Communication Networks

Spring 2020—Online Edition

Laurent Vanbever

nsg.ee.ethz.ch

ETH Zürich (D-ITET)

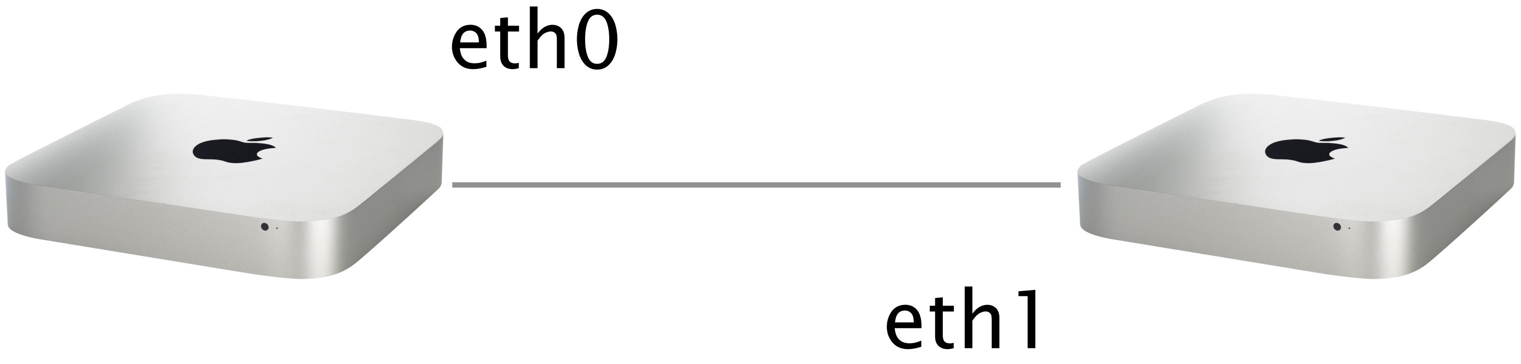
March 16 2020

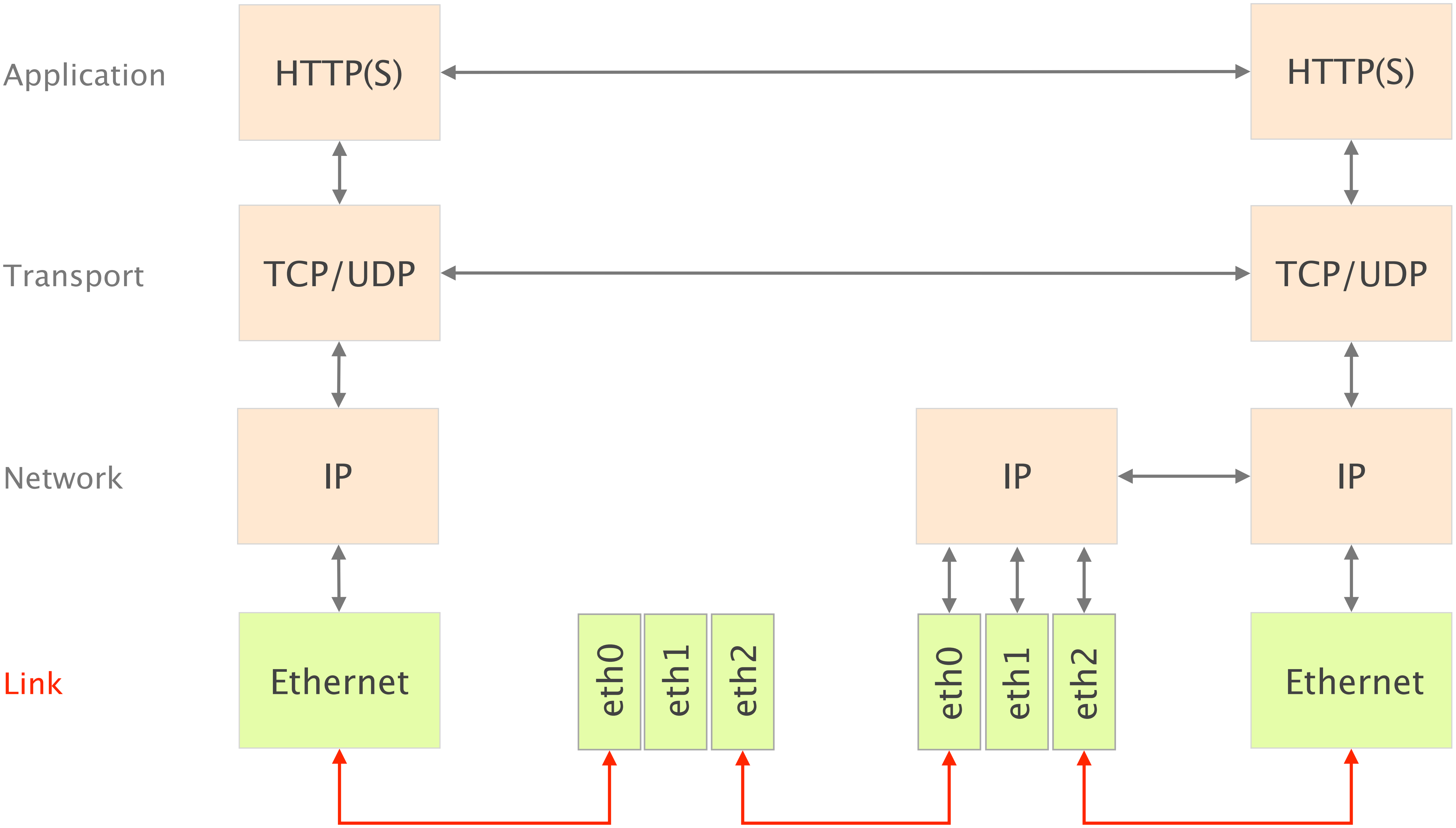
Materials inspired from Scott Shenker & Jennifer Rexford

Last week on

Communication Networks

How do local computers communicate?




Application
HTTP(S)
HTTP(S)
Transport
TCP/UDP
TCP/UDP
Network
IP
IP
IP
Link
Ethernet
eth0
eth1
eth2
eth0
eth1
eth2
Ethernet

Communication Networks

Part 2: The Link Layer

#1 What is a link?

#2 How do we identify link adapters?

#3 How do we share a network medium?

#4 What is Ethernet?

#5 How do we interconnect segments at the link layer?

Communication Networks

Part 2: The Link Layer

#1 What is a link?

How do we identify link adapters?

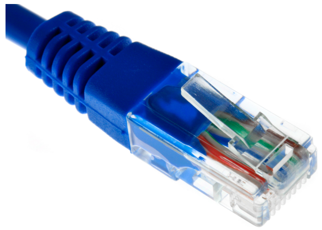
How do we share a network medium?

What is Ethernet?

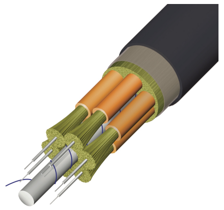
How do we interconnect segments at the link layer?

Link Communication medium and Network adapter

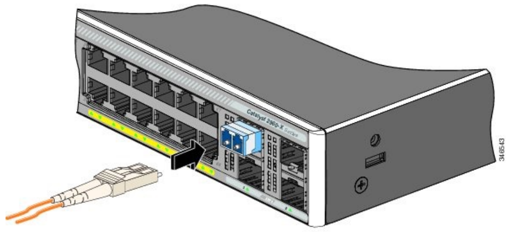
Wifi



Ethernet



Fiber



Communication Networks

Part 2: The Link Layer

What is a link?

#2 How do we identify link adapters?

How do we share a network medium?

What is Ethernet?

How do we interconnect segments at the link layer?

MAC addresses…

identify the sender & receiver adapters

used within a link

are uniquely assigned

hard-coded into the adapter when built

use a flat space of 48 bits

allocated hierarchically

Why don’t we simply use IP addresses?

Links can support any protocol (not just IP)

different addresses on different kind of links

Adapters may move to different locations

cannot assign static IP address, it has to change

Adapters must be identified during bootstrap

need to talk to an adapter to give it an IP address

You need to solve two problems when you bootstrap an adapter

Who am I?

MAC-to-IP binding

How do I acquire an IP address?

Who are you?

IP-to-MAC binding

Given an IP address reachable on a link,

How do I find out what MAC to use?

Who am I?

MAC-to-IP binding

How do I acquire an IP address?

Dynamic Host Configuration Protocol

Who are you?

IP-to-MAC binding

Given an IP address reachable on a link,

How do I find out what MAC to use?

Address Resolution Protocol

This week on

Communication Networks

Link Layer

Network Layer

See last week slides, starting from "How do we identify link adapters"

page 5/13 in the handout

The Beginning

The Local Area Networks we have considered so far define single broadcast domains

If one user broadcast a frame, every other user receives it

As the network scales,
network operators like to segment their LANs

Why?

Improves security

smaller attack surface (visibility & injection)

Improves performance

limit the overhead of broadcast traffic (e.g. ARP)

Improves logistics

separates traffic by role (e.g. staff, students, visitors)

Organizational changes are too frequent to segment networks purely physically—rewiring is a major pain

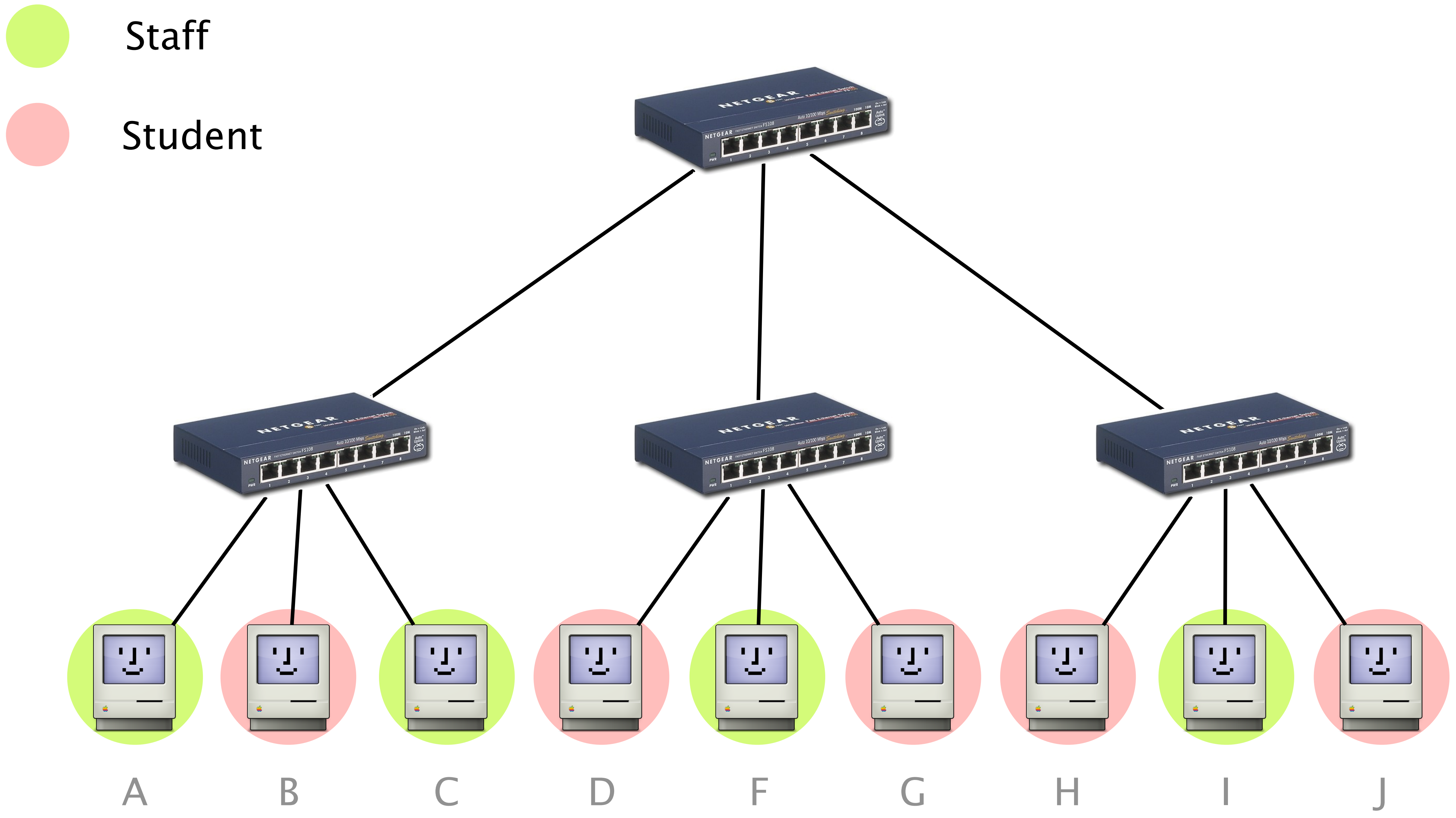
What about doing this in software though?

Enters “Virtual Local Area Networks” (VLANs)

Definition

A VLAN logically identifies
a set of ports attached to
one (or more) Ethernet switches,
forming one broadcast domain

A VLAN identifies a set of ports attached to one or more Ethernet switches

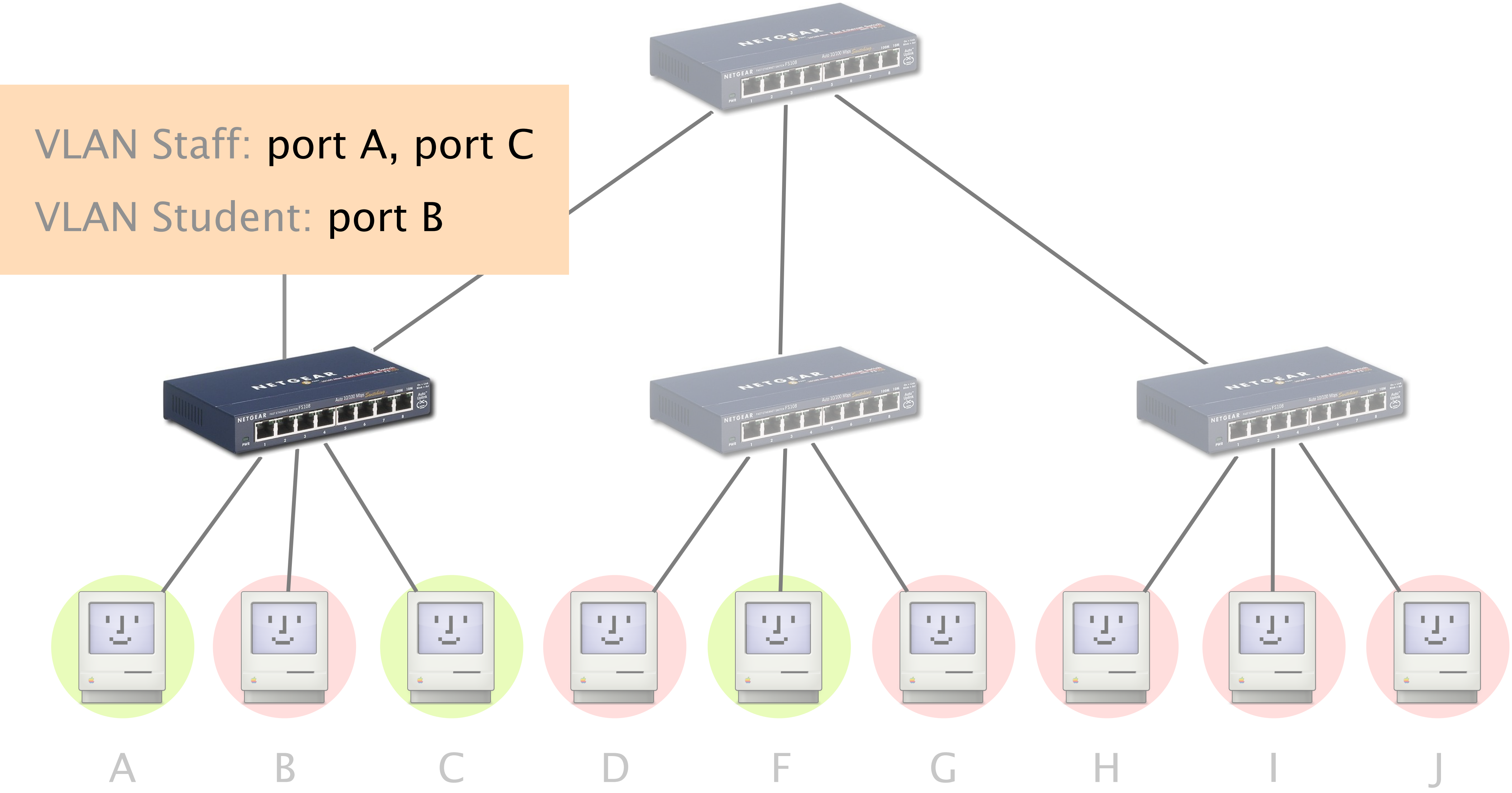


🟢 Staff

🔴 Student

Switches need configuration tables telling them which VLANs are accessible via which interfaces

Switches need configuration tables telling them which VLANs are accessible via which interfaces

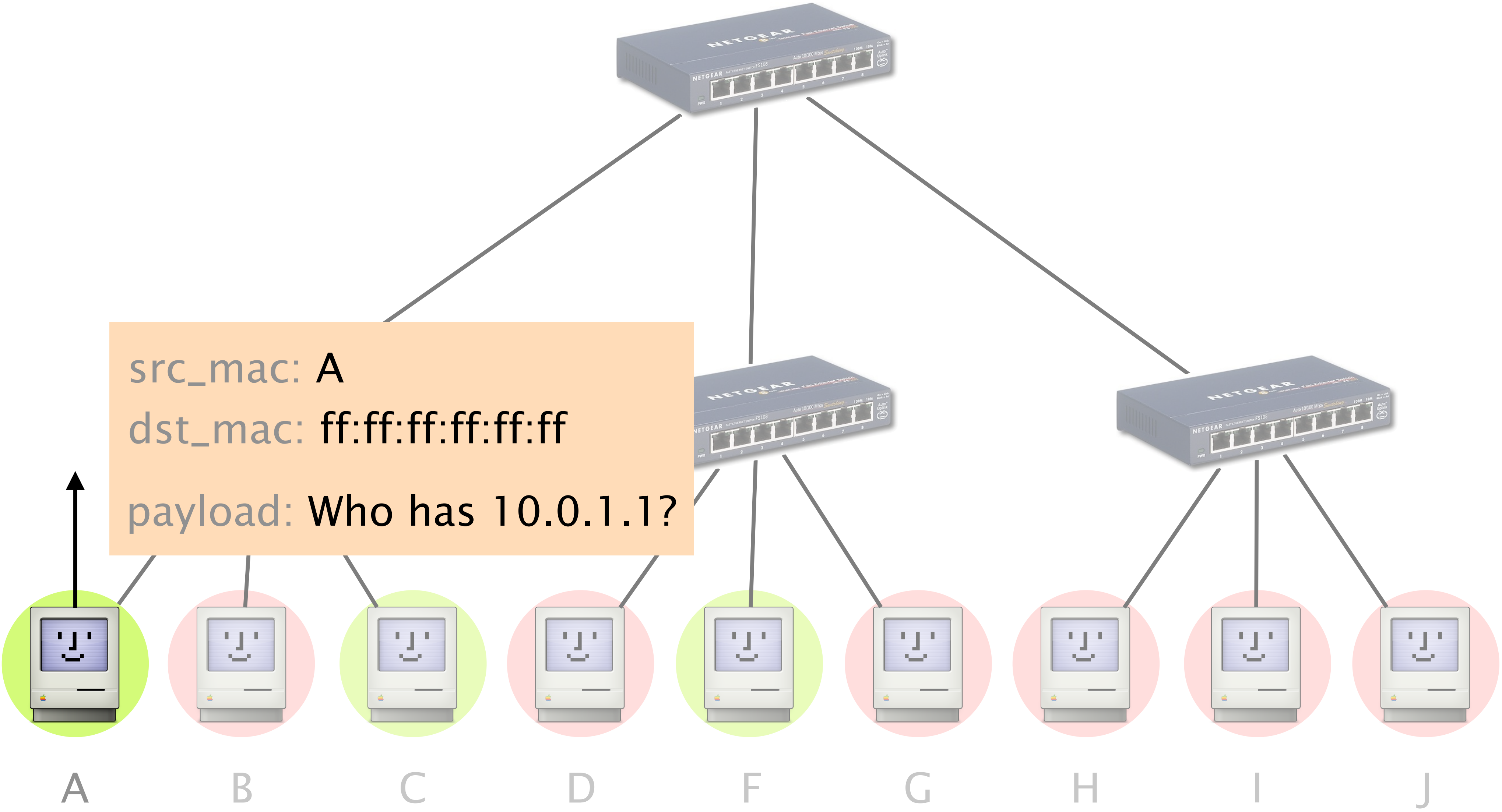


VLAN Staff: port A, port C

VLAN Student: port B

Consider that A sends a broadcast frame

say, an ARP request

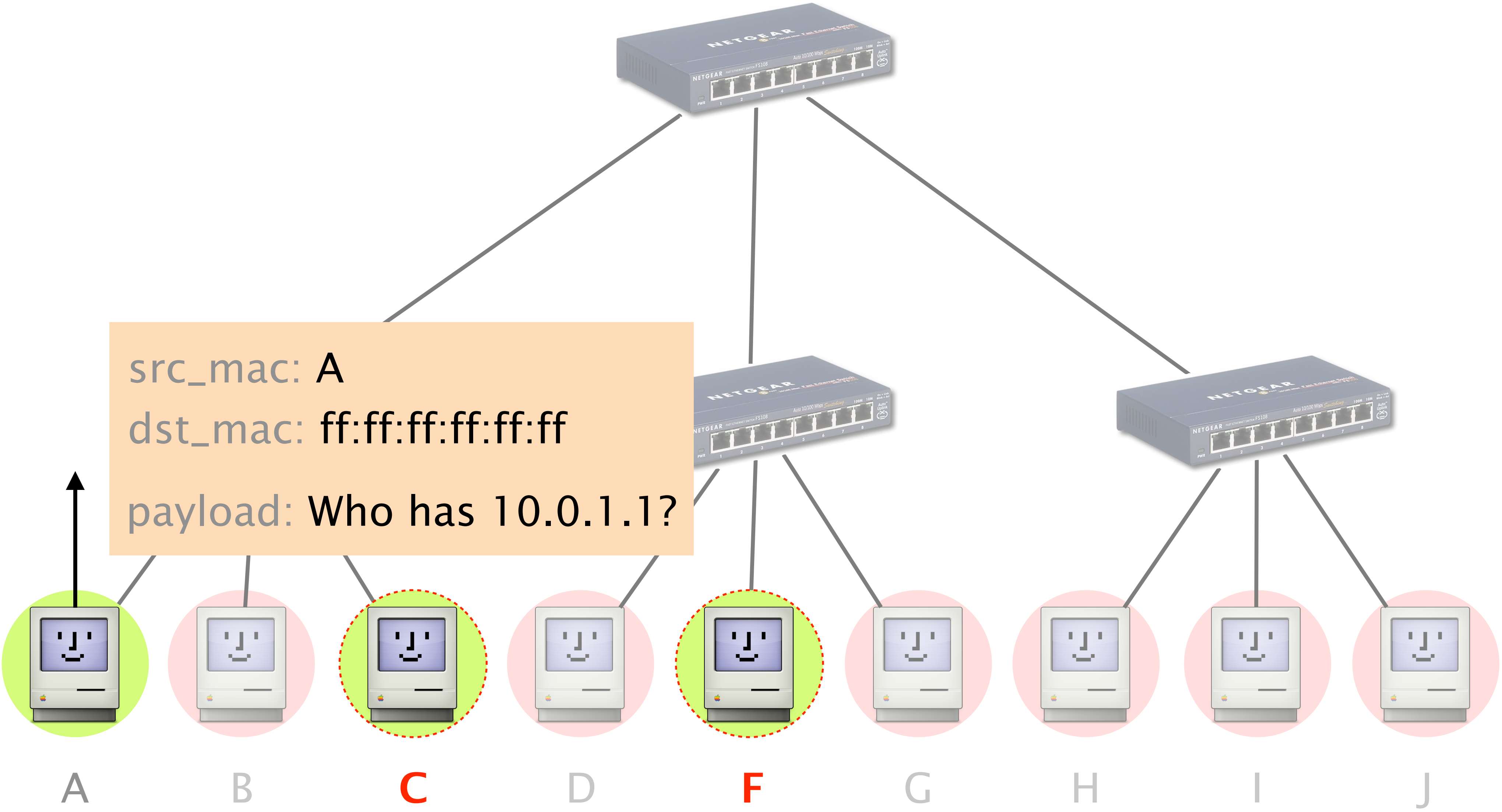


src_mac: A

dst_mac: ff:ff:ff:ff:ff:ff

payload: Who has 10.0.1.1?

That frame should be received by all staff members: i.e. C and F, and *only* them



src_mac: A

dst_mac: ff:ff:ff:ff:ff:ff

payload: Who has 10.0.1.1?

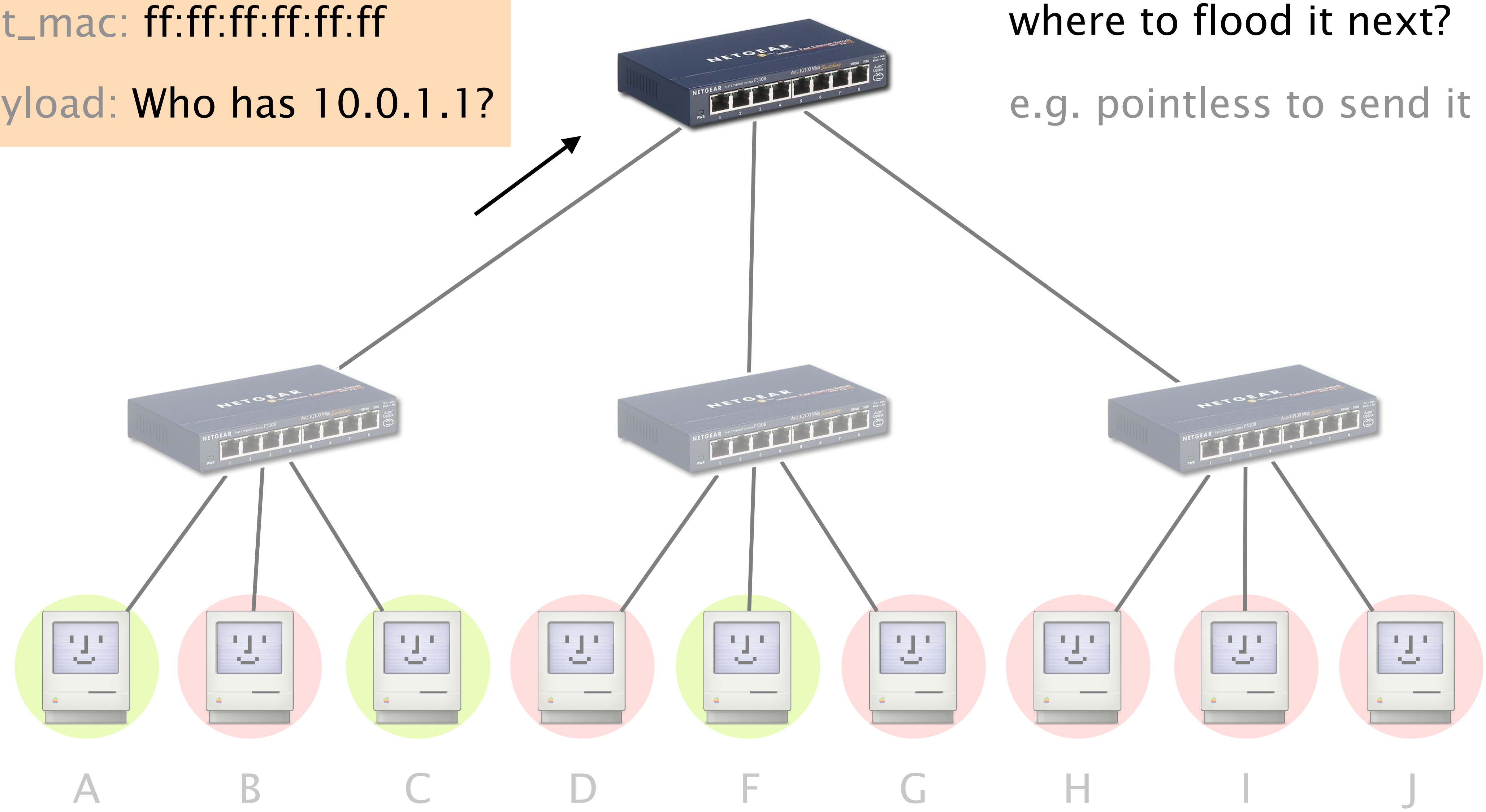
src_mac: A

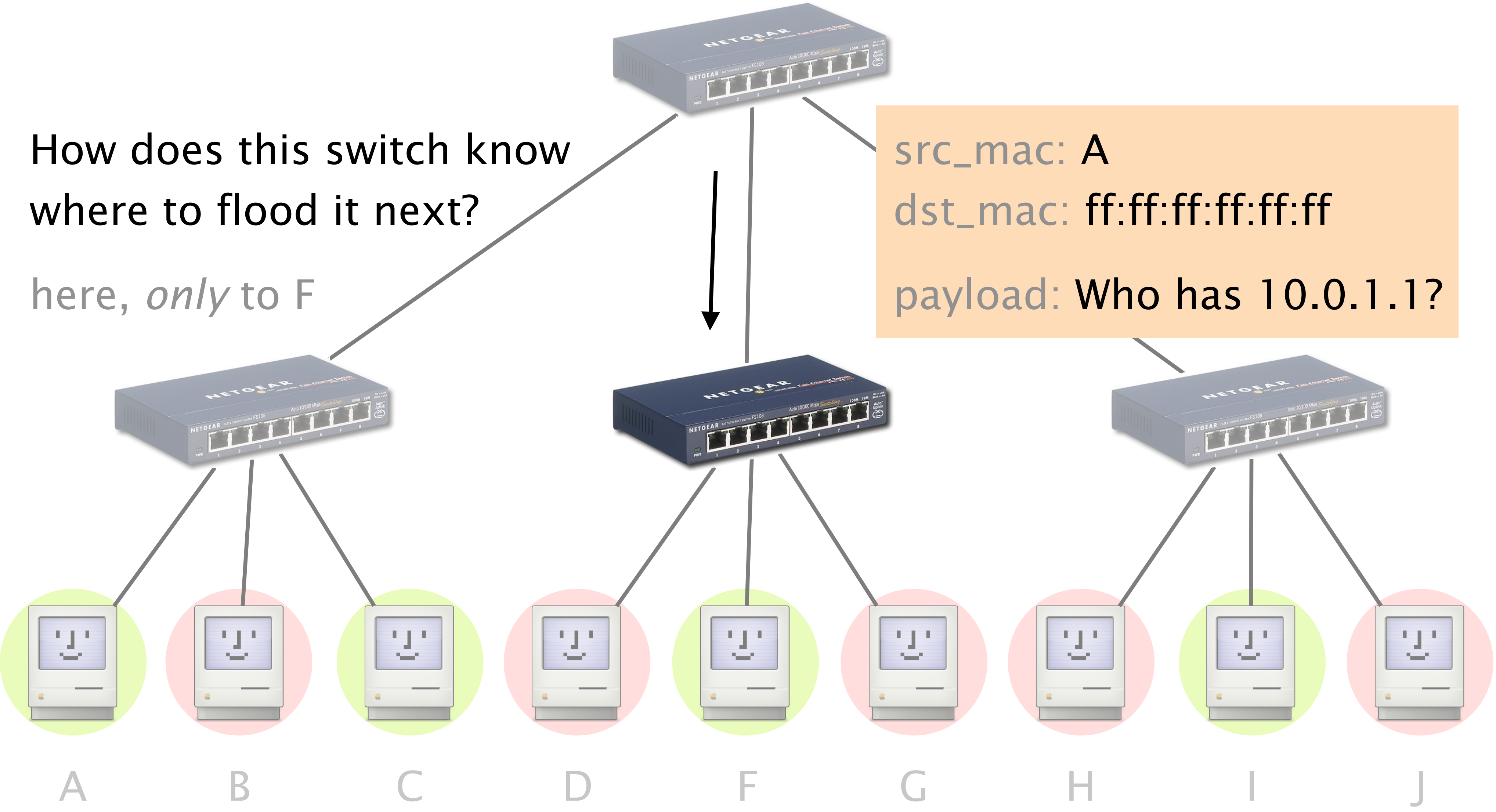
dst_mac: ff:ff:ff:ff:ff:ff

payload: Who has 10.0.1.1?

How does this switch know where to flood it next?

e.g. pointless to send it right





How does this switch know where to flood it next?

here, *only* to F

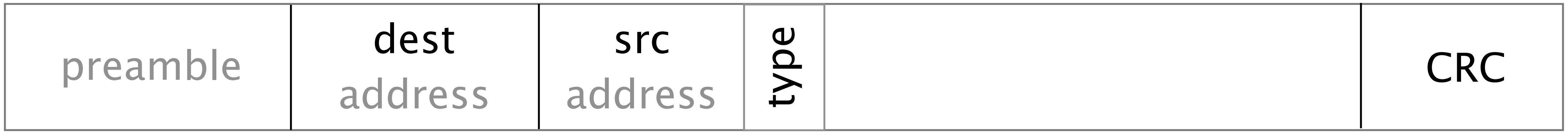
src_mac: A

dst_mac: ff:ff:ff:ff:ff:ff

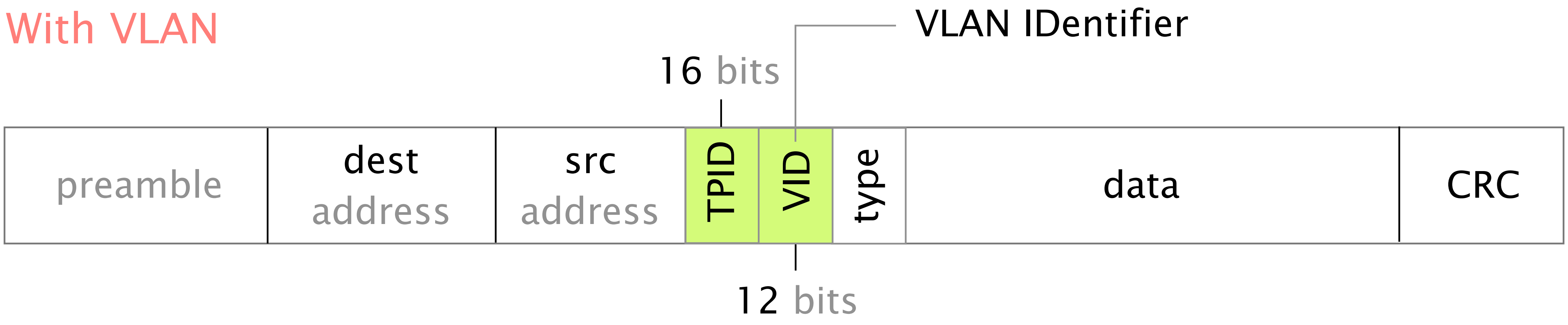
payload: Who has 10.0.1.1?

To identify VLAN, switches add new header when forwarding traffic to another switch

Without VLAN

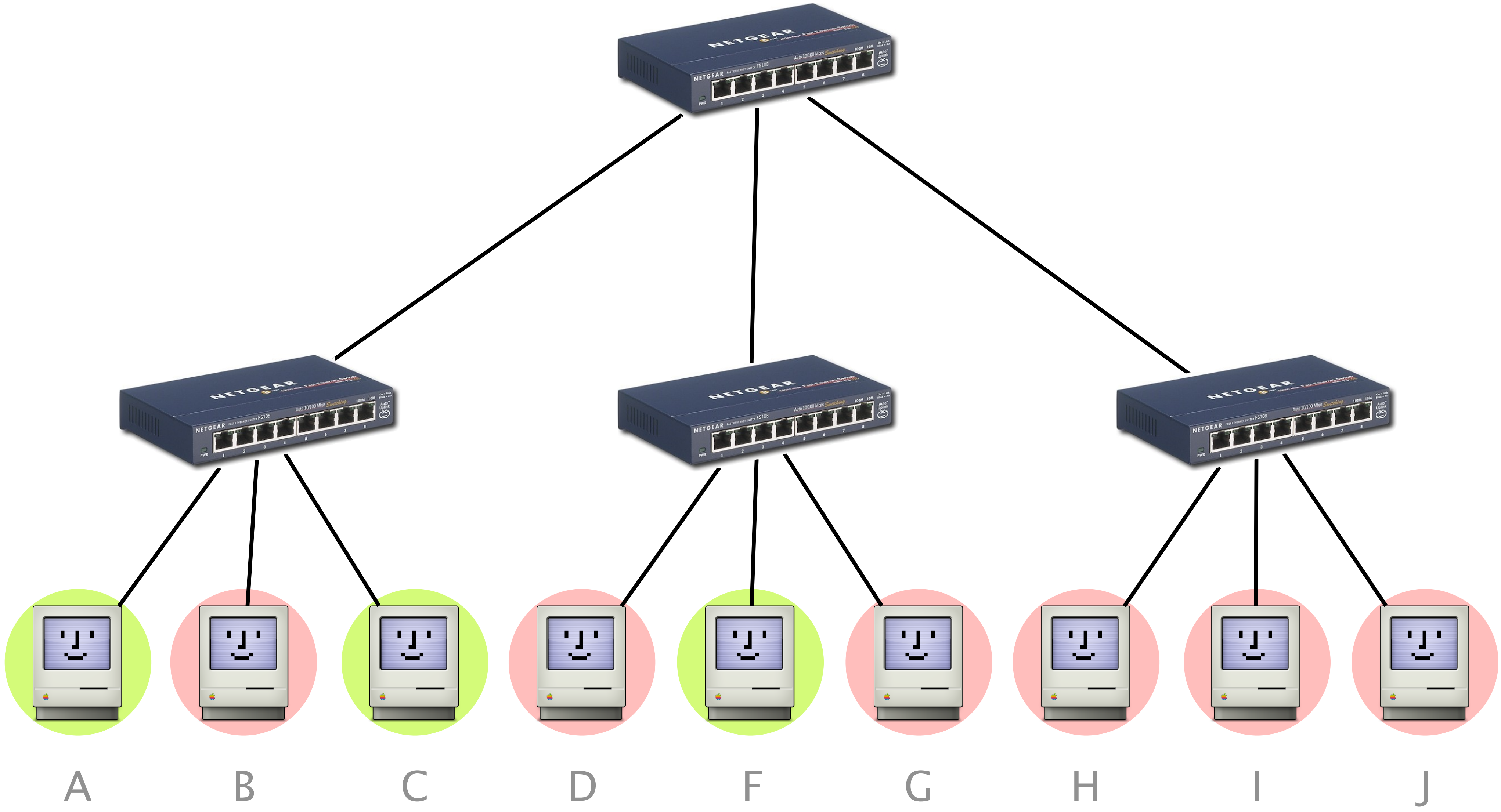


With VLAN



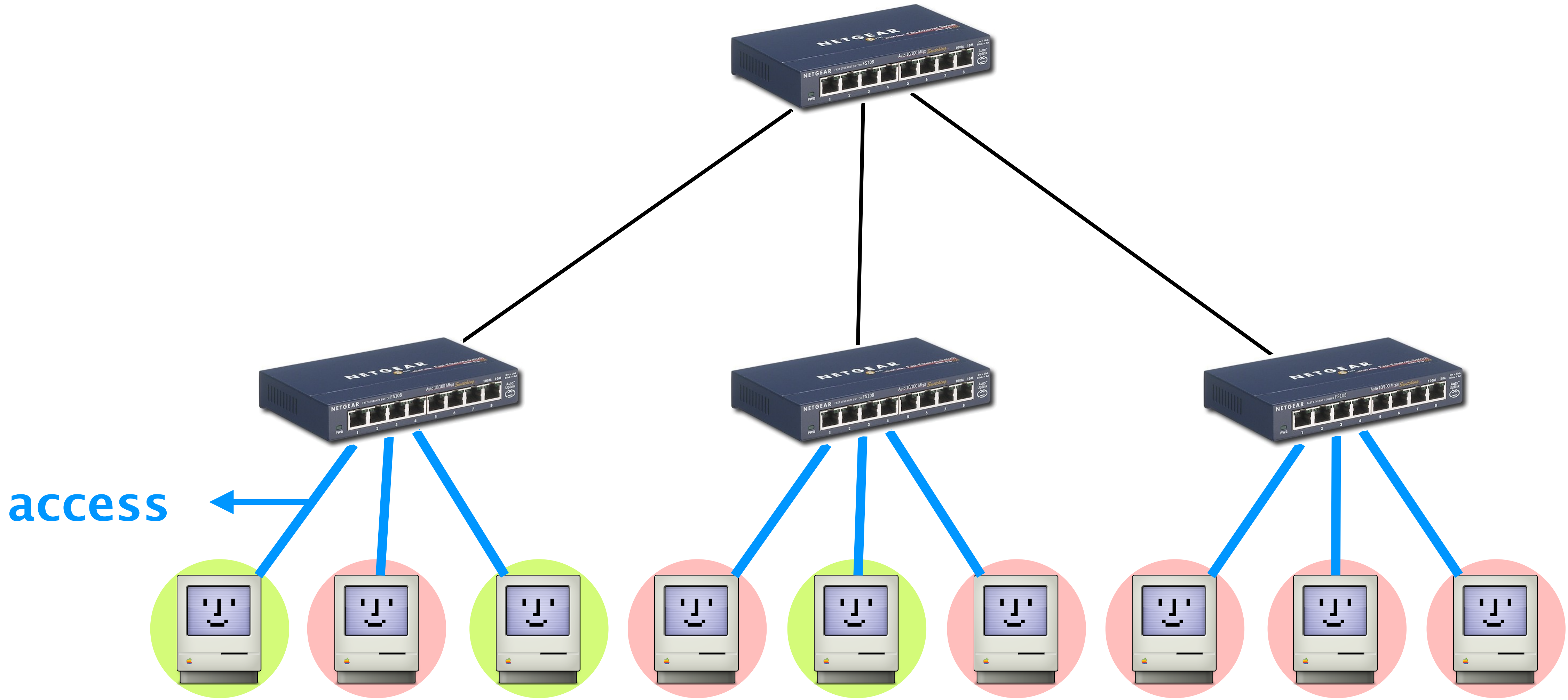
802.1q Header (4 bytes)
(4 bits missing)

With VLANs, Ethernet links are divided in two sets: access and trunks (inter switches) links



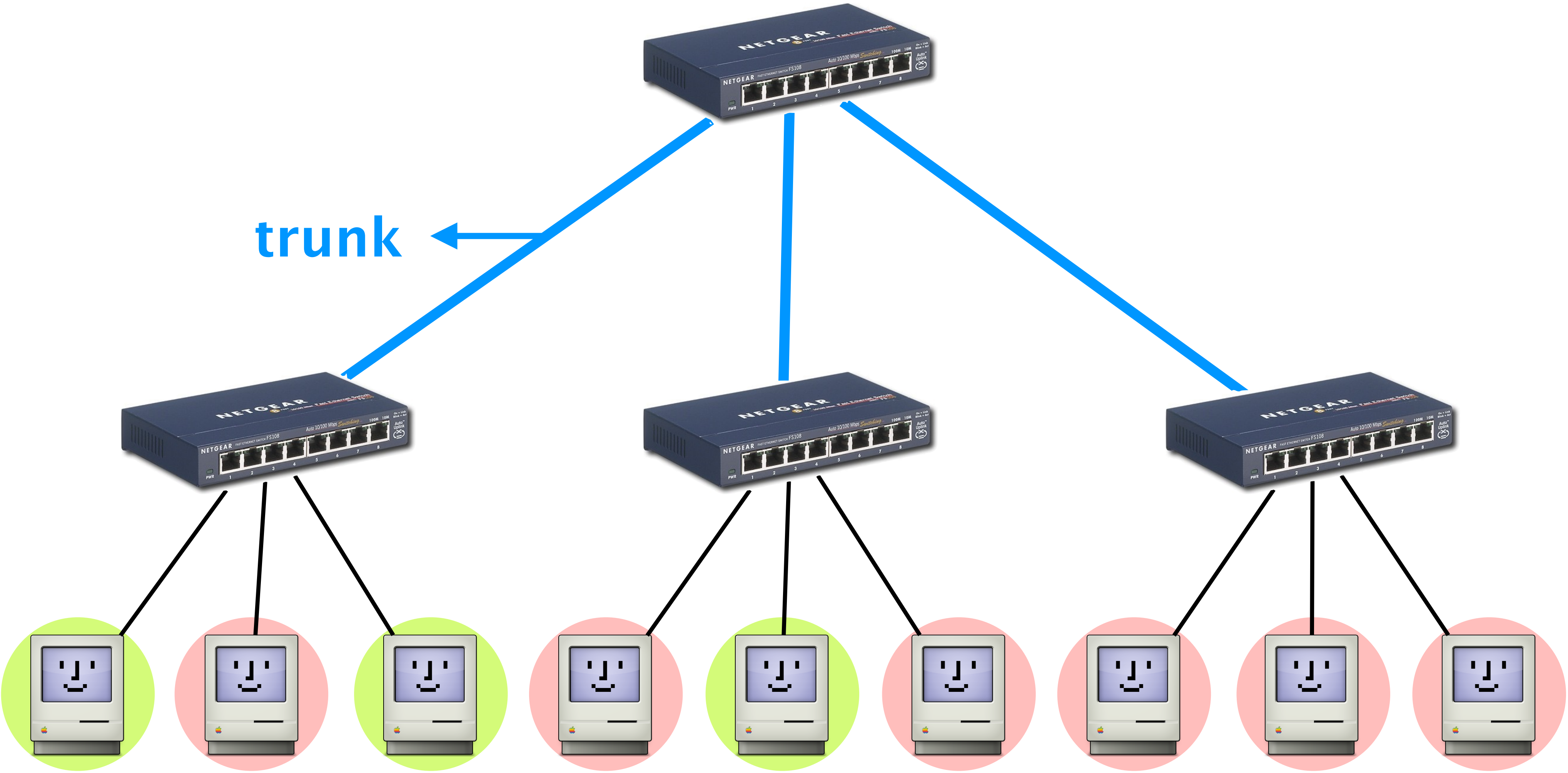
Access links belong to one VLAN

they do not carry 802.1q headers



Trunk links carry traffic for more than one VLAN

and as such carry 801.1q tagged frames



Each switch runs
one MAC learning algorithm for each VLAN

When a switch receives a frame with
an unknown or a broadcast destination,

it forwards it over all the ports that
belong to the same VLAN

When a switch learns a source address on a port

it associates it to the VLAN of this port and
only uses it when forwarding frames on this VLAN

VLAN “Staff” 🟢

port 1: A, C   port 2: F

VLAN “Student” 🔴

port 1: B   port 2: D,G

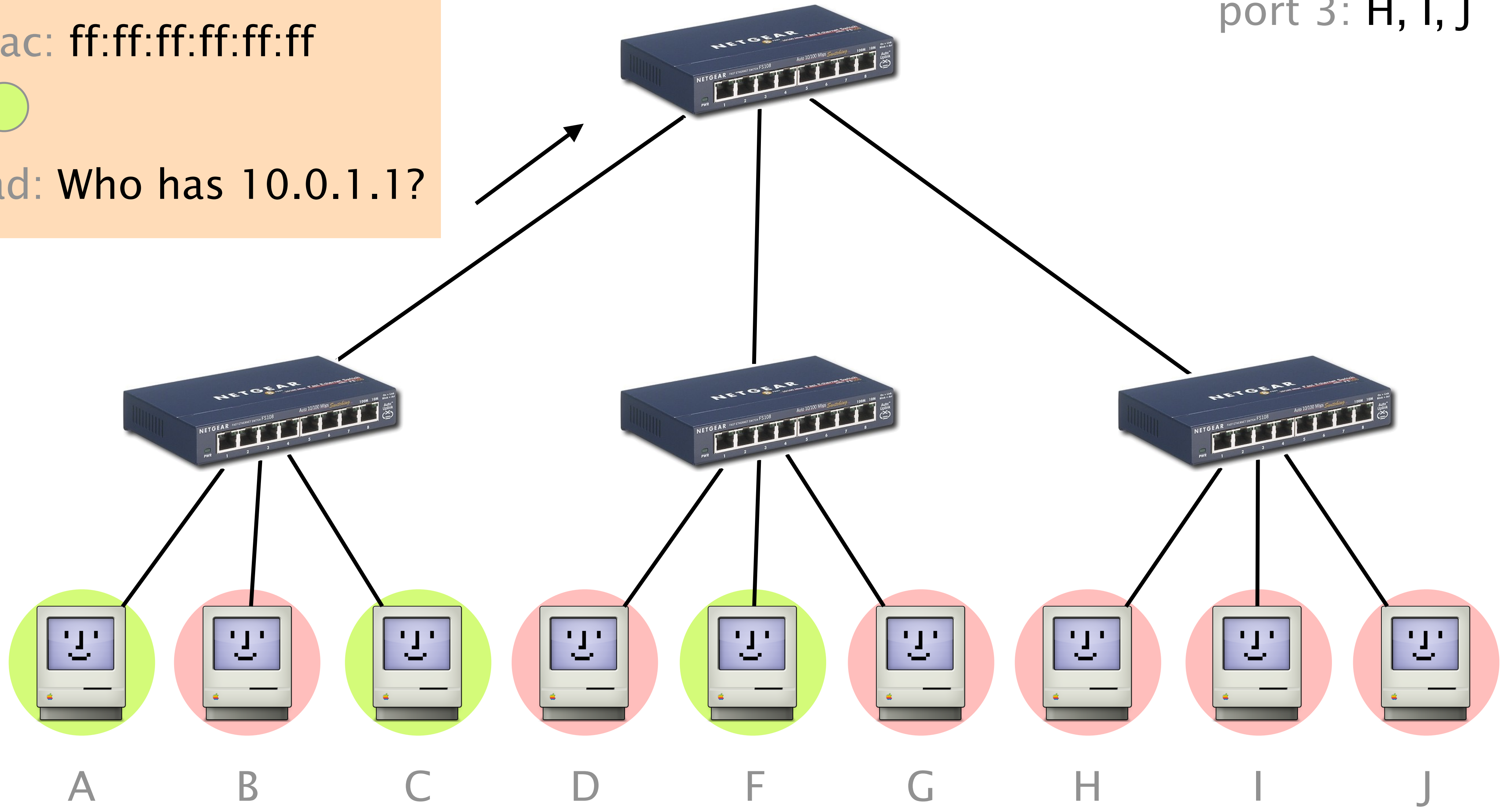
port 3: H, I, J

src_mac: A

dst_mac: ff:ff:ff:ff:ff:ff

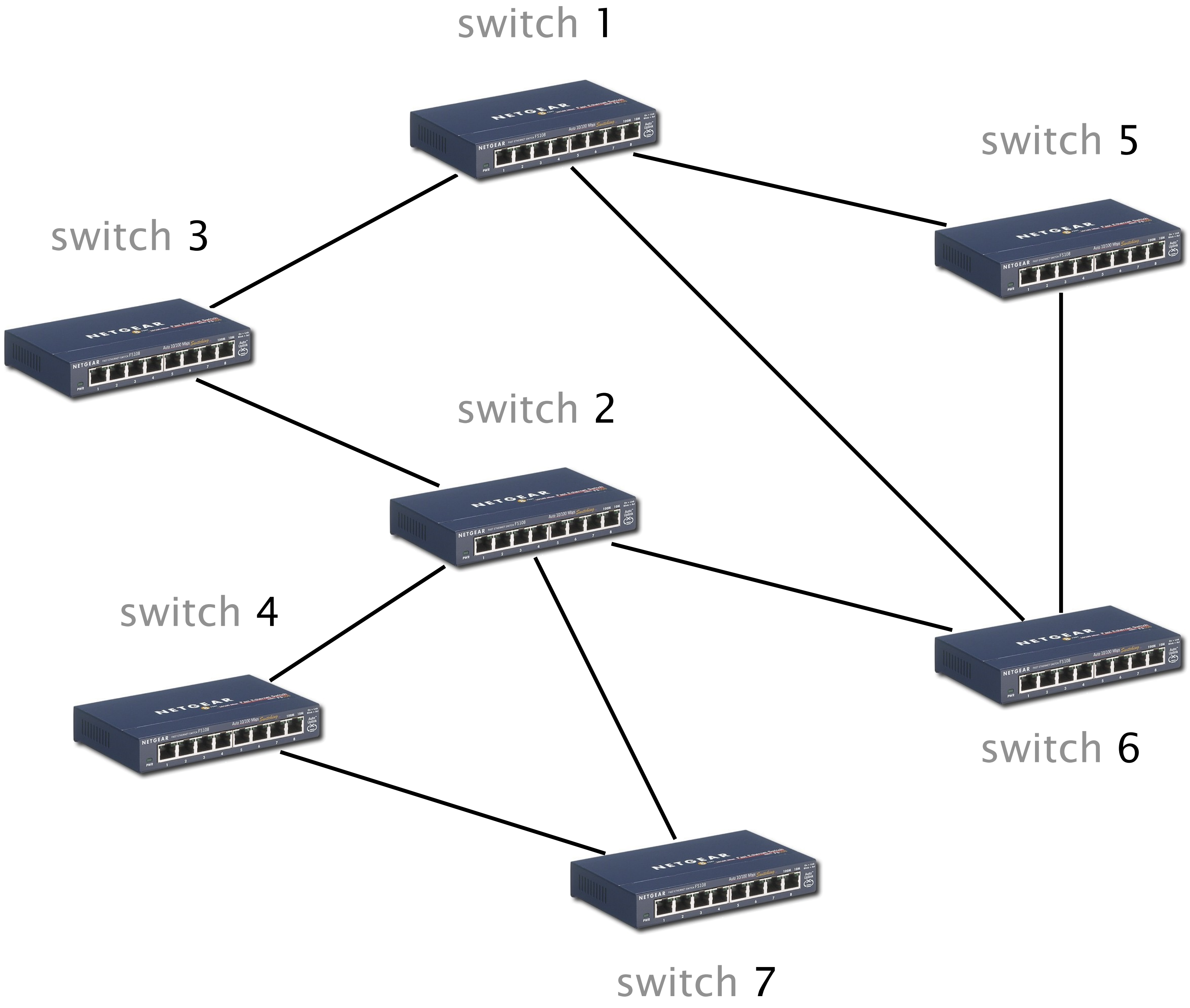
VID: 🟢

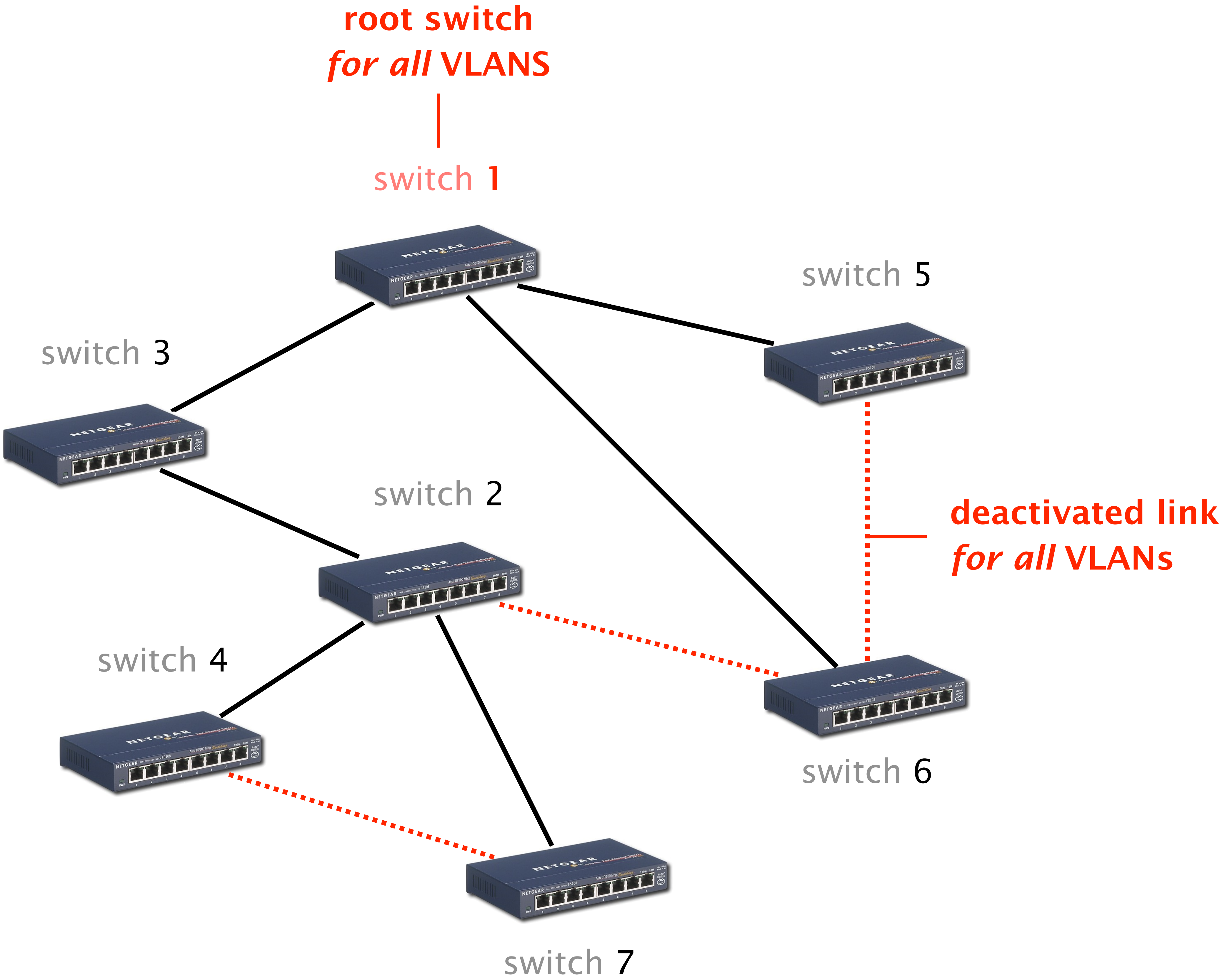
payload: Who has 10.0.1.1?



Switches can also compute per-VLAN spanning-tree allowing a distinct SPT for each VLAN

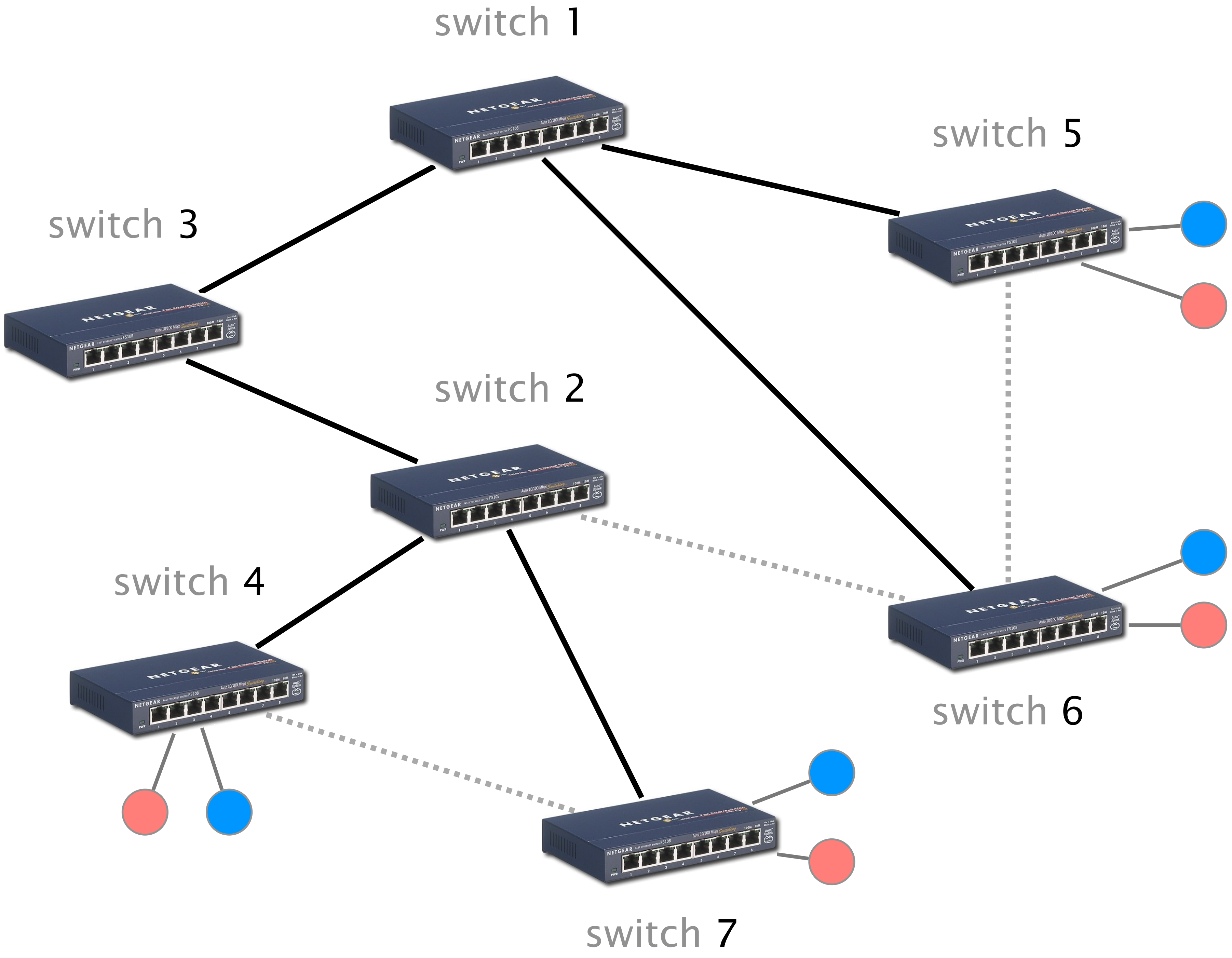
allow the operators to use more of their links


switch 1
switch 5
switch 3
switch 2
switch 4
switch 6
switch 7

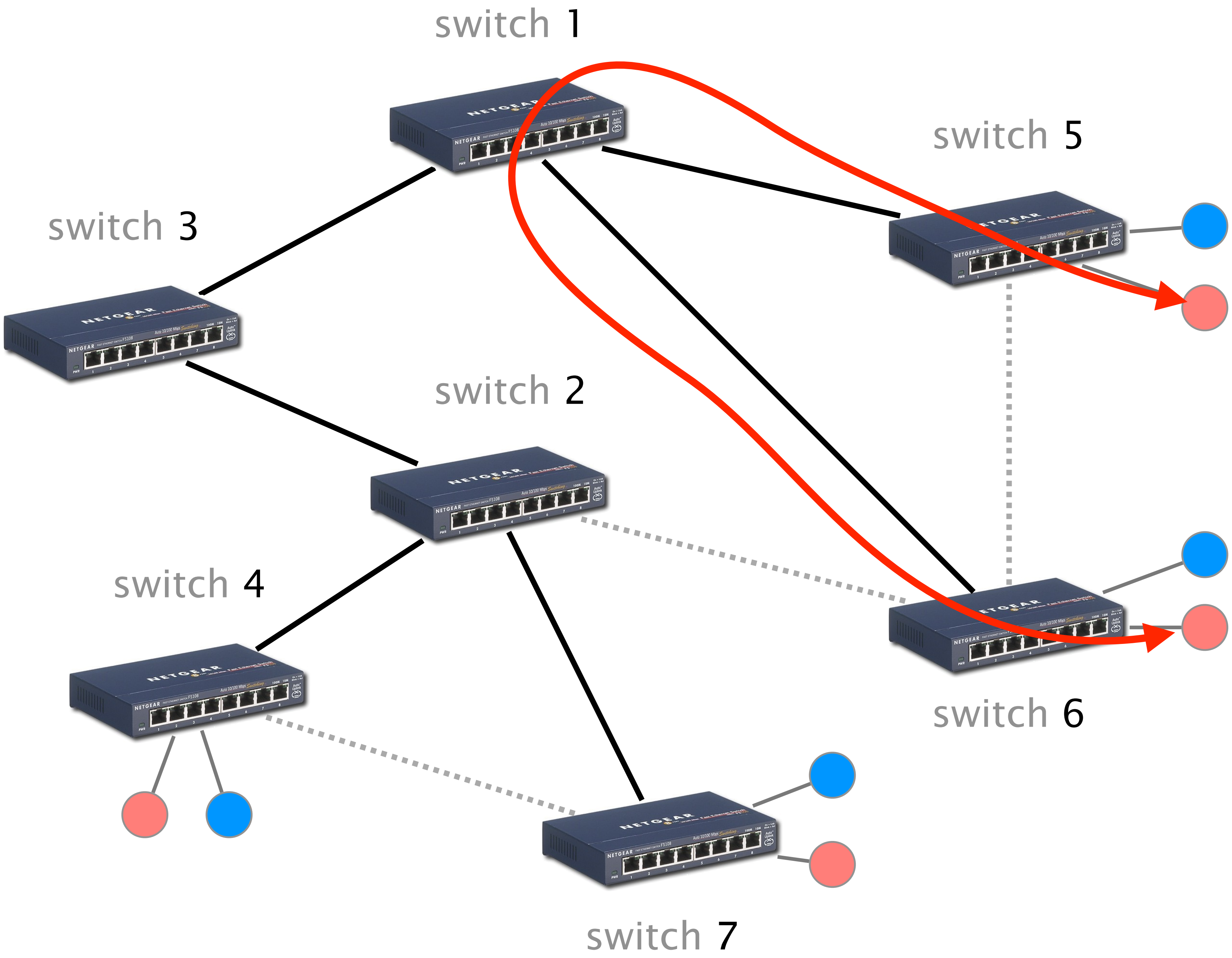


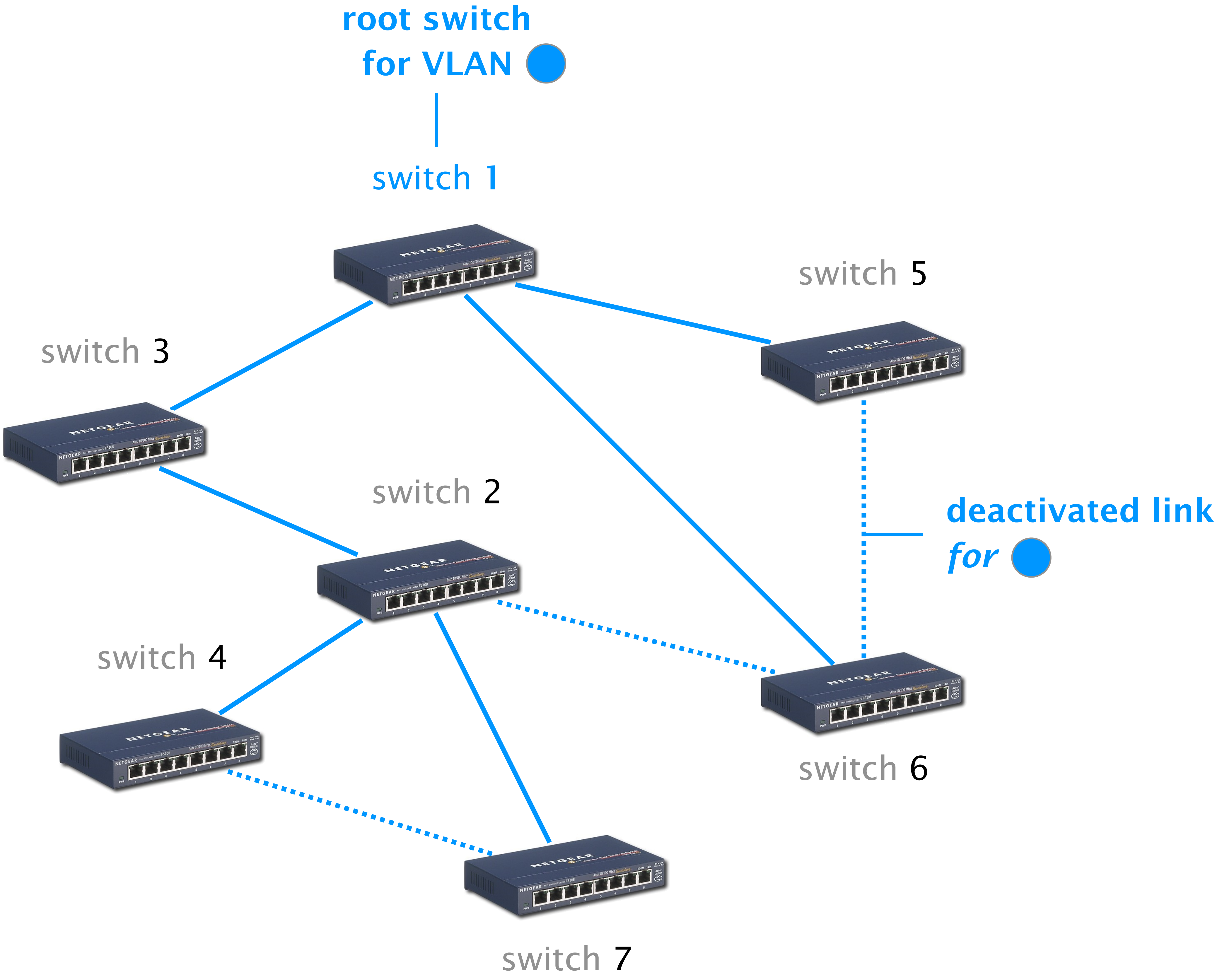
root switch
for all VLANS

deactivated link
for all VLANs


switch 1
switch 5
switch 3
switch 2
switch 4
switch 6
switch 7

Any communication between the red hosts on switch 5 and 6 need to go via switch 1…





root switch for VLAN ●

switch 1

switch 2

switch 3

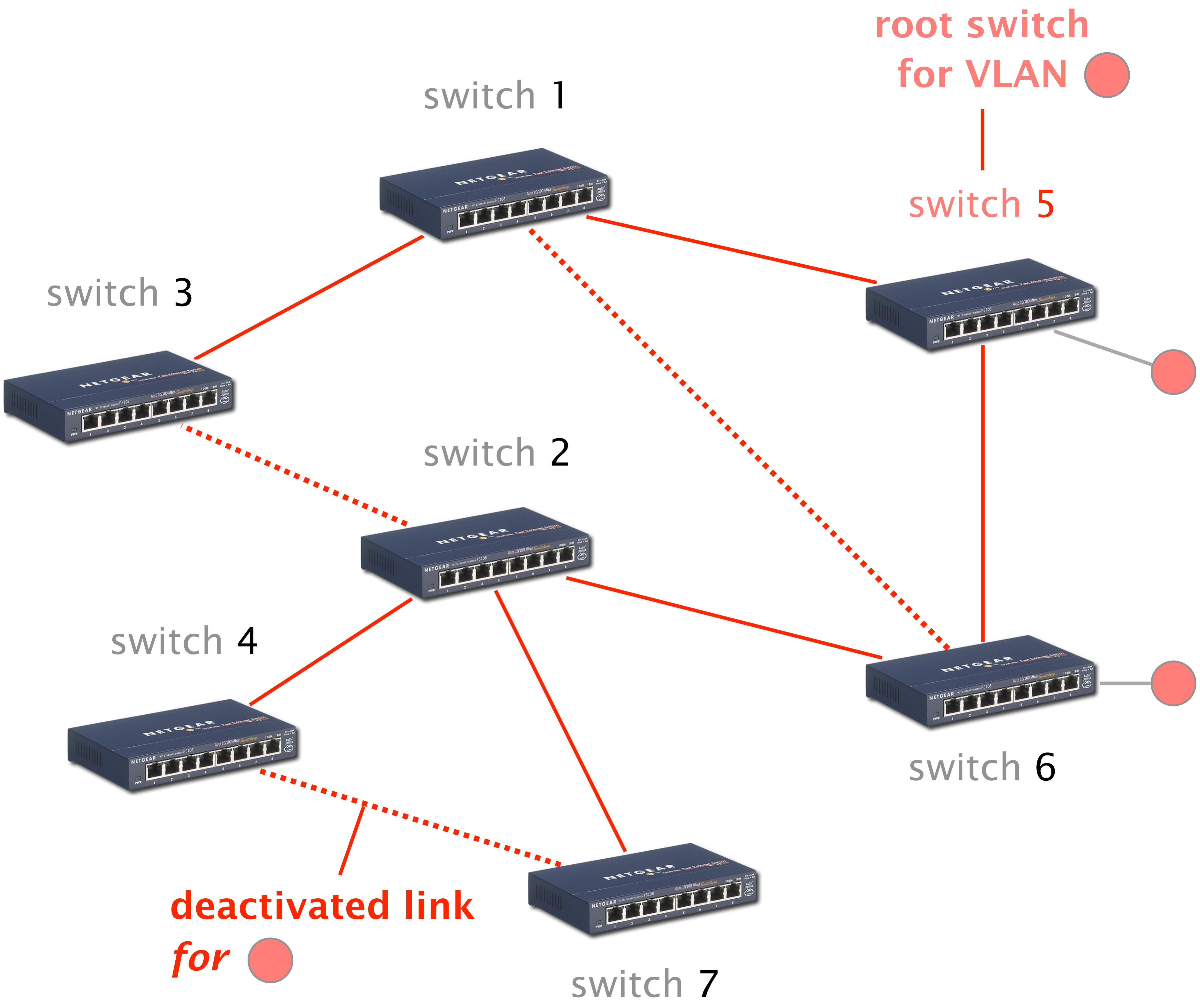
switch 4

switch 5

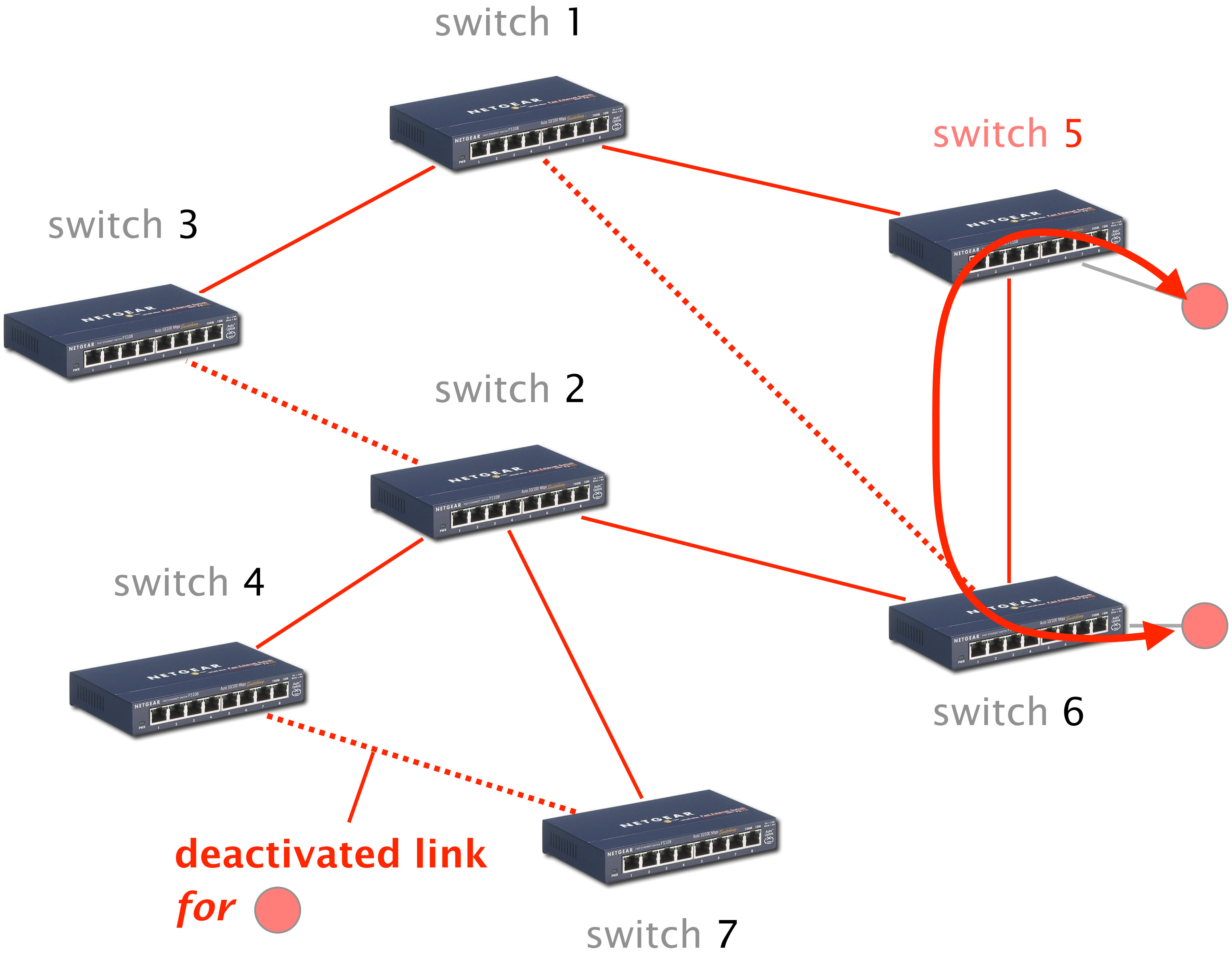
switch 6

switch 7

deactivated link *for* ●


root switch
for VLAN ●
switch 1
switch 5
switch 3
switch 2
switch 4
switch 6
deactivated link
for ●
switch 7

Now any communication between the red hosts on switch 5 and 6 go via the direct link

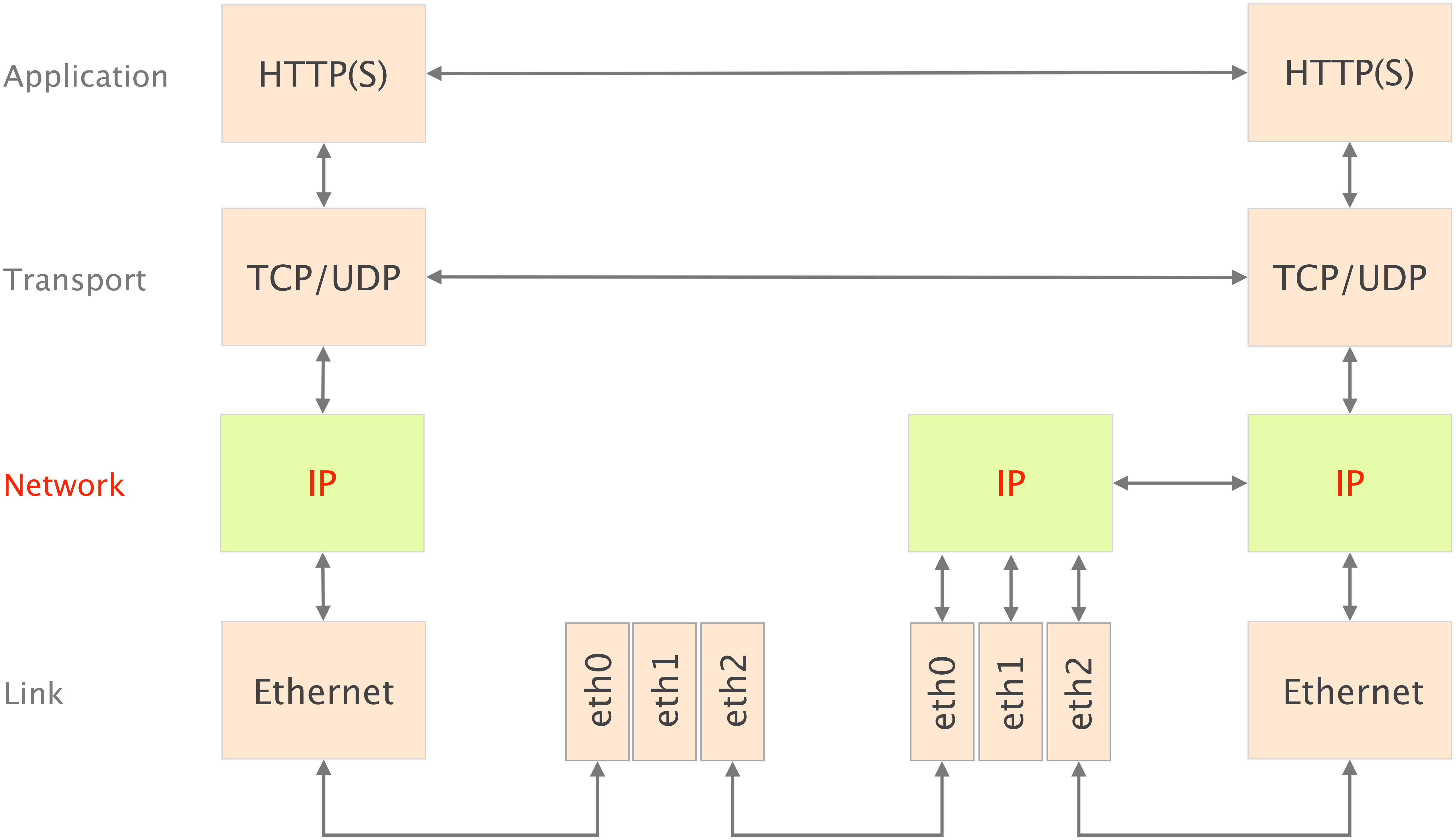


Link Layer

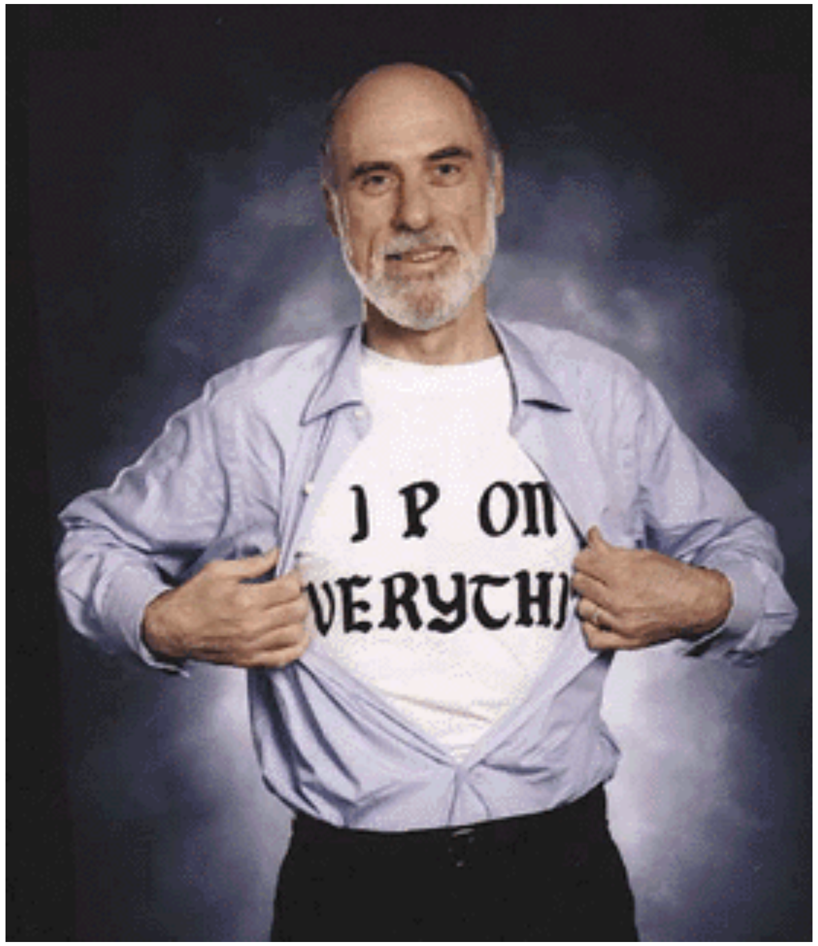
Network Layer

The Beginning

Moving on to IP and the network layer



Internet Protocol and Forwarding



source: Boardwatch Magazine

1 **IP addresses**

use, structure, allocation

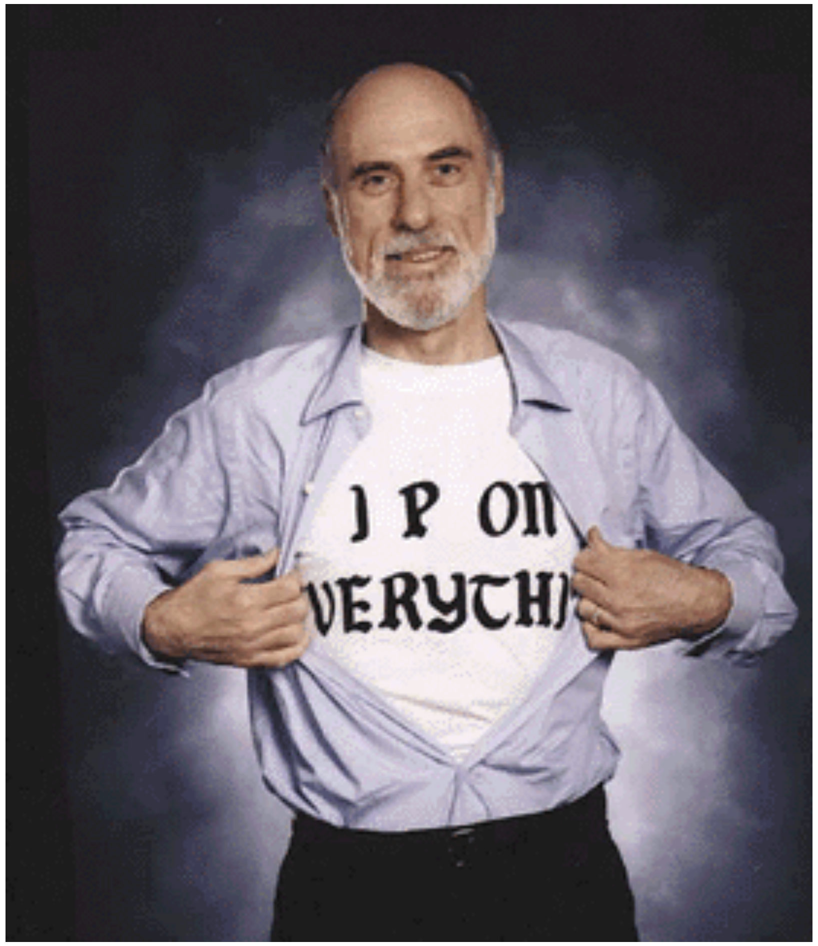
2 **IP forwarding**

longest prefix match rule

3 **IP header**

IPv4 and IPv6, wire format

Internet Protocol and Forwarding



1 IP addresses

use, structure, allocation

IP forwarding

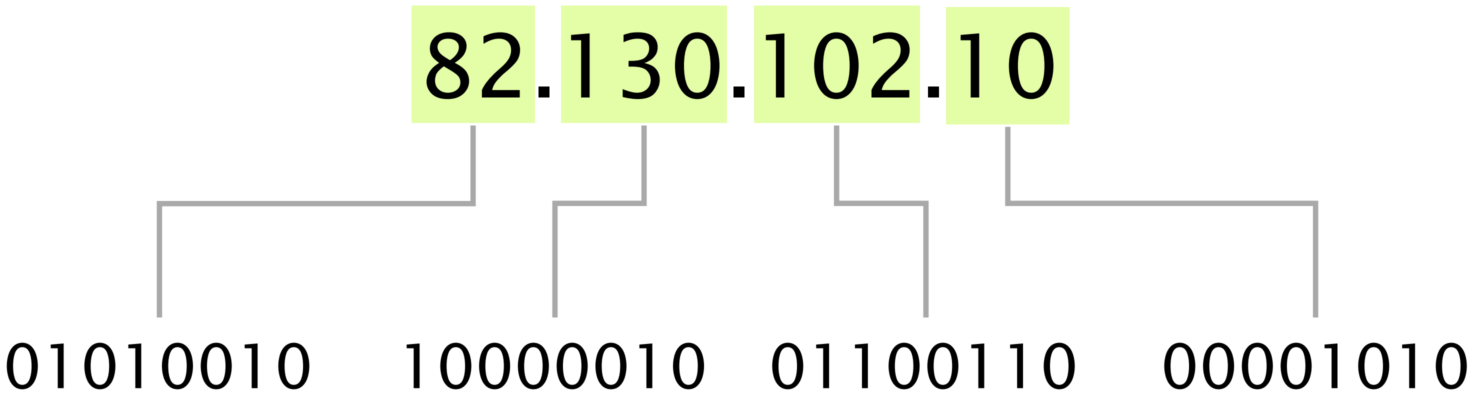
longest prefix match rule

IP header

IPv4 and IPv6, wire format

IPv4 addresses are unique 32-bits number associated to a network interface (on a host, a router, …)

IP addresses are usually written using dotted-quad notation

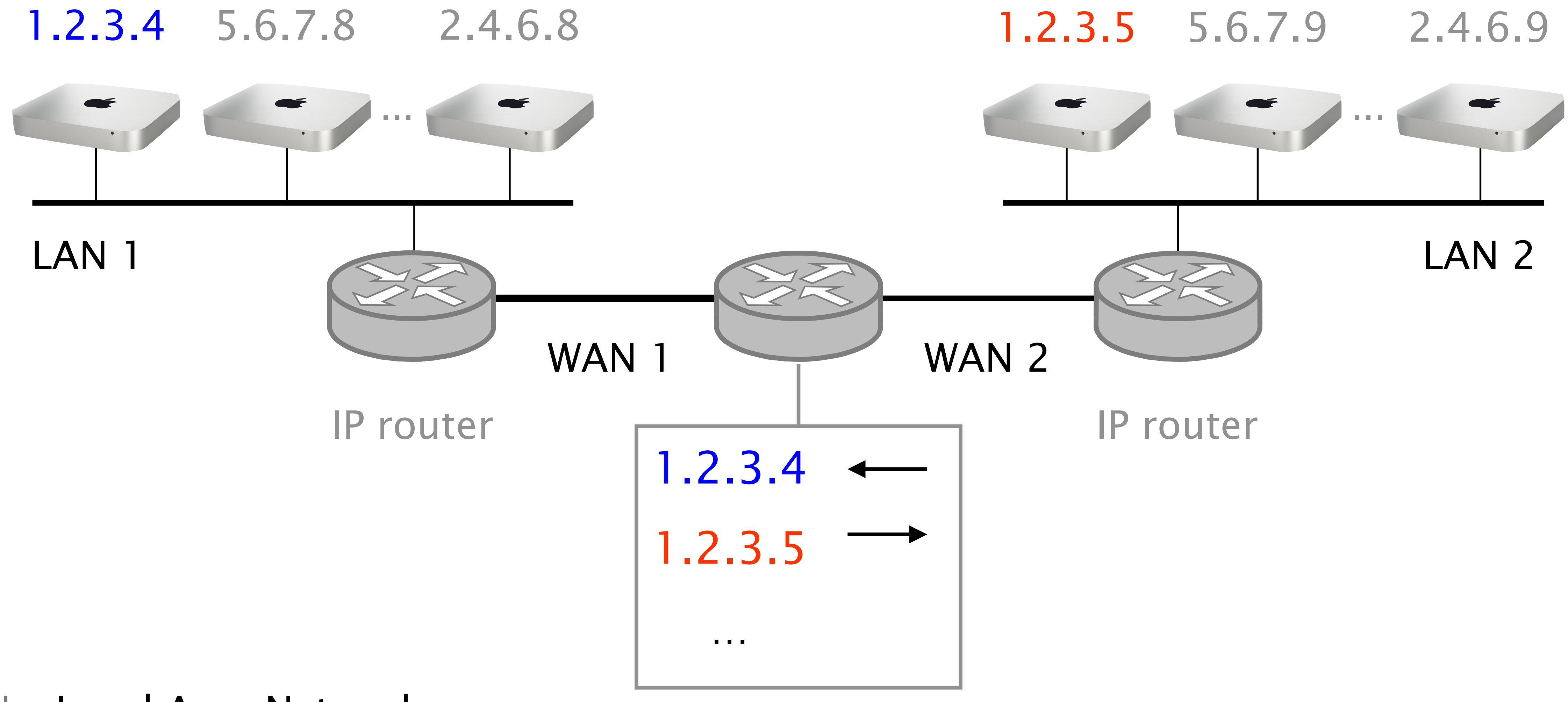


82.130.102.10

01010010 10000010 01100110 00001010

Routers forwards IP packets
based on their destination IP address

If IP addresses were assigned arbitrarily, routers would require forwarding entries for all of them



forwarding table

LAN Local Area Network

WAN Wide Area Network

18 billion

estimated* # of Internet connected devices

in 2017

[*] Cisco Visual Networking Index 2017—2022

28.5 billion

estimated* # of Internet connected devices

in 2022

* Cisco Visual Networking Index 2017—2022

Two universal tricks you can apply to any computer sciences problem

When you need… more flexibility,

you add… a layer of indirection

When you need… more scalability,

you add… a hierarchical structure

When you need… more scalability,

you add… a hierarchical structure

IP addresses are hierarchically allocated, similarly to the postal service

Address

Zip	8092
Street	Gloriastrasse
Building	35 (ETZ)
Location in building	G 90
Name	Laurent Vanbever

Nobody in the Swiss mail system knows where every single house or building is

principle

Routing tables are separated
at each level of the hierarchy

each one with a manageable scale

Forwarding in the Swiss mail

in 4 steps

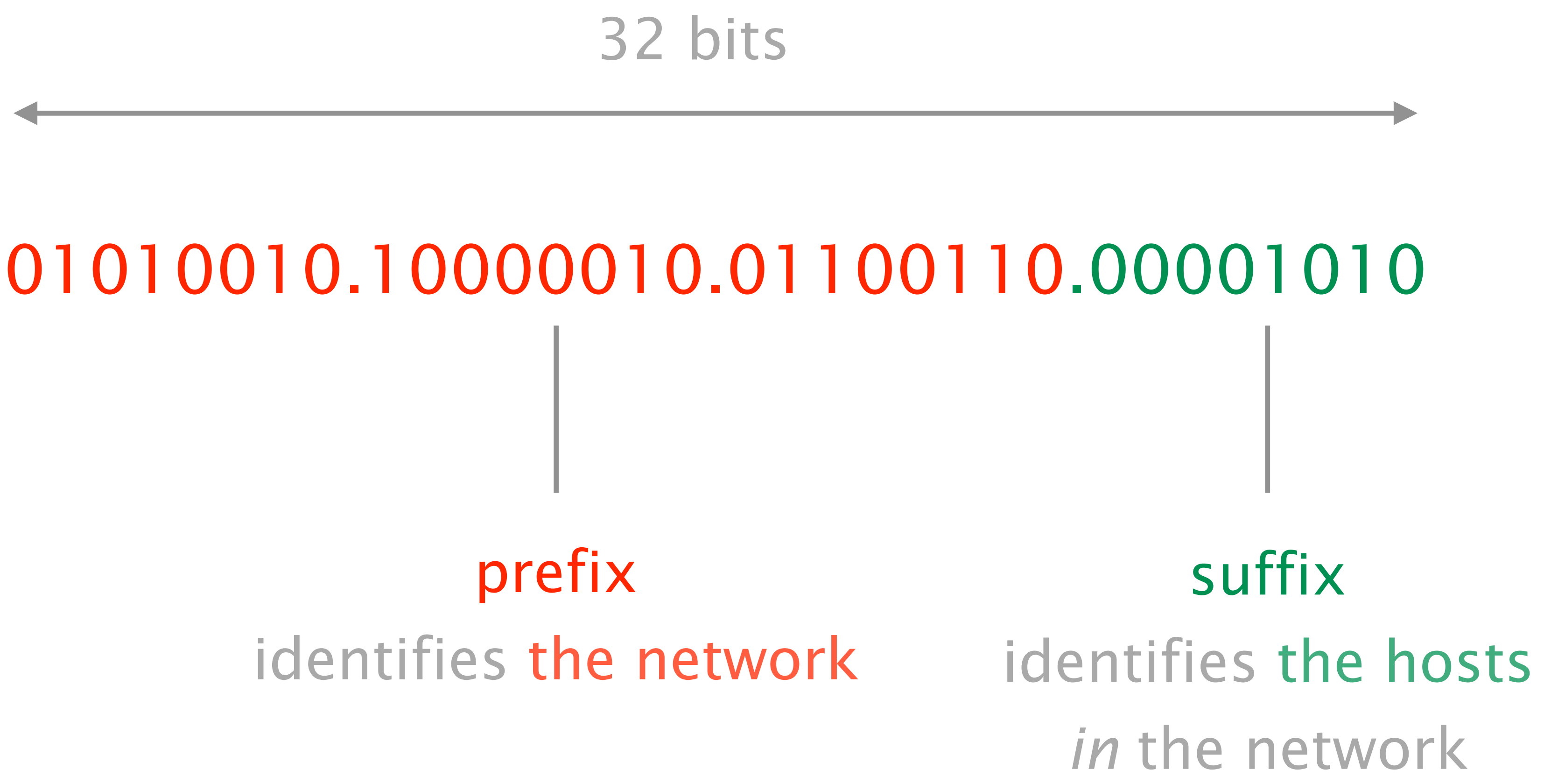
1 Deliver the letter to the post office responsible for the zip code

2 Assign letter to the mail person covering the street

3 Drop letter into the mailbox attached to the building

4 Hand in the letter to the appropriate person

IP addressing is hierarchical, composed of a prefix (network address) and a suffix (host address)



32 bits

01010010.10000010.01100110.00001010

prefix

identifies the network

suffix

identifies the hosts *in* the network

Each prefix has a given length,
usually written using a “slash notation”

IP prefix 82.130.102.0 /24

prefix length (in bits)

Here, a /24 means that we have 8 bits left to address hosts address, enough for 256 hosts

82.130.102.0 /24

prefix part	host part	IP address
01010010.10000010.01100110.	00000000	82.130.102.0
01010010.10000010.01100110.	00000001	82.130.102.1
01010010.10000010.01100110.	00000010	82.130.102.2
01010010.10000010.01100110.	11111110	82.130.102.254
01010010.10000010.01100110.	11111111	82.130.102.255

In practice, the first and last IP address of a prefix are not usable

prefix part	host part	IP address
01010010.10000010.01100110.	00000000	82.130.102.0
01010010.10000010.01100110.	11111111	82.130.102.255

The address with the host part being all 0s identifies the network itself

prefix part	host part	IP address
01010010.10000010.01100110.	00000000	82.130.102.0

The address with the host part being all 1s identifies the broadcast address

prefix part	host part	IP address
01010010.10000010.01100110.	11111111	82.130.102.255

A /24 has therefore only 254 addresses
that can be allocated to hosts

Prefixes are also sometimes specified using an address and a mask

Address 82.130.102.0

01010010.10000010.01100110. 00000000

11111111.11111111.11111111. 00000000

Mask 255.255.255.0

ANDing the address and the mask gives you the prefix

Address 82.130.102.0

01010010.10000010.01100110. 00000000

11111111.11111111.11111111. 00000000

Mask 255.255.255.0

Given this IP prefix 82.130.0.0/17

Compute

# of addressable hosts

the prefix mask

network address

1st host address

last host address

broadcast address

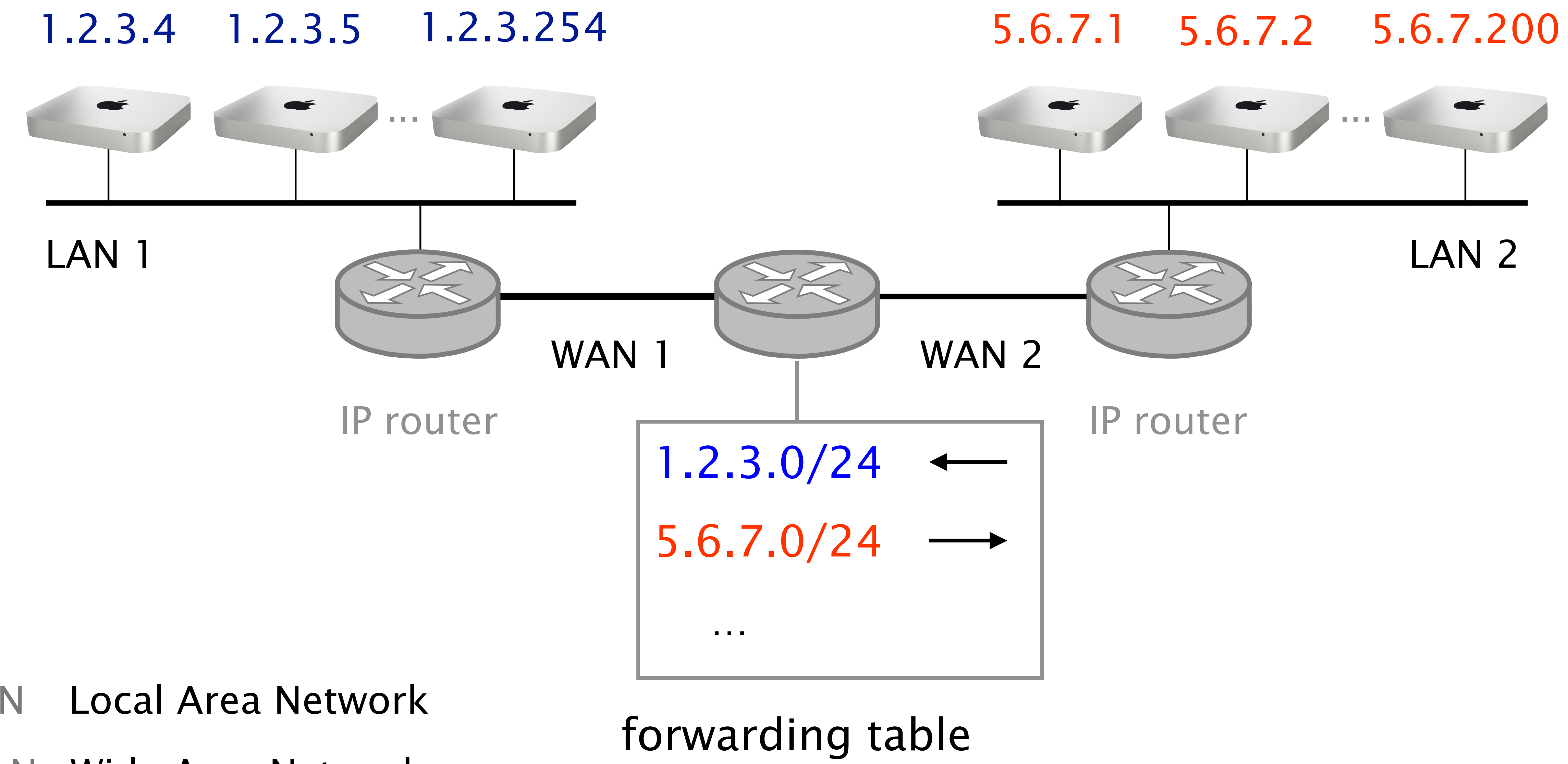
Given this IP prefix 82.130.0.0/17

Compute

# of addressable hosts	32,766
the prefix mask	255.255.128.0
network address	82.130.0.0
1st host address	82.130.0.1
last host address	82.130.127.254
broadcast address	82.130.127.255

Routers forward packet to their destination according to the network part, *not* the host part

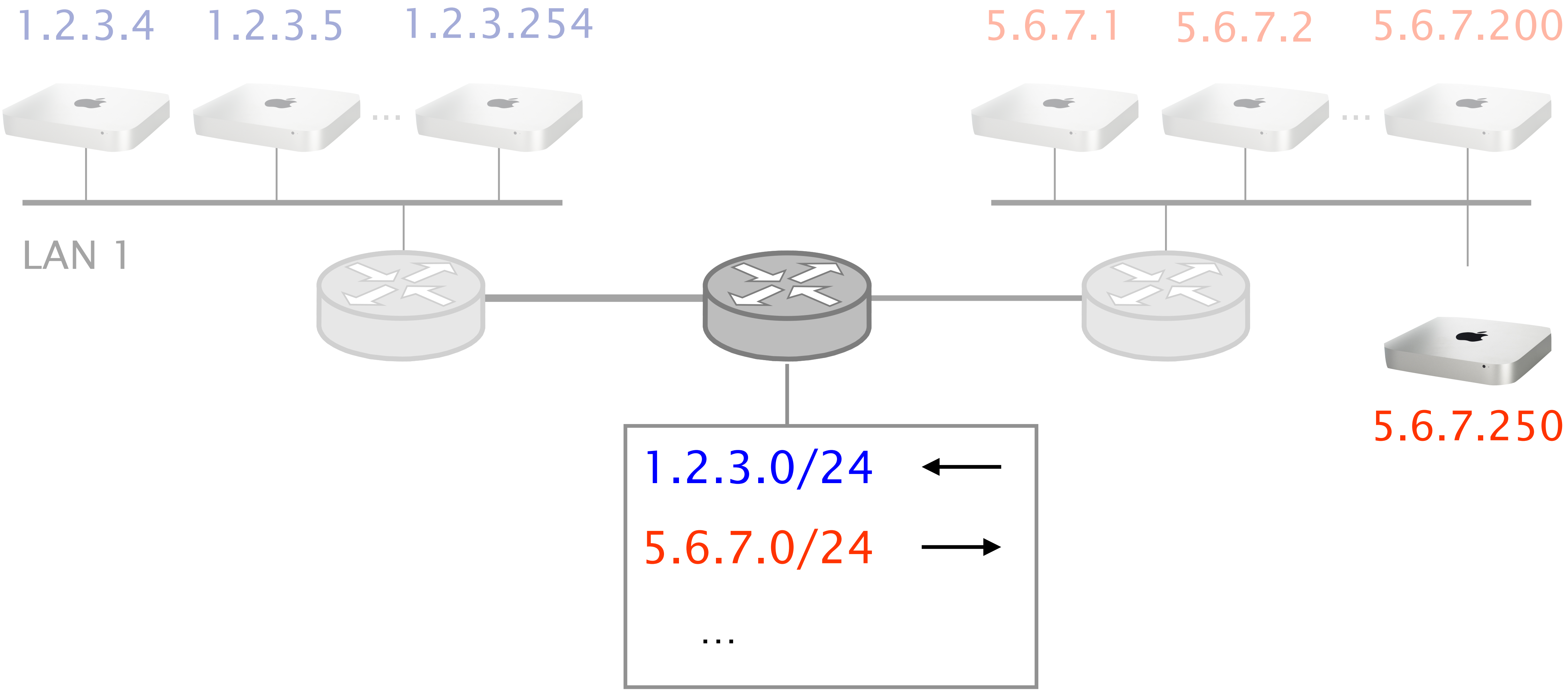
Doing so enables to scale the forwarding tables



LAN Local Area Network

WAN Wide Area Network

Hierarchical addressing enables to add new hosts without changing or adding forwarding rules



forwarding table

Originally, there were only 5 fixed allocation sizes, (or classes)—known as classful networking

	leading	prefix	#	start	end
	bits	length	hosts	address	address
class A	0	8	2^{24}	0.0.0.0	127.255.255.255
class B	10	16	2^{16}	128.0.0.0	191.255.255.255
class C	110	24	2^{8}	192.0.0.0	223.255.255.255
class D multicast	1110			224.0.0.0	239.255.255.255
class E reserved	1111			240.0.0.0	255.255.255.255

Classful networking was quite wasteful leading to IP address exhaustion

problem

Class C was too small, so everybody requested class B

which where: *i)* too big and *ii)* too few (wasted space)

solution

Classless Inter-Domain Routing (CIDR)

introduced in 1993

CIDR enabled flexible division between network and hosts addresses

CIDR must specify both the address and the mask

classful was communicating this in the first address bits

Masks are carried by the routing algorithms

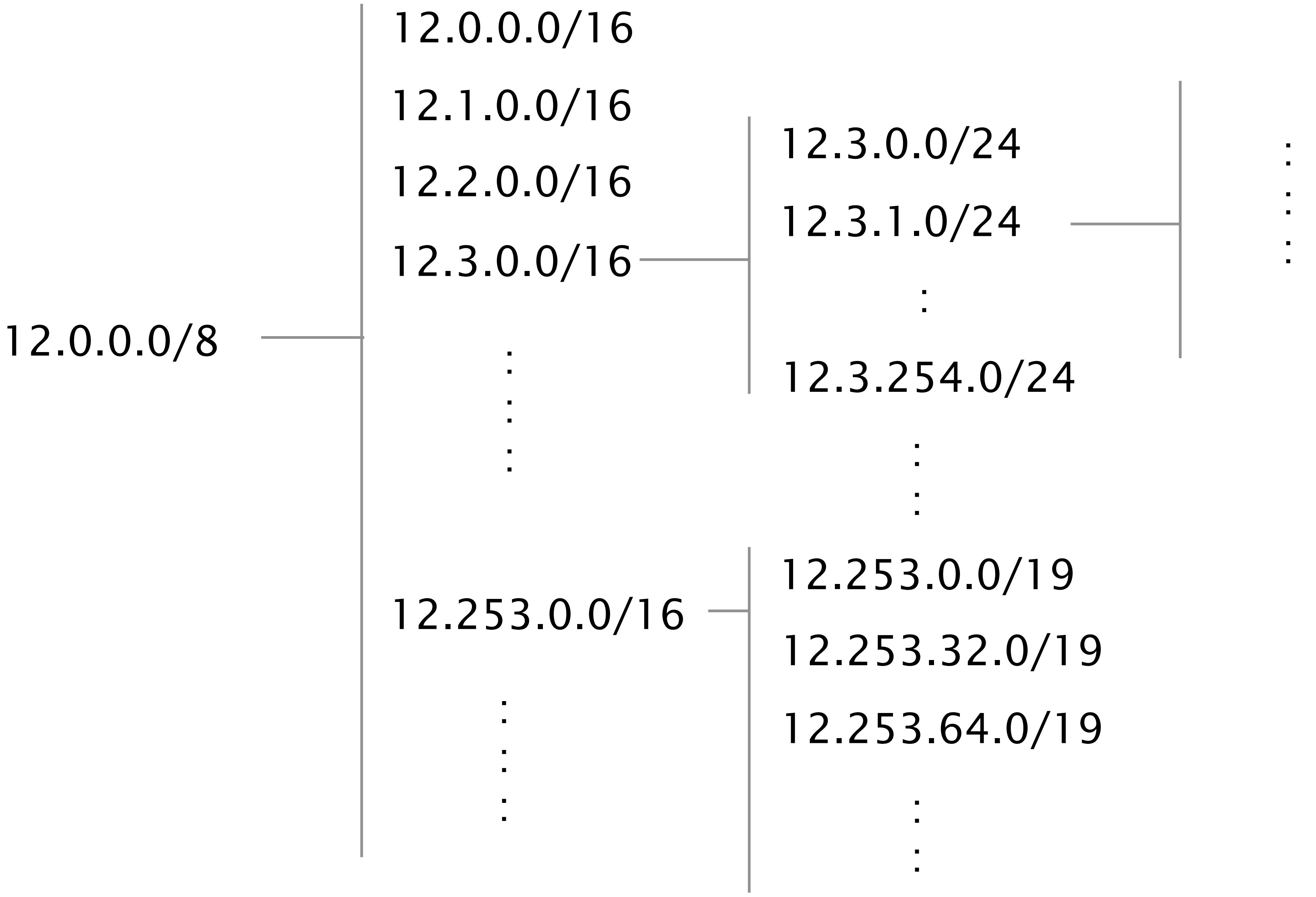
it is *not* implicitly carried in the address

Say that an organization needs 500 addresses…

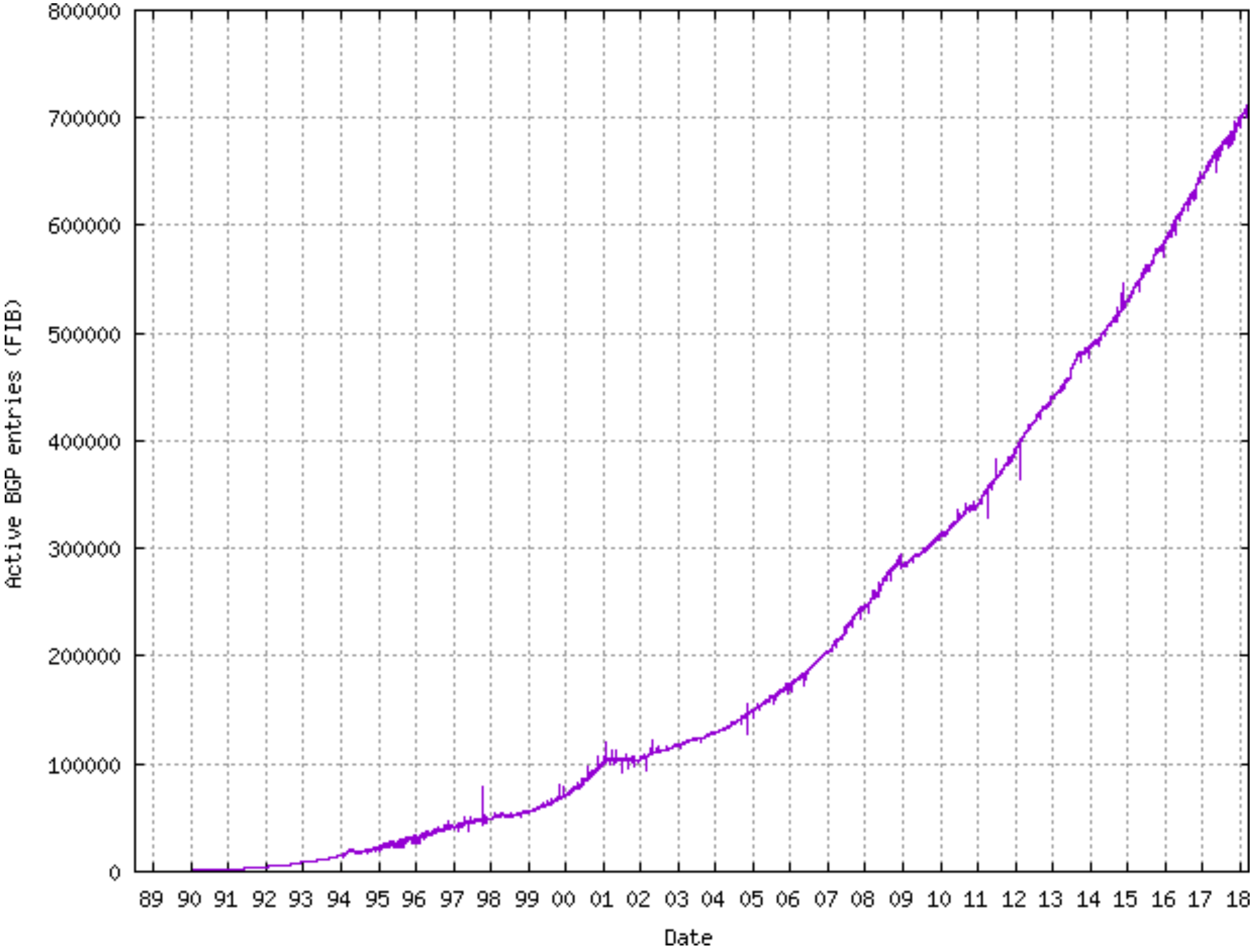
with…	it gets a…	leading to a waste of…
classful	class B (/16)	99%
CIDR	/23 (=2 class C’s)	2%

With CIDR, the max. waste is bounded to 50% (why?)

Today, addresses are allocated in contiguous chunks



As of now,
the Internet has around 710,000 IPv4 prefixes

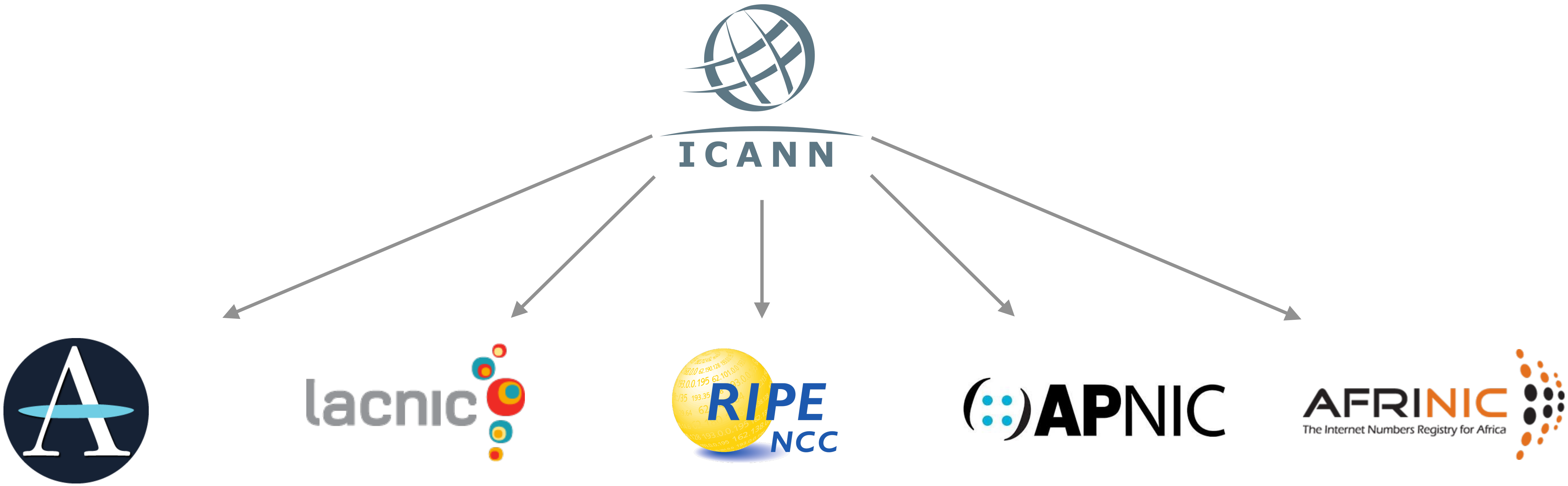


source http://www.cidr-report.org/

The allocation process of IP address is also hierarchical

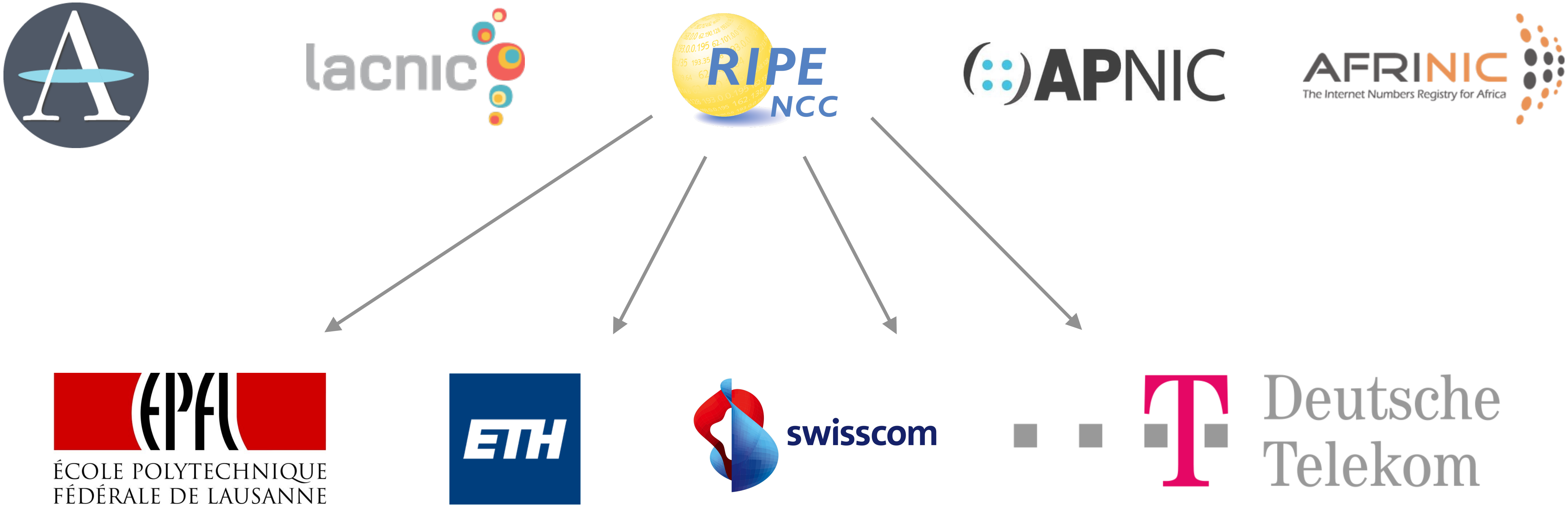
The root is held by Internet Corporation for Assigned Names and Numbers, aka ICANN

ICANN allocates large prefixes blocks to Regional Internet Registries (RIRs)

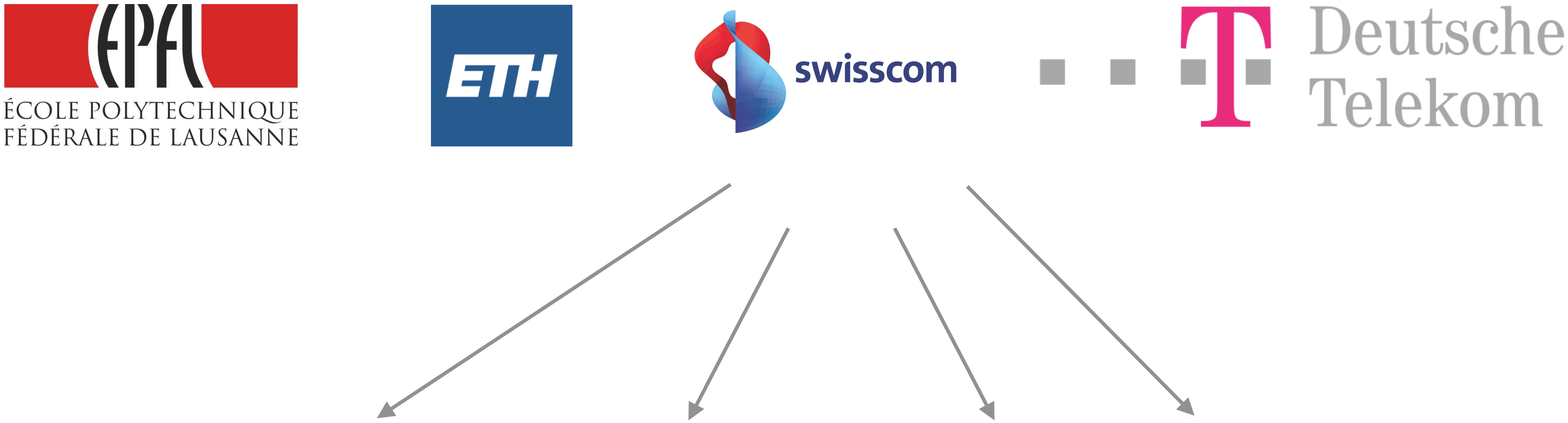


ARIN	LACNIC	RIPE NCC	APNIC	AFRINIC
America	Latin America	Europe	Asia-Pacific	Africa

RIRs allocates parts of these prefixes blocks to Internet Service Providers (ISPs) and large institutions


A
lacnic
RIPE NCC
APNIC
AFRINIC
The Internet Numbers Registry for Africa
ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE
EPFL
ETH
swisscom
T
Deutsche Telekom

ISPs and large institutions may, in turn,
allocate even smaller prefixes to their own customers



ICANN gives RIPE	82.0.0.0/8
Prefix	01010010
RIPE gives ETHZ	82.130.64.0/18
Prefix	010100101000001001
ETHZ gives ITET/TIK	82.130.102.0/23
Prefix	01010010100000100110011
ITET gives me	82.130.102.254
Address	01010010100000100110011011111110

IP prefixes @ ETH

1 82.130.64.0/18

2 129.132.0.0/16

3 148.187.192.0/19

4 195.176.96.0/19

5 192.33.87.0/24

6 192.33.88.0/21

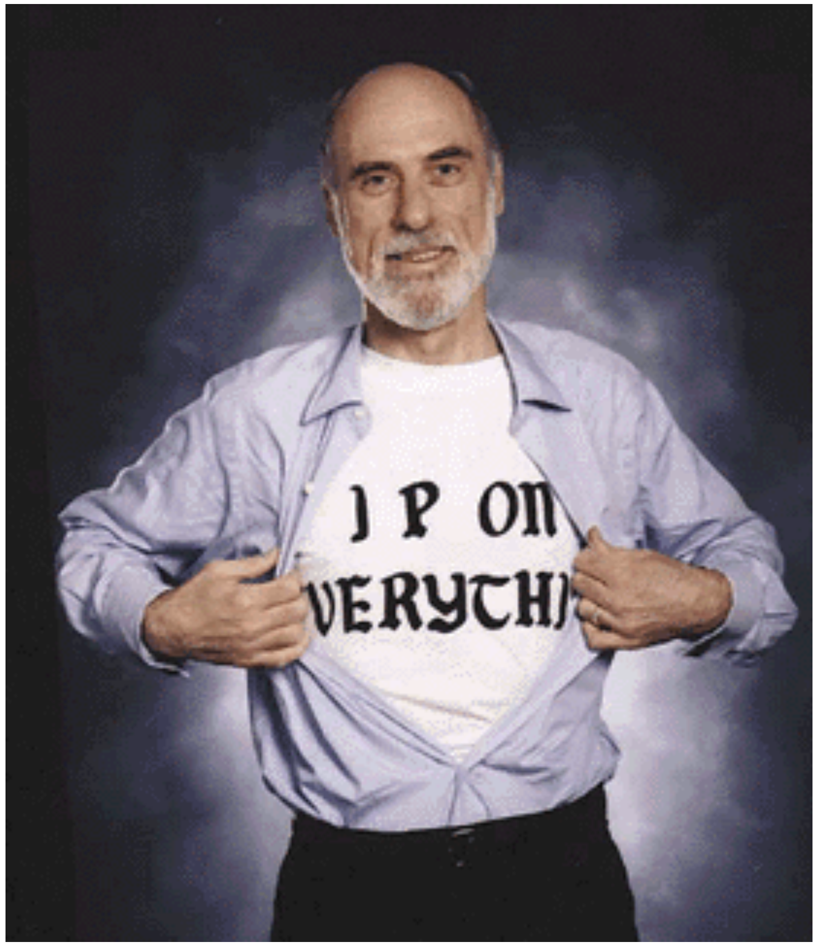
7 192.33.96.0/21

8 192.33.104.0/22

9 192.33.108.0/23

10 192.33.110.0/24

Internet Protocol and Forwarding



IP addresses

use, structure, allocation

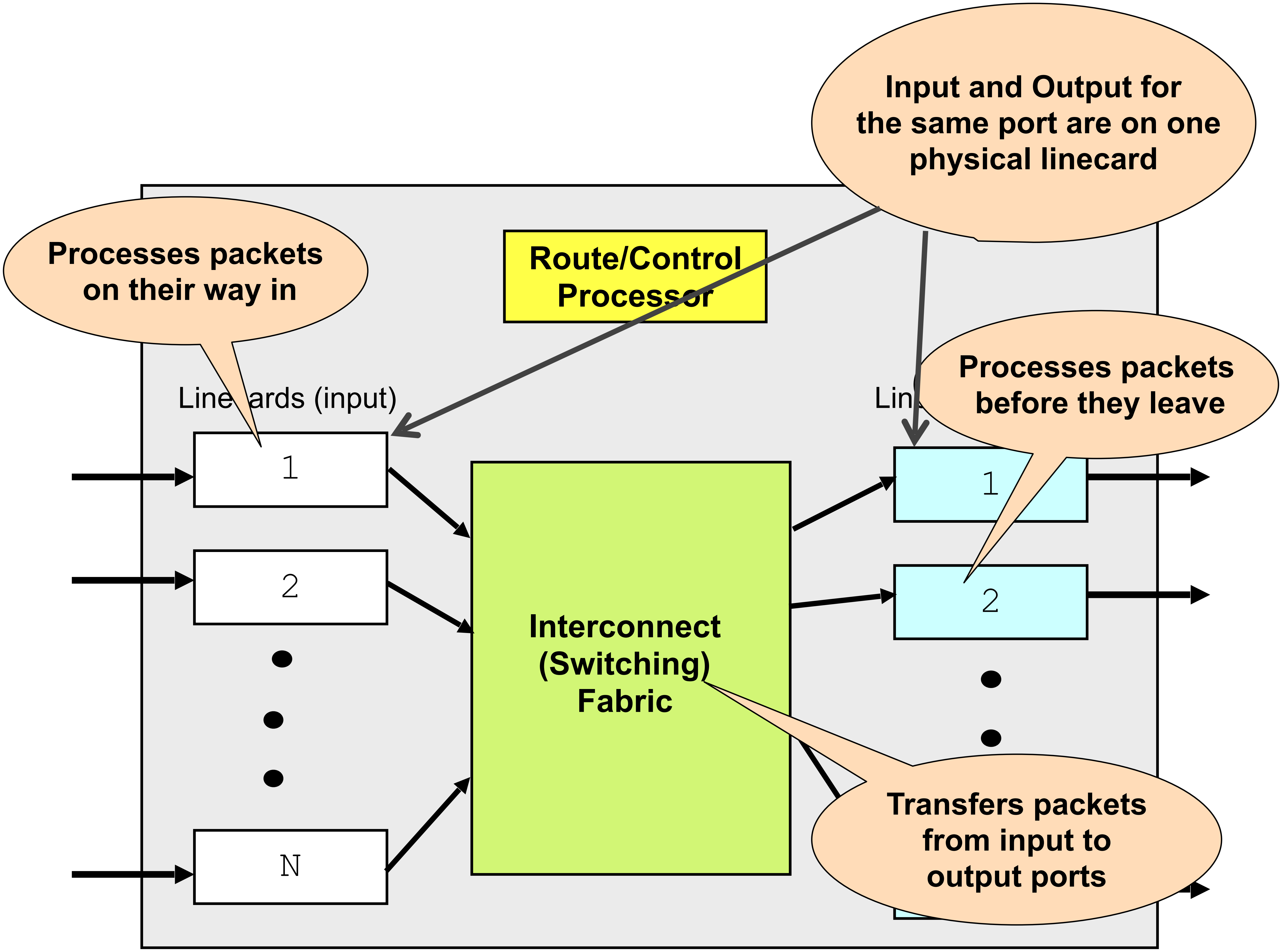
2 IP forwarding

longest prefix match rule

IP header

IPv4 and IPv6, wire format

What’s inside an IP router?



Input and Output for the same port are on one physical linecard

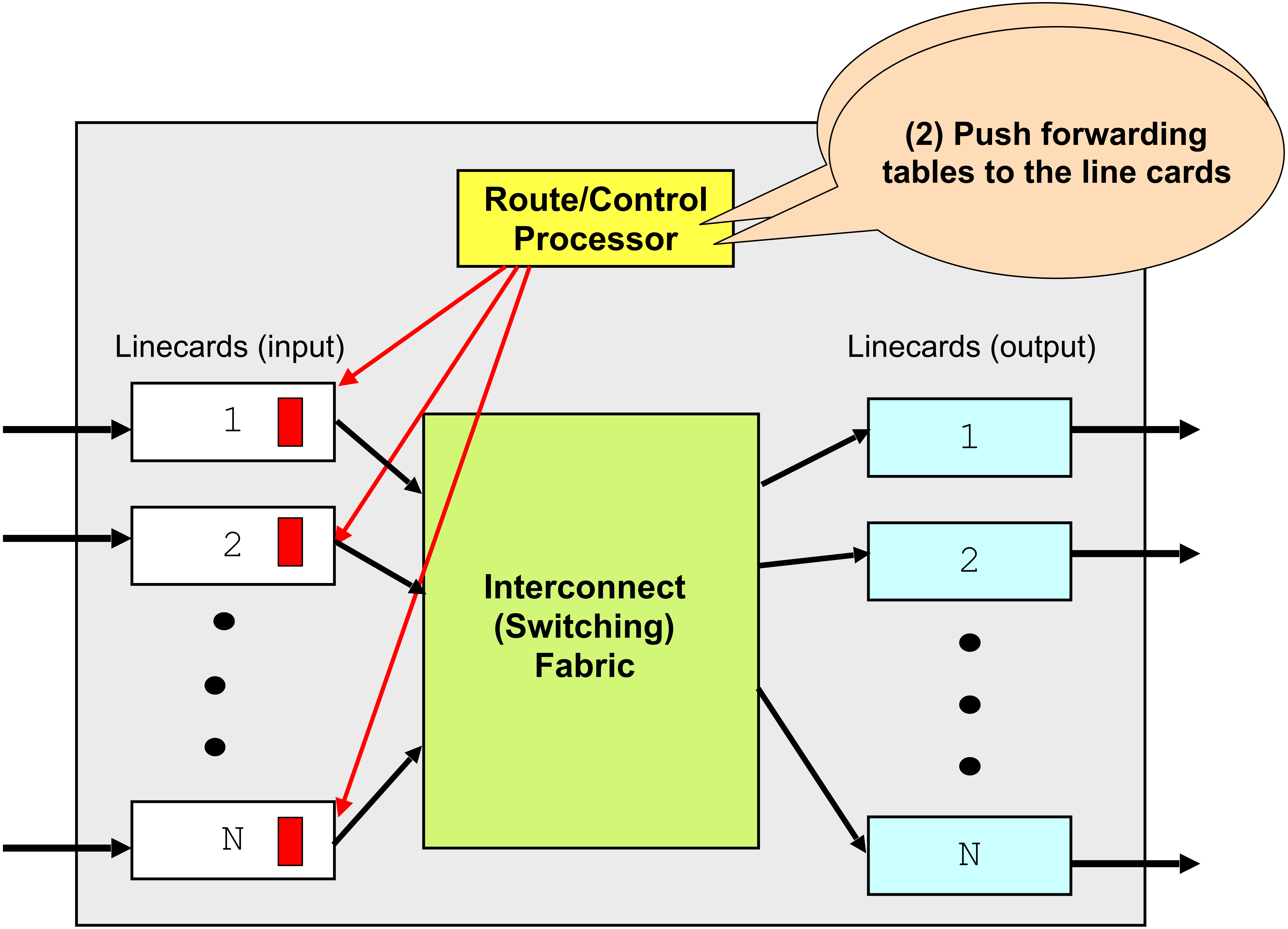
Processes packets on their way in

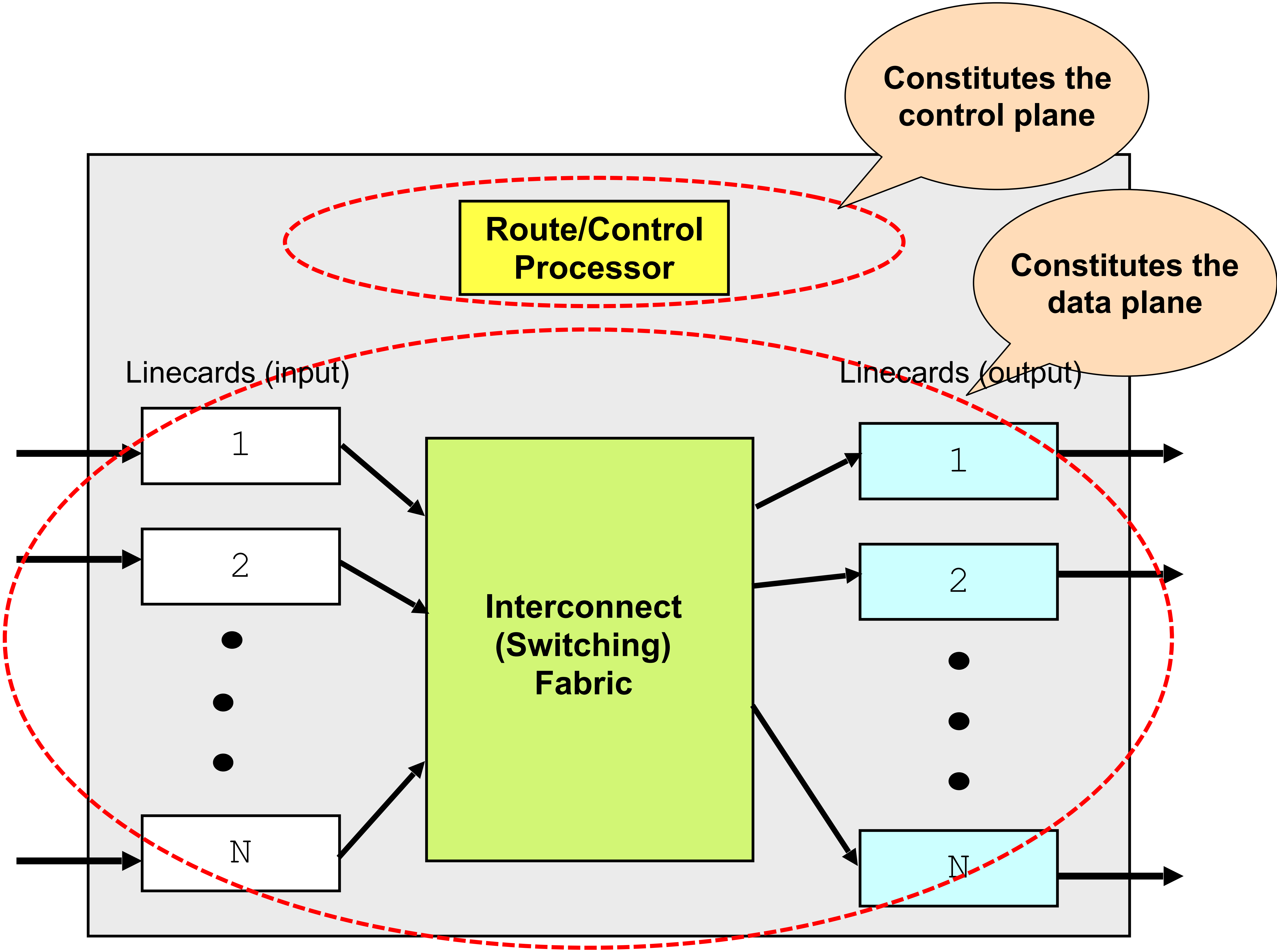
Route/Control Processor

Processes packets before they leave

Interconnect (Switching) Fabric

Transfers packets from input to output ports

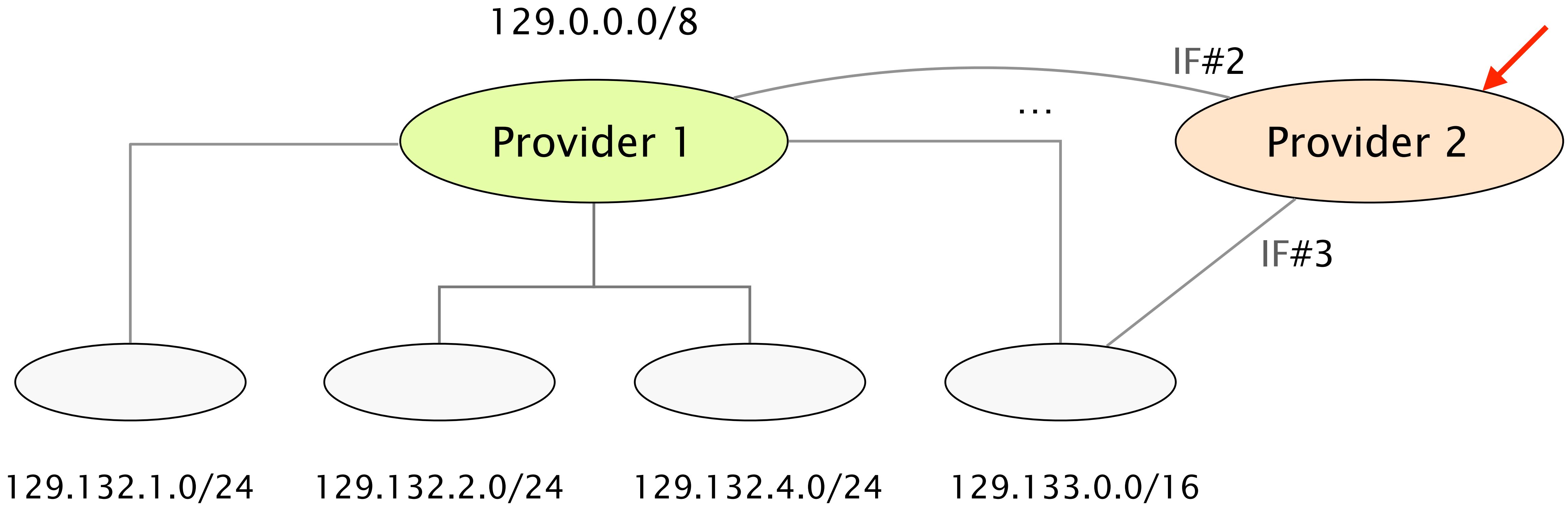

(2) Push forwarding tables to the line cards
Route/Control Processor
Linecards (input)
Linecards (output)
1
2
N
Interconnect (Switching) Fabric
1
2
N


Constitutes the control plane
Constitutes the data plane
Route/Control Processor
Linecards (input)
Linecards (output)
1
2
N
Interconnect (Switching) Fabric
1
2
N

Routers maintain forwarding entries for each Internet prefix

Provider 2’s Forwarding table

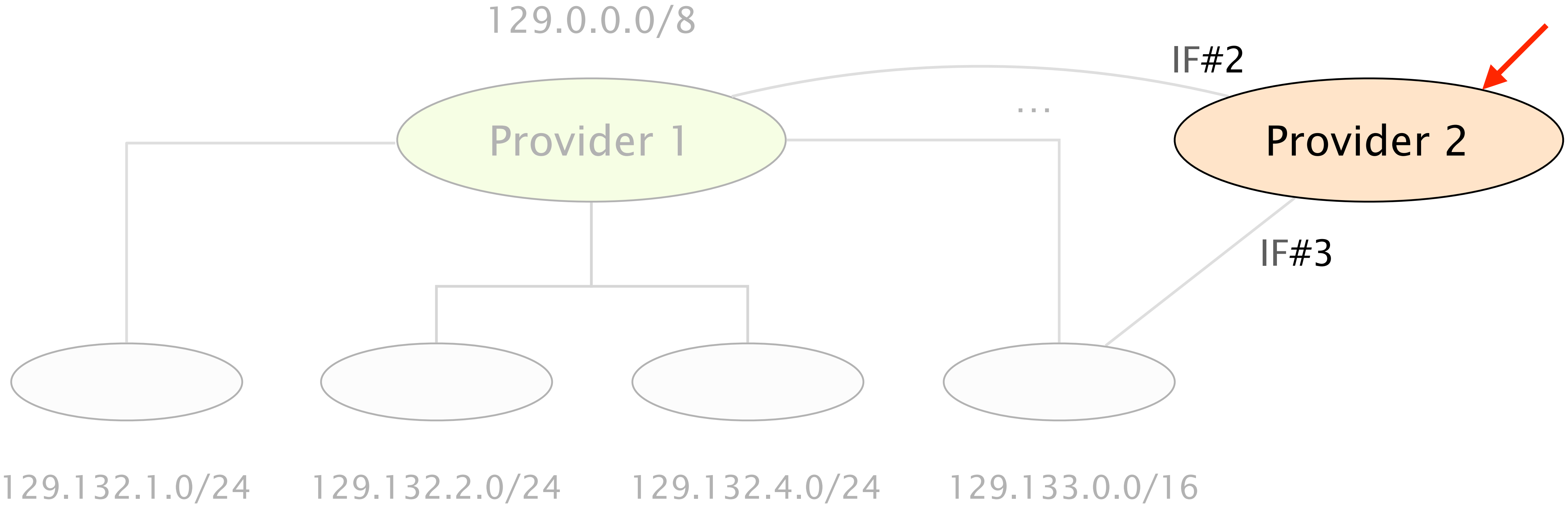
IP prefix	Output
129.0.0.0/8	IF#2
129.132.1.0/24	IF#2
129.132.2.0/24	IF#2
129.133.0.0/16	IF#3



Let’s say a packet for 129.0.1.1 arrives at Provider 2

Provider 2’s Forwarding table

IP prefix	Output
129.0.0.0/8	IF#2
129.132.1.0/24	IF#2
129.132.2.0/24	IF#2
129.133.0.0/16	IF#3



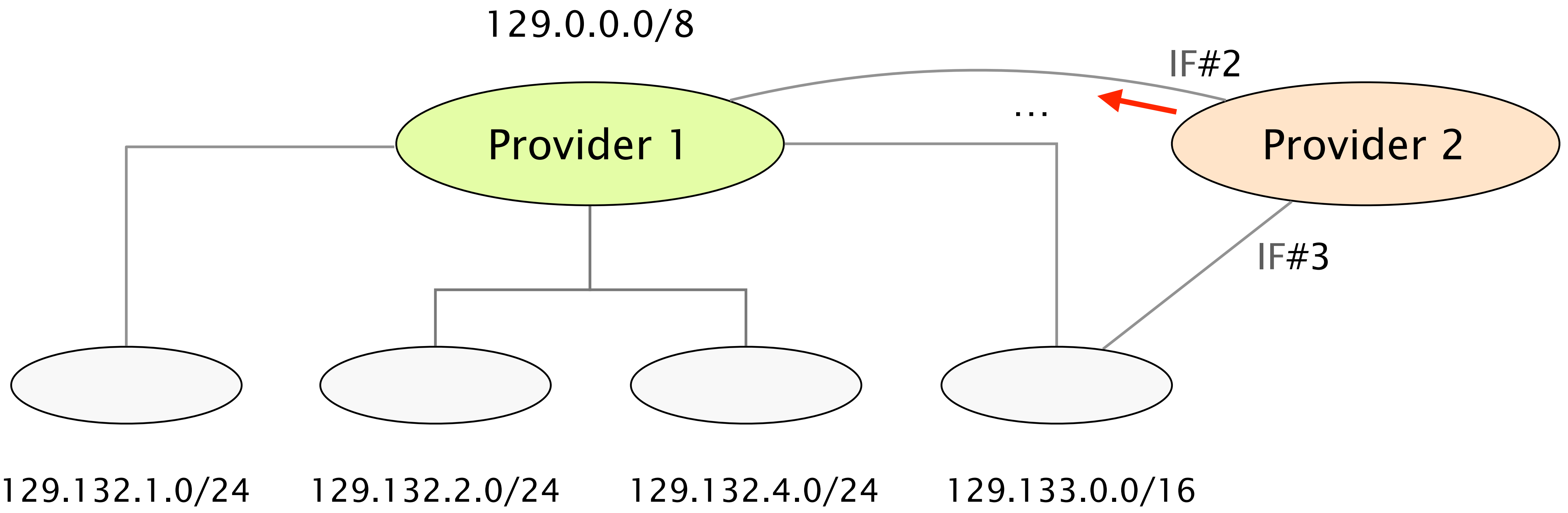
When a router receives an IP packet, it performs an IP lookup to find the matching prefix

Let’s say a packet for 129.0.1.1 arrives at Provider 2

> Provider 2 forwards it to IF#2

Provider 2’s Forwarding table

IP prefix	Output
129.0.0.0/8	IF#2
129.132.1.0/24	IF#2
129.132.2.0/24	IF#2
129.133.0.0/16	IF#3

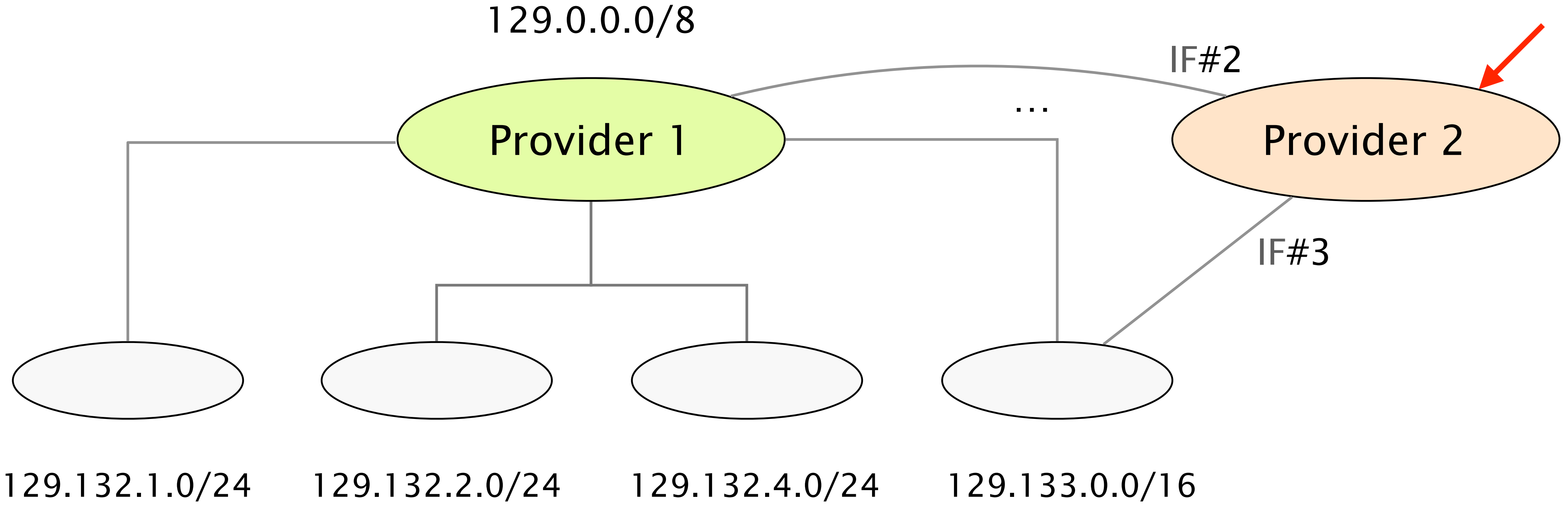


CIDR makes forwarding harder though,
as one packet can match many IP prefixes

Let’s say a packet for 129.133.0.1 arrives at Provider 2

Provider 2’s Forwarding table

IP prefix	Output
129.0.0.0/8	IF#2
129.132.1.0/24	IF#2
129.132.2.0/24	IF#2
129.133.0.0/16	IF#3

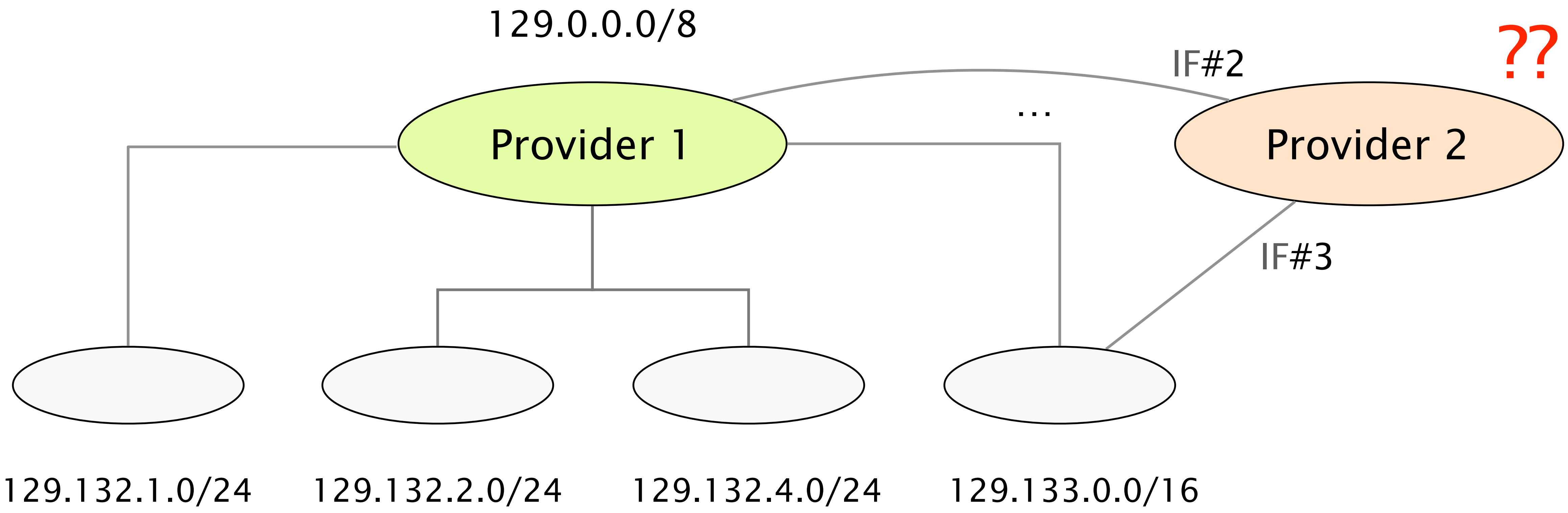


Let's say a packet for 129.133.0.1 arrives at Provider 2

We have two matches!

Provider 2's Forwarding table

IP prefix	Output
129.0.0.0/8	IF#2
129.132.1.0/24	IF#2
129.132.2.0/24	IF#2
129.133.0.0/16	IF#3



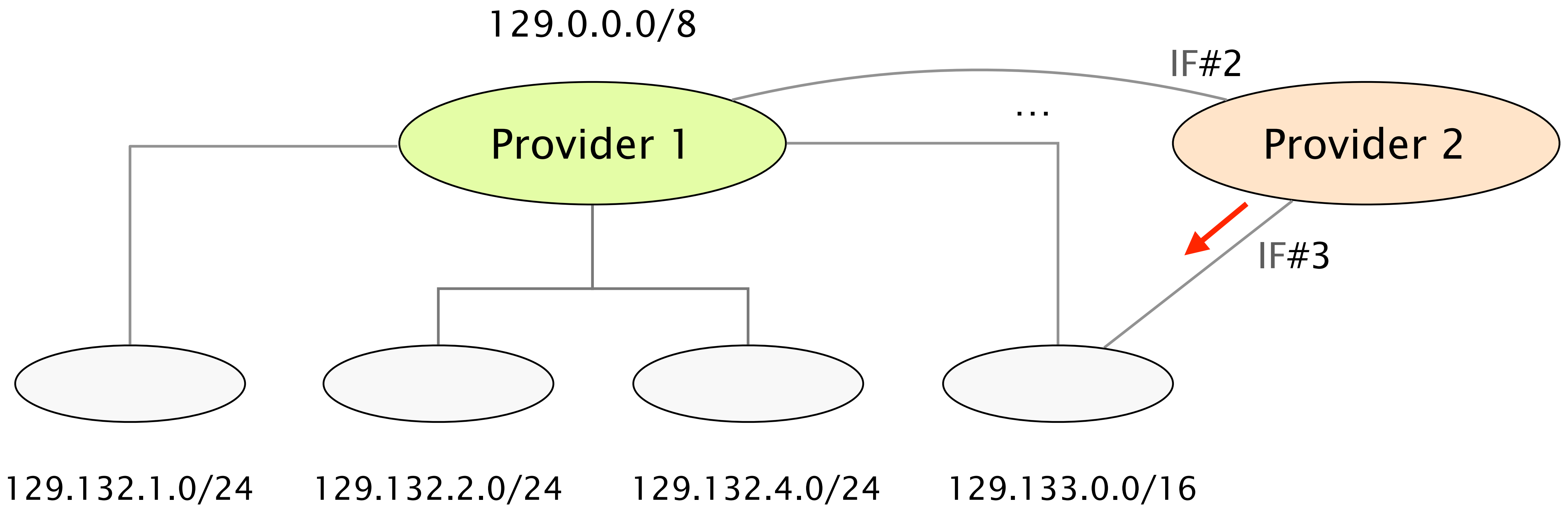
To resolve ambiguity, forwarding is done along the *most specific* prefix (*i.e.*, the longer one)

Let’s say a packet for 129.133.0.1 arrives at Provider 2

> Provider 2 forwards it to IF#3

Provider 2’s Forwarding table

IP prefix	Output
129.0.0.0/8	IF#2
129.132.1.0/24	IF#2
129.132.2.0/24	IF#2
129.133.0.0/16	IF#3

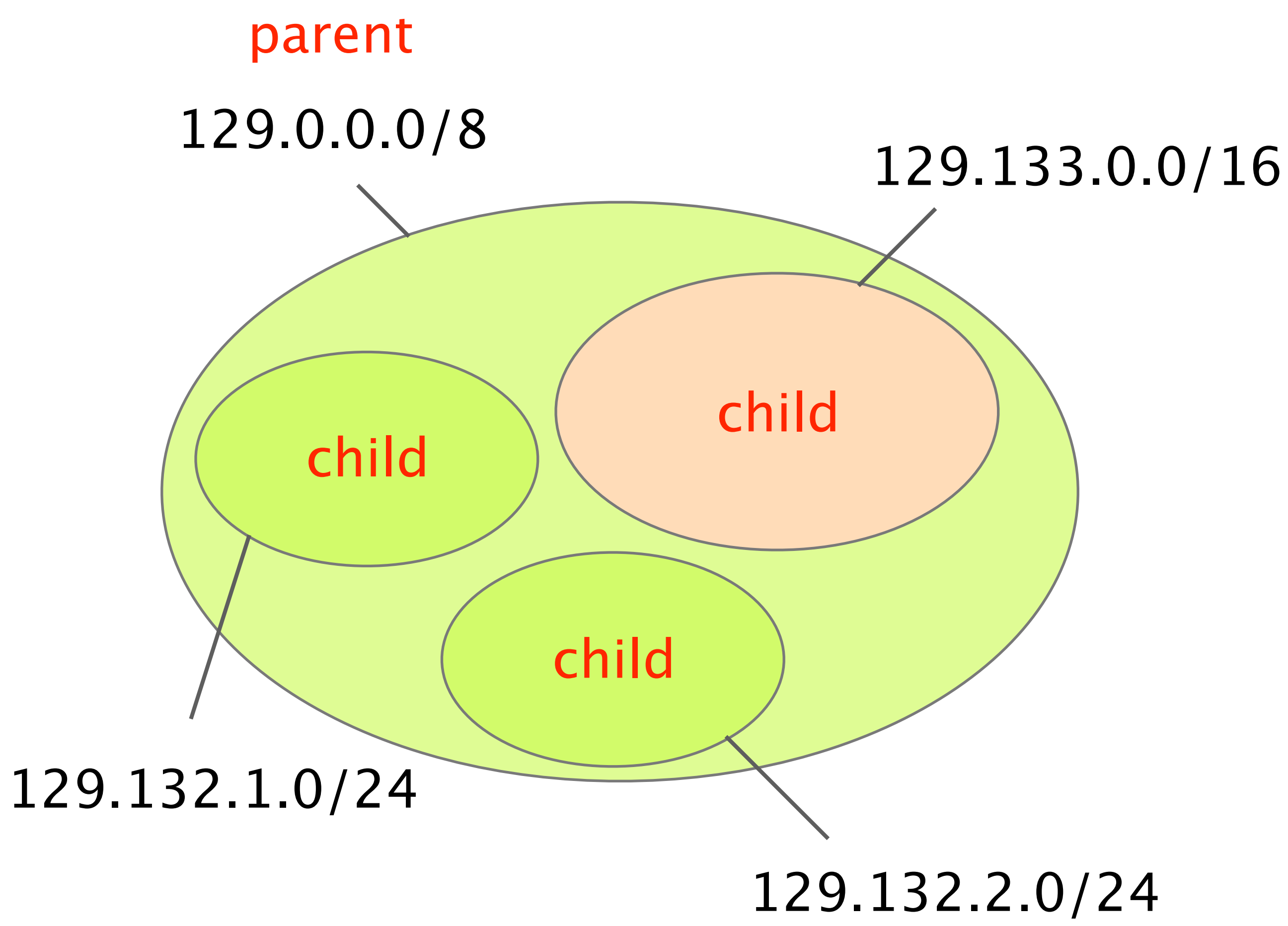


Could we do something better than maintaining one entry per prefix? *Yep!*

A child prefix can be filtered from the table whenever it shares the same output interface as its parent

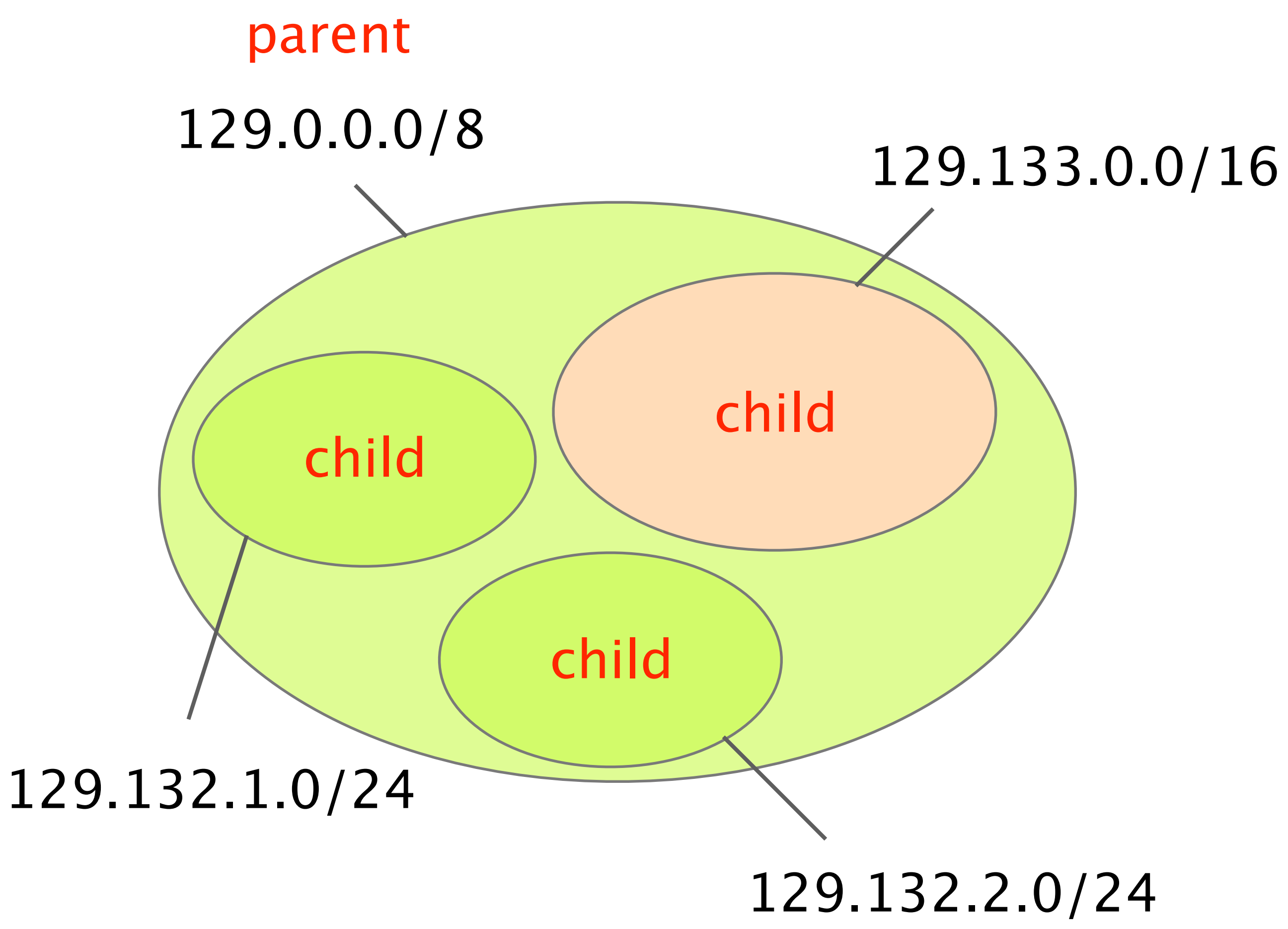
Routing Table

IP prefix	Output Interface
…	
129.0.0.0/8	IF#2
129.132.1.0/24	IF#2
129.132.2.0/24	IF#2
129.133.0.0/16	IF#3
…	



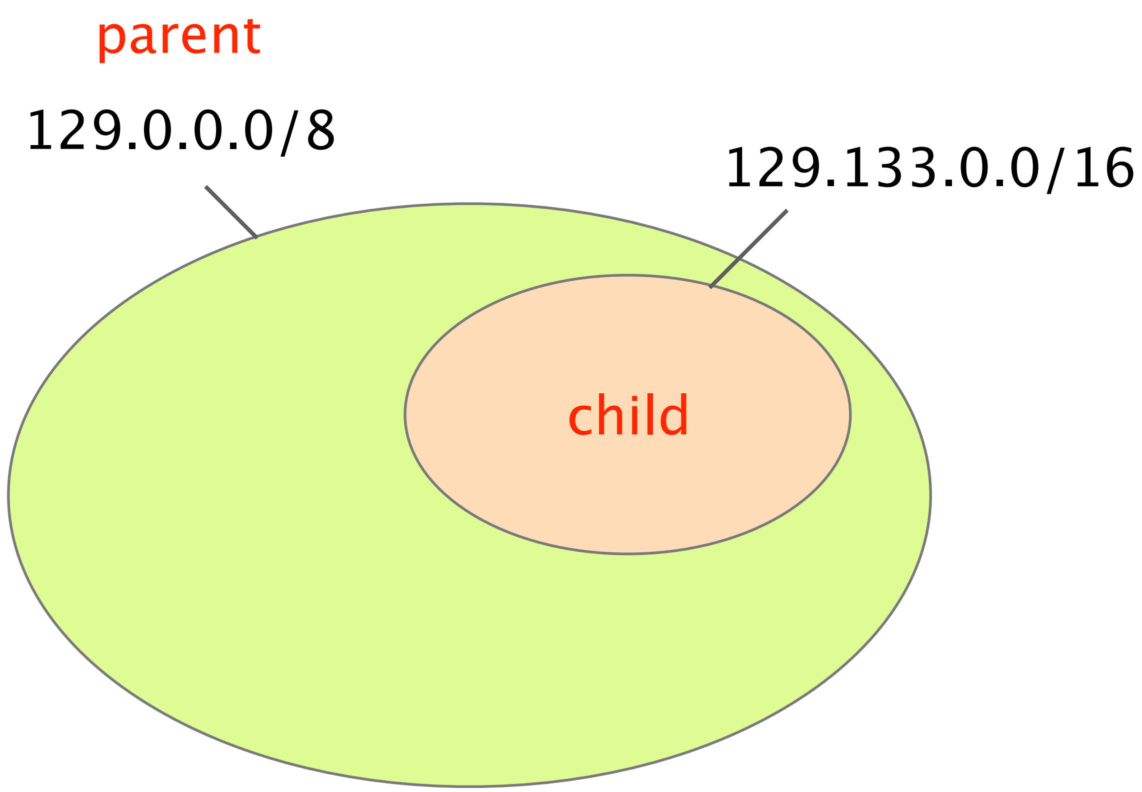
Routing Table

IP prefix	Output Interface
…	
129.0.0.0/8	IF#2
~~129.132.1.0/24~~	~~IF#2~~
~~129.132.2.0/24~~	~~IF#2~~
129.133.0.0/16	IF#3
…	



Routing Table

IP prefix	Output Interface
…	
129.0.0.0/8	IF#2
129.133.0.0/16	IF#3
…	



Exactly the same forwarding as before

Check out www.route-aggregation.net,
to see how filtering can be done automatically

Communication Networks

Spring 2020—Online Edition

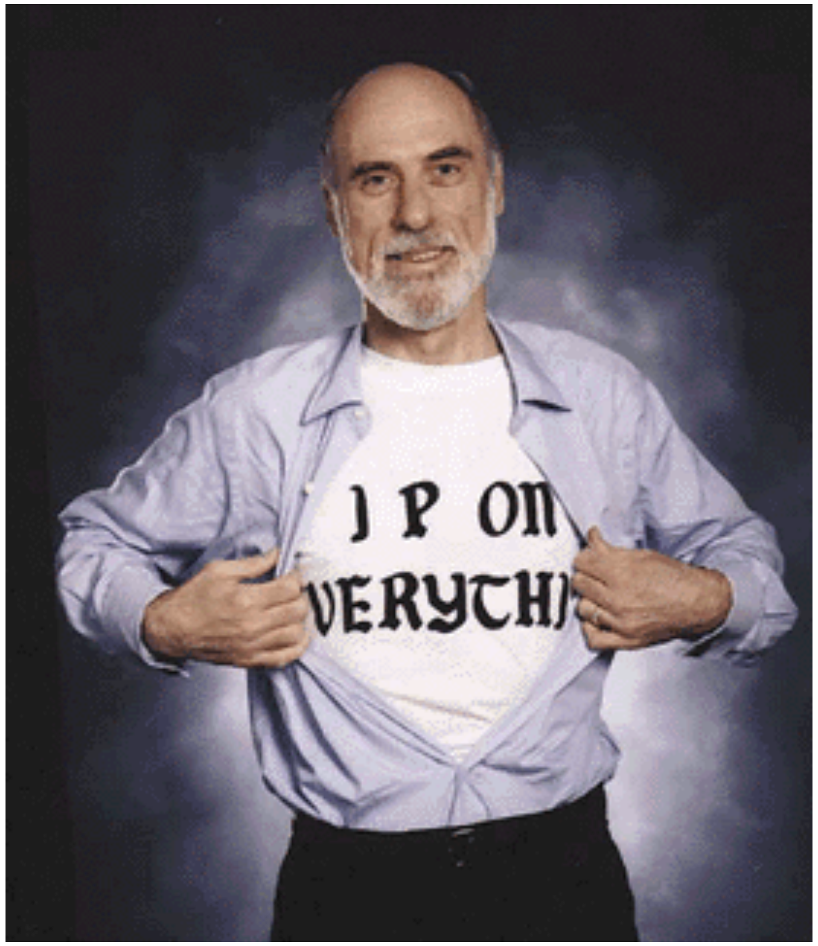
Laurent Vanbever

nsg.ee.ethz.ch

ETH Zürich (D-ITET)

March 16 2020

Internet Protocol and Forwarding



IP addresses

use, structure, allocation

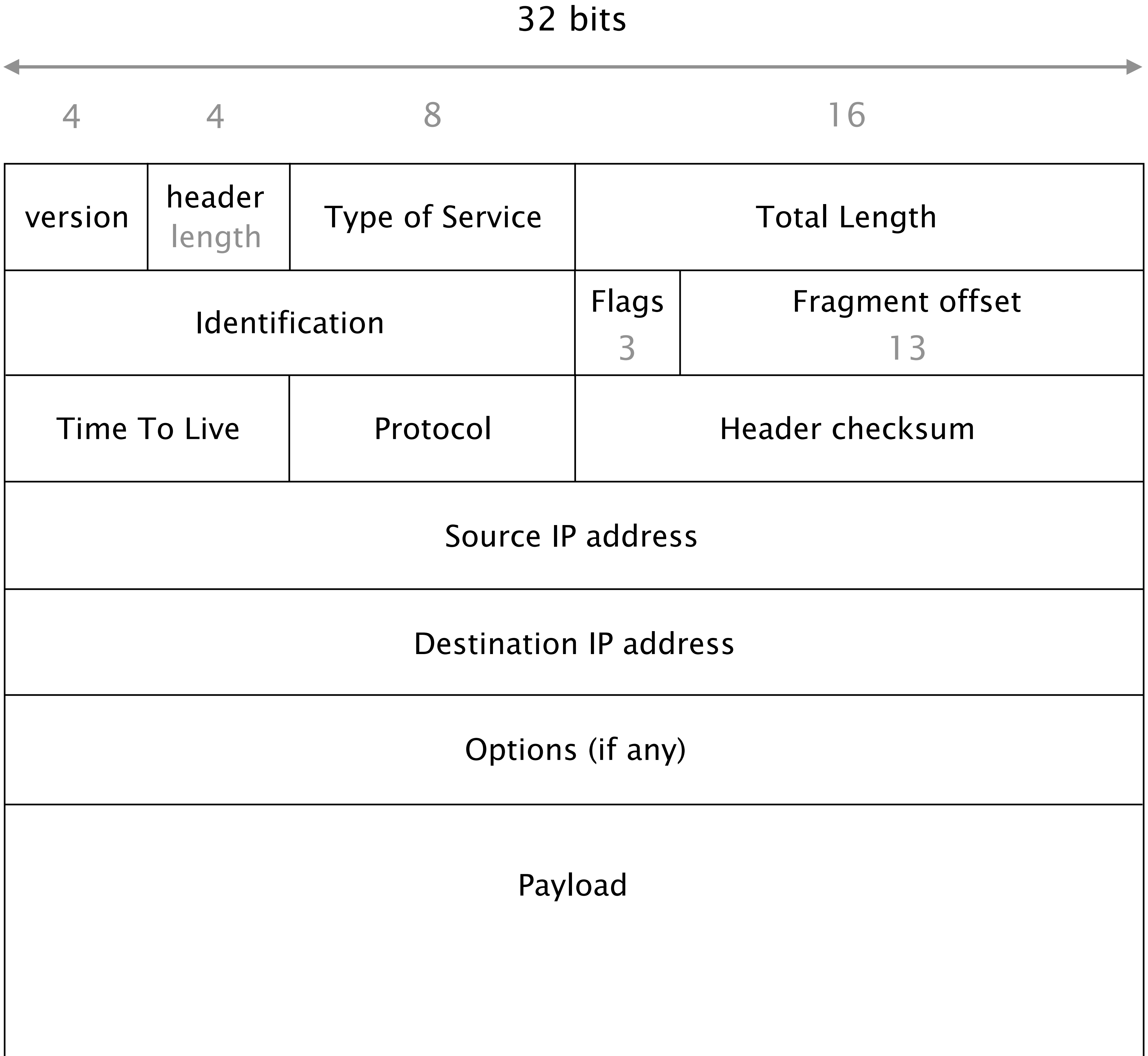
IP forwarding

longest prefix match rule

3 IP header

IPv4 and IPv6, wire format

Here is what an IPv4 packet look like on a wire



32 bits

4 4 8 16

version

header length

Type of Service

Total Length

Identification

Flags 3

Fragment offset 13

Time To Live

Protocol

Header checksum

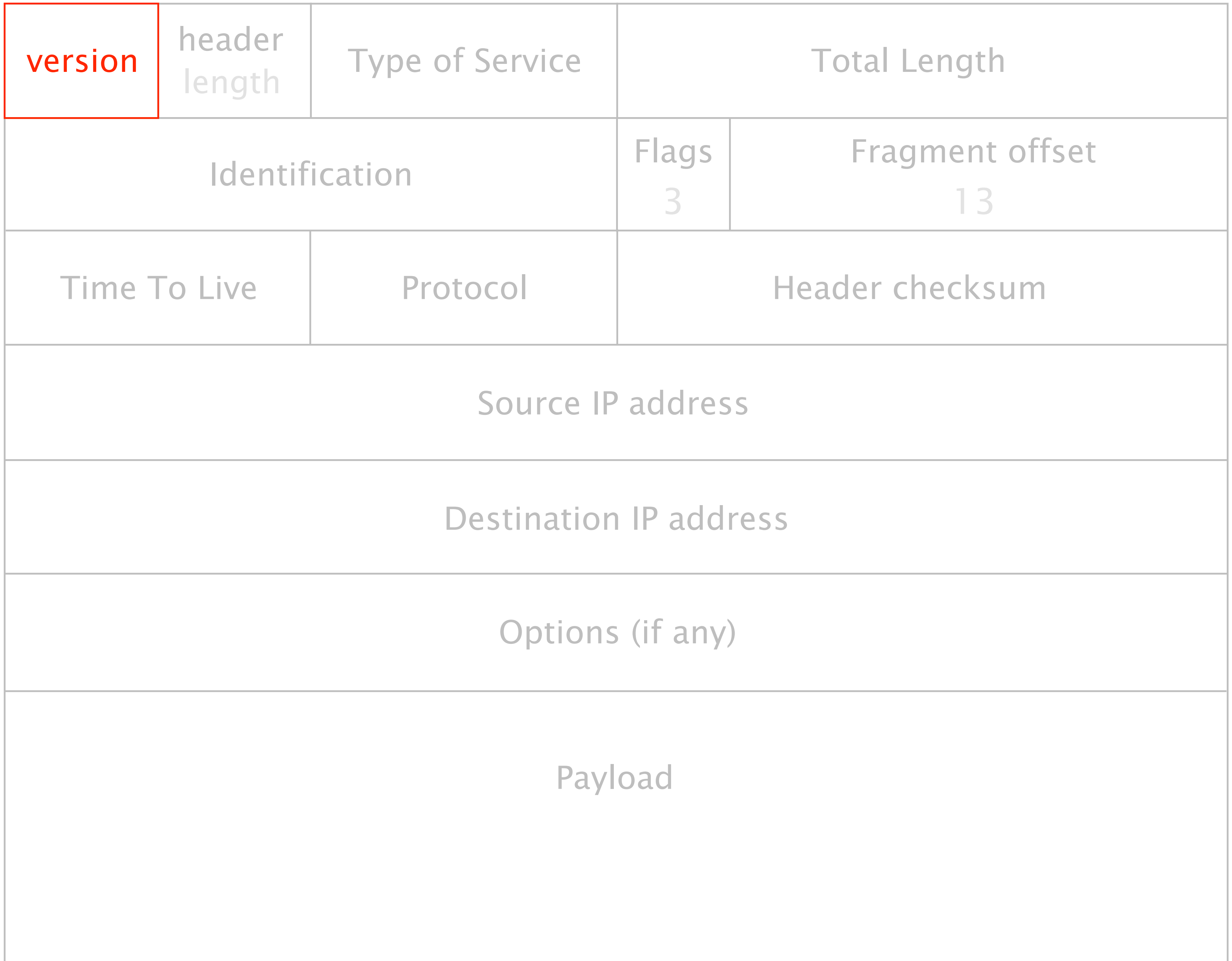
Source IP address

Destination IP address

Options (if any)

Payload

The version number tells us what other fields to expect, typically it is set to “4” for IPv4, or “6” for IPv6



version | header length | Type of Service | Total Length

Identification | Flags 3 | Fragment offset 13

Time To Live | Protocol | Header checksum

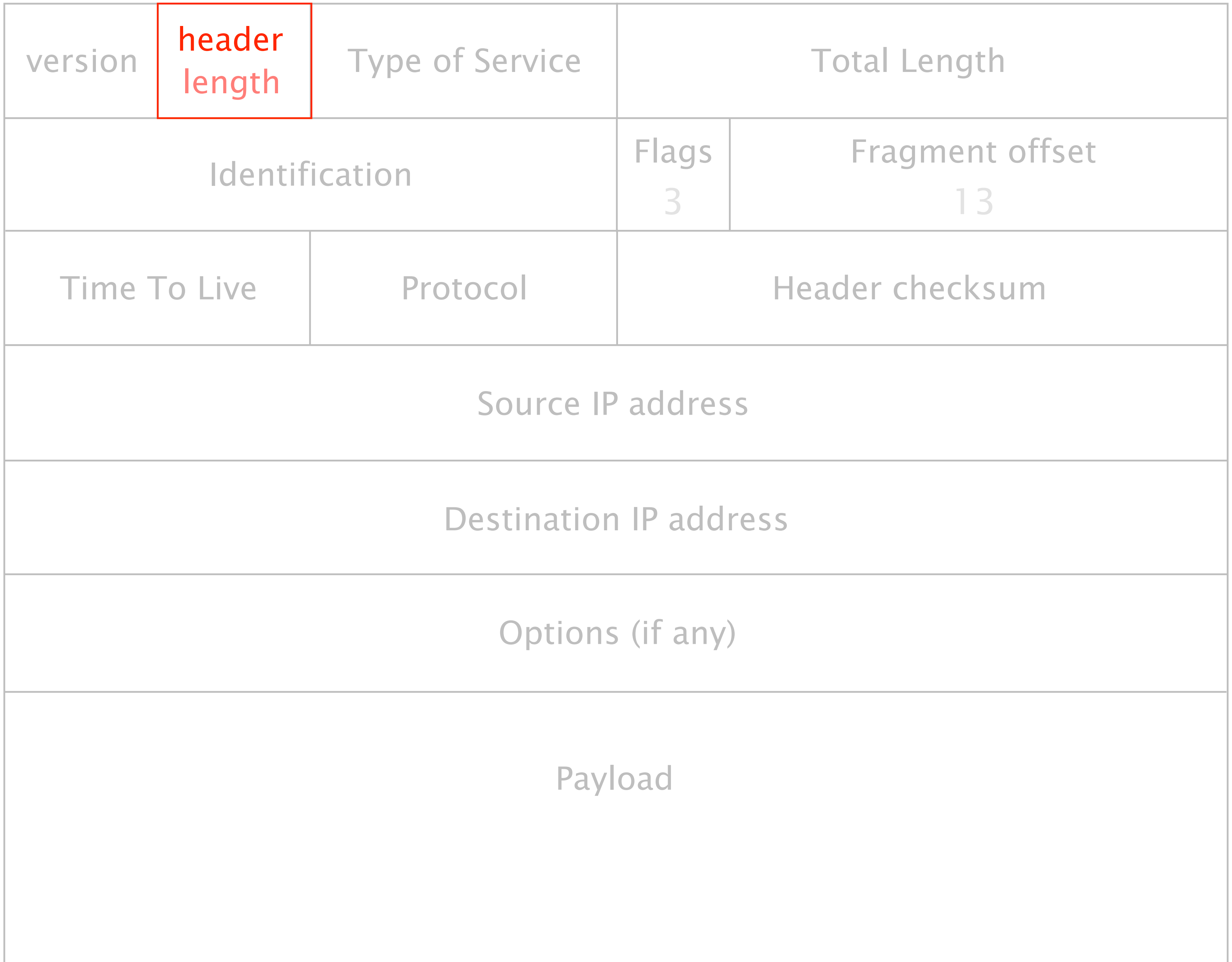
Source IP address

Destination IP address

Options (if any)

Payload

The header length denotes the number of 32-bits word in the header, typically set to 5 (20 bytes header)



version

header length

Type of Service

Total Length

Identification

Flags 3

Fragment offset 13

Time To Live

Protocol

Header checksum

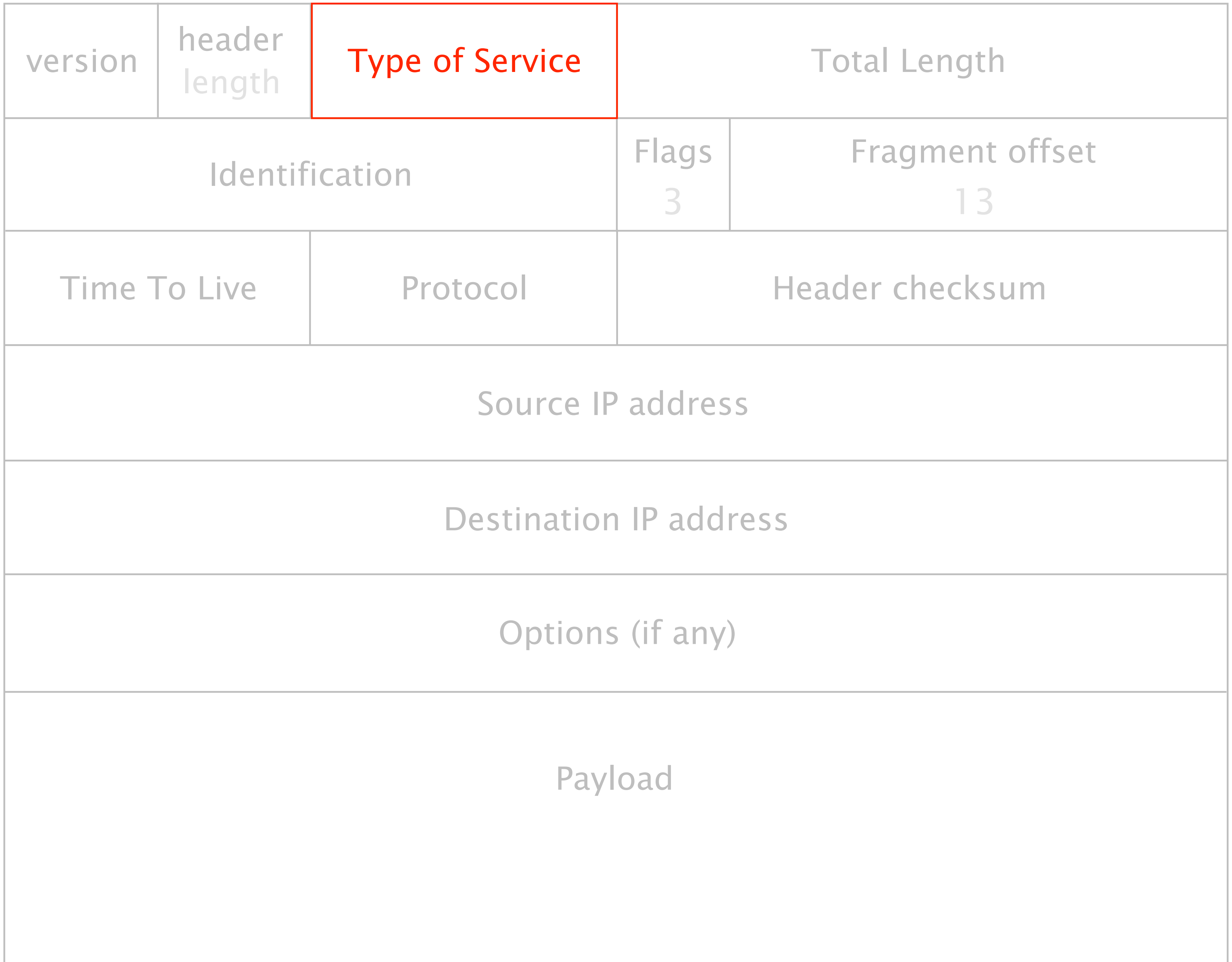
Source IP address

Destination IP address

Options (if any)

Payload

The ToS allows different packets to be treated differently, e.g., low delay for voice, high bandwidth for video



version

header length

Type of Service

Total Length

Identification

Flags 3

Fragment offset 13

Time To Live

Protocol

Header checksum

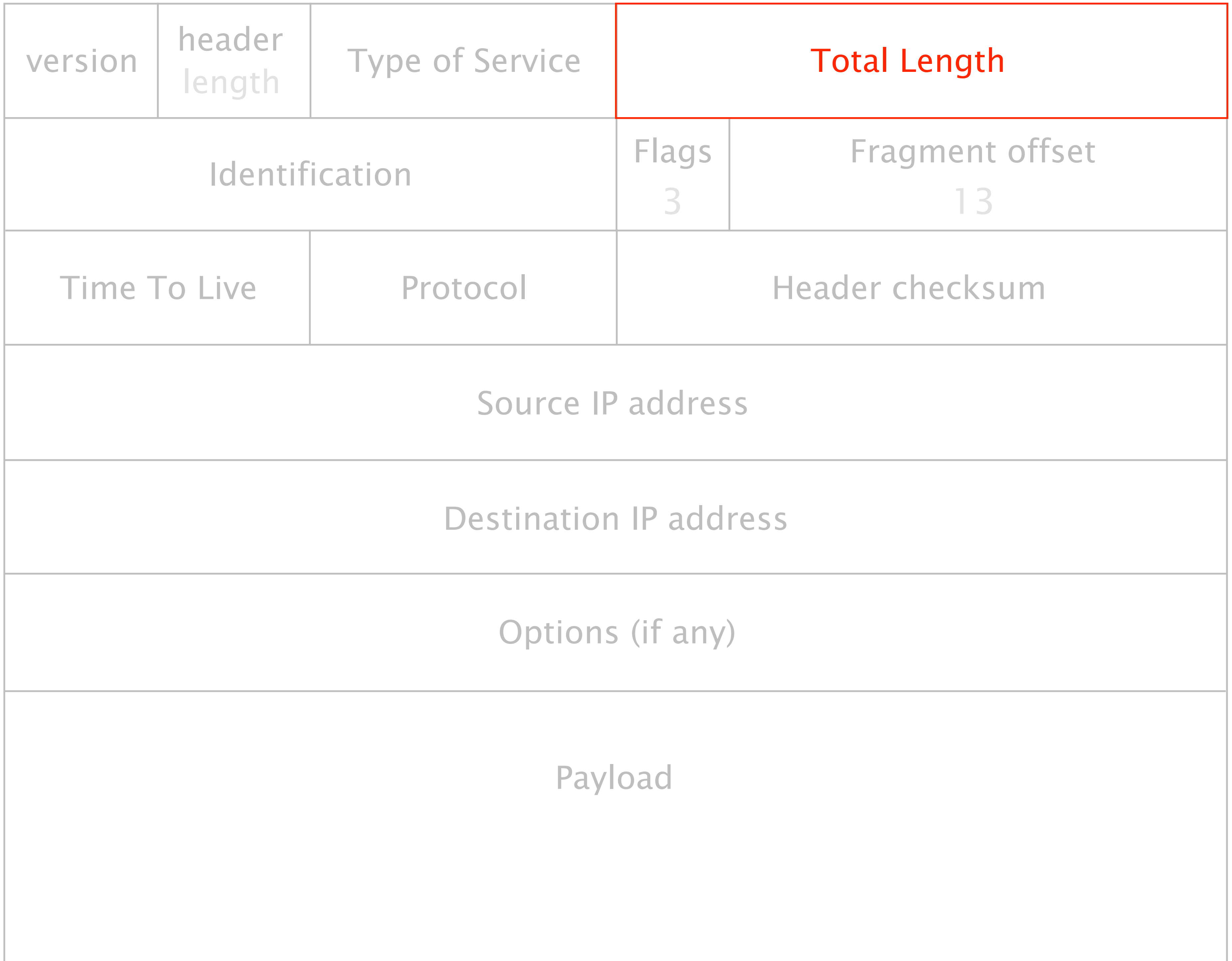
Source IP address

Destination IP address

Options (if any)

Payload

The total length denotes the # of bytes
in the entire packet, with a maximum of 65 535 bytes



version | header length | Type of Service | Total Length

Identification | Flags 3 | Fragment offset 13

Time To Live | Protocol | Header checksum

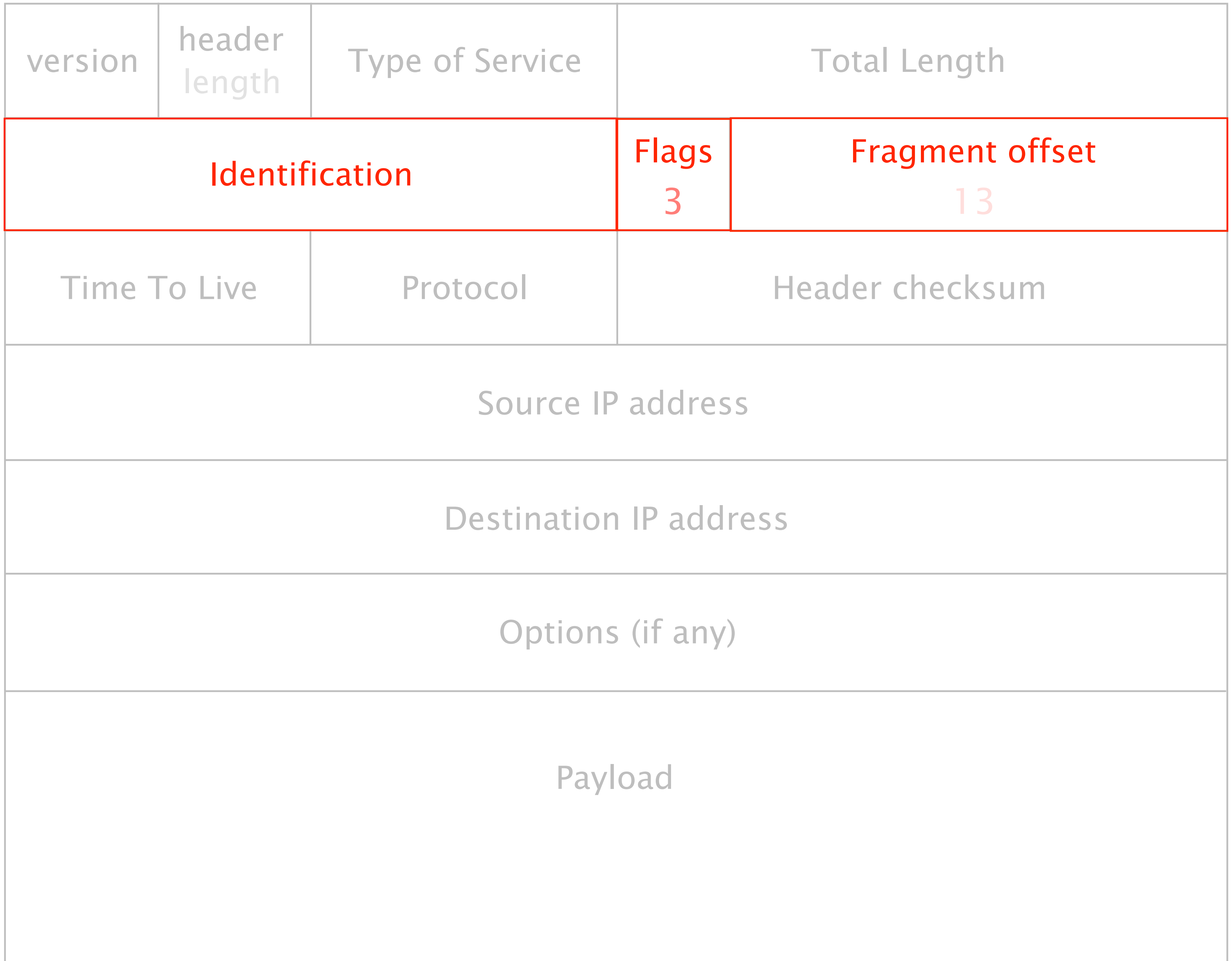
Source IP address

Destination IP address

Options (if any)

Payload

The next three fields are used when packets get fragmented



version

header length

Type of Service

Total Length

Identification

Flags 3

Fragment offset 13

Time To Live

Protocol

Header checksum

Source IP address

Destination IP address

Options (if any)

Payload

Every link in the Internet has a Maximum Transmission Unit (MTU)

MTU is the max. # of bytes a link can carry as one unit

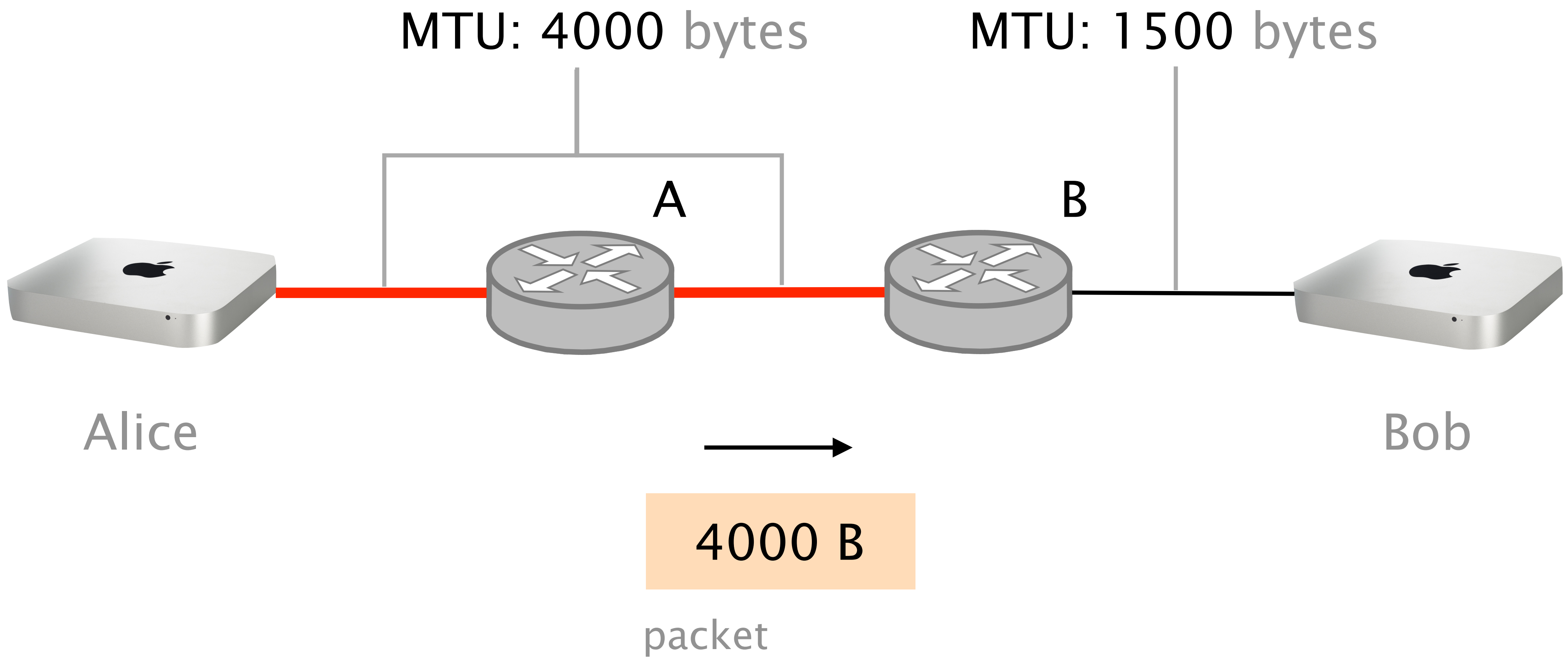
e.g., 1500 bytes for normal Ethernet

A router can fragment a packet if the outgoing link MTU is smaller than the total packet size

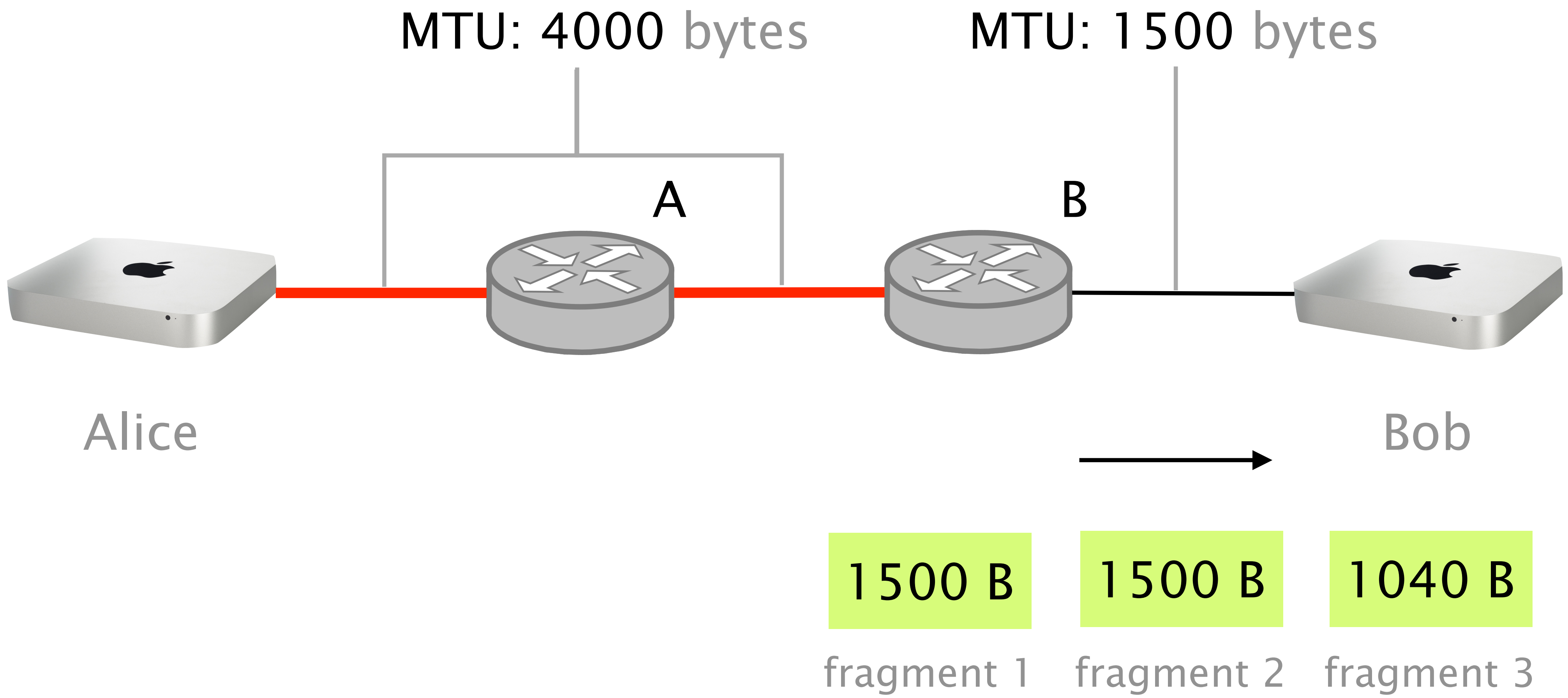
Fragmented packets are recomposed at the destination

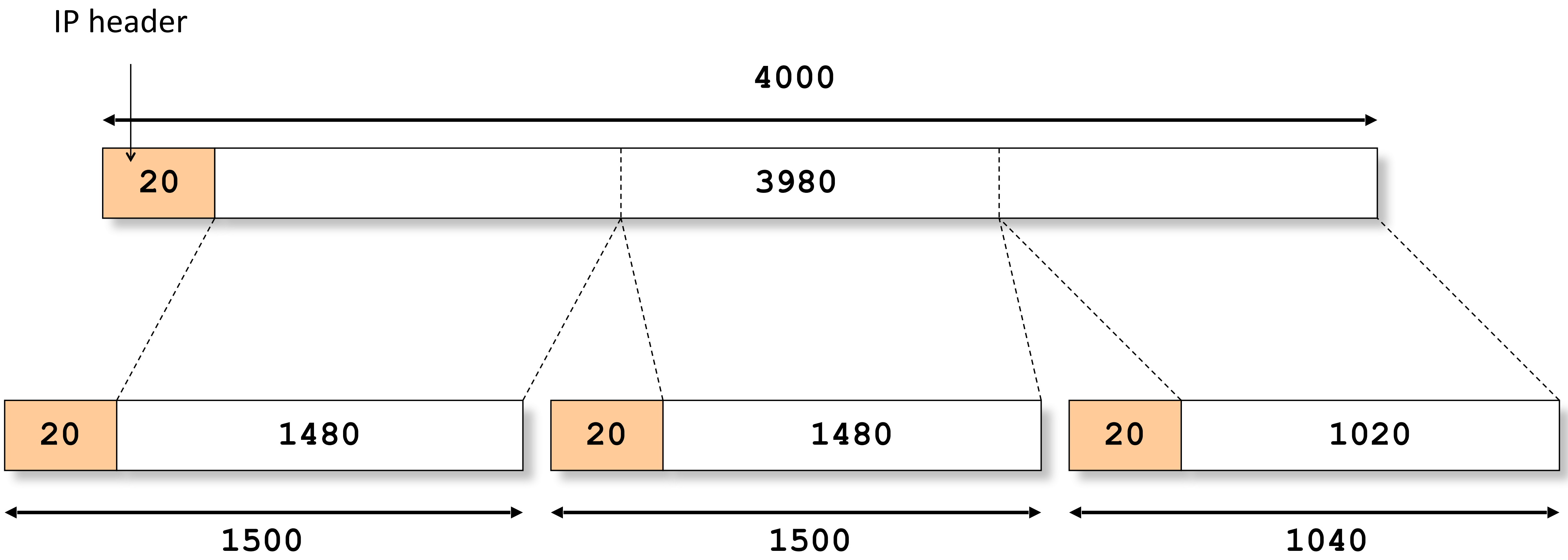
why not in the network?

Assume Alice is sending 4000B packets to Bob, who is connected to a 1500B MTU link

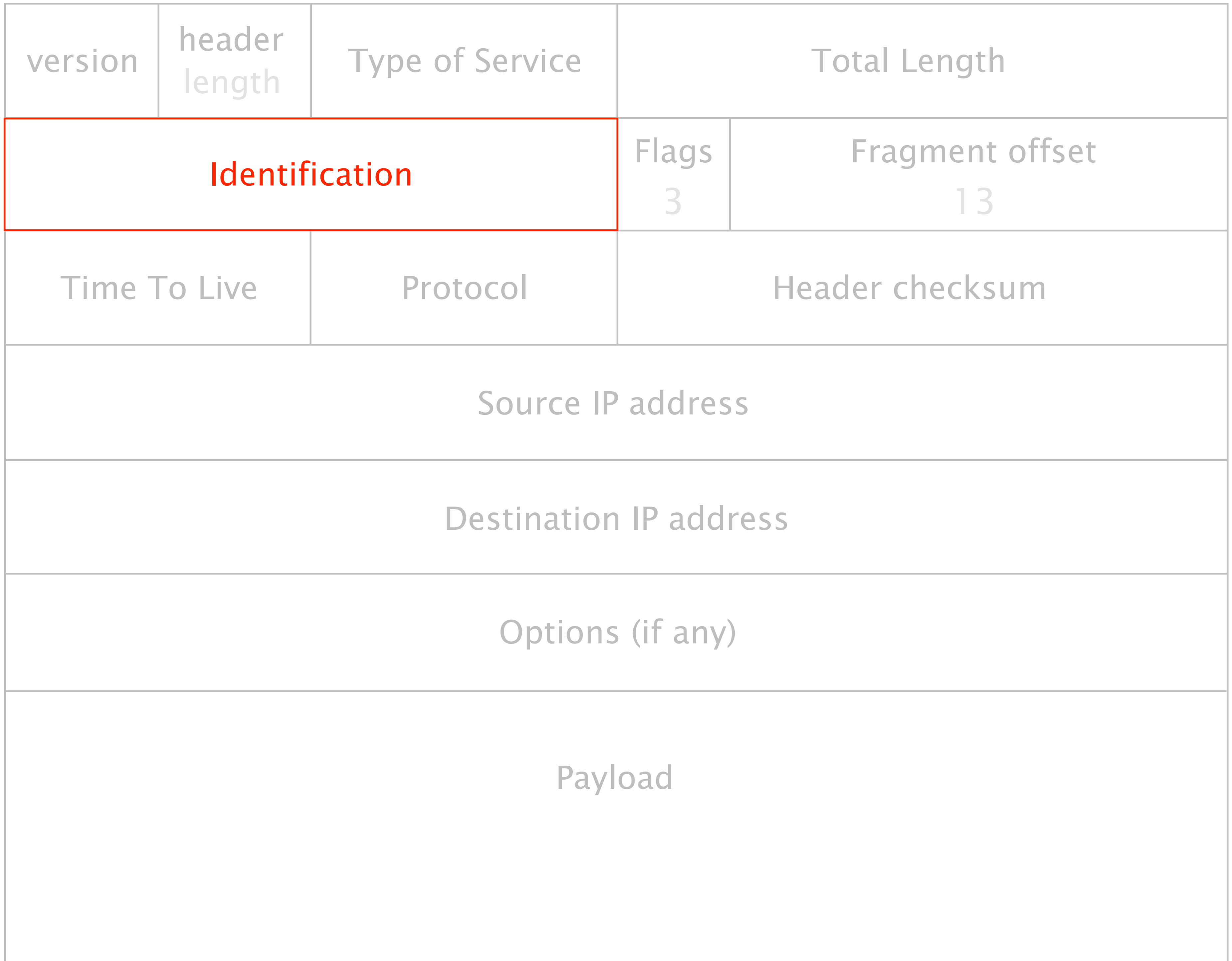


Because the packet is larger than the MTU, router B will split the packet into fragments




IP header
4000
20
3980
20
1480
20
1480
20
1020
1500
1500
1040

The Identification header uniquely identify the fragments of a particular packet



version | header length | Type of Service | Total Length

Identification | Flags 3 | Fragment offset 13

Time To Live | Protocol | Header checksum

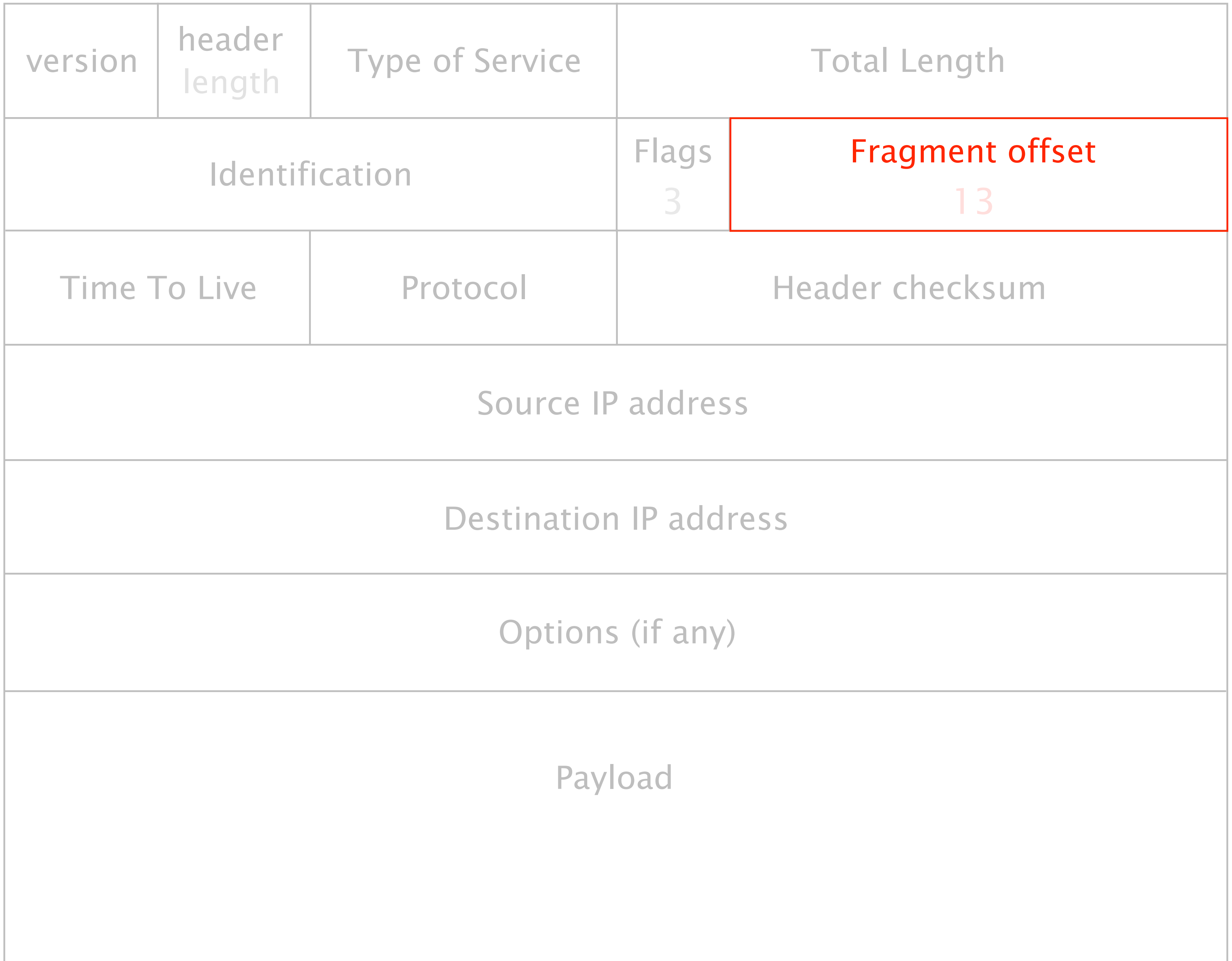
Source IP address

Destination IP address

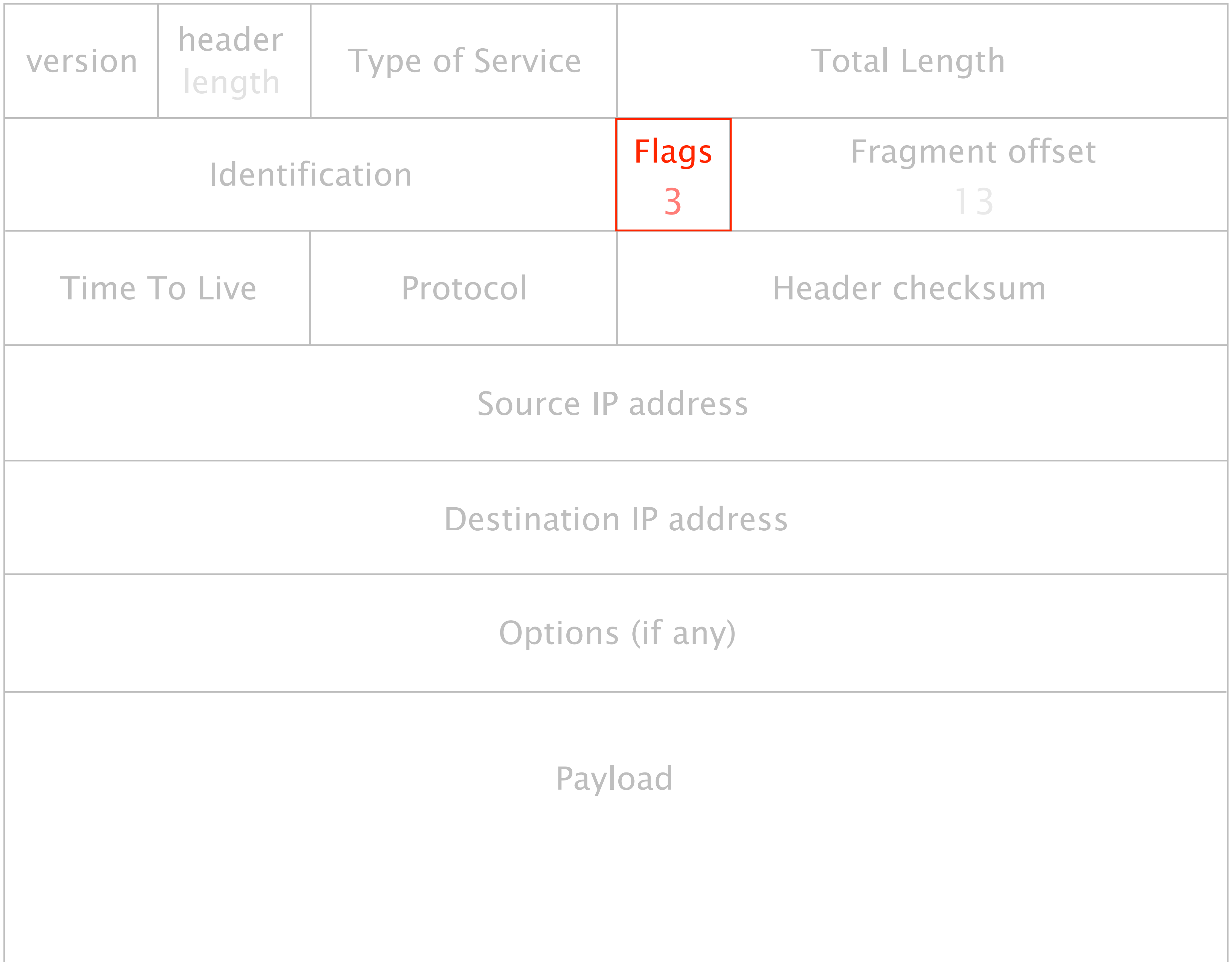
Options (if any)

Payload

The fragment offset is used to put back the fragments in the right order in case of reordering



The flags is used to tell whether there are more fragments coming or not



version

header length

Type of Service

Total Length

Identification

Flags 3

Fragment offset 13

Time To Live

Protocol

Header checksum

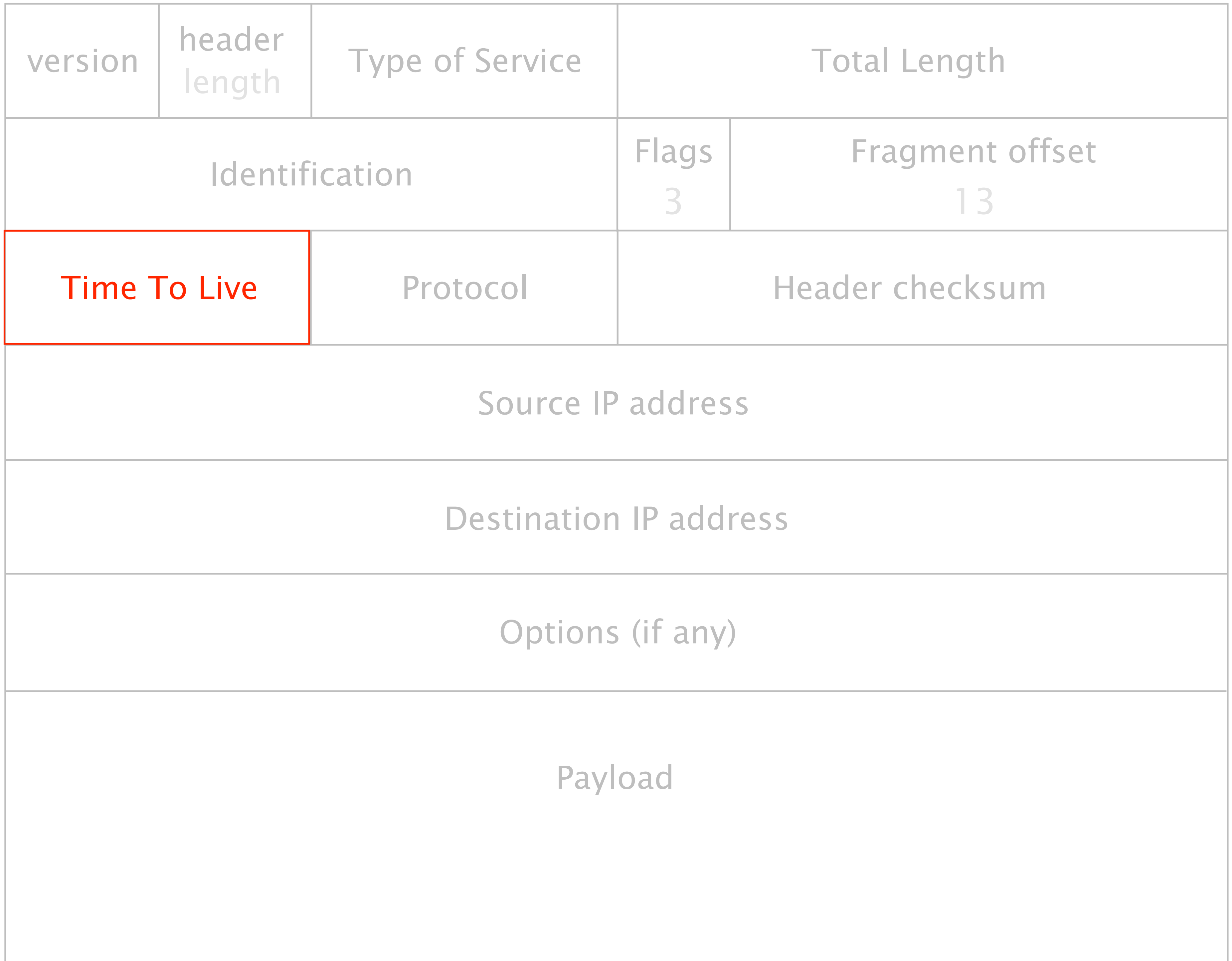
Source IP address

Destination IP address

Options (if any)

Payload

The TTL is used to identify packets trapped in a loop, and eventually discard them



version

header length

Type of Service

Total Length

Identification

Flags 3

Fragment offset 13

Time To Live

Protocol

Header checksum

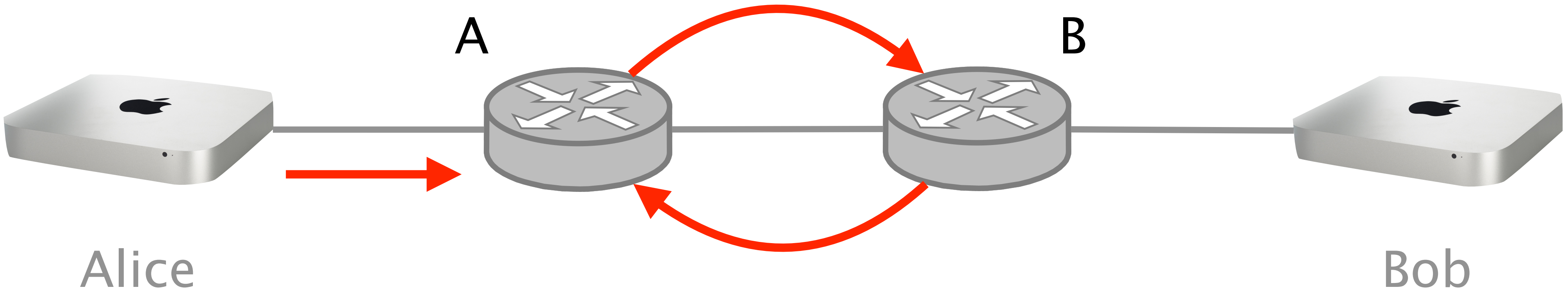
Source IP address

Destination IP address

Options (if any)

Payload

TTL is decremented by 1 at each router, the packet is discarded if it reaches 0

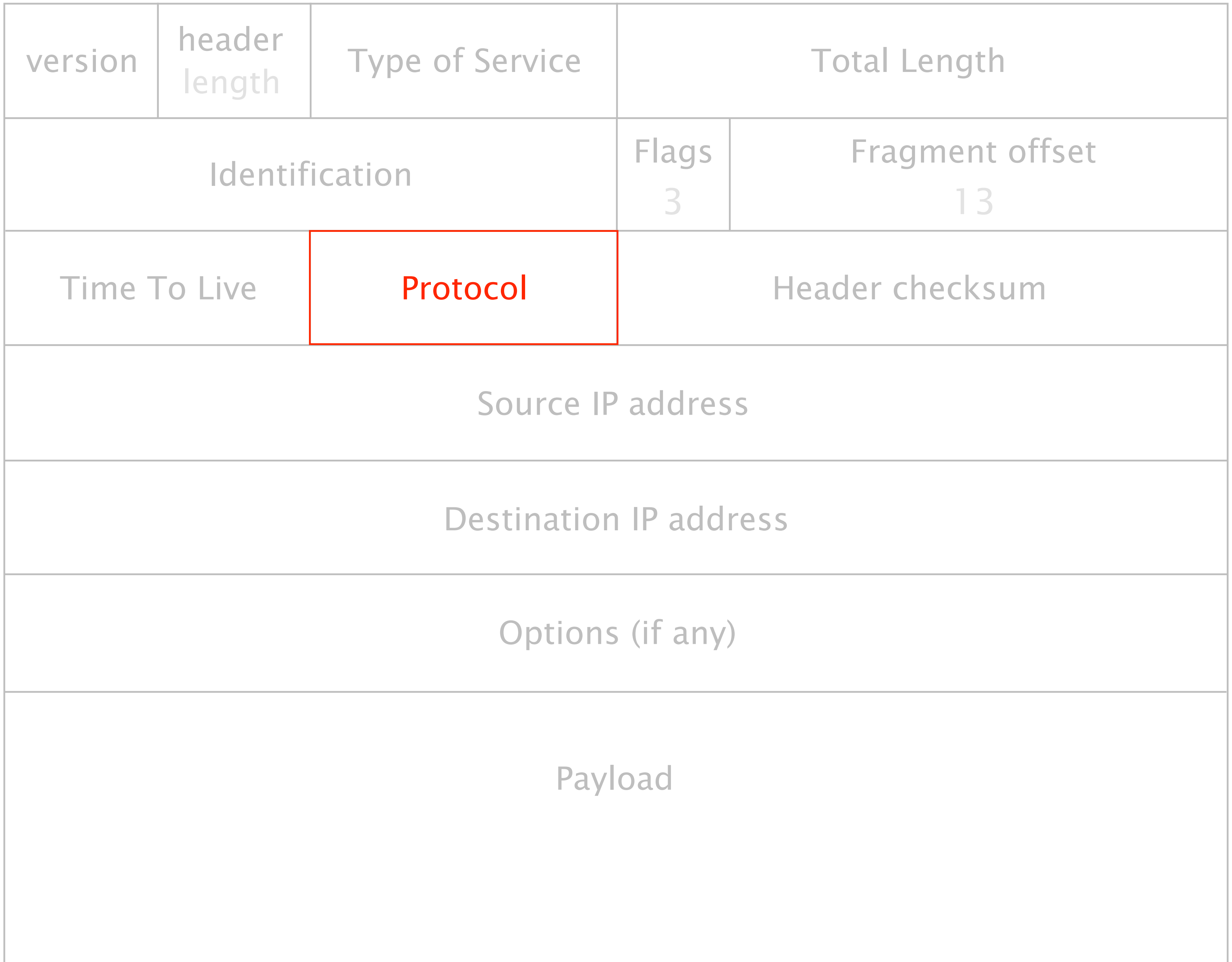


default TTL values

*nix (Linux/Mac)	64
Windows	128

(used for OS fingerprinting)

The protocol field identifies the higher level protocol carried in the packet, “6” for TCP, “17” for UDP



version

header length

Type of Service

Total Length

Identification

Flags 3

Fragment offset 13

Time To Live

Protocol

Header checksum

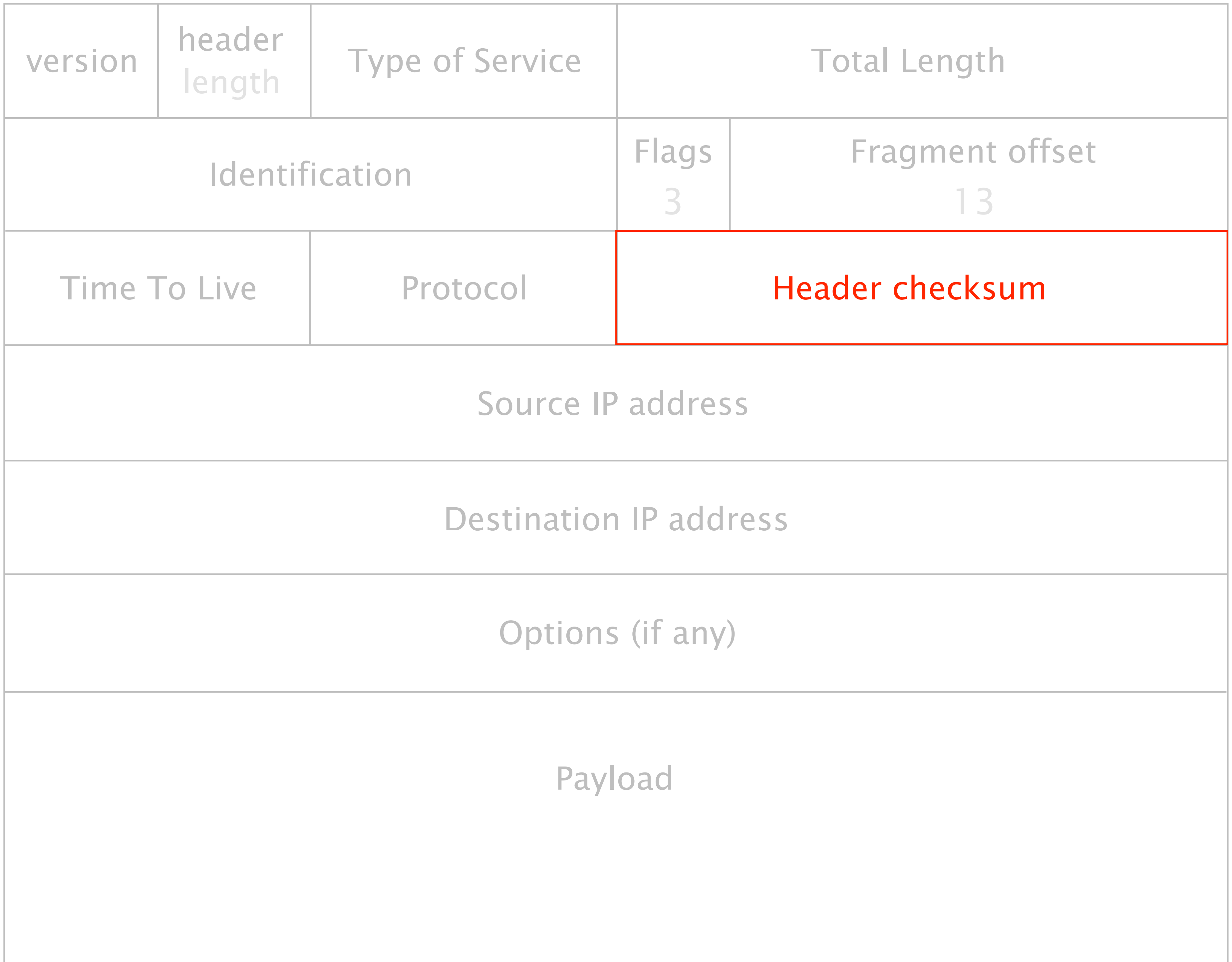
Source IP address

Destination IP address

Options (if any)

Payload

The checksum is the sum of all the 16 bits words in the header (does not protect the payload)



version

header length

Type of Service

Total Length

Identification

Flags 3

Fragment offset 13

Time To Live

Protocol

Header checksum

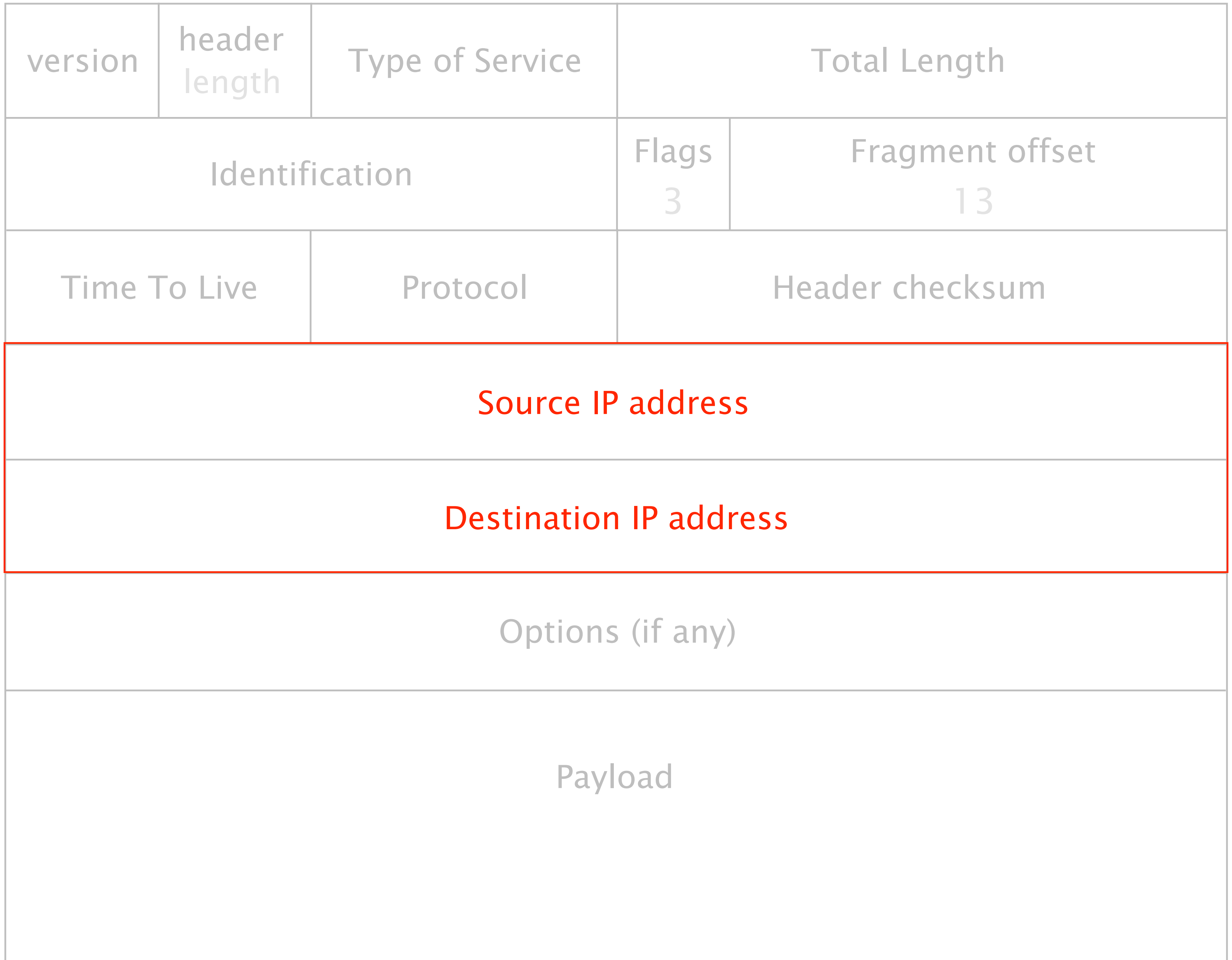
Source IP address

Destination IP address

Options (if any)

Payload

The source and destination IP uniquely identifies the source and destination host



version

header length

Type of Service

Total Length

Identification

Flags 3

Fragment offset 13

Time To Live

Protocol

Header checksum

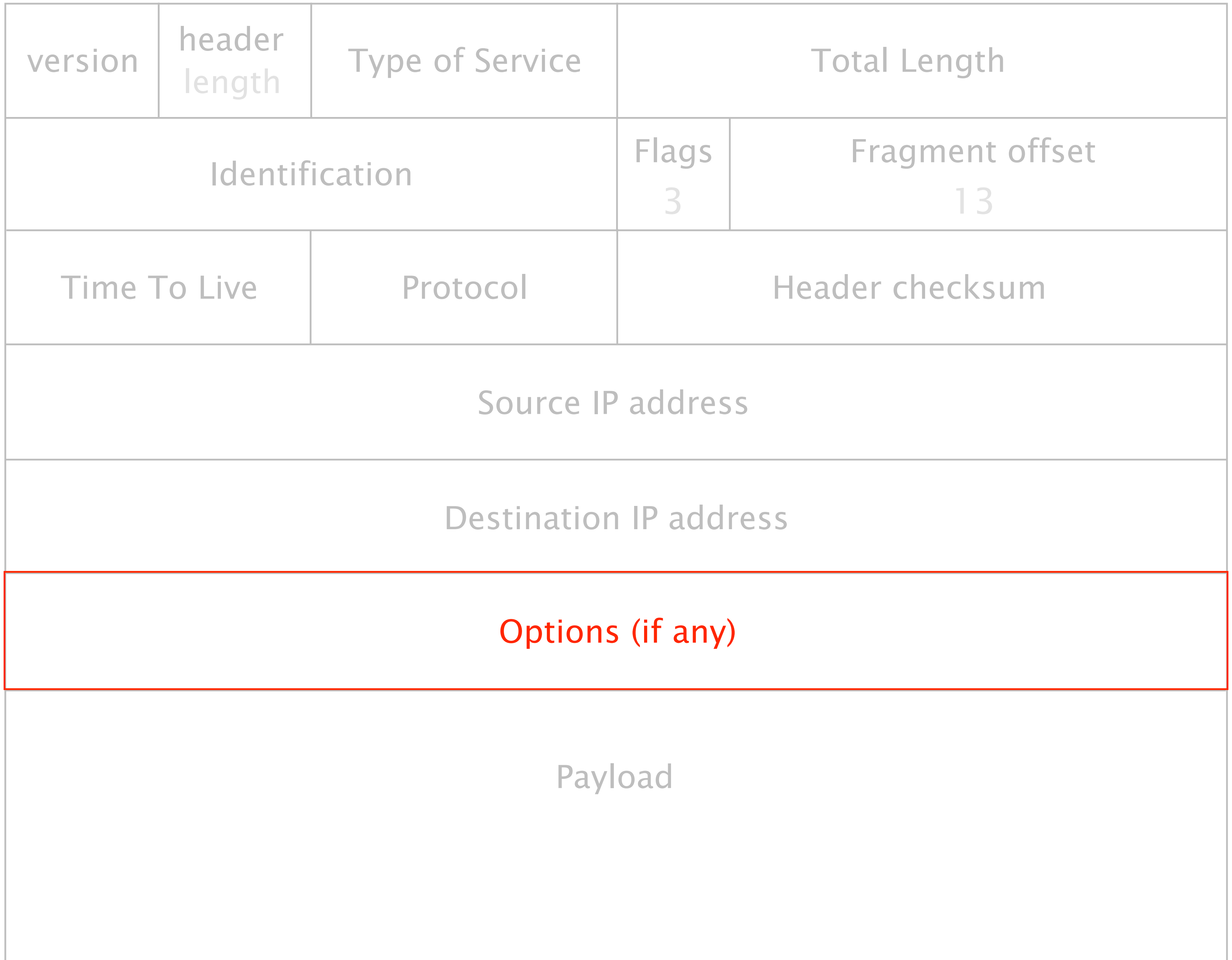
Source IP address

Destination IP address

Options (if any)

Payload

Options were initially put to provide additional flexibility. For security reasons, there are often deactivated.



version

header length

Type of Service

Total Length

Identification

Flags 3

Fragment offset 13

Time To Live

Protocol

Header checksum

Source IP address

Destination IP address

Options (if any)

Payload

IP options

Record route

Strict source route

Loose source route

Timestamp

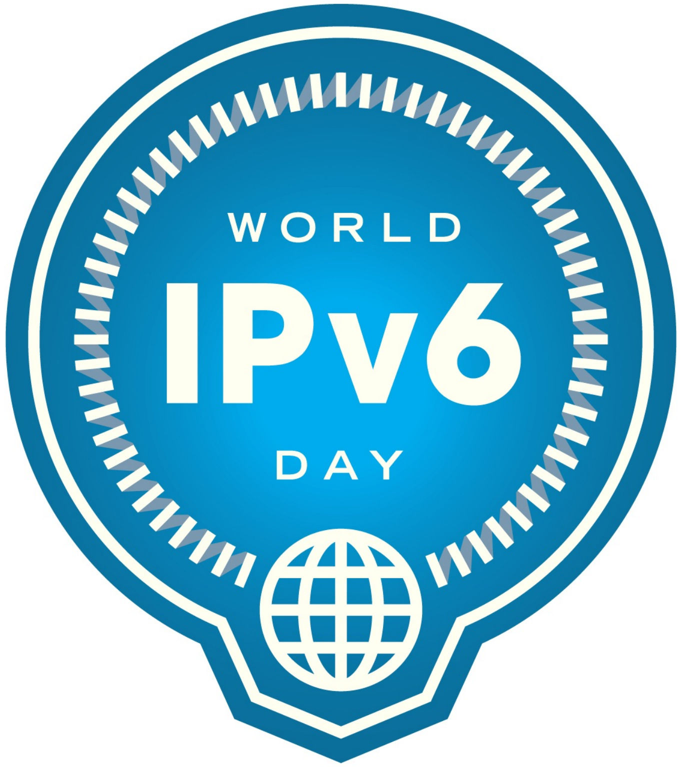
Traceroute

Router alert

…

see http://www.networksorcery.com/enp/protocol/ip.htm#Options for a full list

While there are no new IPv4 available,
IPv4 still accounts for most of the Internet traffic (for now)



With respect to IPv4, IPv6 is simpler

IPv6 was motivated by address exhaustion

IPv6 addresses are 128 bits long, that's plenty!

IPv6 got rid of anything that wasn't necessary

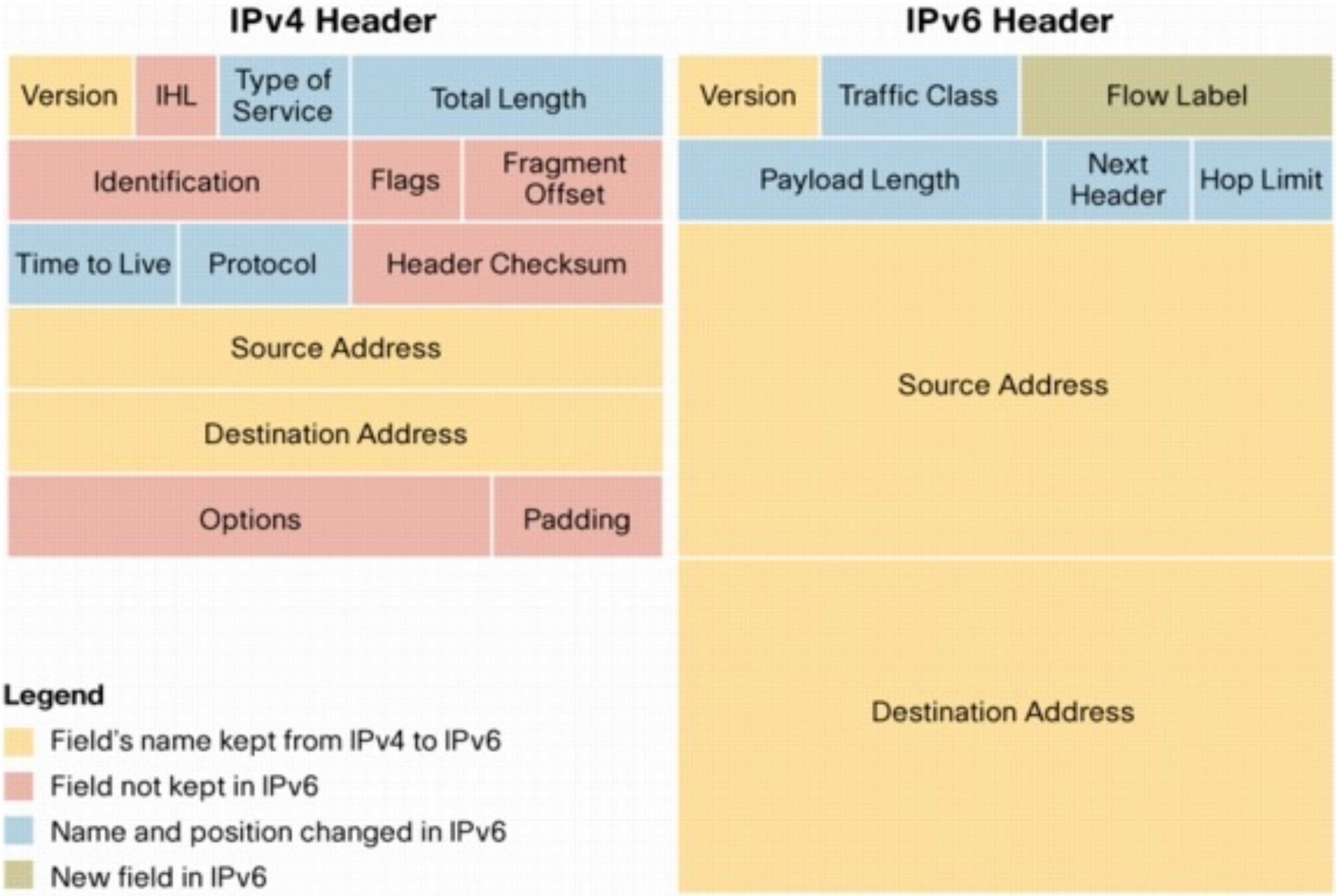
spring cleaning

Result is an elegant, if unambitious, protocol

With respect to IPv4,
IPv6 is simpler

IPv6	removed	reason
	fragmentation	leave problems to the end host
	checksum	
	header length	simplify handling
	added…	
	new options mechanism	simplify handling
	expanded addresses	
	flow label	flexibility

IPv4 vs IPv6



IPv4 Header

Version | IHL | Type of Service | Total Length

Identification | Flags | Fragment Offset

Time to Live | Protocol | Header Checksum

Source Address

Destination Address

Options | Padding

IPv6 Header

Version | Traffic Class | Flow Label

Payload Length | Next Header | Hop Limit

Source Address

Destination Address

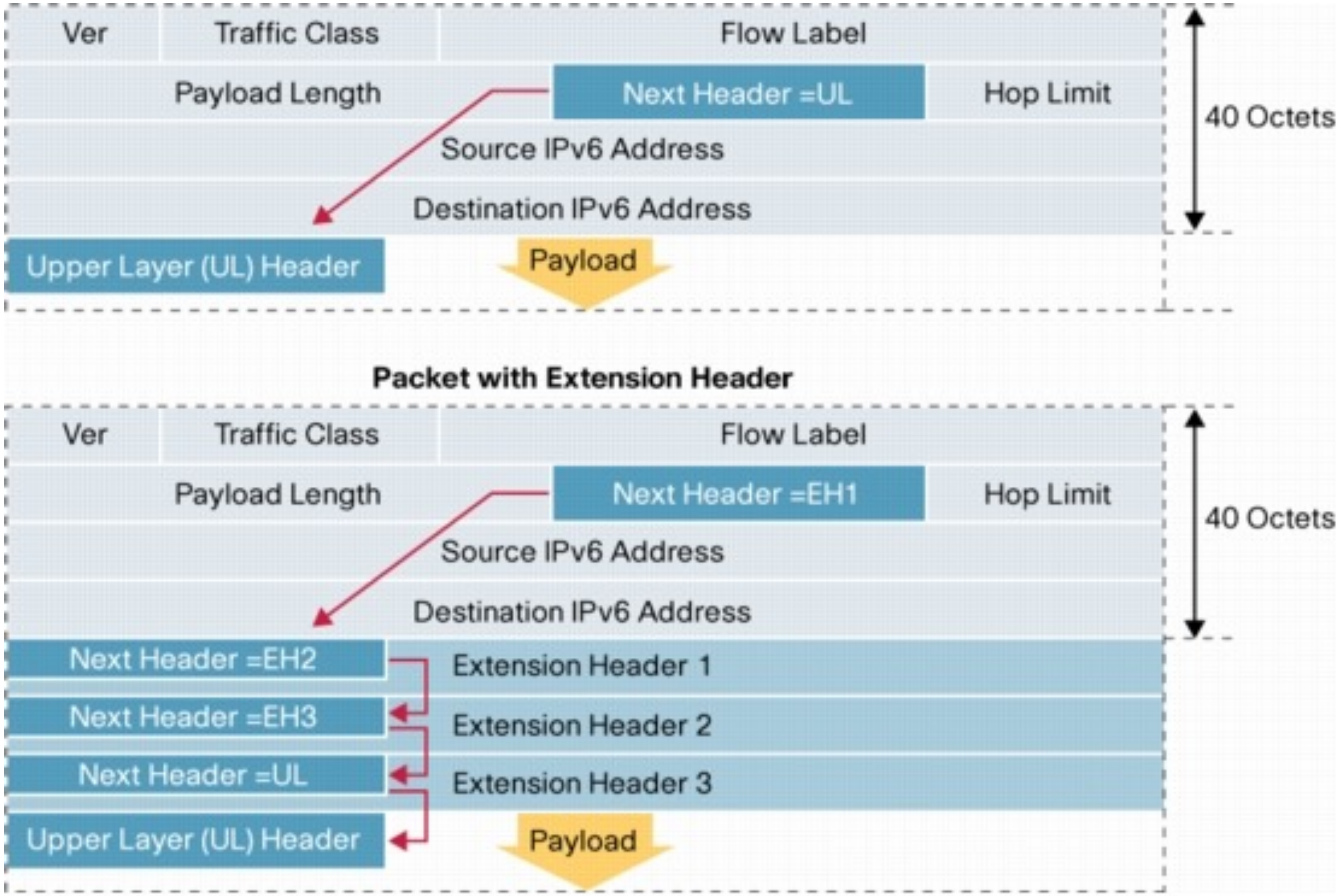
Legend

- 🟨 Field's name kept from IPv4 to IPv6
- 🟥 Field not kept in IPv6
- 🟦 Name and position changed in IPv6
- 🟫 New field in IPv6

source http://bit.ly/1HXc2BS

IPv6 enables to insert arbitrary options in the packet

see RFC 2460

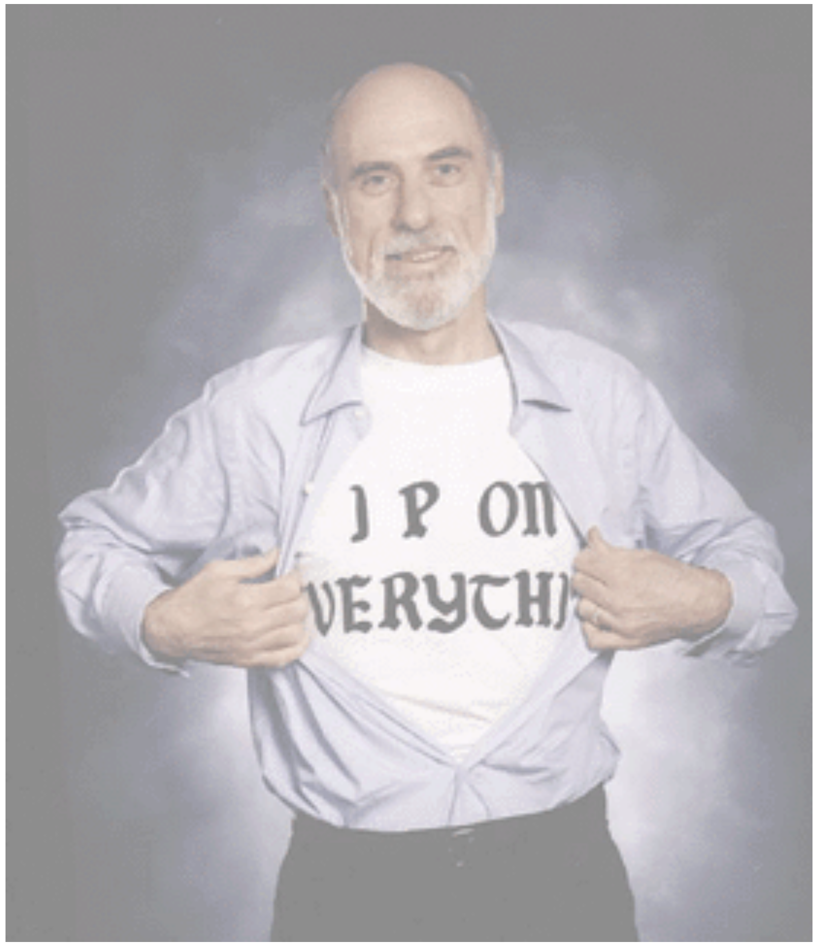


source http://bit.ly/1HXc2BS

The problem with IPv4 options is that all of them must be processed by each router, which is slow

In IPv6, only one type of optional header must be processed by each router

Internet Protocol and Forwarding



IP addresses

use, structure, allocation

IP forwarding

longest prefix match rule

IP header

IPv4 and IPv6, wire format

Next week on

Communication Networks

Internet routing!

Communication Networks

Spring 2020—Online Edition

Laurent Vanbever

nsg.ee.ethz.ch

ETH Zürich (D-ITET)

March 19 2020