Poster Abstract: Harvester – Energy Savings Through Synchronized Low-power Listening

Roman Lim, Matthias Woehrle, Andreas Meier, Jan Beutel
Computer Engineering and Networks Lab
ETH Zurich
8092 Zurich, Switzerland
lim@tik.ee.ethz.ch

***Abstract*— Long-term data gathering with Wireless Sensor Networks (WSNs) requires drastic optimization of power consumption to provide a long system lifetime, especially for battery operated motes. In this work, we discuss Harvester, a low power data collection application that leverages recent advancements over the standard MAC and network layer components provided by the TinyOS-2.x operating system. More specifically we contribute an energy saving strategy based on a per-hop synchronization mechanism on the MAC layer. Our open-source implementation achieves over 70% energy savings over the standard LPL layer currently provided by TinyOS.**

I. Introduction

WSN systems are networks of embedded systems, where each mote has severe resource limitations e. g. memory or energy. Ample energy resources necessitate a focus on low power consumption operation. Especially for data-gathering applications requiring long-term operation, it is not feasible to periodically change the batteries. Instead, the protocols used for collecting data over multiple hops need to be optimized.

TinyOS provides MAC and network layer components to create multi-hop data gathering applications: Low-power listening (LPL) and the Collection Tree Protocol (CTP) [3] are readily available and commonly used. While the TinyOS developers are steadily improving this combination forming a low-power data gathering stack, it is still inferior over custom solutions with respect to power consumption [1], [4]. We present the protocol stack of Harvester, a low-power data gathering application developed over the last 2 years as contributed project to TinyOS[1]. Utilizing time synchronization, our radio stack saves considerable amounts of energy compared to the standard LPL+CTP combination on the order of 70%.

In the following, we present Harvester and especially the synchronization enhancements made on the LPL layer. We present a case study based on a data collection application underlining the substantial savings in energy savings.

II. Synchronized Low Power Listening

Low power MAC protocols for WSNs duty cycle the radio as it is the main power consumer of the sensor node. LPL, introduced in BMAC [6], is a random access MAC protocol, where nodes wake-up independently and have no coordination for message exchange. Using LPL, a node signals the transmissions of a packet by sending a preamble. Its neighbors periodically sample the channel, listening for preambles. WiseMAC [2] is a particular LPL scheme, which tracks wake-up phase offset between the neighbors. Hence, a sender can adapt its wake-up time to send a preamble just before the destined neighbor wakes up, considerably saving energy and bandwidth. In packet-based radios such as the CC2420 used on the Tmote Sky, long preambles can be generated by transmitting the actual packet in succession [5].

In order to achieve a lower energy footprint, we added per hop synchronization to the default TinyOS CC2420 radio stack. We used local synchronization (similar to WiseMAC) and adapted it to packet-based radios. Differing from global, local synchronization is independent of network size.

The set of modifications introduced in the LPL protocol are the following: (1) We changed of the power cycling to a *predictable wake-up schedule*, (2) added the ability to *send packets exactly at a given time* with 32 kHz clock granularity, (3) added *time information* to each packet sent, in order to share wake-up times and clock skews with neighboring nodes and (4) added a robust *wake-up prediction for neighboring nodes* utilizing drift estimation techniques.

A. Short vs. Long Preambles

Every sent packet is extended by time information about the wake-up schedule of the sender node including the offset since the last wake-up and the sleep interval. Initially, packets are sent with a "long preamble" (cf. Fig. 1), e. g. as a packet burst. Every node gradually builds up a table of time information about its neighborhood. Using the augmented neighborhood table and timed packet transmission, it is possible to exactly schedule packets without long preambles. Packets which cannot be sent due to channel occupation are sent in a later period.

Short packets can only be used for unicast traffic, broadcast messages still occupy a whole period.

B. Wake-up prediction

Consecutive time information of a node allows to calculate its drift by comparing the measured wake-up intervals against the claimed interval. In order to get accurate drift measurements by using the limited measurement resolution of

[1] http://tinyos.cvs.sourceforge.net/ in /tinyos-2.x-contrib/ethz/snpk/

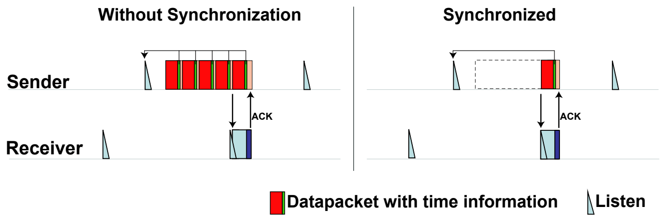

Fig. 1. All messages contain information about the senders wake-up offset. As long as there is no information about the wake-up offset of a neighbor, long preambles are used (1). Gradually, the offsets of frequent communication partners are known and a long preamble can be omitted (2).

the 32 kHz clock, a history of several measurement points is stored. From this information, we calculate the average drift. The longer the history, the more accurately the drift to every neighbor can be computed. Hence, a node can longer sustain local synchronization without message exchange.

C. Timed Packet Transmission

Switching the CC2420 hardware on and off as well as sending a packet consists of a sequence of actions, which have non-deterministic execution times. Additionally, there could also be tasks interfering. Hence we introduce an upper time bound for the whole sequence. Within this window, the radio is turned on, and the packet is loaded into the FIFO buffer. Once the FIFO is loaded, a very short command sequence, triggered by a timer, starts the actual transmission.

D. Tuning for Collection Protocols

We add two features that provide additional performance when used with a collection protocol. Collection protocols in wireless sensor networks generally only generate transmissions towards the sink. This leads to parent nodes being well informed about wake-up times and drift of their children, but not vice-versa. We add a resynchronization flag to the packet header to cope with this asymmetry. If time information becomes inaccurate, a node can request a synchronization update. A node replies with a packet containing time information.

Secondly, in very low duty cycle networks, throughput can be reasonably increased by transmitting several data packets in one wake-up interval. In order to make this possible, the receiver node has to stay awake after packet reception. We introduced another header flag called "More-Flag" to support this feature. In conjunction with CTP, this requires an enhancement to the forwarding engine.

III. Comparison

As Harvester uses CTP for data gathering, we used this protocol to compare our synchronized LPL with the existing MAC. CTP creates a routing tree based on estimated numbers of transmissions (ETX) to the sink. Link quality estimation based on sequence numbers of control packets and data acknowledgement render CTP platform independent. The Trickle algorithm allows for adapting the amount of control traffic to the fluctuations in network connectivity. CTP achieves more than 97% reliability[3] with a non-duty cycled MAC.

The following test runs were all made on a 25 node Tmote office building testbed. We introduced a patch to the original CTP to allow low power listening. We compared the TinyOS CTP/LPL combination against CTP with our synchronized MAC protocol. In both cases, the sink node does not duty-cycle the radio. The nodes run for ten minutes without data collection for initialization. Subsequently, nodes generate and send data packets with a rate of 1 packet per 20 seconds for a period of 30 minutes. We measured the effective radio duty cycle on every node. Fig. 2 displays boxplots for the individual measured duty-cycles. Our tests show that both protocols achieve comparable data yield (97.20% - 99.78%) but the synchronized LPL is superior as it achieves energy savings of 71.6% to 75.6%.

The essence of this work is very similar to the local synchronization mechanisms proposed in recent related work [1], [4]. However the solutions provided here are (i) more general in their applicability to a wide family of TinyOS-2.x based applications than custom solutions [1] and (ii) freely available as open-source. Further information and source code is provided in the contributed TinyOS project.

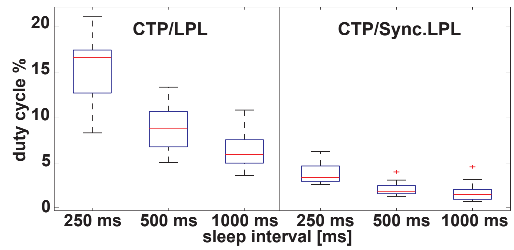

Fig. 2. Boxplot of effectively measured duty cycle. The values are measured over all 24 non-sink nodes with different sleep periods.

IV. Acknowledgments

The work 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.

References

[1] N. Burri, P. von Rickenbach, and R. Wattenhofer. Dozer: ultra-low power data gathering in sensor networks. In *Proc. 6th Int'l Conf. Information Processing Sensor Networks (IPSN '07)*, pages 450–459. ACM Press, New York, Apr. 2007.

[2] A. El-Hoiydi and J.-D. Decotignie. Wisemac: An ultra low power mac protocol for multi-hop wireless sensor networks. In *ALGOSENSORS*, pages 18–31, 2004.

[3] O. Gnawali et al. Ctp: Robust and efficient collection through control and data plane integration. Technical report, Univ. of Southern California, UC Berkeley, MIT CSAIL, Stanford Univ., 2008.

[4] J. Hui and D. Culler. Ip is dead, long live ip for wireless sensor networks. In *Proc. 6th ACM Conf. Embedded Networked Sensor Systems (SenSys 2008)*, pages 15–28, New York, NY, USA, Nov. 2008. ACM Press, New York.

[5] D. Moss et al. *TinyOS 2.x, TEP 126: CC2420 Radio Stack.*

[6] J. Polastre, J. Hill, and D. Culler. Versatile low power media access for wireless sensor networks. In *Proc. 2nd ACM Conf. Embedded Networked Sensor Systems (SenSys 2004)*, pages 95–107. ACM Press, New York, 2004.