Networks and Systems Security

Dr Sudipta Saha

https://www.iitbbs.ac.in/profile.php/sudipta/

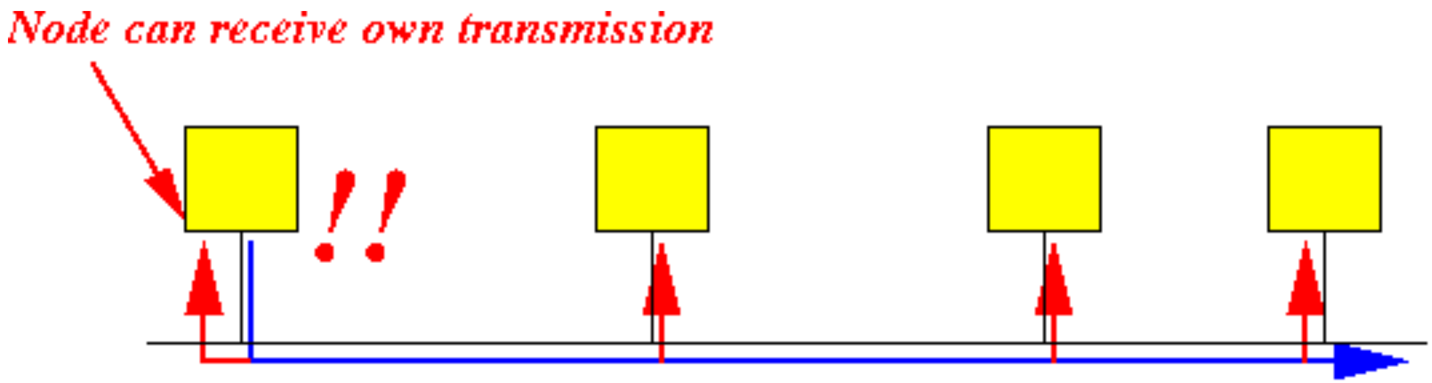
Decentralized and Smart Systems Research Group (DSSRG)

https://sites.google.com/iitbbs.ac.in/dssrg

Computer Science & Engineering, School of Electrical Sciences
Indian Institute of Technology Bhubaneswar

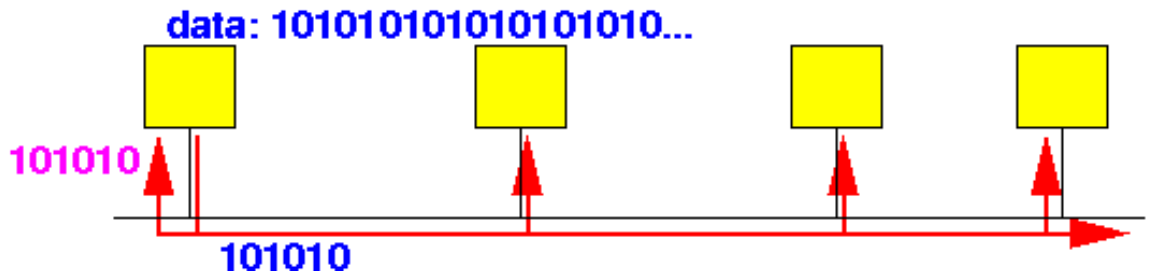
Intro to *Collision Detection*

A **transmitting node** ***can*** receive its ***own*** **transmission**:



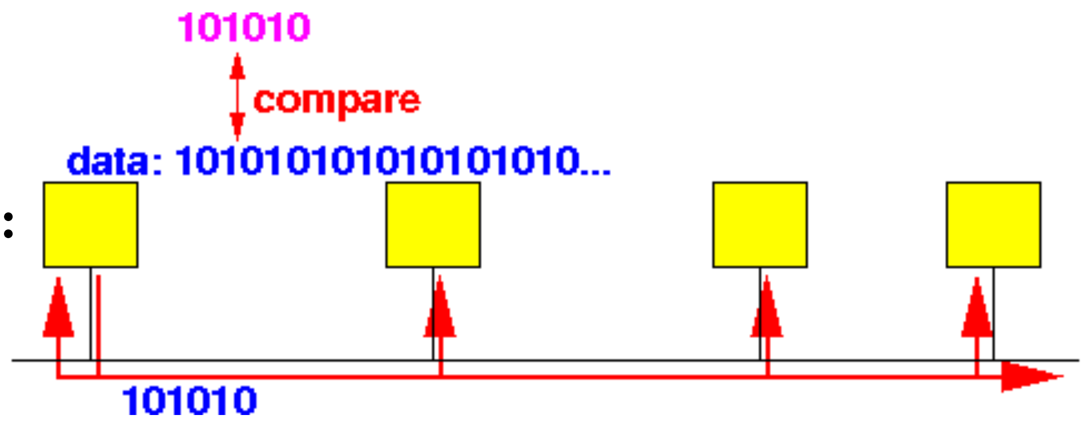
How to detect a **collision:**

- While **transmitting**,
  the **sender receives** its *own* **transmission**:



- *Compare* the **data received** against the **data transmitted**:

Collision **will be detected through: Bit *errors***

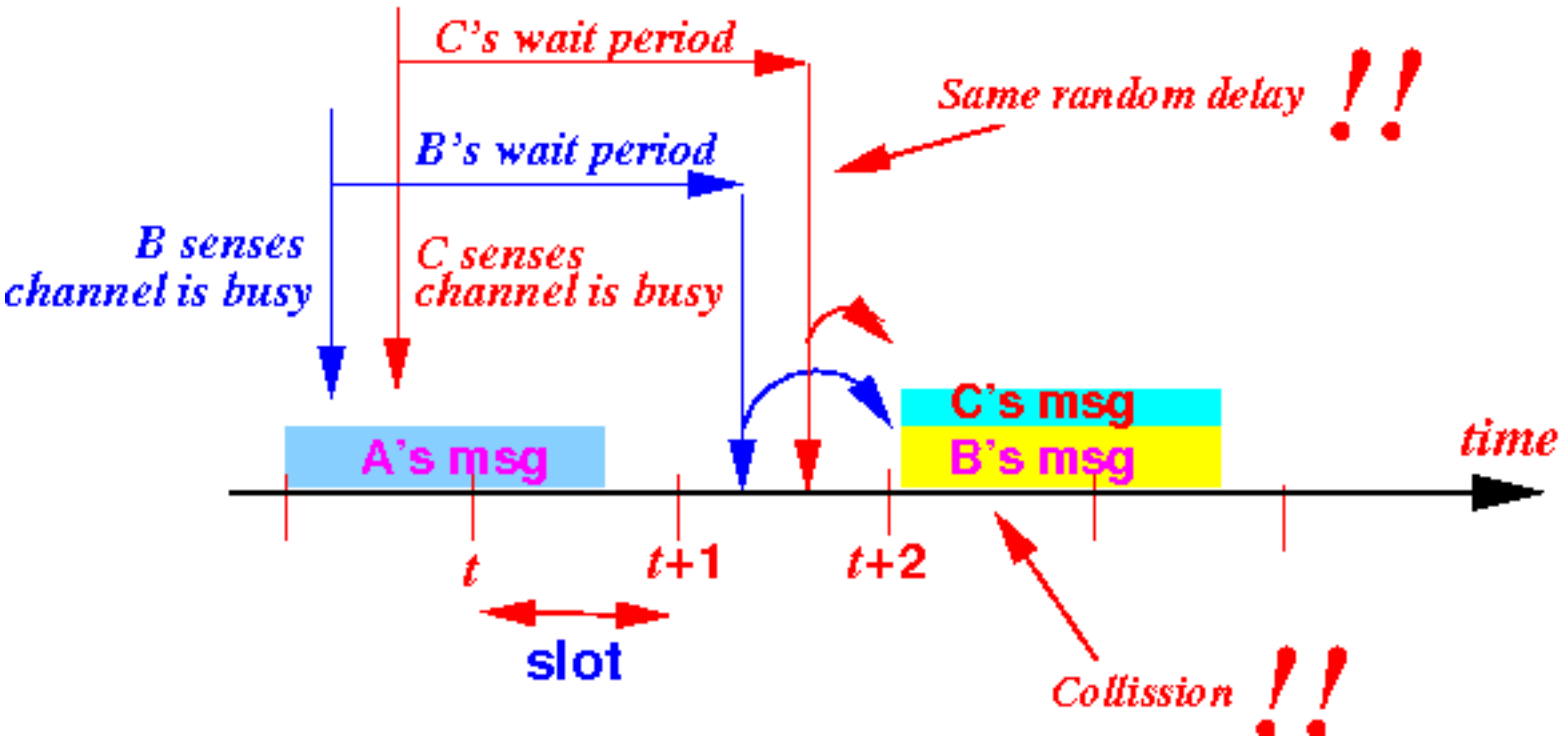


Collisions in the CSMA protocols

- ***Every* variant** of the **CSMA protocols** can have **collisions**

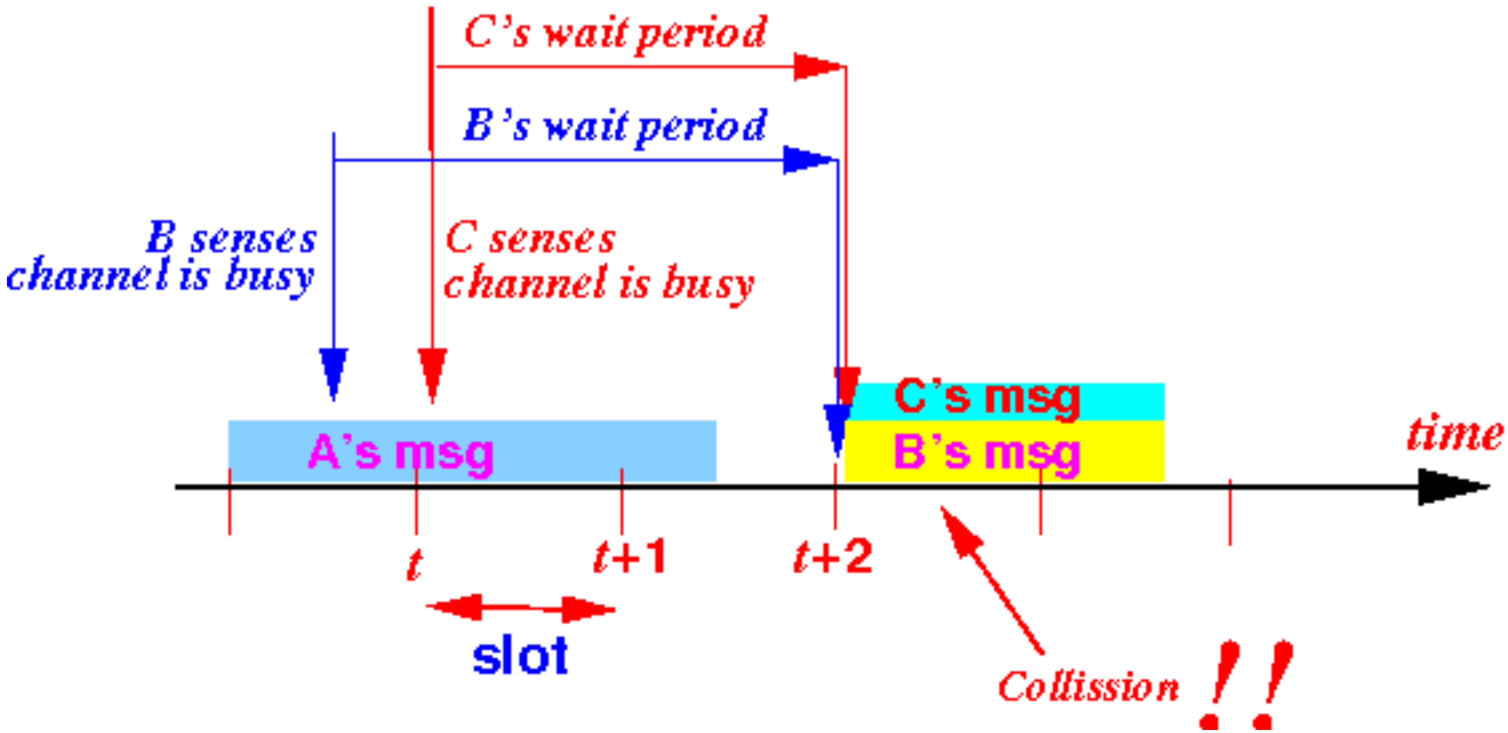
Collision in Non-Persistent CSMA

- In ***non*-persistent CSMA**, a **collision** happens when **2 nodes** pick the *same* **random number**:



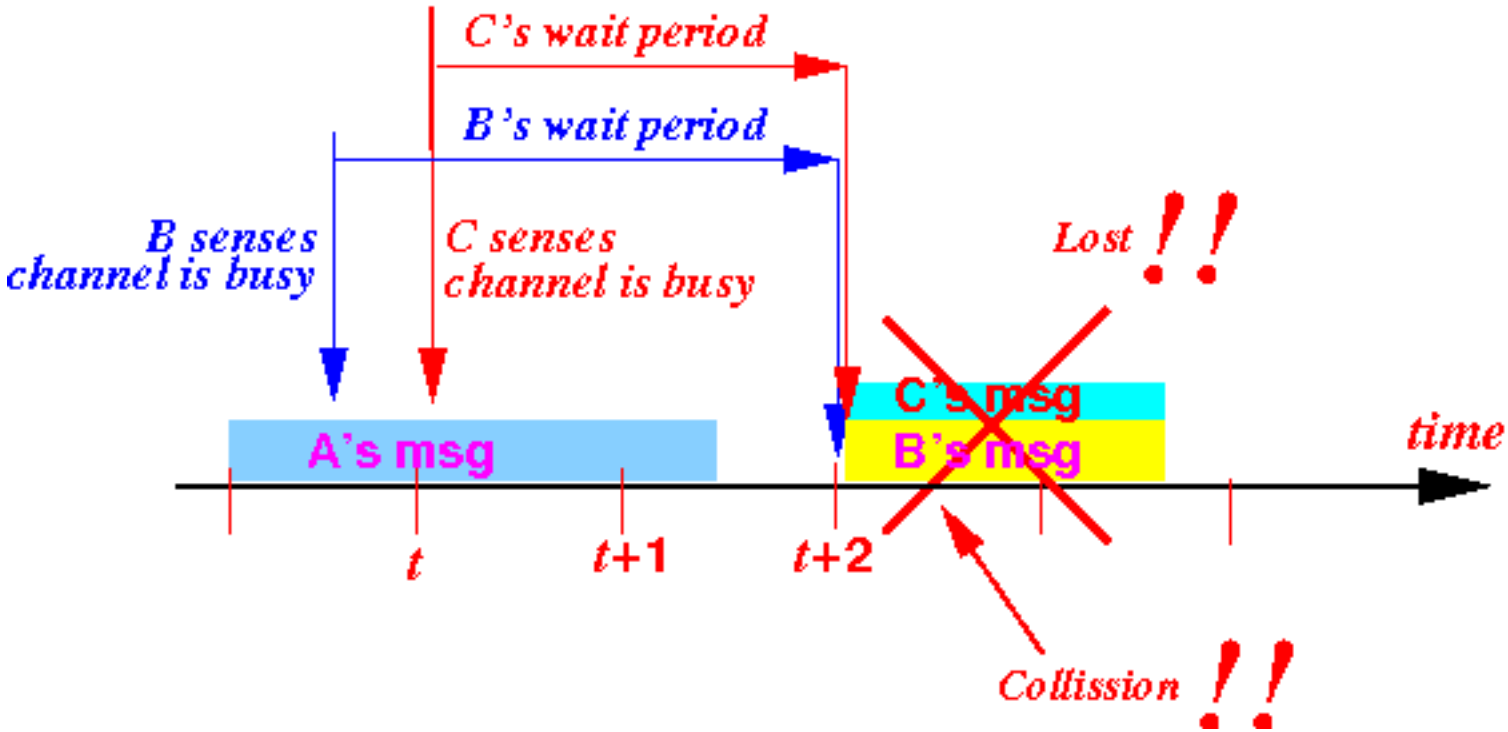
Collision in 1-Persistent CSMA

- In *1*-**persistent CSMA**, a **collision** happens when **2 nodes starts sensing** during the ***same*** **transmission**:
-

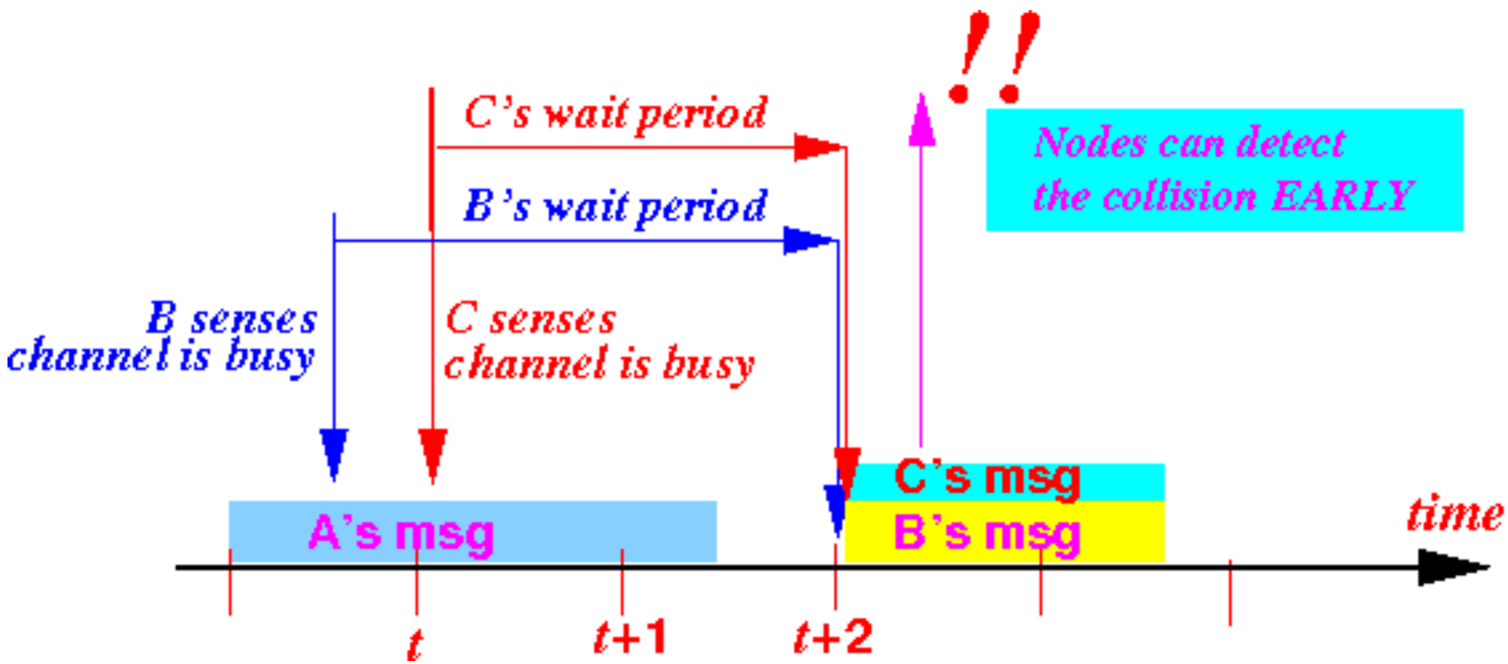


Result of **collisions**:

- The **transmission (frames)** involved in a **collision** are *lost*:
-

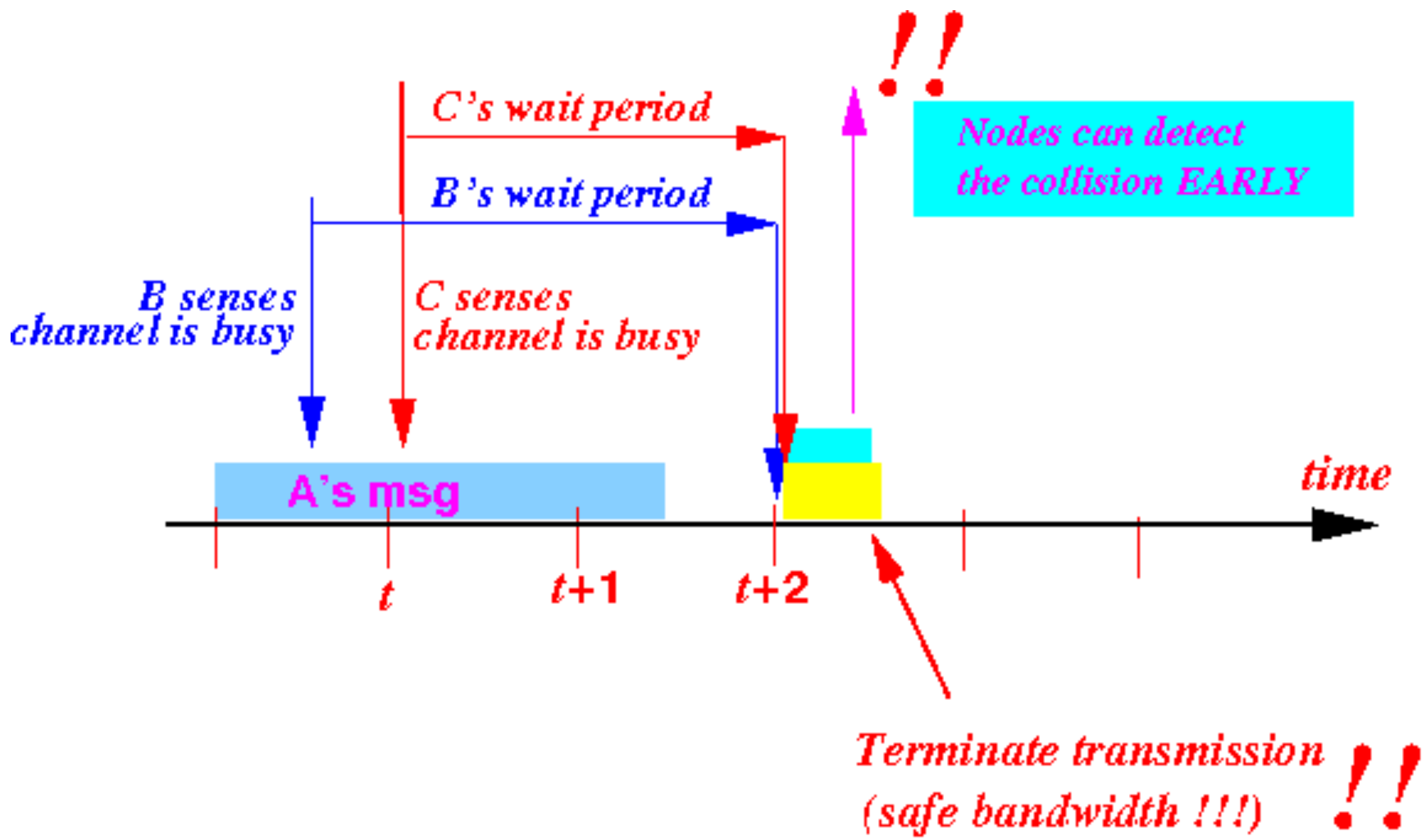


How CD helps (Collision Detection)

- ***Improving* the CSMA protocol with *Collision Detection***
- The **nodes** can **detect** a **collision** very ***early***:
- 

How CD helps

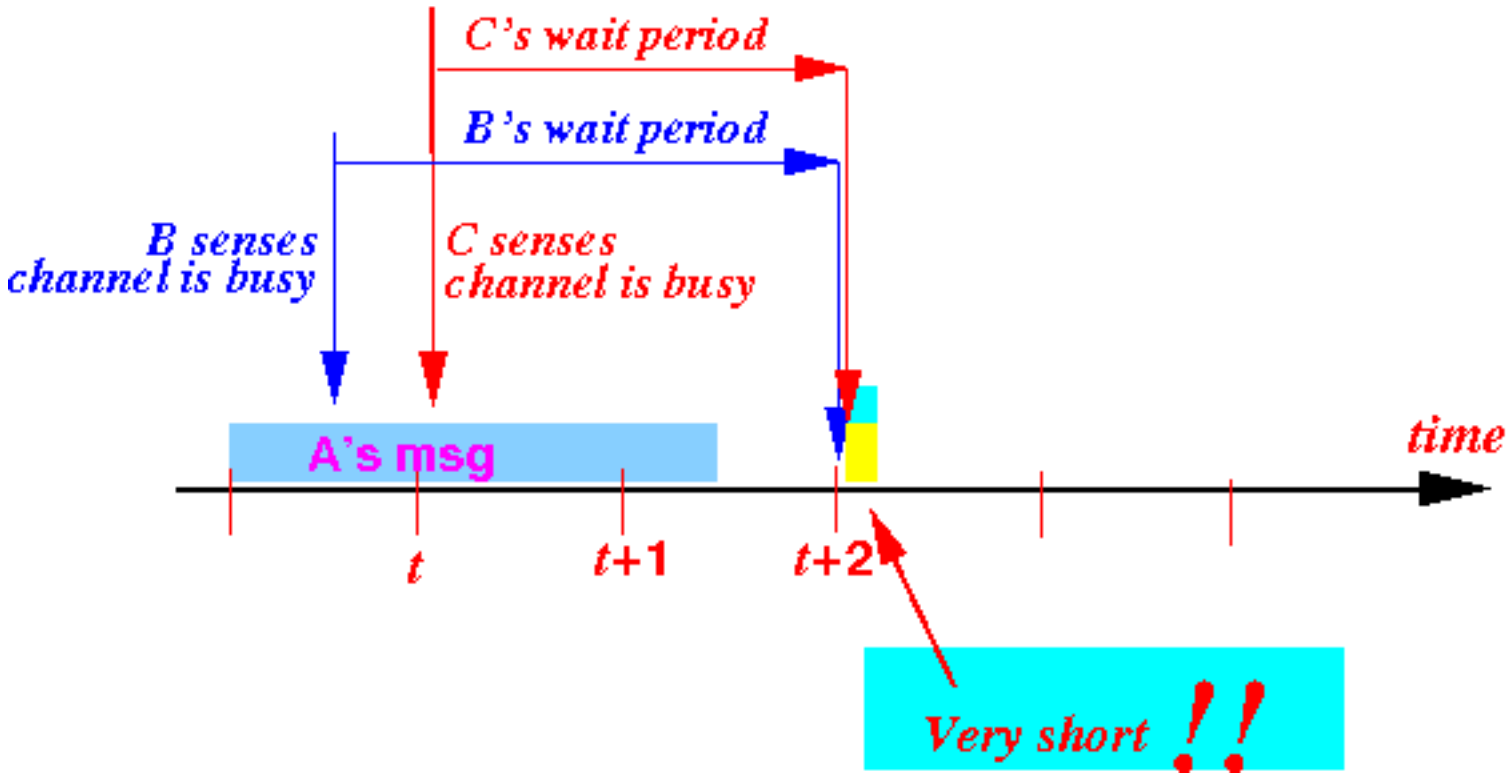
- The **nodes** can ***terminate*** the **transmission prematurely**:



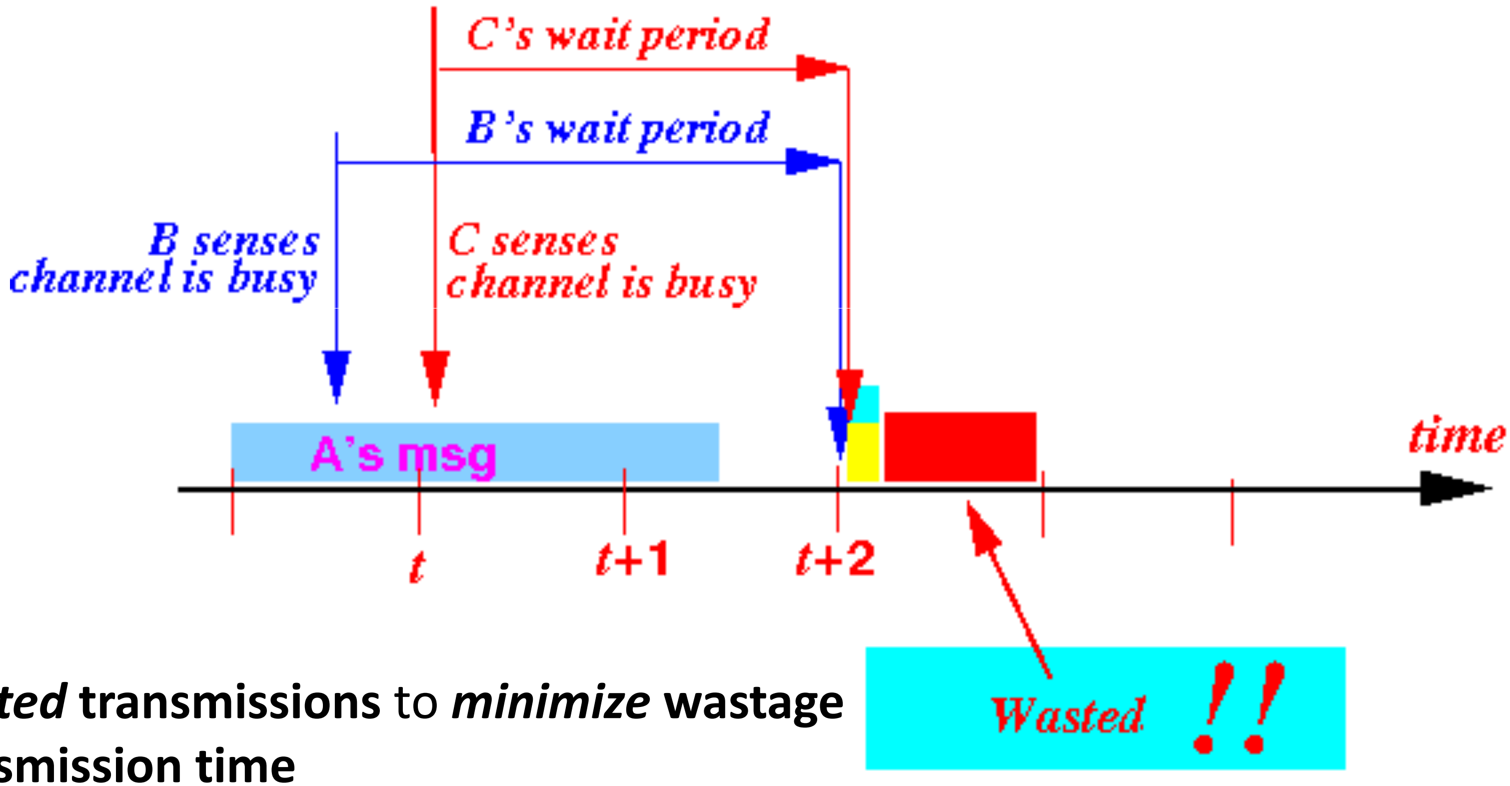
- **Result:**
- ***Less* transmission time** are ***wasted*** in a ***useless*** **collision**
-

Why CSMA/CD uses unslotted transmissions

- The *collision* **period** is *very* **short** compared to a **slot time**:
-

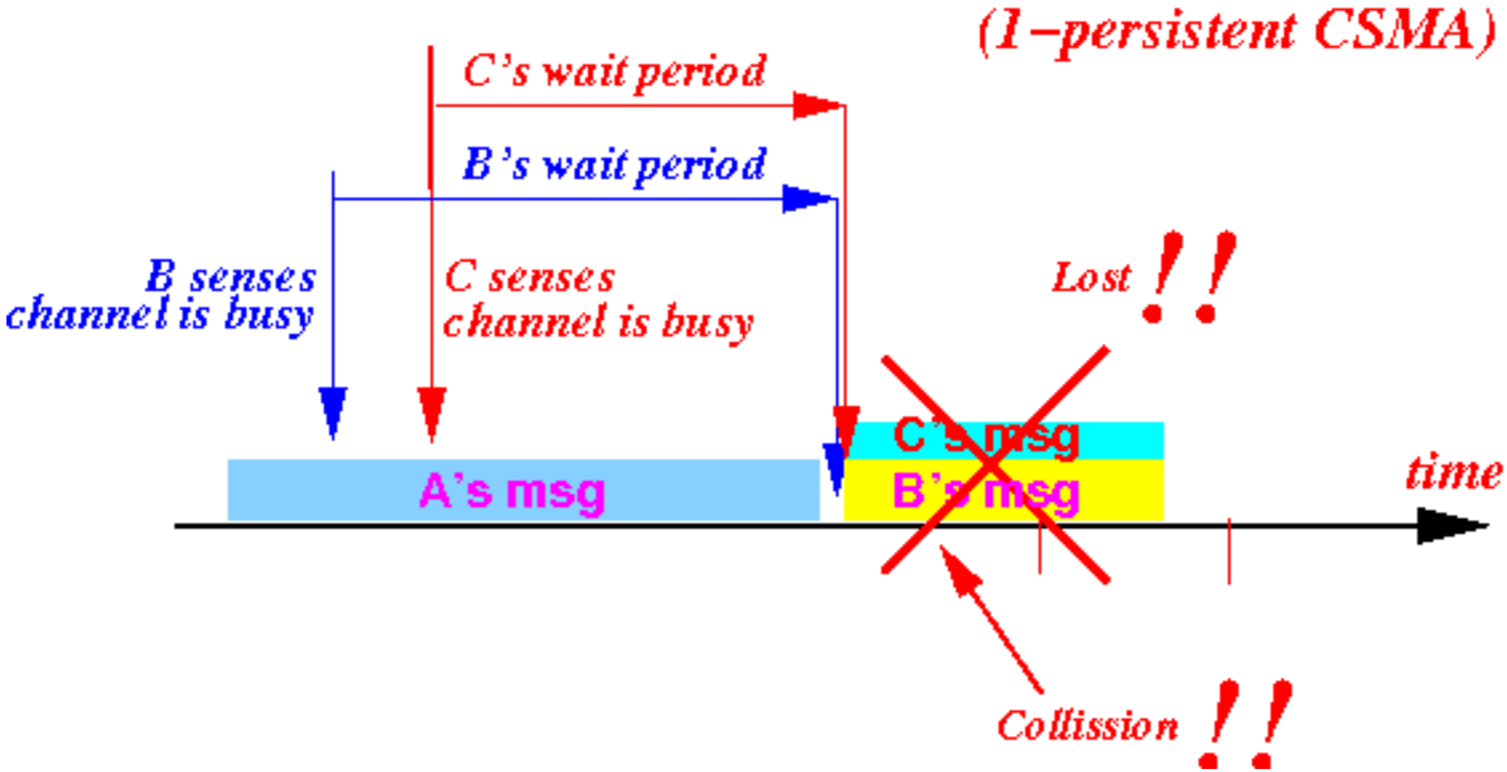


Why CSMA/CD uses unslotted transmissions

- *Slotted* **transmissions** will ***waste*** the ***remainder*** of the **slot**:
- 

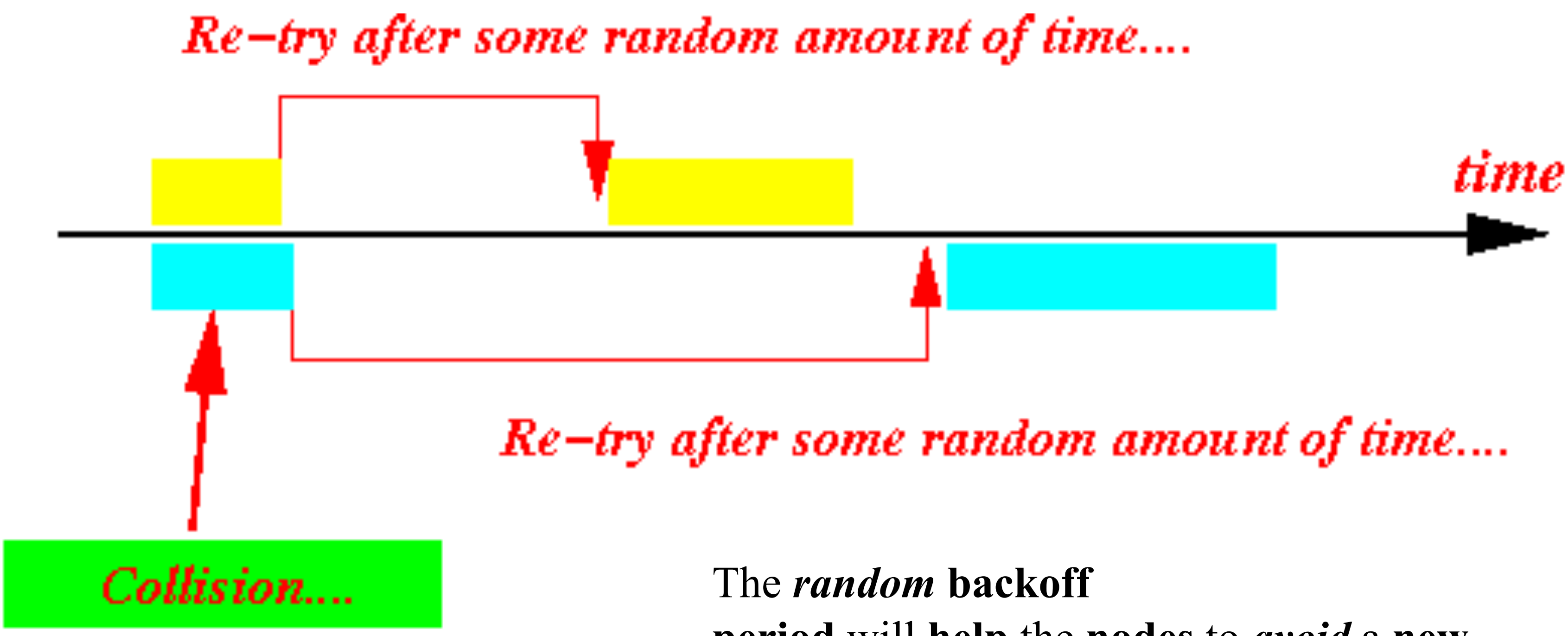
•*Unslotted* **transmissions** to *minimize* **wastage** of **transmission time**

What happens *after* a node detects a *collision* ?

- *Action taken* after a positive collision detection --- the CSMA/CD protocol
- Colliding frames get lost …
- 

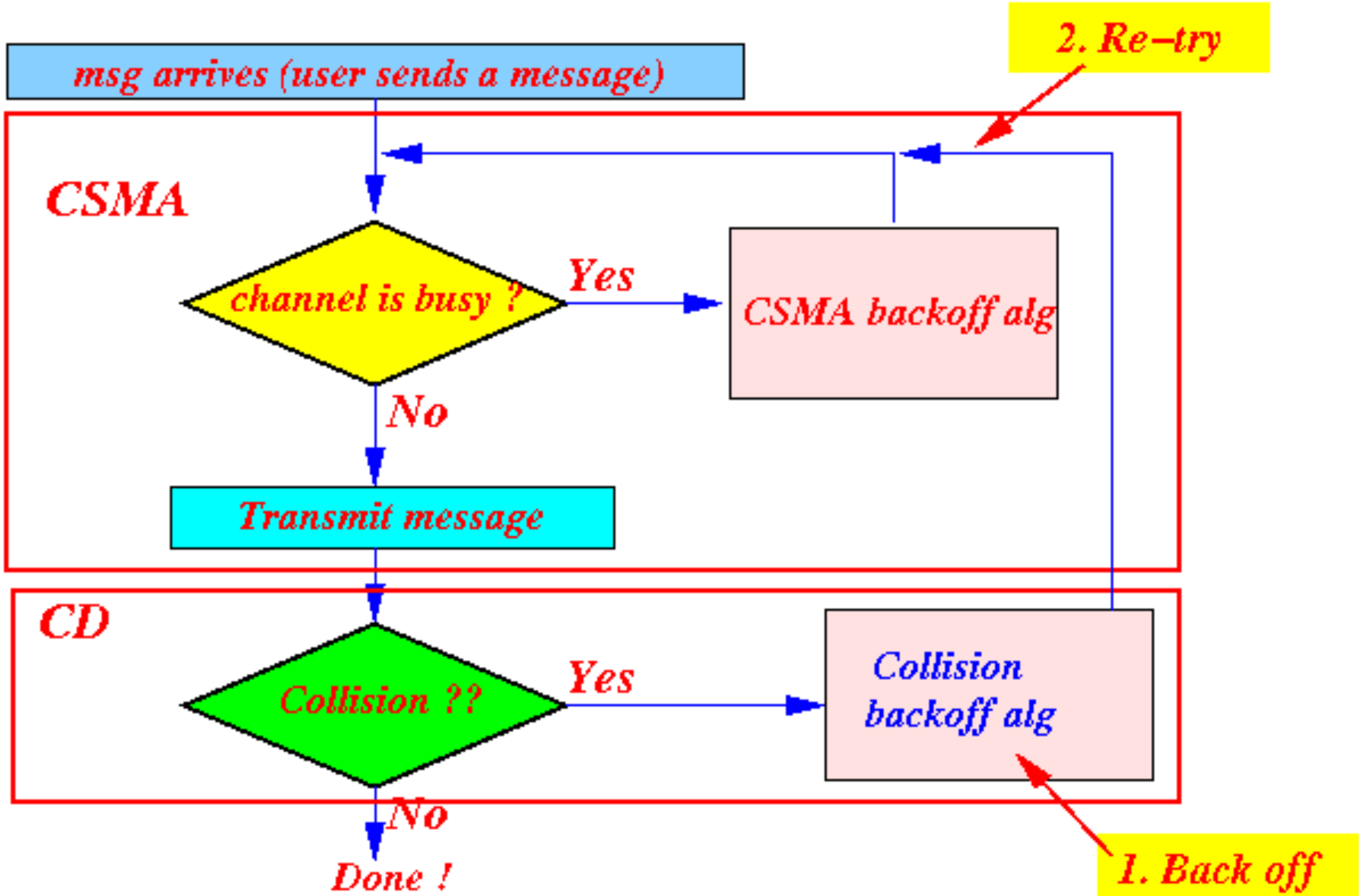
What to **do** when a **node** detects a **collision**:

- ***Back off*** (= **wait**) for a **random period**
- **Re-try** again

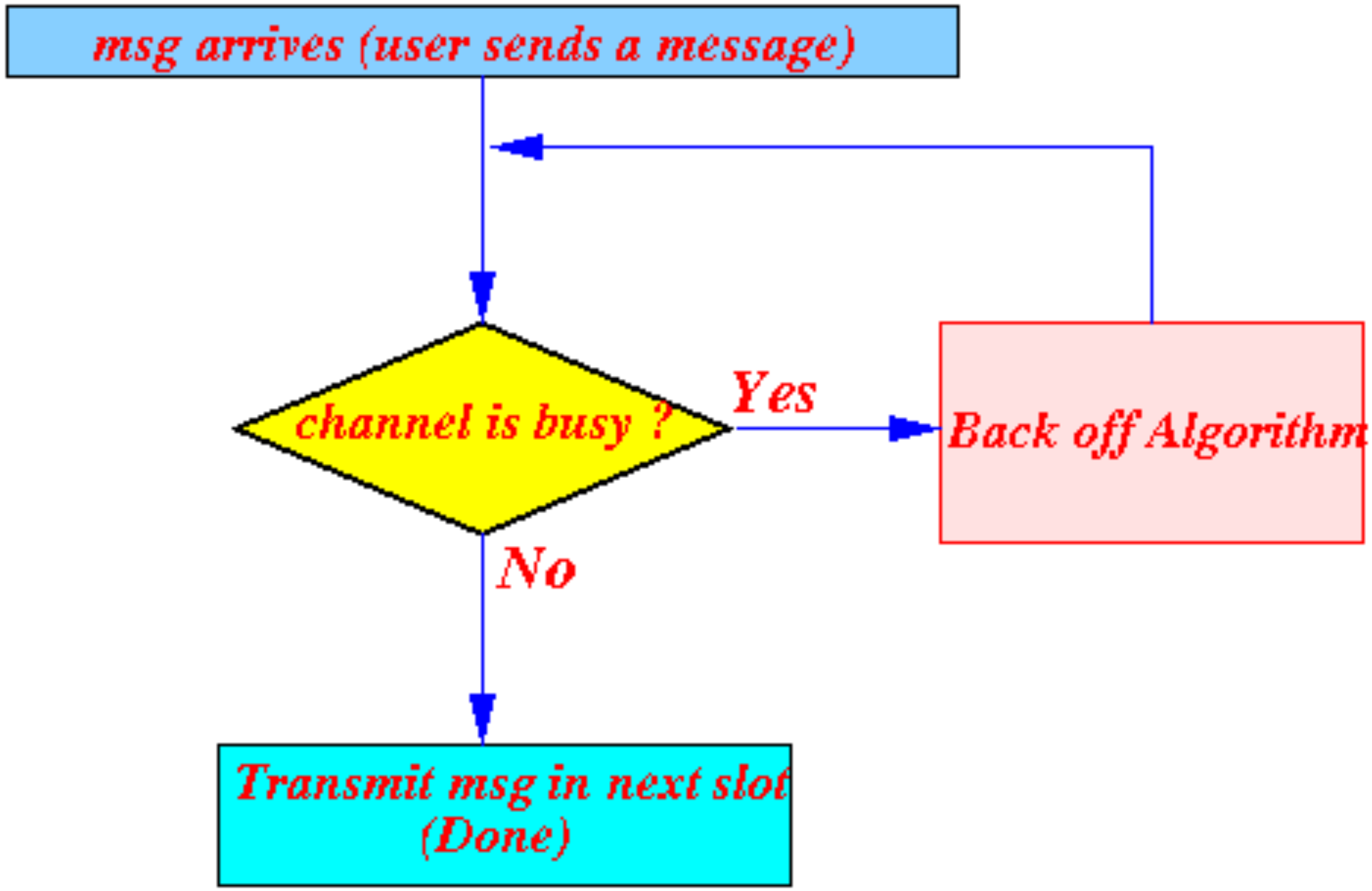


The ***random*** **backoff period** will **help** the **nodes** to ***avoid*** a **new collision** when they **attempt** to **re-transmit** their **frame**

The *CSMA* protocol with collision detection (CD)



Compare that to the **flow chart** for **CSMA**:



Two issues to address now

1. What is the unit of the back-off time?

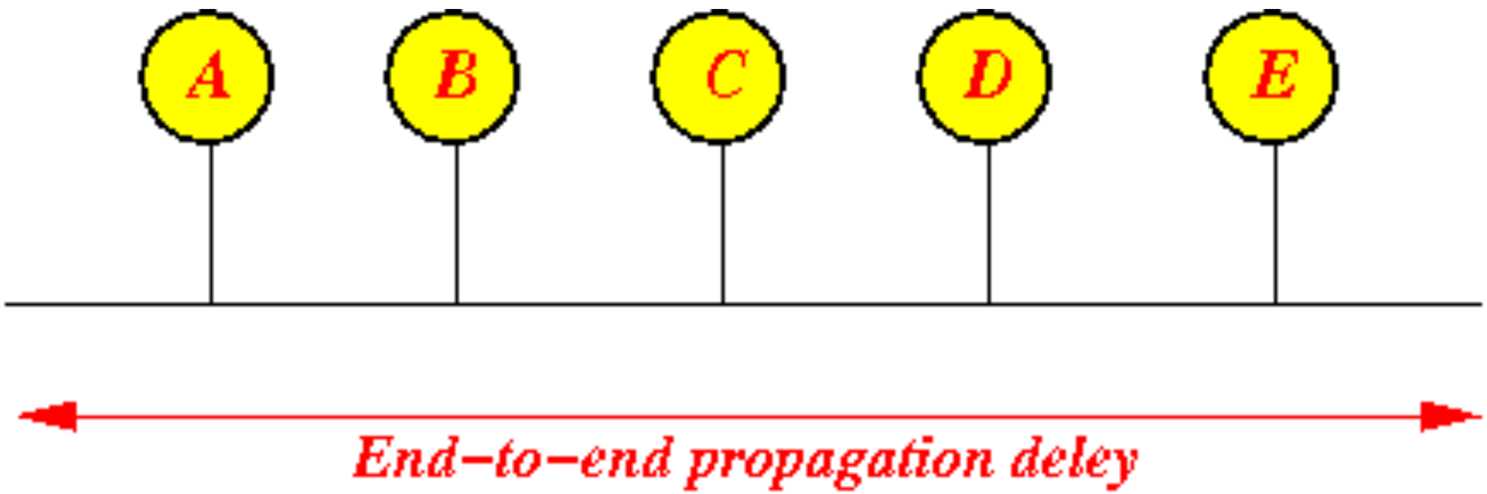
[The answer you know already]

2. How long a node needs to sense to detect a collision has happened or not?

- **We discuss these two issues next …**

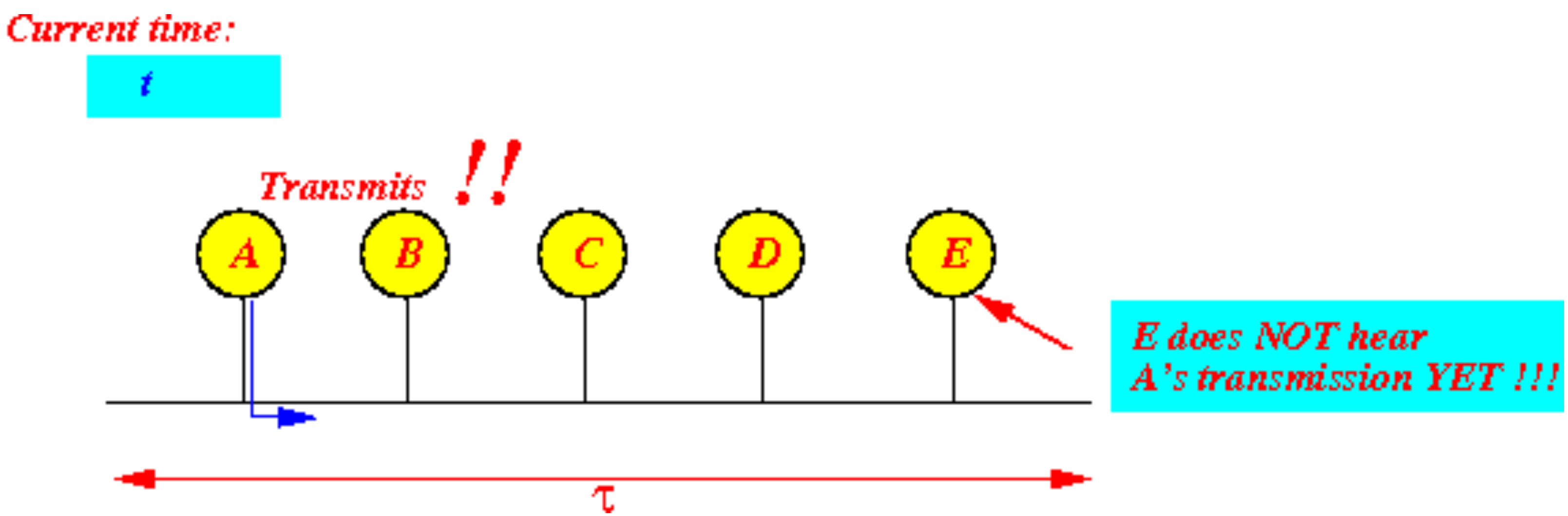
The time unit used in the Back Off Algorithm in CSMA/CD

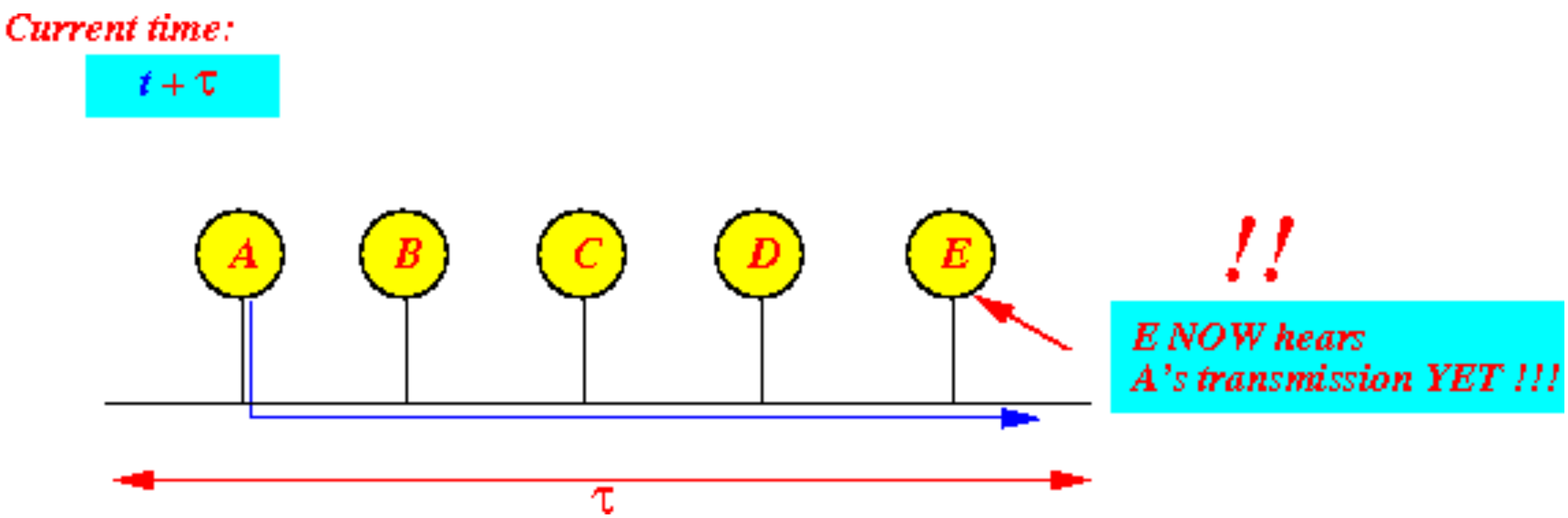
- **Recall: End-to-end propagation delay**
- **End-to-end propagation delay** = **amount of time** that it takes for the **electric signals** to **travel** from **one-end** of the **network** to the ***other*** **end** of the **network**:
-



Notation: τ = end-to-end propagation delay

- **Property** of the **end-to-end propagation delay (τ)**
- ***All* nodes** on the **LAN** will **hear** a **node's transmission *after* τ sec** after the **start** of the **transmission**
- A **(any) node** on the **LAN *starts* transmitting** at **time *t***:
-



- At **time (*t* + τ) sec**, *every* **node** on the **LAN** will **hear (= detect)** the **transmission**:
- 

Back off time period:

- **Back off time duration** = $r \times \tau$
- (r = a **random number**)
- We **choose τ** to be the **"time unit"** of the **back off algorithm**
- We see the **reason why next**...
-

The generic Back Off Algorithm in CSMA/CD

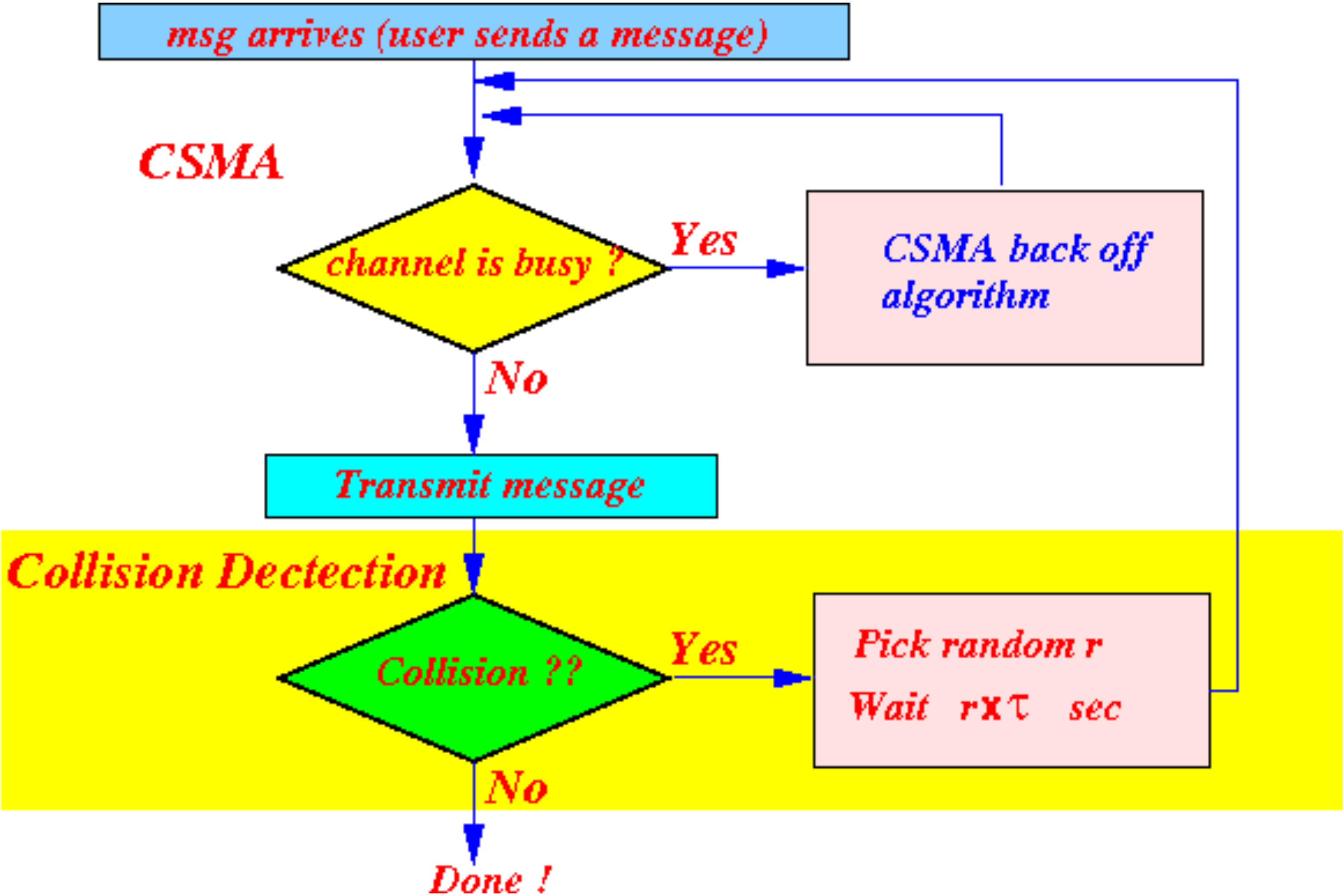
- Pick a random integer r
- Wait r × τ seconds
- (Start the transmission (CSMA/CD) protocol from the start)

- If **2 different nodes pick** ***different*** **random integers**:

- The **back off periods** used by these **nodes** will **ensure (guarantee)** the ***later*** **starting node** will **hear (= detect !)** the **transmission** of the ***earlier*** **starting node** when the ***later*** **starting node re-tries** its **transmission** !!!

- This will **allow** the ***later*** **starting node avoid colliding** with the ***earlier*** **starting node** !!!

Updated (improved) flow chart for CSMA/CD:



How long a node needs to sense for collision?

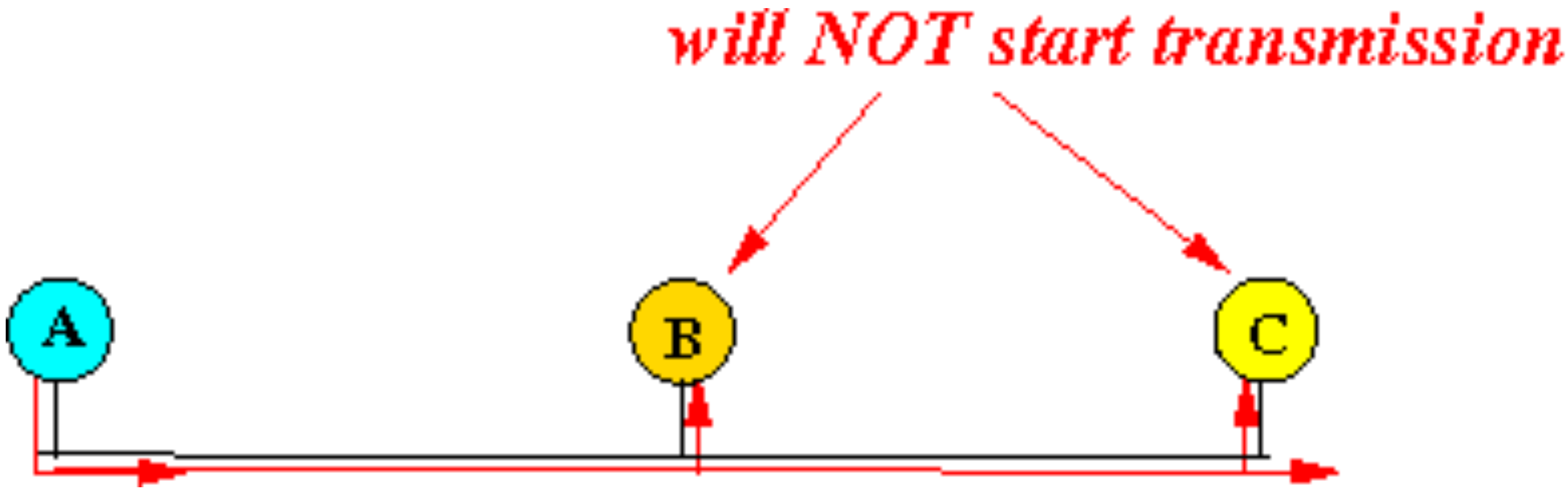
- **A node does not need to perform collision detection forever ....**
- **A node can stop sensing for collision after 2 × τ time**

Claim

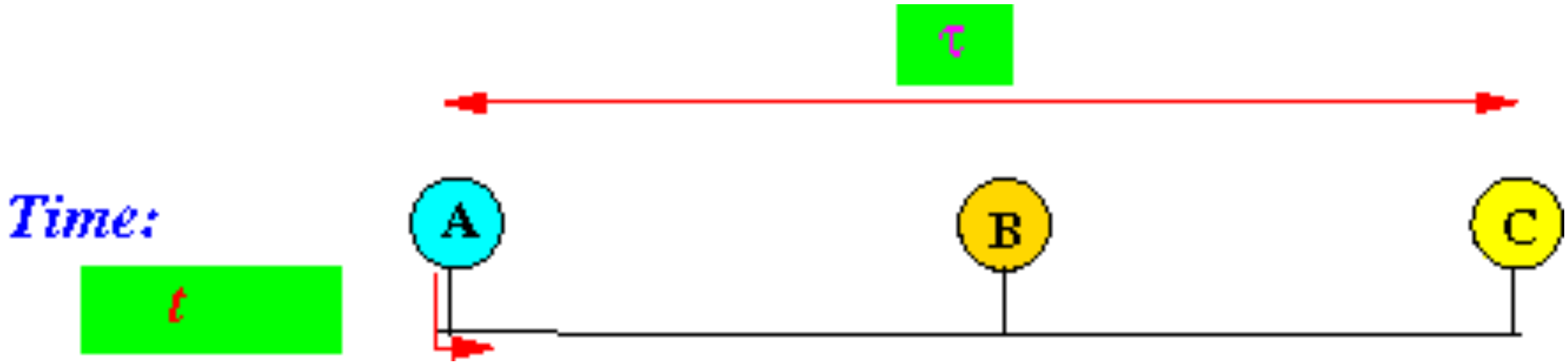
- ***If* node** has **transmitted** for **2 × τ** (**τ** = end-to-end-delay), and it **did *not*** detect a **collision**,
- ***then***
- The **transmission** will **not** suffer a **collision** for ***certain*** !!

Proof:

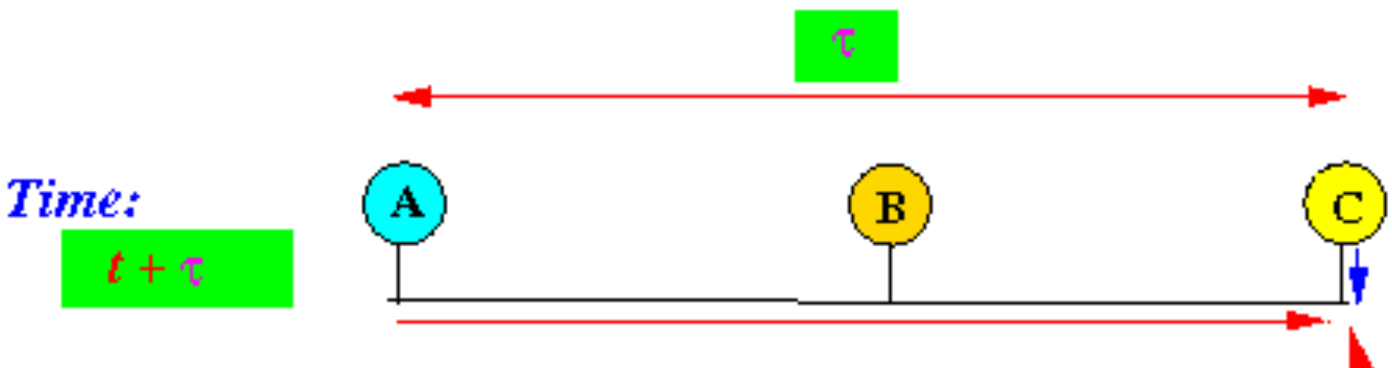
- In **CSMA/CD**, a **node** will **not** transmit if it **hears** an ***on-going*** **transmission**:
-

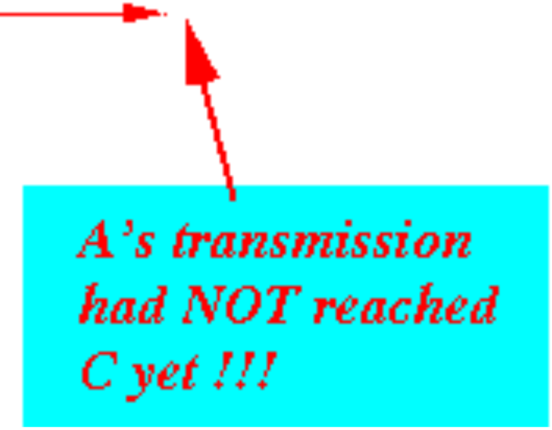


- We **consider** the ***worst case*** **scenario** where:
- a **collision** takes place at the ***last*** **possible moment**
- The **node** ***A*** at **one end** of the **network** start **transmitting** at so ***arbitrary*** time ***t***:
-



- ***Just*** before the ***last*** **node (*C*)** hears ***A*'s transmission**, it **starts** to **transmit**:
-

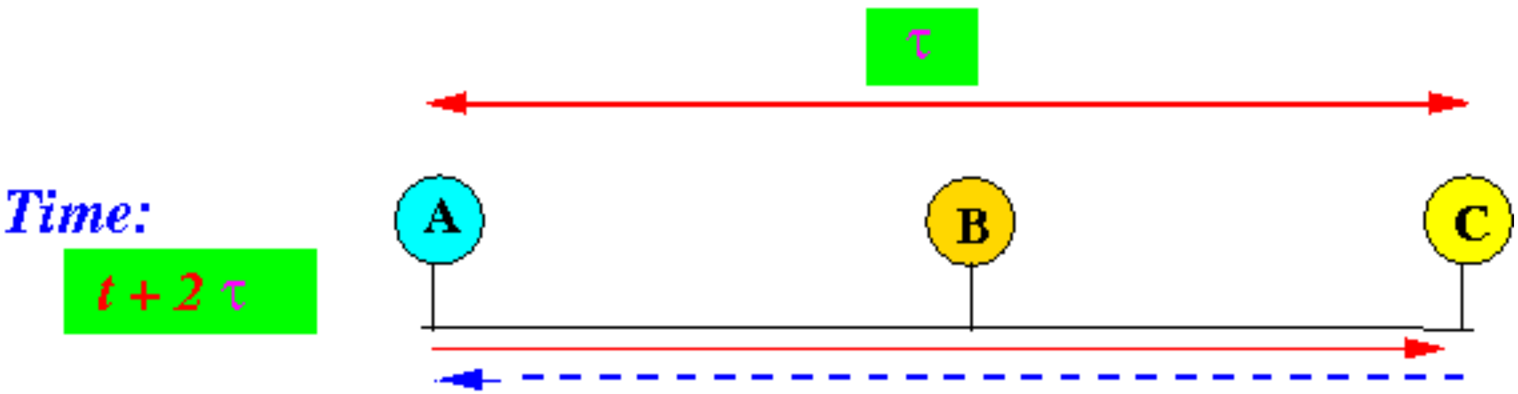




This will **happen** just ***before*** **t+τ** sec

There is **now** a **collision** !!

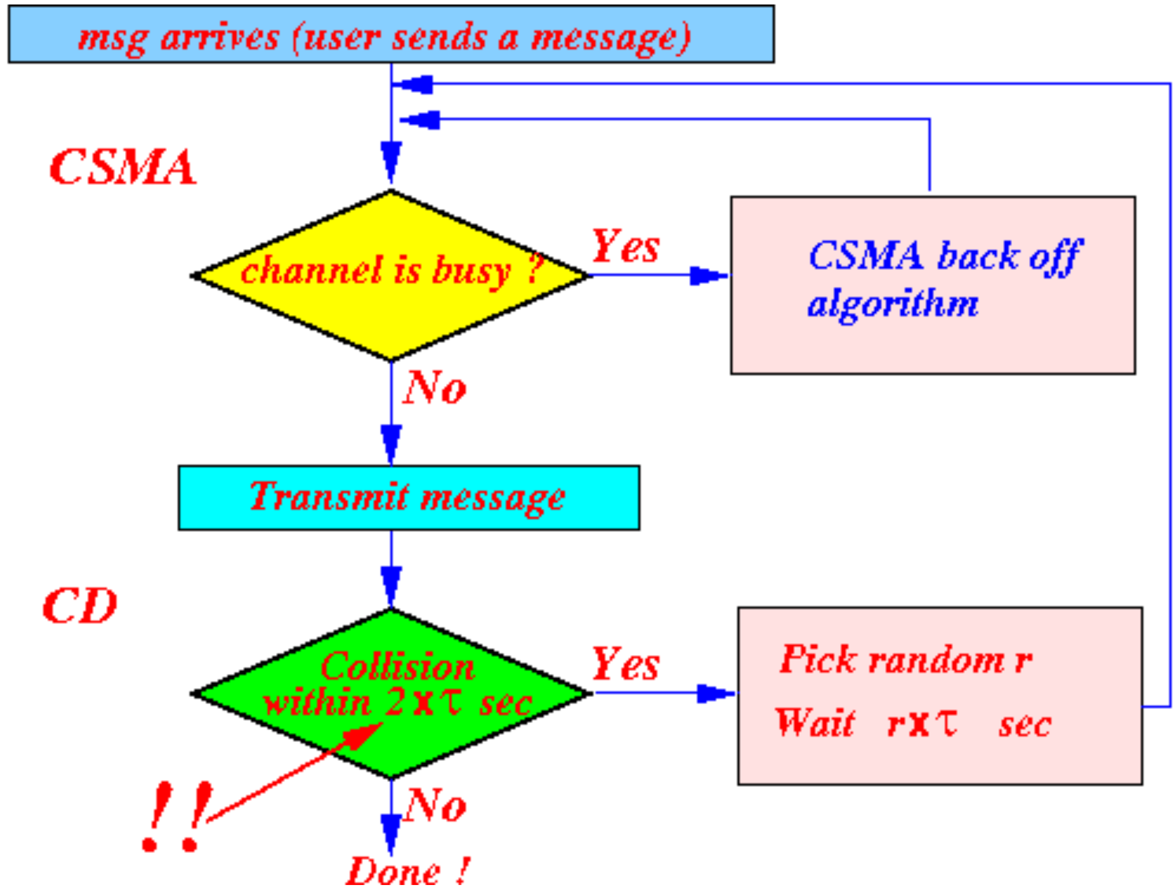
- The **node** ***A*** will **detect** the ***collision*** when ***C*'s** **transmission** reaches *A*:



•This will **happen** at time **t+2×τ** sec

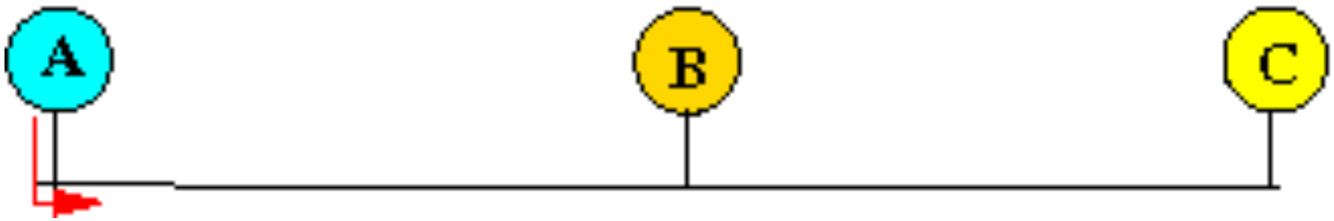
Therefore:

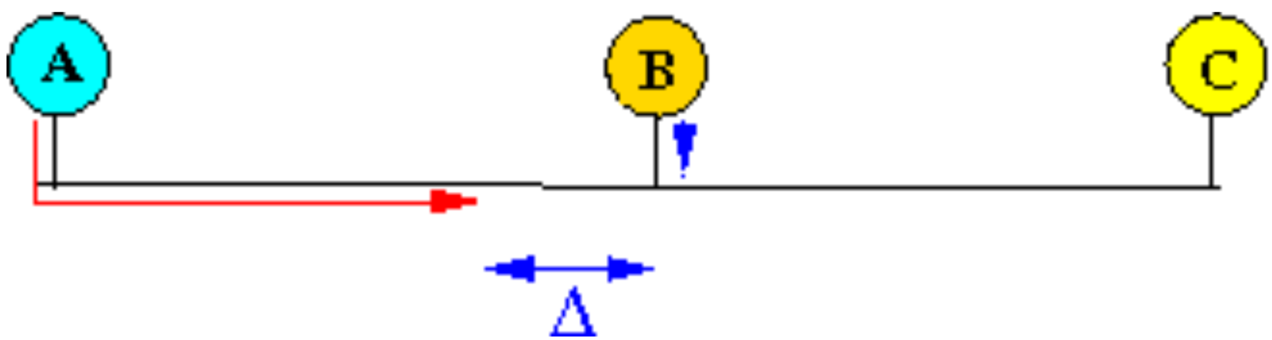
If the **node *A*** has ***not* detected a collision** after **2× τ** time, there **will *not*** be any ***future* collision !!!**



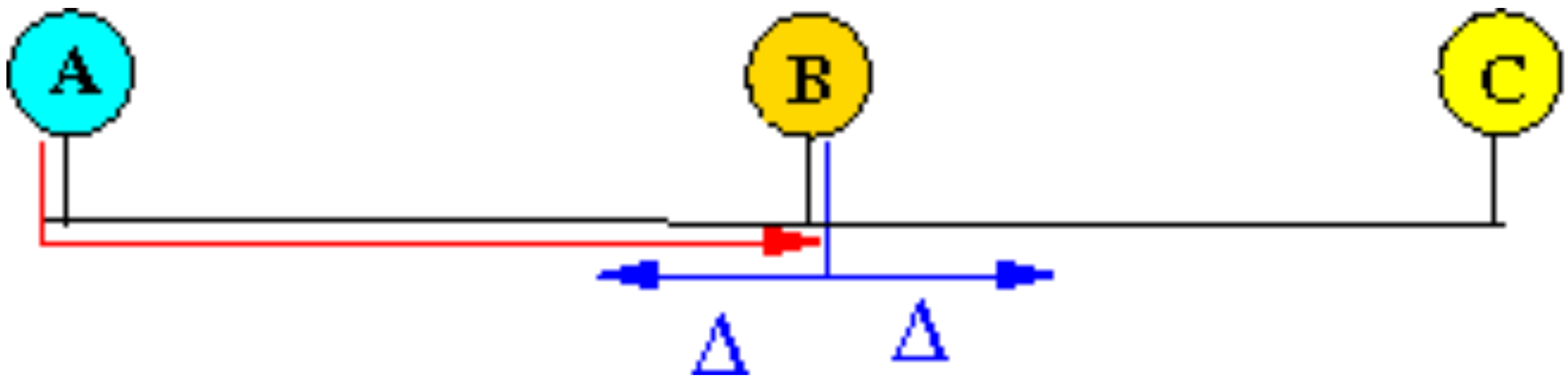
Jamming to ensure collision detection

- **What is the duration of a collision,**
- **Suppose node** ***A*** starts a **transmission**:

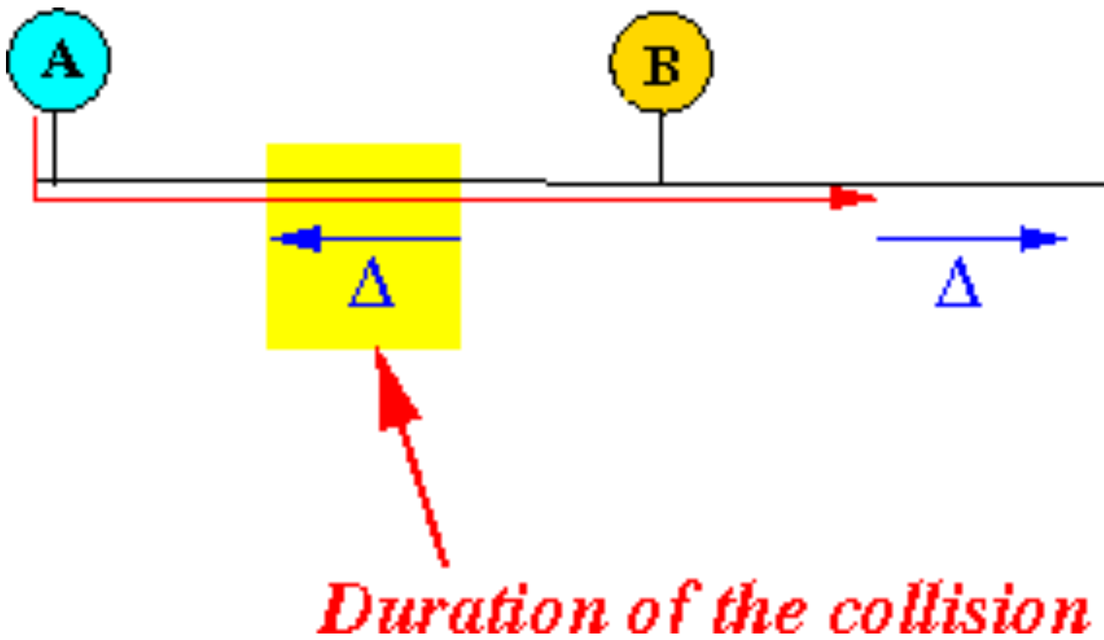


- **Node** ***B*** starts to **transmit** at Δ time ***before A***'s **transmission** arrives at ***B***:
- 

- **Suppose node *B* stops transmitting** when ***B*** hears ***A*'s transmission**:
-



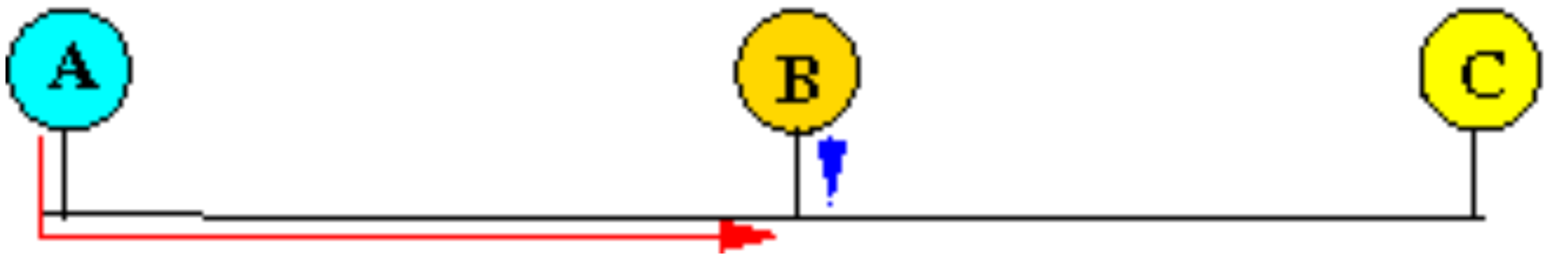
- **Then:**
-



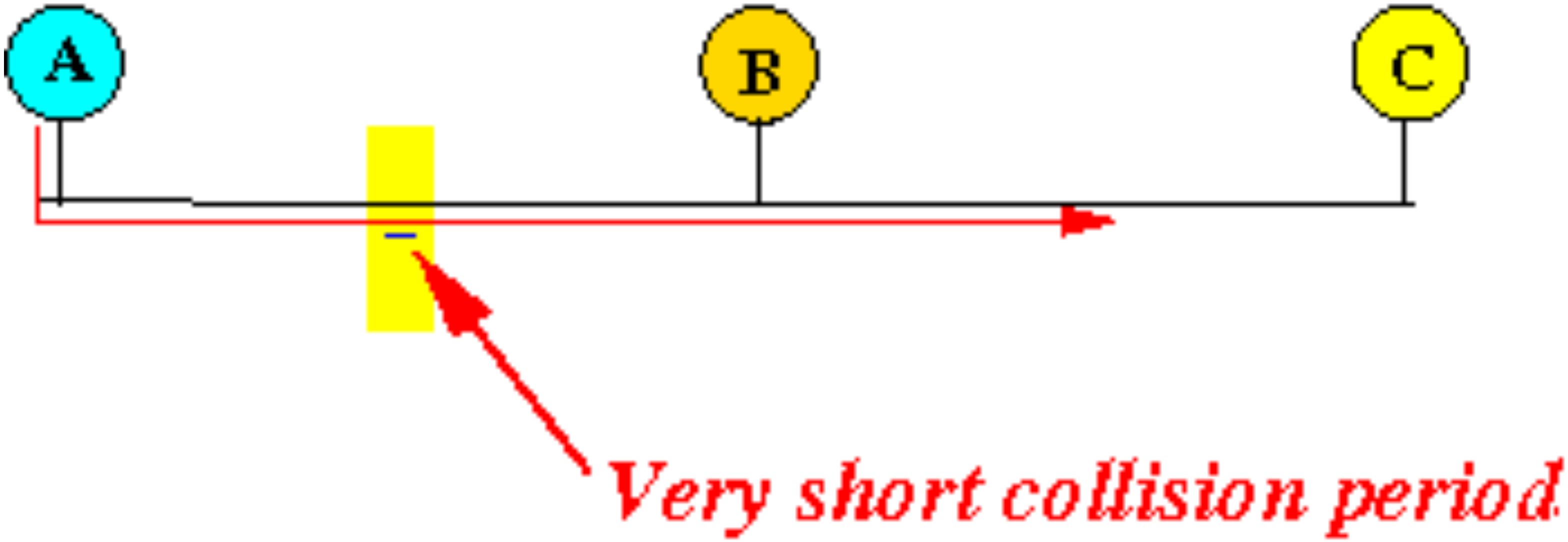
the ***duration*** of the **collision** is **equal** to: Δ sec

The duration of collision can be *very* short

- If a **node** (***B***) ***starts*** a **transmission** ***just*** **before** a **transmission** ***arrives***:



- the ***duration*** of **collision** will be ***very*** **short**


A
B
C
Very short collision period

Collision detection may fail…

- ***Short* collision periods** generates ***few* bit errors** and ***may* escape detection** !!!

- **How to improve collision detection?**

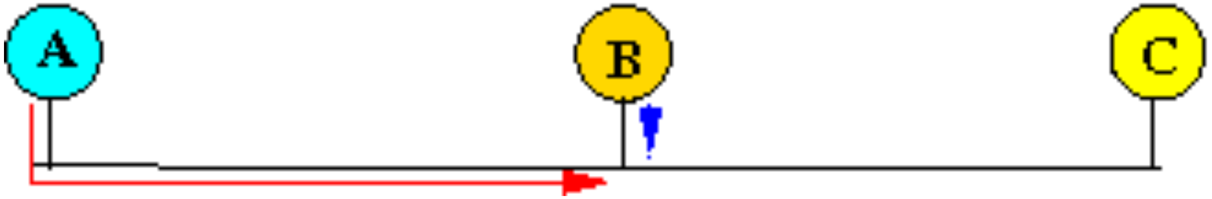
-

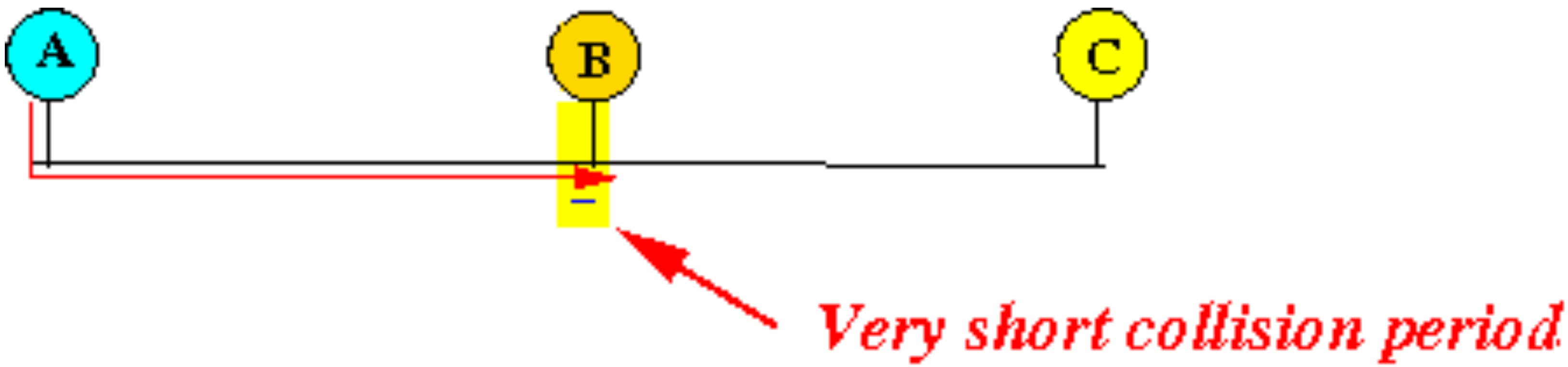
Making sure that a **collision** is *easily* **detectable**:

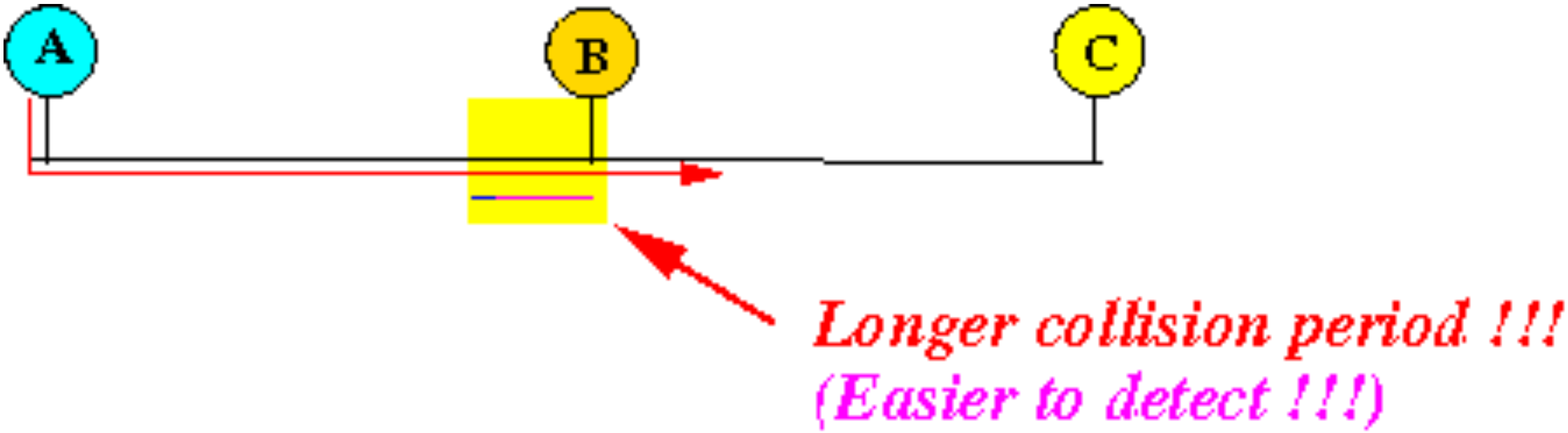
- When a ***transmitting*** **node** has **detected** a **collision**, it will:

1. **Transmit** a **(short) jam signal**
2. **Stop** transmitting

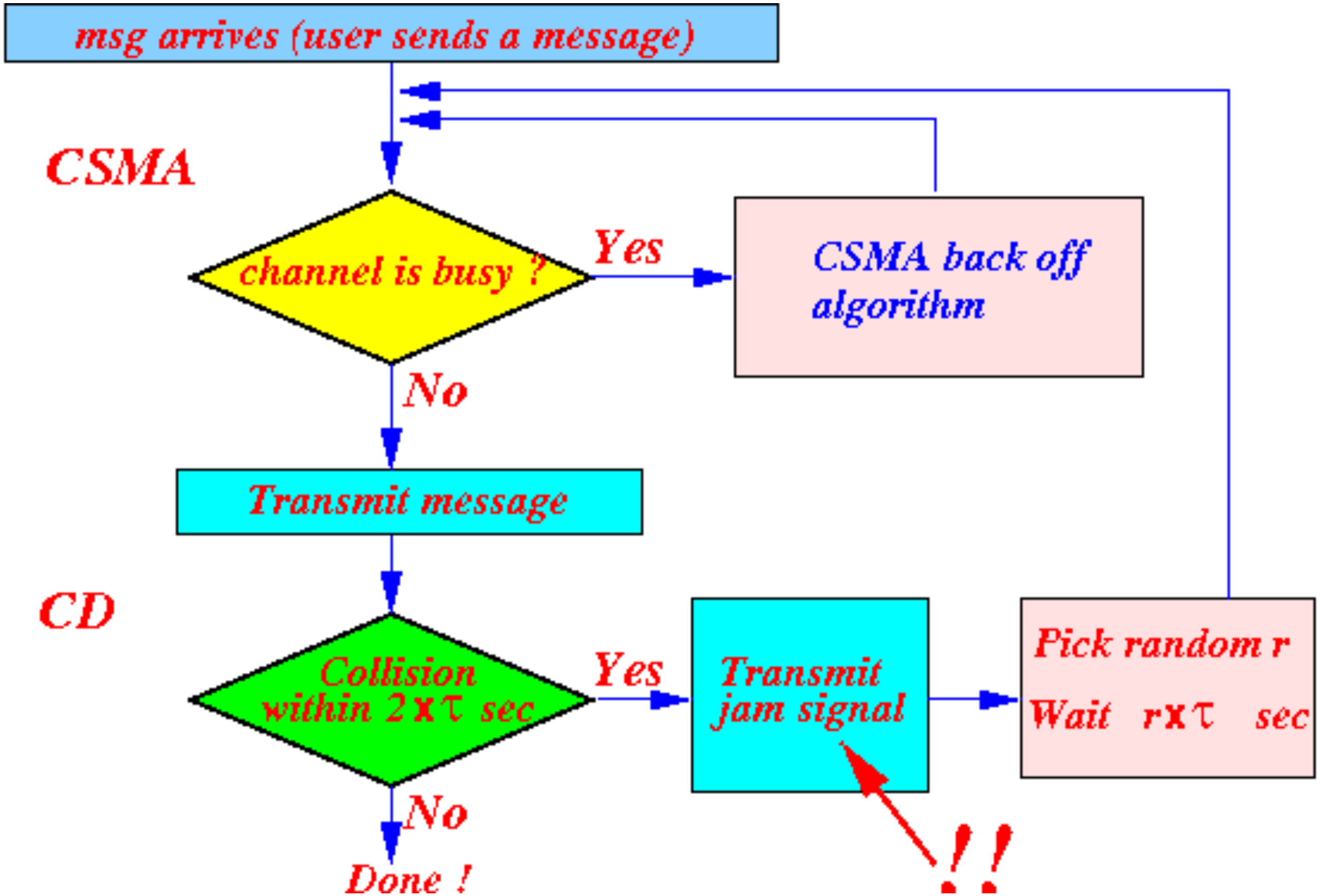
- The **jam signal** is used to **help** the *other* **transmitting nodes** detect the **collision**

- **Example:**
- **Node *B*** starts **just** before ***A*'s transmission** arrives:
- 

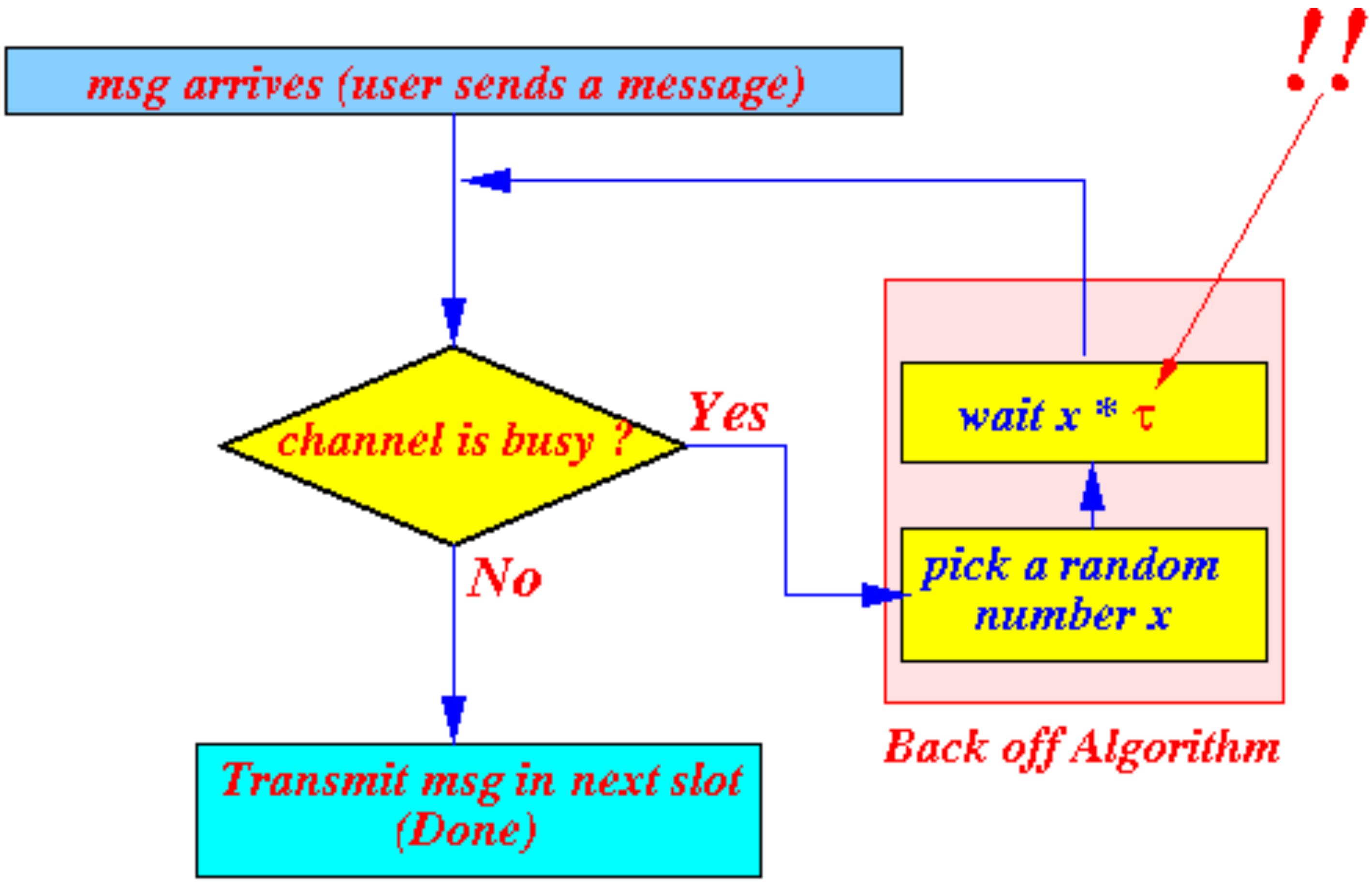
- **Node *B*** has **detected** the **collision**:
- 

- **Node *B*** transmits a (short) **jam signal** to improve **collision detection**:
- 

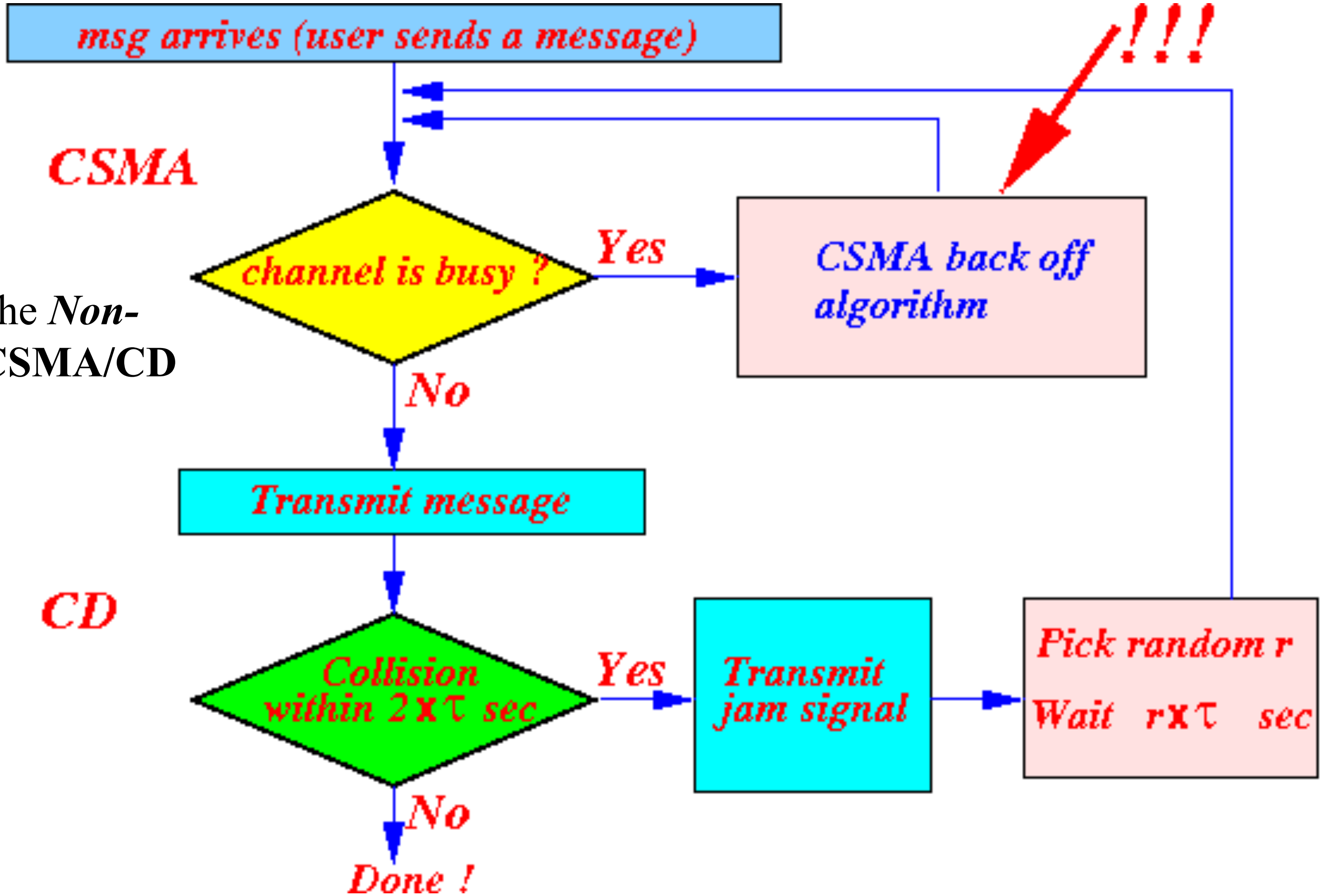
The final CSMA/CD protocol (chart)



Variants of the CSMA/CD protocols

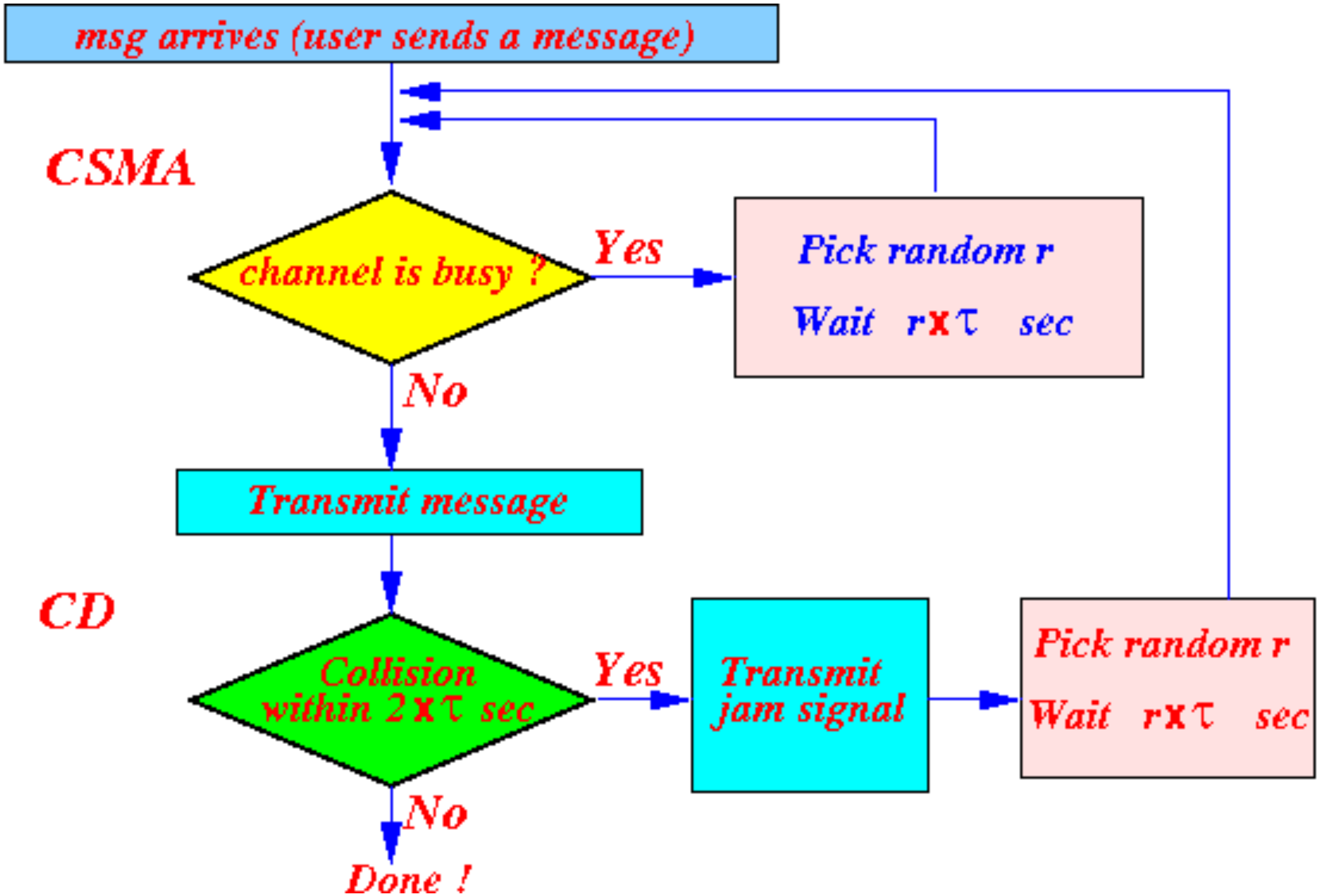
- **Recall** the ***non-persistent*** **CSMA** protocol:
- 

When we use the *non-persistent* **CSMA backoff algorithm** in the **CSMA/CD** flow chart:

- 

we **obtain** the *Non-persistent* CSMA/CD protocol:

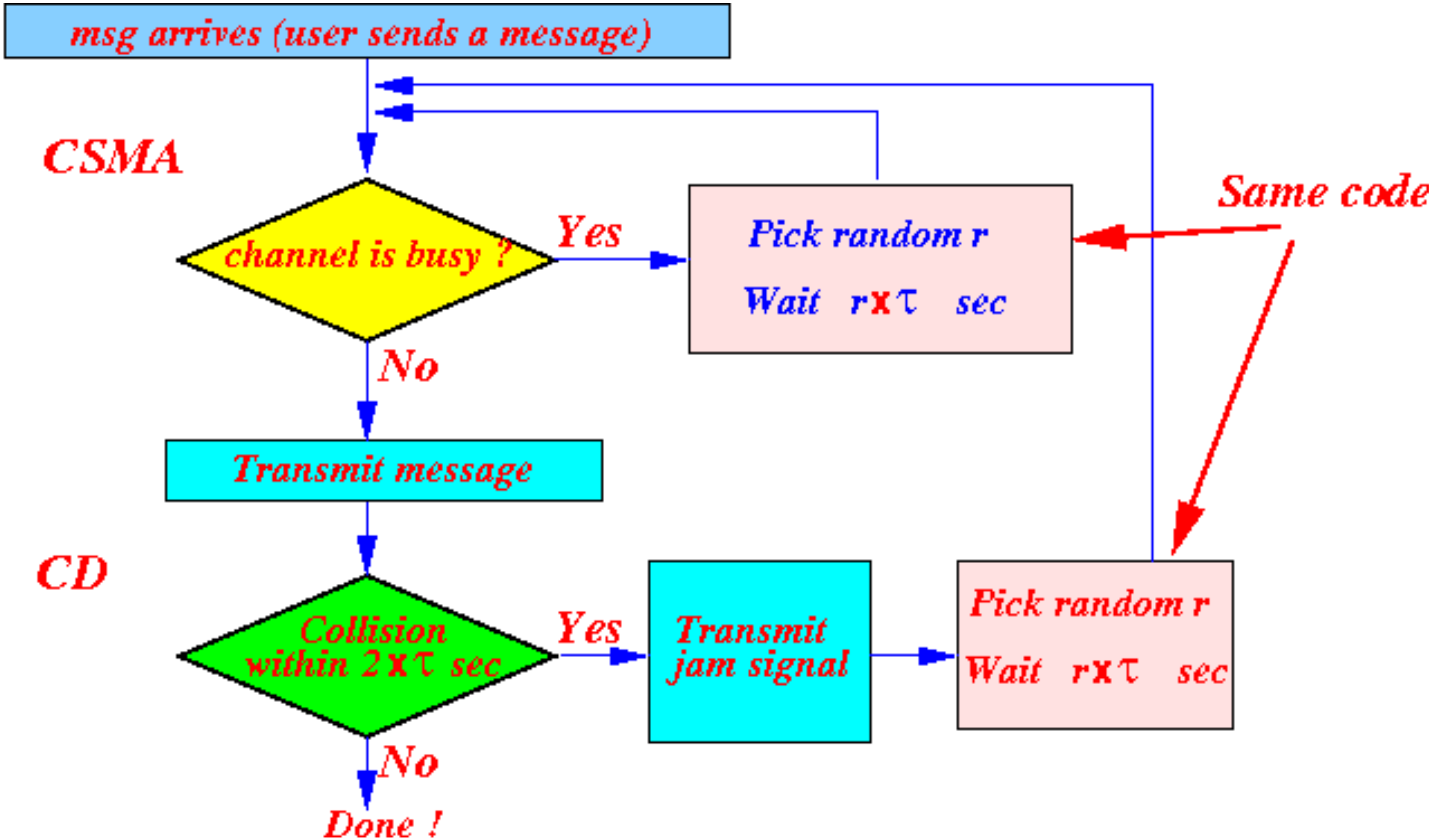
•



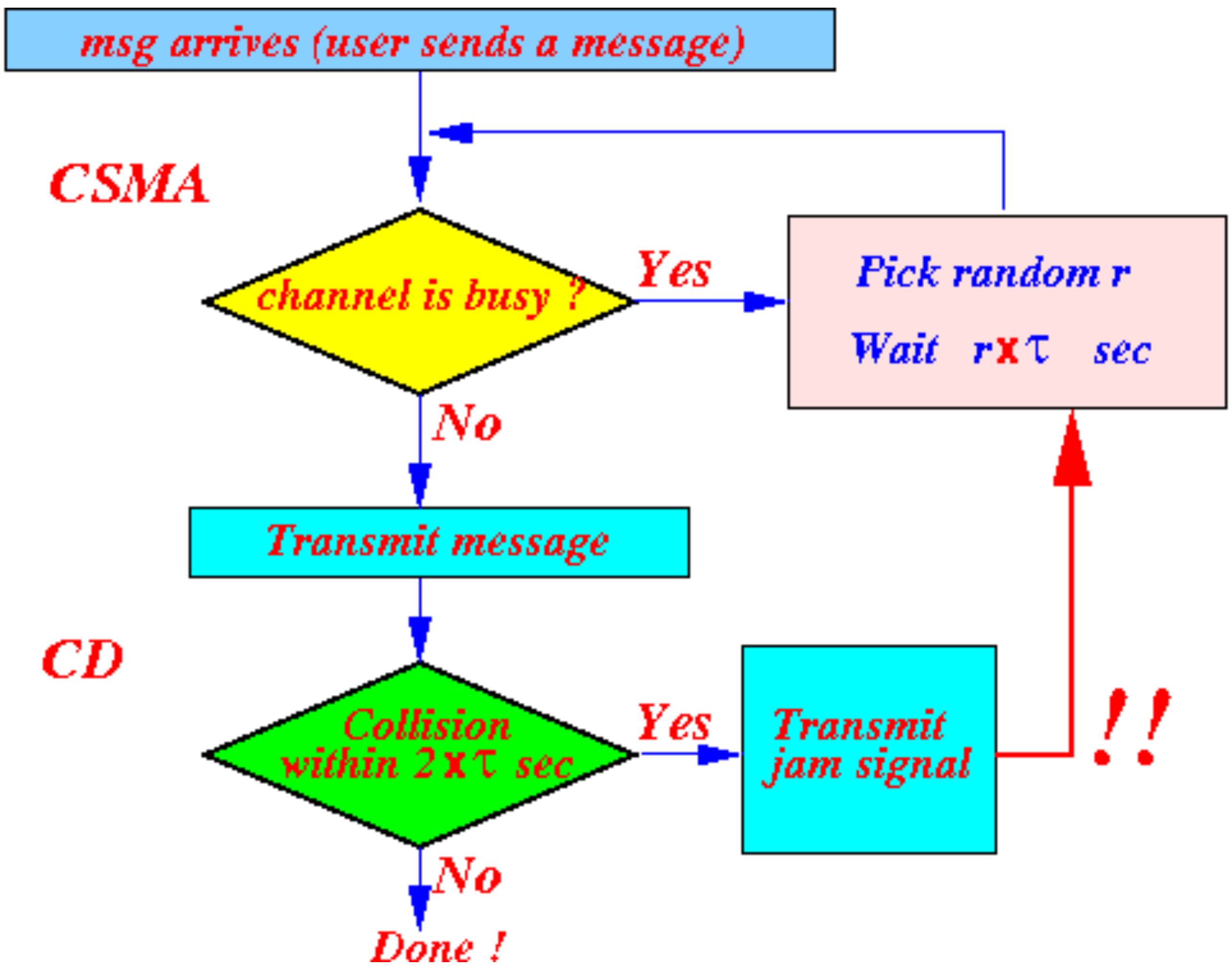
•

•τ = the **end-to-end-delay**

We can **simplify** the ***previous*** **flow chart**:

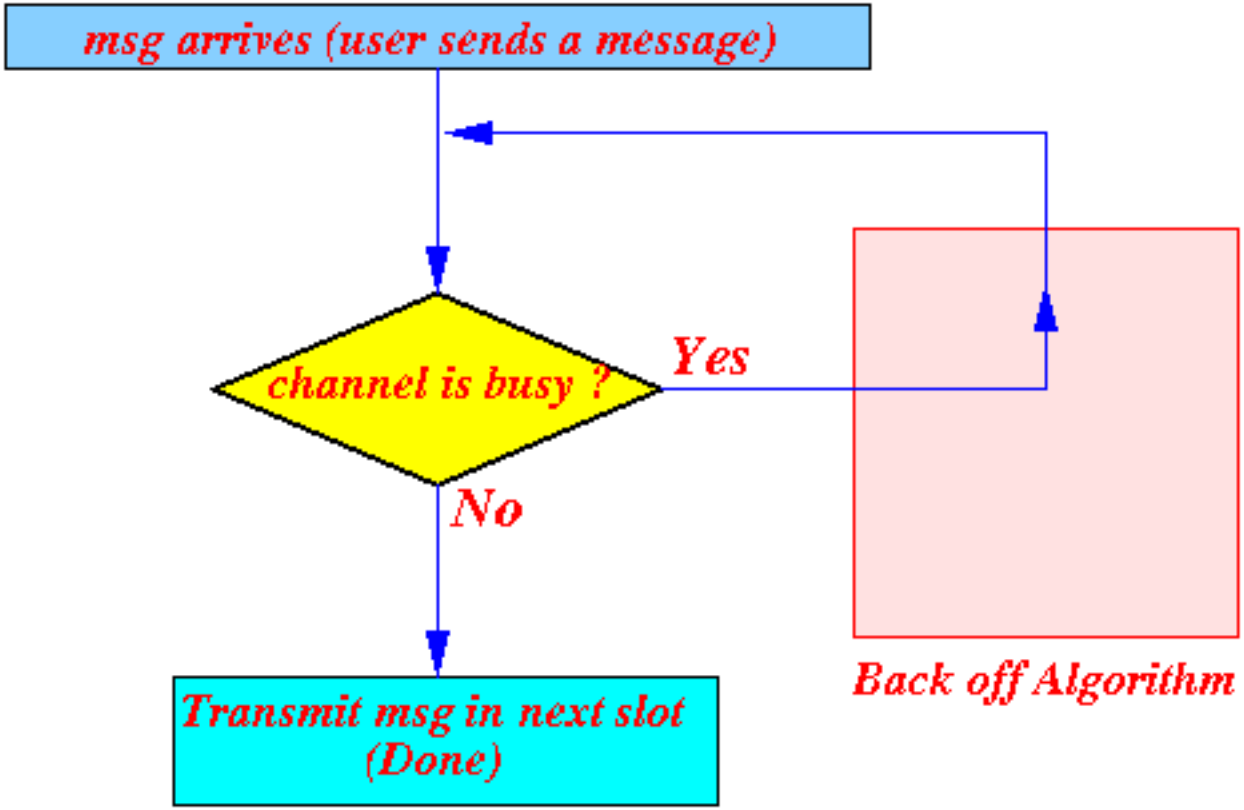


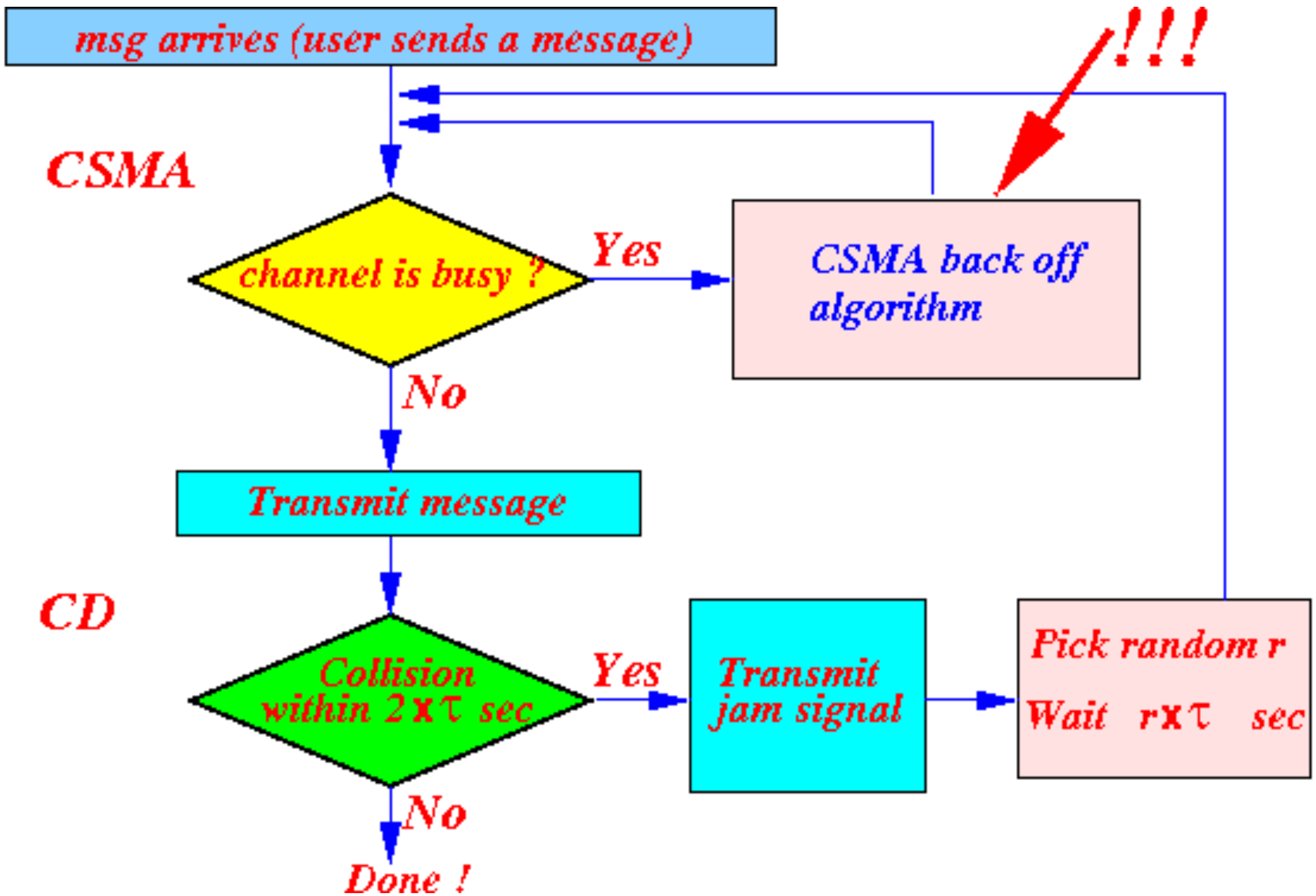
Simplify to the following



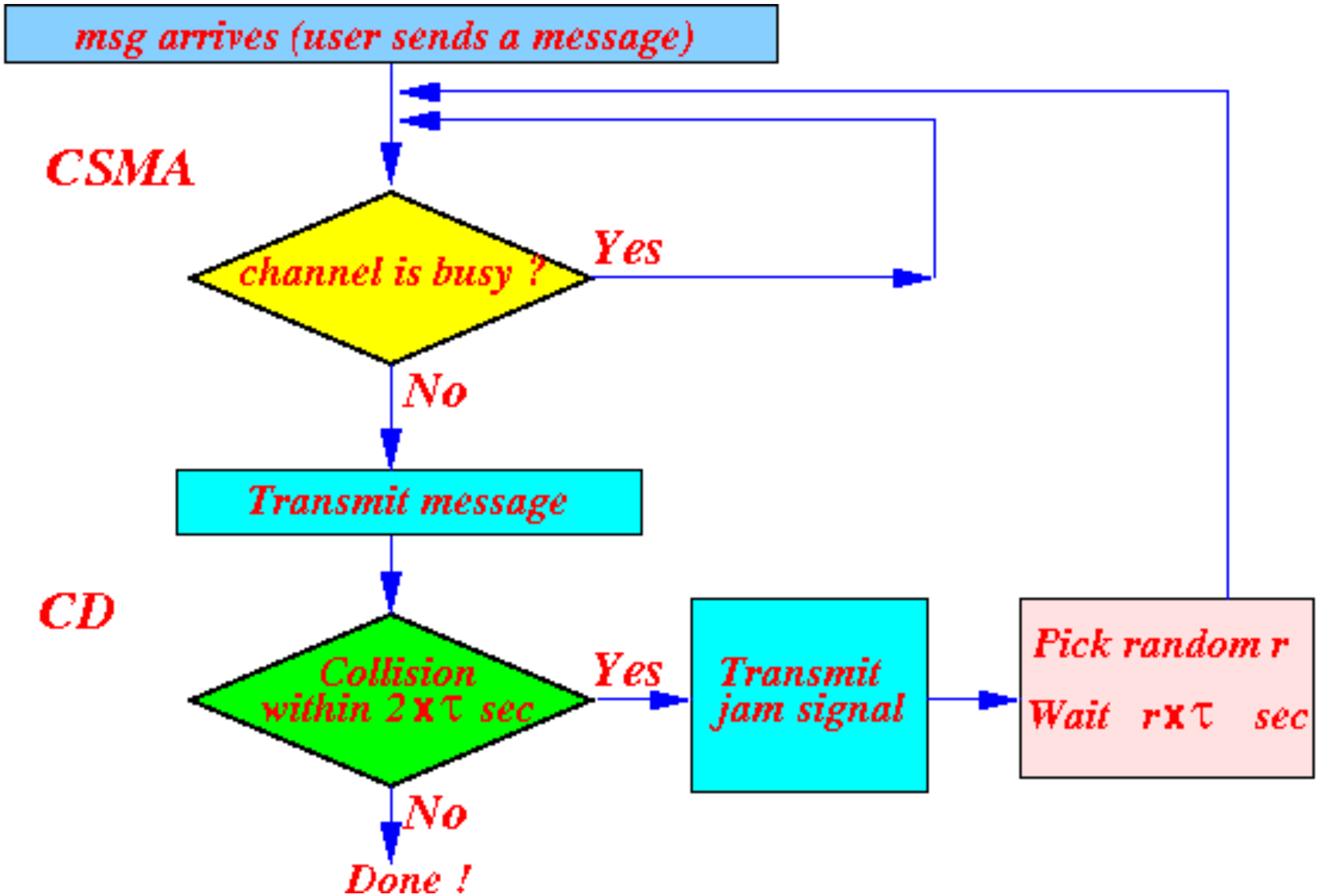
1-persistent CSMA/CD:

- **Recall** the ***1-persistent*** **CSMA** protocol:
-



- When we use the ***non-persistent*** **CSMA backoff algorithm** in the **CSMA/CD** flow chart:
- 

The *1-persistent* **CSMA/CD** protocol:



Intro to Ethernet

- **Ethernet** was developed at **Xerox PARC** between **1973 and 1974**
- (PARC: **P**alo **A**lto **R**esearch **Centre** Incorporated is a branch of the **Xerox company**)
- The **Ethernet protocol** was invented by **Robert Metcalfe**

- **Metcalfe** was **inspired** by **Aloha** --- which he studied as part of his **PhD dissertation**

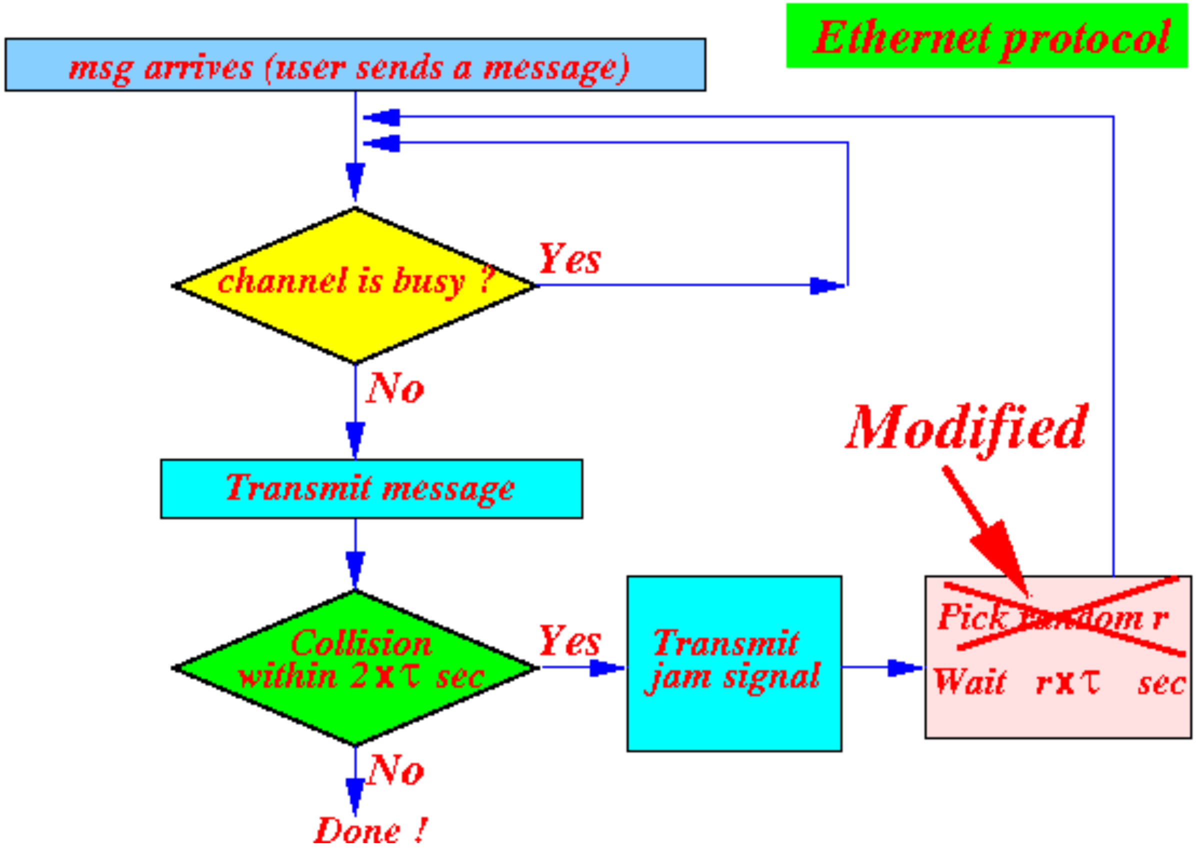
Documentation on the **Ethernet**

- "The Ethernet, A Local Area Network. Data Link Layer and Physical Layer Specifications" - **1980**
- **Standardization dates**:
- **International standard** in **1983**
- The **IEEE 802.3 standard** on: **June 23, 1983**
-
-

The Ethernet *Protocol*

- A ***modified 1-persistent*** **CSMA/CD protocol** !!
- The **Ethernet protocol flow chart:**
-

Ethernet has
a ***modified"*** **back
off algorithm**



The Ethernet *Frame* format

Meaning of the fields:

Preamble: discussed next

Start: 10101011 (Fixed pattern)

Src: Ethernet address of the sender

Dest: Ethernet address of the intended receiver

Length: Length of the data field

Data: Data field

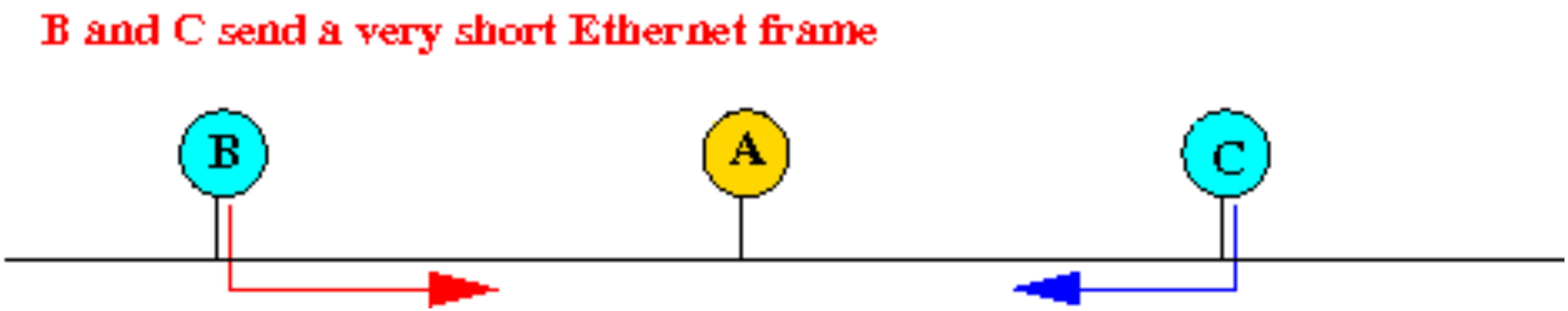
CRC: Check sum

The *pre-amble*

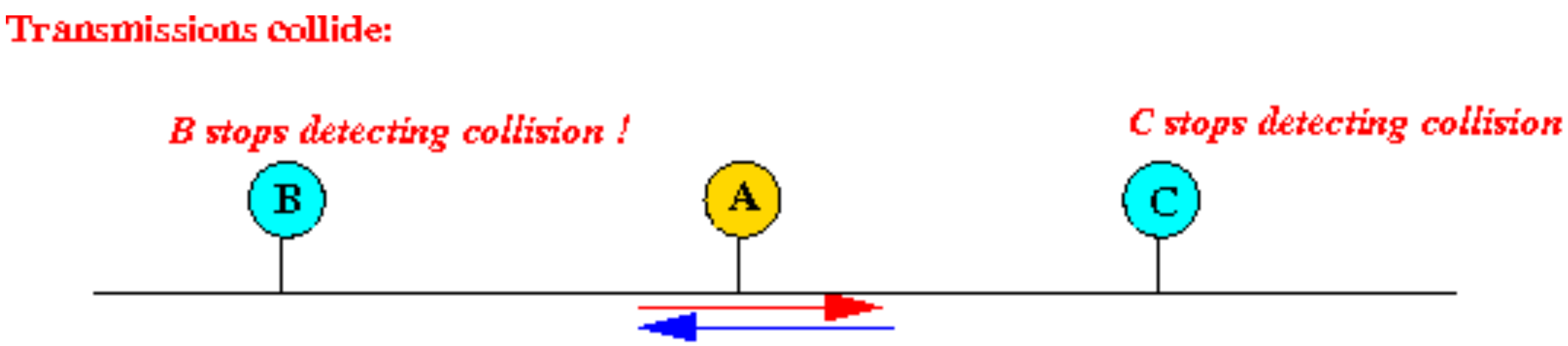
- The **pre-amble** is used to ***pad*** the **Ethernet frame** so that the frame is **long *enough*** for **collision *detection*** purposes
- **Usage** of the **pre-amble:**
- A **transmitting node** can ***only*** **detect** a **collision** ***while*** the **node** is (still) **transmitting**
- The **pre-amble** will **make sure** that ***simultaneous*** **senders** will transmit **long *enough*** that the ***simultaneous*** **senders** will **detect a collision**

Example: how nodes can *miss* a collision

- **Nodes *B* and *C*** transmit ***very* short frames**:
-

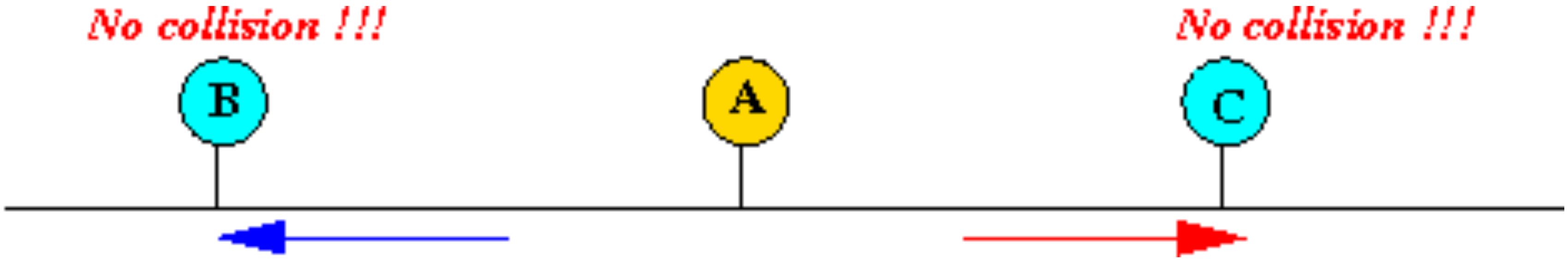


Their **transmissions** will **collide**:

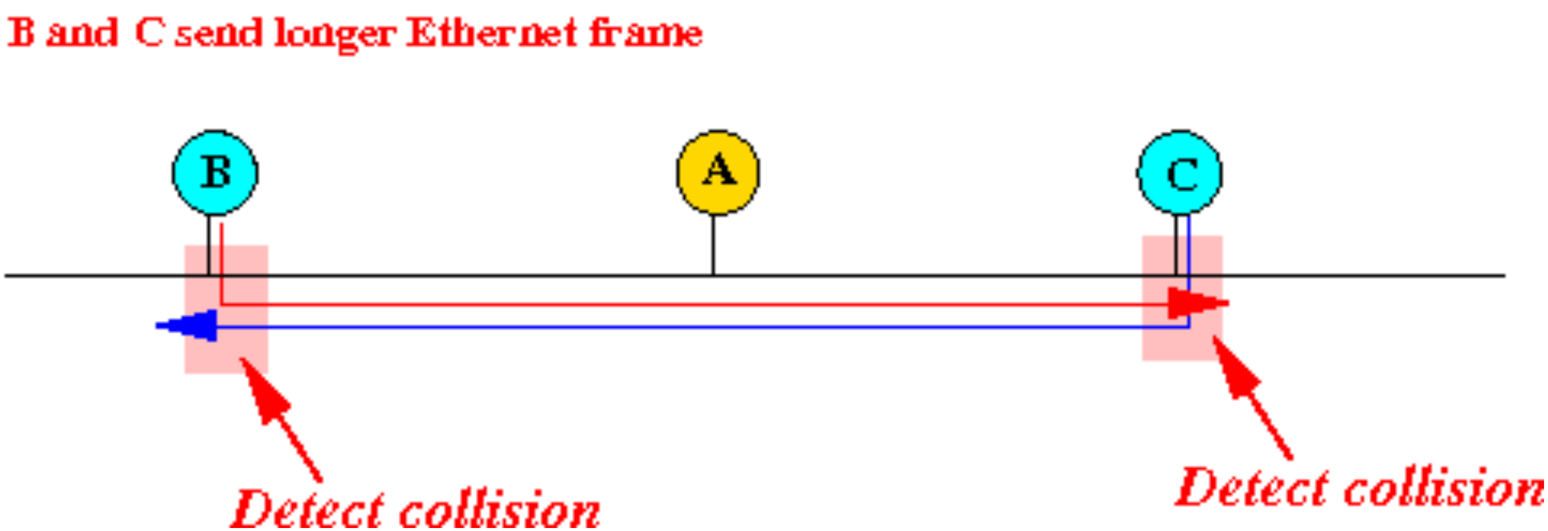


However: the **sending nodes** (***B*** and ***C***)
did **not** detect the **collision**:

- **However**: the **sending nodes** (***B*** and ***C***) did **not** detect the **collision**:
-



Nodes *B* and *C* will *not* retransmit !

- The **pre-amble** will **make sure** that **transmitting node** will be **transmitting** ***long enough*** to **detect** a **collision**:
- **Nodes *B* and *C*** transmit ***longer*** **frames**: and hears the collision
- 

Ethernet Address

- ***Each* nodes** on a **Ethernet** is ***identified*** by a ***unique* 48 bits Ethernet address**
- The **Ethernet address** is ***permanently* inscribed** in the **Ethernet card** of a **computer**
-

- **UNIX command** to find the **Ethernet Address** on a **computer**:
- $ ifconfig -a

- eth0
- Link encap:Ethernet HWaddr 2C:41:38:8B:05:38
- inet addr:170.140.150.36
- Bcast:170.140.151.255 Mask:255.255.254.0
- inet6 addr: fe80::2e41:38ff:fe8b:538/64
- Scope:Link UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
- RX packets:681868507 errors:0 dropped:0 overruns:0 frame:0
- TX packets:628584348 errors:0 dropped:0 overruns:0 carrier:0 collisions:0 txqueuelen:1000
- RX bytes:335075333658 (312.0 GiB) TX bytes:540484678811 (503.3 GiB)
- Interrupt:20 Memory:fb000000-fb020000 lo Link encap:Local Loopback inet addr:127.0.0.1 Mask:255.0.0.0 inet6 addr: ::1/128 Scope:Host UP LOOPBACK RUNNING MTU:16436 Metric:1 RX packets:267964 errors:0 dropped:0 overruns:0 frame:0 TX packets:267964 errors:0 dropped:0 overruns:0 carrier:0 collisions:0 txqueuelen:0 RX bytes:107815976 (102.8 MiB) TX bytes:107815976 (102.8 MiB)

Explanation:

- Ethernet address = 2C:41:38:8B:05:38 (Hexadecimal digits)
- 2 C 4 1 3 8 8 B 0 5 3 8 =
- 0010 1100 0100 0001 0011 1000 1000 1011 0000 0101 0011 1000

Why are there *no* send/receiver sequence numbers ???

There are:

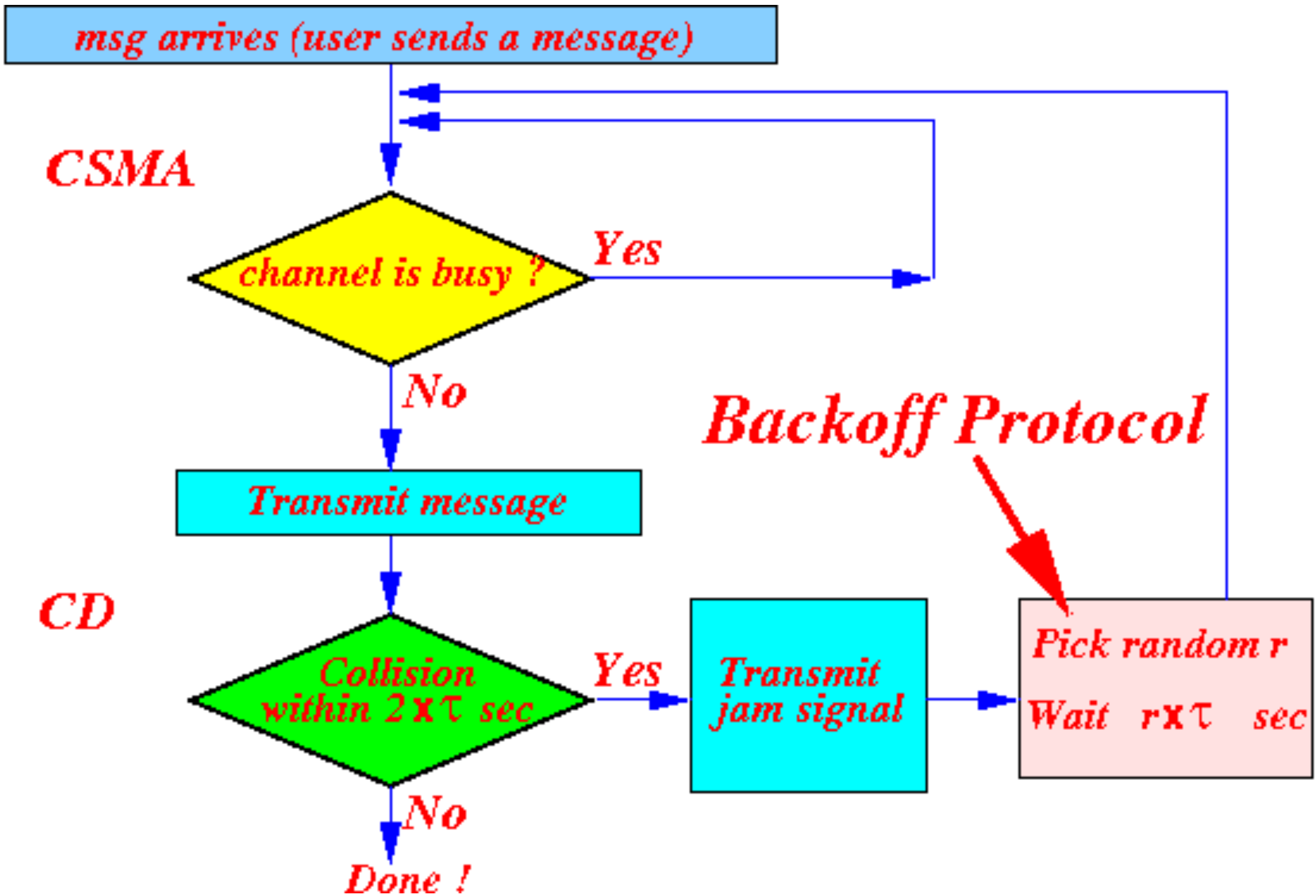
- ***No Send*** **sequence number**
- ***No ACK*** **sequence number**

in Ethernet frames

Reason:

- The **Ethernet (LAN) protocol does** ***not*** provide **reliability** function !!
- **Ethernet** relies on the ***Transport*** **Layer** to ensure **reliable communication**
- There is an **ARQ protocol** in the ***Transport*** **layer** !

The *Exponential* Backoff Algorithm of the Ethernet

- The Backoff Protocol
- 

- The **backoff protocol** is **executed** when a **node have (already) detected** a **collision** !!!

- ***Multiple*** **nodes** are ***ready*** to **transmit** !!

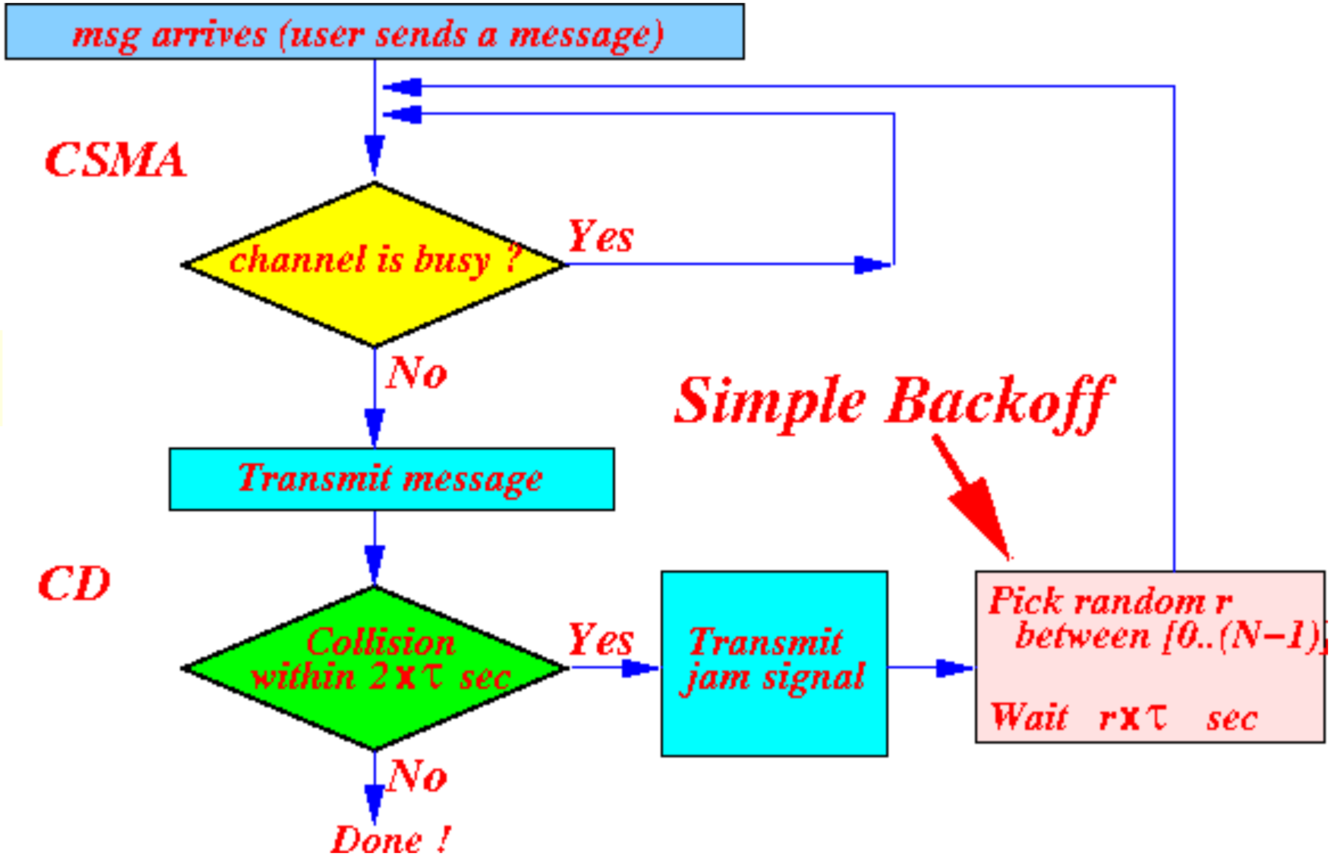
- (***One*** **transmission** can ***never*** result in a **collision**.....)

- ***Goal*** of the **Backoff protocol**:
- ***Re-schedule*** the **transmissions** of the ***collided*** **nodes** so that:
- the **likelihood (chance)** of ***subsequent*** **collisions** is **minimized**

A *simple-minded* backoff protocol

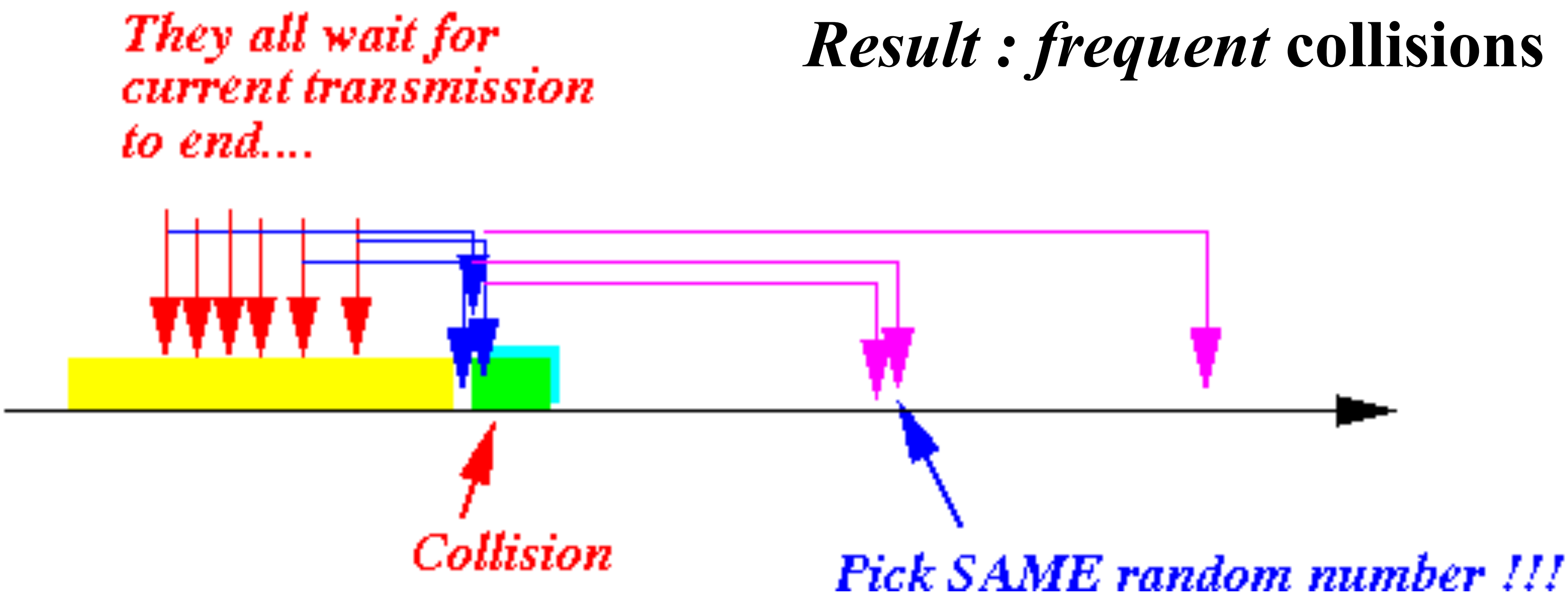
- **Pick** a **random number** *r* between **[0..(N−1)]**   (N is **fixed**)
-

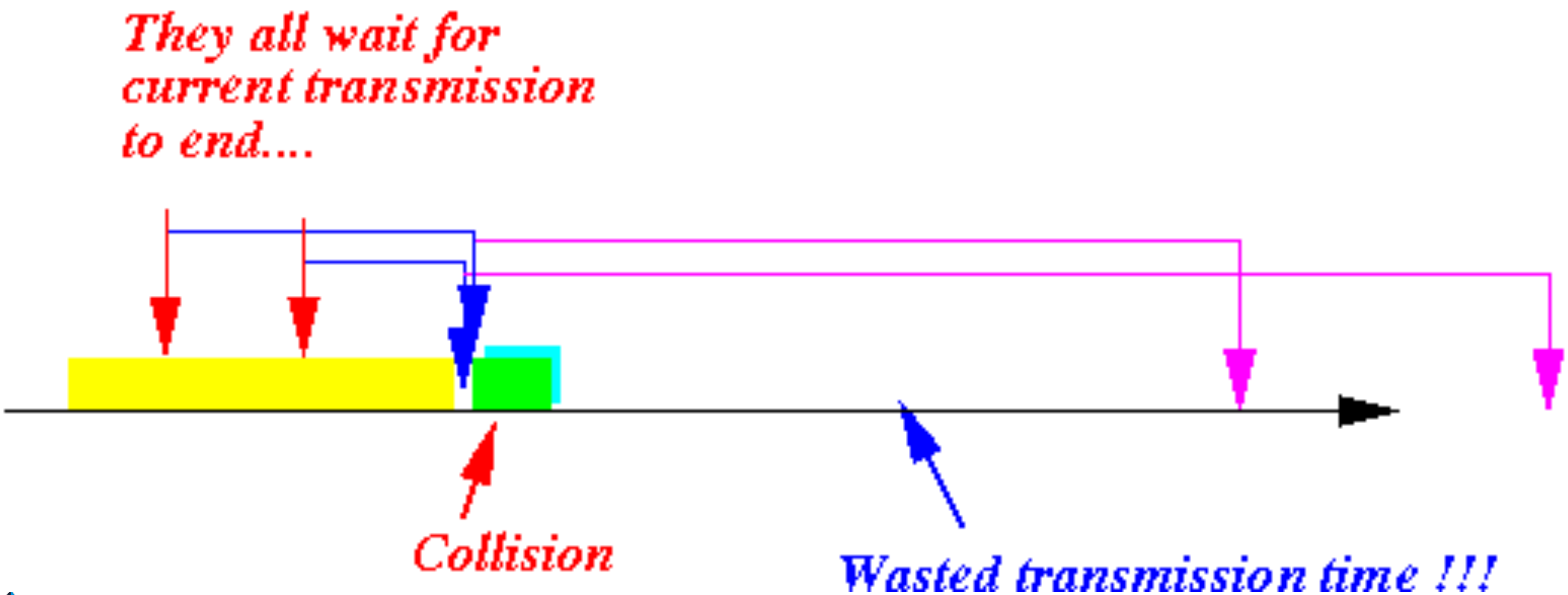
•**Wait** ***r*** **×** **τ** time



Weakness of the ***simple-minded*** **backoff** algorithm

- If *N* is ***small***, and the **system** is **very busy** (= ***many*** **nodes** have ***collided***),
- **then** it will be ***very*** **likely** that ***more*** **than one node** will **select** the ***same*** **random number**
- ***Result : frequent*** **collisions**



- **On the other hand**, if *N* is ***large***, but the **system** is ***lightly*** **loaded**, **then**:
- Nodes ***may*** select an ***large*** random value
- 

- **Result:**
- The **node** will **back off** for a ***long*** **time** (for nothing)...

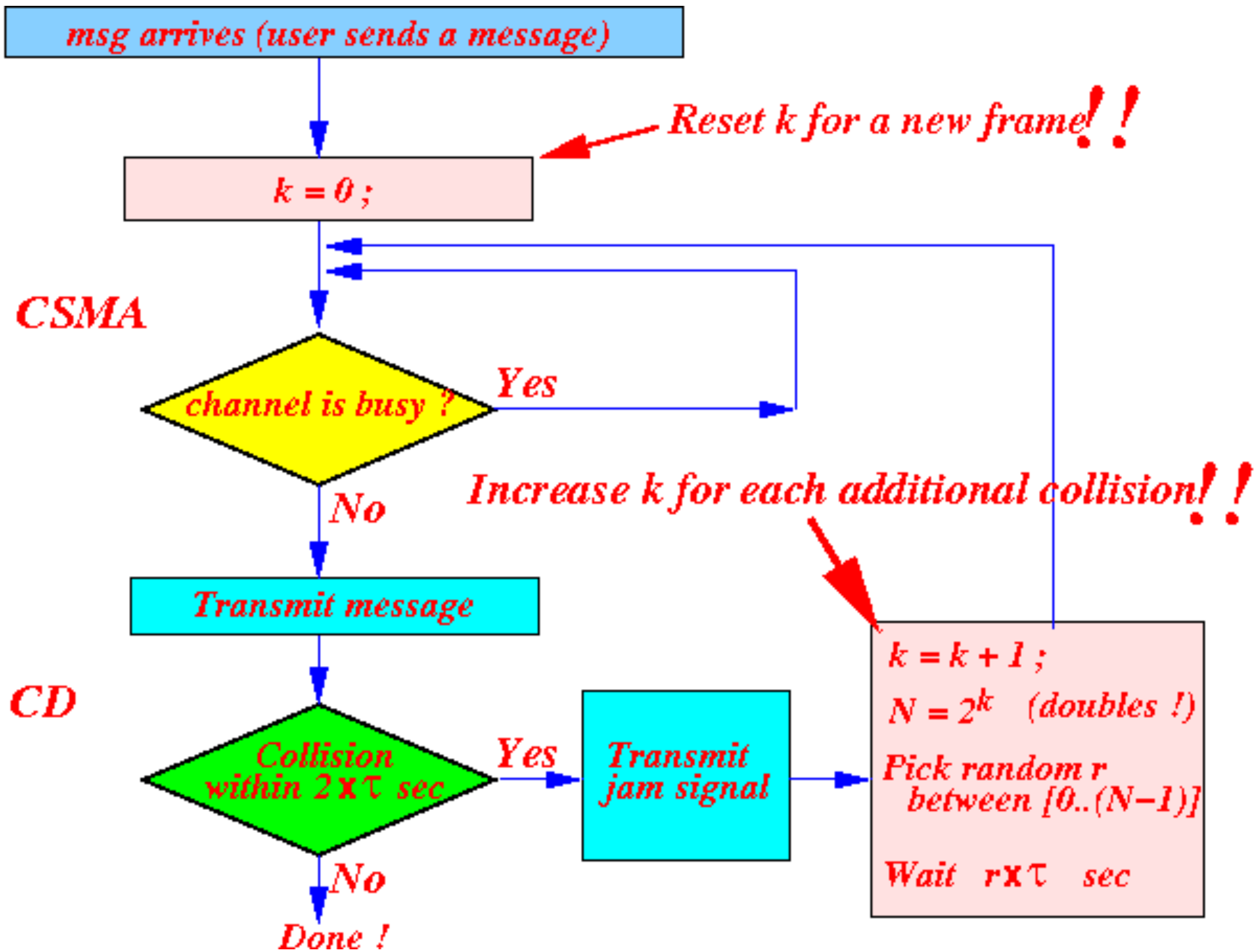
- **Result:**
- **Wasting** of **bandwidth**...

- **Conclusion**
- We need a ***dynamic*** **backoff algorithm** that can ***adjust*** to ***different*** **network operating conditions** !!!

The Exponential Backoff Protocol

- *Adaptive* **back off**:
- Allow **collided nodes** to pick ***small*** **back off** periods in ***earlier*** **re-transmission attempts**
- And pick ***larger*** **back off** periods when there are ***more*** **re-transmission attempts**
- The **exponential backoff algorithm flow chart**:
-

The **exponential backoff algorithm flow chart**:



Explanation

- When a **frame** is **transmitted** for the ***first*** **time**:
- Reset k = 0
- ***If*** there was a **collision**, the ***first*** **retransmission attempt** will use:
- k = k + 1 = 0 + 1 = 1
- N = 2^k = 2^1 = 2
- For the ***first*** **re-transmission attempt**, the **node** will pick a **random number** ***r*** from the range:
- $r \in [0, 1]$
- This **range** is **suitable** when there are ***very few*** **(about 2) nodes** involved in the **collision**

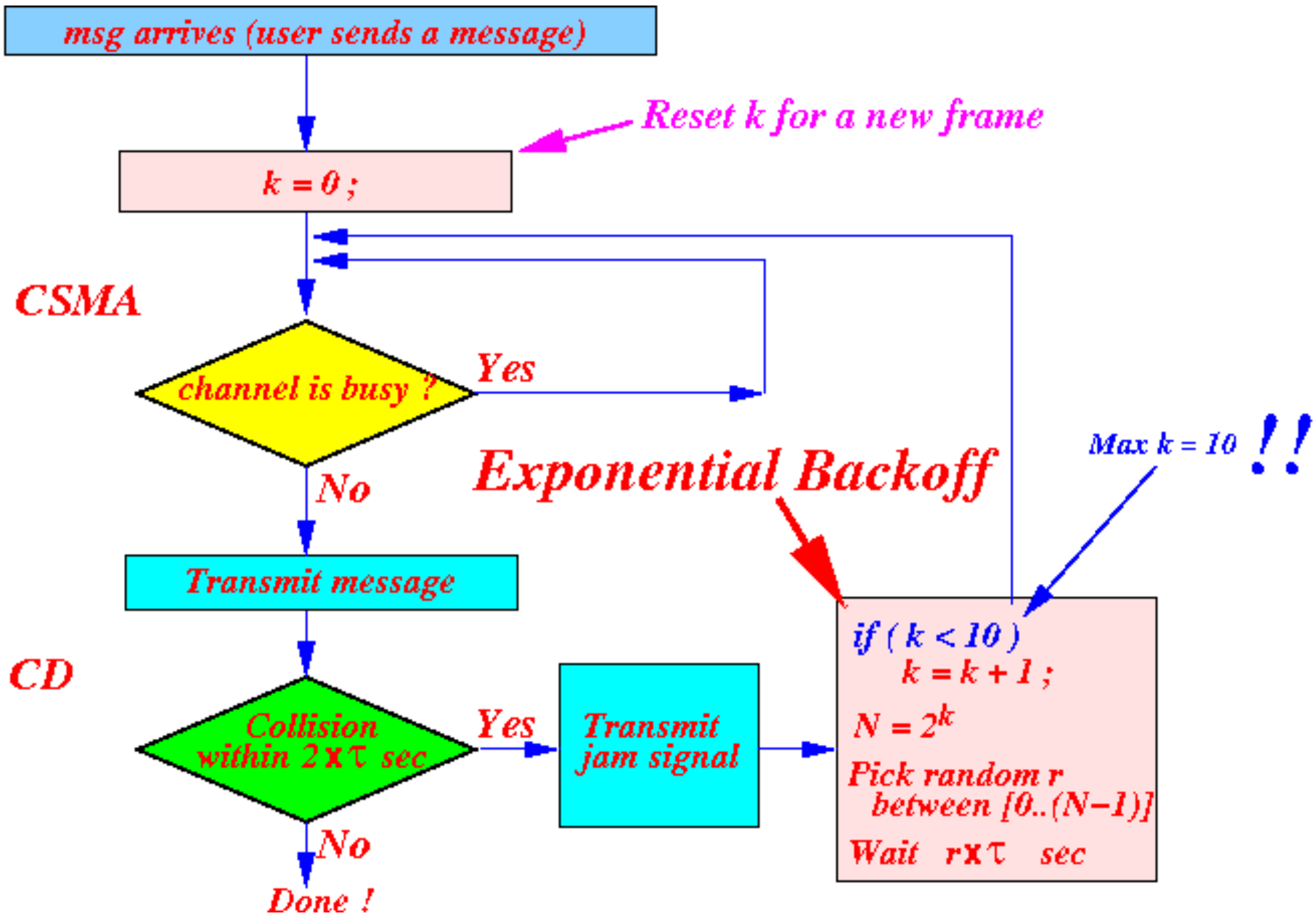
- It the **node** is **involved** in ***another*** **collision** while trying to **transmit** the ***same*** **frame**, **then**:
- k = k + 1 = 1 + 1 = 2
- $N = 2^k = 2^2 = 4$

- So the ***second*** **retransmission attempt** for the ***same*** **frame** will use a **random number** ***r*** from the range
- ***r*** ∈ **[0..3]**
- The **node** is ***adjusting*** to **accommodate** for a ***larger*** **number** of **nodes** (i.e., a ***heavier loaded*** **situation**)

- If there is **another collision**, the ***third*** **retransmission attempt** of the ***same*** **frame** will use:
- $r \in [0..7]$
- **And so on...**

- **A *small* adjustment: upper bound on random range**
- **One Ethernet network** can **support** at most **500 node**
- **Therefore**
- The **maximum range** from which the **random numbers** are selected is:
- $[0..2^{10}-1] \quad = \quad [0..1023]$

The *final* version of the **Ethernet Medium Access Control (MAC)** algorithm:

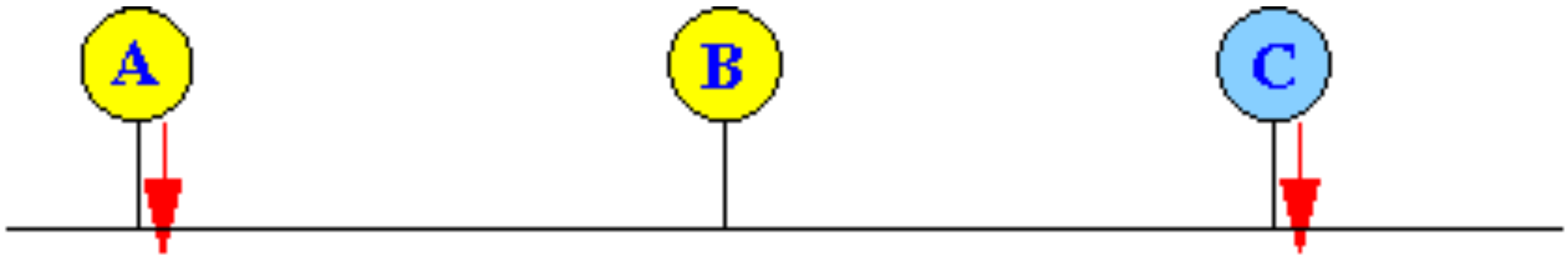
- 

The *unfairness* of the Ethernet Backoff Algorithm

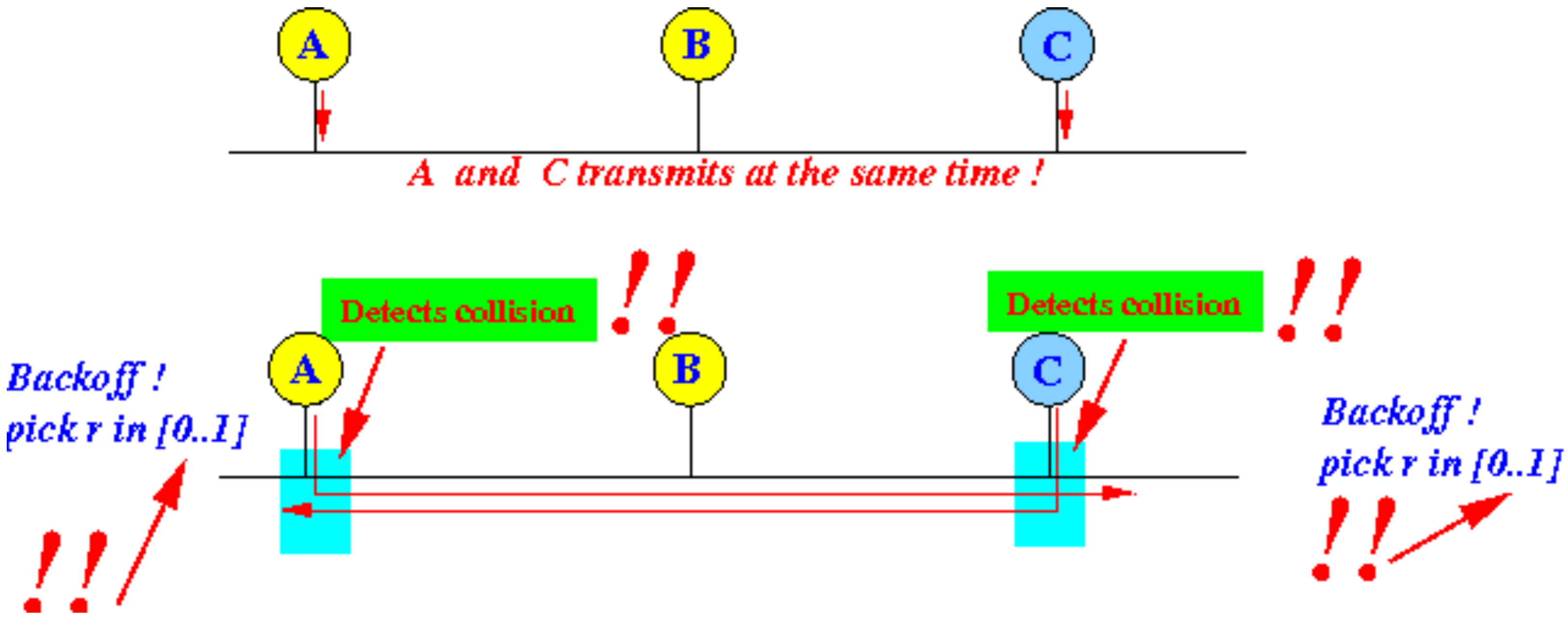
- **Channel capture effect**
- **Channel capture effect** = a **phenomenon** where **one user** of a ***shared*** **(= broadcast) medium *"captures"* (= hogs)** the **medium** for a **significant time**

- ***Unfairness* of the Ethernet's backoff algorithm**
- **Suppose:**
- The **nodes *A*** and ***C*** in an **Ethernet** has a ***large*** **number** of **frames** to **transmit**
-

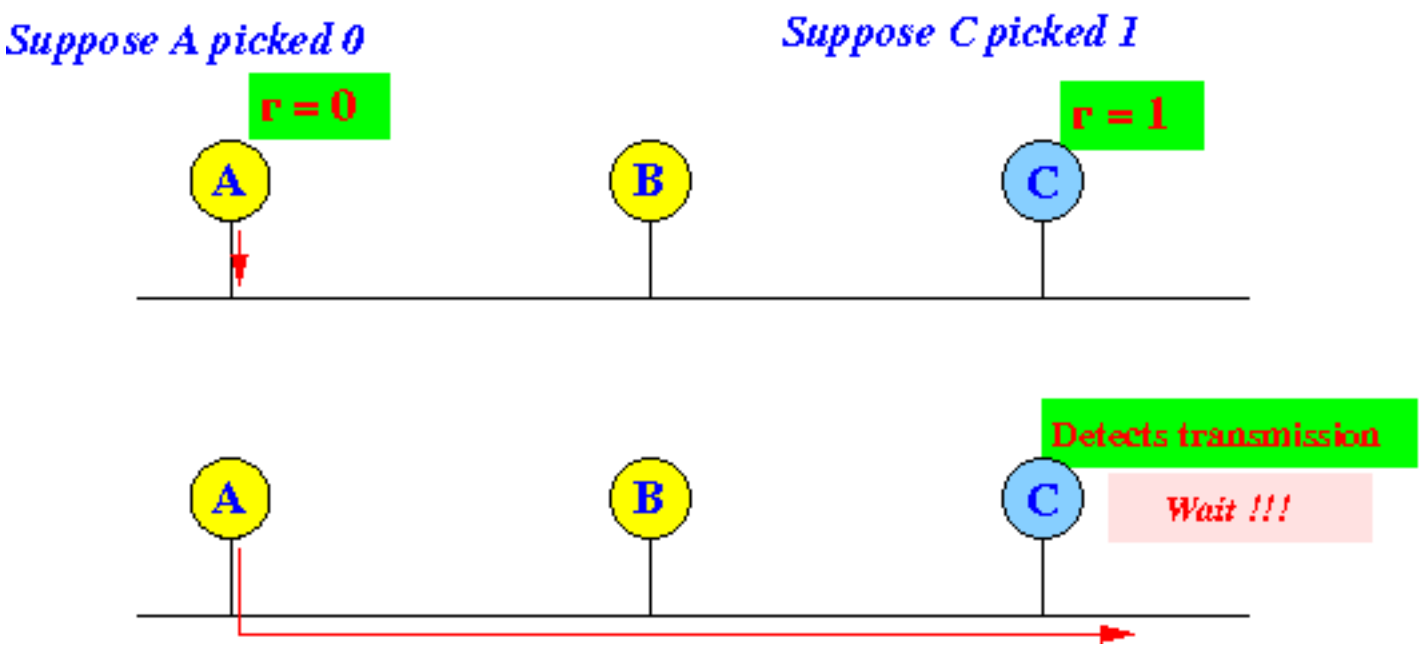
- **Consider** the following **scenario**:
- **Nodes** *A* and *C* **transmits** at the ***same*** **time**:

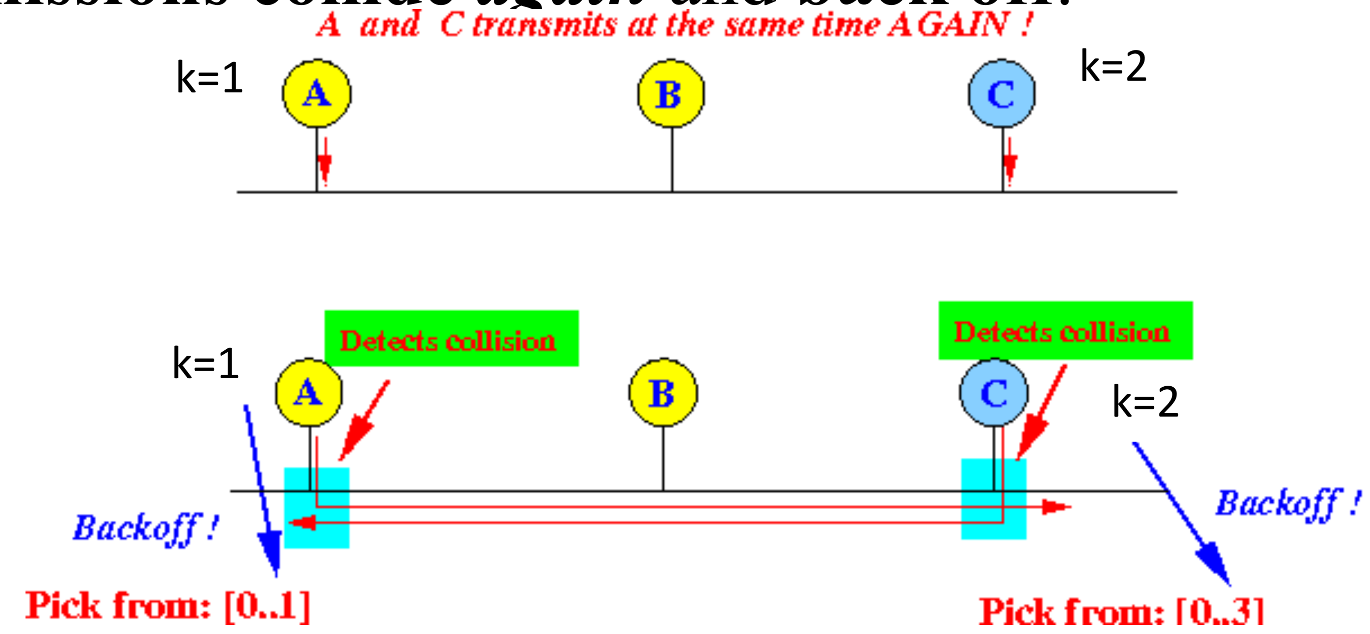


- The **transmissions** of **nodes** *A* and *C* will **collide** and they will **back off**
-

- The **transmissions** of **nodes** *A* and *C* will **collide** and they will **back off**
- 

- **Suppose:**
- Node *A* picked **0**
- Node *C* picked **1**
- **Then node** *A* will **transmits** ***first***:
-



- **Node** ***A*** will **complete** its **transmission**
- When node *A* is **done**, *A* will **transmit** ***again***
- ***Then*** node *C* will ***also*** **transmit**.
- Their **transmissions collide** ***again*** and **back off**:
- 

Notice that:

- **Node *A*** transmits the **frame** for the ***first*** **time**, so:
- Node ***A*** will pick a **random number** from the range **range [0..1]**
- **Node *C*** transmits the **frame** for the ***second*** **time**, so
- Node ***C*** will pick a **random number** from the range **range [0..3]**
- **Problem**
- The ***likelihood*** that **node *A*** will **transmit *before*** the **no de *C*** is ***higher*** !!
- The **probability** is **calculated** in the **section below**..
- This is ***unfair*** for **node *C*** !!!

Unfairness calculation....

- **Question:**
- If **node** ***A*** picks from **[0..1]** and **node** ***C*** picks from **[0..3]**, then:
- What is the **probability** that node ***A*** will transmit ***before*** node ***C*** ?
- **Pre-question:** ***when*** will **node** ***A*** transmit **before node** ***C***:
- Node ***A*** transmits **before** node ***C*** if
- Node ***A*** picks a ***smaller*** **(random) value** than node ***C***

Answer

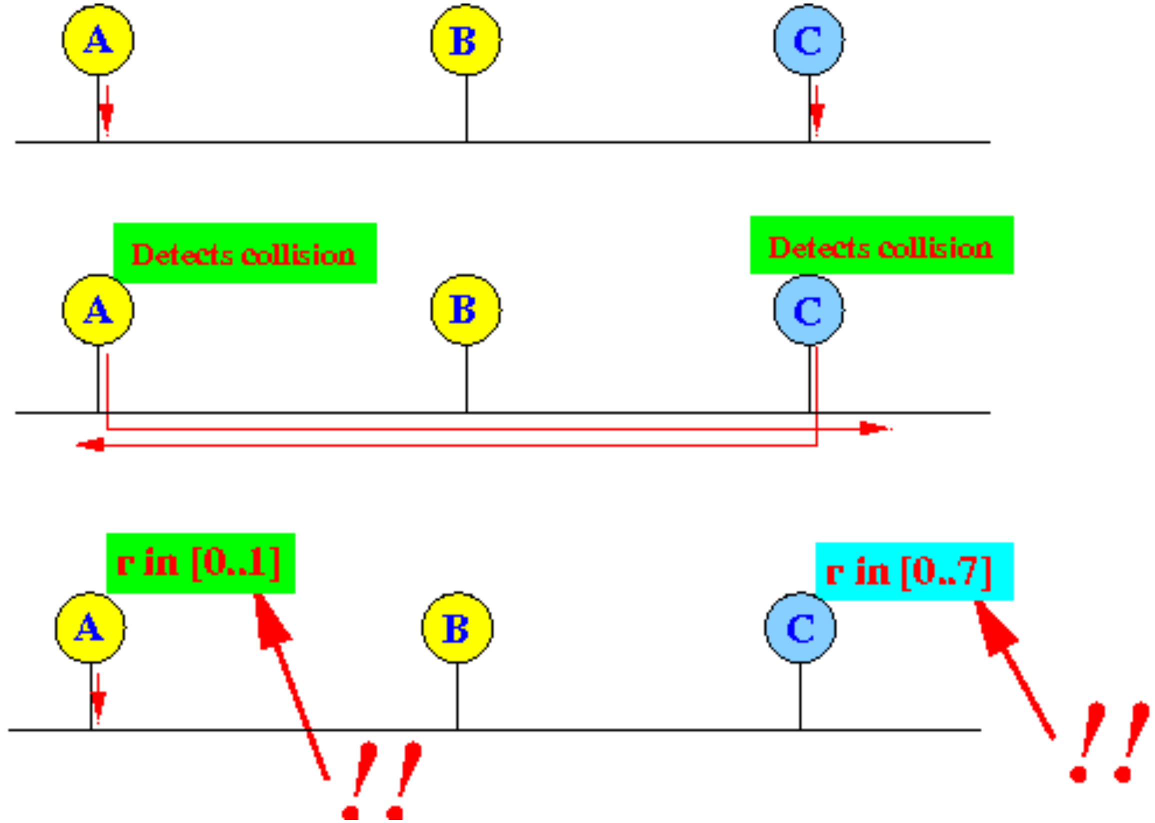
- *A* will transmit ***before C*** when
- *A* **picks 0** and *C* **picks 1**
- *A* **picks 0** and *C* **picks 2**
- *A* **picks 0** and *C* **picks 3**
- *A* **picks 1** and *C* **picks 2**
- *A* **picks 1** and *C* **picks 3**
- The **probability** that **C** will pick **any one number** is **0.25**The **probability** that **B** will pick **any one number 0.5**
- The chance that **C** will transmit **before B** is:
- $(0.25 \times 0.5) + (0.25 \times 0.5) + (0.25 \times 0.5) + (0.25 \times 0.5) + (0.25 \times 0.5) \quad = \quad 0.625$

Summary:

- If **node** *A* picks from **[0..1]** and **node** *C* picks from **[0..3]**, then
- The **probability** that node **A** transmits ***before*** node **C** = **0.625**
- The **probability** that node **C** will transmit ***before*** node **A** = **0.125**
- **Conclusion**
- This is ***not* fair** !!

And the **bad news** is:

- It will get **even worse** if **C** loses **again** ...
- **Suppose** node **B** transmits first and during **B's transmission** the node **A**, scheduled a transmission.
-



- The **likelihood** that **node C** will **transmit first** is now **very very *small*** !!!
- **NOTE**
- This **problem** is **very well-known**, and it is called the
- **Ethernet Capture Effect**