Speculative Precision Time Protocol: submicrosecond clock synchronization for the IoT

Davi Resner, Antônio Augusto Fröhlich
Software/Hardware Integration Lab
Federal University of Santa Catarina
PO Box 476, 88040-900 – Florianópolis, SC, Brazil
{davir,guto}@lisha.ufsc.br

Lucas Francisco Wanner
Institute of Computing
University of Campinas
Av. Albert Einstein, 1251, 13083-852, Brazil
lucas@ic.unicamp.br

Abstract—**Time synchronization is a keystone of Wireless Sensor Networks (WSN). It is fundamental to coordinate the action of nodes in a network and it is also a critical element of several security mechanisms. In this paper, we discuss and evaluate the time synchronization strategy behind the *Trustful Space-Time Protocol* (TSTP), which explores the protocol's cross-layer architecture to speculatively peek through the timestamps and geographic info present in message headers, implementing high-accuracy clock synchronization with minimal insertion of explicit messages. We evaluate the protocol analytically and experimentally. The analytic evaluation is based on the model defined by Schmid [15] for the Virtual High-resolution Time (VHT), while the experimental evaluation was performed on the IEEE 802.15.4-compliant EPOSMote platform running EPOS and TSTP. Our results demonstrate that nodes in the network can be consistently synchronized with sub-microsecond precision while exchanging far less messages than they would with an ordinary, non-speculative implementation, resulting in energy savings. Indeed, precision and energy savings are higher for networks with higher traffic, since more messages are available for peeking. In an experiment scenario in which messages were exchanged between devices every 15 seconds, nodes in the network achieved a synchronization error of approximately 15 microseconds in the worst case, while in a scenario in which messages were exchanged every 3 seconds, synchronization error was less than 0.5 microseconds in the worst case, and approximately 0.25 microseconds on average.**

Keywords-**Clock Synchronization; Time Synchronization; Wireless Sensor Networks; Cross-layer Communication Protocols;**

I. Introduction

Wireless Sensor Networks and virtually any other sort of embedded distributed system, such as vehicles, smart buildings, smart cities, and the smart grid, depend on a consistent perception of time in order to operate properly. These systems rely on it to schedule tasks, to send and listen to messages, and to trigger virtually any sensing and actuation operations. On the eve of the Internet of Things, they will also depend on very precise time synchronization to implement several security algorithms [12].

In this paper, we discuss and evaluate the time synchronization strategy behind the *Trustful Space-Time Protocol* (TSTP) [11], an application-oriented, cross-layer communication protocol initially developed for the Embedded Parallel Operating System (EPOS). TSTP was designed to deliver authenticated, encrypted, timed, georeferenced, SI-compliant data communication support to IoT devices interacting with an IoT gateway, and it was designed to do so in a resource-efficient way. Actually, it reaches beyond a communication protocol as it defines a user interface inspired by the IEEE 1451 Smart Transducer concept of "transducer electronic data sheets". Applications simply declare interest in a given physical quantity in a portion of space-time that is to be measured with a minimum precision and at a given frequency. Nodes matching the criteria periodically send the corresponding data that is selectively forwarded to the gateway. In this scenario, it is fundamental that nodes are aware of their location in space and time in relation to the network. From its design, TSTP implicitly causes timestamps to be constantly transmitted throughout the network, enabling clock synchronization with no additional messages in most cases.

One of TSTP's goals is to eliminate replication of information along the network stack by providing to applications features often necessary in the WSN/IoT context. TSTP itself leverages precise clock synchronization to save energy at the Medium Access Control level [14] and to provide security [12]. Given that precise synchronization is present, cheap, and beneficial, implementing another higher-layer protocol for the same endeavour would replicate timing information unnecessarily: applications can instead simply ask TSTP what time is it, even if that application does not use timestamping with the fine granularity provided.

We discuss TSTP's speculative clock synchronization strategy in depth along with the initialization phase and a prediction mechanism to deal with clock skew during the interval between two messages. A back-off mechanism for long silence intervals is also described. We evaluate the protocol's precision and bandwidth utilization both analytically and experimentally. The analytic evaluation is based on the model defined by Schmid [15] for the Virtual High-resolution Time (VHT), while the experimental evaluation was performed with IEEE 802.15.4 compliant motes running EPOS and TSTP. The key contributions of this work are:

- An analysis of the sources of clock and timestamping synchronization imprecision in networks in general and in IEEE 802.15.4 in particular.

- The speculative time synchronization strategy used in TSTP, which is able to keep nodes in a network consistently synchronized at sub-microsecond ranges while exchanging less messages than they would with an ordinary, non-speculative PTP implementation.

The remainder of this paper is organized as follows: section II presents related work, focusing on strategies for low-cost, high-precision synchronization. Section III presents a discussion on the sources of imprecision in time keeping, showing our strategies to tackle each of the identified issues. Section IV shows the time synchronization model used in our work. Section V presents the design and implementation of the Speculative Precision Time Protocol, which is built upon the Trustful Space-Time Protocol (TSTP) for IoT devices. In section VI we evaluate time synchronization quality with IoT devices using TSTP. Finally VII presents our conclusions.

II. Related Work

Network time synchronization protocols typically work through the exchange of timestamped messages between pairs of nodes [1]. Each message typically contains a local timestamp of the sender. A series of messages are exchanged to estimate the time offset and drift between a pair of nodes [5]. The most widely used protocol of this kind is the Network Time Protocol (NTP) [10]. NTP targets Internet hosts and, due to its multihop nature, suffers from constantly varying communication delays between hosts. Typical synchronization accuracy for NTP is on the order of tens of milliseconds, making it unsuitable for time-critical sensing and actuation applications.

In the wireless embedded sensing context, the Flooding Time Synchronization Protocol (FTSP) [8] is the most widely cited time synchronization protocol. FTSP aims to synchronize an entire multihop network of wireless nodes to a single root node. The root node periodically broadcasts time synchronization messages, each of which contains multiple timestamps. Because FTSP targeted platforms using radios with software-defined medium access control (MAC), it relies on MAC-level timestamping combined with a characterization of interrupt handling timing. Each transmitted timestamp is acquired as close to the physical send event as possible. This combination of precise interrupt handling timing and multiple timestamps per message allowed FTSP to achieve a 1-hop synchronization accuracy of $\sim$1.5 μs. In our work we likewise make use of software-defined MAC to perform timestamping very close to the send event.

Virtual High-resolution Time (VHT) [15] improved on the accuracy and energy performance of FTSP by relying on a combination of two clocks: a low-frequency, low-power clock used to keep the system synchronized and a high-frequency, high-power clock used for high-resolution timestamping. The low-frequency clock (e.g. a 32 kHz crystal) is kept synchronized through a combination of infrequent timestamp exchanges between nodes and local temperature compensation. The high-frequency clock is activated only when fine-grained timestamps are needed and can therefore be duty cycled at very low rates, which leads to significant energy savings (more than 10x improvement with a 0.1% duty cycle). Results with VHT using Epic sensor node cores [2] achieved a mean accuracy of 0.125 μs (identical to the precision provided by the high-frequency 8MHz crystal clock used in the experiments) with a standard deviation of 0.625 μs.

Reference Broadcast Synchronization (RBS) [4] is a *receiver-receiver* synchronization protocol. Instead of synchronizing a pair of sender-receiver nodes, it synchronizes a set of receiver nodes with each other. In RBS, a reference beacon is broadcast and received by multiple nodes in the network. The reference beacon contains no timing information by itself. Instead, each of the receiving nodes captures a local timestamp when a beacon is received. Since the propagation time in wireless networks is negligible under practical conditions (< 0.1 μs for distances of 30m), the beacon arrives at all nodes effectively at the same time, and therefore by exchanging local timestamps receivers can calculate their relative clock offsets. Experimental results with RBS using commodity WiFi hardware showed that it can synchronize two nodes to the order of a few microseconds [4].

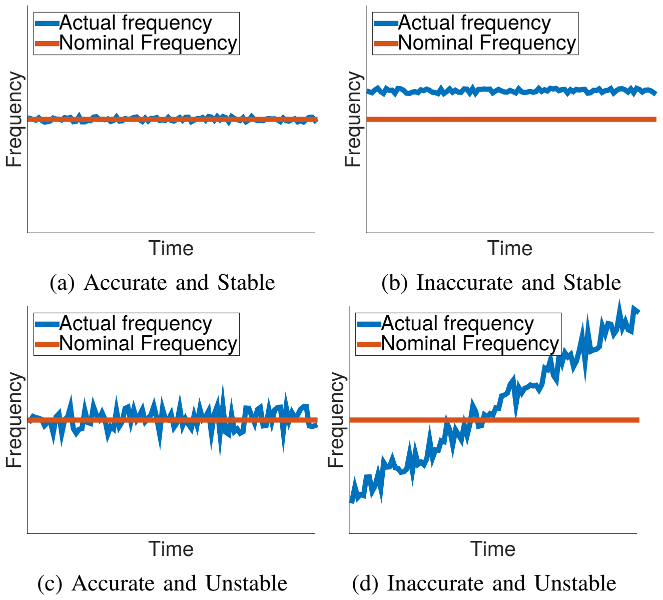
The recent introduction of the IEEE 1588 Precision Time Protocol (PTP) standard [6] emphasized the need for high precision, submicrosecond synchronization for distributed sensing and control systems [3]. Compared to NTP, which uses application-level timestamping of network packets to account for drift and offset, PTP can make use of advanced timestamping capabilities in the network interface hardware in order to reduce the temporal interference introduced by various layers of software. PTP defines a hierarchy of clocks, in which a grandmaster periodically broadcasts *Sync* messages, and other nodes send *Delay Request* messages which are replied by higher clocks in the hierarchy, so that each node can measure with high accuracy both network round-trip time and their own clock offset.

The main focus of the present work is to reduce as much as possible the explicit exchange of time synchronization messages in a context where timestamps are already present in passing messages, while still keeping synchronization accuracy close to the maximum possible offered by the hardware platform.

III. Synchronization and Sources of Imprecision

Time in computing systems is typically kept by counting cycles of a piezoelectric crystal oscillator. The frequency of oscillation is determined by the cut, vibration mode (longitudinal, transverse), and the size of the crystal wafer [16]. Imprecisions and defects in the manufacturing process therefore lead to a deviation in oscillation frequency across different parts with the same nominal frequency. Furthermore, environmental factors such as temperature, aging, drive level, power supply noise, and vibration-induced noise also affect crystal stability and accuracy.

The accuracy of a crystal is its offset from the target nominal frequency, while its stability is the spread of its frequency over time [16]. Figure 1 shows examples for accuracy and stability scenarios for crystal oscillators. Oscillators may be stable and accurate, stable but inaccurate, instable but accurate on the average, and instable and inaccurate. Inaccuracy in frequency leads to the fact that two independent clocks, once synchronized, will drift apart without limit. Figure 2 shows the clock between nodes of out target IoT platform, EPOSMote III, drifting apart over time.



(a) Accurate and Stable

(b) Inaccurate and Stable

(c) Accurate and Unstable

(d) Inaccurate and Unstable

Figure 1: Accuracy and Stability of Crystal Oscillators

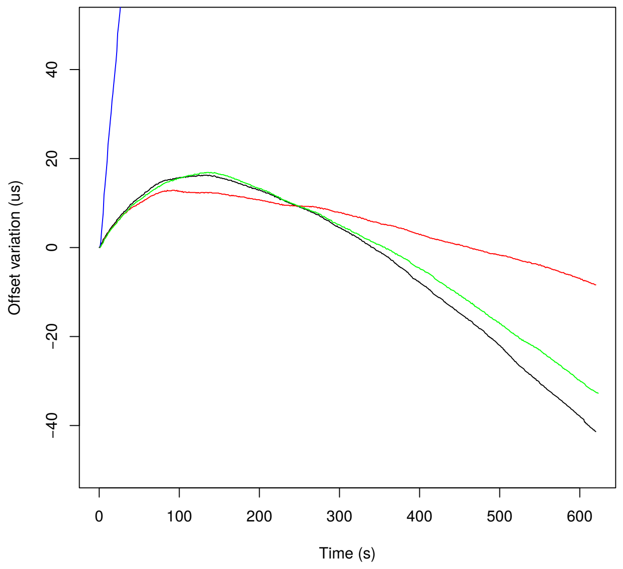


Figure 2: Clock drift of four different EPOSMote III devices in relation to a fifth. One of the motes (in blue) diverged particularly quick in this scenario.

Improving stability and precision of clocks has been the target of research and development, including Temperature-Compensated Crystal Oscillators (TCXO), Microcomputer-compensated crystal oscillator (MCXO), and Oven-controlled crystal oscillator (OCXO), all of which attempt to compensate for systematic and environmental variations in oscillators [7]. These mechanisms come at a cost in terms of system volume, cost, and energy consumption.

In the domain of energy-constrained IoT devices, no compensation mechanism can achieve perfect target frequencies under practical conditions. The IEEE 802.15.4 standard, for example, has a precision requirement for devices of 40 ppm, including temperature and aging variations [9]. Because any two clock sources with differing frequencies will always drift apart, a common measure of time between two independent systems requires a synchronization mechanism. The synchronization process itself is subject to temporal inaccuracies and variations. In the case of wireless IoT devices, these variations arise from the message exchange process itself, which includes the following steps:

1) Signal radio to enter transmission mode
2) Read the local timestamp
3) Copy message to the radio with the timestamp
4) Send Start of Frame Delimiter (SFD)
5) SFD is received
6) Receiver's current timestamp is recorded

The fundamental source of inaccuracy at the time of synchronization is the variation in the time interval taken between acquiring the sender's timestamp (step 2) and the receiver timestamp (step 6). If this time was always constant and quantifiable, synchronization at the time of reading the timestamp would be perfect. This is however not the case in any synchronization strategy. In NTP, for example, this delay is widely variable due to dynamic traffic and congestion in the multiple hops between sender and receiver. In single hop scenarios with a shared medium (such as wireless communications), the delay is variable due to medium access and propagation delays. Finally, even hosts using switched networks with stricter timing guarantees will still suffer variations due to variations in software interrupt handling times.

In IEEE 802.15.4, the quality of the timestamps (steps 2 and 6) is dictated by the standard, which imposes a minimum frequency accuracy requirement. The transmission and reception of messages is bounded by the Start of Frame Delimiter (SFD). Without direct timestamp insertion from either the MAC layer or the hardware itself, message timestamping must be done by software in the higher layers, and is thus subject to variable medium access delays due to network contention. Assuming that the transmission and reception timestamps can be obtained immediately before/after the SFD is actually sent/received, the elapsed time between transmission and reception timestamps would be determined by delays provenient from signal propagation and processing at the physical layer. The later delay is negligible for the small distances between nodes imposed by low transmission powers used in IEEE 802.15.4 networks. The former is affected by the well-defined delays present in the standard:

$s_r = 62.5\frac{symbol}{ms}$	IEEE 802.15.4 Symbol Rate
$T_u = \frac{12}{s_r} = 0.192ms$	IEEE 802.15.4 Turnaround Time
$S_{PHR} = 10symbol$	PHY header + preamble size
$T_{PHR} = 0.160ms$	PHY header + preamble time

and a more random component between SFD transmission and reception representing radio hardware synchronization. This component dictates a physical limit of synchronization made available by the hardware (we measure this quantity in our platform in Section VI).

Acquiring timestamps close to SFD transmission and reception times requires precise control and predictability in the software stack. In general-purpose systems such as Linux, the wide variability in I/O and interrupt handling timing makes it virtually impossible to accurately quantify the time between these two operations, and hence protocol implementations such as PTP require hardware timestamping for high-precision synchronization. With hardware timestamping, the network interface itself acquires a timestamp as soon as SFD or equivalent frame delimiters are sent or received. In this work, by disabling interrupts during the short period between message timestamping and transmission and using only deterministic software instructions during this time, we are able to accurately compensate for the delay between steps 2 and 4. The radio hardware itself records timestamps of received messages, and we consider that the delay between SFD reception and timestamp recording (steps 5 and 6) is zero.

In addition to the variation in the time interval between timestamping operations in senders and receivers, synchronization accuracy over time is also influenced by the rate at which synchronization messages are exchanged. In our strategy, timing information is included with every message. Timestamping is a fundamental requirement for IoT applications, not only for synchronization, but also for sensing and control, localization, and security. Our synchronization strategy "piggybacks" on the timestamping required for these other functionalities. Because there are normally no explicit synchronization messages (as every message is a potential synchronization message), the frequency of synchronization will depend on network traffic, with the effect that networks with higher traffic will perform synchronization more frequently and therefore will have higher quality of time. We define an explicit synchronization mechanism for nodes to use in cases where synchronization is particularly important and the network goes silent for too long.

IV. Time Model

To achieve ϵ-precision time synchronization, a node needs [15]:

1) a high-resolution clock source with frequency f_0, and
2) message time-stamping with accuracy $\epsilon \pm 1/f_0$

Let d_{TX} be the total time delay between immediately before step 2 and immediately after step 6 (Section III), and $c_N(t_m)$ represent the value of the timestamp counter of node N at physical time t_m. If d_{TX} is known and has a small enough jitter, when node B receives a timestamp $c_A(t_1)$ from node A, it can trivially determine its clock offset ϕ

$$\phi = c_A(t_1) - (c_B(t_1') + d_{TX}) \tag{1}$$

where $c_B(t_1')$ is the time recorded by node B upon reception of the message. The accuracy of this estimation is determined by the jitter in d_{TX}.

Once the offset is corrected, node B is ready to estimate its clock drift in relation to A. After it receives a second message containing $c_A(t_2)$, the clock drift is given as:

$$\hat{f}_e = \frac{(c_A(t_2) - c_B(t_2')) - (c_A(t_1) - c_B(t_1'))}{c_A(t_2) - c_A(t_1)} \tag{2}$$

The accuracy of this estimation is [15]:

$$\delta_Q = \frac{1}{(t_2 - t_1) \cdot f_A} \tag{3}$$

where f_A is the frequency of clock A.

In the case of duty-cycled WSNs where nodes listen to the channel at most once every P units of time, there is a lower bound $\delta_Q \geq (P \cdot f_A)^{-1}$

V. TSTP's Speculative Precision Time Protocol

TSTP messages are preceded by a MAC preamble and a Header containing information such as sender time and location (Figure 3). Besides time synchronization, TSTP provides geographic localization to every node [11], and nodes also know the location of the gateway. Gateways declare interest in physical quantities by sending Interest messages, which are geographically routed to a portion of space, and are subscribed to by sensors in the area that can deliver the requested measurements. These sensors periodically respond with Data messages, which are routed to the gateway's location.

TSTP's Medium Access Control mechanism (TSTP MAC) [13] is designed to run directly on top of an IEEE 802.15.4 2450MHz DSSS PHY layer. It is a cross-layer MAC protocol which handles collision avoidance, implicit acknowledgement, duty cycling and greedy geographic routing. Nodes only turn their radio receivers on every Checking Interval *CI*, and transmitters send a long preamble before the actual message to make sure that every node will wake up in time to detect the transmission. Preambles consist of Microframes (*MF*) which are carefully timed [13] and contain useful information, as depicted in Figure 4: a direction flag (*All-Listen*) which indicates whether the message is destined to the gateway or another node, countdown to data transmission (*Count*), sender distance to the destination (*Last Hop distance*), message sequence number (*ID*) and a checksum (*FCS*).

All neighboring nodes within communication range of the sender sense the channel every CI interval, obtain a Microframe and extract the information; then, only eligible receivers (nodes closer to the destination) go back to sleep and wake up to receive the data at the time indicated by

Header Format								
Bits: 3	**1**	**2**	**2**	**8**	**3*sb**	**64**	**3*sb + tb**	**0 or 32**
Message Type	Time Request	Spatial Scale	Temporal Scale	Location Confidence	Last Hop x,y,z	Last Hop Time-stamp	Origin x,y,z,t	Location Deviation

Figure 3: TSTP message header format. *sb* and *tb* are variables defined by the *Spatial Scale* and *Temporal Scale* codes [11].

Microframe				
Bits: 1	**8**	**32**	**15**	**16**
All Listen	Count	Last Hop distance	ID	FCS

Figure 4: TSTP MAC Microframe format.

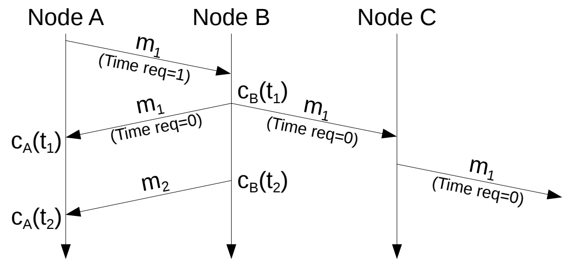


Figure 5: TSTP explicit time synchronization example. Time flows from top to bottom and nodes to the right are closer to the gateway.

the countdown. Nodes that receive the data without error are relay candidates, and start a back-off timer based on their own distance to the destination, which when elapsed will trigger a channel check (CCA). The node with the shortest back-off timer will sense no channel activity and proceed to forward the message by transmitting the preamble for CI units of time. Other relay candidates will detect the winner's preamble containing the same sequence number, drop the data and go back to sleep since the packet is already being forwarded. The Microframes also serve as a passive acknowledgment to the previous sender of the message. As a consequence of the way relay candidates are determined, packets are geographically routed to the final destination in a greedy way.

TSTP offers two modes of clock synchronization: speculative and explicit. Speculative mode runs continually, taking advantage of timestamps present in messages that pass through the nodes. Each node N selects a peer P_N closer to the gateway to synchronize to by observing traffic from the neighbors. Every time N overhears a transmission from P_N, say at time t_1, it records $c_N(t'_1)$ as the reception time of the message and updates its clock offset according to Equation 1. Starting from the second message N overhears from P_N, it is able to update its clock drift in relation to P_N as

$$\hat{f}_e = \frac{(c_{P_N}(t_i) - c_N(t'_i)) - (c_{P_N}(t_1) - c_N(t'_1))}{c_{P_N}(t_i) - c_{P_N}(t_1)} \qquad (4)$$

and the corresponding accuracy of the estimation δ_Q (Equation 3). If P_N detects that the network went silent for so long that it might have drifted out of sync more than its acceptable limit, it tries an explicit time synchronization request by sending a message with the *Time request* bit set. This message may be either a new message destined to P_N or any other message N would already send. When a node receives a message of this kind, it clears the *Time request* bit and forwards it as usual, but after the forwarding, it inserts a new message destined only for N to fetch all the necessary timestamps. Since the minimum time between two messages is *CI* (usually hundreds of milliseconds), N is now synchronized with a known minimum accuracy $\delta_Q \geq (CI{\cdot}f_{P_N})^{-1}$. Figure 5 illustrates this process.

VI. Evaluation

We analyzed the characteristics exposed in Section III and implemented the protocol in EPOS and the EPOSMoteIII platform, which is based on Texas Instrument's CC2538 SoC featuring an ARM Cortex-M3 32MHz processor and an IEEE 802.15.4 transceiver with 2450MHz DSSS PHY. For timestamping, we use the SoC's 32MHz timer with ±40ppm accuracy dedicated to network software.

We have control over the MAC, and the software and hardware involved are highly deterministic. The time from the start of step 1 to the end of step 4 (Section III) was measured with an oscilloscope to be exactly 352.17μs, only 170ns above the expected value ($T_u + T_{PHR}$), with a detected jitter equal to the oscilloscope's own period (5ns). Recording of the receiver timestamp is handled by hardware, so the time between steps 5 and 6 is assumed to be zero.

We replicated the experiments carried out by Schmid [15] for the CC2420 platform to determine the time between the hardware SFD signal (exported by the platform as GPIO) at the sender and at the receiver in our platform. To do so, we set up one node to transmit messages periodically and two to simply listen. The nodes were distributed close to each other (less than 30cm away) and with no obstacles in between. We observe the SFD signals at each of the three nodes with an oscilloscope and mark their time differences (Figure 6). The delays observed range from 3.038μs to 3.225μs, representing a jitter of 187ns. Using the average value of 3.1315μs, the maximum unpredictable variation of the actual delay for a given transmission is 93.5ns. This number represents the hardware-imposed limit of synchronization we can consistently get at the instant of offset determination between two nodes.

To evaluate the performance of the clock drift estimation component, we set up four motes in a star topology distributed in positions similar to the previous experiment. We set three motes as receivers and a single transmitter, which

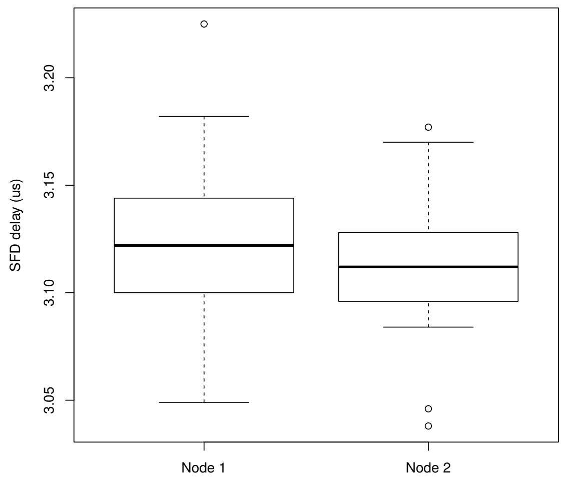


Figure 6: Jitter in Start of Frame Delimiter transmission.

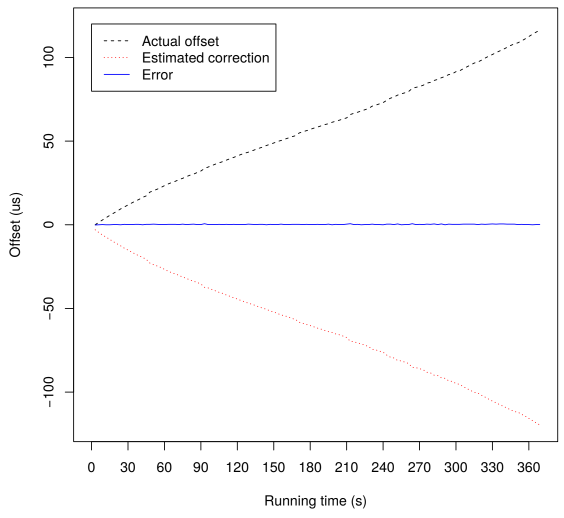


Figure 7: Actual offset from the gateway before calibration, estimated correction value based on previous observations and its corresponding error for Node 1 in Figure 8.

sends timestamps at a constant interval of 3 (Figure 8) or 15 seconds (Figure 9). Each receiver node listens to every message and independently synchronizes with the transmitter. When a message arrives at time t_i, the receiver first gets $c_N(t'_i)$ without any offset correction, then estimates with its previous calibration variables what the value of $c_N(t_i)$ is going to be after instant offset determination:

$$\hat{c}_N(t_i) = \hat{f}_e \cdot (c_N(t'_i) - c_N(t'_{i-1}) + \phi)$$

It then calculates its immediate offset from the sender (Equation 1), updates its clock drift estimation (Equation 4), gets the actual $c_N(t_i)$, and gets the error as $c_N(t_i) - \hat{c}_N(t_i)$. Figure 7 compares the instant offset at the moment of message reception with the previously estimated correction values during execution of Node 1 in Figure 8. The error is the difference between the two. In Figures 8 and 9, only the error and instant offset values are shown.

More than achieving a sub-microsecond instant synchronization per-hop close to the physical limit that is comparable to PTP, VHT and RBS, the results show that with relatively light traffic of one message every three seconds, we keep the network synchronized to sub-microsecond accuracy at *all times* without insertion of any extra message in the network (while PTP, for instance, requires at least 3 explicit synchronization messages). This accuracy gets worse as traffic becomes more sparse, reaching almost 15μs with a message period of 15 seconds. In the worst case observed, the precision of error estimation is 0.06% of the actual clock offset.

VII. Conclusion

In this work, the clock synchronization strategy of the *Trustful Space-Time Protocol* was presented in detail. TSTP is an application-oriented communication protocol for the IoT which closely integrates distinct components, such as Medium Access Control and time synchronization, to efficiently deliver functionality often needed by WSN and IoT applications.

We carried out an analysis of sources of synchronization errors in networks in general and IEEE 802.15.4 networks in particular, with experiments to characterize the delay parameters of the EPOSMote III platform.

We also presented a speculative time synchronization strategy which benefits from network traffic to synchronize the clocks across the network with minimal injection of explicit messages. We showed that in a case where one packet was sent every three seconds, nodes could synchronize their clocks to sub-microsecond precision without inserting any time-synchronization-specific messages.

VIII. Acknowledgements

This work was supported in part by CNPq under grants 449996/2014-0, 404874/2013-4, by CAPES PROCAD under grant 2966/2014, and by CTIC under grant 002571.

References

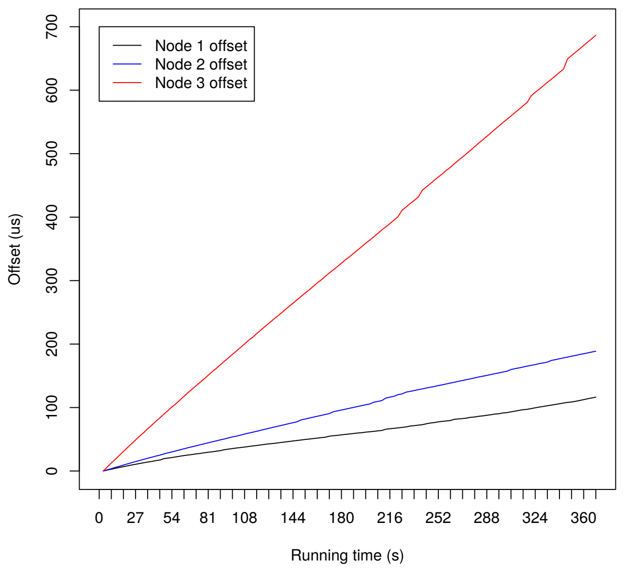
[1] Flaviu Cristian. Probabilistic clock synchronization. *Distributed Computing*, 3(3):146–158, September 1989.

[2] Prabal Dutta, Jay Taneja, Jaein Jeong, Xiaofan Jiang, and David Culler. A building block approach to sensornet systems. In *Proceedings of the 6th ACM Conference on Embedded Network Sensor Systems*, SenSys '08, pages 267–280, New York, NY, USA, 2008. ACM.

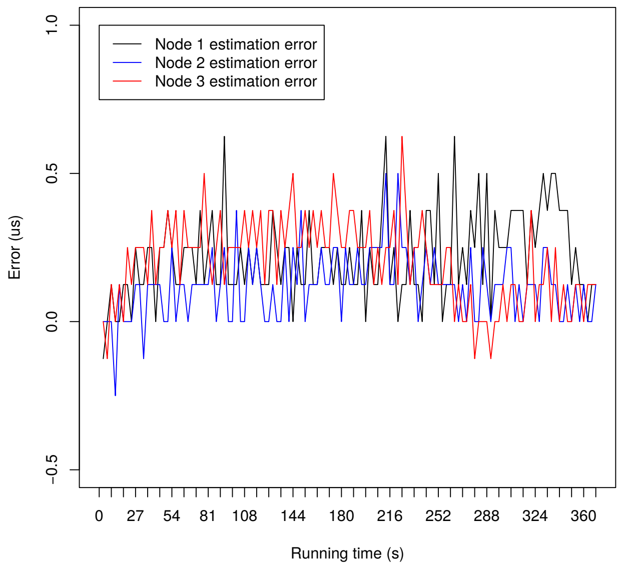
[3] John C. Eidson. *Measurement, Control, and Communication Using IEEE 1588 (Advances in Industrial Control)*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2006.

[4] Jeremy Elson, Lewis Girod, and Deborah Estrin. Fine-grained network time synchronization using reference broadcasts. *SIGOPS Oper. Syst. Rev.*, 36(SI):147–163, December 2002.

[5] R. Gusella and S. Zatti. The accuracy of the clock synchronization achieved by tempo in berkeley unix 4.3bsd. *IEEE Transactions on Software Engineering*, 15(7):847–853, Jul 1989.

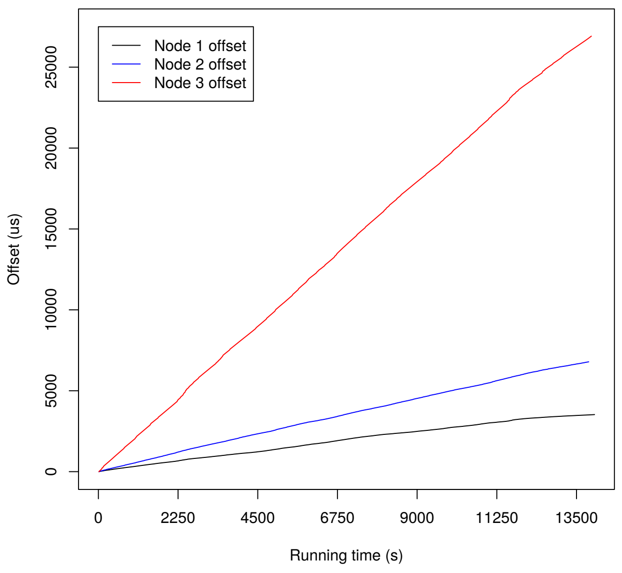


(a) Clock offsets for each node in relation to the gateway before calibration.

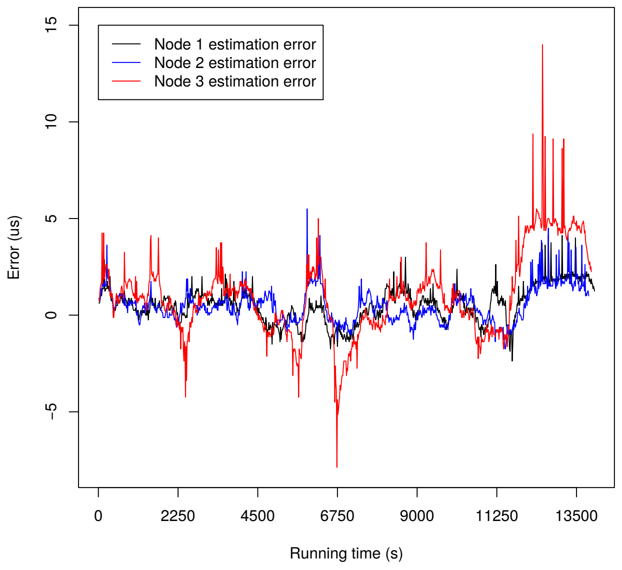


(b) Error in the clock offset for each node after calibration using a previous estimation of clock drift.

Figure 8: Three EPOSMoteIII devices synchronizing with a fourth. Messages with timestamp are sent by the synchronizer node every 3 seconds.



(a) Clock offsets for each node in relation to the gateway before calibration.



(b) Error in the clock offset for each node after calibration using a previous estimation of clock drift.

Figure 9: Three EPOSMoteIII devices synchronizing with a fourth. Messages with timestamp are sent by the synchronizer node every 15 seconds.

[6] IEEE. Ieee standard for a precision clock synchronization protocol for networked measurement and control systems. *IEEE Std 1588-2008 (Revision of IEEE Std 1588-2002)*, pages c1–269, July 2008.

[7] L. L. Lewis. An introduction to frequency standards. *Proceedings of the IEEE*, 79(7):927–935, Jul 1991.

[8] Miklós Maróti, Branislav Kusy, Gyula Simon, and Ákos Lédeczi. The flooding time synchronization protocol. In *Proceedings of the 2Nd International Conference on Embedded Networked Sensor Systems*, SenSys '04, pages 39–49, New York, NY, USA, 2004. ACM.

[9] A. M. Mehta and K. S. J. Pister. Frequency offset compensation for crystal-free 802.15.4 communication. In *International Conference on Advanced Technologies for Communications (ATC)*, pages 45–47, Aug 2011.

[10] D.L. Mills. Internet time synchronization: the network time protocol. *IEEE Transactions on Communications*, 39(10):1482–1493, Oct 1991.

[11] Davi Resner and Antônio Augusto Fröhlich. Design Rationale of a Cross-layer, Trustful Space-Time Protocol for Wireless Sensor Networks. In *20th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA 2015).*, pages 1–8, Luxembourg, Luxembourg, September 2015.

[12] Davi Resner and Antônio Augusto Fröhlich. Key Establishment and Trustful Communication for the Internet of Things. In *4th International Conference on Sensor Networks (SENSORNETS 2015)*, pages 197–206, Angers, France, February 2015.

[13] Davi Resner and Antônio Augusto Fröhlich. TSTP MAC: a Cross-Layer, Geographic, Receiver-Based MAC Protocol for WSNs. In *Brazilian Symposium on Computing Systems Engineering.*, Foz do Iguaçu, Brazil, November 2015.

[14] Davi Resner and Antônio Augusto Fröhlich. TSTP MAC: A Foundation for the Trustful Space-Time Protocol. In *14th IEEE/IFIP International Conference on Embedded and Ubiquitous Computing (EUC 2016)*, To appear, Paris, France, August 2016.

[15] Thomas Schmid, Prabal Dutta, and Mani B. Srivastava. High-resolution, low-power time synchronization an oxymoron no more. In *Proceedings of the 9th ACM/IEEE International Conference on Information Processing in Sensor Networks*, IPSN '10, pages 151–161, New York, NY, USA, 2010. ACM.

[16] Hui Zhou, Charles Nicholls, Thomas Kunz, and Howard Schwartz. Frequency accuracy and stability dependencies of crystal oscillators. Technical report, Carleton University, 11 2008.