SCoRe: Scheduling Commands and Responses for Multihop Low-power Wireless Network

Mingyu Park, Jeongyeup Paek

Department of Computer Science & Engineering, Chung-Ang University

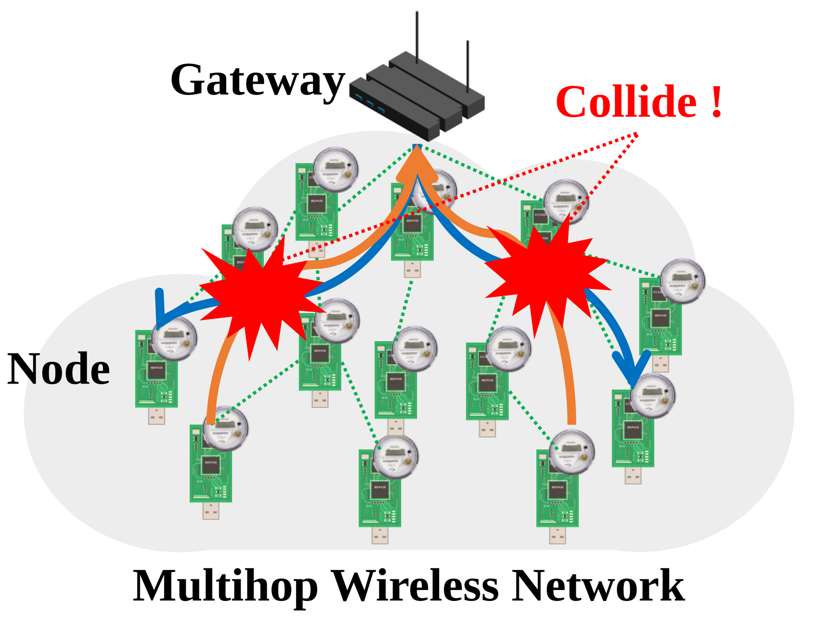
http://nsl.cau.ac.kr

Introduction

Collision with commands and responses

- Internet of Things (IoT) technology is being applied to a variety of fields. (e.g., smart factory, smart grid AMI, smart market, smart hospital, etc.)
- Most such IoT applications require a mechanism to **disseminate commands** and **collect responses**.
- If a large number of nodes send data simultaneously over a wireless network, severe collisions may occur. On the other hand, if the nodes wait long enough to avoid collision, the responses will be delayed.
  ─ Trade off between **reliability** and **latency**

- Most of recent LLN studies focused only on either the **command dissemination phase** or the **response collection phase**.
- Since both command and response packets collide not only among themselves but also with each other, they mush be considered '**jointly**' for real-world IoT applications.



Multihop Wireless Network

SCoRe Design

Recursive gathering

- SCoRe allocates response timeslots as each node's **hop count** H, and command timeslots as M **which is the number of transmissions needed for dissemination**.
- A SCoRe node gathers *the number of desired timeslots for command dissemination and response collection* (***desired slots***) from each child for assignment.
- Each node **piggybacks** its *desired slots*, together with an **aggregate sum** of slots 1-hop children needs, in the upward route-notification packet. (e.g., DAO in RPL)

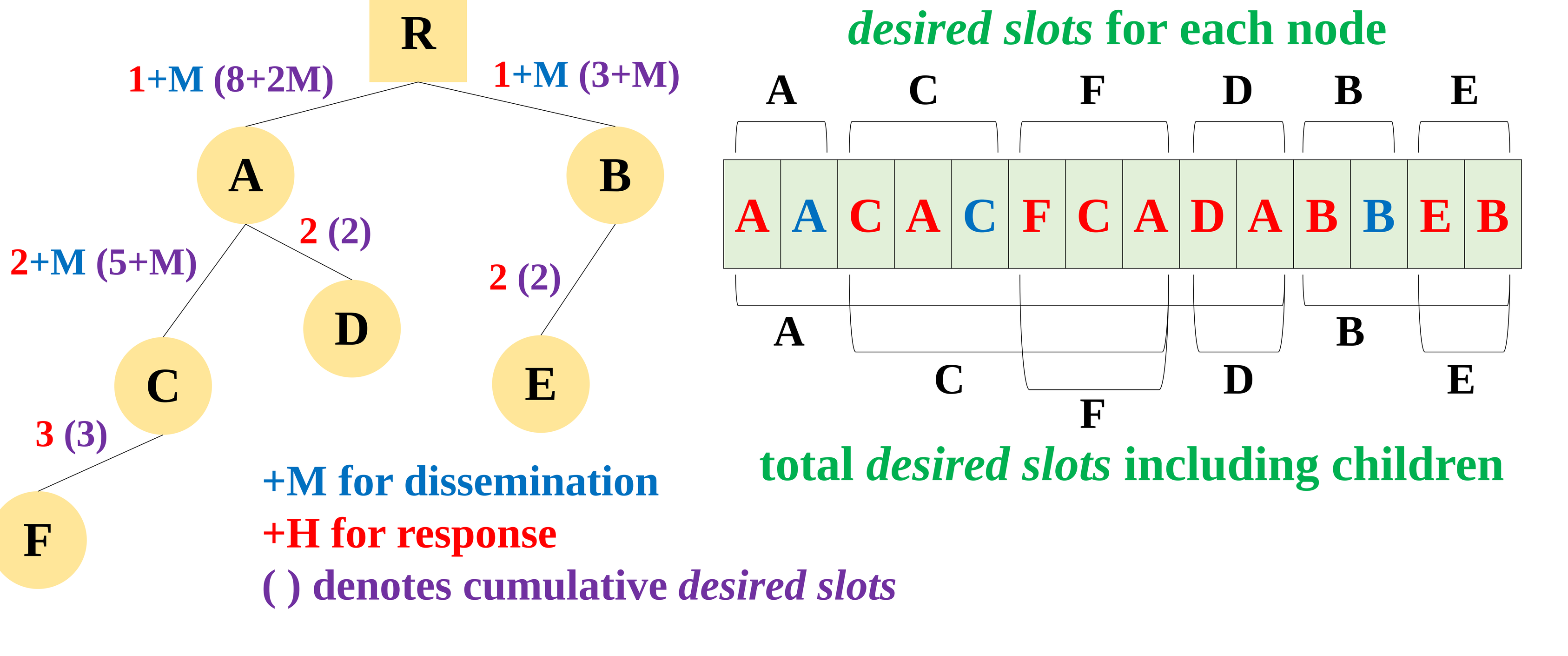
$$D_n = \sum_{i=0}^{K} D_i + H_n + M$$

D_n: *Desired slots* of node n ('s subtree)
D_i: *Desired slots* of i'th child
H_n: Hop count of node n
K: the number of children of node n
M: the number of transmissions for dissemination

Recursive scheduling

- A SCoRe root assigns timeslots to each 1-hop child, and the child uses the slots for its transmission.
- The child re-assigns **remaining timeslots** after it uses until the leaf node.

─ Reduces **overhead**, works in **non-storing** mode as well



Evaluation

Simulation environment

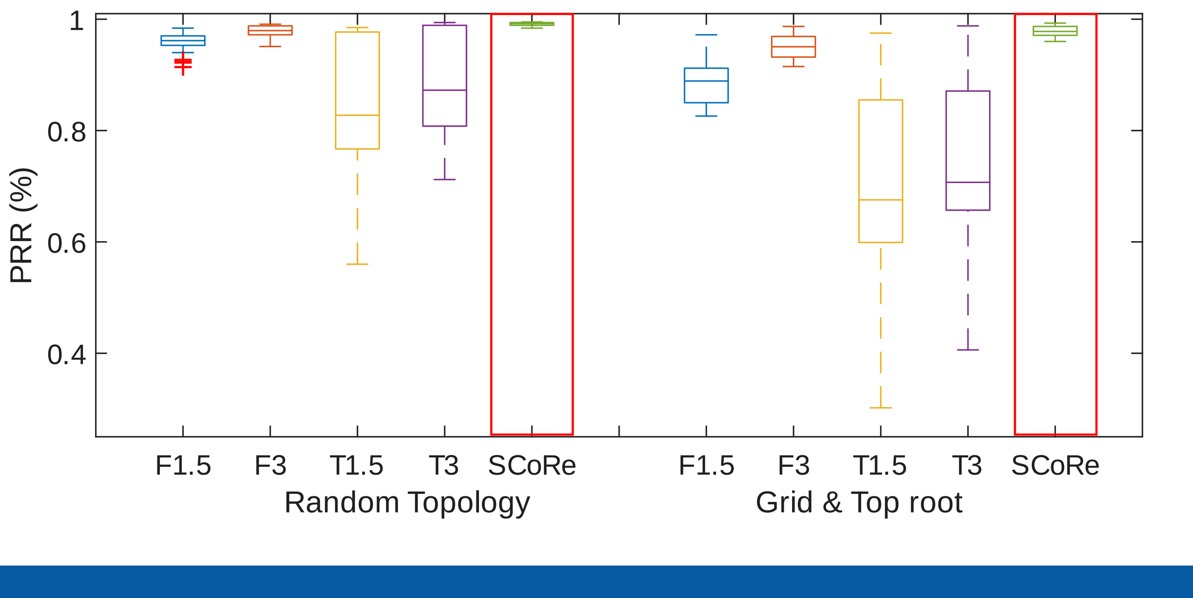
- Implemented on TinyOS 2.1.2, over RPL.
- Simulated on Cooja simulator with 31 Tmote Sky devices on both grid and random topology. (30 nodes and 1 root)
- Compared with two Algorithms : '**Flooding + RPL-collection**' and '**Trickle + RPL-collection**'

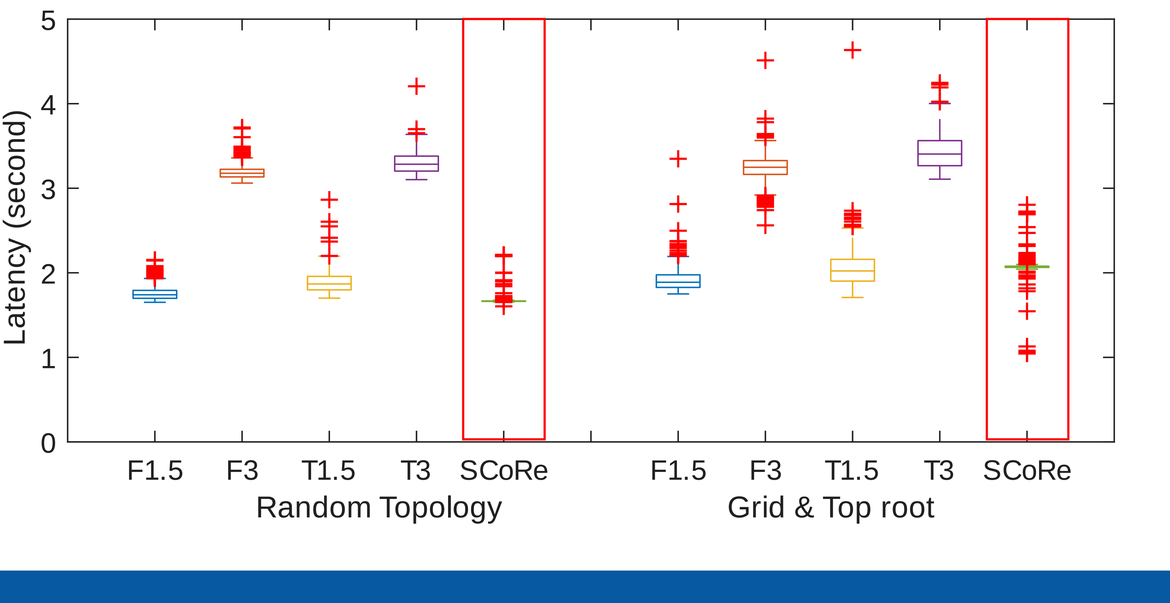
Parameters

- Random jitter range to mitigate the packet explosion problem for each compared algorithm.
  R, for response : **1.5 and 3 sec** (for trade-off between latency and PRR)
  C, for command : **100 ms**
- The number of dissemination M is set to **1**.
- Each timeslot is **20ms** for SCoRe's scheduling.

Simulation results

- F stands for 'Flooding + RPL and T is 'Trickle + RPL', the numbers after each letter denote R.
- SCoRe reduces latency by scheduling while keeping 99%, and adapts to network without any parameter modification.





Summary and Future work

Summary

- SCoRe schedules command and response packets 'jointly'.
- SCoRe has less overhead because desired slots is updated in routing-process.
- Preliminary results show that SCoRe improves performance (reliability & latency) while adapting to the network topology.

Future work

- Adaptive retransmission and recurrent slot assignment packet for reliability.
- Parallel transmission scheduling and tight synchronization to reduce latency.
- Real-world experiments on embedded devices.