Poster Abstract: If You Have Time, Save Energy with Pull

David Hasenfratz, Andreas Meier, Matthias Woehrle, Marco Zimmerling, Lothar Thiele
Computer Engineering and Networks Lab, ETH Zurich, Switzerland
hdavid@ee.ethz.ch {a.meier,woehrle,zimmerling,thiele}@tik.ee.ethz.ch

Abstract

We analyze push and pull for data collection in wireless sensor networks. Most applications to date use the traditional push approach, where nodes transmit sensed data immediately to the sink. Using a pull approach, nodes store the data in their local flash memory, and only engage in communication during dedicated collection phases. We show how one can transform an existing push-based collection protocol into a pull-based one, and compare the power consumption of both approaches on a 35-node testbed. Our results show that substantial energy gains are possible with pull, provided that the application can tolerate a long latency.

1 Introduction

Data collection is the mainstream application scenario in wireless sensor networks (WSNs), where sensor nodes collect and forward data to a base station for further processing and analysis. Since the nodes are usually battery-powered, energy efficiency is a major requirement for long-term data collection. While latency plays only a minor role in data collection, a data yield close to 100 % is typically expected.

In this paper, we analyze two different approaches for energy-efficient data collection in WSNs: push and pull. Using the *push* approach, nodes immediately transmit the sensed data to the sink. This requires the nodes to be continuously active to maintain up-to-date routing paths. The *pull* approach instead operates in two alternating phases, as illustrated in Figure 1. During regular sleep phases the nodes do not transmit sensed data but store these in the local flash memory. At the beginning of each data collection phase, the sink wakes up the network, whereupon the nodes transmit all data backlogged in the preceding sleep phase. Afterwards, the sink puts the network back to sleep.

The major advantage of push is the lower latency. Its downside is the energy overhead to maintain up-to-date routing paths. This is not required in the pull approach, which allows the nodes to limit their activity to a minimum in the sleep phase to save energy. These savings are paid by i) control overhead incurred by the central administration, ii) energy overhead due to flash data storage and routing path initialization before each data collection phase, and iii) higher latency as determined by the pull interval t_{pull}.

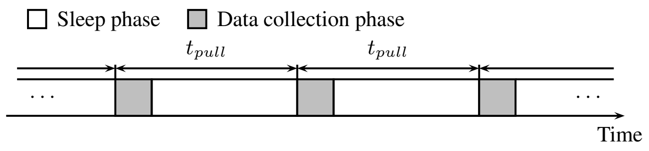

Figure 1. Basic structure of pull-based data collection.

Most data collection applications use push, even though the application requirements would allow for pull (e.g., no latency requirement). Dutta et al. [3] present first insights into a pull-based approach. We build on these promising results by comparing the energy consumptions of push and pull based on testbed experiments. We also show how one can modify an existing, push-based data collection protocol to make it work like a pull-based one. These modifications are applicable to any push-based protocol running on top of any medium access control (MAC) protocol based on low power listening. We demonstrate the feasibility of our techniques with a basic implementation of the Collection Tree Protocol (CTP) [4] running on top of the X-MAC [1] protocol.

2 Making it Work

To transform a push-based data collection protocol such as CTP into a pull-based one, we start by splitting its operation in two distinct phases: a sleep phase and a data collection phase. These phases must satisfy the following conditions:

- Sleep phase: Nodes do not communicate while waiting for the sink to initiate the data collection phase. Sensed data is stored locally in flash memory.
- Data collection phase: Nodes first initialize the routing paths as done when starting the push-based data collection. As soon as routing paths are available, nodes start to transmit backlogged data.

In a second step, we leverage the beacons periodically broadcast by the routing protocol to exchange routing information for the central administration of pull. Specifically, we extend these beacons with phase IDs to distinguish the two protocol phases. To trigger a phase change, the sink sets the appropriate phase ID in its beacons.

The energy consumption in the sleep phase is heavily dictated by the idle listening of the MAC protocol. While a long MAC wake-up interval T_w saves energy in the sleep phase, it limits the available bandwidth in the data collection phase. To overcome this limitation, we propose to use two different

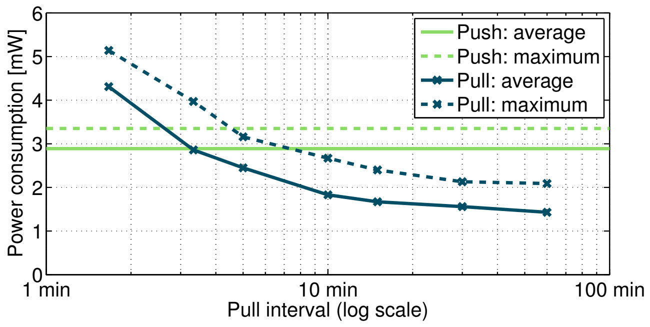

Figure 2. Power consumption comparison of the push and pull approach for varying pull intervals.

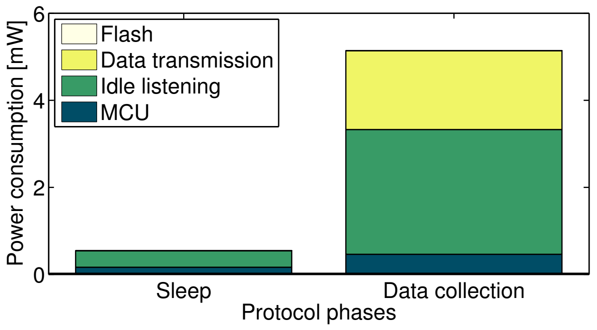

Figure 3. Average power consumption of pull in the sleep and the data collection phases.

wake-up intervals: T_w^s in the sleep phase and T_w^c in the data collection phase. The longer T_w^s saves energy by achieving a low duty cycle in the sleep phase. The shorter T_w^c provides sufficient bandwidth in the data collection phase.

For a fair comparison between push and pull with respect to energy, we perform two additional modifications to achieve close to 100 % data yield for both approaches:

- A data packet is only deleted from the sender's buffer after the reception of a data packet acknowledgment.
- If the receiver's forwarding buffer [4] is full, it replies with an explicit "not acknowledged" message. This forces the sender to back off before sending the data packet again, and hence works as a back-pressure flow-control mechanism.

3 Experimental Comparison

We compare the power consumptions (i.e., microcontroller, radio transceiver, and flash) of push and pull on a 35-node testbed. The nodes are Tmote Sky [5] devices distributed in several offices on one floor, running the Contiki operating system [2]. Nodes generate data with a sampling interval of 45 seconds. One node acts as an always-listening sink, i.e., its radio is always on. We do not consider the power consumption of the sink as it has an unlimited energy supply. We receive with both approaches in all experiments 100 % of the generated packets, due to the reliability modifications described in Section 2.

We first analyze the power consumption of pull for intervals t_{pull} ranging from 100 seconds to 60 minutes and compare these with push. This is shown in Figure 2. We look at the average and maximum power consumption. Average denotes the average power consumption of all nodes and maximum is the maximum power consumption among all nodes.

Every data collection phase incurs an energy overhead for waking up the network and putting it back to sleep. The fewer data collection phases are performed, i.e., the longer the pull interval, the lower the total energy consumption. Pull uses less energy than push with pull intervals above 5 minutes. For example, with a pull interval of 60 minutes, the average and the maximum power consumptions of pull are 50 % and 40 % below those of push.

The distribution of power consumption for the pull approach with a pull interval of 60 minutes is depicted in Figure 3. It shows the average power consumptions of the microcontroller (MCU), the radio (for idle listening and data transmission), and the flash in the two phases. Nodes consume 90 % less power in the sleep phase compared to the data collection phase, because there is no data transmission and especially due to the longer MAC wake-up interval. Notably, the power used by the flash is negligible in both phases.

4 Conclusions

Even using a simple pull approach achieves an energy gain of almost a factor of two. The energy overhead due to recurring initialization phases is amortized if the pull interval is above a couple of minutes. This is because the energy saved in the sleep phases is much larger. With respect to energy consumption it is favorable to have as few collection phases as possible. The maximum pull interval is bounded by the latency requirement and the available flash storage. The increasing amount of available flash storage relaxes the storage requirement. For example, having just 1 MB flash storage and sampling 100 Bytes of data once per minute would require to collect data only once a week. Furthermore, the pull approach provides a wide range of power optimization opportunities [3], such as efficient data stream transmissions and effective data compression due to the large amount of available data.

5 Acknowledgements

The work presented here was supported by the National Competence Center in Research on Mobile Information and Communication Systems (NCCR-MICS), a center supported by the Swiss National Science Foundation under grant number 5005-67322.

6 References

[1] M. Buettner, G. V. Yee, E. Anderson, and R. Han. X-MAC: A short preamble MAC protocol for duty-cycled wireless sensor networks. In *Proc. of SenSys '06*.

[2] A. Dunkels, B. Grönvall, and T. Voigt. Contiki – a lightweight and flexible operating system for tiny networked sensors. In *Proc. of LCN '04*.

[3] P. Dutta, D. Culler, and S. Shenker. Procrastination might lead to a longer and more useful life. In *Proc. of HotNets-VI '07*.

[4] O. Gnawali, R. Fonseca, K. Jamieson, D. Moss, and P. Levis. Collection tree protocol. In *Proc. of SenSys '09*.

[5] J. Polastre, R. Szewczyk, and D. Culler. Telos: Enabling ultra-low power wireless research. In *Proc. of IPSN '05*.