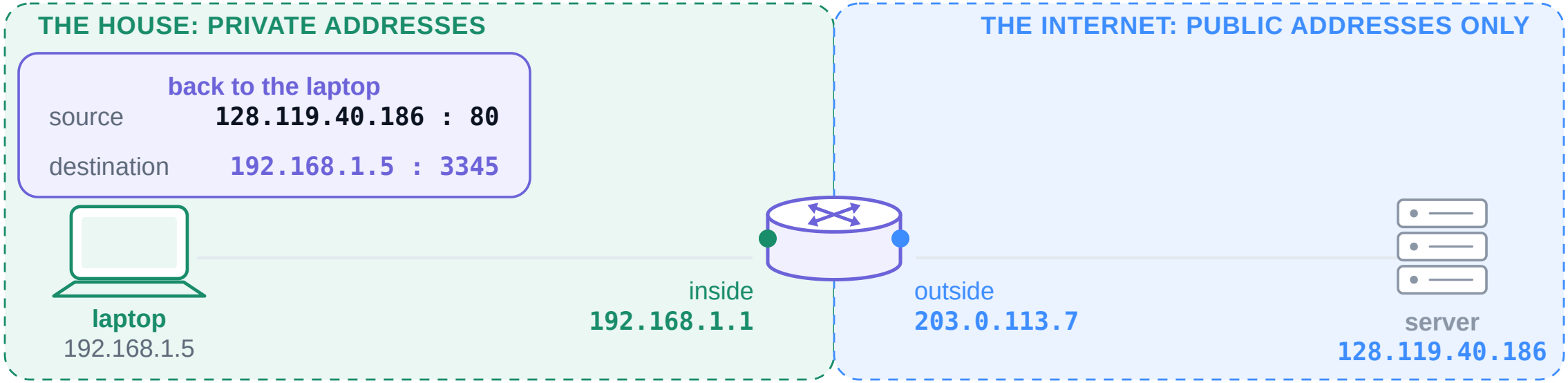
Step 4: the row rewrites the destination



Port 5001 finds the row. Destination becomes 192.168.1.5 : 3345, and the reply arrives.

1 TTL | 2 Checksum | 3 IP allocation | 4 NAT | 5 IPv6

The whole round trip



the translation table in the router

inside: private address : port	outside: public address : port
192.168.1.5 : 3345	203.0.113.7 : 5001

Out: the source is rewritten and a row is written. Back: the row rewrites the destination.

1 TTL | 2 Checksum | 3 IP allocation | 4 NAT | 5 IPv6

Who writes the table?



written

by the router itself, the moment a packet leaves the house



used

by every reply that comes back to that outside port



forgotten

after a while with no packets on that row

Nobody types the table. One row per conversation, made from the packets going out.
That is also why the port is in the row: the address alone would be the same for the whole house.

The router, from what goes out: one row per conversation, forgotten when it goes quiet.

1 TTL | 2 Checksum | 3 IP allocation | 4 NAT | 5 IPv6

Two devices pick the same port. Which reply is whose?

the laptop

source	192.168.1.5 : 3345
destination	128.119.40.186 : 80

the phone

source	192.168.1.9 : 3345
destination	128.119.40.186 : 80

both leave the house as 203.0.113.7

and both chose port 3345, which they may

the router gives each conversation its own outside port

inside	outside
192.168.1.5 : 3345	203.0.113.7 : 5001
192.168.1.9 : 3345	203.0.113.7 : 5002

a reply to port 5001 is the laptop's; to 5002, the phone's

the address says which house; the port says which row

Outside ports are never reused while in the table, so the port alone picks the row.

1 TTL | 2 Checksum | 3 IP allocation | 4 NAT | 5 IPv6

How much does that save?

6 → 1

a home

six devices,
one public address

1000s → few

UB wireless

every laptop is 10.x;
a few public addresses

> 2^{32}

the internet

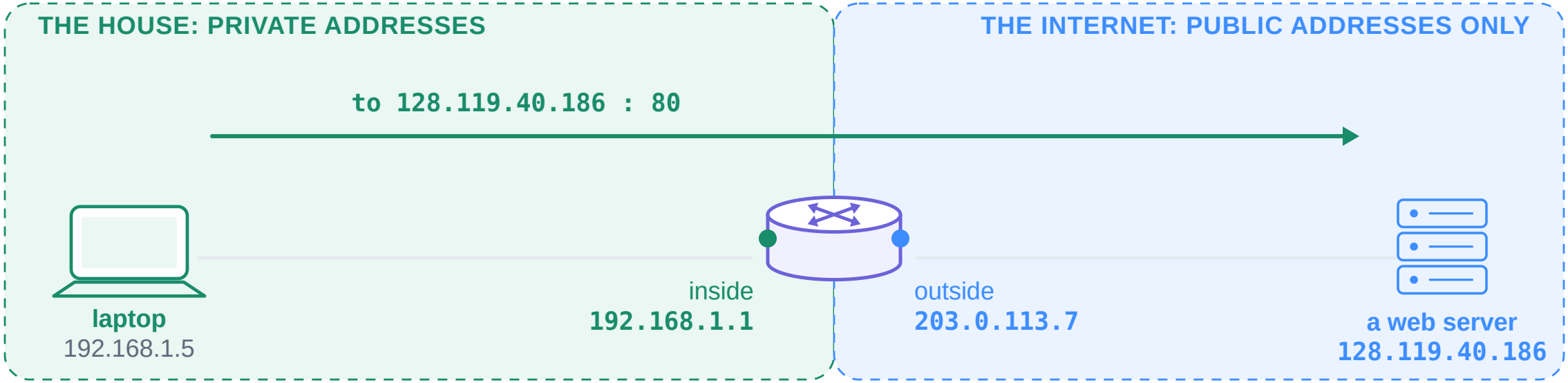
more devices than addresses,
and it still works

This is how the internet kept growing after 2011.
One box at the edge, installed by whoever owns the network; nobody else has to agree, upgrade, or even know.

A whole network behind one address. Almost every device you own is behind a NAT.

1 TTL | 2 Checksum | 3 IP allocation | 4 NAT | 5 IPv6

Why did every connection so far just work?



1 **we know its address**
✓ before we start

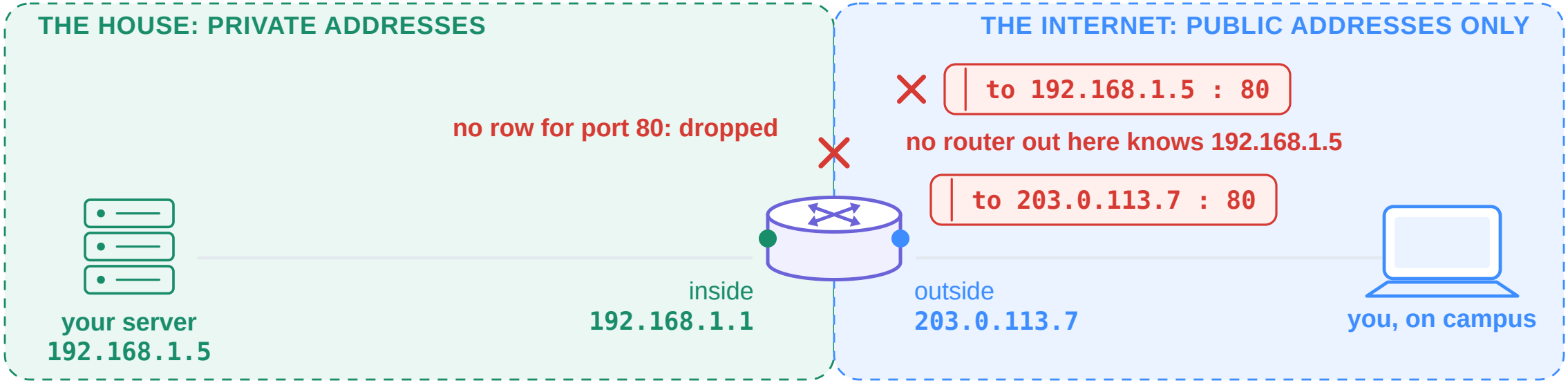
2 **that address is public**
✓ any router can deliver to it

3 **we start the conversation**
✓ so our router writes the row

The server's address was known and public, and we started, so our router had the row.

1 TTL | 2 Checksum | 3 IP allocation | 4 NAT | 5 IPv6

What if the server is the one at home?



1 you know its address
✓ 192.168.1.5

2 but it is private
✗ routers out here drop it

3 and you start from outside
✗ so there is no row for you

Its own address is private, and the house's public address has no row: both dropped.

1 TTL | 2 Checksum | 3 IP allocation | 4 NAT | 5 IPv6

Where does this hurt in real life?



A server at home

your own web site,
your photos, your files



A video call

straight from
one home to another



Sending a file

straight to a
friend's laptop



A game

hosted on
your own console

Each one needs someone outside to start talking to a device behind a NAT.
A device behind NAT can call out, but it cannot be called.

A device behind NAT can call out but cannot be called: NAT's biggest pain point.

1 TTL | 2 Checksum | 3 IP allocation | 4 NAT | 5 IPv6

NAT traversal: how do we get through anyway?

1



Port forwarding

send to the router's public address and port; a row written by hand sends it in

2



Relay

both sides call out to a machine with a public address; it passes packets on

3



Hole punching

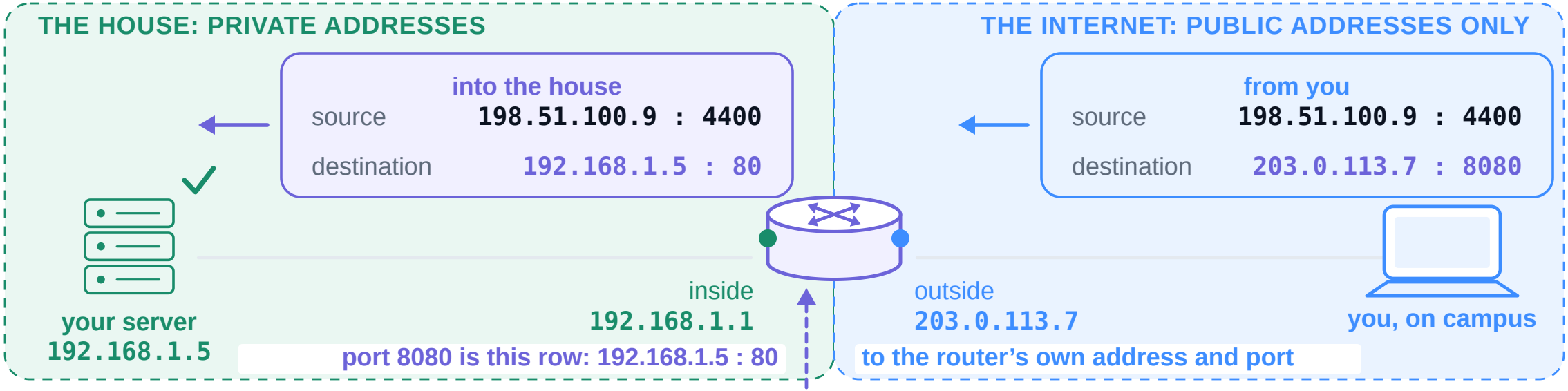
both sides call each other at the same moment, so both routers have a row

All three get around one rule: a router lets a packet in only if a row says where it goes.

Write the row by hand, meet at a relay, or have both sides punch a hole at once.

1 TTL | 2 Checksum | 3 IP allocation | 4 NAT | 5 IPv6

Port forwarding: what if the row is written in advance?



the translation table in the router

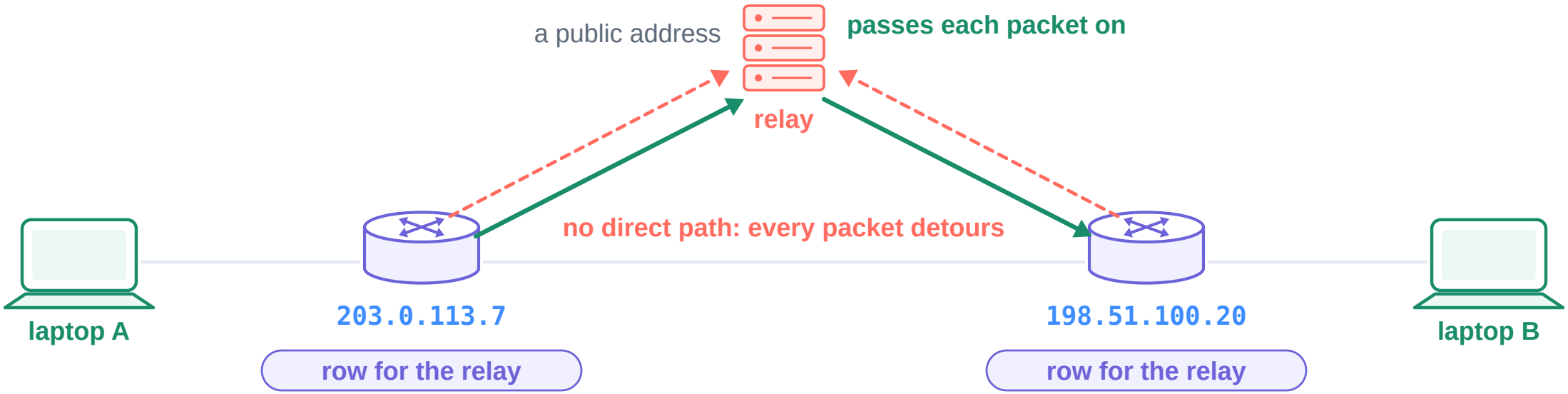
inside: private address : port	outside: public address : port
192.168.1.5 : 80	203.0.113.7 : 8080

typed in by hand, once, in its settings

We send to the router's address and port; a row typed in by hand says where inside.

1 TTL | 2 Checksum | 3 IP allocation | 4 NAT | 5 IPv6

Relay: what if both sides call a third machine?

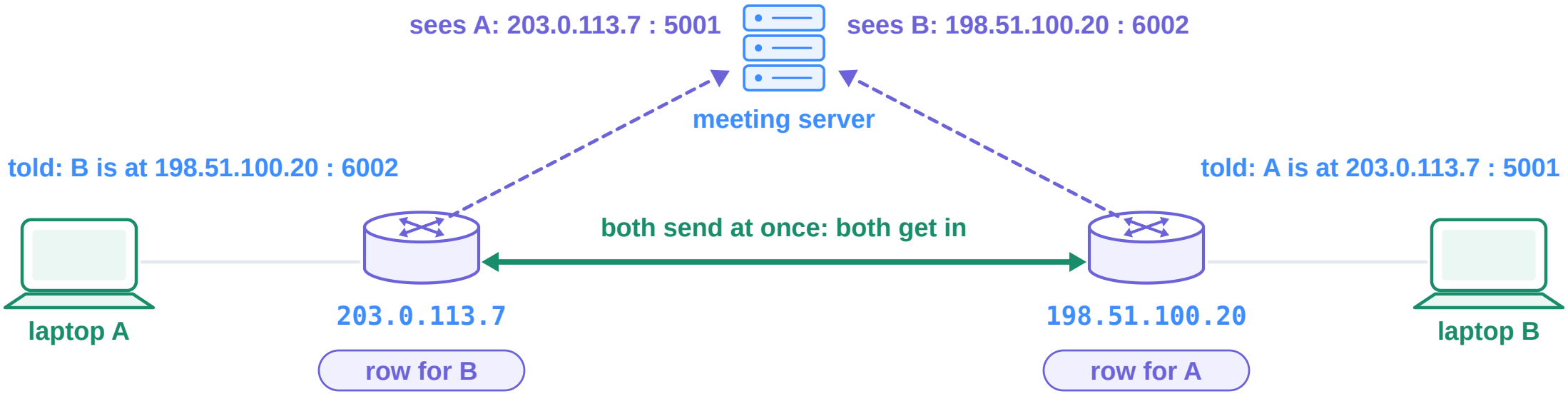


Step 3: it always works, at a price.
Every packet takes the detour, and somebody pays for the relay's bandwidth.

Each side calls out to a machine with a public address, and it passes the packets on.

1 TTL | 2 Checksum | 3 IP allocation | 4 NAT | 5 IPv6

Hole punching: what if both sides call at the same time?



Step 3: both send to each other at the same moment.
Each send writes a row in its own router, so the other side's packet finds one and gets in.

Each side sends first, so each router already has a row when the other's packet arrives.

1 TTL | 2 Checksum | 3 IP allocation | 4 NAT | 5 IPv6

What did NAT give up to make this work?

an address is globally unique	✕	**the same address is in use in millions of homes**
a router reads only its own layer	✕	**it reads and writes transport ports**
anyone can start talking to anyone	✕	**only the inside can start; nobody can call in**

NAT bought time by giving up three properties the design assumed.

NAT is a way to keep going, not a fix. Part 05 is the fix.

1 TTL · 2 Checksum · 3 IP allocation · 4 NAT · 5 IPv6

What is the fix, rather than the workaround?

05

IPv6

A longer address: so many that running out is not on the list of problems. Every device public again, no NAT. Then why, thirty years on, it is still only half of the internet.

What is the real fix?

340,282,366,920,938,463,463,374,607,431,768,211,456

IPv6 addresses: 2 multiplied by itself 128 times

IPv4 had 4,294,967,296 (2 multiplied by itself 32 times)

the address goes from 32 bits to 128 bits

Make the address 128 bits long. That is not four times as many addresses; it is this many.

1 TTL | 2 Checksum | 3 IP allocation | 4 NAT | 5 IPv6

How many is that, really?

share the 340-undecillion addresses out, one equal part to every …

… grain of sand on Earth
about 7.5 quintillion grains
and each one gets
45,000,000,000,000,000,000

… person alive
8.2 billion people
and each one gets
41,000,000,000,000,000,000,000,000,000

… square metre of the planet
510 trillion of them, oceans included
and each one gets
670,000,000,000,000,000,000,000

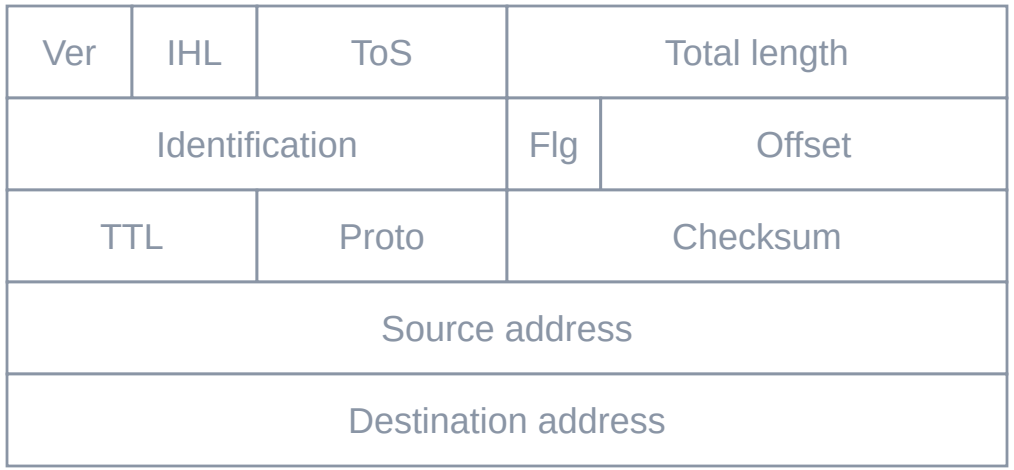
IPv4 ran out at 4.3 billion. IPv6 has 79 billion billion billion times as many.

Give every grain of sand on Earth its own share, and each grain still gets 45 billion billion.

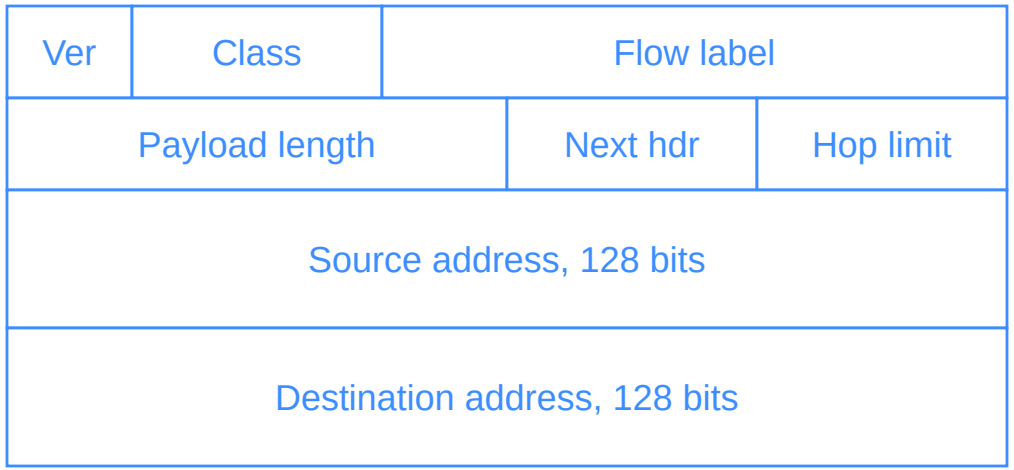
1 TTL | 2 Checksum | 3 IP allocation | 4 NAT | 5 IPv6

What does the IPv6 header look like?

IPv4: 20 bytes, 13 fields



IPv6: 40 bytes, 8 fields



Two much longer addresses, and less of everything else.

The two addresses are 32 of the 40 bytes. The checksum of Part 02 is gone, and so is anything else a router had to rewrite.

Two addresses four times as long, fewer fields around them. That is all we need of it today.

1 TTL | 2 Checksum | 3 IP allocation | 4 NAT | 5 IPv6

Why is this the real fix, when NAT only patches?

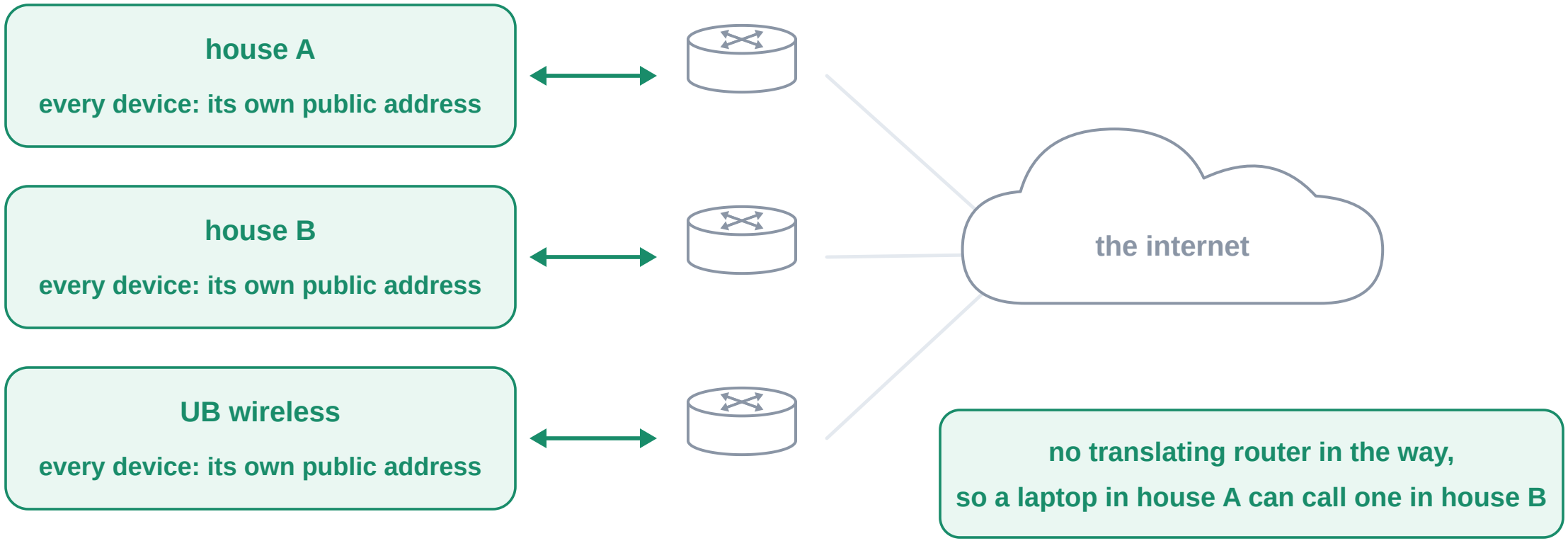
NAT: the patch	IPv6: the fix
✗ one public address, shared by the whole house	✓ every device has a public address of its own
✗ a translation table in the router	✓ no table, nothing rewritten
✗ ports doing the addressing work	✓ the address does its own job
✗ nothing outside can call in	✓ either side can call the other

With enough addresses, the internet works the way it was designed to.

Every device public again: no sharing, no table, no port tricks; anyone can call anyone.

1 TTL | 2 Checksum | 3 IP allocation | 4 NAT | 5 IPv6

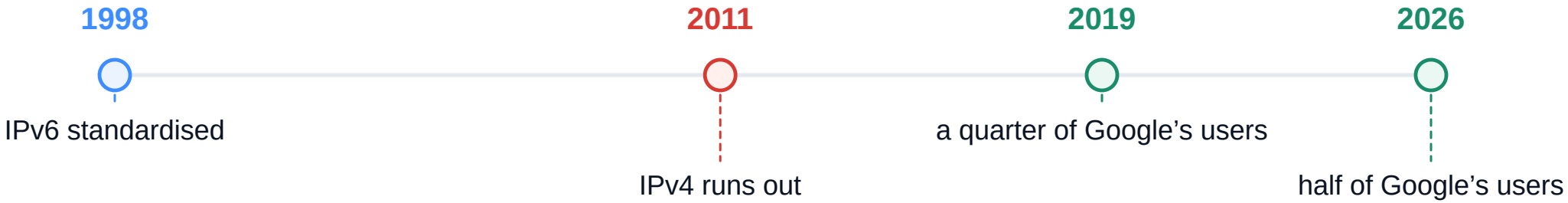
Does IPv6 still need NAT?



No. Every device gets a public address, and the three things NAT gave up come back.

1 TTL | 2 Checksum | 3 IP allocation | 4 NAT | 5 IPv6

So why is it not everywhere already?



Twenty-eight years to reach half. The other half is still IPv4 and NAT.

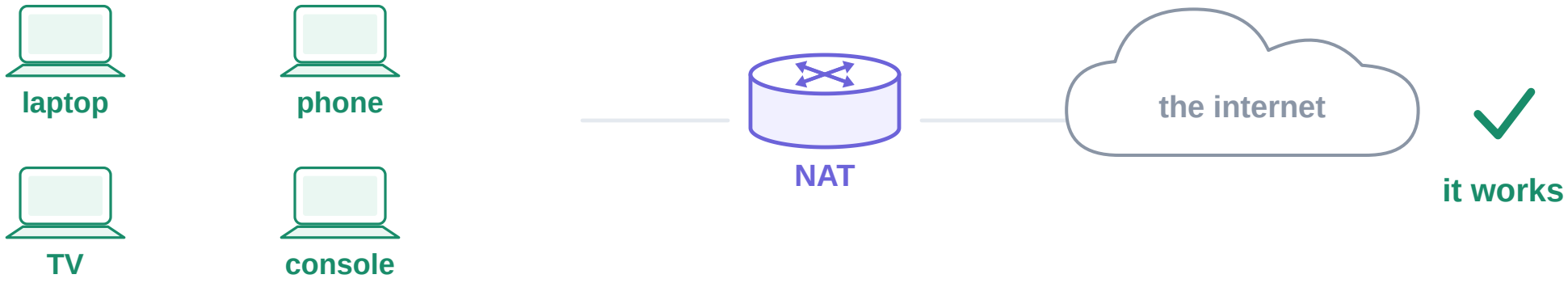
Google measures how many of its users arrive over IPv6: 50.1 % on 28 March 2026, the first time it passed half.

Google’s IPv6 statistics page; the average over several measurements is lower, about 43 % (Internet Society Pulse, April 2026).

Standardised in 1998; half of Google’s users arrived over IPv6 for the first time in 2026.

1 TTL | 2 Checksum | 3 IP allocation | 4 NAT | 5 IPv6

Why so slow, one: nothing is broken



“Why would I pay to change something that works?”

NAT took away the pain, so it took away the hurry.
The addresses ran out in 2011 and almost nobody noticed. That is the patch working against the fix.

With NAT, IPv4 is good enough for almost everything, so nobody is in a hurry.

1 TTL | 2 Checksum | 3 IP allocation | 4 NAT | 5 IPv6

Why so slow, two: the chicken and the egg

THE WEBSITES

“our visitors come over IPv4, so why add IPv6?”

THE NETWORKS

“the sites are on IPv4, so why give users IPv6?”

WHO PAYS

whoever upgrades: new gear, new software

WHO BENEFITS

everyone else, later

Each side waits for the other; whoever moves first pays, and someone else gains.

1 TTL | 2 Checksum | 3 IP allocation | 4 NAT | 5 IPv6

Why so slow, three: IP is the one thing everybody shares



The hourglass: many applications, many links, one protocol in the middle. That is what made IP win, and what makes IPv6 slow.

Everything above and below runs over IP, so changing IP means changing everything.

1 TTL | 2 Checksum | 3 IP allocation | 4 NAT | 5 IPv6

How do the two live side by side?

your phone, today

an IPv4 address

an IPv6 address

IPv6 if the other side has it; IPv4 and NAT if not

Both at once, on every device, and you never notice which one you used.
That is why the switch can take thirty years without anything breaking.

Every device speaks both. Nothing has to switch on one day, and nothing ever forces it.

1 TTL | 2 Checksum | 3 IP allocation | 4 NAT | 5 IPv6

Five topics, five mechanisms

01

TTL

a counter, so loops end; traceroute

02

Checksum

the header, checked at every hop

03

IP allocation

a chain of cuts, then DHCP

04

NAT

private inside, one address out

05

IPv6

a longer address, and why so slow

Two header fields a router rewrites, then the address: handed out, run out, and the fix.

1 TTL | 2 Checksum | 3 IP allocation | 4 NAT | 5 IPv6