CoSiNeT: A Lightweight Clock Synchronization Algorithm for Industrial IoT

Rahul Nandkumar Gore, Elena Lisova, Johan Åkerberg, Mats Björkman
Mälardalen University, Västerås, Sweden
{Rahul.Nandkumar.Gore, Elena.Lisova, Johan.Akerberg, Mats.Bjorkman}@mdh.se

***Abstract*—Recent advances in the industrial internet of things (IIoT) and cyber-physical systems drive Industry 4.0 and lead to advanced applications. The adequate performance of time-critical automation applications depends on a clock synchronization scheme used by control systems. Network packet delay variations adversely impact the clock synchronization performance. The impact is significant in industrial sites, where software and hardware resources heavily contribute to delay variations and where harsh environmental conditions interfere with communication network dynamics. While existing time synchronization methods for field IIoT end-devices, e.g., Simple Network Time Protocol (SNTP), provide adequate synchronization in good operating conditions, their performance degrades significantly with deteriorating network conditions. To overcome this issue, we propose a scalable, software-based, lightweight clock synchronization method, called CoSiNeT, for IIoT end-devices that maintains precise synchronization performance in a wide range of operating conditions. We have conducted measurements in local network deployments such as home and a university campus in order to evaluate the proposed algorithm performance. The results show that CoSiNeT matches well with SNTP and state-of-the-art method in good network conditions in terms of accuracy and precision; however, it outperforms them in degrading network scenarios. In our measurements, in fair network conditions, CoSiNeT improves synchronization performance by 56% and 73% compared to SNTP and state-of-the-art method. In the case of poor network conditions, it improves performance by 76% and 74%, respectively.**

***Keywords*—Clock Synchronization, Industrial Automation, Cyber-physical systems, Industrial internet of things, Wireless networks, SNTP, NTP, Round Trip Delay**

I. Introduction

Internet of Things (IoT) has revolutionized businesses by changing how data is utilized in order to make products and services more efficient, reliable, and profitable. While IoT is mainly used for commercial applications, Industrial Internet of Things (IIoT) is used for industrial purposes, e.g., plant operations, manufacturing, and material management. IIoT promises to achieve improved productivity, reliability, and revenues in the automation business by connecting time-constrained embedded devices to "the Internet" [1].

Fig. 1 shows a typical IoT system in an industrial plant. The bottom layer of the IoT system, 'IoT end-devices,' comprises physical assets such as robots, drives, motors, and transformers. Typically, the asset-related data is acquired by

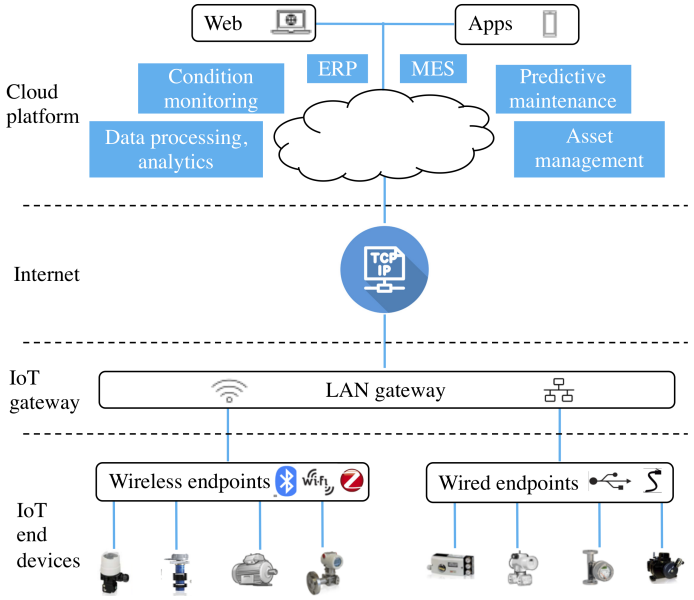

Fig. 1: Typical industrial internet of things architecture

wired and wireless local endpoints in the field and sent to a local or remote server via a local communication gateway. Such a server is implemented using industrial-level computers or cloud-based storage servers. The communication between sensor devices and the on-field data acquisition unit is termed local plant communication. Standard communication technologies used for local plant communication can be both wired or wireless, e.g., Bluetooth, Zigbee, and RFID. The data received at the server is used to run various analytical services in order to make informed decisions. The output of analytical services is used for various plant related intelligent functions such as enterprise resource planning (ERP), management enterprise systems (MES), asset management, condition monitoring, and predictive maintenance.

The advancement of cyber-physical systems and IoT is expected to enable future industrial automation evolution through the fourth evolution (Industry 4.0) [2]. Industry 4.0 has enabled the intelligent applications like cloud robotics and drones for manufacturing through immensely improved connectivity. Most of the industrial applications are based on the transfer of time over a network, so the alignment of time

or time synchronization is critical for successful adaptation of automation functions in IIoT [3].

Achieving adequate time synchronization for such applications within the new paradigm is challenging due to following reasons:

1) The process industries typically use software-based synchronization methods such as Network Time Protocol (NTP), Simple Network Time Protocol (SNTP), or vendor-specific time synchronization protocols, for which software, as well as hardware resources significantly contribute to delay variations with network part.

2) The exposure to harsh environmental conditions in the process industry brings interferes with communication network resulting in the worst delay conditions.

3) IIoT incorporates hundreds of IoT end-devices across factory sites to collect data from various subsystems. These typically are inexpensive devices that have low-cost oscillators and are low on computation and communication resources. Given these limitations, IIoT devices often become a source of additional synchronization errors, e.g., under extreme temperatures, oscillators introduce significant offset errors in the synchronization process [4]. Besides this, the lower memory and communication capabilities limit the deployment of computationally extensive and hence accurate time synchronization algorithms.

4) The resource-constrained IIoT end-devices in field network typically use lightweight SNTP rather than computation heavy NTP-based clock synchronization and achieves accuracy and precision in the order of few milliseconds. However, they cannot maintain the same performance in the worst-case network behaviors. Their synchronization performance degrades significantly with deterioration on networks.

In order to address these challenges, this paper proposes a lightweight, scalable yet accurate, and precise clock synchronization algorithm for IIoT end-devices called CoSiNeT that can maintain its performance even in poor network conditions. Typical industrial networks are heterogeneous comprising wired and wireless sub-networks at field level. Considering the network heterogeneity at field level networks, for CoSiNeT evaluation, we conducted data measurements in real wired and wireless network deployments, home and a university campus. The results demonstrated that CoSiNeT outperforms SNTP and state-of-the-art methods by showing more than 23% improved performance.

The contributions of this paper are as follows:

1) We propose a lightweight, scalable and precise clock synchronization algorithm for inexpensive, less resourceful IIoT end-devices that provides precise synchronization over a wide range of harsh environmental conditions in a factory.

2) State-of-the-art methods typically use simulated network data or data from controlled environments, e.g., laboratories. The proposed algorithm is evaluated based on the data from real wired and wireless networks with different degrees of network qualities - from good to poorly performing networks.

3) The algorithms' performance was benchmarked against widely used in-practice time synchronization protocol in IIoT end-devices from field network such as SNTP as well as state-of-the-art method SPoT [5] available in the literature. The proposed algorithm's superior performance with methods from practice and literature strengthens the new algorithm's positioning.

The paper is organized as follows: First, we present related work in Section II. In Section III, we describe the measurements from local area networks and characterize them. Section IV introduces the CoSiNeT algorithm, and Section V evaluates its performance based on the measured network data. We provide conclusions at the end.

II. Related Work

Many methods have been described in the literature addressing the clock synchronization performance.

In simple clock synchronization methods, the raw clock offset measurements can be averaged or filtered by a low-pass filter. Within these methods, the raw offset measurements are clipped to the empirical 3σ level before passing through the filter to reduce the effect of delay outliers. However, such simple approaches result in lower synchronization precision and do not guarantee the same performance in networks with different quality of communication channels. While IoT devices are typically constrained in Wireless Sensor Networks (WSNs), there have been several time synchronization methods developed in the WSN context. The flooding time synchronisation protocol is a common time-sync protocol in WSNs. In the flooding protocols [6], [7], [8] all the network nodes synchronize with the reference when the synchronization is convergent. However, the clock drift on the flooding path degrades the synchronization quality in such methods.

P. Jia et al. [9] proposed a digital-twin-enabled model-based scheme to achieve an intelligent clock synchronization in fast-changing IIoT environments. However, the success of this scheme depends on accurate clock modelling so that its behavior under dynamic operating environments is predictable.

Sridhar et al. presented the CheepSync time synchronization protocol [10] for Bluetooth Low Energy (BLE) devices. It utilizes the broadcast MAC to deal with network issues causing time synchronization errors.

Kalman filters have been used for time synchronization in order to model time offset and frequency offset and to handle missing information [11]. Such algorithms are computationally extensive and hence may not be suitable for field devices in IoT deployments.

S. K. Mani et al. [5] developed a synchronization system, including a lightweight client, a new packet exchange protocol called SPoT, and a scalable reference server. However, this method was not found efficient in dealing with packet errors and spikes in offsets introduced during bad network conditions.

III. LANs: Time data measurement and analysis

We conducted time data measurements in wired and wireless (wi-fi) networks deployed at home and university campus. We selected four different network traces that represented different network conditions ranging from good to bad.

A. Network data capture method

We used a Java-based client/server application and two computers to capture time-related network data. The Java program running on the static host or client computer sends periodic timing requests containing the start time T1, to the static server computer via a communication network. The server acknowledges the request at the local time T2. The server then sends a reply at the time T3 using the same timing packet with timestamps T2 and T3. The host receives the timing packet back at time T4. Thus, the host machine accumulates T1, T2, T3, and T4 timestamps corresponding to all periodic timing requests and responses over the same network for a particular period in a log file. The timestamps were used to calculate time offset and round trip delays (RTD) between host and server machines. The effect of java implementation on measurements was not considered in this study.

B. Trace 1: Wired home network

A typical home network was analyzed; the network included multiple devices such as laptops, TV, printer, and smartphones. The network was substantially occupied since all these devices were active during measurement. The server and client computers were found to be apart by a single hop.

Fig. 2 shows time offset and RTD values for the home network where the devices were connected using network cables. The offset between host and server device is linearly decreasing from 70ms without any variations. RTD values range from a minimum of 0.3ms to a maximum of 1.15ms. There is a small variation in RTD values with an average of 0.8ms and a standard deviation of 0.134ms.

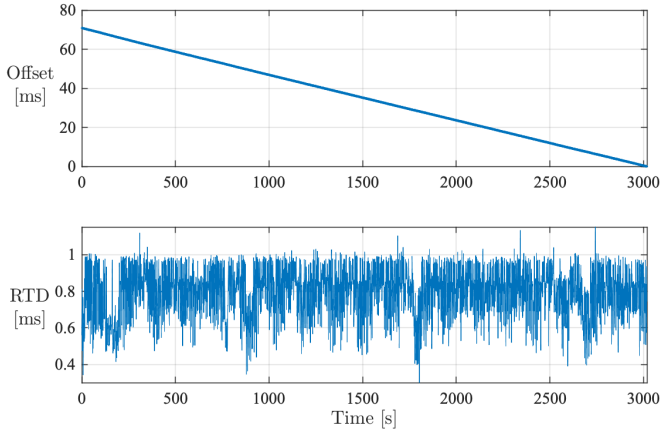

Fig. 2: Time data measurement: Wired home network

C. Trace 2: Wireless home network

When the devices were communicating using wireless links, the time offset and RTD values measured from a home network are shown in Fig. 3. The offset between host and server device shows multiple spikes due to errors (shown by red circles) related to timing requests and responses. RTD values range from a minimum of 0.5ms to a maximum of 2000ms. There are few spikes in RTD data with an average of 7ms and a standard deviation of 85ms. The offset and RTD data from the wireless home networks are higher in magnitude and variation than its wired counterpart.

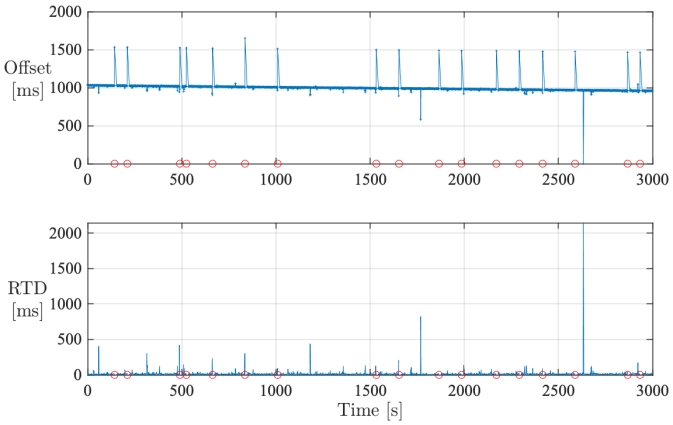

Fig. 3: Time data measurement: Wireless home network

D. Trace 3: Wired campus network

The university campus network that was analyzed included various departments, laboratories, libraries, offices, and students/staff accessing the wired and wireless (wi-fi) network for various activities. The server and client computers were found to be apart by 3 to 5 hops.

Fig. 4 shows time offset and RTD values for the campus network where the devices were connected using network cables. The offset between host and server device is linearly increasing with minor variations. RTD values range from a minimum of 0.86ms to a maximum of 24.82ms. There is a slight variation in RTD values, and RTD has an average of 1.98ms and a standard deviation of 1.70ms. The offset and RTD data from the campus network is higher in magnitude and variation than the wired home network, mainly due to the bigger network size.

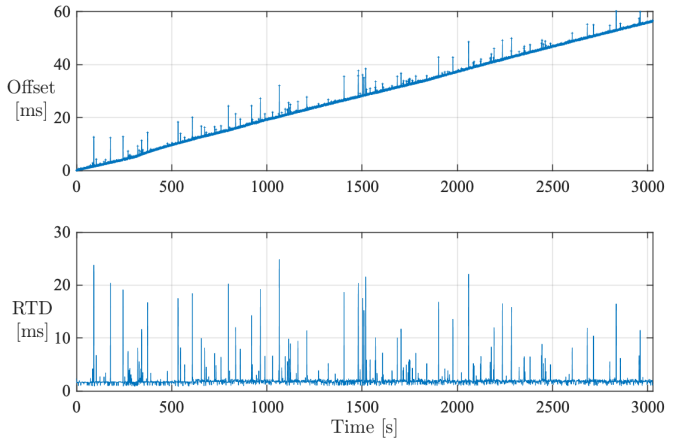

Fig. 4: Time data measurement: Wired campus network

E. Trace 4: Wireless campus network

When the devices were communicating using wireless links, the offset and RTD values measured from a campus network are captured in fig. 5. The offset between host and server device shows multiple spikes due to errors (shown by red circles) related to timing requests and responses. RTD values range from a minimum of 1ms to a maximum of 5000ms. There are few spikes in RTD data with an average of 50.44ms and a standard deviation of 205.99ms. The offset and RTD data from the wireless campus network is higher in magnitude and variation than its wired counterpart due to multiple packet exchange errors.

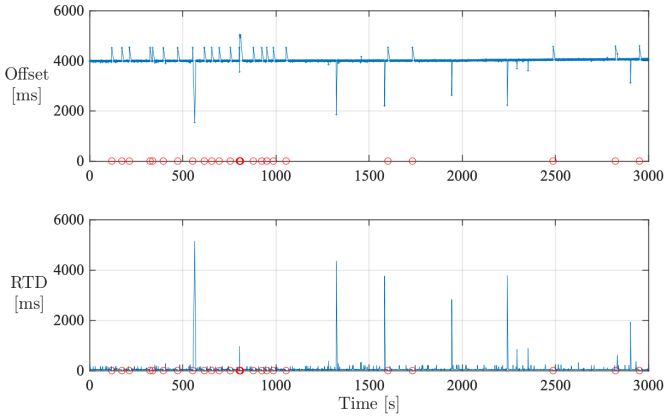

Fig. 5: Time data measurement: Wireless campus network

F. Summary: Network time data measurement

All the measured network traces by their network types, and communication medium have been summarized in Table I.

TABLE I: Measured network traces

Trace Nr	Communication medium	Network type	TRER*	Network Condition**
1	Wired	HAN	0	Good
2	Wireless	HAN	0.0057	Fair
3	Wired	CAN	0	Good
4	Wireless	CAN	0.0923	Poor

* Timing request error rate (TRER) $= \frac{\text{Timing requests with errors}}{\text{Total no. of timing requests}}$

** Good: TRER $= 0$; Fair: $0.05 >$ TRER > 0; Poor: TRER > 0.05

The timing request error rate (TRER) is 0 in the wired network for home and campus. Typically, wireless networks are non-deterministic and susceptible to various noise sources, often missing and delaying packets. Thus, wireless networks have finite TRER indicating one or more failures in exchanging timing requests and responses. We develop criteria to define network conditions (good, fair, or poor) based on that network's TRER value. Using this criterion, we define traces 1 and 3 as from a 'good' network, trace 3 as from a 'fair' network, and trace 4 as from a 'poor' network.

IV. CoSiNeT algorithm

The proposed clock synchronization algorithm CoSiNeT is designed for IIoT devices that are inexpensive and low on computational and communication resources. Hence, it is lightweight in terms of an exchange of timing messages, e.g., time requests and responses over network and algorithm complexity. Since it is a software-based synchronization approach, it can be scaled easily for large IIoT deployments. The algorithm is executed on the client device and includes periodical exchanges of timing messages with the server device. Based on these messages, it computes an offset and a RTD between client and server devices. Further, the algorithm takes raw offset and RTD values as inputs and estimates new offset by removing jitters introduced by oscillators, device stacks, switches, and other entities as described in Algorithm 1. It uses a special spike detection and removal module that compares the difference between estimated and raw offset values with a threshold to detect a spike in raw offset values due to errors and replaces it with a previous valid offset. Thus, the estimated time offset is free from most offset errors and is further used to correct the client device's clock to synchronize with the server device.

Algorithm 1 CoSiNeT algorithm

Inputs: Packet delays RTT, Measured offset Raw_Offset, Offset data size N, Window size W

Outputs: Filtered time offset Time_Offset, Filtered frequency offset Freq_Offset

for $i \leftarrow W+1:1:N$ **do**
  Calculation of minimum offset and 1-sigma offset tolerance:
  $Offsetmin \leftarrow minimum(Raw\_Offset(1) : Raw\_Offset(W))$
  $RTTmin \leftarrow minimum(RTT(1) : RTT(W))$
  $Offsetthr \leftarrow stddev(Raw\_Offset(1) : Raw\_Offset(W))$
  Time offset estimation:
  if $(Raw\_Offset(i) \geq Offsetmin)$ **then**
    $Updated\_Offset(i) \leftarrow Raw\_Offset(i) + (RTT(i) - RTTmin)/2$
  else
    $Updated\_Offset(i) \leftarrow Raw\_Offset(i) - (RTT(i) - RTTmin)/2$
  end if
  Spike detection and removal:
  if $(Updated\_Offset(i) - Raw\_Offset(i)) \geq Offsetthr)$ **then**
    $Time\_Offset(i) \leftarrow Updated\_Offset(i-1)$
  else
    $Time\_Offset(i) \leftarrow Updated\_Offset(i)$
  end if
  Frequency offset estimation:
  $Freq\_Offset \leftarrow (Time\_Offset(i) - Time\_Offset(i-1))/(measurement\_duration)$
end for

V. CoSiNeT Evaluation

We evaluated the CoSiNeT algorithm's performance using the above-mentioned four network traces of varying degrees of operating conditions. We also compared the performance of the proposed algorithm with SNTP and state-of-the-art method SPoT [5] to derive the baselines.

A. Trace 1: Wired home network

The measurements from trace 1 were fed to CoSiNeT, SNTP, and SPoT algorithms. Fig. 6 shows estimated/corrected offset outputs from different methods when evaluated using trace 1. Since this measurement trace is from a 'good' wired home network, the raw offset does not significantly vary. The performance of SNTP, SPoT, and CoSiNeT differs negligibly for this trace.

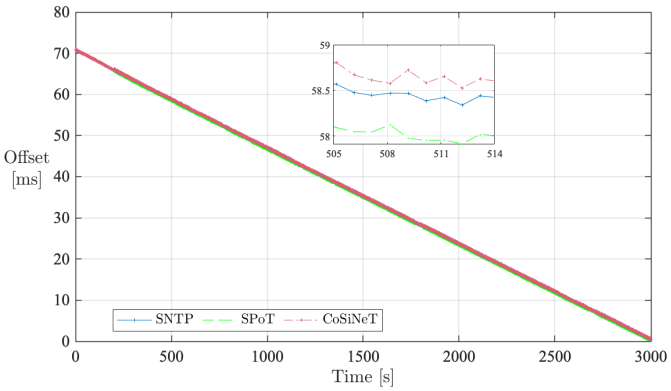

Fig. 6: Algorithm evaluation: Wired home network

B. Trace 2: Wireless home network

The new algorithm along with SNTP and SPoT were evaluated using trace 2 and the resulting estimated offsets are shown in Fig. 7. Given that this measurement trace is classified as 'fair', the raw offset has significant variations represented by spikes due to timing message failures. The magnified view shows that CoSiNeT effectively curbs the spikes in offset, whereas SNTP and SPoT are ineffective.

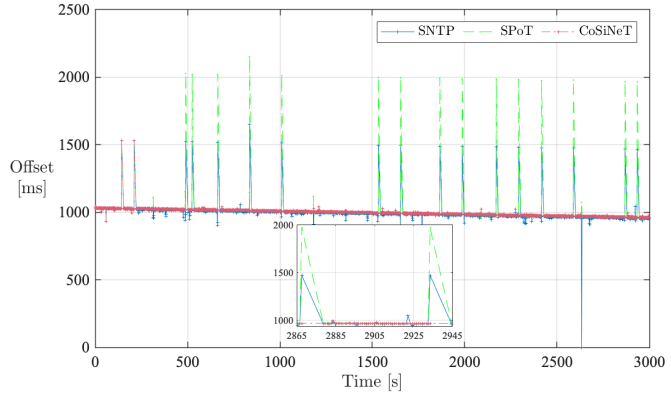

Fig. 7: Algorithm evaluation: Wireless home network

C. Trace 3: Wired campus network

Fig. 8 shows estimated/corrected offset outputs from CoSiNeT, SNTP and SPoT when evaluated using trace 3. Although this measurement trace is characterized as 'good', the raw offset does have a small variation, and this is visible through multiple short spikes. The performance of SNTP, SPoT, and CoSiNeT is identical; however, CoSiNeT performs better in dealing with spikes than others.

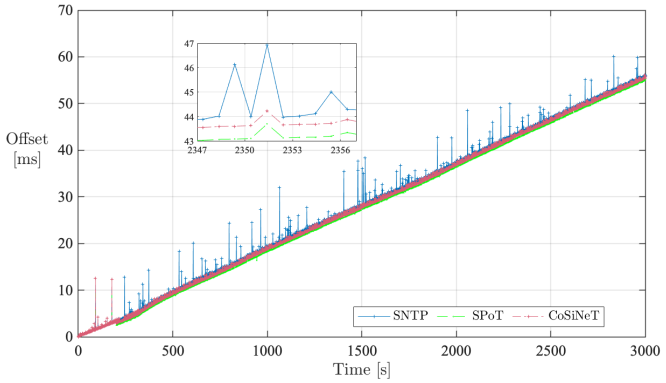

Fig. 8: Algorithm evaluation: Wired campus network

D. Trace 4: Wireless campus network

The network timestamp measurements from trace 4 were applied to inputs of CoSiNeT, SNTP and SPoT algorithm as if they were deployed in those networks. Fig. 9 shows estimated/corrected offset outputs from different methods when evaluated using trace 4. Since this measurement trace belongs to the 'poor' group, the raw offset has significant variations in multiple long spikes due to timing message failures. The magnified view shows that CoSiNeT successfully smoothens the spikes in offset, whereas SNTP and SPoT were ineffective in such cases.

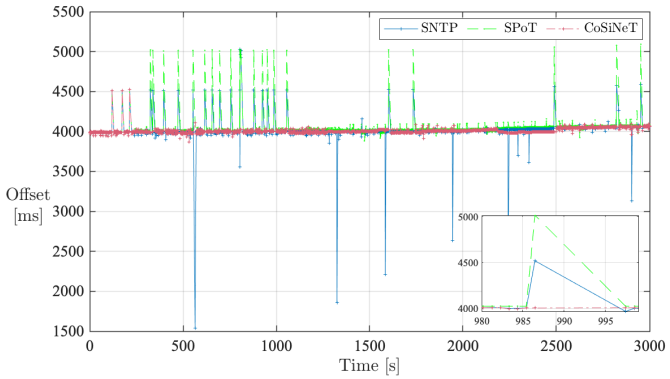

Fig. 9: Algorithm evaluation: Wireless campus network

E. Summary

The clock synchronization algorithm's essential function is reducing the variability in raw offset introduced by packet delay variation in a network. A better-performing clock synchronization algorithm is the one that curbs this variability effectively. For performance comparison of different clock synchronization methods, we use corrected offset statistics in terms of mean, standard deviation, and

TABLE II: Performance comparison of synchronization protocols using corrected offset statistics (ms)

Trace Nr	Performance parameter	SNTP	SPoT	CoSiNeT
1	**Mean**	32.731	32.327	32.903
	Std Dev	18.983	18.98	18.985
2	**Mean**	989.79	998.43	991.75
	Std Dev	48.214	77.061	20.787
3	**Mean**	30.415	29.417	29.932
	Std Dev	15.18	15.164	15.176
4	**Mean**	4017.4	4048.2	4010.2
	Std Dev	114.28	102.76	26.656

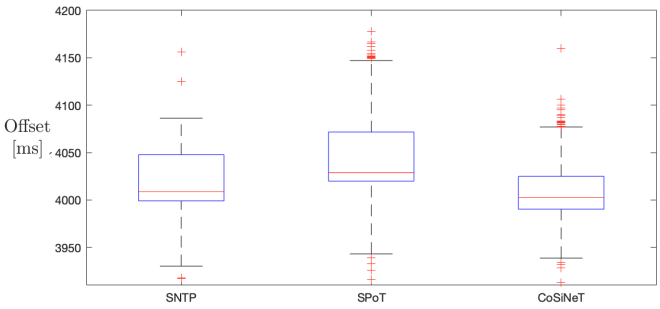

Fig. 10: Wireless campus network

median as together they effectively describe the variability in data.

Table II summarizes the performance of different synchronization methods using estimated/corrected offset statistics. The mean and standard deviation values are nearer to each other for trace 1 and 3 from 'good' networks. In trace 2, from a 'fair' network, CoSiNet shows the lowest standard deviation and, hence, variability. In 'fair' network conditions, CoSiNeT improves the synchronization performance by 56% and 73% over SNTP and SPoT. For trace 4, from a 'poor' network, CosiNeT again shows the lowest standard deviation. In 'poor' network conditions, CoSiNeT improves the synchronization performance by 76% and 74% over SNTP and SPoT. The same can be seen from Fig. 10, which shows median and interquartile range (IQR) for trace 4. The lowest IQR for CoSiNeT indicates a minimum statistical spread of corrected offset compared to others. Thus, CoSiNeT matches well with SNTP and state-of-the-art methods in good network conditions; however, it outperforms them in degrading network scenarios.

VI. Conclusion

Time and frequency alignment is essential for ensuring QoS for automation functions. It has become critical for the resource-constrained field devices due to the introduction of new technologies like drones, cloud robotics following advances in CPS and IIoT. Since existing software-based synchronization means such as SNTP and vendor-specific solutions fail to maintain the performance in field networks with poor channel qualities, we propose a scalable, lightweight clock synchronization algorithm called CoSiNeT for less resourceful and inexpensive IIoT devices. Our evaluation, based on networks of varying degrees of operating conditions, shows that the CoSiNeT algorithm in our evaluations performs better than SNTP and state-of-the-art method SPoT by 56% and 73% in a fair network environment and by 76% and 74% respectively in poor network conditions. The algorithm can successfully deal with offset changes due to step changes in RTD and multiple consecutive or random errors in timing messages due to network deterioration, leading to improved system reliability and safety.

Acknowledgments

This work has been financed by the Future Industrial Networks project, grant number 2018-02196, within the Strategic innovation program for Process industrial IT and Automation, PiiA, a joint program by Vinnova, Formas and Energimyndigheten.

References

[1] T. Lennvall, M. Gidlund, and J. Åkerberg, "Challenges when bringing iot into industrial automation," in *2017 IEEE AFRICON*, 2017, pp. 905–910.

[2] M. Wollschlaeger, T. Sauter, and J. Jasperneite, "The future of industrial communication: Automation networks in the era of the internet of things and industry 4.0," *IEEE Industrial Electronics Magazine*, vol. 11, no. 1, pp. 17–27, 2017.

[3] R. N. Gore, E. Lisova, J. Åkerberg, and M. Björkman, "Clock synchronization in future industrial networks: Applications, challenges, and directions," in *2020 AEIT International Annual Conference (AEIT)*, 2020, pp. 1–6.

[4] J. M. Castillo-Secilla, J. M. Palomares, and J. Olivares, "Temperature-aware methodology for time synchronisation protocols in wireless sensor networks," *Electronics Letters*, vol. 49, no. 7, pp. 506–508, 2013.

[5] S. K. Mani, R. Durairajan, P. Barford, and J. Sommers, "An architecture for iot clock synchronization," in *Proceedings of the 8th International Conference on the Internet of Things*, 2018, pp. 1–8.

[6] W. Masood, J. F. Schmidt, G. Brandner, and C. Bettstetter, "Disty: Dynamic stochastic time synchronization for wireless sensor networks," *IEEE Transactions on Industrial Informatics*, vol. 13, no. 3, pp. 1421–1429, 2017.

[7] K. S. Yildirim and A. Kantarci, "Time synchronization based on slow-flooding in wireless sensor networks," *IEEE Transactions on Parallel and Distributed Systems*, vol. 25, no. 1, pp. 244–253, 2014.

[8] K. S. Yıldırım, R. Carli, and L. Schenato, "Adaptive proportional–integral clock synchronization in wireless sensor networks," *IEEE Transactions on Control Systems Technology*, vol. 26, no. 2, pp. 610–623, 2018.

[9] P. Jia, X. Wang, and X. Shen, "Digital twin enabled intelligent distributed clock synchronization in industrial iot systems," *IEEE Internet of Things Journal*, pp. 1–1, 2020.

[10] S. Sridhar, P. Misra, G. S. Gill, and J. Warrior, "Cheepsync: a time synchronization service for resource constrained bluetooth le advertisers," *IEEE Communications Magazine*, vol. 54, no. 1, pp. 136–143, 2016.

[11] S. Yang, C. Xu, J. Guan, and T. Zhang, "Event-based diffusion kalman filter strategy for clock synchronization in wsns," in *2018 International Conference on Networking and Network Applications (NaNA)*, 2018, pp. 270–276.