Time Synchronization for Resource-Constrained Multi-Hop Wireless Sensor Networks based on Hop Delay Estimation

Ville Kaseva, Timo D. Hämäläinen, Marko Hännikäinen
Tampere University of Technology, Department of Computer Systems
P.O.Box 553, FI-33101 Tampere, Finland
{ville.a.kaseva, timo.d.hamalainen, marko.hannikainen}@tut.fi

***Abstract*—Ultra low-power Wireless Sensor Networks (WSNs) use duty cycled medium access protocols and dynamic routing for multi-hop communication. Transmitted packets can have variable delays and routes. Thus, data timestamps and temporal order of the packets are unknown. However, timing information is essential for many WSN applications. In this paper, we present a novel time synchronization protocol for application packets in multi-hop WSNs. The proposed protocol is based on calculating delays as packets traverse from node to node. It provides high energy-efficiency by minimizing communication overhead. Fault tolerance is achieved by fully distributed operation. The protocol computational complexity is low, including only simple operations. We experimented the proposed protocol using real WSN hardware communicating in 2.4 GHz radio band. The maximum synchronization errors ranged from 72 μs to 909 μs, and the average errors from 20 μs to 153 μs when the hop count was varied from two to six and varying protocol parameters were used. With values extrapolated from the experimental results, the maximum error was 15 ms, which occurred with 100 hops. The paper presents the design, implementation, and experimental results for low-complexity synchronization protocol with maximum errors ranging from 72 μs to 15 ms with varying network sizes.**

***Keywords*-Wireless Sensor Networks; Time synchronization.**

I. INTRODUCTION

Ubiquitous computing requires small and low-cost devices embedded into everyday objects. Wireless Sensor Networks (WSNs) form a key building block in the construction of these smart environments. WSNs consist of densely deployed, independent, and collaborating low-cost sensor nodes, which are highly resource-constrained in terms of energy, processing, and data storage capacity [1]. The nodes can sense their environment, process data, and communicate over multiple short distance wireless hops. The network self-organizes and implements its functionality by co-operative effort. WSN nodes must operate for years with small batteries or by harvesting their energy from the environment. Minimizing communication is essential in achieving energy-efficiency since the radio transceiver is the most power-consuming component in a WSN node [2].

Ultra low-power WSNs use duty cycled Medium Access Control (MAC) protocols and autonomous dynamic routing. The nodes are active only a fraction of the time and the routes of subsequent packets can differ. This results in unknown sensing times, variable forwarding delays, and unguaranteed temporal order of the packets. However, many applications require that timing information of sensed phenomena can be resolved. In basic monitoring, it is usually critical to know the actual sensing times of the reported values. Furthermore, e.g., inferring target velocity from a series of proximity detections [3] requires that the order of (inferring) the sensed events is known.

The most straightforward way for obtaining accurate time information would be to equip every node with a Global Positioning System (GPS) receiver, Universal Time Coordinated (UTC) signal receiver or an accurate atomic clock. However, this would be infeasible in WSN nodes due to increased size, cost, and energy consumption. Although effective in the Internet, the widely used Network Time Protocol (NTP) [4] is too inflexible for WSNs with ad-hoc operation [5].

Commonly, WSN synchronization protocols target at achieving common time locally among a set of neighboring nodes or globally to the whole network. In these, the synchronization protocol operation is fully de-coupled from application-level operation resulting in continuous common time keeping among the nodes. Thus, there is constant synchronization overhead and the overhead is also included in nodes that do not require time information. In these protocols, the time service quality is common to every node and application. This may result in unnecessary accuracy and overhead for some applications, and redundant synchronization overhead during periods time information is not actually needed.

In this paper, we present a delay-based time synchronization (D-DYNC) protocol for application-level time resolution in multi-hop WSNs. The proposed protocol is based on calculating delays as packets traverse from node to node. D-SYNC provides following key benefits compared to related protocols:

- **High energy-efficiency:** D-SYNC does not require explicit messaging to achieve synchronization. The delay information is piggybacked in application packets. The timing accuracy can be adjusted with simple delay field shift operations according to application requirements. This adjustment minimizes the packet delay field size,

 ISBN: 978-1-61208-144-1

and thus, the transmitted overhead data while providing required accuracy. Synchronization can be switched on on-demand only for required time periods, e.g., when a specific event is sensed. These characteristics minimize the communication overhead, and thus, energy consumed by the radio.

- **Fault tolerance:** D-SYNC is fully distributed having no dedicated time reference nodes nor single points of failure. In dynamic WSNs, there may be periods when nodes are not in the radio range of any other nodes in the network or multiple re-transmissions are required due to poor links. D-SYNC can still maintain timing for the buffered packets and the buffers can be unloaded as connectivity is re-established.
- **Minimal computational complexity:** The complexity of the D-SYNC protocol is low including only timestamp value saving, delay calculation using subtraction, and shifting for timing accuracy adjustment.

We present a mathematical analysis and prototype experiments for the D-SYNC protocol. The mathematical analysis provides tools for estimating the maximum achievable accuracy and the required minimum delay field width for variable WSN deployments. We experimented the D-SYNC protocol with real resource-constrained WSN nodes. Accuracy results are presented for varying hop and delay field shift amounts. The used hardware is typical for resource-constrained WSN nodes having a simple 8-bit microcontroller with 2 MIPS performance for processing and a low-power low-cost 2.4 GHz radio for communication. The radio frame length is 32 B requiring minimal protocol overhead for maximizing application payload data content in the packet.

The rest of this paper is organized as follows. The related work is presented in Section II. The D-SYNC design and mathematical analysis are presented in Section III. Section IV introduces the used prototype hardware platform. The experiments and results are presented in Section V. Finally, Section VI concludes the paper and present the future work.

II. Related Work

Next, we present the most essential multi-hop time synchronization protocols proposed for WSNs surveyed, e.g., in [5] and [6]. The protocols represent fundamental approaches to clock synchronization [6]. We compare them against D-SYNC benefits listed in the previous section.

The Timing-sync Protocol for Sensor Networks (TPSN) [7], the Lightweight Tree-based Synchronization (LTS) protocol [8], the Delay Measurement Time Synchronization (DMTS) protocol [9], and the TSync protocol [10] use a dedicated time reference node, a tree structure, and periodical synchronization messages forwarded throughout the tree. In these, the periodical synchronization message exchanges inflict continuous energy consumption overhead to the nodes. Fault tolerance against lost time reference node is not considered at all or is mitigated by using several reference nodes, which are chosen using a leader election algorithm. In either case, single points of failure, the reference node(s), exist in the network.

LTS and TSync provide also a de-centralized version where nodes can query time information from a reference node. In these, the messaging overhead is large since the queries have to be first forwarded to the reference node, possibly via multiple hops, and then synchronization is achieved by reversing the path of the query.

The Flooding Time Synchronization Protocol (FTSP) [11] floods synchronization messages, originating from a reference node, periodically through the network inflicting significant continuous energy consumption overhead. There is no single point of failure since any node in the network can act as the time reference.

In Reference-Broadcast Synchronization (RBS) [12], clock parameters are exchanged with every neighboring node. This method incurs significant messaging overhead and energy consumption penalty, which increases with the density of the nodes. RBS reduces the overhead with on-demand post-facto synchronization in situations where continuous synchronization is not not needed. Still, when continuous synchronization is required, a constant messaging overhead occurs also in RBS.

The protocol presented in [13] achieves multi-hop synchronization using a tree structure. The periodical synchronization message exchanges inflict continuous energy consumption overhead to the network. Also, since clock parameters are exchanged with every neighboring node, with which synchronization is required, the overhead increases with the synchronized neighbor amount.

The Time-Diffusion synchronization Protocol (TDP) [14], and the protocol proposed by Li et al. [15] achieve network wide time by co-operative effort of all the nodes in the network. The complexity of both protocols is high requiring lot of messaging [5]. The protocols do not rely on a single reference node increasing fault tolerance.

III. D-SYNC Design

The D-SYNC protocol is based on calculating cumulative delay for a packet as it traverses via a multi-hop network. Each synchronized packet includes a delay field, which is updated at every hop. Every node forwarding a synchronized packet adds the local delay incurred by that node using its local clock. Using this delay value included in the packet, a destination node can resolve the source event time in its local time base.

A. Single-Hop Delay Calculation

The undeterministic delays during a packet exchange are the main contributors for synchronization inaccuracy. The total delay incurred by one packet exchange can be divided

 ISBN: 978-1-61208-144-1

into send, access, transmission, propagation, and receive delays [16]:

- The *send delay* includes the time spent at the source node to construct the packet, the delays incurred by the operating system, the buffering delays including re-tranmsissions, and the time required to transfer the packet to the MAC layer for transmission. The send delay is undeterministic and highly variable.
- The *access delay* includes the time spent waiting for access to the wireless communication medium. It depends on the used MAC protocol. The access delay is undeterministic and can be highly variable.
- The *transmission delay* (t_{tx}) refers to the time it takes to transmit a single packet over the wireless medium. It is directly proportional to the length of the transmitted packet and inversely proportional to the radio data rate making it relatively deterministic. This delay also includes deterministic and undeterministic radio specific delays. An example of a deterministic radio delay is the radio start-up transient time, during which the radio hardware is powered up.
- The *propagation delay* changes as a function of distance. For radio waves it is below 1 μs for distances under 300 m. The ns scale error caused by it does not contribute significantly to the error budget of synchronization in WSNs [12] and is considered to be zero.
- The *reception delay* consists of the time it takes for the radio receiver to process the incoming bits and notify the host of packet arrival. Commonly, much of the physical packet processing is done in the radio hardware and a packet is delivered to the host via a digital interface. Thus, this delay is relatively small. The undeterministic components can be mainly contributed to the small variability in interrupt latencies.
- The *receive delay* includes the time required to transfer the packet to the receiving layer. It includes the delays incurred by the operating system and the buffering delays. The receive delay is undeterministic and highly variable.

The operation principle for calculating single hop delay in the D-SYNC protocol is illustrated in Figure 1. A packet is triggered for transmission at time t_0 at the source Node i. The time value is saved to internal packet data structure in the local time of Node i, $L_i(t_0)$, where $L_i(t_k)$ denotes real time t_k in the time base of node i. After t_0, the packet is constructed and scheduled for transmission. Just prior to transmission, at time t_1, the delay value is calculated using the previously saved and current time values. This delay is inserted to the packet at the MAC protocol or the radio driver (depends on implementation specifics). The calculated delay value is $L_i(t_1) - L_i(t_0)$. Now, the packet is transferred to the radio transceiver hardware, which transmits it after a radio specific delay. Calculating the delay just before transmission

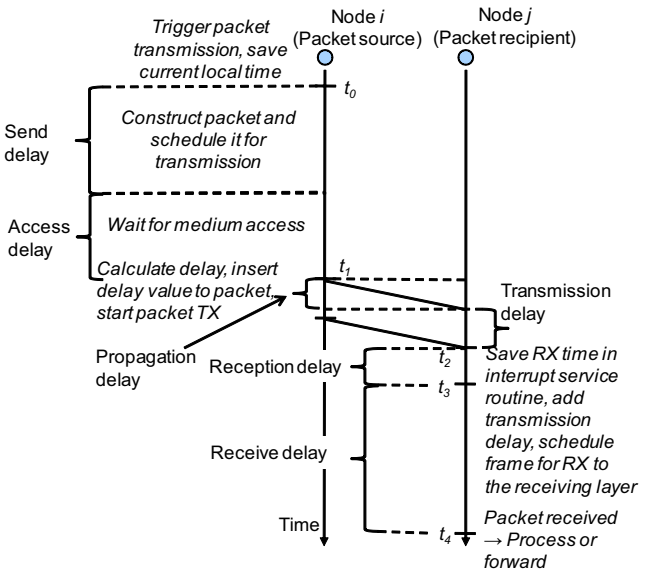

Figure 1. Example of single hop delay calculation. The total delay incurred by one packet exchange can be divided into send, access, transmission, propagation, and receive delays. By calculating delay values just prior to transmission, the highly undeterministic delays can be mitigated and delay estimation accuracy improved.

mitigates the highly variable send and access delays.

Furthermore, the time $L_i(t_1)$ is again saved to the internal packet structure in case the packet is not actually received by the next hop node and re-transmission is needed. In the case of re-transmission the additional delay can be calculated and added to the current delay value using the saved time value $L_i(t_1)$ and the new transmission moment of the packet.

After the transmission delay (t_{tx}) and the propagation delay, the packet is received by the recipient Node j at time t_2. After interrupt latency, the recipient saves the reception time of the packet $L_j(t_3)$. This saved value can be again used for internal delay calculation at node j. Saving the time value just after reception mitigates the highly variable reception and receive delays. Furthermore, the transmission delay is added to the packet delay at this stage due to successful packet exchange. Thus, the only remaining inaccuracies are the propagation delay and the possibly undeterministic timing inflicted by the radio hardware.

B. Multi-Hop Delay Calculation and Time Resolution

To calculate the multi-hop delay, the single hop delay calculation is performed at every hop and the resulting delays added to the delay field in the packet. This means that also the inaccuracies accumulate hop-by-hop but are minimized by low-level timestamping. An example of the multi-hop delay calculation is presented in Figure 2. Node 1 schedules packet for transmission at time t_0 and saves time value $L_1(t_0)$ to the internal packet data structure. The physical packet transmission to the next hop (Node 2) is started at at time t_1. Thus, calculated delay from packet

 ISBN: 978-1-61208-144-1

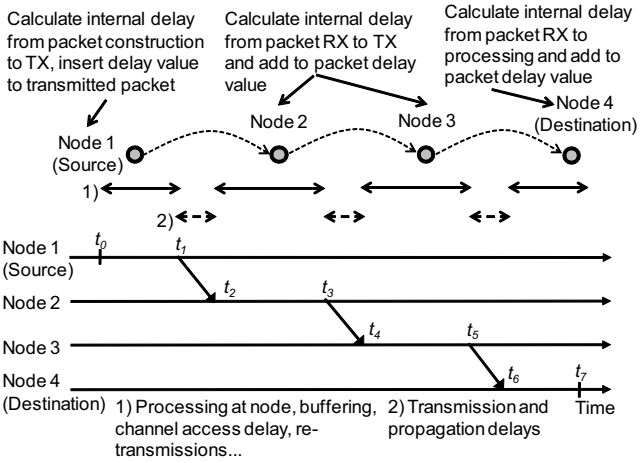

Figure 2. To calculate the multi-hop delay, the single hop delay calculation is performed at every hop and the resulting delays added to the delay field in the packet.

construction to transmission, inserted to the packet at Node 1, is $L_1(t_1) - L_1(t_0)$.

A packet is received at Node 2 at time t_2 and the time value $L_2(t_2)$ is saved. Also, the transmission time (t_{tx}) of the previous wireless packet exchange is added to the total delay at this stage. The physical packet transmission to the next hop (Node 3) is started at at time t_3. Thus, calculated delay from packet reception to transmission, added to the packet delay field at Node 2, is $L_2(t_3) - L_2(t_2) + t_{tx}$. Similarly, the added delay at Node 3 is $L_3(t_5) - L_3(t_4) + t_{tx}$.

Finally, the packet is received at the destination, Node 4, at time t_6. Again, the packet reception time is saved as the local time of Node 4, $L_4(t_6)$. The final delay, from reception to actual processing moment, added at Node 4, is $L_4(t_7) - L_4(t_6) + t_{tx}$.

Thus, the remove event time (t_0), as estimated by the destination Node 4 at time t_7 in its local time base, is $L_4(t_7) - t_{delay}$, where t_{delay} is the cumulative delay value obtained from the received packet. Since all the delay calculations are done locally no continuous synchronization nor time base conversions are required.

In general, the remote event time for packet k ($t_{pkt(k)}$) in the time base of destination node i is

$$L_i(t_{pkt(k)}) = L_i(t_{pr}) - t_{delay(k)}, \quad (1)$$

where t_{pr} is time the packet is processed at the destination, and $t_{delay(k)}$ is the cumulative delay value for packet k obtained from the packet.

C. Accuracy and Resource Consumption

There is a tradeoff between the maximum achievable accuracy and the amount of used bits for synchronization messaging (overhead). These can be be varied by simple shift operations. When the delay field is shifted to the right the least significant bits are lost but most significant bits are gained. This results in lower accuracy but less overheard bits and larger maximum delay value.

Thus, the delay field overhead can be minimized when the scale of required accuracy, maximum number of hops in the WSN, and the worst case hop delay can be estimated. On the other hand, the shifting enables graceful degradation of accuracy at run-time. With a fixed delay field width, the field overflow can be monitored hop-by-hop and additional shifting can be used when required.

With shift amount of S bits, the maximum achievable accuracy in ticks (ε) is given by

$$\varepsilon = 2^S \quad (2)$$

and the maximum presentable delay in ticks (D) is

$$D = 2^{N_{bits}+S} - 1, \quad (3)$$

where N_{bits} is the number of bits in the delay field.

E.g., for a WSN where maximum number of hops is 20, the worst case hop delay is 10 s, and the internal clock tick in the nodes is 1 μs, the maximum achievable accuracy values and presentable delays with varying delay field widths are as follows.

With an 8-bit delay field the required shift amount is 20 bits resulting in maximum accuracy of 1 s and maximum presentable delay of 268 s. The corresponding values for 16-bit and 24-bit delay fields are shift of 12 and 4 bits, and maximum accuracy of 4 ms and 16 μs, respectively. For 16-bit and 24-bit delay fields the maximum presentable delay is 268 s with the given shift values. With a 32-bit delay field, no shift is required, maximum accuracy is 1 μs corresponding to one clock tick, and the maximum presentable delay is 4295 s.

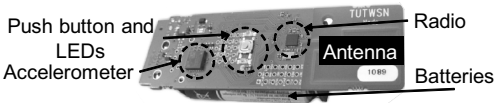

Figure 3. Prototype hardware platform circuit board.

IV. Prototype Hardware Platform

The prototype hardware platform is presented in Figure 3. The platform uses a Microchip PIC18F8722 MCU, which integrates an 8-bit processor core with 128 kB of FLASH program memory, 4 kB of RAM data memory, and 1 kB EEPROM. The used clock speed of the MCU is 8 MHz resulting in 2 MIPS performance.

For wireless communication the platform uses a Nordic Semiconductor nRF24L01 radio transceiver operating in the 2.4 GHz ISM frequency band. The radio data rate is 1 Mbps and there are 80 available frequency channels. Transmission power level is selectable from four levels between -18 dBm and 0 dBm with 6 dBm intervals and $\pm$4 dBm accuracy. Loop type antenna is implemented as a trace on the Printed Circuit Board (PCB).

 ISBN: 978-1-61208-144-1

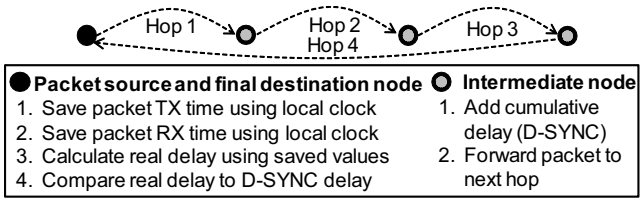

Figure 4. Example experimental scenario for 4 hops. The packet traverses a loop and source node local clock is used as a reference.

The platform is equipped with a 32.768 kHz crystal for real-time clock implementation. This results in clock resolution of approximately 30 μs. The resolution is enhanced by using the MCU's own oscillator and delay loops for fine timing below 30 μs.

V. Experiments and Results

For the experiments, the D-SYNC protocol was implemented on the prototype hardware platform. For communication, a multi-hop protocol stack implementation with fixed routing and fixed one second delay per hop was used. A ring topology was used to implement scenarios with varying amount of hops. An example of the topology with 4 hops is illustrated in Figure 4. The scenarios are general as the path traversed by a single packet can always be reduced to a simple chain of nodes independent of the actual network topology used.

The experimental scenarios consisted of 2, 4, and 6 hops. Each of the three scenarios was experimented with no shift, giving a time resolution of 1 μs, and with shift of 8 bits, giving a time resolution of 256 μs. For each test, the packet was transmitted 100 times, of which maximum delay errors and average delay errors were calculated. It must be noted that the experimented scenarios represent a subset of possible scenarios that can occur in a WSN. The number of hops, the length of channel access delays, and the required shift amount can have variations. These are inherently handled by the D-SYNC protocol. The presented experimental setup provides a well-defined and reproducible framework for estimating the orders of magnitude of achievable accuracy. Furthermore, it gives total control over the scenario parameters.

In the experiments, the source node was set to be also the final destination node letting the transmitted packet traverse a loop. Upon transmitting a packet, the source node saved the initial packet transmission time and after reception the reception time. Using these times a reference delay was calculated. The intermediate nodes forwarded the packet and added the cumulative delay value given by the D-SYNC protocol. The reference time value was compared to the value given by the D-SYNC protocol for calculating accuracy results.

The results are presented in Figure 5. With shift of 8 bits, the maximum synchronization errors were 300 μs for 2 hops, 573 μs for 4 hops, and 909 μs for 6 hops. The corresponding values with no shift were 72 μs, 194 μs, and 223 μs, respectively. With the 8-bit shift, average errors were 60 μs, 76 μs, and 153 μs with the given hop amounts. The corresponding average values with no shift were 20 μs, 105 μs, and 118 μs, respectively.

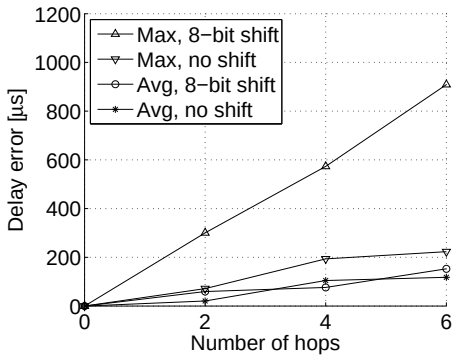

Figure 5. D-SYNC delay error as a function of hop count.

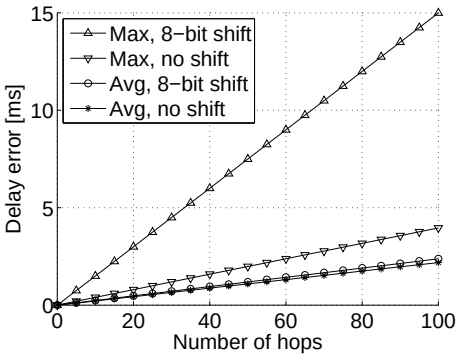

Figure 6. D-SYNC delay error results with least squares linear fitting and extrapolation to 100 hops.

The results show an almost linear increase in delay as the hop count increases. Figure 6 presents least squares linear fit of the results extrapolated to 100 hops. Using these values the synchronization error can be estimated beyond the experimented 6 hops. With shift of 8 bits, for 25, 50, 75, and 100 hops, the estimated maximum errors are 3.7 ms, 7.5 ms, 11.2 ms, and 15.0 ms, respectively. The corresponding values with no shift are 1.0 ms, 2.0 ms, 3.0 ms, and 4.0 ms. With the 8-bit shift, for 25, 50, 75, and 100 hops, the estimated average errors are 0.6 ms, 1.2 ms, 1.8 ms, and 2.4 ms, respectively. The corresponding values with no shift are 0.5 ms, 1.1 ms, 1.6 ms, and 2.2 ms.

The main error source in the experiments was the real-time clock of the used prototype platform. The usage of delay loops for fine timing produces inaccuracies when clock resolution is below 30 μs. First, this incurs error in the actual delay calculation. Furthermore, the same clock is used at the

 ISBN: 978-1-61208-144-1

radio driver. This causes errors in the packet transmission times and packet reception time registering.

The results show that the shifting has an obvious impact on the maximum synchronization errors, which accumulate when the hop count increases. However, the difference between the average synchronization errors between the 8-bit shift and no shift is much smaller. With the experimented four hops the average synchronization error was even smaller with the 8-bit shift compared to no shift. The surprisingly small difference in the average synchronization errors can be explained by the fact that with the 8-bit shift the maximum error fluctuated between positive and negative and was almost equal to both directions whilst with no shift the maximum error was almost always positive.

VI. Conclusion and Future Work

In this paper, we presented D-SYNC time synchronization protocol for multi-hop WSNs. The proposed protocol is based on multi-hop delay calculation for synchronized packets. High energy-efficiency is achieved by minimizing communication overhead. Accuracy versus overhead tradeoff can be adjusted using simple shift operations. Timing is maintained on-demand removing redundant overhead when synchronization is not needed. D-SYNC operation is fault tolerant having no single points of failure and supporting disconnected periods. The protocol complexity is low, including only timestamp value saving, delay calculation using subtraction, and shifting for timing accuracy adjustment.

We presented a mathematical analysis and tools for estimating the D-SYNC accuracy and overhead tradeoff. D-SYNC was implemented on real resource-constrained WSN nodes and its accuracy was experimented with variable hop and shift amounts. The maximum synchronization errors ranged from 72 μs to 909 μs, and the average errors from 20 μs to 153 μs when the hop count was varied from two to six and the shift amount was varied for zero to eight bits. With values extrapolated from the experimental results, the maximum error was 15 ms, which occurred with 100 hops and 8-bit shift. The experiments show that the errors have a predictable change when the hop amount increases and the error remains in the ms-scale even for large networks.

Our future work concentrates on clock drift compensation in the delay calculation. Synchronized MAC protocols rely on accurate local synchronization among neighboring nodes. This information can also be used to derive clock drift values for the D-SYNC protocol.

References

[1] I. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, "A survey on sensor networks," *Communications Magazine, IEEE*, vol. 40, no. 8, pp. 102–114, 2002.

[2] M. Kohvakka, J. Suhonen, M. Kuorilehto, V. A. Kaseva, M. Hannikainen, and T. D. Hamalainen, "Energy-efficient neighbor discovery protocol for mobile wireless sensor networks," *Elsevier Ad Hoc Networks*, 2008.

[3] A. Cerpa, J. Elson, M. Hamilton, J. Zhao, D. Estrin, and L. Girod, "Habitat monitoring: application driver for wireless communications technology," in *SIGCOMM LA '01: Workshop on Data Communication in Latin America and the Caribbean*. New York, NY, USA: ACM, 2001, pp. 20–41.

[4] D. Mills, "Internet time synchronization: the network time protocol," *Communications, IEEE Transactions on*, vol. 39, no. 10, pp. 1482–1493, Oct 1991.

[5] B. Sundararaman, U. Buy, and A. D. Kshemkalyani, "Clock synchronization for wireless sensor networks: a survey," *Ad Hoc Networks*, vol. 3, no. 3, pp. 281 – 323, 2005.

[6] Y.-C. Wu, Q. Chaudhari, and E. Serpedin, "Clock synchronization of wireless sensor networks," *Signal Processing Magazine, IEEE*, vol. 28, no. 1, pp. 124 –138, 2011.

[7] S. Ganeriwal, R. Kumar, and M. B. Srivastava, "Timing-sync protocol for sensor networks," in *SenSys '03: Proceedings of the 1st International Conference on Embedded Networked Sensor Systems*. New York, NY, USA: ACM, 2003, pp. 138–149.

[8] J. V. Greunen and J. Rabaey, "Lightweight time synchronization for sensor networks," in *WSNA '03: Proceedings of the 2nd ACM International Conference on Wireless Sensor Networks and Applications*. New York, NY, USA: ACM, 2003, pp. 11–19.

[9] S. Ping, "Delay measurement time synchronization for wireless sensor networks," Intel Research Berkeley Lab, Tech. Rep. IRB-TR-03-013, june 2003.

[10] H. Dai and R. Han, "Tsync: a lightweight bidirectional time synchronization service for wireless sensor networks," *SIGMOBILE Mob. Comput. Commun. Rev.*, vol. 8, no. 1, pp. 125–139, 2004.

[11] M. Maróti, B. Kusy, G. Simon, and A. Lédeczi, "The flooding time synchronization protocol," in *SenSys '04: Proceedings of the 2nd International Conference on Embedded Networked Sensor Systems*. New York, NY, USA: ACM, 2004, pp. 39–49.

[12] J. Elson, L. Girod, and D. Estrin, "Fine-grained network time synchronization using reference broadcasts," *SIGOPS Oper. Syst. Rev.*, vol. 36, no. SI, pp. 147–163, 2002.

[13] M. Sichitiu and C. Veerarittiphan, "Simple, accurate time synchronization for wireless sensor networks," in *WCNC '03: Proceedings of the IEEE Conference on Wireless Communications and Networking*, March 2003.

[14] W. Su and I. F. Akyildiz, "Time-diffusion synchronization protocol for wireless sensor networks," *IEEE/ACM Trans. Netw.*, vol. 13, no. 2, pp. 384–397, 2005.

[15] Q. Li and D. Rus, "Global clock synchronization in sensor networks," *IEEE Trans. Comput.*, vol. 55, no. 2, pp. 214–226, 2006.

[16] H. Kopetz and W. Schwabl, "Global time in distributed realtime systems," Technishe Univesität Wien, Tech. Rep. 15/89, 1989.

 ISBN: 978-1-61208-144-1