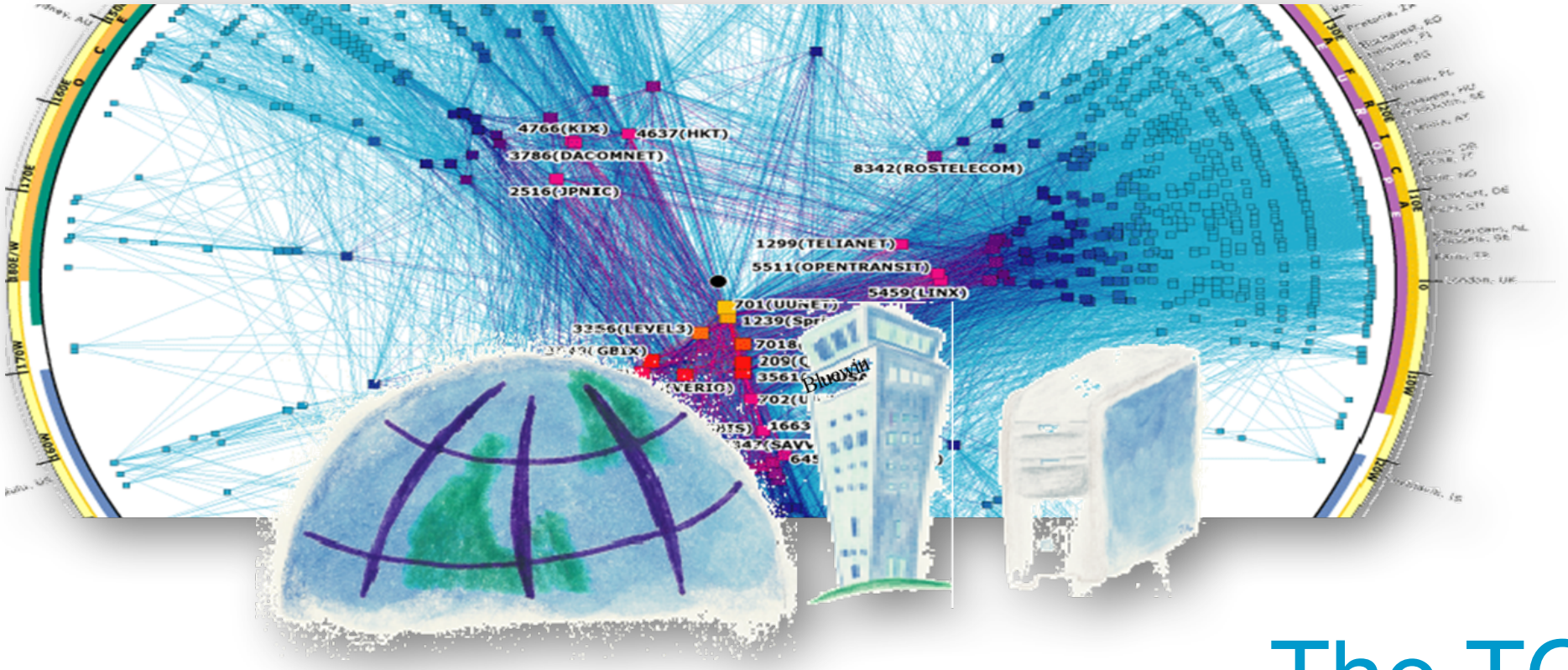


The TCP/IP Architecture

COM-407

2024

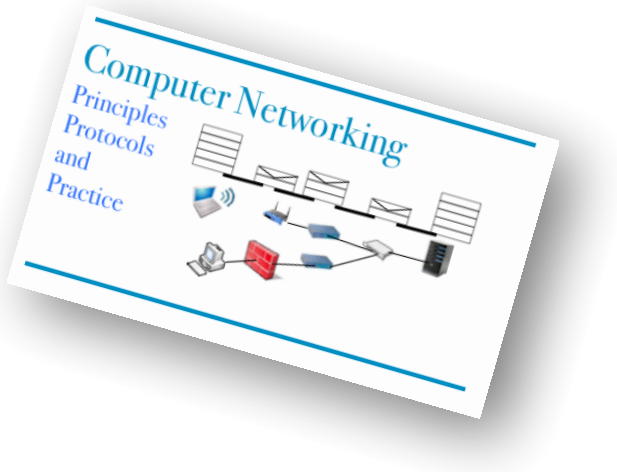
Objective of this lecture

Understand Layered Model of Computer Networks

Know what MAC addresses, IP addresses and DNS names are

Textbook

Chapter 2: Introduction of edition 1



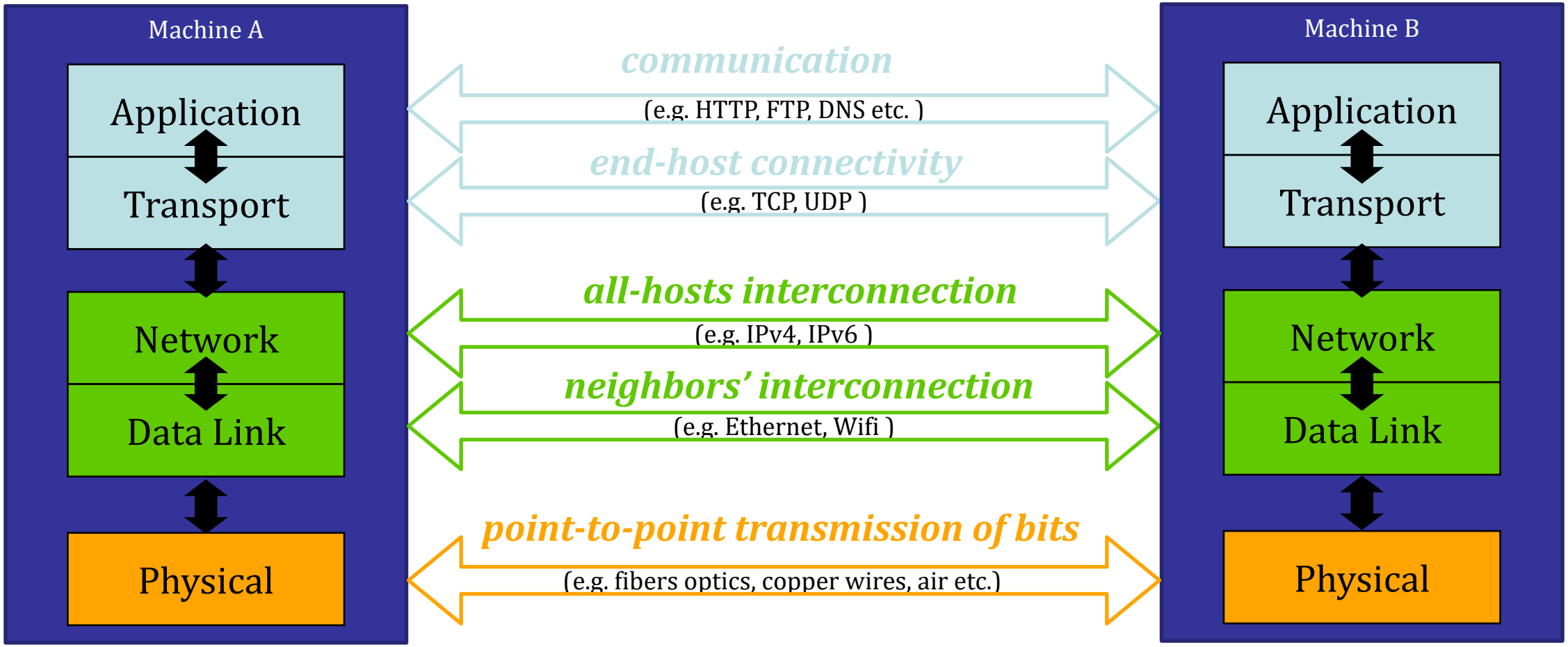
Chapter 1: Computer Networks and the Internet

Computer Networking: A Top-Down Approach
8th edition
Jim Kurose, Keith Ross
Authors' website

Computer Networks: A layered architecture of protocols

(= sets of rules and exchanged messages)

What is it?

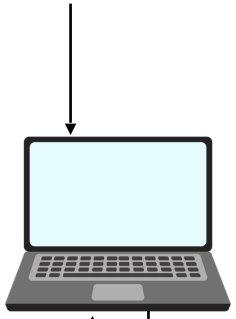


Why it exists? Enables flexibility and modularity

Application Layer helps machines communicate

user types into the browser:
http://www.google.com

- Well-defined protocols e.g.: DNS, HTTP, email, etc., all network *applications*, and *e2e security* reside here
- In lab 3, we will build our "Application Layer"



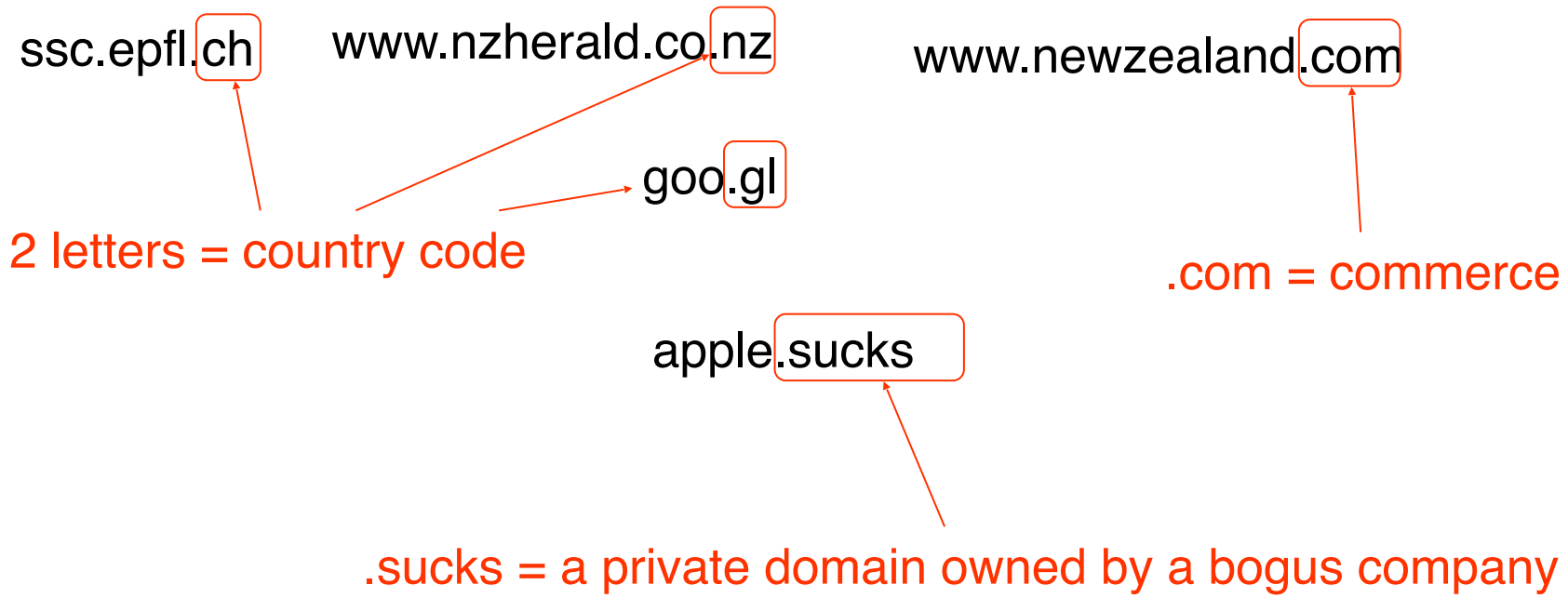
IP address = 216.58.215.228
GET www.google.com

data (HTML page)

Web server
www.google.com

E.g.: DNS, a protocol for resolving names to addresses

- *DNS servers* map names to addresses
  - e.g.: ssc.epfl.ch <—> 128.178.222.83, smtp.sunrise.ch <—> 212.35.39.69
- Names are human readable *synonyms* for IPv4 or IPv6 addresses and have some *structure*:



Transport Layer helps the application layer

Provides:

- *programming interface* to application layer, named *socket* (= specific software structure)
- (along with all layers below) data delivery to the applications of two communicating hosts

Exists mainly in two versions:

TCP (Transmission Control Protocol)

- *Reliable*: if some data is lost somewhere, TCP retransmits it
- *In-order delivery* (stream service): data is delivered at destination in the order sent by source
- Unit of information is a *byte*—data is written to the socket, TCP sends blocks of data at will

UDP (User Datagram Protocol)

- *Unreliable*: message may be lost
- *No in-order delivery* guaranteed: data may arrive at destination out of order
- Unit of information is a *message*—blocks of data are sent as written into the socket

We will also study QUIC, which is a kind of “super-TCP” and is over UDP

Transport Layer uses *port numbers*

- A port number is assigned to a process to ***uniquely*** identify it within a host machine
- The UDP/TCP packet carries ***both*** the source and destination port numbers (at its header)
- Using ports, multiple network programs may run on a host ***at the same time***
- Some port numbers are reserved (e.g. 80 is for HTTP Web servers, 53 is for DNS servers)

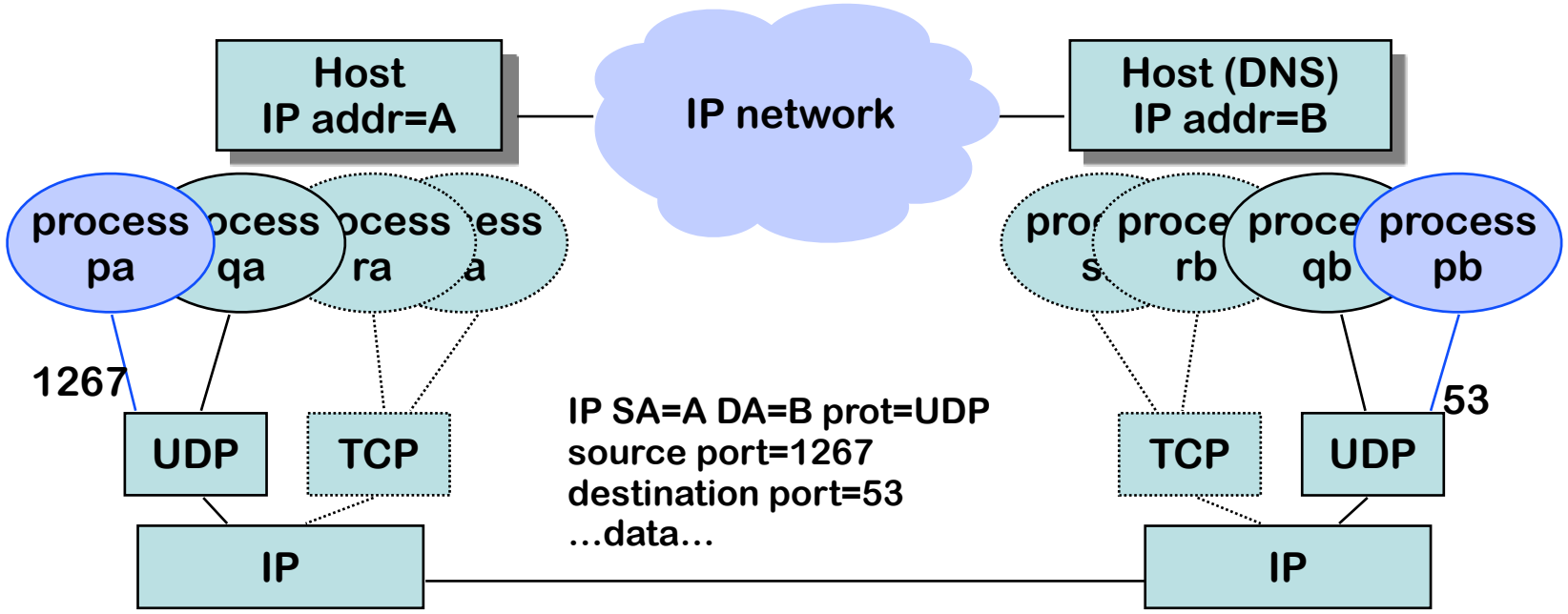
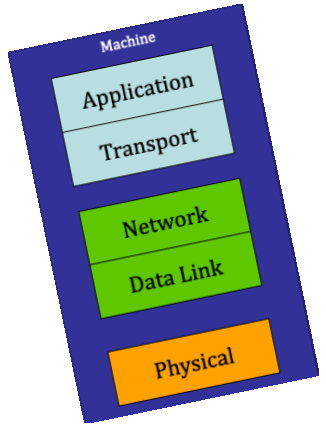


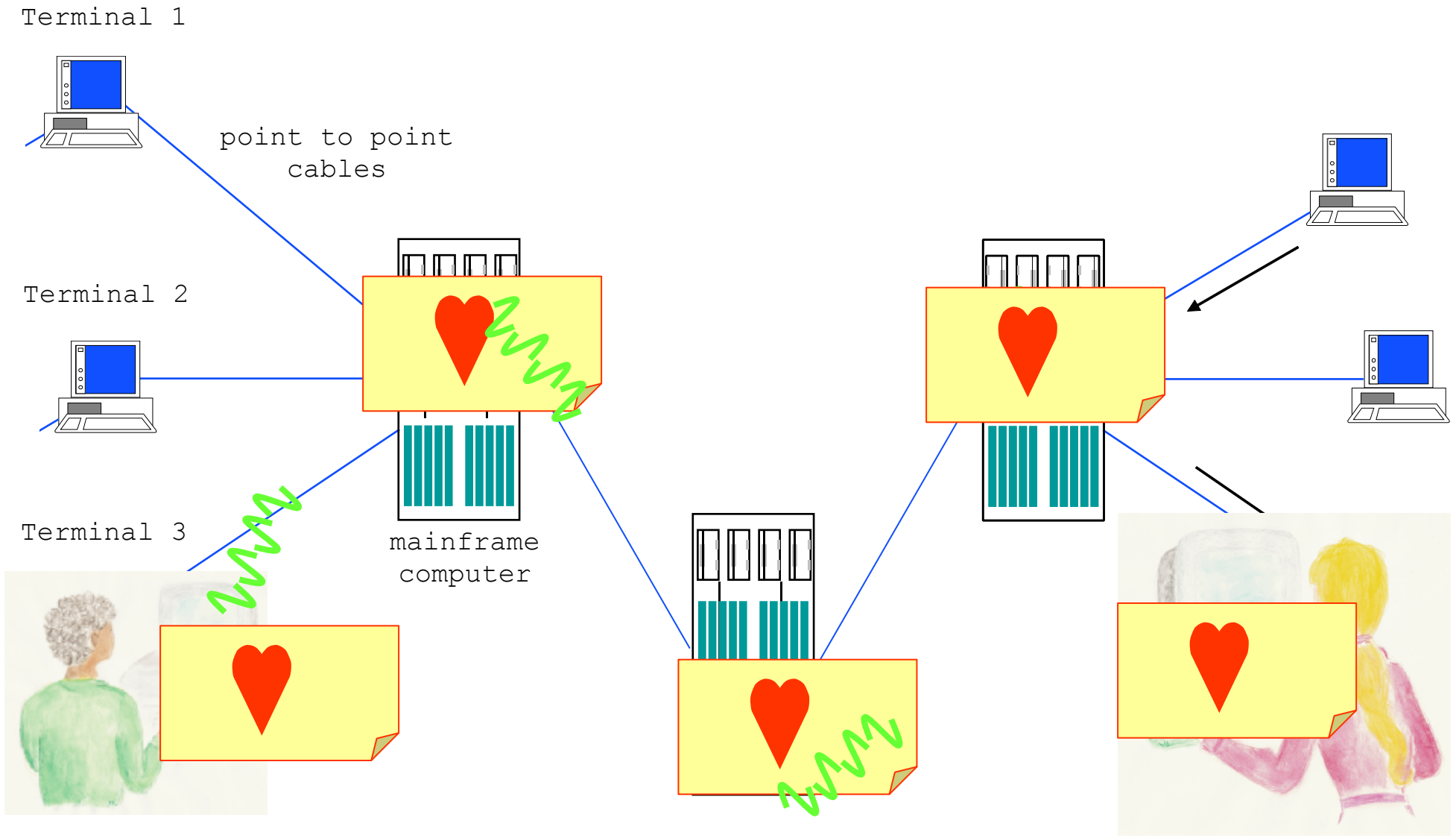
Fig.: Process “pa” (port=1267) on host A sends a request to process “pb” (port=53) on host B.

Network Layer provides interconnection among *all* network devices



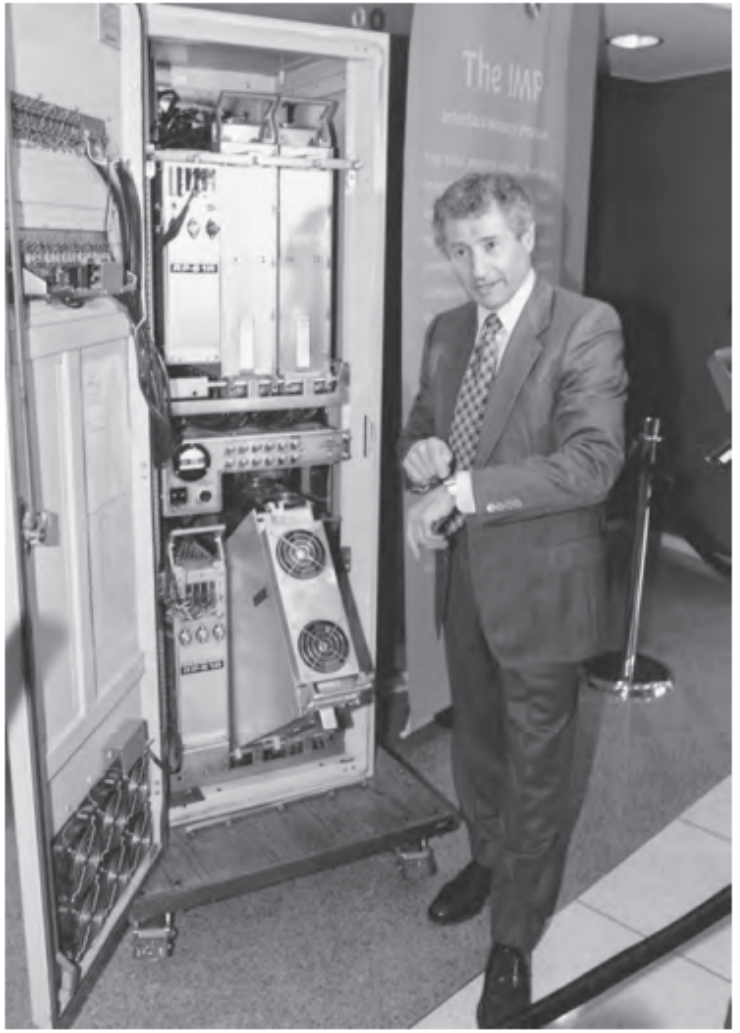
Drawing direct cable connections does not scale.
Let's do something similar to the Postal Services!

First Networks (Bitnet, SNA) relayed *entire* messages



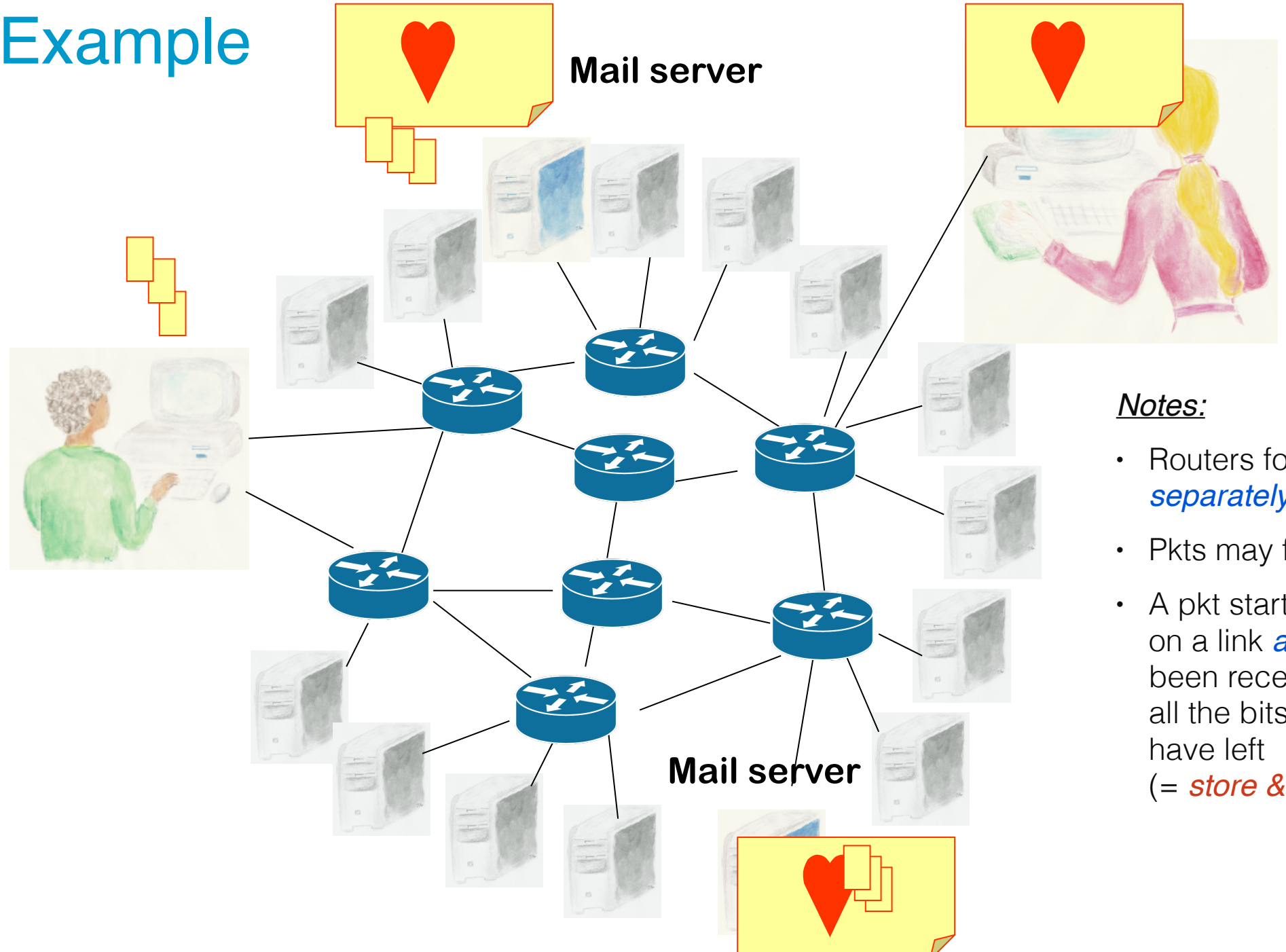
The Internet Uses *Packet Switching*

- data is broken into chunks called *packets* of size ≤ 1500 bytes
- packets are *forwarded* from one intermediate node (a.k.a. *router*) to another, until they reach destination
- similar to Postal Service: a packet ≈ a postcard, contains information that helps its delivery to the destination



An early packet router

Example



Notes:

- Routers forward packets *separately* to other routers
- Pkts may follow *different paths*
- A pkt starts being forwarded on a link *after* its last bit has been received and all the bits of the previous pkt have left (= *store & forward* principle)

Network layer: 2 basic functions

Forwarding

- When a packet arrives at a router's input link, the router moves the packet to the appropriate output link
- Forwarding is based on information carried in the packet's *header*

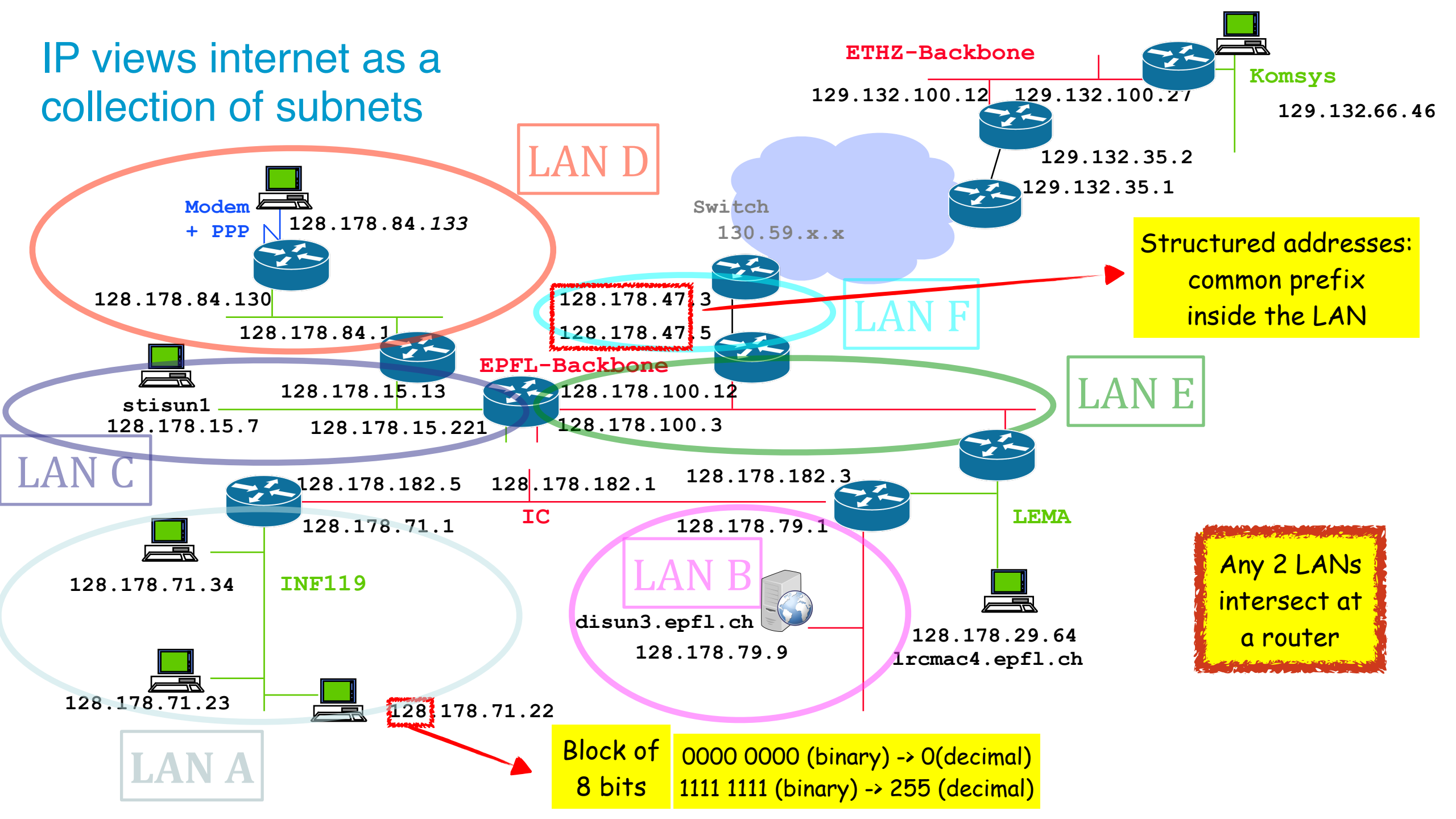
Routing

- The network layer determines the *route/path* taken by packets, as they flow from a source to a destination
- The algorithms that calculate these paths are called *routing algorithms*

Most important protocol: IP (Internet Protocol)

- interconnects multiple *local area networks* (LANs)
- uses packet switching
- delivers packets from a source to a destination via a series of routers
- forwards packets from router to router based on *IP addresses*
- offers *no reliable*-delivery guarantees (*best-effort* approach)
    - packets are briefly stored in routers' buffers
    - packets of the same source-destination flow may follow different routes/paths
    - so, packets may be *dropped*, *delayed* or *reordered*

IP views internet as a collection of subnets



Two IP versions: IPv4 and IPv6

IPv4 addresses have:

- a length of 32 bits
- dotted *decimal* notation – one number in {0, 1,…, 254, 255} = 8 bits
  - an EPFL (public) address: 128.178.156.23

IPv6 addresses have:

- a length of 128 bits
- *hexadecimal* notation – one hexadecimal digit in {0,1,…, e,f} = 4 bits
  - an EPFL (public) address: 2001:0620:0618:01a6:0a00:20ff:fe78:30f9

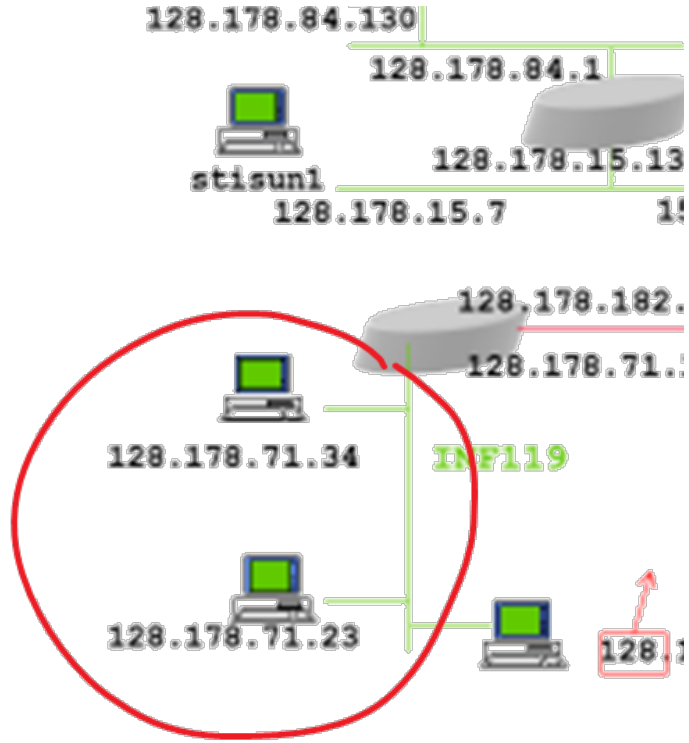
IPv4 and IPv6 are *incompatible*

- IPv6 is another protocol (not IPv4 with a larger address space)
- devices that run IPv6 cannot parse IPv4 packets and vice versa

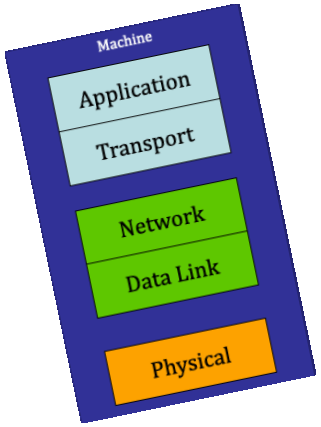
Apart from addresses, IP also uses *subnet* masks

- *subnet* <— a LAN, i.e. a set of devices:
  - connected at the Data-link layer
  - sharing the same IP-address *prefix*, e.g.: 128.178.71.X

- The prefix is specified using a *subnet mask*
  (= sequence of bits, where 1s indicate fixed positions of the prefix)
  - e.g. for an EPFL IPv4 LAN, the subnet mask is
    1111 1111 1111 1111 1111 1111 0000 0000
  - The size (in bits) of the prefix is not always the same, e.g.:
    ETHZ IPv4 LANs = 26 bits
    EPFL IPv6 LANs = 64 bits



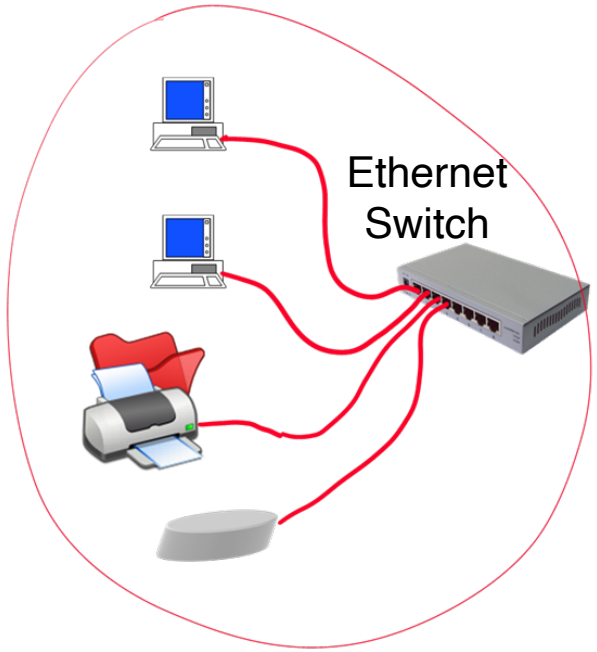
- Different notations for the subnet mask:
  - dotted, decimal: e.g., address =128.178.71.34, mask = 255.255.255.0
  - *"/"* (slash): e.g. 128.178.71.34/24
    or 2001:620:618:1a6:0a00:20ff:fe78:30f9/64



Data link layer delivers data from hop to hop

a.k.a. MAC (= Medium Access Control) layer

WiFi LAN



Ethernet LAN

Responsible for:

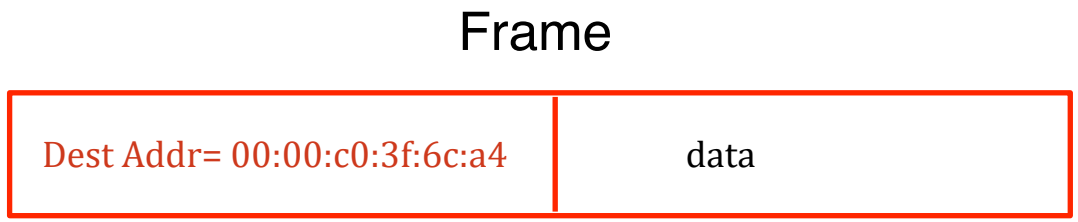
- moving data from one node to an *neighbor* node over a shared link (wired or wireless)
- controlling the physical layer and providing services to the network layer that depend on the actual physical link

Possible functions/services:

- error detection and correction
- reliable delivery
- collision avoidance
- flow control
- link access and multiplexing of the shared medium/link

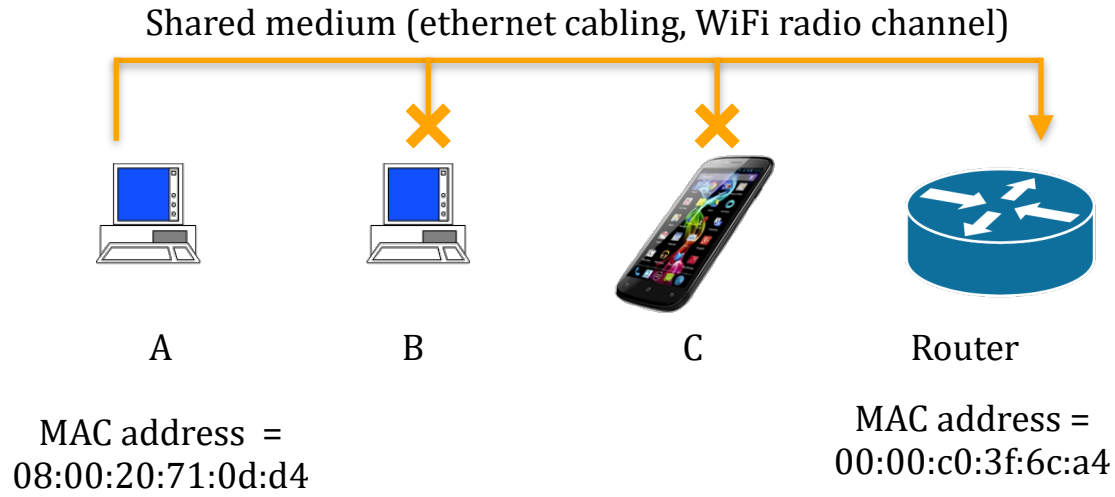
Data link layer uses MAC addresses

- MAC address (≈ device's serial no):
  - 48 bits
  - set by manufacturer
  - unique, in principle
  - no special structure



How?

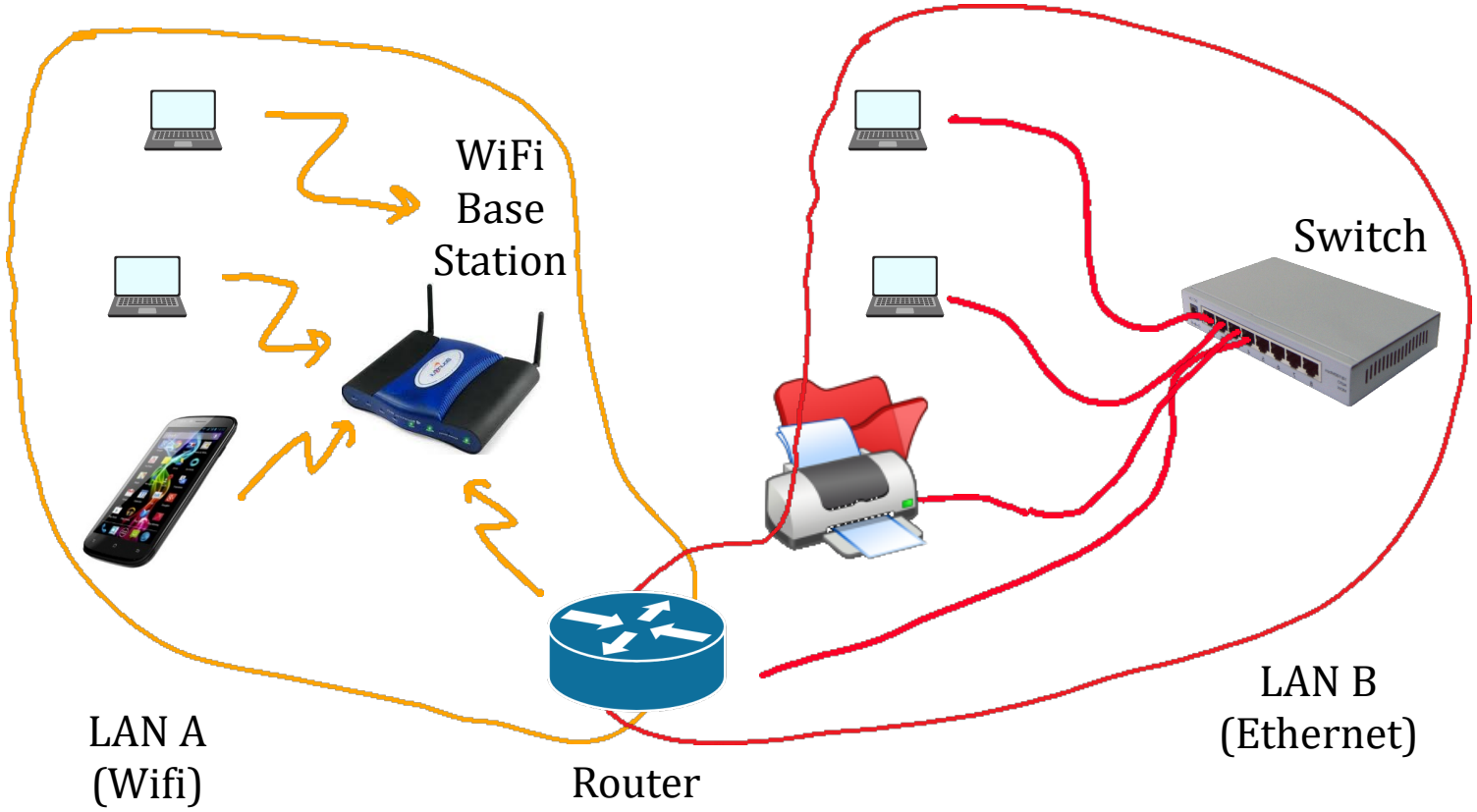
- Sender puts destination MAC address in header of the packet (called *frame*)
- All hosts connected via the same/shared medium read all frames; keep only if destination MAC address matches their own
- Switches can forward frames, if the LAN links are too long or too many (and not only for that)
- Destination MAC Address is sent in cleartext, no encryption (but data can be encrypted)



Routers vs Switches

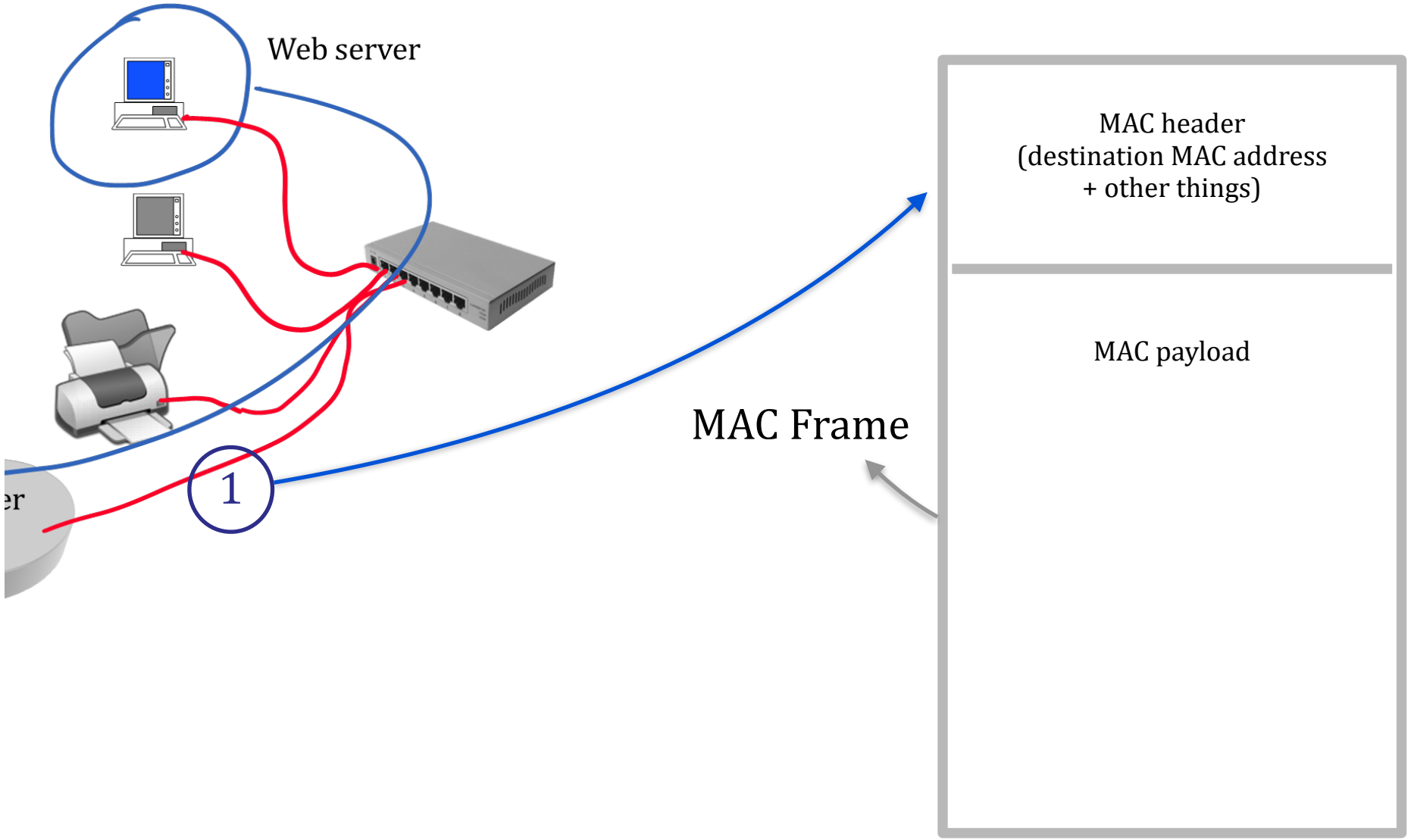
Router =
a *network-layer* system (or program) that forwards *packets* based on *IP addresses*

Switch =
a *data-link-layer* system (or program) that forwards *frames* based on *MAC addresses*

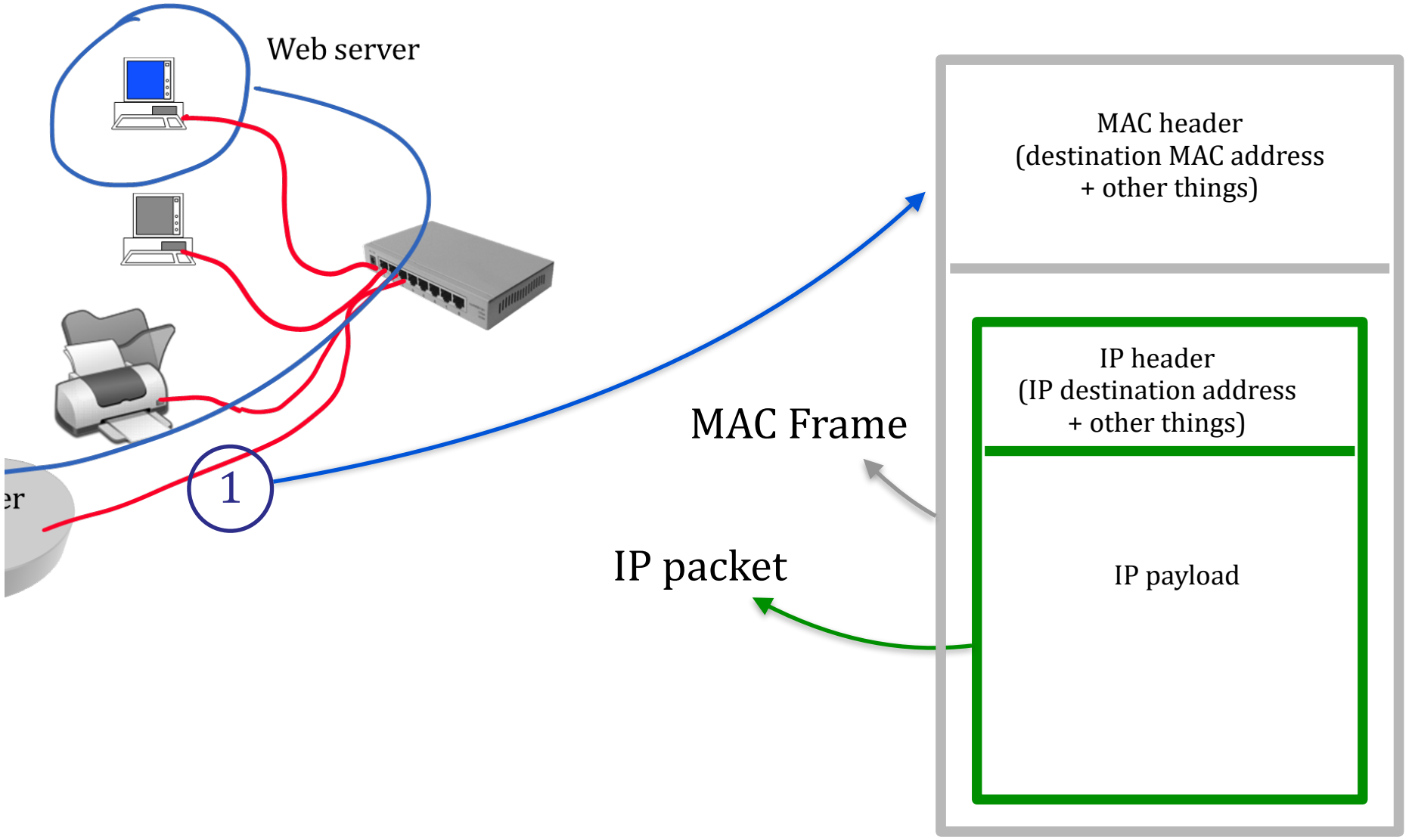


- Routers interconnect *different LANs*
- Switches interconnect devices of the *same* LAN
  - if devices from different LANs want to communicate, they go through a router (a.k.a. *gateway* router)

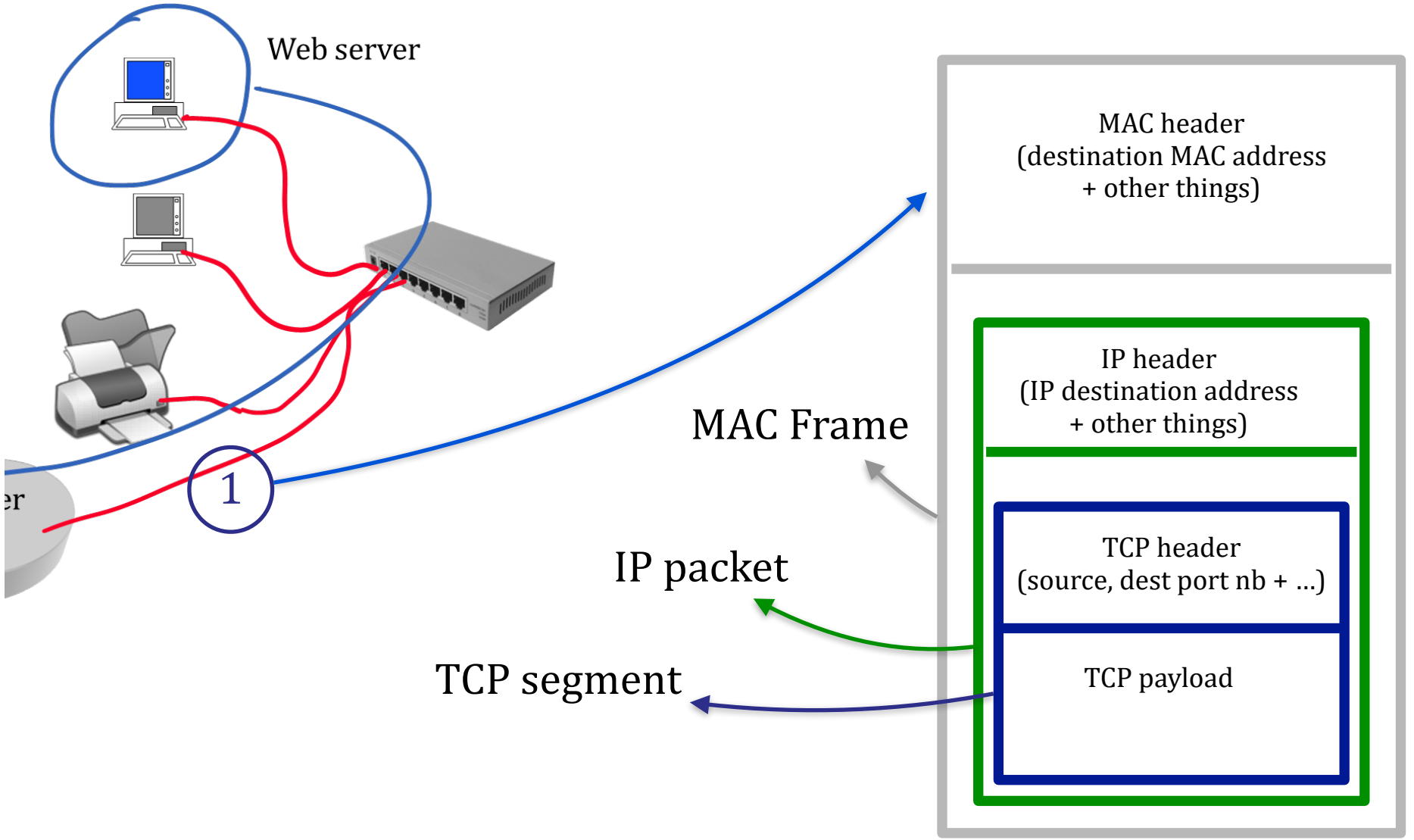
Encapsulation: an onion view with headers and payloads



Encapsulation: an onion view with headers and payloads



Encapsulation: an onion view with headers and payloads

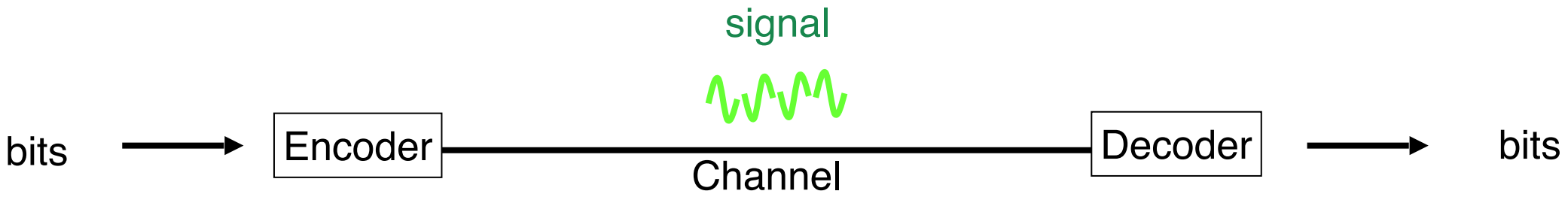


A frame captured and prettily displayed

```
ETHER:  ----- Ether Header -----
ETHER:
ETHER:  Packet 4 arrived at 19:03:32.40
ETHER:  Packet size = 60 bytes
ETHER:  Destination = 0:0:c:2:78:36, Cisco
ETHER:  Source      = 0:0:c0:b8:c2:8d, Western Digital
ETHER:  Ethertype = 0800 (IP)
ETHER:
IP:   ----- IP Header -----
IP:
IP:   Version = 4
IP:   Header length = 20 bytes
IP:   Type of service = 0x00
IP:         xxx. .... = 0 (precedence)
IP:         ...0 .... = normal delay
IP:         .... 0... = normal throughput
IP:         .... .0.. = normal reliability
IP:   Total length = 44 bytes
IP:   Identification = 2948
IP:   Flags = 0x0
IP:         .0.. .... = may fragment
IP:         ..0. .... = last fragment
IP:   Fragment offset = 0 bytes
IP:   Time to live = 64 seconds/hops
IP:   Protocol = 6 (TCP)
IP:   Header checksum = cec2
IP:   Source address = 128.178.156.7, lrcpc3.epfl.ch
IP:   Destination address = 129.132.2[illegible]72, ezinfo.ethz.ch
IP:   No options
IP:
TCP:  ----- TCP Header -----
TCP:
TCP:  Source port = 1268
TCP:  Destination port = 23 (TELNET)
TCP:  Sequence number = 2591304273
TCP:  Acknowledgement number = 0
TCP:  Data offset = 24 bytes
TCP:  Flags = 0x02
```

Hostnames are never inside the IP header; they are shown here, only because of the software that created this user-friendly packet view.

Physical Layer transforms Bits and Bytes into signals



- Signals are electromagnetic or acoustic (under water)
- Technology-specific layer:
  - various Ethernet physical layers, various WLAN 802.11 physical layers
  - with various *bit rates*, measured in b/s, 1 kb/s = 1000 b/s, 1 Mb/s = 10^6 b/s, 1Gb/s=10^9 b/s

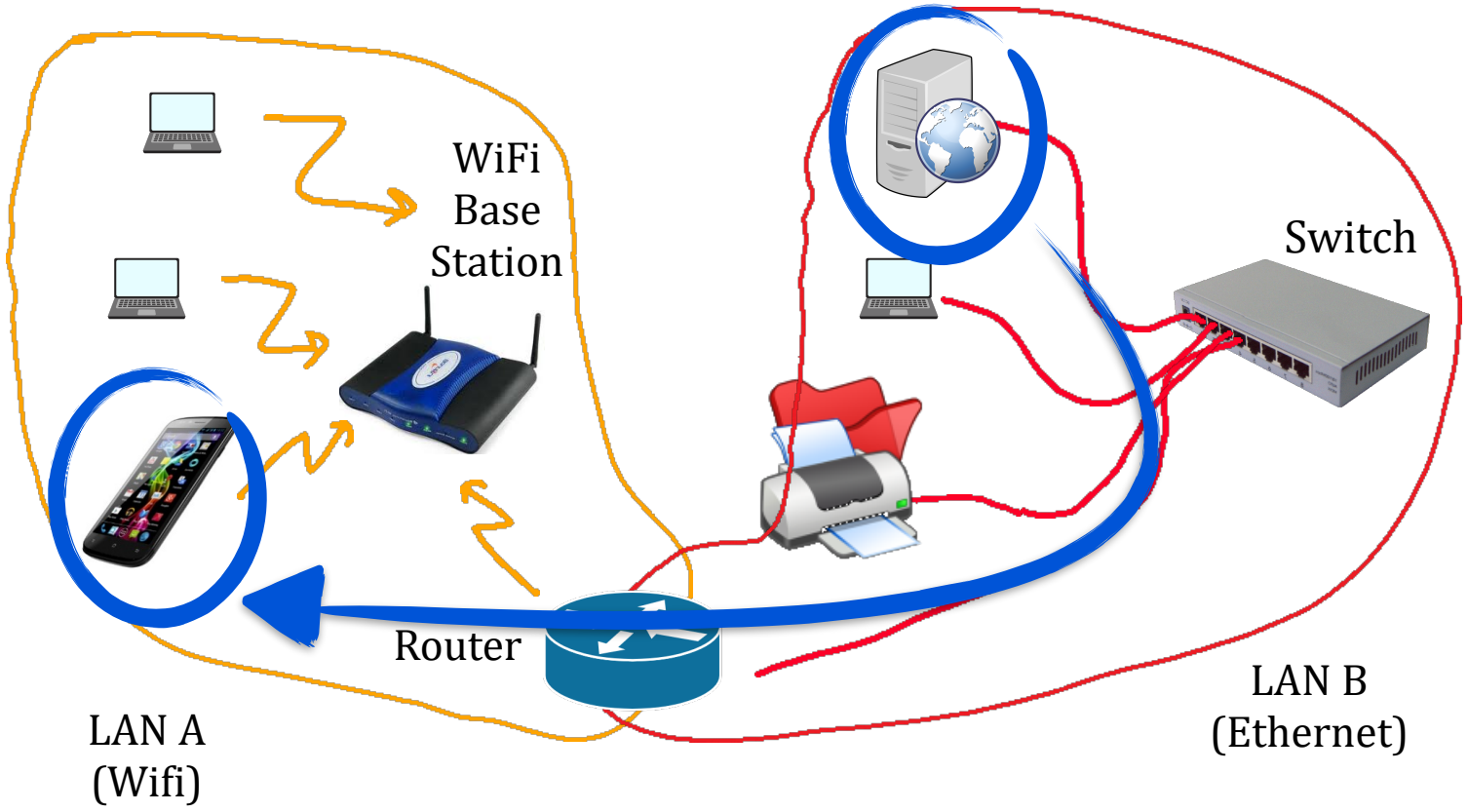
Bit Rate vs Bandwidth—for us they will be the same, but…

Bit rate = number of bits that can be transmitted per sec inside a channel

Bandwidth = the width of the frequency range that can be used for transmission over a channel

- The bandwidth limits the *capacity* (= max achievable bit rate) of a channel
- Information Theory computes the capacities of various channels:
  - e.g. [Shannon-Hartley law], for a channel with bandwidth B (Hz) submitted to Gaussian noise, the capacity in b/s is: $C = B \log_2(1 + SNR)$,
    where: SNR= signal to noise ratio (ratio of power of emitted signal over power of noise);
  - in practice, for a ADSL Line: $B =$ *1 MHz,* $SNR =$ *45 dB,* $C =$ *15 Mb/s*

Putting everything together:
Suppose the web server sends a file to Eliane's smartphone



Putting everything together

Eliane (Client)

Web server

Application

read(s1, dataBlock)

Transport (TCP)

1 2 3

Network (IP)

1 2 3

MAC (WiFi)

1 2 3

Physical

Router

Network (IP)

1 2 3

MAC (WiFi)

1 2 3

Physical

Switch

MAC (Ethernet)

1 2 3

Physical

Application

send(s2, dataBlock)

Transport (TCP)

1 2 3

Network (IP)

1 2 3

MAC (Ethernet)

1 2 3

Physical

Terminology

Eliane (Client)

Web server

Application

read(s1, dataBlock)

Transport (TCP)

1 2 3 4 5

MAC frame

Router

TCP segment

Application

send(s2, dataBlock)

Transport (TCP)

1 2 3 4 5

Network (IP)

2 3 4 5

Network (IP)

1 2 3

IP packet

Switch

Network (IP)

1 2 3 4 5

MAC (WiFi)

2 3 4 5

MAC

1 2 3

MAC (Ethernet)

1 2 3

MAC (Ethernet)

1 2 3 4 5

Physical

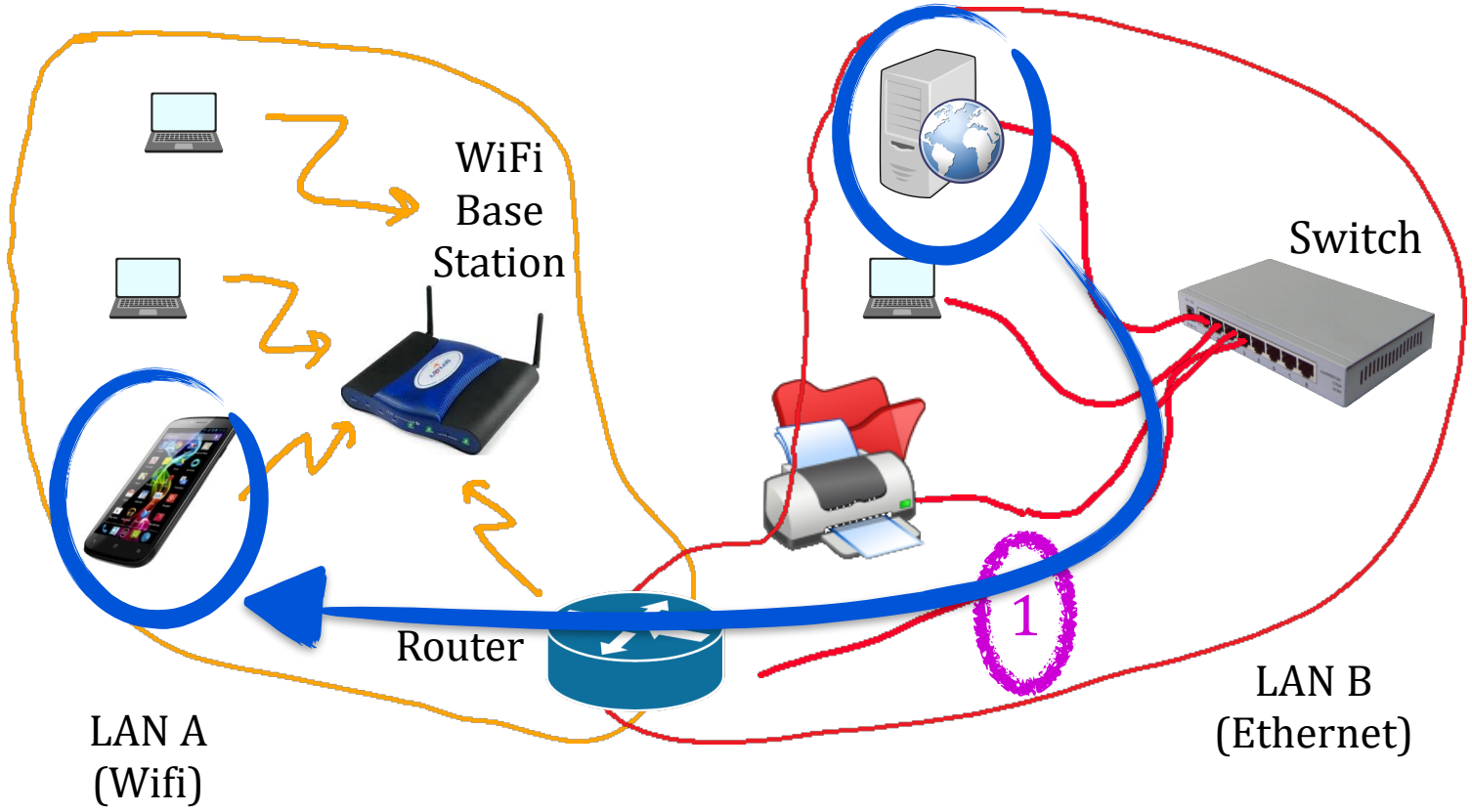
Physical

Physical

Physical

We observe a packet from Web server to Eliane at 1. What is true about its headers?

A. The destination MAC address is the MAC address of the router
B. The destination IP address is the IP address of the router
C. Both A and B
D. None

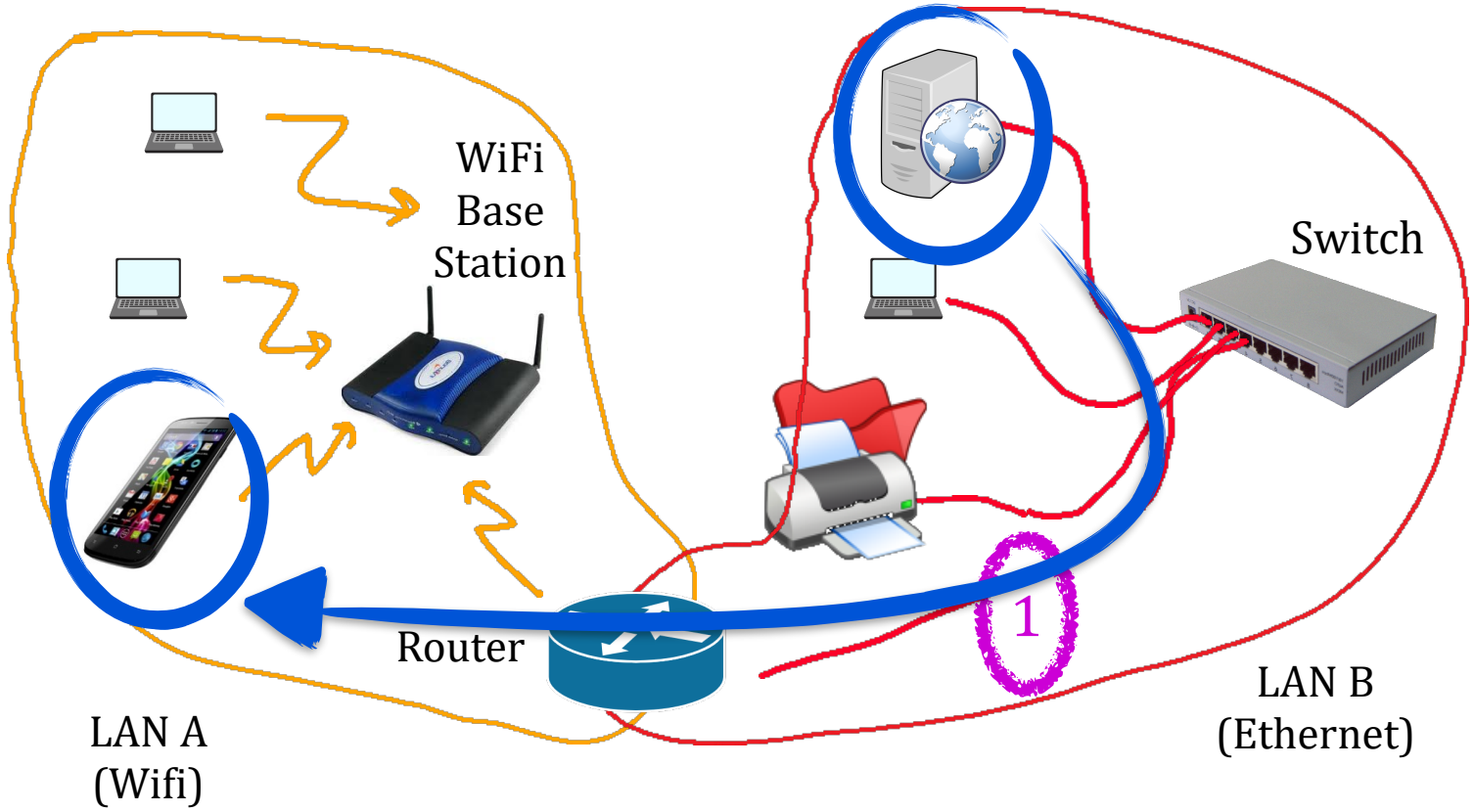


Solution

Answer A

The packet is sent by Web server to Eliane;

- the MAC destination address is the router's MAC address (MAC layer is local)
- the IP destination address is Eliane's device address (network layer is global)



Revisit another example: Access a URL

Name server (DNS server) with IP: 128.178.15.8

ic.epfl.ch is 128.178.222.83

Names are mapped to addresses by DNS servers— present only in payload, not in IP header

2

Dest: 128.178.15.8	What is the IP of ic.epfl.ch?

3

Dest: 128.178.56.9	ic.epfl.ch is 128.178.222.83

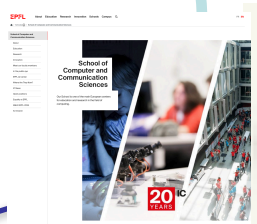
1

click

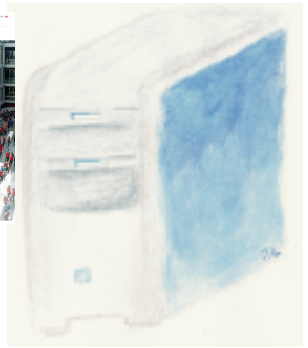
Client's machine with IP:128.178.56.9

Dest: 128.178.56.9	200 OK
Dest: 128.178.56.9	Other data

5



web server ic.epfl.ch with IP: 128.178.222.83



Dest: 128.178.222.83	HTTP get /index.html

4

Concepts that span all layers

important for the rest of the course and the labs…

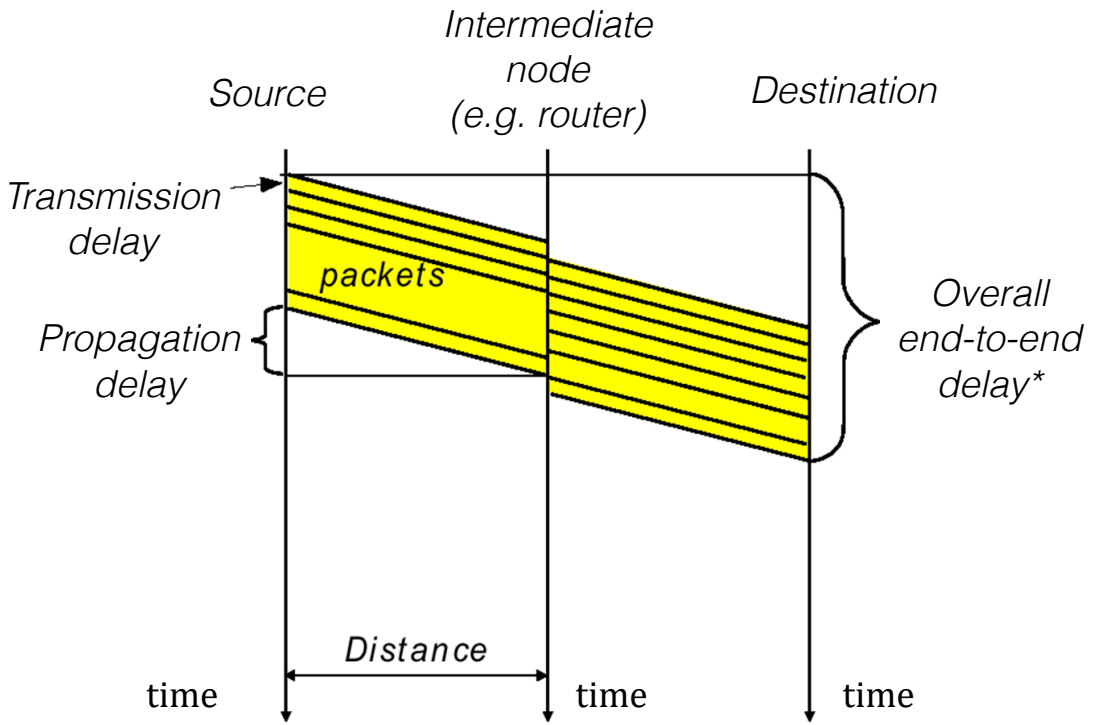
Concept 1: Network delays

- **Processing delay**
  Time required to examine a packet's IP header and determine where to forward it

- **Queueing delay**
  Time a packet needs to wait at a queue until it is transmitted into the outgoing link

- **Transmission delay** $= \frac{packetSize}{bitRate}$
  Time needed to push (i.e, transmit) all of the packet's bits into the outgoing link

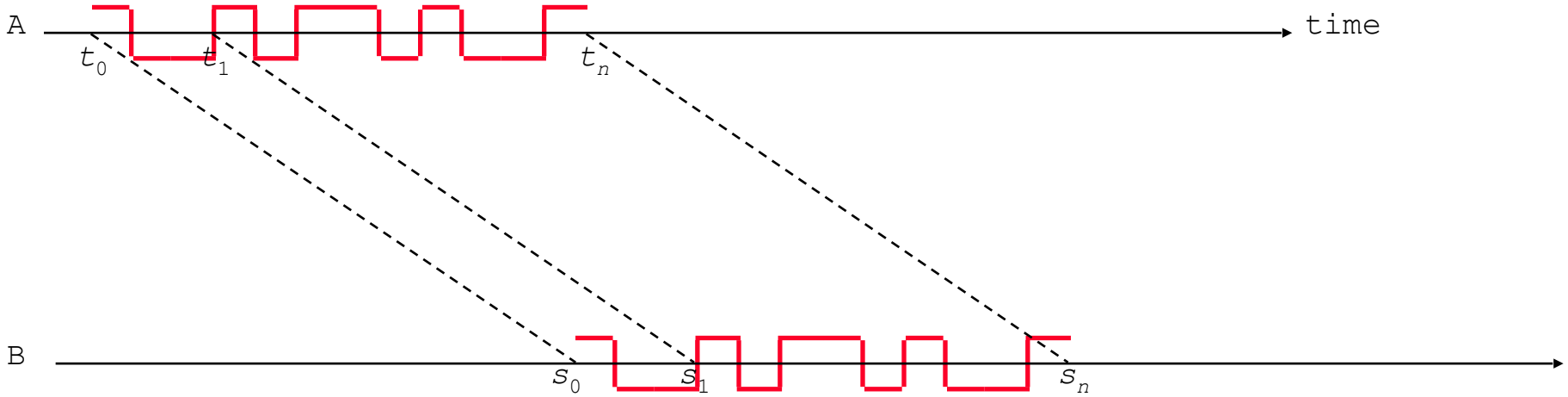
- **Propagation delay**
  Time for a bit to propagate from the beginning of the link to the other end, with the speed of light



*assuming ***no*** processing delays

Propagation delay: in more detail

Propagation between A and B = time for the head of signal to travel from A to B



$$D = s_0 - t_0 = \frac{d}{c} = \frac{distance}{speed\ of\ light}$$ (propagation delay for non acoustic channels)

In copper: c = 2.3e+08 m/s; in glass optical fiber: c = 2e+08 m/s;

Rule of thumb: 5 μs/km; around the globe = 200 msec

Example: Time it takes to send one packet of 1kB(≈8000bits)

* assuming *no* queueing, *no* processing delays; say there is only one link…

	Data Center	ADSL	Modem	Internet
Distance	20 m	2 km	20 km	20000 km
Bit rate	1 Tb/s	10 Mb/s	10 Kb/s	1 Mb/s
Propagation delay	**0.1µs**	0.01ms	0.1ms	**100ms**
Transmission delay	0.008µs	**0.8ms**	**800ms**	8ms
Total*	0.108 µs	0.81ms	800.1ms	108ms

- Each time, the total delay may be *dominated* by a different type of delay
- We always want to know what the *bottleneck* is

Pigeon outruns South African ADSL

11 September 2009 | 14:28

A South African information technology company has proved it's faster for them to send data by carrier pigeon than using the country's leading internet provider.

Winston the pigeon has easily outpaced South Africa's leading broadband netw ork it moving data (AAP)

A South African information technology company has proved it's faster for them to transmit data by carrier pigeon than to send it using Telkom, the country's leading internet service provider.

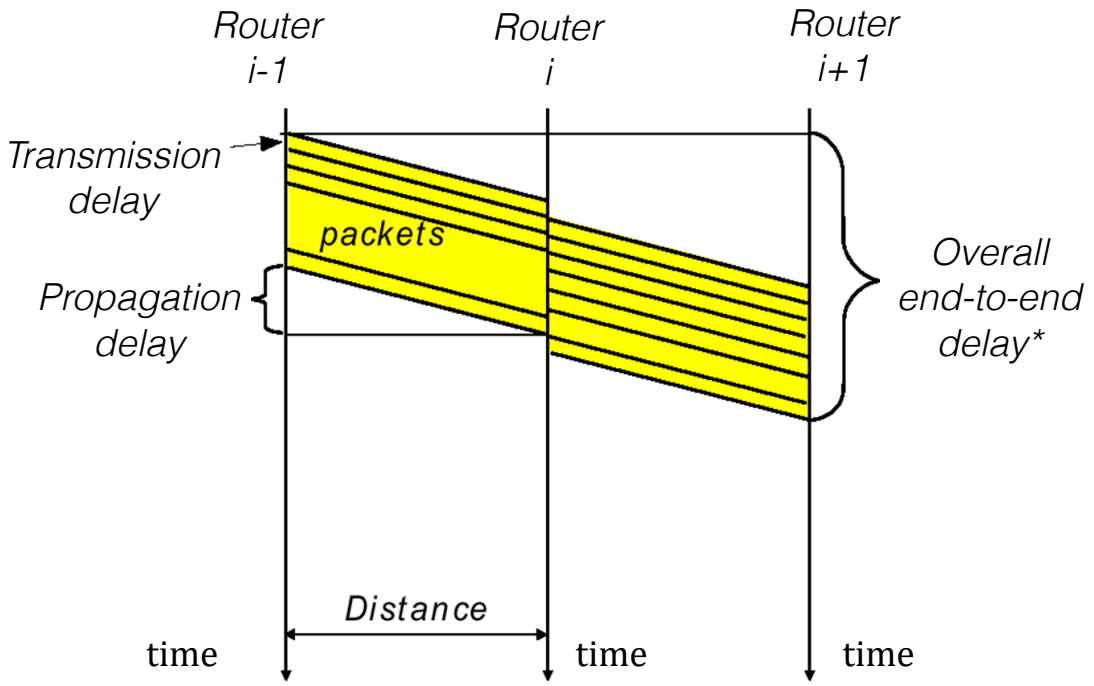
Internet speed and connectivity in Africa's largest economy are poor because of a bandwidth shortage. It is also expensive.

An 11-month-old pigeon, Winston, took one hour and eight minutes to fly the 80 km (50 miles) from Unlimited IT's offices near Pietermaritzburg to the coastal city of Durban with a data card strapped to its leg.

Including downloading, the transfer took two hours, six minutes and 57 seconds – the time it took for only four percent of the data to be transferred using a Telkom line.

Suppose that processing delays are insignificant, why is packet switching more efficient than relaying the entire message?

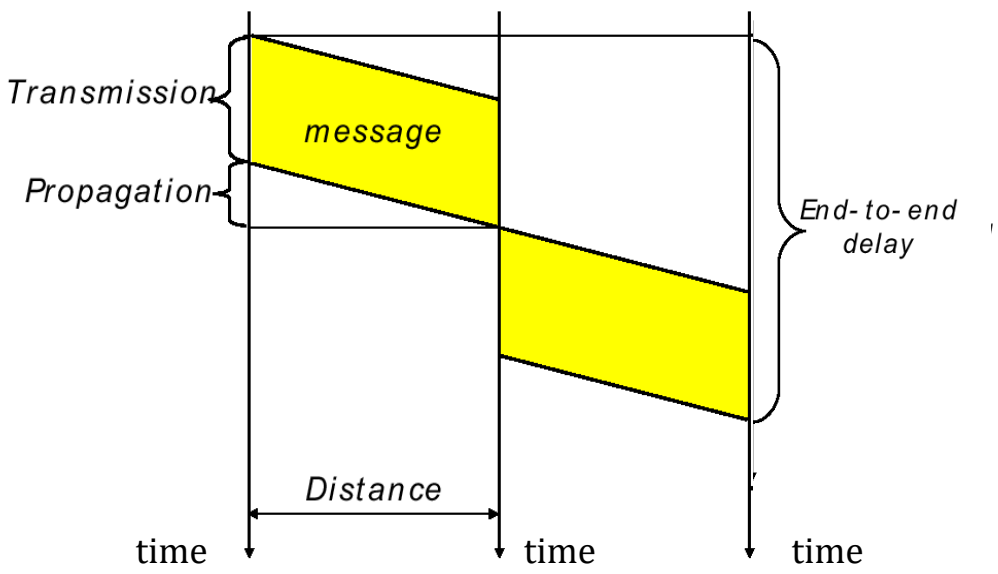
A. It reduces buffer required in routers
B. It reduces the bit error rates
C. It increases the (information-theoretic) capacity of the links
D. The end-to-end overall delay is reduced

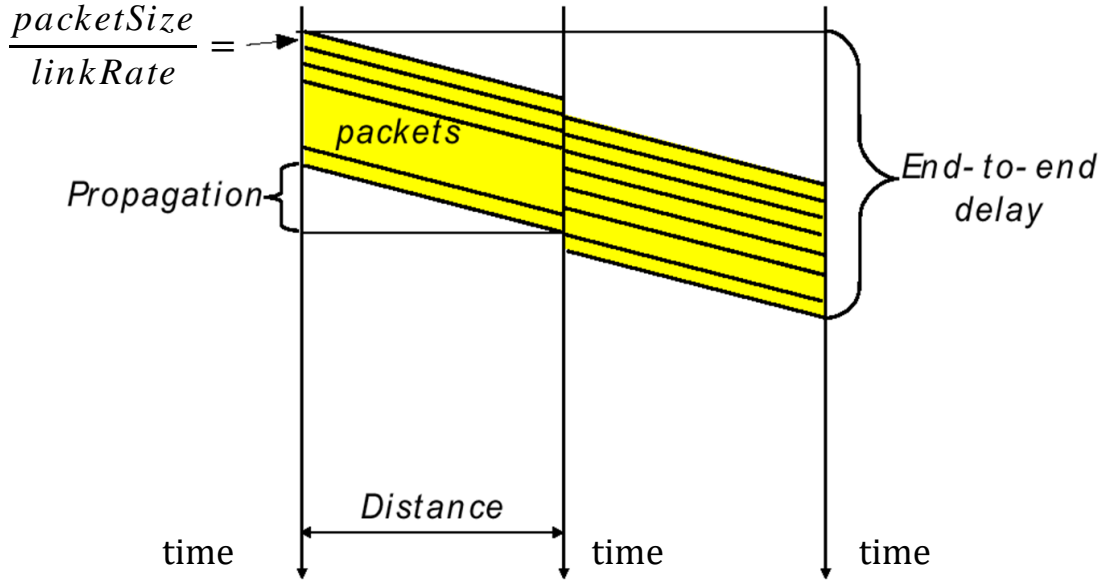


*assuming ***no*** processing delays

Solution

Entire message relay vs Packet Switching





Correct answers = A and D

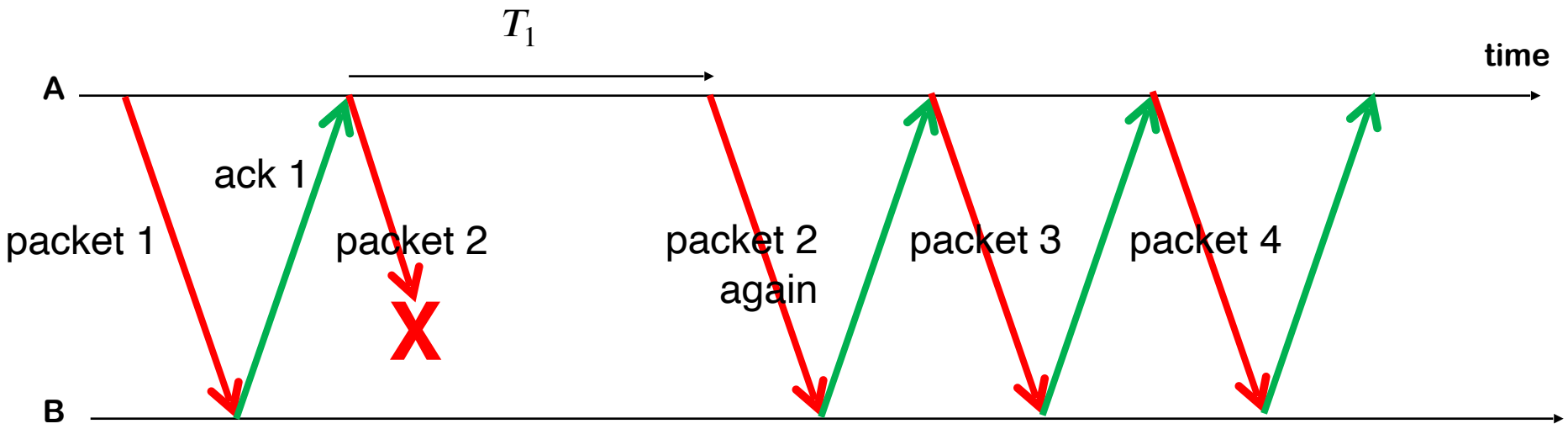
- End-to-end delay is reduced
- Required buffer space at intermediate systems is reduced
- Bit error rate remains the same, but the probability that a packet is corrupted because of a bit error is smaller than for an entire message (as a message is typically larger than a single packet).
- Capacity is not increased (see “Physical layer”).

Concept 2: Throughput

- Throughput = number of useful data bits at destination / time unit
- Same units as the bit rate: b/s, kb/s, Mb/s
- Even if computed over a single link, it is not the same as the bit rate. Why?
  - *overhead:* all protocols use some extra bytes to exchange metadata
  - *waiting times:* a host in one end usually waits a response from the other before transmitting

Example: The Stop and Go Protocol

- Simple protocol for repairing packet losses:
  - A sends packets to B; B returns an *acknowledgement* packet immediately to confirm that B has received the packet;
  - A waits for acknowledgement before sending a new packet;
    if no acknowledgement comes after a delay T_1, then A *"timeouts"* and *retransmits*



Throughput of the Stop and Go Protocol

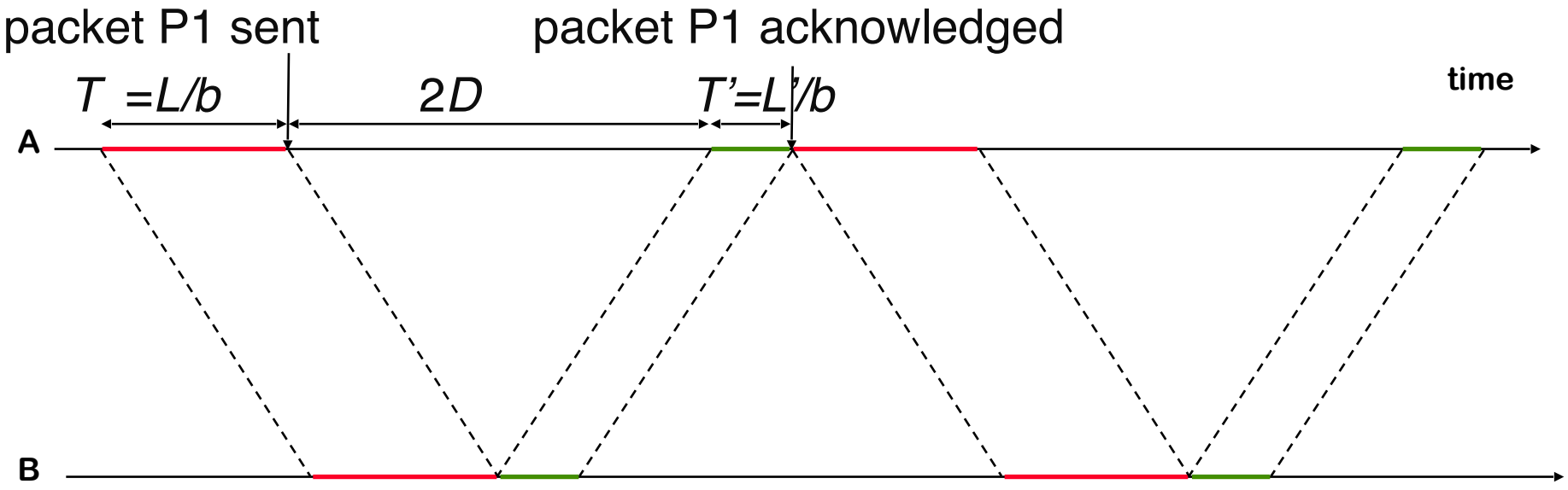
$L =$ packet size; $b =$ channel bit rate; $D =$ propagation delay; $L' =$ ack size

Assume: always one packet to transmit, no loss.

- In each cycle, L useful bits are transmitted.
- The cycle lasts $T + 2D + T'$.
- Throughput $= \dfrac{L}{T + 2D + T'} = \dfrac{b}{1 + \frac{L'}{L} + \frac{2Db}{L}}$

Overhead

"Bandwidth-Delay Product" = #bits transmitted, but not yet ack'ed.
(Note: bandwidth is "mistakenly" used instead of bit rate)



Throughput of Stop and Go

	Data Center	ADSL	Modem	Internet
Distance	20 m	2 km	20 km	20000 km
Bit rate	1 Tb/s	10 Mb/s	10 Kb/s	1 Mb/s
Propagation delay	**0.1µs**	0.01ms	0.1ms	**100ms**
Transmission delay	0.008µs	**0.8ms**	**800ms**	8ms
Total*	0.108 µs	0.81ms	800.1ms	108ms
BW-delay product	200Kb	200b	2b	200Kb
Throughput of Stop&Go*	**3.8%**	97.56%	99.98%	**3.8%**

* with packets of size 1kB ≈ 8'000 bits, and assuming no processing delays and negligible overhead

We will see that TCP does better than Stop and Go using a smarter mechanism (*sliding window*)