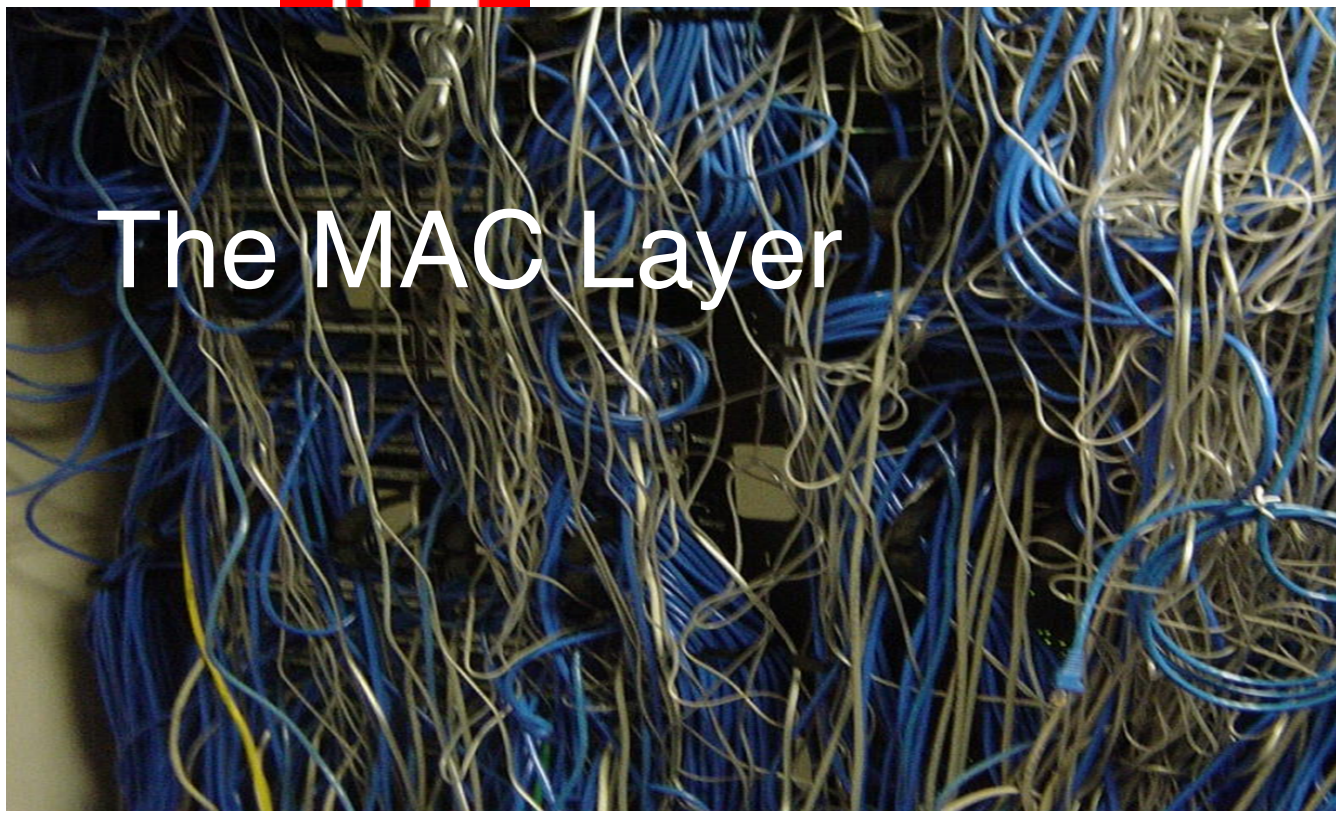
EPFL

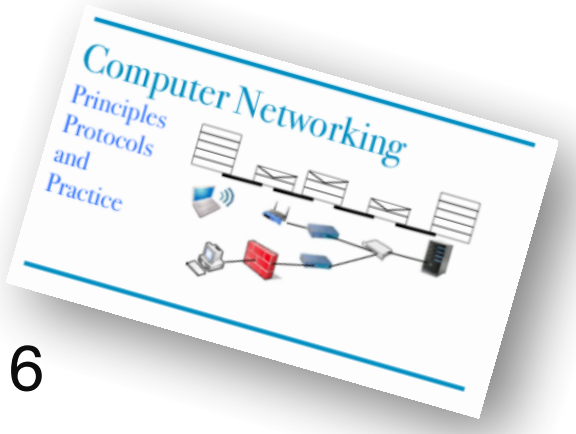


The MAC Layer

Contents

1. MAC as Shared Medium
2. Switches
3. Frame format and MAC addresses
4. Virtual LANs
5. WiFi Distribution Systems
6. Security aspects

Textbooks



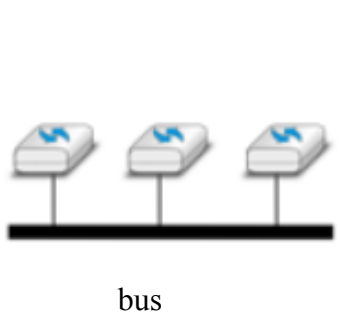
Part 6

Chapter 6

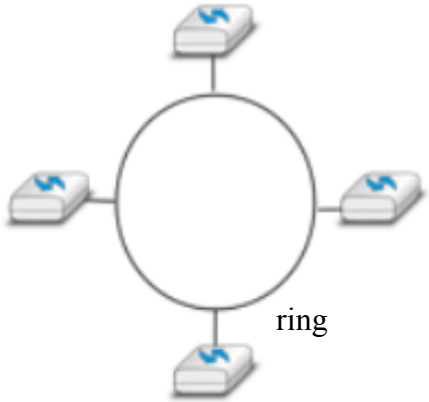
1: MAC: Medium Access Control

Why invented? So that multiple LAN devices *share* a single physical medium:

- cable (old/coaxial Ethernet, Powerline); or
- wireless radio link (GSM, WiFi, Bluetooth)



bus



ring

star

What is the problem ?

If several systems transmit at the same time, several frames will appear on the transmission medium (channel) and they will *collide/interfere*.

Solution 1: joint decoding (allow everyone to talk and decode) -- used in cellular networks (e.g. CDMA)

Solution 2: *mutual exclusion* (only one system can talk at a time), ideally via a *distributed* protocol

How to implement mutual exclusion?

Deterministic protocols

- use an *arbitration* for accessing the transmission medium (channel)
  - Time Division Multiple Access (**TDMA**): each host is given a ***dedicated timeslot*** to transmit
  - Frequency Division Multiple Access (**FDMA**): each host is given a ***dedicated frequency band*** to use
  - CAN bus: the ***highest-priority*** transmitting host dominates other transmissions
  - Token ring, FDDI: every host takes turn and passes a ***token*** to next host; whoever has the token, can transmit
    - ✦ more flexible than TDMA and FDMA, may achieve higher throughput
- *avoid* collisions by construction, *proactively*
- but…work only with few users, as they require synchronization and pre-configuration
  (if a host goes down or a new one appears, we need to take the network down and reconfigure it)

Stochastic (random access) protocols

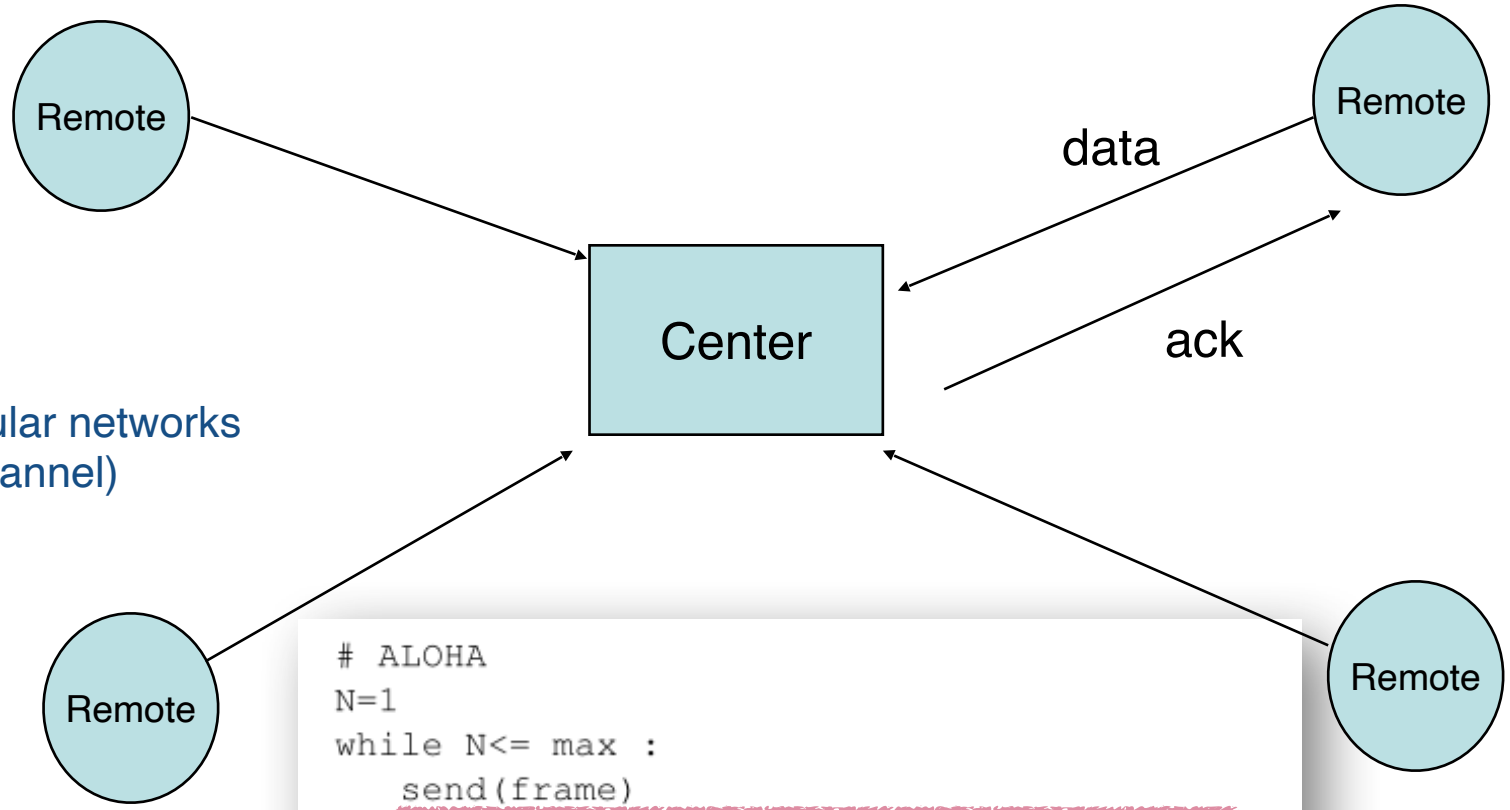
- *allow everyone* to transmit at any time, *detect collisions*, and *react*
- hosts transmit at full rate and if a collision is detected, they retransmit after some *random* time ("back-off")
  - WiFi, coaxial/historical Ethernet, Zigbee, signaling channel on cellular networks, Ethernet over Powerline

Principles of random-access protocols

… step by step

ALOHA

(used today in cellular networks for the signaling channel)



shared channel from remotes to center

```
# ALOHA
N=1
while N<= max :
    send(frame)
    wait(ack_on_return_channel or timeout)
    if (ack_on_return_channel):
        break  # transmission was successful
    else:
        # timeout
        wait(random_time)
        N=N+1
 else:
    # Too many transmission attempts
```

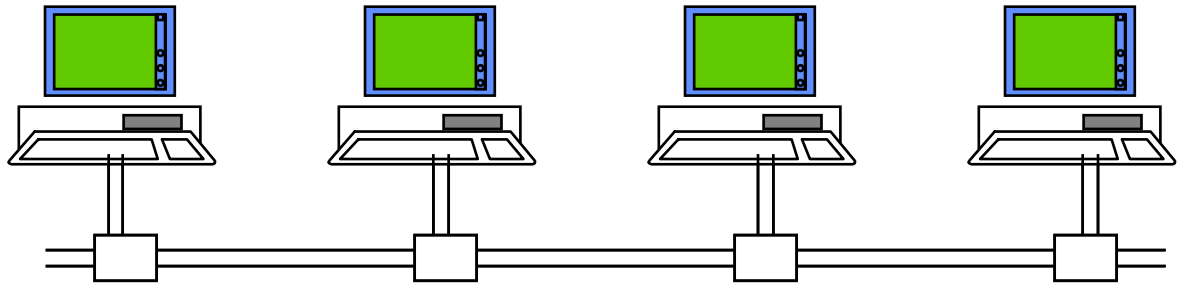
Transmission procedure at remote

ALOHA

- Aloha is the basis of all non-deterministic access methods. The Aloha protocol was originally developed for communications between islands (University of Hawaï) that use radio channels at low bit rates.
- It assumes one shared channel from remote to center, and a separate (non interfering) channel from center to remotes.
- Collisions occur when two packet transmissions overlap, and if a packet is lost, then source has to retransmit; the retransmission strategy is not specified here; many possibilities exist about how to pick the “random” waiting time. We will see one approach later when discussing CSMA/CD.
- The picture shows a radio transmission scenario; Aloha can also be used on a cable (bus). It is used nowadays in cases where simplicity is more important than performance (for example: machine bus).
- The *maximum utilization* is 18% of the channel capacity, which is very small. This is assuming independent and exponential inter-arrival times of packets and retransmissions.

➡ Slotted ALOHA can improve utilization by a factor of 2.

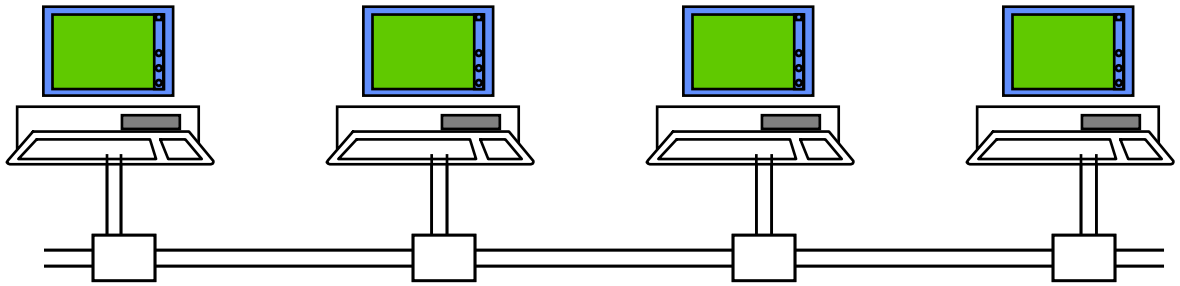
CSMA (carrier sense multiple access)



```
# persistent CSMA
N=1
while N<= max :
  listen(channel)
  wait(channel_becomes_free)
  send(frame)
  wait(ack or timeout)
  if ack :
      break  # transmission was successful
  else :
      # timeout
      N=N+1
# end of while loop
  # Too many transmission attempts
```

- assumes a single *transitive* channel (everyone can hear everyone)
- requires that hosts *"sense"/listen* the channel before transmitting
- drastically *improves efficiency* over Aloha

In CSMA, collisions…



A. cannot occur because the medium is transitive

B. cannot occur if transmission delays are very small

C. can still occur sometimes

D. I don't know

Go to web.speakup.info or download speakup app

Join room 87072

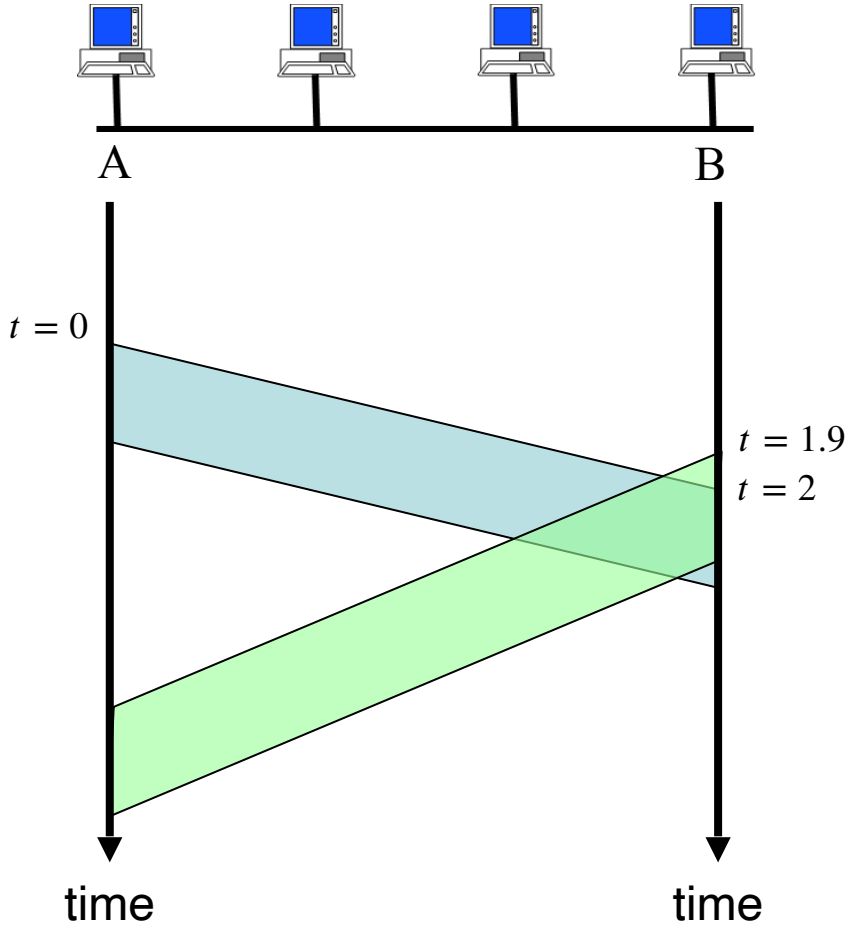
Solution

Many collisions can be avoided, but not all.

Two or more stations may sense that the medium is idle and start transmitting at time instants that are close enough for a collision to occur.

<u>Example in figure:</u>

- Assume propagation delay between A and B is 2 ms and that all stations are silent until t = 0.
- At t=0, station A starts transmitting.
- At t=1.9ms, station B has not received any signal from A yet, so it also starts transmitting.
- At t=2ms, station B senses the collision but it is too late —> a collision has occurred.

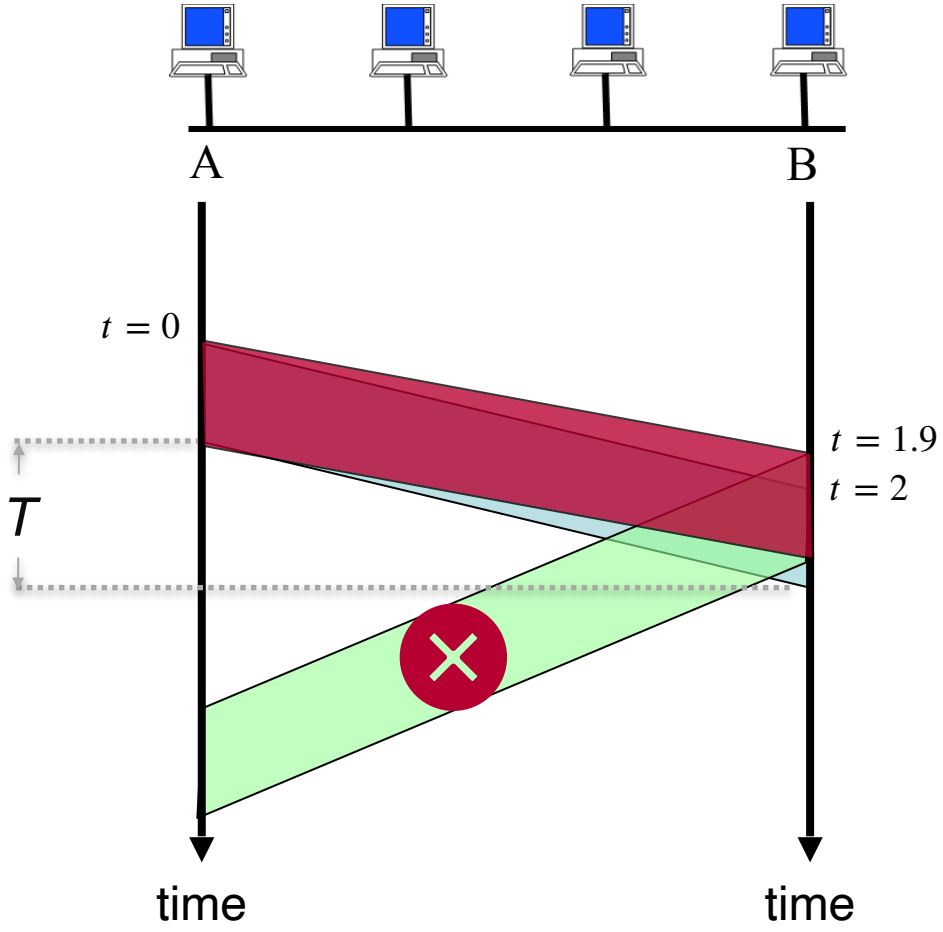


Importance of propagation delay

The effect of CSMA can be expressed as follows:

- Let *T* be maximum propagation delay from station *A* to any other stations;
- if no collision occurs during a time duration *T* after *A* starts transmitting,
  then *A* has *successfully* seized the channel (= no other station can send)

➡ CSMA works better if: propagation delay is *small*



Non-persistent CSMA

- uses *random* waiting times if channel is busy
- avoids hosts' synchronization
  - if all stations choose the random delays *independently*, and
  - if the waiting times are *larger* than the propagation delay,
  - then, with high probability, only one of the transmitting stations seizes the channel
- improves utilization

```
# Non persistent CSMA
N=1
while N<= max :
   listen(channel)
   if free(channel):
      send(frame)
      wait(ack or timeout)
      if received(ack) :
         break  # transmission was successful
      else :
         # timeout
         N=N+1
   else:
      wait(random_time)
 # end of while loop
   # Too many transmission attempts
```

CSMA / CD (Collision Detection) detects collisions as they occur

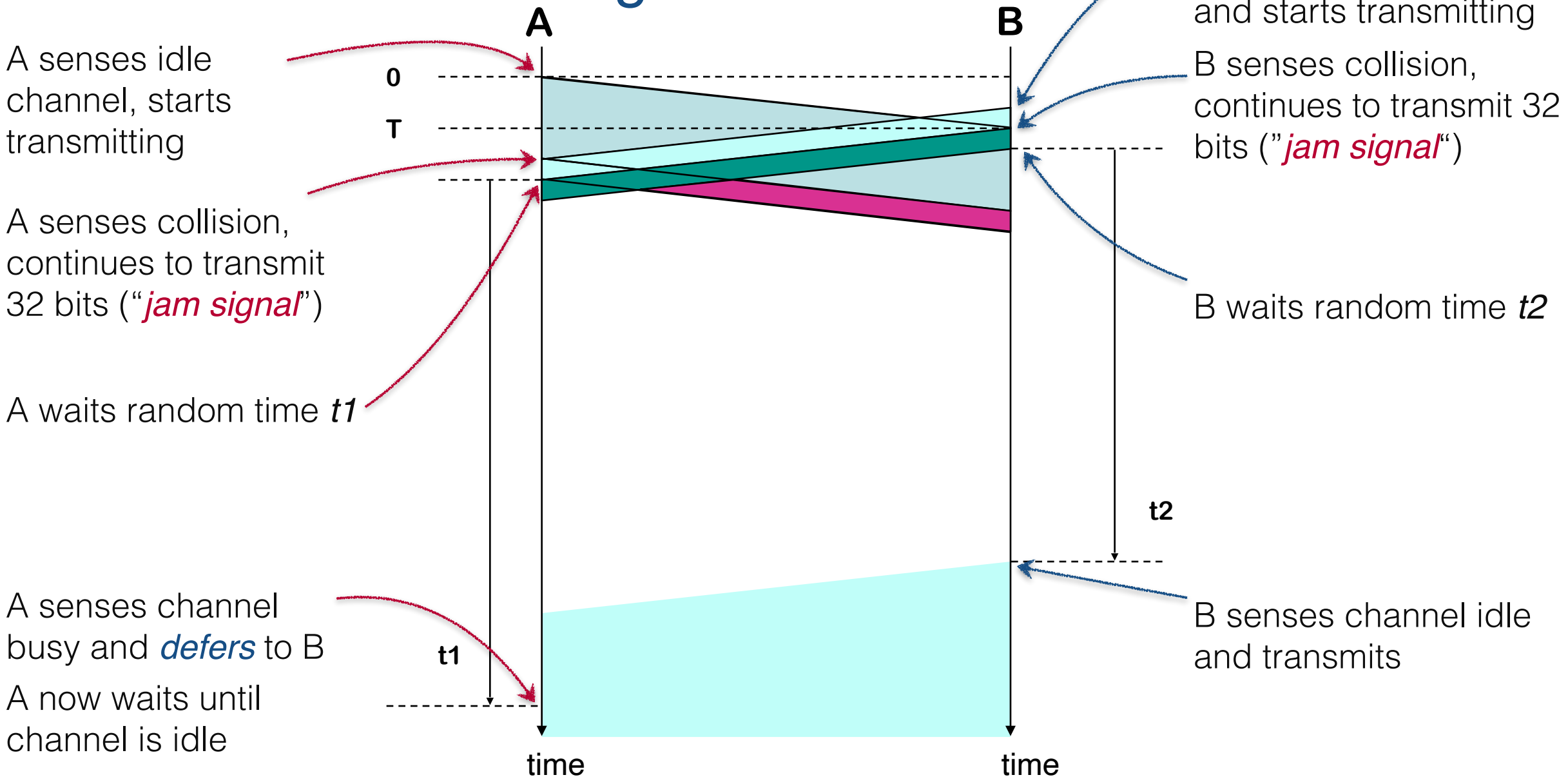
ACKs replaced by CD:

channel is monitored even during transmission to detect collisions

```
# CSMA/CD pseudo-code
N=1
while N<= max :
    wait(channel_becomes_free)
    send(frame)
    wait_until (end_of_frame) or (collision)
    if collision detected:
        stop transmitting
        send(jamming)
        k = min (10, N)
        r = random(0, 2^k - 1) * slotTime
        wait(r*slotTime)
        N=N+1
    else :
        wait(inter-frame_delay)
        break
 # end of while loop
   # Too many transmission attempts
```

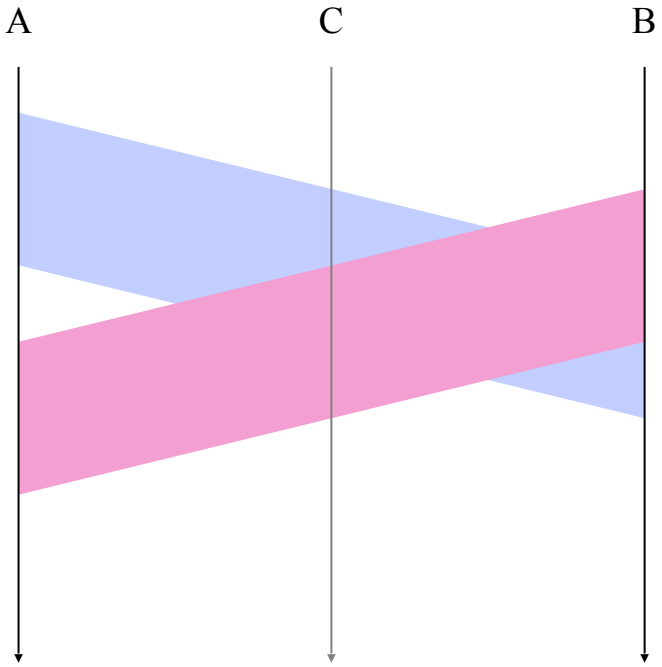
Exponential back-off

CSMA / CD Time Diagram

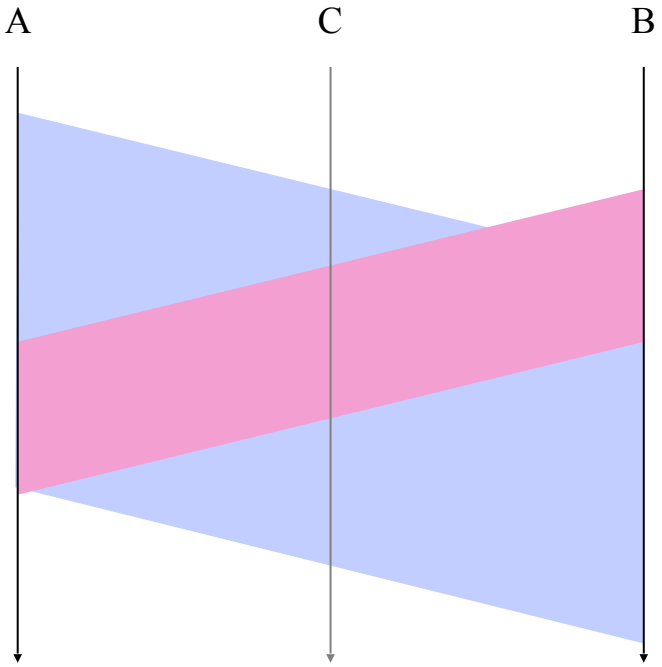


A Minimum Frame Size is needed to guarantee collision detection

Problem: Frame sent by A is *too short*;
collision is not visible at A (but only at C)



Solution: Frame sent by A is *larger*;
collision is visible at A



➡ To avoid undetected collisions, we impose a minimum frame size that depends on the link's *bit rate* and round-trip propagation delay *(a.k.a. slotTime)*:

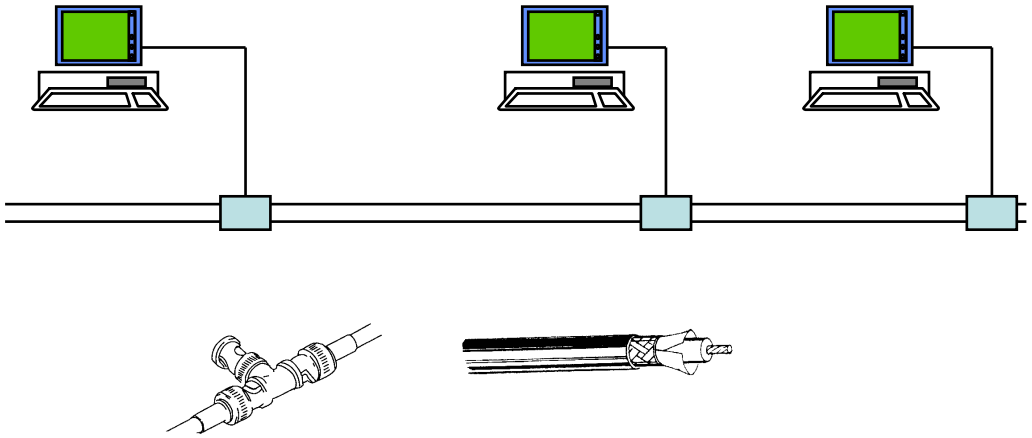
```
transmission delay = frameSize/bitRate > slotTime
=> frameSize > slotTime*bitRate  ⟶  ≈ bandwidth-delay product
```

CSMA / CD

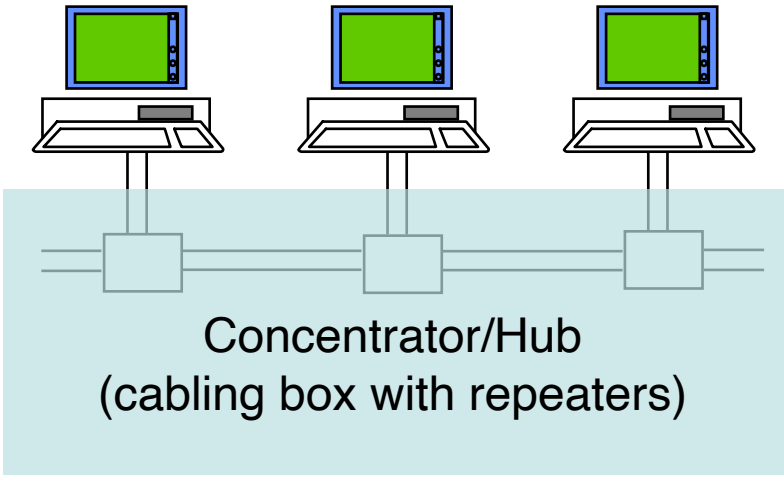
- CSMA/CD improves on CSMA by requiring that stations detect collisions and stop transmitting (after 32 bits, called ***jam*** bits, in order to ensure that all systems properly recognize the presence of collisions).
- The jam bits (or jam signal) is a 32-bit binary pattern indicating that a collision has occurred.
- CSMA/CD has a *better performance* than Aloha or CSMA
- After a collision is detected, stations will re-attempt to transmit after a random time.
- The random time before retransmission is chosen in such a way that if repeated collisions occur, then this time increases exponentially *(“exponential backoff”)*. To see this, check again the pseudocode from a couple of slides earlier.
  The downside of the exponential backoff is that: in case of congestion (too many collisions), the access to the channel is slowed down.
- Acknowledgements are not necessary because absence of collision means that the frame can be transmitted (see “Minimum Frame Size“ in previous slide).
- An inter-frame delay (“gap”) of 9.6 µs is used to avoid blind times, during which adapters are filtering typical noise at transmission ends.

CSMA/CD = early Ethernet (with coax cables and hubs)

Coax cables with T-connectors



Concentrator/Hub-based topology



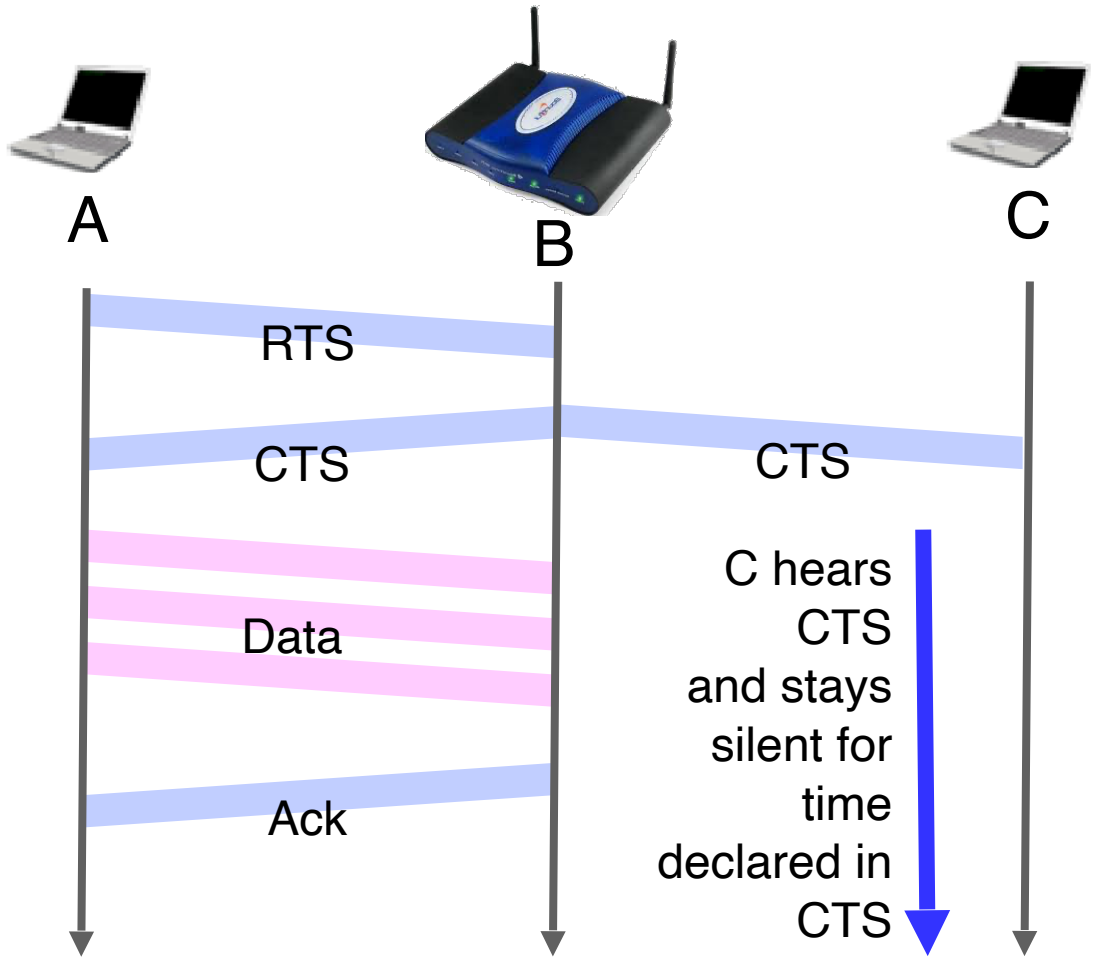
Early Ethernet, a.k.a. *half-duplex* Ethernet, addresses *collisions* by using:

- CSMA/CD
- with restrictions about the slotTime (hence also the min frame size)
  - e.g. 10Mbps half-duplex Ethernet uses *slotTime = 51.2ms => min frame size = 64Bytes*

WiFi = CSMA/CA (Collision Avoidance) = variant of CSMA/CD

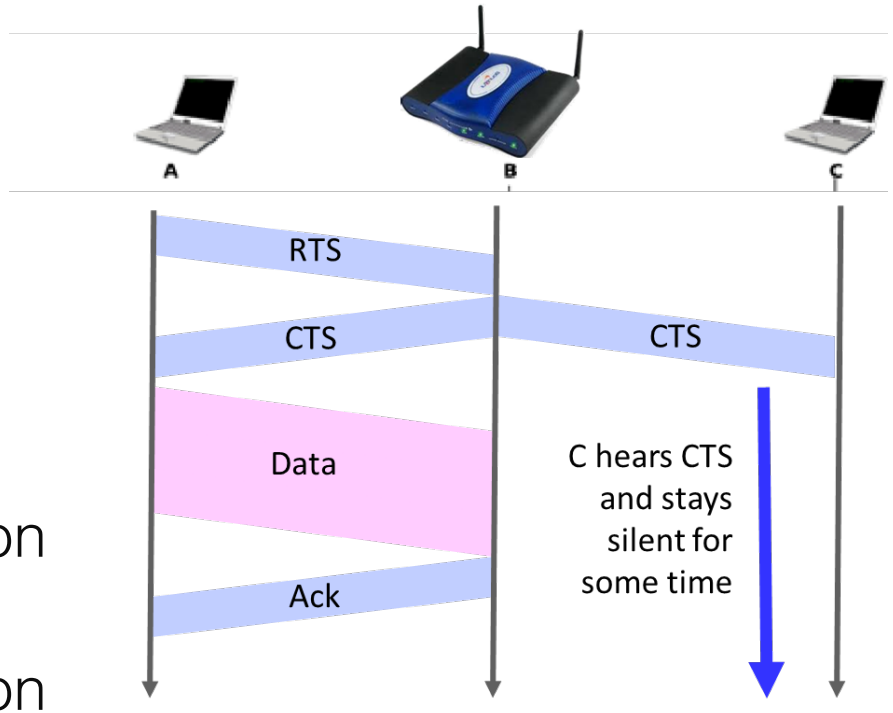
Uses:

- ACKs to detect collisions
  (i.e. if they are absent)
- Carrier sensing, but does not always work:
  - e.g., C may be “hidden” from from A;
  - and this is *mitigated* through *RTS/CTS*
- Further optimizations to avoid collisions:
  - more *conservative channel backoff*:
    even if the channel is idle,
    still wait for random time before transmitting
  - *network allocation vector*:
    MAC header contains the transmission time
    needed for the frame;
    it is a form of *virtual* carrier sensing



In a WiFi network with a single channel and a single base station, how many data frames can be transmitted concurrently ?

A. At most one in total
B. One from mobile to base station and one from base station to mobile
C. At most one per mobile to base station and one from base station
D. At most one per mobile to base station and one per mobile from base station
E. I don't know



Go to web.speakup.info or download speakup app

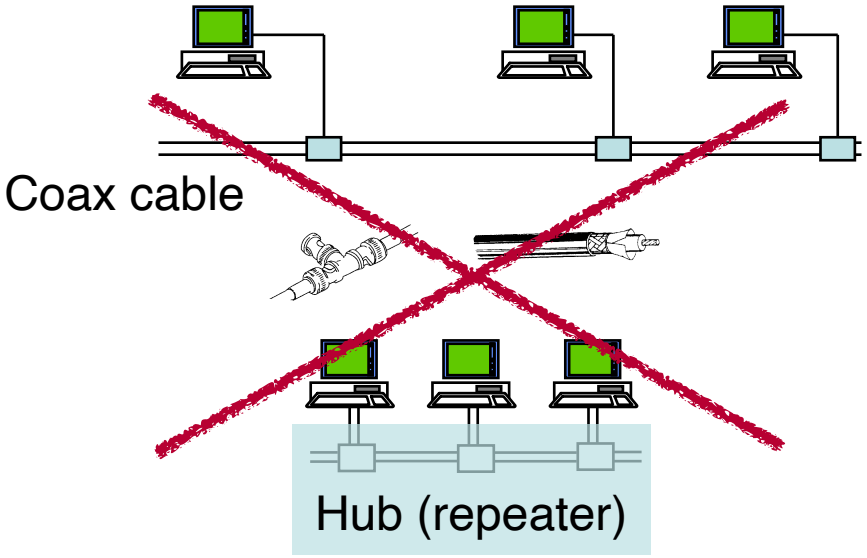
Join room 87072

Solution

Answer A
The WiFi MAC layer controls the access of all senders to the radio channel. So, it makes sure that only one can transmit at a time.

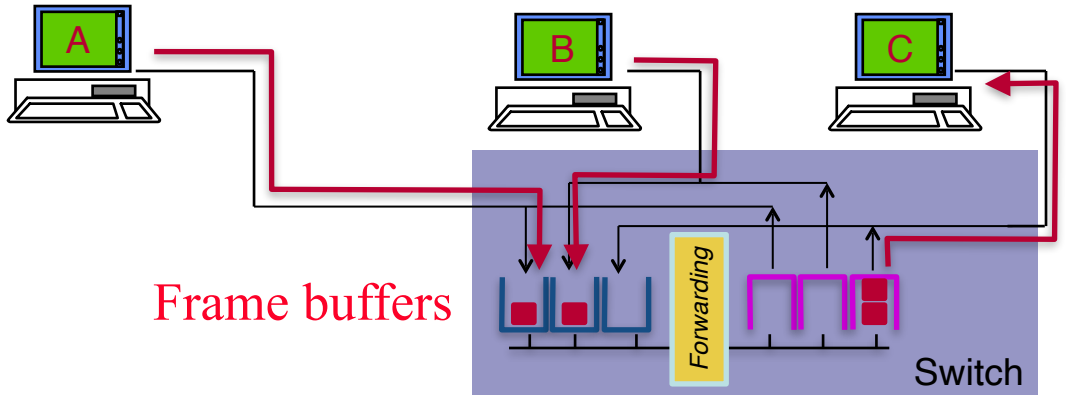
2. (Today's) Switched Ethernet

Early Ethernet was shared medium
(either used coax cables or repeaters/hubs
with twisted-pair copper wires)



Today, Ethernet is based on:

- *switches* (or *bridges*):
  - = middle boxes of the MAC layer
  - = queuing systems (with frame buffers) that forward frames based on MAC addresses



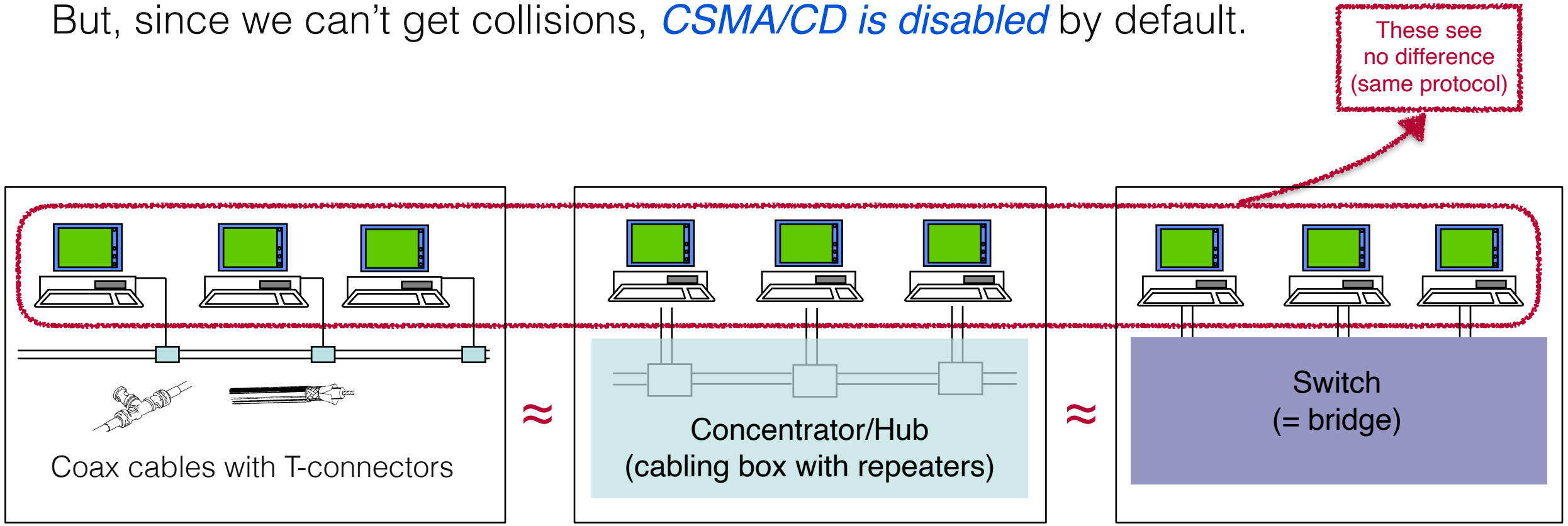
- *full-duplex cables*:
  - frames flow simultaneously in both directions without suffering collisions (*collision-less* ethernet)
  - e.g. the well-known **UTP** cables with separate downlink/uplink

Is switched Ethernet a different protocol?

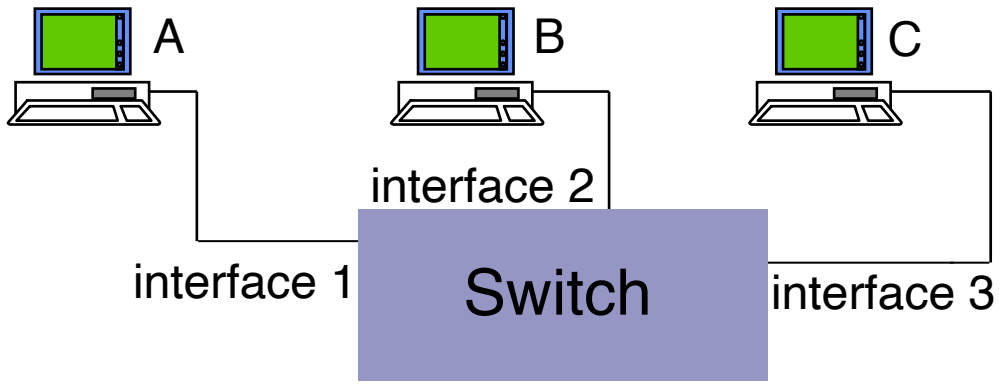
No, switches are designed to be *transparent*:

i.e. there should be *no difference* for the protocol used by end-systems when they are connected to a coax cable, concentrator/hub or a switch/bridge

But, since we can't get collisions, *CSMA/CD is disabled* by default.



How does a switch forward a frame?



Switch's Forwarding Table

Destination MAC address	Outgoing Interface/Port
MAC of A	1
MAC of B	2
MAC of C	3

- Switches use:
  - a forwarding table with *exact match*

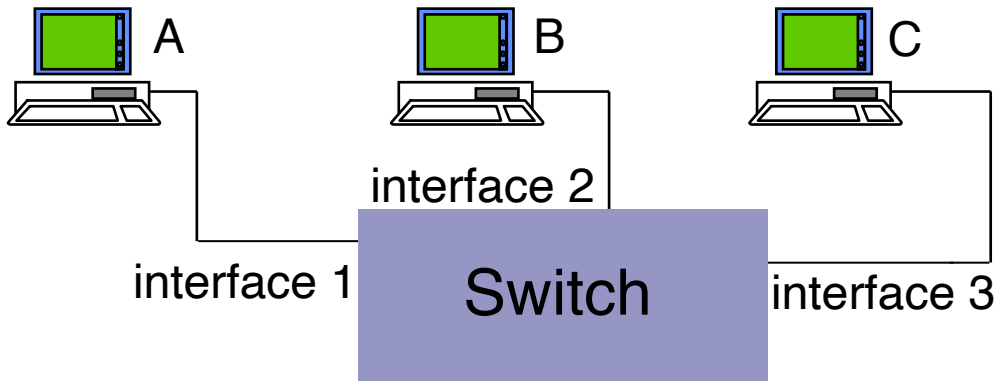
    which maps destination MAC addresses to outgoing interfaces (a.k.a. "ports")
  - and a *forwarding algorithm*

- Smart switches handle *multicast frames differently* [see later]

Switch's Forwarding algorithm

```
Observe all traffic at all ports
For each incoming frame:
  read destination MAC address in MAC header
  if entry for MAC exists in forwarding table:
     obtain matching outgoing port
     if outgoing port = incoming port:
          discard frame (no need to forward)
     else
          forward frame to outgoing port
  else (destination MAC not in the table):
     broadcast to all ports, except for the
     incoming port
```

How does a switch populate its forwarding table?



Switch's Forwarding Table

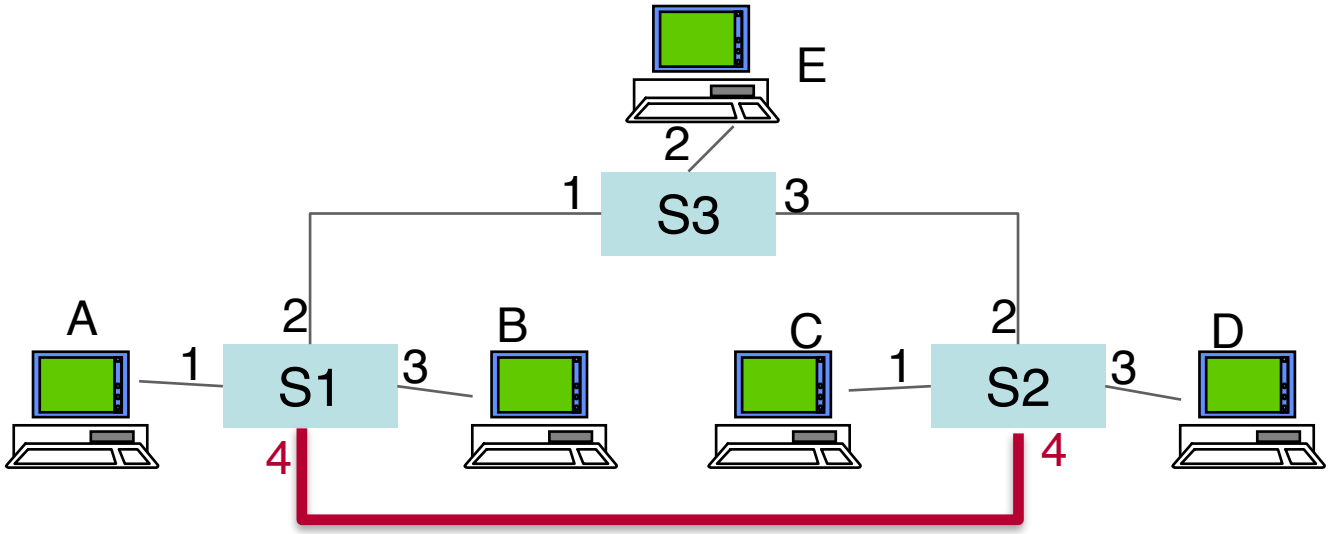
Destination MAC address	Outgoing Interface/Port
MAC of A	1
MAC of B	2
MAC of C	3

By observing traffic (*plug-and-play device*):

- table built by self-learning from Source MAC Address field in incoming frames
- learnt addresses time out if not re-learnt after an ageing time

➡ *different* from IP routers that use routing algorithms

Can self learning be extended to inter-connected switches?



Forwarding Table at S3

Destination MAC address	Outgoing Interface/Port
MAC of A	✗ 3
MAC of B	1
MAC of C	3
MAC of D	3
MAC of E	2

A. Yes because learning propagates
B. Yes because MAC addresses are unstructured
C. Only in some special cases
D. No, never
E. I don't know

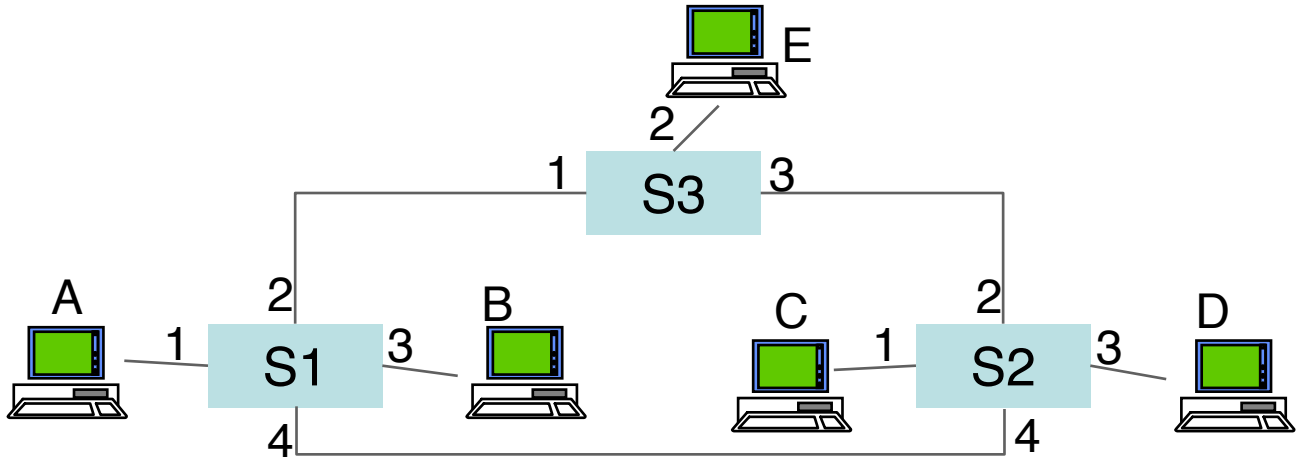
Go to web.speakup.info or download speakup app

Join room 87072

Solution

Answer C

It does not work if there are *loops* in the topology



E.g.: here S3 does not know whether A is on port 1 or 3, as flooded/broadcast packets may arrive from either side.

(Recall that switches are queueing systems, hence there exist non-deterministic delays)

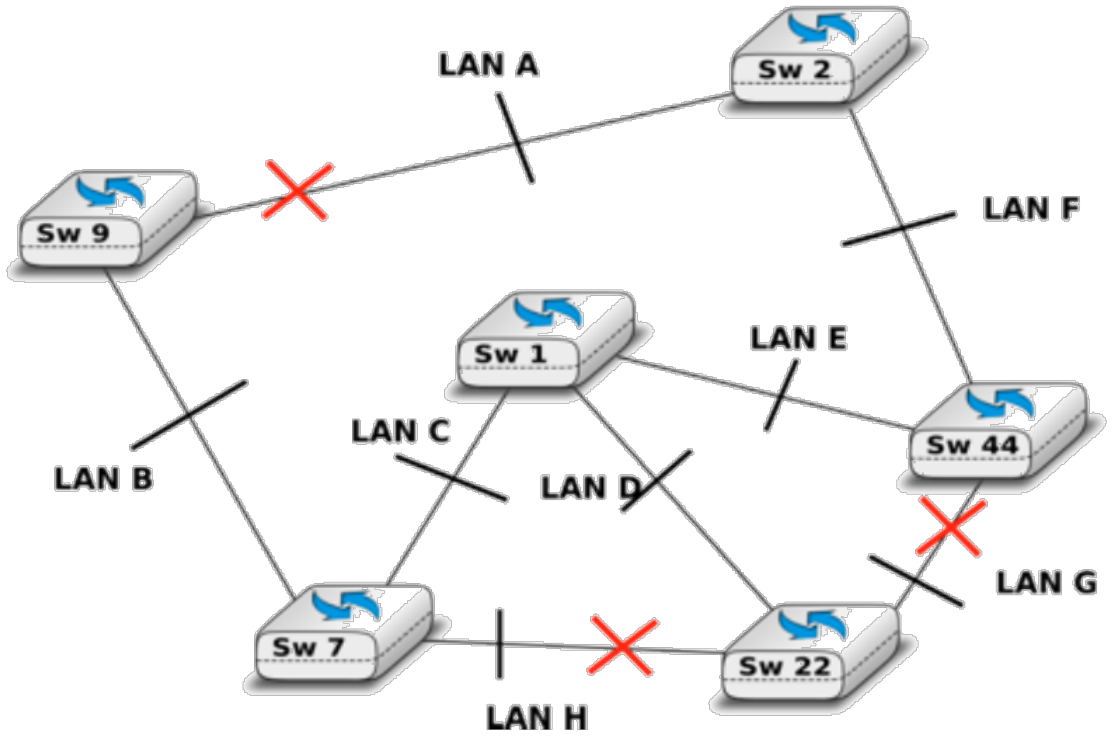
➡ The entries of the forwarding table may change *indefinitely* and packets may be trapped in a link, as we saw when talking about TTL in IP layer

Spanning Tree Protocol (STP): the common way to prevent loops

What does it do?

- forces the active topology to be a *tree* (i.e. loop-free) that spans all switches by *deactivating* some links

- adapts to link failures and additions whenever needed

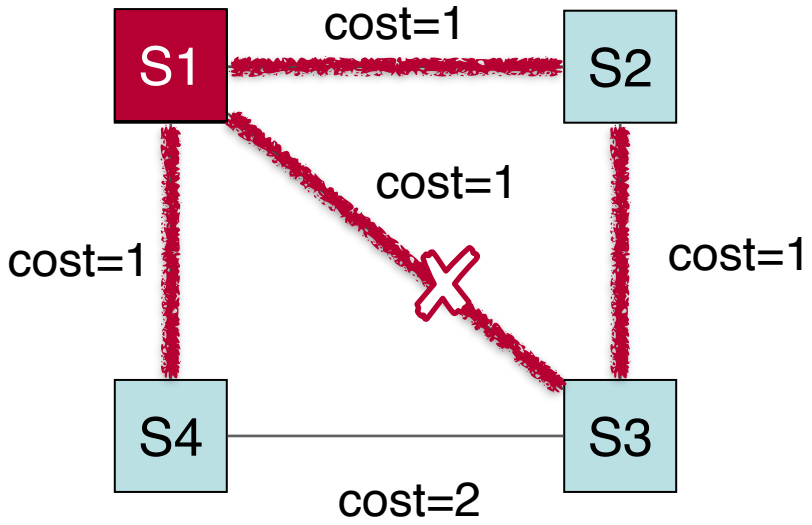


How does the Spanning Tree Protocol work?

The switches:

1. elect a *root* switch = switch with smallest (configurable) label ID
2. activate only the links that are along the shortest path to root switch

   spanning tree = set of *shortest paths* to root switch
3. monitor whether the root is reachable, and
   if not, they trigger a re-computation of a new spanning tree



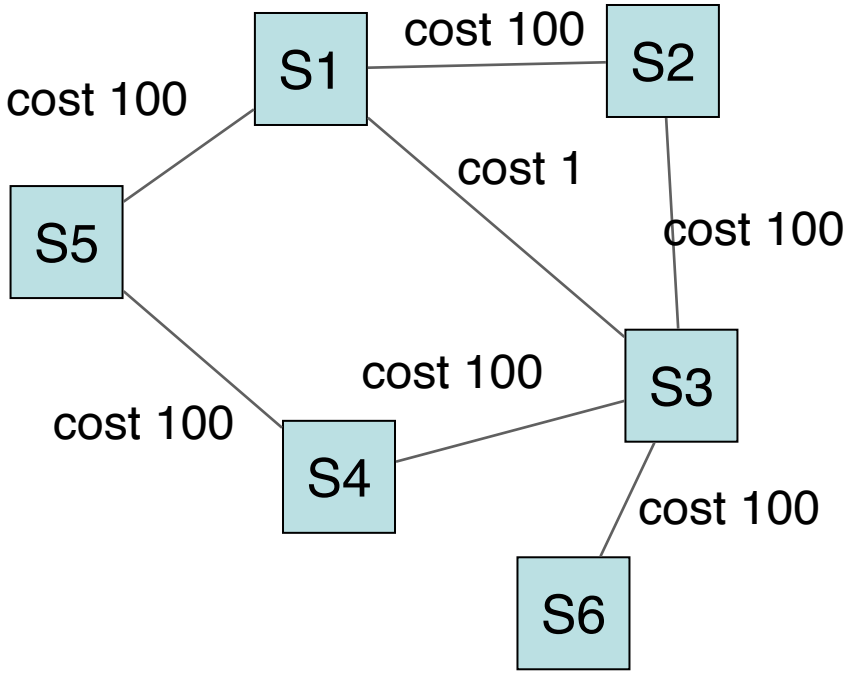
A few important points

- STP is a *distributed* algorithm
  - no central control,
  - everything is *plug and play*
  - a *gossip* protocol is used to learn the IDs
- Shortest paths = paths incurring the smallest aggregate cost
  - each link between switches has a (configurable) *cost*,
  - typically, the cost is a decreasing function of bit rate
- Switches compute the shortest paths using variant of *Bellman-Ford* algorithm

[Details are more nuanced when links between switches are shared medium and not point to point: see Bonaventure's textbook.]

Which of the following links are on the spanning tree ?

A. S4—S5
B. S4—S3
C. Both
D. None
E. I don't know



Go to web.speakup.info or download speakup app

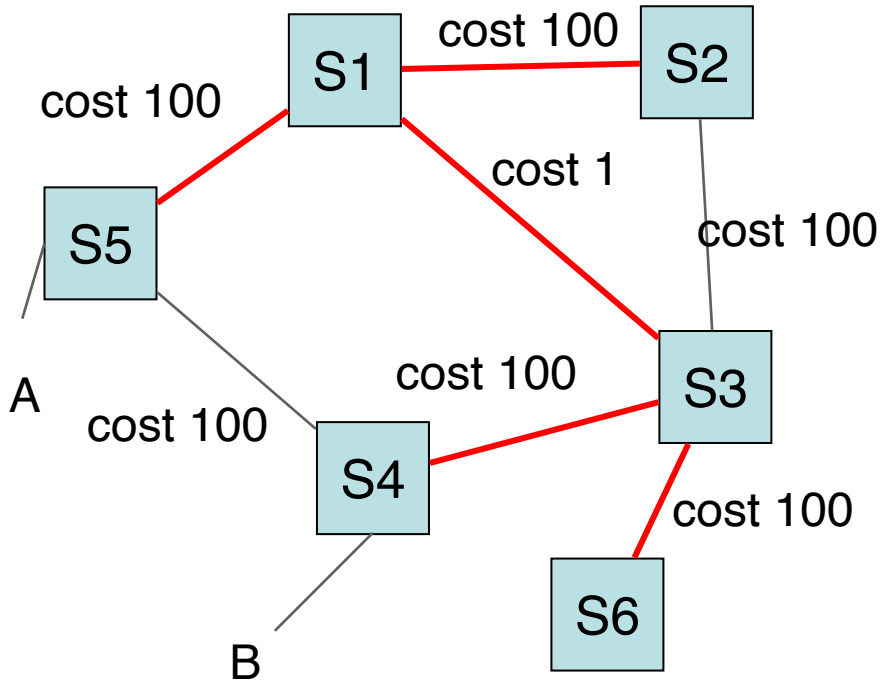
Join room 87072

Solution

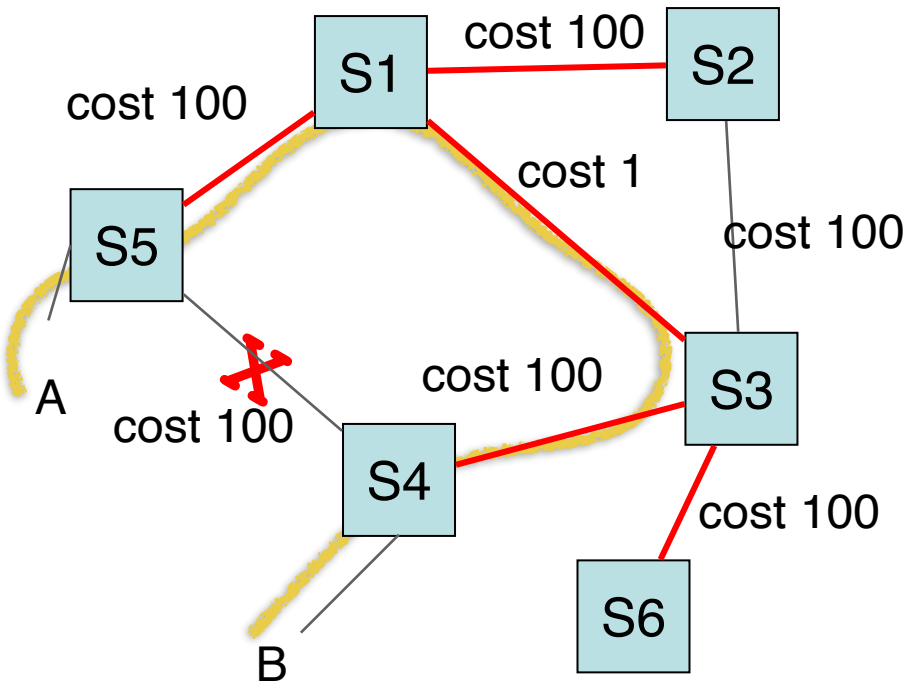
Answer B
The root is S1 (smallest label)
The spanning tree is shown in red →

The path for packets from A to B is:

S5—S1—S3—S4



A side-effect of the Spanning Tree Protocol (STP)



- All frames go through the spanning tree
- A direct link between two (non-root) switches may not be used even if it is optimal

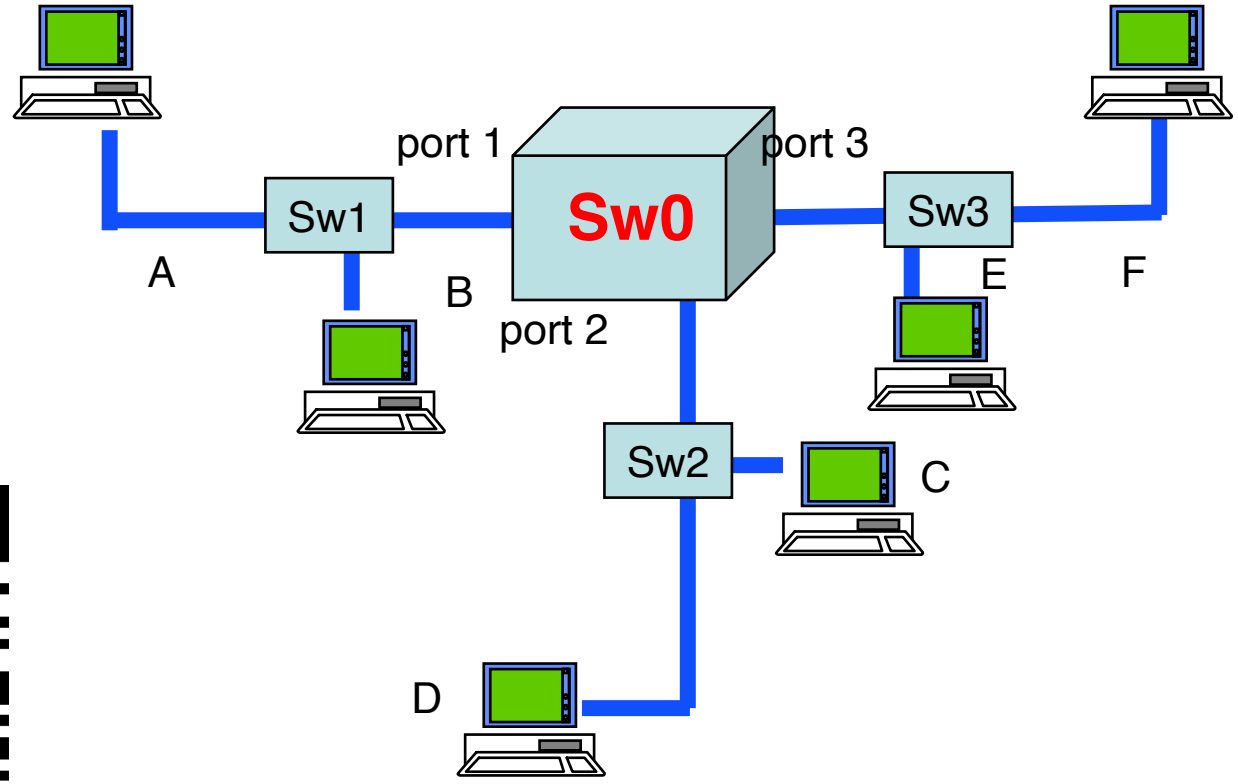
  e.g., the path from A to B, here, is not optimal

- *Less efficient* than shortest path
- some more sophisticated switches implement *Shortest Path Bridging* instead of STP, which is the default

  [see “link state” lecture]

How many frames can be transmitted in parallel in this switched Ethernet LAN (all boxes are switches)?

A. 1
B. 3
C. 6
D. > 6
E. I don't know



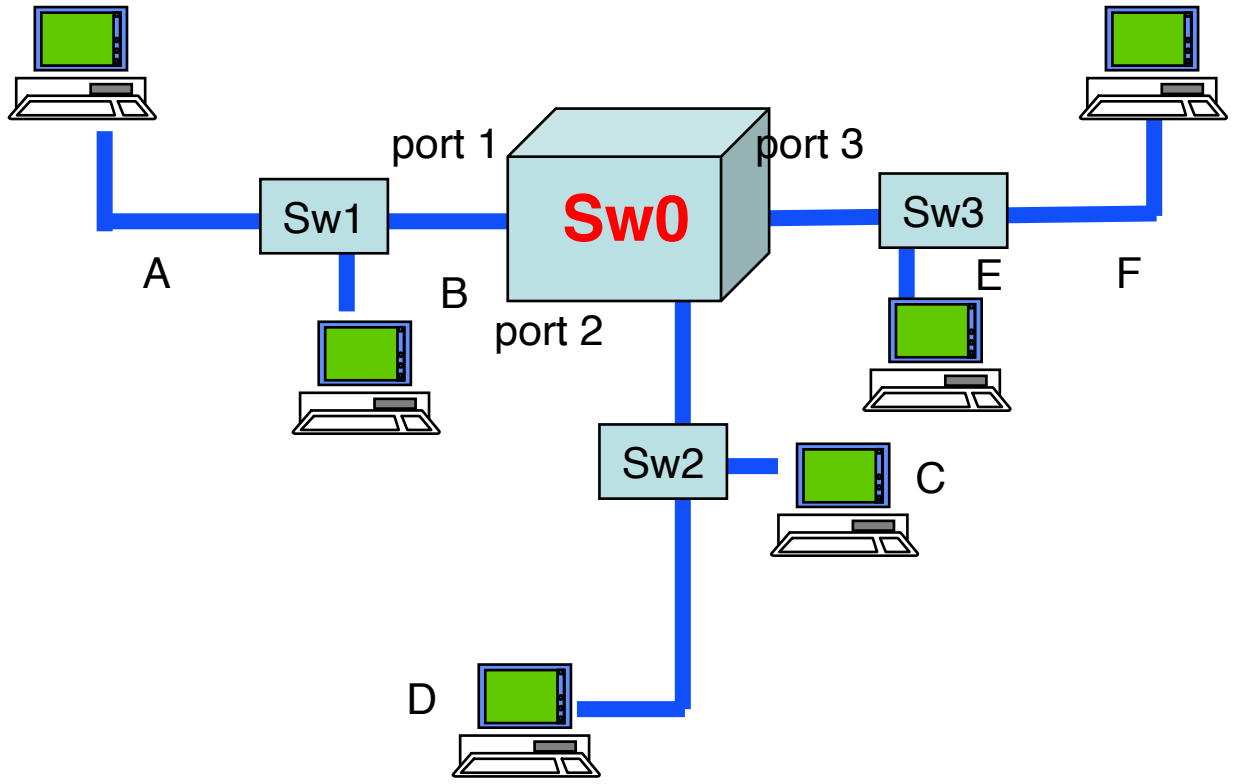
Go to web.speakup.info or download speakup app

Join room 87072

Solution

Answer D
Assume that switches can transmit on all ports in parallel (one LAN adapter per port).

Since switches are *full duplex*, there can be up to 2 frame transmissions on every link (one in each direction), therefore up to 18 transmissions in parallel.



➡ *This is the benefit of today's switches over shared-medium, old-style Ethernet*

The Spanning Tree Protocol: How

Algorhyme

I think that I shall never see
a graph more lovely than a tree.
A tree whose crucial property
is loop-free connectivity.
A tree that must be sure to span
so packets can reach every LAN.
First, the root must be selected.
By ID, it is elected.
Least-cost paths from root are traced. → using the Bellman-Ford algorithm, see "Distance Vector"
In the tree, these paths are placed.
A mesh is made by folks like me,
then bridges find a spanning tree.

Radia Perlman
(inventor of the Spanning Tree Protocol)

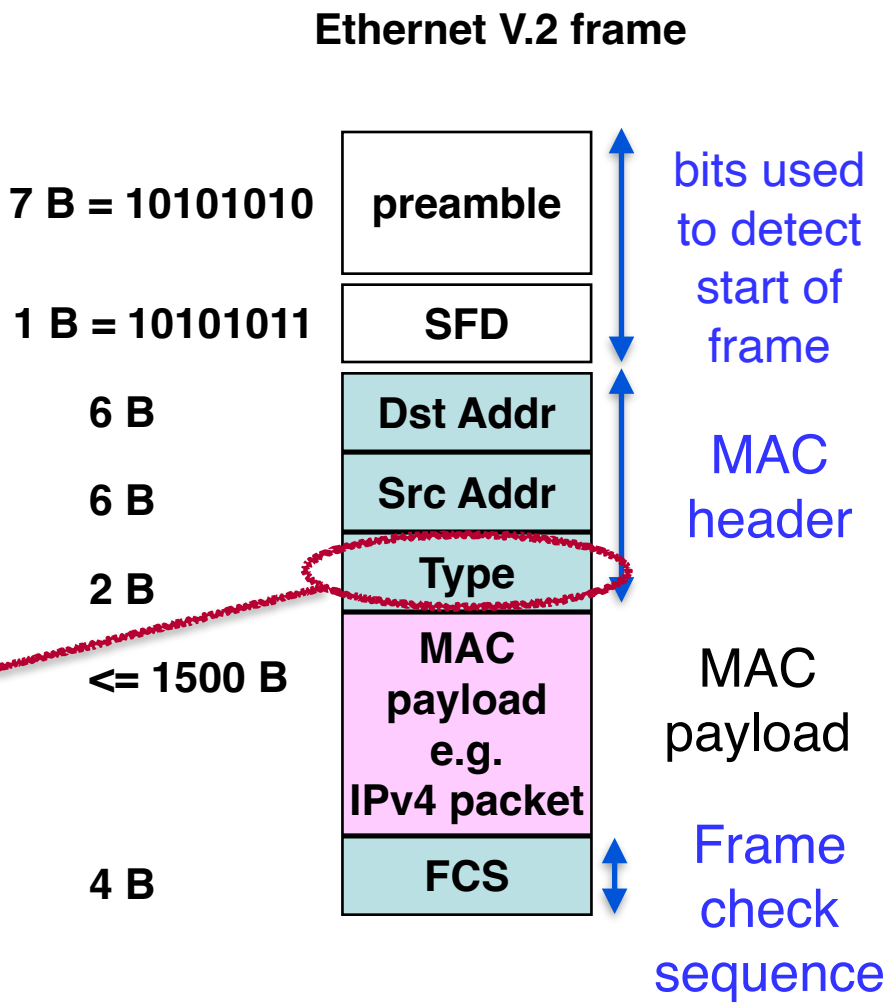
https://youtu.be/ERFohfN_H40

3. Ethernet Frame format

- Ethernet frame = Ethernet's Protocol Data Unit (PDU)
- An Ethernet frame typically carries an IP packet in its payload, sometimes also other:

Type of protocol contained in the Ethernet packet (hexa):

0800: IPv4
0806: ARP (used by IPv4)
86dd: IPv6
88e5: MACSEC (authenticated/encrypted frame)
88f7: Precision Time Protocol



Ethernet Frame format: more details

The *preamble* is used for the receivers to synchronize their clocks. It is a bit sequence of 7 bytes, where each byte is equal to: 10101010… and it is terminated by 0.
Following the preamble, an additional byte is used to indicate the beginning of the frame. This is called SFD (=start frame delimiter) and it is similar to a preamble byte, but its last two bits are 11 (instead of 10).

In Ethernet, transmission starts *asynchronously* (hosts transmit independently), and between transmissions, the channel is idle. This is why the clocks of the sender and the receiver have to get synchronized at the beginning of each frame, so that the receiver can correctly “read” the received data.

There is no frame length field in the Ethernet header—it is up to the higher layer using Ethernet to find out the length of the payload.

Frames have to be at least 64B (to see why, refer to earlier slide about minimum frame size) and at most 1500B. If needed, padding is inserted to make the frame long enough, and only in that case the Type field is used to indicate the length of the size of the payload before padding.

The format shown of the previous slide is called Ethernet v2. There exists another format on Ethernet called IEEE 802.1. With WiFi the format is similar, but the MAC header contains additional fields.

Error Detection via Frame Check Sequence (FCS)

Why ?

- The channel may cause bit errors during transmission => a frame may be "*corrupted*".
- We want to detect and discard corrupted frames.

How ?

- The sender computes a Cyclic Redundancy Checksum (CRC, 32 bits) of the frame and inserts it in the FCS field.
- The receiver recomputes the CRC checksum and checks if it is equal to the FCS.
- CRC can ***detect*** all single, double, triple errors, all error bursts of length ≤ 32, most double bursts of length ≤ 17.
    - The probability that a random collection of bit errors goes undetected is ≈2e-32.

✳ *Error recovery* in Ethernet is simple: If a frame fails the check, it is *dropped*

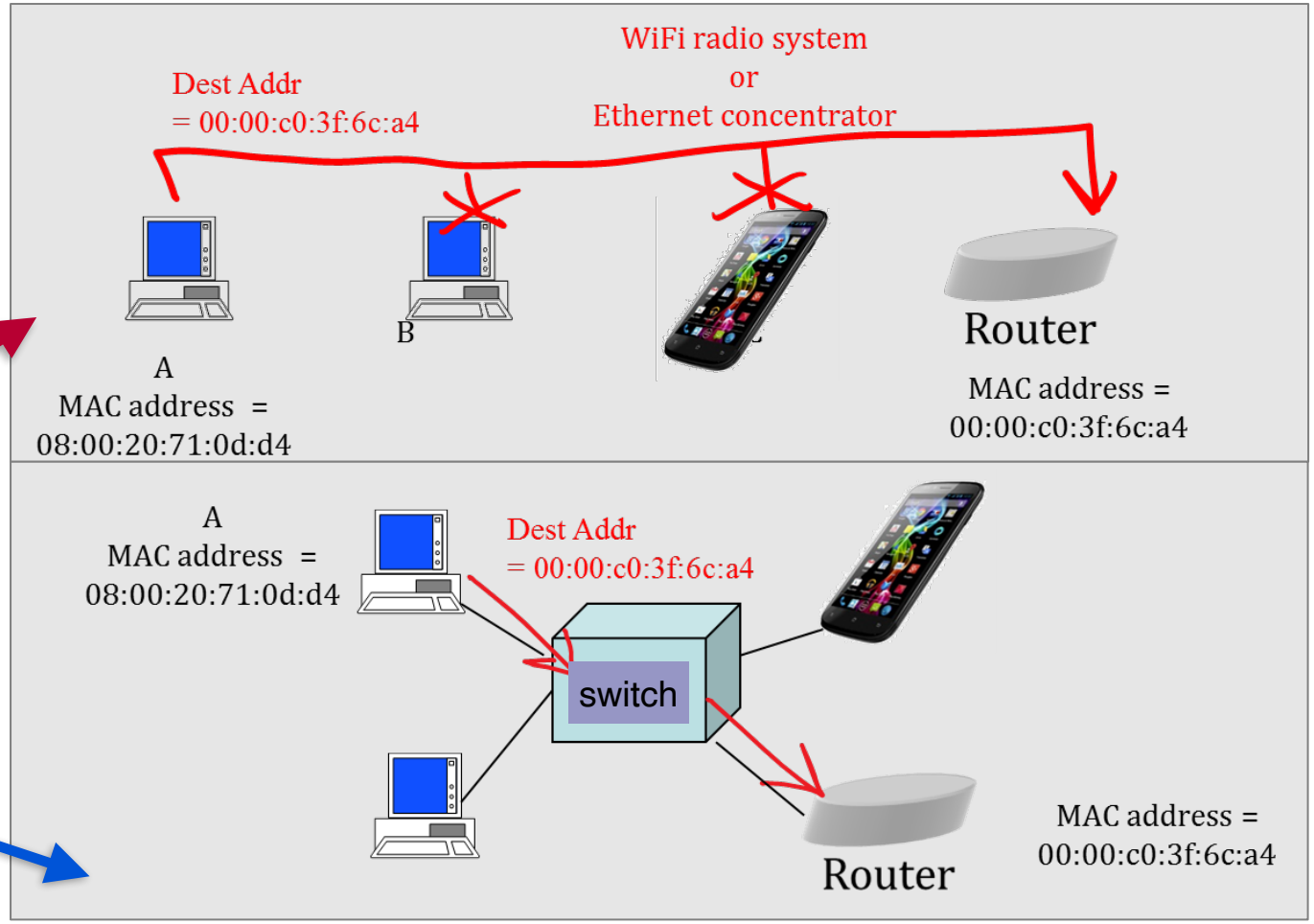
MAC Unicast

- MAC address: 48 bits = adapter ID
  Unique worldwide (in principle, but can be also configured sometimes)

How?

- Sender puts destination MAC address in the frame
- Address sent in the clear,
  no encryption

- In *shared-medium* LAN:
  all stations read *all* frames, but keep only if destination address matches
- In *switched Ethernet*:
  switches forward frames *only* to ports that need it (in principle)



Addressing: more details (for the interested students)

Ethernet addresses are known as MAC addresses. Every Ethernet interface has its own MAC address, which is in fact similar to the serial number of the adapter, set by the manufacturer.

MAC addresses are 48 bit-long. The 1st address bit is the individual/group bit, used to differentiate normal addresses from group addresses. The second bit indicates whether the address is globally administered (which is the usual case, and also called a “burnt-in MAC address”) or locally administered. Group addresses are always locally administered.

When A sends a data frame to B, A creates a MAC frame with source addr = A, dest addr = B. The frame is sent on the network and recognized by the destination.

The data is transmitted in Ethernet with the least significant bit of first octet first—this is a strange behavior dictated by Intel processors. Canonical representation thus inverts the order of bits inside a byte (the first bit of the address is the least significant bit of the first byte);

The first 24 bits are the Organization Unique Identifier (OUI); the remaining 24 bits are allocated by organization that owns the OUI.

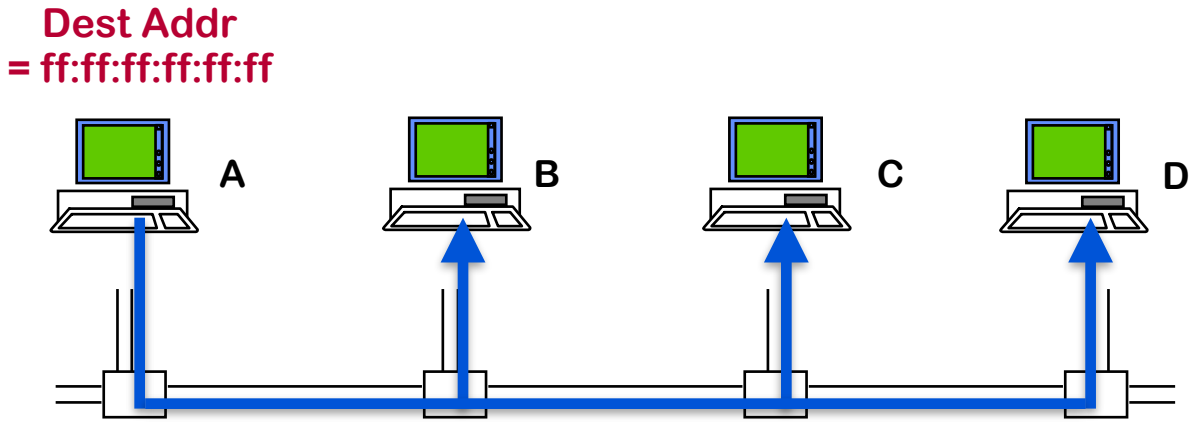
Examples of addresses:

08:00:20:71:0d:d4 (a SUN machine)
00:00:c0:3f:6c:a4 (a PC)
00:00:0c:02:78:36 (a CISCO router)
ff:ff:ff:ff:ff:ff the broadcast address

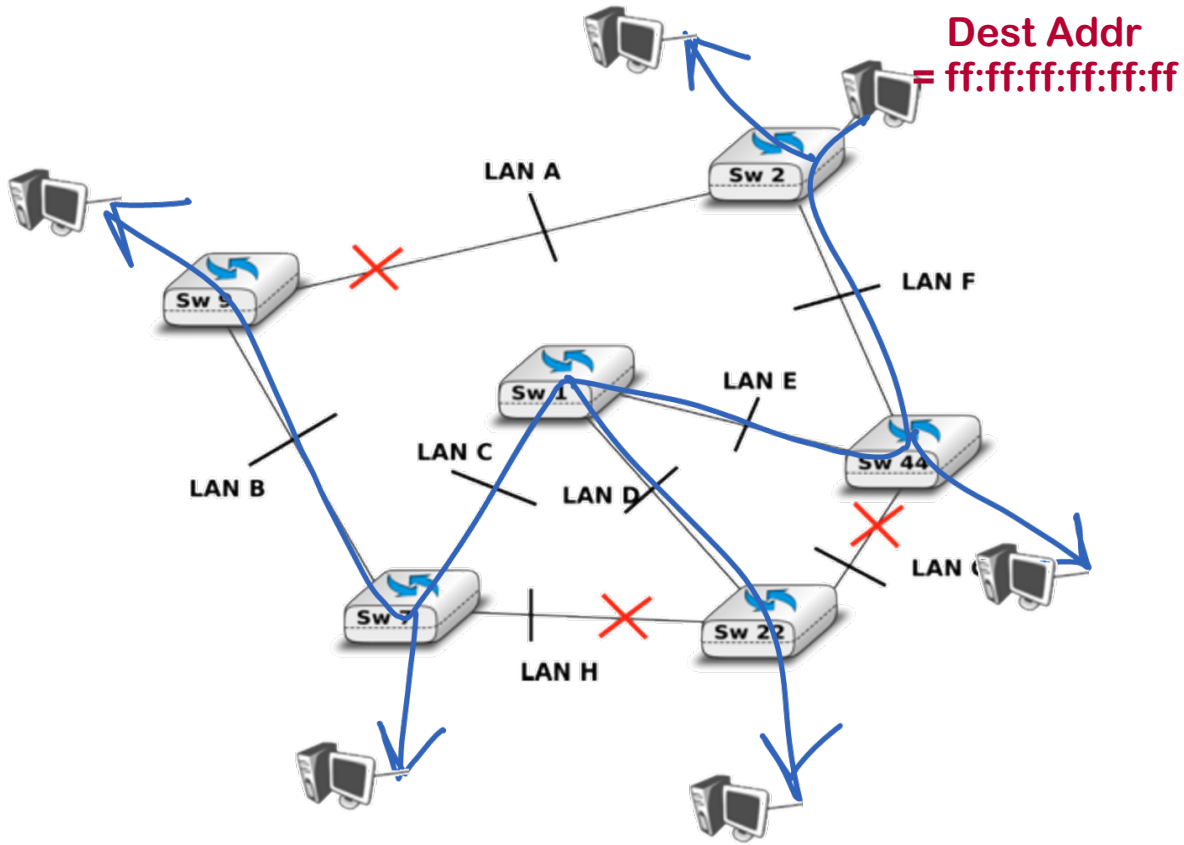
MAC Broadcast

- ff:ff:ff:ff:ff:ff = broadcast address

- In *shared-medium* LAN:
  all machines receive packet
  and do *not* discard it



- In *switched Ethernet*:
  broadcast frames are sent
  on all nodes and ports
  on the spanning tree

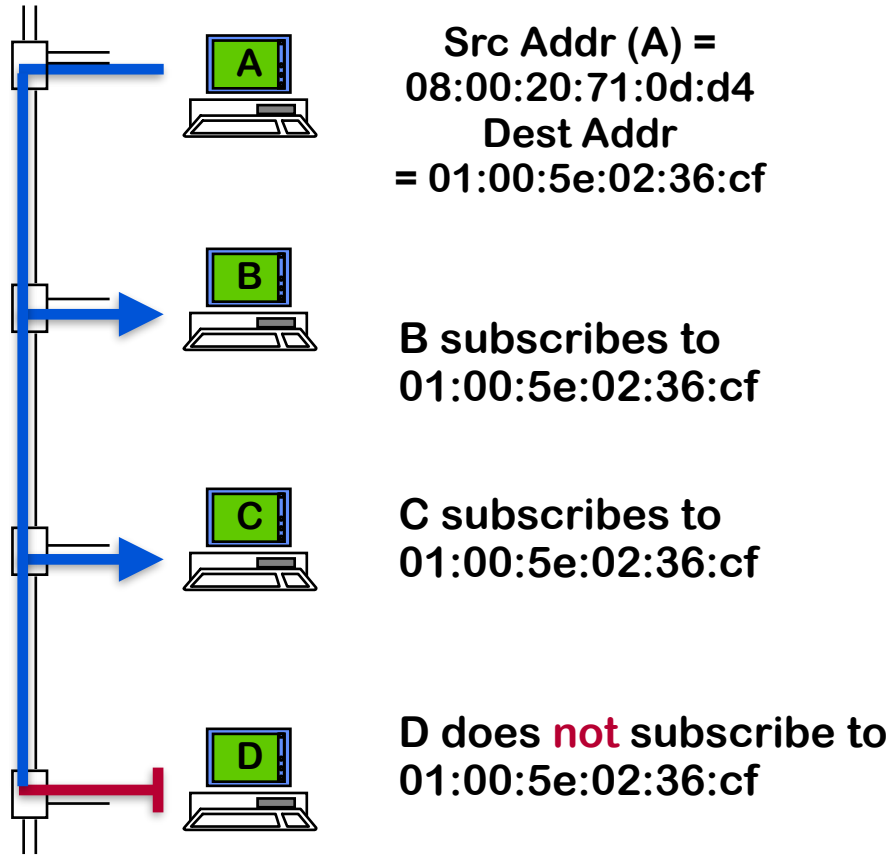


MAC Multicast (= sending to multiple hosts)

- uses:
  - MAC Addresses with 8th bit == 1 are multicast addresses
  - various multicast addresses to support various protocols (e.g. IPv4, IPv6, PTP)

- makes sense only in *switched Ethernet*:
  - Ethernet adapter discards multicast packets, unless host subscribes to address
  - non-smart switches broadcast such frames
  - smart switches send only to relevant nodes (the ones that have subscribed to the multicast address using IPv4 or IPv6 subscription protocols—see “multicast” lecture

01-00-5e-XX-XX-XX	IPv4 multicast
33-33-XX-XX-XX-XX	IPv6 multicast
01-80-c2-00-01-81	Precision Time Protocol



The protocol type in an Ethernet frame header indicates whether…

A. … the data is a TCP packet or not
B. … the data is an IPv4 packet or not
C. Both
D. None
E. I don't know

Go to web.speakup.info or download speakup app

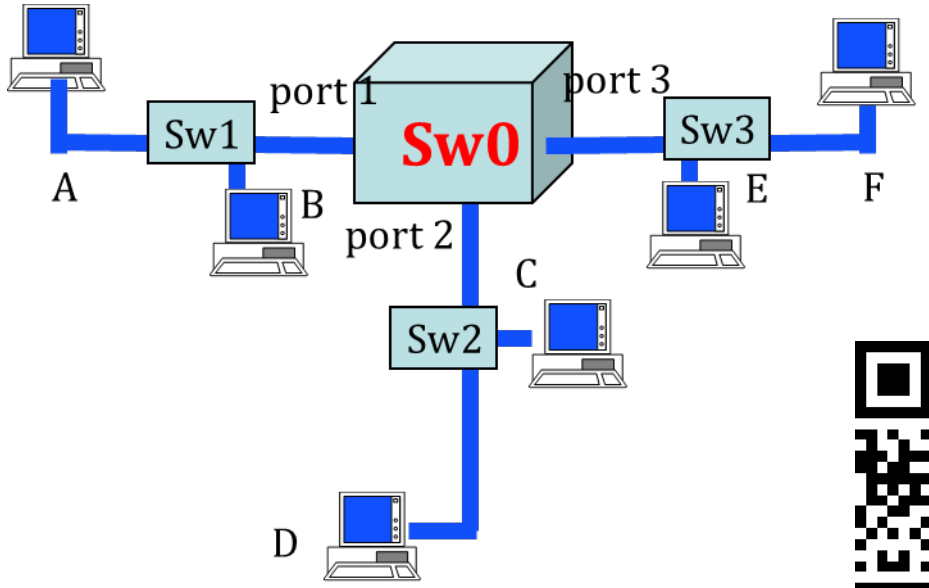
Join room 87072

Solution

Answer B: the Ethertype indicates an IPv4 or IPv6 packet

We have a working environment with IPv4, using Ethernet switches plus other things. We add IPv6 to the network. Do we need to change the switch configurations ?

A. No, since switches look only at MAC addresses
B. Yes since the protocol type is different
C. It depends whether SLAAC is used
D. I don't know



Go to web.speakup.info or download speakup app

Join room 87072

Solution

Answer A: the switches work only at MAC layer and continue to work the same whether IPv4, IPv6 (or something else) is used.

One *exception*: “smart” switches are cheating—they build multicast trees by looking at IP layer group membership (see multicast lecture later)

How do you find the *content* of the lectures so far?

A. Not interesting at all
(I know all this stuff from my undergrad)

B. Somewhat interesting
(There are few things I missed from my undergrad and I am happy to learn them now)

C. Interesting
(I thought I knew much about networks, but I now realize that there are a lot more things to learn)

D. Other

How do you find the *way* of lecturing so far?

A. **Very slow** pace, I am bored.

B. **Rather slow**, many times I feel that time is spent on useless stuff. I want a faster pace.

C. **Engaging**, it helps me catch up with all concepts.

D. **Very fast** pace, I do not understand critical ideas.

E. Other

How do you find the *content* of the labs so far?

A. Not interesting at all.
(I know all this stuff from my undergrad)

B. Somewhat interesting.
(There are few things I missed from my undergrad and I am happy to have a hands-on experience about them now)

C. Interesting.
(I had limited hands-on experience with networks and I am happy to dive into these details)

D. Other

How do you find the *labs* so far?

A. **Completely useless**, I am losing my time

B. **Somewhat useful**, they help me catch up with things I already knew

C. **Useful**, they help me understand in detail what is taught in the lectures and all new things I learn

D. Other