TiredZebra

Exploring Gossip Protocols in Sensor Networks

Paul Crews
Travis Lanham

Background: Trickle

- Protocol for software updates in wireless networks
  - General algorithm for eventual consistency

- Longevity of sensors is paramount, need to be ultra low power
  - Ex. Tracking sensors for animals (need to last on the order of years)

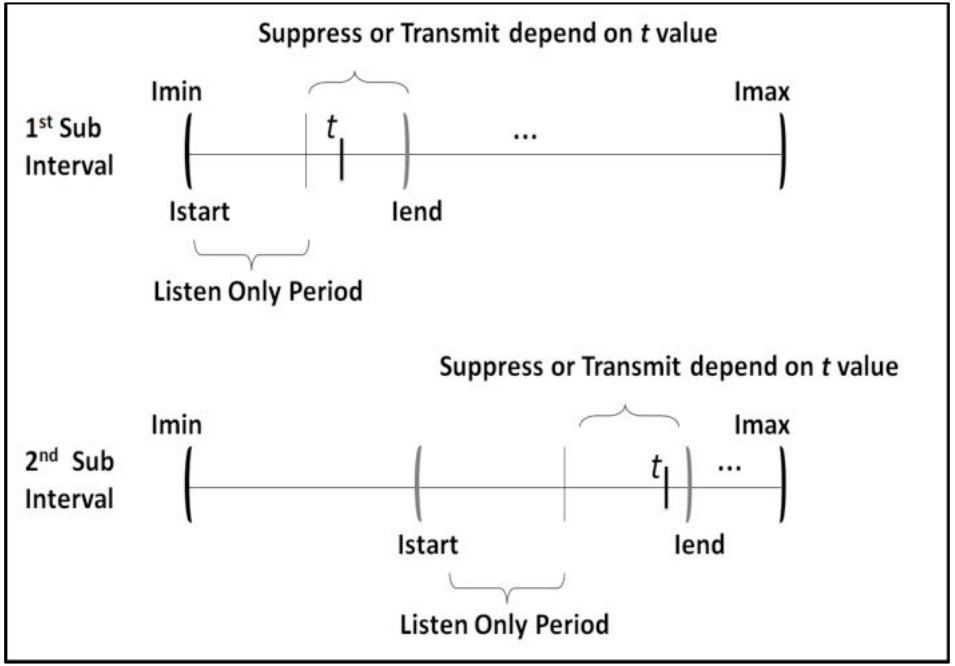
Background: TockOS

- Research OS at Stanford
- Embedded operating system for IoT devices
- Written in Rust for memory safety
- Good platform for low-power devices

Trickle Algorithm

- “Polite Gossip”: at random time point in interval, broadcasts most recent update (unless it has heard update broadcast by other nodes recently)
- Tunable parameters
  - T: transmission interval, higher T means fewer transmissions and longer propagation time, small T means shorter propagation time and more energy consumption
    - Tl is lower bound, Th is upper bound, when T expires it doubles next timeout (up to Th); if new update is heard, resets to Tl
  - K: when node is deciding to broadcast, if it has heard fewer than k broadcasts for the update then broadcasts it, otherwise remains silent

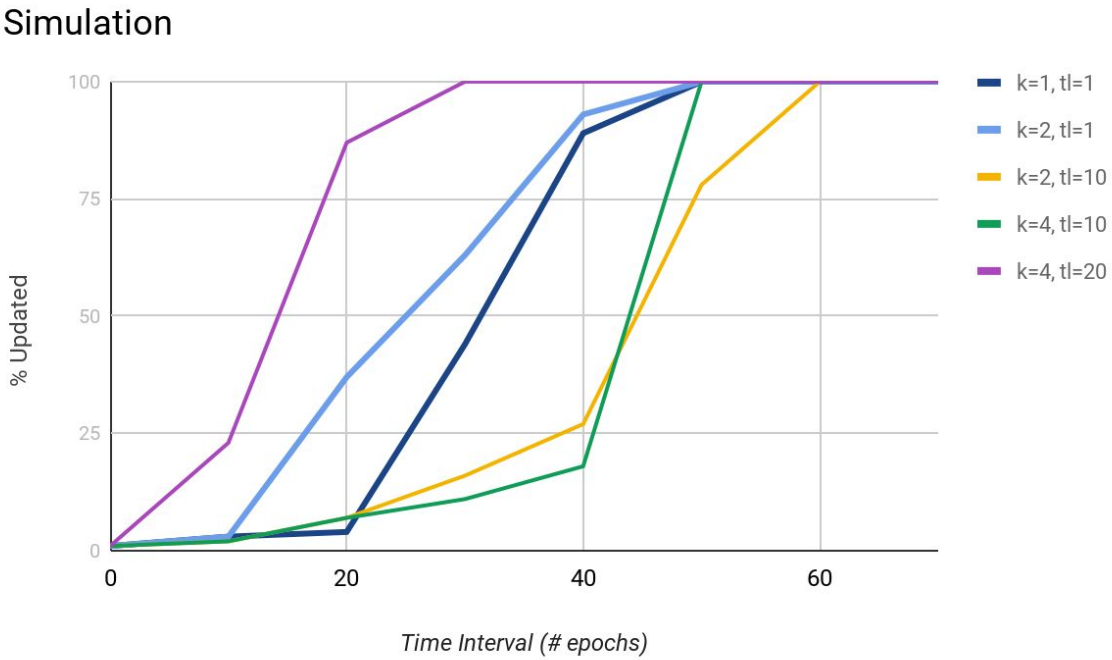
Trickle Algorithm



ZebraSim

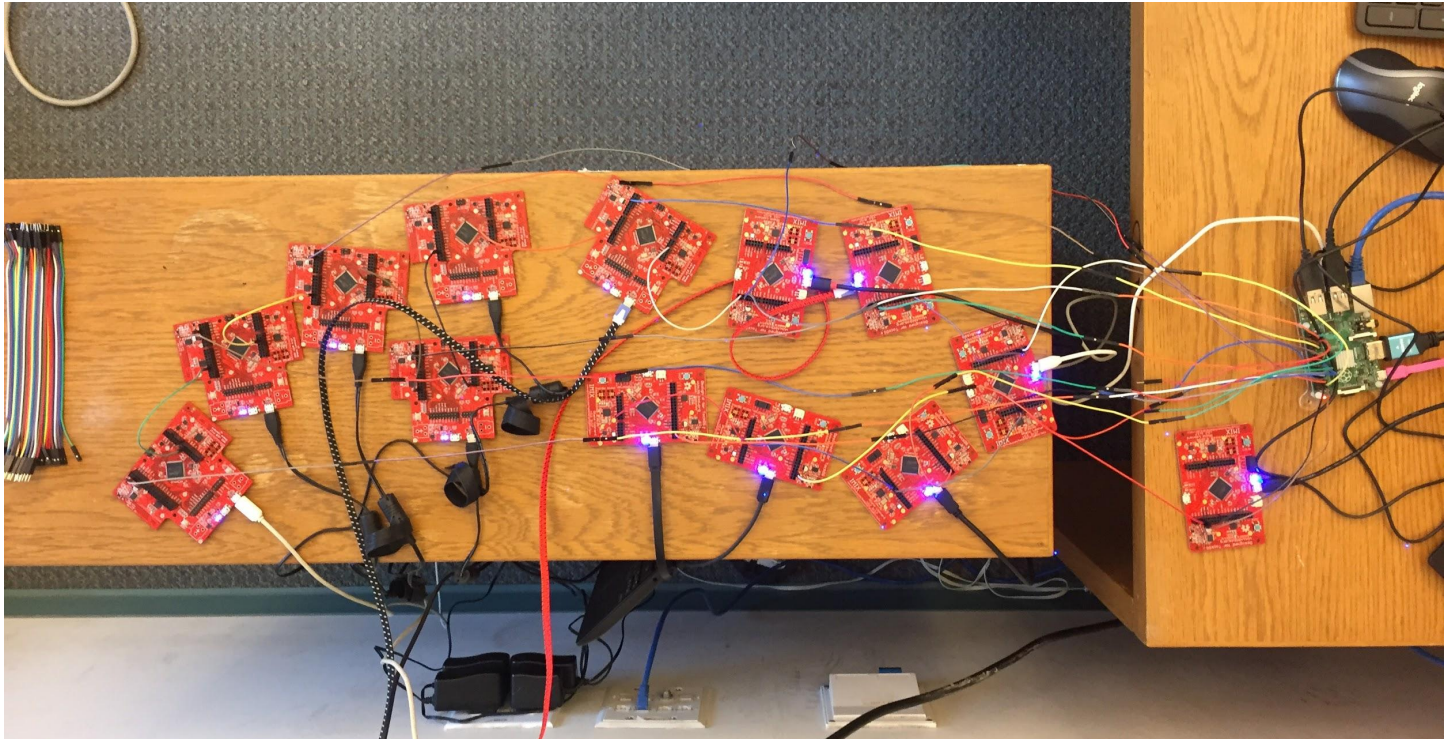
- Software simulation for algorithm
- Original paper had lower fidelity simulator
- Configurable k, Th, Tl, distribution of nodes (and movement), simulation duration (number of epochs), loss rate

ZebraSim Results



Tock Simulation

OBJ



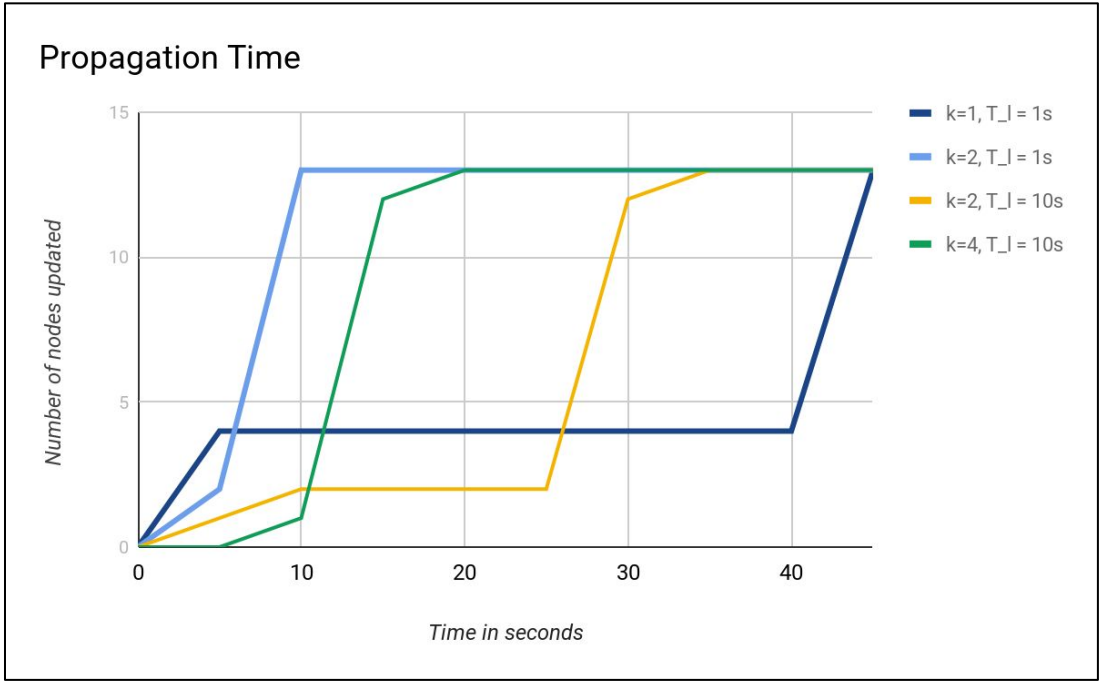
Tock Simulation

- Used 13 Imix boards running Tock
- Evaluated three versions of the Trickle algorithm:
  - Single-hop, dense network
  - Multi-hop network
  - Sleepy Trickle
- Measured total propagation time, # packets sent

Single-Hop Network

- Tested four different pairs of k, T_l
  - Low k: susceptible to hidden terminal
  - High T_l: high(er) latency
- Claim: For dense networks, best option is large K, T_l

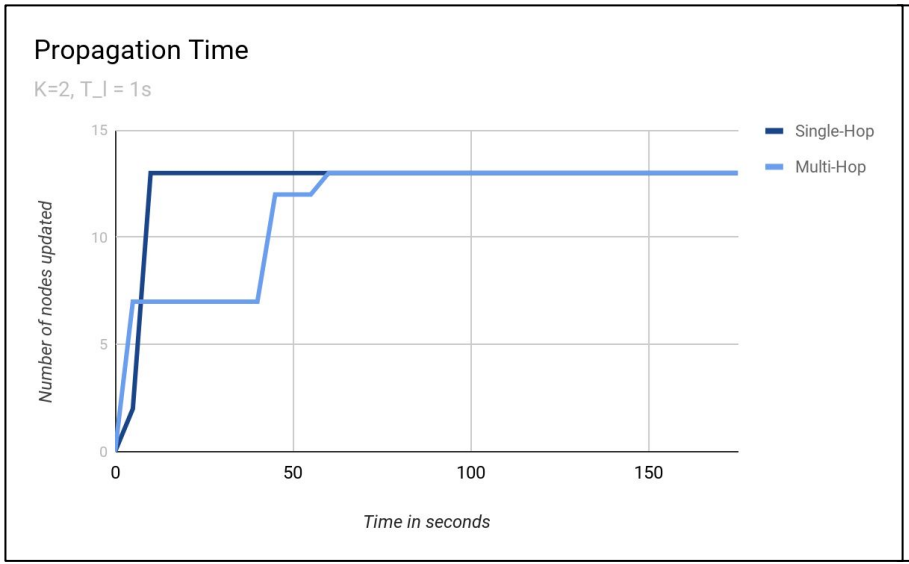
Single-Hop Network

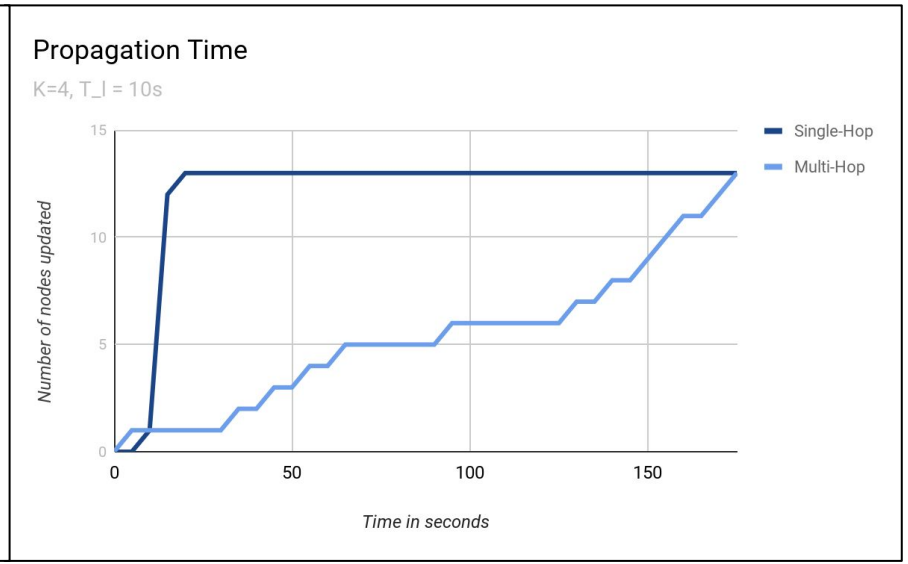


Multi-Hop Network

- Simulated a sparse, multi-hop network
- Greatly increased tail latency
- Vulnerable to bad nodes, “hidden” terminal

Multi-Hop Network





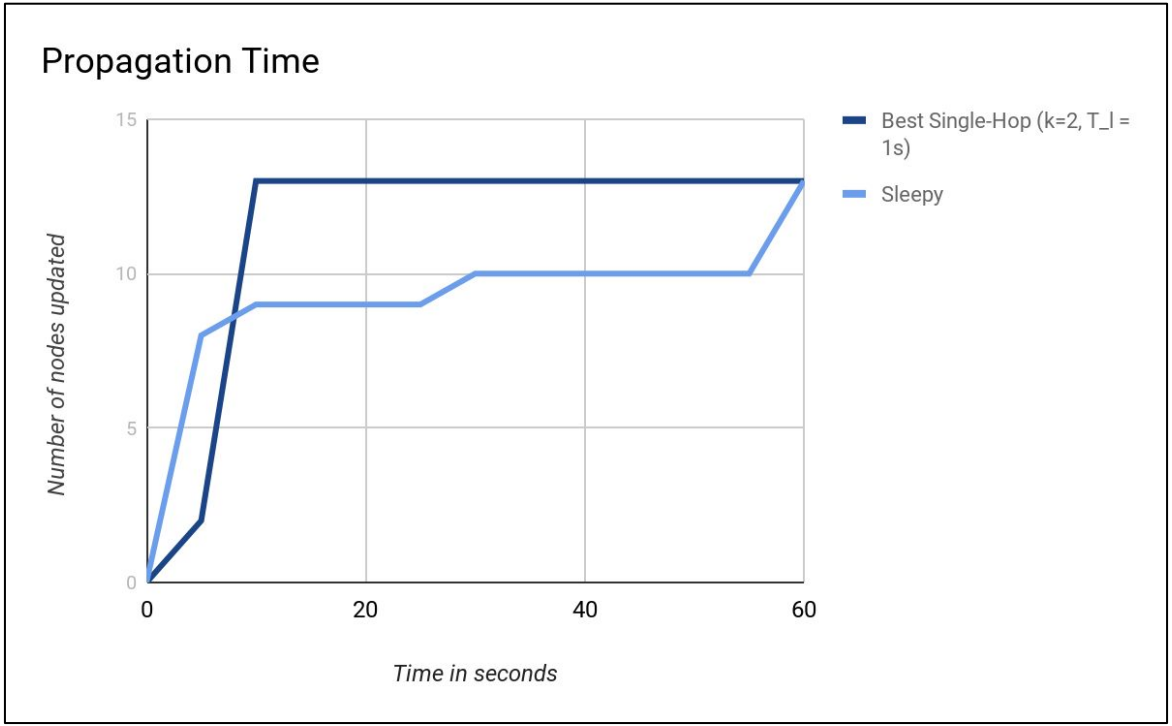
Sleepy Trickle

- Problem: Trickle is transmit efficient, still have to listen constantly
- Insight: Make more assumptions about network
  - There are router nodes and leaf nodes
  - Leaf nodes are connected to >= 1 router node
- Idea: Leaf nodes can sleep
  - Set k = 1, sleep after not transmitting for a full interval of T_h

Sleepy Trickle

- Setup
  - K=1 for leaf nodes, K=4 for router nodes
  - 2 router nodes
  - T_l = 1s for everyone
- During update
  - 8 total messages for normal (~1 message/sec)
  - 24 messages for sleepy (~0.4 message/sec)
- Stable state
  - ~2 messages per interval for both
  - Dominated by router nodes for sleepy

Sleepy Trickle



Demo

Conclusions

- Trickle offers ability to propagate information through sensor network at low cost
- Eventually consistent
- In the normal case converges quickly, sparser networks require more resources for fast propagation (clearly demonstrates the tradeoff and provides parameters for appropriate configuration)
- Our contributions
  - Algorithm modifications (sleepy)
  - Improved simulator
  - Ported to new OS/environment
  - Higher fidelity hardware testbed

Questions

Links

- TockOS
  - GitHub: https://github.com/helena-project/tock
  - Webpage: tockos.org
- Code
  - Code: https://github.com/lanhamt/trickle
  - Tock branch: https://github.com/ptcrews/tock/tree/cs244b_trickle