Low Overhead and Application-Oriented Synchronization in Heterogeneous Internet of Things Systems

Haide Wang, The University of Western Ontario

Supervisor: Wang, Xianbin, The University of Western Ontario
A thesis submitted in partial fulfillment of the requirements for the Master of Engineering Science degree in Electrical and Computer Engineering
© Haide Wang 2022

Follow this and additional works at: https://ir.lib.uwo.ca/etd

Part of the Systems and Communications Commons

Recommended Citation
https://ir.lib.uwo.ca/etd/8973

This Dissertation/Thesis is brought to you for free and open access by Scholarship@Western. It has been accepted for inclusion in Electronic Thesis and Dissertation Repository by an authorized administrator of Scholarship@Western. For more information, please contact wlsadmin@uwo.ca.
Abstract

Recent evolution in the Internet of Things (IoT) and Cyber–physical systems (CPS) is expected to change everyday life of its users by enabling low latency and reliable communication, coordinated task execution and real time data processing among pervasive intelligence through the communication network. Precise time synchronization, as a prerequisite for a chronological ordering of information or synchronous execution, has become a vital constituent for many time-sensitive applications.

On one hand, Internet of Things (IoT) systems rely heavily on the temporal coherence among its distributed constituents during data fusion and analysis, however the existing solutions for data synchronization, do not easily tailor to resource-constrained scenarios. On the other hand, timestamping accuracy is of the utmost importance to achieve accurate time synchronization of large-scale connected systems, however the heterogeneity and complexity inherent to Internet of Things (IoT) systems lead to multi-source timestamping uncertainties and significantly deteriorate performance of traditional inflexible synchronization methods.

Therefore, this thesis aims at solving these challenges by proposing a low overhead and application-oriented synchronization in heterogeneous IoT systems. First, a low-overhead data synchronization scheme is proposed to achieve accurate temporal consistency prior to fusing the massive data collected from the distributed IoT devices. More specifically, a task period is scheduled for each sensor device to deliver the sampled data to Sink Node (SN). By comparing the difference between the predefined period and the real observed one, the clock parameters can be estimated accurately so that the misalignment of the data can be compensated accordingly. Simulation results show that the proposed scheme can enhance the data fusion accuracy to tens of microseconds with significantly reduced network overhead by up to 90%.

Next, a situation-aware hybrid time synchronization protocol is designed based on multi-source timestamping uncertainty modeling and integrated time information exchange mechanism for heterogeneous IoT systems. More specifically, the multi-source timestamping error inherent to the overall synchronization process are accurately modeled by exploring the impact of the multi-faceted operating conditions. By analyzing the real-time timestamping uncertainties, a hybrid time synchronization scheme is actualized, which can achieve optimal synchronization strategy for clock parameters estimation. In addition, an integrated time information
exchange mechanism is designed to reduce timestamping redundancy during time synchronization. Simulation results show that the proposed scheme can enhance the synchronization accuracy for heterogeneous operating scenarios.

**Keywords:** Internet of Things, Time synchronization, Data synchronization, Delay modeling, Low overhead.
The unexamined life is not worth living

— Socrates
Acknowledgments

I would like to express my deepest appreciation to my supervisor, Dr. Xianbin Wang, for his guidance, patience and encouragement in developing my research. It was his enlightening supervisions that inspired me to explore novel research areas and broadened my views in the research area. It was a wonderful and rewarding journey to learn from him.

I also feel grateful to Dr. Anestis Dounavis and Dr. Raveendra K. Rao for being my advisory committee. I really appreciate their constructive suggestions on my research.

Sincere thanks to thank my colleague, Dr. Pengyi Jia, for his countless and selfless help on my research and my paper submissions. Without the regular meetings and discussions with him, I cannot make such progress in my study.

As always, I feel very grateful to my parents and my family. I really appreciate their love and support throughout my master study and my life abroad, especially during the pandemic period. Although it’s a hard time, I know you are always there and be with me.
# Contents

Abstract ii  
Dedication iv  
Acknowledgments v  
List of Figures viii  
List of Appendices xi  

## 1 Introduction 1  
  1.1 Overview of Internet of Things Systems .......................... 1  
  1.2 Challenges of Synchronization in Time-sensitive Applications ........... 3  
  1.3 Research Objectives ............................................. 6  
  1.4 Technical Contributions of the Thesis .................................... 7  
  1.5 Thesis Outline .................................................. 8  

## 2 State of Art of Clock Synchronization in Time-critical IoT Applications 10  
  2.1 Overview of Clock Synchronization .................................. 10  
  2.2 State of art in time synchronization strategy ........................... 12  
    2.2.1 Timestamping-based Synchronization ............................... 12  
    2.2.2 Temperature-compensated Synchronization ............................ 14  
    2.2.3 Timestamp-free Synchronization ................................... 15  
  2.3 Source of Synchronization Errors .................................... 16  
  2.4 Transmission of time reference ...................................... 19  
    2.4.1 Two-Way Message Exchanges ...................................... 20  
    2.4.2 One-Way Message Dissemination .................................. 21  
    2.4.3 Receiver-receiver Synchronization .................................. 22  
  2.5 Data Synchronization ............................................. 24  
  2.6 Chapter Summary ................................................ 25  

## 3 Low-Overhead Data Synchronization 26  
  3.1 Introduction .................................................. 26  
  3.2 System Model .................................................. 29  
    3.2.1 Data Fusion and Sampling Model .................................. 29  
    3.2.2 Sensor Clock Model ........................................... 31
3.3 Data Synchronization based on Prescheduled Task
   3.3.1 Network Initialization ....................................... 32
   3.3.2 Estimation and Compensation of Clock Parameters ........... 34

3.4 Performance Evaluation ........................................... 36
   3.4.1 Data Fusion Accuracy ....................................... 36
   3.4.2 Network Overhead Analysis .................................. 38

3.5 Chapter Summary .................................................. 39

4 Situation-Aware Hybrid Time Synchronization ................. 40
   4.1 Introduction .................................................... 40
   4.2 System Model .................................................... 42
      4.2.1 Heterogeneous IoT Systems ................................ 42
      4.2.2 Timestamping Uncertainties Modeling .................... 43
   4.3 Hybrid Time Synchronization Based on multi-source Uncertainty Modeling ................. 47
      4.3.1 Time Information Exchange and Skew Estimation .......... 47
      4.3.2 Hybrid Time Synchronization .............................. 49
   4.4 Performance Evaluation ........................................ 50
      4.4.1 Uncertainty Modeling Accuracy ............................ 50
      4.4.2 Synchronization Accuracy in Heterogeneous Scenarios .... 54
   4.5 Conclusion ...................................................... 54

5 Low-Overhead and Reliable Data Synchronization ........... 56
   5.1 Introduction .................................................... 56
   5.2 System Model .................................................... 58
      5.2.1 System model .................................................. 58
      5.2.2 Data Sampling Model ....................................... 59
      5.2.3 Clock-Induced Data Misalignment ......................... 61
   5.3 Task Period Enabled Data Synchronization ..................... 63
      5.3.1 Network Initialization ....................................... 63
      5.3.2 Data Synchronization ....................................... 64
   5.4 Performance Evaluation ........................................ 67
      5.4.1 Data Fusion Accuracy ....................................... 67
      5.4.2 Network Overhead Analysis ................................ 69
   5.5 Chapter Summary ................................................ 70

6 Conclusion and Future Work ....................................... 71
   6.1 Conclusion ..................................................... 71
   6.2 Future Work .................................................... 72

Bibliography .......................................................... 74

Curriculum Vitae ......................................................... 79
List of Figures

1.1 An Internet of Things platform with physical objects deployed at the edge. . . . 2
1.2 Critical machine type communication (cMTC) use cases requiring device-level
time synchronization and an enabling solution over 5G-TSN network. [1] . . . 4

2.1 Different clock behavior. .......................................................... 11
2.2 Delays of exchanging a timestamp over a wireless link. ......................... 17
2.3 Various factors affecting packet-based synchronization. [2] ...................... 19
2.4 MAC layer timestamping with a compliant transmitter and receiver pair. [3] . 20
2.5 Two-way messaging scheme, where a node initiates a synchronization message
exchanges by sending a synchronization request to another node while also
storing its time report locally. The receiver responds back to the sender with its
time report. In (a), a typical operation of the scheme, and in (b), the timeline of
message exchanges. [3] .............................................................. 20
2.6 One-way messaging scheme, where a root node emits a synchronization mes-
sage, and all of its neighbors receive. Each receiver synchronizes to the time of
the root node. In (a), a typical operation of the scheme, and in (b), timeline of
message exchanges. [3] .............................................................. 21
2.7 Receiver-receiver messaging scheme, where a root node emits a synchroniza-
tion message, and all of its neighbors receive. Then, the receiver nodes ex-
change their local time along with the received reference time. In (a), a typical
operation of the scheme, and in (b), timeline of message exchanges. [3] . . . . 22
2.8 Drifted data timestamping due to the unsynchronized sensor clock. . . . . . 24

3.1 The overview of the data fusion architecture, where the distributed data col-
lected are associated with its sampling timestamps that are generated by the
embedded inaccurate OX-driven clocks. ....................................... 30
3.2 During the task period of node 𝑖, a series of timestamped data will be gen-
erated and recorded, which will not be transmitted to the sink node until the
prescheduled transmission point. .............................................. 31
3.3 The transmission of the data and the associated timestamps during one task
period. The first pair of communication signals is used for initial clock offset
estimation, network delay observation, and task scheduling. .................. 34
3.4 The workflow of Data Synchronization based on Timestamp Calibration and
Clock Parameters Estimation. ................................................. 35
3.5 The accumulated timestamp error associated with the data to be fused in the sink node. The timestamp accuracy can be significantly enhanced by adopting the proposed data synchronization scheme, which can help to remain the overall error to be lower than the application-oriented requirement. 37

3.6 The comparison of network overhead induced by temporal alignment processes. Because of avoiding frequent timestamp exchanges, the proposed task period-based method can achieve accurate synchronization with the least network resource consumption. 38

4.1 Various timestamping scenarios in heterogeneous IoT systems. The source of timestamping uncertainties for distributed IoT devices will be different from each other. 43

4.2 Timestamping uncertainties inherent to the synchronization process due to network uncertainties, CPU jitter, and temperature variation. 46

4.3 The timestamp exchange mechanism designed in the proposed hybrid time synchronization scheme, including initial offset estimation and three skew estimation methods. 48

4.4 The modeling accuracy of CPU-induced timestamping uncertainties before hybrid time synchronization. 51

4.5 Overall synchronization error. 52

4.6 Performance different temperatures. 52

4.7 Performance with network uncertainties. 53

5.1 The overview of the data fusion architecture, where the distributed data collected are associated with its sampling timestamps that are generated by the embedded inaccurate OX-driven clocks. 57

5.2 During the task period of node \( i \), a series of timestamped data will be generated and recorded, which will not be transmitted to the sink node until the prescheduled transmission point. 60

5.3 The illustration of the proposed TP-enabled data synchronization scheme. (a). Data transmission schedule between each IoT node and the edge device during one data synchronization cycle, including one network initialization period and a few TPs followed by data transmission. (b). The information exchange between IoT nodes and the edge device during network initialization, aiming at estimating network condition and initial clock parameters. (c). The simplified network architecture with a few edge-formed groups. The interactions between the edge device and its subordinates include bidirectional information exchange during network initialization period and unidirectional data upload for data synchronization afterward. 61

5.4 Group scheduling of Edge based data transmission. 65

5.5 The accumulated timestamp error associated with the data to be fused in the sink node. The timestamp accuracy can be significantly enhanced by adopting the proposed data synchronization scheme, which can help to remain the overall error to be lower than the application-oriented requirement. 68
5.6 The comparison of network overhead induced by temporal alignment processes. Because of avoiding frequent timestamp exchanges, the proposed task period-based method can achieve accurate synchronization with the least network resource consumption.
# List of Abbreviations

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AI</td>
<td>artificial intelligence</td>
</tr>
<tr>
<td>AMS</td>
<td>advanced manufacturing system</td>
</tr>
<tr>
<td>CPS</td>
<td>cyber-physical systems</td>
</tr>
<tr>
<td>CS</td>
<td>clock synchronization</td>
</tr>
<tr>
<td>CAP</td>
<td>contention access period</td>
</tr>
<tr>
<td>CFP</td>
<td>contention free period</td>
</tr>
<tr>
<td>D2D</td>
<td>device-to-device</td>
</tr>
<tr>
<td>DCS</td>
<td>distributed clock synchronization</td>
</tr>
<tr>
<td>DTS</td>
<td>differential timestamping mechanism</td>
</tr>
<tr>
<td>FTSP</td>
<td>flooding time synchronization protocol</td>
</tr>
<tr>
<td>ICT</td>
<td>information and communication technologies</td>
</tr>
<tr>
<td>IoT</td>
<td>Internet of Things</td>
</tr>
<tr>
<td>MAC</td>
<td>media access control</td>
</tr>
<tr>
<td>NWK</td>
<td>network</td>
</tr>
<tr>
<td>NI</td>
<td>network initialization</td>
</tr>
<tr>
<td>PAN</td>
<td>personal area network</td>
</tr>
<tr>
<td>PCA</td>
<td>principal component analysis</td>
</tr>
<tr>
<td>PHY</td>
<td>physical</td>
</tr>
<tr>
<td>PIB</td>
<td>personal area network information base</td>
</tr>
<tr>
<td>PTP</td>
<td>precision time protocol</td>
</tr>
<tr>
<td>RBS</td>
<td>reference broadcast synchronization</td>
</tr>
<tr>
<td>RSSI</td>
<td>received signal strength indicator</td>
</tr>
<tr>
<td>SD</td>
<td>superframe duration</td>
</tr>
<tr>
<td>SN</td>
<td>sink node</td>
</tr>
<tr>
<td>Abbreviation</td>
<td>Description</td>
</tr>
<tr>
<td>--------------</td>
<td>-------------</td>
</tr>
<tr>
<td>TP</td>
<td>task period</td>
</tr>
<tr>
<td>TADAS</td>
<td>task period-enabled data synchronization scheme</td>
</tr>
<tr>
<td>Wi-Fi</td>
<td>wireless fidelity</td>
</tr>
<tr>
<td>WSN</td>
<td>wireless sensor network</td>
</tr>
<tr>
<td>WSAN</td>
<td>wireless sensor and actuator network</td>
</tr>
</tbody>
</table>
Chapter 1

Introduction

1.1 Overview of Internet of Things Systems

Recent developments in connection, interactivity, and embedded intelligence technologies have made it possible for ubiquitous sensors and actuators from our daily lives to communicate with each other virtually in order to accomplish tasks specified by innovative applications and services. The Internet serves as a communication and virtualization platform for this new computing and communication paradigm, known as the Internet of Things (IoT), which connects the real world to the informationalized virtual world [4]. In order to provide a Common Operating Picture (COP) for many unique applications and services, a various number of IoT devices are creating and processing informative data in this radically immense and highly dynamic distributed system [5].

The Internet of Things (IoT) is enabling multifarious real-world applications to promote productivity and scalability. We can connect sensors and edge intelligence through the Internet of Things (IoT), digitally transform input information into data, and apply intelligent analytical tools to make accurate estimations. For instance, industrial IoT (IIoT), healthcare IoT, and citywide IoT are offering a cutting-edge improvement to support the productivity and efficiency of smart industry, smart manufacturing, and smart cities. Low-cost sensor nodes facilitate the physical world in many of these applications through their ubiquitous sensing system. Unlike our personal computers, which are connected to the Internet, embedded computers used as nodes are reduced to the size that facilitate particular purposes, utilizing the fewest compo-
Battery-powered IoT sensor nodes, for instance, are always manufactured with inexpensive components, have constrained computation and memory capabilities. Therefore, energy-efficient solutions will prolong the life of the nodes and the deployed application as a whole, while also lowering maintenance expenses. Additionally, it is clear that a traditional IoT design must contend with unprecedented challenges brought by the diversity and dynamics of the various use cases and deployment scenarios, such as large-scale deployment, heterogeneous devices, and dynamic operating scenarios. In order to support the anticipated functionalities, IoT needs customized designs. The three main issues in the design of wireless communication and networking systems are achieving resilience against heterogeneity, offering predictable quality-of-service (QoS), and maintaining connectivity that robust to the environment.
1.2 Challenges of Synchronization in Time-sensitive Applications

Internet of Things (IoT) technologies are attracting growing research interests nowadays due to the proliferation of wireless communications and intelligent devices. As the use of these devices continues to increase and as the Internet continues to develop, microsecond-level time synchronization will become more crucial for many applications. With the support of accurate time synchronization, novel kinds of applications can be created that improve user coordination and provide better cooperating experiences. While the use of various Internet-connected devices has increased dramatically over the past ten years, little focus has been placed on the unique difficulties they present for time synchronization. Understanding the effectiveness of standard approaches in achieving fine-grained time synchronization accuracy requires careful consideration of the important factors affecting time synchronization mentioned above, in particular the possibility of dramatic changes in environmental conditions and highly variable path conditions between clients and servers. To overcome the difficulties in time synchronization brought on by the influx of new device kinds, we are inspired in this thesis to develop methods, protocols, and architectures for time synchronization.

Diversity: Clock synchronization is facing various of difficulties due to the diversity of emerging device categories, such as mobile and heterogeneous IoT devices. The variety of these devices may be seen in important distinctions like: (i) processor capabilities, (ii) energy requirements, (iii) communication protocols, (iv) hardware quality of clocks, and (v) synchronization precision requirements. This diversification leads to a variety of needs for precise synchronization and frequently leads to completely different design decisions for time synchronization techniques. Let’s use the synchronization of devices with differing clock hardware, computing power, and energy budgets with a timing source to demonstrate this idea. Standard methods would treat every device the same way, taking into account the possibilities for selecting an interval for polling the timing source. However, given that each device has a unique set of capabilities and constraints, there are a number of options available, including I adjusting the polling interval for energy-constrained devices by using clock-skew tracking
reference (absolute or local domain time) via the 5G system to the devices using OTA synchronization is currently being investigated in 3GPP Release 16 [4]. Nevertheless, securing robust distribution of reference time into the network has many associated challenges. Following the existing procedures in cellular systems, the devices must compensate for the propagation delay in the reference time. Typically, this is achieved using timing advance (TA) — the frame alignment procedure as used in LTE and the 5G NR radio interface. However, TA is an approximation of device-to-BS (D2B) propagation time since each TA value corresponds to a certain range of time of arrival (TOA) values. Both the limited granularity of TA and the random perturbations in TOA due to the measurement and multipath errors, respectively, can introduce inaccuracy in reference time. Therefore, the impact of TA-related errors need careful investigation and correction to meet the device-level synchronization target.

In the rest of this article, we present the role of time synchronization in multiple cMTC use cases. We discuss the transition of 5G into industrial networks. We present the 5G NR procedures and the associated timing errors to enable device-level synchronization. We quantify the errors in propagation delay due to TA and propose an improvement. An outline of new research directions to enhance synchronization accuracy in 5G concludes the article.

algorithms, (ii) running skew tracking algorithms at the timing source for computationally constrained devices, and (iii) using standard approaches for devices with relatively higher energy budgets or higher-quality clock hardware. Heuristics to pick amongst such possibilities get complex and need substantial human adjustment under additional limitations like server load at the time source, etc. Also anticipated to expand during the next years are mobile, IoT, and other Internet-connected device kinds. By 2023, the total amount of interconnected IoT devices is expected to reach 60 billion. Numerous essential Internet services and infrastructures, including time synchronization, may face scalability issues as a result of this expanded expansion. In order to achieve the required synchronization goals, the first goal of this work is to design algorithms and protocols that can recognise diversity and adapt accordingly. A second goal is to design an architecture for IoT device synchronization that takes scalability into account as a key design criterion.
Heterogeneity: Although there has been considerable interest in Internet of Things (IoT) synchronization over the past decade, there have been relatively few studies that have concentrated on designing low overhead, situation-aware synchronization methods that solve the difficulties in the current IoT application. For example, in a heterogeneous IoT environment, maintaining synchronized networks is difficult in two aspects. 1) Device diversity: The vast majority of previous research makes the assumption that networks are homogenous, made up of the same kind of nodes outfitted with clocks that have the same performance requirements [6], [7]. IoT, on the other hand, comprises of a variety of devices that are each made for a particular function, which implies that there are various clock specifications and, consequently, various needs for timekeeping. 2) Environment heterogeneity: In addition, the deployment of IoT nodes for specified functions takes place in a variety of places. This adds environment heterogeneity into the IoT. Clocks are known to drift as a result of the effects of environmental conditions such temperature, supply voltage, vibration, and pressure [8]. Controlling the time at various network sites is difficult as a result of these considerations. Since temperature is a significant factor that adversely affects the stability of crystal oscillators, its impact on clocks should be assessed theoretically or empirically prior to the deployment of nodes on the sensing field so that appropriate synchronization solutions can be tailored for the particular environment. 3. Network architectural heterogeneity: Despite the fact that contemporary IoT designs concentrate on single hop links to set up networks with a star topology, there are still difficulties in maintaining a sizable one-hop network. First off, considering cost and deployment challenges, a small number of gateways might not be able to adequately cover the entire environment. Second, the connection between nodes may be hampered by non-line-of-sight propagation induced by moving objects due to the complexity of both interior and outdoor surroundings. To distribute reference time signals across a greater distance under certain circumstances, multi-hop connections are used. The accuracy of clock skew and offset estimates is compromised in these situations because timing inaccuracy propagates exponentially from the central reference down the dissemination channel [9]. Additionally, compared to devices with a continuous power supply, resource-constrained IoT systems may have smaller energy budgets. Low duty-cycle methods used by networks to save energy can lead to connection and topological changes over time. As IoT network architectures move toward being centralised
and having a control unit, such as an access point or gateway, synchronization methods must overcome various obstacles in order to support the new network design. As a result, achieving precise but effective network synchronization for these heterogeneities in IoT is a significant and unsolved issue.

**Resource Constrained network:** The observation of clock inaccuracy relies heavily on the explicit interactions and timestamps exchange among the involved IoT devices. Therefore, it is challenging to choose an appropriate instance to start the clock calibration process, especially in resource-constrained networks, because the characteristics in distributed clocks cannot be fully estimated, leaving the unknown and non-deterministic offsets throughout the network unpredicted by the conventional clock synchronization techniques. Inappropriate synchronization frequencies for the dispersed industrial equipment will unavoidably result from the improper network-wide timestamp exchange. Some of the difficulties might be related to the excessive use of network resources that results from excessive clock calibration as well as the erroneous and crooked local data creation brought on by insufficient synchronization. Additionally, as the network size increases, the time-consuming interactions involved in clock offset calculation will take up a increasing portion of the scarce communication resources, making it more challenging that resources will be available for other critical heterogeneous applications.

### 1.3 Research Objectives

The research objectives of this thesis are mitigating the aforementioned synchronization challenges in Time-sensitive IoT applications, namely, low overhead data synchronization and situation-aware clock synchronization.

**Low overhead data synchronization:** Time-sensitive applications in Internet of Things (IoT) systems rely heavily on the temporal coherence among its distributed constituents during data fusion and analysis. Due to the ubiquitous deployment of IoT devices, a large amount of heterogeneous data will be continuously sampled and delivered from the distributed devices to support centralized data processing. The inconsistent clock output inherent to the unstable and
heterogeneous clock oscillator embedded at each IoT device will inevitably lead to inaccurate data processing and deteriorated overall performance. Traditional data synchronization always achieve low efficiency by dedicated clock synchronization protocols. Therefore, our objective is to design lightweight data synchronization protocols without clock synchronization to reduce network overhead

**Situation-aware clock synchronization:** The expeditious evolution of Internet of Things (IoT) and wireless communication technologies is expected to further boost the proliferation of connected devices for the foreseeable future. Heterogeneity is becoming one of the fundamental characteristics of the future IoT systems, due to the diverse communication standards [10], various processing capabilities, and different operating conditions of the IoT devices involved [11]. With the increasingly stringent and multifarious requirements, maintaining an accurate temporal consistency among the involved heterogeneous IoT devices becomes indispensable in achieving timely information exchange for collaboration. Therefore, our objective is to design a situation-aware hybrid time synchronization protocol is designed based on multi-source timestamping uncertainty modeling and integrated time information exchange mechanism for heterogeneous IoT systems.

1.4 Technical Contributions of the Thesis

The main contributions of this thesis are summarized below:

- A low overhead data synchronization scheme is proposed to tackle the challenges inherent to the traditional data synchronization methods. The main component of this chapter can be summarized in two aspects. First, a prescheduled task period is proposed to serve as a time reference. By comparing the task period and the real observed information, the temporal information of the collected data could be compensated for only at the sink nodes according to the estimated clock parameters. Second, contrary to traditional approaches that necessitate frequent synchronization to adjust the clocks, the proposed data synchronization can be achieved more efficiently. There is no extra communication
overhead introduced during the clock compensation due to the reason that additional transmission of dedicated timestamps is eliminated.

- A novel hybrid time synchronization protocol enabled by multi-source uncertainty modeling and integrated time information exchange is designed to achieve accurate time synchronization in heterogeneous IoT systems. Specifically, the contributions of this thesis can be summarized in three aspects. First, the multi-source timestamping uncertainties inherent to the overall synchronization process due to the non-deterministic communication process, temperature variation, and CPU load are accurately modeled to analyze their corresponding impact on the timestamping accuracy. Moreover, a hybrid time synchronization scheme is proposed to achieve adaptive and situation-aware clock calibration in terms of operating conditions. The optimal combination of different skew estimation methods is achieved according to the real-time multi-source uncertainties observed at each IoT device. In addition, an integrated time information exchange mechanism is designed by merging redundant timestamps to reduce the network resource consumption during the hybrid time synchronization.

1.5 Thesis Outline

The rest of the thesis is organized as follows:

In chapter 2, a comprehensive study of clock synchronization in industrial IoT systems is conducted. An overview of the technical background of clock synchronization is given first, followed by the current challenges of clock synchronization in IoT systems, including the cost efficient and low overhead synchronization protocol design as well as the clock uncertainty induced from the operating environment.

In chapter 3, a novel prescheduled task period-based data synchronization method have been proposed to achieve accurate temporal alignment among the distributed data before data fusion and processing in supporting time-sensitive IoT applications. More specifically, by comparing the predefined task period of each sensor node, the real observed information at SN will be used for clock skew and offset estimation. By compensating the clock inaccuracy only through comparing the task periods, temporal alignment with extremely low network overhead
is achieved. Simulation results demonstrated that the proposed method can help to significantly enhance the data fusion performance with the presence of inaccurate clocks. Moreover, the induced network overhead is dramatically reduced during clock parameter estimation, as compared to the traditional timestamp-based synchronization approaches.

In chapter 4, a situation-aware hybrid time synchronization protocol is designed based on multi-source timestamping uncertainty modeling and integrated time information exchange for heterogeneous IoT systems. More specifically, the multi-faceted operating conditions inherent to the overall synchronization process are accurately modeled to explore their impact on the timestamping accuracy. By analyzing the real-time timestamping uncertainties, a hybrid time synchronization scheme is achieved, which can help achieve optimal synchronization strategy for clock parameters estimation. In addition, an integrated time information exchange mechanism is designed to reduce timestamping redundancy during time synchronization. Simulation results show that the proposed scheme can achieve accurate timestamping error modeling and enhance the synchronization accuracy for heterogeneous operating scenarios.

In chapter 5, a pre-scheduled task period enabled data synchronization scheme is proposed to achieve accurate, efficient, and reliable synchronization for distributed data in edge-enabled IoT systems. The main contributions of this chapter are threefold, including the pre-scheduling of task periods for IoT devices in achieving timestamp-efficient data synchronization, the design of the differential timestamping mechanism for efficiency enhancement, as well as two unreliability detection schemes enabled by scheduled data transmission and timestamp verification.

Lastly, all the contributions that presented in the previous chapters are concluded in chapter 6. The plan for the future research is discussed in this chapter as well.
Chapter 2

State of Art of Clock Synchronization in Time-critical IoT Applications

2.1 Overview of Clock Synchronization

The coordination of distributed systems rely heavily on accurate clock synchronization. Each node in a distributed system, such as wireless sensor or IoT devices, has its own clock to keep its own notion of time. However, the different nodes network collaborate to complete tasks such as data exchange or data fusion, where the data will not be meaningful if they are not synchronized with the same time reference. The performance of these distributed applications is determined by the accuracy to which the nodes are synchronized. As a result, the network needs be synced in order to agree on a global time, also known as coordinated universal time (UTC). The purpose of synchronization is to determine the relative clock skew and offset between two nodes using a simple message exchange method.

The clock model between two network devices can be represented as follows:

\[ C_i(t) = \alpha_i(t)C(t) + \beta_i(t) \]  

where \( C_i(t) \) and \( C(t) \) are, respectively, the local of node \( i \) and the universal time at time \( t \), \( \alpha_i \) is the relative clock skew, and \( \beta_i \) is the relative clock offset between node \( i \) and the reference node. The initial clock offset \( \beta_i(t) \) is measured as the time difference from the universal time.
\[ \beta_i(t) = C_i(t) - C(t). \] Consequently, As a result, the overall objective of the synchronization process is to remove any relative clock skew and offset between every pair of nodes in the system, and to guarantee that the synchronized condition can be maintained for as long as possible.

Each node’s clocks should ideally be ticking at the same rate. In real scenario, a node has a physical oscillator to provide local time signals. Typically, crystal oscillators or digitally controlled oscillators (DCO) are used for this. A deviation in frequency between two clocks can be caused by external variables that affect oscillators, such as supply voltage, temperature, and atmospheric pressure. Parts-per-million (ppm) metric is defined with a frequency tolerance range w.r.t. the nominal frequency to measure the accuracy of oscillators. In the study that follows, we can first assume that the scenario in which environmental influences are time-invariant. A clock \( C_i \) on node \( i \)’s instantaneous output frequency may be described as:
\[ f_{c_i}(t) = f_0 + \delta_{c_i}(t) \]  

where \( \delta_{c_i} \) is the oscillator frequency’s inaccuracy of clock \( C_i \) and \( f_0 \) is the clock’s nominal frequency. A normally distributed variable is used to model the oscillator’s inaccuracy. As a result, their clocks are drifting at different rates. To avoid clocks from drifting apart and losing synchronization, it is crucial to determine the network’s relative clock skew and compensate it.

The terms accuracy and stability are often used in describing the quality of a clock. We propose to define each term and illustrate the difference between them.

The accuracy of an oscillator represents how much the frequency of the oscillator can deviate from its nominal value and it is expressed in parts per million (ppm). Stability is the measurement of random uncertainty of a clock value, expressed by the standard deviation or by a multiple of the standard deviation.

2.2 State of art in time synchronization strategy

We begin by first reviewing recent research on time synchronization for IoT. Synchronization is essential in both wired and wireless networks, and it has been a major research area for decades.

2.2.1 Timestamping-based Synchronization

Because of their ease implementation and versatility, timestamping-based synchronization techniques derived from the NTP protocol are the most often utilised methodology in wireless sensor networks. Several clock synchronization systems have been developed, most notably TPSN [12], which tries to decrease clock offsets by two-way timestamp exchange and enables network-wide clock synchronization across all nodes in a sensor network. This protocol has two phases: “level discovery” and “synchronization”. Level 0 is allocated to a node known as the root node. The root node sends a level discovery packet containing its node ID and level number to its neighbours at the start of the “level discovery phase.” Following receipt of this level discovery packet, each neighbour changes its level to one higher than the level obtained
from its parent. After a node has determined its own level, it can transmit a level discovery packet to its neighbours. Every node updates its levels in the same way. A node stops processing additional level discovery signals from any other nodes after updating its level. All of the nodes are placed in a hierarchical framework in this fashion. Each node synchronizes pairwise with a parent node that is present in the hierarchy during the "synchronization phase." The sender-receiver root node is the starting point for this paired synchronization as given in [13]. Each node computes the offset corresponding to its immediate upper-level node in the hierarchy using this method, and then updates its physical clock. These methods’ major drawback is the absence of clock skew correction, which leads to more frequent re-synchronization efforts, higher energy costs, and greater bandwidth utilisation. By incorporating a time dissemination technique based on flooding, FTSP [6] strengthens the shortcomings of TPSN. To lessen the jitter of interrupt handling and encoding/decoding time in the clock drift and offset estimate, FTSP uses multiple timestamps and linear regression. The one-way delay jitter was further represented as being regularly distributed. Starting with the root node, each sender node sends a single radio broadcast message that is timestamped with the sender’s local clock to all of its neighbours in order to distribute the time synchronization packet. The receiver uses its local clock to record the packet’s arrival time at the time of reception. The receiver node employs a linear regression approach to determine the offset and the relative drift rate with respect to the source node after acquiring a certain number of synchronization packets from the source. Each node keeps a logical clock that is based on the offset and the relative drift rate. However, synchronization packets from the root node can be received by a node that is within its broadcast range. The network broadcasts synchronization packets to the other nodes in order to synchronize them. To distribute the global time across all the nodes, the FTSP protocol constructs an ad hoc structure with a root node.

Aside from centralised and distributed synchronization protocols, one type of method improves the efficiency of synchronizing large-scale networks by building network clusters. Cluster heads are chosen based on the stability of the clocks, which allows them to act as a trustworthy sender of reference time. Each cluster head will be in charge of getting the global timing reference from the network root and synchronizing all nodes in the cluster. The most common clustering-based mechanisms are Reference-Broadcast Synchronization (RBS) and the global
synchronization protocol. Cluster-based solutions have been found to assist reduce the number of participating nodes during each synchronization period. Each receiver node records the time of packet arrival using its local clock when it receives the request packet. Then, each receiver node communicates the recorded arrival time of the request packet with the intermediate node’s other neighbours. It is expected that all receiver nodes receive the broadcast request packet at around the same time in real time. Each node computes the offset corresponding to all other nearby nodes of the intermediate node based on these local clock timestamped values. The accuracy of this protocol is mostly determined by the uncertainty in message delay. The greater the uncertainty in message delay, the lower the synchronization accuracy. The intermediate node broadcasts numerous request packets to its neighbours in order to improve synchronization accuracy. Each receiver node computes numerous offsets with respect to other receivers based on these request packets and computes the average of these offsets for each other receiver node. Furthermore, the protocol presents a way for estimating the drift rate at each node using the least square regression method.

### 2.2.2 Temperature-compensated Synchronization

Accurate clock skew estimation is the key for providing long-term synchronization since the clock offset is accumulated through clock drifting; therefore, it is important to accurately model the clock skew according to its physical characteristics.

The key variables that affect the frequency variation of crystal oscillators from their normal frequency are temperature, supply voltage, manufacturing flaws, and ageing. The common assumption that the clock drift remains constant over a certain period of time is not valid in applications with the frequent temperature fluctuations. Recent studies have described and predicted the relationship between these parameters and the frequency deviation. In [14], Schmid et al. used the temperature and manufacturing defect models to create a clock skew correction system that is software-based. The temperature-assisted clock self-calibration (TACSC) method created in [15] also used temperature data to self-calibrate the clock skew on sensor nodes. It was demonstrated that by utilising the temperature information, the synchronization precision may be increased by more than one order of magnitude. A system for self-calibration
and clock skew adjustment in [16] that made use of temperature and voltage information. The results indicated that taking into account both supply voltage and temperature might increase the accuracy of clock skew estimates, allowing for a longer re-synchronization interval and lower the communication costs.

EACS was proposed by Yang et al [17]. Taking into account temperature variations, Yang et al. describe a highly accurate clock skew estimation. The authors propose an additional information-aided multi-model Kalman filter (AMKF) algorithm to dynamically compensate for the clock skew. The benefit of EACS is the possibility of using it as a component of any conventional WSN synchronization protocol based on the clock skew estimation. This allows the update of the local clock with local information before the clock resynchronization is carried out, which improves the lifetime of the mote by increasing the synchronization period. The EACS algorithm builds up an initial table of each node of the network based on the correlation between temperature and clock skew. A Kalman filtering is applied to obtain the theoretical relationship between those components. This procedure has very large computational requirements, which does not make it a suitable method for low cost WSN. The above mentioned proposals improve the behavior of protocols under temperature variations. However, both approaches (i) have high computational costs and energy consumption, (ii) require a large amount of memory per node to store the relationship between temperature and clock skew, (iii) cannot manage new temperatures without a recalibration process and (iv) are constrained to always use the same reference node. Any change in these restrictions triggers the need for a new calibration.

### 2.2.3 Timestamp-free Synchronization

Another class of synchronization protocols that tries to mitigate the overhead of frequent timestamp usage is timestamp-free synchronization. Timestamp-free techniques collect the implicit timing information from round-trip time measurements of bi-directional message exchange rather than communicating timing information to the other nodes in the network.

The greatest advantage of this approach over the aforementioned packet-based synchronization techniques is that no timestamps are exchanged between nodes, so that a significant
amount of energy and communication bandwidth can be optimized. At the same time, the implementation of this novel approach can be embedded into existing network traffic, which also avoids the possibility of timestamps being attacked during transmission. The basic idea of timestamp-free synchronization is to convey timing information implicitly through the receiver’s predefined response to the sender, rather than exchanging explicit timestamps. Brown and Klein [18] presented a timestamp-free synchronization algorithm considering propagation delay to compensate for local clock drift, in which, the response time was pre-defined such that the total duration of packet reception and transmission time at the receiver follows a determined schedule.

2.3 Source of Synchronization Errors

In practice, the clock sources in each of the nodes in a distributed system run at slightly different frequencies, causing the clock values to gradually diverge from each other. This divergence has different names depends on the source which causes the deviation.

The error caused by timestamping message transmission is often made up of both stochastic and deterministic uncertainties, and the latter may be maintained below a certain threshold. The primary goal of messaging protocols is to reduce the negative effects of message delivery errors, which may occur for a variety of reasons. Elson et al [19] has determined the most common error causes chronologically, which is made up of six main error components:

- **Send time**: This is the amount of time it takes the sender at the application layer to construct the packet and deliver it to the MAC layer. This time is entirely predictable because it is mostly determined by the processor load.

- **Access time**: This is the time spent at the MAC layer waiting to access the transmission channel. Unless a fixed time slot is allocated to each node for channel access, this time period is also a random variable.

- **Transmission time**: The time it takes to push a packet into a wireless channel. This time duration is predictable and is determined mostly by the packet size and data rate of the channel.
2.3. Source of Synchronization Errors

Figure 2.2: Delays of exchanging a timestamp over a wireless link.

- Propagation time: This is the amount of time it takes to transfer a packet from source to destination through a channel. It is mostly determined by the distance between the source and destination, as well as the channel’s propagation speed. The propagation time is very short because the propagation speed is very high and the distance between the source and destination nodes is relatively close.

- Reception time: This is the time required to push the packet from the wireless channel to the MAC layer. This time period, like transmission time, is predictable.

Send time is the time it takes to compile a message at the application layer and send it to the Medium Access Control (MAC) layer. This delay is caused by kernel processing, application-induced delays, and variable delays caused by the operating system scheduler. The variable delay imposed by MAC protocols owing to their propagation medium access restrictions is referred to as the access time. The propagation time is the time it takes for a message to go from the transmitter to the receiver and is often insignificant in comparison to the other delays. Finally, the receive time is the time it takes to process the received message and transfer it to the application layer. It should be noted that the access time is included in the transmitter side delays.

The preceding list has been expanded by include transmission and reception timings. The transmission time denotes the time necessary to convey the message bit by bit in the physical layer. The reception time is the time required to receive a message in the physical layer and pass it to the MAC layer. The transmission time is a frame length dependent delay that follows the access time on the transmitter side. The reception time is the first delay of the frame during
reception. These two delays are predictable and equivalent.

The delays on the transmitter side have been extended with encoding time, which indicates the time required for encoding the current message, before transmission time, which covers the processing delay associated with transforming the bit information to the electromagnetic waveform right after raising the transmitter side interrupt. The receiver side delays are increased to account for decoding time, which refers to the time required to translate the electromagnetic waveform into bit information and decode the message on the receiver side. The byte alignment time, which represents the time spent on the receiving side owing to the differing byte alignment of the transmitter and the receiver. The interrupt handling time on the receiver side is the time it takes for the radio chip to wait for the processor to complete the current instruction or crucial section before switching to the interrupt handling subroutine.

The software delays in transmission (software jitter) and reception (interrupt handling time) are random, and their variability is determined by the overall system architecture. The diversity of the delays caused by these components is particularly dependent on the processor load. Under typical working circumstances, the calibration time usually has a guaranteed delay and a minimal variability. The access time is determined by the transmission medium, while the characteristics are determined by the MAC layer implementation.

- **Timestamping errors**

The time stamps should ideally be recorded when the message emission at the transmitter begins and when the message reception at the receiver is detected. In this manner, the chance error source that affects the time records is completely eradicated. In reality, however, the frames are created long before the transmission really begins, and they are buffered to effectively manage the MAC layer rules for accessing the propagation media. It is not possible to prevent random access time in this situation. Similar to the previous point, more time faults are added to the time information when the timestamps are removed from the physical layer. In order to reduce the possibility of messaging errors, it is best to store the timestamps for both the transmitter and receiver sides as close as feasible to the node’s physical layer.

It should be highlighted that the causes of errors can be reduced by switching software parts to hardware implementations. In this scenario, the interrupt handling time and software delay
on both the transmitter and receiver sides would be predictable. The access time is the sole source of delay that is immutable. But it’s not always possible to implement such a solution. For instance, because certain IoT devices have really limited resources, the hardware producers wish to restrict software access to specific hardware capabilities linked to communication timings. The timing of the timestamp collection is also uncontrollable with some wireless communication methods because they are largely implemented in hardware. As a result, since the wireless communication technique and underlying hardware both influence the timestamping method, the possible time synchronization precision relies on them.

2.4 Transmission of time reference

The objectives of time synchronization strategies are very diverse and, therefore, synchronization strategies can be applied in very different ways. The information gathered on messaging error origins is heavily influenced by the system’s message design. Three well-known synchronization schemes are introduced in the section: two-way timestamping message exchanges, one-way timestamping message broadcasting and receiver-receiver synchronization.
2.4.1 Two-Way Message Exchanges

This timestamping method is the most extensively used mechanism for sending time reports between two neighboring nodes known as sender and receiver nodes. A sender node in this approach seeks the most recent time report from one of its neighbours. The receiver node responds with its most recent time report as well as the request’s reception time. The clock relation model parameters are calculated by the sender node utilising all four time information
2.4. Transmission of time reference

displayed in Fig. 2.5. Since there are two time reports for each node ($T_1$ and $T_4$ for sender, and $T_2$ and $T_3$ for receiver), this scheme can achieve very high synchronization accuracy. Furthermore, suppose that $d = d_{12} = d_{21}$, then we have

$$ \theta = \frac{(T_2 - T_1) - (T_4 - T_3)}{2} \tag{2.3} $$

$$ d = \frac{(T_2 - T_1) + (T_4 - T_3)}{2} \tag{2.4} $$

when the uncertainties of time are neglected. As a result, this message exchange mechanism may be used to estimate time offset and messaging delay, both of which are required for highly accurate clock synchronization.

Most existing time synchronization systems are based on two-way timestamping message exchanges and pairwise communication among different nodes. This, however, requires that each node must handle both the receiving and transmission of clock information, which may be difficult to implement for some IoT applications.

2.4.2 One-Way Message Dissemination

Figure 2.6: One-way messaging scheme, where a root node emits a synchronization message, and all of its neighbors receive. Each receiver synchronizes to the time of the root node. In (a), a typical operation of the scheme, and in (b), timeline of message exchanges. [3]

Due to the energy limitations of low-power networks, a root node broadcasts synchronization signals, which are received by all of its neighbors, as illustrated in Fig.2.6. This strategy
reduces the power needs of the time synchronization method by half by having the recipient nodes synchronize to the root node using just one-way communications.

on the transmitter side, the energy can be saved at the expense of increasing message unpredictability. The timing error causes in two-way message exchanges are limited to a single messaging period i.e., \( T_4 - T_1 \) in Fig. 2.6 b. However, for one-way timestamping mechanism, timing error accumulates and must be addressed by other techniques in order to achieve better synchronization precision. For example, in the software stack, the time stamping can be placed as near to the radio interface as possible, e.g., to the MAC layer.

### 2.4.3 Receiver-receiver Synchronization

![Figure 2.7: Receiver-receiver messaging scheme, where a root node emits a synchronization message, and all of its neighbors receive. Then, the receiver nodes exchange their local time along with the received reference time. In (a), a typical operation of the scheme, and in (b), timeline of message exchanges. [3]](image)

The wireless sensor network’s broadcast nature may be utilized to avoid transmitter side uncertainties by adopting the receiver-receiver mechanism shown in Fig. 2.7. In this approach, a root node broadcasts the reference time firstly, which will then be received by its neighbours. At the start of the reception, the neighbours record their local timestamps. The receivers then share the time data they have saved with one another. Since both neighbours get the sent frame at the same time, the transmitter side delays are removed. Even if there is a timing difference owing to various propagation delays, it is insignificant for short-range networks. Because the transmitter side random delays represent the majority of the random message
delays, eliminating them can improve the overall performance.

The four types of messaging systems have various effects on time synchronization accuracy, power consumption, and network architecture. In this regard, one-way message distribution necessitates the fewest number of transmissions in the network and does not impose a topology at the expense of controlled degradation (through proper timestamping mechanism) in synchronization accuracy. Two-way message exchanges give the most information on clock parameters, however each synchronization period needs two message exchanges, implying increased energy and processing requirements, as well as pair-wise operation. However, it may be utilised in circumstances when the lower levels of the communication protocol cannot be adjusted, as proved by Son et al. [20] over constraint application layer protocol to obtain sufficient synchronization precision. The receiver-receiver synchronization technique can achieve more accuracy than the one-way message distribution scheme, but it consumes more energy and computer resources. Because this mechanism is dependent on the existence of a transmitter node, its node failure resilience is better than that of two-way message exchanges but lower than that of one-way message dissemination. In receiver-only synchronization, the benefits of two-way message exchanges and one-way message dissemination are combined. This technique offers limited resistance to node failures but minimal power consumption and great synchronization precision.

The attributes of these four message exchange schemes suggest that if one has the option of selecting a messaging strategy, they can base their decision on the needed synchronization precision for a particular energy and computational resource budget. However, for the majority of IoT implementations, a system designer is forced to select off-the-shelf components to integrate with the system. Another factor to consider in this regard is the required level of modification to the standard compliant-systems. Receiver-receiver synchronization is an appealing solution for such deployments since it eliminates transmitter side delays without the need for additional timestamping. When some nodes have more resources than others, receiver-only synchronization can be selected. When the deployment supports two-way messaging, two-way message exchanges are the preferable method. As a result, an IoT practitioner must choose a messaging strategy that takes into account various practical considerations, including multi-hop synchronization support.
2.5 Data Synchronization

The data collected from several sensors is often fused in order to generate virtual objects using implicit measurements of some occurrence. Data from several sources is integrated and correlated to provide extra knowledge that is typically not visible in the original data [21]. The cross-correlation of data from diverse sources must be computed using the obtained samples, which necessitates a shared understanding of time. As a result, implicit information sources may rely on physical things sharing a shared sense of time. Because WSNs are spread, they may collect samples from a spatiotemporal field, for example, for structural health monitoring [22]. These applications necessitate the collection of synchronous samples from all sensors in order to estimate the spatial parameters of interest [23]. As a result, the simultaneous execution of some periodic activities, such as sampling, necessitates a coherent view of time within the network.

![Diagram](image)

**Figure 2.8:** Drifted data timestamping due to the unsynchronized sensor clock.

Through the common operational picture (COP), the information content of data provided by an IoT network is used by applications and services. The COP does not distinguish between actual and virtual (information) objects semantically, thus it is natural to virtualize all information sources and sinks. These entities are known as virtual objects, and they can either
proxy physical things or be tied to a software component. In either case, they are a type of information unit in the COP. In turn, such an information-based abstraction necessitates agreement on how to rank the data in relation to a certain argument, such as time or frequency. The gathered data for the majority of physical information sources is naturally arranged in time, thus the information content allocated to distinct virtual objects must be ordered according to their chronology.

2.6 Chapter Summary

Clock synchronization, as one of the most critical enablers in IoT systems, may improve overall network cohesiveness, seamless interaction, and timely collaborations across many heterogeneous devices and processes. Because of the nature of these heterogeneous applications, which typically have higher quality of service (QoS) and more complicated operating scenarios, more stringent requirements are imposed on the synchronization protocol design, such as higher achievable synchronization accuracy, reduced network overhead for exchanges of packets containing timestamps, and enhanced security against external attack and internal abnormalities. These criteria make synchronization protocol design more difficult than standard point-to-point synchronization techniques. Furthermore, the dynamic and severe working conditions of heterogeneous IoT systems will induce additional challenges. Temperature-induced clock drift will be more severe in industrial scenarios, resulting in a longer re-synchronization interval and resource consumption. As a result, innovative synchronization strategies for heterogeneous and resource-constrained IoT systems are necessary.
Chapter 3

Low-Overhead Data Synchronization

3.1 Introduction

Internet of Things (IoT) technologies are attracting growing research interests nowadays due to the proliferation of wireless communications and intelligent devices [24]. As a critical constituent of IoT systems, wireless sensor networks (WSN) are increasingly important in supporting a wide variety of time-sensitive applications, including data fusion, anomaly detection [25], industrial condition monitoring [26], and healthcare services [27]. These applications pose stringent requirements on the temporal consistency among the associated data for accurate analysis and timely decision-making.

Due to the ubiquitous deployment of IoT devices, a large amount of homogeneous data will be continuously sampled and delivered from the distributed devices to support centralized data processing, the accuracy of which depends heavily on the temporal consistency of the collected data [28]. However, the heterogeneous qualities of the clock oscillators and their diverse operating conditions will inevitably lead to incoherent clock behaviors among the IoT devices as time passes [29]. The asynchronous data acquisition and lack of perfect clock synchronization will significantly degrade the cohesion of the obtained data measurements, prohibiting accurate temporal analysis. Consequently, data synchronization aiming at achieving temporal alignment among the massive data measurements is essential to support time-sensitive IoT applications.

To enhance the temporal correlation among the distributed devices while supporting time sensitive applications, e.g., data fusion, numerous works have been proposed in the literature to
achieve accurate and cost-efficient clock synchronization. Evolved from Precision Time Protocol (PTP), a clock skew correction scheme is designed for TSNs in [30] by incorporating the relative clock drift factor between the neighboring nodes. However, lacking the consideration of network structure and channel asymmetry may lead to defect synchronization accuracy. Moreover, aiming at achieving low-overhead synchronization, authors in [31] proposed a PANSO scheme by fusing the observation instants from distributed IoT devices regarding the same event in time-sensitive industrial networks. However, period timestamping and synchronization is still required, which will keep occupying the limited network resources.

The inherent limitation of traditional clock synchronization methods is significantly hindering the efficiency of time sensitive applications. To address this issue, data synchronization protocols are becoming increasingly attractive. By estimating the clock drift to investigate the real sampling instants, the data synchronization and temporal ordering of the massive sensing data is achieved with the help of Kalman filters. However, the synchronization is achieved based on the assumption that the real sampling time can be accurately obtained by the filters, which may lead to high computational burden and thus degrading the data fusion efficiency. Moreover, a data sampling rate synchronization scheme is designed by compensating the clock offset with the help of a pre-trained clock skew error model, which is obtained by periodic observation and Kalman filter prediction. In addition, a lightweight data synchronization is achieved by observing the clock difference based on the encapsulated local timestamps at each relay node. The variation of network latency and propagation delay are not considered throughout the design, which will cause uncertain synchronization performance, especially for large-scale IoT systems.

Traditionally, data synchronization is achieved by designing dedicated clock synchronization protocols, which try to substantially calibrate the local clock of each IoT device according to the common time reference by exchanging a sequence of timestamps for clock inaccuracy estimation. For example, evolved from Precision Time Protocol (PTP), a clock skew correction mechanism is achieved in [30] by incorporating the relative clock drift factor between the neighboring nodes in TSNs. Meanwhile, by fusing the observation instants regarding the same event, network-wide synchronization is achieved in time-sensitive industrial networks [31]. Unfortunately, due to the excessive communication overhead inherent to the frequent
timestamp transmission, packet-switching-based clock synchronization will become infeasible in resource-constrained scenarios. Moreover, in time-sensitive IoT systems, an application-dependent synchronization requirement must always be maintained, resulting in the necessity of periodic resynchronization due to the instability of the clock oscillators. An overwhelming number of timestamping packets must be shared within the network to guarantee the required temporal consistency. As a result, the efficiency and timeliness of the time-sensitive applications will be suspicious, especially for large-scale IoT systems.

Based on this observation, designing data synchronization protocols with reduced network overhead is preferred to achieve accurate temporal alignment and efficient processing of the distributed data measurements generated from multiple IoT devices [32]. Numerous packet-switching-free clock calibrating mechanisms have been proposed in accomplishing accurate and efficient synchronization for WSNs. In most of these approaches, including reference broadcast synchronization (RBS) [33], timing-sync protocol for sensor networks (TPSN) [34], and flooding time synchronization protocol (FTSP) [35], the sink node, which is responsible for collecting all sensing data throughout the network by multi-hop routing, will broadcast beacons containing timestamps to other sensors for clock calibration. Because of avoiding bidirectional timestamp exchange among the distributed devices, the efficiency and feasibility of clock synchronization are expected to be enhanced. However, periodic synchronization that occupies network resources is still necessary to maintain a certain level of clock accuracy, which is non-ideal for resource-constrained IoT devices. In addition, a lightweight data synchronization is achieved by observing the clock difference based on the encapsulated local timestamps at each relay node without sharing dedicated packets for timestamps [36]. Nevertheless, the estimation bias caused by network-related issues, e.g., network latency and propagation delay will significantly affect the data analysis accuracy.

In this chapter, a low overhead data synchronization scheme is proposed to tackle the challenges inherent to the traditional data synchronization methods. The contributions of this chapter can be summarized in two aspects. First, a prescheduled task period is proposed to serve as a time reference. By comparing the task period and the real observed information, the temporal information of the collected data could be compensated for only at the sink nodes according to the estimated clock parameters. Second, contrary to traditional approaches that necessi-
tate frequent synchronization to adjust the clocks, the proposed data synchronization can be achieved more efficiently. There is no extra communication overhead introduced during the clock compensation due to the reason that additional transmission of dedicated timestamps is eliminated.

The remainder of this article is organized as follows. The system model is presented in Section 3.2, including data acquisition and fusion system using temporal correlation of the sampled data. The proposed method is described in Section 3.3 in detail, including the estimation of clock parameters and the corresponding compensation. In Section 3.4, the performance of the proposed method is evaluated via experimental results in terms of data fusion accuracy and network overhead analysis, followed by the conclusion in section 3.5.

3.2 System Model

In this section, the overall data fusion architecture and the clock model of each sensor node are described in detail, aiming to show the severeness of temporal inconsistency caused by the clock inaccuracy among the distributed data during centralized data processing.

3.2.1 Data Fusion and Sampling Model

A centralized data fusion system is considered in this chapter, which consists of two components, i.e., the sink node SN and multiple sensor nodes $S_i$, $i \in \{1, 2, ..., n\}$, as shown in Fig. 4.1. SN is usually associated with sufficient computational resources and an accurate clock, which can serve as the reference clock during data synchronization. Meanwhile, it plays a role as the fusion center responsible for collecting raw data generated from distributed sensors. In supporting various time-sensitive IoT applications, these distributed data should be aligned and correlated into a data set for further processing, such as data aggregation and data association.

In the proposed system, a large number of inexpensive sensors are deployed in the target area, executing different time-sensitive acquisition tasks like critical object tracking or condition monitoring. The sampling instant of each sensor will be triggered locally by its own clock. Meanwhile, the time instant of each sampling will be captured and recorded with the observed data, denoted as $(O_j, TS_j)$. As a result, the physical time of each measurement can be recorded
sequentially. However, the excessive data inherent to large-scale sensor networks with frequent samplings will inevitably pose overwhelming burden on the network resources, which lead to the necessity of joint consideration between the task fulfillment and the sampling capacity of each sensor device.

In the presence of limited communication resources, the distributed sensors are usually prohibited to transmit the sampled data immediately to SN after its generation. Instead, the transmission process will start if a certain amount of data have been collected. In other words, the sampled data will wait for a predefined interval in the local data register until the predefined transmission instant, as demonstrated in Fig. 4.2. The task period $p_i$ is defined as the time interval between two consecutive transmission points, which is scheduled by SN. After the predefined $p_i$, a series of timestamped data $TD_i$ will be delivered from the sensor node $i$, represented as

$$TD_i = S_j[(O_j, TS_j), j \in \{1, 2, ..., m\}$$ (3.1)
where \( m \) is the total number of observations at node \( i \).

### 3.2.2 Sensor Clock Model

One of the critical issues leading to data fusion inaccuracy is the timestamp error induced by the non-ideal crystal oscillator (OX)-driven clock embedded at each sensor device. Typically, the local clock equipped is responsible for tracking the standard time and provide a temporal reference to each local observation. However, the low-cost clocks are usually associated with original manufacturing flaws and vulnerable to environmental variations. Therefore, clock drift becomes a non-negligible impact leading to inaccuracy of local time information. According to [?], the drifted clock can be expressed using an empirical model, given by

\[
C_i(t) = \alpha_i t + \beta_i
\]

where \( \alpha_i \) and \( \beta_i \) represent the local clock skew and clock offset, respectively.

Moreover, the timestamp will be inevitably distorted by its local clock deviation from the standard time, meaning that it cannot reflect the real instant of the sampled observation. Given a certain sampling frequency \( f_i \) of sensor node \( i \), the error between the timestamp of its \( k^{th} \) measurement and the real sampling instant can be estimated by

\[
\epsilon_i = \left| T_i - TS_{i,k} \right| = \left| \frac{k}{f_i} - (\alpha_i \frac{k}{f_i} + \beta_i) \right|
\]

Figure 3.2: During the task period of node \( i \), a series of timestamped data will be generated and recorded, which will not be transmitted to the sink node until the prescheduled transmission point.
\( \epsilon_i \) can be used to evaluate the improvement of data fusion after adopting the data synchronization scheme.

### 3.3 Data Synchronization based on Prescheduled Task Period

As previously mentioned in section II-A, timestamped data generated from distributed sensor nodes will be transmitted to SN for data fusion and further processing. Due to the non-ideal sensor clocks, the associated timestamps will deviate from their real sampling instants. In time-sensitive IoT applications, the cohesive temporal correlation among the distributed data serves as one of the most critical prerequisites to guarantee that the information extracted from the raw data is meaningful. Hence, in achieving accurate data aggregation and efficient data fusion, timestamps should be compensated accurately according to the observed clock errors to cohere the distributed data in the temporal domain.

Different from traditional clock synchronization techniques, a low-overhead data synchronization scheme is proposed in this chapter, with the goal of achieving accurate temporal consistency among the distributed data without exchanging excessive timestamps. The core idea of the proposed scheme is to adjust the timestamps of the collected data based on the predefined task scheduling period, instead of synchronizing the physical clocks embedded at sensor devices. As shown in Algorithm 1, the proposed scheme consists of two successive phases, namely, network initialization and data synchronization. These two successive stages are explained in detail as follows.

#### 3.3.1 Network Initialization

The overview of the transmission process during data synchronization is demonstrated in Fig. 4.3. In the network initialization stage, each sensor node will first report its configuration information to SN by sending a sampling request packet, including its sampling capacity, memory size, and communication parameters. Once SN receives all sampling request packets from the distributed sensor nodes, it will broadcast an initialization signal to schedule an individualized
3.3. Data Synchronization based on Prescheduled Task Period

Algorithm 1 Timestamp Calibration based on the Estimation of Clock Parameters

Require: Number of sensor nodes: n
1: **Network Initialization:**
2: for each sensor node \( i = 1 : n \) do
3: Report the local configuration information
4: Assign the local sampling frequency \( f_i \)
5: Define the task scheduling period \( p_i \)
6: Calculate the propagation delay \( \tau_i \)
7: end for
8: **Clock Parameters Estimation and Compensation:**
9: for each sensor node \( i = 1 : n \) do
10: for each task period \( j = 1 : \#p_i \) do
11: Calculate \( \alpha_i \) and \( \beta_i \) according to Eq. (5.4) and Eq. (5.5)
12: Calibrate \( ts_i^j \) according to Eq. (7)
13: end for
14: end for

Sampling frequency and task period to each sensor node based on its characteristics, including the sampling capacity and the importance of its local task. Meanwhile, this initialization signal also contains the current time information of SN, which can help the sensors to calibrate their local clocks in achieving initial clock synchronization. It is worth noting that the sensor clocks will be synchronized once during the whole scheme to avoid the impact of the initial clock phase error, and no more calibrations will be applied to the physical clocks in the subsequent stages. Moreover, the network initialization will be executed again only if the sampling tasks need reallocation or the network topology changed, e.g., a sensor node joins or leaves the group.

Moreover, the network uncertainties will be a non-negligible factor deteriorating the overall synchronization performance. In the proposed scheme, since the transmission instant for each node is prescheduled by SN, it is reasonable to assume that there is no potential contention when trying to access the channel. Moreover, the propagation delay between each node and SN is assumed to be time-invariant and symmetric, which can also be estimated at the initialization stage. The timestamps of the round-trip communication signals exchanged between the node \( i \) and SN will be used to calculate the propagation delay, given by

\[
\tau_i = \frac{(t_i^4 - t_i^1) - (t_i^3 - t_i^2)}{2} \tag{3.4}
\]
which will be further used to improve the accuracy during the estimation of clock parameters.

### 3.3.2 Estimation and Compensation of Clock Parameters

In achieving temporal consistency among the distributed data, the drifted timestamps should be calibrated before data processing. Thus, an accurate estimation of the clock parameters is necessary for achieving clock calibration at Sink Node (SN). Moreover, due to the fact that the unstable oscillation frequency inherent to the low-cost clocks is the dominating source of the time-varying clock drifts, the estimation of clock parameters must be conducted periodically for information updates after each task period.

In the proposed scheme, the clock parameters are estimated by comparing the pre-scheduled task period and the real observed one for each sensor node. Once the initialization phase is finished, the sampling task will start according to the preallocated setups. Denoted by $T_{i,0}^j$, the starting timestamp will serve as an indicator to show the initial instant of the $j^{th}$ task period of node $i$. After the sampling duration and buffering time, each sensor will upload the collected data to SN at the prescheduled transmission time $T_{i,TX}^j$. Once the data packet is received at SN, it will record its local receiving time $T_{i,RX}^j$. Therefore, Based on the previously estimated clock offset $\beta_j^{i-1}$ and the latest time information associated with the received data, the clock skew of

![Diagram](image-url)

**Figure 3.3:** The transmission of the data and the associated timestamps during one task period. The first pair of communication signals is used for initial clock offset estimation, network delay observation, and task scheduling.
3.3. Data Synchronization based on Prescheduled Task Period

**Network Initialization**
- Report the local configuration information
- Assign the local sampling frequency $f_i$
- Define the task scheduling period $p_i$

**Clock Parameters Estimation**
- For each node, calculate the propagation delay $\tau_i$
- For each task, calculate $\alpha_i$ and $\beta_i$ based on Eq.() and Eq.()

**Clock Parameters Compensation**
- For each task, calibrate $t_s^j$ according to Eq.()

Figure 3.4: The workflow of Data Synchronization based on Timestamp Calibration and Clock Parameters Estimation.

Node $i$ in $j^{th}$ task period can be estimated and updated as

$$
\alpha^j_i = \frac{T_{i,Tx}^j - (T_{i,0}^j - \beta_i^{j-1})}{p_i} = \frac{T_{i,Rx}^j - \tau_i - (T_{i,0}^j - \beta_i^{j-1})}{p_i} \tag{3.5}
$$

where $\tau_i$ is the propagation delay between the node $i$ and SN calculated according to (5.4).

Based on (5.5), the accumulated clock offset can be further calculated by

$$
\beta^j_i = \sum_{i=1}^{j-1} (\alpha^j_i - 1) \cdot p_i \tag{3.6}
$$
where $p_i$ is the scheduled task period of node $i$.

Based on the clock parameters of each node estimated by SN, the distorted timestamps can be calibrated accurately for each task period. The reconstruction of the data sampling instant is given by

$$T^{j}_{S_{i,k}} = \frac{T^{j}_{S_{i,k}} - T^{j}_{i,0} - \beta^{j-1}}{\alpha^{j}_{i}} + T^{j}_{i,0} \quad (3.7)$$

where $T^{j}_{S_{i,k}}$ indicates the time information associated with the corresponding data. The calibrated timestamps will be updated and recorded for further data fusion and processing.

### 3.4 Performance Evaluation

In this section, the proposed task period based data synchronization scheme is evaluated from two perspectives. Firstly, the effectiveness of the proposed scheme is evaluated by comparing the data fusion performance after adopting the data synchronization scheme. Moreover, the network overhead incurred by temporal alignment is demonstrated and compared in detail among three different synchronization techniques.

#### 3.4.1 Data Fusion Accuracy

In this simulation, a total number of 30 sensor devices are randomly deployed in an IoT system for environmental condition monitoring. The sensed data will be collected and fused in SN for different applications. The acceptable data fusion accuracy is set to be in microsecond level, which is a general requirement in supporting time-sensitive applications.

Due to the temporal inconsistency among the distributed devices, the collected data will be mismatched in temporal domain as a consequence. As shown in Fig. 5.5, the accumulated timestamp error without adopting data synchronization will increase significantly with time, meaning that the data processing accuracy is intolerable. By contrast, after adopting the proposed data synchronization scheme, the accumulated timestamp error will keep lower than the predefined application requirement due to the temporal compensation by clock parameter estimation as compared to the case without synchronization[37]. Meanwhile, this accumulated error grows much slower than the case without synchronization due to the periodic clock com-
3.4. Performance Evaluation

Figure 3.5: The accumulated timestamp error associated with the data to be fused in the sink node. The timestamp accuracy can be significantly enhanced by adopting the proposed data synchronization scheme, which can help to remain the overall error to be lower than the application-oriented requirement.
Figure 3.6: The comparison of network overhead induced by temporal alignment processes. Because of avoiding frequent timestamp exchanges, the proposed task period-based method can achieve accurate synchronization with the least network resource consumption.

pensation in the presence of the ever-growing clock error. We can conclude that the proposed data synchronization scheme can further support applications with an even higher synchronization requirement, which can be typically observed in more time-sensitive scenarios.

### 3.4.2 Network Overhead Analysis

On the other hand, achieving accurate temporal alignment among the massive data will inevitably pose additional burden on the network resources. Adopting synchronization schemes with excessive network resource consumption will lead to high network overhead and occupy resources of critical applications. The overall performance of the time-sensitive applications will be suspicious, especially in resource-constrained scenarios. As a result, analyzing the resource consumption during synchronization is of the utmost importance.

In this simulation, the network overhead caused by the temporal alignment is defined as
the total number of timestamps required during synchronization. A comparison among the
proposed sampling-period-based data synchronization, traditional timestamp-based data syn-
chronization [35], and packet-switching-based clock synchronization, i.e., PTP, is conducted.
As shown in Fig. 5.6, we can observe that the lowest network overhead is incurred by adopt-
ing the proposed scheme since only a few timestamps are required to be exchanged between
SN and each sensor node for clock information sharing. By contrast, the network resources
required by traditional synchronization methods will be much higher to calibrate the unsta-
ble clocks. Furthermore, the extremely low accumulated network overhead of the proposed
scheme during the long-term operation will lead to a significant benefit for network resources
saving, meaning that it is a promising candidate to support resource-constrained applications.

3.5 Chapter Summary

In this chapter, a novel prescheduled task period-based data synchronization method have been
proposed to achieve accurate temporal alignment among the distributed data before data fusion
and processing in supporting time-sensitive IoT applications. More specifically, by comparing
the predefined task period of each sensor node, the real observed information at SN will be used
for clock skew and offset estimation. By compensating the clock inaccuracy only through com-
paring the task periods, temporal alignment with extremely low network overhead is achieved.
Simulation results demonstrated that the proposed method can help to significantly enhance
the data fusion performance with the presence of inaccurate clocks. Moreover, the induced
network overhead is dramatically reduced during clock parameter estimation, as compared to
the traditional timestamp-based synchronization approaches.
Chapter 4

Situation-Aware Hybrid Time Synchronization

4.1 Introduction

The expeditious evolution of Internet of Things (IoT) and wireless communication technologies is expected to further boost the proliferation of connected devices for the foreseeable future. Heterogeneity is becoming one of the fundamental characteristics of the future IoT systems, due to the diverse communication standards [10], various processing capabilities, and different operating conditions of the IoT devices involved [11]. With the increasingly stringent and multifarious requirements, maintaining an accurate temporal consistency among the involved heterogeneous IoT devices becomes indispensable in achieving timely information exchange for collaboration.

Accurate time synchronization, which can cohere the distributed IoT devices in the temporal domain, serves as one of the critical foundation to enable effective interactions throughout heterogeneous IoT systems [38]. By adjusting the different clocks network-wide according to the same time reference, the distributively gathered information can be precisely aligned in supporting various collaboration and coordination, e.g., remote monitoring, data fusion, and networked control.

Traditional synchronization methods, including precision time protocol (PTP) and flooding time synchronization protocol (FTSP) [6], are achieved with fixed routine without fully
considering the heterogeneity and diversity of IoT systems. However, unlike traditional local networks where nodes and their operating requirement are mostly homogeneous, *multi-source timestamping uncertainties* will be induced in heterogeneous IoT systems, impeding the accuracy of traditional fixed synchronization methods. First, the dynamic and unpredictable network conditions, including asymmetric latency and packet delay variation, will severely affect the timeliness and reliability of timestamp exchange over Internet. Second, the software timestamping accuracy may become less reliable in complicated IoT systems, due to the timestamping jitter inherent to the time-varying CPU load and timestamping access time [2]. Finally, oscillator instability induced by environmental dynamics will further reduce the timestamping inaccuracy. The aggregation of these multi-source timestamping uncertainties will substantially degrade the performance of traditional synchronization methods.

To alleviate the impact of these timestamping uncertainties, different synchronization methods have been proposed in the literature. For example, a digital-twin-enabled distributed time synchronization protocol is proposed in [39] to overcome the impact of network uncertainties and temperature-induced clock drifting. By establishing an accurate virtual clock model for each IoT device considering various operating conditions, the impact of timestamping uncertainties on the synchronization is reduced. Moreover, timestamp-free synchronization methods are designed in [40] and [41], which can achieve accurate time synchronization without using explicit timestamps for clock parameter estimation. As a result, the issue of inaccurate software timestamping can be theoretically eliminated. However, none of these methods can simultaneously address the multi-source timestamping errors induced by various uncertainties inherent to the IoT systems. Meanwhile, an adaptive synchronization strategy is required to achieve accurate synchronization in highly dynamic operating scenarios.

In this chapter, a novel hybrid time synchronization protocol enabled by multi-source uncertainty modeling and integrated time information exchange is designed to achieve accurate time synchronization in heterogeneous IoT systems. Specifically, the contributions of this chapter can be summarized in three aspects. First, the multi-source timestamping uncertainties inherent to the overall synchronization process due to the non-deterministic communication process, temperature variation, and CPU load are accurately modeled to analyze their corresponding impact on the timestamping accuracy. Moreover, a hybrid time synchronization
scheme is proposed to achieve adaptive and situation-aware clock calibration in terms of operating conditions. The optimal combination of different skew estimation methods is achieved according to the real-time multi-source uncertainties observed at each IoT device. In addition, an integrated time information exchange mechanism is designed by merging redundant timestamps to reduce the network resource consumption during the hybrid time synchronization.

The remainder of this chapter is organized as follows. The system model focusing on multi-source uncertainty analysis is presented in Section II. A novel situation-aware time synchronization method is described in Section III, including the time information exchange and the hybrid time synchronization protocol design. In Section IV, the performance of the proposed method is evaluated via simulation results, followed by the conclusion in section V.

4.2 System Model

In this section, the illustration of a heterogeneous IoT system and the modeling of multi-dimensional timestamping uncertainties will be given in detail.

4.2.1 Heterogeneous IoT Systems

Typical timestamping scenarios in an IoT system are shown in Fig. 4.1, where different timestamping errors will be caused for distributed IoT devices. As a result, devices in the same IoT system will generate timestamps with different qualities. Moreover, for different IoT devices, the major contributor to the timestamping error will be different due to the uncertainties inherent to the highly heterogeneous IoT. For example, the timestamping uncertainty in one IoT device may be caused by one of the uncertainties, e.g., CPU load variation, which is more straightforward to design dedicated synchronization strategy. However, in the case that the timestamping uncertainty is jointly originated from multiple sources, fixed and unified synchronization strategy will be less effective, especially when involving high dynamics.

To deal with the dynamic and multi-dimensional uncertainties associated with the synchronization process, a situation-aware hybrid time synchronization protocol is proposed in this chapter based on the modeling of the timestamping uncertainties. As shown in Fig. 4.1, the network coordinator, which serves as the master clock throughout the network, will be respon-
4.2. System Model

Figure 4.1: Various timestamping scenarios in heterogeneous IoT systems. The source of timestamping uncertainties for distributed IoT devices will be different from each other.

It is possible for collecting real-time uncertainty information about each IoT device. By analyzing the potential impact of the uncertainty on the timestamping accuracy, proper synchronization strategies will be adopted for each IoT device, while each slave clock will be calibrated by utilizing the time information exchanged to achieve accurate and situation-aware time synchronization.

4.2.2 Timestamping Uncertainties Modeling

Electronic devices in IoT systems are typically equipped with a quartz crystal oscillator-driven clock to keep tracking of physical time. Due to the defective manufacturing and aging issue inherent to the oscillator, the clocks will continuously deviate from the ideal situation, which can be approximately modeled by a linear function, given by

\[ C_i(t) = \alpha_i t + \beta_i \] (4.1)
where $C_i(t)$ is the local clock value in IoT device $i$ affected by the local skew $\alpha_i$ and local offset $\beta_i$. Packet-switching-based time synchronization methods exclusively hinge on the accurate timestamp exchange between the local IoT clock and the reference clock, as shown in Fig. 4.2. Due to various uncertainties inherent to the highly complicated IoT system, the timestamp gathered and transmitted from the local IoT devices will be inaccurate compared to the ideal scenario, leading to degraded synchronization performance. The timestamping uncertainty induced by timestamp exchange can be generally categorized into the following three aspects.

- **Temperature-dependant Oscillator Drifting**

To limit the implementation cost of IoT systems, a simple packaged crystal oscillator (SPXO) is typically embedded in each local IoT device, which is inexpensive but unreliable in harsh and dynamic operating environments. The vulnerability of SPXO to external temperature variation will inevitably cause unstable clock skew. Therefore, the timestamp generated at the local IoT device will deviate from the ideal clock model given in (4.1). As validated in [39] and [42], external operating temperature has a quadratic impact on the oscillating frequency, which can lead to the corresponding clock skew variation given by

$$\alpha_i^T = \alpha_i + \eta T_d^2$$

where $T_d$ is the temperature difference between the ideal operating temperature and ideal scenario, while $\eta$ is the temperature sensitivity factor depends on the initial manufacturing. Due to this temperature-dependent oscillator drifting, inaccurate timestamps will be generated at the local IoT device, which will be dynamic due to the environmental variation.

- **Software Timestamping Error**

To share local time information between two connected IoT devices, proper timestamping techniques should be adopted during packet transmission and reception. Typically, there are two approaches to obtain timestamps, including hardware timestamping, where timestamps are recorded at the physical layer, and software timestamping. Due to hardware limitation, the
availability of software timestamping is much higher for low-cost IoT devices, with reduced timestamping accuracy as a drawback.

One of the key elements contributing to the software timestamping error is the variation of access time in drawing timestamps, which can be affected by the real-time CPU load and timestamping layer inside the operating system, e.g., in application or network layer. For the same IoT system, devices generally prefer to draw timestamps in a consistent layer, while CPU load will dominate the software timestamping error. For example, the ideal transmission time $t_{1,I}$ in Fig. 1 will be recorded with a timestamp $t_1$, given by

$$t_1 = t_{1,I} + \epsilon_i$$

where $\epsilon_i$ is the CPU load-induced timestamp jitter that can be modeled into a piecewise Gaussian distribution based on the real-time CPU usage, according to [43] and [44].

- **Communication-Induced Uncertainty**

Due to the complexity of IoT networks, packet exchange will be impeded due to asymmetric propagation latency and stochastic packet losses. Asymmetric propagation latency and packet delay variation will play significant roles in affecting the clock offset and skew estimation, respectively. As shown in Fig. 4.1, the offset estimation under communication uncertainty can be written as

$$\beta_i(\delta) = \frac{(t_2 - t_1) - (t_4 - t_3)}{2} - \frac{\delta_i}{2}$$

where $\delta_i$ indicates the packet delay variation between two successive packet transmission between the local device $i$ and the reference node. Similarly, skew estimation will be affected in the same amount by transmitting two unidirectional packets between the two devices. In this chapter, it is assumed that $\delta_i$ can be approximately estimated during packet exchange, e.g., by transmitting a series packets before time synchronization.

By jointly considering these three timestamping uncertainties, the modified clock model under heterogeneous source of error can be represented as

$$C_i(t) = (\alpha_i + \eta_i T_d^2)(t - \beta_i(\delta_i)) + \epsilon_i$$
where it can be observed that the multi-dimensional uncertainties, including temperature-dependent variation $\eta_i$, network uncertainty $\delta_i$, and CPU load-induced jitter $\epsilon_i$, will respectively contribute to the timestamping error to different extents, leading to biased timestamp generation. For large-scale IoT system with heterogeneous devices in terms of diverse operating environments, cross-standard communication standards, and various collaborative applications, the timestamping uncertainties will significantly deteriorate the overall synchronization performance.

Figure 4.2: Timestamping uncertainties inherent to the synchronization process due to network uncertainties, CPU jitter, and temperature variation.
4.3 Hybrid Time Synchronization Based on multi-source Uncertainty Modeling

Traditional synchronization methods are vulnerable to the heterogeneous timestamping uncertainties. Therefore, the goal of the proposed method is to design a situation-aware hybrid synchronization scheme that can accommodate the multifaceted operating conditions.

4.3.1 Time Information Exchange and Skew Estimation

The hybrid time synchronization can be achieve by multiple timing information exchange process as illustrated in Fig. 4.3. First, the synchronization procedure is initialized from the slave node by sending a \textit{Sync\_Request} packet with its current time \( t_1 \). Upon receiving the \textit{Sync\_Request}, the master node will reply with a \textit{Sync\_Response}, which embedded with two timestamps \( t_2 \) and \( t_3 \), then the slave node will record its receiving time as \( t_4 \). This round-trip timestamp packet exchange can provide a initial clock offset estimation, meanwhile each node can observe its state information including its interrupt handling delay and queuing delay by monitoring the current system load and traffic load during the transmission of the timestamped packet. However this initial estimation will be inevitably deteriorated by multi-source timestamp uncertainties, therefore two subsequent synchronization process will be performed with auxiliary timing information.

To address the synchronization inaccuracy incurred by software timestamping uncertainty, two consecutive packets, namely, \textit{Sync\_Report} and \textit{Report\_Followup}, will be firstly transmitted from from the slave node to the master node in skew estimation phase I. By defining a deterministic interval \( \tau_s \) known by both the slave and master node, the relative clock skew can be estimated at the master node by comparing the two packet reception instants denoted as \( t_5 \) and \( t_6 \), given by

\[
\alpha^k_1 = \frac{(t_6 - t_5)}{\tau_s}
\]  

(4.6)

It is worth noting that the transmission interval \( \tau_s \) can be pre-defined in the initialization stage so that no timestamps will be required at the slave node. As a result, the software timestamping uncertainty can be easily mitigated. Meanwhile, the clock skew can be regarded as invariant
due to the small transmission interval between the two successive instants.

Another significant issue leading to timestamping error is asymmetric propagation delay, which is mainly caused by nondeterministic communication conditions. Fortunately, this asymmetric delay can be avoided if the timestamps are transmitted unidirectionally. Based on this observation, a periodic beacon message will be broadcast from the master node in skew estimation phase II, which is robust against complicated network conditions and can be easily implemented. By recording the broadcasting interval $\tau_m$, the skew of the slave node compared to its master can be derived as

$$\alpha_2^k = \frac{\tau_m}{t_8 - t_7}$$  \hspace{1cm} (4.7)

where $t_7$ and $t_8$ are the timestamps recorded at the slave nodes. Compared to (4.6), skew estimated in (4.7) will be more reliable in highly complicated communication environments, at the cost of degraded performance with heavy CPU loads. Moreover, the estimated clock parameter in (4.6) can also be piggybacked in the beacon message to further calculate the clock skew.

---

**Table 4.2**: Synchronization Process

<table>
<thead>
<tr>
<th>Offset estimation:</th>
<th>$\beta^0 = \frac{t_2 - t_1 + t_3 - t_4}{2}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Skew estimation I:</td>
<td>$\alpha_1^k = \frac{t_6 - t_5}{\tau_s}$</td>
</tr>
<tr>
<td>Skew estimation II:</td>
<td>$\alpha_2^k = \frac{\tau_m}{t_8 - t_7}$</td>
</tr>
<tr>
<td>Skew estimation III:</td>
<td>$\alpha_3^k = \alpha_2^{k-1} + \eta(T - T_0)^2$</td>
</tr>
</tbody>
</table>

**Figure 4.3**: The timestamp exchange mechanism designed in the proposed hybrid time synchronization scheme, including initial offset estimation and three skew estimation methods.
4.3. Hybrid Time Synchronization Based on multi-source Uncertainty Modeling

In addition, the overall synchronization accuracy exclusively depends on the skew estimation consistency in one synchronization period. Therefore, the temperature-induced oscillator drifting also need to be considered to ensure the skew estimated is accurate and reliable. The temperature-related skew variation can be estimated based on the quadratic relationship between the operating temperature and the clock skew [45] shown as (4.2), where the ideal operating temperature and temperature sensitivity coefficient of the slave node can be considered as known for most of the IoT devices. In the case without sufficient manufacturing specifications, the critical parameters can be observed by estimating the clock skew under different operating temperatures [39].

4.3.2 Hybrid Time Synchronization

Given the highly complicated IoT environments with multi-source timestamping uncertainties, jointly utilizing different skew estimation methods can significantly enhance the synchronization performance under heterogeneous scenarios. The core idea of the proposed hybrid synchronization method is to analyze the real-time multi-source timestamping error and assign proper synchronization strategy accordingly.

Based on this consideration, three adaptive weighting factor will be assigned to each node for the integrated estimation of each clock parameters to achieve an optimal synchronization performance. Specifically, each weighting factor serves as the reliability indicator of the skew estimation phase under multiple uncertainties, given by

\[
\mathbf{w}^k = \begin{bmatrix} w_{k1}^k & w_{k2}^k & w_{k3}^k \end{bmatrix}
\]  

(4.8)

which can be obtained based on the timestamping error induced the real-time operating conditions written as

\[
\mathbf{e}^k = \begin{bmatrix} e_{CL}^k & e_{NL}^k & e_T^k \end{bmatrix}
\]  

(4.9)

where \( e_{CL}^k \), \( e_{NL}^k \), and \( e_T^k \) are the timestamping errors induced by software uncertainty, communication uncertainty, and temperature-induced drifting, respectively. The uncertainties are estimated from the previous synchronization processes, since for each skew estimation phase,
the synchronization error will be mainly affected by one specific environmental uncertainty, which can guarantee the error observation accuracy.

Therefore, the weighting factors can be calculated as the proportion of each synchronization error compared with the total clock error in one synchronization period, given by

$$w_k(i) = \frac{e_k(i)}{\sum e_k(i)}$$

(4.10)

Therefore, the clock skew can be jointly estimated based on the three successive skew estimation phases, given by,

$$\hat{\alpha}_k = w_1\alpha^k_1 + w_2\alpha^k_2 - w_3\eta_i(T - T_0)^2$$

(4.11)

which can be used to calibrate the clock error for each device.

### 4.4 Performance Evaluation

In this section, the proposed hybrid time synchronization scheme is evaluated from two aspects. Firstly, the modeling of multi-source timestamping uncertainties is evaluated. Moreover, the synchronization accuracy with heterogeneous operating conditions will be given by comparing the proposed hybrid scheme to other existing techniques.

#### 4.4.1 Uncertainty Modeling Accuracy

Timestamping uncertainty modeling lays foundation for the proposed hybrid synchronization scheme. In this simulation, the timestamping uncertainty is affected by three different conditions, namely, operating temperature, communication variation, and CPU load. Specifically, the temperature will change from $-10^oC$ to $70^oC$ for environmental variations. Moreover, the communication uncertainty at each IoT device $i$ during synchronization is simulated as a Gaussian distribution with $\delta_i \sim (50, 5)$, while two different CPU loads (50% and 100%) are considered.

The modeling accuracy of CPU-induced timestamping error is demonstrated in Fig. 5.5, while the results for the other two factors are omitted here to avoid redundancy. It can be
Figure 4.4: The modeling accuracy of CPU-induced timestamping uncertainties before hybrid time synchronization.
Figure 4.5: Overall synchronization error.

Figure 4.6: Performance different temperatures.
Figure 4.7: Performance with network uncertainties.
observed that the synchronization accuracy will be affected by the variation of operating conditions due to the induced timestamping error. From Fig. 5.5, the estimation of the timestamping uncertainty will follow the proposed models, which can support the successive hybrid synchronization scheme.

### 4.4.2 Synchronization Accuracy in Heterogeneous Scenarios

Based on the accurate timestamping uncertainty modeling, hybrid time synchronization can be designed accordingly. In this simulation, a total number of 30 IoT devices are deployed in a heterogeneous IoT system, which are assigned with different operating environments, communication conditions, and application requirements. A scenario of the synchronization with all three uncertainties is shown in Fig. 4.5, where the timestamping error induced will lead to different synchronization errors. However, the proposed hybrid method can achieve much stable synchronization accuracy compared to the other two existing solutions, including packet-switching-based and one-way beacon-based methods [46].

Furthermore, the proposed approach can achieve accurate clock calibration in different and time-varying operating scenarios. The averaged synchronization accuracy throughout the network is demonstrated in Fig. 4.6 and Fig. 4.7, by considering the variation of temperature and network uncertainty. Therefore, the proposed method will be more advantageous in highly complicated and heterogeneous operating environments.

### 4.5 Conclusion

In this chapter, a situation-aware hybrid time synchronization protocol is designed based on multi-source timestamping uncertainty modeling and integrated time information exchange for heterogeneous IoT systems. More specifically, the multi-faceted operating conditions inherent to the overall synchronization process are accurately modeled to explore their impact on the timestamping accuracy. By analyzing the real-time timestamping uncertainties, a hybrid time synchronization scheme is achieved, which can help achieve optimal synchronization strategy for clock parameters estimation. In addition, an integrated time information exchange mechanism is designed to reduce timestamping redundancy during time synchronization. Simulation
results show that the proposed scheme can achieve accurate timestamping error modeling and enhance the synchronization accuracy for heterogeneous operating scenarios.
Chapter 5

Low-Overhead and Reliable Data Synchronization

5.1 Introduction

A pre-scheduled task period enabled data synchronization scheme is proposed in this chapter to achieve accurate, efficient, and reliable synchronization for distributed data in edge-enabled IoT systems. The main contributions of this chapter are threefold, including the pre-scheduling of task periods for IoT devices in achieving timestamp-efficient data synchronization, the design of the differential timestamping mechanism for efficiency enhancement, as well as two unreliability detection schemes enabled by scheduled data transmission and timestamp verification. Highlights of this chapter can be specifically summarized from the following three aspects:

A novel data synchronization scheme is proposed by pre-scheduling each IoT sensor for an adaptive task period (TP), which is defined as the time allocated for the sensing task at each device. By comparing the TP predefined and the information observed at the edge device, the non-ideal characteristics of each local clock can be estimated with significantly reduced timestamp exchange within the network. Distributed data collected from local IoT devices can be accurately aligned at the remote processing center without frequently synchronizing the local clocks. Compared to our previous work [47], in this chapter, adaptive TP scheduling for each IoT device is achieved by edge-enabled group-wise coordination. Several critical
design specifications, including timeliness of application data, clock skew consistency, data transmission duration, and holistic schedulability of all involved devices, are accomplished for a more organized and optimal TP design.

Moreover, an efficient and compact differential timestamping mechanism (DTS) is designed in this chapter to enhance the overall synchronization efficiency substantially. By leveraging the previous time information recorded at both the edge and local devices, a partial timestamp containing only differential information that changed from the known information will be uploaded from each IoT device. By adopting DTS, timestamps with unnecessarily lengthy digits can be pruned without sacrificing the value of timestamps, which is expected to minimize the network overhead induced during data synchronization.

Furthermore, a dual unreliability detection mechanism is designed to avoid using uncertain timestamps during the proposed data synchronization method considering the vulnerability of
IoT devices. On the one hand, given the pre-scheduled TP, IoT data and the occasionally associated timestamps are transmitted in deterministic time slots. Any unexpected information delivered beyond the schedule will be potentially unreliable and will not be processed by the remote device. On the other hand, a verification digit is retained in the DTS, which can help to validate the correctness of the time information. The combination of these two mechanisms can effectively filter unreliable timestamps and ensure data synchronization performance under uncertain operating conditions.

5.2 System Model

In this section, the sampling model for each IoT device and the misalignment among distributed data induced by drifting clocks will be introduced in detail.

5.2.1 System model

This chapter studies edge-enabled IoT systems, shown in Fig. 5.1, where a wide variety of \( N \) distributed IoT sensors \( S = \{s_1, ..., s_N\} \) are assigned to diverse sensing tasks. Time-series IoT data collected from distributed locations will be transmitted to the cloud center for centralized decision-making with the help of \( M \) edge devices \( E = \{e_1, ..., e_M\} \), including smart gateway and vehicles. Edge-based data fusion is critical to avoid data overload in the cloud center.

Serving as the sink node for local IoT sensors, edge devices are assumed to have sufficient processing capability and accurate clocks, i.e, \( C_e(t) = t \). By contrast, the clock of the local IoT sensor \( s_i \) will deviate from the ideal time due to the embedded inexpensive crystal oscillator. Considering the first order affine model that has been widely recognized in the literature [48],[49], the drifted clock generation at instant \( t \) can be written as

\[
C_i(t) = (1 + \alpha_i)t + \beta_i, \tag{5.1}
\]

where \( \alpha_i \) and \( \beta_i \) represent the clock skew and offset, respectively. It is worth noting that, although \( \alpha_i \) for each node is typically small (around 1ppm), it will affect the clock ratio over time and continuously contribute to the non-negligible clock deterioration, which requires periodic
packet exchange to achieve time synchronization.

The data transmission between the sensor \( s_i \) and its edge device will encounter non-deterministic end-to-end latency \( \tau_i \), which is the accumulation of propagation delay \( \tau^p_i \), data transmission delay \( \tau^d_i \), network access delay \( \tau^a_i \), and queuing delay \( \tau^q_i \), written as

\[
\tau_i = \tau^p_i + \tau^d_i + \tau^a_i + \tau^q_i.
\]  

(5.2)

Despite dominating by the distance between \( s_i \) and its edge device, propagation delay \( \tau^p_i \) is also affected by hardware specifications and communication environments. It is assumed that \( \tau^p_i \) will follow a Gaussian distribution, i.e., \( \tau^p_i \sim \mathcal{N}(\mu_i, \sigma^2_i) \), where \( \mu_i \) and \( \sigma_i \) are the mean and standard deviations of the propagation delay, respectively. Due to the lack of a temporal consensus, parameters of \( \tau^p_i \) cannot be accurately measured by simply recording the transmission and reception instants of one packet.

On the other hand, data transmission delay \( \tau^d_i \) is more relevant to the data to be transmitted, expressed by

\[
\tau^d_i = \frac{D_i}{r_i},
\]

(5.3)

where \( D_i \) is the data size of the packet from sensor \( s_i \) and \( r_i \) is the data transmission rate, which can be further given by

\[
r_i = B W_i \log_2(1 + \gamma_i)
\]

(5.4)

according to Shannon’s theory, where \( B W_i \) is the channel bandwidth. The signal to noise ratio \( \gamma_i \) can be given by \( \gamma_i = \frac{P_0 h_i}{\theta_i^2} \), where \( P_0 \), \( h_i \), and \( \theta_i^2 \) are transmission power, channel gain, and noise power of \( i \), respectively. Network access and queuing delays will be negligible in the proposed scheme, thereby omitted here.

### 5.2.2 Data Sampling Model

In this study, a large number of inexpensive sensor nodes \( n_i \) with drifted clocks \( C_i, i \in \{1, 2, ..., N\} \) are densely deployed to support different IoT applications. The sampling instant of each sensor will be triggered locally by its own clock. Meanwhile, the time instant of the sampling \( j \) at node \( n \) will be stamped together with the data observation, denoting as \( (s_{ij}, TS_{ij}) \). As
a result, the physical time of each measurement can be recorded sequentially. However, the excessive amount of data inherent to large-scale IoT networks and high sampling frequency will inevitably pose an overwhelming burden on the network resources and lead to significant network access contention. A joint consideration between the task fulfillment and the sampling capacity for each sensor device is necessary.

In the presence of limited communication resources, local IoT sensors are prohibited to transmit the sampled data to the edge device immediately after its generation. Instead, the transmission process will be triggered after a predefined task period (TP) \( p_i \) is elapsed. In other words, the sampled data will wait in the local data register until the given TP, as demonstrated in Fig. 5.2. TP is defined as the time interval between the first sampling instant and the transmission instant, which is scheduled by the edge device. After \( p_i \), the recorded sensing data \( s_{ij} \) and the timestamp associated with the first sampling \( TS_{i1} \) will be delivered from the sensor node \( i \), represented as

\[
s_i = \{s_{i1}, TS_{i1}\}, s_{ij}, j \in \{1, 2, ..., m\},
\]

where \( m \) is the total number of data sampled at node \( i \). After receiving the local data from each IoT sensor, the edge device will recover the timestamp for each data sample, given by

\[
TS_{ij} = TS_{i1} + \frac{j-1}{f_i}, j \in \{1, 2, ..., m\}.
\]

Ideally, \( f_i \) is the fixed sampling frequency of node \( i \) known by the edge device. However, drifted clocks will lead to biased interval between two samples, which will be discussed in the

![Buffering](image)

Figure 5.2: During the task period of node \( i \), a series of timestamped data will be generated and recorded, which will not be transmitted to the sink node until the prescheduled transmission point.
5.2. System Model

Figure 5.3: The illustration of the proposed TP-enabled data synchronization scheme. (a). Data transmission schedule between each IoT node and the edge device during one data synchronization cycle, including one network initialization period and a few TPs followed by data transmission. (b). The information exchange between IoT nodes and the edge device during network initialization, aiming at estimating network condition and initial clock parameters. (c). The simplified network architecture with a few edge-formed groups. The interactions between the edge device and its subordinates include bidirectional information exchange during network initialization period and unidirectional data upload for data synchronization afterward.

Section III-B during timestamp recovery. Unknown sampling rates can be straightforwardly estimated by generating a few data samples at each IoT device and observing the averaged sampling interval among them. By transmitting only the first data sample of each sensor, overall network resource consumption is expected to reduce without information missing.

5.2.3 Clock-Induced Data Misalignment

The recovered data information \( \{s_i, TS_i\} \) from each sensor should be coherently aligned with each other for accurate data processing. However, timestamp error will be inevitably induced by the low-cost clock embedded in each sensor device, especially in inhospitable operating environments due to its susceptibility to external variations.

As indicated in (5.1), the timestamp generated by distorted clocks cannot reflect the real instant of the real sampled observations. Compared to the standard time at the edge device, the
clock distortion-induced timestamp error for the $l$th sampling can be expressed as

$$
\epsilon_i(l) = TS_{e,l} - TS_{i,l} = \frac{l}{f_i} - \left(\frac{l}{f_i} + \beta_i\right) = \frac{l(1 - \alpha_i)}{f_i} - \beta_i,
$$

(5.7)

where $\epsilon_i$ can serve as a specific indicator during collaborative data processing illustrating the clock-induced temporal misalignment among distributed data.

To specify the impact of timestamp error, we consider a sensor fusion method referred to as inverse-variance weighting (IVW), which is widely adopted for fusing distributed sensing data[50], [51]. Given a series of data samples $s_i(C_i(t_j)), i \in \{1, 2, ..., N\}$ generated by $N$ sensors for the $l$th sampling instant $t_l$, the fused data after utilizing IVW can be written as

$$
s(t_l) = \frac{\sum_i s_i(C_i(t_l))/\xi^2_i}{\sum_i 1/\xi^2_i},
$$

(5.8)

where $\xi^2_i$ is the variation of data sampled in node $i$. Due to the timestamp error given in (5.1), the output of IVM will deviate from its ideal case, written as

$$
\hat{s}(t_l) = \frac{\sum_i s_i(C_i(t_l - \epsilon_i))/\xi^2_i}{\sum_i 1/\xi^2_i},
$$

(5.9)

which will clearly lead to a fusion error given by

$$
e(l) = s(t_l) - \hat{s}(t_l) = \frac{1}{\sum_i 1/\xi^2_i} \sum_i \frac{s_i(C_i(t_l - \epsilon_i)) - s_i(C_i(t_l))}{\xi^2_i}.
$$

(5.10)

It can be observed that the fusion error will be only related to the temporal misalignment caused by $\epsilon_i$, while the variance of data $\hat{\xi}_i^2$ will remain unchanged. Moreover, since timestamps are locally generated at IoT sensors and only reflect the instant of each data sample, network-induced delays will not affect the fusion accuracy.
5.3 Task Period Enabled Data Synchronization

In this section, a task period-enabled data synchronization scheme (TADAS) is designed to address the temporal misalignment during centralized data processing for resource-constrained IoT sensors. In Fig. 4.3, the overall data transmission and simplified network architecture for the proposed TADAS scheme are demonstrated. Unlike traditional time synchronization protocols that rely exclusively on excessive timestamp exchange to correct timestamp errors caused by non-ideal clocks, TADAS aims to achieve accurate temporal consistency among distributed IoT data with significantly reduced network overhead. As illustrated in Algorithm 1, TADAS mainly comprises two successive phases, namely, network initialization and data synchronization.

5.3.1 Network Initialization

In TADAS, the overall transmission process of one data synchronization cycle, as shown in Fig. 4.3(a), consists of the network initialization period (NI) and a few data synchronization processes (DS), while the latter one can be further divided into TP and data transmission. The main goal of NI is to estimate the network conditions and initial clock parameters to support the succeeding data synchronization design.

The information exchange in NI phase contains two main processes, namely, delay estimation and offset calculation, shown as Fig. 4.3(b). On the one hand, each IoT sensor will transmit a series of packets to the edge device to estimate the latency and its variation between the two devices. For the $k^{th}$ packet transmission of sensor $i$, the transmission and reception instants will be given as $T_{0-k,Tx}^i$ and $T_{0-k,Rx}^i$, respectively. The use of 0− aims to indicate the transmissions are conducted before data synchronization. During packet transmission, the local configuration of each sensor, including its sampling frequency $f_i$, buffer size $b_i$, and time sensitivity (deadline) of its associated data $dl_i$ will be reported piggyback. Given the packet delay of the $k^{th}$ packet transmission as $T_{0-k,Rx}^i - T_{0-k,Tx}^i$, the variation of propagation latency
can be obtained by
\[ q_i = \frac{1}{K-1} \sum_{k=2}^{K} \left| (T_{0-k,Rx}^i - T_{0-k,Tx}^i) - (T_{0-k-1,Rx}^i - T_{0-k-1,Tx}^i) \right|, \]  
\[ (5.11) \]
where \( K \) is the total number of packets transmitted from one sensor node during the delay estimation period.

On the other hand, after delay variation estimation, the initial clock parameters will be calculated by adopting precision time protocol (PTP)-based bidirectional packet exchange. By exchanging a pair of packets between each sensor and the edge device, the initial clock offset \( \beta_0^i \) and the network latency \( \tau_i \) can be respectively represented as
\[
\begin{align*}
\beta_0^i &= \frac{1}{2} \left( (T_{0,2}^i - T_{0,1}^i) - (T_{0,4}^i - T_{0,3}^i) \right) \pm q_i, \\
\tau_i &= \frac{1}{2} \left( (T_{0,2}^i - T_{0,1}^i) + (T_{0,4}^i - T_{0,3}^i) \right) \mp q_i.
\end{align*}
\[ (5.12) \]
The involvement of delay variation \( q_i \) indicates the confidence interval of the clock parameter estimation. After estimating delays and offsets, the length of TP and data transmission instants will be determined for each sensor according to its intrinsic characteristics. In the end, the edge device will broadcast determined design specifications back to all its subordinates to finish the NI phase.

It is worth noting that the network latency estimation and task period assignment achieved in one NI phase will become unreliable over time due to operational dynamics, e.g., network topology variation and unstable clock skew caused by operating temperature, necessitating periodic re-initialization for each sensor node. The detailed definition and optimization of the task period as well as re-initialization considerations will be designed based on network and clock conditions, as given in Section IV.

### 5.3.2 Data Synchronization

After network initialization, each IoT sensor will start to generate local samples at its specific sampling frequency. The temporal correlation among distributed data will be disordered due to local clock errors, which require careful synchronization before further processing. In TADAS,
5.3. Task Period Enabled Data Synchronization

Algorithm 2: Timestamp Calibration based on the Estimation of Clock Parameters

**Network initialization:**
Each sensor node $i = 1 : n$
1. Transmit a series of packets
2. Piggyback local configuration information
3. Estimate delay variation $q_i$ based on Eq. (5.11)
4. Calculate the propagation delay $\tau_i^p$ and initial offset $\beta_i^0$ based on Eq. (5.12)
5. Define the task scheduling period $p_i$

**Data synchronization:**
Each task period $j = 1 : \#p_i$
Each sensor node $i = 1 : n$
1. Record timestamp for the first sample $TS_i,1$
2. Recover each timestamp according to Eq. (5.6)
3. Calculate $\alpha_i$ and $\beta_i$ based on Eq. (5.18) and Eq. (5.19)
4. Calibrate $\hat{TS}_j^i$ according to Eq. (5.20)

Figure 5.4: Group scheduling of Edge based data transmission

the predefined TP will serve as the time reference to achieve holistic temporal consistency.

As previously introduced in Section II-A, for IoT sensor $i$, a series of local samples will be generated starting at time $t_{i,1}$, which will be recorded as $TS_{i,1}$ at the local device. All data sampled generated thereafter will be stored in the local register until TP is elapsed, leading to the transmission instant given by

$$t_{i,Tx} = TS_{i,1} + p_i$$

with a timestamp $TS_{i,Tx}^i$ at the transmitter side. Once the data packets from sensor $i$ are received at the edge device $e$, it will record the reception instant $TS_{i,Rx}^e$. Looking at one specific device, the difference between the transmission and reception instants will be dominated by the end-
to-end propagation delay $\tau_i$, given by

$$TS_{i,Rx}^e = TS_{i,Tx}^e + \tau_i, \quad (5.14)$$

which is obtained according to the clock at the edge device, and

$$TS_{i,Rx}^i = TS_{i,Tx}^i + \tau_i \quad (5.15)$$

for the clock at sensor $i$. By observing the clock relation between sensor $i$ and edge $e$ given in 5.1, we can straightforwardly obtain

$$TS_{i,Tx}^i = (1 + \alpha_i)TS_{i,Tx}^e + \beta_i \quad (5.16)$$

describing the clock deviation of sensor $i$, and

$$TS_{i,Tx}^i = TS_{i,1}^i + (1 + \alpha_i)p_i, \quad (5.17)$$

which shows the evolution of the clock at sensor $i$ after one TP. Clearly, by combining (5.14), (5.16), and (5.17), the clock skew at sensor $i$ of the $P^{th}$ TP can be calculated as

$$\alpha_i^P = \frac{TS_{i,1,p} - \beta_{i-1}^P}{TS_{i,Rx,p}^e - \tau_i - p_i}, \quad (5.18)$$

where $TS_{i,Rx,p}^e$ and $TS_{i,1,p}^i$ are initial sampling timestamp at sensor $i$ and the reception timestamp at edge $e$, which are known for the edge device. $\beta_{i-1}^P$ is the accumulated clock offset after $P-1$ TPs, which can be generalized as

$$\beta_i^P = \begin{cases} \sum_{r=1}^{P-1} \alpha_r^i p_i, & P > 0, \\ \beta_i^0, & P = 0. \end{cases} \quad (5.19)$$

$\beta_i^0$ and $\tau_i$ can be obtained by (5.9).

Based on the clock parameters of each node estimated by SN, the distorted timestamps can be calibrated accurately for each task period. The reconstruction of the data sampling instant
is given by

\[ T^S_{i,k} = \frac{TS^j_{i,k} - T^j_{i,0} - \beta'^{j-1}_i}{\alpha'_i} + T^j_{i,0}, \]

(5.20)

where \( TS^j_{i,k} \) indicates the time information associated with the corresponding data. The calibrated timestamps will be updated and recorded for further data fusion and processing.

As shown in Fig. 4.3(c), unlike network initialization where packet exchange is required between each node and the edge device, data synchronization is achieved by unidirectional timestamp and data upload from each IoT sensor to the edge device. Meanwhile, only one timestamp is required to be transmitted, which can enhance the synchronization efficiency substantially. It should be noted that, due to the unstable oscillation frequency inherent to low-cost clocks leading to time-varying clock drifts, data synchronization should be conducted periodically after each task period.

### 5.4 Performance Evaluation

In this section, the proposed task period based data synchronization scheme is evaluated from two perspectives. Firstly, the effectiveness of the proposed scheme is evaluated by comparing the data fusion performance after adopting the data synchronization scheme. Moreover, the network overhead incurred by temporal alignment is demonstrated and compared in detail among three different synchronization techniques.

#### 5.4.1 Data Fusion Accuracy

In this simulation, a total number of 30 sensor devices are randomly deployed in an IoT system for environmental condition monitoring. The sensed data will be collected and fused in SN for different applications. The acceptable data fusion accuracy is set to be in microsecond level, which is a general requirement in supporting time-sensitive applications.

Due to the temporal inconsistency among the distributed devices, the collected data will be mismatched in temporal domain as a consequence. As shown in Fig. 5.5, the accumulated timestamp error without adopting data synchronization will increase significantly with time, meaning that the data processing accuracy is intolerable. By contrast, after adopting the pro-
Figure 5.5: The accumulated timestamp error associated with the data to be fused in the sink node. The timestamp accuracy can be significantly enhanced by adopting the proposed data synchronization scheme, which can help to remain the overall error to be lower than the application-oriented requirement.
5.4. Performance Evaluation

Figure 5.6: The comparison of network overhead induced by temporal alignment processes. Because of avoiding frequent timestamp exchanges, the proposed task period-based method can achieve accurate synchronization with the least network resource consumption.

5.4.2 Network Overhead Analysis

On the other hand, achieving accurate temporal alignment among the massive data will inevitably pose additional burden on the network resources. Adopting synchronization schemes with excessive network resource consumption will lead to high network overhead and occupy
resources of critical applications. The overall performance of the time-sensitive applications will be suspicious, especially in resource-constrained scenarios. As a result, analyzing the resource consumption during synchronization is of the utmost importance.

In this simulation, the network overhead caused by the temporal alignment is defined as the total number of timestamps required during synchronization. A comparison among the proposed sampling-period-based data synchronization, traditional timestamp-based data synchronization [35], and packet-switching-based clock synchronization, i.e., PTP, is conducted. As shown in Fig. 5.6, we can observe that the lowest network overhead is incurred by adopting the proposed scheme since only a few timestamps are required to be exchanged between SN and each sensor node for clock information sharing. By contrast, the network resources required by traditional synchronization methods will be much higher to calibrate the unstable clocks. Furthermore, the extremely low accumulated network overhead of the proposed scheme during the long-term operation will lead to a significant benefit for network resources saving, meaning that it is a promising candidate to support resource-constrained applications.

5.5 Chapter Summary

In this chapter, a pre-scheduled task period enabled data synchronization scheme is proposed to achieve accurate, efficient, and reliable synchronization for distributed data in edge-enabled IoT systems. The main contributions of this chapter are threefold, including the pre-scheduling of task periods for IoT devices in achieving timestamp-efficient data synchronization, the design of the differential timestamping mechanism for efficiency enhancement, as well as two unreliability detection schemes enabled by scheduled data transmission and timestamp verification. Simulation results demonstrated that the proposed method can help to significantly enhance the data fusion performance with the presence of inaccurate clocks. Moreover, the induced network overhead is dramatically reduced during clock parameter estimation, as compared to the traditional timestamp-based synchronization approaches.
Chapter 6

Conclusion and Future Work

6.1 Conclusion

In this thesis, a novel prescheduled task period-based data synchronization method have been proposed to achieve accurate temporal alignment among the distributed data before data fusion and processing in supporting time-sensitive IoT applications. More specifically, by comparing the predefined task period of each sensor node, the real observed information at SN will be used for clock skew and offset estimation. By compensating the clock inaccuracy only through comparing the task periods, temporal alignment with extremely low network overhead is achieved. Simulation results demonstrated that the proposed method can help to significantly enhance the data fusion performance with the presence of inaccurate clocks. Moreover, the induced network overhead is dramatically reduced during clock parameter estimation, as compared to the traditional timestamp-based synchronization approaches.

Moreover, a situation-aware hybrid time synchronization protocol is designed based on multi-source timestamping uncertainty modeling and integrated time information exchange for heterogeneous IoT systems. More specifically, the multi-faceted operating conditions inherent to the overall synchronization process are accurately modeled to explore their impact on the timestamping accuracy. By analyzing the real-time timestamping uncertainties, a hybrid time synchronization scheme is achieved, which can help achieve optimal synchronization strategy for clock parameters estimation. In addition, an integrated time information exchange mechanism is designed to reduce timestamping redundancy during time synchronization. Simulation
results show that the proposed scheme can achieve accurate timestamping error modeling and enhance the synchronization accuracy for heterogeneous operating scenarios.

6.2 Future Work

The thesis solves the technical challenges of distributed clock synchronization in time-sensitive IoT systems, using a series of low-overhead situation-aware synchronization schemes to enhance the synchronization performance from various aspects. To further improve the performance of the overall IoT system as well as the clock synchronization, there are still many more issues that need to be looked at and solved. While some of the potential future study areas are recognized and described in this part, the current research proposed in this thesis can be expanded in several ways.

- Recently, the timeliness of information update, in terms of age of information (AoI), is becoming an important performance metric for real-time systems and wireless networks. AoI, which is defined as the elapsed time after information was created, has been presented as a crucial criteria to assess the freshness of information updates, and it has been demonstrated that there is an ideal update rate for status-update systems with constrained network resources. Numerous works have been created thus far for keeping information current in a variety of applications, including age-optimal link scheduling, transmission scheduling for energy-harvesting sensors, and optimum cache updates for mobile caching. The implementation of AoI in resource-constrained IoT systems is particularly difficult because of the inaccurate timekeeping caused by cheap clock sources. Existing AoI studies assume perfect timing and synchronization across all nodes and treat the rate of age change as linear. It’s important to schedule events correctly since the information age is a measure of time. The effects of synchronization problems on reducing age have not yet received much attention, despite the fact that non-linear age and an age penalty function were taken into consideration to address the non-linearity of age evolution. Due to incompatibilities between the clocks of nearby devices, age considerations in practise lead to age inaccuracy. Age mistakes that are unique to each node are also a result of the heterogeneity of clock skew and offset in each device. The
heterogeneity of synchronization demand at various nodes is ignored by the majority of clock synchronization protocols, which typically distribute synchronization messages on a periodic basis throughout the network.

- In some emerging applications, node may collaborate and interact in more flexibly way such as an UAV network or ad-hoc network, where node can move and be free to dynamically associate and share temporal information with any other devices in their communication range. Future extension of this thesis could be focused on building a mutual understanding of the concurrent events with data alignment and association system in a self-organized way when considering the devices’ mobility and with no fixed infrastructure for the network deployment.
Bibliography


Curriculum Vitae

Name: Haide Wang

Post-Secondary Education and Degrees:
2018 - present, M.E.Sc
Electrical and Computer Engineering
The University of Western Ontario
London, Ontario, Canada

Related Work Experience:
Teaching Assistant
The University of Western Ontario
2018 - Present

Research Assistant
The University of Western Ontario
2018-Present

Publications: