Evaluation of a Consensus-based Protocol for Clock Synchronization in Wireless Sensor Network

Mengheng Xue (661541223)

December 11, 2017

1 Introduction

Nowadays, with the development of sensor technology, the low cost, low power wireless sensors are widely applied in the area of wireless sensor networks such as mobile target tracking, effect time division multiple access (TDMA) scheduling and event detection [1]. It is essential for the sensor nodes to operate in a coordinated and synchronized fashion. Therefore, all these applications require a global clock synchronization, which drives all the nodes in the network to a common time [2].

In this project, a novel consensus-based protocol called Average TimeSync for synchronizing wireless sensor networks is evaluated [3]. Three main features of this protocol will be presented. Firstly, it is fully distributed, accordingly robust to node failure and new node appearance. Secondly, it compensates both clock skew difference and offset difference among nodes, which guarantees the long-time synchronization. Thirdly, it is computationally efficient. Simulations in simple and complex networks will be conducted to test the accuracy and efficiency of the evaluated algorithm.

2 System Model

In this section, the mathematical model of wireless sensor network (WSN) clocks is described. As shown in Fig. 1, We assume there are totally N nodes in the WSN, and each node i has its own local clock whose first order dynamic is obtained by

$$\tau_i(t) = \alpha_i t + \beta_i \ , \quad \forall i = 1, \cdots, N \ , \tag{1}$$

where τ_i denotes the local clock reading, α_i denotes the local clock skew, i.e., clock speed, and β_i denotes the local clock offset. Since the absolute reference time t is not given, it is impossible to obtain the parameter α_i and β_i based on Eq. (1). However, if we have local clocks of two nodes i and j, the relative local clock of node i with respect to node j can be computed by

$$\tau_j = \alpha_{ij}\tau_i + \beta_{ij} \ , \quad \forall i, j = 1, \cdots, N \ , \tag{2}$$

where $\alpha_{ij} = \frac{\alpha_j}{\alpha_i}$ and $\beta_{ij} = \beta_j - \frac{\alpha_j}{\alpha_i}\beta_i$. We can see that (2) is still linear. Since the absolute reference is not obtainable, we want to synchronize all the node clocks with respect to a

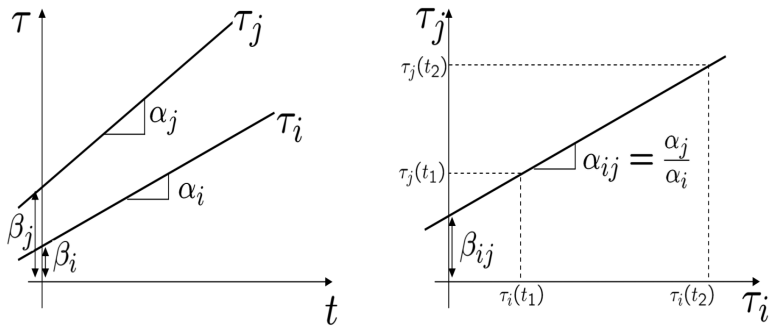

Fig. 1: Clock dynamics as function of t and relative to each other

so-called virtual reference clock τ_v, i.e.,

$$\tau_v(t) = \alpha_v t + \beta_v \ . \tag{3}$$

Each node in the WSN keeps estimating the virtual clock τ_v using their own clock τ_i by a linear function

$$\hat{\tau}_i = \hat{\alpha}_i \tau_i + \hat{o}_i \ . \tag{4}$$

Therefore, the target of this project is to find optimal pair of parameters $(\hat{\alpha}_i, \hat{o}_i)$ such that for each node in the WSN

$$\lim_{t\to\infty} \hat{\tau}_i = \tau_v(t) \ , \quad \forall i = 1, \cdots, N \ . \tag{5}$$

Thus, based on Eq. (5), all the nodes can have a global reference time which is given by $\tau_v(t)$. If we substitute Eq. (1) into Eq. (5), we can obtain that

$$\hat{\tau}_i(t) = \hat{\alpha}_i \alpha_i t + \hat{\alpha}_i \beta_i + \hat{o}_i \ . \tag{6}$$

Thus Eq. (5) can be equivalently stated as

$$\lim_{t\to\infty} \hat{\alpha}_i(t) = \frac{\alpha_v}{\alpha_i} \ , \forall i = 1, \cdots, N \ , \tag{7}$$

$$\lim_{t\to\infty} \hat{o}_i(t) = \beta_v - \frac{\alpha_v}{\alpha_i}\beta_i \ , \forall i = 1, \cdots, N \ . \tag{8}$$

Before describing the ATS protocol which is applied to update $(\hat{\alpha}_i, \hat{o}_i)$, there are some remarks worth to be mentioned. First of all, let us recall the local clock model in Eq. (1). The parameters $\alpha_i(t)$ and $\beta_i(t)$ are time-variant due to the surrounding condition and aging problem. Therefore, it requires the updating period of the protocol must be shorter than the variations of these parameters. Secondly, the virtual reference clock in Eq. (3) is actually a fictitious clock which is not fixed. In other words, the accuracy of parameters (α_v, β_v) is not important, since the ultimate goal is to make all clocks converge to one common virtual reference clock. Lastly, we assume that reading of $\tau_i(t_1)$, packet transmission and reading of $\tau_j(t_2)$ must be instantaneous, i.e., $t_1 = t_2$. Otherwise, the ATS protocol needs to be modified to handle the packet delivery delay problem.

3 Average TimeSync Protocol

The Average TimeSync protocol consists of the following three parts: the relative skew estimation, the skew compensation, and the offset compensation.

3.1 Relative Skew Estimation

This subsection introduces the part of the protocol to compute the estimate for the relative skew of each clock i with respect to its neighbours j. Assuming there are N_i nodes that can transmit packets to node i with a single hop. The neighbouring nodes j broadcast their current local time $\tau_j(t_1)$, then node i receives this packet and immediately record its own local time $\tau_i(t_1)$. As discussed in the previous section, we assume the process of the two local clock reading is instantaneous. Thus, $(\tau_i(t_1), \tau_j(t_1))$ is recorded by node i in its memory. Then after a new packet from node j arrives to node i and obtain a new pair $(\tau_i(t_2), \tau_j(t_2))$, the estimate of the relative skew α_{ij} can be obtained by

$$\eta_{ij}^{+} = \rho_\eta \eta_{ij} + (1 - \rho_\eta) \frac{\tau_j(t_2) - \tau_j(t_1)}{\tau_i(t_2) - \tau_i(t_1)} \ , \tag{9}$$

where η_{ij}^{+} denotes an updated value of variable η_{ij} and $\rho_\eta \in (0, 1)$ is a tuning parameter. If there is no measurement error and the skew is constant, then η_{ij} converges to α_{ij}, i.e.,

$$\lim_{k \to \infty} \eta_{ij}(t_k) = \alpha_{ij} \ . \tag{10}$$

The detailed proof is shown in [3]. In reality, Eq. (9) is regarded as a low pass filter where ρ_η is used to tune the trade-off between the speed of convergence ($\rho_\eta \to 0$) and the noise immunity ($\rho_\eta \to 1$) [3].

In addition, it is worth to mention that in this algorithm, the updated inter-arrival time $t_2 - t_1$ can vary, which is useful for asynchronous communication. Also, it requires very little memory, i.e., only the current N_i relative skew estimates η_{ij} and the last local clock pairs $(\tau_i(t_1), \tau_j(t_1))$, which can make this algorithm expend to large-scale networks.

3.2 Skew Compensation

This part is the core of the Average TimeSync protocol, which forces all local clocks to converge to a common virtual clock rate α_v defined in Eq. (3). The main idea is to apply a distributed consensus algorithm based only on local information exchange [4]. More specifically, each node bootstraps each other until all of them converge to a common value, i.e., till they agree upon a global value. In this protocol, each node stores its own virtual clock skew estimate $\hat{\alpha}_i$ defined in Eq.(4), and update it as soon as it receives a package from node j as follows

$$\hat{\alpha}_i^{+} = \rho_v \hat{\alpha}_i + (1 - \rho_v) \eta_{ij} \hat{\alpha}_j \tag{11}$$

where $\hat{\alpha}_i^{+}$ denotes the updated value of $\hat{\alpha}_i$ which is the estimated virtual skew of node i, and $\rho_v \in (0, 1)$ is a tuning parameter. The initial condition for all noes is set to be $\hat{\alpha}_i(0) = 1$. Then we can obtain that

$$\lim_{t \to \infty} \hat{\alpha}_i(t) \alpha_i = \alpha_v, \ \forall i = 1, \cdots, N \ . \tag{12}$$

The detailed proof is shown in [3].

It is noticeable that this algorithm does not depend on the order with which the nodes transmit, nor the exact time instants when they transmit. It shows that this algorithm is indeed applicable to the asynchronous communication for nodes with different transmission rates. The only important condition is that the graph must be sufficiently connected. Another conclusion is that even some messages are lost during transmission, Eq. (12) can be guaranteed. It implies that this algorithm is robust to link failure and packet collision.

3.3 Offset compensation

After skew compensation, each local clock will run at the same speed α_v, then Eq. (6) can be rewritten as

$$\hat{\tau}_i(t) = \alpha_v t + \frac{\alpha_v}{\alpha_i}\beta_i + \hat{o}_i \ . \tag{13}$$

Then we apply the consensus algorithm to update the virtual clock offset defined in Eq. (4) as

$$\begin{aligned} \hat{o}_i^+ &= \hat{o}_i + (1-\rho_o)(\hat{\tau}_j - \hat{\tau}_i) \\ &= \hat{o}_i + (1-\rho_o)(\hat{\alpha}_j\tau_j + \hat{o}_j - \hat{\alpha}_i\tau_i - \hat{o}_i) \end{aligned} \tag{14}$$

where $\hat{\tau}_j$ and $\hat{\tau}_i$ are computed at the same time instant, $\hat{o}_i^+$ denotes the updated value of $\hat{o}_i$ which is the estimated virtual offset of node i, and $\rho_o \in (0,1)$ is a tuning factor. Then it can be shown that

$$\lim_{t\to\infty} \hat{o}_i + \frac{\alpha_v}{\alpha_i}\beta_i = \beta_v, \ \forall i = 1, \cdots, N \ . \tag{15}$$

The detailed explanation is shown in [3].

Finally, based on the above algorithm, we can conclude that if the underlying sensor network is strongly connected, i.e., there exists at least one communication path from any node i to any other node j in the network, then

$$\lim_{t\to\infty} \hat{\tau}_i(t) = \hat{\tau}_j(t), \ \forall (i,j) \ . \tag{16}$$

4 Simulation

In this section, I have conducted several simulation cases to demonstrate the robustness, efficiency, and accuracy of the Average TimeSync algorithm.

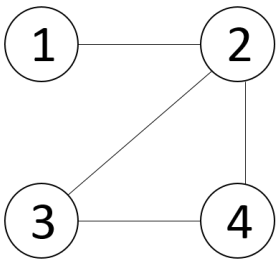

Fig. 2: Network topology of four nodes

4.1 4-Node Network Analysis

In this case, I adopt network with only four nodes whose network topology is shown in Fig. 2.The synchronization period is set as $T = 40s$. Every $0.01s$ the four nodes will update their estimate virtual time $\hat{\tau}_i$ using the Average TimeSync algorithm. In all the experiments, we set $\rho_\eta = \rho_\alpha = \rho_o = 0.6$ for convenience.

As shown in Fig. 3, I first simulate the situation that local clock of each node has different local clock skews and offsets, which is consistent with Fig. 1.

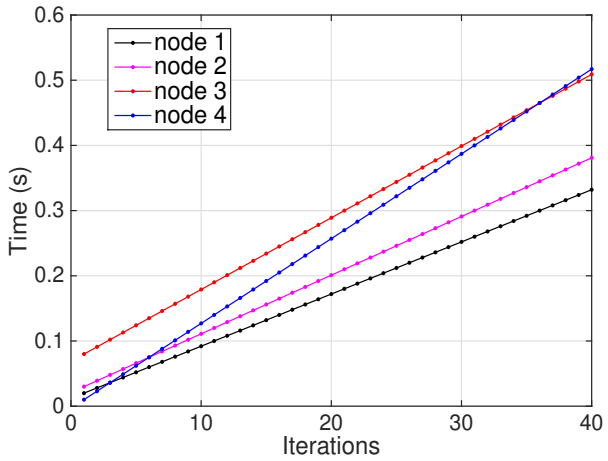

Figure 3: Local Time within Nodes

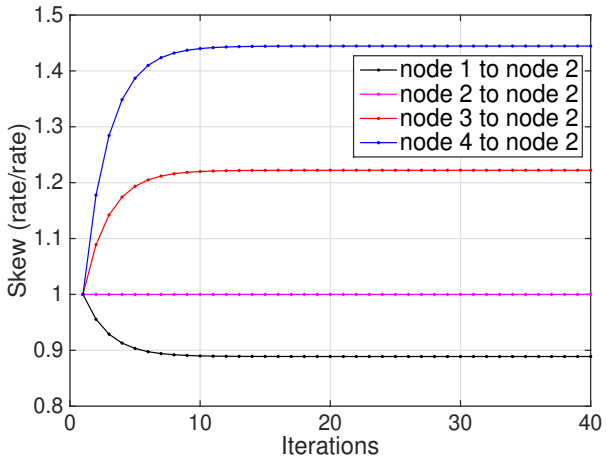

Figure 4: Relative Skew Estimation as Seen by Node 2

Then following the algorithm part to obtain the relative skew estimate, i.e., Eq. (9), we can obtain the results showing in Fig. 4. It demonstrates all the relative clock skew estimates with respect to node 2 as the iteration time increases. As we can see from Fig. 4, at the initial time, the relative clock skews of other nodes compared with node 2 changes dramatically, then with the iteration time increases, the relative skew parameter η_{ij} will converge to a constant which is α_{ij}. It also demonstrates the correctness of Eq. (10).

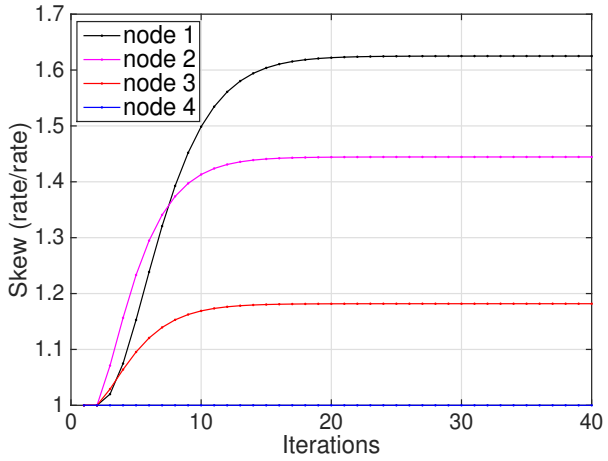

Figure 5: Virtual Skew Estimation

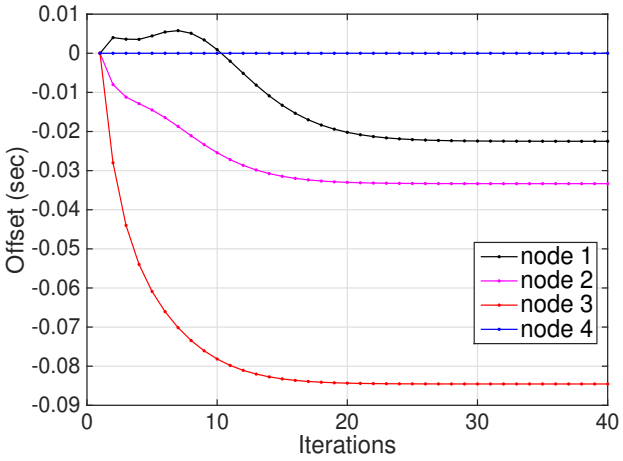

Figure 6: Virtual Offset Estimation

Then Fig. 5 illustrates the virtual skew estimation of each node in the network after skew compensation as time passes. We can observe that all nodes learn their relative skews and use this information to improve their virtual skew estimation performance. Then the virtual skew estimation of each node will converge to a common value α_v as expected. Similar performance for the virtual offset estimation after offset compensation for each node is demonstrated in Fig. 6 as expected from the theoretical analysis of the ATS algorithm.

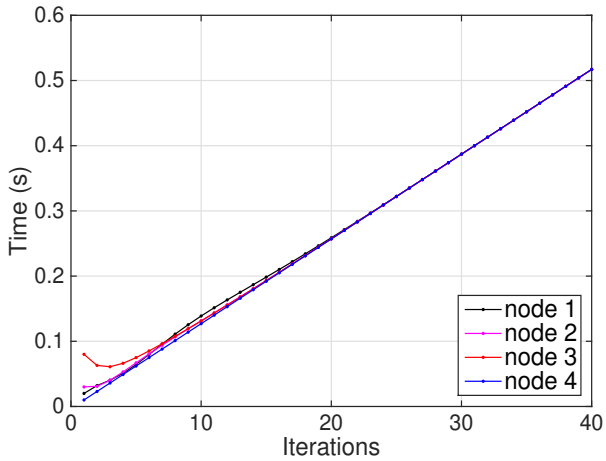

Figure 7: Virtual Time within Nodes

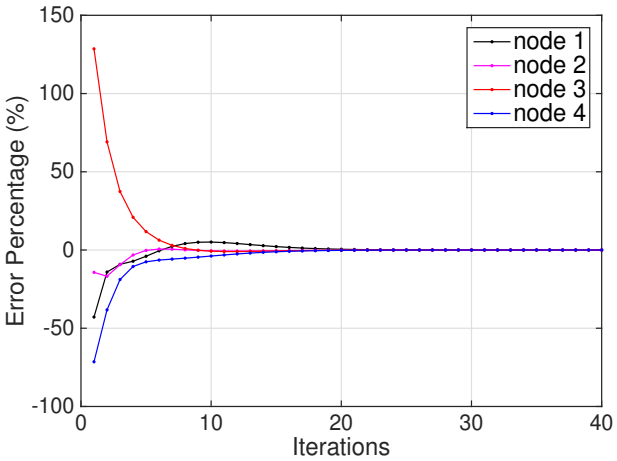

Figure 8: Error Percentage from Instantaneous Mean of Local Times

Since the virtual skew estimation and virtual offset estimation converges to α_v and β_v for each node, respectively, we can expect the virtual time estimation for each node will converge to a common one which is shown in Fig. 7. The error between each virtual clock $\hat{\tau}_i(t)$ and the nodes instantaneous mean

$$\hat{\tau}_{mean}(t) = \frac{1}{N}\sum_{i=1}^{N}\hat{\tau}_i(t) \tag{17}$$

is shown in Fig. 8. It shows clearly that the error of each virtual clock decrease to 0 simultaneously as time passes. Therefore, we can conclude that for simple network topology, the Average TimeSync Protocol is efficient and accurate for clock synchronization.

4.2 Fully Connected 4-Node Network

As shown Fig. 9, in the 4-node network, node 1, 2, 3 and 4 are fully connected. When I apply the same parameter setting as the previous subsection to conduct simulation on this fully connected network, the performances of virtual time and error of each node are shown in Fig. 10 and Fig. 11, respectively.

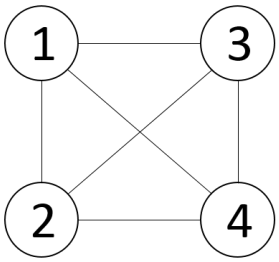

Fig. 9: Fully Connected Network Topology of Four Nodes

We can see that even with the same initial skew and offset parameters as before for each node, when the network topology changes, the final virtual time for each node will change, i.e., virtual skew and virtual offset for each node will converge to different common values. Another observation is that with a further connectivity of the network, the Average TimeSync algorithm works more efficient with less iteration ($\approx$ 15) compared with the topology in Fig. 2 ($\approx$ 20). The underlying reason is the stronger connection of each node can make their communication more efficiently to update their local clock information.

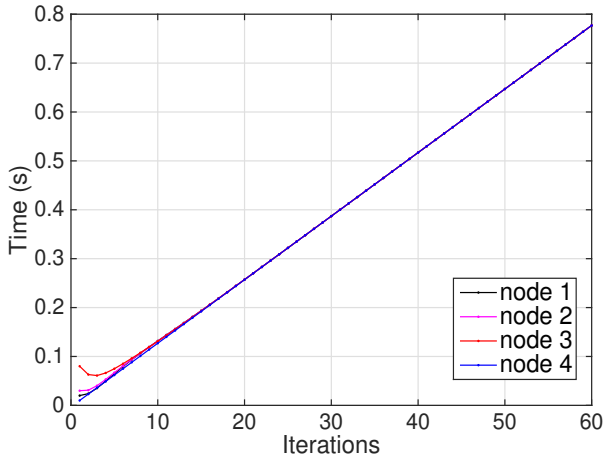

Figure 10: Virtual Time within Nodes

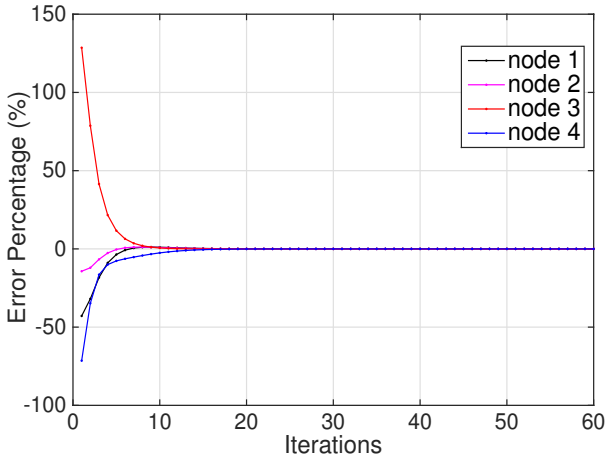

Figure 11: Error Percentage from Instantaneous Mean of Local Times

4.3 Insufficient Connected 4-Node Network

As mentioned before, the Average TimeSync algorithm works efficient and effective under one important condition that the graph is sufficiently connected. Fig. 12 shows one network topology in which node 1, 2, and 4 only have the connection to node 3, which means node 1, 2, and 4 have limited communication with each other (only can through node 3). When I conduct simulation on this network topology with the same initial local skew and offset parameters as before, the performances of virtual time and error of each node are shown in Fig. 13 and Fig. 14, respectively.

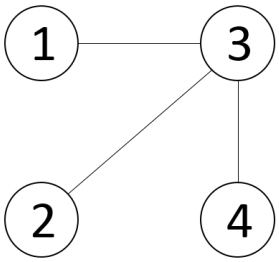

Fig. 12: Insufficiently Connected Network Topology of Four Nodes

We can see that with an insufficient connection of the topology, the Average TimeSync algorithm may not work. As shown in in Fig. 13, the virtual times of each node after applying the algorithm do not converge. The underlying reason for this performance is that since the graph is not strongly connected, the communication of each node in the network may be limited, which causes the relative skew estimation for each node may be not guaranteed to converge. There exist drift for the estimation as time passes. Accordingly, the same drifting performance happens to the virtual skew and virtual offset estimation. Therefore, the final virtual time of each node cannot be synchronized. Besides the non-convergence, the iteration time is also longer which is caused by the limited communication of each node in the network.

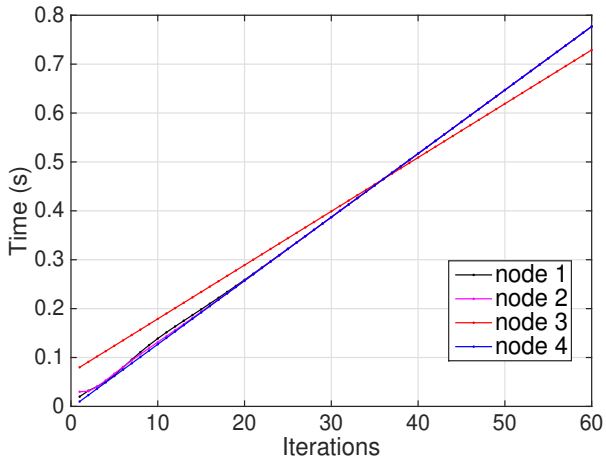

Figure 13: Virtual Time within Nodes

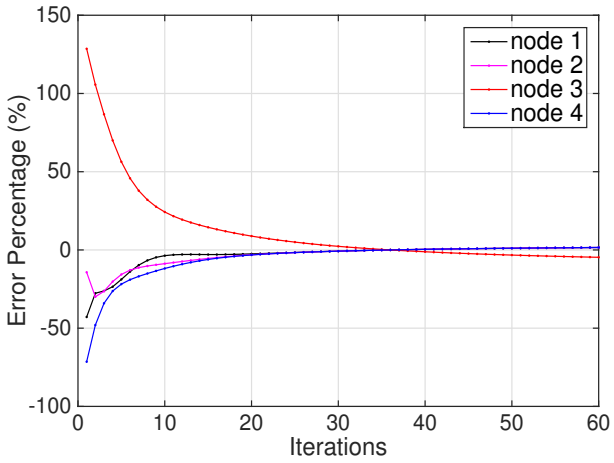

Figure 14: Error Percentage from Instantaneous Mean of Local Times

4.4 10-Node Network Analysis

In this subsection, I will conduct the similar analysis as 4-node network for a more complex network which consists of 10 nodes and the topology is shown in Fig. 15. It guarantees that the topology is sufficiently connected.

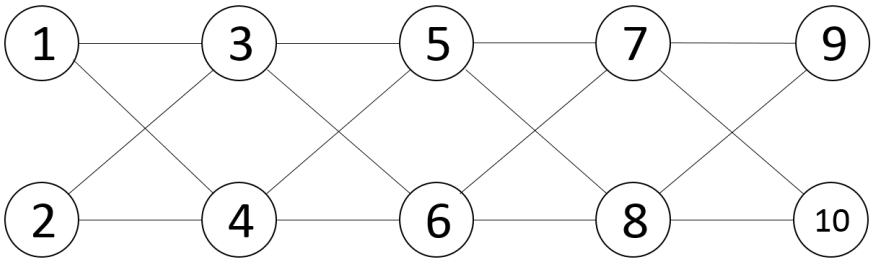

Fig. 15: Network topology of ten nodes

As shown from Fig. 16 to Fig. 21, we can see that the Average TimeSync algorithm is also efficient for large-scale networks. More iterations need to be done for full convergence of virtual time of each node (≈ 25) compared with the 4-node network (≈ 20) as expected.

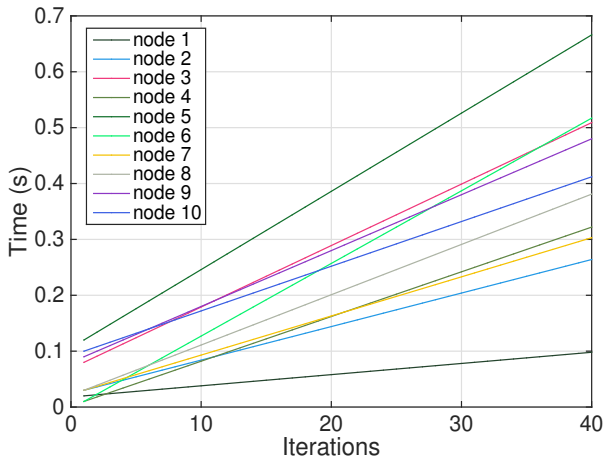

Figure 16: Local Time within Nodes

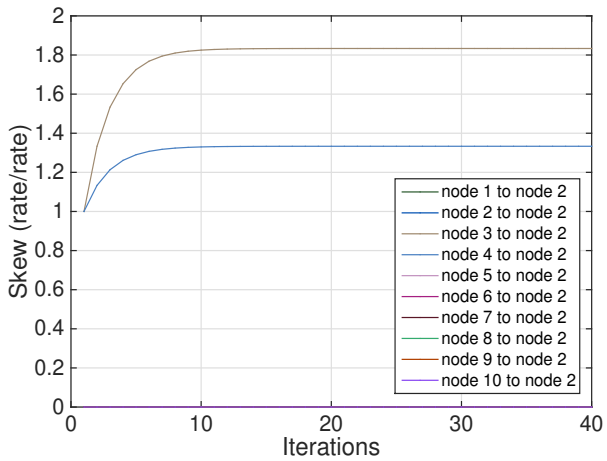

Figure 17: Relative Skew Estimation as Seen by Node 2

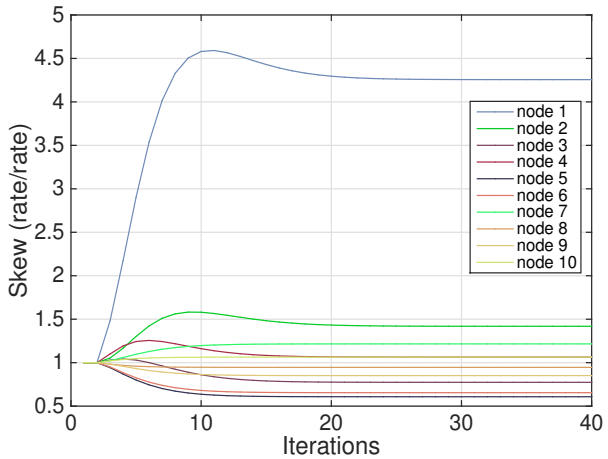

Figure 18: Virtual Skew Estimation

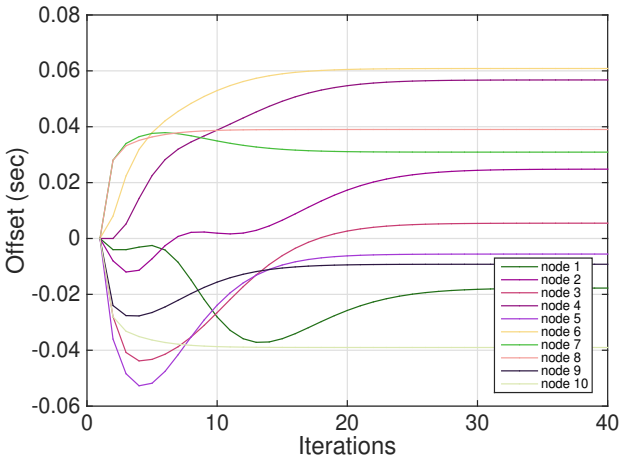

Figure 19: Virtual Offset Estimation within Nodes

5 Conclusion

In this project, I have reviewed one novel synchronization algorithm, the Average TimeSync protocol, based on a promising distributed consensus algorithm. This reviewed algorithm is fully distributed, asynchronous including skew and offset compensation. Also, it is shown to be computationally efficient. Moreover, it is robust to dynamic network topologies such as node failure and new node appearance. Simulations are conducted for 4-node and 10-node networks to test the efficiency and accuracy of the algorithm under different topology conditions. I show that the performance of this algorithm is dependent on the connectivity and complexity of the network topology. The stronger the connection of nodes in the network, the faster the synchronization can perform.

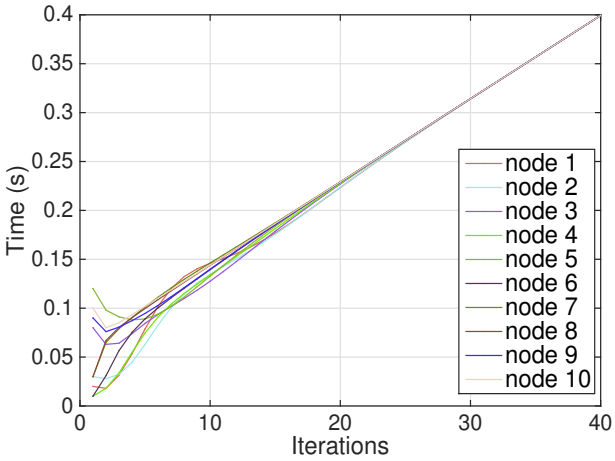

Figure 20: Virtual Time (10 nodes)

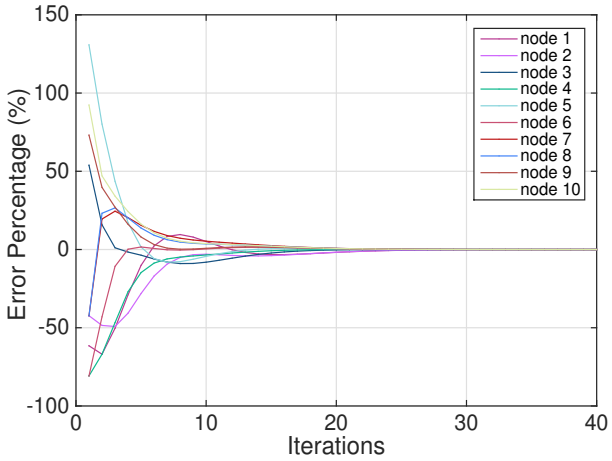

Figure 21: Error Percentage from Instantaneous Mean of Local Times (10 nodes)

References

[1] S. Ganeriwal, R. Kumar, and M. B. Srivastava, "Timing-sync protocol for sensor networks," in *Proceedings of the 1st international conference on Embedded networked sensor systems*. ACM, 2003, pp. 138–149.

[2] R. Solis, V. S. Borkar, and P. Kumar, "A new distributed time synchronization protocol for multihop wireless networks," in *IEEE Conference on Decision and Control*. IEEE, 2006, pp. 2734–2739.

[3] L. Schenato and G. Gamba, "A distributed consensus protocol for clock synchronization in wireless sensor network," in *IEEE Conference on Decision and Control*. IEEE, 2007, pp. 2289–2294.

[4] R. Olfati-Saber, J. A. Fax, and R. M. Murray, "Consensus and cooperation in networked multi-agent systems," *Proceedings of the IEEE*, vol. 95, no. 1, pp. 215–233, 2007.