Adaptive Synchronization in IEEE802.15.4e Networks

David Stanislowski, Xavier Vilajosana, *Member, IEEE*, Qin Wang, Thomas Watteyne, *Member, IEEE*, and Kristofer S. J. Pister

***Abstract*—Industrial low-power wireless mesh networks are shifting towards time-synchronized medium access control (MAC) protocols which are able to yield over 99.9% end-to-end reliability and radio duty cycles well below 1%. In these networks, motes use time slots to communicate, and neighbor motes maintain their clocks' alignment, typically within 1 ms. Temperature, supply voltage, and fabrication differences cause the motes' clocks to drift with respect to one another. Neighbor motes need to re-synchronize periodically through pairwise communication. This period is typically determined *a priori*, based on the worst case drift. In this paper, we propose a novel technique which measures and models the relative clock drift between neighbor motes, thereby reducing the effective drift rate. Instead of resynchronizing at a preset rate, neighbor motes resynchronize only when needed. This reduces the minimum achievable duty cycle of an idle network by a factor of 10, which in turn lowers the mote power consumption and extends the network lifetime. This Adaptive Synchronization is implemented as part of IEEE802.15.4e in the OpenWSN protocol stack and is validated through extensive experimentation.**

***Index Terms*—Duty cycle, energy consumption, IEEE802.15.4e, synchronization, time-synchronized channel hopping (TSCH), wireless sensor networks (WSNs).**

I. Introduction

IN the last decade, contention-based wireless medium access control (MAC) layers have been used in many low-power wireless mesh protocols. ZigBee[1] has been the *de-facto* standard for these types of networks, but has failed to fulfill the industrial reliability requirements.

Manuscript received December 17, 2012; revised February 06, 2013; accepted March 15, 2013. Date of publication March 27, 2013; date of current version December 12, 2013. This work was supported by Projects CALIPSO-288879, OUTSMART-285038, RELYONIT-317826 and SWAP-251557, supported in part by the European Community and the Spanish Ministry of Education under Fullbright-ME Grant INF-2010-0319. Paper no. TII-12-0855.

D. Stanislowski and K. S. J. Pister are with the Berkeley Sensor and Actuator Center, University of California, Berkeley, CA 94720 USA.

X. Vilajosana is with the Berkeley Sensor and Actuator Center, University of California, Berkeley, CA 94720 USA, and also with the Universitat Oberta de Catalunya, Barcelona 08018, Spain, and Worldsensing, Barcelona 08013, Spain.

Q. Wang is with the Berkeley Sensor and Actuator Center, University of California, Berkeley, CA 94720 USA, and also with the and University of Science and Technology, Beijing 100083, China.

T. Watteyne is with the Berkeley Sensor and Actuator Center, University of California, Berkeley, CA 94720 USA, and also with the Dust Networks/Linear Technology, Hayward, CA 94544 USA.

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TII.2013.2255062

[1][Online]. Available: http://www.zigbee.org/.

Things started to change with the development of the time-synchronized channel hopping (TSCH) technique that was adopted by major industrial low-power wireless standards such as WirelessHART [1] and recently as a part of the published IEEE802.15.4e standard [2]. As of today, several commercial networking providers are offering 99.9% reliable MAC layers that provide radio duty cycles well below 1%, thereby reducing the mote power consumption and increasing the network lifetime.[2]

The IEEE802.15.4e standard [2] is an amendment to the MAC protocol of IEEE802.15.4-2006 [3]. It achieves better reliability and lower power consumption through time-synchronized channel hopping (TSCH). All motes in an IEEE802.15.4e network are synchronized and time is split into time slots, each typically 10 ms long. Time slots are grouped into a super-frame which continuously repeats over time.

A schedule instructs each mote what to do in each time slot: send to a particular neighbor, receive from a particular neighbor, or sleep [4]. Channel diversity is obtained by specifying, for each send and receive slot, a channel offset. The same channel offset translates into a different frequency on which to communicate at each iteration of the super-frame. The resulting channel-hopping communication reduces the impact of external interference and multipath fading, thereby increasing the reliability of the network [5].

Crystal oscillators are commonly used for timing since they offer a good tradeoff between power consumption, frequency stability, and cost. The frequency of a crystal is affected by manufacturing differences, temperature, and supply voltage. A crystal oscillator is rated by its frequency stability: a 32-kHz crystal rated 30 ppm will pulse somewhere between 32 768.99 Hz and 32 767.01 Hz. Two motes equipped with these crystal can *drift* by 60 ppm to one another (one going fast, one going slow), i.e., they will desynchronize by 60 μs each second. Therefore, motes need to resynchronize periodically.

In a TSCH network, all transmitting motes are scheduled to begin transmission at the same time in a slot (typically about 2 ms into the slot, called the TsTxOffset). To allow for some desynchronization, receivers start listening some time before this instant (see Fig. 1) and keep listening some time after. This duration is called the "guard time"

Assuming a guard time of 1 ms, and if two motes are equipped with 30 ppm crystals, it takes 16 s for them to desynchronize by more than 1 ms. Since they cannot communicate if they get desynchronized past the guard time, they periodically resynchronize. The following equation can be used to determine the

[2][Online]. Available: http://www.linear.com/products/smartmesh_ip.

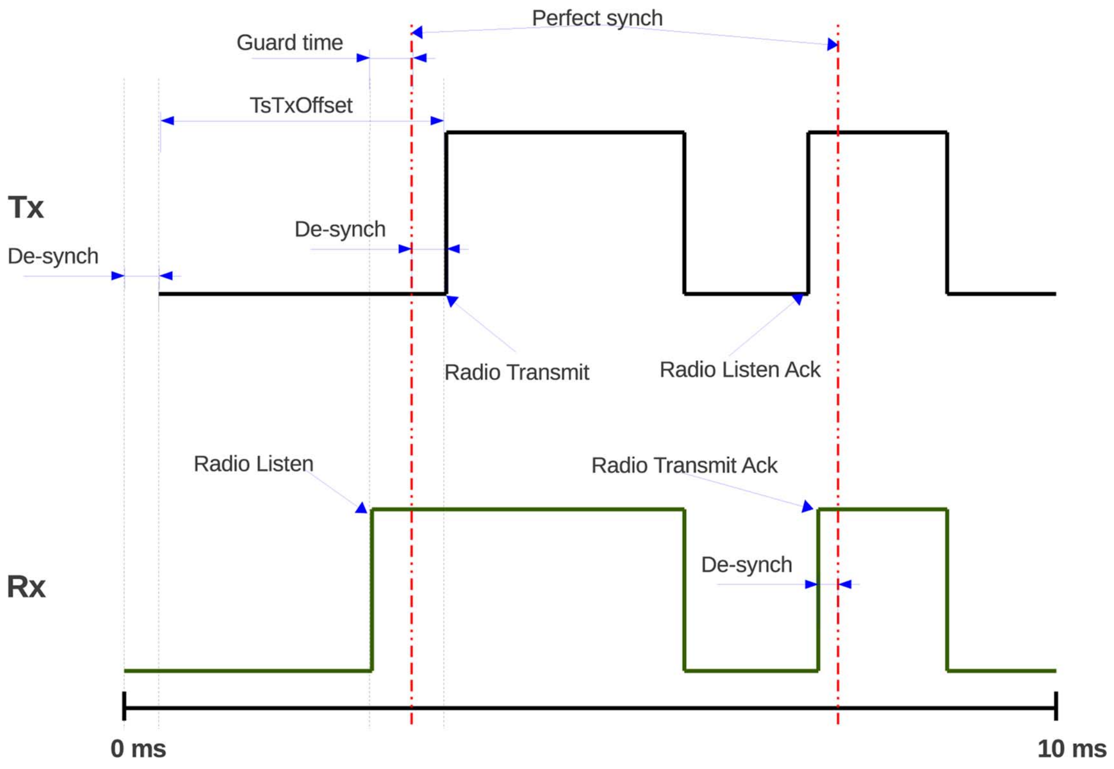

Fig. 1. Timeline of an IEEE802.15.4e slot showing how two motes synchronize by exchanging a data packet.

maximum allowable synchronization period τ_{ka} as a function of the guard time T_g and the drift rate $\Delta\nu$:

$$\tau_{ka} = \frac{T_g}{\Delta\nu} \tag{1}$$

Resynchronizing more frequently than the limit obtained with (1) guarantees that the motes stay synchronized within their guard time. The larger the guard time T_g or the smaller the drift $\Delta\nu$, the less frequent motes need to resynchronize.

As detailed in Section III, resynchronizing in a TSCH network involves exchanging a data packet and an acknowledgment. Since the radio consumes power, it is best not to have to resynchronize too often. We call the idle duty cycle the portion of time the radio of a mote needs to be on just to keep synchronized to its neighbors. The smaller this value, the longer the lifetime of the mote.

Typically, the worst case drift rate is used to hard-code the resynchronization period in the motes. This often results in "over-synchronization" and hence a waste of energy. We propose in this paper a technique which allows a mote to measure its clock drift rate with a neighbor. Using this information, it tracks its neighbor's drift by periodically applying internal correction to its clock, which allows it to resynchronizes less often. This estimated correction helps to ensure that, even in very large networks, the total desynchronization between two arbitrary nodes is reduced as individual parent–child drifts are reduced.

This technique, called *Adaptive Synchronization*, is added to the TSCH mode of the IEEE802.15.4e implementation of the OpenWSN protocol stack [6] and used on real hardware. Experimental results presented in Section V show a reduction of the minimum achievable duty cycle of an idle network by a factor of 10.

II. Related Work

Network synchronization is a well-studied topic. Lindsey *et al.* published early theoretical work on synchronization, covering independent clocks settings, master–slave hierarchical settings (centralized), time reference distribution, and mutual synchronization (decentralized) [7], [8]. The emergence of cellular and wireless networks came with new requirements on network synchronization, such as multiframe synchronization [9] or network-wide synchronization in *ad-hoc* [10] networks. Recently, synchronization has also been studied in the context of industrial real-time systems, which introduces strict accuracy constraints [11]. The time-synchronized mesh protocol (TSMP) [12] and the timing-sync protocol for sensor networks (TPSN) [13] were the first to apply synchronization techniques to low-power mesh networks. By synchronizing the transmitter to the receiver, both motes spend most of their time with their radio OFF, only turning it ON when a communication is about to take place. This yields very low radio duty cycles and long network lifetimes. The ideas developed from these protocols are the foundation for standards such as IEEE802.15.4e [2]. The technique presented in this paper builds on top of TSMP, TPSN, and IEEE802.15.4e. Instead of always resynchronizing at a "worst case" rate, adaptive synchronization lowers the radio's duty cycle even more by modeling the clock drift between neighbor motes and reducing the effective drift rate by making small corrections that do not require the radio to be powered. This results in a lower idle duty cycle and thus a longer network lifetime.

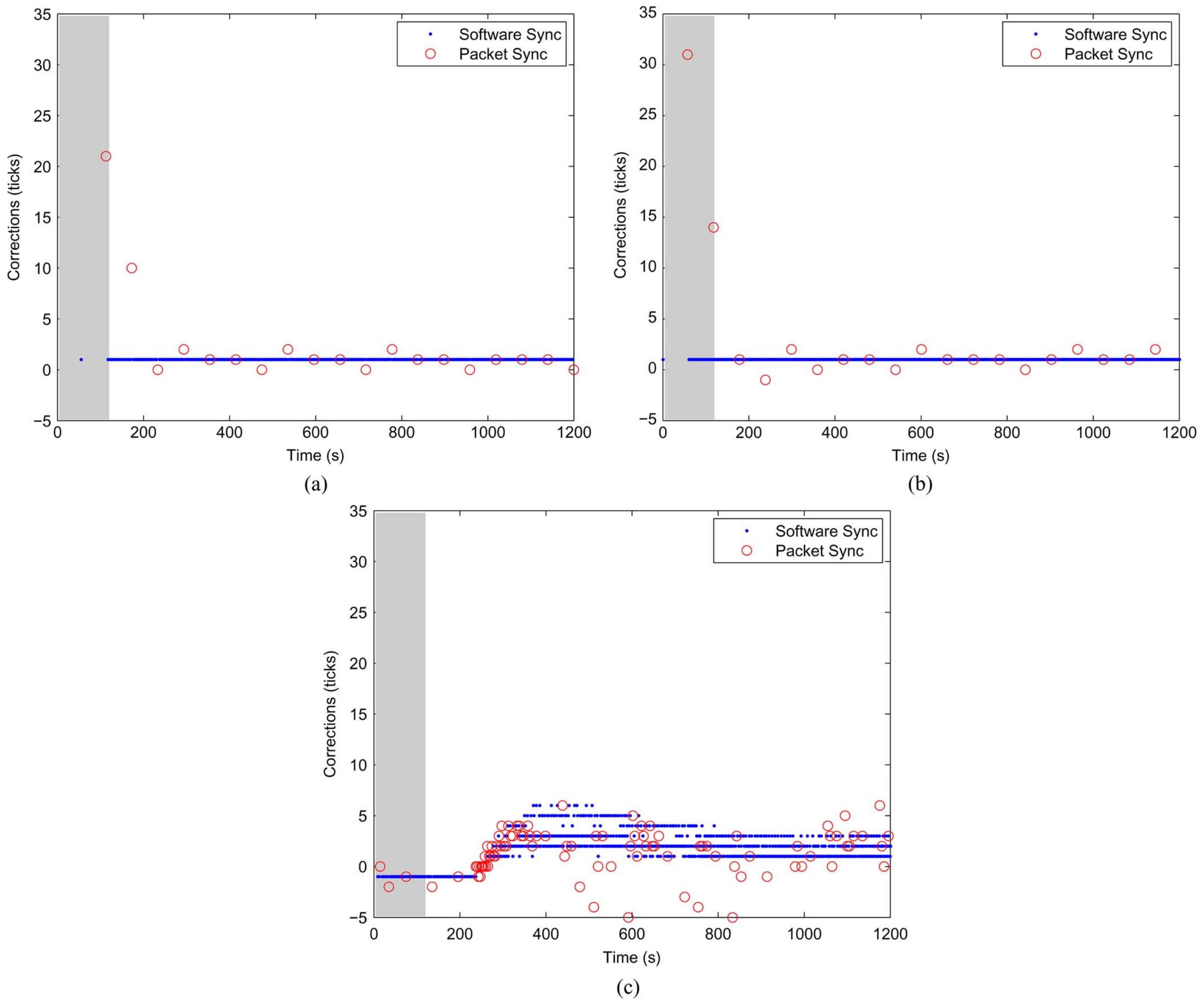

Fig. 2. Corrections are applied to the clock of a mote which uses adaptive synchronization to track the drift with a neighbor mote. Packet-based synchronization, which happens every 60 s or when mote temperature has changed 2 °C since last packet sync, whichever comes first, causes the offset (red circles) to be applied; software corrections are used to further track the calculated drift in between packet-based synchronizations. The same experiment is conducted in three environments with different temperature variations. Corrections are expressed in 32-kHz clock ticks. (a) Indoor setup: the temperature is constant. (b) Outdoor setup: the temperature varies by 15 °C over the course of the experiment. (c) Oven setup: the temperature varies by 55 °C over the course of the experiment.

Recently, Kerkez [14] investigated how to achieve tight clock synchronization without a precise clock source. This allows for a board manufacturer to swap an (expensive) crystal for a (cheap) on-chip oscillator. Similar to our approach, Kerkez proposes to account clock drift to be able to correct clock misalignment by modifying slot phase and duration, but requires motes to communicate at every time frame. Very interesting approaches have been published recently, Medina *et al.* [15] present a synchronization algorithm based on an estimation of the relative clock drift of each node, their scheme uses periodic beacons to determine clock drift with respect to a global clock shared by all the motes in the network. In addition, in [16], the same authors present a theoretical error characterization of clock drift in similar scenarios. Adaptive synchronization presented in this paper takes a similar approach and complements estimated clock drift with periodic temperature monitoring in order to compensate temperature effects on the drift rate of crystals. Liu *et al.* [17] uses a Kalman filter to compensate clock drift, their scheme named AdaSynch shows how clock drift can be modeled and fit to a certain value using different Kalman filters. The important aspect is the deep analysis of clock drift that authors carry on a set of 100 telosb motes which can be further used to improve the Adaptive synchronization estimation scheme. Temperature compensation schemes have been studied recently by Brunelli *et al.* [18]. Authors propose to divide operation in two phases, a one-time calibration where nodes learn the drifting pattern and a operation phases where nodes apply the learnt pattern according to the temperature reading. Adaptive synchronization composes a temperature synchronization scheme with an adaptive synchronization which benefits from the fact that no calibration is needed and that temperature readings can be spaced in time so energy consumption is not compromised.

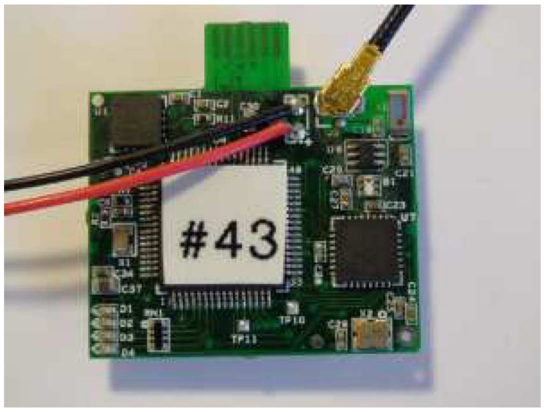

Fig. 3. GINA mote.

III. Synchronizing in IEEE802.15.4e

In an IEEE802.15.4e TSCH network, communication happens in time slots. A network-wide schedule instructs each mote what to do in each slot. The schedule is built to satisfy the throughput and delay requirements of the different traffic flows. A time slot is sufficiently long for the transmitter to transmit the longest possible packet and for the receiver to send back an acknowledgment indicating reception.

All motes in an IEEE802.15.4e network are synchronized, i.e., neighbor motes are desynchronized by at most a *guard time*, which is typically 1 ms. The network schedule indicates the "time parents" of each mote, the neighbor motes to which this mote needs to keep synchronized. This results in a synchronization-directed acyclic graph. Once a mote has been assigned its time parents, it needs to ensure its clock never drifts by more than a guard time with respect to its time parents.

IEEE802.15.4e is designed to cope with clock drift. It includes timing information in *all* packets, to allow for two neighbor motes to resynchronize to one another whenever they communicate. When a transmitter transmits a packet, the receiver timestamps the instant at which it receives the packet and indicates the offset between that measured time and the theoretical reception time `TsTxOffset` in its acknowledgment. As a result, upon each communication, both the transmitter and the receiver know how desynchronized they are. Depending on which mote is the time parent, either the transmitter or the receiver aligns its clock to the other. After resynchronizing, the clocks of both motes are perfectly aligned.

When data packets flow through the network, keeping synchronized comes at no extra cost.[3] When the network sits idle (no data packets are flowing), motes send empty data packets periodically to keep synchronized. The period at which these "keep-alive" packets are exchanged is the object of this paper. We call "idle duty cycle" the portion of the time a mote's radio is on just to keep synchronized, when the network sits idle.

Let us take a typical case where a mote has two time parents and two children. We assume the mote runs IEEE802.15.4e with a 1-ms guard time and is equipped with a 30-ppm crystal. As per the calculation above, it can expect, each 16 s, to participate in four resynchronizations: to both parents and both children. Each resynchronization involves exchanging the keep-alive packet (around 15 bytes) and the acknowledgment (around 15 bytes). At 250 kpbs and taking into account radio startup and turn-around times, this translates to a radio on time of around 2 ms per synchronization. To participate in all synchronizations, it will have its radio on for around 8 ms each 16 s, or an idle radio duty cycle of 0.050%.

[3]Strictly speaking, the time offset between transmitter and receiver is encoded as a 2-byte signed value, which is added to the acknowledgment. Transmitting this extra field costs some energy.

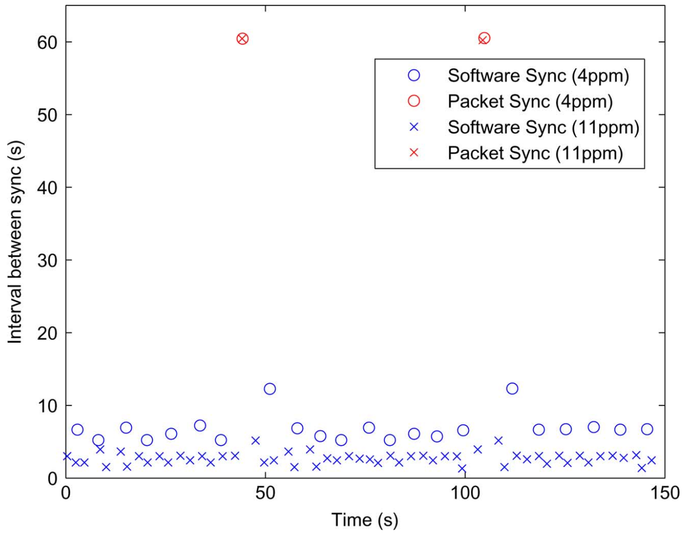

Fig. 4. Correction interval is the interval between each type of sync. For Software Syncs, which in this case has a consistent correction value of a single clock cycle, the interval is closely related to the clock drift rate. This figure shows an overlay of correction intervals for two mote-pairs of different drift rates.

The goal of the adaptive synchronization technique presented in Section IV is to bring this idle duty cycle down by extending the period between two resynchronization between neighbor motes.

IV. Adaptive Synchronization

Adaptive synchronization allows a mote to track the drift rate to its neighbor, which it then uses to increase the time between keep-alive messages.

This is done by measuring the *effective* clock drift between two consecutive keep-alive messages, and then using that rate to make periodic "software" adjustments (which do not require communication) in-between "packet-based" resynchronizations.

At each packet synchronization, the timestamping of the packets indicates to a mote how offset its clock is with respect to its neighbor's. It can calculate the experimental clock drift rate $r_{\exp}$ by dividing this offset ε by the duration since the previous packet-based synchronization ΔT, as shown in

$$r_{\exp} = \frac{\varepsilon}{\Delta T}. \quad (2)$$

If a mote uses (2) and determines that it is fast with respect to a neighbor, it will periodically adjust its own clock in order to "slow down." It can do so by periodically adding a clock tick. This allows it to track its neighbor's clock, thereby reducing the drift. Since the correction is not perfect and the drift rate changes (e.g., with temperature), packet-based re-synchronizations is still needed, but can be less frequent.

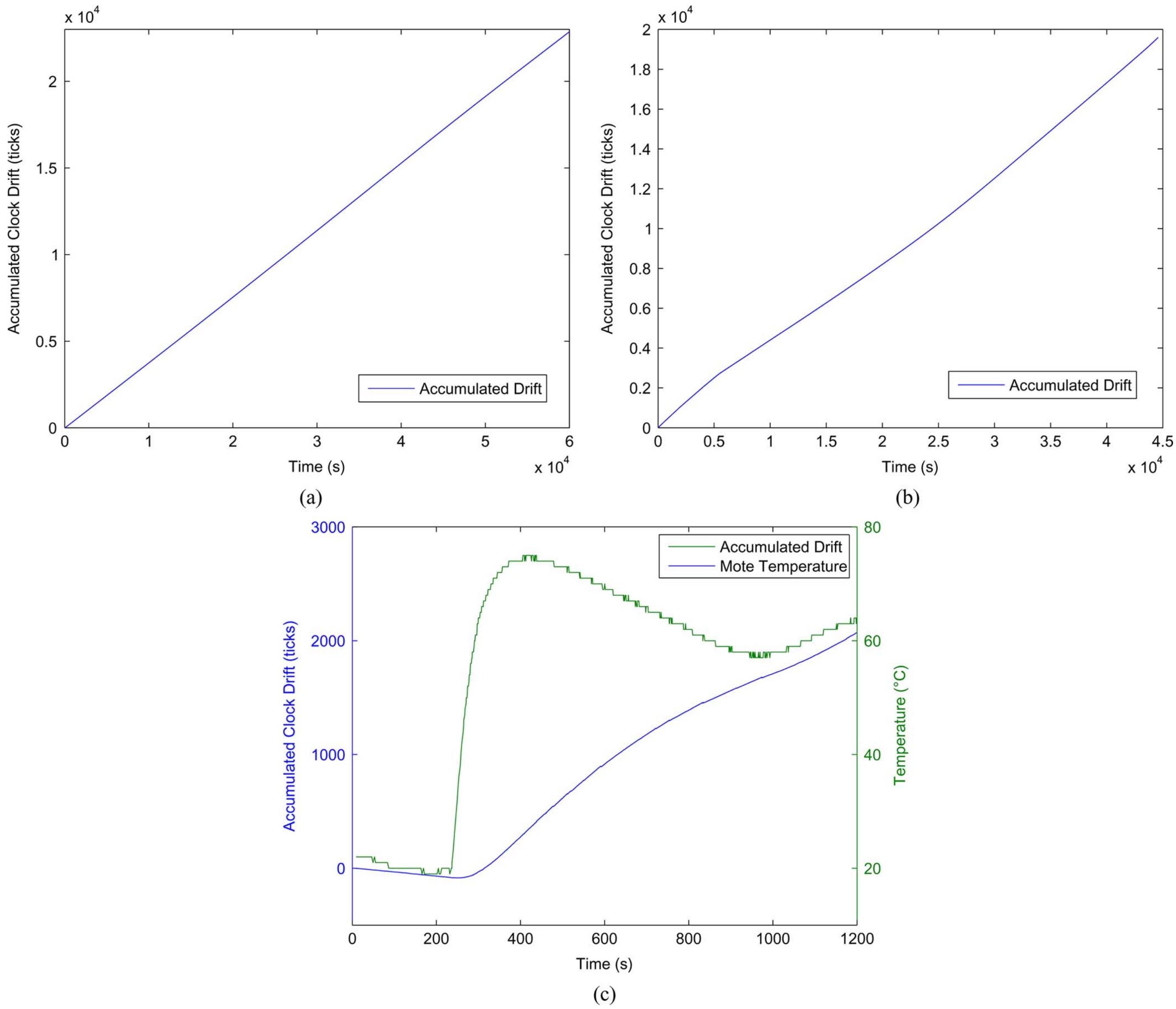

Fig. 5. Accumulated clock drift for a mote pair. This is the accumulated value of corrections made by both software adjustments and packet-based resynchronizations which shows how two motes would drift when corrections are not applied. The figures show how the drift rate, i.e., the slope, changes with temperature. (a) Indoor setup. (b) Outdoor setup. (c) Oven setup.

Fig. 2(a) shows experimental results gathered on a pair of GINA motes [19] running in a temperature-controlled environment. The blue dots show the periodic software corrections applied to the mote's clock; the red circles represent the offset indicated by each packet-based synchronization (which is pre-set to happen every 60 s). After a learning period—during which the mote measures the drift rate—the software corrections track the drift, and the offset measured by the packet-based synchronization drops from 22 ticks (671 μs) to three ticks (91 μs). This corresponds to an effective drift dropping from 11 to 1.5 ppm.

Temperature causes the clock drift rate to change. To account for temperature changes, Adaptive synchronization records the temperature during the most recent packet-sync event and then measures the temperature periodically. If the difference between the recorded temperature and any measurement thereafter is larger than φ, a keep-alive message is triggered to re-synchronize and determine the new effect drift rate. The threshold φ—set to 2 °C in our experiments—can be tuned to match the crystal's temperature sensitivity and the application requirements.

The other sources of clock drift (supply voltage and crystal aging) change much more slowly and are unlikely to be noticed even with extended keep-alive packet intervals. For this reason, we did not make any attempt to measure or correct for these sources.

There is an error associated with the experimental clock drift rate that increases as the packet interval decreases: short intervals make for inaccurate predictions. This could be an issue if a short message interval were immediately followed by a long one. To prevent such a situation, the interval between keep-alive packets is controlled so that a short interval is followed by one twice as long, then four times as long, etcetera, until the target interval length is reached. In our experimental setting we started at a period of 5 s, increasing it up to 60 s. This is similar to TCP's "slow start" [20].

V. Evaluation

Here, we experimentally explore the tradeoff between synchronization accuracy and energy consumption, for different crystal oscillator configurations and temperature conditions.

All experiments are conducted with GINA motes [19] (shown in Fig. 3), which feature a Texas Instruments MSP430 microcontroller and an Atmel AT86RF231 IEEE802.15.4 radio chip.

Data were collected at each synchronization event, whether packet or software based. Clock correction amount (in units of ticks), time of correction, a unique sync event ID, temperature, and whether the correction was packet or software based were all recorded.

A. Impact of Manufacturing Differences

To isolate crystal manufacturing, all experiments in this section are conducted with motes operating in the same temperature-controlled environment, corresponding to the indoors case shown in Fig. 2(a).

By letting the motes synchronize without adaptive synchronization, we can measure the drift rate. We cherry-pick two pairs of motes: motes H_1 and H_2 exhibit a *high* clock drift of 11 ppm; motes L_1 and L_2 have a *low* effective clock drift of 4 ppm. The motes in both pairs run the same firmware, and are configured to exchange a keep-alive every 60 s.

Fig. 4 depicts the interval between two synchronizations. Because Adaptive Synchronization speeds up synchronization when the drift is large, the interval between software synchronization events is smaller for the (H_1, H_2) 11 ppm pair than for the (L_1, L_2) 4 ppm pair. The frequency of synchronization is proportional to the clock drift, so in this case the 11 ppm pair makes corrections about 2.75 time as frequently as the 4 ppm pair.

B. Impact of Temperature

A pair of motes is set up to send keep-alive messages to keep synchronized using adaptive synchronization; no application data are exchanged. To measure the impact of temperature variation, the same experiment is repeated in the following cases.

- In the **indoor** case, the pair of motes is placed on a desktop in an office environment. The experiment is conducted over the course of 20 h, during which the temperature is constant.
- In the **outdoor** case, one mote is placed indoors, the other one outdoors. The experiment is conducted over the course of 12 h, during which the temperature drops by 15 °C.
- In the **oven** case, one mote is placed at a constant 20 °C, the other one is placed in an oven. From 20 °C, the temperature of the oven is brought to 75 °C over the course of 2 min and then down to 55 °C over 10 min.

In each case, adaptive synchronization is running with a preset keep-alive interval of 60 s. As detailed in Section IV,

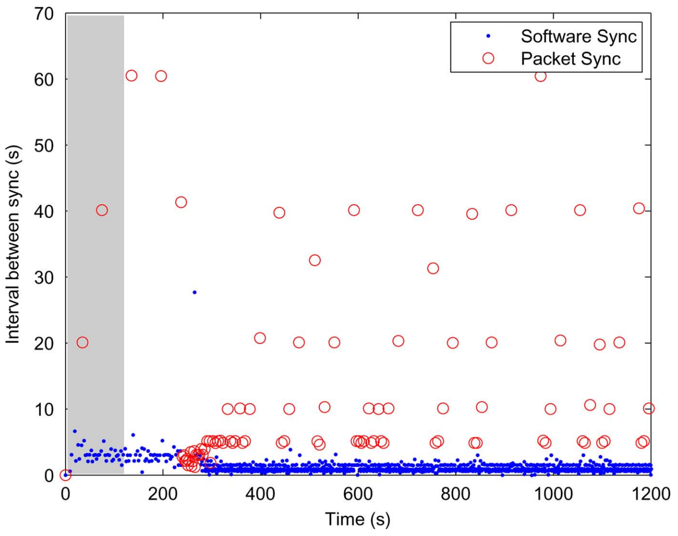

Fig. 6. Interval between time correction events in the oven setup scenario. Packet-based corrections are separated from software-based corrections. At around 250 s, the temperature is increased using the oven. The slow start process can be seen when temperature reaches a steady state at around 350 to 400 s.

sudden changes in temperature trigger extra keep-alive messages.

Fig. 5 shows the total accumulated correction conducted by adaptive synchronization in the three cases. The total amount that adaptive synchronization corrected is equivalent to how the two motes would drift when corrections are not applied, therefore, Fig. 5 also presents the accumulated drift between two motes in the presented scenarios. A change in temperature alters the accumulation rate. In the indoor case, the value of accumulated correction over time is increasing linearly due to stable temperature. There is slow variation in the accumulation rate of correction in the outdoor case, caused by the slow temperature drift during the transition between day and night. In the oven case, the green line shows the sharp temperature change when the oven is switched on.

Fig. 6 shows the interval between time correction events in the oven setup. As the temperature increases, it can be seen that the packet synchronizations happen frequently at around 250 s as the temperature rapidly increases. When temperature stabilizes, adaptive synchronization starts the slow start process to achieve the maximum keep alive interval.

C. Energy Consumption Discussion

Fig. 7 shows a direct comparison between synchronizing with and without Adaptive Synchronization. Because drift is not compensated when not using adaptive synchronization the offset is large at each packet-based synchronization. In Fig. 7, this is around 20 ticks or a drift of 11 ppm. When using adaptive synchronization, drift is compensated, and the offset triggered by each packet-synchronization drops to two ticks or 1 ppm.

This means that the guard time can be much smaller. In this case, without adaptive synchronization, the guard time must be at least 660 μs long (i.e., the clock drift in 60 s); it can be 60 μs when using adaptive synchronization, using the same 60-s

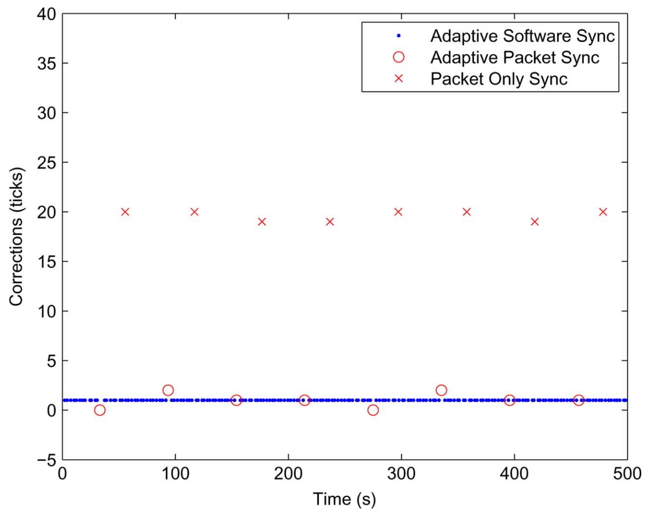

Fig. 7. Correction values for the 11-ppm mote-pair running adaptive synchronization versus packet-only correction.

keep-alive interval. This translates into a 90% reduction of the energy spent by the mote idle listening for packets.

VI. Conclusion

TSCH networks such as those defined by the IEEE802.15.4e standard require tight synchronization of their motes, which limits the lowest achievable duty cycle and hence the minimum energy consumption of the network. Clock drift is attributed to crystal manufacturing differences and temperature, mainly. TSCH uses periodic keep-alive messages to resynchronize neighbor motes in the absence of application traffic.

We presented in this paper a simple adaptive synchronization technique which can either reduce the keep-alive interval or shorten the guard time. In both cases, this translates directly in a reduction of the idle duty cycle. Experimental results show an idle duty cycle reduced by a factor of 10, with the ability to maintain synchronization even in rapidly varying temperature settings. The technique is implemented in the IEEE802.15.4e MAC layer of the OpenWSN protocol stack and evaluated on off-the-shelf motes.

Acknowledgment

The authors are especially thankful for the preliminary work carried out by T. Beyazoglu and X. Vilajosana

References

[1] *WirelessHART Specification 75: TDMA Data-Link Layer*, HART Communication Foundation Std., Rev. 1.1, hCF, 2008, _SPEC-75.
[2] *802.15.4e-2012: IEEE Standard for Local and Metropolitan Area Networks—Part 15.4: Low-Rate Wireless Personal Area Networks (LRWPANs) Amendment 1: MAC Sublayer*, IEEE Std., Apr. 16, 2012.
[3] *802.15.4-2006: IEEE Standard for Information Technology, Local and Metropolitan Area Networks, Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low Rate Wireless Personal Area Networks (WPANs)*, IEEE Std., 2006.
[4] E. Toscano and L. L. Bello, "Multichannel superframe scheduling for IEEE802.15.4 industrial wireless sensor networks," *IEEE Trans. Ind. Inf.*, vol. 8, no. 2, pp. 337–350, May 2012.
[5] T. Watteyne, A. Mehta, and K. Pister, "Reliability through frequency diversity: Why channel hopping makes sense," in *Proc. 6th ACM Symp. Performance Eval. Wireless Ad Hoc, Sensor, Ubiquitous Netw.*, New York, NY, USA, 2009, pp. 116–123.
[6] T. Watteyne, X. Vilajosana, B. Kerkez, F. Chraim, K. Weekly, Q. Wang, S. D. Glaser, and K. Pister, "OpenWSN: A standards-based low-power wireless development environment," *Trans. Emerging Telecommun. Technol.*, vol. 23, no. 5, pp. 480–493, 2012.
[7] W. C. Lindsey, F. Ghazvinian, W. Hagmann, and K. Dessouky, "Network synchronization," *Proc. IEEE*, vol. 73, no. 1, pp. 1445–1467, Oct. 1985.
[8] J.-H. Chen and W. Lindsey, "Mutual clock synchronization in global digital communication networks," *Eur. Trans. Telecommun.*, vol. 7, no. 1, pp. 25–37, Jan.–Feb. 1996.
[9] Z. Zhang, H. Kayama, and C. Tellambura, "New joint frame synchronisation and carrier frequency offset estimation method for OFDM systems," *Eur. Trans. Telecommun.*, vol. 20, no. 4, pp. 413–430, 2009.
[10] C. H. Rentel and T. Kunz, Carleton Univ., "Network synchronization in wireless ad hoc networks," Tech. Rep., Jul. 2004.
[11] M. M. H. P. van den Heuvel, R. J. Bril, and J. J. Lukkien, "Transparent synchronization protocols for compositional real-time systems," *IEEE Trans. Ind. Inf.*, vol. 8, no. 2, pp. 322–336, May 2012.
[12] K. S. J. Pister and L. Doherty, "TSMP: Time synchronized mesh protocol," in *Proc. Int. Symp. Distrib. Sensor Networks*, Orlando, FL, USA, Nov. 2008, pp. 391–398.
[13] S. Ganeriwal, R. Kumar, and M. B. Srivastava, "Timing-sync protocol for sensor networks," in *Proc. Int. Conf. Embedded Networked Sensor Syst.*, New York, NY, USA, 2003, pp. 138–149.
[14] B. Kerkez, "Adaptive time synchronization and frequency channel hopping for wireless sensor networks," Master's thesis, Electr. Eng. Comput. Sci. Dept., Univ. of California, Berkeley, CA, USA, Jun. 2012.
[15] C. Medina, J. C. Segura, and A. De La Torre, "Accurate time synchronization of ultrasonic tof measurements in ieee based wireless sensor networks," *Ad Hoc Netw.* vol. 11, no. 1, pp. 442–452, Jan. 2013.
[16] C. Medina, J. C. Segura, and A. de la Torre, "A synchronous tdma ultrasonic tof measurement system for low-power wireless sensor networks," *IEEE Trans. Instrum. Meas.*, vol. PP, no. 99, pp. 1–13, 2012.
[17] Q. Liu, X. Liu, J. L. Zhou, G. Zhou, G. Jin, Q. Sun, and M. Xi, "Adasynch: A general adaptive clock synchronization scheme based on kalman filter for wsns," *Wireless Pers. Commun.*, vol. 63, no. 1, pp. 217–239, Mar. 2012.
[18] D. Brunelli, D. Balsamo, G. Paci, and L. Benini, "Temperature compensated time synchronisation in wireless sensor networks," *Electron. Lett.*, vol. 48, no. 16, pp. 1026–1028, 2012, 2.
[19] A. Mehta and K. Pister, "WARPWING: A complete open-source control platform for miniature robots," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2010, pp. 18–22.
[20] M. Allman, V. Paxson, and E. Blanton, "TCP Congestion Control, RFC (Draft Standard)," Internet Eng. Task Force.

David Stanislowski received the B.S. degree in physics from the University of California, Santa Cruz, CA, USA, in 2007, and the M.Sc.Eng. degree in electrical engineering from the University of California, Berkeley, CA, USA, in 2012.

He is currently an Electrical Engineer working in the San Francisco Bay Area in the areas of automation and embedded systems design. He worked at a solar manufacturing startup where he gained extensive experience in solar manufacturing, automation, machine vision, and metrology. During his time at the University of California, Berkeley, CA, USA, he got involved in OpenWSN as part of a project to wirelessly monitor outlet power usage and continued involvement due to interests in firmware design and low-power communications protocols.

Xavier Vilajosana (M'13) received the Ph.D. degree in computer sciences from the Universitat Oberta de Cataluny, Cataluna, Spain, in 2009.

He is currently an Associate Professor with the Universitat Oberta de Catalunya (UOC) in the area of computer networks and distributed systems. He is also Chief Innovations Officer and co-founder of Worldsensing and a Fulbright-BE Visiting Professor with the Electrical Engineering and Computer Science Department, University of California, Berkeley, CA, USA, until 2014. During 2008, he was a Visiting Researcher with France Telecom Labs, Paris, France. He is now working on the development of the OpenWSN project at the University of California, Berkeley, CA, USA, Berkeley, an open-source initiative to promote the use of fully standards-based protocol stacks in M2M applications. During the last years of research, he has acquired extensive experience in distributed systems, wireless ad hoc networks, delay tolerant networks and cloud computing. In the area of wireless sensor networks, he proposed several decentralized load balancing schemes for the RPL protocol and actively contributed to the IETF 6LowApp and 6TSCH working groups. He has published several journal and conference papers, has contributed to two books and participated in standardization activities. Xavier is an IEEE member and chair in several prestigious international conferences. His research interests include low power communication protocols, routing, scheduling, and optimization problems in distributed systems at large.

Qin Wang received the B.S. degree from the University of Science and Technology Beijing (USTB), Beijing, China, in 1982, the M.S. degree from Peking University, Beijing, China, in 1985, and the Ph.D. degree from USTB in 1998, all in computer science and engineering.

He is a Professor of School of Computer and Communication, University of Science & Technology Beijing (USTB), China. She has been director of Micro-Architecture and IC Laboratory in USTB since 2000. As a Visiting Scientist (2005–2006) with Cornell University and a Visiting Researcher (2006–2007) with Harvard University, her research and contributions were on wireless sensor network technology and related power consumption modeling from both device and network perspectives. In recent years, she has focused on low-power wireless sensor networks and multiprocessor system-on-chip technology in communications and networking systems. She is currently working on the OpenWSN project with University of California, Berkeley, CA, USA. She has been involved in international wireless network standard development since 2007, including ISA100.11a, IEEE 802.15.4e, and industrial wireless standard WIA-PA proposed to IEC by China.

Thomas Watteyne (M'09) received the Ph.D. degree in computer science from INSA Lyon, France, in 2008.

He is a Senior Networking Design Engineer with Linear Technology, Dust Networks Product Group, Hayward, CA, USA, a group specializing in ultra-low-power and highly reliable wireless sensor networking. He cochairs the IETF 6TSCH group, which is standardizing how to use IEEE802.15.4e TSCH in a 6LoWPAN network. He coordinates Berkeley's OpenWSN project, an open-source initiative which promotes the use of fully standards-based IoT protocol stacks. He is fluent in 4 languages. In 2009–2010, he was a Postdoctoral Researcher with the Berkeley Sensor & Actuator Center, University of California, Berkeley, CA, USA, in Prof. Kristofer S. J. Pister's team. From 2005 to 2008, he was a Research Engineer with France Telecom R&D/Orange Labs working on energy efficiency and self-organizing for wireless multihop networks, together with the CITI Laboratory, France. At that time, he was member of the IEEE Region 8 Student Activity Committee, and serving as the IEEE Region 8 Electronic Communications Coordinator. He has published several journal and conference papers, holds two patents, has contributed to three books, has given several international short-courses, and participated in standardization activities.

Kristofer S. J. Pister received the B.A. degree in applied physics from the University of California at San Diego, La Jolla, CA, USA, in 1982, and the M.S. and Ph.D. degrees in electrical engineering from the University of California, Berkeley, CA, USA, in 1989 and 1992, respectively.

From 1992 to 1997, he was an Assistant Professor of electrical engineering with the University of California, Los Angeles, CA, USA. In 1997, he joined the Department of Electrical Engineering and Computer Sciences, University of California, Berkeley, CA, USA, where he is currently a Professor and Co-Director of the Berkeley Sensor and Actuator Center. He created the term "Smart Dust" and pioneered the development of ubiquitous networks of communication sensors, a concept that has since become a global area of technology R&D. During 2003 and 2004 he was on industrial leave as CEO and then CTO of Dust Networks, a company that he cofounded to commercialize low-power wireless sensor networks. In addition to wireless sensor networks, his research interests include MEMS-based microrobotics and low-power circuit design.