Future Generation Computer Systems 99 (2019) 1–10

Contents lists available at ScienceDirect

Future Generation Computer Systems

journal homepage: www.elsevier.com/locate/fgcs

Time-slotted software-defined Industrial Ethernet for real-time Quality of Service in Industry 4.0

Peng Zeng [a], Zhaowei Wang [a,b,*], Zhengyi Jia [c], Linghe Kong [d], Dong Li [a], Xi Jin [a]

[a] Key Laboratory of Networked Control System, Shenyang Institute of Automation, Chinese Academy of Sciences, Shenyang 110016, China
[b] University of Chinese Academy of Sciences, Beijing 100049, China
[c] National Network New Media Engineering Technology Research Center, Institute of Acoustics, Chinese Academy of Sciences, Beijing 100190, China
[d] Department of Computer Science and Engineering, Shanghai Jiao Tong University, Shanghai 200240, China

HIGHLIGHTS

- Software defined network is an appropriate choice to establish a RT communication system in Industry 4.0.
- Time slot based packet switching policy guarantees the forwarding delay of RT data.
- Time synchronization is the foundation to realize time slot based packet scheduling.
- Precision Time Protocol based link compensation can achieve time synchronization at microsecond level.

ARTICLE INFO

Article history:
Received 11 May 2018
Received in revised form 26 February 2019
Accepted 3 April 2019
Available online 9 April 2019

Keywords:
Industrial Ethernet
Time synchronization
Software defined networking
Real-time
Quality of Service
Industry 4.0

ABSTRACT

Reliable and timely data delivery, namely, real-time communication, is a key aspect for achieving the agile manufacturing goals of Industry 4.0. Moreover, reducing the maintenance, infrastructure and reconfiguration costs, and increasing the flexibility are general goals for communication systems in Industry 4.0. Although some works have extended the Ethernet standard IEEE 802.3 to satisfy the real-time requirements of industrial automation systems, the existing works have one or more of the following drawbacks: poor scalability, insufficient self-configuration capabilities, or increased costs due to their use of proprietary hardware. In this paper, we propose a Time-slotted Software-defined Industrial Ethernet (TS-SDIE) for real-time Quality of Service (QoS) in Industry 4.0. The paper addresses two key technologies of TS-SDIE, time synchronization and time slot-based packet switching. First, we develop a precision time protocol-based link delay compensation mechanism to handle the delay variance in networks. Precise synchronization is achieved at the microsecond level. Second, we design a system architecture for time slot-based industrial switches with four modules responsible for time synchronization, flow classification, time slot assignment, and packet forwarding, respectively. The experimental results show that the switch delay and its jitter can be reduced to 12.2 μs and 0.12 μs, respectively, for packets with a size of 100 B, which satisfies the requirements of time-sensitive networking for industrial automation systems.

1. Introduction

Recently, with the development of advanced industrial technologies such as Cyber–Physical Systems (CPS) and the Industrial Internet of Things (IIoT), Industry 4.0 was proposed to realize intelligent manufacturing and smart factories [1,2]. The ultimate goals of Industry 4.0 are to improve the level of automation and production quality and to be able to meet the customized demands from customers in shorter time intervals while using less energy consumption and fewer resources [3]. To implement Industry 4.0, different production systems should be interconnected to construct a distributed network horizontally, which increases the scale and complexity of industrial automation system sharply [1]. So, the trend of flexibility, reconfiguration, and maintainability in industrial networks is unalterable. In addition, real time and reliability are two key requirements in industrial applications [3,4]. For example, many industrial applications (such as machine control, power generation, process control and transportation) require timely message delivery, namely, real time transmission. In the welding process of automobile production line, symmetrical welding of a car body is effective to minimize deformation. In the drilling and riveting process, two robots work together to finish the process of drilling, sending

* Corresponding author at: Key Laboratory of Networked Control System, Shenyang Institute of Automation, Chinese Academy of Sciences, Shenyang 110016, China.
E-mail address: wangzhaowei@sia.cn (Z. Wang).

https://doi.org/10.1016/j.future.2019.04.009
0167-739X/

nail, and riveting. The cooperative operation of these robots is based on the command from the controller. At the moment, the command transmission should satisfy the real time requirement. Usually, the real time requirements of motion control systems in automobile production lines are less than 1 ms and those of process control systems in petrochemical fields are below 100ms [4]. Moreover, unreliable communication of sensor data or control signal from production facilities would cause unpredictable consequences. Hence, the main problem in Industry 4.0 is to establish a flexible, reliable real-time (RT) communication system that requires less maintenance, less infrastructure, and can be reconfiguration at less cost.

At present, Ethernet is widely used in industry as Ethernet can connect all levels of a factory, which provides support for systems interconnection in Industry 4.0. However, current Ethernet systems cannot satisfy the requirements of RT communication system in Industry 4.0. Thus far, no real-time support is available in standard IEEE Ethernet. This lack has led to the use of a number of proprietary Ethernet modifications in industrial systems such as Profinet, EtherCAT, Powerlink, and Ethernet/IP [4]. Nevertheless, the existing Industrial Ethernet (IE) systems have one or more drawbacks [5]. For example, almost all IE systems require special hardware and have millisecond delivery time; Profinet, EtherCAT, and Powerlink have single point of failure; Powerlink has poor scalability; and Profinet and Ethernet/IP have insufficient self-configuration capabilities. Currently, IEEE Time-Sensitive Networking (TSN) [6] provides deterministic services for industrial automation systems. Although TSN defines some RT functions such as guaranteed message latency with a jitter less than 1 μs, coexistence of real-time and non-real-time traffic, and shared network infrastructure through higher-layer protocols, it is still in being standardized. Moreover, TSN focuses only on the RT demands of Industry 4.0.

Software-defined Networking (SDN) is a new network paradigm for improving network performance by augmenting network management, control and data handling capabilities [2,7–9]. SDN's features make it possible to gain a globally oriented view of network management and control, which has avoided the best effort and locally optimal transmission under distributed control. Meanwhile, fine-grained data stream forwarding rules govern the information flow from physical ports to the application layers, which improve the accuracy of data transmission. Furthermore, the unified data-forwarding plane makes SDN-based networks scalable and the infrastructure shareable [1]. Therefore, SDN is an appropriate choice for establishing an RT communication system for Industry 4.0.

Although SDN has been considered for industrial automation networks in recent works, most of these studies analyze only the applicability of SDN [10–13]. Thus, the following challenges still exist in SDN based IE systems.

- Many private networks intended for different applications exist in industrial fields, e.g., video surveillance, device control, and state awareness. A single controller makes it difficult to meet the sharply increasing computing requirements for managing and controlling multiple data streams crossing network domains [5].
- The centralized control logic provided by SDN is effective for managing real-time services in IE, but the time delay caused by the competition mechanism in traditional Ethernet still exists [3,14], and has a huge impact on hard real-time requirements.
- When the intend is to run different applications on shared infrastructures, it is difficult to concurrently guarantee sufficient bandwidth to data-intensive applications or upper-bounded latencies to RT applications.

To address these challenges, this paper proposes a Time-slotted Software-defined Industrial Ethernet (TS-SDIE) for real-time Quality of Service (QoS) in Industry 4.0. The main idea underlying TS-SDIE is to employ a time-aware shaper before transmitting the packets. This concept is similar to the classical Time Division Multiple Access (TDMA) protocol used in cellular networks. Two key technologies, time synchronization and time slot-based packet switching, are addressed. First, an enhanced Precision Time Protocol (PTP) is proposed to realize precise time synchronization. Second, the SDN controller discovers the network topology and collects the various application requirements. Then, it executes a time slot-based scheduling algorithm and issues appropriate flow rules to switches. The switches allocate the transmission time slots for different packets to match the flow rules and forward packets at the precise time point.

The main contributions of this paper are summarized as follows:

- We provide a method to achieve highly accurate time synchronization over IE. The proposed method does not require the support of the hierarchical network structure and can synchronize the network to approximately 2.5 μs within one hop.
- We propose the TS-SDIE for RT QoS in Industry 4.0. Experimental results show that the switch delay can achieve 12.2 μs with a jitter of 0.12 μs, for packets with a size of 100 B.

The rest of this paper is organized as follows. Related work is reviewed in Section 2. The basic concepts of TS-SDIE architecture and problem analysis are introduced in Section 3. In Section 4, we describe the enhanced PTP mechanism, and we propose the time slot-based packet switching policy in Section 5. Section 6 presents the experimental results. Finally, Section 7 provides conclusion.

2. Related work

SDN-based industrial systems and time-slotted systems have been studied extensively in the literature. Enabling time-slotted systems also requires scheduling data flows in a synchronized manner. Hence, we divide the related studies into three categories: SDN-based industrial networks, TDMA-based SDN, and time synchronization.

2.1. SDN-based industrial networks

Guck et al. [14] proposed an SDN-based QoS control framework for industrial networks. To guarantee hard RT QoS, an accurate network model is maintained through network calculus that avoids control loops over forwarding and control planes and reduces the time required to make routing and admission control decisions. Li et al. [3] proposed a software-defined industrial network for intelligent manufacturing that supports customized production. This scheme adopts fixed routing paths and reserves special logical links for real-time industrial communication transmission. Wan et al. [1] also proposed a software-defined IIoT architecture and described the interface for information exchange. Both [3] and [1] discussed the basic problems that occur in SDN-based industrial networks and their possible solutions.

Nonetheless, in [14], the link delay is still affected by queuing delays in the priority based route planning mechanism; therefore, it cannot provide guaranteed transmission delay levels. The resource reserve mechanism in [3] wastes resources and reduces network utilization. Finally, [1] provides no description of RT communication capabilities.

2.2. TDMA-based SDN

Yang et al. [15] and Schweissguth et al. [5] introduced SDN-based architectures that enable TDMA for wireless LANs and IE, respectively. The object of [15] is to overcome the reduction in network efficiency caused by exposed links and hidden links in the Point Control Function of IEEE 802.11 standards. In [5], a real-time industrial communication system was designed that avoids the drawbacks of the current IE systems. Their key idea is similar to ours, where the data flow accesses the network based on the allocated time slots.

However, [15] does not solve any delay related issues. In [5], the computational complexity of the single controller increases sharply when the industrial network is scaled to the level of an entire factory. Meanwhile, it utilizes the Network Time Protocol for time synchronization which has a lower synchronization precision. Additionally, the TDMA-based Media Access Control (MAC) scheme does not require the time synchronization of all switches.

2.3. Time synchronization

For TDMA based RT Ethernet systems, the PTP is usually adapted to different applications [16]. For instance, Fontanelli et al. [17] proposed a servo-clock model for long chains of transparent clocks affected by uncertainty contributions, such as clock timing noises, imperfect link delay compensation, variable data rates, and packet losses. Xu et al. [18] presented a Kalman filter based PI clock servo to compensate for the quantization errors in the time stamping process of real-time networks with long paths. This approach assumes that peer delays are constant and that the asymmetry can be compensated. Giorgi et al. [19] designed a Kalman filter-based clock servo to estimate the clock offset and skew affected by uncertainties in time stamping [20].

Nevertheless, the current PTP-based time synchronization mechanisms mostly focus on the design of a clock servo in slave nodes, but they ignore the effects from intermediate node interference [18,19]. Moreover, these schemes involve additional communication overhead to maintain the tree structure based network topology. In [17], the transparent clocks also need to be synchronized with the master node, which increases the burden of intermediate nodes for time-division-based channel access networks.

3. TS-SDIE architecture and problem analysis

We begin this section with an overview of the TS-SDIE architecture, and then present a problem analysis for the design of TS-SDIE. Finally, we propose key technologies to solve the problems of TS-SDIE design. Table 1 gives some important abbreviations.

3.1. Overview of TS-SDIE architecture

TS-SDIE is inspired by the concept of SDN, which provides a centralized way to control the access of network devices, and removes the control functionality from network devices. As shown in Fig. 1, TS-SDIE separates the control functionality from the data plane. TS-SDIE consists of three planes. The application plane abstracts the application requirements of networks, such as critical hard RT controls, RT configuration and diagnostic, audio and video stream, and bandwidth guarantee. The control plane manages the interaction between the application plane and the data plane. It controls the physical network, provides specific requirements from the application plane, and issues the flow rules to the data plane. The data plane is composed of network devices with no control functionality that forwards data based on flow rules sent from the control plane.

Table 1
Abbreviations.

Abbreviations	Complete expression
TS-SDIE	Time-slotted Software-defined Industrial Ethernet;
IE	Industrial Ethernet;
RT	Real-time;
CPS	Cyber–Physical Systems;
IIoT	Industrial Internet of Things;
TSN	Time-Sensitive Networking;
SDN	Software-defined Networking;
QoS	Quality of Service;
TDMA	Time Division Multiple Access;
PTP	Precision Time Protocol;
MAC	Media Access Control;
PHY	Physical;
E2E	End-to-End;
OS	Operating System;
UDP	User Datagram Protocol.

Industrial automation systems often need to manage collaborative action for devices that belong to different domains, e.g., polishing and welding in automobile production lines. A single controller is ineffective for satisfying the RT transmission requirements of devices in different network domains. To overcome this problem, we first divide the data plane of TS-SDIE into several domains and assign a controller to each domain: namely, a domain controller, as shown in Fig. 1. The domain division is based on the functions of local network devices; for example, one domain might monitor machinery equipment and another domain might control industrial robot operations. All the domain controllers connect to a master controller that is responsible for resource management among the various domains. Each domain controller is responsible for the resource management of devices in its local domain.

For each application, the data plane is an abstraction of the network devices that can provide a logically independent transmission channel. In this way, developers can accelerate the design of new applications by customizing the data transmission and processing strategies. Industrial energy consumption can also be reduced by optimizing the networks.

3.2. Problem analysis

Unlike the requirements for improving throughput in existing SDN solutions, the primary demand of industrial networks involves RT transmissions. We detail the end-to-end (E2E) network delay in a decomposition similar to that presented in [21], as shown in Fig. 2, as follows: (1) $T_{process}$: this time is determined by the network protocol stack at the application layer and cannot be externally altered; (2) $T_{transmission}/T_{reception}$: these are the times required for the switch to respectively transmit or receive a packet at the physical layer (PHY layer), and these times depend on the packet size and the link rate; (3) $T_{propagation}$: this is the time required to transmit a packet from sender to receiver after it has left the sender, and it depends on the distance between these two devices; (4) T_{queue}: this is the time incurred from waiting in a packet queue for access to the transmit channel at the MAC layer. From the above analysis, $T_{process}$ is related to the operating system, while $T_{transmission}/T_{reception}$ and $T_{propagation}$ are constants that have little impact on the E2E delay. However, the traditional competition-based access mechanism at the MAC layer can cause a larger delay jitter and further affect the E2E delay.

To address this problem, TS-SDIE adopts a time slotted data plane that enables different devices connecting to the network in different time slots. To make our method clearer, we utilize an example to illustrate how the RT communication can achieve. In the welding process of automobile production line, two robots

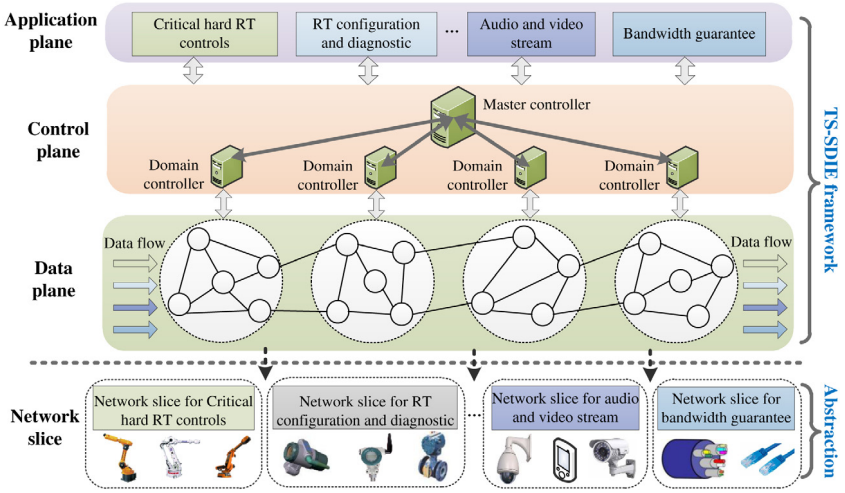

Fig. 1. The TS-SDIE framework.

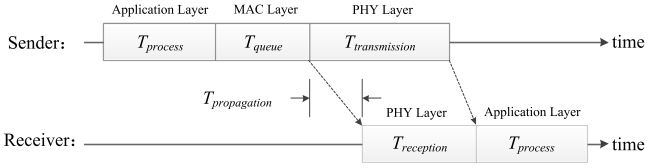

Fig. 2. Causes of E2E network delay.

perform symmetrical welding of a car body based on the closed-loop control with the controller. These two robots are devices that need to access the TS-SDIE. First, the application plane of TS-SDIE abstracts the RT demand of these two robots and sends the demand to the control plane. Then, the control plane specifies the data flow rules based on the RT demand and current network state, and issues the flow rules to the data plane. Finally, the data plane transmits the data from these two robots at specified time slots. This approach eliminates queuing delay because no two flows contend for access to a network link.

3.3. Key technologies

We use an intuitive comparison to illustrate the benefits brought by TS-SDIE, as shown in Fig. 3. The existing IE systems can be divided into three categories [22]. In class 1 (Ethernet), RT services transmit data via User Datagram Protocol (UDP) which achieves RT performance in the range of 100 ms. In class 2 (IE systems built on the top of Ethernet), RT services are achieved by adding a scheduling entity above the Ethernet link layer, and the achievable RT performance is close to 10 ms. In contrast, class 3 systems can achieve an RT performance below 1 ms by modifying the Ethernet link layer using schemes such as scheduling methods. These IE systems transmit RT data in best-effort manner, strict priority, and scheduling, respectively. Hence, congestion still exists, especially when data types and traffic increase sharply. In addition, the control and data planes are coupled, which increases the complexity of network reconfiguration, maintenance, and expansion.

Compared with existing IE systems, TS-SDIE decouples the control plane and data plane. The proposed two-layer controller architecture solves the problem of poor scalability of existing IE systems. The unified time-slotted physical devices in TS-SDIE make the network scalable and reconfigurable. Shared data plane reduces the cost of using proprietary hardware. Because no channel contention occurs, the time slotted forwarding rules can achieve higher RT performance.

To implement TS-SDIE on off-the-shelf Ethernet devices, two key technologies need to be addressed. TS-SDIE requires that the data flows must accurately access the networks in designated time slots, i.e., TS-SDIE needs to implement time synchronization between the source nodes of data flows and the border access devices. Consider the scenario that we do not have time synchronization, the data flows access the switch disorderly. The channel competition still exists. Next, time-slot based packet switching needs to be designed for TS-SDIE's data plane. In the next two sections, we elaborate on these two key technologies.

4. Enhanced precision time protocol

Time synchronization is important for TS-SDIE. Because PTP has been widely used in IE systems, we also adopt PTP to enable high accuracy time synchronization [23].

Standard PTP utilizes four packets to transmit four timestamps: $t1$, $t2$, $t3$, and $t4$, as shown in Fig. 4. It assumes that the E2E delay in both directions is symmetric. Then the offset between the master and slave clock is given by

$$\theta = [(t_2 - t_1) - (t_4 - t_3)]/2. \quad (1)$$

Obviously, this assumption is incompatible with the asymmetry that exists between the uplink and downlink in current devices. In tree-based networks, the synchronization error accumulates as the distance between the leaves and the master increases, because the forwarding delay is uncertain in PTP devices. Although frequent synchronization is effective for improving the synchronization precision, it also causes a dramatic increase in communication traffic. To improve the synchronization accuracy and reduce the communication overhead, we propose an enhanced PTP with link delay compensation.

In this section, we describe how we compensate and amend the link delay and this method's compatibility with traditional PTP. Table 2 gives some important notations.

4.1. Link delay compensation

Standard PTP utilizes the forwarding nodes to measure the ingress–egress message latency for *Sync* and *Delay-Req* packets in forwarding nodes. Then, it compensates for the latency in the slave clock. However, the clock error between the master

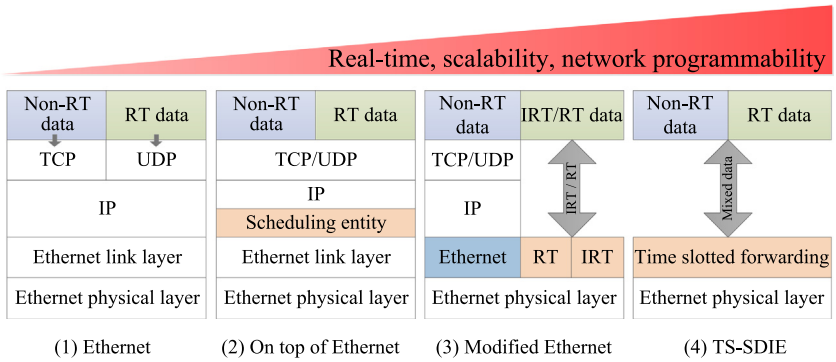

Fig. 3. Comparison with existing IE systems. IRT is the abbreviation for isochronous RT. IRT requires strict time synchronization for motion control and other applications.

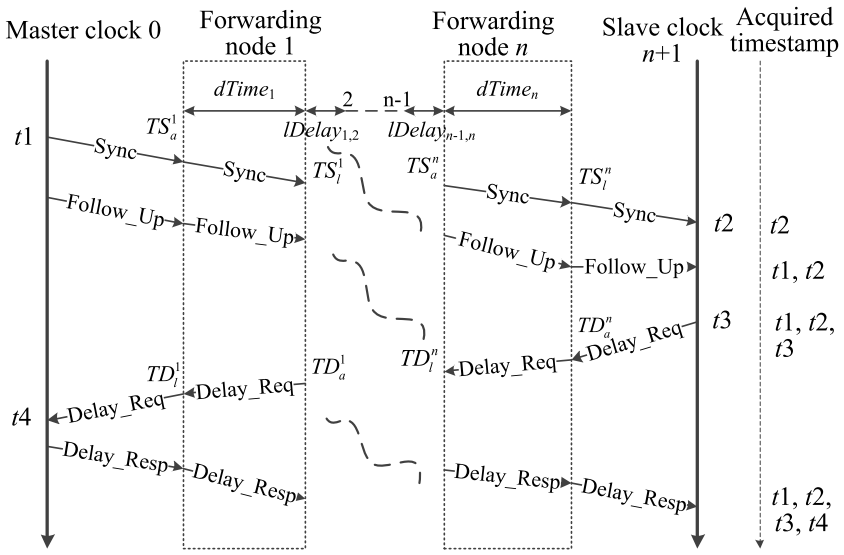

Fig. 4. An enhanced PTP-based message exchange.

node and forwarding nodes has a negative impact on the latency calculated by the forwarding nodes. For a hierarchical topology, the communication delay accumulates as the number of hops increase, which eventually interferes with the synchronization process.

PTP has also been extensively used in existing industrial control and monitoring devices. These PTP devices do not undertake any packet transfer task, but they need to synchronize cross TS-SDIE. In other words, traditional PTP devices and TS-SDIE physical devices coexist in the industrial field. Hence, compatibility is an important aspect of the PTP design for TS-SDIE.

To preserve compatibility with traditional PTP, we also utilize the forwarding nodes to measure the forwarding delay and communication delay between two adjacent nodes. Then, we divide the enhanced PTP into two parts, which respectively compensate for the forwarding delay and the communication delay.

As shown in Fig. 4, forwarding node i records the times at which it receives and sends the *Sync* packet, namely, TS_a^i and TS_l^i, respectively, and the times at which it receives and sends the *Delay-Req* packet, namely, TD_a^i and TD_l^i, respectively. Then, it calculates the forwarding delay as follows:

$$dTime_i = T_l^i - T_a^i, \tag{2}$$

where for the *Sync* packet, T_a^i and T_l^i denote TS_a^i and TS_l^i, respectively, and for the *Delay-Req* packet, T_a^i and T_l^i denote TD_a^i and TD_l^i, respectively.

Similarly, in Fig. 4, the communication delay for transmitting the *Sync* and *Delay-Req* packets between two adjacent nodes are

$$lDelayS_{i-1,i} = TS_a^i - TS_l^{i-1}, \tag{3}$$

$$lDelayD_{i-1,i} = TD_a^{i-1} - TD_l^i. \tag{4}$$

It is obvious that the measurement accuracy of the forwarding delay is relevant to the consistency of the clock rates of the forwarding node and the master. When the consistency error of the clock skews is too large, calculating the forwarding delay will result in a much larger error. The calculation error of communication delay is relevant to both the clock offset between two adjacent nodes and the consistency between the clock rates of the forwarding node and the master. Therefore, it is necessary to amend these two delays. Though PTP has deemed it necessary that the adjustment of the residence time and the estimation of the ratio of the master and local clock rates, none relevant technologies have been proposed.

We denote the estimate of clock offset between the adjacent nodes i and $i-1$ as $\widehat{O}_{i-1,i}$, and the estimate of the relative clock rate of the master–slave clocks as $\widehat{\alpha}_{0,i}$. Hence, the amended forwarding delay and communication delay satisfy the following:

$$\widehat{dTime}_i = \left(T_l^i - T_a^i\right)\widehat{\alpha}_{0,i}, \tag{5}$$

$$\widehat{lDelayS}_{i-1,i} = (TS_a^i - TS_l^{i-1} + \widehat{O}_{i-1,i}) \cdot \widehat{\alpha}_{0,i}, \tag{6}$$

Table 2
Notation definitions.

Symbol	Definitions
θ	The offset between two nodes;
TS_a^i	The time that node i receives the *Sync* packet;
TS_l^i	The time that node i sends the *Sync* packet;
$TS_l^{i-1,k}$	The time that node $i-1$ sends the kth *Sync* packet;
$TS_a^{i,k}$	The time that node i receives the kth *Sync* packet;
$\overline{TS_l^{i-1}}$	The mean of $TS_l^{i-1,k}$;
$\overline{TS_a^{i-1}}$	The mean of $TS_a^{i,k}$;
TD_a^i	The time that node i receives the *Delay-Req* packet;
TD_l^i	The time that node i sends the *Delay-Req* packet;
$dTime_i$	The forwarding delay of node i;
$lDelayS_{i-1,i}$	The communication delay of *Sync* packet;
$lDelayD_{i-1,i}$	The communication delay of *Delay-Req* packet;
$\widehat{dTime}_i$	The amended forwarding delay;
$\widehat{lDelayS}_{i-1,i}$	The amended communication delay of *Sync* packet;
$\widehat{lDelayD}_{i,i-1}$	The amended communication delay of *Delay-Req* packet;
$\tau_i(t)$	The time of node i;
α_i	The clock rate of node i;
β_i	The clock offset of node i;
$\alpha_{i-1,i}$	The relative clock rate of nodes i and $i-1$;
$\beta_{i-1,i}$	The relative clock offset of nodes i and $i-1$;
$O_{i-1,i}$	The clock offset of nodes i and $i-1$;
$\widehat{\alpha}_{i-1,i}$	The estimation of relative clock rate;
$\widehat{\beta}_{i-1,i}$	The estimation of relative clock offset;
$\widehat{O}_{i-1,i}$	The estimation of clock offset of nodes i and $i-1$.

$$\widehat{lDelayD}_{i,i-1} = (TD_a^{i-1} - TD_l^i - \widehat{O}_{i-1,i}) \cdot \widehat{\alpha}_{0,i}. \tag{7}$$

The sum of the amended forwarding delay and communication delay should be added to the correctionField of the corresponding *Follow_Up* and *Delay_Resp* packets. For the slave clock, it computes the clock offset to synchronize with the master in the same manner as in IEEE 1588. The detailed process is no longer repeated here. Next, we detail the high accuracy estimation of the relative clock parameters $\widehat{O}_{i-1,i}$ and $\widehat{\alpha}_{0,i}$.

4.2. Relative clock parameters estimation

Each node's time $\tau(t)$ is obtained by counting the output pulse of an internal crystal oscillator. Due to the hardware manufacturing process and environmental interference, the timing rates of $\tau(t)$ and the absolute time t are different. For the convenience of analysis, $\tau(t)$ is always modeled as an increasing function of t, as given by [24]:

$$\tau_i(t) = \alpha_i t + \beta_i, \tag{8}$$

where α_i and β_i are the clock rate and offset, which determine the clock speed and the initial synchronization error, respectively. The times of two adjacent nodes (e.g., i and $i-1$) satisfy the following relationship:

$$\tau_{i-1}(t) = \alpha_{i-1,i}\tau_i(t) + \beta_{i-1,i}, \tag{9}$$

where $\alpha_{i-1,i} = \alpha_{i-1}/\alpha_i$ denotes the relative clock rate of nodes i and $i-1$, and $\beta_{i-1,i} = \beta_{i-1} - \alpha_{i-1,i}\beta_i$ denotes the relevant clock offset.

Relative clock rate estimation has been applied in many protocols to adjust the offset that accumulates over time and reduce the communication overhead. In the hierarchical synchronization mechanism, each node synchronizes with its upper layer node and finally synchronizes with the master clock. That is, this mechanism constructs a chain of clocks along the communication link from the slave clock to the master clock. Similarly, estimating the relative clock rate at a higher precision results in a higher synchronization precision. However, the clock errors in this chain of clocks have a negative impact on the relative clock rate of the master–slave clocks. The error is magnified at each hop, ultimately leading to exponential error proliferation.

To overcome this problem, we estimate the relative clock rate of the master–slave clocks directly without synchronizing the slave clock with the forwarding node. The independent relative clock rate estimation is not influenced by the clock errors in the chain of clocks along the communication link. Next, we detail the estimation of the relative clock rate.

As shown in Fig. 4, the forwarding node i records the time of receiving and sending the *Sync* packet, respectively, namely, TS_a^i and TS_l^i. From Eq. (8), there is a linear relationship between two clocks. Here, we adopt a linear regression method to estimate the relative clock rate,

$$\widehat{\alpha}_{i-1,i} = \frac{\sum_{k=1}^{N}\left(TS_l^{i-1,k} - \overline{TS_l^{i-1}}\right)\left(TS_a^{i,k} - \overline{TS_a^{i}}\right)}{\sum_{k=1}^{N}\left(TS_a^{i,k} - \overline{TS_a^{i}}\right)^2}, \tag{10}$$

where i denotes the downstream node, $TS_l^{i-1,k}$ denotes the time of the kth *Sync* packet leaving node $i-1$, $TS_a^{i,k}$ denotes the time of the kth *Sync* packet arriving at node i, $\overline{TS_l^{i-1}}$ and $\overline{TS_a^{i-1}}$ are the means of $TS_l^{i-1,k}$ and $TS_a^{i,k}$, respectively, where $k = 1, 2, \cdots, N$.

The relative clock rate for any node i that is i hops away from the master node 0 is $\alpha_{0,i} = \alpha_0/\alpha_i$. Based on the relative clock rate definition, the relative clock rate of the master–slave clocks satisfies the following iterative equation:

$$\Pi_{k=1}^{i}\widehat{\alpha}_{k-1,k} = \widehat{\alpha}_{0,1} \cdot \widehat{\alpha}_{1,2} \cdots \widehat{\alpha}_{i-2,i-1} \cdot \widehat{\alpha}_{i-1,i} = \widehat{\alpha}_{0,i}, \tag{11}$$

$$\widehat{\alpha}_{0,i} = \widehat{\alpha}_{0,i-1} \cdot \widehat{\alpha}_{i-1,i}. \tag{12}$$

Based on the obtained relative clock rate, we can estimate the clock offset between any two nodes, which further improve the synchronization accuracy.

From Eq. (8), the clock offset of nodes i and $i-1$ is

$$O_{i-1,i} = (\alpha_{i-1} - \alpha_i)\, t + \beta_{i-1} - \beta_i = \alpha_{o,i}\tau_i(t) + \beta_{o,i}, \tag{13}$$

where $\alpha_{o,i} = (\alpha_{i-1} - \alpha_i)/\alpha_i$ denotes the relative rate regarding the time offset and $\tau_i(t)$, $\beta_{o,i} = \beta_{i-1,i}$ denotes the relevant clock offset, and $O_{i-1,i} = \tau_{i-1}(t) - \tau_i(t)$.

From Eqs. (10) and (13), we have

$$\widehat{\alpha}_{o,i} = \widehat{\alpha}_{i-1,i} - 1, \tag{14}$$

$$\widehat{\beta}_{o,i} = \overline{TS_l^{i-1}} - \widehat{\alpha}_{i-1,i}\overline{TS_a^i}. \tag{15}$$

Thus, upon receiving the Sync packet, node i estimates its clock offset relevant to node $i-1$, namely,

$$\widehat{O}_{i-1,i} = \left(\widehat{\alpha}_{i-1,i} - 1\right)\left(TS_a^i - \overline{TS_a^i}\right) + \left(\overline{TS_l^{i-1}} - \overline{TS_a^i}\right). \tag{16}$$

Then, the relative clock parameters $\widehat{O}_{i-1,i}$ and $\widehat{\alpha}_{0,i}$ can be obtained by Eqs. (12) and (16), and the obtained delays are modified using Eqs. (5)–(7). In this way, the slave clock synchronizes with the master clock directly without being affected by the error caused by the forwarding nodes.

4.3. Compatibility

Compared with the standard PTP, the enhanced PTP must transmit the relative clock rate and the packet sending time of the local node. By analyzing the message format, we can see that several reserved fields exist in the message header, i.e. a 4-byte reserved field and a 1-byte reserved field.

We adopt the 4-byte reserved field in the message headers of the *Follow_Up*, *Delay_Resp* and *Sync* packets to transmit the

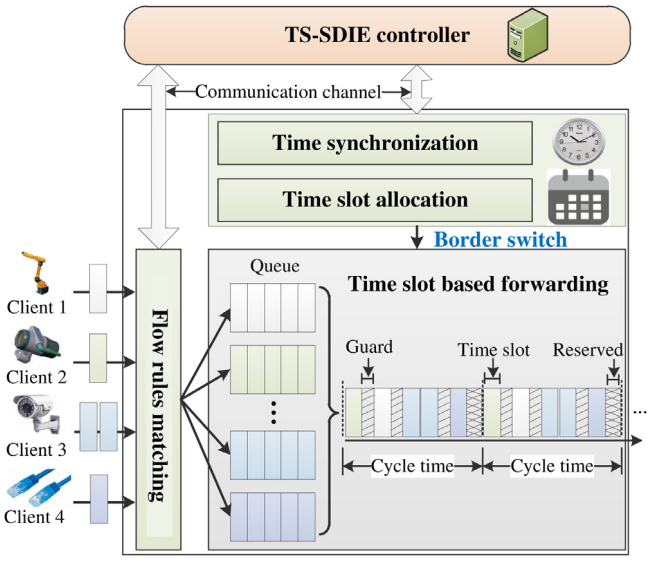

Fig. 5. Time slot-based forwarding.

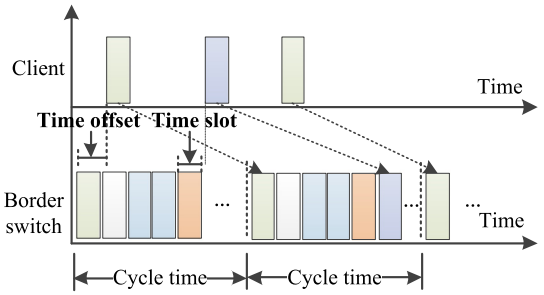

Fig. 6. Mismatched time slots.

leaving time of the *Sync* packet, the arriving time of the *Delay_Req* packet, and the relative clock rate, respectively. Here, the message IDs of the *Follow_Up* and *Sync* packets and the *Delay_Resp* and *Delay_Req* packets should remain consistent. Finally, the offset calculation and compensation used in enhanced PTP are the same as those used in standard PTP.

5. Time slot-based packet switching

The design of time slot-based Ethernet devices is a core aspect for achieving the packet scheduling based on the time slot in TS-SDIE. As shown in Fig. 5, a time-slotted Ethernet device includes four modules: flow rules matching, time synchronization, time slot allocation, and time slot-based forwarding. The flow rules matching module processes the incoming data flow according to the flow rules from the controller; then, it puts corresponding flows into different queues. The time synchronization and time slot allocation policy modules determine when and how the data flows access the network, namely, time slot-based forwarding. The idea of modularization used in the design of a TS-SDIE physical device implements programmable data communication. Meanwhile, the forwarding rules can change dynamically to meet various QoS requirements.

In TS-SDIE, the border switches determine the access of data flows. First, border switches and clients generating the data synchronize to the master clock in enhanced PTP. Then clients can achieve time synchronization with the border switches. The master clock can be the controller or some other time sources. Second, the border switches assign time slots and the start transmission time to different clients. Clients access to TS-SDIE based on the distributed rules.

As shown in Fig. 5, the border switch controls the flows based on the time and the schedule. Each queue is defined within the schedule. When the packets in a queue are forwarding during scheduled time windows, other queues will typically be blocked, thus eliminating the chance of a scheduled flow being impeded by an unscheduled flow. This means that the delay through each switch is deterministic, which means that TS-SDIE can guarantee the message delay.

Formally, the time slot size is important in controlling the communication traffic of the clients. A long time slot can accommodate multiple packets, while a short time slot provides fine-grained traffic control. The typical time slot size is 2 ms∼10 ms [15]. In our design, the minimum time slot size can achieve 10 μs based on considering the RT demand. Therefore, it is necessary to synchronize the network within several microseconds, which our enhanced PTP can achieve.

Next, we describe the time slot operation policy and the schedule design.

5.1. Time slot operation

The common control policies for time slots consist of four parts: polling, priority, guard, and reservation.

Competition- and priority-based access mechanisms introduce congestion in queues, resulting in long-term channel occupation, which has a negative impact on load balancing and real-time guarantee. However, polling is an effective way to ensure fairness. Therefore, we utilize a polling-based method to control data flows access. As shown in Fig. 5, we divide the time into periodic frames. Then, each frame is further divided into some number of time slots. In this way, each data flow has the opportunity to access TS-SDIE in different frame cycles based on its RT demand.

For packets in the flow from a single client, we can assign priorities to different packets to support finer-grained QoS. Suppose that we allocate three time slots (1, 2, 3) to a client, and all the packets in that client's data flow can be transmitted in those three time slots. Then, packets with a higher priority will be transmitted earlier, and packets with the same priority will be transmitted in a FIFO (first in first out) manner. The TS-SDIE controller determines the packet priorities by referencing the QoS requirements.

Based on the time slot allocation rule, a client transmits its data at time

$$T_K^s = T_0 + K \times S, \tag{17}$$

where T_0 is an arbitrary time, K is the index of a time slot, and S is the size of the time slot. As shown in Fig. 6, with the existence of time offset between the client and the border switch, the transmit time between the client and the border switch is unmatched. The current packet transmission will take place in the following cycle, namely, increasing the forwarding delay of the real-time packet. In addition, if the packet transmission is not completed in the current time slot, it will affect subsequent packet transmissions for other clients, as shown in Fig. 6. Therefore, a guard window $t_{guardWindow}$ is required before the transmission starts a new flow. A large guard window will reduce the bandwidth utilization; consequently, we set $t_{guardWindow} = S$ in our design, as shown in Fig. 5.

The design described above solves the problem of fairness and interference for periodic data. It should be noted that the data types in industrial systems are often known. So, the flow rules in the switch can be predefined. Flow rules matching module processes the data flow directly. Now, the problem that obtaining

global view in SDN systems affects the RT property is solved in TS-SDIE. For sporadic data with critical hard RT demand, the controller will redesign and redistribute the schedule policy, which will introduce additional computing and communication overhead and delays. To solve these problems, we reserve time slots for sporadic data at the end of the cycle without disturbing the other flows, as shown in Fig. 5. The number of reserved time slots is determined by the controller by referencing the corresponding application.

5.2. Scheduling method

The design of a scheduling method involves finding an appropriate route and a time slot configuration for each flow.

The number of time slots for each flow is set according to the corresponding requirement parameters, e.g., the communication traffic or packet number. This configuration guarantees that the sender can transmit all the packets for a single data interval in the allocated time slot. To avoid packets accumulating in the queue and to ensure that all the data packets are sent in their allotted time slots, the time interval during which the application generates data packets and the cycle period of time slots should satisfy the following:

$$t_{cycleSlot} \geq \min t_{dataInterval}, \tag{18}$$

where $\min t_{dataInterval}$ denotes the minimum value of the time interval.

6. Experimental results

We demonstrate the feasibility and benefit of TS-SDIE with experiment. The testbed of TS-SDIE consists of a controller, a switch, and two clients. The controller is a Dell server based on the POX framework. The compile environment supports Python 2.7 or above. The switch and clients are three multicore network processors (Cavium OCTEON CN68XX). The switch has two operating systems (OS): A Linux OS takes charge of the communication with the controller and receiving the flow table rules, and SE OS takes charge of querying the flow table and forwarding the packets. The communication channel between the controller and the switch uses the POF protocol [25]. For the clients, one is responsible for collecting and sending packets and delay statistics, while the other one is responsible for receiving packets.

The remaining parts in this section detail the performance analysis of the enhanced-PTP and TS-SDIE.

6.1. Enhanced PTP

We use the time offset to evaluate the synchronization accuracy: the time offset describes the difference between the master clock and the slave clock. Because our goal is to synchronize all devices to the same master, the offset value is the synchronization error. We set the size of the regression table to $N = 10$ considering the memory overhead and guaranteed computational precision, and we set the time synchronization period to $T = 2^{-4}$ s. Then, we observe and analyze the synchronization precision of the enhanced PTP mechanism based on the collected data.

We first compare norm PTP with the enhanced PTP under one hop. Here, the switch functions as the master clock, and the client is the slave clock. The maximum synchronization error of the norm PTP is approximately 100 μs. Here, to reduce the development cost, the timestamp of each incoming or outputting packet is added in the driver layer rather than the PHY–MAC interface. The synchronization error of enhanced PTP is several microseconds due to the compensation of link delay. The mean value of maximum offset is 2.5 μs.

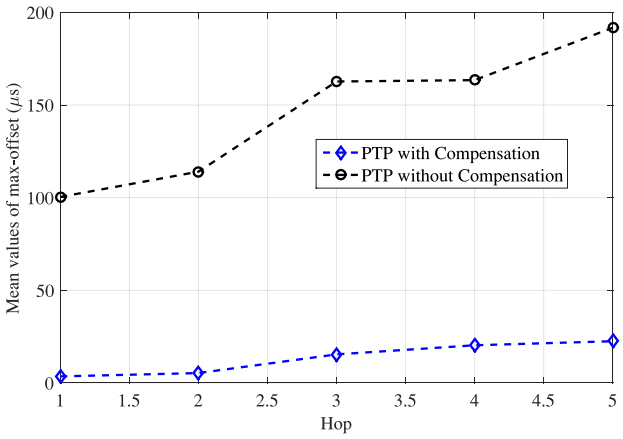

Fig. 7. Time synchronization errors of enhanced PTP and norm PTP under different hops.

In the performance evaluation of enhanced PTP under different hop numbers (i.e., 1, 2, 3, 4, and 5), we set another two Conemtech DK5 development kits as the master and slave clocks, respectively. The TS-SDIE switch is a COMPEX WP546HV, and message exchange is realized through TS-SDIE. Then, we used the Saleae Logic 16 logic analyzer to collect the output second-pulses from the master and slave node clocks. We used Matlab to process and analyze the collected data.

Fig. 7 shows the performances of the two methods under different hops. Obviously, the synchronization error of norm PTP without compensation is seriously affected by the forwarding nodes' clocks. In contrast, enhanced PTP with compensation compensates for the forwarding delay and communication delay based on the relative clock rate between the forwarding node and the master; thus, it avoids the interference from local clocks and achieves time synchronization at the microsecond level.

6.2. TS-SDIE

Packet transmission delay and delay jitter are two important metrics of RT QoS [4–6]. In this section, we conduct an experiment using these two metrics to illustrate the feasibility and effectiveness of TS-SDIE. During the experiment, considering the synchronization precision, we set the time slot size to 10 μs and set the cycle period to 10 ms.

First, we compare the performance of a time slot operation with a guard window for a single flow. The packet size is set to 100 B, and the data rate is 100 kpps. As shown in Fig. 8, without a guard window, mismatched time slots occur frequently, which further affect the packet transmission delay. However, with the guard window, packet transmission does not need to wait for an idle channel, and the switch forwards the packet quickly. Fig. 9 shows stable packet transmission with a guard window. The mean values of the forwarding delay and its jitter are 12.2 μs and 0.12 μs, respectively. In the following experiment, we provide a guard window for each flow.

Second, we conduct an experiment to illustrate the properties with different packet sizes and background traffic. As shown in Fig. 10, the packet sizes are 100 B, 200 B, 500 B, 1000 B, and 1500 B, respectively, and the data rate is 100 kpps. As the packet size increases, the mean value of forwarding delay presents a linearly increasing trend. It is common for the packet transmission time to be proportional to the packet size given a

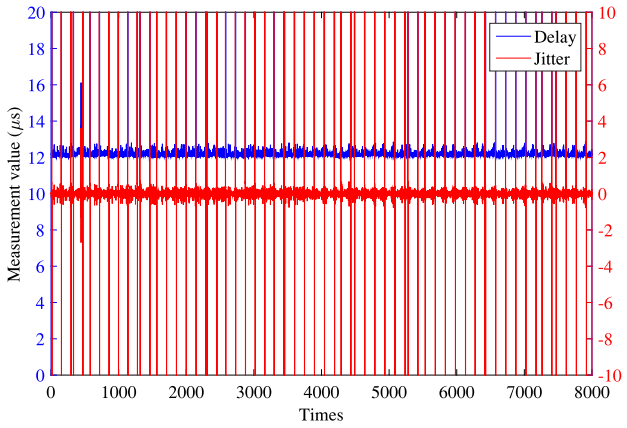

Fig. 8. Delay and jitter of the packet transmission without a guard window.

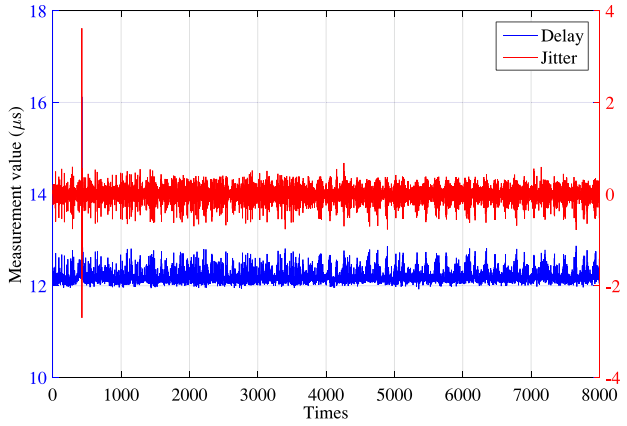

Fig. 9. Delay and jitter of the packet transmission with a guard window.

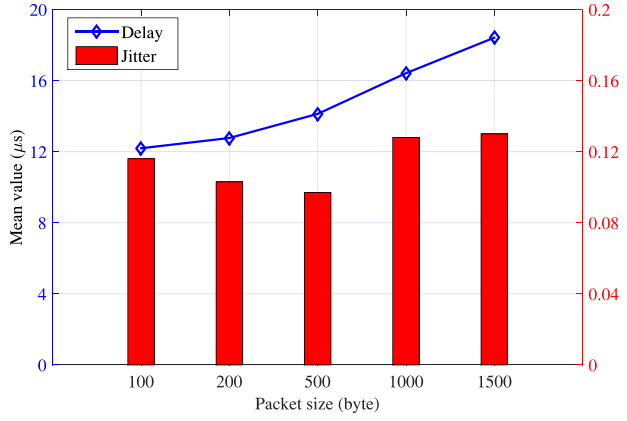

Fig. 10. Delay and jitter of the packet transmission with different packet sizes.

constant bandwidth. The mean jitter value is almost stable, which demonstrates the stability of the packet transmission.

In Fig. 11, we varied the background traffic among 50 kpps, 100 kpps, 200 kpps, 500 kpps, and 1000 kpps. The packet size of the flow and background traffic is 500 B, and the flow data rate is 100 kpps. The results show that the delay and jitter remain almost the same(approximately 14 μs and 0.078 μs, respectively) because the flow and background traffic transmissions occur in different communication channels. Therefore, there is no risk of interference for flow transmission.

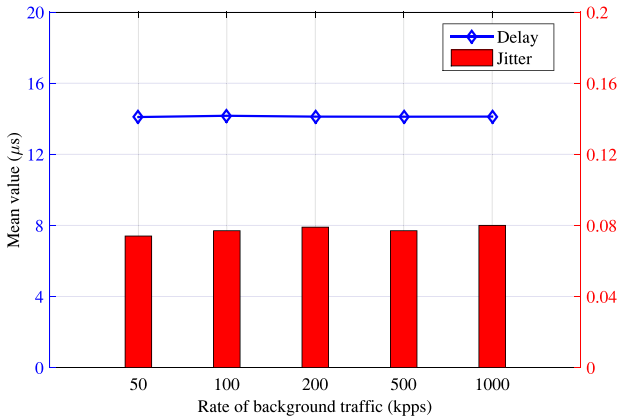

Fig. 11. Delay and jitter of the packet transmission with different background traffic.

Table 3
Measurement results for different flows (in μs).

Flow name	Type	Mean value	Standard deviation
Flow1	Delay	14.056	0.216
	Jitter	0.172	0.160
Flow2	Delay	14.054	0.215
	Jitter	0.169	0.159
Flow3	Delay	14.056	0.216
	Jitter	0.167	0.158
Flow4	Delay	14.098	0.232
	Jitter	0.196	0.176
Flow5	Delay	14.076	0.223
	Jitter	0.176	0.163

Finally, Table 3 shows the performance when multiple flows are accessing the network. The packet size of each flow is the same as 500 B, and the data rate is 100 kpps. As shown in Table 3, the forwarding delay is stable; however, the jitter here varies slightly from that in Fig. 10 because the transmission of multiple flows increases the process time uncertainty in the switch.

These experimental results indicate that TS-SDIE can provide a guaranteed message delay with the jitter satisfying the requirement of TSN.

7. Conclusion

In this paper, we proposed TS-SDIE, a Time-slotted Software-defined Industrial Ethernet (TS-SDIE) for RT communication. TS-SDIE provides communication channels for different applications that do not interfere with each other, which guarantees the forwarding delay for RT data. Additionally, we designed an enhanced PTP to achieve precise synchronization. The enhanced PTP compensates for the link delay based on the relative clock rate obtained in an independent channel, thus avoiding the interference from polluted local clocks. Extensive experimental results demonstrate the TS-SDIE's effectiveness.

There are multiple future research directions. First, we plan to devise a routing and scheduling mechanism that satisfies all the flow constraints. Second, considering a factory with many subnets for different applications, it would be interesting to devise a control plane that considers RT operational requirements.

Acknowledgments

This work was supported by the National Natural Science Foundation of China [grant numbers 61533015, 61502474] and the Foundation of CAS Youth Innovation Promotion Association, China.

References

[1] J. Wan, S. Tang, Z. Shu, et al., Software-defined industrial internet of things in the context of industry 4.0, IEEE Sens. J. 16 (20) (2016) 7373–7380.

[2] L. Thames, D. Schaefer, Software-defined cloud manufacturing for industry 4.0, Procedia CIRP 52 (2016) 12–17.

[3] D. Li, M. Zhou, P. Zeng, et al., Green and reliable software-defined industrial networks, IEEE Commun. Mag. 54 (10) (2016) 30–37.

[4] P. Danielis, J. Skodzik, V. Altmann, et al., Survey on real-time communication via ethernet in industrial automation environments, in: Proc. of IEEE ETFA, IEEE, Barcelona, Spain, 2014, pp. 1–8.

[5] E. Schweissguth, P. Danielis, C. Niemann, et al., Application-aware industrial ethernet based on an SDN-supported TDMA approach, in: Proc. of IEEE WFCS, IEEE, Aveiro, Portugal, 2016, pp. 1–8.

[6] W. Steiner, P.G. Pen, M. Gutirrez, et al., Next generation real-time networks based on it technologies, in: Proc. of IEEE ETFA, IEEE, Berlin, Germany, 2016, pp. 1–8.

[7] F. Hu, Q. Hao, K. Bao, A survey on software-defined network and openflow: from concept to implementation, IEEE Commun. Surv. Tutor. 16 (4) (2014) 2181–2206.

[8] D. Kreutz, F.M.V. Ramos, P.E. Verissimo, et al., Software-defined networking: a comprehensive survey, Proc. IEEE 103 (1) (2015) 14–76.

[9] F. Shang, L. Mao, W. Gong, Service-aware adaptive link load balancing mechanism for software-defined networking, Future Gener. Comput. Syst. 81 (2018) 452–464.

[10] M. Herlich, J.L. Du, F. Schörghofer, et al., Proof-of-concept for a software-defined real-time ethernet, in: Proc. of IEEE ETFA, IEEE, Berlin, Germany, 2016, pp. 1–4.

[11] G. Kalman, Applicability of software defined networking in industrial Ethernet, in: Telecommunications Forum Telfor, IEEE, Serbia, Yugoslavia, 2014, pp. 340–343.

[12] D. Henneke, L. Wisniewski, J. Jasperneite, Analysis of realizing a future industrial network by means of software-defined networking (SDN), in: Proc. of IEEE WFCS, IEEE, Aveiro, Portuga, 2016, pp. 1–4.

[13] J. Vestin, A. Kassler, J. Akerberg, Resilient software defined networking for industrial control networks, in: Proc. of International Conference on Information, Communications and Signal Processing, IEEE, Singapore, 2015, pp. 1–5.

[14] J.W. Guck, M. Reisslein, W. Kellerer, Function split between delay-constrained routing and resource allocation for centrally managed QoS in industrial networks, IEEE Trans. Ind. Inf. 12 (6) (2016) 2050–2061.

[15] Z. Yang, J. Zhang, K. Tan, et al., Enabling TDMA for today's wireless LANs, in: Proc. of IEEE INFOCOM, IEEE, Rio de Janeiro, Brazil, 2009, pp. 1436–1444.

[16] P. Ferrari, A. Flammini, S. Rinaldi, et al., On the seamless interconnection of IEEE1588-based devices using a PROFINET IO infrastructure, IEEE Trans. Ind. Inf. 6 (3) (2010) 381–392.

[17] D. Fontanelli, D. Macii, S. Rinaldi, et al., A servo-clock model for chains of transparent clocks affected by synchronization period jitter, IEEE Trans. Instrum. Meas. 63 (5) (2014) 1085–1095.

[18] X. Xu, Z. Xiong, X. Sheng, et al., A new time synchronization method for reducing quantization error accumulation over real-time networks: theory and experiments, IEEE Trans. Ind. Inf. 9 (3) (2013) 1659–1669.

[19] G. Giorgi, C. Narduzzi, Performance analysis of kalman-filter-based clock synchronization in IEEE 1588 networks, IEEE Trans. Instrum. Meas. 60 (8) (2011) 2902–2909.

[20] A. Mahmood, R. Exel, T. Sauter, Delay and jitter characterization for software-based clock synchronization over WLAN using PTP, IEEE Trans. Ind. Inf. 10 (2) (2014) 1198–1206.

[21] S. Ganeriwal, P. Kumar, M.B. Srivastava, Timing-sync protocol for sensor networks, in: Proc. of the 1st International Conference on Embedded Networked Sensor Systems, Los Angeles, USA, 2003, pp. 138–149.

[22] M. Wollschlaeger, T. Sauter, J. Jasperneite, The future of industrial communication: automation networks in the era of the internet of things and industry 4.0, IEEE Ind. Electron. Mag. 11 (1) (2017) 17–27.

[23] IEEE Standard for a Precision Clock Synchronization Protocol for Networked Measurement and Control Systems. IEEE Standard (2008) 1588-2008.

[24] J. He, P. Cheng, L. Shi, et al., Time synchronization in WSNs: a maximum-value-based consensus approach, IEEE Trans. Automat. Control 59 (3) (2014) 660–675.

[25] H. Song, Protocol-oblivious forwarding: Unleash the power of SDN through a future-proof forwarding plane, in: Proc. of the Second ACM SIGCOMM Workshop on Hot Topics in Software Defined Networking, ACM, Hong Kong, China, 2013, pp. 127–132.

Peng Zeng received his B.E. degree in computer science from Shandong University and his Ph.D. degree from Shenyang Institute of Automation (SIA), Chinese Academy of Sciences (CAS), in 1998 and 2005, respectively. He is currently the director of the Department of Industrial Control Network and System of SIA, CAS. He is also a professor at SIA, CAS, an expert member of the IEC TC65 WG16, a member of the standards committee of SP100, and a member of the Wireless WG of FieldBus Foundation. His research interests include software defined networking, industrial communication, and wireless sensor networks.

Zhaowei Wang received his B.E. degree in measurement & control technology and instrumentation from Zhengzhou University, in 2012. He is currently pursuing his Ph.D. degree in control theory and control engineering at the SIA, CAS. His research interests include software defined networking, industrial control networks, and wireless sensor networks.

Zhengyi Jia received his B.E. degree in Electronic Information Science and Technology from the University of Science and Technology of China, in 2014. He is currently pursuing his Ph.D. degree in signal and information processing at the National Network New Media Engineering Technology Research Center, Institute of Acoustics, CAS. His research interests include software defined networking, next-generation mobile networks, and protocol oblivious forwarding switch.

Linghe Kong received his B.E. degree from Xidian University, China, in 2005, his M.S. degree from TELECOM SudParis, France, in 2007, and his Ph.D. degree from Shanghai Jiao Tong University, China, in 2012. He is currently a research professor with Department of Computer Science and Engineering at Shanghai Jiao Tong University. Before that, he was a postdoctoral researcher at Columbia University, McGill University, and Singapore University of Technology and Design. His research interests include 5G communication, sensor networks, mobile computing, Internet of things, big data, and smart energy systems.

Dong Li received his B.E. degree in electronics engineering from Fudan University, China, in 2008, and his Ph.D. degree in mechatronic engineering at SIA, CAS, in 2016. He is currently an associate professor at SIA, CAS. His research interests include industrial networks and software defined networking for intelligent manufacturing.

Xi Jin received her M.S. and Ph.D. degrees in computer science from Northeastern University, China, in 2008 and 2013, respectively. She is currently an associate professor at SIA, CAS. Her research interests include wireless sensor networks and real-time systems, especially the real-time scheduling algorithms, and worst case end-to-end delay analysis.