Analyzing Worst-Case Delay Performance of IEC 61850-9-2 Process Bus Networks Using Measurements and Network Calculus

Huan Yang
Computer Science and Engineering Department
Lehigh University
19 Memorial Drive West
Bethlehem, Pennsylvania 18015, USA
huy213@lehigh.edu

Liang Cheng
Computer Science and Engineering Department
Lehigh University
19 Memorial Drive West
Bethlehem, Pennsylvania 18015, USA
cheng@cse.lehigh.edu

Xiaoguang Ma
ABB Inc.
655 Century Point
Lake Mary, Florida 32746, USA
xiaoguang.ma@us.abb.com

ABSTRACT

In power substation automation systems (SASs) based on IEC 61850, conventional hardwired process connections are being replaced by switched Ethernet. To ensure system reliability and responsiveness, transmission of critical information required by protection and control tasks must satisfy hard delay constraints at all times. Therefore, delay performance conformance should be taken into consideration during the design phase of an SAS project. In this paper, we propose to study the worst-case delay performance of IEC 61850-9-2 process bus networks, which generally carry non-feedforward traffic patterns, through the combination of measurements and network-calculus-based analysis. As an Ethernet switch supports dedicated interconnections between its multiple interfaces, our proposed approach converts a non-feedforward network into feedforward ones by introducing service models for its individual output interfaces instead of modeling it in its entirety with a single service model. To derive practical delay bounds that can be validated against measurement results, our approach not only constructs traffic models based on the idiosyncrasies of process bus network and switched Ethernet, but also establishes service models of networking devices by taking measurements. Results from our case studies of both feedforward and non-feedforward process bus networks show that the proposed combination of network calculus and measurement-based modeling generates accurate delay bounds for Ethernet-based substation communication networks (SCNs). The proposed approach can thus be adopted by designers and architects to analytically evaluate worst-case delay performance at miscellaneous stages of SAS design.

CCS CONCEPTS

•Networks →Network performance modeling; Network performance analysis; Network measurement;

KEYWORDS

Substation communication networks, IEC 61850-9-2 process bus network, non-feedforward networks, worst-case delay, switched Ethernet, network measurements, network calculus, smart grids.

ACM Reference format:
Huan Yang, Liang Cheng, and Xiaoguang Ma. 2016. Analyzing Worst-Case Delay Performance of IEC 61850-9-2 Process Bus Networks Using Measurements and Network Calculus. In *Proceedings of the 8th ACM International Conference on Future Energy Systems, Hong Kong, China, May 17–19, 2017 (e-Energy'17),* 11 pages.
DOI: 10.1145/nnnnnnn.nnnnnnn

e-Energy'17, Hong Kong, China

DOI: 10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

To take advantage of modern technologies, such as switched Ethernet, power substation automation systems (SASs) across the globe are being modernized by adopting international standard IEC 61850 [9]. To ensure system reliability and responsiveness, critical information carried by a substation communication network (SCN) has to be delivered within hard time constraints. In IEC 61850-5 [17], maximum transmission times for different class of messages are explicitly specified. For instance, in a substation requiring performance classes P2 and P3, the maximum transmission time for Type 4 sampled values must not exceed 3 milliseconds [13]. The worst-case delay performance of a particular SCN is quantified by the set of worst-case network-induced delays experienced by all its traffic flows. Evidently, for networked systems controlling critical infrastructure such as the power grid, it is the worst-case rather than the average delay performance that is of practical interest to SAS designers and architects. Before an SAS project is

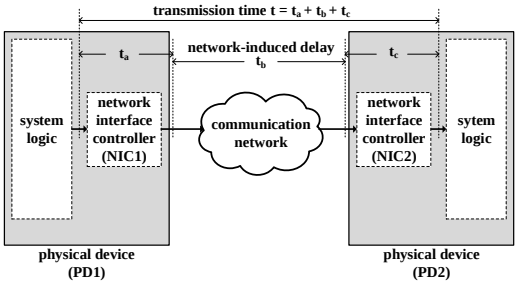

Figure 1: Transmission time and its components defined in IEC 61850-5 [17].

commissioned, it is therefore of vital importance to ensure that all the worst-case delay performance requirements are satisfied.

Two major interfaces defined by IEC 61850 are the IEC 61850-9-2 [16] interface (i.e., process bus) and IEC 61850-8-1 [15] interface (i.e., station bus). Network architecture of SASs based on IEC 61850 has been presented in several research projects, such as [9], [21] and [27]: An IEC 61850-9-2 process bus carries sampled values (SVs) from synchronized merging units (MUs), which monitor a section of a power system through multiple current and voltage sensors, to various protective relays. In addition, an IEC 61850-8-1 station bus transmits generic object-oriented substation event (GOOSE) messages, greatly reducing the need for inter-equipment hardwired signals. As IEC 61850 gains momentum in the energy sector, its application has been expanded to inter-substation communication [14], as well as information exchange between substations and control centers [20]. Network engineering guidelines as well as time constraints for these emerging application scenarios are also defined [18, 19]. As illustrated in Fig. 1, network transmission time (i.e., network-induced delay) is one of the major components of the transmission time defined in IEC 61850-5 [17]. In essence, protective relays (also known as intelligent electronic devices) rely on sampled values delivered by the process bus to infer system status and collaboratively perform assorted control/protection tasks. Hence, it is necessary to analyze the worst-case (i.e., maximum) network-induced delays experienced by sampled value messages (SVMs), which is dependent on various factors such as network topology and traffic characteristics.

At the bare minimum, an IEC 61850-9-2 process bus network organizes multiple merging units and transmits their sampled value messages to one or multiple protective relays. Synchronization among merging units can be achieved via hardwired signals or via precision time protocol (PTP) messages [12]. Similarly, inter-equipment signals among protective relays may be transmitted via hardwired signals, a separate station bus network, or over the same network implementing the process bus [11]. In this work, we focus on finding the worst-case network-induced delays for sampled value messages on a process bus network where synchronization and inter-equipment GOOSE signals are transmitted out-of-band [13]. Even under such settings, the traffic pattern of a process bus network is generally non-feedforward (see Sec. 2.2) because SVMs from synchronized merging units are typically broadcast or multicast to their subscribers (e.g., protective relays) via a process bus consisting multiple Ethernet switches. As state-of-the-art worst-case delay analyses target feedforward networks (e.g., [7, 24]), it is infeasible to directly adopt these techniques to analyze IEC 61850-9-2 process bus networks.

In this paper, we propose an approach to deriving the accurate upper bounds on the worst-case delays for IEC 61850-9-2 process bus networks with arbitrary (i.e., either feedforward or non-feedforward) traffic patterns. In essence, our proposed approach is a combination of measurements and network calculus: Non-queuing delays introduced by Ethernet switches are modeled through measurements, whereas network calculus is applied to find worst-case delay bounds on network-induced delays consisting of both queuing and non-queuing delay components. The contributions of our work are as follows:

(1) To obtain realistic delay bounds that are sufficiently tight using network calculus, we establish service models of Ethernet switches by taking proper measurements and exploiting the fact that Ethernet switches have sufficient capacity to support multiple simultaneous interconnections. In addition, we construct accurate traffic models for merging units, leveraging both their synchronized operations and the serialization effect of Ethernet switches.
(2) Instead of leveraging conventional techniques in network calculus to deal with non-feedforward networks, we show that a non-feedforward traffic pattern on switched Ethernet can be converted into feedforward ones, facilitating the applications of existing analytical techniques targeting feedforward networks.
(3) To make it less laborious to evaluate SCN design alternatives, we propose a hybrid approach that takes measurements only during switch modeling and relies on the theory of network calculus to analytically evaluate different SCN designs.
(4) Our case studies of both feedforward and non-feedforward process bus networks show that the proposed combination of measurement and network calculus produces accurate delay bounds that can be validated against measurements.

We envision that our proposed approach can be utilized by SAS architects as a tool for evaluating worst-case delay performance at various stages of system design.

2 RELATED WORK

2.1 Evaluations and Analyses of Delay Performance of SCNs Based on IEC 61850

The importance of ensuring delay performance conformance of SCNs based on IEC 61850 has motivated investigations exploiting a variety of methods. In [25], discrete-event simulation is conducted to verify the delay performance of SCNs. To facilitate the study of SCNs with different traffic patterns and/or network topology, simulation models of merging units and protective relays are proposed and constructed. In [13], measurements are taken from merging units in a real transmission substation. Delay performance of a process bus network with a feedforward traffic pattern is then investigated in controlled lab environment. Both discrete-event simulation and measurement-based experiment allow us to study the delay performance of a particular SCN. However, it is hard to generalize the knowledge gained from such case-specific evaluations: When changes need to be made to a particular SCN, new experiment has to be set up to re-evaluate the worst-case delay performance because network-induced delays is usually dependent on network topology and device configurations (e.g., data rates).

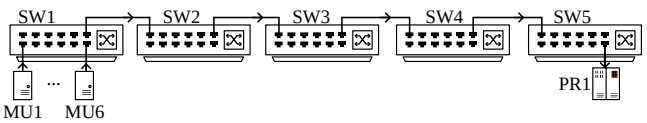

Figure 2: The feedforward process bus network studied in [13]. Six merging units (MUs) transmit sampled value messages (SVMs) to the protective relay (PR1) through the SCN consisting of five Ethernet switches (SW1∼5).

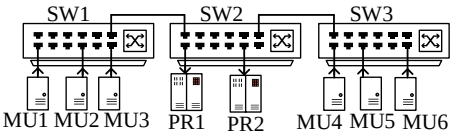

Figure 3: A three-switch non-feedforward process bus network. Under the current scheme of assigning identifiers, the paths (SW3, SW2) and (SW3, SW2, SW1) carrying broadcast sampled values from MUs 4∼6 cannot be represented by increasing sequences.

Furthermore, results obtained from these methods are sometimes not conclusive since boundary or extreme scenarios may still be overlooked even after extensive experiments/simulation.

To find out the upper bounds on worst-case delays, network calculus is applied in [10]. The derived delay bounds are then validated against simulation results, showing the feasibility of employing network-calculus-based analysis in SAS design. However, the SCNs analyzed in [10] consist of a single Ethernet switch, whereas the actual design of an SCN (e.g., the number of switches to use and the topology of the network) depends on various physical constraints (e.g., the locations and number of current and voltage sensors [27]). The traffic patterns on SCNs can be complex and non-feedforward, whereas the toolbox [3] used in [10] is designed for feedforward networks. An approach to analyzing the worst-case delay performance of non-feedforward traffic patterns on SCNs is yet to be devised.

2.2 Feedforward vs. Non-Feedforward Networks

Given a network with multiple traffic processing nodes, we can assign unique integers as node identifiers. We say that a network has a feedforward traffic pattern if the paths traveled by all its traffic flows can be represented by a set of monotonically increasing sequences of node identifiers [7]. Take the multiple-switch process bus network depicted in Fig. 2 as an example. If we model each switch as a network node and assign integer identifiers as shown in Fig. 2, the traffic pattern is feedforward because traffic flows from all merging units traveled through the same path, which can be represented by the sequence (SW1, SW2, SW3, SW4, SW5).

On the other hand, we note that SCNs designed in existing SAS projects (e.g., [27]) generally carry non-feedforward traffic patterns. For instance, the process bus network depicted in Fig. 3 collects sampled values from two groups of merging units separate from each other and far away from the control room, where protective relays are deployed. Suppose that both protective relays subscribe to the sampled values published by all the merging units and that SVMs are broadcast over the process bus, the traffic pattern is non-feedforward because there is always at least a path that has to be represented by a non-increasing sequence. It should be noted that, for the network in Fig. 3, we can interchange the identifiers for SW2 and SW3 so that the flows from MUs $4 \sim 6$ to PR1 can be represented by an increasing sequence. However, the sequence representing the path for the flows from MUs $4 \sim 6$ to MUs $1 \sim 3$ is still not monotonically increasing. Such a non-feedforward traffic pattern is tricky to analyze in that a proper starting point for deduction cannot be found: If we want to find the delay induced by SW1 for SVMs sent by MUs $1 \sim 3$, we need to assess the "interference" of the SVMs arriving at SW1 from MUs $4 \sim 6$. This in turn requires us to analyze the SVM flows from MUs $4 \sim 6$ starting from SW3, which leads us back to MUs $1 \sim 3$ because their passage through SW3 introduces similar interference.

We note that non-feedforward traffic patterns are the most general ones that can be found on switched Ethernet: The path of any traffic flow should not form a cycle since it makes no sense for a device to transmit its own messages back to itself. Once a message is received by a certain sink node, it may be replayed back to the original source, but this scenario can be modeled by two flows with the same (or similar) traffic profile passing through the same set of nodes in forward and inverse orders. Although non-feedforward scenarios can be studied via simulation or measurements, an analytical approach suitable for IEC 61850-9-2 process bus network can significantly reduce the effort on measurements/simulation and is yet to be developed.

2.3 Worst-Case Delay Analysis Using Network Calculus

Based on min-plus algebra, network calculus [4, 8] provides a set of models and theorems on the worst-case delay and backlog performance of communication networks. For feedforward networks with multiple nodes and flows, the problem of tightening the delay bounds derived using network calculus is of practical interest for researchers of real-time networked systems. Two categories of analytical methods, namely compositional methods and optimization-based methods, have been proposed [2]. A compositional method is essentially a set of rules for applying various network calculus theorems along the path traveled by a particular traffic flow (also termed as the flow of interest). Several well-known compositional methods are proposed and discussed in [23, 24]. An optimization-based method finds the worst-case delay by solving multiple optimization problems with constraints derived from network-calculus theorems and definitions. Several optimization-based methods targeting network nodes with different service disciplines, such as arbitrary multiplexing [5, 23], first-come first-served [7], and fixed priority [6], have been proposed. Optimization-based approach achieves tighter bounds by exhaustively searching for the maximum possible delays within constrained trajectory spaces. Our analysis in this work falls into the category of compositional methods, which typically have low computational complexity. We show that delay bounds can be tightened through taking proper measurements for process bus networks and exploiting the traffic characteristics of merging units.

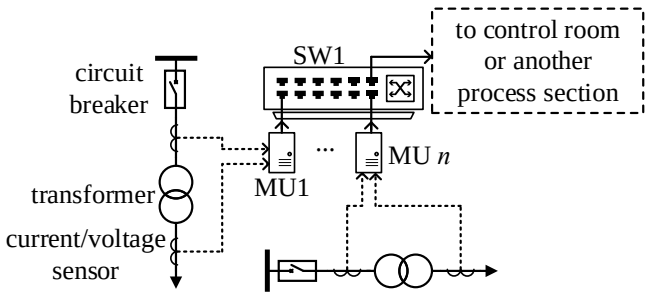

Figure 4: A group of n merging units monitoring a section of a power system process.

The fact that it is hard to find a starting point for deduction on a non-feedforward network has been observed in classic network-calculus work, such as [8], which also demonstrates that simple non-feedforward networks can be analyzed by introducing stopped sequences. This technique is applied in [1] to analyze software-defined networks where the traffic patterns are also non-feedforward. However, applying this approach to non-feedforward networks with broadcast/multicast sources can be intricate because the number of stopped sequences to be introduced grows rapidly with the number of nodes and sources. To address this issue, this work takes a measurement-based approach that properly models Ethernet switches and converts a non-feedforward network into feedforward ones.

3 MODELING TRAFFIC FLOWS FROM MERGING UNITS

In an IEC 61850-9-2 process bus network, merging units are synchronized to ensure that they are able to carry out sampling operations simultaneously and generate a "snapshot" of the power system. For a power system operating at 60 Hz, the length of its system cycle is simply $\frac{1}{60}$ seconds. During each system cycle, a merging unit perform sampling operations at its configured sampling rate (e.g., 80 samples per cycle). The flow of sampled value messages generated by a particular merging unit i can thus be represented by a real-valued, non-negative, non-decreasing function $F_i(t)$, which is the cumulative traffic volume observed from the Ethernet output interface of i up to time instant t. $F_i(t)$ is called the arrival process of the flow generated by i. Without loss of generality, we define any arrival process $F(t)$ in this work for $t \geq 0$ and assume that $F(0) = 0$. In network calculus [4, 8], a bounding traffic model known as arrival curve, is defined to characterize arrival process:

Definition 3.1. Given an arrival process $F(t)$, a real-valued, non-negative, non-decreasing function $\alpha(t)$ defined for $t \geq 0$ is an arrival curve of $F(t)$ if and only if

$$\forall t \geq s \geq 0 : F(t) - F(s) \leq \alpha(t - s).$$

If $\alpha(t)$ is an arrival curve of $F(t)$, we write $F(t) \sim \alpha(t)$. A commonly used form of arrival curve is the leaky-bucket arrival curve [8, 23] $\alpha(t) = \sigma + \rho \cdot t$, where ρ is the average rate component and σ the burstiness component. If an arrival process $F(t)$ has an arrival curve $\alpha(t) = \sigma + \rho \cdot t$, we write $\alpha(t) \sim (\sigma, \rho)$.

3.1 Simultaneous Arrival and Traffic Aggregation

Although arrival curves of traffic flows generated by individual sources are typically constructed separately, we propose to construct arrival curves for different groups of merging units in IEC 61850-9-2 process bus network to exploit its following characteristics:

(1) Merging units are synchronized and generate sampled values simultaneously.
(2) A group of merging units are typically deployed to collectively monitor a section of a power system process.
(3) At each section of the system process, data from the merging units can be collected by an Ethernet switch provided that it has an adequate number of network interfaces.
(4) Merging units of the same group are configured to work at the same sampling rate and generate sampled value messages of the same size.

Let us consider the process section depicted in Fig. 4. Since the n merging units are synchronized, they generate sampled value messages nearly simultaneously at the beginning of each sampling cycle. By connecting them to an Ethernet switch, their sampled value messages are put onto the process bus network for transmission. Suppose that the size of sampled value messages is L and that the length of the sampling cycles is T. For any individual sampled value message flow $F_i(t)$ generated by merging unit i ($1 \leq i \leq n$), we have $F_i(t) \sim (L, \frac{L}{T})$, which is depicted in Fig. 5a. To find the arrival curve of the aggregate output from the n merging units in Fig. 4, the multiplexing/aggregation theorem [8] can be leveraged:

Theorem 3.2. *Given a set of n arrival processes $F_1(t), F_2(t), \ldots, F_n(t)$ and their respective arrival curves $\alpha_1(t), \alpha_2(t), \ldots, \alpha_n(t)$, we always have*

$$\Sigma_{i=1}^{n} F_i(t) \sim \Sigma_{i=1}^{n} \alpha_i(t).$$

Thus, the arrival curve for the aggregate output of n merging units is simply $\Sigma_{i=1}^{n} F_i(t) \sim (n \cdot L, \frac{n \cdot L}{T})$. We note that for synchronized merging units generating sampled values simultaneously, the arrival curve of their aggregate output given by Theorem 3.2 is tight. We term this phenomenon as simultaneous arrival, which is illustrated in Fig. 5b. Since transmission time for all sampled values must not exceed 3 ms [13], it is unnecessary to distinguish individual traffic flows generated by MUs of the same group: The worst-case delay experienced by the aggregate flow upper bounds that of any individual flows. In other words, simultaneous arrival facilitates worst-case delay analysis with aggregate flows.

3.2 Arrival Curve of Serialized Switch Output

After n simultaneously generated sampled value messages pass through switch SW1 in Fig. 4, they will be serialized and put onto the output interface connecting to another switch or a locally deployed protective relay. The inter-frame gap specified by the Ethernet specification is 12 bytes, which is much smaller than the size of a sampled value message (e.g., 126 bytes according to [13]). For 100 Mbps (Fast) Ethernet, this inter-frame gap translates into a $\frac{12 \times 8 \text{ bits}}{100 \text{ Mbps}} = 0.96\ \mu$s time interval. Hence, the arrival curve $\alpha(t) = n \cdot L + \frac{n \cdot L}{T} \cdot t$ is not strictly tight (in other words, slightly loose) for the output of SW1, but it is still a good approximation to its tight arrival curve.

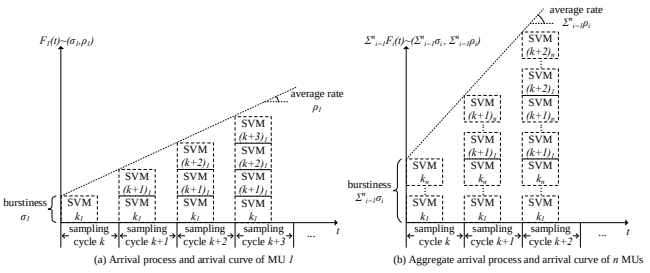

Figure 5: Arrival processes and arrival curves of an individual merging unit (MU1) as well as a group of n merging units. The sampled value message generated by MUi at sampling cycle k is denoted by SVM k_i.

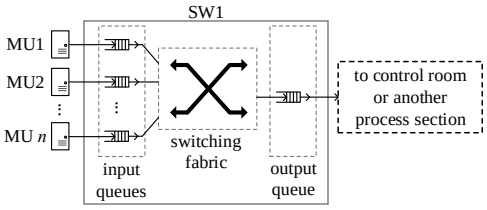

Figure 6: A group of n merging units connecting to Ethernet switch SW1. Note that we do not assume any particular implementation scheme for the internal components of SW1.

In our proposed approach, we always group together merging units connected to the same switch to exploit simultaneous arrival. This is because synchronized merging units monitoring the same power system section typically use the same sampling rate. For the corresponding switch output, we still use the arrival curve of the aggregate input as an approximation. As we will see in Sec. 5.1, accurate arrival curves can help us tighten the delay bounds.

4 MODELING ETHERNET SWITCHES

In network calculus, traffic processing capability of a networking device is modeled by another bounding model, which is known as service curve [4, 8]:

Definition 4.1. Given a traffic processing network node with input arrival process $F^{\text{in}}(t)$ and output arrival process $F^{\text{out}}(t)$, a real-valued, non-negative, non-decreasing function $\beta(t)$ is the service curve of the node if and only if

$$\forall t \geq 0 : F^{\text{out}}(t) \geq \inf_{0 \leq s \leq t} \{F^{\text{in}}(t) + \beta(t-s)\} \equiv (F^{\text{in}} \star \beta)(t),$$

where $[\cdot]^+$ denotes the operation $\max\{\cdot, 0\}$.

The $\star$ operator is used to denote the min-plus convolution operation. A commonly-used type of service curve is the rate-latency service curve $\beta(t) = R[t-T]^+$, where R is the processing capacity and T models constant non-queuing delay. For a rate-latency service curve $\beta(t) = R \cdot [t-T]^+$, we introduce the shorthand $\beta(t) \leftrightarrow (R, T)$.

Sampled values may need to pass through a series of Ethernet switches to reach a subscriber. Take the process bus network depicted in Fig. 2 as an example. The sampled values travel through five switches in tandem in order to reach the protective relay. The concatenation theorem [8] is well-suited to model such a tandem of switches:

Theorem 4.2. *Suppose that a flow $F(t)$ passes through m network nodes in tandem and that the service curves offered by these nodes are $\beta_1(t), \beta_2(t), \ldots, \beta_m(t)$, respectively. The service curve $\beta(t)$ offered by the tandem of these m nodes to $F(t)$ is*

$$\beta(t) = (\beta_1 \star \beta_2 \star \ldots \star \beta_m)(t) \equiv \Pi_{j=1}^{m} \beta_j(t).$$

To apply this theorem to a process bus network, we need to first find out the rate-latency service curves offered by individual Ethernet switches.

4.1 Identifying Rate Component for the Service Curve of Process Bus Ethernet Switch

For an Ethernet switch, we use the rate component of its service curve to capture its traffic processing capacity. Let us consider the single-switch scenario in Fig. 4, which is re-drawn in Fig. 6. For sampled value messages from merging unit i, the entry point into switch SW1 is the input interface connecting i. We assume that the data rate of the output interface of i matches that of the input interface of SW1, which is typically the case. The switching fabric provides dedicated interconnections between input and output interfaces of SW1, allowing different pairs of input/output interfaces to communicate simultaneously. However, as shown in Fig. 6, the traffic pattern of a process bus network inevitably creates a bottleneck at the output interface: Suppose that the switching fabric offers sufficient capacity. As the number of merging unit increases, the output interface connecting to another switch or a protective relay will eventually overflow.

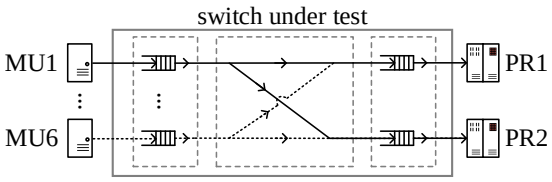

Figure 7: Experiment settings to extract latency component of the service curve of the switch under test. Merging units (MU) and protective relays (PR) are emulated by a computer server. Note that only one MU is activated at a time.

Suppose that the input queue, switching fabric, and output queue, are modeled by three work-conserving links [8] with rate-latency service curves $\beta_{\text{in}}(t) \leftrightarrow (R_{\text{in}}, T_{\text{in}})$, $\beta_{\text{sw}}(t) \leftrightarrow (R_{\text{sw}}, T_{\text{sw}})$, and $\beta_{\text{out}}(t) \leftrightarrow (R_{\text{out}}, T_{\text{out}})$, respectively. Theorem 4.2 suggests that the service curve $\beta(t)$ offered by the tandem of these three work-conserving links is

$$\begin{aligned} \beta(t) &= (\beta_{\text{in}} \star \beta_{\text{sw}} \star \beta_{\text{out}})(t) \\ &= \min\{R_{\text{in}}, R_{\text{sw}}, R_{\text{out}}\} \cdot [t - (T_{\text{in}} + T_{\text{sw}} + T_{\text{out}})]^+. \end{aligned} \quad (1)$$

According to the specifications of Ethernet switches on the market, their switching fabrics are capable of supporting all interfaces to communicate at wire rate in full-duplex mode (in other words, perfect parallelism is offered by the switching fabric and it will not become a bottleneck). Therefore, we have $\min\{R_{\text{in}}, R_{\text{sw}}, R_{\text{out}}\} = R_{\text{out}}$. In process bus networks, the rate component of the service curve offered by an Ethernet switch to a merging unit can thus be the wire rate of the output interface traveled by its sampled values. For a multicast or broadcast flow, an Ethernet switch replicates the flow in its switching fabric, so multiple service curves are offered by the switch, corresponding to all the directions it travels (i.e., for all the output interfaces that process the broadcast/multicast flow).

4.2 Extracting Latency Component for Service Curve from Measurements

From Eq. 1, we also find that the latency component of the service curve offered by a switch is simply the non-queuing latencies induced by its internal components. Note that delays introduced by other necessary physical components, such as Ethernet cables, should also be included in practice. In our approach, we rely on network calculus to find the worst-case delays caused by queuing. However, we still need to factor in non-queuing delay components, which are hard to determine merely based on technical specifications of switches. The latency component models several categories

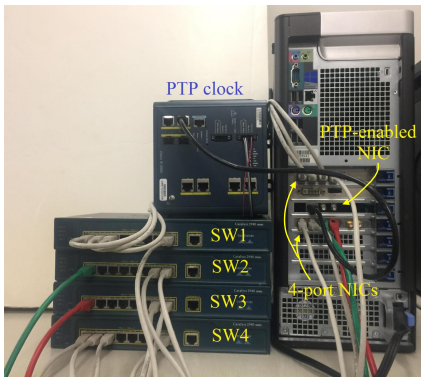

Figure 8: Components and organization of our test bed.

of non-queuing delays, including propagation delay on Ethernet cables, serialization delay at input/output interfaces, as well as processing delays (e.g., checksum verification) caused by various internal components of an Ethernet switch.

In our approach, we propose to capture non-queuing delays by taking measurements: By injecting sampled value messages at low rates and measuring the delays they experience, we use the maximum delay observed as a good approximation to the latency component because few messages are enqueued under light workloads and they are dequeued rapidly. Note that such a measurement-based approach essentially relies on the assumption that additional delays observed during instantaneous traffic burstiness (e.g., simultaneous arrivals of SVMs) are caused by queuing. If this assumption holds for Ethernet switches (at least under normal workloads observed on process bus networks [13]), measuring non-queuing delays intrinsic to the switches will enable us to find an appropriate latency component for their service curves. To verify the validity of this assumption, we conduct experiments on four Ethernet switches of the same model. All the switches come with eight full-duplex 100 Mbps interfaces and one full-duplex 1 Gbps interface. For all the experiments conducted in this work, we only utilize the 100 Mbps interfaces.

4.2.1 Test Bed Settings. The components and organization of our test bed is shown in Fig. 8. We install a PTP-enabled network interface card [26] on a computer server and connect it to a PTP master clock. Two four-port network interface cards (NICs) are installed on the server to inject and receive timestamped packets. To improve system throughput, we write our packet sender and receiver programs using the netmap fast packet I/O framework [22]. To measure the delay experienced by a packet, timestamps are generated when the packet is sent and received. The difference between the two timestamps is used to represent the delay induced by the network.

4.2.2 Measurement Procedures. We inject sampled value messages into the input interface of an Ethernet switch to impose the traffic pattern shown in Fig. 7. The length of the sampled value messages is 126 bytes, and the message layout is detailed in [13]. To emulate broadcast traffic pattern on process bus, we set to destination MAC address of all injected messages to $FF:FF:FF:FF:FF:FF$. For each switch under test (SUT), we take the following measurements step by step:

Table 1: Delay Measurements from One of the Switches Under Test (SUT) Serving MU1 or MU2 (in microseconds)

SVMs per second	MU1			MU2		
	min.	avg.	max.	min.	avg.	max.
1	15.2	16.3	17.2	15.3	16.1	17.1
10	15.1	15.9	16.7	14.6	16.0	17.3
100	16.0	16.6	16.9	15.5	16.1	17.4
1000	15.7	16.2	17.0	14.9	15.8	16.9
2000	15.4	16.3	17.1	15.4	16.6	17.6
3000	15.2	16.3	17.5	15.8	16.2	17.0
4000	14.9	16.1	17.4	15.3	15.9	17.4
4800	15.6	16.5	17.6	15.1	16.4	16.8

(1) Choose the SVM flow generated by merging unit i as our flow of interest, where $i \in \{1, 2, \ldots, 6\}$. Inject SVMs at an initial rate, which should be low enough. All the other MUs are not activated (i.e., they do not generate SVMs and simultaneous arrival at the SUT is avoided). In our experiment, we start with one SVM per second. This step is run for five minutes and the delays experienced by all the injected SVMs are recorded.
(2) Increase the data rate of the selected MU and then repeat Step (1). We increase the data rate up until $60 \times 80 = 4800$ SVMs per second, which is the data rate used by the MUs in the process bus networks studied in our case studies.
(3) Choose another flow of interest and repeat Steps (1) and (2) until interfaces utilized by all the MUs are tested.

The measurement results for one of the SUTs, with MU1 or MU2 activated separately, are summarized in Table 1. We observe that the variances of the delays introduced by the SUT at different data rates are small. At each rate, the majority of the measured values are close to the mean. The results observed on interfaces utilized by the other MUs are identical to those of MU1 and MU2. Similarly, if we change the interfaces utilized by MUs and relays (e.g., interchanging the interfaces for MU1 and PR1) while maintaining the same traffic pattern between sources (i.e., MUs) and sinks (i.e., relays), the distributions of the delays observed have similar characteristics. In addition, if an extra MU is activated to emulate simultaneous arrival, the worst-case delays increase significantly (see Sec. 5). Furthermore, this observation is consistent across all four SUTs. The fact that the maximum delays do not increase significantly at different data rates suggests that the SUTs offer sufficient processing capacity under the imposed workloads. The maximum latencies observed for all four SUTs are about the same, so we use the maximum value from Table 1, i.e., 17.6 μs, as the latency component for the service curves of all the SUTs. We choose the maximum value here because Theorem 5.1 requires that the service curve lower bounds the processing capacity of a switch. Choosing an aggressively small latency component may lead to underestimating the delay bounds.

It should be noted, however, different values may be chosen for individual switches if there are significant differences among the maximum latencies observed (e.g., the Ethernet switches employed by an SCN come from different manufacturers or have different switching fabric implementations). Moreover, we do not consider the scenarios where MUs inject SVMs at wire rate because some of

the SVMs are discarded by the output interfaces of switches due to buffer overflow. The dropped SVMs effectively experience infinite network-induced delays. In a process bus network, we pay special attention not to cause SVM drops because successful delivery of SVMs is of critical importance to the implementation of protection and control tasks. Additionally, we note that Theorem 5.1 is not applicable under such scenarios because the assumption of infinite buffer size is violated (see Sec. 5.1).

5 CASE STUDY I - A FEEDFORWARD SINGLE-SWITCH PROCESS BUS

5.1 Worst-Case Delay Analysis of Feedforward Process Bus Networks

Once arrival curve of aggregate input to an Ethernet switch and its service curve are constructed, the classic result from network calculus gives the upper bound on the worst-case delay incurred by the switch [4, 8]:

Theorem 5.1. *Suppose that a traffic flow $F^{in}(t) \sim \alpha(t)$ passes through a network node with service curve $\beta(t)$ and an infinite-size buffer and that the corresponding output flow is represented by $F^{out}(t)$. The delay experienced by the data bit entering the node at time t is denoted by $d(t) = \inf\{u \geq 0 : F^{in}(t) \leq F^{out}(t+u)\}$ and we have*

$$d(t) \leq \inf\{u \geq 0 : \alpha(s) \leq \beta(s+u), 0 \leq s \leq t\} \equiv h(\alpha(t), \beta(t)).$$

According to this theorem, the delay bound is simply the maximum horizontal distance between the arrival curve and the service curve, i.e., $h(\alpha(t), \beta(t))$. Note that this theorem is only applicable if the buffer of the node can accommodate all the incoming data at all times. To ensure that such an assumption of infinite buffer size holds in practice, we check the input and output traces in our experiments to ensure that no SVMs are dropped. We observe that our switches occasionally discard SVMs when the data rate at its output interface is close to wire rate. Such scenarios are avoided in our experiments since a process bus network design leading to SVM loss should be further refined before commissioning.

Theorem 5.1 also explains why we want to obtain tight arrival curves (see Sec. 3) and service curves (see Sec. 4). A loose arrival curve results in overestimating the worst-case delays and so does an overly tight service curve (e.g., naively assuming that the latency component is zero). By constructing realistic arrival and service curves, existing methods for worst-case delay analysis, such as those reviewed in Sec. 2.3, can be applied to feedforward SCNs and generate accurate delay bounds.

5.2 Worst-Case Delay of a Single-Switch Process Bus Network

To validate the quality of the delay bounds generated by our proposed combination of network calculus and measurements, we first study the simple single-switch process bus network depicted in Fig. 4 and Fig. 6 with up to seven merging units. Note that the traffic pattern on such a process bus is always feedforward because there is only one possible path with a single node.

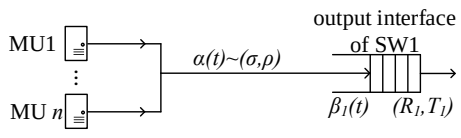

Figure 9: Simplified network diagram of the single-switch process bus network in Fig. 6.

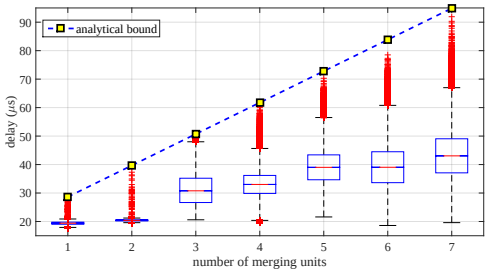

Figure 10: Analytically derived delay bounds vs. measurement results for the single-switch process bus network in Fig. 6 with different numbers of merging units activated.

5.2.1 Network-Calculus-Based Analysis. As proposed in Sec. 3.1, we construct the arrival curve for the aggregate input to SW1. We use our test bed to emulate n merging units monitoring a 60-Hz power system at 80 samples per cycle, for $1 \leq n \leq 7$. The traffic pattern of this process bus can be simplified into the network diagram shown in Fig. 9. The length of the SVMs we use is 126 bytes. To use our traffic model, we take the 8-byte preamble and 4-byte CRC trailer required by Ethernet frames into account. The effective message size is hence $L = (8+126+4)\times 8 = 1104$ bits, and the length of the sampling cycle is $T = \frac{1}{60\times 80} \approx 208.33\mu s$. Therefore, the burstiness component of the aggregate input is $\sigma = n\times 1104$ bits, and the average rate is $\rho = \frac{n\times 1104}{208.33}$ Mbps. According to our discussion in Sec. 4.1 and Sec. 4.2, we take $R_1 = 100$ Mbps and $T_1 = 17.6$ μs for SW1. From Theorem 5.1, we find that the worst-case delay experienced by SVMs sent by any of the n MUs is given by

$$d_{\max} = T_1 + \frac{n \cdot L}{R_1}. \tag{2}$$

5.2.2 Measurement-Based Evaluation. In our evaluation, we start with $n = 1$. After injecting emulated SVMs and recording the delays for 10 minutes, we increase n by 1 and repeat the experiment until $n = 7$. Fig. 10 compares the measured delay values against the analytically derived bounds given by Eq. 2. Our results show that the delay bounds derived using our proposed combination of network calculus and measurements are sufficiently tight: For all tested values of n, the analytical bounds are at most 3.5% larger than the maximum delays we observe. We note that the traffic pattern carried by this process bus is rather simple, and the rest of this paper will focus on non-feedforward traffic patterns.

6 CONVERTING A NON-FEEDFORWARD NETWORK INTO FEEDFORWARD ONES

As sampled value messages from multiple merging units converge toward Ethernet switch output interfaces connecting protective relays or other switches, we can model an individual Ethernet switch by m work-conserving links, each of which corresponds to one of its utilized output interfaces. Take the process bus illustrated in Fig. 3 as an example. We re-draw this network into the simplified network diagram in Fig. 11 using the modeling technique in Sec. 4.1.

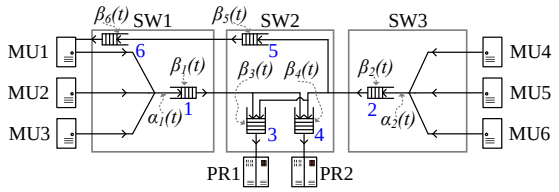

Figure 11: Network diagram of the process bus in Fig. 3, simplified for worst-case delay analysis. Note that we re-assign unique integer identifiers to the work conserving links (i.e., the integers in blue).

Note that most of the paths among merging units are omitted. Only the path from MUs 4∼6 to MU1 is shown for the purpose of discussion. By introducing multiple work-conserving links, the SVM flows within the switching fabric of each switch are completely decoupled because the bottlenecks are the output interfaces. If we assign unique integer identifiers to the work-conserving links, it is now possible to find an assignment scheme such that all the paths can be represented by monotonically increasing sequences. For instance, suppose that we use the assignment scheme shown in Fig. 11. The path (SW3, SW2, SW1) in Fig. 3 can now be represented by the increasing sequence (2, 5, 6), and it is evident that the non-feedforward traffic pattern is converted into a feedforward one.

In switch Ethernet, each switch interface is utilized by a particular device (e.g., merging units, protective relays, and another switch). Therefore, the following theorem can be proved for switched Ethernet:

Theorem 6.1. *By introducing work-conserving links for individual switch output interfaces, any non-feedforward traffic pattern on a switched Ethernet can be converted into a feedforward one.*

Proof. To prove that the network is feedforward after introducing work-conserving links, all we need to show is that there exists an integer identifier assignment scheme such that all paths of the originally non-feedforward network can be represented by monotonically increasing sequences of identifiers. Let us represent Ethernet switches by virtual network nodes with an infinite number of embedded work-conserving links (i.e., output interfaces). For a network with m switch and n sinks, without loss of generality, suppose that each sink has only one input interface. Similarly, each pairwise physical connections between two switches uses up an output interface of the upstream switch. Any particular path connecting k ($1 \le k \le m$) nodes consists of k pairwise physical connections among the nodes (i.e., k–1 connecting the switches and the last one connecting a switch to a sink). For any non-feedforward network that has paths that cannot be represented by increasing node (i.e., switch) identifiers, we can find an integer identifier assignment scheme for the work-conserving links that makes the network feedforward as follows. First, we choose an arbitrary path p and assign integers such that its representation is an increasing sequence. Then, we select a new path p'. We say that a pair of paths are in conflict if all possible assignment schemes enabling one of them to be represented by an increasing sequence make the representation of the other not monotonically increasing. If p' is not in conflict with p, we extend the assignment scheme used for p such that p' is also represented by an increasing sequence. In this case, p' may or may not introduce new work-conserving links. If p' is in conflict with p, then it cannot be implemented using only the physical pairwise connections of p. This can be shown by contradiction: If p' can also be implemented by a subset of the work-conserving links utilized by p, its representation must be a certain subsequence of the representation of p, which contradicts the definition of a conflicting path. In this case, p' introduces certain new physical pairwise connections between the nodes by occupying work-conserving links that have not been utilized. Evidently, these new connections are introduced by the part of p' that cannot be represented by subsequence of the representation of p. We can then walk along p' and assign new integers to the newly utilized work-conserving links, by prepending, inserting, and/or appending new integers, and then adjust the scheme such that both p and p' are represented by increasing sequences. By repeating this procedure, we can eventually represent all paths by increasing sequences. □

Note that if the number of output interfaces of a switch is finite, only the number of possible paths that need to pass through this procedure is constrained. To sum up, for a process bus based on switched Ethernet with a non-feedforward traffic pattern, we can always convert it into a feedforward network by introducing one work-conserving link per utilized switched output interface: For any particular flow of interest, we start introducing work-conserving links from the last node on its path and walk backward. At each node, if the output interface utilized by the flow of interest also serves other flow(s), we also need to trace backward along the paths of these "interfering" flows and introduce work-conserving links accordingly. Since a non-feedforward traffic pattern does not have any cycle and Theorem 6.1 shows that such a network of work-conserving links is feedforward, we are able to reach the source nodes of these paths eventually. At this point, we obtain a network diagram allowing us to analyze the worst-case delay experienced by the flow of interest. Note that this technique has been extensively used in prior work on feedforward networks [5–7], and our proposed combination of network calculus and measurements extends its application to non-feedforward patterns on switched Ethernet. It should also be noted that the feedforward traffic pattern created by introducing work-conserving links at switch output interfaces can be used to created multiple network diagrams for different flows of interest, with work-conserving links as network nodes.

7 CASE STUDIES OF NON-FEEDFORWARD PROCESS BUS NETWORKS

To evaluate the delay bounds derived by our approach for non-feedforward process bus networks, we study two multiple-switch examples. Note that the switches used in our experiments are the ones tested in Sec. 4.2.

7.1 Case Study II - A Three-Switch Process Bus

In this experiment, we analyze the network-induced delays for the network shown in Fig. 3. As discussed in Sec. 6, the originally non-feedforward traffic pattern is first converted to a feedforward one. The flows of our interest and notations for arrival/service curves are explicitly shown in Fig. 11. All six merging units are

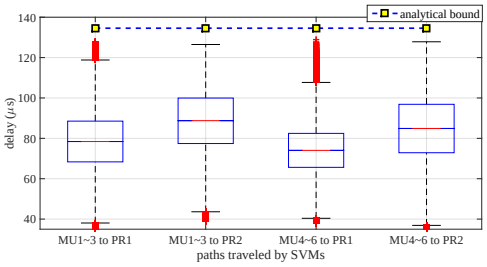

Figure 12: Analytically derived delay bounds vs. measurement results for SVM flows between merging units (MU) and protective relays (PR) on the process bus network in Fig. 3.

activated, and we have $\alpha_1(t) \sim (3 \cdot L, \frac{3 \cdot L}{T})$ and $\alpha_2(t) \sim (3 \cdot L, \frac{3 \cdot L}{T})$. As for service curves, we have $\beta_i(t) \leftrightarrow (R_1, T_1)$, where $R_1 = 100\ Mbps$ and $T_1 = 17.6\ \mu s$, for $1 \leq i \leq 6$.

7.1.1 Network-Calculus-Based Analysis. The worst-case delay bounds for the SVM flows from MUs 1 ~ 3 to PR1 and PR2 as well as from MUs 4 ~ 6 to PR1 and PR2 are the same due to symmetry. Note that an implicit assumption that enables us to exploit such symmetry is that the switching fabrics have sufficient capacity to replicate SVM messages for broadcasting/multicasting, which is verified in Sec. 4.2. Let us analyze the path from MUs 1 ~ 3 to PR1. At node 1, Theorem 5.1 suggest that the worst-case delay it introduces is $d_1 = T_1 + \frac{3 \cdot L}{R_1}$. At node 3, we use $\alpha_1(t)$ and $\alpha_2(t)$ to approximate the arrival curves of the output of node 1 and node 2, respectively (see Sec. 3.2). By applying Theorems 3.2 and 5.1, the worst-case delay introduced by node 3 is $d_3 = T_1 + \frac{6 \cdot L}{R_1}$. The worst-case delay bound for this flow is thus $d_{\max} = d_1 + d_3$.

For the flow from MUs 4 ~ 6 to MU1, note that nodes 2, 5, and 6 serve only this particular flow. Therefore, we can first apply Theorem 4.2 to find the concatenated service curve offered by the tandem of the three nodes. Then, we apply Theorem 5.1 and find that the delay bound for this flow is $d'_{\max} = 3 \cdot T_1 + \frac{3 \cdot L}{R_1}$. Note that this delay bound is not of practical interest in SASs, and we derive this bound simply for the purpose of evaluation.

7.1.2 Measurement-Based Evaluation. We set up the network in Fig. 3 and let it execute for 10 minutes. For the flows between merging units and protective relays, the measured delays and the derived bounds are shown on Fig. 12. We observe that all measured delays are upper bounded by the analytically derived bounds, which are at most 6.4% greater than the maximum observed values. The maximum delay we observe for the flow from MUs 4 ~ 6 to MU1 is 83.2 μs, whereas its derived delay bound is 85.92 μs. The results of this experiment show that network calculus provides accurate delay bounds for non-feedforward IEC 61850-9-2 process bus network once we convert it into a feedforward one.

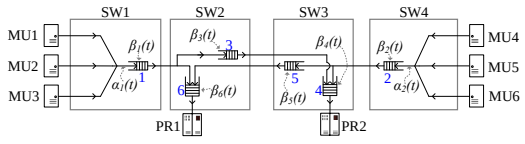

Figure 13: Network diagram of the four-switch process bus studied in Sec. 7.2. Unique integer identifiers are assigned to the work-conserving links to facilitate network-calculus-based worst-case delay analysis.

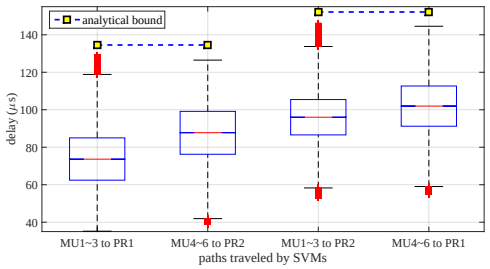

Figure 14: Analytical bounds vs. measurement results for SVM flows received by protective relays in the four-switch process bus network shown in Fig. 13.

7.2 Case Study III - A Four-Switch Process Bus

To further validate our proposed approach, we construct another process bus network consisting of four Ethernet switches. In contrast to the three-switch network studied in the previous case study, the SVM flow from MUs 1 ~ 3 to PR2 travels through three work-conserving links (which is also the case for the flow from MUs 4 ~ 6 to PR1). In [13], it is shown through measurements that inserting an additional Ethernet switch into the feedforward path traveled by SVMs introduces an extra 15-μs latency (using switch of a particular model, which is not reported). This is because SVMs simultaneously arriving at the first switch (i.e., SW1 in Fig. 2) are serialized and do not experience queuing delays at succeeding switches (i.e., SW2~5). This case study shows that the same observation also applies to non-feedforward traffic patterns. The network diagram of this process bus is shown in Fig. 13. In this experiment, we focus on worst-case delays experienced by SVMs traveling toward protective relays, and paths between merging units are all omitted. The configuration of the MUs remains unchanged. By introducing work-conserving links at the output interfaces along our flows of interest, the network in Fig. 13 is transformed into a feedforward one. Using the models discussed in previous sections, we have $\alpha_1(t) = \alpha_2(t) \sim (3 \cdot L, \frac{3 \cdot L}{T})$ and $\beta_i(t) \leftrightarrow (R_1, T_1)$ for $1 \leq i \leq 4$.

7.2.1 Network-Calculus-Based Analysis. Exploiting symmetry, we derive the delay bounds for the SVM flows from MUs 1 ~ 3 to PR1 and PR2. First, let us analyze the flow from MUs 1 ~ 3 to PR1. At node 1, Theorem 5.1 suggests that the worst-case delay it induces is $d_1 = T_1 + \frac{3 \cdot L}{R_1}$. At node 6, we use the arrival curves $\alpha_1(t)$ and $\alpha_2(t)$ to approximate the serialized output from node 1 and node 5, respectively. By applying Theorem 3.2 and then Theorem 5.1, we have $d_6 = T_1 + \frac{6 \cdot L}{R_1}$.

For the flow from MUs 1~3 to PR2, we note that it solely enjoys the service curves of nodes 1 and 3. Therefore, Theorem 4.2 can be applied to obtain their concatenated service curve. Then, Theorem 5.1 gives the worst-case delay incurred by these two nodes, which is $d_{1,3} = 2 \cdot T_1 + \frac{3 \cdot L}{R_1}$. At node 4, we again use $\alpha_1(t)$ and $\alpha_2(t)$ to approximate the serialized output from node 1 and node 2. By applying Theorem 3.2 and then Theorem 5.1, we have $d_4 = T_1 + \frac{6 \cdot L}{R_1}$.

7.2.2 Measurement-Based Evaluation. We set up the four-switch process bus using our test bed and let it execute for 10 minutes. The measured delays and the analytical bounds for the SVM flows received by the protective relays are shown in Fig. 14. All the measured delays are upper bounded by the corresponding analytical bounds. In fact, the delay bounds are at most 6.3% greater

than the observed maximum delays. These analytical and measurement results suggest that the worst-case delay experienced by SVMs increases by about 17.6 μs if an additional switch is inserted, which is consistent with the observation for feedforward SCNs in [13]. Both case studies in this section show that our proposed combination of network-calculus-based worst-case delay analysis and measurement-based switch modeling produces realistic delay bounds that are close to measured values.

8 DISCUSSION

8.1 Incorporating Network-Calculus-Based Analysis into SAS Project

The proposed approach complements existing SAS design tools and significantly reduces the efforts required to find out accurate worst-case delay bounds. Network-calculus-based analysis can be applied throughout multiple phases of an SAS project. At early stages of an SAS design when measurements are not available (e.g., models of Ethernet switches are not finalized or plans for equipment procurement are pending review), arrival and service curves can be constructed from system specifications. As discussed in Sec. 3.1, arrival curve for the SVM flow generated by a particular merging unit can be derived based on its designed sampling rate, operating frequency of the power system, as well as the size of its SVMs. Based on site-specific constraints, such as the locations of primary power system equipment, it is also possible to determine how MUs should be grouped together. Arrival curves for aggregate SVM flows can then be found. Similarly, service curve can be roughly established by setting the latency component to 0 (i.e., temporarily using an overly tight, rate-based service curve). The analytical procedures demonstrated in Sec. 7 can then be applied to obtain estimated delay bounds. Although these analytical results provide insights into the worst-case delay performance of candidate design alternatives, it should be noted that the impacts of non-queuing delays are not factored in and underestimation will probably occur.

At later stages (e.g., when Ethernet switches are procured), more realistic service curves can be established by taking measurements following the steps outline in Sec. 4.2. In addition, arrival curves may also need to be updated to reflect system changes made after the last time network-calculus-based analysis is conducted. Using newly updated arrival and service curves, accurate delay bounds can be derived and whether the deterministic delay performance requirements specified in IEC 61850-5 [17] are satisfied can be verified analytically. For legacy SASs to be renovated with IEC 61850, service and arrival curves established in previous projects using similar devices and configuration may be exploited to analytically assess the worst-case delay performance of process bus networks to be deployed.

8.2 Applicability of the Proposed Approach

In our experiments, the maximum rate at which SVMs are injected into a particular input interface of an Ethernet switch is limited to about 5.3 Mbps. Although our evaluation shows that the analytical bounds are sufficiently tight at this rate, this does not suggest that our approach is always applicable. In theoretical treatments of network calculus [4, 8], it is usually assumed that network nodes have infinitely sized buffers. Under such an assumption, worst-case backlog bounds can be derived for individual nodes using network calculus, addressing the problem of buffer dimensioning. In our proposed approach, Theorem 5.1 applies as long as the assumption of infinite buffer size holds. Suppose that the aggregate input to a network node with a service curve $\beta(t) \leftrightarrow (R, T)$ has an arrival curve $\alpha(t) \sim (\sigma, \rho)$. In general, the theoretical stability condition $\rho \leq R$ suffices to guarantee that there exists some constant C that upper bounds the maximum backlog sizes of all nodes [4, 7].

However, what we observe on our switches is that certain frames will be discarded when the data rate at an output interface approaches its wire rate. To ensure that a particular SCN design does not violate the infinite buffer size assumption, simply checking the theoretical stability condition is insufficient. Measurements must be taken to establish the "safe operating regions" for different switch models in terms of maximum data rates and frame sizes at output interfaces: Using a particular frame size, we increase the data rate of a certain output interface, which can easily be achieved by utilizing multiple input interfaces. The data rate at which packet loss is first observed may be used to establish the boundary of the operating region, which can then be used in future SAS design to assess whether certain output interfaces are heavily loaded or not.

9 CONCLUSIONS

In this work, we show that Ethernet switches can be modeled by multiple work-conserving links provided that they have sufficient processing capacity to support dedicated interconnections for all their input/output interface pairs. To analyze a process bus network, we propose the combination of network calculus and measurements, facilitating the construction of realistic service curve models and producing accurate delay bounds that can be validated against measurements. Instead of applying the conventional technique of stopped sequences [1, 8] to deal with non-feedforward traffic patterns, we convert non-feedforward traffic patterns into feedforward ones, making it easier to apply network calculus theorems.

As our future work, we plan to further extend our proposed worst-case delay analysis. SASs solely relying on IEC 61850 for protection and control are emerging [11, 12]. The traffic flows on switched Ethernet may include PTP messages (for in-band synchronization) and GOOSE messages. Although the traffic patterns, in the most general cases, are still non-feedforward, the arrival curve models need to be further refined to accurately upper-bound PTP and GOOSE message flows. Incorporating analytical methods that are formally proved to produce tight bounds (e.g., [7, 23]) may help us achieve realistic bounds under such scenarios since more sophisticated arrival curves used in these methods are more flexible in capturing fluctuations in traffic profiles.

ACKNOWLEDGMENTS

This work is supported by ABB Inc. and the Pennsylvania Infrastructure Technology Alliance (PITA), a program sponsored by the Pennsylvania Department of Community and Economic Development. The authors would also like to thank ABB Inc. for the technical assistance that greatly facilitated this work. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of ABB Inc. or PITA.

REFERENCES

[1] S. Azodolmolky, R. Nejabati, M. Pazouki, P. Wieder, R. Yahyapour, and D. Simeonidou. 2013. An Analytical Model for Software Defined Networking: A Network Calculus-Based Approach. In *2013 IEEE Global Communications Conference (GLOBECOM)*. 1397–1402.

[2] Steffen Bondorf, Paul Nikolaus, and Jens B. Schmitt. 2016. Delay Bounds in Feed-Forward Networks - A Fast and Accurate Network Calculus Solution. *arXiv preprint arXiv:1603.02094v1 [cs.NI]* (March 2016).

[3] Steffen Bondorf and Jens B. Schmitt. 2014. The DiscoDNC v2: A Comprehensive Tool for Deterministic Network Calculus. In *Proceedings of the 8th International Conference on Performance Evaluation Methodologies and Tools (VALUETOOLS '14)*. 44–49.

[4] Jean-Yves Le Boudec and Patrick Thiran. 2001. *Network Calculus: A Theory of Deterministic Queuing Systems for the Internet*. Springer.

[5] A. Bouillard, L. Jouhet, and E. Thierry. 2010. Tight Performance Bounds in the Worst-Case Analysis of Feed-Forward Networks. In *Proceedings of the 2010 IEEE Conference on Computer Communications (INFOCOM 2010)*. 1–9.

[6] Anne Bouillard and Aurore Junier. 2011. Worst-Case Delay Bounds with Fixed Priorities Using Network Calculus. In *Proceedings of the 5th International ICST Conference on Performance Evaluation Methodologies and Tools (VALUETOOLS '11)*. 381–390.

[7] A. Bouillard and G. Stea. 2015. Exact Worst-Case Delay in FIFO-Multiplexing Feed-Forward Networks. *IEEE/ACM Transactions on Networking* 23, 5 (October 2015), 1387–1400.

[8] Cheng-Shang Chang. 2000. *Performance Guarantees in Communication Networks*. Springer-Verlag.

[9] X. Cheng, W. J. Lee, and X. Pan. 2017. Modernizing Substation Automation Systems: Adopting IEC Standard 61850 for Modeling and Communication. *IEEE Industry Applications Magazine* 23, 1 (January 2017), 42–49.

[10] H. Georg, N. Dorsch, M. Putzke, and C. Wietfeld. 2013. Performance Evaluation of Time-Critical Communication Networks for Smart Grids Based on IEC 61850. In *Proceedings of the 2013 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*. 43–48.

[11] Q. Huang, S. Jing, J. Li, D. Cai, J. Wu, and W. Zhen. 2017. Smart Substation: State of the Art and Future Development. *IEEE Transactions on Power Delivery* 32, 2 (April 2017), 1098–1105.

[12] D. M. E. Ingram, P. Schaub, D. A. Campbell, and R. R. Taylor. 2013. Performance Analysis of PTP Components for IEC 61850 Process Bus Applications. *IEEE Transactions on Instrumentation and Measurement* 62, 4 (April 2013), 710–719.

[13] D. M. E. Ingram, P. Schaub, R. R. Taylor, and D. A. Campbell. 2013. Performance Analysis of IEC 61850 Sampled Value Process Bus Networks. *IEEE Transactions on Industrial Informatics* 9, 3 (August 2013), 1445–1454.

[14] International Electrotechnical Commission (IEC) 2010. *Communication Networks and Systems for Power Utility Automation - Part 90-1: Use of IEC 61850 for the Communication between Substations*. IEC TR 61850-90-1:2010.

[15] International Electrotechnical Commission (IEC) 2011. *Communication Networks and Systems for Power Utility Automation - Part 8-1: Specific Communication Service Mapping (SCSM) - Mappings to MMS (ISO 9506-1 and ISO 9506-2) and to ISO/IEC 8802-3*. IEC 61850-8-1:2011.

[16] International Electrotechnical Commission (IEC) 2011. *Communication Networks and Systems for Power Utility Automation - Part 9-2: Specific Communication Service Mapping (SCSM) - Sampled Values over ISO/IEC 8802-3*. IEC 61850-9-2:2011.

[17] International Electrotechnical Commission (IEC) 2013. *Communication Networks and Systems for Power Utility Automation - Part 5: Communication Requirements for Functions and Device Models*. IEC 61850-5:2013.

[18] International Electrotechnical Commission (IEC) 2013. *Communication Networks and Systems for Power Utility Automation - Part 90-4: Network Engineering Guidelines*. IEC TR 61850-90-4:2013.

[19] International Electrotechnical Commission (IEC) 2015. *Communication Networks and Systems for Power Utility Automation - Part 90-12: Wide Area Network Engineering Guidelines*. IEC TR 61850-90-12:2015.

[20] International Electrotechnical Commission (IEC) 2016. *Communication Networks and Systems for Power Utility Automation - Part 90-2: Using IEC 61850 for Communication between Substations and Control Centres*. IEC TR 61850-90-2:2016.

[21] H. Lei, C. Singh, and A. Sprintson. 2014. Reliability Modeling and Analysis of IEC 61850 Based Substation Protection Systems. *IEEE Transactions on Smart Grid* 5, 5 (September 2014), 2194–2202.

[22] Luigi Rizzo. 2012. netmap: A Novel Framework for Fast Packet I/O. In *2012 USENIX Annual Technical Conference (USENIX ATC '12)*. 101–112.

[23] J. B. Schmitt, F. A. Zdarsky, and M. Fidler. 2008. Delay Bounds under Arbitrary Multiplexing: When Network Calculus Leaves You in the Lurch.... In *Proceedings of the 27th IEEE Conference on Computer Communications (INFOCOM 2008)*. 1669–1677.

[24] J. B. Schmitt, F. A. Zdarsky, and L. Thiele. 2007. A Comprehensive Worst-Case Calculus for Wireless Sensor Networks with In-Network Processing. In *Proceedings of the 28th IEEE International Real-Time Systems Symposium (RTSS 2007)*. 193–202.

[25] T. S. Sidhu and Y. Yin. 2007. Modelling and Simulation for Performance Evaluation of IEC61850-Based Substation Communication Systems. *IEEE Transactions on Power Delivery* 22, 3 (July 2007), 1482–1489.

[26] Oregano Systems. 2017. syn1588® PCIe NIC - Technical Information. (2017). Retrieved January 2017 from http://www.oreganosystems.at/?page_id=2214.

[27] M. R. D. Zadeh, T. S. Sidhu, and A. Klimek. 2011. Implementation and Testing of Directional Comparison Bus Protection Based on IEC 61850 Process Bus. *IEEE Transactions on Power Delivery* 26, 3 (July 2011), 1530–1537.