# 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, two minutes 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, $t$
- Choose a redundancy constant $k$
  - $k = (\text{receptions} + \text{transmissions})$
  - In an interval of length $t$
- Trickle keeps the rate as close to $k/t$ as possible
Trickle Algorithm

- Time interval of length \( t \)
- Redundancy constant \( k \) (e.g., 1, 2)
- Maintain a counter \( c \)
- Pick a time \( t \) from \([0, t]\)
- 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 \( t \), pick a new \( t \)
Example Trickle Execution

transmission  suppressed transmission  reception

c

k=1

time

NSDI, Mar 2004
Example Trickle Execution

\[ k=1 \]

\[ t_{A1} \]
Example Trickle Execution

\[ k=1 \]

transmission \quad suppressed transmission \quad reception

NSDI, Mar 2004
Example Trickle Execution

\[ \text{transmission} \quad \text{suppressed transmission} \quad \text{reception} \]
Example Trickle Execution

- Transmission
- Suppressed transmission
- Reception

$k=1$

At time $t_{A1}$, transmission is suppressed and reception occurs at time $t_{C1}$.
Example Trickle Execution

- Transmissions and receptions are indicated on the time axis.
- Transmissions are marked with a solid line, suppressed transmissions with a dashed line.
- Reception is indicated with a box.

The diagram shows the sequence of transmissions and receptions in a Trickle execution:

- At time $t_{A1}$, there is a transmission from node A to node C.
- At time $t_{B1}$, there is a transmission from node B to node C, suppressed after the reception of $t_{A1}$.
- At time $t_{C1}$, there is a transmission from node C, possibly indicating the end of the reception period.

$k=1$ indicates the initial state of the Trickle execution.
Example Trickle Execution

\[ \text{k=1} \]

\[ t_{A1}, t_{B1}, t_{C1} \]

transmission | suppressed transmission | reception

NSDI, Mar 2004
Example Trickle Execution

\[ \begin{align*}
    t_A^1 & \quad t_B^1 & \quad t_C^1 \\
    t_B^2 & \\
\end{align*} \]

- Transmission
- Suppressed transmission
- Reception

NSDI, Mar 2004
Example Trickle Execution

\[ \begin{align*}
  &k=1 \\
  &t_{A1} \\
  &t_{B1} \\
  &t_{C1}  \\
  &t_{B2} \\
  &t_{C2} \\
\end{align*} \]
Example Trickle Execution

\[ \text{transmission} \quad \text{suppressed transmission} \quad \text{reception} \]

\[ k=1 \]

\[ t_{A1} \quad t_{A2} \]

\[ t_{B1} \quad t_{B2} \]

\[ t_{C1} \quad t_{C2} \]
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)

Transmissions/Interval

Motes

- ■ 60%
- ▲ 40%
- ◆ 20%
- — 0%
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:

```
A
B
C
D
```

![Diagram showing time and transmission order](image_url)
Short Listen Effect Prevention

- Add a listening period: $t$ from $[0.5, t]$
Effect of Listen Period
(algorithmic simulator)

Transmissions/Interval

Motes

- Not Synchronized
- Synchronized
- Listening
Multihop Network (TOSSIM)

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

**Redundancy over Density in TOSSIM**

NSDI, Mar 2004
Empirical Validation (TOSSIM and deployment)

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

![Graph showing redundancy vs motes](image)
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 $t$
  - Lower transmission rate (lower maintenance cost)
  - Higher latency to discovery (slower propagation)

- Small interval $t$
  - Higher transmission rate (higher maintenance cost)
  - Lower latency to discovery (faster propagation)

- Examples ($k=1$)
  - At $t = 10$ seconds: 6 transmits/min, discovery of 5 sec/hop
  - At $t = 1$ hour: 1 transmit/hour, discovery of 30 min/hop
Speeding Propagation

• Adjust $t$, $t_h$
• When $t$ expires, double $t$ up to $t_h$
• When you hear newer metadata, set $t$ to $t_h$
• When you hear newer data, set $t$ to $t_h$
• 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 $\tau = 1$ sec
- Set $\tau_h = 1$ 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

$k=1$, $\square_t=1$ second, $\square_h=1$ minute

Mote Propagation Distribution

Time (seconds)
Empirical Results

$k=1$, $\Box=1$ second, $\Box_h=1$ minute

Mote Propagation Distribution

Time (seconds)
Network Layout
(about 4 hops)
Network Layout
(about 4 hops)
• A single, lossy link can cause a few stragglers

$k=1$, $\square=1$ second, $\square_h=1$ minute
Changing th to 20 minutes

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

NSDI, Mar 2004
Outline

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

- Further examination of $q$, $q_h$ 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

Packet losses

Time (days)

Mote ID

2 5 12 13 15 17 18 19 24 26 29 30 32 33 35 39 40 41 42 44 45 46 47 48 51 53 54 55 57 58
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

\[ \square_h \]

Time

NSDI, Mar 2004
Rate Change Illustration

Hear Newer Metadata

Time

\( h \)
Rate Change Illustration

Hear Newer Metadata

Time

NSDI, Mar 2004
Rate Change Illustration

Hear Newer Metadata

Time

NSDI, Mar 2004
Rate Change Illustration

Hear Newer Metadata

\[ \text{Time} \]

\[ Q_h \]

\[ 2 \]

\[ \overline{Q_h} \]

\[ \overline{Q_h} \]