Mobile and Sensor Systems

Lecture 6: Sensor Network Reprogramming and Mobile Sensors

Dr Sarfraz Nawaz

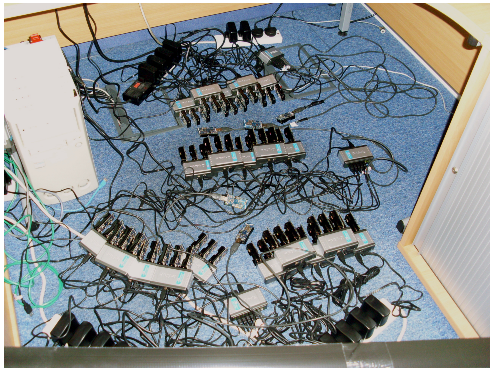
UNIVERSITY OF CAMBRIDGE

In this lecture

- We will describe techniques to reprogram a sensor network while deployed.
- We describe briefly mobile sensor networks and mobile sensor network reprogramming.

UNIVERSITY OF CAMBRIDGE

Sensor Network Programming/Reprogramming



- Long Lifetime requires retasking the sensors.
- However programming each node separately may not be feasible.
- What is reprogramming?
  - Send function parameters ("wake up every X seconds").
  - Sending binaries or code to compile.
- Checking that each node has the right code can be quite costly too.

UNIVERSITY OF CAMBRIDGE

Idea

- The first step is to detect when nodes need updates (continuous process).
- When there is no new code:
  - **Maintenance** cost should approach zero
- When there is new code.
  - **Propagation** should be rapid.

UNIVERSITY OF CAMBRIDGE

Trickle

- Simple, "polite gossip" algorithm.
- "Every once in a while, broadcast what code you have, unless you've heard some other nodes broadcast the same thing, in which case, stay silent for a while."

UNIVERSITY OF CAMBRIDGE

Trickle

- Within a node time period:
  - If a node hears older metadata, it broadcasts the new data.
  - If a node hears newer metadata, it broadcasts its own metadata (which will cause other nodes to send the new code).
  - If a node hears the same metadata, it increases a counter: If a threshold is reached, the node does not transmit its metadata. Otherwise, it transmits its metadata.

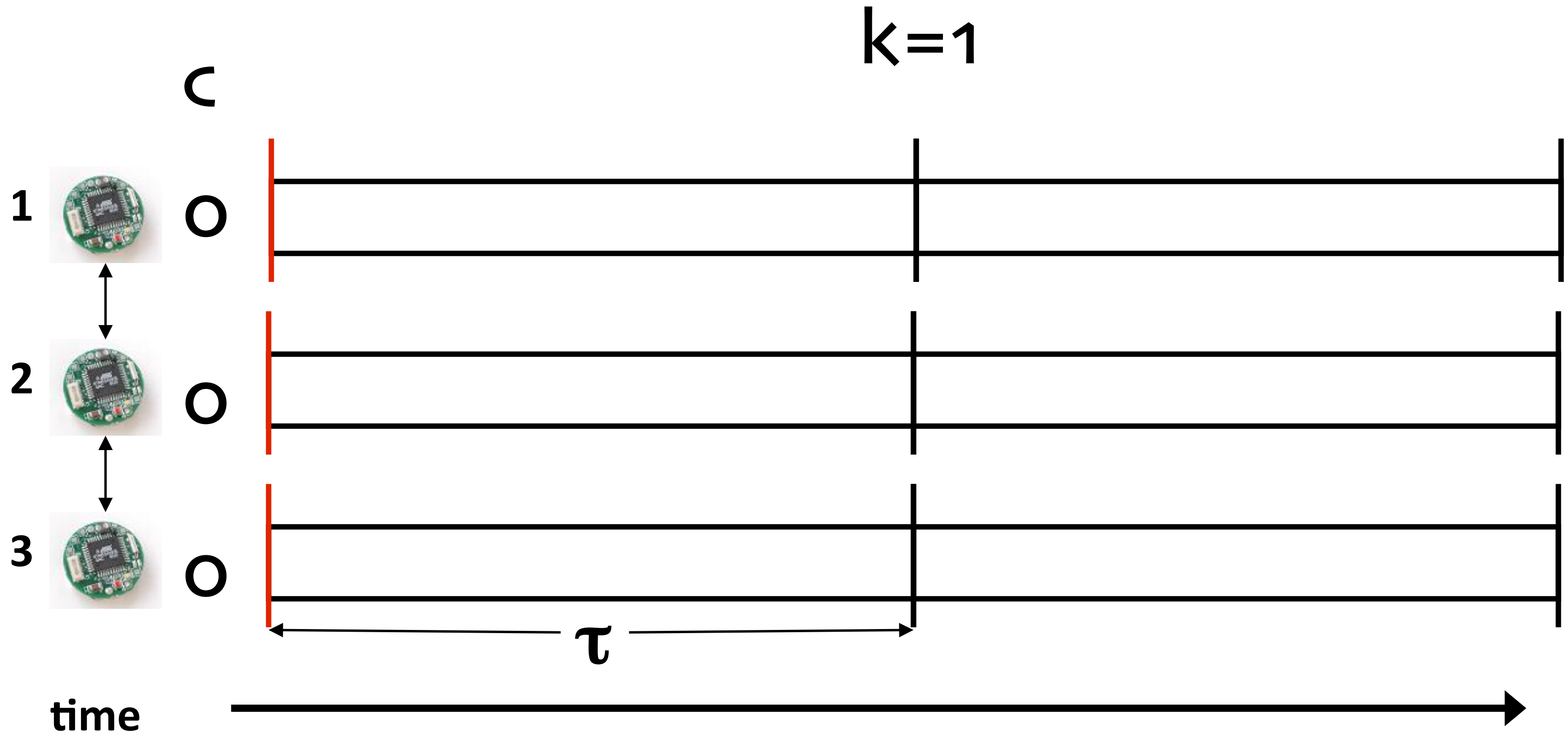
UNIVERSITY OF CAMBRIDGE

Trickle – Main Parameters

- Counter ***c***: Count how many times identical metadata has been heard
- ***k***: threshold to determine how many times identical metadata must be heard before suppressing transmission of a node's metadata
- ***t***: the time at which a node will transmit its metadata. t is in the range of ***[0, τ]***

UNIVERSITY OF CAMBRIDGE

Example Trickle Execution



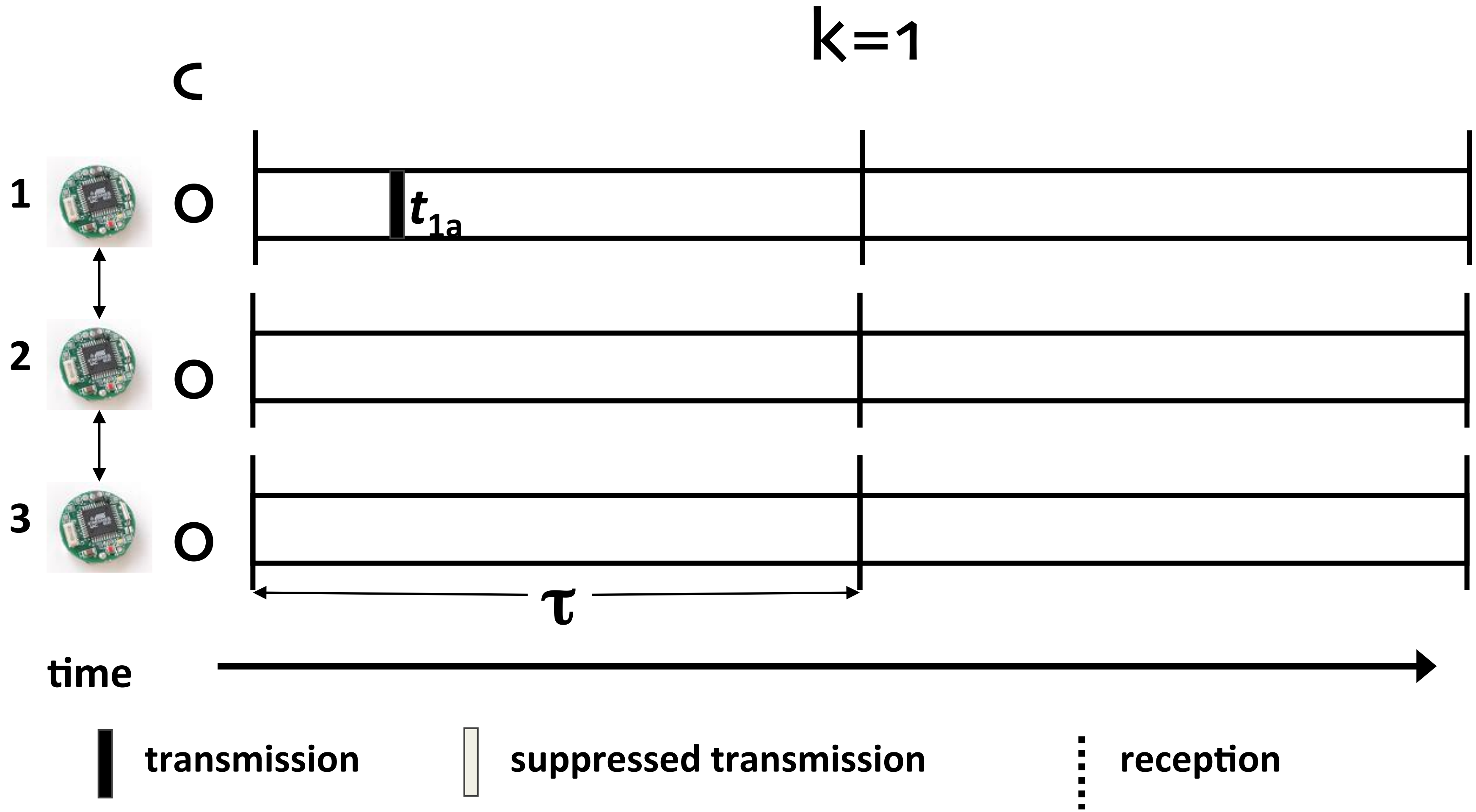
transmission

suppressed transmission

reception

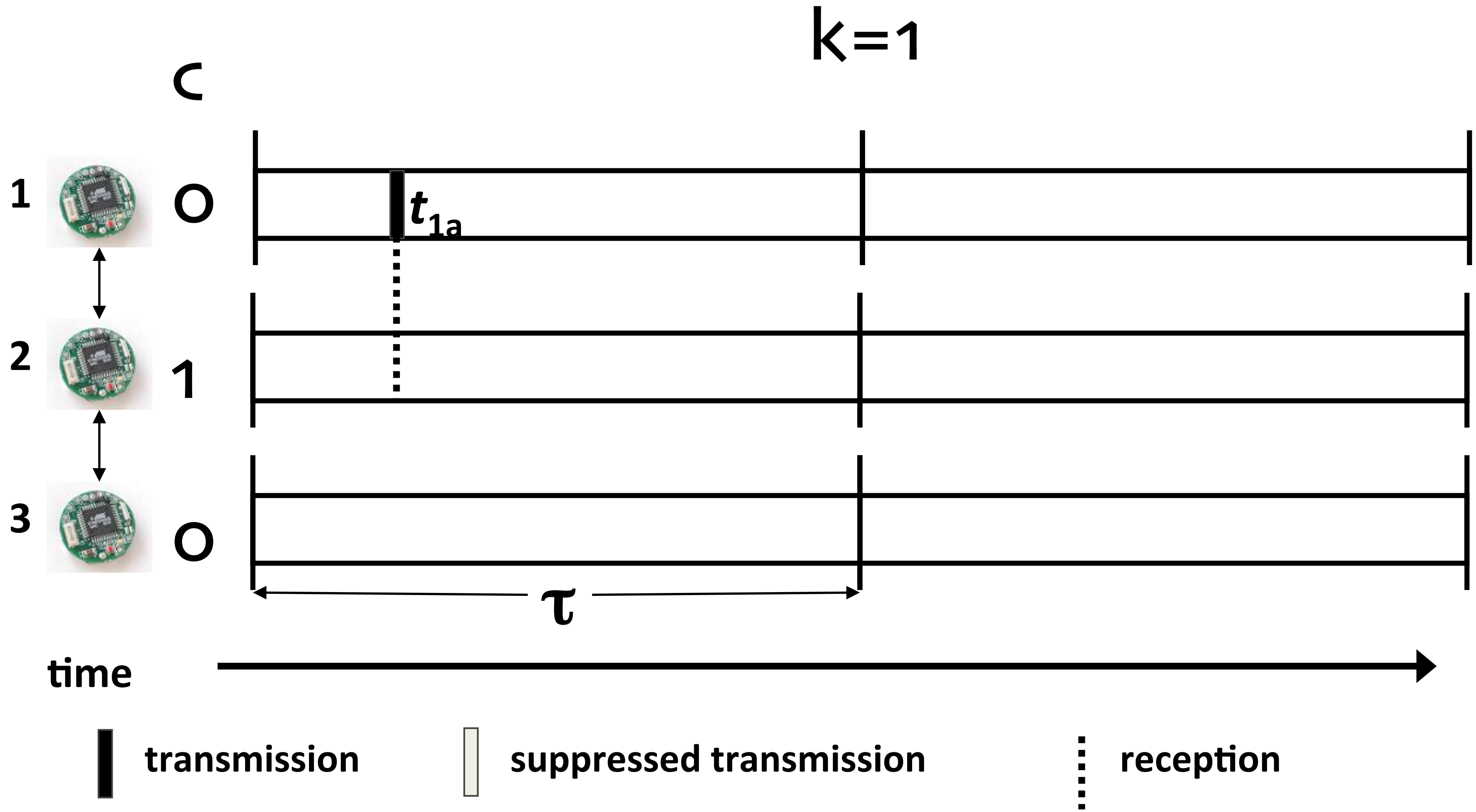
UNIVERSITY OF CAMBRIDGE

Example Trickle Execution

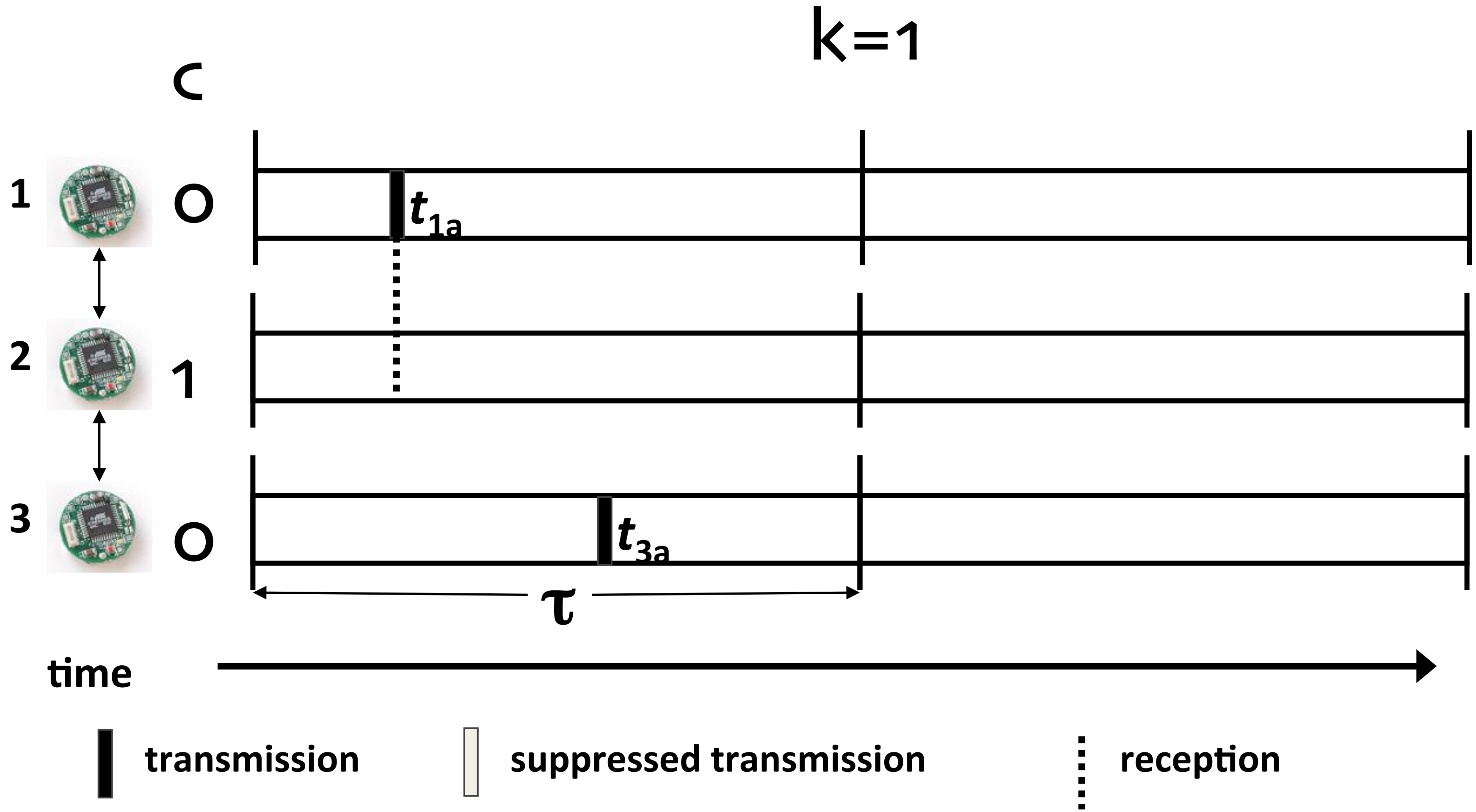


UNIVERSITY OF CAMBRIDGE

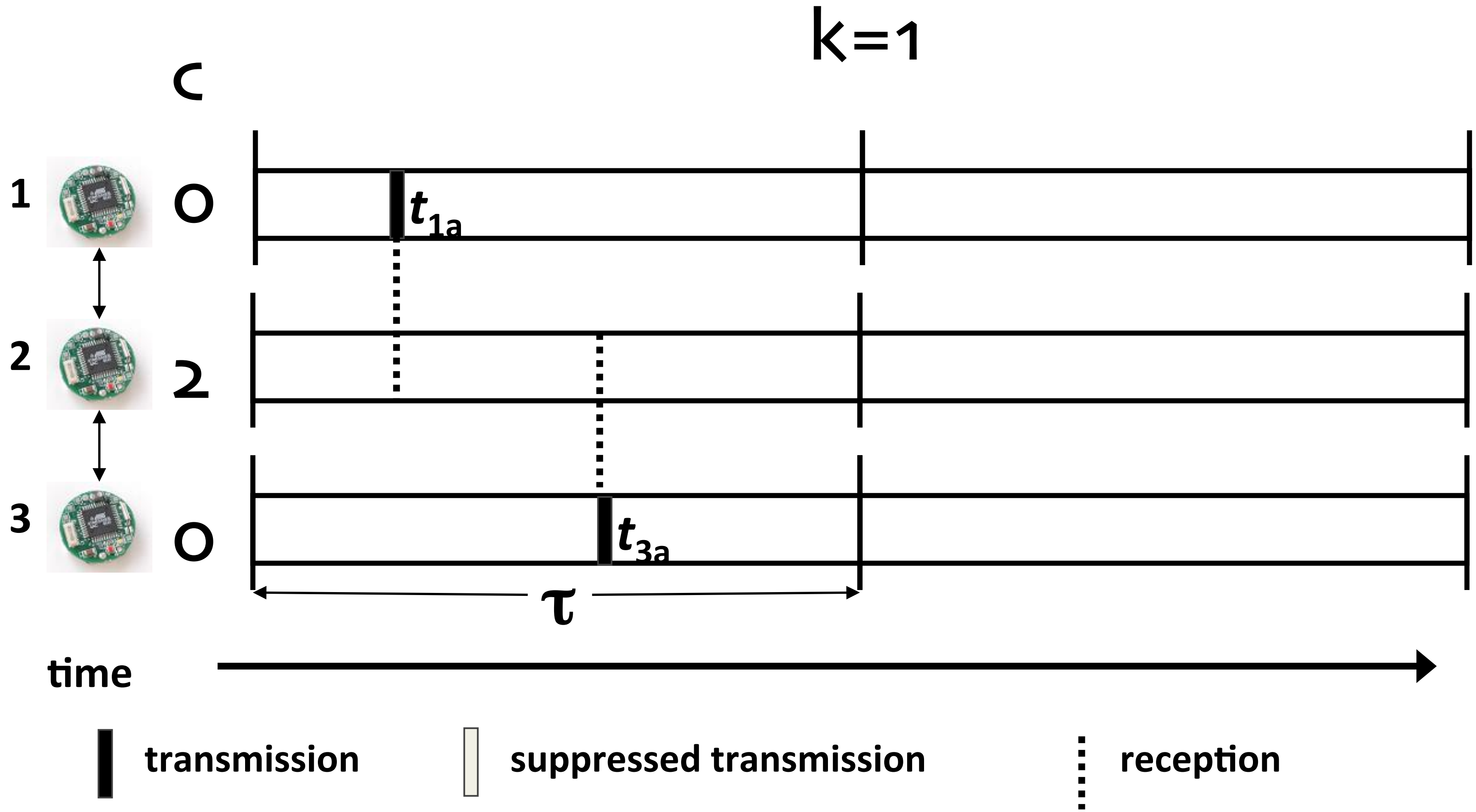
Example Trickle Execution



Example Trickle Execution

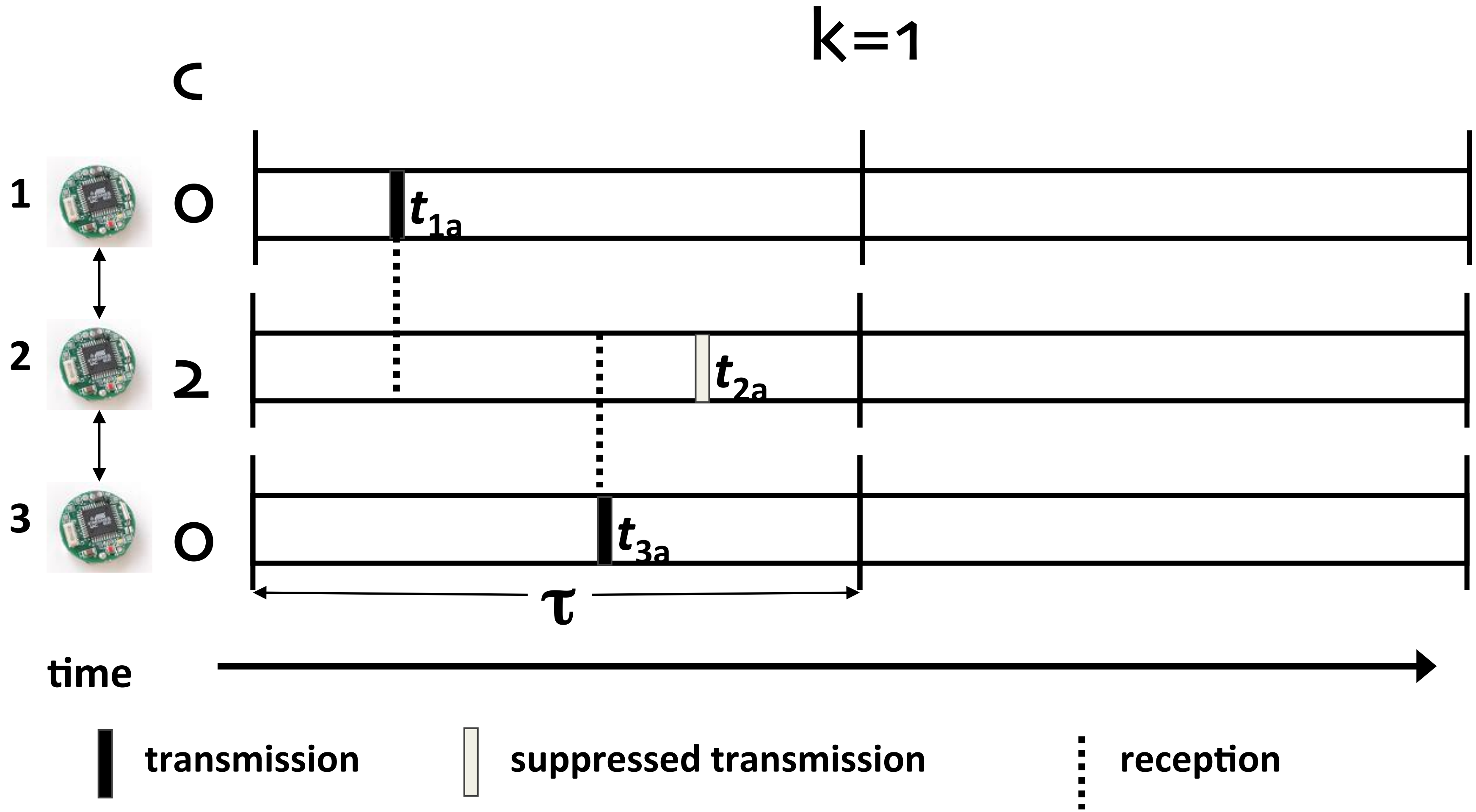


Example Trickle Execution



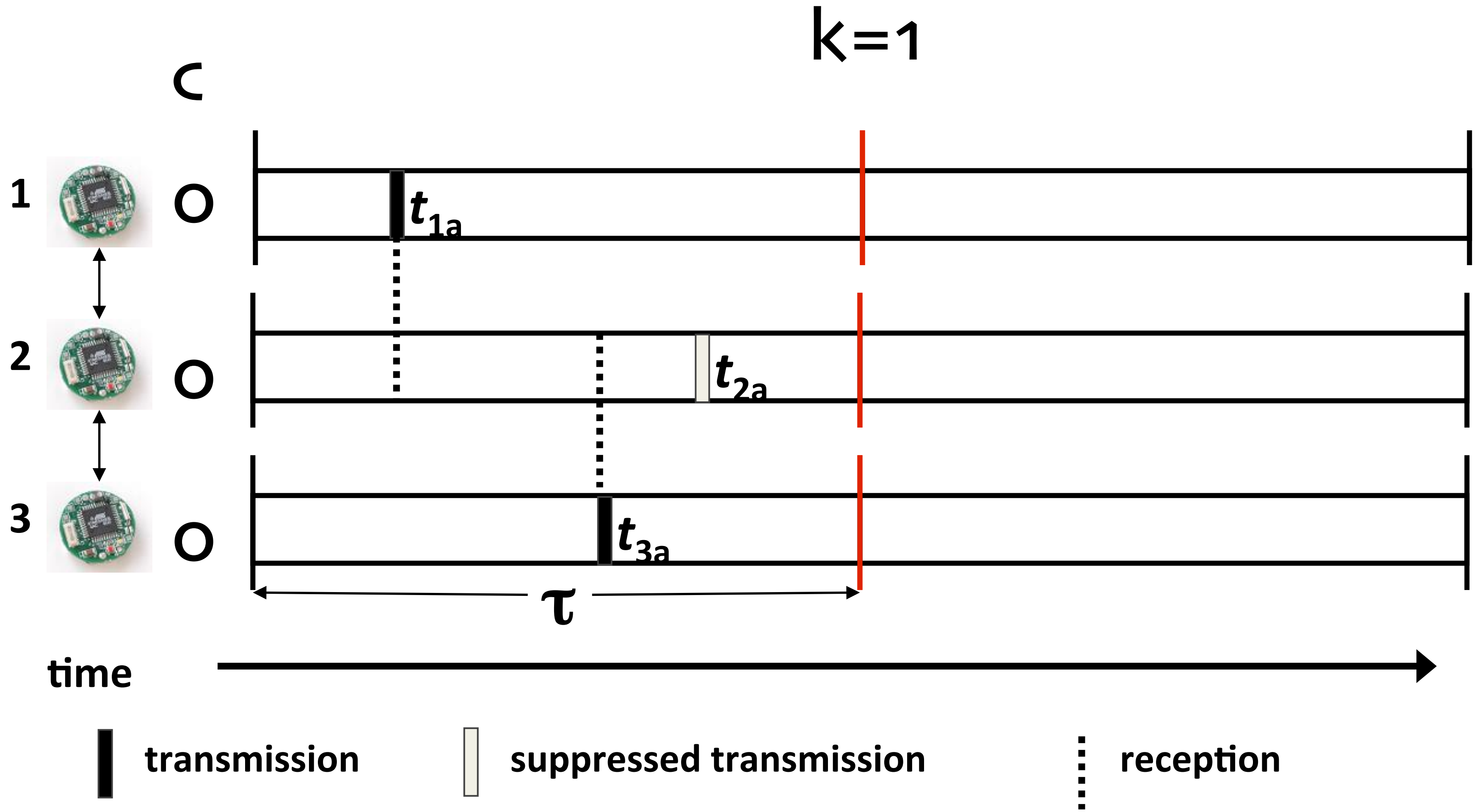
UNIVERSITY OF CAMBRIDGE

Example Trickle Execution

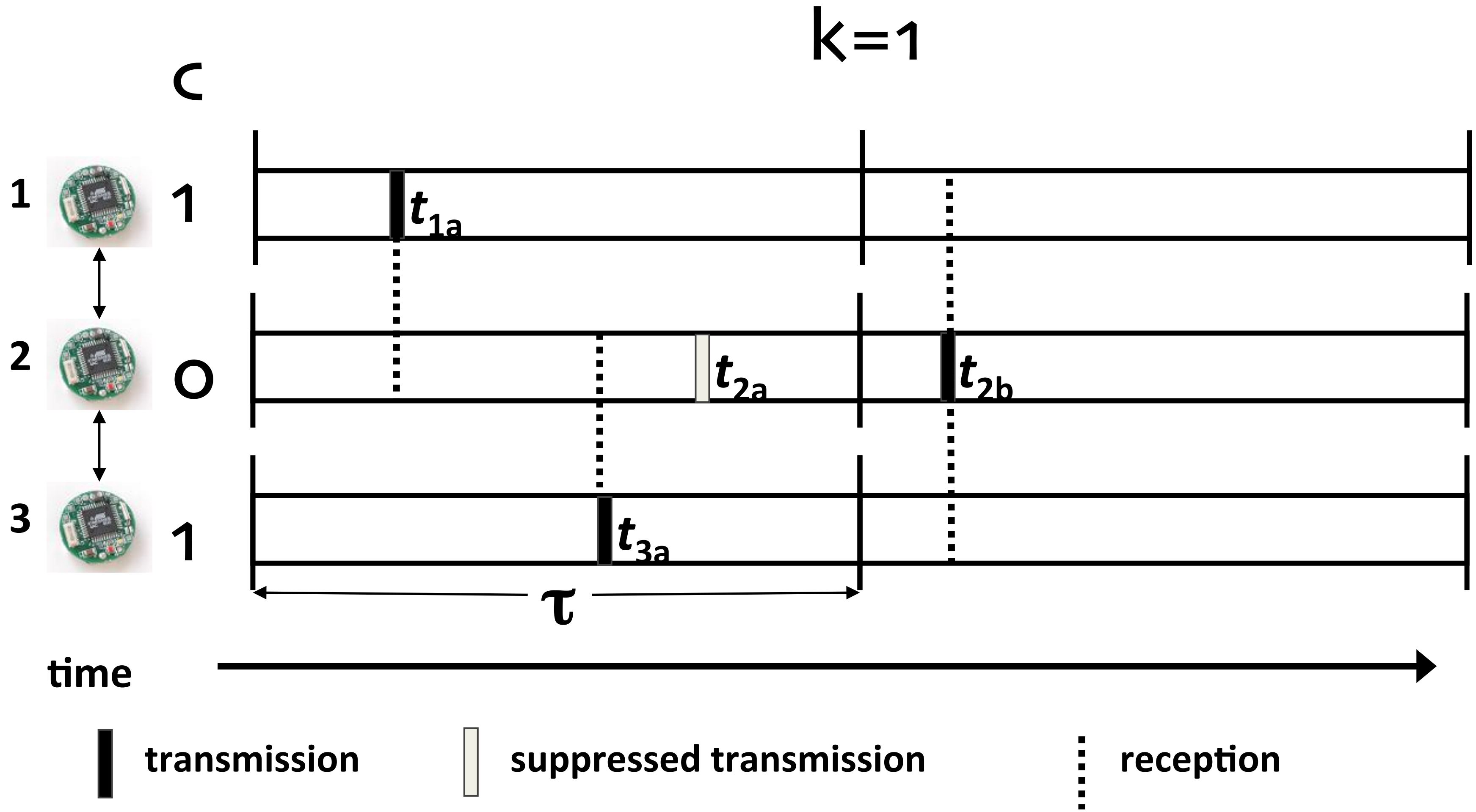


UNIVERSITY OF CAMBRIDGE

Example Trickle Execution

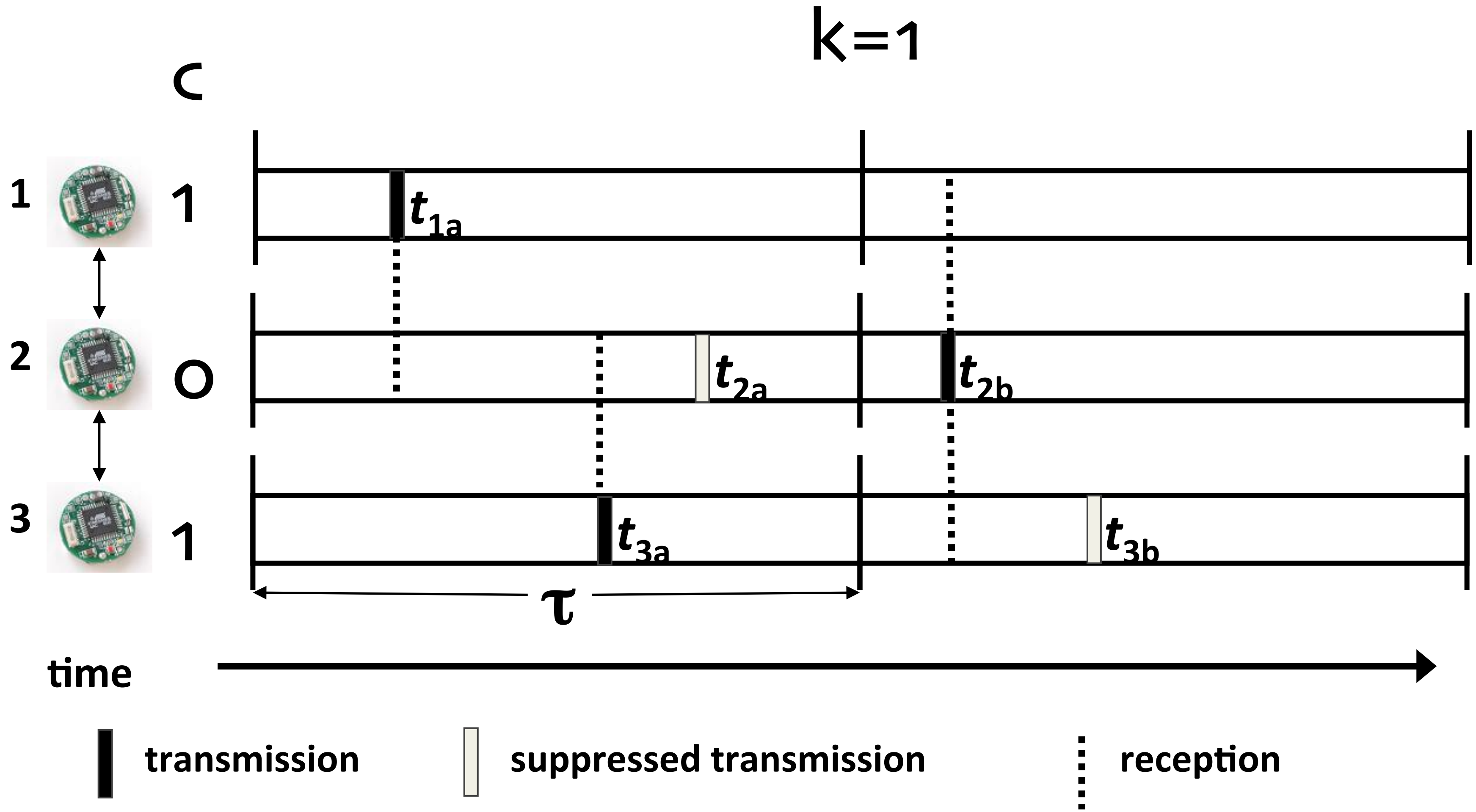


Example Trickle Execution



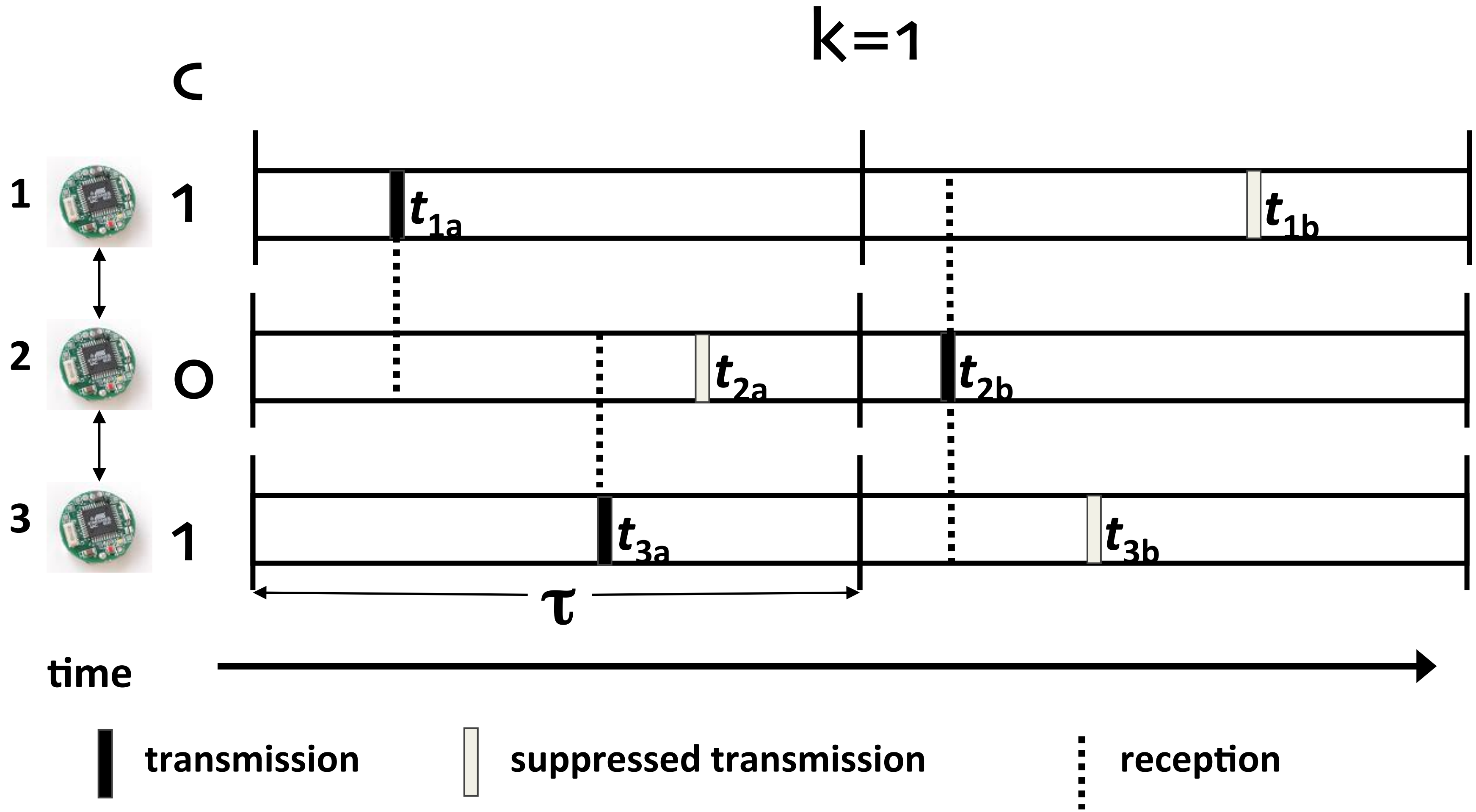
UNIVERSITY OF CAMBRIDGE

Example Trickle Execution



UNIVERSITY OF CAMBRIDGE

Example Trickle Execution



UNIVERSITY OF CAMBRIDGE

Assumptions

- Precise node synchronization
- No packet Loss

- Impact of these assumption?

Trickle: Impact of Packet Loss

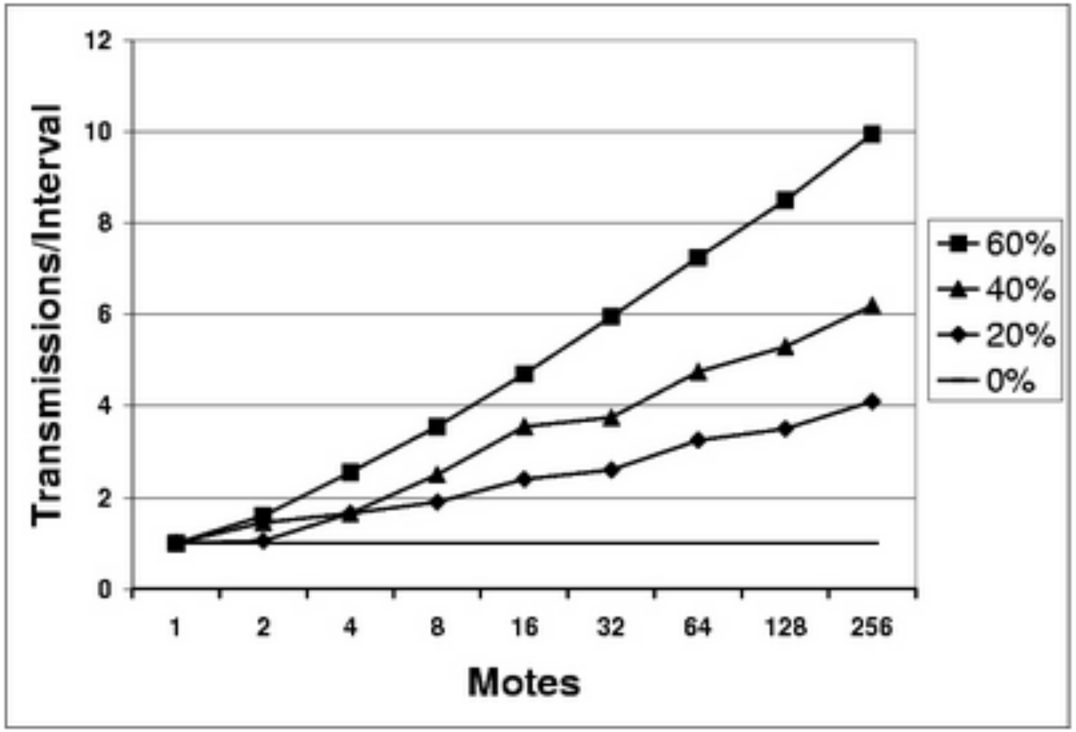


Figure 4: Number of Transmissions as Density Increases for Different Packet Loss Rates.

UNIVERSITY OF CAMBRIDGE

Trickle Maintenance without Synchronization – Short Listen Problem

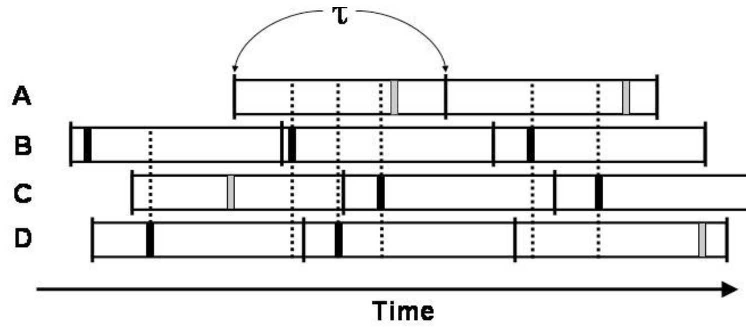


Figure 5: The Short Listen Problem For Motes A, B, C, and D. *Dark bars represent transmissions, light bars suppressed transmissions, and dashed lines are receptions. Tick marks indicate interval boundaries. Mote B transmits in all three intervals.*

- Mote B selects a small *t* on each of its three intervals:
  - Although other motes transmit, mote B's transmissions are never suppressed.
- The number of transmissions per intervals increases significantly.

UNIVERSITY OF CAMBRIDGE

Trickle –
Impact of Short Listen Problem

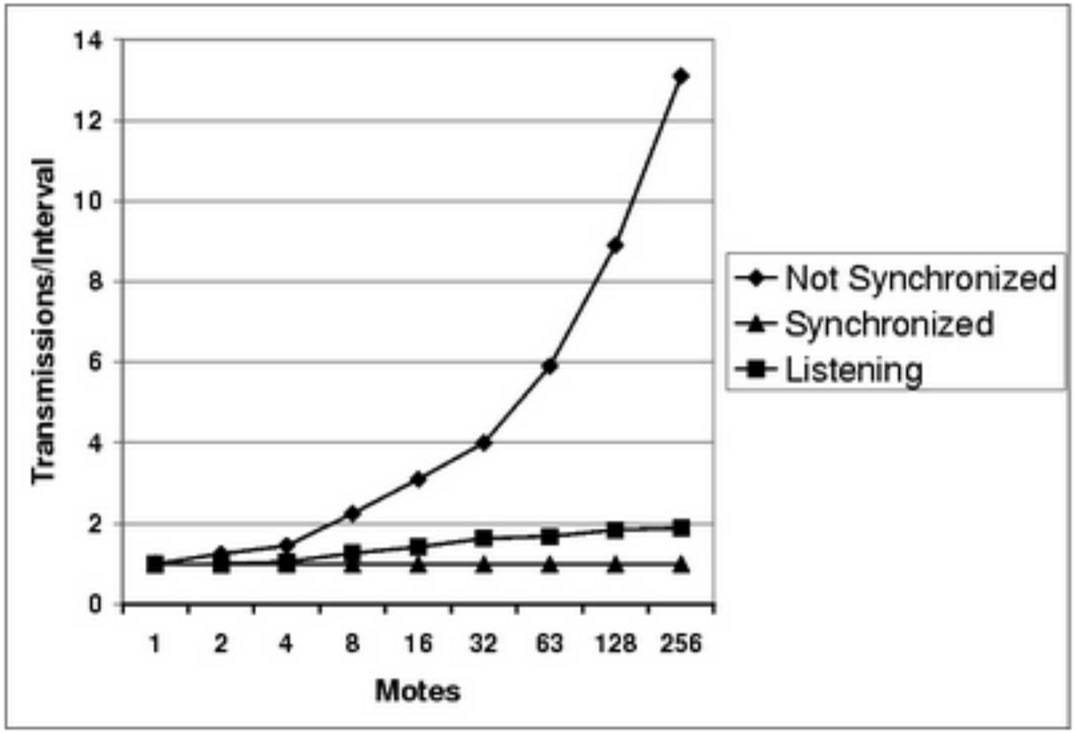


Figure 6: The Short Listen Problem's Effect on Scalability, *k=1. Without synchronization, Trickle scales with O(√n). A listening period restores this to asymptotically bounded by a constant.*

UNIVERSITY OF CAMBRIDGE

Solution to *Short Listen* Problem

- Instead of picking a *t* in the range $[0, \tau]$, t is selected in the range $[\tau/2, \tau]$

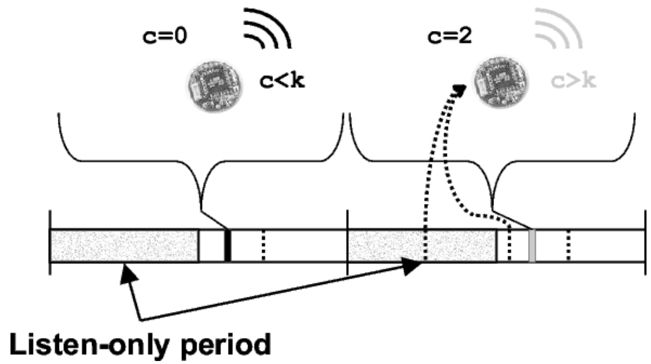


Figure 7: Trickle Maintenance with a k of 1 and a Listen-Only Period. *Dark boxes are transmissions, gray boxes are suppressed transmissions, and dotted lines are heard transmissions*.

Propagation

- Tradeoff between different values of *T*
  - A large *T*
    - Low communication overhead
    - Slowly propagates information
  - A small *T*
    - High communication overhead
    - Propagate more quickly
- How to improve?
  - Dynamically adjust *T*
    - Lower Bound T_l
    - Upper Bound T_h

UNIVERSITY OF CAMBRIDGE

Trickle Complete Algorithm

Event	Action
τ Expires	Double τ, up to τ_h. Reset c, pick a new t.
t Expires	If $c < k$, transmit.
Receive same metadata	Increment c.
Receive newer metadata	Set τ to τ_l. Reset c, pick a new t.
Receive newer code	Set τ to τ_l. Reset c, pick a new t.
Receive older metadata	Send updates.

t is picked from the range $[\frac{\tau}{2}, \tau]$

Figure 12: Trickle Pseudocode.

UNIVERSITY OF CAMBRIDGE

Mobile Sensor Networks

- We have considered fixed sensor networks.
- There are however examples in which the sensor networks are mobile, i.e., the nodes of the networks do not have a fixed position.
- Example of this are when sensors are moved through controlled movement (E.g. a sensor robot) or when sensors are attached to moving entities and the mobility is independent from the sensing activity (E.g. animals or vehicles or humans).

UNIVERSITY OF CAMBRIDGE

Impact of Mobility

- MAC Layer protocols:
  – Mobility impacts the protocol of duty cycling as the neighbours of the nodes are not the same all the time.
  – Adaptation of low power listening protocols are reasonably suitable.
  – **Alternatively, approaches which keep into account periodic encounter patterns.**

UNIVERSITY OF CAMBRIDGE

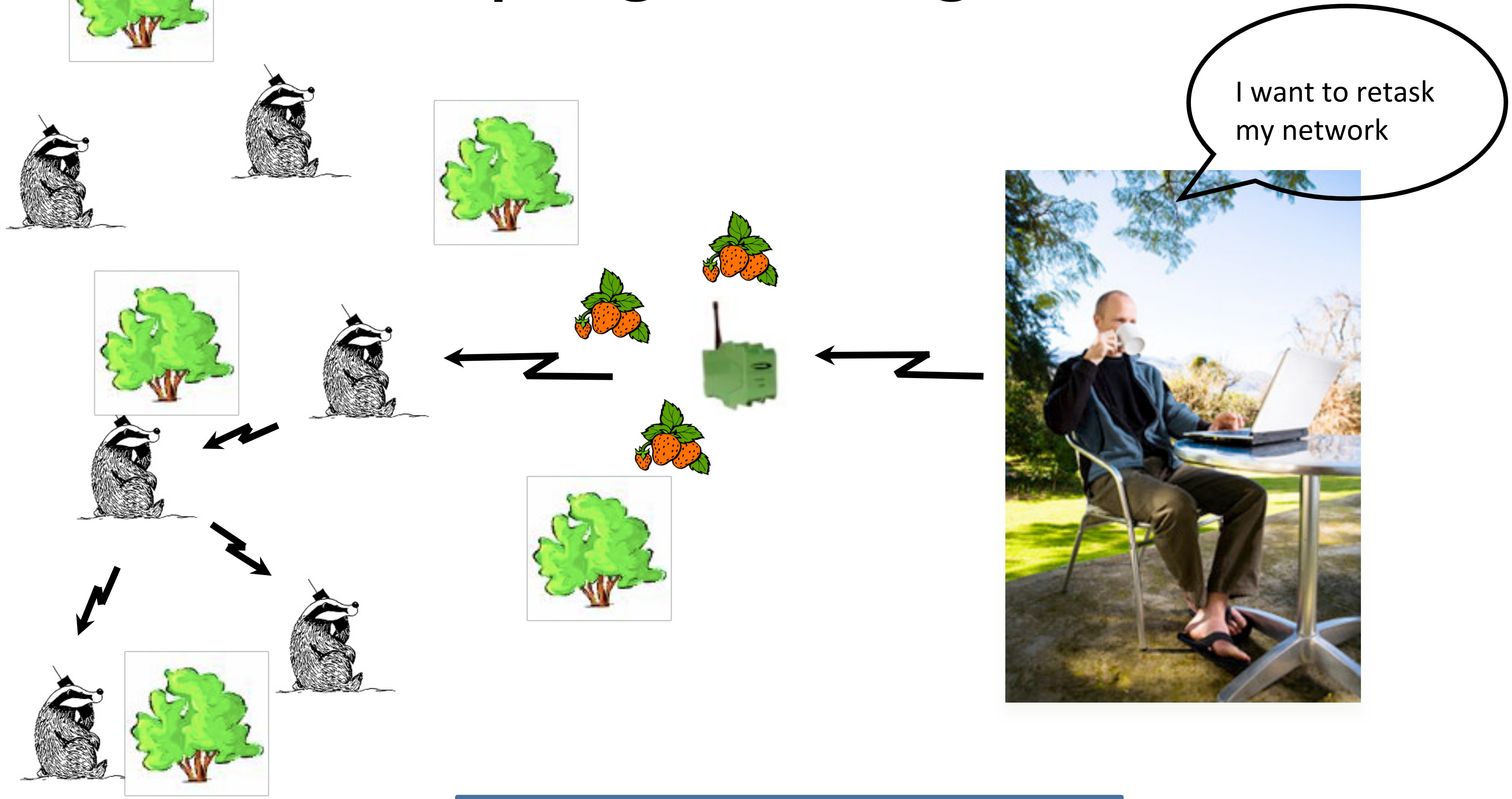
Impact of Mobility (2)

- Routing protocols:
  - All of a sudden establishing a tree structure does not make sense any longer.
  - Delay tolerant routing protocols are applicable (on top of duty cycling approaches).

Impact of Mobility(3)

- Reprogramming:
  - Existing solutions target connected fixed networks.
  - Delay tolerant solutions could be applied however some attention to targeted set of nodes should be applied (eg reprogram only nodes which go to certain areas) and attention to avoid useless code broadcasts should be paid.

Mobile WSN Reprogramming



Network of animals wearing sensors on their collars

UNIVERSITY OF CAMBRIDGE

What's the best way to distribute the update?

- Flooding? No too expensive.

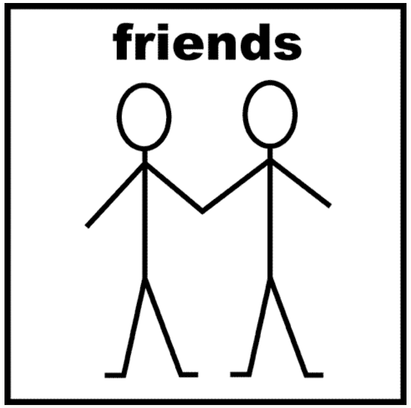
- These animals are social!
- These social groups tend to be stable over time, and more importantly, they spend a lot of time together, regularly.

UNIVERSITY OF CAMBRIDGE

Social Dissemination

- Dissemination:
  - use social characteristics of the network!

- Selective update:
  - use the network to figure out whom to update.



UNIVERSITY OF CAMBRIDGE

Social Dissemination

- Instead of flooding the network, let us try to use the social characteristics: social groups, social links between nodes, as well as group leaders;
- Groups tend to stay connected - perfect for maintenance!
- Animals do not behave all in the same way - some are more active than others:
  - group leaders: identify leaders, and spread code among them. identify clusters: wait until they come together let leaders disseminate code using smart broadcasts to their group.

References

- Levis P., Patel L., Shenker S., Culler D. 2004. Trickle: A Self-Regulating Algorithm for Code Propagation and Maintenance in Wireless Sensor Networks. *In Proceedings of the First USENIX/ACM Symposium on Networked Systems Design and Implementation (NSDI 2004)*. Pages 15-28.
- B. Pasztor, L. Mottola, C. Mascolo, G. P. Picco, S. Ellwood and D. Macdonald. Selective Reprogramming of Mobile Sensor Networks through Social Community Detection. In Proceedings of 7th European Conference on Wireless Sensor Networks (EWSN2010). Coimbra, Portugal. February 2010. Springer.