Achieving Network Consistency

Octav Chipara

Reminders

- **Homework is postponed until next class**
    - if you already turned in your homework, you may resubmit

- **Please send me your peer evaluations**

Next few lectures

- **Start building a wireless stack from the ground up**
  - already covered
    - phy properties
    - mac layer
  - today:
    - network consistency
    - next class: Prof. Ted Herman will talk about timesync
  - future lectures
    - network consistency
    - link estimation
    - topology control
    - routing

Problem formulation

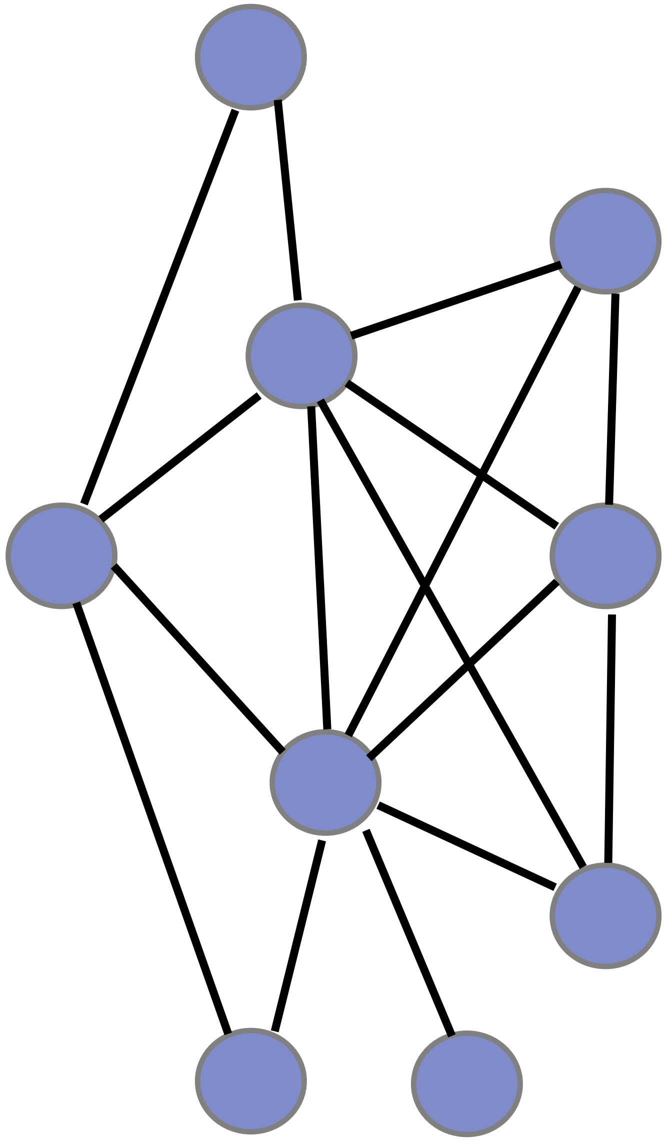
- **Consistency is a foundation for many network protocols**
  - routing tree maintenance => next hop has lower cost
  - network configuration => all nodes have the most recent configuration
  - neighborhood maintenance => a node in all its neighbor’s lists

- **Goal: when a node updates/generates a new piece of data, this information must be relayed to all other nodes**
  - minimize the number of redundant transmissions
    (i.e., a node should receive a packet only once)

  - scales well with network size and density

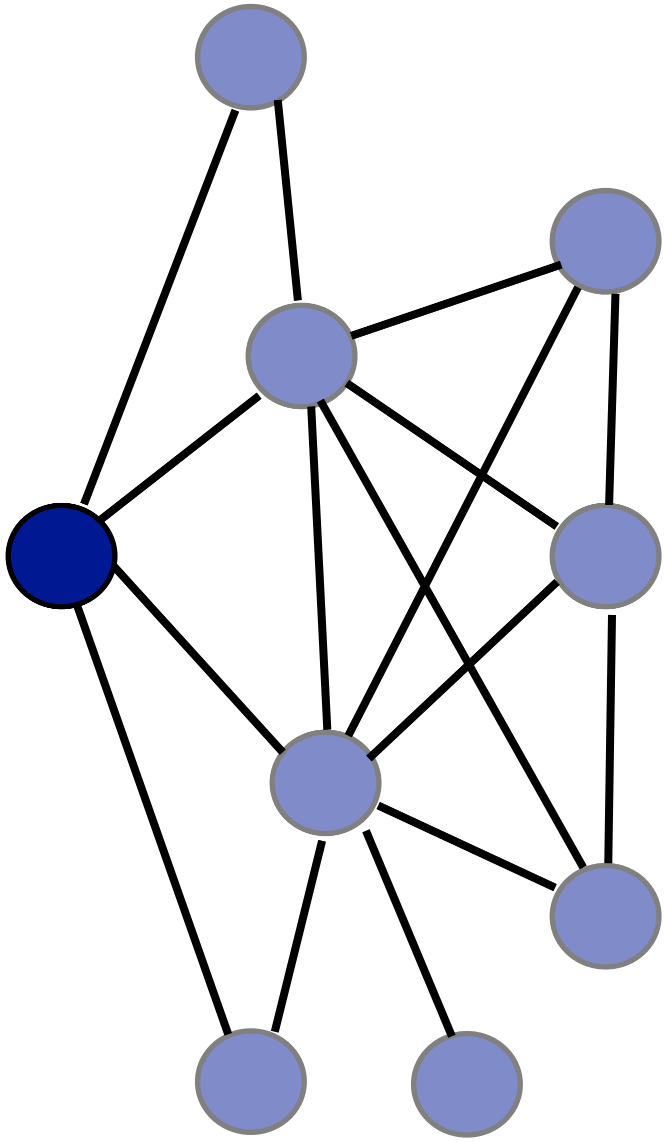
Basic approach - Flooding

- Upon hearing new data a node rebroadcasts it ➜ eventual consistency
- Challenge: wireless is a broadcast medium ➜ ***broadcast storm***



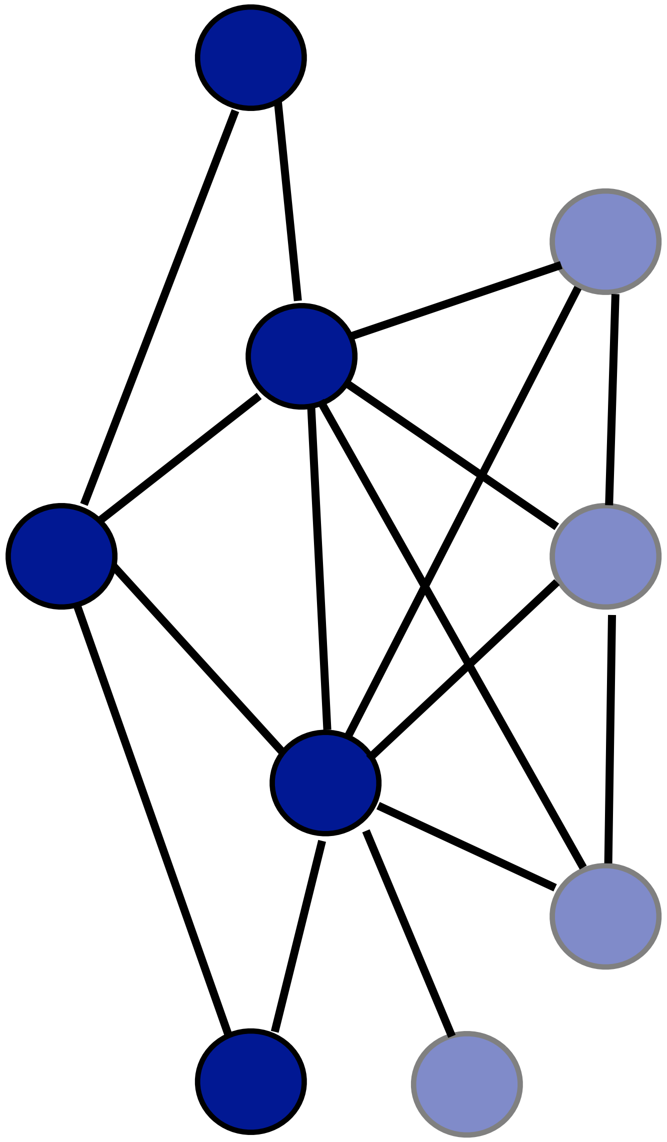
Basic approach - Flooding

- Upon hearing new data a node rebroadcasts it ➜ eventual consistency
- Challenge: wireless is a broadcast medium ➜ ***broadcast storm***



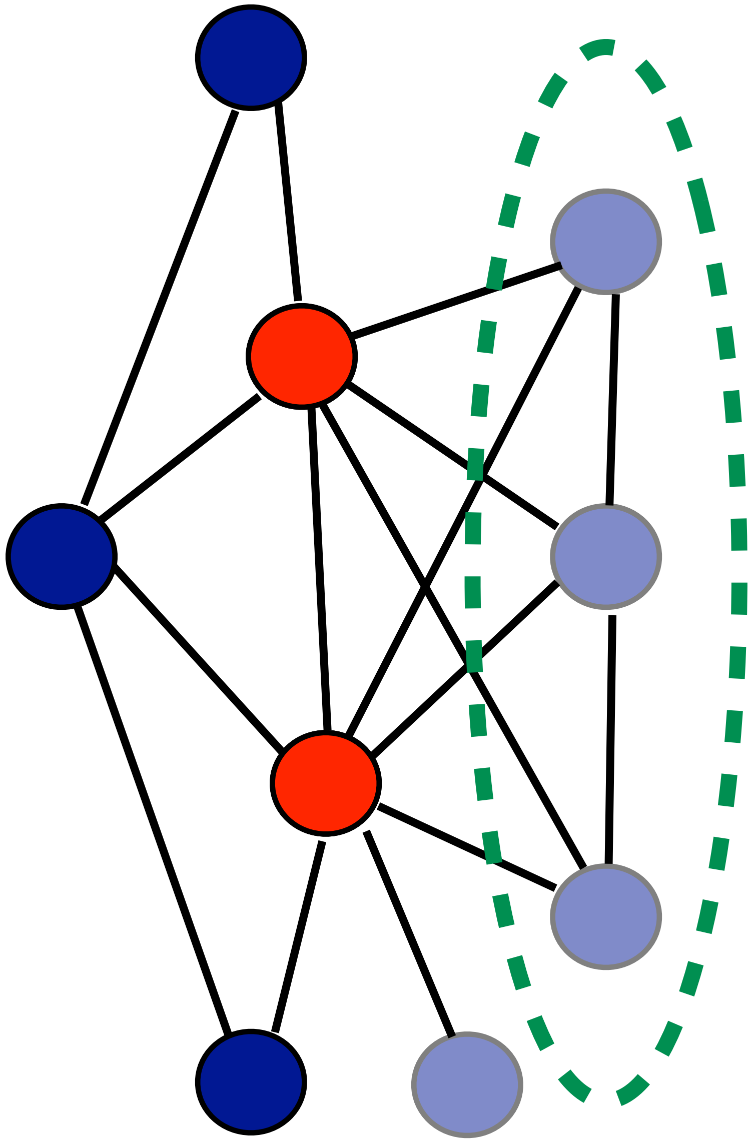
Basic approach - Flooding

- Upon hearing new data a node rebroadcasts it ➜ eventual consistency
- Challenge: wireless is a broadcast medium ➜ ***broadcast storm***



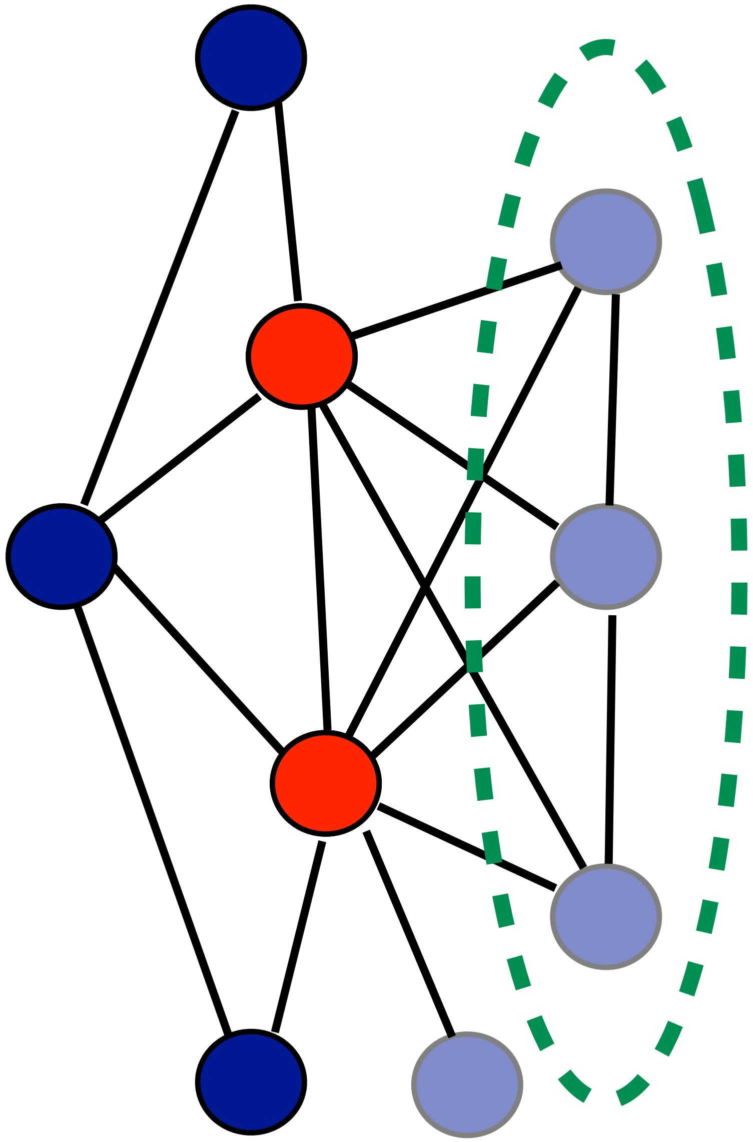
Basic approach - Flooding

- Upon hearing new data a node rebroadcasts it ➜ eventual consistency
- Challenge: wireless is a broadcast medium ➜ ***broadcast storm***



Basic approach - Flooding

- Upon hearing new data a node rebroadcasts it ➜ eventual consistency
- Challenge: wireless is a broadcast medium ➜ ***broadcast storm***



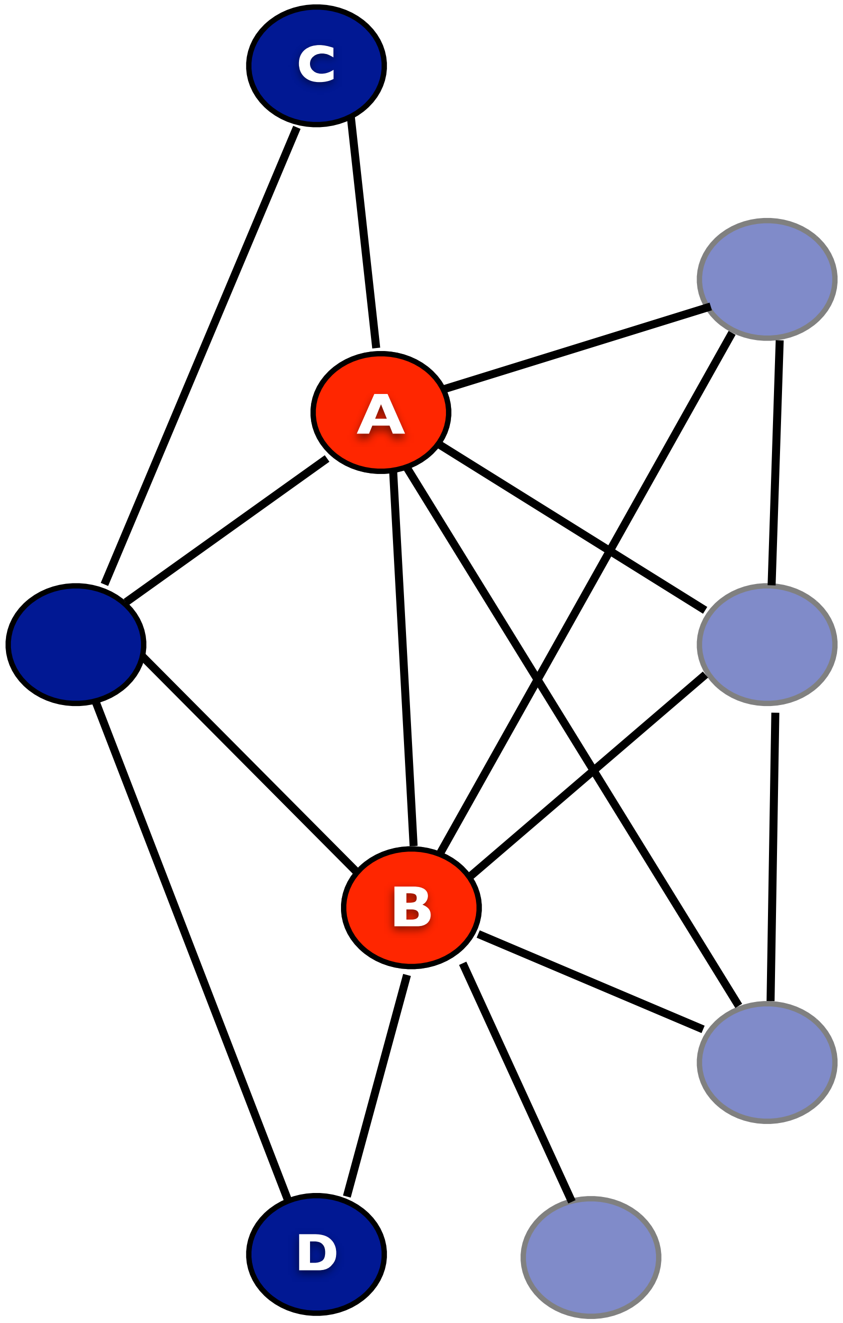
Broadcast storm: every tries to transmit at the same time resulting in numerous collisions

Mitigating the broadcast storm problem

- **Randomized delays: introduce delays before packet transmissions**
  - reduces the likelihood of packet collisions
  - however, it is often hard to determine the optimal delays
    - depends on the the “local node density”

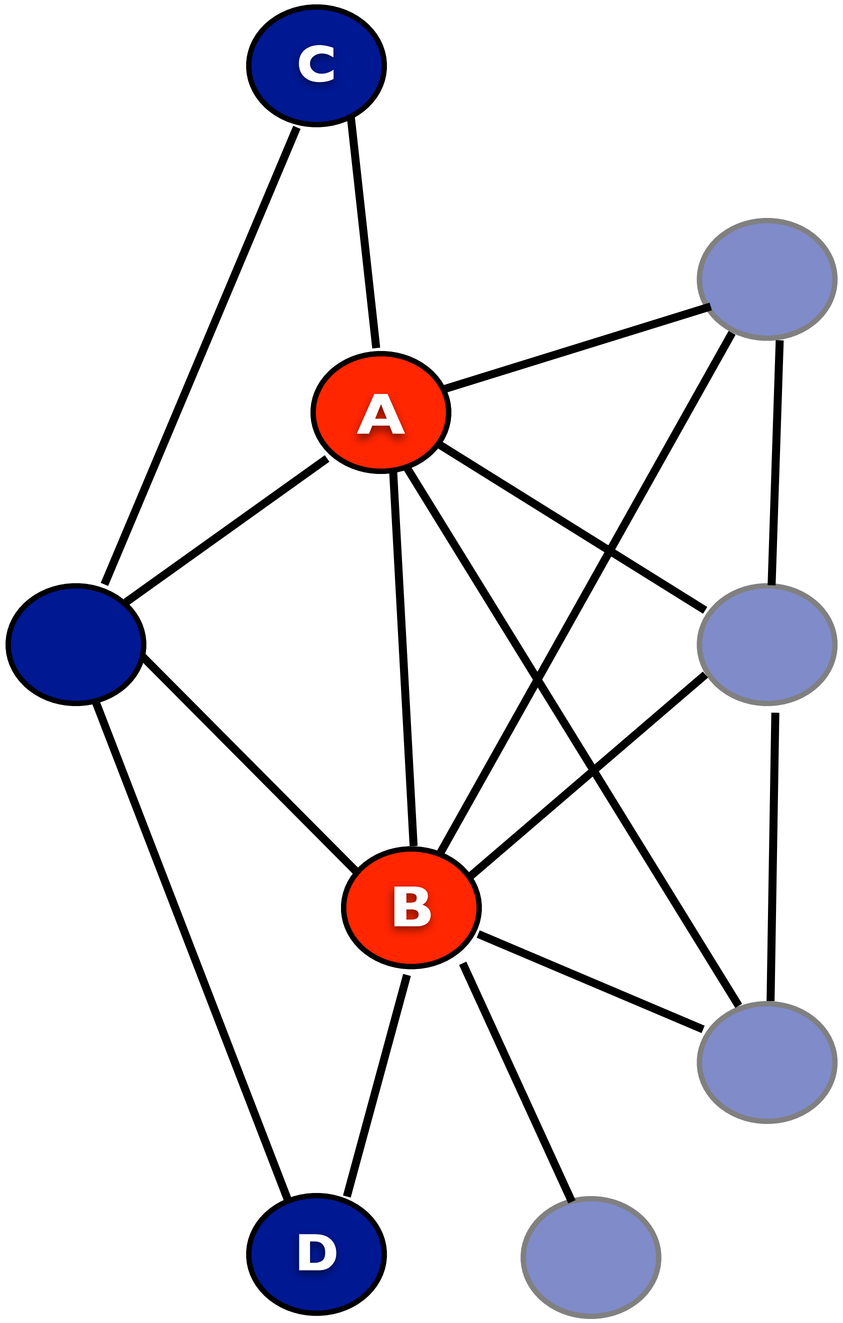
- **Transmission suppression: some nodes do not need to transmit**
  - a node that hears the same data from several neighbors stops transmitting
  - reduces the number of contending nodes
  - however, it may prolong the time to propagate the message

Randomized delays



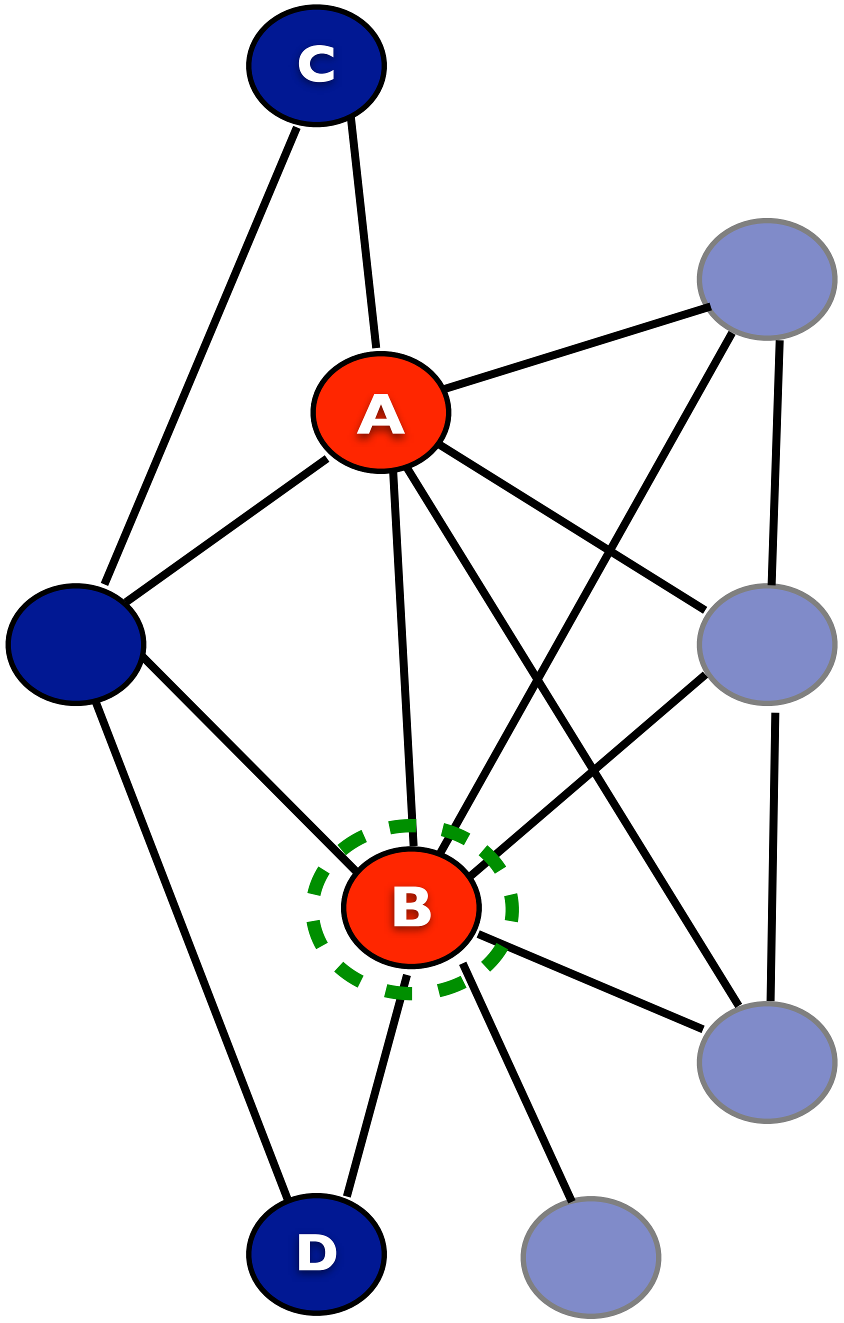
- **Reduces likelihood of collisions**
  - nodes A, B, C, D transmit at different times
- **Still inefficient**
  - node C and D should not transmit
  - node A and B share many neighbors

Transmission suppression



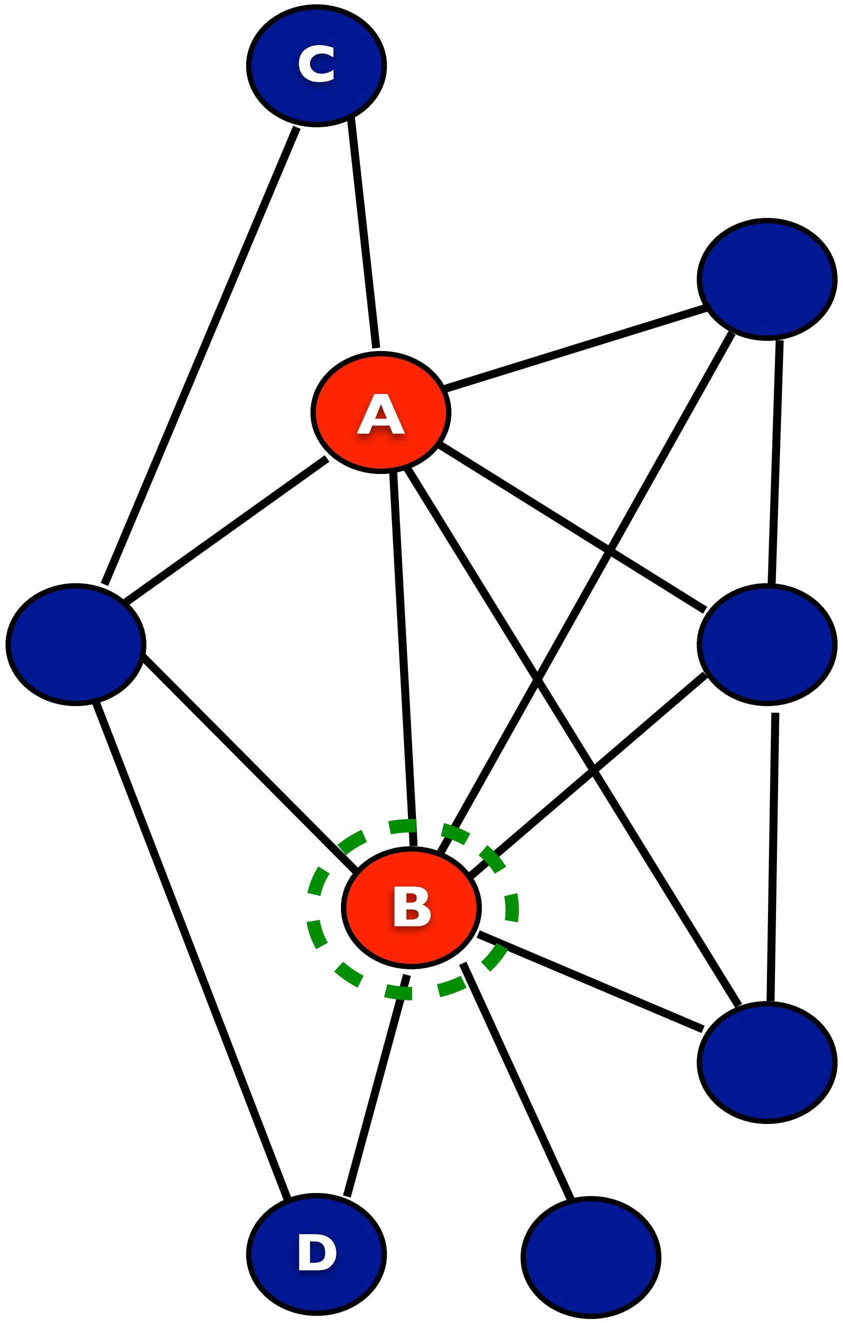
- **Transmission suppression**
  - reduces the number of contenders
  - potential for significant savings
- **Knowing more information may help you make better decisions**
  - e.g., two hop neighborhood info

Transmission suppression



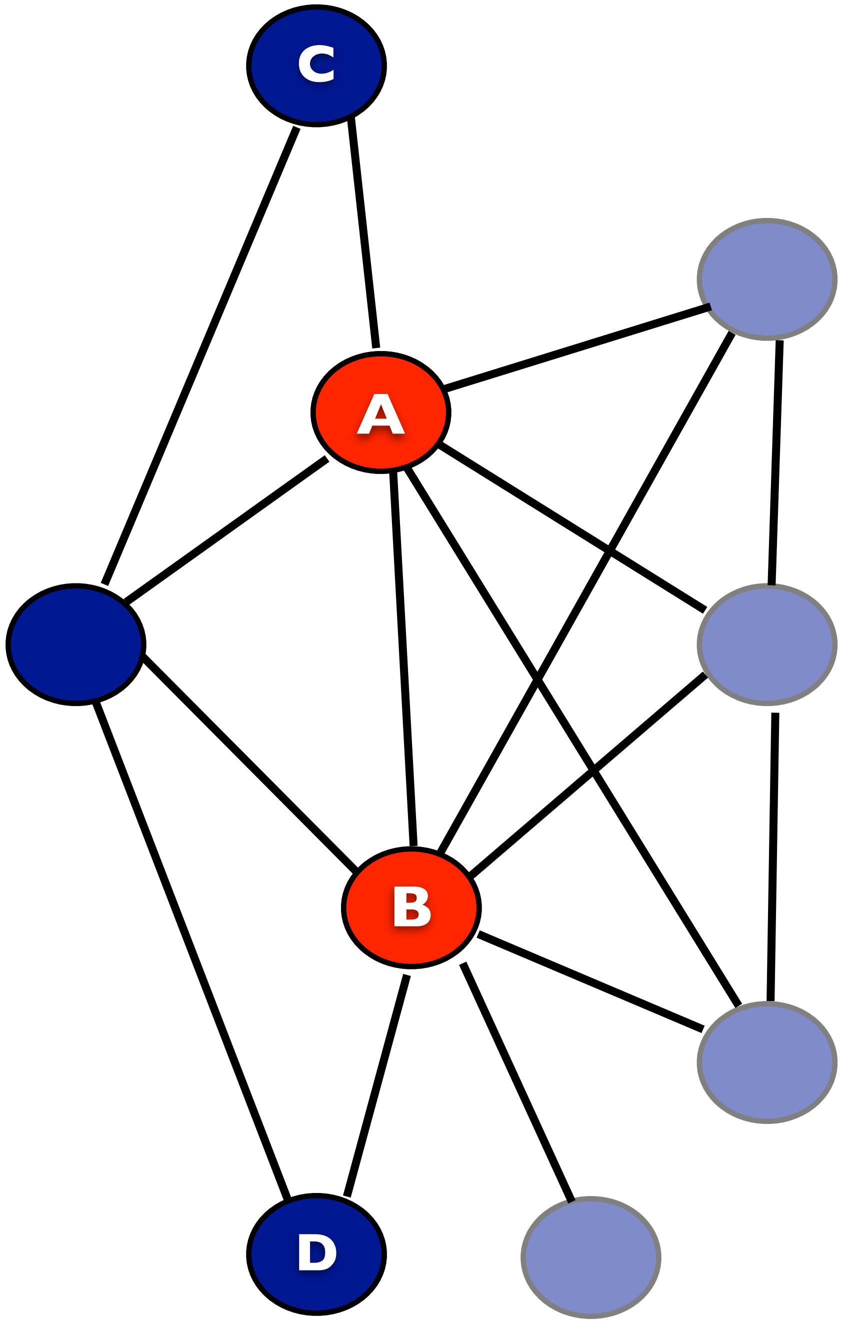
- **Transmission suppression**
  - reduces the number of contenders
  - potential for significant savings
- **Knowing more information may help you make better decisions**
  - e.g., two hop neighborhood info

Transmission suppression



- **Transmission suppression**
    - reduces the number of contenders
    - potential for significant savings
- **Knowing more information may help you make better decisions**
    - e.g., two hop neighborhood info

Transmission suppression

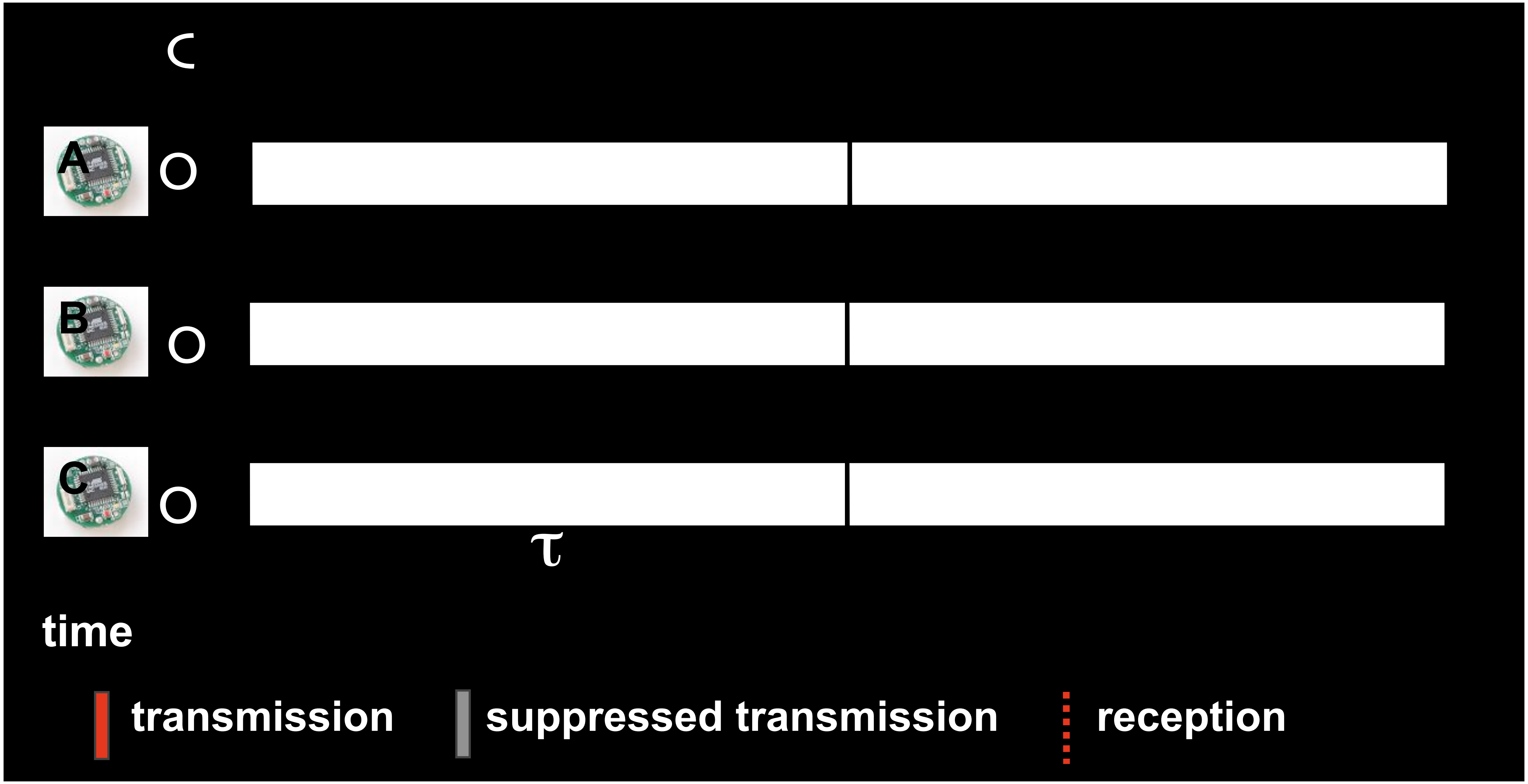


- **Suppressing wrong transmissions will increase propagation delays**
  - e.g., suppressing A and B stops progress

Trickle - algorithm outline

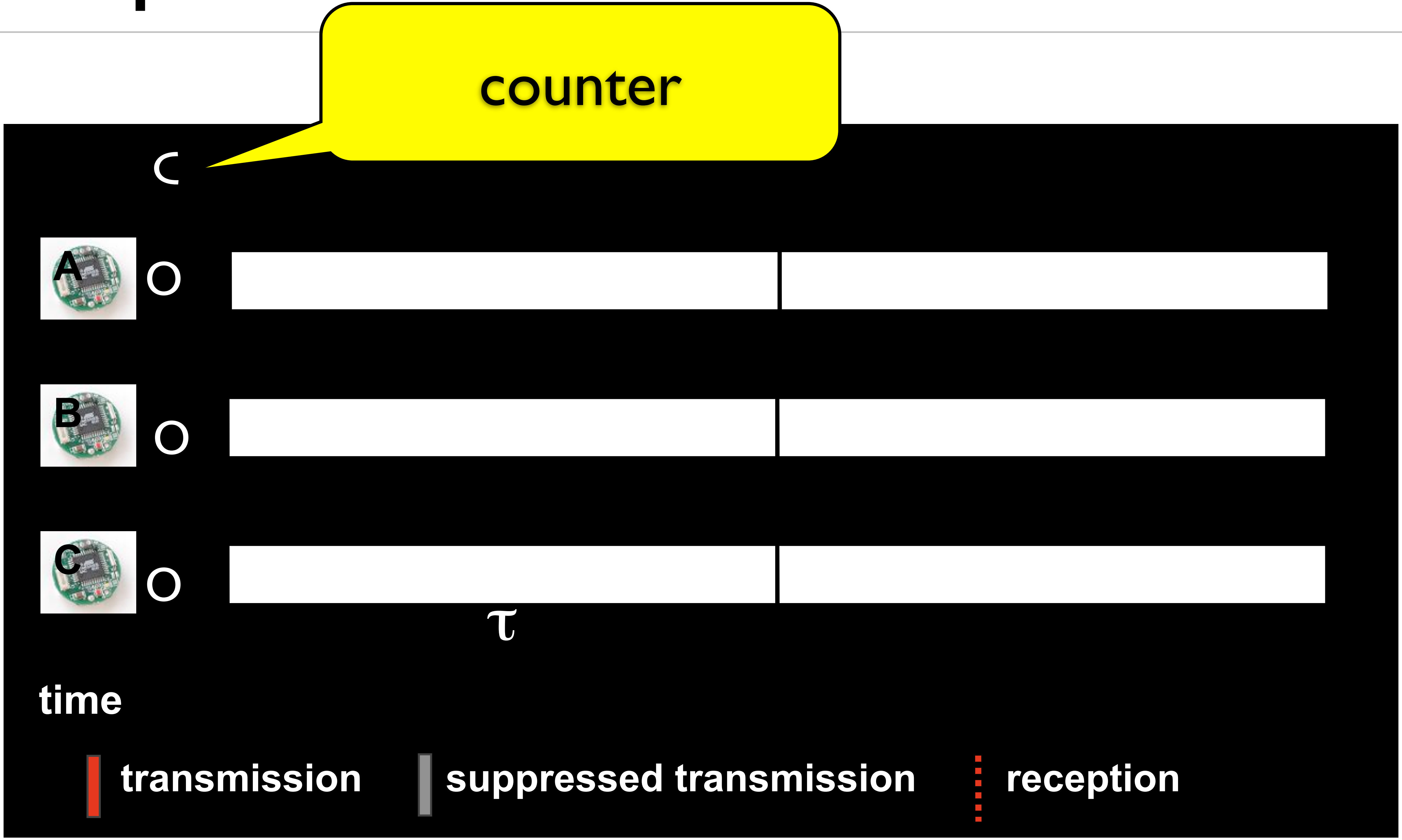
- **Divides the time into intervals, nodes are synchronized**
    - a node transmits **metadata** in each interval
- **In response to a change in metadata**
    - a node picks a random time in its current interval *t* to transmit its data
        - let c be the number of times a node hears a data item
        - if c < threshold, then node transmits the data item
        - else, transmission is suppressed

Example


A
0
B
0
C
0
τ
time
transmission
suppressed transmission
reception

k = 1

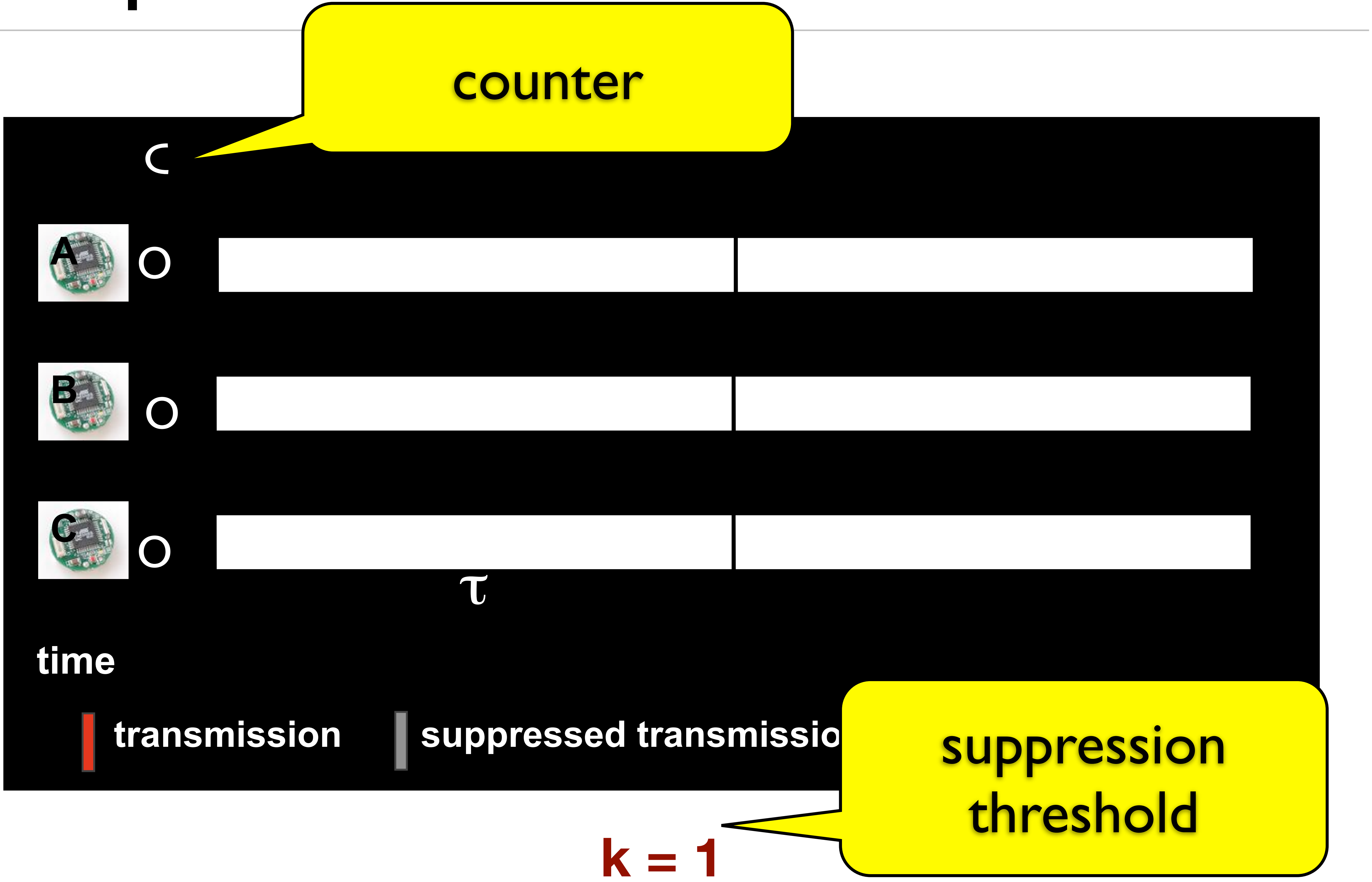
Example



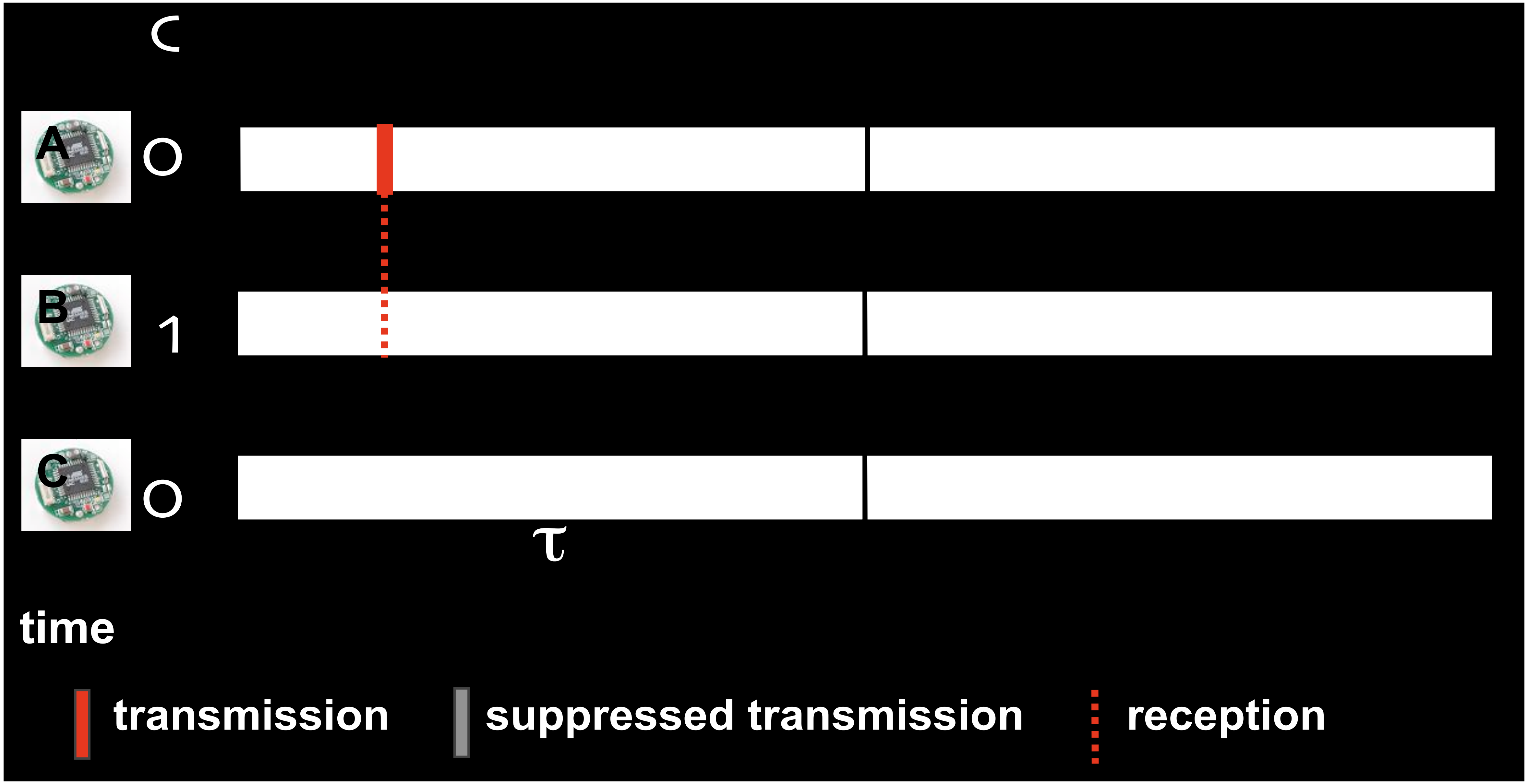
counter

k = 1

Example

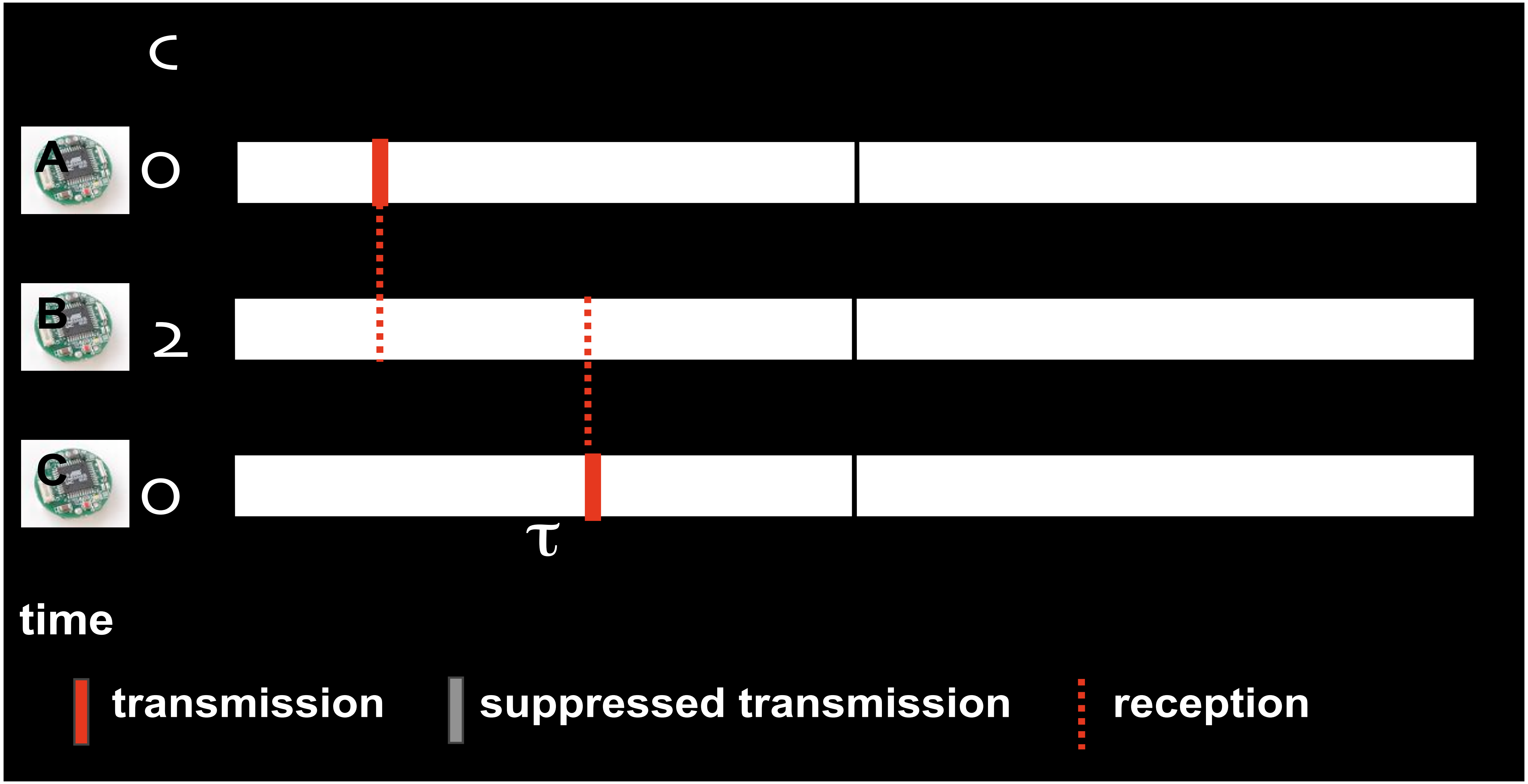

counter
c
A 0
B 0
C 0
τ
time
🟥 transmission
▨ suppressed transmissio
suppression threshold
k = 1

Example


c
A
0
B
1
C
0
τ
time
transmission
suppressed transmission
reception

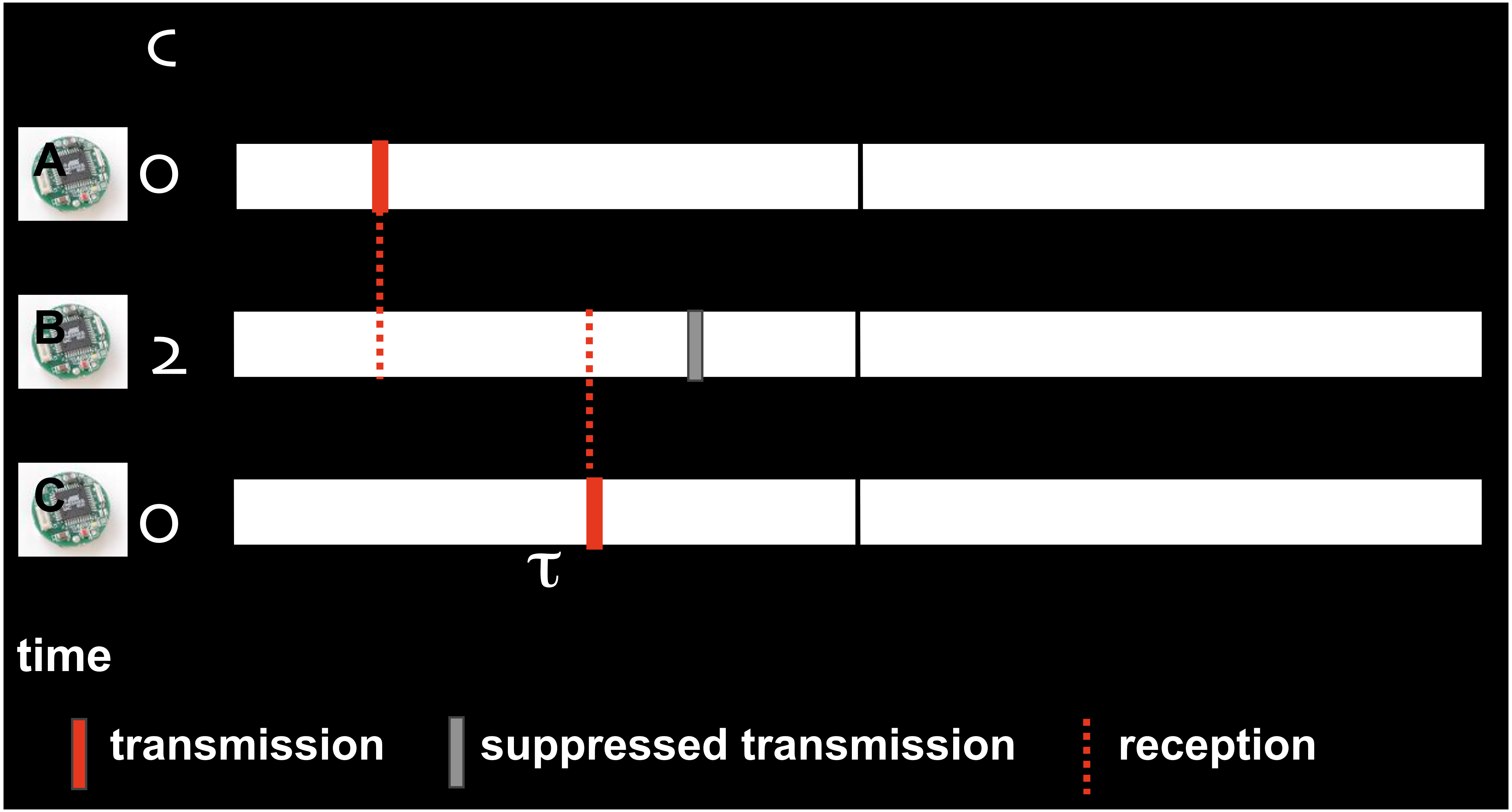
k = 1

Example


c
A
0
B
2
C
0
τ
time
transmission
suppressed transmission
reception

k = 1

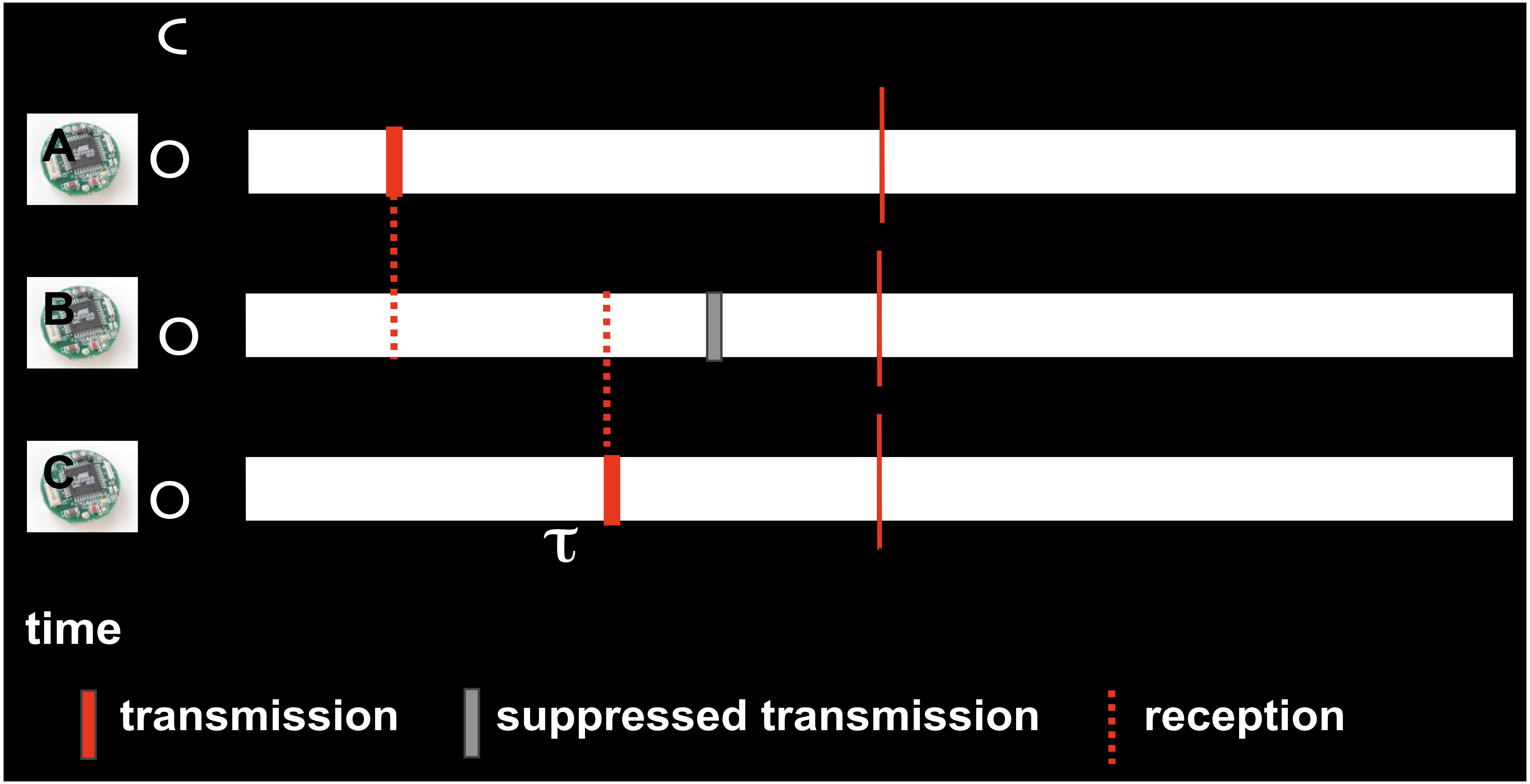
Example



🟥 transmission ▮ suppressed transmission ⋮ reception

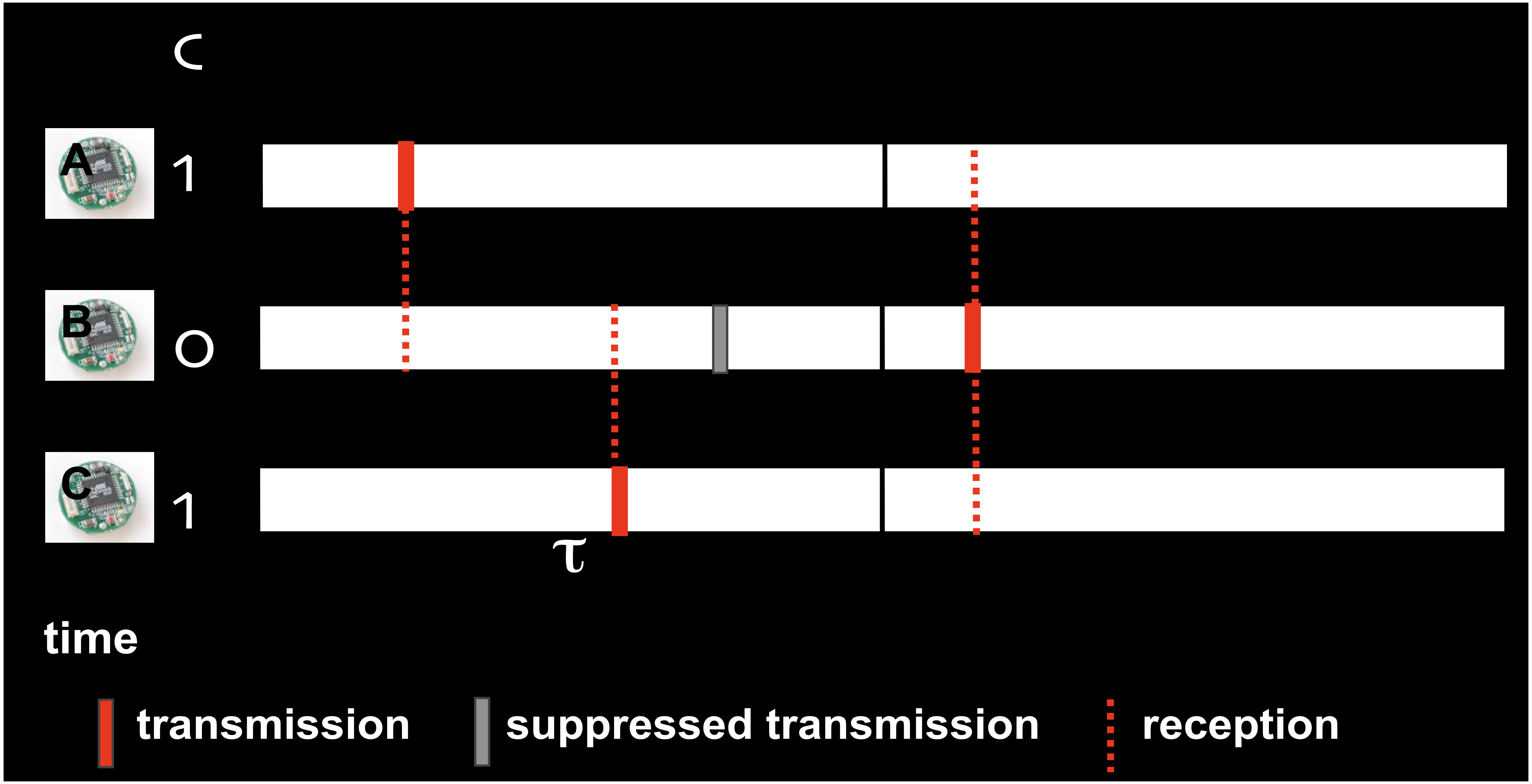
k = 1

Example


c
A
O
B
O
C
O
τ
time
transmission
suppressed transmission
reception

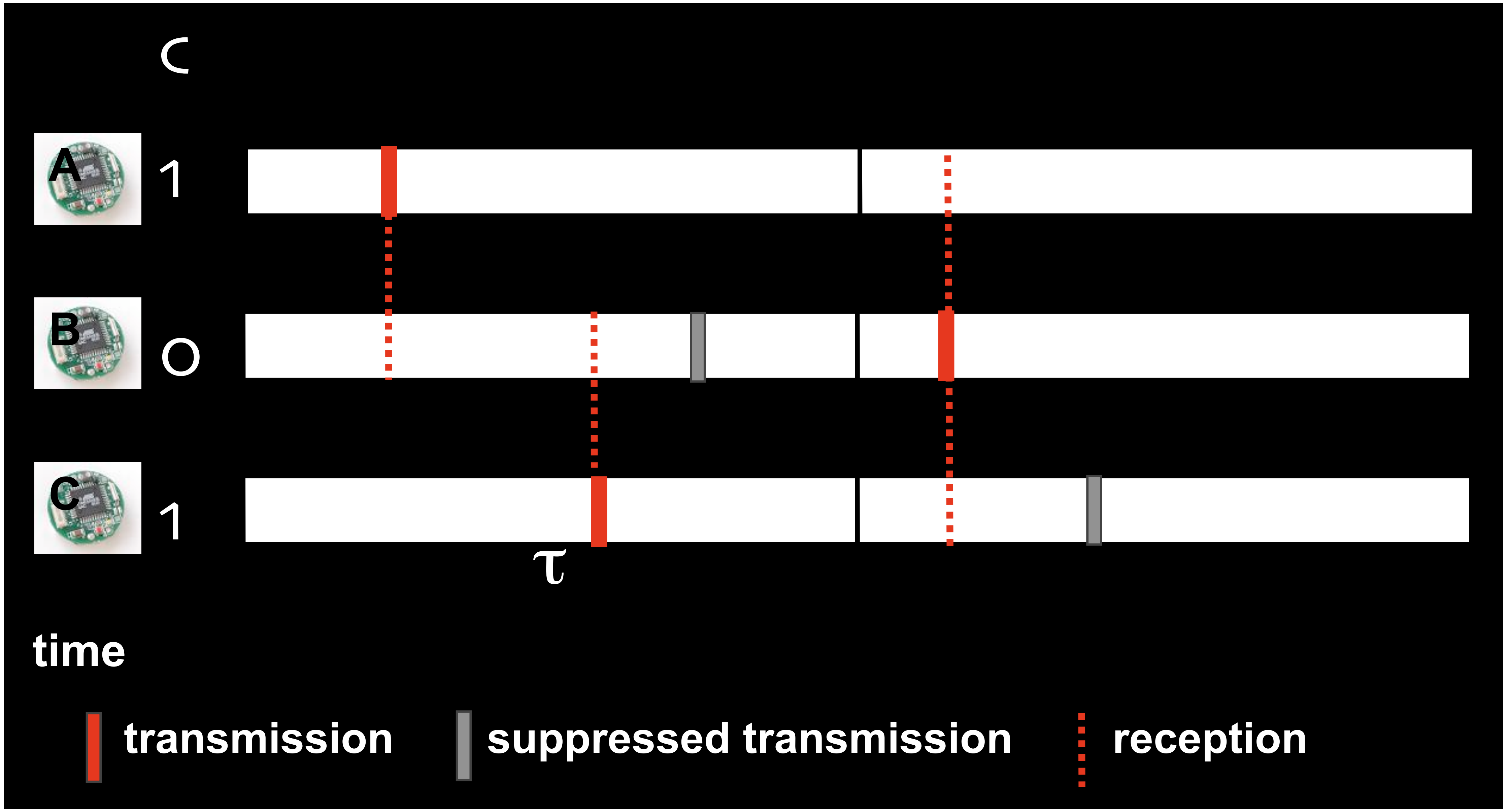
k = 1

Example


c
A
1
B
0
C
1
τ
time
transmission
suppressed transmission
reception

k = 1

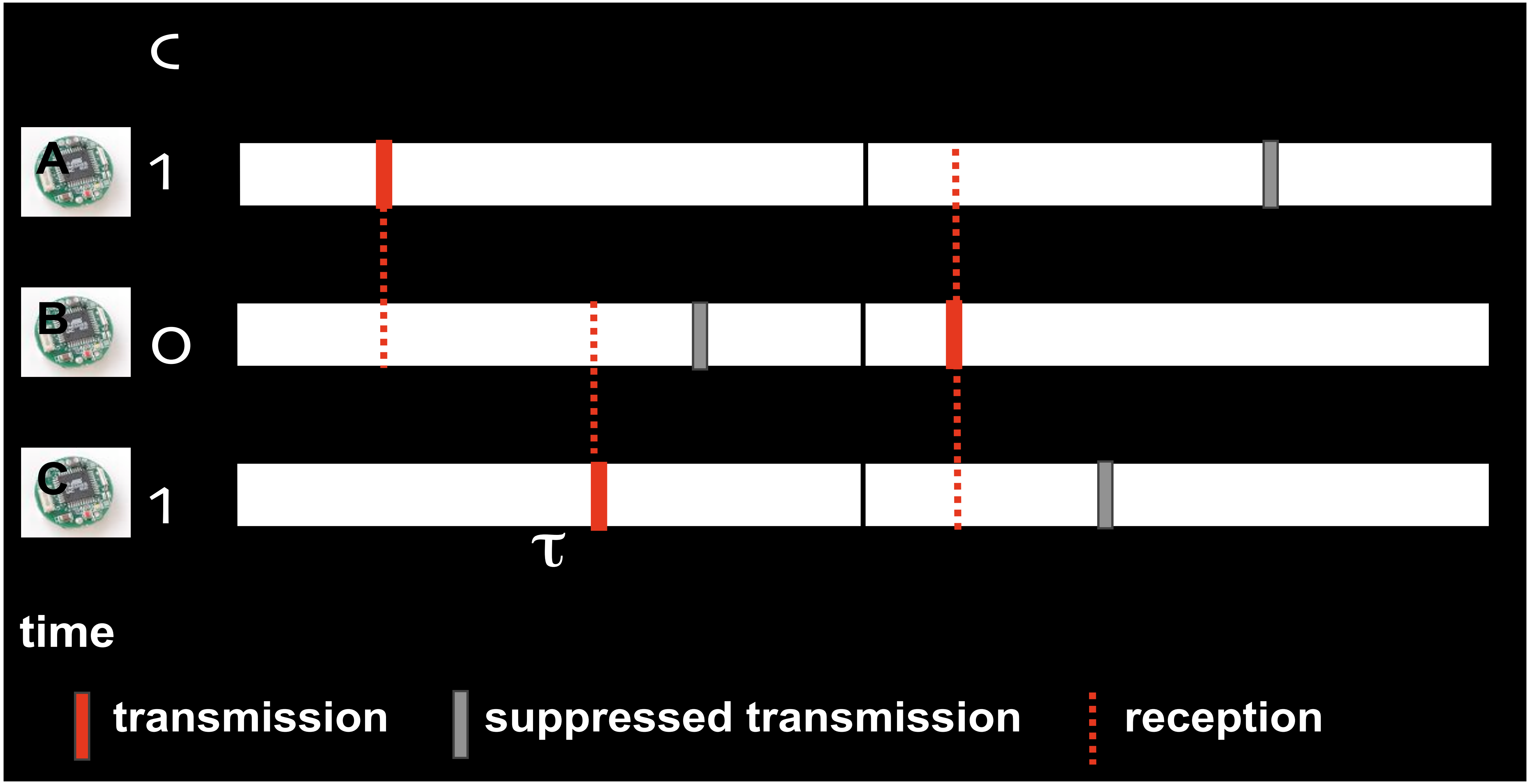
Example



🟥 transmission ▨ suppressed transmission ⋮ reception

k = 1

Example

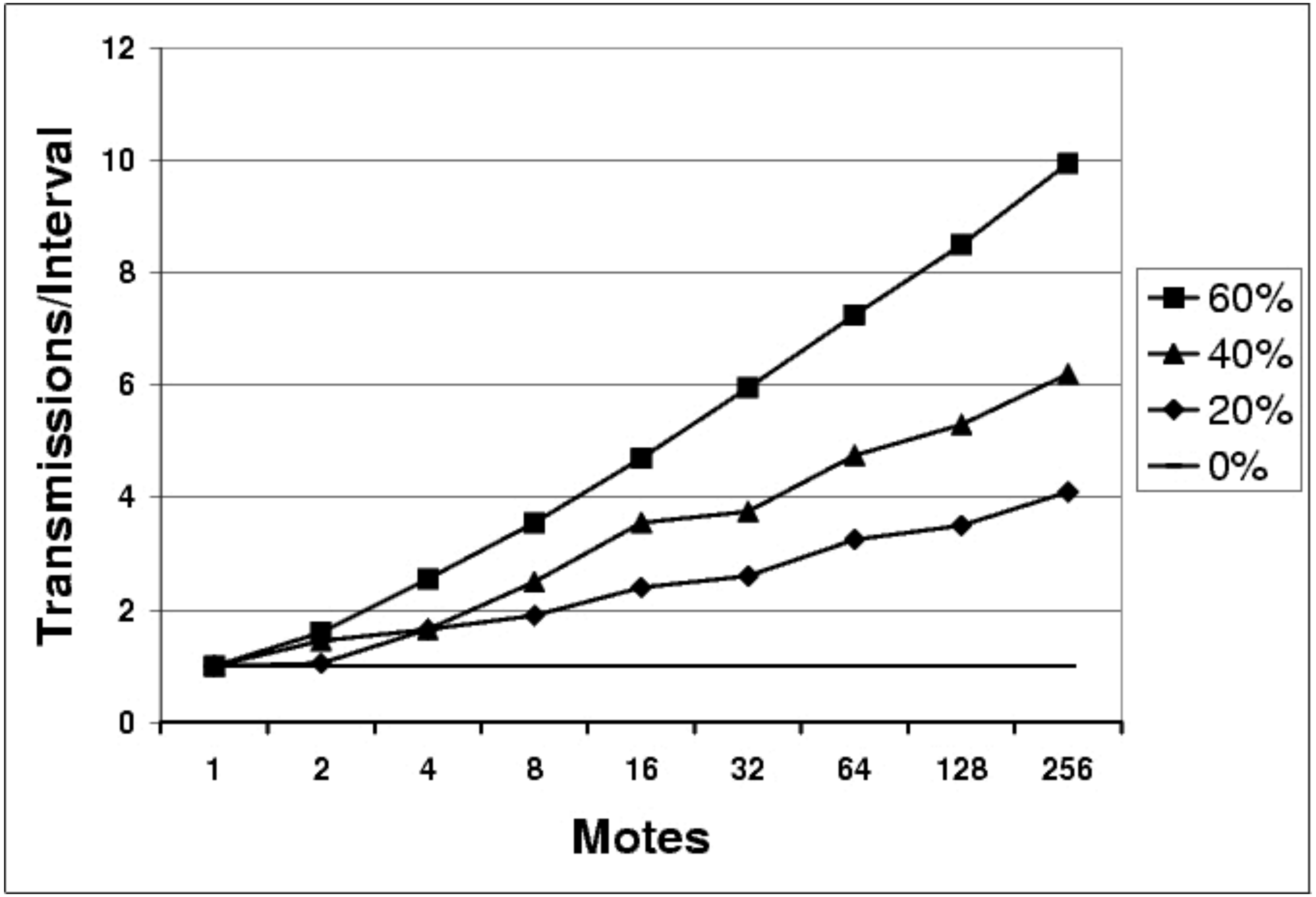

c
A
1
B
0
C
1
τ
time
transmission
suppressed transmission
reception

k = 1

Trickle - features

- **Managing protocol overhead**
    - it is wasteful to transmit state information when nothing changes
    - insight:
        - upon a change ➜ metadata should be transmitted fast
        - after a change ➜ rate of transmitting metadata should decrease
    - solution:
        - exponentially decrease the metadata when state is consistent
        - reset the rate of transmitting metadata upon hearing new data
- **Suppression based on number of overhead packets**
    - relies on minimal topological information ➜ tolerates frequent changes in topology

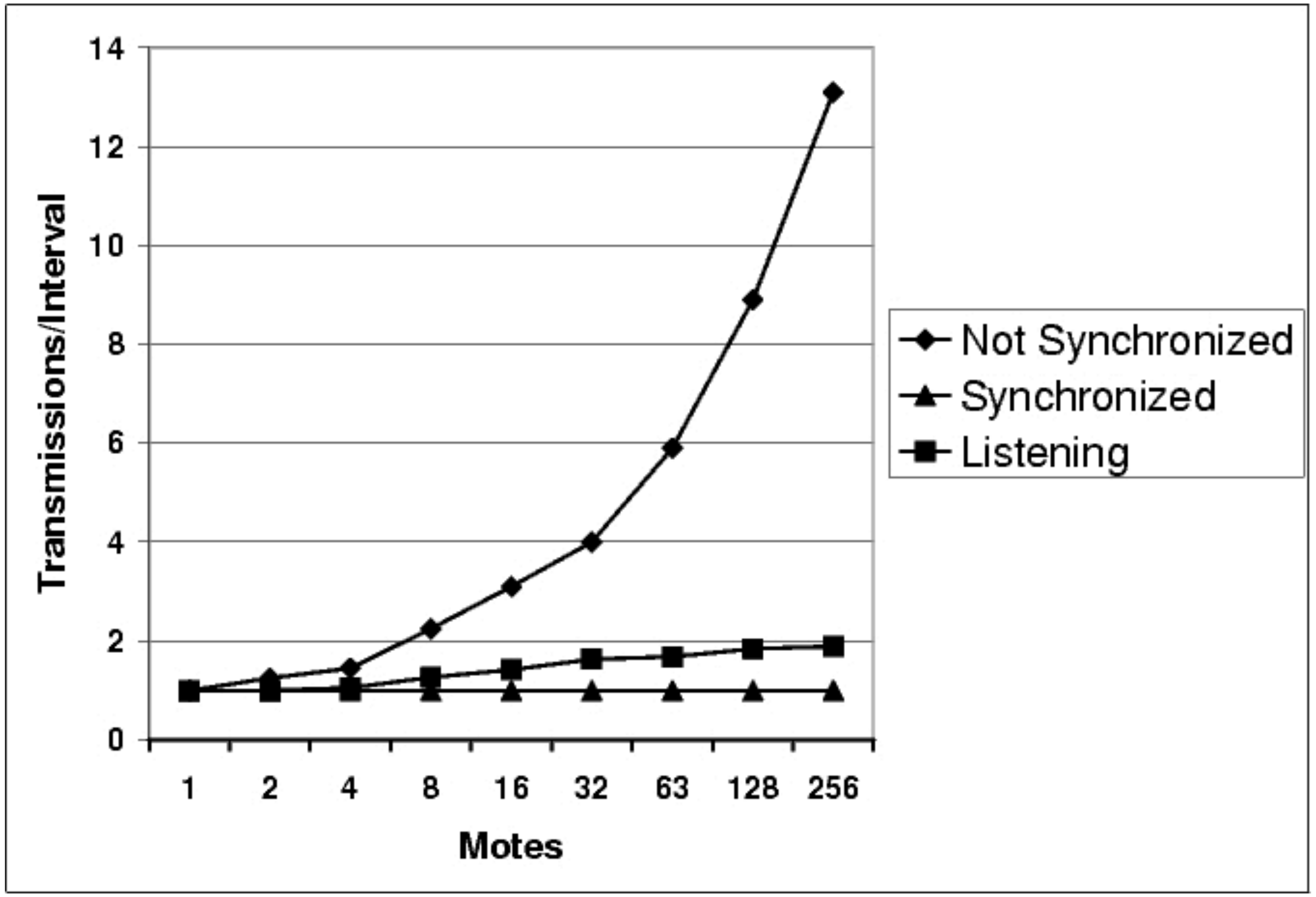
Impact of packet losses [Single hop network]



The number of rounds scale with O(log(n))

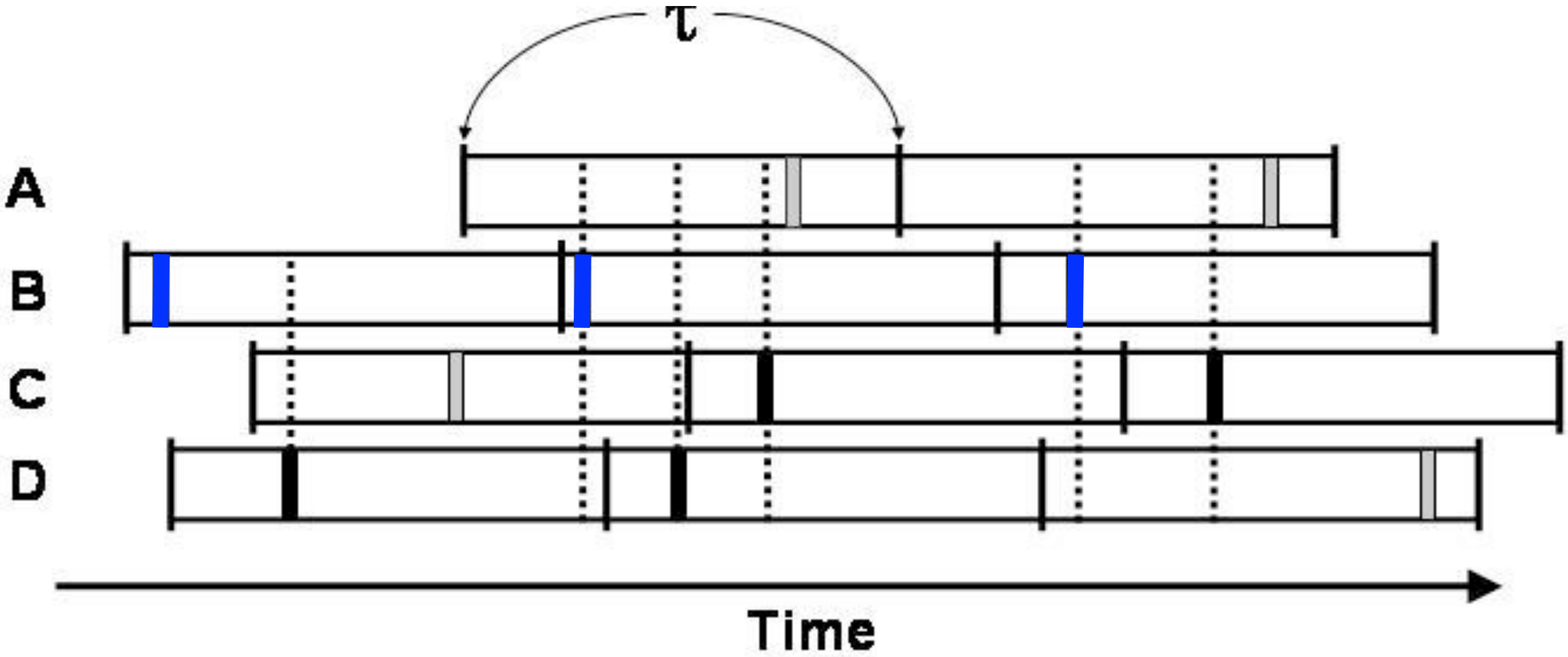
Tickle without synchronization

- **Remove the requirement of nodes operating synchronized**
  - each node operates independently
  - the intervals are not aligned anymore



New problem: short-listening

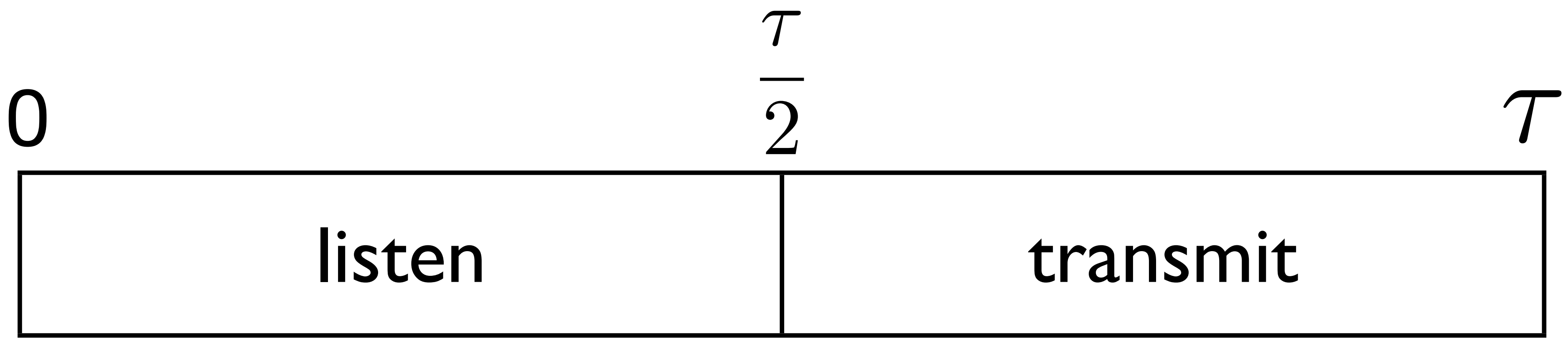
Short-listening problem



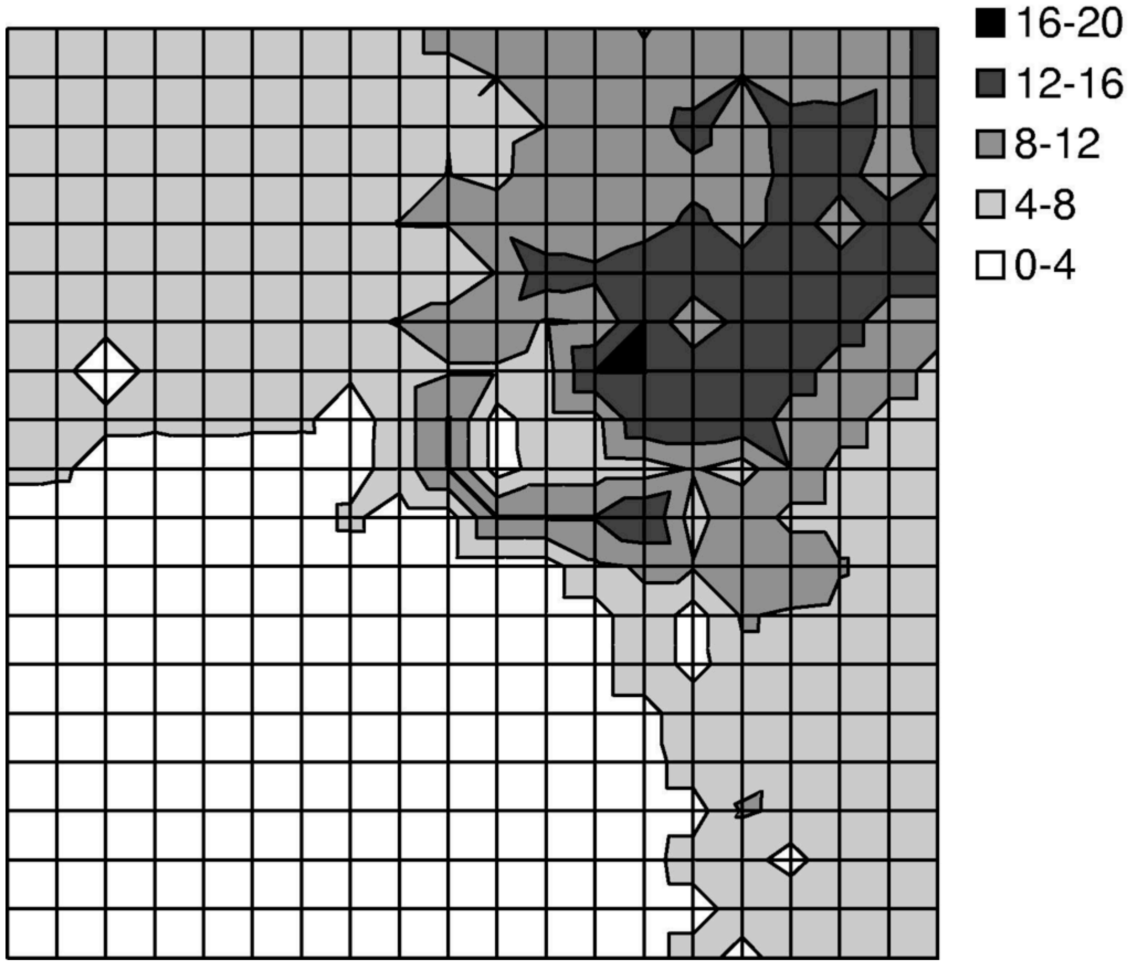
- **B transmits soon after the start of each interval**
    - reduce likelihood for its transmissions to be suppressed

Solution to the short-listen problem

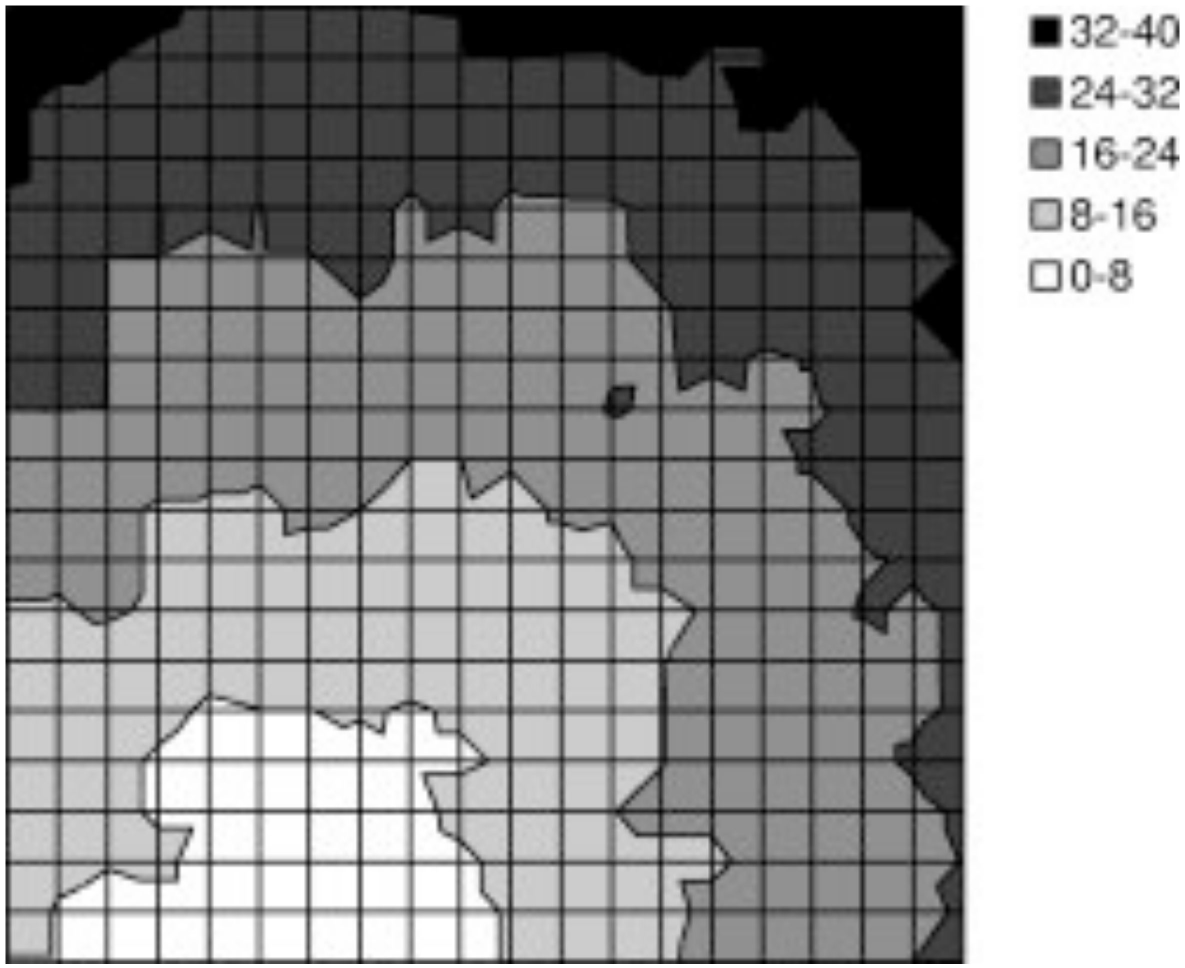
- **Divide a slot in two parts**
  - listen only - nodes only listen during this part of the interval
  - transmit part - nodes transmit randomly within this interval



Simulation results



5’ spacing, 6 hops



20’ spacing, 40 hops

How could we improve Trickle?

- Take advantage of the spatial correlation of packets
- Differentiate between “stable” and “unstable” neighbors