Adaptive Synchronization of Robotic Sensor Networks

Kasım Sinan YILDIRIM*
*Department of Computer Engineering
Ege University, Universite cad. 35100 Bornova,
Izmir, Turkey
sinan.yildirim@ege.edu.tr

Önder GÜRCAN†*
†CEA, LIST, Laboratory of Model driven
engineering for embedded systems
Point Courrier 174, Gif-sur-Yvette, F-91191
France
onder.gurcan@cea.fr

1. INTRODUCTION

Time synchronization is one of the fundamental building blocks for coordinated and power-efficient operation of the networked robots. Ironically, time synchronization itself is also an energy consuming process which demands communication and information processing among the robots. This has led most research in time synchronization literature to focus on developing the less power consuming synchronization method while meeting predefined accuracy requirements.

Roughly, in the existing synchronization methods, participants collect noisy time information propagating through the network and construct a clock in software, so-called *logical clock*, whose input is the value read from the unstable built-in clock and whose output is the network-wide global time. In general, the logical clock can be constructed via collecting stable time information flooded by a special reference node or via peer-to-peer communication where nodes interact with and synchronize to their direct neighbors. Least-squares [4, 3, 9, 7] and consensus based on distributed averaging [5, 6, 10] are the common methods employed for this construction. Recently, we proposed a novel technique for the construction of the logical clock: we considered the problem of synchronization as a *search process* in which each sensor node is trying to adjust the speed of its logical clock without knowing its correct value. We employed the *Adaptive Value Trackers* (AVTs) [2] which find and track *dynamic* searched values in a given search space through successive feedbacks. We proposed a flooding based implementation [8] and a peer-to-peer implementation [1] of this approach. We observed that the synchronization performances of these approaches are similar to the existing solutions in the literature with drastically lower computation overhead, which make them more power-efficient.

However, an aspect that has been often overlooked in the recent time synchronization studies is the high dynamics of the network topology due to the mobility of the nodes. Aforementioned studies assume periodical and almost reliable communication among the nodes. Their performances are evaluated on static and non-mobile topologies. On the other hand, the time information propagating in a mobile network is subject to more noise, collisions and packet losses. Due to mobility and neighborhood changes, nodes may instantaneously start to receive time information from badly synchronized nodes. Besides, nodes may be clustered and may form dense areas where packet collisions and losses occur frequently. What is more, there may be time durations during which nodes become disconnected from the network and do not receive time information. These points are crucial for the performance of time synchronization protocols and have not been explored yet. It is still unknown whether the existing solutions are still applicable under mobile network dynamics or not: Are networked robots still be able to adapt themselves and self-adjust their logical clocks while meeting the pre-defined synchronization performance?

In this paper, we reveal by simulations that AVT synchronization is robust and preferable to existing synchronization methods under high mobile dynamics. Completely blind execution without keeping track of any neighboring node makes it particularly suitable for robotic sensor networks. As a remark, we observed that the performance of synchronization via flooding is better than the peer-to-peer approach in mobile environments.

2. METHOD

We propose a time syncronization method using AVTs. An AVT finds and tracks a *dynamic* searched value, that may change in the time due to the dynamics of the system, in a given search space as fast as possible [2].

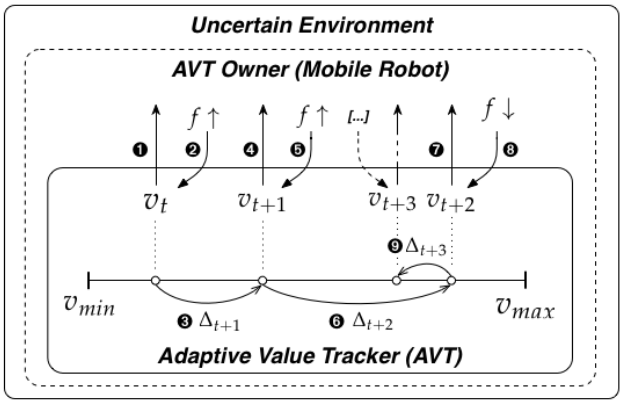

Figure 1: Interaction between a mobile robot that situated in an uncertain environment and its AVT. The AVT value tracking process starts with an initial value v_0 and includes several cycles of search iteration until $v^\star$ is reached.

The tracking is established via the successive feedbacks coming from the *owner* of the AVT (i.e. the robot) that indicate the direction that *probably* lead to the searched value. This decision is not trivial and is made by taking into account the goal of the robot.

Formally speaking, an *avt* searches and tracks a *dynamic* value $v^\star$ inside a given real interval (search space) $AVT_{ss} = [v_{min}, v_{max}] \subset \mathbb{R}$ where v_{min} is the lower boundary and v_{max} is the upper boundary for $v^\star$. At any time instant t, *avt* is able to propose a value $v_t \in AVT_{ss}$ to its owner robot that can be accessed using an action of the form $v_t = avt.value(t)$. The objective of the robot is to determine if the searched value $v^\star$ is smaller than, equal to or greater than the current proposed value v_t, without knowing the value $v^\star$. After this determination, the robot interacts with *avt* using an action of the form $avt.adjust(f_t \in \mathcal{F})$ for sending a *feedback* f_t from the feedback set $\mathcal{F} = \{\uparrow, \downarrow, \approx\}$. The feedback f_t can be about increasing v_t ($\uparrow$), decreasing v_t ($\downarrow$) or informing that v_t is good ($\approx$). A sample search with AVT is presented in Figure 1. The details for the AVT parameters can be found in [1, 8]

In time synchronization with AVTs, each node collects periodically the time information flooded by a reference node as in [8] or the time information of its neighboring nodes in a peer-to-peer manner as in [1]. Whenever a fresh time information is received, the synchronization error is calculated by considering the value of the logical clock. In flooding based approach, the whole error is added while in the peer-to-peer approach half of the error is added to the logical clock to compensate for the clock offset. After offset compensation, each sensor node tries to find the correct speed of the logical clock with respect to its built-in clock without knowing the correct value.

ALGORITHM 1. *Speed tracking code for robot u*
1: ***if*** *error>0* ***then*** $avt_u.adjust(f \uparrow)$
2: ***else if*** *error < 0* ***then*** $avt_u.adjust(f \downarrow)$
3: ***else*** $avt_u.adjust(f \approx)$

A positive error indicates that the logical clock of the robot is progressing at a slower speed than the sender's logical clock. Then, a feedback about increasing the speed of the logical clock is sent to the *avt* of that robot (Algorithm 1, line 1). In contrast, a negative error indicates that the logical clock of the robot is progressing at a faster speed and hence a feedback about decreasing its speed is sent to the *avt* (Algorithm 1, line 2). Otherwise, i.e. the error is zero, *avt* is informed that the speed of the logical clock is good (Algorithm 1, line 3), hence it remains unchanged.

3. SIMULATION RESULTS

In order to evaluate the performance of time synchronization with AVT in mobile environments, we applied the flooding based and peer-to-peer protocols to the mobile robotic sensors using simulations in our discrete event simulator. In this simulator, we implemented a probabilistic radio model (Gaussian wireless channel) and a CSMA based MAC layer. Briefly, the messages are corrupted when two or more neighboring robots are trying to transmit simultaneously and messages are lost with a small probability. For mobility, we have chosen to implement random waypoint mobility model: a robot moves on a straight line to a randomly selected position in the deployment field. Once arrived, it waits for a random amount of time before it selects a new position to move to. We implemented 1 MHz built-in clocks with constant drift clock model where the drift is uniformly distributed within the interval of ± 100 parts per million. For performance comparison, we considered two other popular time synchronization protocols: PulseSync [3] and Gradient Time Synchronization Protocol (GTSP) [6]. PulseSync offers the time information of the reference node to be propagated as fast and reliably as possible through pulses. Receiver nodes performs least-squares regression on the received time information to construct their logical clock. On the other hand, GTSP is a peer-to-peer synchronization protocol and it performs distributed averaging for time synchronization by keeping track of the neighboring nodes.

Our evaluation metrics were the instantaneous and the average instantaneous global synchronization error: the maximum error observed between arbitrary nodes. Each of our simulation runs simulated an execution of 25000 seconds. We deployed nodes to a 300x300 meter square area randomly. The transmission range of the

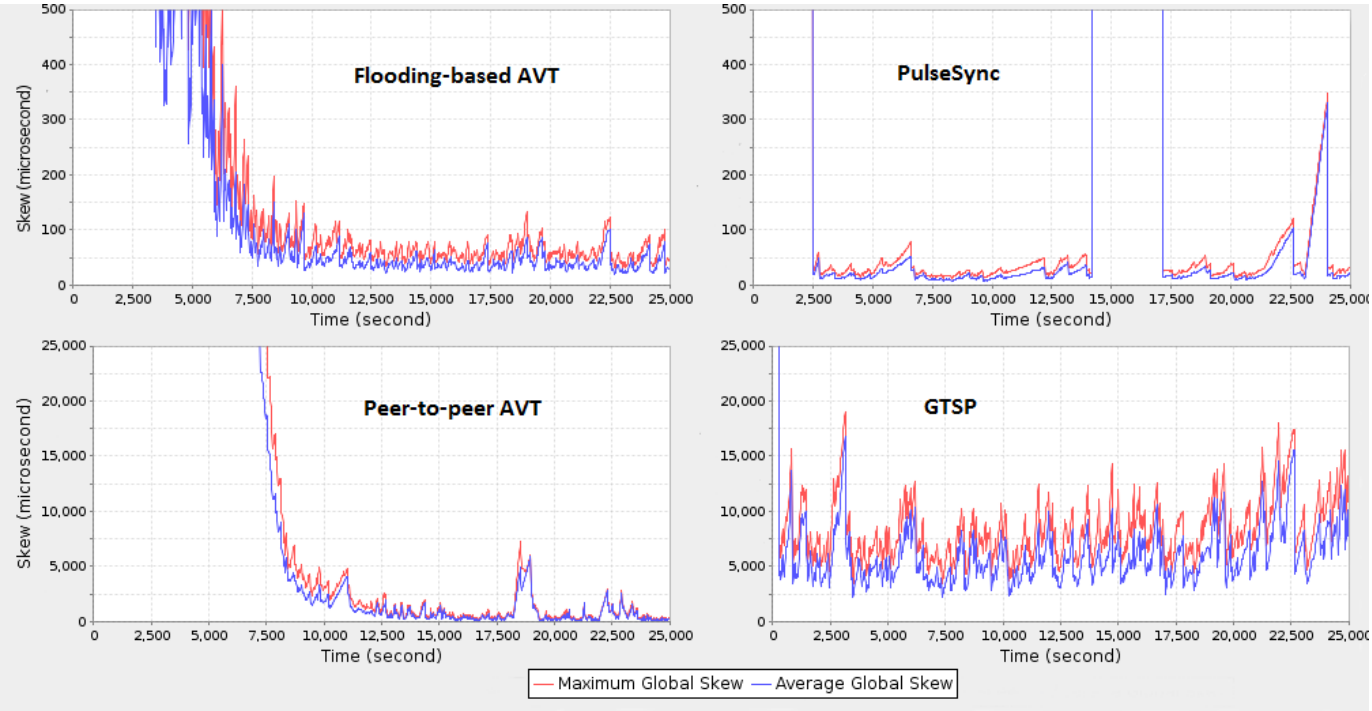

Figure 2: Global synchronization errors observed in our simulations.

nodes were adjusted to 25 meters. The evaluated protocols had an identical beacon period of 30 seconds. Since the hardware clocks of sensor nodes are reported to have a drift of ± 100 parts per million, we defined the upper bound and lower bounds of the searched logical clock speed as $v_{min} = -10^{-4}$ and $v_{max} = 10^{-4}$ for AVT.[1] For GTSP and PulseSync, the least-squares regression tables are composed of 8 entries. Finally, in GTSP, each node tracks at most 10 neighbors and if does not receive message during five beacon periods, it drops the information of that neighbor from the neighbor table.

In contrast to the unmobile networks, the time information propagated in a mobile network is subject to more noise, collisions and packet losses. First, while a robot is receiving a more correct and recent time information from one of its neighbors, it may not receive future information from that node due to mobility and neighborhood change. In this case, it may start to receive time information from a new neighbor whose logical clock is badly synchronized. Secondly, robots may be clustered and form dense areas where packet collisions and losses occur frequently. Last, there may be time durations during which a robot becomes disconnected from the network and may not receive time information. We observed that these points are crucial for the performance of time synchronization protocols.

[1]Please refer to [1, 8] for the details of the parameters chosen for the successful operation of AVT.

Figure 2 presents the global synchronization error observed during the simulations of flooding based and peer-to-peer versions of AVT synchronization, PulseSync and GTSP. In PulseSync, the time information of the reference robot is propagated as fast as possible which reduces the noise of the received time information and decreases the time required for network-wide synchronization [3]. Flooding-based time synchronization with AVT required more time to catch the performance of PulseSync, however it is more robust to packet losses and network disconnections. In PulseSync, all of the network should be connected at pulse times of the reference robot in order to receive fresh time information. During the time interval [13000,17000] in our simulations, some robots became either disconnected from the network or too many packet losses occur in the network, hence they could not receive pulses from the reference robot. This has led these robots to loose synchronization. In contrast, in AVT synchronization, the network do not require to be connected at pulse times. Instead, each robot waits until their broadcast timer to expire in order to propagate the fresh time information of the reference robot. Any robot receiving a massage carrying a higher sequence number updates its logical clock although it might have missed the pulse of the reference robot. This makes our approach more suitable for mobile robotic networks.

Considering peer-to-peer approaches, GTSP has cru-

cial disadvantages compared to peer-to-peer time synchronization with AVT. First, GTSP requires robots to keep track of their neighboring robots in order to employ distributed averaging. However, it suffers from the problem of deciding which neighbors to keep track and which ones to discard in dense areas of the network, since it is not feasible to store information for all of the neighbors. Moreover, detection of the neighborhood change is another crucial problem. For instance, as a simple strategy, when a robot does not receive messages during n broadcast periods from the neighbor it is currently keeping track of, it may delete its information from its internal tables. However, under high mobility, this strategy exhibits poor performance. From our simulations, we realized that GTSP is not suitable for mobile robotic networks and exhibits a poor performance. On the other hand, peer-to-peer AVT synchronization does not require to keep track of the information of the neighboring robots and it works in a *completely blind* manner. Robots update their time information whenever they receive a message from their neighboring robots regardless of the identity of the sender. This strategy achieved better performance than GTSP, but not better than the flooding-based approaches.

4. DISCUSSION AND CONCLUSION

This paper presented the application and the evaluation of the recent flooding-based and peer-to-peer time synchronization methods on robotic sensor networks. It has already been shown that AVT synchronization is simple, easy to implement, memory and CPU efficient, and it establishes synchronization in finite amount of time. [8, 1]. Here in this study, we observed through simulations that AVT synchronization is also robust and efficient under mobility. Hence, deducing the time synchronization problem in robotic sensor networks into a *dynamic value searching* problem is preferable to existing synchronization methods in the literature.

In general, the peer-to-peer approaches are expected to have a better performance in mobile networks. Intuitively, when the robots of the connected components synchronize to themselves, it should become easier to synchronize different connected components of the network. However, we observed that the flooding-based AVT synchronization performs better and establishes network-wide synchronization faster, compared to peer-to-peer strategy. We think that this mainly due to the random waypoint mobility model we applied in our simulation experiments. We believe that if robots apply another mobility model (where they can form group), the success of the peer-to-peer synchronization would improve. However, we leave it as a future work for now.

Besides, as another future work, we plan to explore a hybrid synchronization mechanism in which the flooding and the peer-to-peer strategies are employed together, as in [10].

5. REFERENCES

[1] Ö. Gürcan and K. S. Yildirim. Self-organizing time synchronization of wireless sensor networks with adaptive value trackers. In *Self-Adaptive and Self-Organizing Systems (SASO), 2013 IEEE Sixth Int. Conf. on*, pages 91 –100, sept. 2013.

[2] S. Lemouzy, V. Camps, and P. Glize. Principles and properties of a mas learning algorithm: A comparison with standard learning algorithms applied to implicit feedback assessment. In *Proc. of the 2011 IEEE/WIC/ACM Int. Conf. on Web Intelligence and Intelligent Agent Technology - Vol. 02*, WI-IAT '11, pages 228–235, Washington, DC, USA, 2011. IEEE Computer Society.

[3] C. Lenzen, P. Sommer, and R. Wattenhofer. Optimal clock synchronization in networks. In *Proceedings of the 7th ACM Conference on Embedded Networked Sensor Systems*, SenSys '09, pages 225–238, New York, NY, USA, 2009. ACM.

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

[5] L. Schenato and F. Fiorentin. Average timesynch: A consensus-based protocol for clock synchronization in wireless sensor networks. *Automatica*, 47(9):1878–1886, Sept. 2011.

[6] P. Sommer and R. Wattenhofer. Gradient clock synchronization in wireless sensor networks. In *Proceedings of the 2009 International Conference on Information Processing in Sensor Networks*, IPSN '09, pages 37–48, Washington, DC, USA, 2009. IEEE Computer Society.

[7] K. Yildirim and A. Kantarci. Time synchronization based on slow-flooding in wireless sensor networks. *Parallel and Distributed Systems, IEEE Transactions on*, 25(1):244–253, 2014.

[8] K. S. Yildirim and Ö. Gürcan. Efficient time synchronization in a wireless sensor network by adaptive value tracking. *Wireless Communications, IEEE Tran. on*, to appear.

[9] K. S. Yildirim and A. Kantarci. Drift estimation using pairwise slope with minimum variance in wireless sensor networks. *Ad Hoc Netw.*, 11(3):765–777, May 2013.

[10] K. S. Yildirim and A. Kantarci. External gradient time synchronization in wireless sensor networks. *Parallel and Distributed Systems, IEEE Transactions on*, 25(3):633–641, March 2014.