International Journal of Innovative
Computing, Information and Control
Volume **15**, Number **4**, August **2019**

 ISSN 1349-4198
pp. **1271–1286**

AN SDN-BASED MEASUREMENT SCHEME TO BUILD DELAY DATABASE FOR TIME-SENSITIVE NETWORK SCHEDULING

Zhengyi Jia[1,2], Jinlin Wang[1,2], Xiao Chen[1,2], Yiqiang Sheng[1] and Kang Zheng[1,2]

[1]National Network New Media Engineering Research Center
Institute of Acoustics, Chinese Academy of Sciences
No. 21, North 4th Ring Road, Haidian District, Beijing 100190, P. R. China
{ jiazy; wangjl; xxchen; shengyq; zhengk }@dsp.ac.cn

[2]School of Electronic, Electrical and Communication Engineering
University of Chinese Academy of Sciences
No. 19(A), Yuquan Road, Shijingshan District, Beijing 100049, P. R. China

Received November 2018; revised March 2019

Abstract. *The IEEE 802.1Qbv standard has proposed the Time Aware Shaper (TAS) mechanism that meets the stringent deterministic end-to-end delay requirements. Although the database of delay between the adjacent nodes is critical for the TAS scheduling, we have not found an automatic and reliable measurement scheme to build a delay database dedicated to the TAS scheduling yet. In this paper, different from the delay in IP networks, we define the Time-Sensitive Network Delay (TSND) dedicated to the TAS scheduling and propose an SDN-based TSND measurement scheme using a time-triggered mechanism. Moreover, to apply TSND in different network environments, we derive formulas to quantify the impact of the clock synchronization accuracy and packet size on the TSND measurement. We then implement the scheme, based on the POX SDN controller and the multi-core network processor, and build an experiment scene with the above equipment. The experiment shows: the TSND is effective for predicting the receiving slot; the maximum error of the TSND caused by the clock synchronization accuracy can be kept within two slots as long as the clock offset is less than slot length; if the size of the packet in the Time-Triggered (TT) flow differs significantly, forwarding the TT flow in the slot calculated by the larger TSND can achieve less jitter, at the expense of increasing the delay. The experiment results are in good agreement with the formulas. Furthermore, the TSND measurement scheme is dedicated to building a delay database of the adjacent TSN nodes, which may be suited as a starting point for the TAS scheduling.*

Keywords: Time-sensitive network, Network delay, Time aware shaper, TSN scheduling, Time-triggered

1. **Introduction.** With the rapid development of real-time applications such as autonomous driving, AR/VR, we demand a network transmission control mechanism to support deterministic Ultra-Low Latency (ULL) [1]. When it comes to the deterministic latency system, the DTM project [2] is a typically excellent job. In the DTM, each second is divided into 8000 cycles, and each cycle is subdivided into slots. However, since the arrival time of the packet is unknown, the DTM system assigns the slot randomly. So the packet has to be buffered in the queue if the packet arrives earlier or later than its slot. In this case, the maximum queuing delay may be 125 microseconds at one node, which is not tolerable for ULL. To reduce delay and save the switch buffer, precise allocating of the slot is required according to the arrival time of the packet.

DOI: 10.24507/ijicic.15.04.1271

The IEEE 802.1Qbv [3], which has been standardized by the Time-Sensitive Network (TSN) Task Group, has proposed the Time Aware Shaper (TAS) mechanism that meets the stringent end-to-end latency and jitter requirements [4]. In IEEE 802.1Qbv, there are eight priority queues for each port of the TSN node. Notably, the priority seven is for the Time-Triggered (TT) flow. The frames in different queues are selected according to the transmission gate control list. The state of the transmission gate (0 for close, 1 for open) determines whether or not queued frames can be selected for transmission. Although the TAS scheduling is critical work, it is not in the scope of the IEEE 802.1Qbv standard.

Different from the traditional event-triggered mechanism, the time-triggered mechanism forwards the packet at a pre-configured time no matter the packet arrives early or late. So before scheduling, we need to know the arrival time of the TT flows. Only with the delay database between the adjacent nodes could we get the flow arrival time on each node once the sending time on the starting node is determined. As one can see that the database of delay is critical for the TAS scheduling, nonetheless, there are few descriptions or discussions on delay measurement schemes.

The network delay including One-Way Delay (OWD) and Round-Trip Time (RTT) is the crucial parameter for IP Performance Metrics (IPPM). Consequently, the existing work of delay measurement mainly focuses on IP networks. Although there are similarities between the delay for the TAS scheduling and the IP network delay, they differ in some significant aspects. The IP network delay is the packet delivery time between any two network nodes in the Wide Area Network (WAN), whereas the delay for the TAS scheduling is the packet delivery time between two adjacent nodes in the Local Area Network (LAN). The legal regulation of the Internet is governed by the layers principle [5]. The lower layers should not handle the protocol of the higher layers. That is, the Internet Control Message Protocol (ICMP) or other higher layer protocols which are used to measure network delay should not be introduced into the TSN switch directly. Besides, although the schemes like [6] increase measurement accuracy to sub-microseconds based on ICMP, they neither consider nor take good advantage of the characteristics of the TSN.

Related work: In the latest research of the TAS scheduling, although most contributions mention or rely on network delay, we have not found an automatic and reliable measurement scheme to build a delay database dedicated to the TAS scheduling yet. NW-PSP [7] defines the cumulative network delay between adjacent nodes as an essential parameter for no-wait packet scheduling problem. The flow reception windows of [4] have to be adjusted on the whole path to account for the cumulative delay per link. However, the cumulative delay is only a conceptual idea including transmission, processing, and propagating delays, under simulated or emulated scenarios. Craciunas et al. [8] also define the delay parameters and discuss the effect of delay on the TAS scheduling. Still, there is no discussion on how to measure the delay. Gavrilut and Pop [9] get the transmission time of a frame between two nodes via theoretical calculations rather than measurement. As one can see that most contributions of the TAS scheduling are concerned with algorithm design, there are few descriptions or discussions on delay measurement schemes.

Furthermore, as early as the TSN proposed, there has been much work [10, 11] on the Time-Triggered Ethernet (TTEthernet). In the TTEthernet, there is a network-schedule for defining transmission and reception time windows for each time-triggered frame being transmitted between nodes. However, in a distributed system, the network-schedule is typically built offline which is not flexible. Besides, building the network-schedule online [12] would increase the computing load of the system and lead to convergence and scalability issues.

Contributions: In this paper, by making good use of the characteristics of the TSN, we propose an SDN-based Time-Sensitive Network Delay (TSND) measurement scheme for the TAS scheduling. Our key contributions can be summarized as follows.

• For the first time to our best knowledge, we introduce delay measurement into the time-sensitive network as a critical step for the TAS scheduling. Different from the OWD and RTT in the IP network, we define the TSND dedicated to the TAS scheduling and prove that TSND is effective for the TAS scheduling.

• Instead of using traditional timestamp mechanism [6], we propose an SDN-based measurement scheme using a time-triggered mechanism. We implement the scheme based on Protocol Oblivious Forwarding (POF), which can easily support new protocols on the data plane. Different from pure software simulation, we build an experimental scene, based on Cavium multi-core network processor and POX (open source SDN controller), which could be closer to the real network environment.

• The clock synchronization deviation and the size difference between the TT flow packet and the TSND measurement packet would affect the TSND to predict the receiving slot accurately. To apply TSND in different network environments, we derive formulas to quantify the impact of clock synchronization accuracy and packet size on the TSND and verify it via experiment. In this case, the formulas provide a good reference for the application of TSND in the TAS scheduling.

The remainder of this paper is structured as follows. We introduce the basic concepts and theoretical analysis of the TSND in Section 2. The design of the TSND measurement scheme is discussed in Section 3. We implement the scheme and present the evaluation results in Section 4. At last, we discuss and summarize our work in Sections 5 and 6.

2. Basic Concepts and Theoretical Analysis.

2.1. TSND definition.

The delay for the TAS scheduling is used to evaluate the flow arrival time on each node while the sending time on the starting node is determined. We notice that the clock is synchronized and the scheduling cycle T is the same between the TSN nodes. Therefore, as Figure 1 shows, we define the TSND between two TSN nodes: the relative time offset of the receiving slot and the sending slot in a scheduling cycle T.

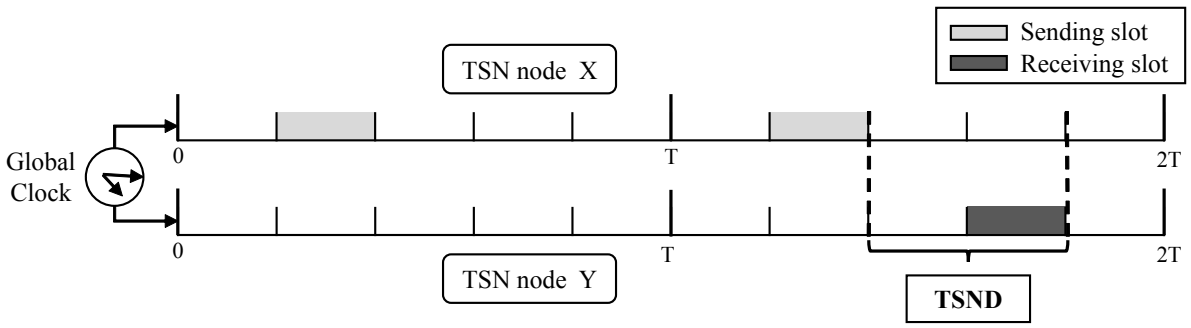

FIGURE 1. The schematic diagram of TSND definition

Further, as shown in Figure 1, in each TSN node, the transmission cycle T is divided into N slots, and each slot is wide enough for an MTU-sized packet to transmit (N can be different in different nodes). In general, we divide T into N slots on average, so that the slot length ℓ is T/N. In the TSN node X, the sending slot is t_x and the slot length is ℓ_x. The receiving slot in the TSN node Y is t_y, and the slot length is ℓ_y. We use Equation (1) to evaluate the TSND.

$$TSND = (\ell_y * t_y + T - \ell_x * t_x)\%T \tag{1}$$

Notably, when the slot number of two TSN nodes is the same, Equation (1) can be simplified to Equation (2) where d is the TSND in units of the slot. To simplify the following analysis, we assume that the slot number of the TSN nodes is N and we use d to represent the TSND. If we have measured the d, we could get the arrival slot t_y of the packet by $(t_x + d)\%N$ while the sending slot t_x on the previous node is determined.

$$TSND = d * \ell = (t_y + N - t_x)\%N * \ell \tag{2}$$

2.2. **TSND analysis.**

2.2.1. *Clock synchronization accuracy.* As we mentioned above, the TSND is not the precise network delay but the relative offset of the sending and receiving slot in the scheduling cycle T, so that the clock of the TSN nodes should be synchronized. For that clock synchronization accuracy is platform-dependent, before we apply the TSND measurement method to different platforms, we need to analyze the impact of clock accuracy on the TSND measurement.

As Figure 2 shows, the clock offset of the receiver relative to the sender, which could be positive or negative, is denoted as ε. Due to space limitations, we only analyze the situation where $\varepsilon < 0$ in detail. We denote ℓ as slot length, t_y as the receiving slot with the clock fully synchronized, and p as the packet arrival time in the t_y. We know that $|\varepsilon|/\ell$ can be expressed by $|\varepsilon|/\ell = k * N + m + r$, where k is a positive integer ($k \geq 0$), m is an integer less than N, and r is a decimal ($0 \leq r < 1$). In this case, the receiving slot t'_y we get with $\varepsilon < 0$ can be expressed by Equation (3).

$$t'_y = \begin{cases} (t_y - m)\%N & (p \geq r) \\ (t_y - m - 1)\%N & (p < r) \end{cases} \tag{3}$$

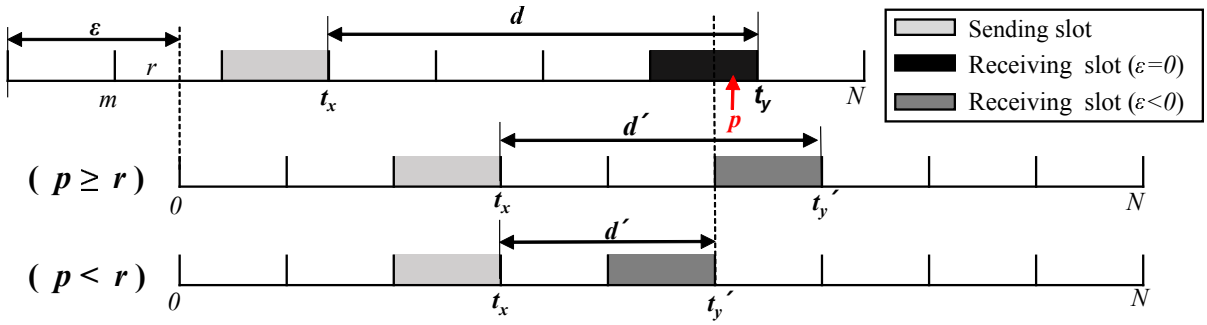

FIGURE 2. The TSND measurement with synchronous clock offset ($\varepsilon < 0$)

Moreover, TSND value d' can be expressed by $d' = (t'_y - t_x)\%N$, where t_x is denoted as the sending slot on TSN node X. With the clock fully synchronized, the TSND value d could be expressed by $d = (t_y - t_x)\%N$. Especially, d is just a reference value that we use to illustrate the effect of clock offset ε on the TSND value d'. Then, according to Equation (3), we can get the relationship of d and d' when $\varepsilon < 0$ by Equation (4), as shown in Figure 2. Similarly, the relationship of d and d' can be expressed by Equation (5) when $\varepsilon \geq 0$.

$$d' = \begin{cases} (d - m)\%N & (p \geq r) \quad (\varepsilon < 0) \\ (d - m - 1)\%N & (p < r) \end{cases} \tag{4}$$

$$d' = \begin{cases} (d + m + 1)\%N & (p \geq (1 - r)) \quad (\varepsilon \geq 0) \\ (d + m)\%N & (p < (1 - r)) \end{cases} \tag{5}$$

It can be known from Equation (4) and Equation (5). Firstly, the TSND value d' eventually approaches the exact value d as m decreases. Secondly, the error between d and d' is mainly determined by the symbol of the offset ε and the value of m and r. When the $|\varepsilon|$ is less than ℓ, m is 0, and the maximum error between d and d' can be kept in two slots. Moreover, since the TT flow arrival time is unchanged in the TSN, while the r changes only within a small range, the TSND value d' would be stable.

2.2.2. *Measurement packet size.* TSND contains the processing delay (d_p) for deciding on which port to forward an incoming packet, the propagation delay d_g of the signal propagating on the link, and the transmission delay (d_t) to serialize the packet on the wire. The d_p and d_g are independent of the packet size. The transmission delay depends on the packet size and port rate. Usually, when the switch is determined, the port rate is also determined. That is, when we use different sizes of the packet to measure TSND, the result would be different. We denote L and M as the size of packets used to measure TSND. Port rate is B for all ports of the TSN node. So the TSND value d (measured with the packet L) and d' (measured with the packet M) can be expressed by Equation (6):

$$\begin{cases} d = \lceil (L/B + d_p + d_g)/\ell \rceil \% N \\ d' = \lceil (M/B + d_p + d_g)/\ell \rceil \% N \end{cases} \tag{6}$$

$$\begin{cases} (L/B + d_p + d_g)/\ell = n + r & (0 \le r < 1) \\ (M/B + d_p + d_g)/\ell = m + z & (0 \le z < 1) \end{cases} \tag{7}$$

As Equation (7) shows, $(L/B + d_p + d_g)/\ell$ equals n slots plus r slots, where n is a positive integer, and r is a decimal ($0 \le r < 1$). So do the m and z. Therefore, the difference (Δd) between d' and d can be expressed by Equation (8) where $\Delta L = M - L$.

$$\Delta d = (\Delta L/(B * \ell) + r - z) \% N \tag{8}$$

As the port rate and slot length increase, the Δd caused by ΔL decreases gradually. Notably, when the value of the $(B * \ell)$ is much larger than ΔL, the maximum Δd can be limited to one slot (since $|z - r| \le 1$). Also, when $\Delta d = 0$, the range of ΔL change can be expressed by Equation (9). It is worth noting that $(z - r)$ is not a value but a range of $[0, 1)$ changes with M and L, so that when k is determined, the TSND value may be the same while the size of measurement packet changes within a range $[0, B * \ell)$.

$$\Delta L = (B * \ell)((z - r) + k * N) \quad (\Delta d = 0) \tag{9}$$

We allocate the sending slot of the TT flows based on the receiving slot calculated by the TSND and the sending slot on the previous node, so the delay and jitter of the TT flows are closely related to the TSND measurement. We denote D as the cumulative network delay of a packet (M) over one TSN node, which contains transmission delay (d_t), processing delay (d_p), and queueing delay (d_p). Same as [7], the time intervals for propagation, processing, queuing, and transmission of each packet are ordered strictly sequentially and do not overlap since we assume a store-and-forward behavior for switches rather than cut-through forwarding. In this case, $D = (d_q + d_p + d_t)$, where $d_t = M/B$. As Figure 3 shows, d_q can be expressed by Equation (10) where d is the TSND value that we measured with the packet L. With the constraints of $(\Delta L/(B * \ell) + r - z) \le N$, D can be expressed by Equation (11). D is related to the relative size of the transmitted packet and the measurement packet rather than the size of the transmitted packet. We can infer that D would have a significant change around the point where $M = L$. Moreover, while the measurement packet (L) is determined and the size of transmitted packet (M) is less

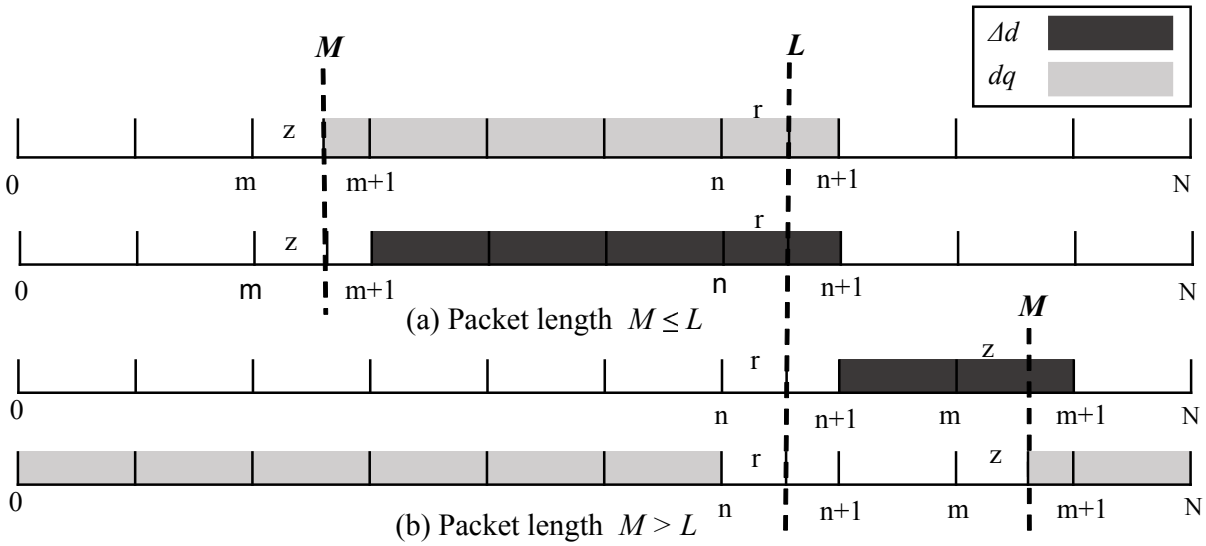

FIGURE 3. The impact of the TSND measurement on the network queuing delay

than L, z would be the only cause of jitter. Since $0 \le z < 1$, the maximum jitter of D is $2 * \ell$.

$$d_q = \begin{cases} (\Delta d + 1 - z) * \ell & (M \le L) \\ (N - \Delta d - z) * \ell & (M > L) \end{cases} \tag{10}$$

$$D = \begin{cases} d_p + L/B + (1 + r - 2z) * \ell & (M \le L) \\ d_p + N * \ell + L/B + (r - 2z) * \ell & (M > L) \end{cases} \tag{11}$$

3. TSND Measurement Scheme.

3.1. **TSND measurement process.** In this paper, we choose a new networking paradigm: Software-Defined Networking [13] (SDN), which effectively decouples the control plane with the data plane and provides a globally optimized resource configuration. Before the measurement of the TSND, the controller has to generate the Link Layer Discovery Protocol (LLDP) [14, 15] flow table and TSND flow table so that the switch can handle the LLDP packet and TSND packet. The event of adding a new TSN switch will trigger the TSND measurement process. As shown in Figure 4, the flow table process and link topology discovery are in the light-colored font. The controller generates a flow table with the prior knowledge of the LLDP and TSND configuration, and then sends the flow table message to switch by the southbound interface protocol, e.g., OpenFlow, POF. The SDN switch stores the flow table in the shared memory that both the management element and forwarding element of the switch can access.

As Figure 4 shows, the solid arrow indicates the processing of the TSND measurement. Topology and Delay Management Information Base (TD-MIB) is responsible for managing and providing the TSND and the topology information for the TAS scheduling. Once the TD-MIB detects that a new TSN switch has joined, the TSND measurement module will be triggered to send a TSND Request packet to the switch. Notably, there are two kinds of the TSND packet: the TSND Request packet from the controller and the TSND Measure packet from the adjacent TSN switch. The SDN switch generates the TSND Measure packet according to the TSND Request packet. Then the TSND Measure packet will be added to the output queue and wait to be scheduled by the TAS engine. Meanwhile, the switch will receive the TSND Measure packet from the adjacent TSN switch. The TSND handler encapsulates the input port, receiving time and other required information

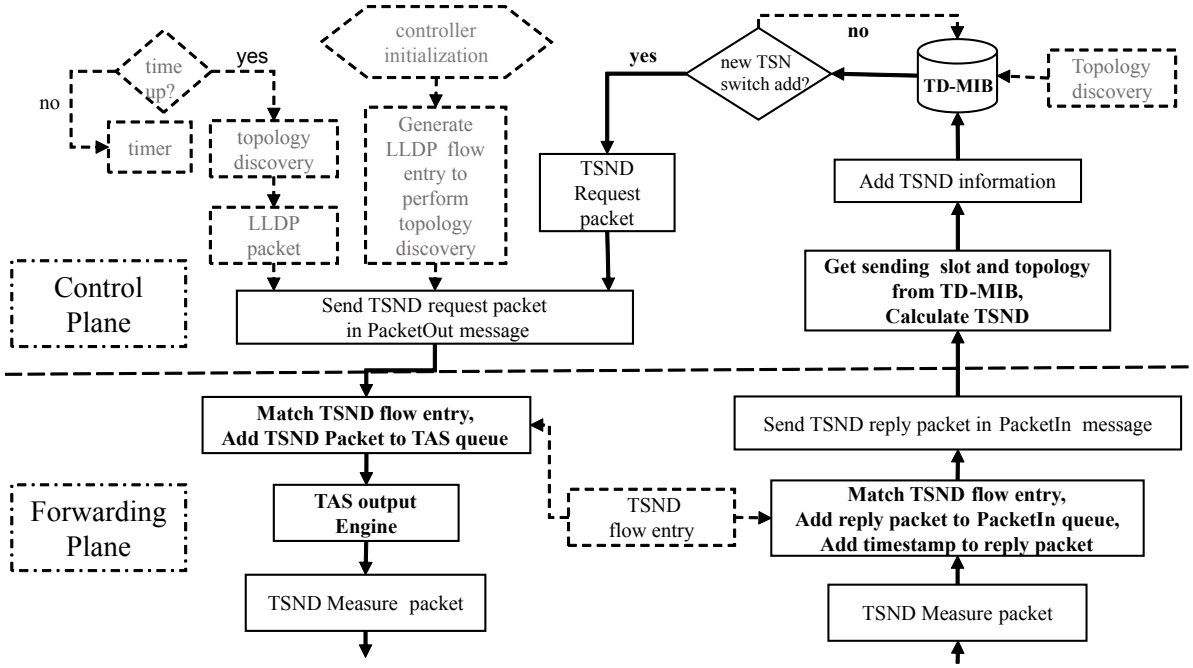

FIGURE 4. The flow chart of the TSND measurement process

	TSN Header			TSND Data Unit			
Preamble	Destination MAC Address (Flow ID)	Source MAC Address	VLAN Tag	Time Slot Info TLV	Time Clock Info TLV	End of TLV	Frame Checksum

TPID	PCP	DEI	VID
2 octets	3 bits	1bit	12 bits
0x8100	7	0	0x000

TLV Type = 1	TLV Info Length (octets)	Receiving Time slot
7 bits	9 bits	4 octet

TSND Data Unit TLV Description		
TLV Type	TLV Name	Usage in TSNDDU
0	End of TSNDDU	Required
1	Time Slot Info	Required
2	Time Clock Info	Optional
3~127	Reserved	Optional

FIGURE 5. The structure of the TSND measurement frame

of the TSND measurement in Packet-In message and sends the message to the controller. The controller parses the Packet-In message, calculates the TSND and adds the TSND information to the TD-MIB.

3.2. **TSND measurement frame structure.** We design the structure of the TSND measurement frame, as Figure 5 shows, the TSND data encapsulated in an Ethernet frame contains a TSN header and a TSND Data Unit (TSNDDU).

The IEEE 802.1Qbv mentions that frames are selected for transmission by the traffic classes, but the TSN frame structure has not been defined yet. There are many ways to distinguish the TSN flow under consideration currently. Craciunas et al. [8] point out that, in particular, a TSN flow is defined by the Priority Code Point (PCP) field and VLAN ID (VID) within the 802.1Q VLAN tag in the Ethernet header. The PCP field and VID are assigned based on the application associated with the flow. Reference to the IEEE 802.1Q [16], the VID value 0x000 indicates that the frame does not carry a VLAN ID. In this case, the 802.1Q tag specifies only a priority (in PCP fields) and is referred to as a priority tag. Therefore, for implementation and experiment, we design

that individual frames belonging to TT flows are identified by VID value 0x000 and PCP value 7 of VLAN-tagged frames.

Furthermore, if the frame belongs to TT flow, the destination MAC will convert to the flow ID. We use flow ID to identify the kind of the TSND packet. The flow ID TSND_REQUEST indicates the TSND Request packet which comes from the controller. The flow ID TSND_MEASURE indicates the TSND Measure packet which comes from the adjacent switch. Nonetheless, the specific value of the TSND_REQUEST and TSND_MEASURE depends on the developing and implementation of the TSN standard.

Given that the function of the TSND may be enhanced as TSN standard is developed and ratified, the TSNDDU is designed as some Type-Length-Value (TLV) structures which contain a required TLV, an optional TLV and an end of TSNDDU TLV. The TLV format is compatible with the LLDP [17] where: type indicates the kind of information; length indicates the length of the information string in octets; value is the specific data of information.

Primarily, the time slot TLV, which is contrasted against other TLVs and trailer fields via a shading in grey, is exclusive required TLV for the TSND measurement that contains the receiving slot used to calculate the TSND. As an optional TLV, the time clock TLV is used for precise network delay measurement so that the network developer can use different time format to measure TSND flexibly.

It is worth noting that the sender chassis ID, output port ID, sending slot, receiver chassis ID, and input port ID are not included in the TSNDDU. The receiver of the TSND Measure packet will send Packet-In message to the controller which contains information such as the receiver chassis ID, input port ID. The network topology is already stored in the TD-MIB, from which we can get the sender chassis ID and output port ID by the receiver chassis ID and port ID. Besides, as for the sending slot, which is configured by the controller, there is no need to tell controller again by the TSND Measure packet.

3.3. **TSND measurement scheme based on POF.** OpenFlow [13] is a widely used southbound interface protocol for the SDN controller and switch. However, the definition of matching field and instruction of the OpenFlow is protocol-specific, which severely impacts the flexibility and programmability of the data plane. Regarding the issues above, Huawei proposed Protocol Oblivious Forwarding [18], as an improvement and extension of the OpenFlow. The POF removes the protocol specific configurations on the forwarding elements by using {offset, length} tuple to define the matching field, where offset indicates the skipped bits from the current cursor within the packet (or metadata) and length indicates the number of bits that should be included in the key starting from the offset position. The POF makes the data plane easily support new protocols and forwarding requirements in the future. Considering that the function of the TSND may be enhanced as the TSN standard is developed and ratified, we choose the POF as an ideal southbound interface protocol in our scheme.

The POF switch processes the TSND measurement in collaboration with the controller. A POF logical switch consists of one or more flow tables, which perform packet lookups and forwarding. Matching starts at the first flow table and may continue to additional flow tables of the pipeline. If a matching entry is found, the instructions associated with the specific flow entry are executed [19]. At the initial stage, the controller sends the TSN configuration and flow tables to switch. The TSN configuration contains Flow_Config and TAS_Config. The Flow_Config contains the mapping of the flow ID and the output queue number which is used to direct the packet to the corresponding output queue. The TAS_Config is the gate control list that contains the mapping of the slot, port ID, and output queue. There are two kinds of the flow table used by the TSND measurement:

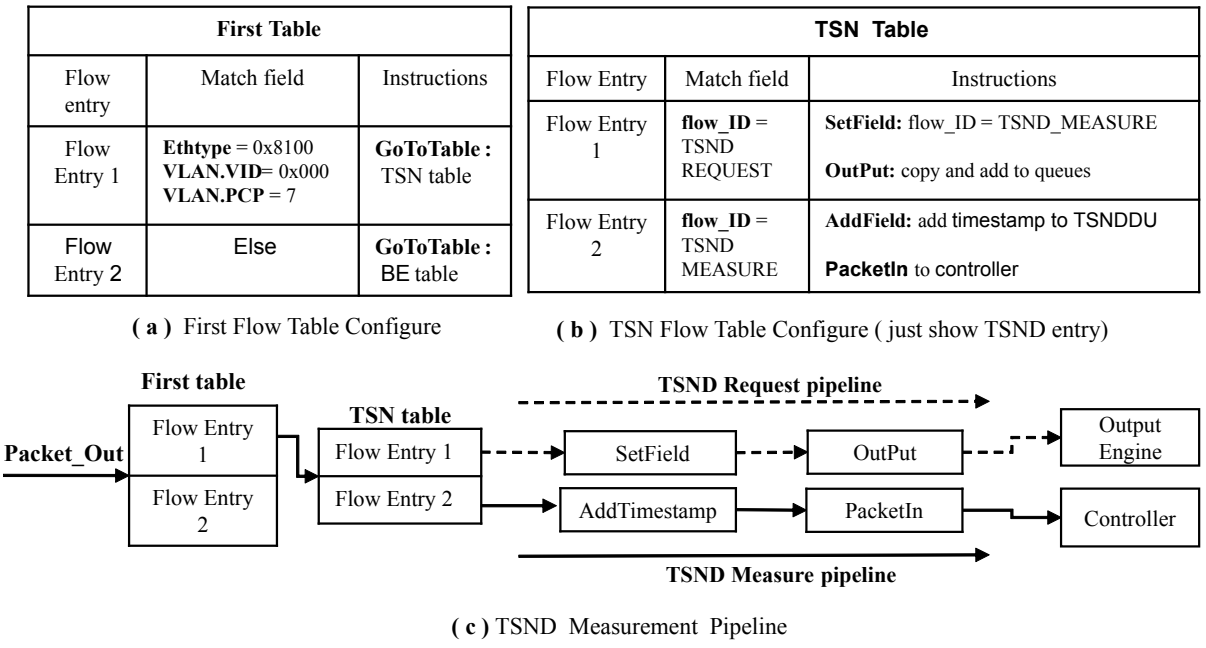

FIGURE 6. The POF flow table and pipeline of TSND measurement process

First Table and TSN Table. All packets need to be processed in the First Table. The First Table is used to distinguish between TSN flows and non-TSN flows, as Figure 6(a) shows. The TSN Table is used to handle the TSN flows including TSND Request flow, TSND Measure flow, and other TT flows, as Figure 6(b) shows.

The controller sends a TSND Request packet with the flow ID TSND_REQUEST to switch in the Packet-Out message. To simplify the data plane, the controller can instruct the switch to submit the packet to the First Table by setting the output port to OFP-P_TABLE in Packet-Out message. Therefore, the TSND Request packet can be treated as if it was received via any of the switch's "normal" ports and is handled by POF process pipeline instead of developing a dedicated TSND module.

TSND Request pipeline. As Figure 6(c) dotted arrow shows, the TSND Request flow will match flow entry 1. The SetField instruction will be executed and the flow ID of the packet will be set to TSND_MEASURE. After that, the packet will be added to the output queue by executing OutPut instruction. In TSN Switch, the OutPut instruction does not send the packet out of switch but buffers the packet to the output queue. We can get the queue number from the TAS_Config by the flow ID in the header of the packet. Notably, for the flow ID TSND_REQUEST, there would be multiple queues that map to each port. In this case, the packet is effectively cloned and added to each queue. The output engine gets a packet from the queue, which is indexed by the port ID and current slot, and sends the packet out through the TAS mechanism.

TSND Measure pipeline. As Figure 6(c) solid arrow shows, the TSND Measure flow matches the flow entry 2. The packet will be added to the Packet-In queue (one kind of output queue). Then we get the current slot from the TAS mechanism, add it to the TSND Measure packet, and send the Packet-In message to the controller. It is worth noting that packets have to be buffered in the output queue before the TAS engine can schedule them so that we record the packet arrival time after the packet has been buffered in the Packet-In queue. The controller gets the receiver chassis ID, port ID and receiving slot by parsing the Packet-In message. Based on the above information, the controller can also get the sender chassis ID, port ID from the TD-MIB since the link topology has

already been recorded. Until now, we get all of the information needed by the TSND, and we can calculate the TSND by Equation (1).

4. Implementation and Evaluation.

4.1. Implementation. The development environment is shown in Table 1. We develop the TSND measurement controller in Python language based on the PCTRL [20], a POF controller extending from the POX [21].

TABLE 1. The TSND measurement development environment

SDN device	Support	Software	Platfrom	Language
POF controller	TSND/flow table configure TD-MIB manage	PCTRL	Linux 4.15	python 2.7
POF switch	TSND measure/PTP flow classifier/meter	PTPv2.0	Cavium OCTEON CN6880 (32 cnMIPS II CPU @1.2GHz)	C

In the data plane, we develop a POF switch based on multi-core network processor Cavium OCTEON CN6880, which contains 32 dual-issue superscalar cores implementing the MIPS64 instruction set and each core has several choices for runtime environment, including Simple Executive standalone mode and Linux mode. We enable one core running in Linux mode to process flow table management and communicate with the controller. We enable three cores running in Simple Executive standalone mode, one for the traffic classification, one for the Precision Time Protocol (PTP) clock synchronization, and one for the TAS scheduling.

4.2. The experiment of TSND measurement. Figure 7 shows the typical TSND measurement scenario. In our experiment, we use the flow ID 0xFFFFFFFFF0 to indicate the TSND Request flow. After the TSND Request pipeline processing in switch 1, the flow ID is set to 0xFFFFFFFFF1 and sent out in the slot 3. The switch 2 identifies the TSND Measure flow by the flow ID 0xFFFFFFFFF1 and adds the packet to Packte-In buffer, after which the switch 2 adds the current slot (such as, slot 9) to the TSND packet and sends a Packet-In message to the controller.

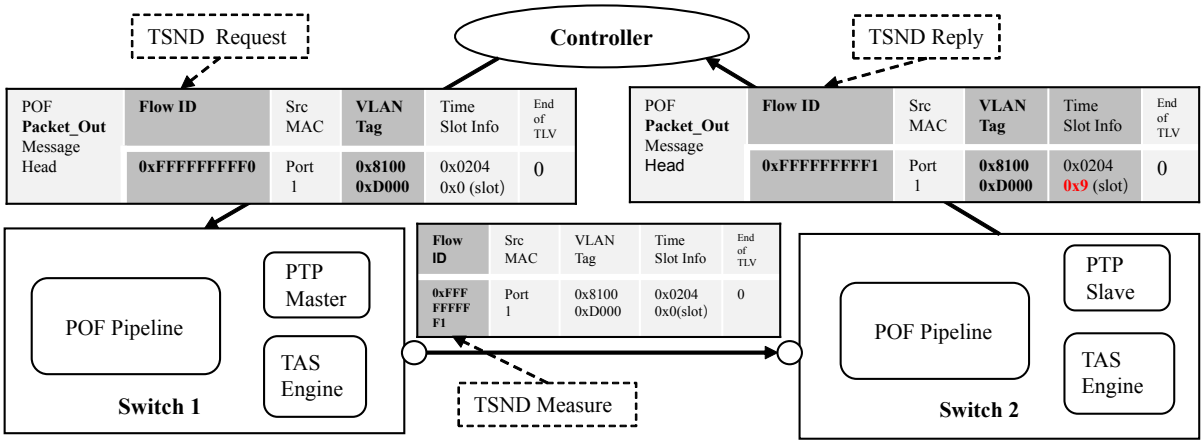

FIGURE 7. The typical TSND measurement scenario

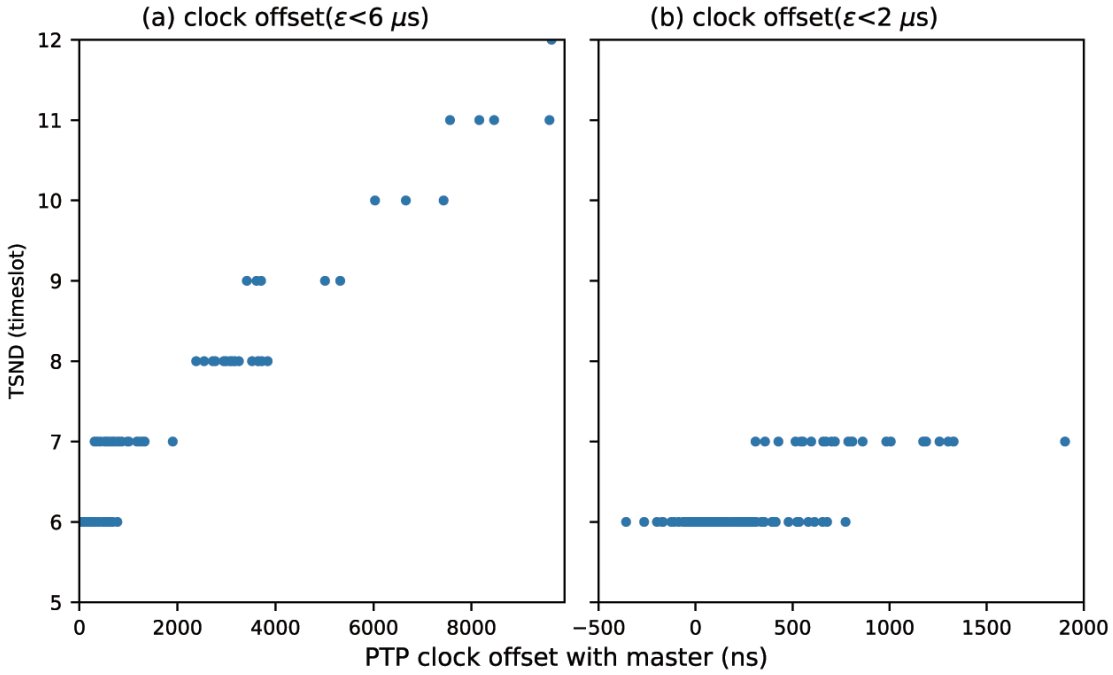

FIGURE 8. The TSND with different PTP clock offset

4.2.1. *PTP clock synchronization.* The Precision Time Protocol (PTP) clock synchronization is required between the TSN nodes before the TSND measurement starts. Figure 7 shows the experimental environment. During the convergence of offset between the PTP master clock and slave clock, we record the TSND results at different PTP clock offset (ε). As Figure 8 shows, the X-axis is the PTP clock offset (ε) and the Y-axis is the TSND value with the clock offset (ε). As Figure 8(a) shows, with the $|\varepsilon|$ decrease, the TSND value begins to decrease. It is in line with our analysis in Section 2.2.1 that the TSND value d' decreases as $|\varepsilon|$ decreases and eventually approaches the exact value d. As we have demonstrated in Section 2.2.1, when the $|\varepsilon|$ is less than ℓ, the error can be kept in two slots. In our experiments, the slot length ℓ is two microseconds. We zoomed in the TSND value while the offset ε is less than two microseconds (Figure 8(b)). It can be seen that the TSND delay changes between slot 6 and 7 and tends to be stable if the ε only changes in a small range (in Figure 8(b), where $|\varepsilon| < 250$ ns).

As we discussed in Section 2.2.1, the higher the clock synchronization accuracy is, the more stable the TSND value will be. However, the high-precision clock synchronization also increases the computational overhead and expensive hardware costs. So we may need to make a trade-off between performance and overhead.

4.2.2. *Different TSND measurement packet size.* The TSND contains the transmission delay which depends on the packet size and port rate. Usually, when the switch is deployed, the port rate is also determined. That is, when we use different sizes of the packet to measure the TSND, the result would be different. Figure 7 shows the experimental environment. Our switch port rate is 1Gbps so that the transmission delay of the 70-byte packet and 1295-byte packet has a maximum difference of 9.8 microseconds. As Figure 9 shows, the TSND varies from 6 to 12 slots while the packet size changes from 70 to 1295 bytes. In our experiment, the slot length is two microseconds, the TSND measurement error caused by packet size is 12 microseconds (6 slots), which is reasonable with one slot error (as the PTP clock offset $|\varepsilon| < 300$ ns). Further, we notice that the TSND is not a linear growth but a step-growth with the packet size increase. It is in good agreement

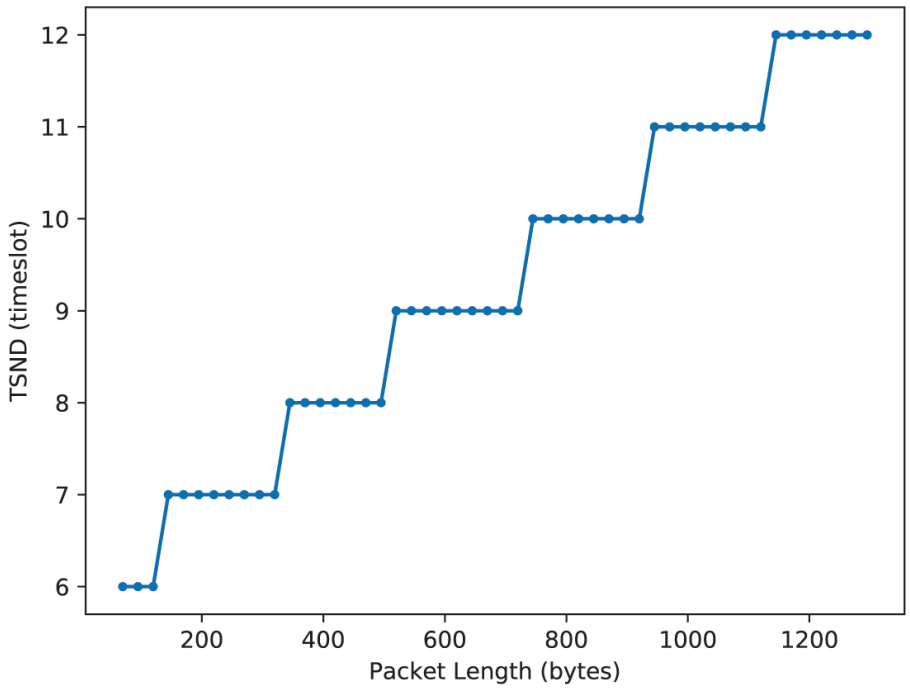

FIGURE 9. The TSND with different measurement packet sizes

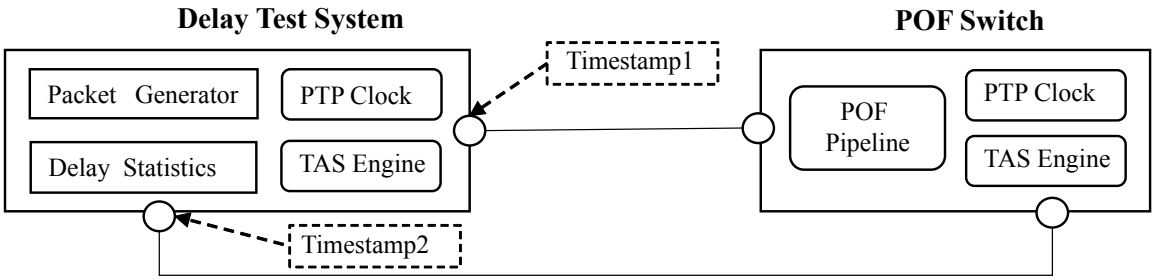

FIGURE 10. The experimental environment of delay and jitter test

with our analysis in Section 2.2.2 that the TSND value may be the same while the size of measurement packet changes within a small range.

4.3. **The experiment of delay and jitter.** As Figure 10 shows, the Delay Test System (DTS) is used to generate TT flows and record the network delay. The Packet Generator constructs the TT flows and buffers the flow to the output queue, and then the TAS engine sends the flow out and returns the timestamp 1 to the Delay Statistics module. Then the TT flow will be processed by the POF pipeline and sent out in the pre-configured slot on the Switch. The DTS records the timestamp 2 and directs the flow to the Delay Statistics module when the returned TT flow arrives. The delay between timestamp 1 and timestamp 2 contains the delay over one TSN switch (D) and two propagation delay (d_g). Since our experimental scene is determined, the d_g is a constant and small enough that we can ignore.

4.3.1. *The TSND evalution.* The TSND is used to predict the receiving slot based on the sending slot and make the optimal assignment of forwarding slot. As Table 2 shows, the TSND is 6 slots between the DTS and the POF Switch, measured with the 70-byte packet, while $|\varepsilon| < 300$ ns and the slot length is two microseconds. As shown in the third

TABLE 2. The data of TSND evaluation

Sending slot	TSND (slots)	Receiving slot (calculate)	Receiving slot (experiment)
8	6	14	14
13	6	19	19
18	6	24	24
23	6	29	29

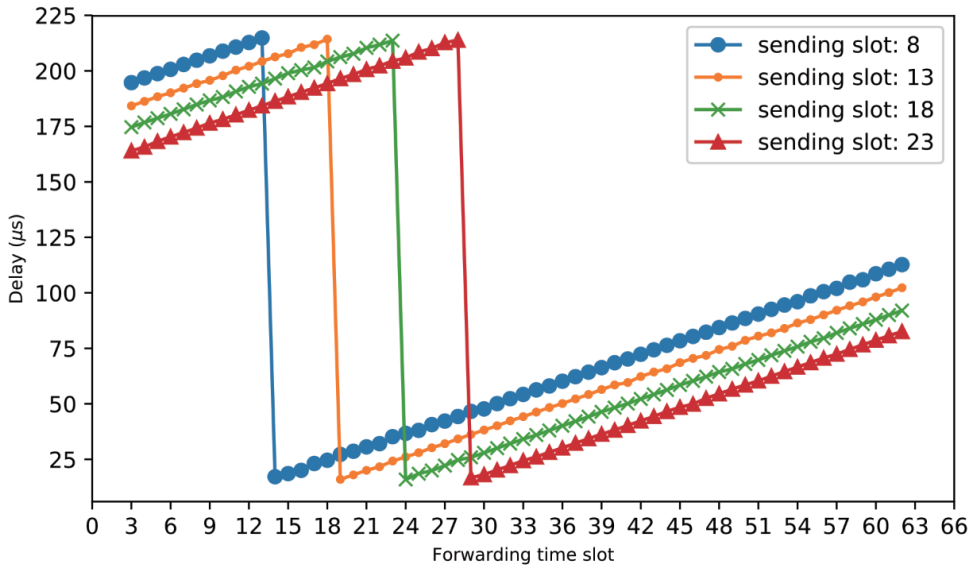

FIGURE 11. The delay of TT flows forwarding in different slots

column of Table 2, we obtain the theoretical receiving slot (marked as "calculate") by the equation: $Receiving\ slot = (Sending\ slot + TSND)\%100$. Then we construct 60 TT flows and send out the flows in the same slot on the DTS. On the Switch, we forward each flow in different slots (from 3 to 62, 0 to 2 reserved for special purposes such as PTP). The delay of each TT flow is shown in Figure 11. The X-axis is the forwarding slot of each TT flow on the Switch. The point on each line represents a TT flow. The different line styles represent a different sending slot on the DTS. We find that every line has the lowest point and we show the forwarding slot of the minimum delay point in the fourth column of Table 2 (marked as "experiment"). The delay is minimum for forwarding packets in the slot in which the packet arrives, since that the queuing delay is minimum. So we can speculate that the forwarding slot of the lowest delay point is the actual receiving slot of the packet in the current experimental environment. By comparing the third column with the fourth column of Table 2, we find that the receiving slots calculated by the TSND are consistent with the receiving slots we obtain from the experiment.

Furthermore, if the TT flow just misses its forwarding slot when it arrives at the switch, the TT flow may need to wait for nearly one TAS cycle ($N-1$ slots). In our experimental scenario, when we send the TT flow in the slot 8 (first line in the legend) on the DTS and forward the flow in the slot 13 on the Switch, the delay is the highest. Moreover, the difference between the highest point and the lowest point is 197.636 microseconds, (in our experimental environment, the cycle is 200 μs), which also proves that the actual receiving slot is 14. That is, the TSND is effective for predicting the arrival slot of the TT flows.

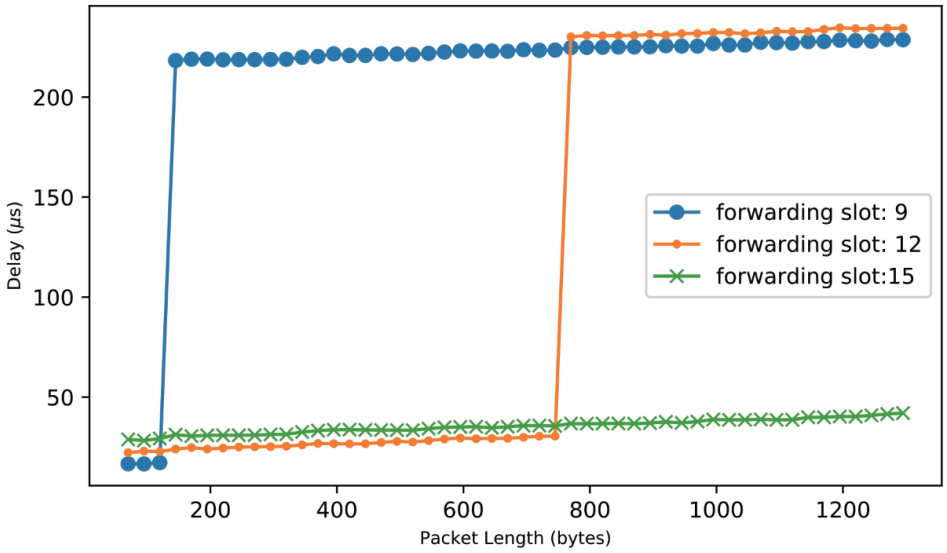

FIGURE 12. The jitter of TT flow with different packet sizes

4.3.2. *Test of delay and jitter.* As we mentioned in Section 2.2.2, the delay and jitter are closely related to TSND measurement packet size. From Figure 9, we can get the TSND value 6, 9, and 12 (slots) measured by the 70-byte, 700-byte, and 1295-byte packet, respectively. We construct a TT flow with 50 packets by Packet Generator of DTS; the packet size increases from 70 bytes to 1295 bytes. On the DTS, we send out the flow in slot 3; on the Switch, we forward the flow in slot 9, slot 12, and slot 15, respectively. As Figure 12 shows, the X-axis is the size of the packet, and the Y-axis is the delay of each packet in the TT flow. The point on each line represents a packet. The different line styles represent a different forwarding slot on the Switch. It can be seen that the delay over the Switch changes significantly around the point where the packet size is 120 bytes and 750 bytes. As we discussed in Section 4.3.1, when the size of the transmitted packet (e.g., 200 bytes) is much larger than the measurement packet (e.g., 70 bytes), the packet arrival slot is no longer 9, but 10, and the packet needs to wait for slot 9 of next cycle before it can be sent, which leads to nearly one TAS cycle delay (in our experimental scenario: 200 microseconds). Unlike the first line and the second line, the third line is very flat. Since all packet sizes of the TT flow we construct are less than 1295 bytes, the actual arrival slot of the TT flow will not exceed the arrival slot (slot 15). In this case, delay jitter would be smooth since all of the packets do not need to wait for an extra cycle.

It is worth noting that the turning point of delay does not appear at the point where the X-axis is 70 or 700 (bytes), but at the point where the X-axis is 120 and 750 (bytes). As discussed in Section 2.2.2, the TSND value may be the same while the size of measurement packet changes within a small range. Besides, from Figure 9 we can also know that the TSND change does occur around the 120 and 750 (bytes).

5. **Discussion.**

5.1. **The advantage of TSND scheme.** The scheme proposed in this paper has limitations that the node performing the TSND measurement needs to support the TSN mechanisms, especially, the precise time synchronization mechanism and the TAS mechanism. Still, by making good use of the characteristics of the TSN, there are some advantages of the scheme.

• The scheme is based on a new networking paradigm: Software-Defined Networking (SDN) which unloads the computational load of the data plane to the controller and provides a globally optimized resource configuration.

• Instead of adding extra delay measurement module like [6], the scheme takes advantage of TAS time-triggered mechanism. The measurement flow can be processed and transmitted as a normal TT flow, in which case, the processing delay of TSND measurement flow is almost the same as the TT flows, and the delay measurement can be performed online at any time.

• The sending slot can be pre-configured, and the measurement packet could be sent in any cycle rather than at a fixed time point so that neither does the measurement packet have to carry sending timestamp nor do we need to send an extra packet to carry the sending timestamp, which reduces the difficulty of implementation.

5.2. **The application of TSND.** Although we do not discuss the TAS scheduling in this paper, we would like to discuss how to choose the appropriate forwarding slot based on the TSND. From the experiment 4.3.1, we verified that the TSND is effective for predicting arrival slot. Nonetheless, we do not suggest forwarding the TT flows in the arrival slot calculated by the TSND. Since if the packet arrives at the end of the slot, forwarding this packet will encroach on the next slot, which would affect the TT flow of that slot. Besides, in Section 4.2.1, we verified that as long as the $|\varepsilon|$ is less than slot ℓ, the accuracy of the TSND measurement can be kept within two slots. That is, in a real network environment where clock synchronization errors exist, the arrival slot may be two slots (maximum) earlier or later than the slot calculated by TSND. Therefore, it is recommended to take the two slots caused by the clock synchronization error into account. For example, if the receiving slot calculated by the TSND is 10, we should set the forwarding slot to 12. In other words, we can reduce the jitter caused by the clock synchronization at the expense of increasing the delay.

Furthermore, in experiment 4.3.2, although the jitter of the third line is minimum, the delay is the longest before the turning point (120 bytes and 750 bytes). In general, if the packet size is uniform, we can get both low delay and low jitter at the same time by allocating slots according to the TSND measured by the average or larger packet. However, if the size of the packet in the TT flow differs significantly, inevitably, we need to make a trade-off between low delay and low jitter.

6. **Conclusion and Outlook.** In this paper, we defined the time-sensitive network delay and proposed an SDN-based TSND measurement scheme. Instead of developing the measurement module separately, the scheme is based on the time-triggered mechanism, which makes good use of the TAS characteristics. Moreover, to apply the TSND in the real network environment effectively, we derived formulas to describe the impact of the clock synchronization accuracy and packet size on TSND measurement. Further, we also discussed how to choose the appropriate forwarding slot based on the TSND and suggested making a trade-off between low delay and low jitter according to the network environment and target application.

However, due to the limitation of experimental environments, we just performed the functional verification of TSND measurement under light load conditions, without introducing the influence of background flows. With the establishment of a deterministic delay channel, we will further study how to improve the performance of the TSND in complex network scenarios.

Acknowledgment. This work was supported by the National Science and Technology Major Project under Grant 2017ZX03001019. We also gratefully acknowledge the editor and all reviewers for their valuable suggestions.

REFERENCES

[1] A. Nasrallah, A. Thyagaturu, Z. Alharbi, C. Wang, X. Shao, M. Reisslein and H. Elbakoury, Ultra-Low Latency (ULL) networks: A comprehensive survey covering the IEEE TSN standard and related ULL research, *IEEE Communications Surveys Tutorials*, 2018.

[2] N. Yamanaka and K. Shiomoto, DTM: Dynamic transfer mode based on dynamically assigned short-hold time-slot relay, *IEICE Trans. Communications*, vol.E82-B, no.2, pp.439-446, 1999.

[3] IEEE standard for local and metropolitan area networks – Bridges and bridged networks – Amendment 25: Enhancements for scheduled traffic, *IEEE Std 8021Qbv-2015 (Amendment to IEEE Std 8021Q-2014 as amended by IEEE Std 8021Qca-2015, IEEE Std 8021Qcd-2015, and IEEE Std 8021Q-2014/Cor 1-2015)*, 2015.

[4] B. Bansal, *Divide-and-Conquer Scheduling for Time-Sensitive Networks*, Master Thesis, 2018.

[5] L. B. Solum and M. Chung, The layers principle: Internet architecture and the law, *SSRN Electronic Journal*, vol.79, no.3, p.815, 2003.

[6] N. I. Vinogradov, E. S. Sagatov and A. M. Sukhov, Device for measuring one-way network delay with microsecond accuracy, *Telecommunications Forum Telfor (TELFOR)*, pp.133-136, 2015.

[7] F. Dürr and N. Nayak, No-wait packet scheduling for IEEE time-sensitive networks (TSN), *Proc. of the 24th International Conference on Real-Time Networks and Systems*, pp.203-212, 2016.

[8] S. S. Craciunas et al., Scheduling real-time communication in IEEE 802.1Qbv time sensitive networks, *Proc. of the 24th International Conference on Real-Time Networks and Systems*, pp.183-192, 2016.

[9] V. Gavrilut and P. Pop, Scheduling in time sensitive networks (TSN) for mixed-criticality industrial applications, *The 14th IEEE International Workshop on Factory Communication Systems (WFCS)*, pp.1-4, 2018.

[10] H. Kopetz, A. Ademaj, P. Grillinger and K. Steinhammer, The time-triggered Ethernet (TTE) design, *The 8th IEEE International Symposium on Object-Oriented Real-Time Distributed Computing (ISORC'05)*, pp.22-33, 2005.

[11] K. Steinhammer, P. Grillinger, A. Ademaj and H. Kopetz, A time-triggered Ethernet (TTE) switch, *Proc. of the Conference on Design, Automation and Test in Europe: Proceedings*, European Design and Automation Association, pp.794-799, 2006.

[12] Z. Zheng, F. He and Y. Xiong, The research of scheduling algorithm for time-triggered Ethernet based on path-hop, *IEEE/AIAA the 35th Digital Avionics Systems Conference (DASC)*, 2016.

[13] N. McKeown et al., OpenFlow: Enabling innovation in campus networks, *ACM SIGCOMM Computer Communication Review*, vol.38, no.2, pp.69-74, 2008.

[14] G. Tarnaras, E. Haleplidis and S. Denazis, SDN and ForCES based optimal network topology discovery, *The 1st IEEE Conference on Network Softwarization (NetSoft)*, 2015.

[15] F. Pakzad, M. Portmann, W. L. Tan and J. Indulska, Efficient topology discovery in OpenFlow-based software defined networks, *Computer Communications*, vol.77, pp.52-61, 2016.

[16] IEEE standard for local and metropolitan area networks – Media access control (MAC) bridges and virtual bridged local area networks, *IEEE Std 802.1Q-2011 (Revision of IEEE Std 802.1Q-2005)*, 2011.

[17] IEEE standard for local and metropolitan area networks – Station and media access control connectivity discovery, *IEEE Std 802.1AB-2009 (Revision of IEEE Std 802.1AB-2005)*, 2009.

[18] H. Song, Protocol-oblivious forwarding: Unleash the power of SDN through a future-proof forwarding plane, *Proc. of the 2nd ACM SIGCOMM Workshop on Hot Topics in Software Defined Networking*, pp.127-132, 2013.

[19] *OpenFlow Switch Specification 1.5.1*, Open Networking Foundation, https://www.opennetworking.org/wp-content/uploads/2014/10/openflow-switch-v1.5.1.pdf, 2015.

[20] *https://github.com/USTC-INFINITELAB/PCTRL*.

[21] *https://github.com/noxrepo/pox*.