Data Dissemination in Wireless Sensor Networks

Philip Levis
UC Berkeley
Intel Research Berkeley

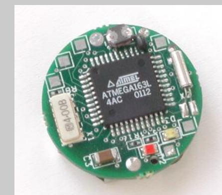
Neil Patel
UC Berkeley

David Culler
UC Berkeley

Scott Shenker
UC Berkeley
ICSI

Sensor Networks

- Sensor networks are large collections of small, embedded, resource constrained devices
  - Energy is the limiting factor



- A low bandwidth wireless broadcast is the basic network primitive (not end-to-end IP)
  - Standard TinyOS packet data payload is 29 bytes
- Long deployment lifetimes (months, years) require retasking
- Retasking needs to disseminate data (a program, parameters) to every node in a network

To Every Node in a Network

- Network membership is not static
  - Loss
  - Transient disconnection
  - Repopulation
- Limited resources prevent storing complete network population information
- *To ensure dissemination to every node, we must periodically maintain that every node has the data.*

The Real Cost

- Propagation is costly
  - Virtual programs (Maté, TinyDB): 20-400 bytes
  - Parameters, predicates: 8-20 bytes
  - To every node in a large, multihop network…
- But maintenance is more so
  - For example, one maintenance transmission every minute
  - Maintenance for 15 minutes costs more than 400B of data
  - For 8-20B of data, <u>two minutes</u> are more costly!
- *Maintaining that everyone has the data costs more than propagating the data itself.*

Three Needed Properties

- Low maintenance overhead
  - Minimize communication when everyone is up to date
- Rapid propagation
  - When new data appears, it should propagate quickly
- Scalability
  - Protocol must operate in a wide range of densities
  - Cannot require *a priori* density information

Existing Algorithms Are Insufficient

- Epidemic algorithms
  - End to end, single destination communication, IP overlays
- Probabilistic broadcasts
  - Discrete effort (terminate): does not handle disconnection
- Scalable Reliable Multicast
  - Multicast over a wired network, latency-based suppression
- SPIN (Heinzelman et al.)
  - Propagation protocol, does not address maintenance cost

Solution: Trickle

Solution: Trickle

- "Every once in a while, broadcast what data you have, unless you've heard some other nodes broadcast the same thing recently."

Solution: Trickle

- "Every once in a while, broadcast what data you have, unless you've heard some other nodes broadcast the same thing recently."
- Behavior (simulation and deployment):
  - Maintenance: a few sends per hour
  - Propagation: less than a minute
  - Scalability: thousand-fold density changes

Solution: Trickle

- "Every once in a while, broadcast what data you have, unless you've heard some other nodes broadcast the same thing recently."
- Behavior (simulation and deployment):
  - Maintenance: a few sends per hour
  - Propagation: less than a minute
  - Scalability: thousand-fold density changes
- Instead of flooding a network, establish a trickle of packets, just enough to stay up to date.

Outline

- Data dissemination
- Trickle algorithm
- Experimental methodology
- Maintenance
- Propagation
- Conclusion

Trickle Assumptions

- Broadcast medium
- Concise, comparable metadata
  - Given A and B, know if one needs an update
- Metadata exchange (maintenance) is the significant cost

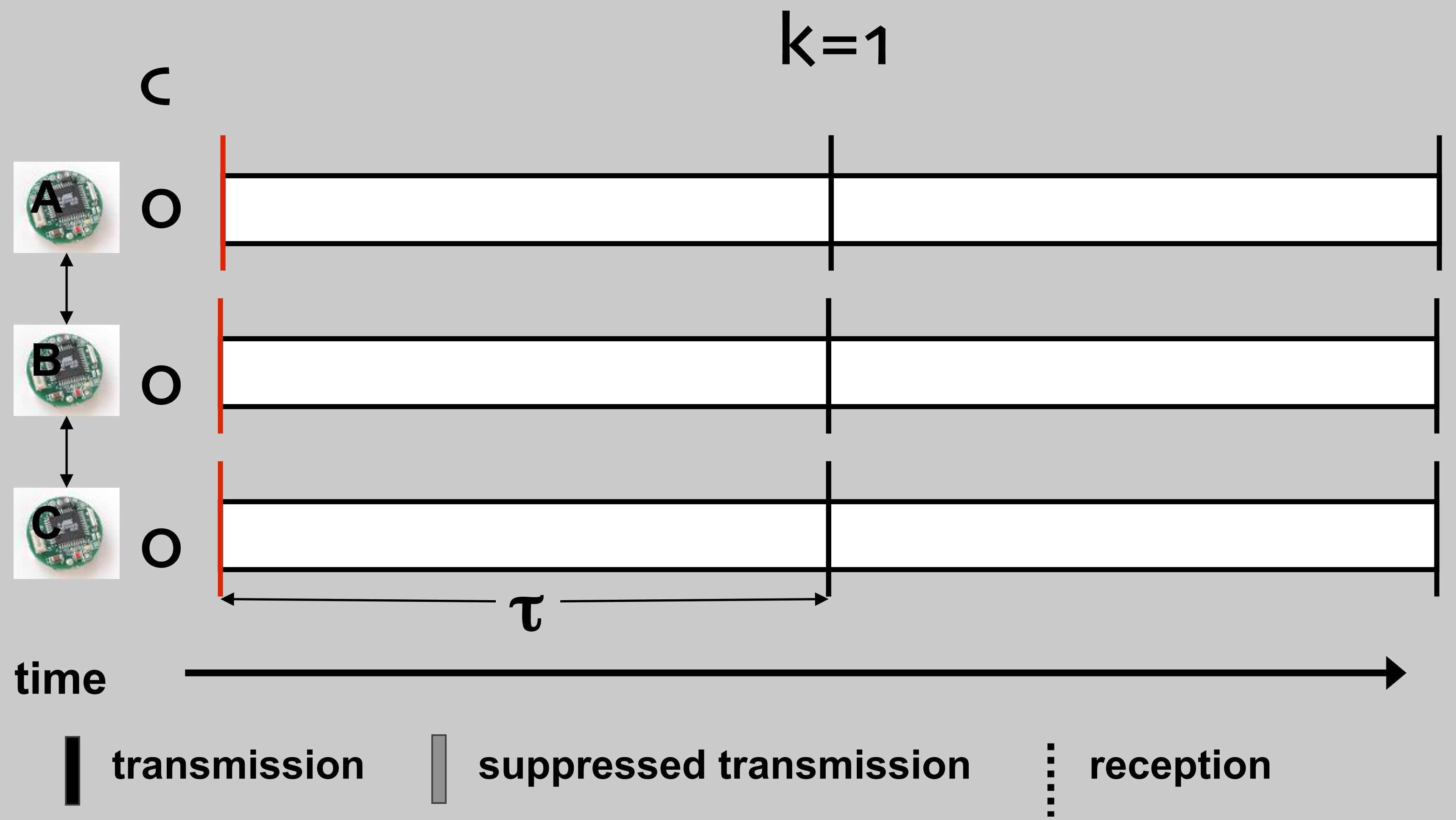
Detecting That a Node Needs an Update

- As long as each node *communicates* with others, inconsistencies will be found
- Either reception or transmission is sufficient
- Define a desired detection latency, τ
- Choose a redundancy constant k
  - k = (receptions + transmissions)
  - In an interval of length τ
- Trickle keeps the rate as close to k/ τ as possible

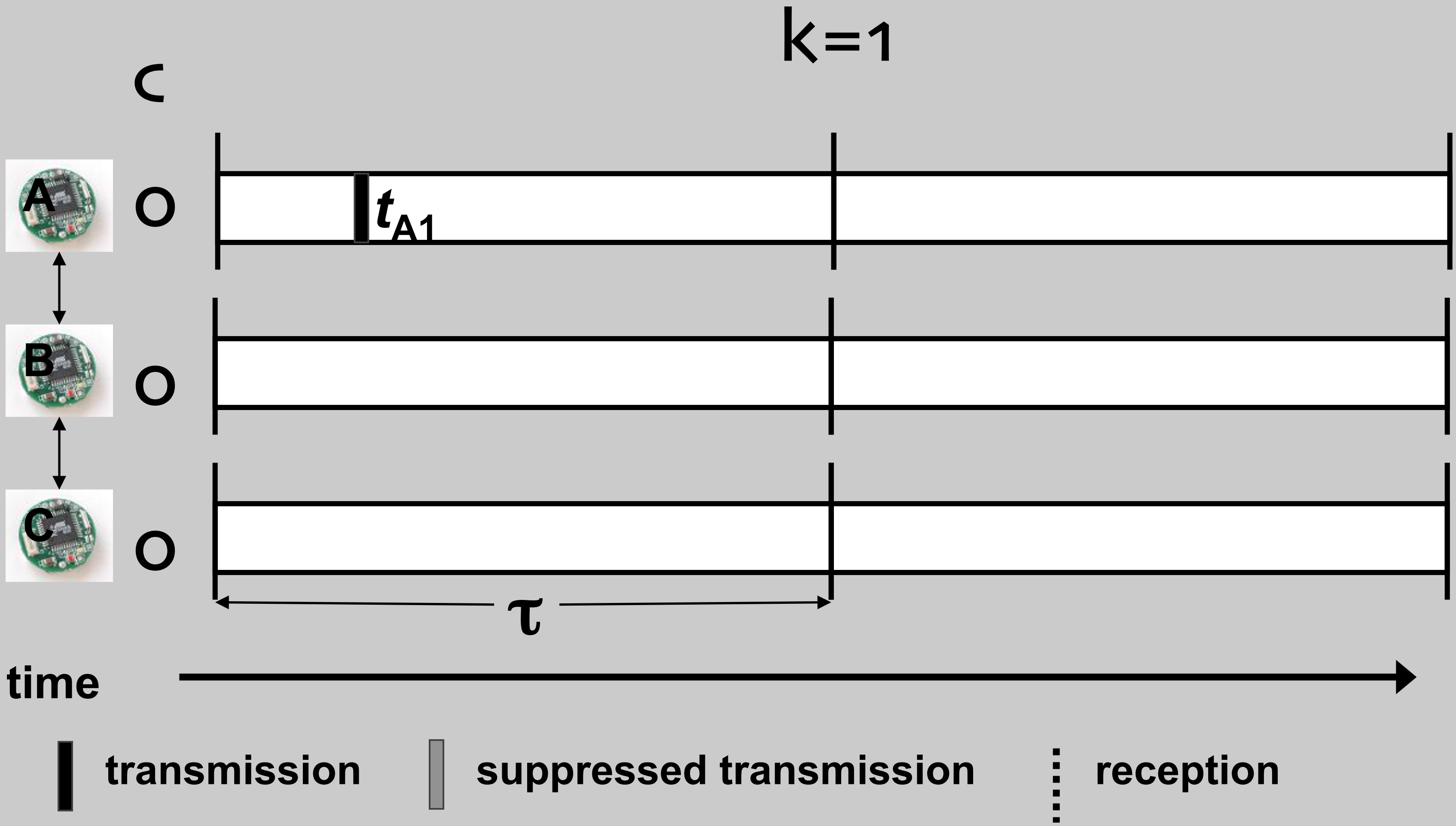
Trickle Algorithm

- Time interval of length τ
- Redundancy constant *k* (e.g., 1, 2)
- Maintain a counter *c*
- Pick a time *t* from $[0, \tau]$
- At time *t*, transmit metadata if $c < k$
- Increment *c* when you hear identical metadata to your own
- Transmit updates when you hear older metadata
- At end of τ, pick a new *t*

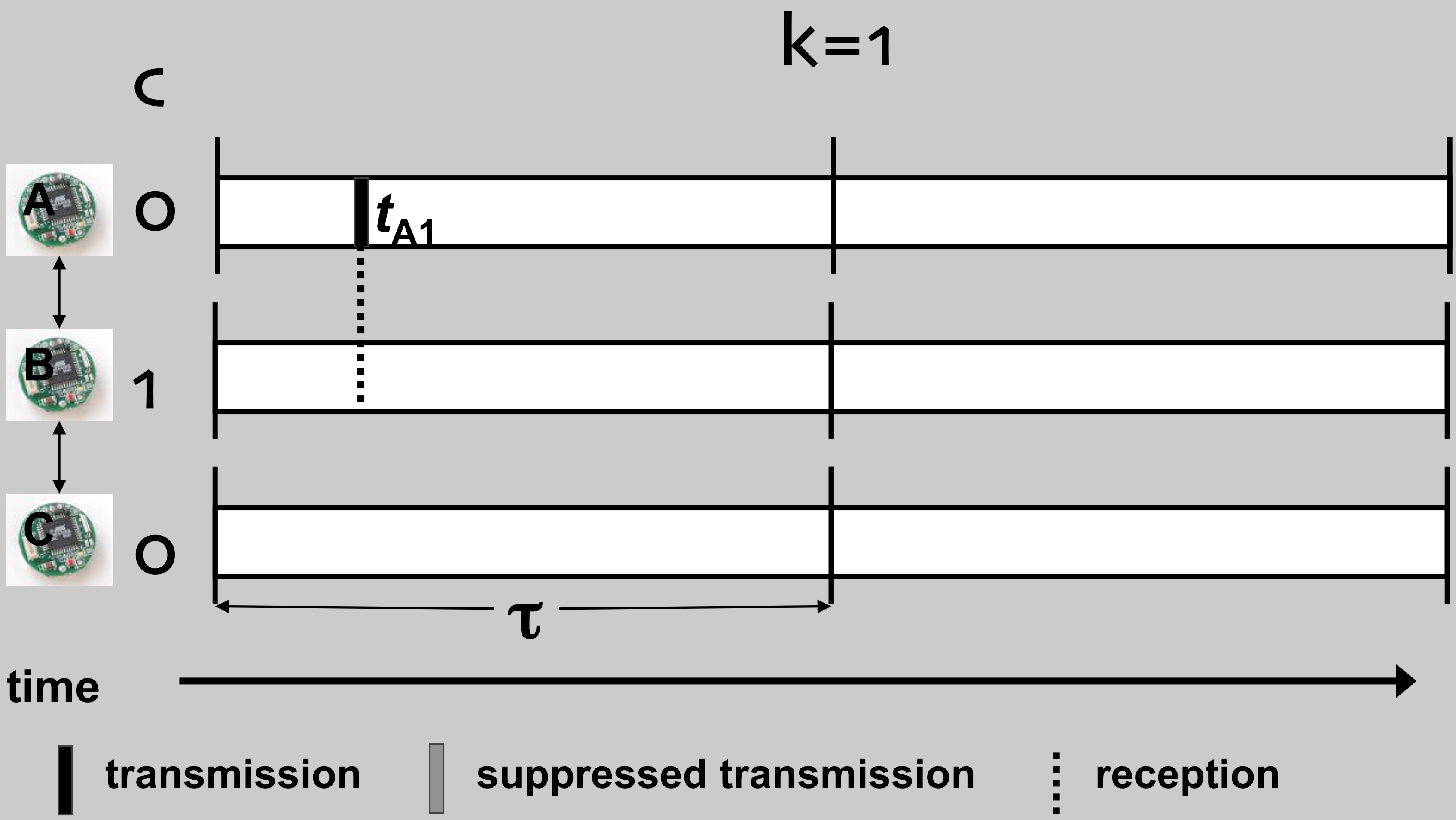
Example Trickle Execution



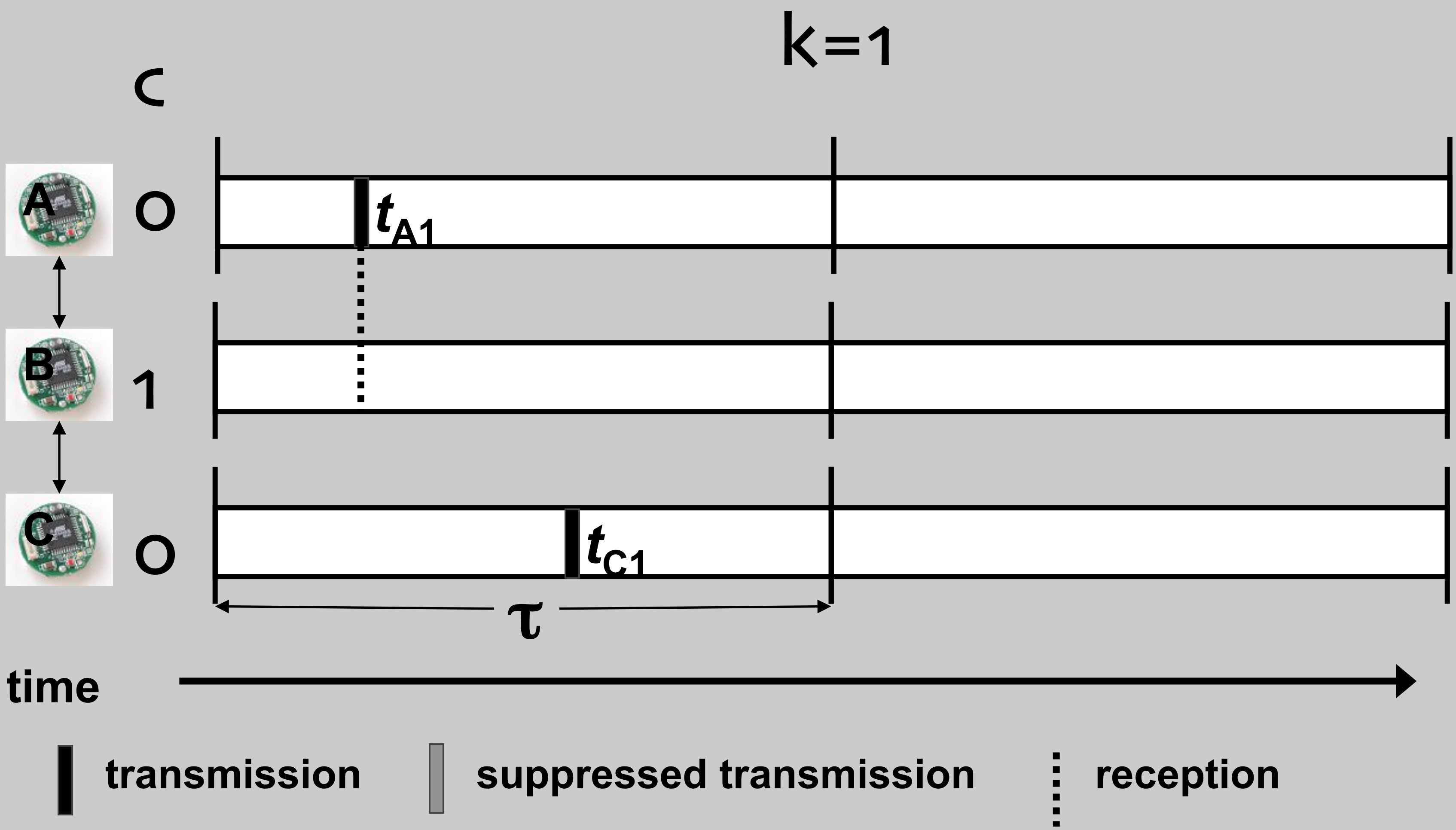
Example Trickle Execution



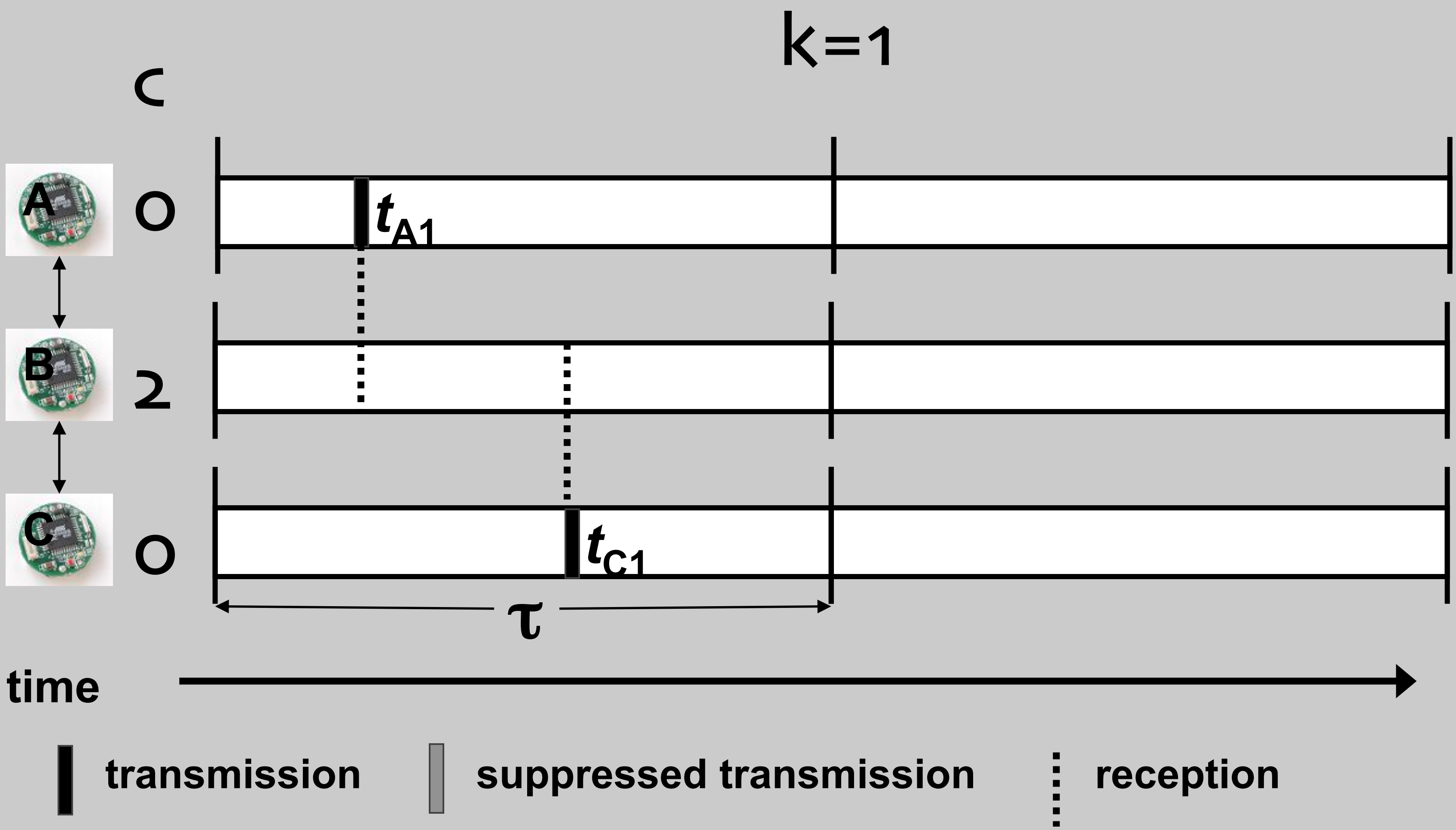
Example Trickle Execution



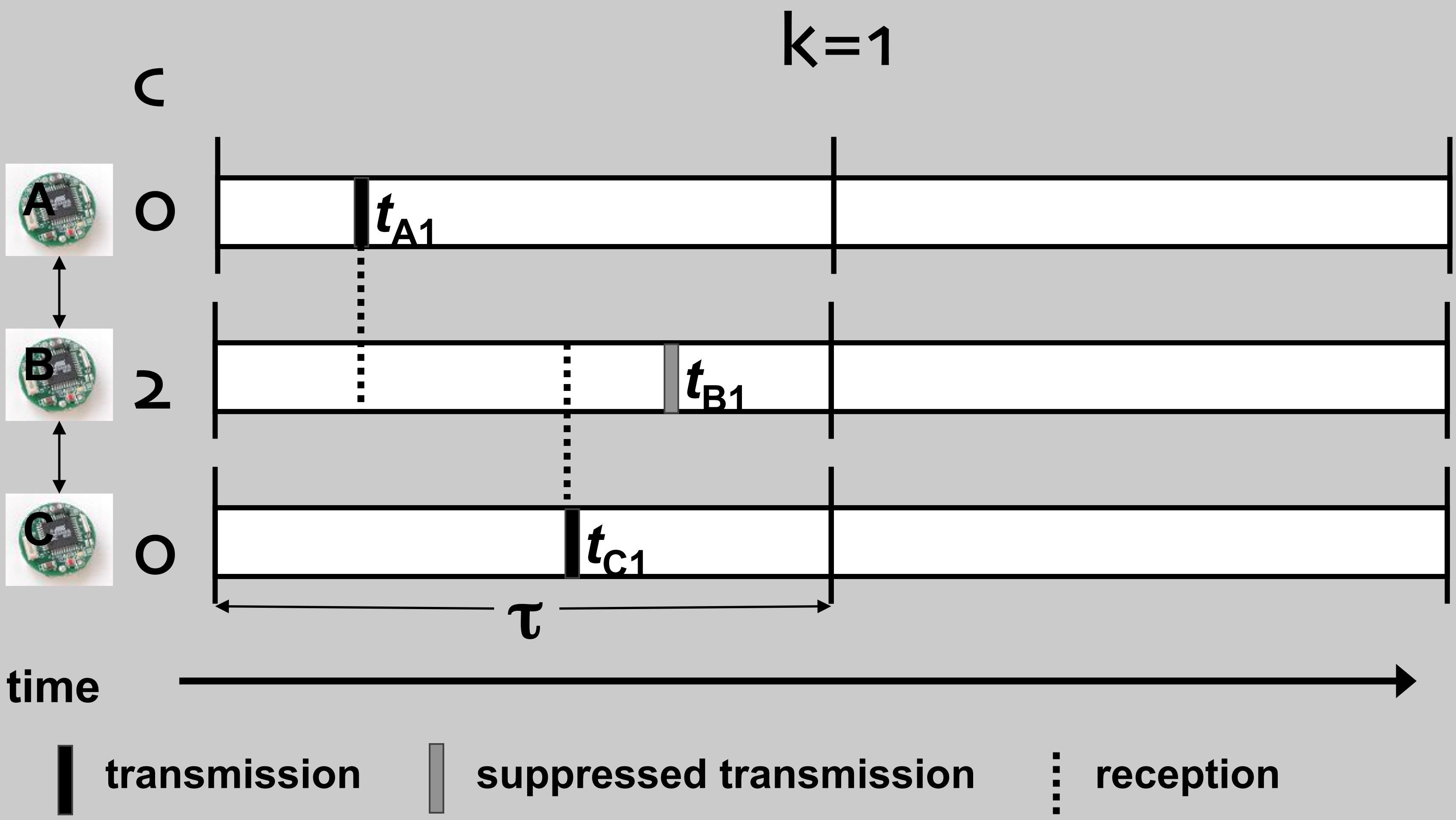
Example Trickle Execution



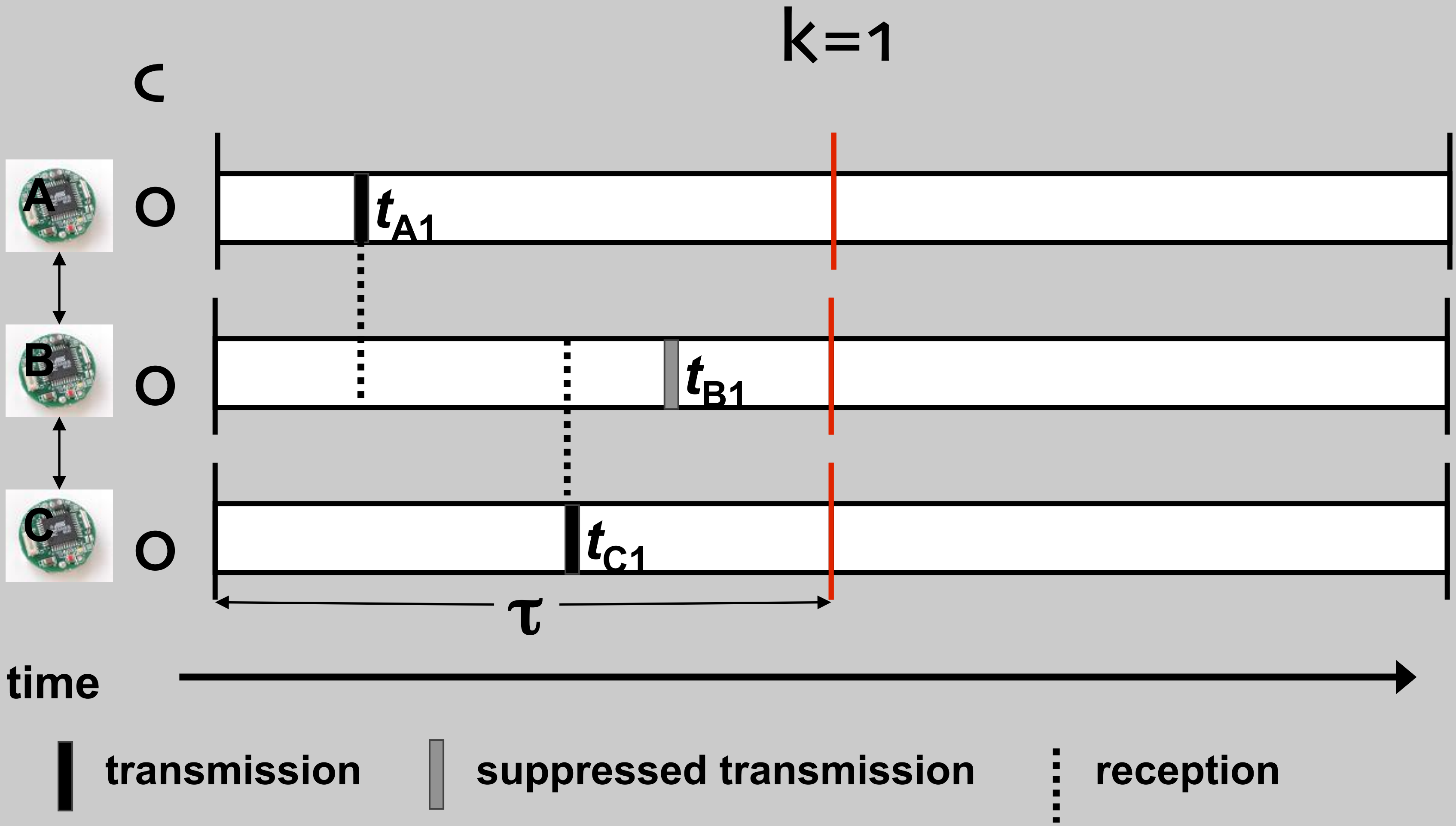
Example Trickle Execution



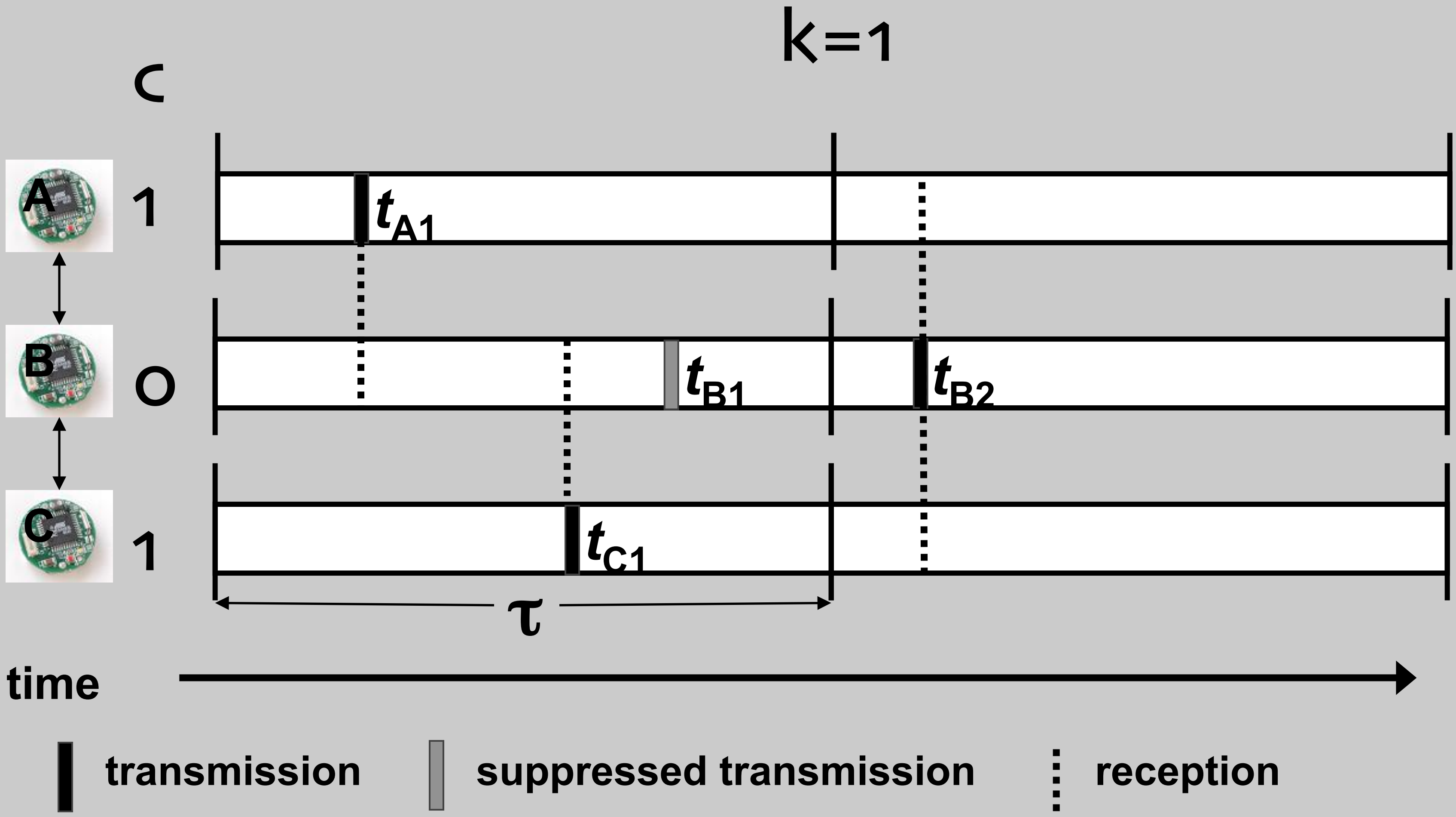
Example Trickle Execution



Example Trickle Execution

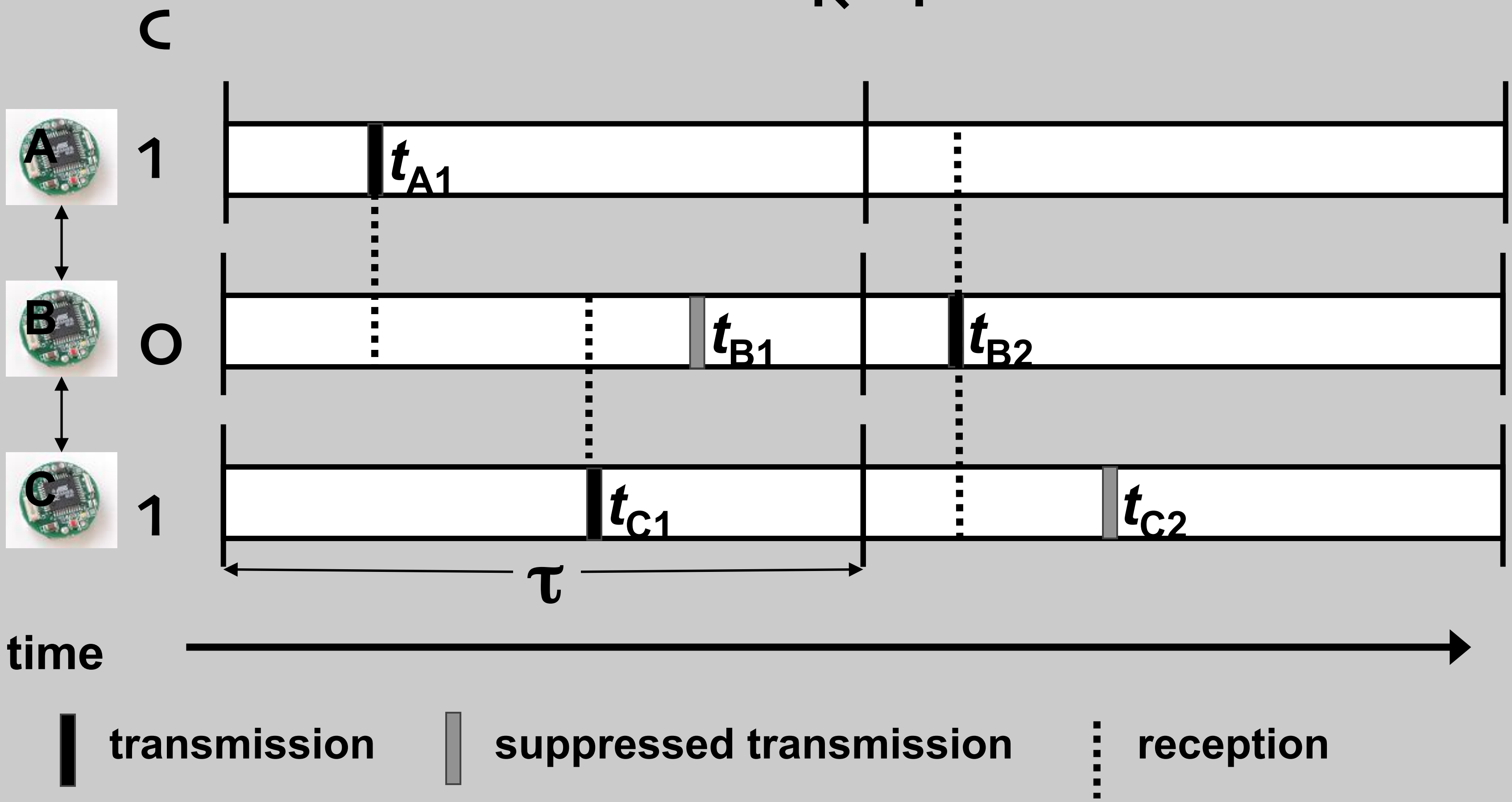


Example Trickle Execution

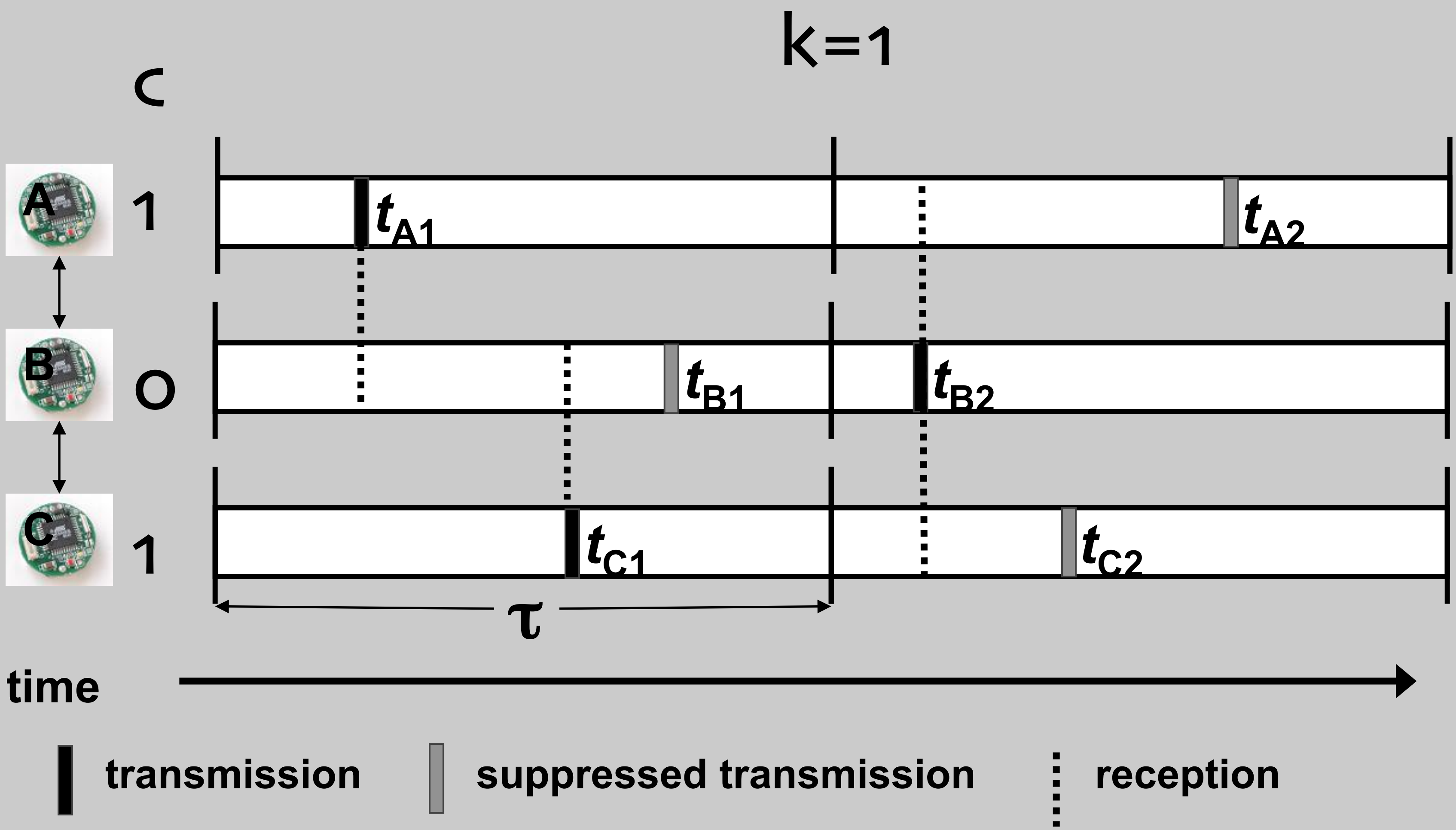


Example Trickle Execution

k=1



Example Trickle Execution



Outline

- Data dissemination
- Trickle algorithm
- Experimental methodology
- Maintenance
- Propagation
- Future Work

Experimental Methodology

- High-level, algorithmic simulator
  - Single-hop network with a uniform loss rate
- TOSSIM, simulates TinyOS implementations
  - Multi-hop networks with empirically derived loss rates
- Real world deployment in an indoor setting
- In experiments (unless said otherwise), *k* =1

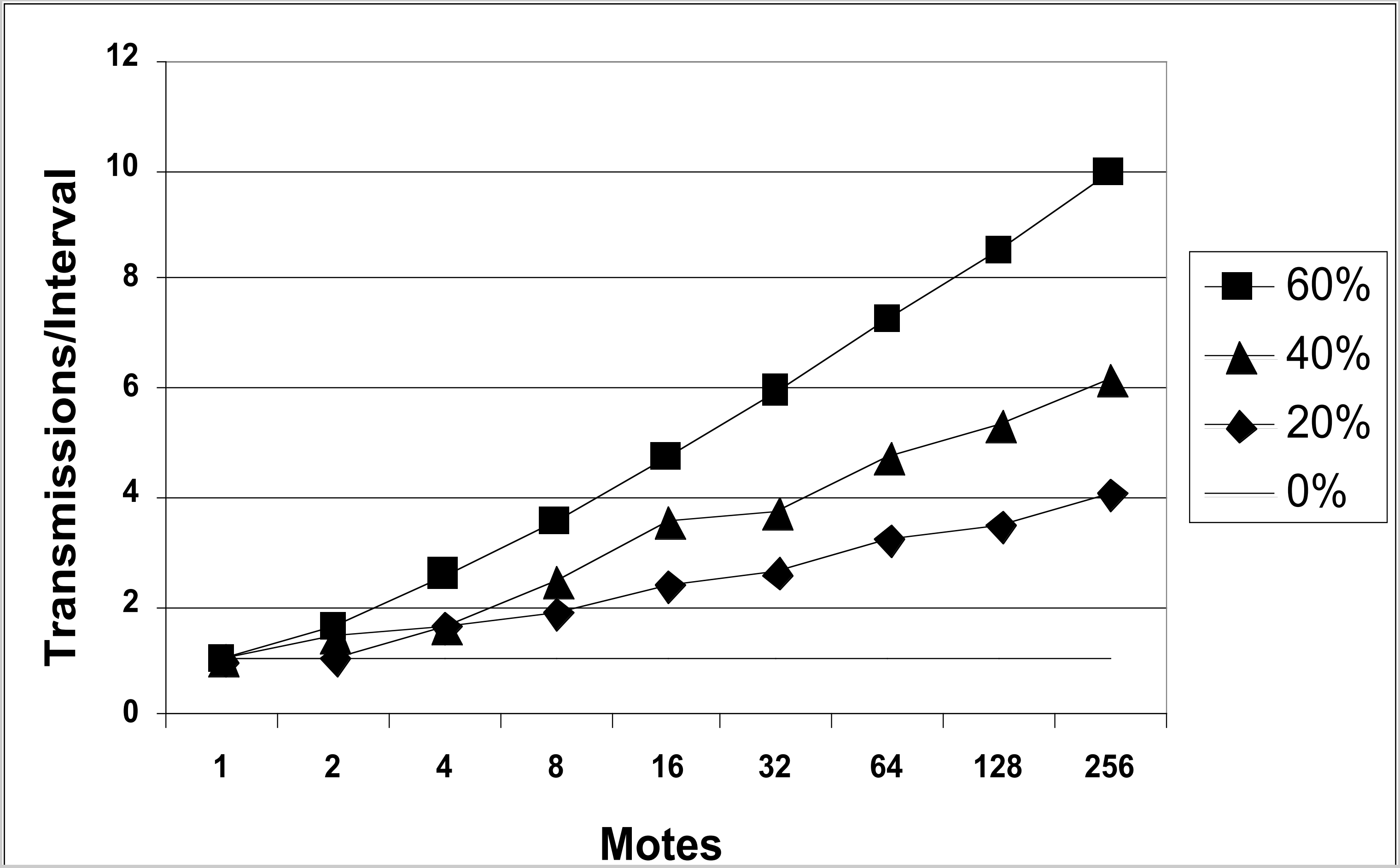
Outline

- Data dissemination
- Trickle algorithm
- Experimental methodology
- Maintenance
- Propagation
- Future Work

Maintenance Evaluation

- Start with idealized assumptions, relax each
  - Lossless cell
  - Perfect interval synchronization
  - Single hop network
- Ideal: Lossless, synchronized single hop network
  - *k* transmissions per interval
  - First k nodes to transmit suppress all others
  - Communication rate is independent of density
- First step: introducing loss

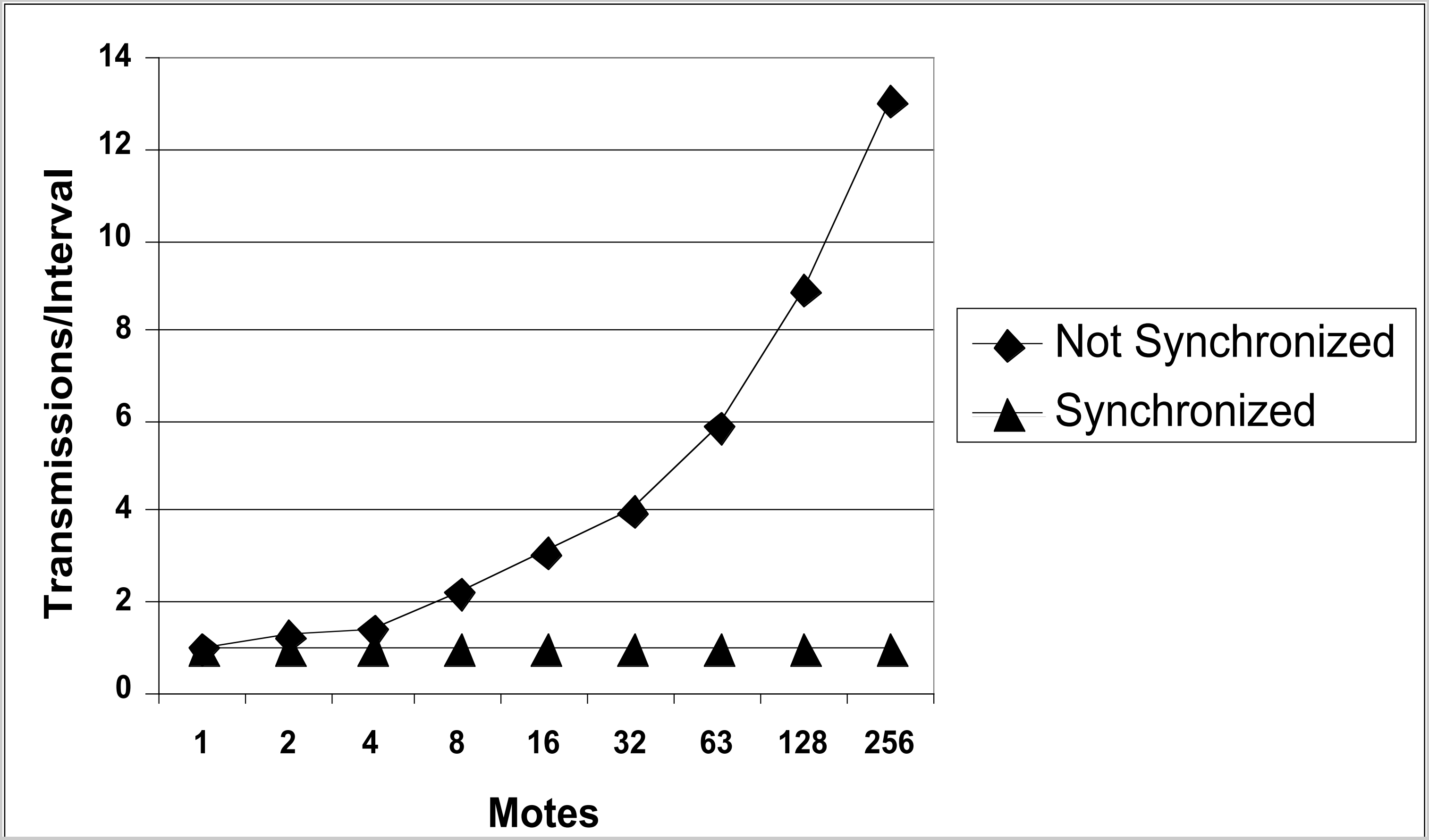
Loss
(algorithmic simulator)



Logarithmic Behavior of Loss

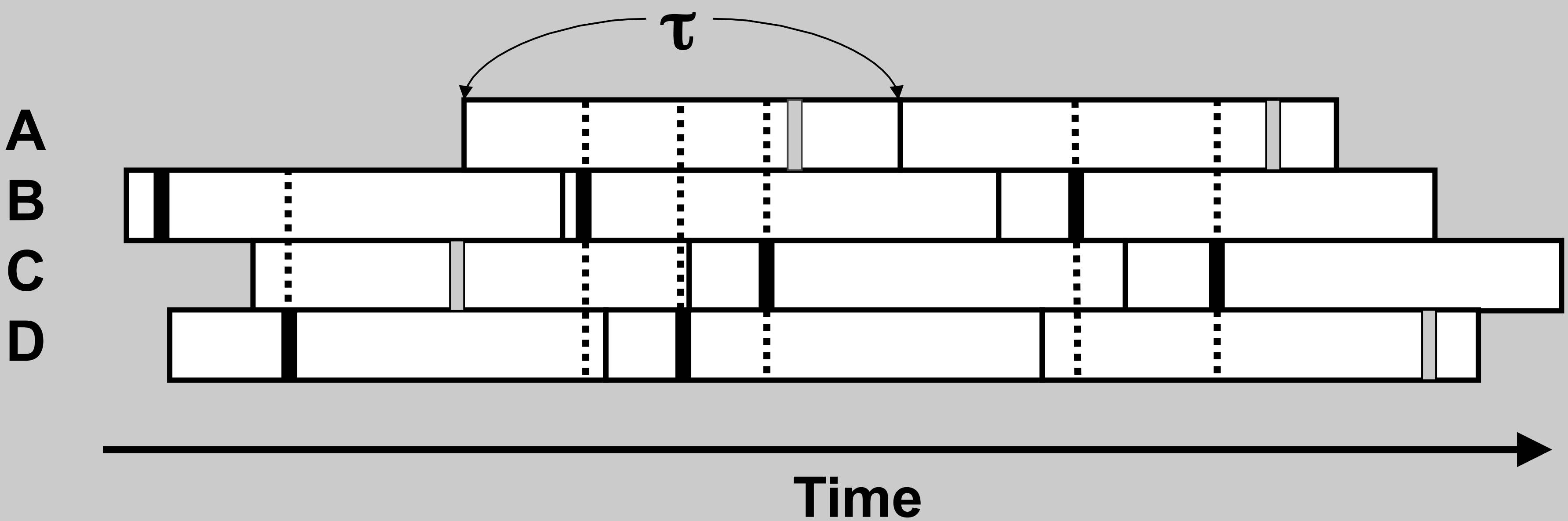
- Transmission increase is due to the probability that one node has not heard *n* transmissions
- Example: 10% loss
  - 1 in 10 nodes will not hear one transmission
  - 1 in 100 nodes will not hear two transmissions
  - 1 in 1000 nodes will not hear three, etc.
- Fundamental bound to maintaining a per-node communication rate

Synchronization
(algorithmic simulator)



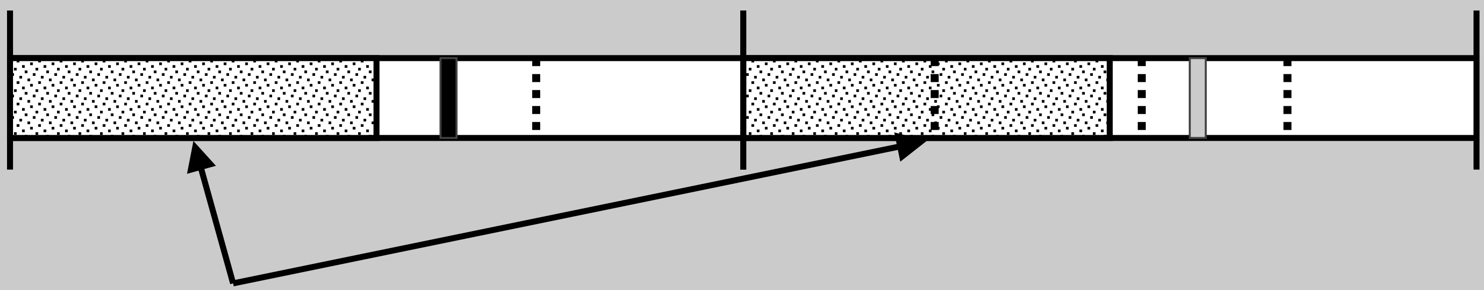
Short Listen Effect

- Lack of synchronization leads to the “short listen effect”
- For example, B transmits three times:



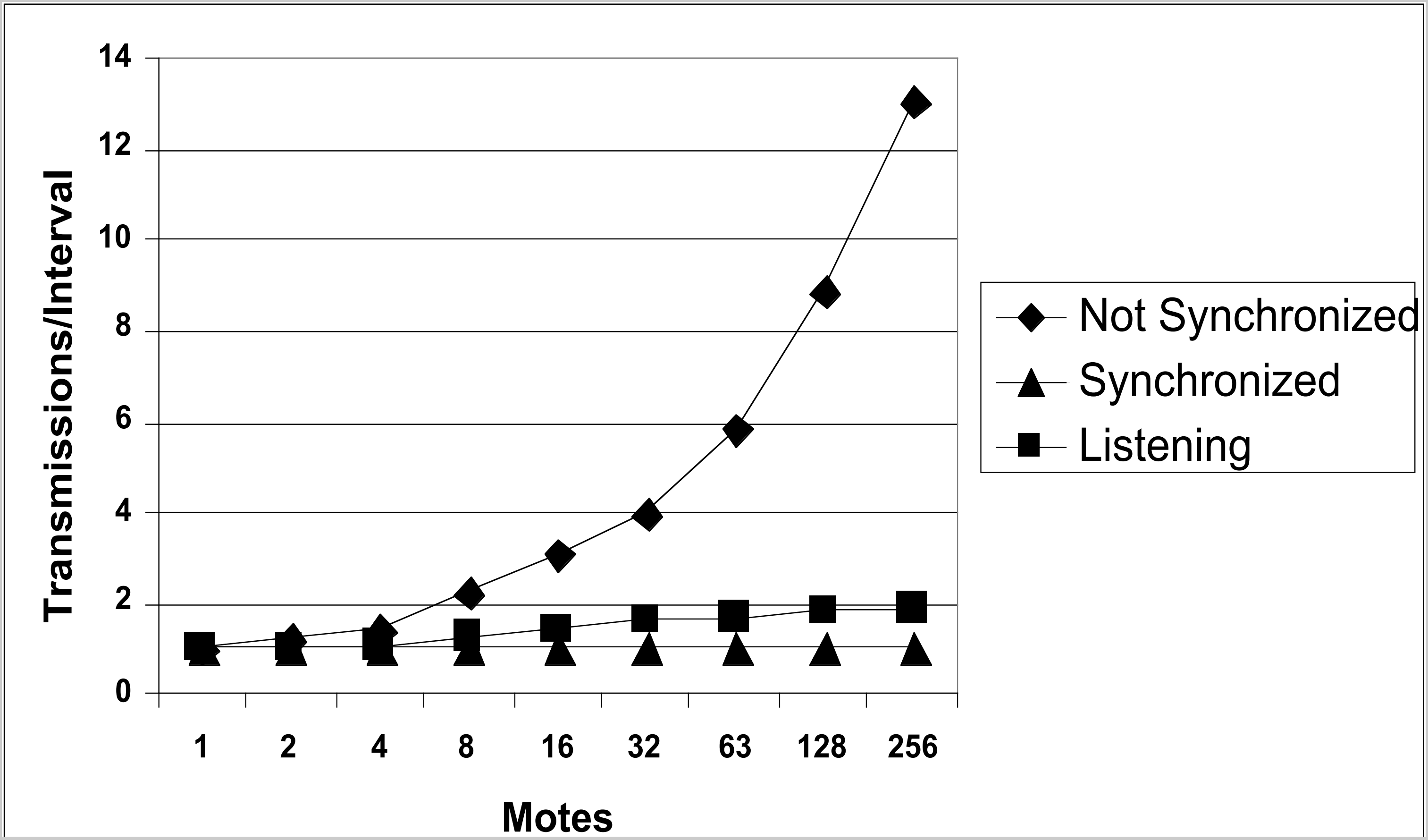
Short Listen Effect Prevention

- Add a listening period: *t* from [0.5τ, τ]



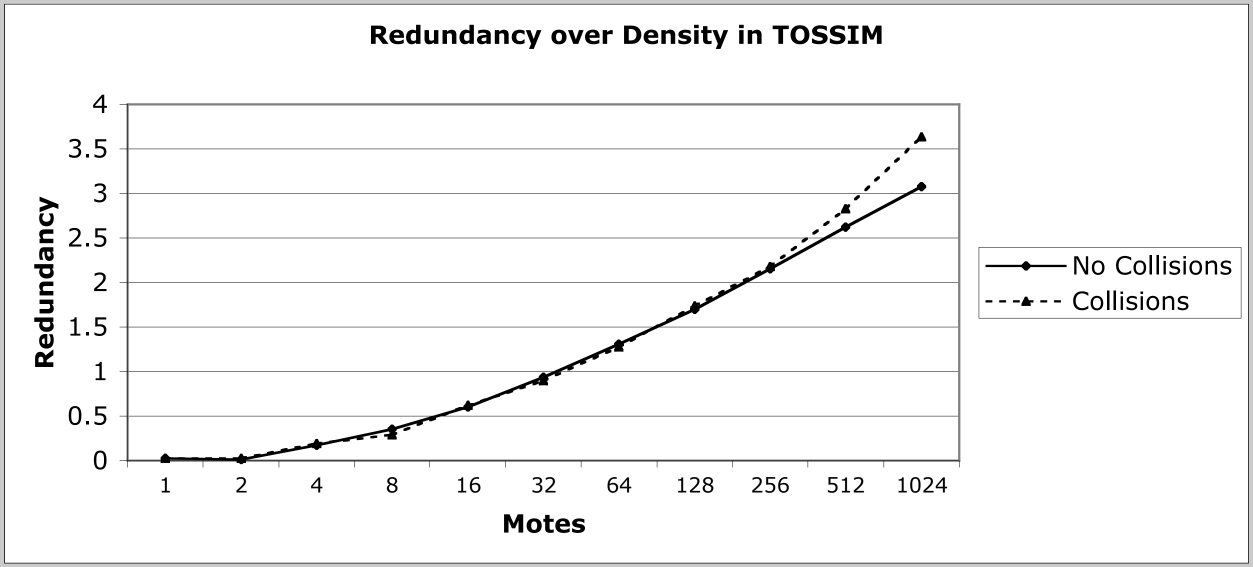
Listen-only period

Effect of Listen Period
(algorithmic simulator)



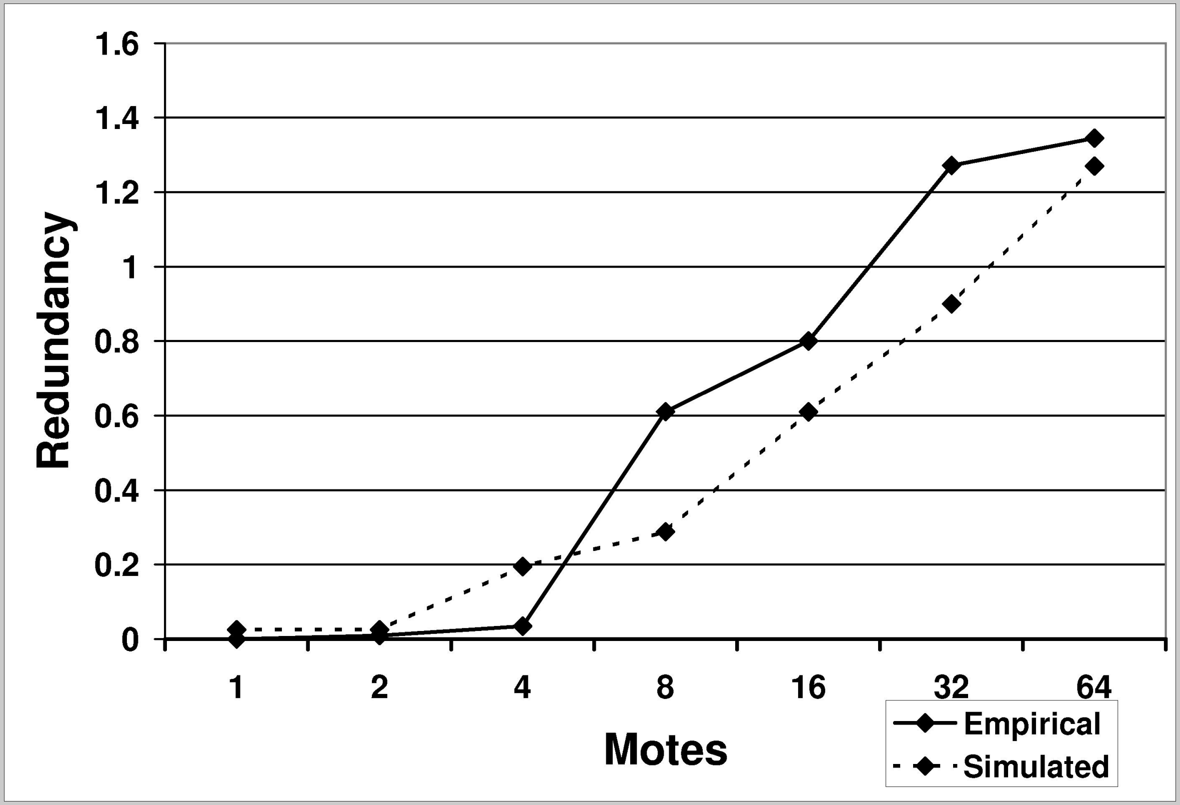
Multihop Network
(TOSSIM)

- Redundancy: $\frac{(transmissions + receptions)}{intervals} - k$
- Nodes uniformly distributed in 50'x50' area
- Logarithmic scaling holds



Empirical Validation (TOSSIM and deployment)

- 1-64 motes on a table, low transmit power



Maintenance Overview

- Trickle maintains a per-node communication rate
- Scales logarithmically with density, to meet the per-node rate for the worst case node
- Communication rate is really a number of transmissions *over space*

Outline

- Data dissemination
- Trickle algorithm
- Experimental methodology
- Maintenance
- Propagation
- Future Work

Interval Size Tradeoff

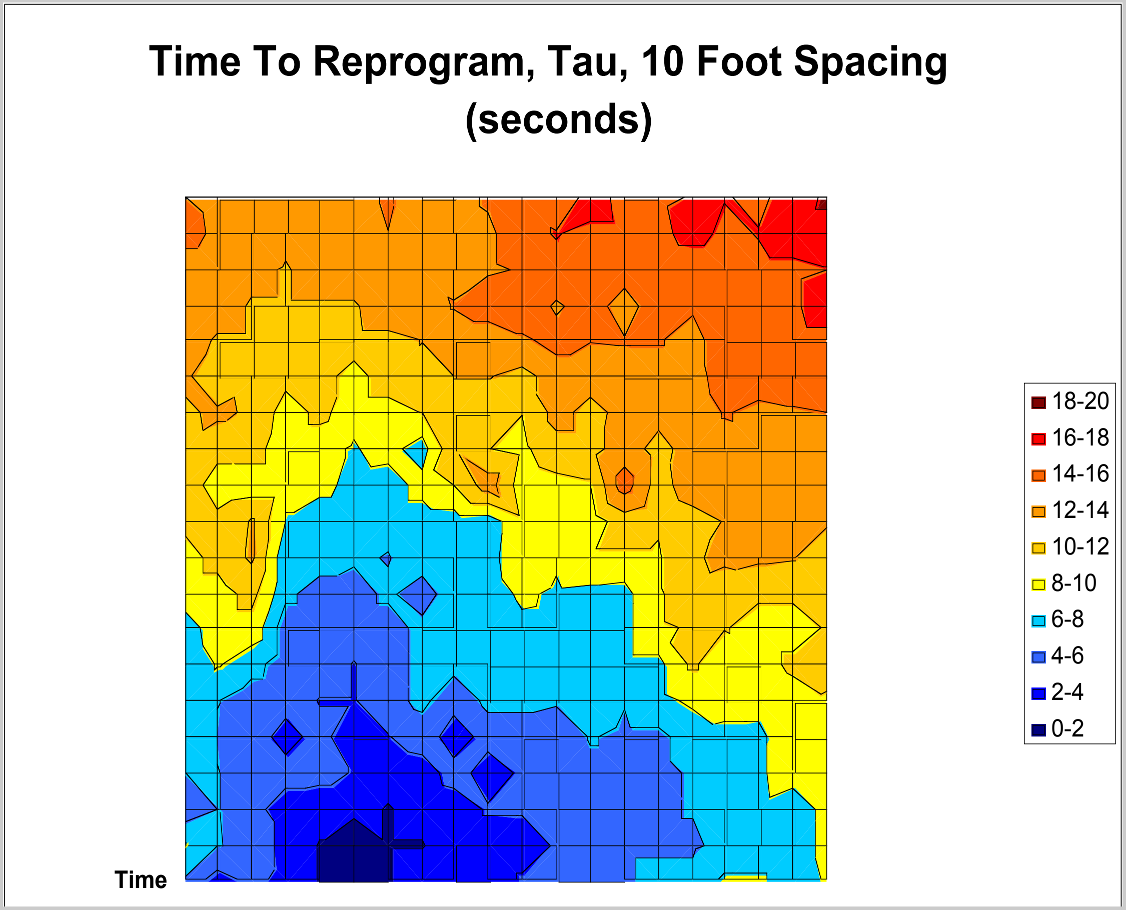
- Large interval τ
  - Lower transmission rate (lower maintenance cost)
  - Higher latency to discovery (slower propagation)
- Small interval τ
  - Higher transmission rate (higher maintenace cost)
  - Lower latency to discovery (faster propagation)
- Examples (k=1)
  - At τ = 10 seconds: 6 transmits/min, discovery of 5 sec/hop
  - At τ = 1 hour: 1 transmit/hour, discovery of 30 min/hop

Speeding Propagation

- Adjust τ: τ_l, τ_h
- When τ expires, double τ up to τ_h
- When you hear newer metadata, set τ to τ_l
- When you hear newer data, set τ to τ_l
- When you hear older metadata, send data

Simulated Propagation

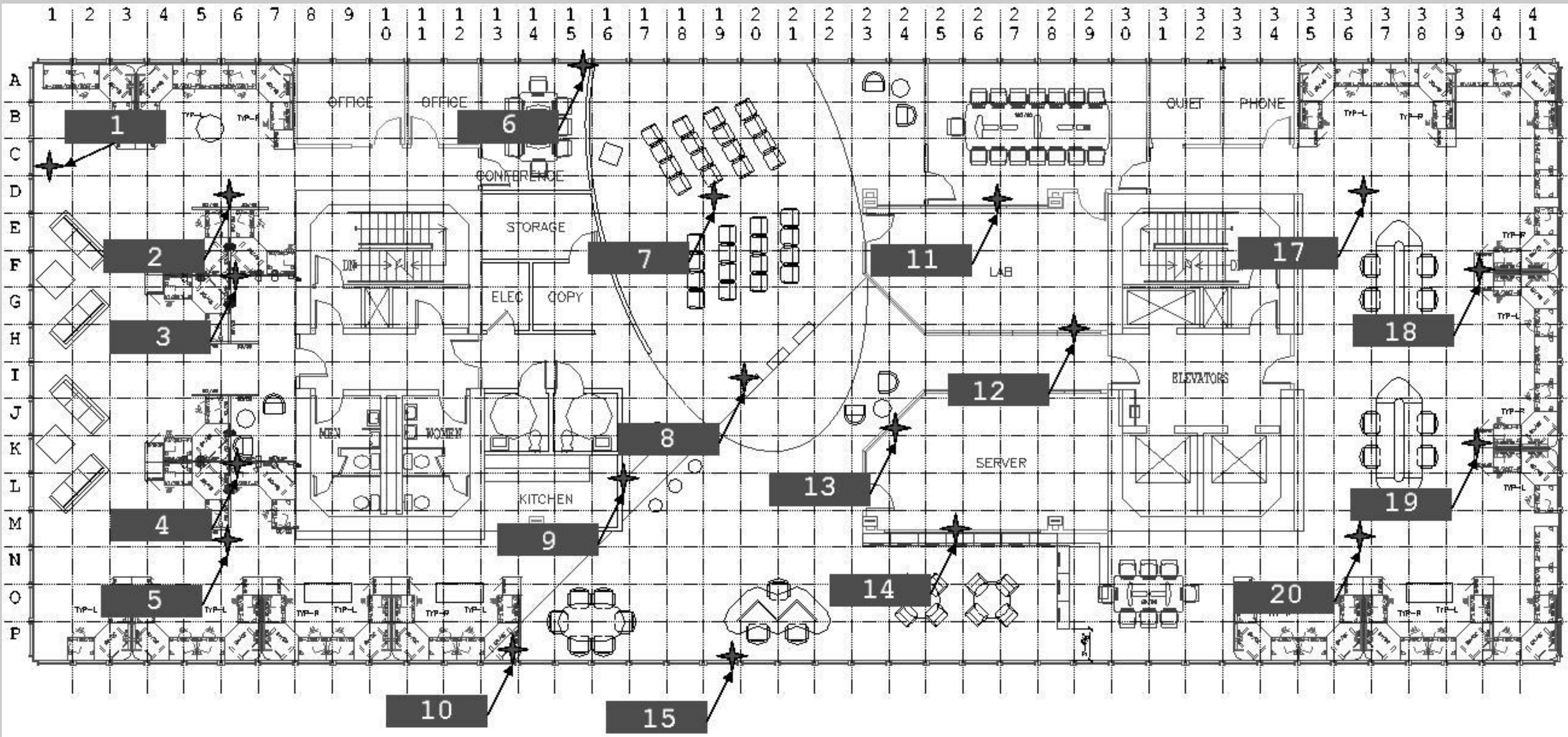
- New data (20 bytes) at lower left corner
- 16 hop network
- Time to reception in seconds
- Set τ_l = 1 sec
- Set τ_h = 1 min
- 20s for 16 hops
- Wave of activity



Empirical Propagation

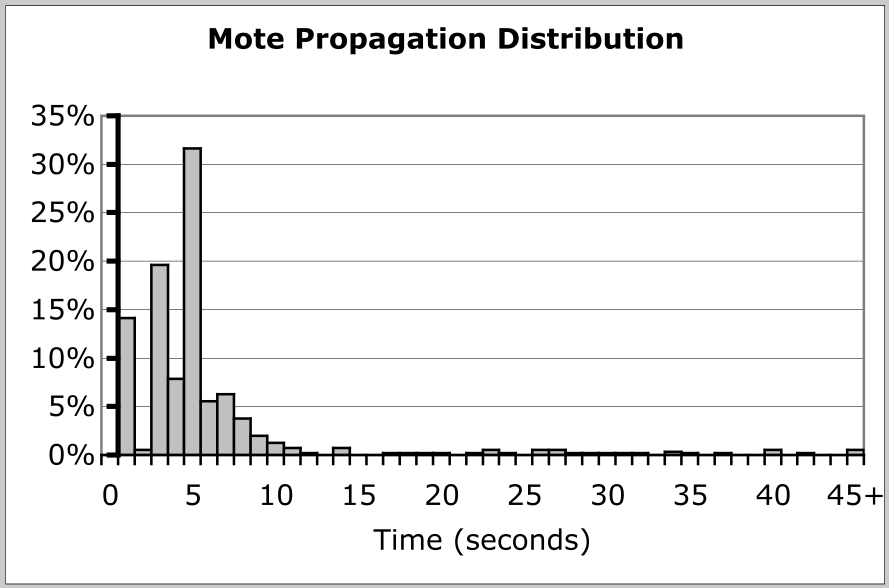
- Deployed 19 nodes in office setting
- Instrumented nodes for accurate installation times
- 40 test runs

Network Layout
(about 4 hops)



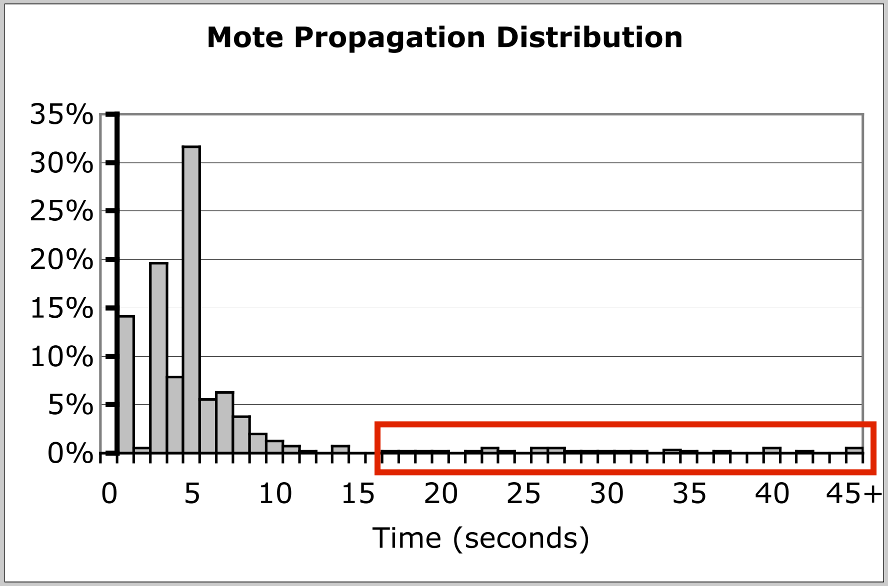
Empirical Results

k=1, τ_l=1 second, τ_h=1 minute

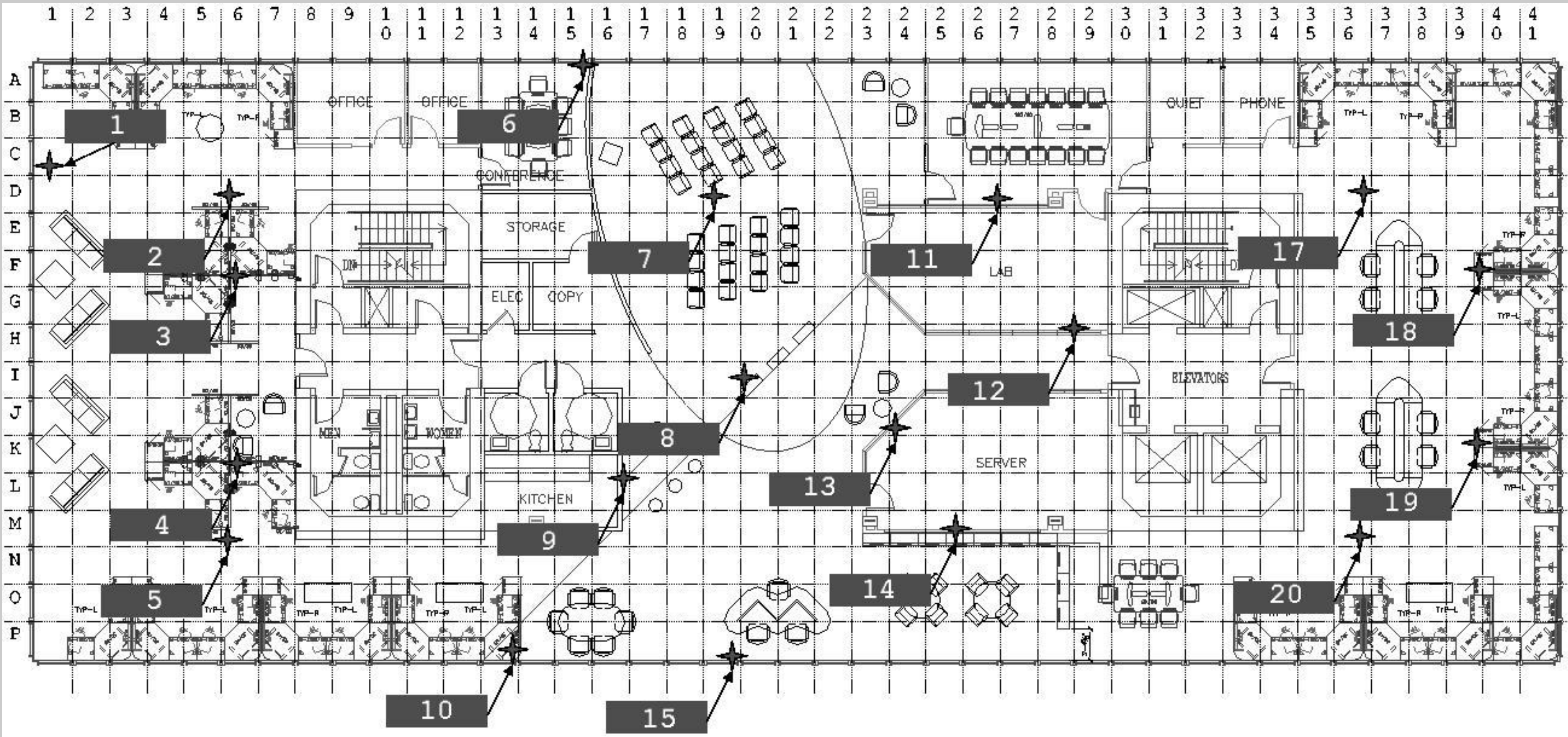


Empirical Results

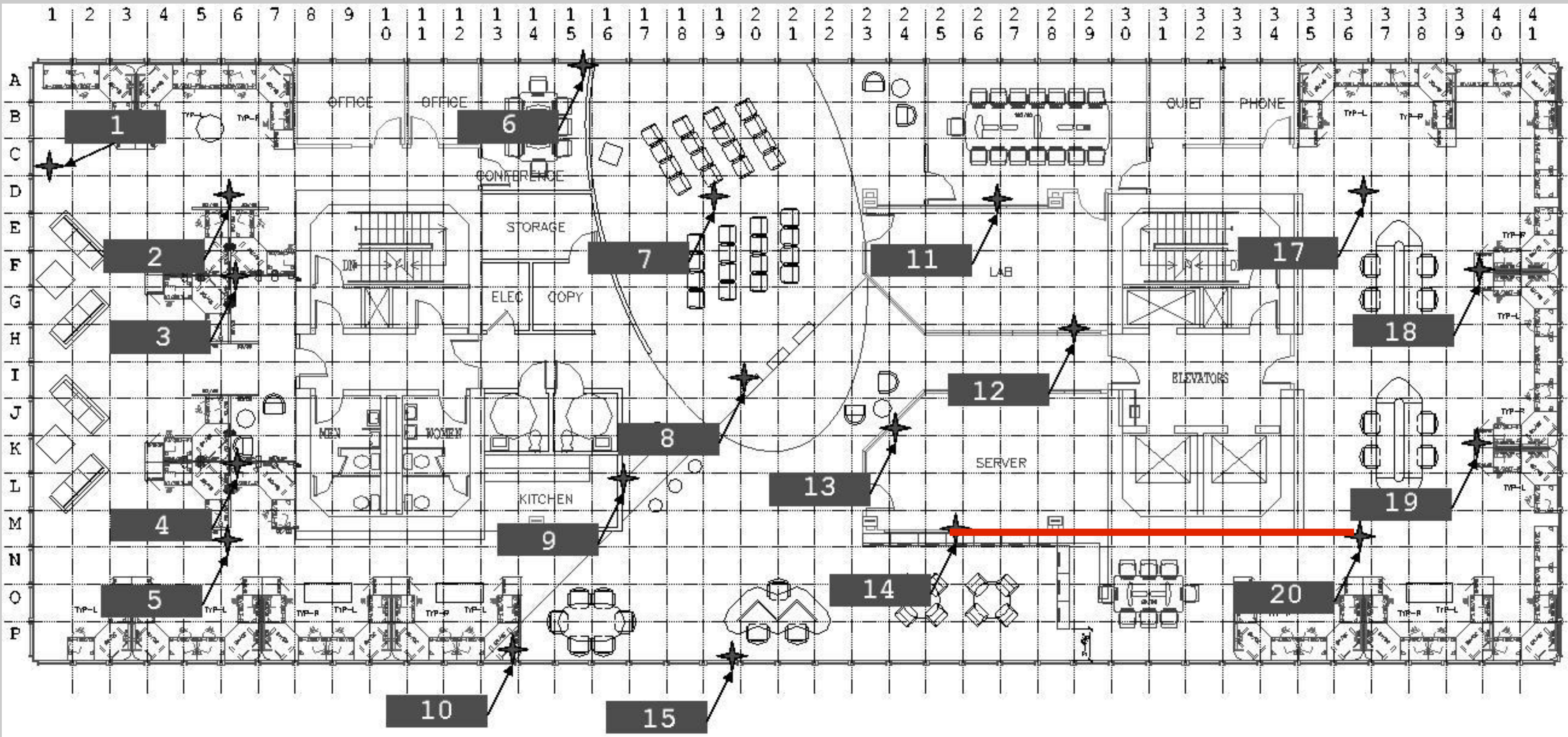
k=1, τ_l=1 second, τ_h=1 minute



Network Layout
(about 4 hops)

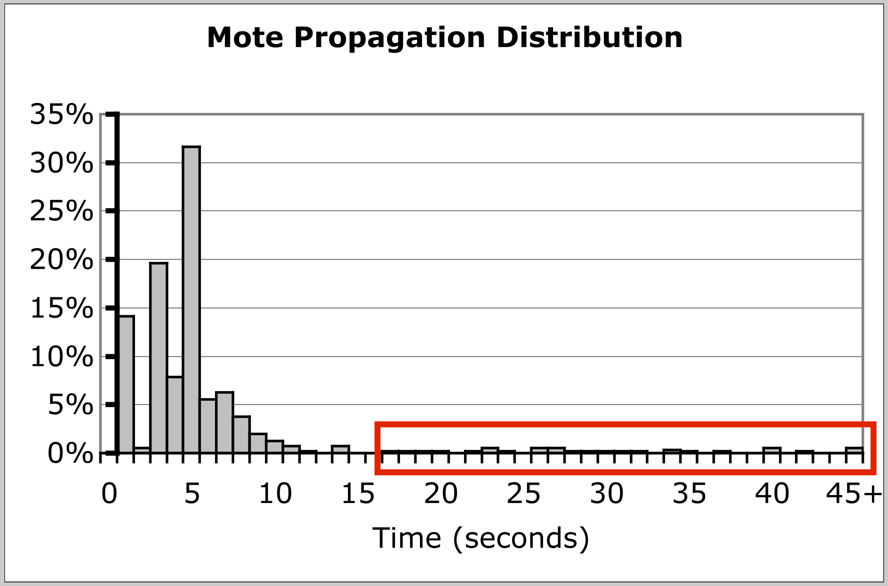


Network Layout
(about 4 hops)



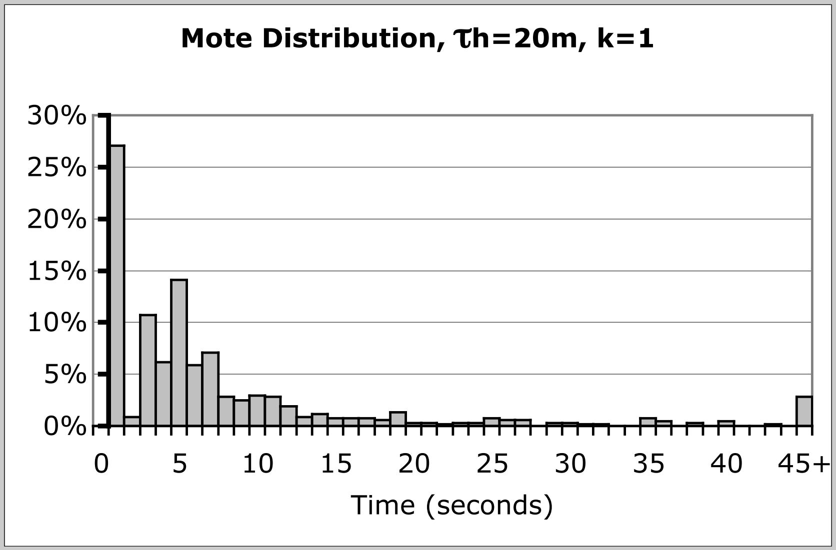
Empirical Results

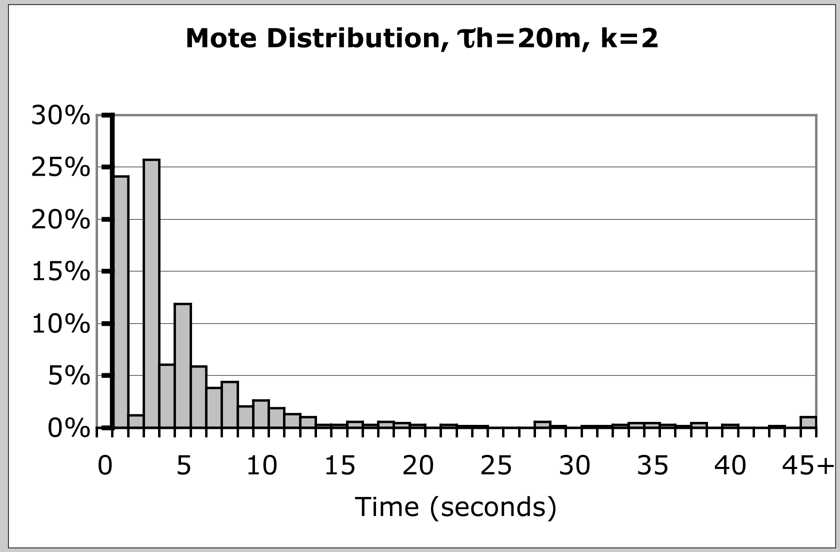
k=1, τ_l=1 second, τ_h=1 minute



- A single, lossy link can cause a few stragglers

Changing th to 20 minutes





- Reducing maintenance twenty-fold degrades propagation rate slightly
- Increasing redundancy ameliorates this

Outline

- Data dissemination
- Trickle algorithm
- Experimental methodology
- Maintenance
- Propagation
- Future Work and Conclusion

Extended and Future Work

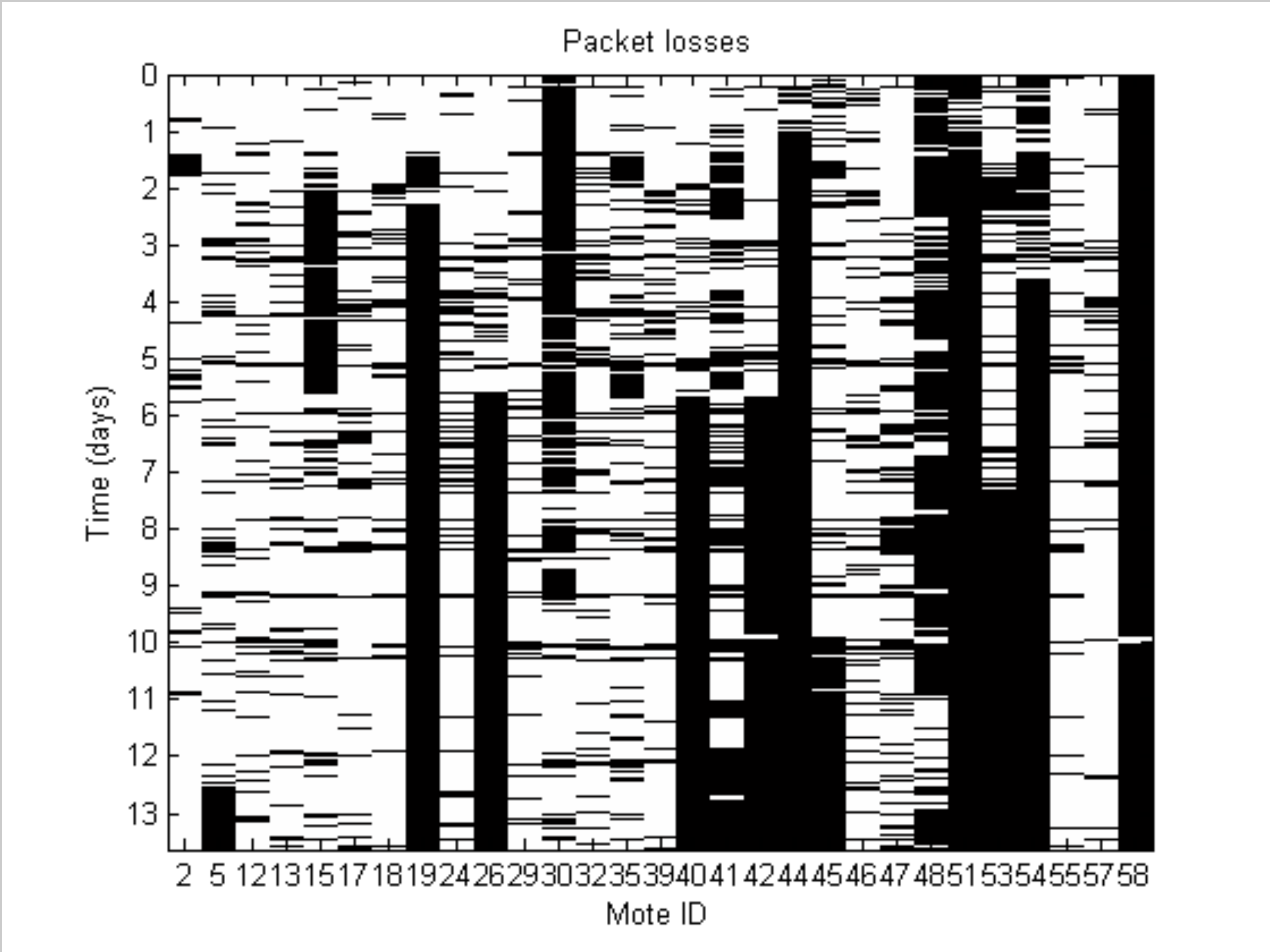
- Further examination of τ_l, τ_h and *k* needed
- Reducing idle listening cost
- Interaction between routing and dissemination
  - Dissemination must be slow to avoid the broadcast storm
  - Routing can be fast

Conclusions

- Trickle scales logarithmically with density
- Can obtain rapid propagation with low maintenance
  - In example deployment, maintenance of a few sends/hour, propagation of 30 seconds
- Controls a transmission rate over space
  - Coupling between network and the physical world
- Trickle is a nameless protocol
  - Uses wireless connectivity as an implicit naming scheme
  - No name management, neighbor lists…
  - Stateless operation (well, eleven bytes)

Questions

Sensor Network Behavior



Energy Conservation

- Snooping can limit energy conservation
- Operate over a logical time broken into many periods of physical time (duty cycling)
- Low transmission rates can exploit the transmit/receive energy tradeoff

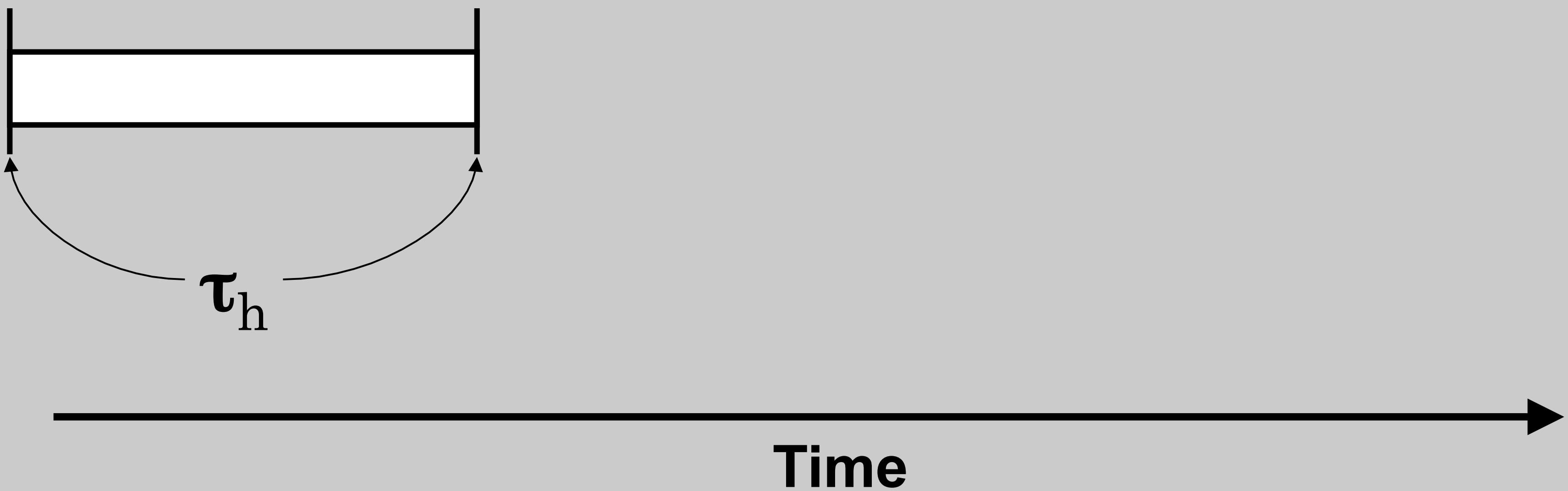
Use an Epidemic Algorithm?

- Epidemics can scalably disseminate data
- But end to end connectivity is the primitive (IP)
  - Overlays, DHTs, etc.
- Sensor nets have a local wireless broadcast

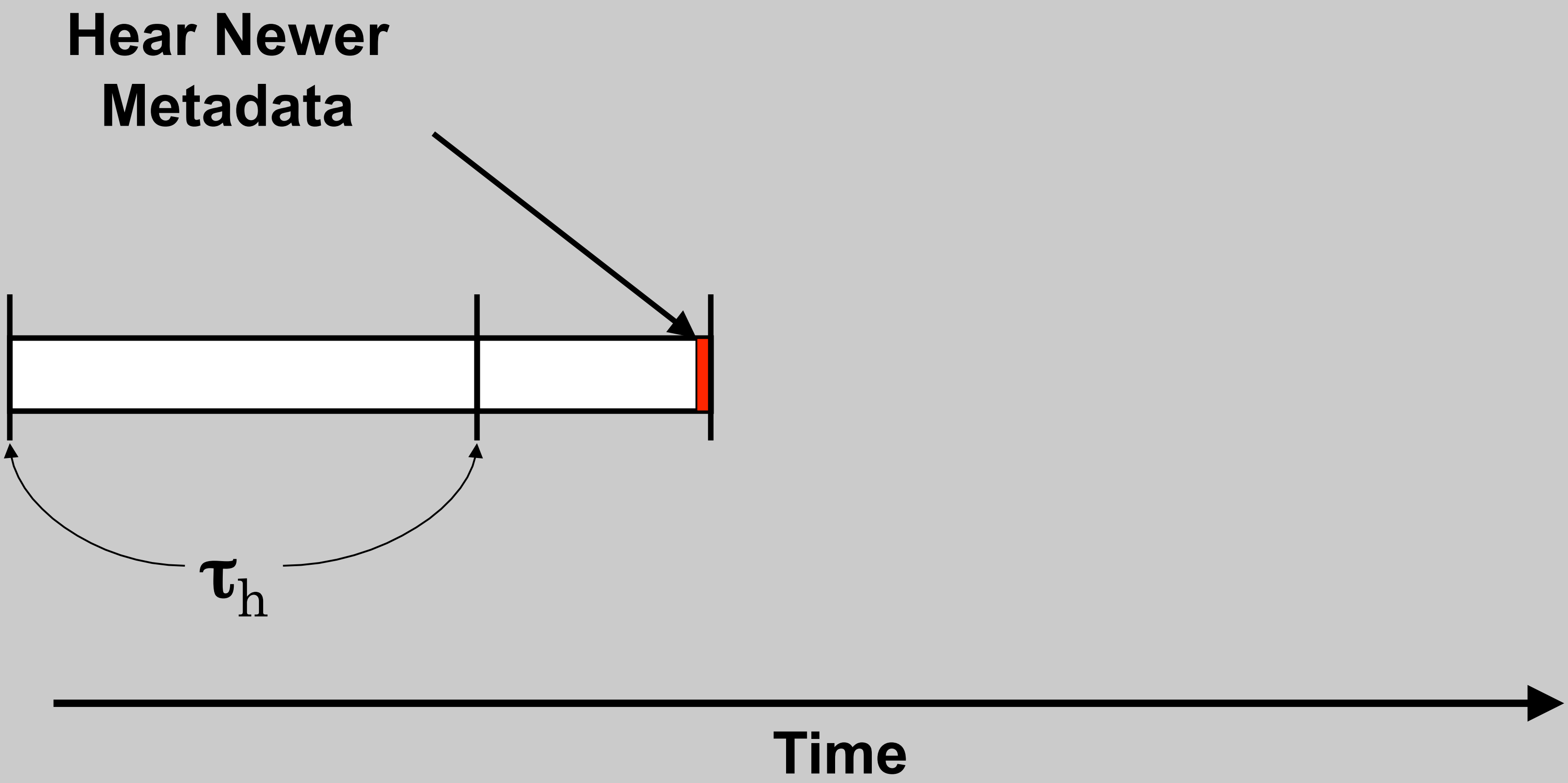
Use a Broadcast?

- Density-aware operation (e.g., pbcast)
  - Avoid the broadcast storm problem
- Broadcasting is a discrete phenomenon
  - Imposes a static reachable node set
- Loss, disconnection and repopulation
- We could periodically rebroadcast…
  - When to stop?

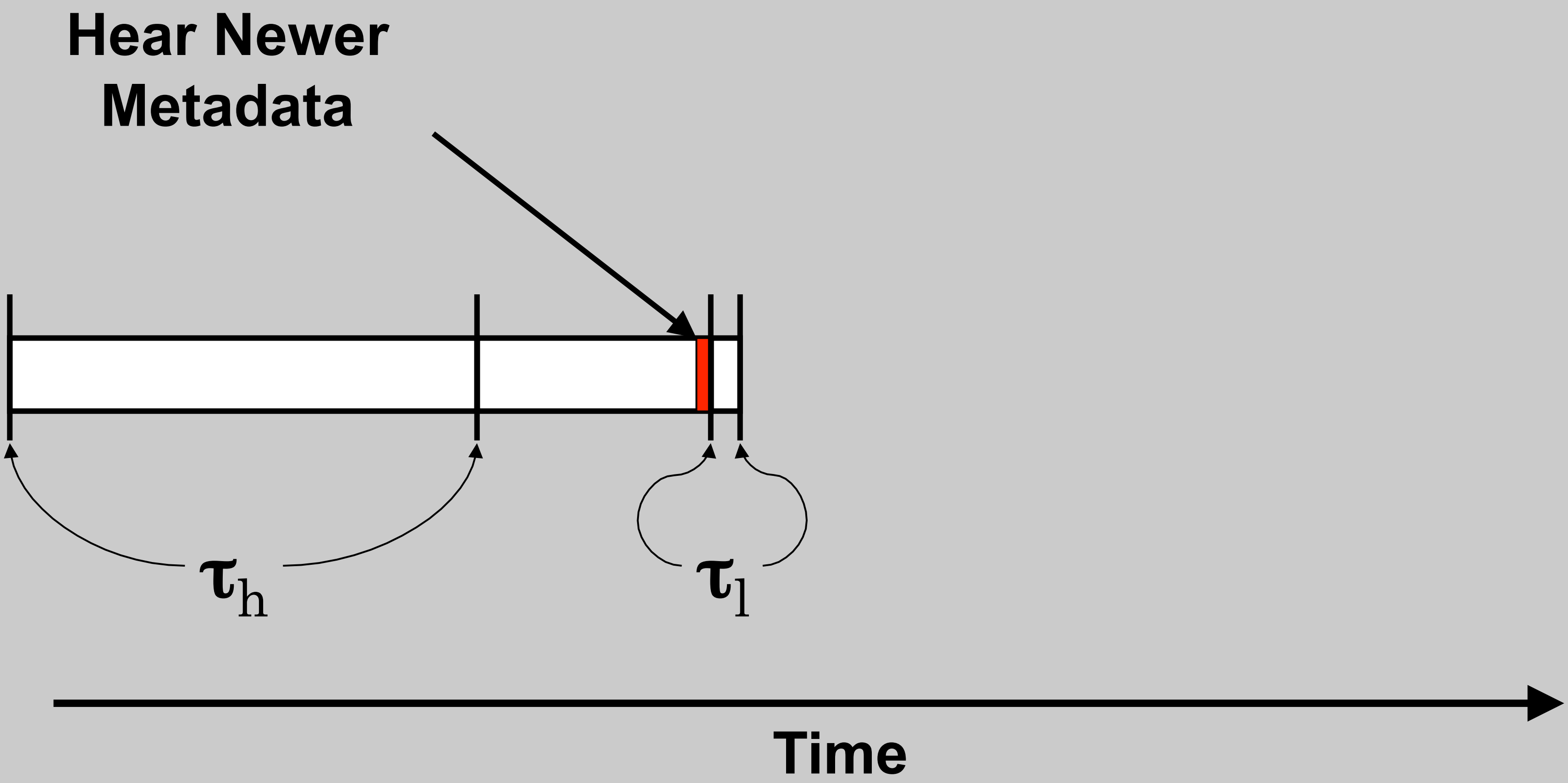
Rate Change Illustration



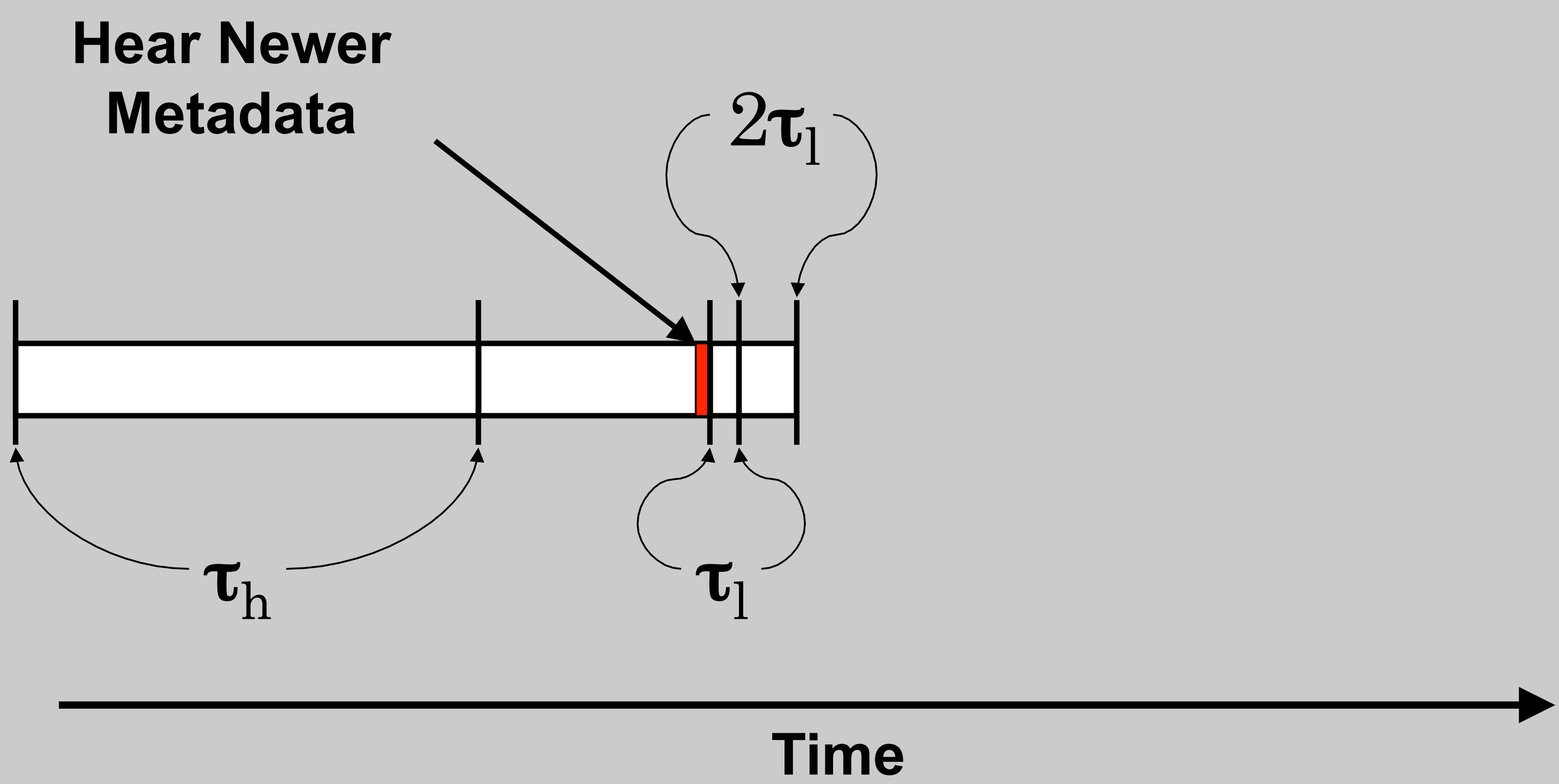
Rate Change Illustration



Rate Change Illustration



Rate Change Illustration



Rate Change Illustration