International Journal of Contemporary Research in Computer Science and Technology (IJCRCST) e-ISSN: 2395-5325
Volume1, Issue 5 (August '2015)

A STUDY ON DA-SYNC SCHEME FOR MOBILE UNDERWATER SENSOR NETWORKS

K.Premalatha,
M.Phil Scholar,
Department Of Computer Science,
Dr. R.A.N.M. Arts & Science College,
Erode, Tamilnadu, India.

M.Suriya,
Head and Assistant Professor,
Department Of Computer Applications,
Dr. R.A.N.M. Arts & Science College,
Erode, Tamilnadu, India.

Abstract: Time synchronization plays a critical role in distributed network systems. In this paper, we investigate the time synchronization problem in the context of underwater sensor networks (UWSNs). Although many time-synchronization protocols have been proposed for terrestrial wireless sensor networks, none of them can be directly applied to UWSNs. This is because most of these protocols do not consider long propagation delays and sensor node mobility, which are important attributes in UWSNs. In addition, UWSNs usually have high requirements in energy efficiency. To solve these new challenges, innovative time synchronization solutions are demanded. In this paper, we propose a pair wise, cross-layer, time-synchronization scheme for mobile underwater sensor networks, called DA-Sync. The scheme proposes a framework to estimate the Doppler shift caused by mobility, more precisely through accounting the impact of the skew. To refine the relative velocity estimation, and consequently to enhance the synchronization accuracy, the Kalman filter is employed. Further, the clock skew and offset are calibrated by two runs of linear regression. Simulation results show that DA-Sync outperforms the existing synchronization schemes in both accuracy and energy efficiency.

Keywords: Sensor networks, Time Synchronization, data fusion, energy efficiency

I. INTRODUCTION

Time synchronization in all networks either wired or wireless is important. It allows for successful communication between nodes on the network. It is, however, particularly vital for wireless networks. Synchronization in wireless nodes allows for a TDMA algorithm to be utilized over a multi-hop wireless network. Wireless time synchronization is used for many different purposes including location, proximity, energy efficiency, and mobility to name a few. In sensor networks when the nodes are deployed, their exact location is not known so time synchronization is used to determine their location. Also time stamped messages will be transmitted among the nodes in order to determine their relative proximity to one another. Time synchronization is used to save energy; it will allow the nodes to sleep for a given time and then awaken periodically to receive a beacon signal. Many wireless nodes are battery powered, so energy efficient protocols are necessary. Lastly, having common timing between nodes will allow for the determination of the speed of a moving node[1].

The need for synchronization is apparent. Besides its many uses like determining location, proximity, or speed, it is also needed because hardware clocks are not perfect. There are variations in oscillators, which the clocks may drift and durations of time intervals of events will not be observed the same between nodes. The concept of time and time synchronization is needed, especially in wireless networks. The definition of time synchronization does not necessarily mean that all clocks are perfectly matched across the network. This would be the strictest form of synchronization as well as the most difficult to implement. Precise clock synchronization is not always essential, so protocols from lenient to strict are available to meet one's needs [2].

There are three basic types of synchronization methods for wireless networks. The first is relative timing and is the simplest. It relies on the ordering of messages and events. The basic idea is to be able to determine if event 1 occurred before event 2. Comparing the local clocks to determine the order is all that is needed. Clock synchronization is not important. The next method is relative timing in which the network clocks are independent of each other and the nodes keep track of drift and offset. Usually a node keeps information about its drift and offset in correspondence to neighbouring nodes. The nodes have the ability to synchronize their local time with another nodes local time at any instant. Most synchronization protocols use this method. The last method is global synchronization where there is a constant global timescale throughout the network. This is obviously the most complex and the toughest to implement. Very few synchronizing

algorithms use this method particularly because this type of synchronization usually is not necessary [3].

II. TIME SYNCHRONIZATION

As the advances in technology have enabled the development of tiny, low power devices capable of performing sensing and communication tasks, sensor networks emerged and received high attention of many researchers. Sensor networks are a special type of ad-hoc networks, where wireless devices get together and spontaneously form a network without the need for any infrastructure. Because of the lack of infrastructure, such as routers in traditional networks, nodes in an ad-hoc network cooperate for communication, by forwarding each other's packets for delivery from a source to its destination. This yields a multihop communication environment [2].

Though being a special type of ad-hoc networks, sensor networks have their own characteristics, such as much limited energy sources, high density of node deployment, cheap and unreliable sensor nodes. Having these extra limiting factors for their operation, sensor networks are designed to perform complex tasks such as emergency applications, environment monitoring, information gathering in battle fields, and many other uses, connecting the physical world to the virtual world of computers. As in all distributed systems, time synchronization is an important component of a sensor network. Time synchronization in a computer network aims at providing a common time scale for local clocks of nodes in the network. Since all hardware clocks are imperfect, local clocks of nodes may drift away from each other in time, hence observed time or durations of time intervals may differ for each node in the network. However, for many applications or networking protocols, it is required that a common view of time exists and is available to all -or some- of the nodes in the network at any particular instant.

Computing devices are mostly equipped with a hardware oscillator assisted computer clock, which implements an approximation C(t) of real-time t. The angular frequency of the hardware oscillator determines the rate at which the clock runs. The rate of a perfect clock, which can be denoted as dC dt , would equal 1, however all clocks are subject to a clock drift; oscillator frequency will vary unpredictably due to various physical effects. Even though the frequency of a clock changes over time, it can be approximated with good accuracy by an oscillator with fixed frequency .

Then, for some node i in the network, we can approximate its local clock as:

$$C_i(t) = a_i t + b_i \quad (1)$$

Where ai(t) is the clock drift, and bi(t) is the offset of node i's clock.

Drift denotes the rate (frequency) of the clock, and offset is the difference in value from real time t. Using equation (1), we can compare the local clocks of two nodes in a network, say node 1 and node 2 as: $C1(t) = a12 \cdot C2(t) + b12$ (2) We call a12 the relative drift, and b12 the relative offset between the clocks of node 1 and node 2. If two clocks are perfectly synchronized, then their relative drift is 1 -meaning the clocks have the same rate- and their relative offset is zero -meaning they have the same value at that instant. Some studies in the literature use "skew" instead of "drift", defining it as the difference between clock rates . Also, the "offset" may equivalently be mentioned as "phase offset". The synchronization problem on a network of n devices corresponds to the problem of equalizing the computer clocks of different devices.

The synchronization can be either global; trying to equalize Ci(t) for all i = 1..n, or it can be local; trying to equalize Ci(t) for some set of nodes -mostly the ones that are spatially close. Equalizing just the instantaneous values (correcting the offsets) of clocks is not enough for synchronization since the clocks will drift away afterwards. Therefore a synchronization scheme should either equalize the clock rates as well as offsets, or it should repeatedly correct the offsets to keep the clocks synchronized over a time period[5][6].

a). Synchronization Methods for Sensor

Networks Time synchronization in sensor networks has attracted attention in last few years. Post-facto synchronization was a pioneering work by Elson and Estrin . They proposed that unlike in traditional synchronization schemes such as NTP, local clocks of the sensor nodes should normally run unsynchronized – in their own pace, but should synchronize whenever necessary. This way local timestamps of two nodes at the occurrence time of an event are synchronized later by extrapolating backwards to estimate the offset between clocks at a previous time (at the time of the event). This synchronization scheme has led afterwards to their RBS protocol, This work also presented a synchronization algorithm for ad hoc networks around the same time as , suggesting the timestamps in the messages to be transformed between nodes, instead of adjusting the clocks. However, his scheme used the traditional two-way message exchange for drift and offset estimation, with the assumption that the drifts are bounded by some constant p. This synchronization scheme achieves around 1ms precision with little overhead [4].

In, the authors report 2μs precision for synchronization, achieved by tight coupling between application and MAC layers in the protocol stack. The achievement in precision is basically due to the proposed architecture that enables time stamping messages at the instant the message is actually sent at the MAC layer, thereby eliminating uncertainties due to the sender. gives an overview of the time synchronization problem in sensor networks, defining the requirements, and various issues for designing synchronization algorithms for sensor networks. The authors argue that such an algorithm should be multi-modal, tiered and tunable, so that it can satisfy the diverse needs of various sensor network applications. Moreover, they suggest that the local clock of each node should be free-running, i.e. one should not adjust the local clocks. Instead, the synchronization scheme should build up a table of parameters that enables each node to convert its local clock to that of another node, and vice versa. In [10] a message ordering scheme for sensor networks is proposed. The intention is not to synchronize clocks but to be able to reason about the relative order between messages or events. The scheme 4 described in this work complies with the most relaxed version of synchronization and is not applicable for most synchronization needs in sensor networks. A recent interesting study has a more theoretical approach to the problem.

In this work, the authors consider an infinitely large sensor network, and propose an approach in which nodes collaborate to generate a waveform that carries enough synchronization information to all nodes in the network. They argue that as the number of nodes goes to infinity, optimal synchronization is possible at reasonable complexity. In the rest of this section, we present in detail the synchronization methods explicitly designed and proposed for sensor networks[4].

b) Timing-Sync Protocol for Sensor Networks (TPSN)

Ganeriwal et.al. proposed a network-wide time synchronization protocol for sensor networks, which they call Timing-Sync Protocol for Sensor Networks (TPSN). Their protocol works in two phases: "level discovery phase" and "synchronization phase". The aim of the first phase is to create a hierarchical topology in the network, where each node is assigned a level. Only one node is assigned level 0, called the root node. In the second phase, a node of level i synchronizes to a node of level i−1. At the end of the synchronization phase, all nodes are synchronized to the root node and the network-wide synchronization is achieved.

c) Discovery Phase

This phase is run once at the network deployment. First a node should be determined as the root node. This could be a sink node in the sensor network, and the sink may have a GPS receiver, in which case the algorithm will synchronize all nodes to an external time (time in physical world). If such a sink is not available, sensor nodes can periodically take over the functionality of the root node. An existing leader election algorithm might be used for this periodic root node election step. The root node is assigned level 0, and initiates the level discovery phase by broadcasting a level discovery packet. This packet contains the identity and level of the sender node. Upon receiving this packet, the neighbors of the root node assign themselves level 1. Then each level 1 node broadcasts a level discovery packet with its level and identity in the packet. Once a node is assigned a level, it discards further incoming level discovery packets. This broadcast chain goes on through the network, and the phase is completed when all nodes are assigned a level.

d) Lightweight Tree-Based Synchronization (LTS)

Lightweight Tree-based Synchronization (LTS), proposed by Greunen and Rabaey is distinguished from other work in the sense that the aim is not to maximize accuracy, but to minimize the complexity of the synchronization.

Thus the needed synchronization accuracy is assumed to be given as a constraint, and the target is to devise a synchronization algorithm with minimal complexity to achieve given precision. This approach is supported by the claim of authors that the maximum time accuracy needed in sensor networks is relatively low (within fractions of a second), and therefore it is sufficient to use a relaxed, or lightweight, synchronization scheme in sensor networks. Two LTS algorithms are proposed for multihop synchronization of the network based on pair wise 8 synchronization scheme of , as also explained . Both algorithms require nodes to synchronize to some reference point(s) such as a sink node in the sensor network. The first algorithm is a centralized algorithm, and needs a spanning tree to be constructed first.

Then pairwise synchronization is done along the $n - 1$ edges of the spanning tree. In the centralized algorithm, the reference node is the root of the spanning tree and has the responsibility of initiating a "resynchronization" as needed. Using the assumption that the clock drifts are bounded, and given the required precision, the reference node calculates the time period that a single synchronization step will be valid. Since the depth of the spanning tree affects the time to synchronize the whole network, and also the precision error at the leaf

nodes, the depth of the tree is communicated back to the root node so that it can use this information in its resynchronization time decision.

The second multihop LTS algorithm performs network-wide synchronization in a distributed fashion. Each node decides the time for its own synchronization, and a spanning tree structure is not used in this algorithm. When a node i decides that it needs to synchronize (using the desired accuracy, its distance from the reference node, and the clock drift), it sends a synchronization request to the closest reference node (by any routing mechanism available). Then, all nodes along the path from that reference node to i must be synchronized before node i can be synchronized. The advantage of this scheme is that some nodes may have less frequent events to deliver and therefore may not need frequent synchronization. Since nodes have the opportunity to decide on their own synchronization, this saves unnecessary synchronization effort of such nodes. On the other hand, letting each node decide on resynchronization may boost the number of pairwise synchronizations, since for each synchronization request, all nodes along the path from reference node to the initiator of resynchronization need to be synchronized. As the number of synchronization requests increase, the overall effect of synchronizations along these paths may be a significant waste of resources.

e) Data Collection

In the beginning, an ordinary node initiates message exchanges by sending a "Sync-Req" message to a neighboring reference node. The ordinary node records the sending time stamp T1, obtained at the MAC layer, right before the message leaves. Upon receiving the Sync-Req message, the reference node estimates and records the ordinary node's relative moving velocity v0 with Doppler shifts as specified in Section 3. Meanwhile, it marks its local time as t2. Then, after a time interval tr (waiting for the hardware sending receiving transition and avoiding collisions), the reference node sends back a "Sync-Res" message which contains t2, v0 and its sending time t3. When receiving the Sync-Res message, the ordinary node records its receiving time T4 and its relative moving velocity to the reference node, the message exchange process between the ordinary node and the reference node [6][7][8].

III. PROBLEM STUDIED

UWSNs facilitate or enable a wide range of aquatic applications, including coastal surveillance, environmental monitoring, undersea exploration, disaster prevention, and mine reconnaissance. However, due to the high attenuation of radio waves in water, UWSNs have to rely on acoustic communications . The unique characteristics of underwater acoustic communications and networking, such as low available bandwidth, long propagation delays, high error probability, and sensor node (passive or proactive) mobility (in mobile networks) pose grand challenges to almost every layer of network protocol stack and applications Most of them claim to be able to achieve high accuracy with reasonable energy expenditure. However, these algorithms cannot be directly applied to UWSNs. This is because most of these approaches assume negligible propagation delays among sensor nodes, which is not true in UWSNs. UWSNs often feature long propagation delays due to the low transmission speed of sound in water (about 1500 m/s).

In addition, for mobile UWSNs, propagation delays between nodes are time-varying because of sensor node mobility. Furthermore, acoustic transmissions are power demanding, which requires high energy efficiency. All of these features in UWSNs bring new challenges for time synchronization algorithms. Recently, some time synchronization algorithms, such as TSHL, MU-Sync , Mobi-Sync , and D-Sync have been proposed for UWSNs. In these algorithms, the issue of long propagation delays is often well addressed. However, they all ignore one issue or another. For example, TSHL assumes that nodes are fixed, which makes it not suitable for mobile networks. While MU-Sync is designed for mobile underwater networks, it is not energy efficient, and Mobi-Sync is for dense network.

a) Existing Method

Existing time-synchronization schemes use half of the round trip time to calculate one way propagation delay. Due to the node mobility, propagation delays on the way forth and back are not necessarily identical, especially when nodes move at high speed. This issue severely decreases the accuracy of most time synchronization approaches. The technique used by many of these systems is collaborative filtering (CF), which analyzes past community opinions to find correlations of similar users and items to suggest k personalized items (e.g., movies) to a querying user u. Community opinions are expressed through explicit ratings represented by the triple (user, rating, item) that represents a user providing a numeric rating for an item. Myriad applications can produce location-based ratings that embed user and/or item locations.

- Both accuracy and energy efficiency is less.
- Long propagation delay.
- The existing systems are ill-equipped to produce location aware recommendations.

- The existing system provides more expensive operations to maintain the user partitioning structure.
- The existing system does not provide spatial ratings.

b) Proposed Method

A novel time-synchronization scheme, called DA-Sync, which is a fundamental cross-layer-designed time-synchronization protocol specific for mobile UWSNs. DA-Sync provides a fundamental method to synchronize two sensor nodes, i.e., an ordinary node and a reference node. The scheme proposes a framework to estimate the doppler shift caused by mobility, more precisely through accounting the impact of the skew.
(a) A user partitioning technique that exploits user locations in a way that maximizes system scalability while not sacrificing recommendation locality

(b) A travel penalty technique that exploits item locations and avoids exhaustively processing all spatial recommendation candidates.

Advantages

- Different algorithms have different sync message (including request and response messages) packet sizes since they need to carry different amounts of information.
- High accuracy and high energy efficiency.
- Reduce the nondeterministic errors that are commonly encountered by time synchronization algorithms which rely on message exchanges.

CONCLUSION

DA-Sync is the first time synchronization algorithm to utilize the spatial correlation characteristics of underwater objects, improving the synchronization accuracy as well as the energy efficiency. The simulation results show that this new approach achieves higher accuracy with a lower message overhead. Explore other underwater mobility patterns, including one that involves vertical movement to examine the suitability of our design. Investigate the influence of errors on super node localization as well as velocity estimation, and also the influence on MAC layer activities such as packet loss and retransmission. In this project, we presented DA-Sync, a novel time synchronization scheme for mobile UWSNs. DA-Sync is a fundamental cross-layer time-synchronization scheme, which leverages Doppler effects and employs the Kalman filter to improve the accuracy of propagation-delay estimation in time synchronization. Our simulation results showed that this new approach can achieve high accuracy with low message overhead [9][10].

REFERENCES

[1] John Heidemann, Wei Ye, Jack Wills, Affan Syed, Yuan Li, "Research Challenges and Applications for Underwater Sensor Networking," *IEEE Communications Society (2006), 228-235.*

[2] Li Wang, Zhi Bin Wang, et al., "A survey of time synchronization of wireless sensor networks", Conference on Wireless, Mobile and Sensor Networks (CCWMSN07)2007.

[3] Zhengbao Li, Zhongwen Guo, Feng Hong, Lu Hon, "E2DTS: An energy efficiency distributed time synchronization algorithm for underwater acoustic mobile sensor networks," Ad Hoc Networks 11 (2013) 1372– 1380

[4] Lasassmeh, S.M., Conrad, J.M., "Time Synchronization in wireless sensor networks: a survey",IEEE SoutheastCon, 2010.

[5] M. Arulampalam, S. Maskell, N. Gordon, and T. Clapp, "A Tutorial on Particle Filters for Online Nonlinear/Non-Gaussian Bayesian Tracking," vol. 50, no. 2, pp. 174-188, Feb. 2002.

[6] A.C. Bagtzoglou and N. A., "Chaotic Behavior and Pollution Dispersion Characteristics in Engineered Tidal Embayments: A Numerical Investigation," J. Am. Water Resources Assoc., vol. 43, pp. 207-219, 2007.

[7] P. Bahl and V.N. Padmanabhan, "RADAR: An In-Building RFBased User Location and Tracking System," Proc. IEEE INFOCOM, pp. 775-784, Mar. 2000.

[8] Ming Xu, Guangzhong Liu, Daqi Zhu, Huafeng Wu, "A Cluster-Based Secure Synchronization Protocol for Underwater Wireless Sensor Networks," Hindawi Publishing Corporation International Journal of Distributed Sensor Networks Volume 2014, Article ID 398610, April 2014

[9] Y. Liu, J. Li, and M. Guizani, "Lightweight secure global time synchronization for wireless sensor networks," in *Proceedings ofthe IEEE Wireless Communications and Networking Conference:Mobile and Wireless Networks*, pp. 2312–2317, 2012.

[10] A. S. Uluagac, R. A. Beyah, and J. A. Copeland, "Secure sourcebased loose synchronization (SOBAS) for wireless sensor networks," *IEEE Transactions on Parallel and Distributed Systems*, vol. 24, no. 4, pp. 803–813, 2013.