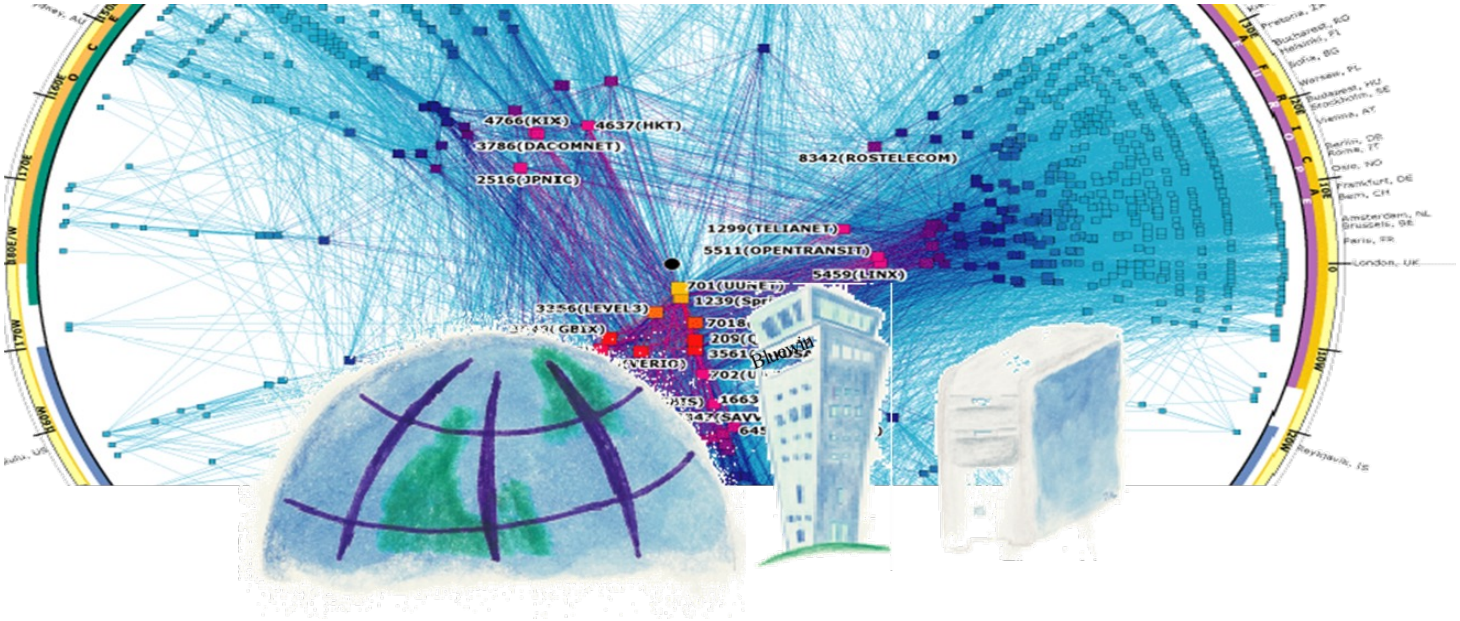


EPFL

Introduction to TCP/IP

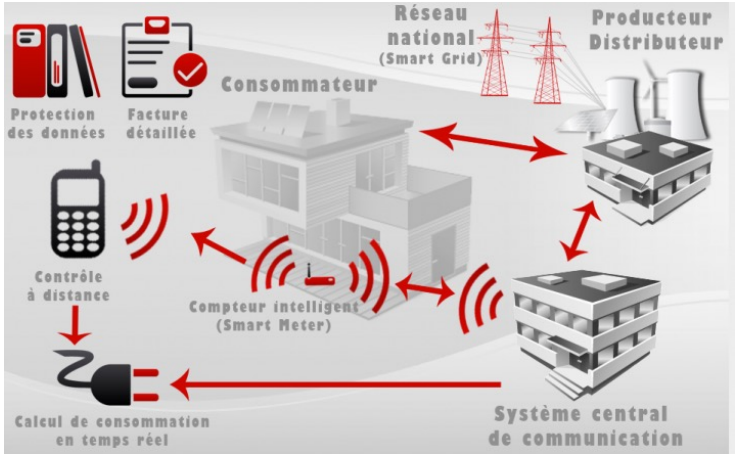
Jean-Yves Le Boudec

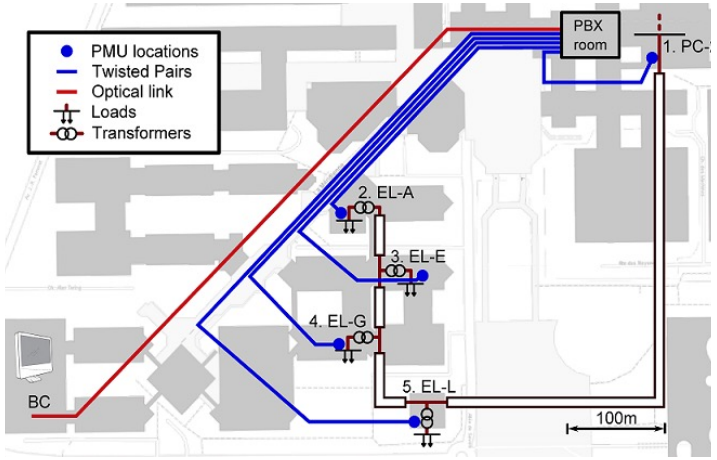
February 2023

This slide show is available at: https://leboudec.github.io/leboudec/resources/tutorial.html

Objective

Understand the principles of TCP/IP in order to use it well





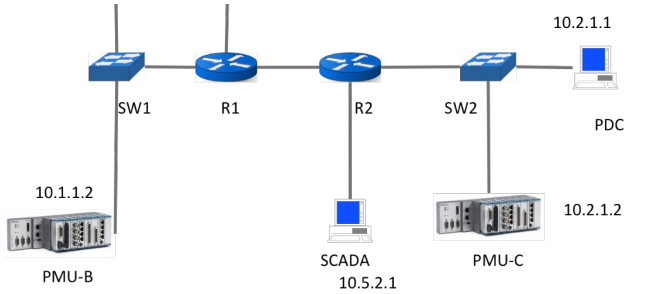
Textbooks

Kurose and Ross, Computer Networking, a top down approach (8th edition)

Olivier Bonaventure, Computer Networking : Principles,Protocols and Practice, open book, https://www.computer-networking.info/

Contents

1. Computer Communication



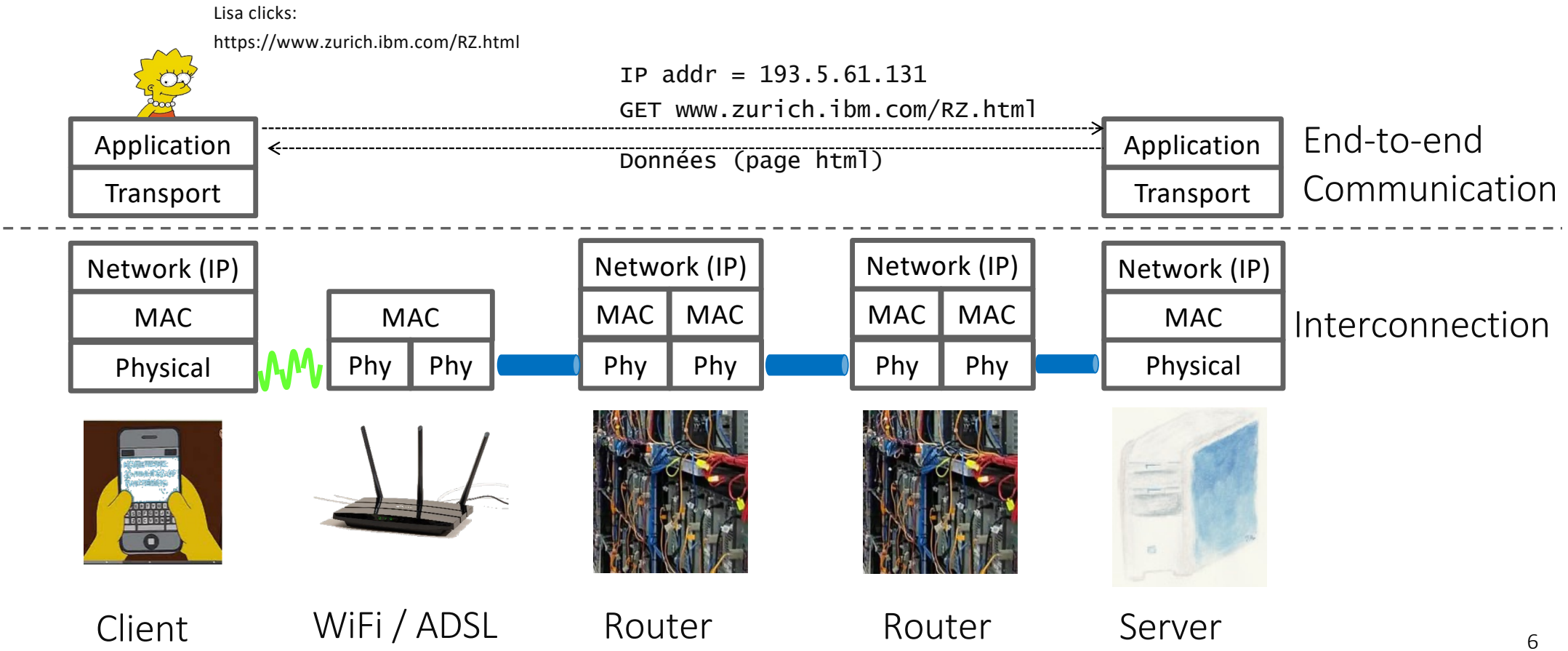
Clients (smartphones, PCs, PMUs) and servers (web sites, PDCs)

Server: a device that waits for messages to arrive

Cables, wireless links, base stations, routers,…

Communication functions are organized in layers:

- a layer entity speaks to entities of the same layer + uses the service of the layer below
- TCP/IP is the name of the architecture used in the Internet and in quasi-all networks



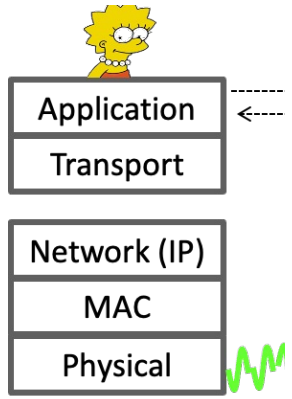
The Transport Layer

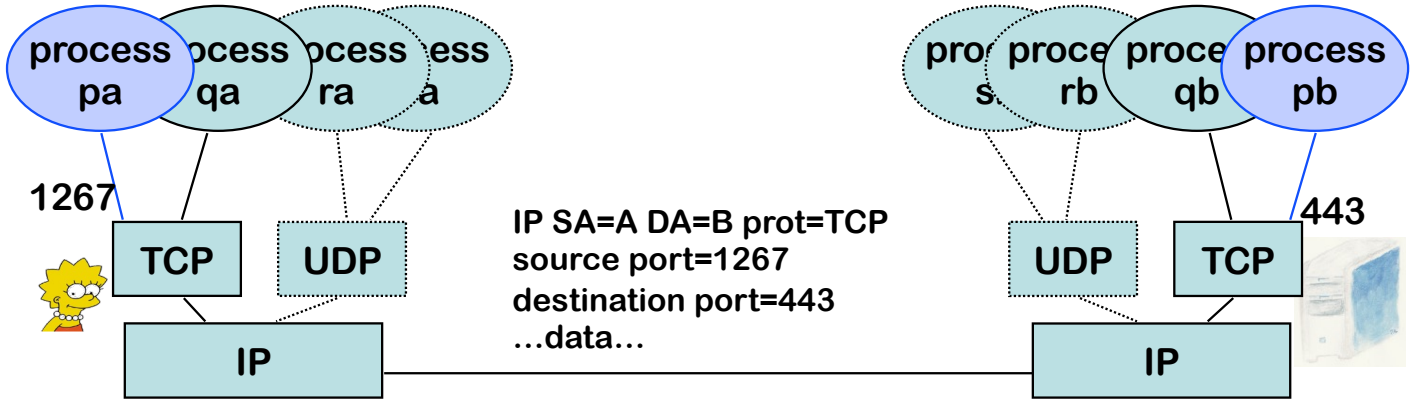
The application layer is what we use the network for (web, PMU monitoring system, etc). It uses the transport layer, which provides a programming interface.

The transport layer exists in 2 variants:

- UDP: simplistic, non reliable;
- TCP: repairs all packet losses that can be repaired, more complex.

The transport layer uses port numbers to differentiate processes inside a machine.





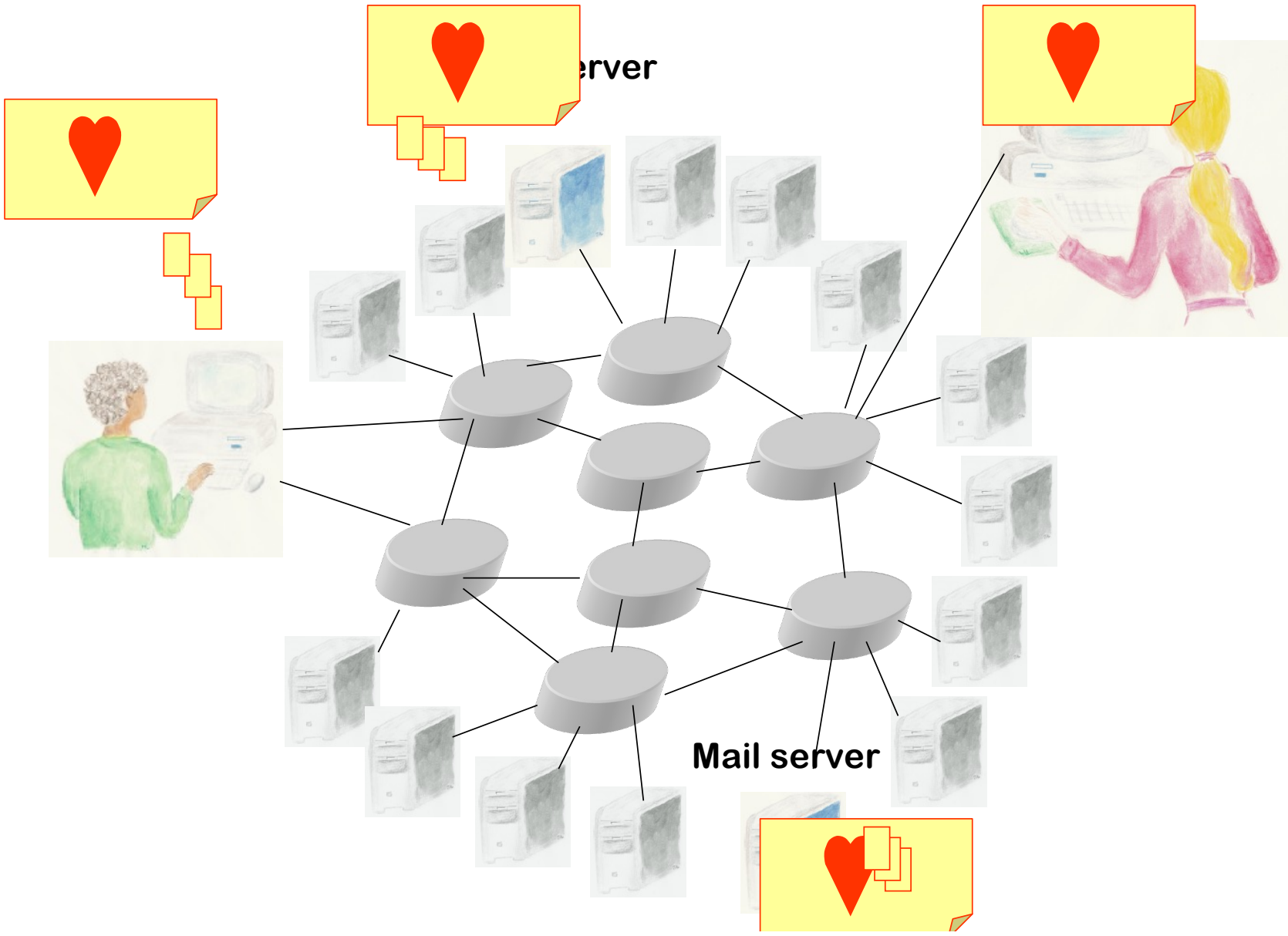
2. The Network and MAC Layers

The network layer interconnects all systems in a given universe (Internet, intranet, smart grid network). It uses the IP protocol (Internet Protocol).

IP Principle #1:

- Every interface has a number, called “IP address”, which is unique in a given universe.
- The data received from the transport layer are broken into chunks called “IP packets” of size $\leq$ 1500 bytes (in principle).
- Every IP packet contains the destination and source addresses. Intermediate systems, called routers, forward packets based on the destination address (similar to postcard handling by the Post office).

Louis Pouzin 1973, first packet switching communication network, Cyclades, France
Vint Cerf and Bob Kahn, TCP/ IP, 1974


erver
Mail server

Address Format: IPv4

The old (but most widespread) numbering plan is IPv4.

Every address is 32 bits long, and is often written in dotted decimal notation, where one integer in {0, 1,…, 254, 255} represents 8 bits.

An EPFL address: 128.178.151.1
Private addresses: 192.168.1.23, 172.16.0.0 to 172.31.255.255, 10.201.121.98.
This host (localhost): 127.0.0.1

System	Alphabet	Example
Binary	$\{0,1\}$	1000 0000 1011 0010 1001 0111 0000 0001
Dotted decimal	$\{0,1,2,3,4,5,6,7,8,9,.\}$	128.178.151.1
Hexadecimal	$\{0,1,2,3,4,5,6,7,8,9,a,b,c,d,e,f\}$	80b2 9701

$$1011\ 0010_{bin} = b2_{hex} = 128 + 32 + 16 + 2 = 178$$

$$ff_{hex} = 1111\ 1111_{bin} = 255$$

IP Principle #2: «longest prefix match»

- Every system (host, router) has a routing table, which is a list of couples (prefix, where to forward).
- When an IP packet is received, if the destination address is not one of the addresses of this system, the routing table is searched for the longest prefix match

Destination	*Next-Hop / Interface*
128.178.29/24	128.178.100.2 / south
128.178/16	128.178.100.3 / south
0/0	128.178.47.3 / north

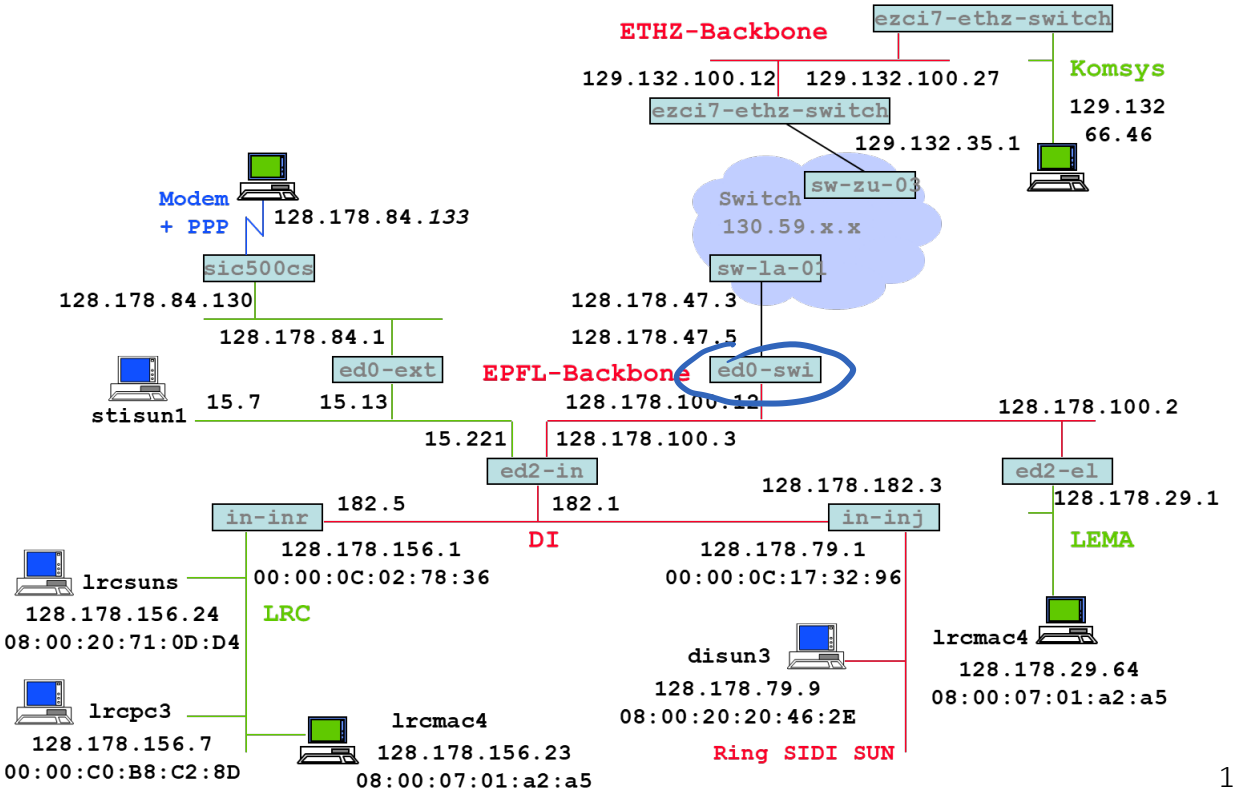
Routing table of router ed0-swi

= router

/24 means a prefix of 24 bits

0/0 (empty prefix): default

Ex: to 128.178.* → ed2-in
to 128.178.29.* → ed2- el
else → sw-la-01

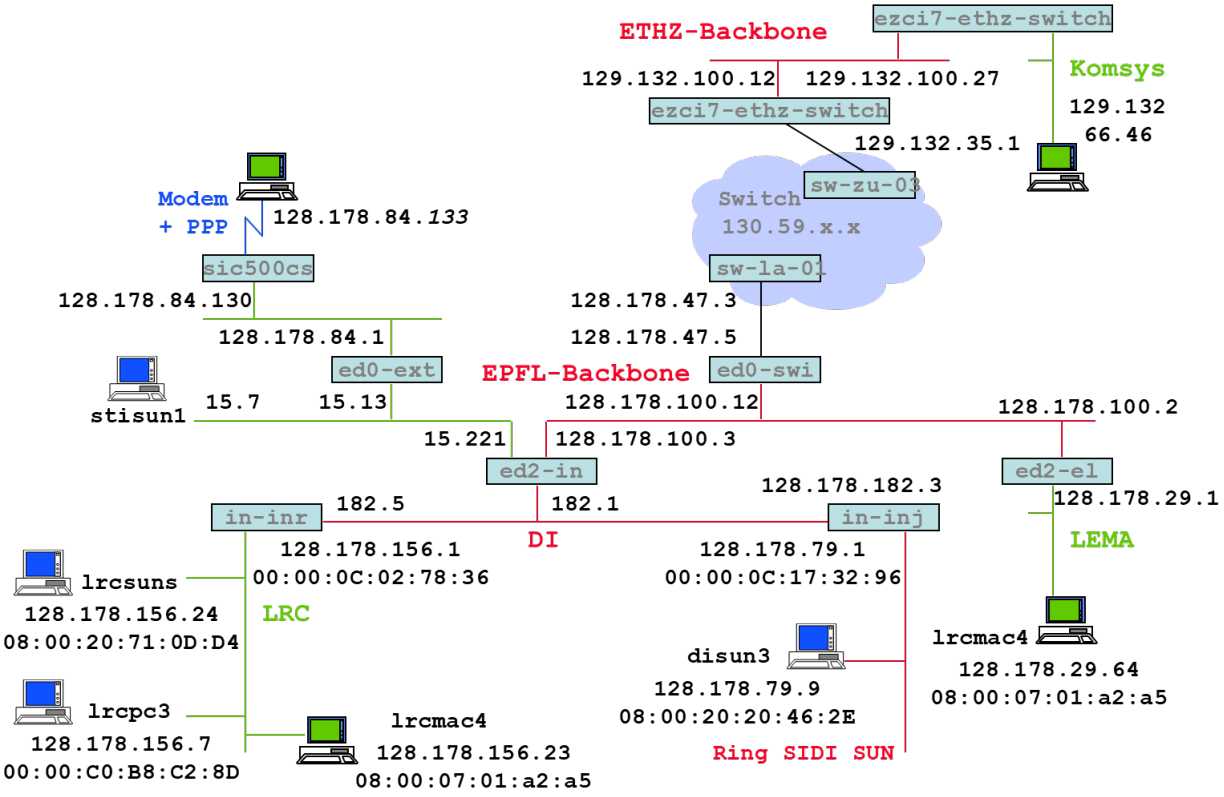


Corollary: IP addresses are structured

My IP address is not the same when I am at EPFL, or at home, because it reflects the router to which I am connected.

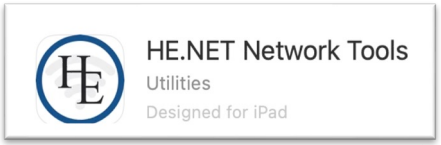
My device address must be configured, either automatically or manually (DHCP).

In routers, routing tables must also be configured, either manually, or more typically, by routing protocols (OSPF, BGP).



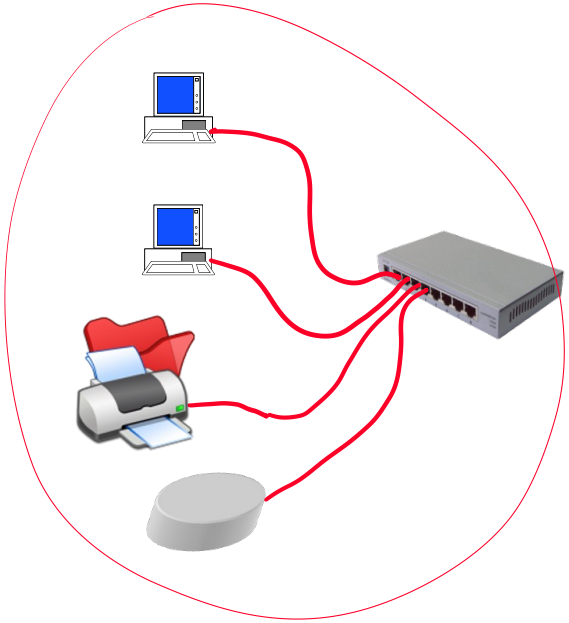
Lookup your IP address:

- Windows: ipconfig
- Mac, Linux: ifconfig
- Smartphone: HE.NET app



We could build a network by connecting all systems to routers, then connecting the routers to routers et cetera. However, this is not how things are done. Usually, systems are first connected to Ethernet switches, or Wi-Fi stations, which are intermediate systems that do not use IP addresses (therefore, are not routers).

WiFi Station



Ethernet Switch

Why ? How does it work ?

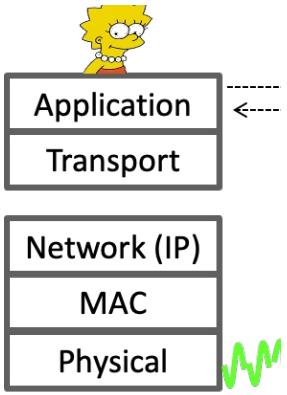
The MAC layer

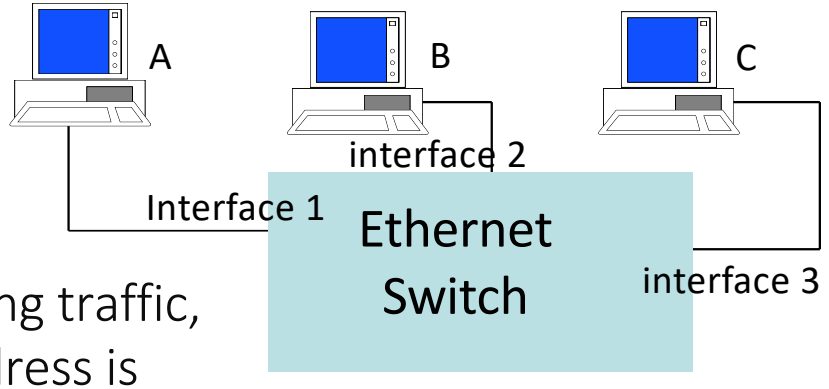
Goal = interconnect a small number of system, without configuration («plug and play»)

Principle:

- Every communication interface has a MAC address (48 bits), also called physical address, which in principle is its serial number. Written in hexa, e.g. 08:00:20:71:0d:d4
- MAC layer packets (called “frames”) contain source and destination MAC addresses.
- An Ethernet switch builds a MAC layer table by observing traffic, and uses it to forward frames. If a MAC destination address is unknown, the frame is broadcast to all ports.

  The Ethernet switch is plug-and-play, but does not scale to large networks as addresses have no structure.





Forwarding Table

Destination	i/f
08:00:20:71:0d:d4	1
00:00:c0:3f:6c:a4	2
2a:c8:00:99:88:e2	3

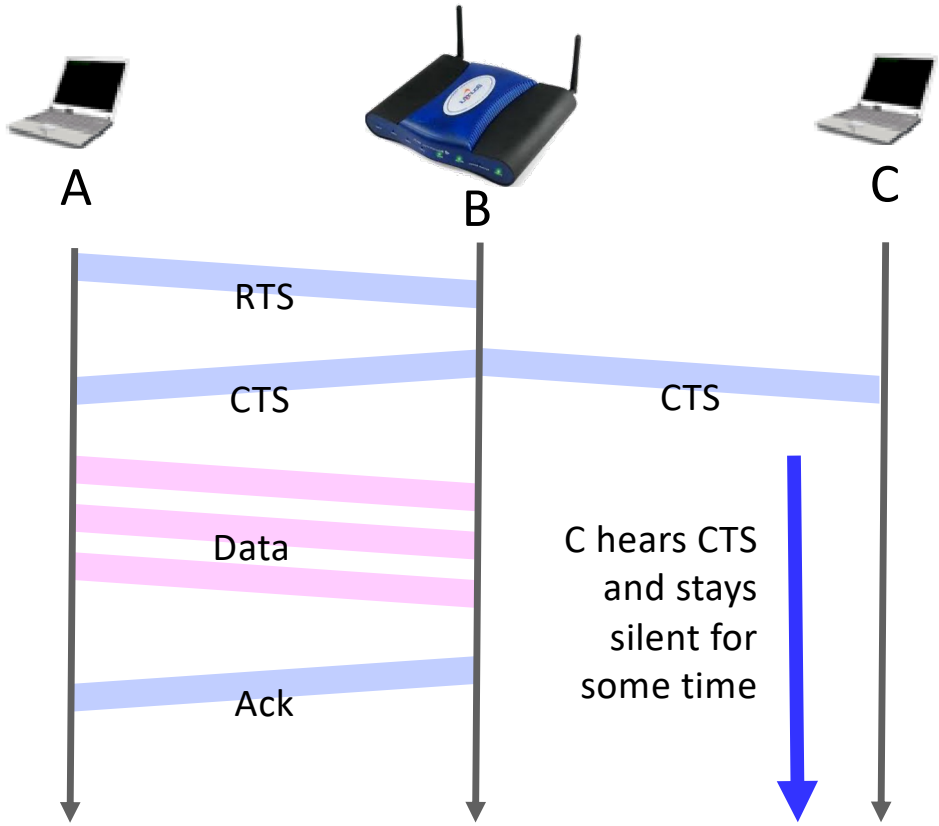
WiFi

The MAC layer exists in many variants: Ethernet, WiFi, Powerline Communication (PLC), Bluetooth, etc.

They all use only MAC addresses only in order to handle a packet.

In addition, the WiFi, Bluetooth and PLC MAC layers handle collisions on the wireless or wired medium.

(MAC = Medium Access Control).



How MAC and IP layers interact

Local area networks can be interconnected by means of routers.

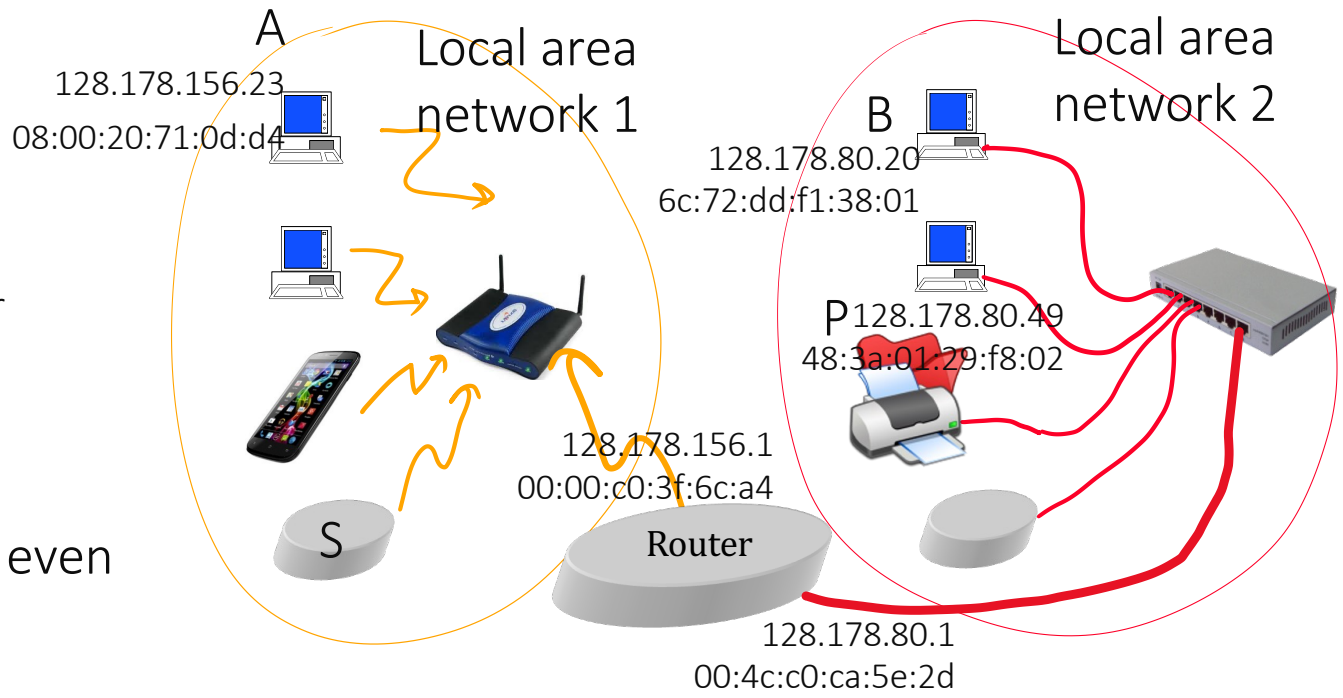
Every host must be configured with the address of its default router (sometimes called “default gateway”)

For A it is 128.178.156.1

A→P: A creates an IP packet with destination address 128.178.80.49, sends it to the router, i.e. to MAC address 00:00:c0:3f:6c:a4. The router forwards the IP packet on its Ethernet interface, to 6c:72:dd:f1:38:01

Communication always uses IP, even inside a local area network.

E.g.: B→P, B must know P’s IP address

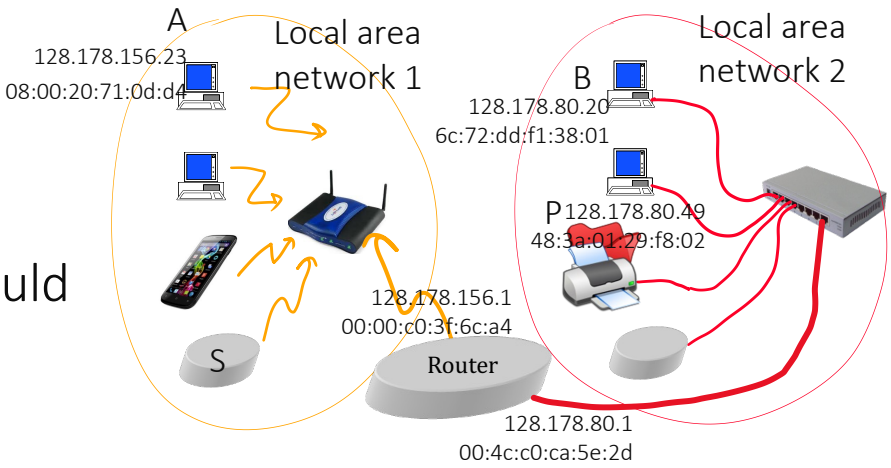


IP Principle #3

IP was designed as a method for interconnecting local-area networks.

IP = internet protocol

"subnet"= local area network = a collection of systems that can communicate at the MAC layer

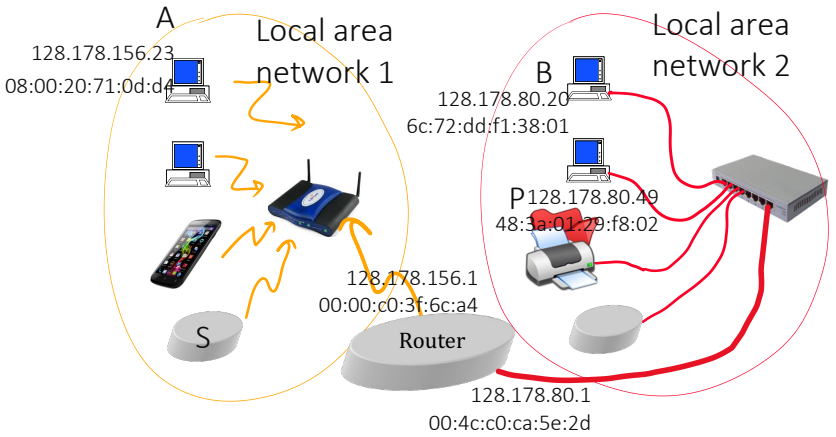


IP Principle #3:

- Communication between hosts in same subnet should not go via any router
- Between subnets, use routers

Consequence: a host must be know whether an IP correspondent is in the same subnet or not

Subnet mask



To every subnet must be correspond one subnet prefix

All hosts in same subnet must have same prefix

The size (in bits) of the subnet prefix is not always the same;
must be specified in the configuration;
EPFL-IPv4: 24 bits: Example: 128.178.71.34 /24
ETHZ IPv4: 26 bits

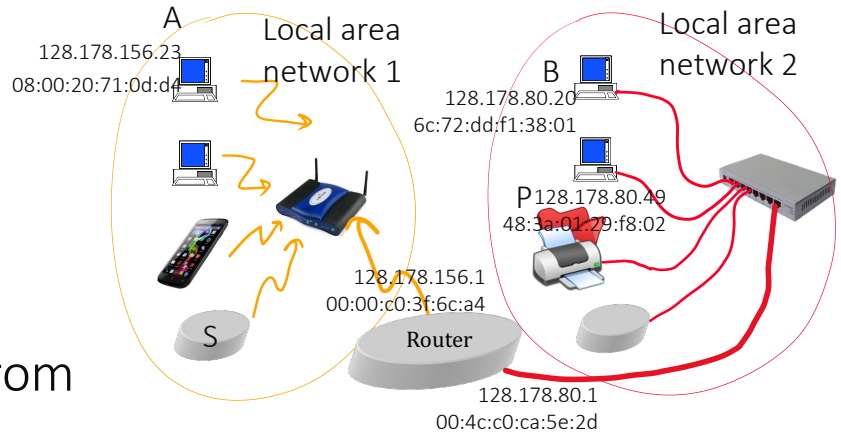
The size of IP subnet prefix is (still) often specified using a subnet mask
Mask = sequence of bits where 1s indicate the position of the prefix, often written in dotted decimal notation.

Ex: EPFL: 1111 1111 1111 1111 1111 1111 0000 0000 → 255.255.255.0
ETHZ: 1111 1111 1111 1111 1111 1111 1100 0000 → 255.255.255.192

Practical Consequence of Principle #3

Destination	*Next-Hop*
128.178.156/24	onlink
0/0	128.178.156.1

Routing table at A



For a host, the routing table is automatically derived from

- Its addresses and masks
- Its default router

A → P

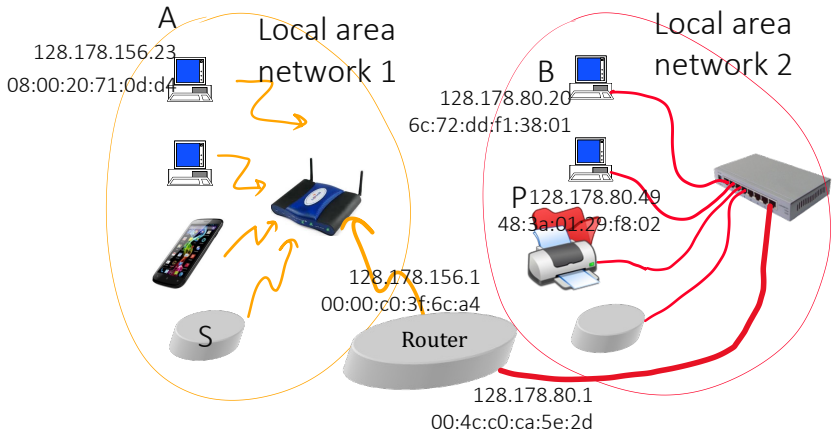
A receives from the transport layer the job of sending an IP packet to 128.178.80.49 (this address is known by the application layer, which passes it to the transport layer)

Longest prefix match ⇒ A sends IP packet to its default router, i.e. A puts it into a MAC packet with destination address 00:00:c0:3f:6c:a4 (address of router's Wifi interface).

Practical Consequence of Principle #3, continued

Destination	*Next-Hop*
128.178.80/24	onlink
0/0	128.178.80.1

Routing table at B



B→P

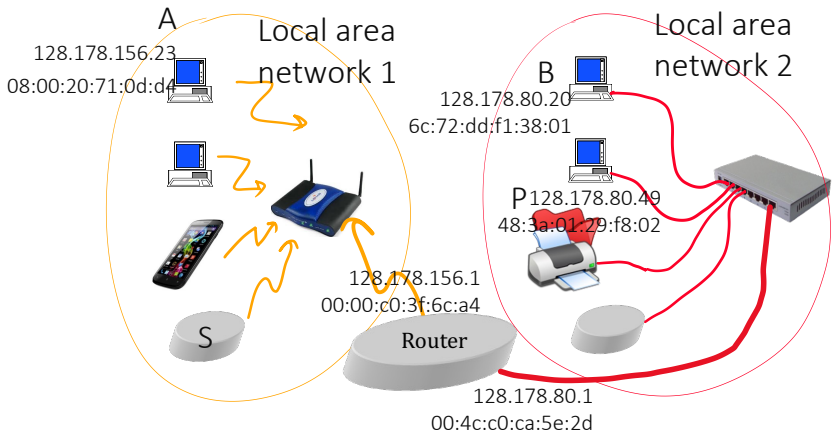
Longest prefix match ⇒ B sends IP packet directly to P,
i.e. B puts it into a MAC packet with destination address 48:3a:01:29:f8:02
(P's MAC address)

ARP

How does A know the MAC address of router's Wifi interface ?

How does B know P's MAC address ?

By ARP (Address Resolution Protocol)



B broadcasts an ARP request to its entire subnetwork, asking: "who has IP address 128.178.80.49". P answers with her MAC address: We hope that the answer does come from P and not from an attacker. The mapping IP address ↔ MAC address is kept in the "ARP cache" as long as it is confirmed by active traffic between B and P.

Look at your ARP cache:
Command shell in Windows, Mac, Linux: arp -a

Configuration of an IP Host

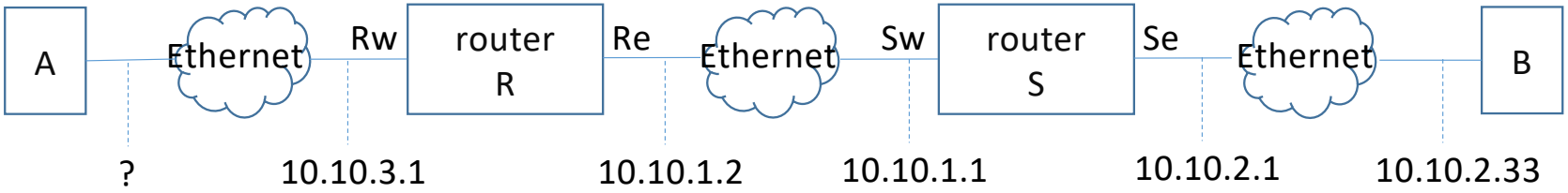
An IP host must be configured with:

- IP address and mask, for every interface
- IP address of default router
- IP address of one or several DNS servers

Can be entered manually in system settings, or automatically via DHCP: when an interface boots, it contacts a configuration server (e.g. in the WiFI base station or the Ethernet switch).

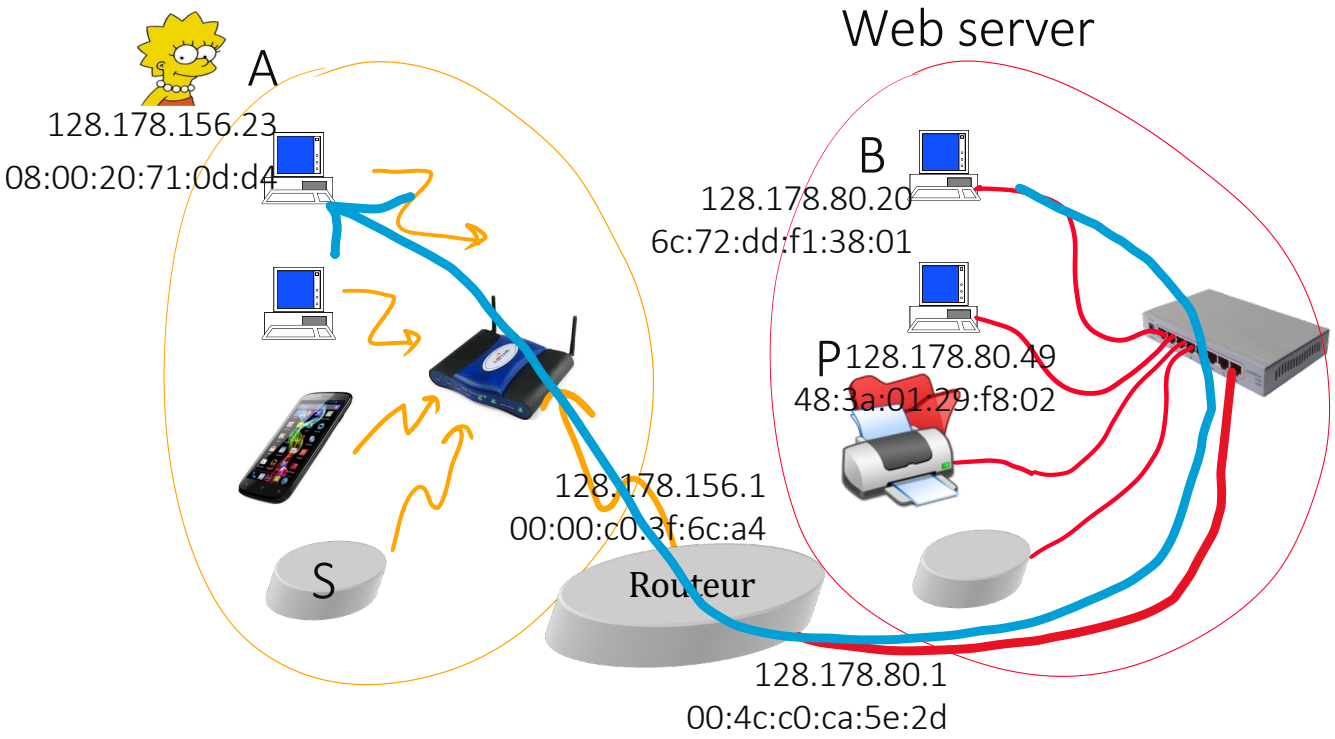
In all cases, a numbering plan must be made explicitly.

The subnet masks are all equal to 255.255.255.0. Which is a valid address for A ?

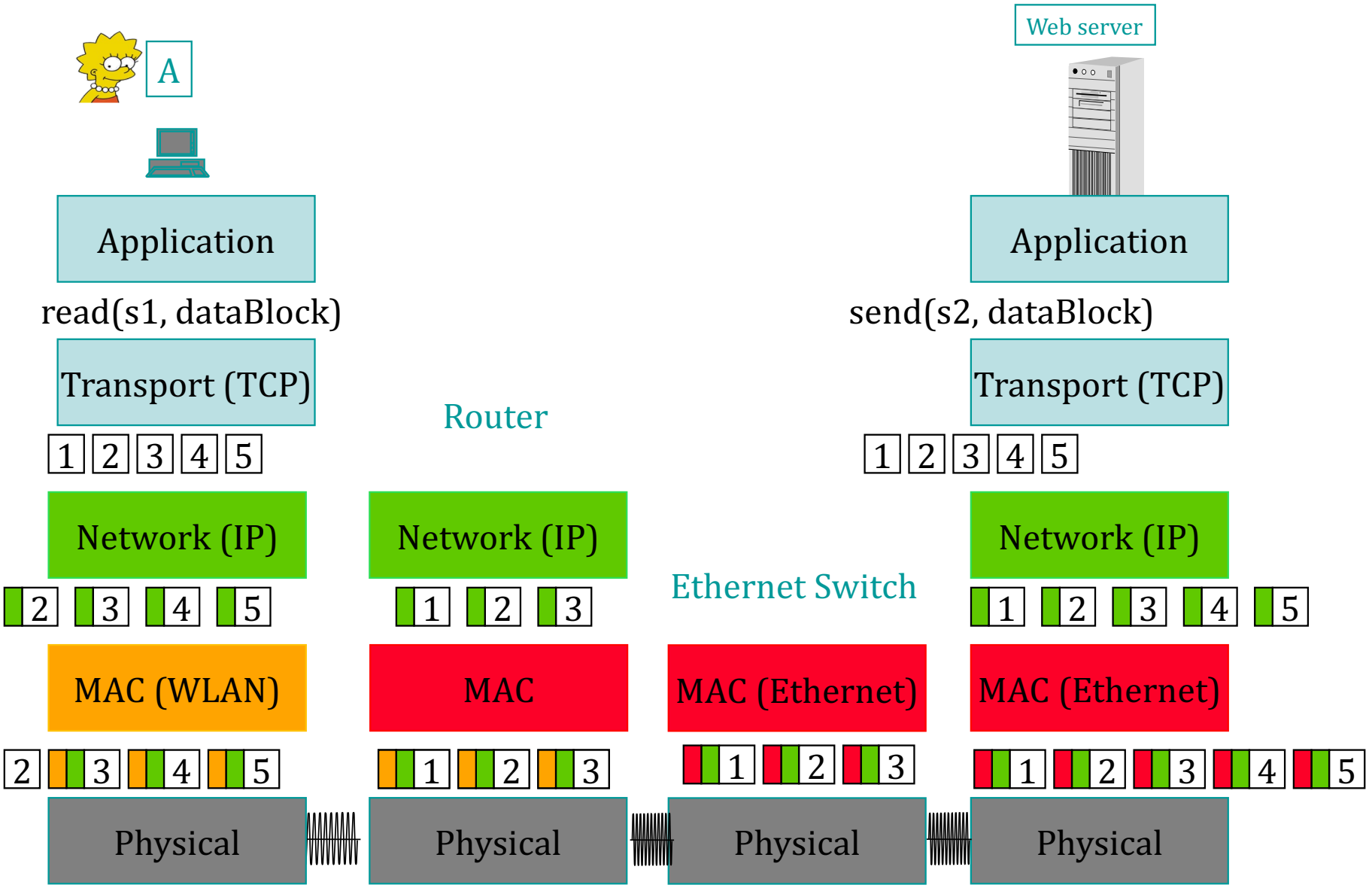


A. 10.10.3.33
B. 10.10.2.22
C. A and B
D. None
E. I don’t know

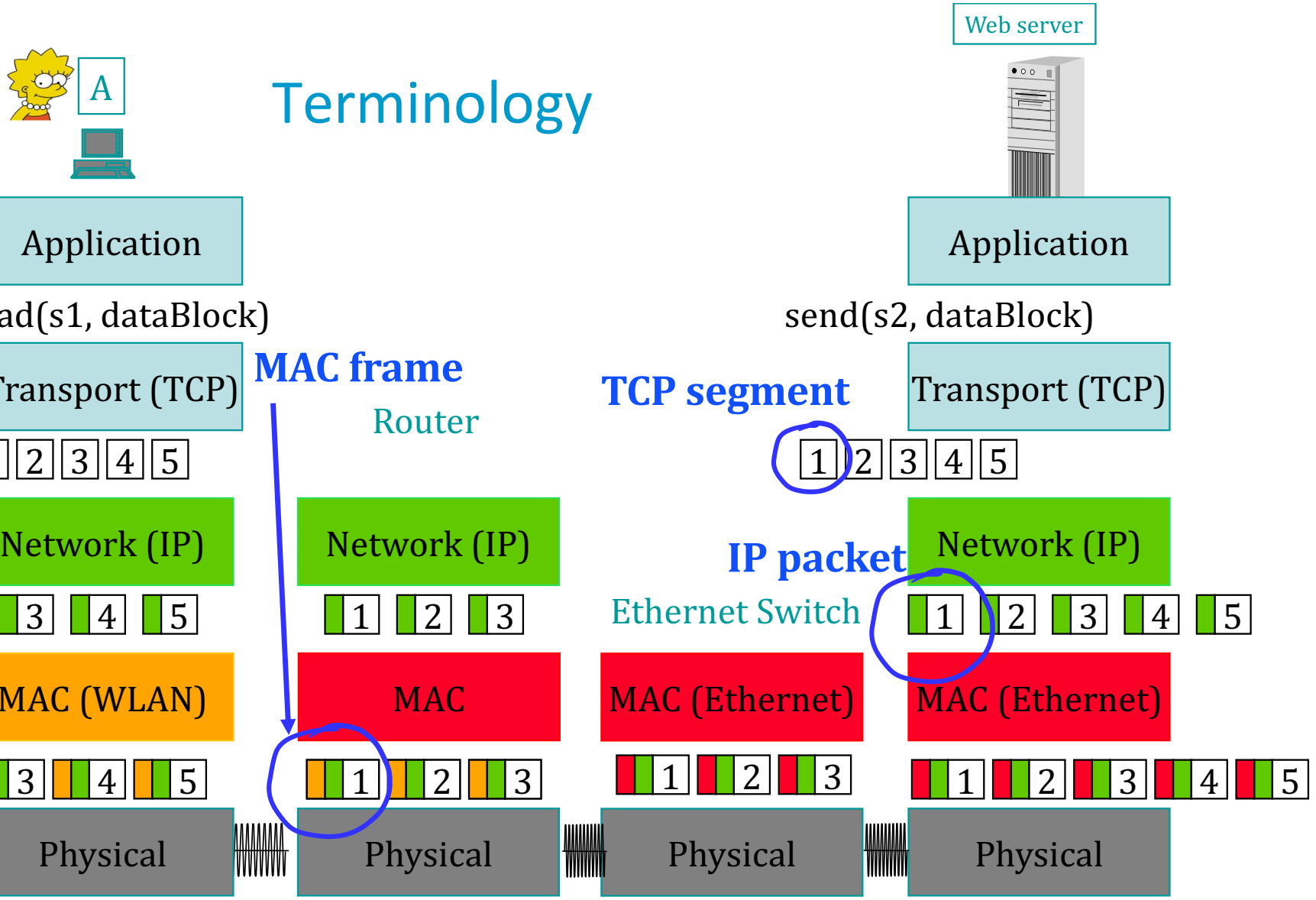
Putting things together …



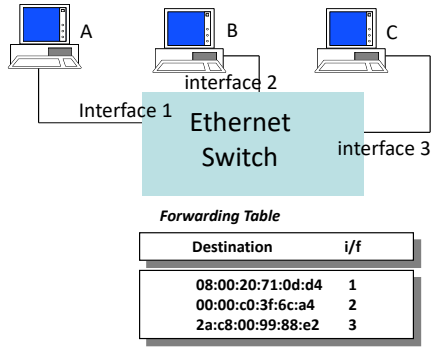
A web server in B sends data to A


A
Web server
Application
Application
read(s1, dataBlock)
send(s2, dataBlock)
Transport (TCP)
Transport (TCP)
Router
1 2 3 4 5
1 2 3 4 5
Network (IP)
Network (IP)
Network (IP)
Ethernet Switch
2 3 4 5
1 2 3
1 2 3 4 5
MAC (WLAN)
MAC
MAC (Ethernet)
MAC (Ethernet)
2 3 4 5
1 2 3
1 2 3
1 2 3 4 5
Physical
Physical
Physical
Physical

Terminology



Bridges

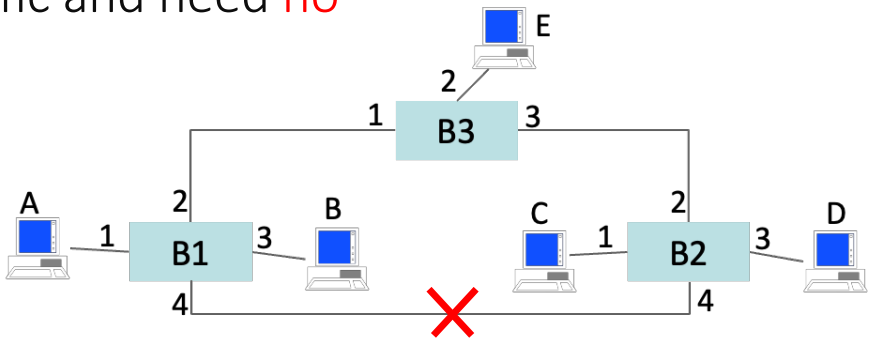


The Ethernet concentrator is an example of a bridge

Bridge = an intermediate system that forwards packets based on MAC addresses.

Bridges learn addresses automatically by observing traffic and need no configuration.

They can be networked, in which case they run the spanning tree protocol, which automatically disable loops.

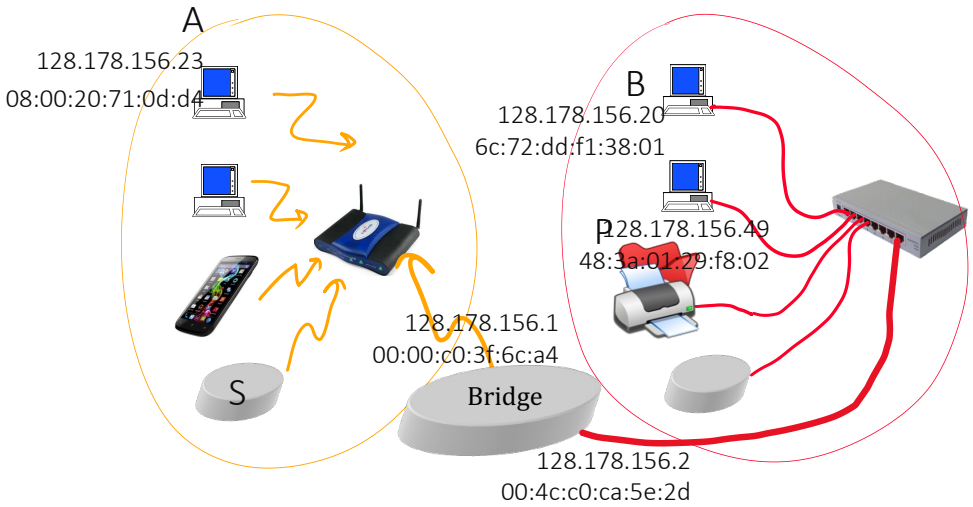


Bridges can be used between different MAC layer technologies

Bridge can be used to interconnect e.g. WiFi and Ethernet.

Bridges are transparent: A cannot distinguish (with wireshark) whether P is on WiFi or is behind bridges.

A bridged network = one subnet = a LAN (local area network).



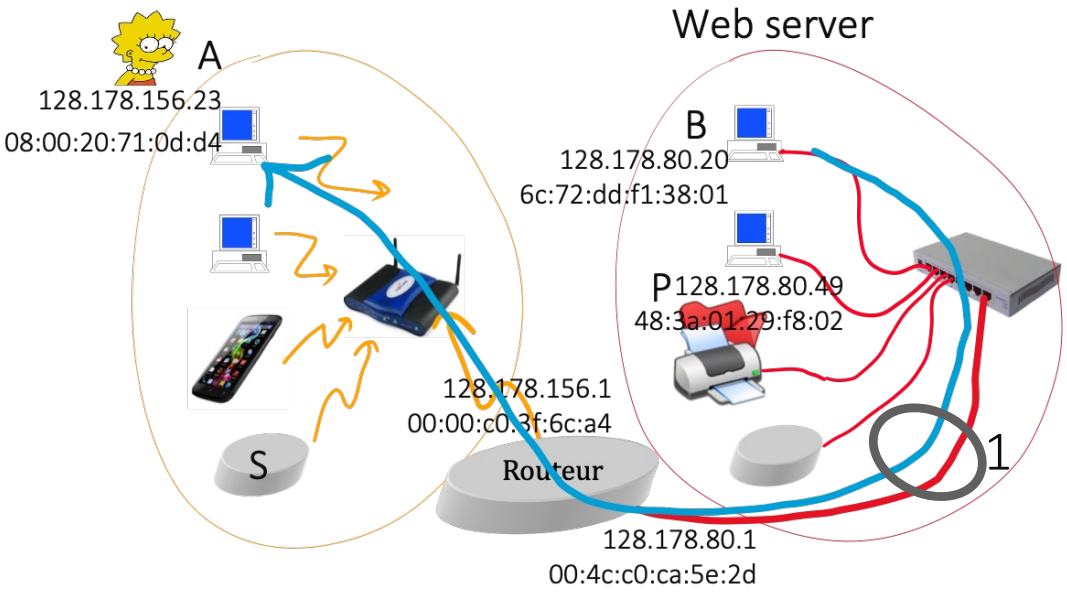
Here all systems are in the same subnet and should have the same subnet prefix.

A→ P

A sends IP packet directly to P,
i.e. A puts it into a MAC packet with destination address 48:3a:01:29:f8:02 (P's MAC address)

We analyze the packets at point 1.

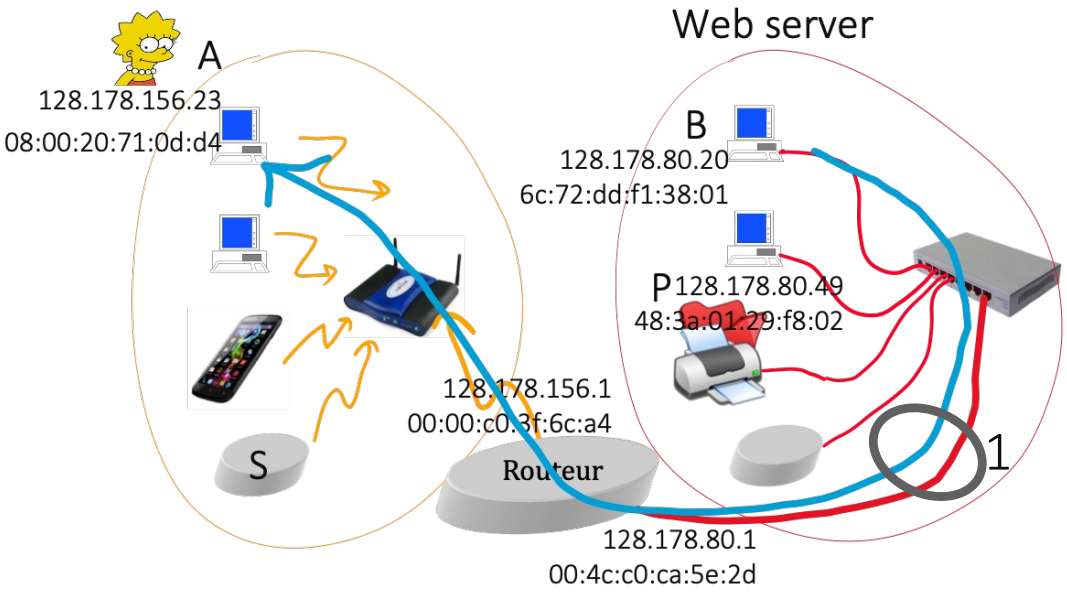
The destination MAC address is …



A. 00:4c:c0:ca:5e:2d
B. 00:00:c0:3f:6c:a4
C. 08:00:20:71:0d:d4
D. I don’t know

We analyze the packets at point 1.

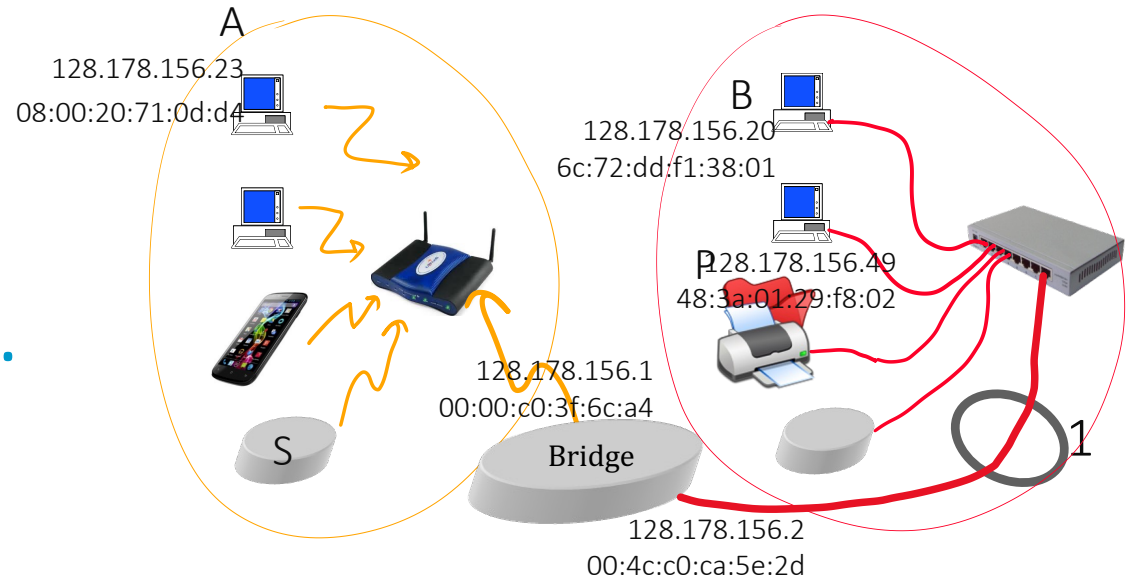
The destination IP address is …



A. 128.178.80.1
B. 128.178.156.1
C. 128.178.156.23
D. I don't know

We analyze the packets at point 1.

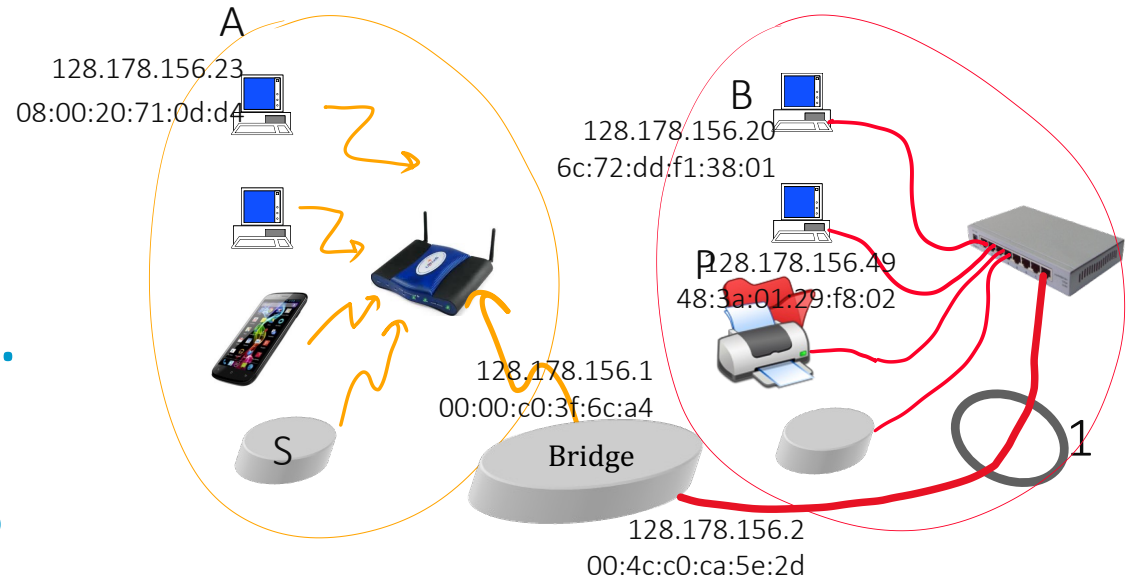
The destination MAC address is …



A. 00:4c:c0:ca:5e:2d
B. 00:00:c0:3f:6c:a4
C. 08:00:20:71:0d:d4
D. I don't know

We analyze the packets at point 1.

The destination IP address is …



A. 128.178.156.1
B. 128.178.156.2
C. 128.178.156.23
D. I don't know

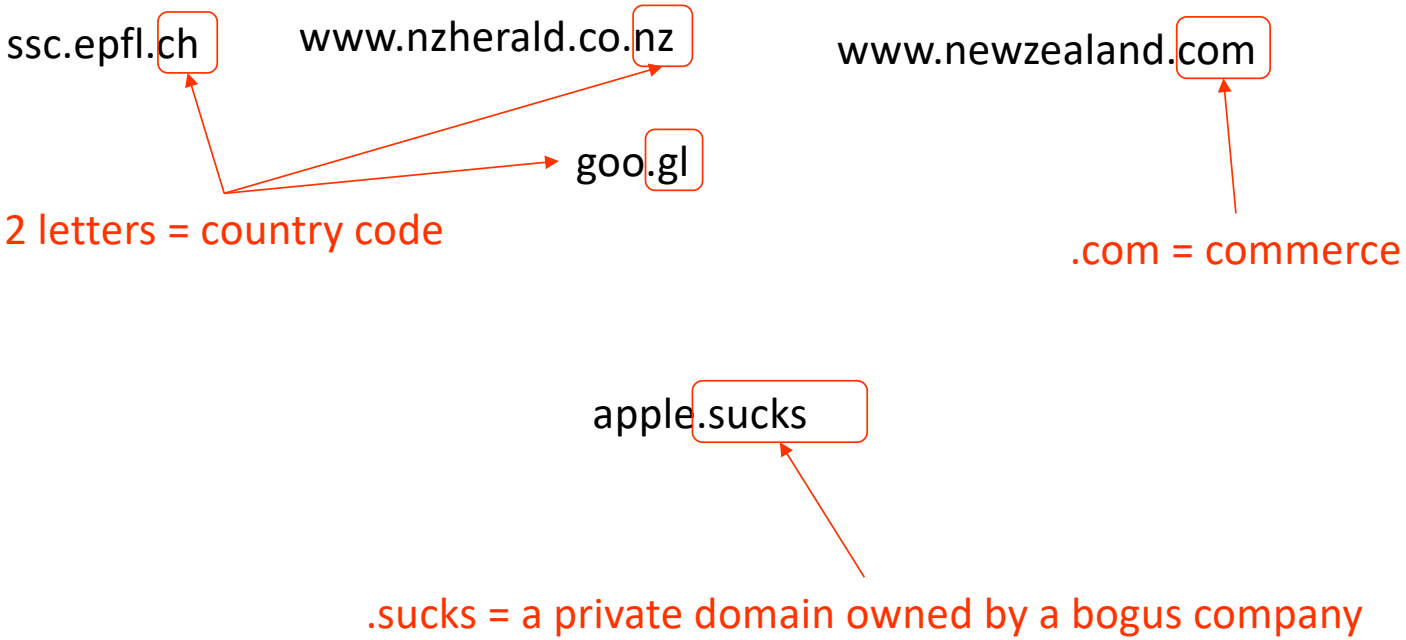
3. Addresses and Names

Names are human readable synonyms for IP addresses

Examples:

ssc.epfl.ch

smtp.sunrise.ch

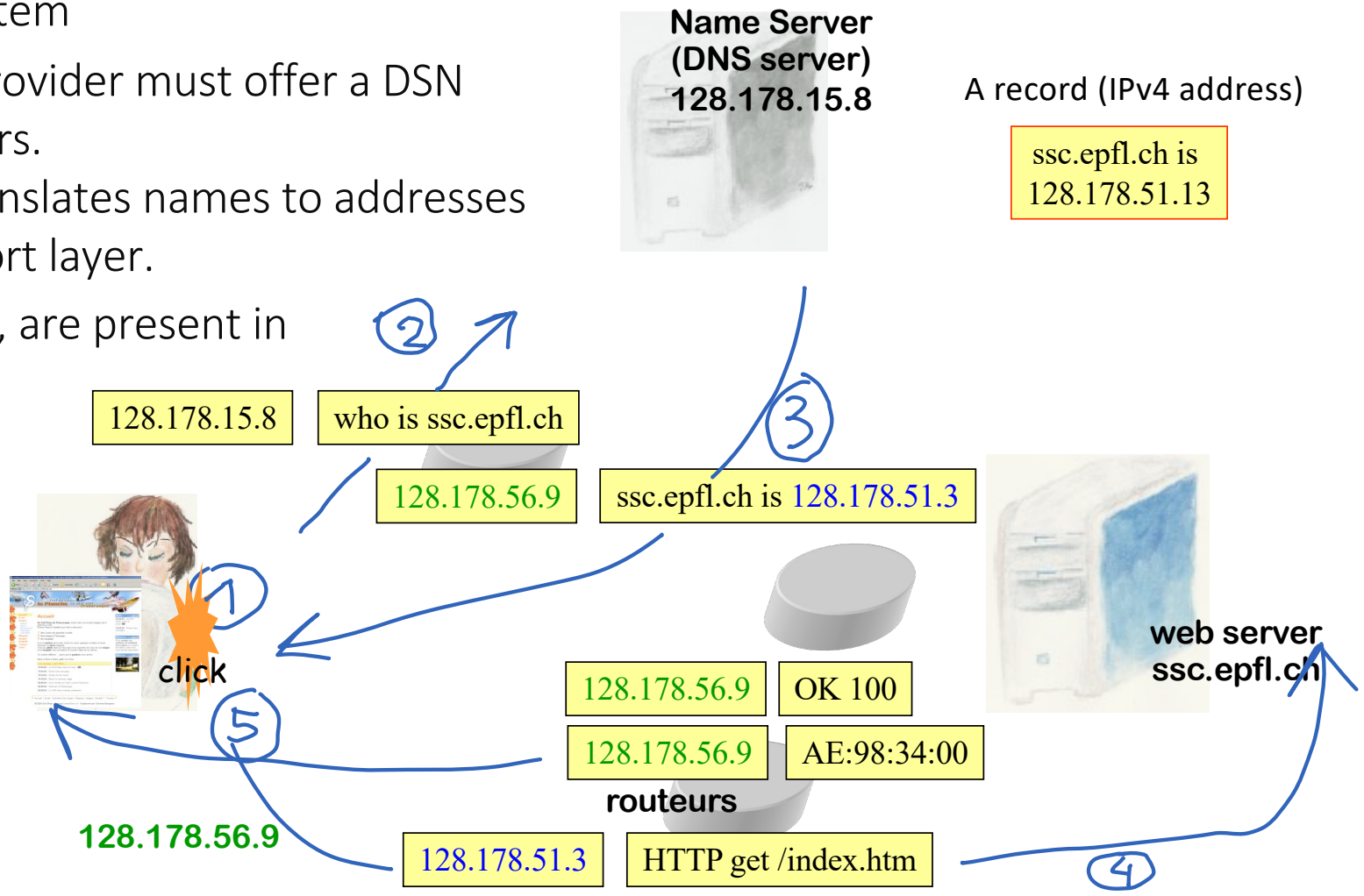


Names are mapped to addresses by DNS servers

DNS = domain name system

Every Internet service provider must offer a DSN service to their customers.
The application layer translates names to addresses before using the transport layer.

IP addresses, not names, are present in IP packet headers.



Try:
Windows, Mac, Linux:
nslookup
Smartphone: HE.NET app

ping

The ping command (windows, Mac, Linux, HE.NET app) sends a test message to a system identified by an IP address or a name

```
leboudec@icsil1noteb240 ~ % ping -c 3 www.canterbury.ac.nz
PING www.canterbury.ac.nz (132.181.106.9): 56 data bytes
64 bytes from 132.181.106.9: icmp_seq=0 ttl=223 time=302.115 ms
64 bytes from 132.181.106.9: icmp_seq=1 ttl=223 time=301.977 ms
64 bytes from 132.181.106.9: icmp_seq=2 ttl=223 time=302.241 ms
--- www.canterbury.ac.nz ping statistics ---
3 packets transmitted, 3 packets received, 0.0% packet loss
round-trip min/avg/max/stddev = 301.977/302.111/302.241/0.108 ms

leboudec@icsil1noteb240 ~ % ping -c 3 www.ethz.ch
PING www.ethz.ch (129.132.19.216): 56 data bytes
64 bytes from 129.132.19.216: icmp_seq=0 ttl=48 time=4.517 ms
64 bytes from 129.132.19.216: icmp_seq=1 ttl=48 time=4.356 ms
64 bytes from 129.132.19.216: icmp_seq=2 ttl=48 time=4.311 ms
--- www.ethz.ch ping statistics ---
3 packets transmitted, 3 packets received, 0.0% packet loss
round-trip min/avg/max/stddev = 4.311/4.395/4.517/0.088 ms
```

traceroute

The traceroute command (tracert in windows) detects the routers in the path between this system and a destination.

```
leboudec@icsil1noteb240 ~ % traceroute  www.canterbury.ac.nz
traceroute to www.canterbury.ac.nz (132.181.106.9), 64 hops max, 52 byte packets
 1  cv-dist-b-v151-ro (128.178.151.1)  0.598 ms  0.349 ms  0.241 ms
 2  cv-core-dist-b-168 (10.0.2.168)  0.335 ms  0.352 ms  0.307 ms
 3  forti-core-epfl-6 (10.0.2.6)  0.266 ms  0.204 ms  0.255 ms
 4  cv-core-intranet-181 (10.0.2.181)  0.449 ms  0.531 ms  0.397 ms
 5  cv-internet-intranet-127 (10.0.2.127)  0.814 ms  0.478 ms  0.400 ms
 6  forti-internet-intranet-178 (10.0.2.178)  0.254 ms  0.264 ms  0.263 ms
 7  cv-internet-internet-133 (10.0.2.133)  0.799 ms  0.740 ms  0.767 ms
 8  swiel3 (192.33.209.17)  0.954 ms  1.075 ms  1.057 ms
 9  swige3-100ge-0-0-1-1.switch.ch (130.59.36.82)  2.103 ms  2.017 ms  2.038 ms
10  swice2-100ge-0-0-0-10.switch.ch (130.59.38.193)  2.241 ms  3.127 ms  4.041 ms
11  switch.mx1.gen.ch.geant.net (62.40.124.21)  3.247 ms  1.872 ms  34.086 ms
12  ae7.mx1.par.fr.geant.net (62.40.98.239)  9.890 ms  9.817 ms  9.745 ms
13  hundredge-0-0-0-22.102.core1.newy32aoa.net.internet2.edu (198.71.45.236)  83.302 ms  86.952 ms  83.599 ms
14  fourhundredge-0-0-0-2.4079.core1.ashb.net.internet2.edu (163.253.1.116)  158.890 ms  158.650 ms  157.387 ms
15  fourhundredge-0-0-0-18.4079.core2.ashb.net.internet2.edu (163.253.1.107)  158.884 ms  158.413 ms  158.692 ms
16  fourhundredge-0-0-0-1.4079.core2.clev.net.internet2.edu (163.253.1.139)  158.654 ms  158.798 ms  158.848 ms
17  fourhundredge-0-0-0-2.4079.core2.eqch.net.internet2.edu (163.253.2.17)  182.013 ms  173.136 ms  198.211 ms
18  fourhundredge-0-0-0-2.4079.core2.chic.net.internet2.edu (163.253.2.18)  160.076 ms  158.350 ms  158.850 ms
19  fourhundredge-0-0-0-1.4079.core1.kans.net.internet2.edu (163.253.1.245)  158.972 ms  160.021 ms  160.103 ms
20  fourhundredge-0-0-0-1.4079.core1.denv.net.internet2.edu (163.253.1.242)  160.490 ms  158.259 ms  158.155 ms
21  fourhundredge-0-0-0-3.4079.core1.salt.net.internet2.edu (163.253.1.171)  159.987 ms  157.754 ms  159.299 ms
22  fourhundredge-0-0-0-1.4079.core1.seat.net.internet2.edu (163.253.1.157)  159.423 ms  158.744 ms  158.431 ms
23  fourhundredge-0-0-0-16.4079.core2.seat.net.internet2.edu (163.253.2.55)  159.511 ms  158.893 ms  158.472 ms
24  reannz-1-lo-jmb-706.sttlwa.pacificwave.net (207.231.240.33)  156.902 ms  157.074 ms  156.801 ms
25  210.7.37.209 (210.7.37.209)  302.161 ms  302.010 ms  302.132 ms
26  210.7.37.210 (210.7.37.210)  384.072 ms  319.099 ms  320.738 ms
27  202.36.179.65 (202.36.179.65)  302.168 ms  302.182 ms  302.007 ms
28  132.181.3.236 (132.181.3.236)  303.059 ms  302.994 ms  304.032 ms
29  132.181.106.9 (132.181.106.9)  302.111 ms  302.115 ms  302.263 ms
```

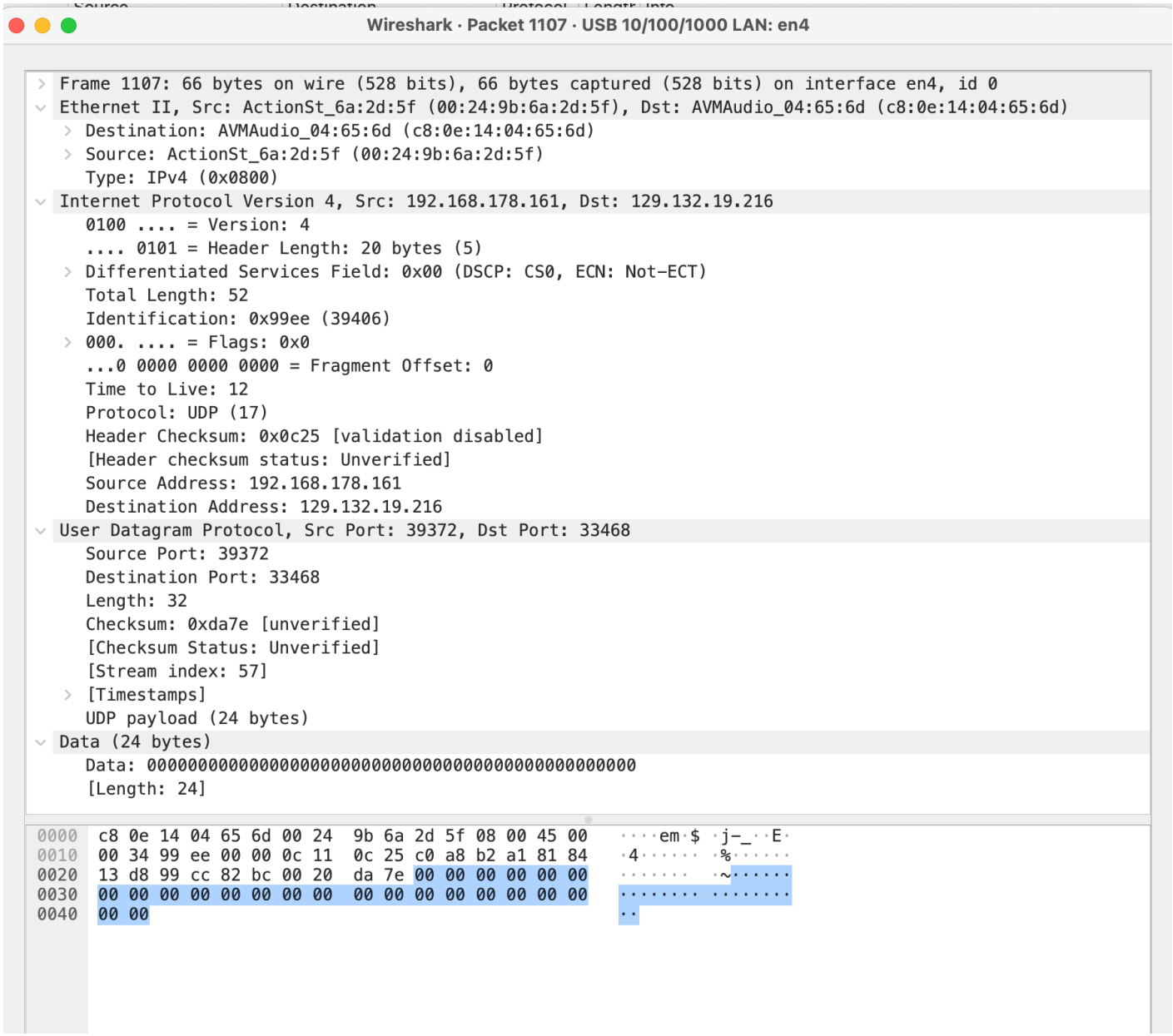
Try:

www.wireshark.org

```
leboudec@icsil1noteb240 ~ %
traceroute  www.ethz.ch
traceroute to www.ethz.ch (129.132.19.216),
64 hops max, 52 byte packets
```

then wireshark with display filter:

ip.addr == 129.132.19.216



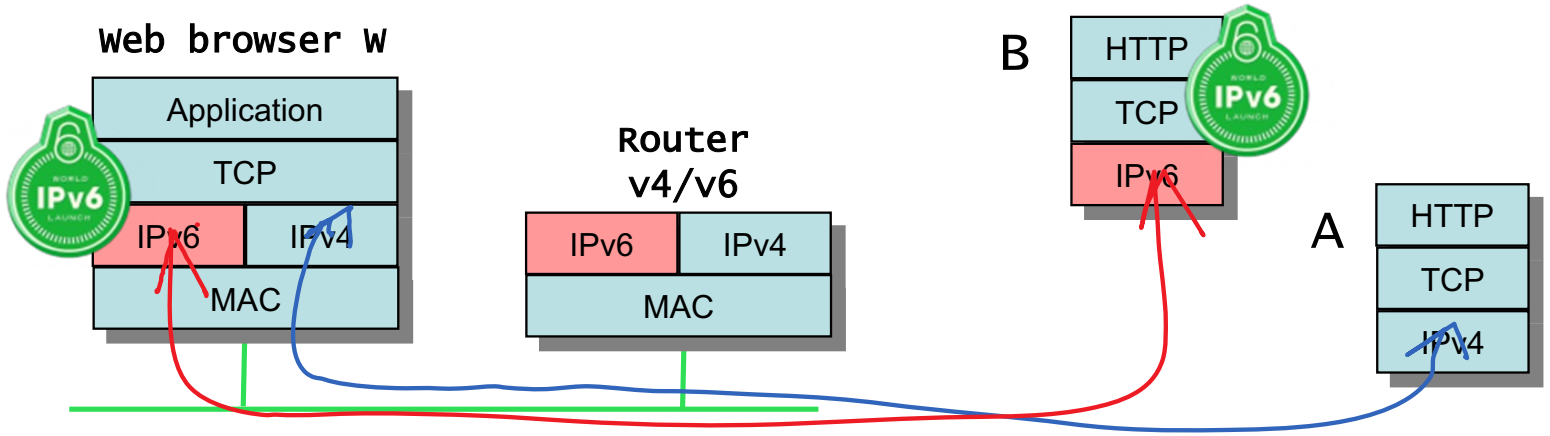
4. IPv6

The most widespread version of IP is IPv4, the most recent one is IPv6.

Why a new version ? The IPv4 address space has $2^{32} \approx 4 \cdot 10^9$addresses and is exhausted (no new address can be allocated).

How ? IPv6 redefines the packet format and extends the address to 128 bits ($\approx 3 \cdot 10^{38}$addresses). Otherwise, is the same as IPv4 (except for some details).

But IPv6 is incompatible with IPv4: W can communicate with A and B but A and B cannot communicate directly. Up-to-date systems support both IPv4 and IPv6, but some networks support only IPv6 (or IPv4).



IPv6 addresses

128 bits = 8 groups of 16 bits = 8 groups of 4 hexa digits.
A public address at EPFL: 2001:0620:0618:01a6:0a00:20ff:fe78:30f9
A private address at EPFL: fd24:ec43:12ca:01a6:0a00:20ff:fe78:30f9
This host (localhost): 0000:0000:0000:0000:0000:0000:0000:0001

Compression rules for writing and displaying addresses:

- Groups (16 bits = 4 hexa digits) are separated by colons (:)
- leading zeroes can be omitted
- :: replaces any number of groups equal to 0; may appear at most once

uncompressed	*compressed*
2002:0000:0000:0000:0000:ffff:80b2:0c26	2002::ffff:80b2:c26
2001:0620:0618:01a6:0000:20ff:fe78:30f9	2001:620:618:1a6::20ff:fe78:30f9
0000:0000:0000:0000:0000:0000:0000:0001	::1

Link-local addresses

With IPv6, every interface has several addresses. It has at least:

- a global, configured e.g. manually or automatically e.g. with DHCP
- a link-local address, obtained without configuration, of the form fe80::xxxx:xxxx:xxxx:xxx.

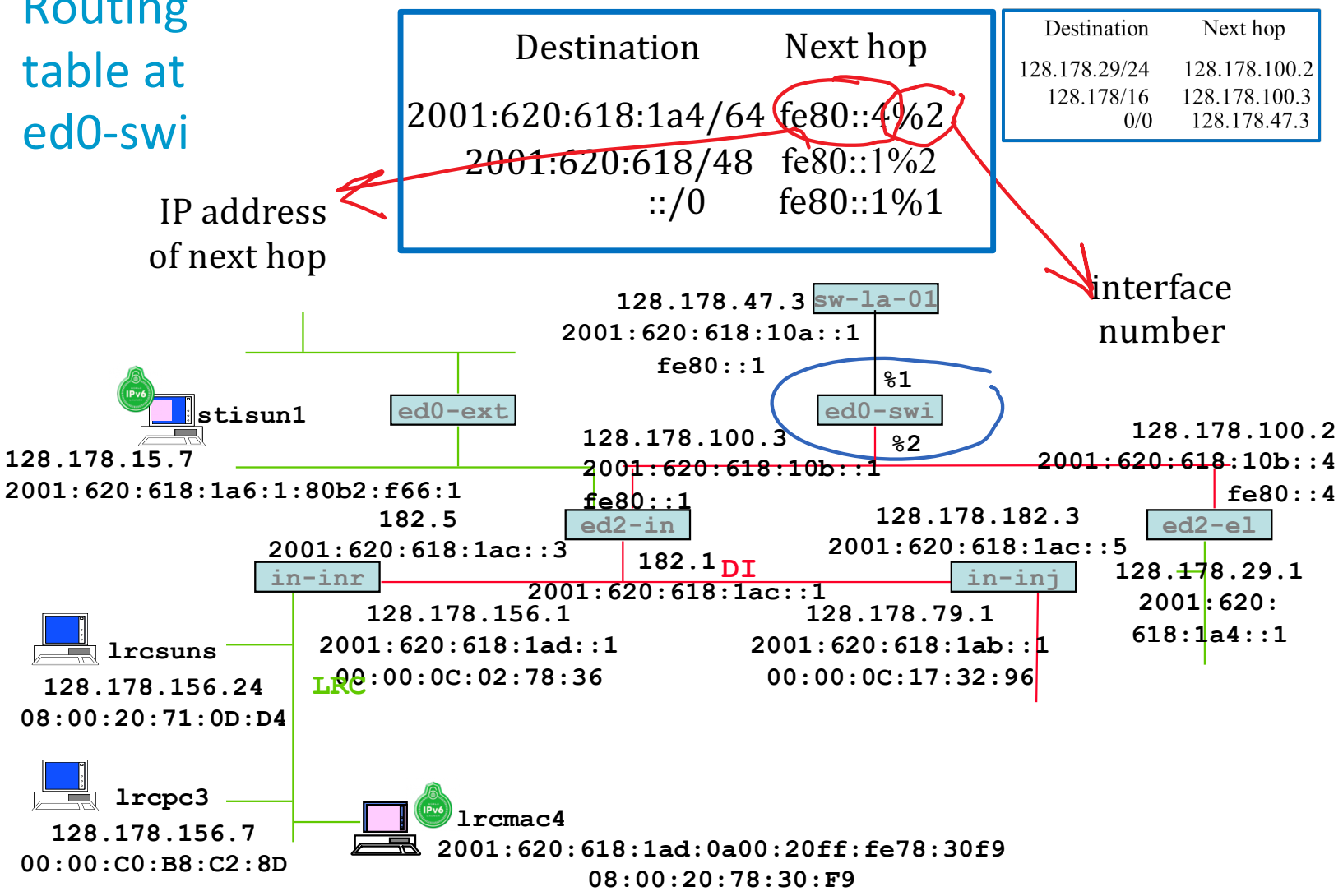
The link-local address can be used only within one subnet. A packet with a link-local destination address is never forwarded by a router. It is used to perform automatic configuration tasks, and also to operate plug-and-play when there is no router

```
leboudec@icsil1noteb240 ~ % ifconfig
en1: flags=8863<UP,BROADCAST,SMART,RUNNING,SIMPLEX,MULTICAST> mtu 1500
ether 14:98:77:64:d7:48
inet6 fe80::cdb:b44a:3a8c:3c7e%en1 prefixlen 64 secured scopeid 0xd
inet 128.179.167.28 netmask 0xfffff800 broadcast 128.179.167.255
inet6 2001:620:618:5a0:2:80b3:0:157 prefixlen 64 optimistic dynamic
```

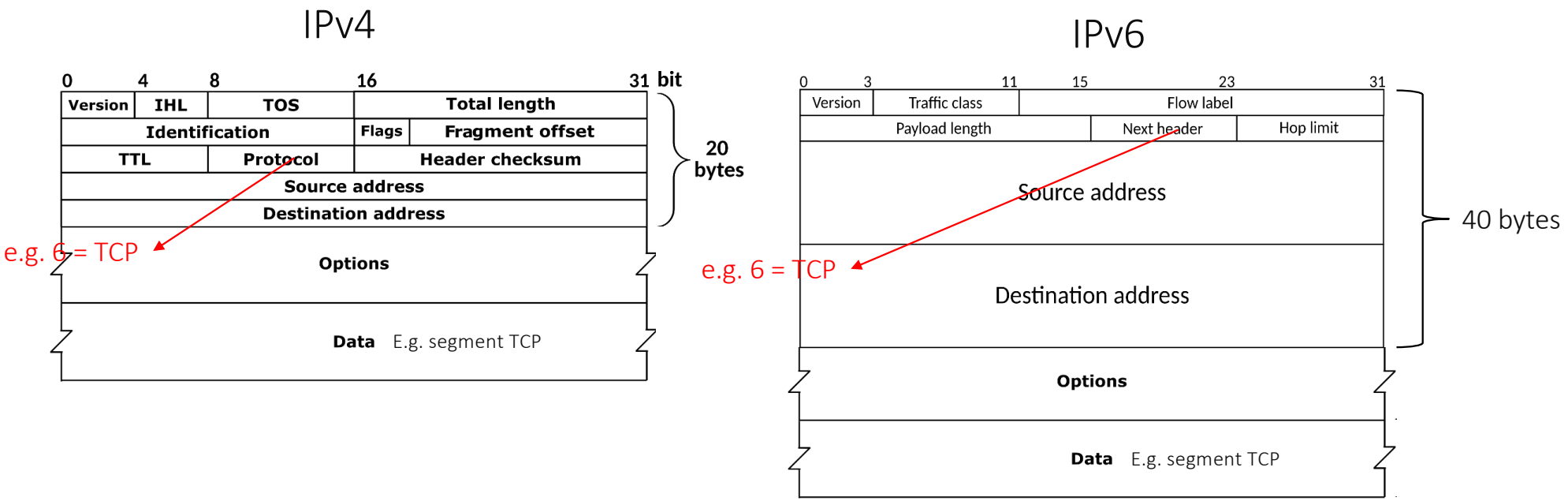
Lookup your IP addresses:

- Windows: ipconfig
- Mac, Linux: ifconfig
- Smartphone: HE.NET app

Routing table at ed0-swi



IP Packet Format



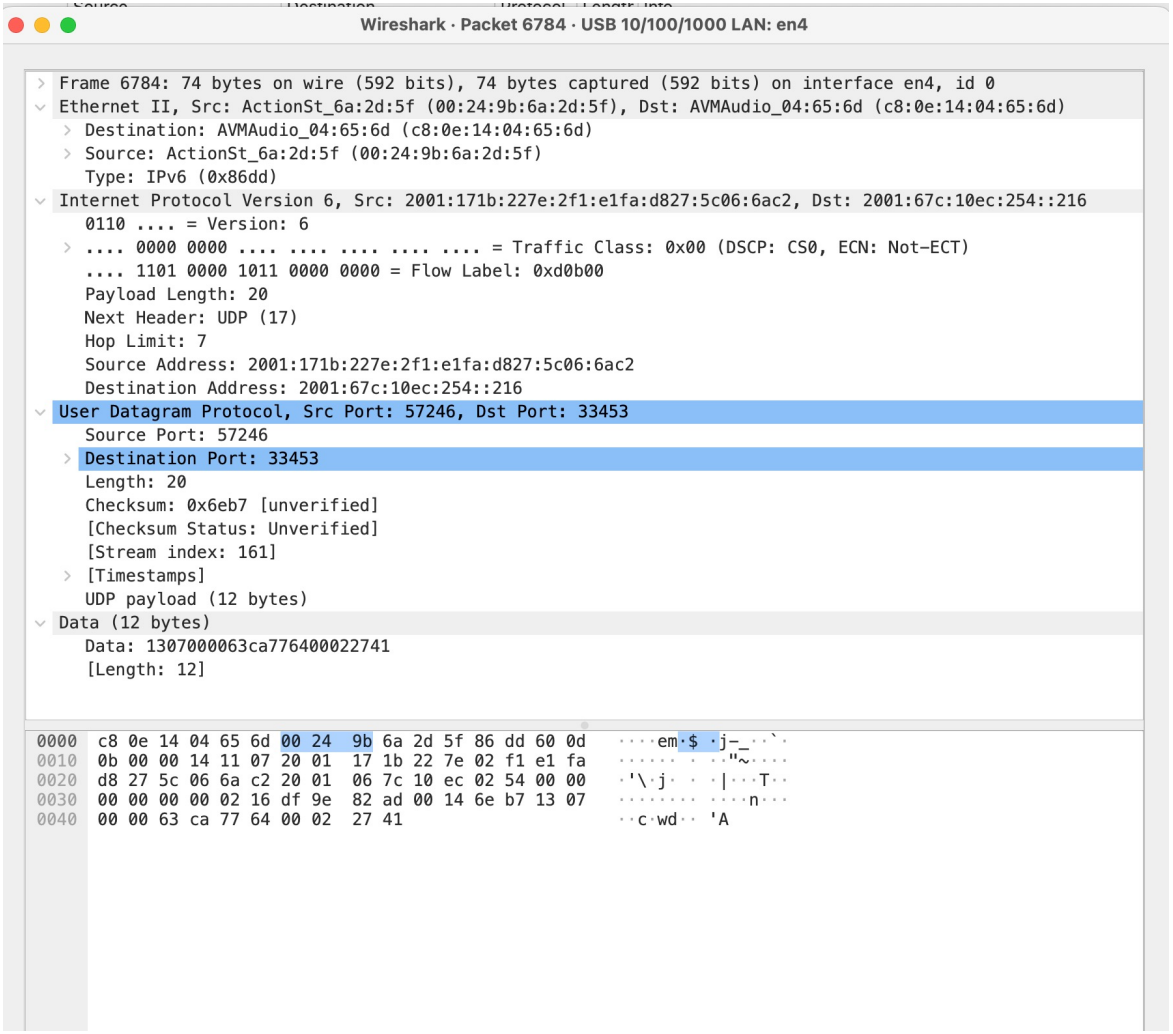
The header contains 20 bytes (IPv4) / 40 bytes (IPv6) + options.

The TTL/HL field (time-to-live/ hop limit) is set to 64 (typically) at source and decremented by 1 by every router. When it reaches 0 the packet is discarded – used to flush looping packets, also used by traceroute.

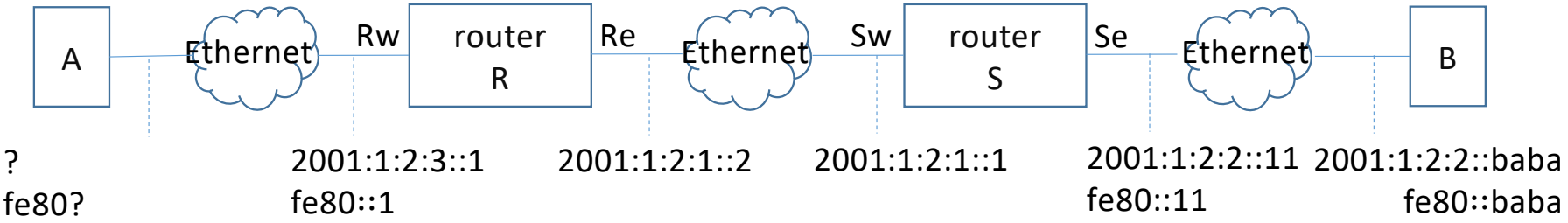
```
leboudec@icsil1noteb240 ~ % traceroute6  www.ethz.ch
traceroute6 to www.ethz.ch (2001:67c:10ec:254::216)
from 2001:171b:227e:2f1:e1fa:d827:5c06:6ac2, 64 hops
max, 12 byte packets
```

then wireshark with display filter:

ipv6.addr == 2001:67c:10ec:254::216

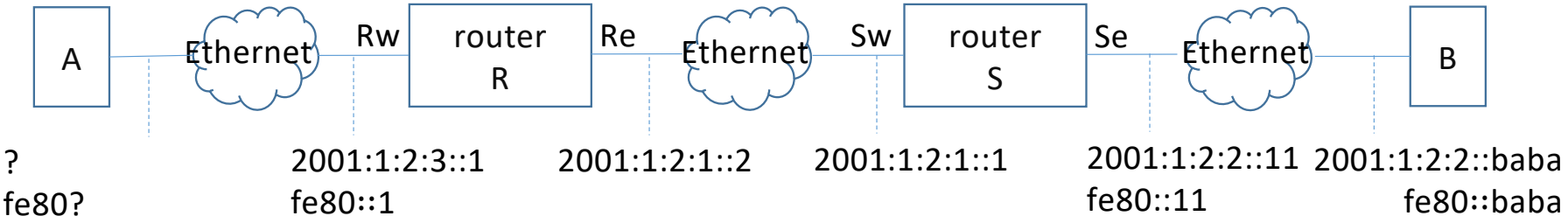


All subnet mask lengths are 64. Which is a valid address for A ?



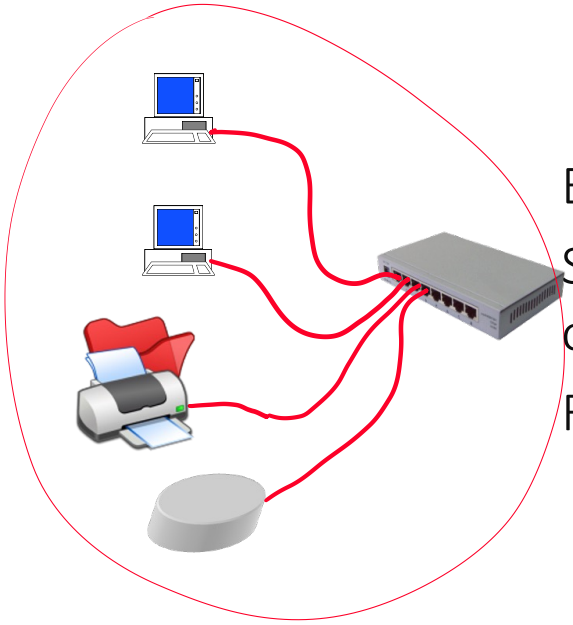
A. 2001:1:2:2::2
B. 2001:1:2:3::2
C. A and B
D. None
E. I don’t know

Which are valid addresses for A ?



A. fe80::1
B. fe80::2
C. fe80::11
D. A et B
E. A et C
F. B et C
G. All
H. None
I. I don’t know

A box interconnects n systems running IPv4; it is configured either as an Ethernet switch or as a router. Then we migrate all devices from IPv4 to IPv6. Should the box also be reconfigured ?



Ethernet

Switch

or

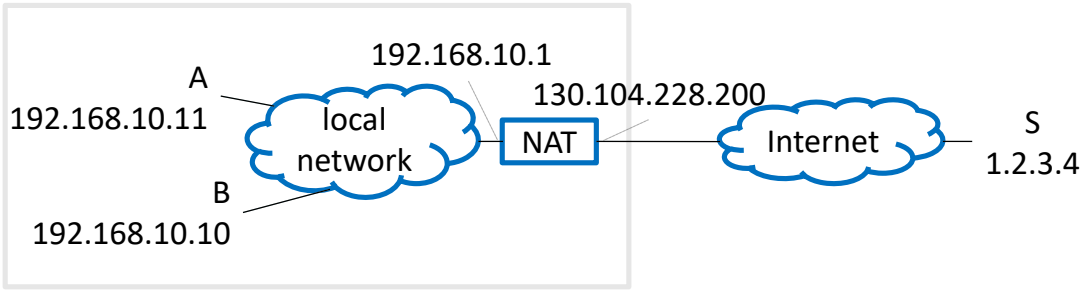
Router

A. Yes if it is a router, no if it is an Ethernet switch
B. No if it is a router, yes if it is an Ethernet switch
C. Yes in both cases
D. No in both cases
E. I don't know

5. NATs (“Network Address Translation”)

Why ? Use n devices with only one IPv4 address.

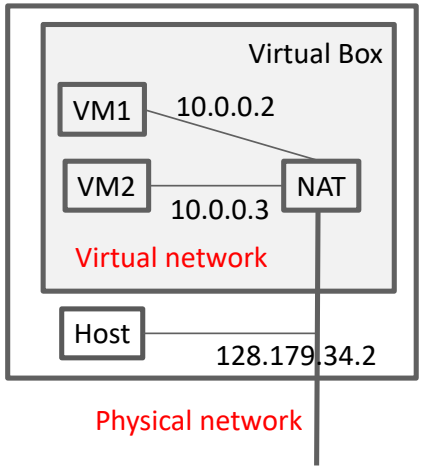
Allow virtual machines to use the host’s physical interface.



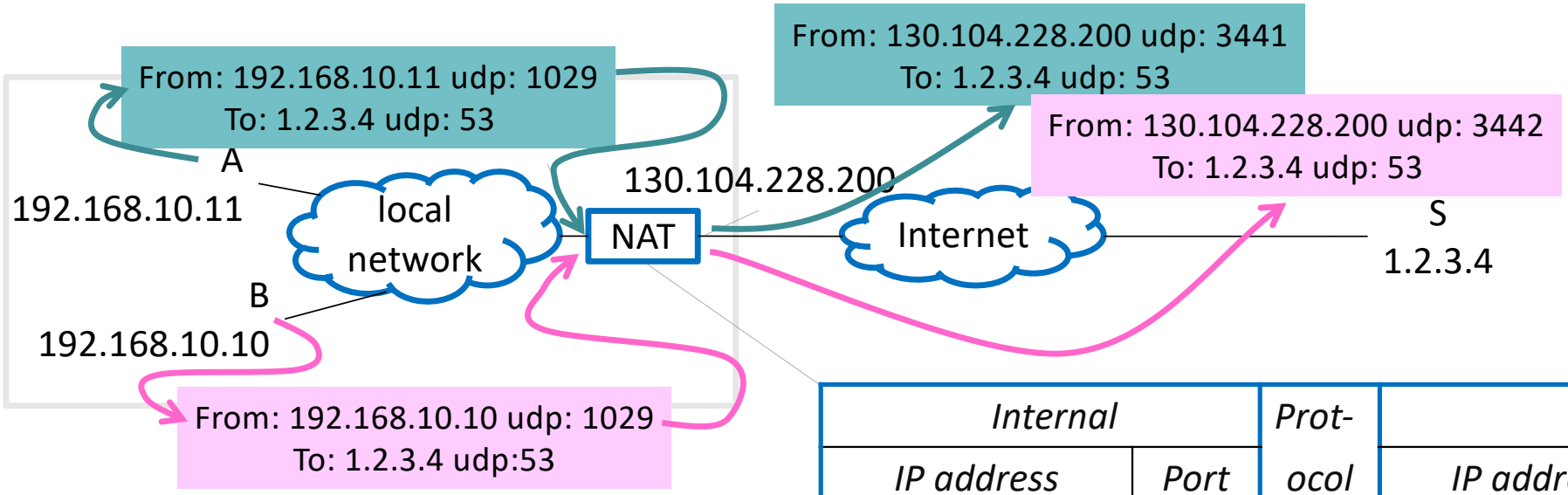
The NAT allows to

- share one IP address between several machines,
- use private addresses to connect to the public internet

It is a deviation from IP’s original principles.



The NAT translates IP addresses and cheats with ports



NAT Table

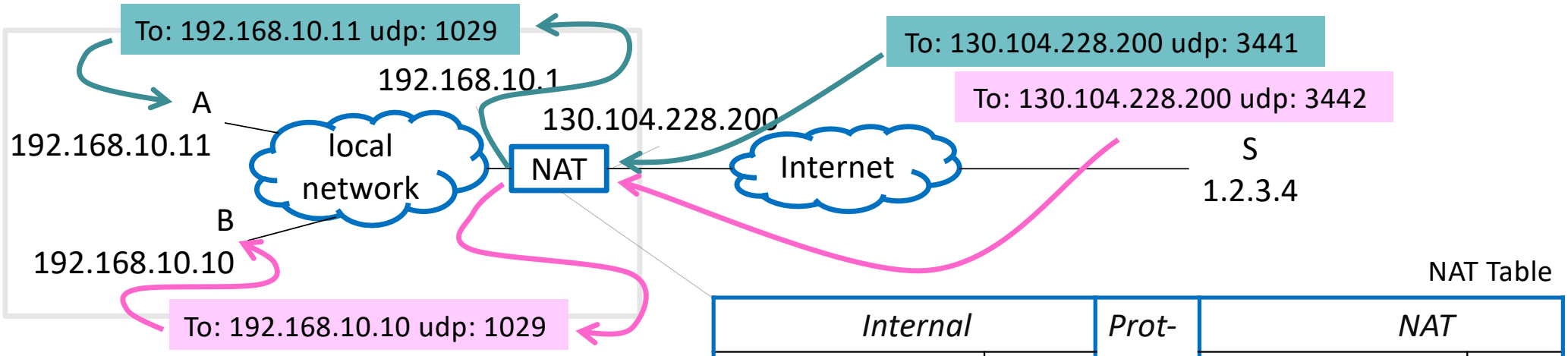
Internal		*Prot-ocol*	*NAT*	
IP address	*Port*		*IP address*	*Port*
192.168.10.11	1029	udp	130.104.228.200	3441
192.168.10.10	1029	udp	130.104.228.200	3442

The NAT replaces A's source IP address with its own IP address.

Problem: A et B now appear to have the same IP address !

Solution: manipulate port numbers (which belong to transport layer !)

The NAT translates IP addresses (continued)



NAT Table

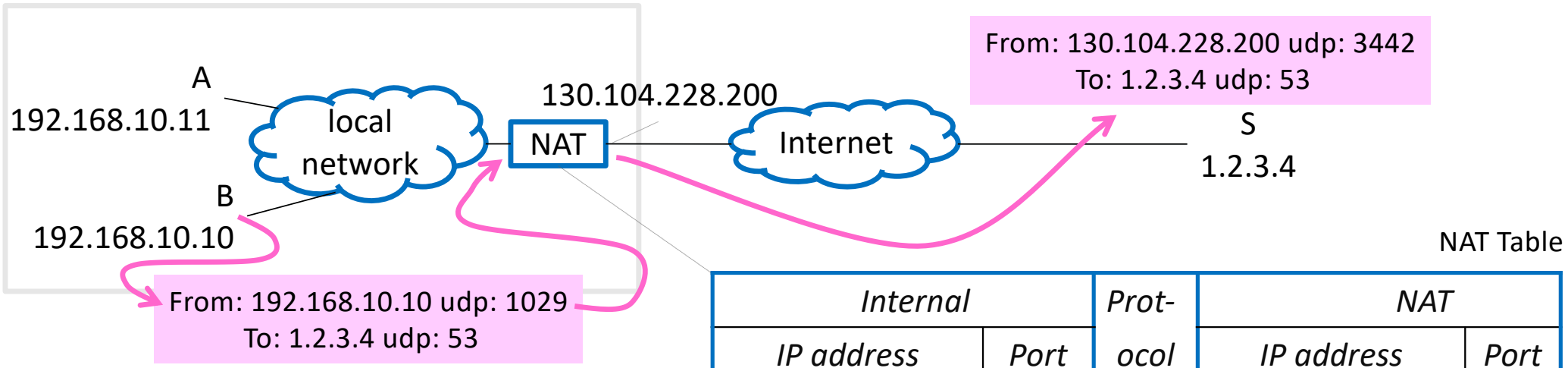
Internal		*Prot-ocol*	*NAT*	
IP address	*Port*		*IP address*	*Port*
192.168.10.11	1029	udp	130.104.228.200	3441
192.168.10.10	1029	udp	130.104.228.200	3442

From external to internal, the NAT translates destination IP addresses and port numbers.

To S, A and B appear as if they were two processes on one machine.

The NAT is often called “router” but formally it is not (a router is an intermediate system of the IP layer).

The NAT works automatically if communication is initiated from inside

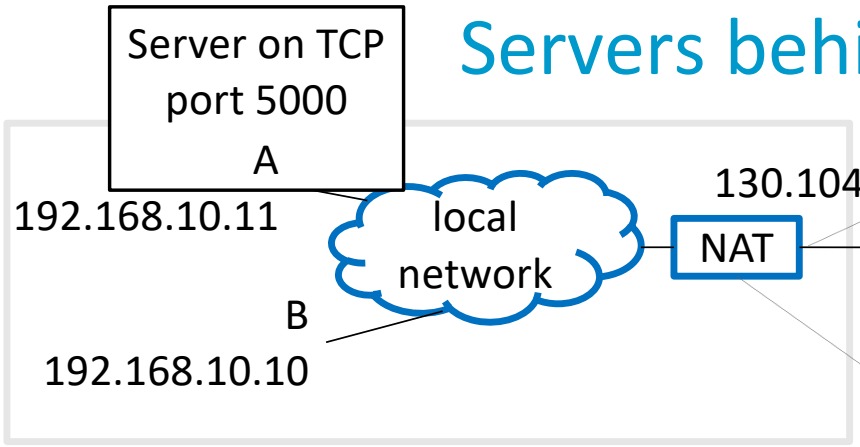


NAT Table

Internal		*Prot-ocol*	*NAT*	
IP address	*Port*		*IP address*	*Port*
192.168.10.11	1029	udp	130.104.228.200	3441
192.168.10.10	1029	udp	130.104.228.200	3442

When the NAT sees a packet from 192.168.10.10 udp: 1029 for the first time, it chooses a port number that is not yet used and creates a new line in its NAT table. After some inactivity time, the entry is deleted.

Servers behind a NAT



NAT Table

Internal		*Prot-*	*NAT*	
IP address	*Port*	*ocol*	*IP address*	*Port*
192.168.10.11	1029	udp	130.104.228.200	3441
192.168.10.10	1029	udp	130.104.228.200	3442
192.168.10.11	5000	tcp	130.104.228.200	5000

If there is a server program at A, communication is initiated by C. The NAT cannot learnt, it must be configured (port forwarding).

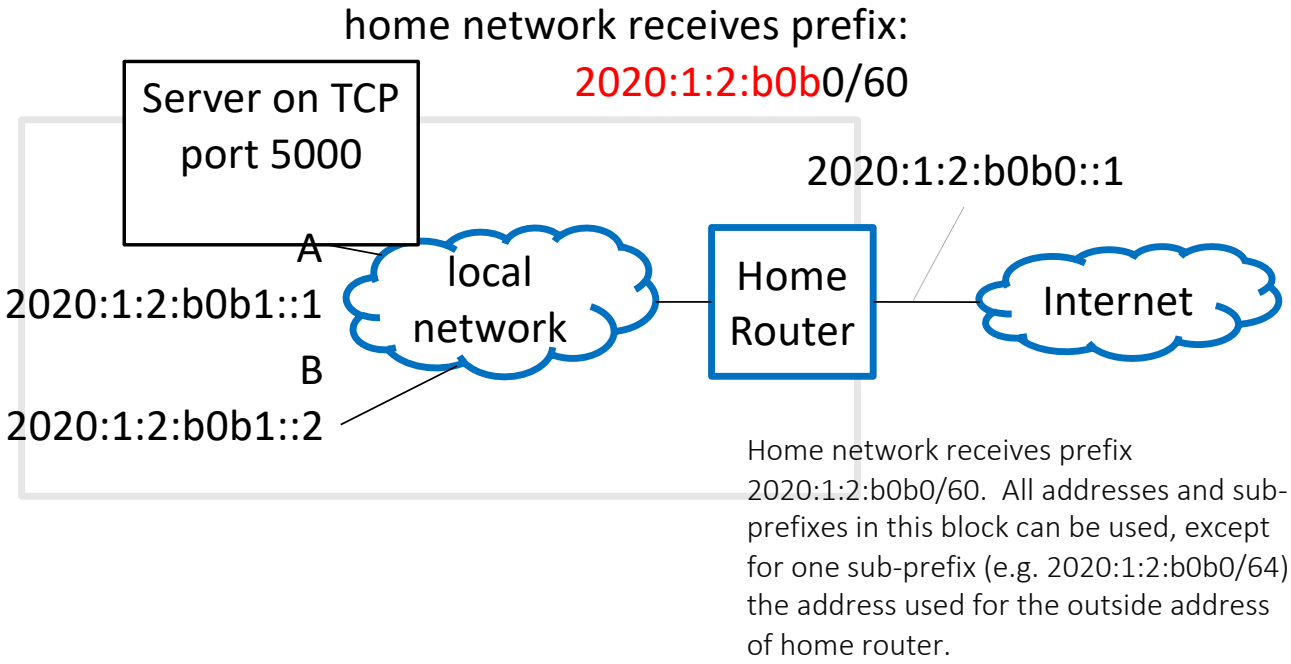
Incidentally, this brings a level of protection: only server ports configured in the NAT can be accessed from outside.

Setup by manual configuration of *port forwarding* on NAT

IPv6 NATs

NATs were developed because of IPv4 address scarcity.

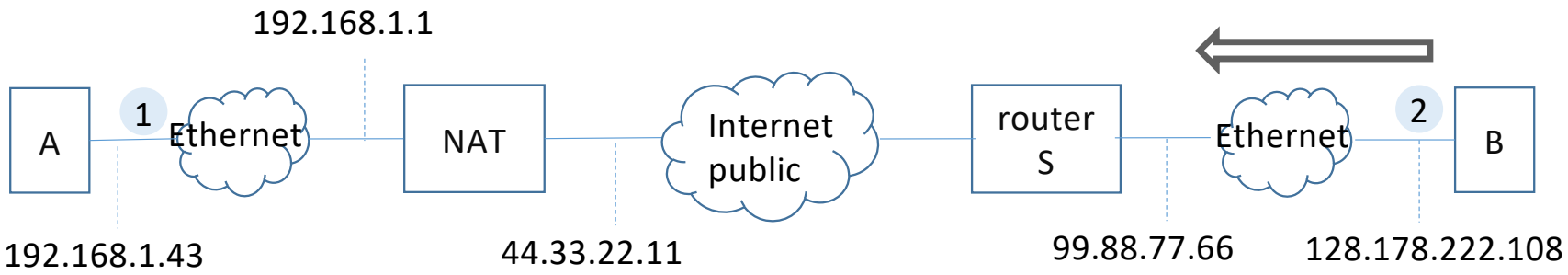
With IPv6, home routers need not operate as NATs, since the home network typically receives not one address but an entire prefix.



The “home router” works as a true router. It also implements port filtering: communication initiated from the outside is possible only if explicitly authorized.

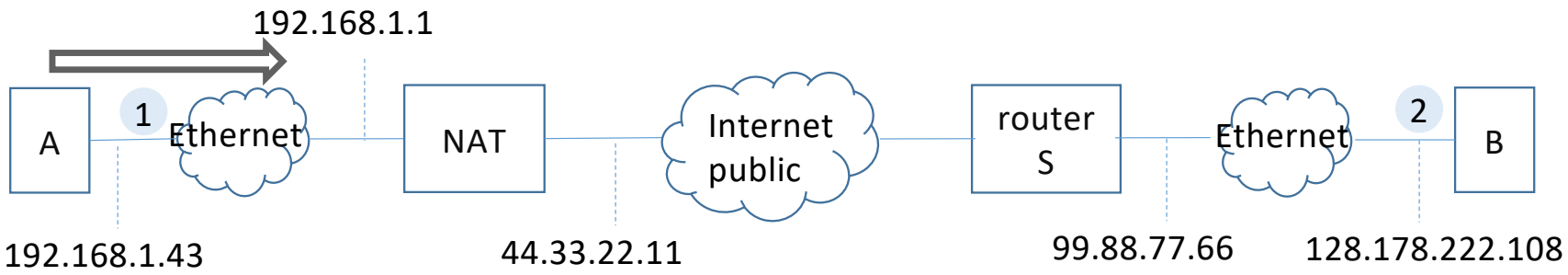
In virtualization environments, IPv6 NATs are often used as the physical host receives only one IP6 address.

B sends IP packets to A. What is the IP destination address observed at (2) ?



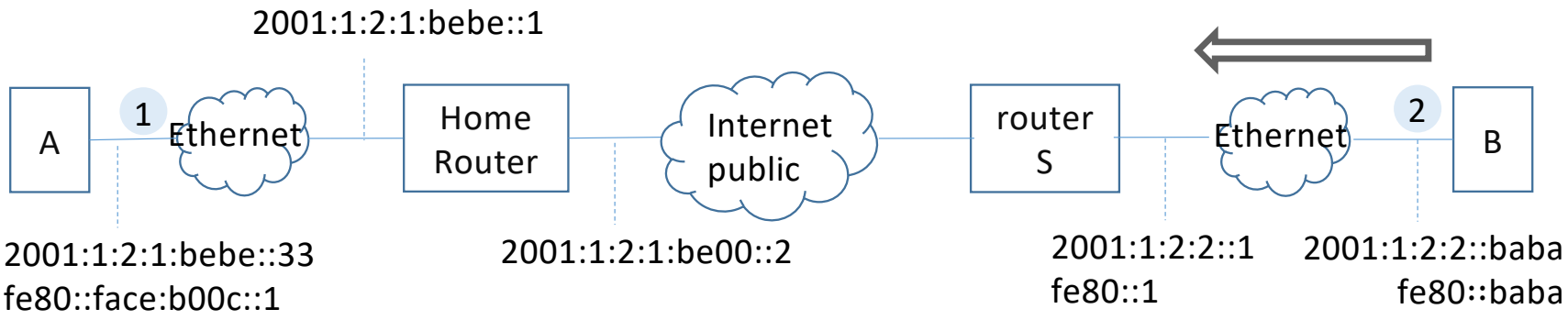
A. 99.88.77.6
B. 44.33.22.11
C. 192.168.1.43
D. None of these
E. I don’t know

A sends IP packets to B. What is the IP destination address observed at (2) ?



A. 99.88.77.6
B. 44.33.22.11
C. 128.178.222.108
D. None of these
E. I don’t know

B sends IP packets to A. What is the IP destination address observed at (2) ?



A. 2001:1:2:2::1
B. fe80::1
C. 2001:1:2:1:be00::2
D. 2001:1:2:1:bebe::33
E. fe80::face:b00c::1
F. None of these
G. I don't know

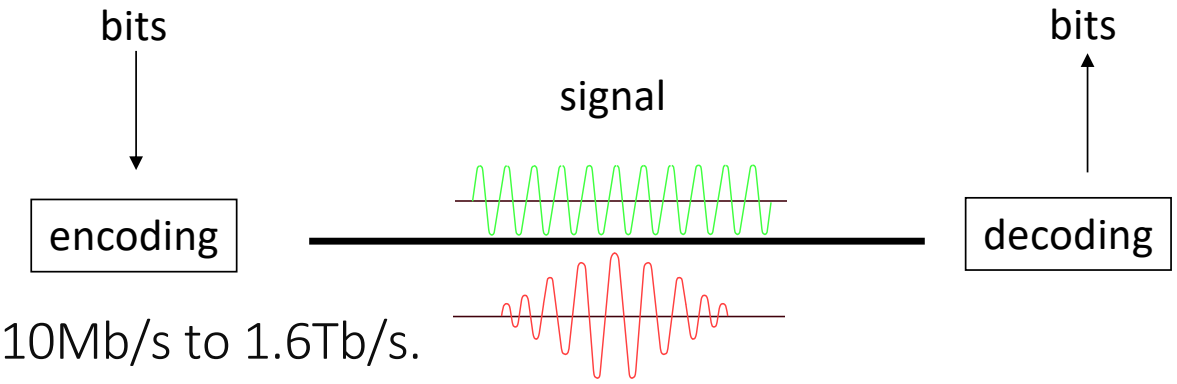
Solution

Answer B: A's address is a private address and cannot be used outside A's private network. To the outside, A appears as being "inside" the NAT.

Answer C: B's address is public and is not modified by the NAT.

Answer D: here there is no NAT and A's address is public. A's link-local address cannot be used because A and B are not in the same subnet.

6. The Physical Layer



It maps blocks of bits (from the MAC layer) into a physical signal, typically an electromagnetic signal.

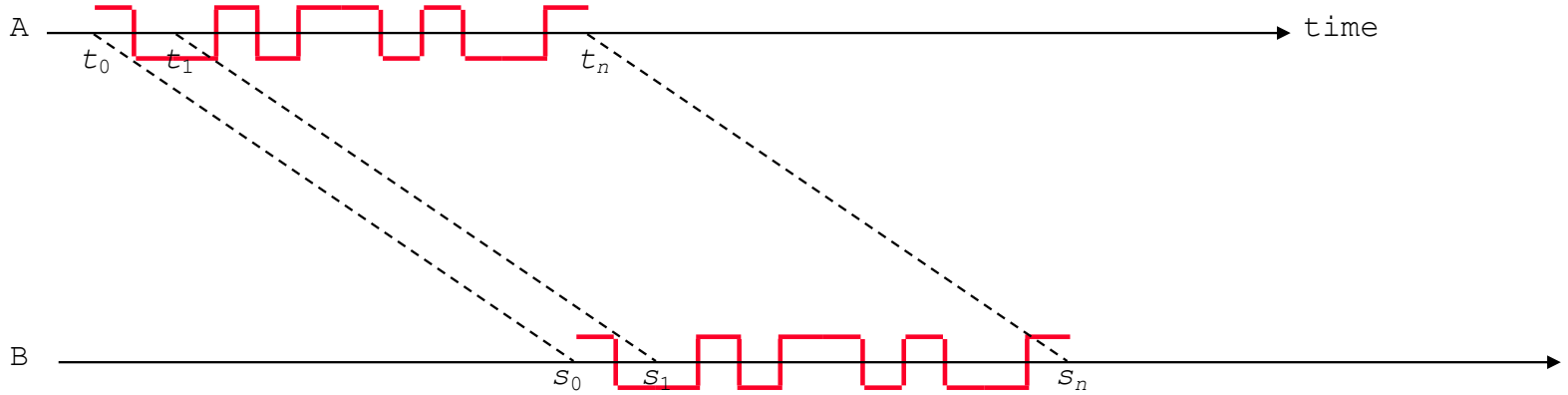
Evolves with technology: ex: Ethernet 10Mb/s to 1.6Tb/s.

The bit rate is the number of bits transmitted per second ≠ bandwidth. For a simple channel with only disturbance being gaussian noise, Information Theory gives a tight upper bound on the bit rate $C_{\max} = B \log_2(1 + \mathrm{SNR})$, where B is the bandwidth (in Hz), $C_{\max}$ in b/s and SNR is the signal-to-noise ratio (unitless).

Example: ADSL line of fair quality: B= 2 MHz, SNR=50 dB, $C_{\max}$ =33.2 Mb/s.

Try: https://www.cnlab.ch/speedtest

Propagation



Propagation: $s_n - t_n = s_0 - t_0$ is given by the speed of light in the channel: c =3E+08m/s in vacuum or air, c = 2.3E+08 m/s in copper, c = 2eE08 m/s in glass optical fibers (round the earth in 200ms).

Transmission: $t_n - t_0$ is equal to the number of bits to be transmitted divided by the bit rate. Depends on techonology.
Total one-way delay $t_n - t_0 =$ propagation + transmission

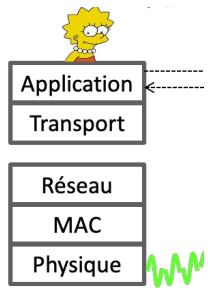
The **ping** command measures the two-way delay, including propagation , transmission, processing by the destination and queuing delays in switches and routers.

How long does it take to send 1kB (8000bits) ?

	data center	*ADSL*	*modem*	*Internet*
distance	20 m	2 km	20 km	20'000 km
bit rate	1Tb/s	10Mb/s	10kb/s	1Mb/s
propagation	0.1μs	0.01ms	0.1ms	100ms
transmission	0.008 μs	0.8ms	800ms	8ms
total	0.108 μs	0.81ms	800.1ms	108ms

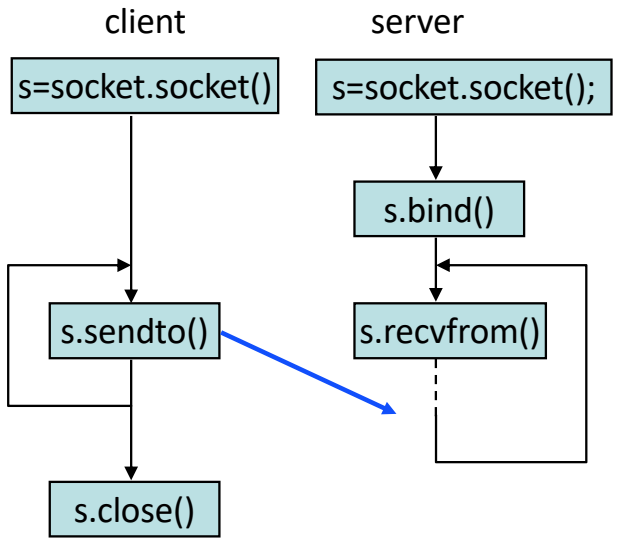
```
leboudec@icsil1noteb240 ~ % ping -c 3
www.canterbury.ac.nz
64 bytes from 132.181.106.9: icmp_seq=0 ttl=223 time=302.035 ms
64 bytes from 132.181.106.9: icmp_seq=1 ttl=223 time=301.972 ms
64 bytes from 132.181.106.9: icmp_seq=2 ttl=223 time=301.911 ms

leboudec@icsil1noteb240 ~ % ping -c 3 www.nzherald.co.nz
64 bytes from 195.176.255.75: icmp_seq=0 ttl=49 time=4.619 ms
64 bytes from 195.176.255.75: icmp_seq=1 ttl=49 time=4.418 ms
64 bytes from 195.176.255.75: icmp_seq=2 ttl=49 time=4.460 ms
```

7. The Transport Layer



Reminder: the transport layer offers a programming interface to the applications. UDP offers a basic service, TCP adds packet loss recovery.

UDP and TCP are accessed via the “socket” API in C, Python etc.



UDP Exemple: client sends a message, servers prints it on screen.
socket(socket.AF_INET,…) creates an IPv4 socket and returns a socket descriptor or an error code.
socket(socket.AF_INET6,…) creates an IPv6 socket.

bind() associates the socket with a local port number and an IP address– usually omitted at client;

sendto() sends a message to some IP address and port number, passed in argument;

recvfrom() blocks until one message has arrived, then returns the message, the source IP address port number.

Try it !

udpc.py

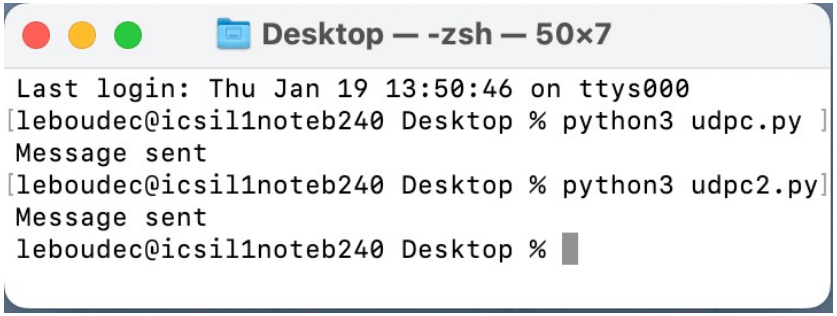
```
import socket
HOST = 'localhost' # The remote host
PORT = 50007 # The same port as used by the server
s = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
s.sendto(b'Bonjour les amis', (HOST,PORT) )
s.close()
print('Message sent')
```

udpc2.py

```
import socket
HOST = 'localhost' # The remote host
PORT = 50007 # The same port as used by the server
s = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
s.sendto(b'Hoi zamme', (HOST,PORT) )
s.close()
print('Message sent')
```

udps.py

```
import socket
HOST = '' # Symbolic name meaning all available interfaces
PORT = 50007 # Arbitrary non-privileged port
s = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
s.bind((HOST, PORT))
while True:
  data, addr = s.recvfrom(1024)
  print('From: ', addr)
  print('Received: ', data.decode('utf-8'))
```

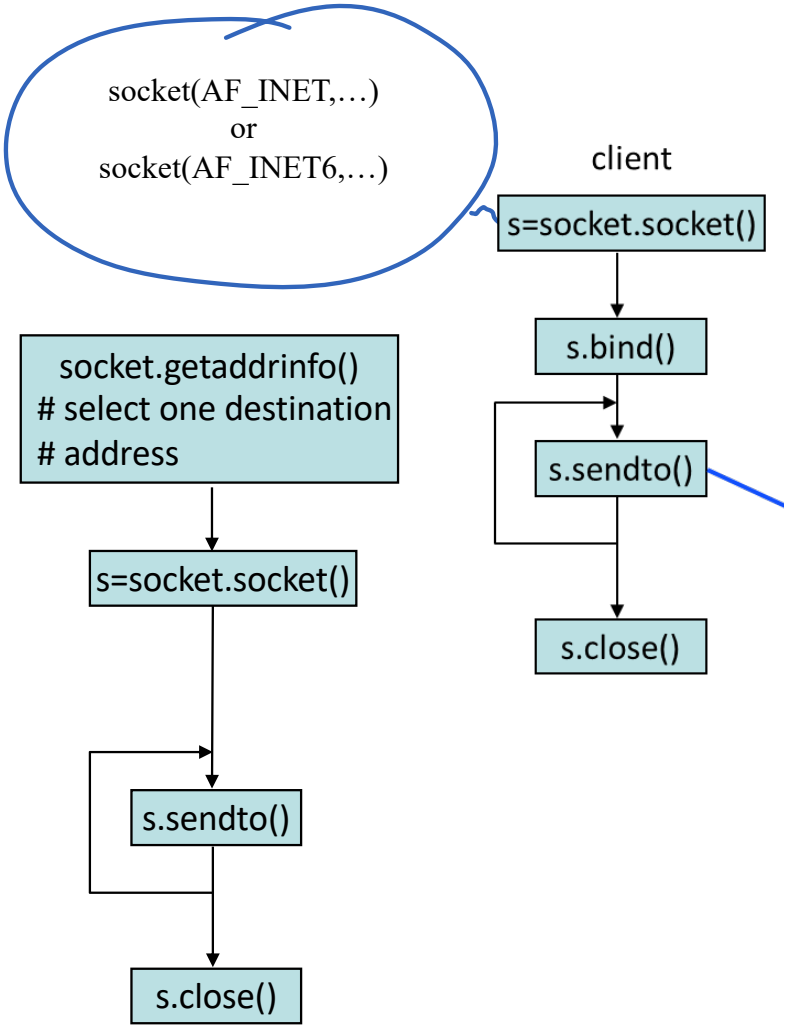


Is there UDPv6 ?

No, the transport layer is the same for IPv4 and IPv6:

But a socket program must know whether it is working with IPv4 or IPv6 addresses.

The application selects IPv4 or IPv6, based on what DNS returns.



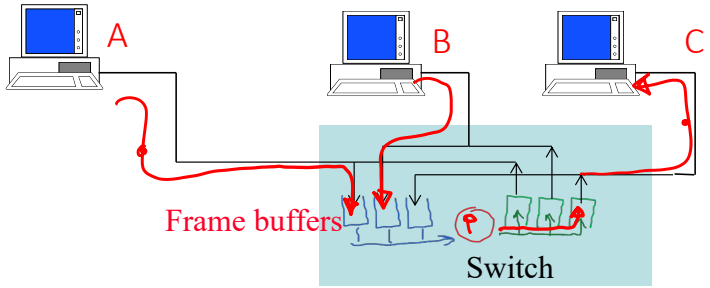
Essayez:

```
leboudec@lca2-2111021 ~ % python3
>>> import socket
>>> socket.getaddrinfo("lca.epfl.ch",None)
[(<AddressFamily.AF_INET6: 23>, 0, 0, '',
('2001:620:618:521:1:80b3:2127:1', 0, 0, 0)),
(<AddressFamily.AF_INET: 2>, 0, 0, '', ('128.179.33.39', 0))]
```

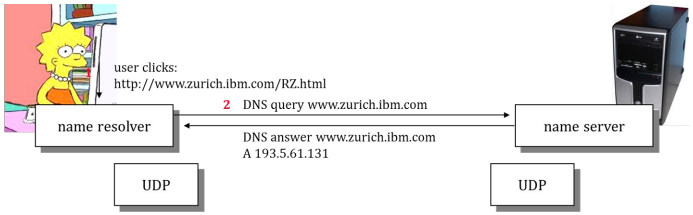
TCP

IP packets may be lost in flight:

- transmission errors (rare on cables, frequent on wireless)
- congestion: buffer overflow in switch or router buffers (normal)



UDP applications must manage losses.

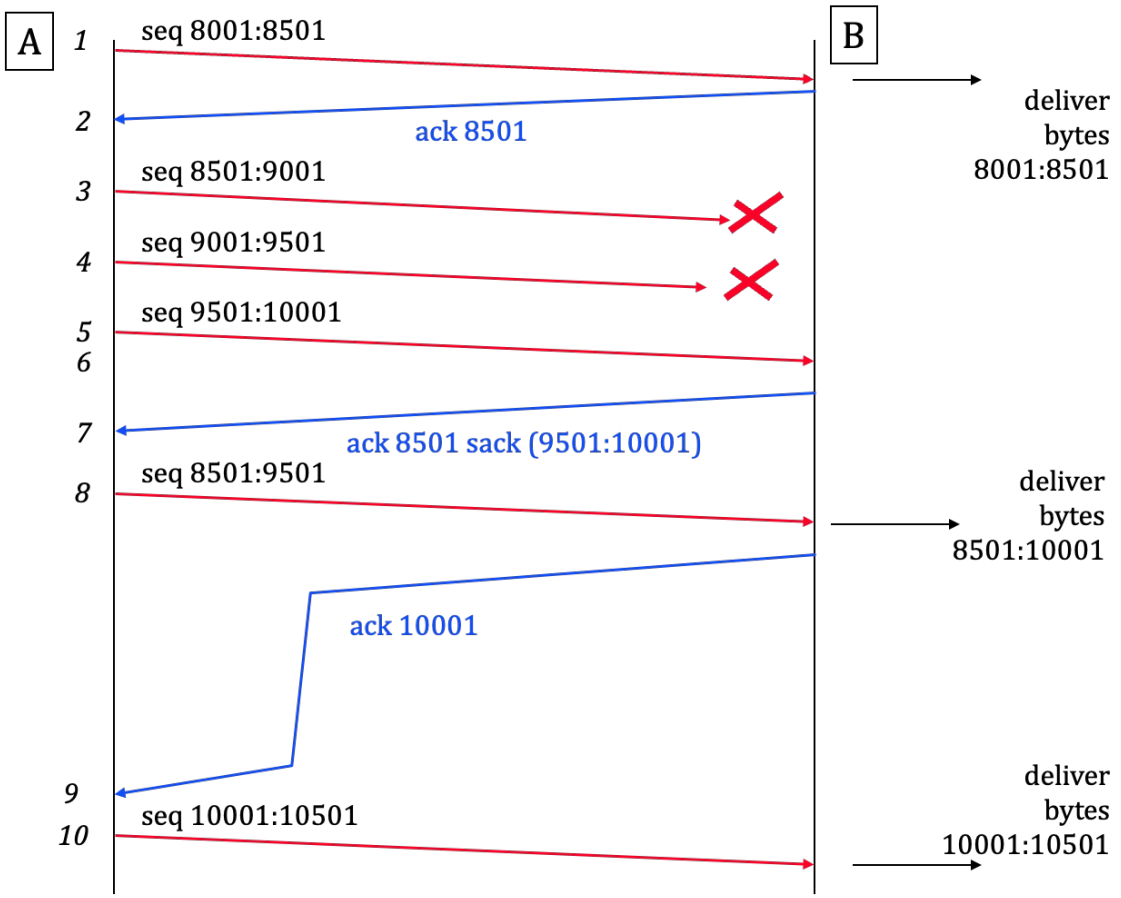


The UDP application "name resolver" expects an answer and tries again if no answer comes after some delay.

TCP does it once for all.

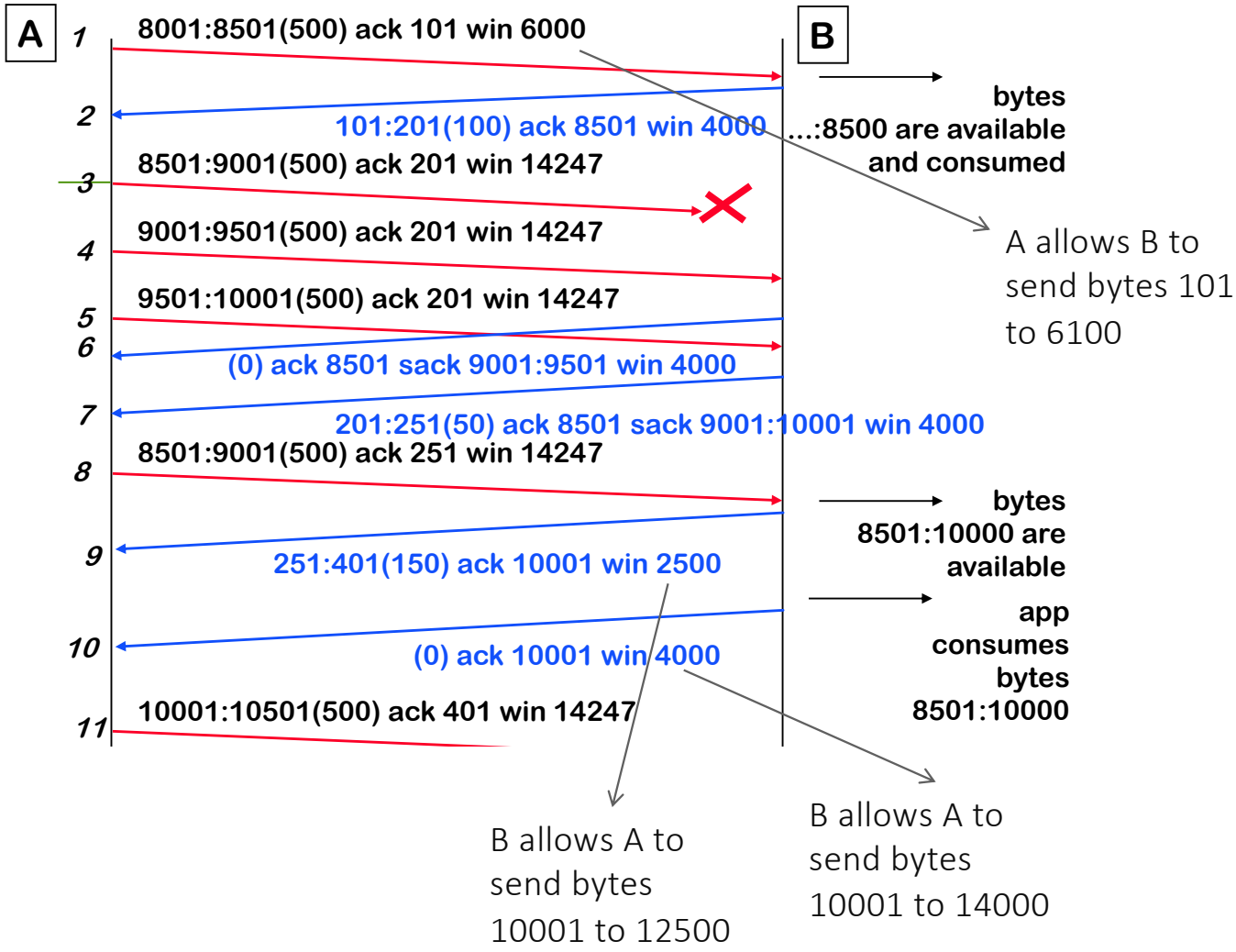
TCP Operation

- Data is numbered (per byte)
- TCP-source waits for every sent byte to be acknowledged, detects losses, and retransmits if needed.
- TCP-destination stores data in socket buffer until they can be delivered in-sequence.



TCP uses a Sliding Window

- Sliding window limits the number of bytes in flight. This avoids socket buffer overflow at receiver.
- TCP is bidirectional, A and B are both sources and destinations.



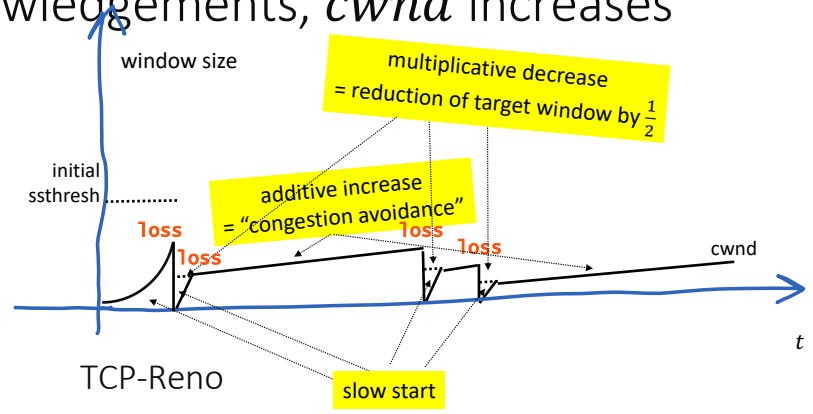
Congestion Control

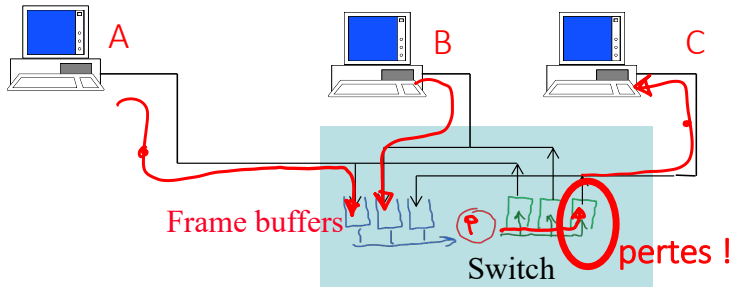
TCP is also used to control the volume of traffic in the network (congestion control). The size of the window used by TCP is $\mathbf{min}(win, cwnd)$, where win is the window size announced by TCP-destination and $cwnd$ is computed by the TCP congestion control algorithm at TCP-source.

Principles:

- when TCP detects a loss, $cwnd$ decreases
- As TCP rexceives acknowledgements, $cwnd$ increases

Different algorithms exist, most widespread are *Reno* and *Cubic*.

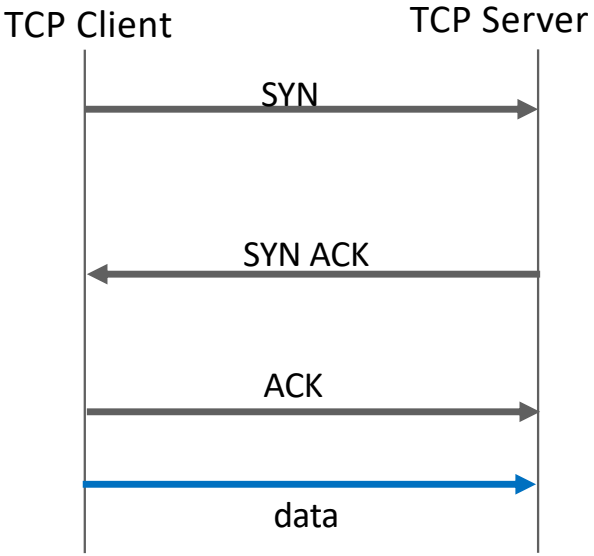


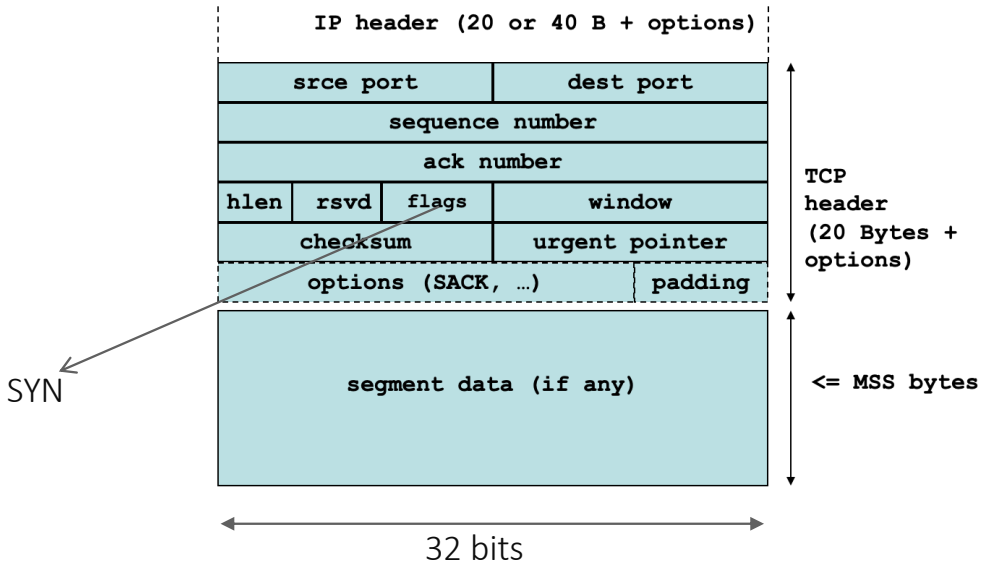


A et B send data to C atC à 100 Mb/s each. Without congestion control (e.g. with UDP), there is 50% packet loss at the switch output buffer (200 Mb/s are trying to be transmitted at 100 Mb/s). If A and B use TCP, congestion control reacts to losses and limits the rates of A and B to approximately 50 Mb/s each. There are still some losses from time to time as TCP congestion control needs a loss signal to operate.

TCP requires a connection setup phase

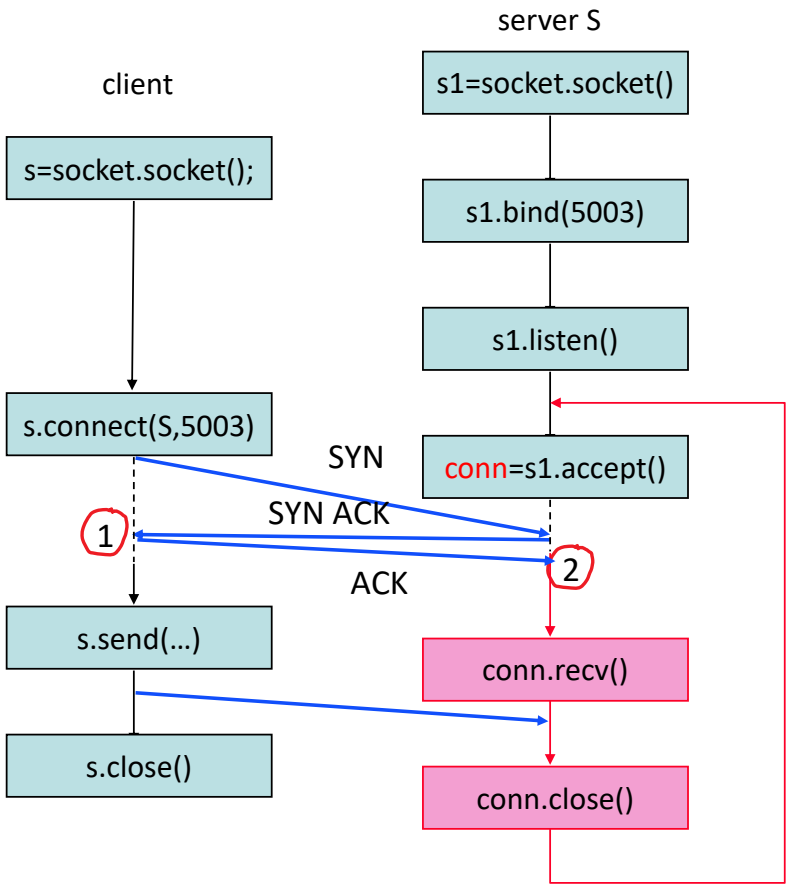
Before sending data with TCP, the counters on both ends need to be synchronized. This is achieved by sending SYN, SYN-ACK and ACK packets, which contain no data.





TCP Sockets

Data exchange starts at 1 (client) and 2 (server).



Example: : client sends a message, servers prints it on screen.

connect() associates the TCP socket with the servers's IP address and port number, sends a SYN packet, and blocks until a SYN-ACK packet is received.

send() sends a block of data (the data is actually transmitted when the sliding window allows it)

listen() prepares the buffer used for receiving SYN packets

accept() blocks until a SYN packet is received, then creates a new socket conn which will be used for exchanging data. The socket conn includes the client's IP address and port number (it is *connected*).

read() blocks until some data is available, i.e. has been received in sequence, and delivers it. If some data is received but is not in sequence, it cannot be read.

IP camera streams to Video wall controller using TCP. One packet, containing video frame number n, is lost . Frames $n+1, n+2$, etc are not lost.

A. Video frames n+1, n+2, etc are not delivered to video wall controller until frame n is successfully re-transmitted by IP camera.

B. Video frames $n+1$, $n+2$, etc are delivered to video decoder as soon as they arrive, and video decoder can mask the loss.

C. It depends whether ECN is used.

D. I don't know

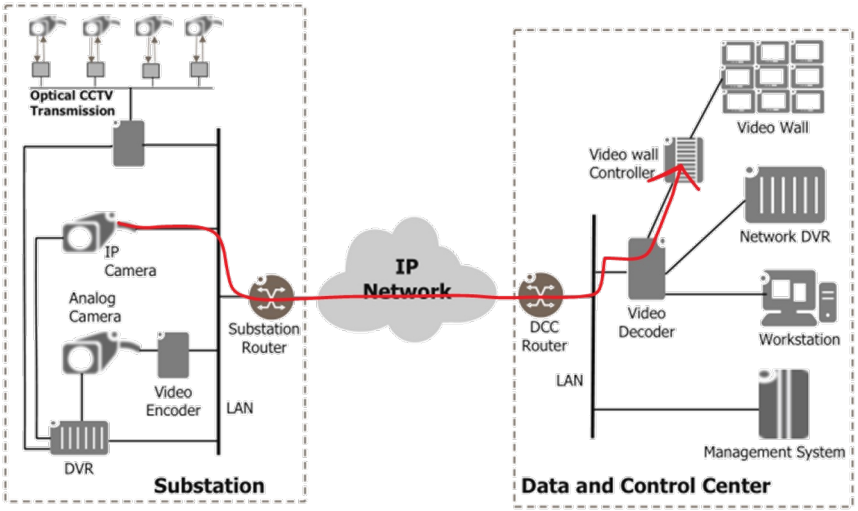
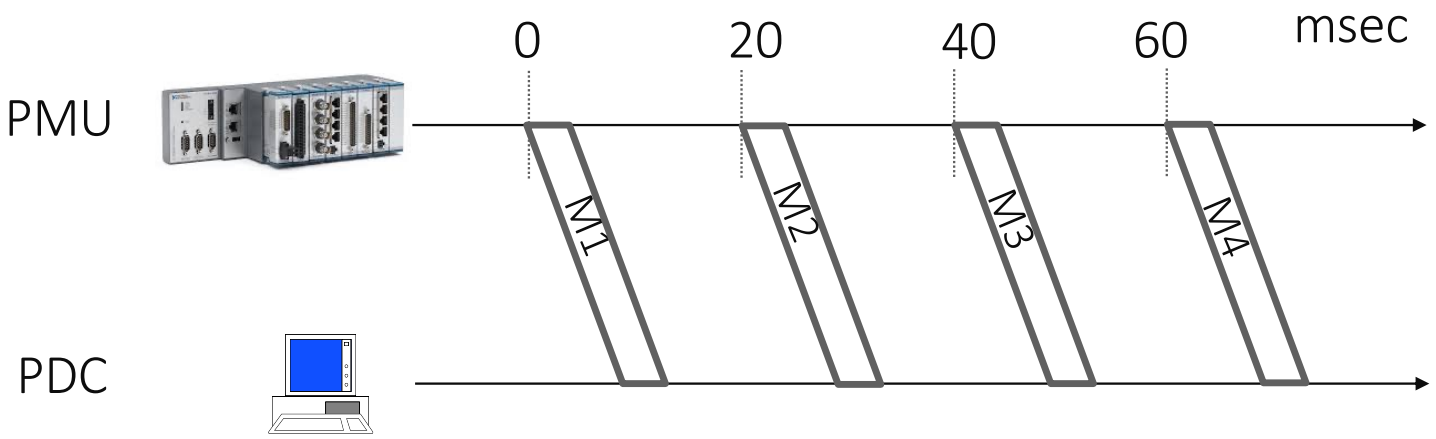


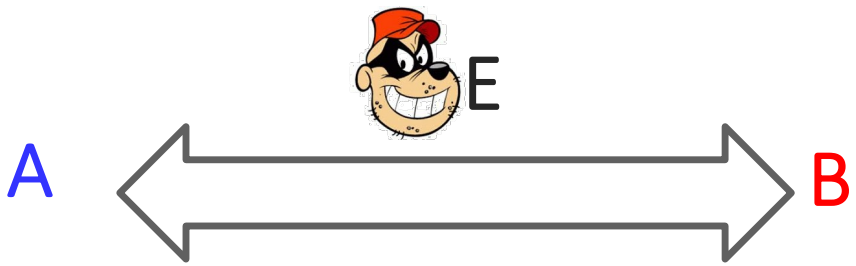
Fig. 4.7 Components of a CCTV system

Which is a better transport protocol for real-time PMU streaming?

1. TCP
2. UDP
3. Neither
4. Doesn't matter
5. I don't know



8. Secure Transport

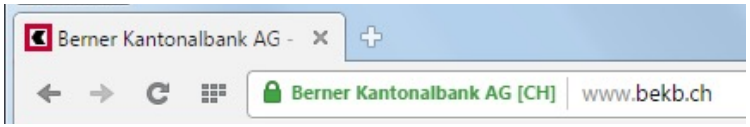


Communication must secured against attackers that may want to

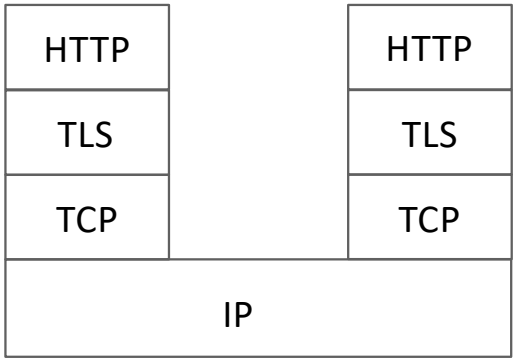
- read the message in flight,
- or even modify it.

TCP (as UDP) provides no security and should be complemented with e.g. TLS, which adds:

- confidentiality, by encrypting the data in flight, using secret keys known only by A and B;
- integrity, which ensures that the received data is exactly the one that was sent;
- and authentication, which allows to verify the identity of the remote correspondent.



http used over TLS and port 443 = https



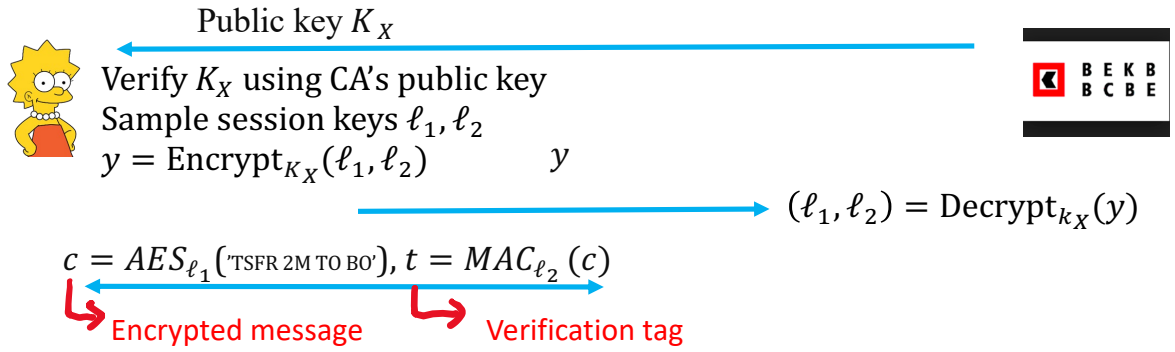
Crypto Background

Symmetric cryptography (e.g. AES) uses a secret key, known only by sender and receiver. Messages are encrypted using the secret key and can (only) be decrypted using the same secret key; this provides confidentiality. Can also be used to authenticate a message by adding a verification tag computed with the secret key, which can be verified using the same secret key. It is very fast but has the problem of key distribution.

Asymmetric cryptography (e.g. RSA) uses two different keys: one can be used for encryption, the other for decryption. One of the keys must be secret (private), the other one is public. The private key cannot be computed from the public key other than by brute-force (in principle). This eases the problem of key distribution but encryption/decryption is computationally intensive.

In all cases, keys must be long enough to resist brute-force attacks today (e.g. 256 bits for AES, 2000 bits for RSA).

TLS uses private/public key pairs + certificate

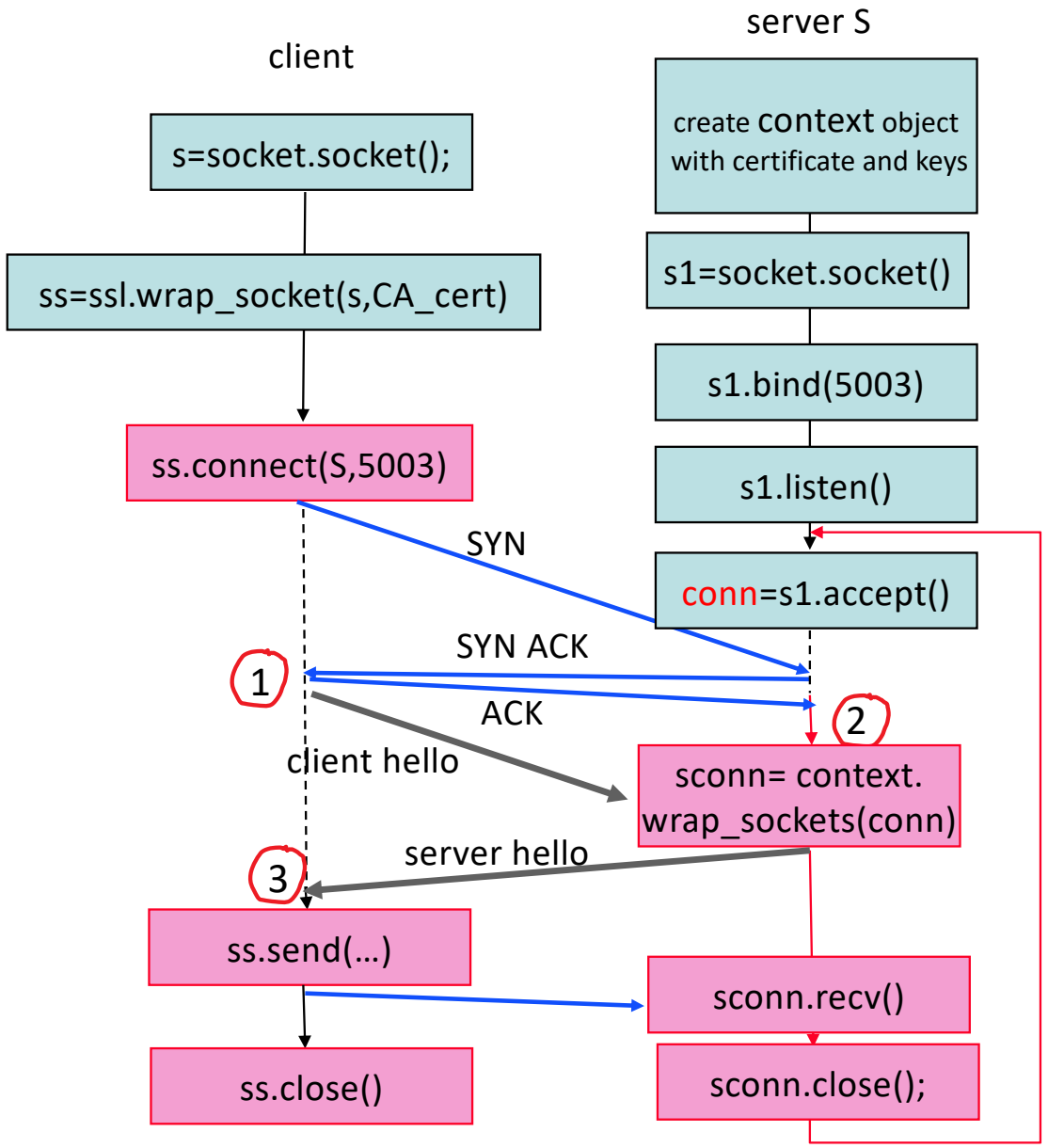


1. Lisa verifies that K_X is indeed the key of bank by using a certificate, which contains public key of bank signed by Certification Authority (CA). Lisa verifies signature of certificate by using CA's public key, pre-installed in Lisa's web browser.
2. Lisa uses public-key of bank to share secret session keys used to encrypt and authenticate the e-banking transaction.
3. Two-way communication uses secret keys

TLS Sockets

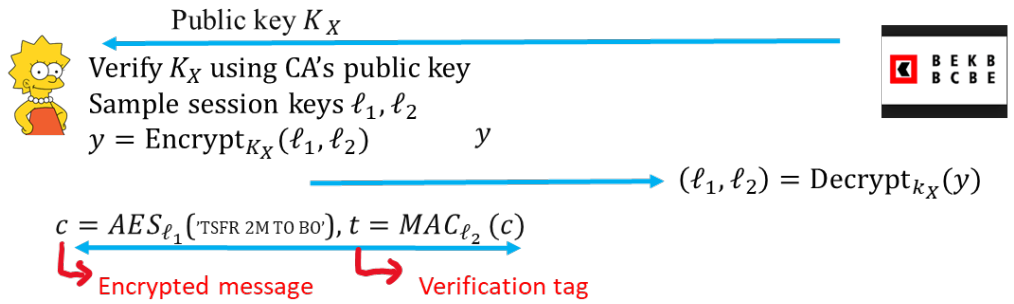
Data exchange starts in 3.

TLS also exists over UDP (D-TLS). In this case, there is no packet loss repair, and no congestion control.



Lisa's browser uses the bank's certificate to obtain...

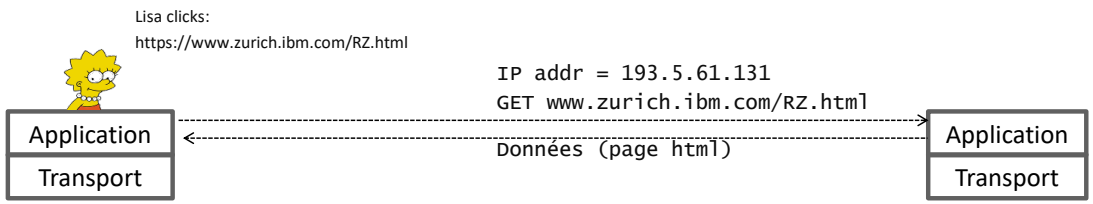
A. The bank's public key
B. The identity of BEKB (official name as of Registre du Commerce)
C. The public key of the issuing CA
D. A and B
E. A and C
F. B and C
G. All
H. None
I. I don't know



Example: WWW on TLS on TCP

WWW uses HTTP

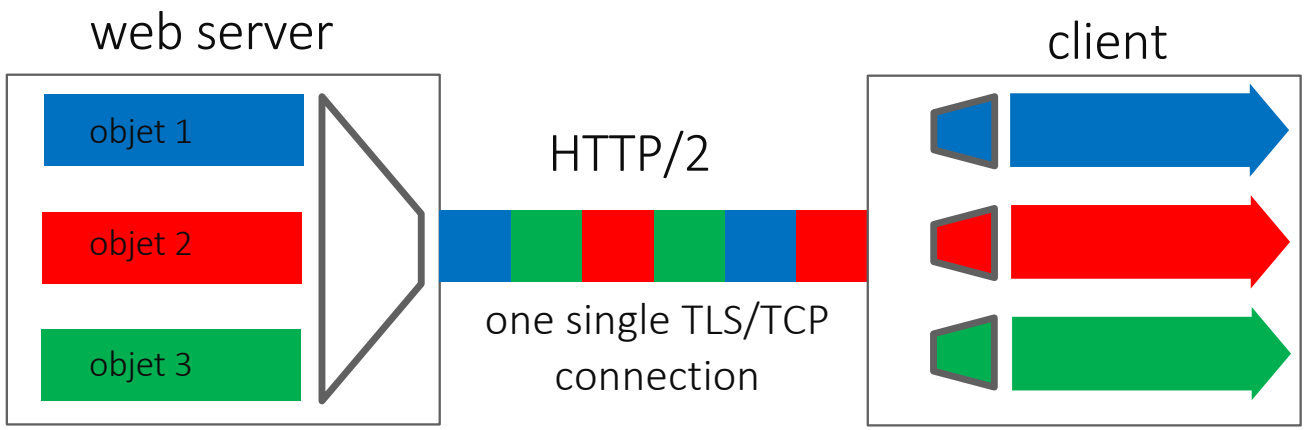
HTTP/1.1 is simple: uses one TLS/TCP connection per object



HTTP/2 uses one single TLS/TCP connection for all objects

- reduces latency and server load
- but web app is more complex as it needs to manage the multiplexing of objects

⇒ «head of the line blocking»



QUIC

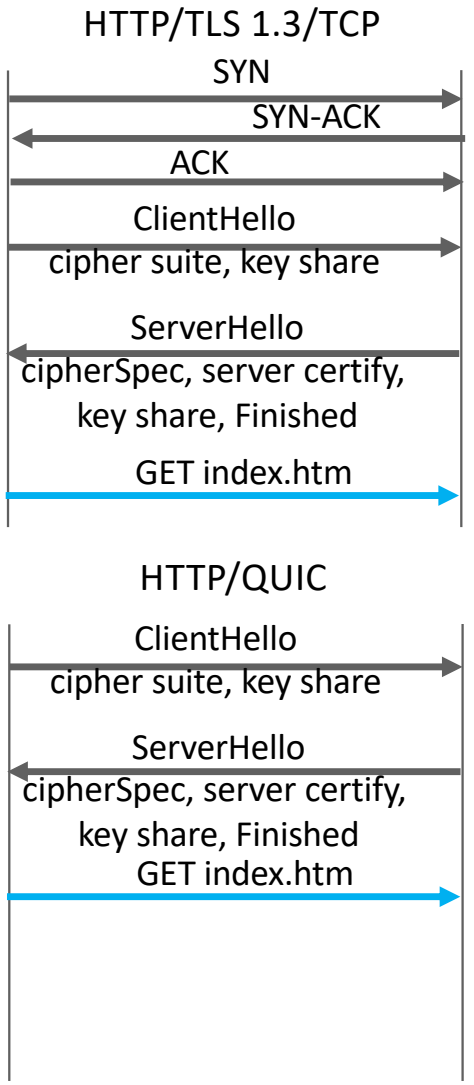
A replacement for TCP+TLS, developed by Google

Why ?

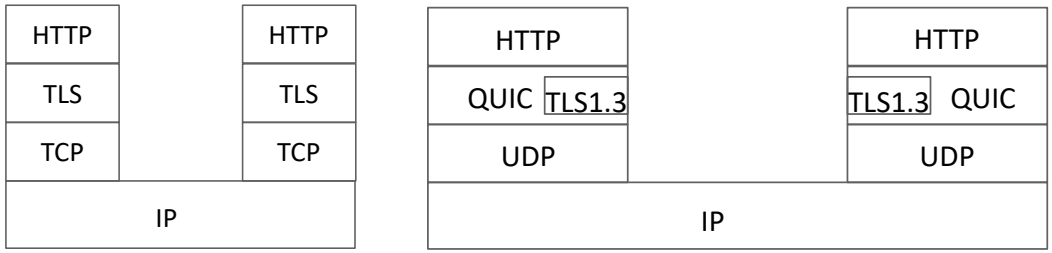
- reduce setup latency
- reduce head-of-the-line blocking

How ?

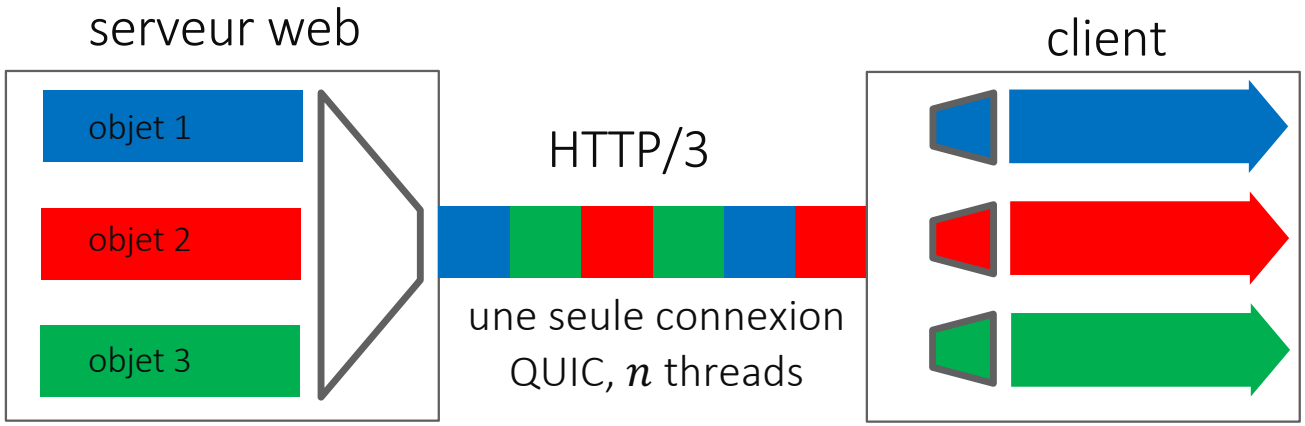
- 2 in 1: in one interaction, QUIC opens both a security association and a reliable transport association;
- For security, QUIC uses TLS;
- For reliable transport, QUIC re-uses the mechanisms of TCP, including congestion control, but supports parallel streams (in-sequence delivery is per stream);
- QUIC runs on UDP.



Example: HTTP on QUIC



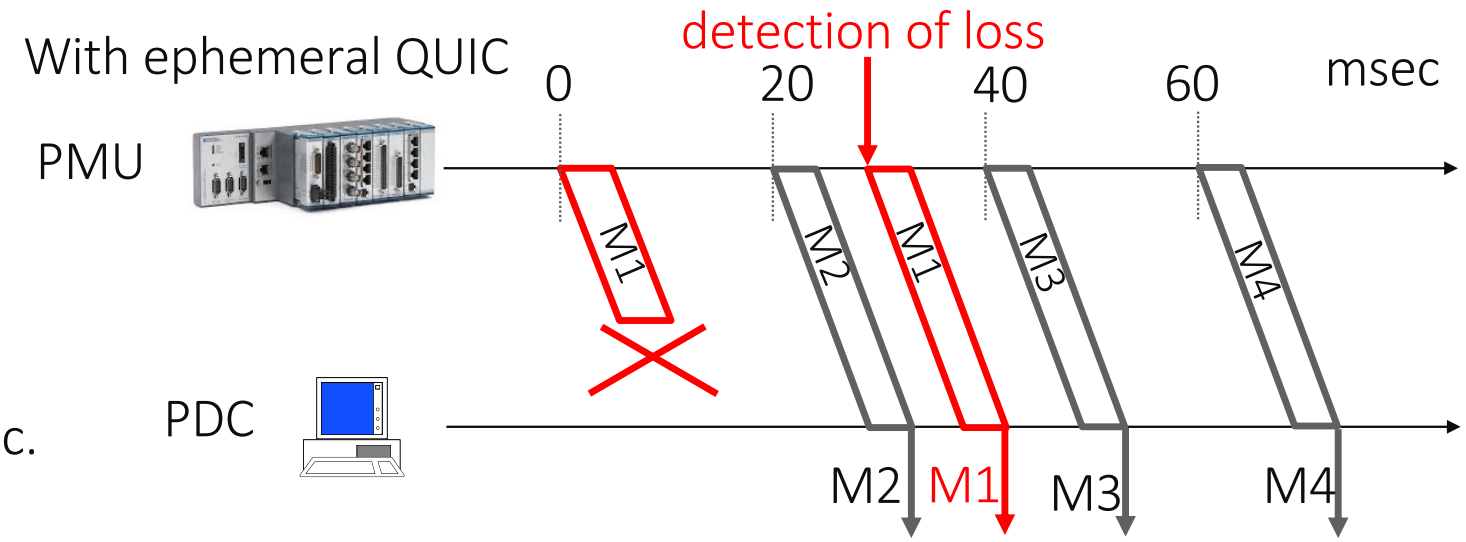
- With QUIC, security is mandatory
- HTTP/3 (with QUIC) avoids head-of-the-line-blocking



- Implemented in google apps and more (chrome, youtube app, etc)

try: access youtube then run wireshark or, from chrome:
chrome://net-export/ suivi de https://netlog-viewer.appspot.com/

Example: Ephemeral-QUIC



PMU streaming application sends one measurement in one packet every 20 msec. If one packet is lost but not recovered before 20msec, it is better to forget about it and use the newer measurement. With TCP we have head-of-the line blocking, a new measurement is delayed if previous one is lost.

Weiyu Zhang's solution with QUIC [EPFL master thesis 2018]

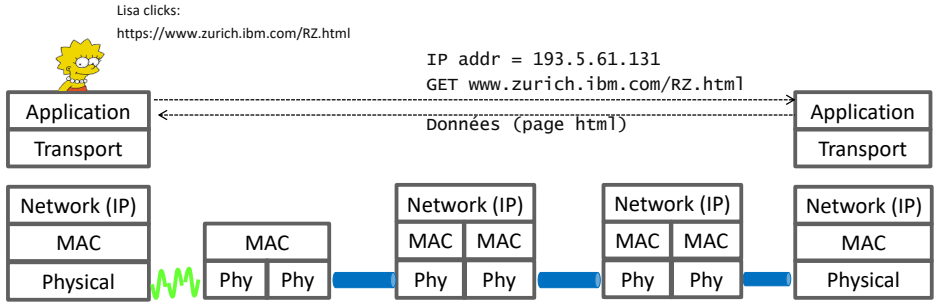
- send one measurement in one reliable stream
- one stream per measurement
- kill stream if a new measurement is available and previous stream is not acknowledged

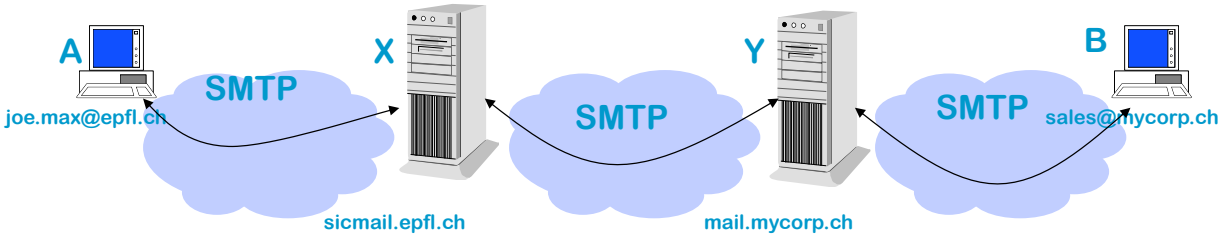
The Youtube app uses QUIC whenever possible, otherwise TLS/TCP. Lisa uses the Youtube app at home, behind a NAT. Will the Youtube app use QUIC ?

A. No because the NAT will not see TCP port numbers
B. No because QUIC uses UDP and the UDP port numbers are encrypted
C. Yes because QUIC uses UDP and the NAT can manipulate UDP port numbers
D. I don’t know

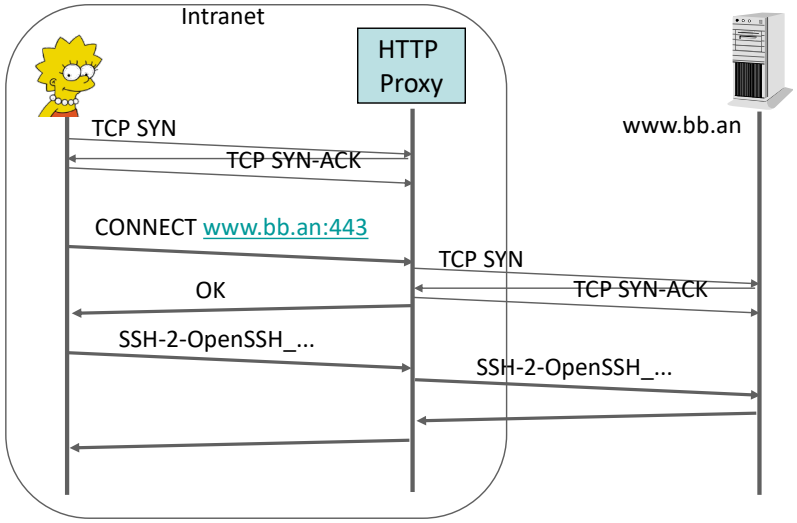
9. The Application Layer

The end-to-end principle says that applications should be in end-systems, not in intermediate systems (unlike minitel or BlackBerry)





Exceptions for mobility, security, performance, customer capture, surveillance, …



The "HTTP tunnel" forces all web clients to establish a TCP connection with the proxy. The proxy decides whether or not to forward the connection.

The Meter Concentrator should be…

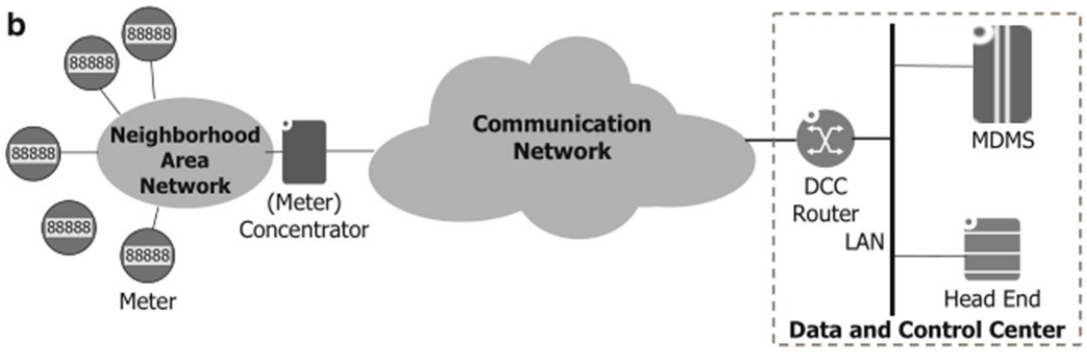
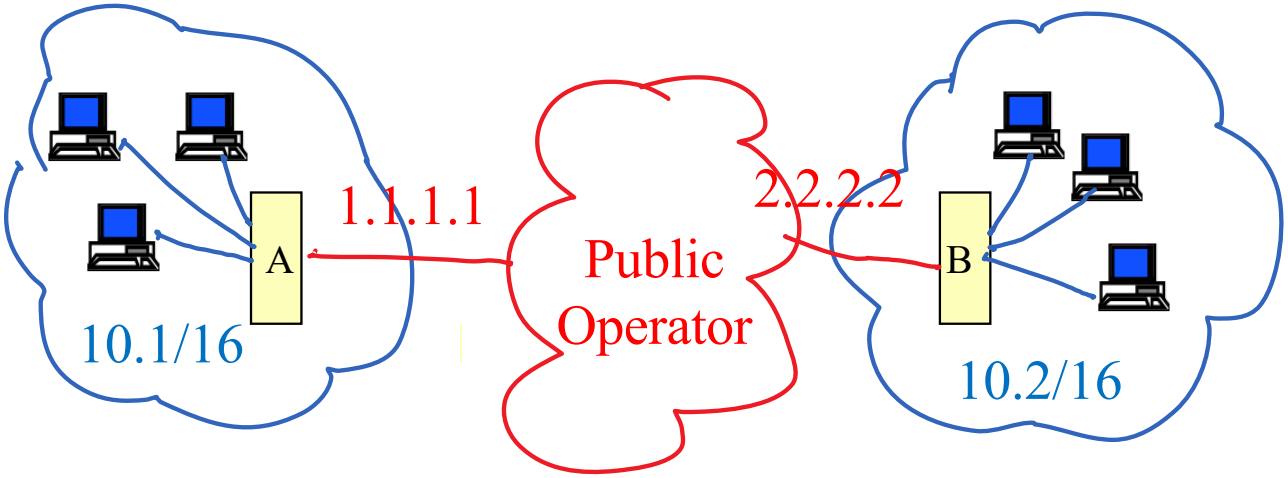


Fig. 5.1 Networking for AMI – two options. (**a**) Meters connects directly to utility MDMS. (**b**) Vendor-proprietary "AMI solution" connecting to the MDMS

A. A switch
B. A router
C. An application layer gateway
D. Any of the above
E. None of the above
F. I don't know

10. VPNs (Virtual Private Networks)

Homer deploys 10.x addresses in two sites and would like to interconnect them as one (closed) private network.

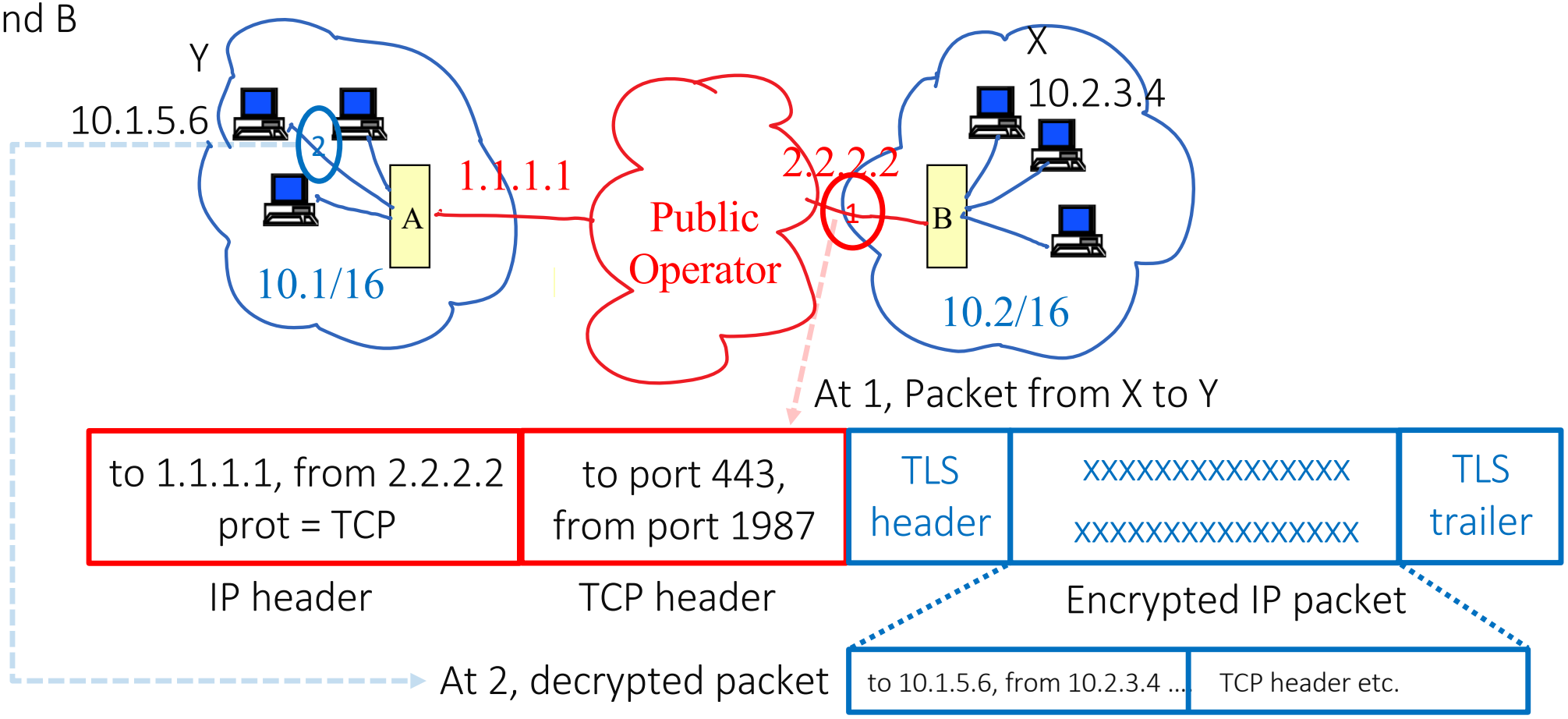


- Private addresses cannot be forwarded
- Traffic should be protected while transiting the public network

Common solution: VPN

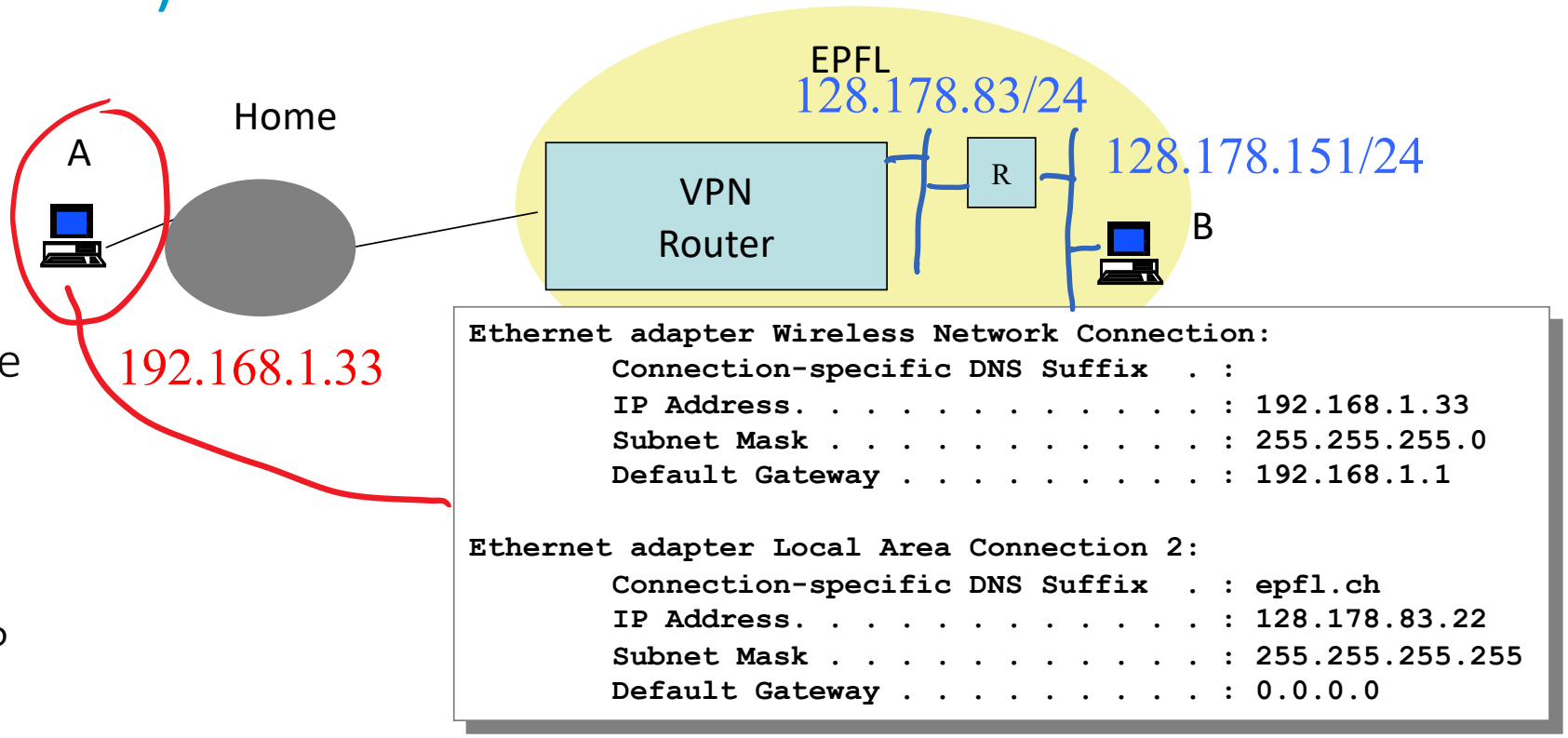
VPNs use tunnels + encryption + authentication

TLS- VPN: Packets from X to Y are tunneled = placed inside a TLS connection, between A and B



VPNs (continued)

VPN can be used to build a private network + to give remote access to a private network.



VPN software provides to A a virtual interface, allowing A to be in EPFL network

VPN variants

- IP in TLS in TCP
- IP in QUIC
- IP in D-TLS in UDP
- IP in IPSEC in UDP

The last two are often preferred because of undesirable side-effects: the TLS-TCP and QUIC tunnels are congestion controlled.

Conclusion

The internet protocols are based on a layered model and a few simple principles

The same technologies are used in industrial networks (electrical grid, factory automation, aviation, car, …)

This fascinating world can be explored with a few simple but powerful tools, available in quasi-all systems.