Poster Abstract: C-Sync: The Resilient Time Synchronization Protocol

Nitin Shivaraman, Patrick Schuster, Saravanan Ramanathan, Arvind Easwaran, Sebastian Steinhorst
{nitin.shivaraman,saravanan.ramanathan}@tum-create.edu.sg,arvinde@ntu.edu.sg
{patrick.schuster,sebastian.steinhorst}@tum.de

ABSTRACT

Time synchronization is paramount for communication in Internet of Things (IoT) networks. Existing synchronization protocols in the IoT are designed to be accurate, energy-efficient and scalable with absolute trust on the time source(s). If a byzantine node becomes a time source, it can cause synchronization errors with false time, leading to system malfunctions or network crashes. In this paper, we introduce C-Sync: a clustering time synchronization protocol for decentralized IoT networks that incorporates resilience against byzantine nodes. We show that C-sync achieves a worst-case synchronization accuracy of a few tens of microseconds (μs).

KEYWORDS

Time synchronization, IoT, Fault detection, Byzantine Faults

1 INTRODUCTION

Communication is the primary source of energy expenditure among resource-constrained Internet of Things (IoT) nodes [1]. Time synchronization among the nodes aids in energy conservation of a node by keeping track of the exact time to communicate with other nodes. This allows the nodes to remain in sleep mode for the remaining duration.

In general, synchronization protocols such as flooding [1, 2], global consensus (common logical time) and compensation methods [3, 4] are designed to improve one or more combinations of accuracy, energy efficiency and scalability. Flooding mechanisms have a large quantity of messages exchanged before synchronization is complete, leading to high energy consumption. Flooding is a process of rapidly transmitting received information immediately upon message reception. Although consensus protocols are resilient to faults, they take longer to converge with less energy efficiency and have increasing drifts as the network scales. However, the above protocols are not resilient against byzantine nodes. Synchronizing based on the malicious time information could compromise the network.

In this paper, we present a synchronization mechanism called Clustering-Synchronization (C-Sync) for a decentralized IoT network. C-sync uses clustering to achieve energy

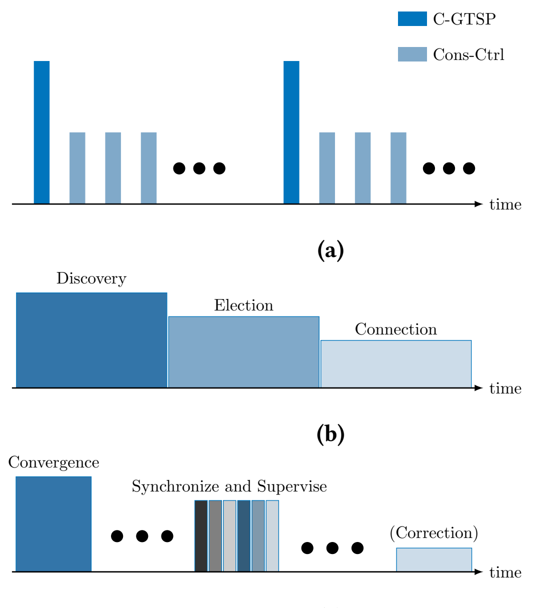

Figure 1: (a) C-sync protocol with two phases of operation. (b) Clustering phase with state transitions. (c) Convergence and consensus phase with TDMA communication.

efficiency and resilience to faulty nodes in the system. Clustering techniques are known to reduce the energy expenditure of the nodes and provide routing through representative nodes called Cluster heads (CH). Since communication in C-sync involves only CH nodes, the number of messages exchanged is reduced significantly compared to other protocols. Our preliminary results show that C-sync achieves a worst-case synchronization error of 37 μs in a multi-hop network of up to 10 hops.

2 C-SYNC

C-sync protocol operates in two phases (clustering and consensus) and adopts a slotted operation mode where synchronization is performed in fixed time slots while applications are executed in the remaining time slots as shown in Figure 1.

Clustering Phase

In the first phase of C-sync, the nodes are grouped into clusters. The clustering process is a state machine of 3 states: neighbor discovery, election and connection as indicated in

This work was financially supported in part by the Singapore National Research Foundation under its Campus for Research Excellence And Technological Enterprise (CREATE) programme.

Figure 1b. Nodes become aware of their neighboring nodes in the neighbor discovery state. Time information is added to the messages to achieve an implicit synchronization during this state similar to Gradient Time Synchronization Protocol (GTSP) [3]. GTSP achieves time synchronization by averaging neighbor time information and provides resilience for the clustering phase (C-GTSP). During the election phase, a representative node is chosen as a cluster head (CH) based on the maximum degree (number of connections) of a node. Lastly, the connection state selects a representative bridge node (nodes that belong to multiple clusters) as cluster bridge (CB) and other bridge nodes as supervision cluster bridges (SCB) to ensure inter-cluster connectivity and resilience.

Consensus Phase

Consensus Control (Cons-Ctrl) is the second phase of C-sync which repeats periodically after certain time slots to maintain the synchronization. The cluster heads exchange time information among each other to find *local centers* of their neighborhood. Local centers are CHs at a configurable number of hops from the edge of the network. The configuration limits the maximum number of hops before which the consensus is reached. Local centers bound the maximum error across the network. The time information of the local centers are propagated back to the CHs in the reverse direction after the convergence. The order of the CH receiving information is the reverse of the order of finding local centers and hence, the TDMA slots for each of the cluster is created accordingly for the *synchronize and supervise* phase. Thus, the protocol scales very well for larger network sizes without additional loss of energy and additional delay for consensus. However, given the dynamic nature of the network, some nodes may fail or new nodes may be added to the network. If there are significant changes (e.g. change in CH), clustering is triggered again to establish new clusters with the optional *correction* state of the consensus phase in Figure 1c.

Detection of Byzantine nodes. The elected cluster bridge (CB) and supervision cluster bridge (SCB) nodes act as supervision nodes to monitor the broadcast data of CH nodes to their cluster member nodes. SCB also ensures reliable and accurate information propagation during consensus. In the case of a byzantine fault, e.g. the CH sending corrupt time information, CBs and SCBs intervene and share the legitimate value of time. As CBs are connected to other clusters, they further forward the valid time information along the network path. Hence, the impact of a byzantine event can be detected and contained within a single cluster.

3 PRELIMINARY RESULTS AND CONCLUSION

In this section, we present the experimental setup and initial results of the C-sync protocol. For experiments, we use the Contiki software with a simplified communication stack to minimize software delays, deployed on the Tmote Sky [5].

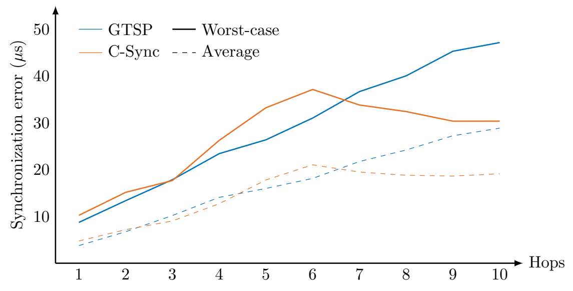

Figure 2: Comparison of synchronization error of GTSP and C-sync for a ring topology with 10 hops.

As an initial experiment, we compare the synchronization error between GTSP and C-sync (inter-cluster synchronization) over a ring network of 10 hops. GTSP uses neighbor averaging while C-sync uses convergence to find local centers and, further, consensus to synchronize. The results of the experiment are shown in Figure 2.

The synchronization error increases with hop distance for GTSP and is in agreement with the results in [3]. It is observed that the error for C-sync increases till the local center is found (6 hops) and reduces beyond. This is due to equal dispersion of information across both sides of the local center, leading to nodes having similar error as their hop distance from the center. The variation in average error is lower since the deviation of synchronization error from the average can be lower or higher. As local centers are found again as the network scales, the synchronization error is bounded at 37 μs.

In summary, the C-sync time synchronization protocol is presented in this paper and it is shown that the worst-case synchronization error is bounded. As the current results show promising potential, we aim to further improve the protocol and test its performance for byzantine faults.

REFERENCES

[1] M. Maróti, B. Kusy, G. Simon, and A. Lédeczi, "The flooding time synchronization protocol," in *Proceedings of the 2Nd International Conference on Embedded Networked Sensor Systems*, ser. SenSys '04. ACM, 2004, pp. 39–49.
[2] F. Ferrari, M. Zimmerling, L. Thiele, and O. Saukh, "Efficient network flooding and time synchronization with glossy," 05 2011, pp. 73 – 84.
[3] P. Sommer and R. Wattenhofer, "Gradient clock synchronization in wireless sensor networks," in *International Conference on Information Processing in Sensor Networks*, April 2009, pp. 37–48.
[4] L. Schenato and G. Gamba, "A distributed consensus protocol for clock synchronization in wireless sensor network," in *2007 46th IEEE Conference on Decision and Control*, Dec 2007, pp. 2289–2294.
[5] *Ultra low power IEEE 802.15.4 compliant wireless sensor module*, Moteiv Corporation, 11 2006.