Leveraging IP for Sensor Network Deployment

Simon Duquennoy, Niklas Wirstrom,
Nicolas Tsiftes, Adam Dunkels
Swedish Institute of Computer Science

Presenter - Bob Kinicki

Outline

- Introduction
- Addressing the Deployment Problem with IP
- Experimental Setup
- Incremental Network Deployment with RPL
- Software Installation over Low-Power IP
- In-Network Caching
- Conclusions

Introduction

- Authors are interested in the network deployment problem that includes node configuration and software updates.
- The argument is that currently at the network layer you have a practical need for multiple protocols:
  - A collection protocol (e.g. CTP (Collection Tree Protocol) in TinyOS and Contiki collection protocol) runs from sensors towards the sink (Base Station).

Introduction

- A configuration protocol that runs from the sink enabling the sink to individually configure sensor nodes.
- A software update protocol that enables multicasting from the sink to sensor nodes.

- Emerging sensor applications include heterogeneous sensors and applications. This implies the ability to dynamically change sensor software at deployment.

Introduction

- Three research contributions of this paper:
  - Measure RPL performance.
  - Show that HTTP/TCP and CoAP/UDP performance can be improved by adding a low-power streaming mechanism at the radio duty cycling layer.
  - Introduce an in-networking caching scheme.

Addressing the Deployment Problem with IP

- Authors argue against using dedicated protocols for software updates.
  - Likely, not to be adequately tested.
- IP provides a generic network layer on which applications can be built to provide low-level details (e.g., routing).
- CoAP is a new protocol developded to provide light weight RESTful interactions in a constrained environment.

Addressing the Deployment Problem with IP

- CoAP provides a bulk data transfer mechanism over UDP.
- CoAP performs its own loss detection and retransmission to avoid the problems TCP has in wireless networks.

Experimental Setup

- Authors study performance of deployment scenarios over low-power IP by using the Contiki simulation environment which simulates the Contiki OS (which provides an IPv6 implementation).
- Contiki simulation environment consist of the Cooja network simulator and MSPsim node-level emulator.

Experimental Setup

- Mote software in the simulator is msp430 binary file that includes Contiki, the uIPv6 stack and ContikiRPL.
- RPL builds a directed acyclic graph through which packets can be efficiently routed to sink nodes.
- From the sink, RPL builds routes to nodes inside the network which can distribute software to sensor nodes.

Experimental Setup

- ContikiMAC used as radio cycling protocol.
- Energy consumption is measured using Contiki's built-in power profiler.

Incremental Network Deployment with RPL

- 10 nodes deployed in a line with sink on one end.
- Three deployment scenarios:
  - *Sink-first* :: incremental starting with sink node.
  - *Sink-last* :: incremental starting with node farthest from the sink.
  - *Random* :: random starting with the sink.
- Deployment rate – one node per 30 secs.
- Energy measured per node over 8 minutes.

RPL Routing

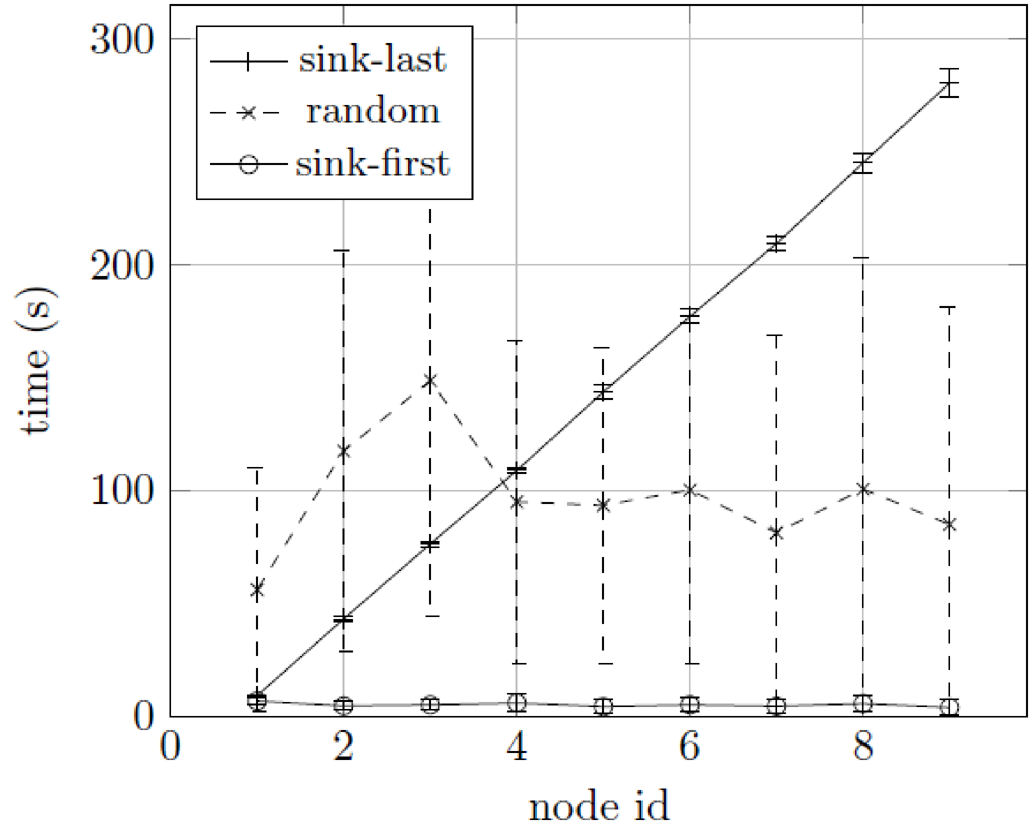


Figure 1: The time between node boots up and until it becomes part of the routing graph. The x-axis shows the number of hops from sink.

RPL Routing

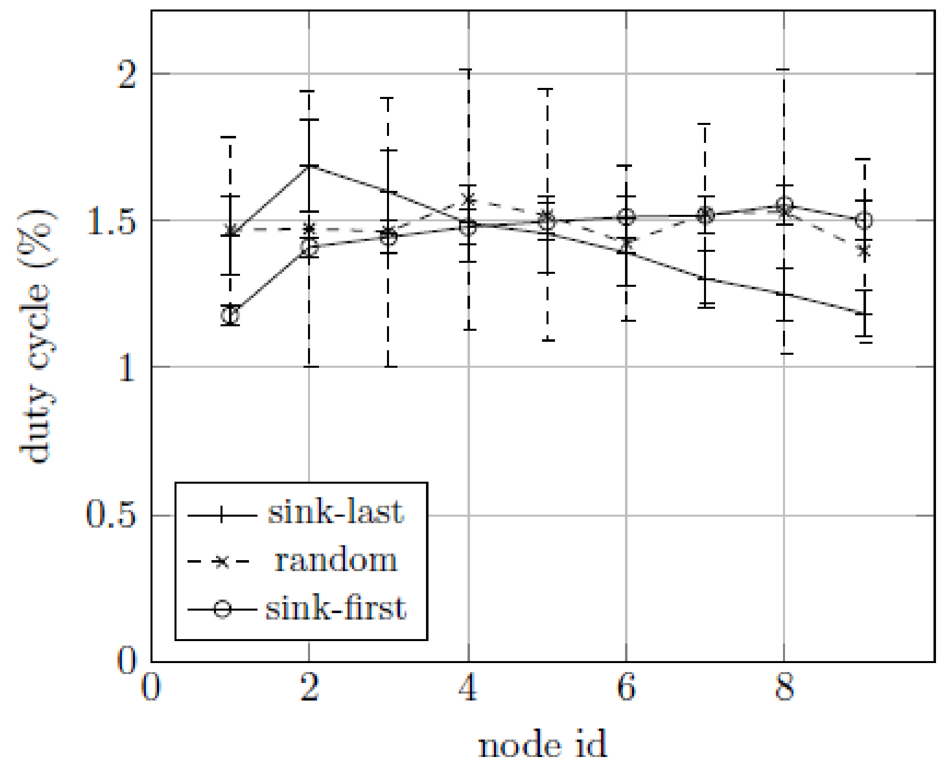


Figure 2: The radio duty cycle of the RPL nodes in Figure 1.

Software Installation over Low-Power IP

- Study performance of software updates in low-power IP networks.
- CoAP used for control commands while both TCP and CoAP used to download to node.
- CoAP sends consecutively requested single chunks of file.
- TCP sets advertised window to 1.
- ContikiMAC receivers periodically check every 125 ms.

Accelerate Multi-Hop Forwarding

- Mechanism is added to ContikiMAC such that duty cycling behaves differently during busy periods.
- Busy :: when a node has sent or transmitted at least one frame within one second.

Three Possible Behaviors

Default:: no busy period adaptation.

Streaming:: keep the radio ON during busy period.

Snooping:: increase the channel check frequency (i.e., the receivers' cyclic probe) by 8 (namely, change from a receiver cycle of 0.125 sec to 0.0156 sec.)

Synchronization on sender is disabled for streaming and snooping.

File Transfer Time and Energy

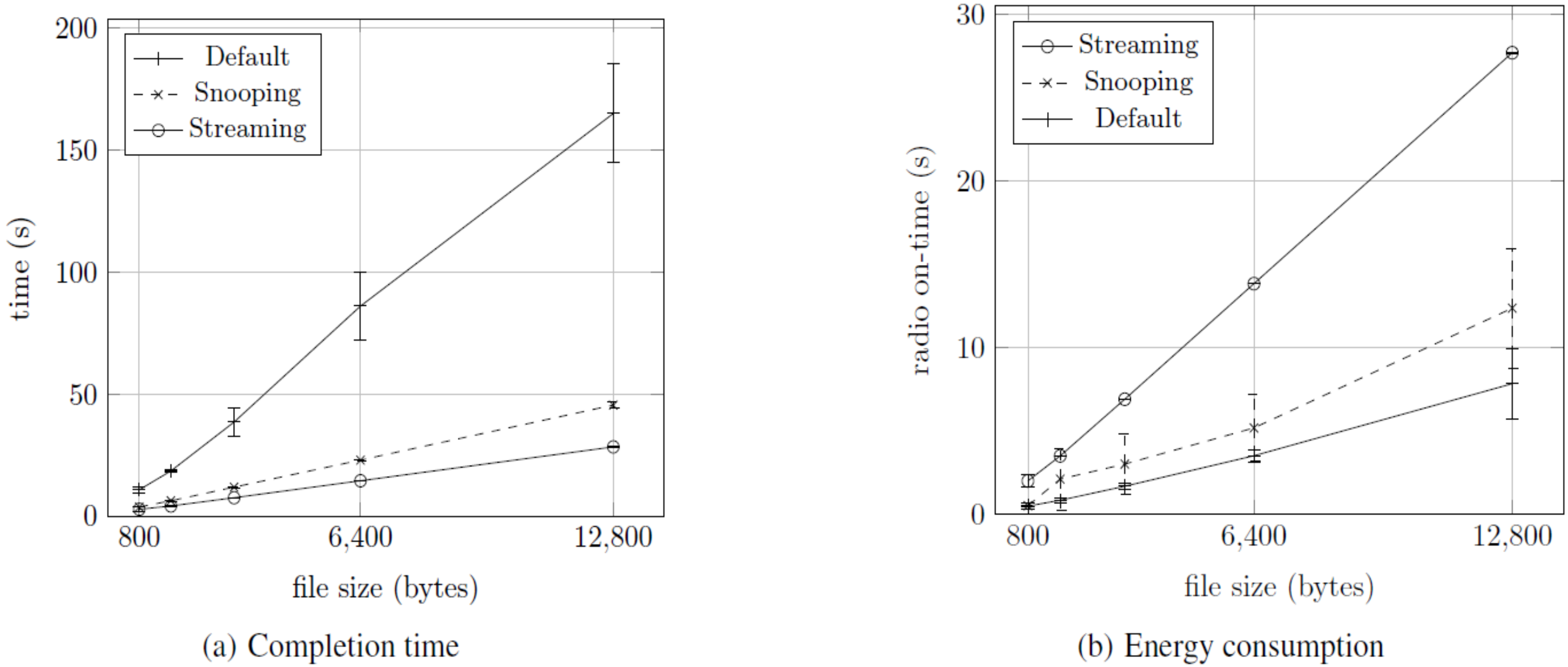
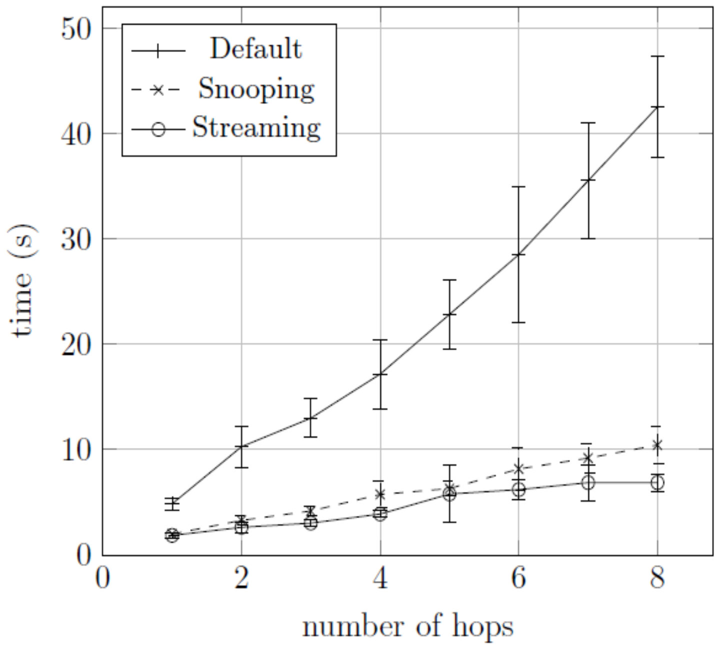


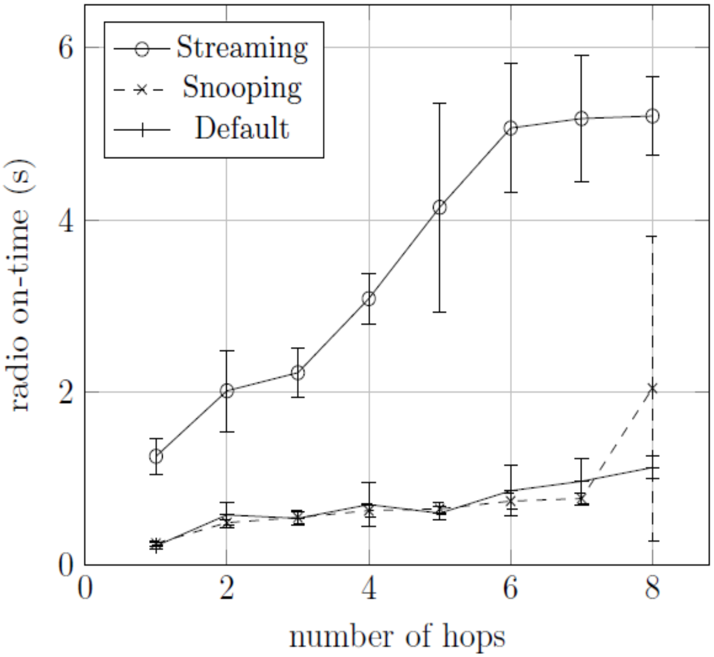
Figure 3: Time and energy as a function of file size over 4 hops. Both time and energy grow linearly with the file size.

Measurements go from request to the final notification that indicates that the downloaded application has been installed on requesting node.

Lossy Network Performance



(a) Completion time



(b) Energy consumption

Figure 4: Time and energy as a function of the number of hops with 5% per-hop loss. Both CoAP and TCP-based approaches work well in a lossy environment. The default duty cycling mechanism is the slowest while streaming provides the highest energy consumption. Snooping arguably presents the best time-energy trade-off.

TCP vs CoAP Performance

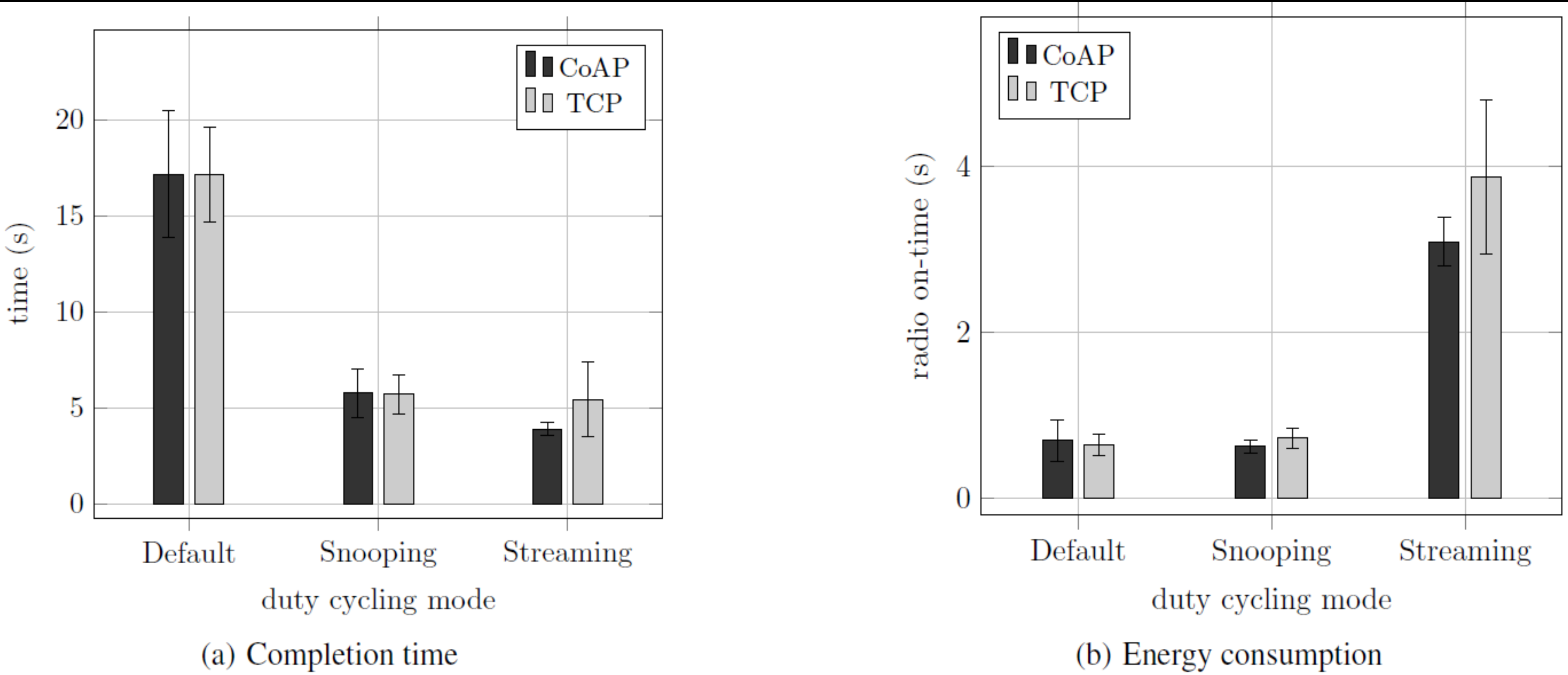


Figure 5: Comparing TCP with CoAP chunks. Both protocols provide similar results.

800-byte file,four hops and 5% packet loss rate

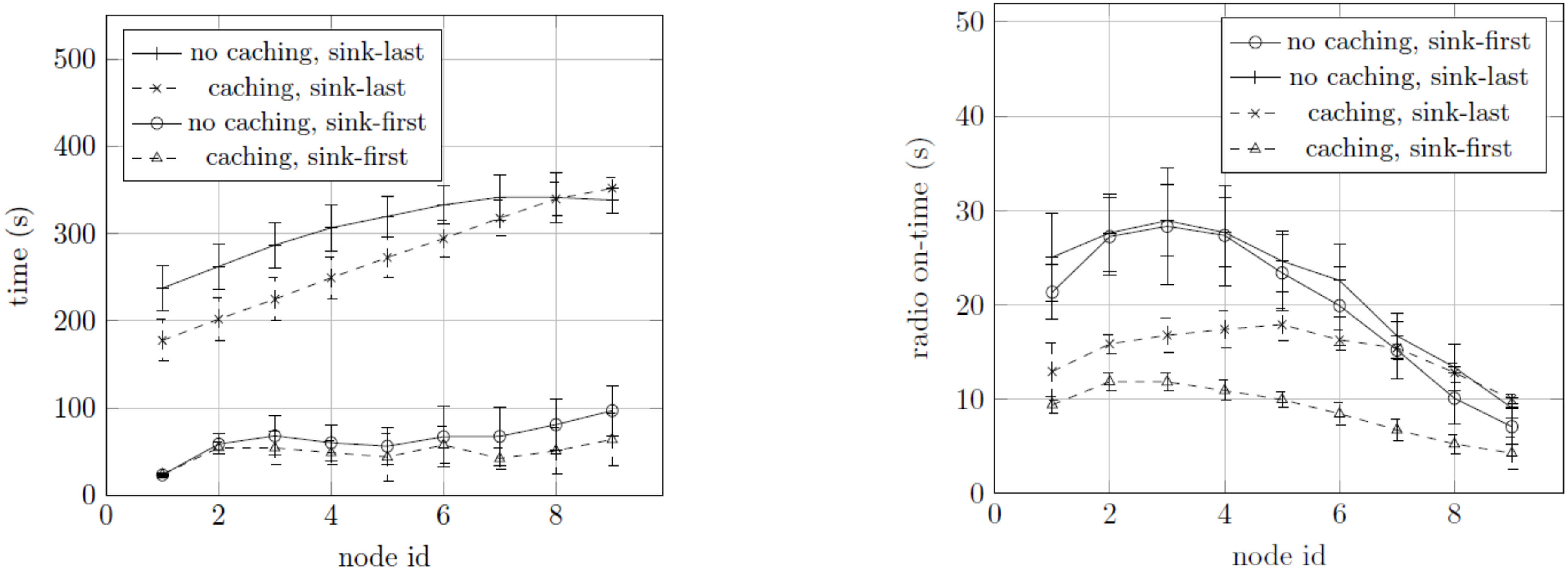
Standard solutions can transfer data over duty-cycled networks.

However, performance improves with 'adaptations'.

In-Network Caching

- Two upload strategies evaluated:
  - No caching :: all nodes download the application only from the sink.
  - Caching :: nodes store the application to secondary storage once downloaded. Then nodes set up a local CoAP server to let other nodes download from it. Sink sends a message to a newly deployed node specifying from which host the new node should download the application.
- Strategy selects physically nearest node as the host for the download.

In-Network Data Caching



(a) Completion time

(b) Energy consumption

Figure 6: In-network data caching reduces both installation time and energy consumption.

800-byte file and 15% packet loss rate

In-network caching uses only 43.5% of energy in sink-first case.

In-network caching uses only 70% of energy in sink-last case.

Conclusions and Future Work

- This paper evaluates the feasibility of an IP-based deployment solution for duty-cycled sensor networks via simulation.
- RPL can quickly find routes during deployment.
- A simple adaptation in the duty-cycle layer can improve both TCP and UDP performance.

Conclusions and Future Work

- Performance of bulk data dissemination using standard protocols can be improved using in-network caching.
- Since these were ONLY simulation experiments with an unrealistic loss model, the next step should be a testbed implementation.
- Leveraging mechanisms provided by low-power IP should simplify future sensor network deployments.