INVITED: A Testbed to Verify the Timing Behavior of Cyber-Physical Systems

Aviral Shrivastava[1], Mohammadreza Mehrabian[1], Mohammad Khayatian[1]
Patricia Derler[2], Hugo Andrade[2], Kevin Stanton[3],
Ya-Shian Li-Baboud[4], Edward Griffor[4], Marc Weiss[5], and John Eidson[6]

[1] Arizona State University, [2] National Instruments, [3] Intel, [4] NIST, [5] Qulsar, [6] UC Berkeley

ABSTRACT

Time is a foundational aspect of Cyber-Physical Systems (CPS). Correct time and timing of system events are critical to optimized responsiveness to the environment, in terms of timeliness, accuracy, and precision in the knowledge, measurement, prediction, and control of CPS behavior. However, both the specification and verification of timing requirements of the CPS are typically done in an ad-hoc manner. While feasible, the system can become costly and difficult to analyze and maintain, and the process of implementing and verifying correct timing behavior can be error-prone. Towards the development of a verification testbed for testing timing behavior in tools and platforms with explicit time support, this paper first describes a way to express the various kinds of timing constraints in distributed CPS. Then, we outline the design and initial implementation of a distributed testbed to verify the timing of a distributed CPS analytically through a systematic framework. Finally, we illustrate the use of the verified timing testbed on two distributed CPS case studies.

1. INTRODUCTION

Intelligent software-based control systems are enabling an increasingly connected and automated world with smart devices, smart healthcare, smart transportation, smart buildings, smart grid, smart defense, etc. Software adds considerable flexibility to hardware devices since arbitrarily complex control algorithms can be implemented in software. The integration of computational algorithms with distributed physical platforms is referred to as Cyber-Physical Systems (CPS). Many CPS are safety-critical and hard real-time. Hard real-time means that correct system behavior depends not only on the logical "functionality" of the computation, but also on the "timing" of the computation, sensing, and actuation, with deadlines that may have no or little tolerance for uncertainty. As an example, electric power system is a distributed CPS that has critical timing requirements since a power outage will affect sensitive infrastructures like transportation, industry, communication, water supply, etc. Some of the required timing constraints in power system include traveling wave fault detection and location that needs 100 to 500 nanoseconds time accuracy, line differential relays that need 10 to 20 microseconds time accuracy and anti-islanding that needs sub-microsecond time accuracy [1].

Traditionally, timing requirements for CPS are either an afterthought once the main functions are met – for soft real-time or hard real-time systems–, they are designed with costly customized hardware platforms (such as Application-Specific Integrated Circuits (ASICs) or Field Programmable Gate Arrays (FPGAs)) and operating systems (including but not limited to real-time operating systems). This is because modern computing systems are built for average performance or power consumption at the cost of timing predictability. This happens at multiple levels of abstractions in the computer design. At the hardware-level, microarchitectural structures like processor caches lead to variability in execution times, but they are indispensable in improving the average performance of the application. In operating systems with unbounded preemptions, shared resources, the Interrupt Service Routines significantly degrade the ability to estimate the execution time of tasks executing in an arbitrary environment, but they are the most flexible and scalable means to implement I/O in modern processors. Consequently, and most importantly, programming languages have no notion of time. Common programming languages like C/C++/Java/Python etc. do not allow the programmer to specify any timing behavior of the application[2]. For example, in C, it is not possible to specify that a certain instruction should execute at a certain time, or that a loop body must be executed exactly every 10 ms. All this must be implemented through operating system calls, and therefore, the timing of the system depends on the hardware-software platform, and the behavior of other executing tasks on the computing system. This makes it difficult to guarantee any timing property with platform customization.

Irrespective of the implementation of the CPS, there is always a need to verify the timing behavior. Particularly for safety-critical CPS, testing whether the implementation meets all of its timing requirements is important. The heterogeneity of CPS components and the diverse design environments, particularly make the testing of a CPS a difficult challenge. Since a universal standard methodology has not been introduced yet, time testing of CPS is typically done in an ad-hoc manner. Even though the methods and equipment used for analyzing/testing the timing of CPS may be quite sophisticated (typically oscilloscopes and signal/frequency analyzers), the testing methodology is often custom and lacks standardization, which can make tests difficult to reproduce or to have confidence in the verification process. Difficulties arise when verifying the temporal behavior for a signal with high-frequency components, or when capturing the exact occurrence time of an event (a signal value crossing a threshold at a single point in time) on a dynamically changing signal [3]. For digital signals, rise time can affect the event time. Rise time indicates how fast the signal value changes from 0 to 1, and noise in the power supply or even signal cross-talk can affect the rise time. For physical signals, such as electrical signals, the length of the measurement cable can affect the impedance and therefore the timing of events on the signal. Discretization rate and precision also affect the timing of an event. Additional challenges arise in geographically distributed CPS. How can we ascertain the timing of synchronous measurements on geographically distributed components? How do we combine and make sense of data measured with different monitoring equipment, each with its

own clock, precision, and latency properties? Although possible, ad-hoc and customized testing approaches quickly become very complicated, error-prone to formally specify, analyze and verify.

This paper is an effort towards standardizing the process of testing the timing properties of CPS. The ultimate objective of the testbed is that if the timing specifications can be expressed formally as a small number of primitives, then it would be possible to generate canonical timing tests for each primitive. Such canonical tests would guide modular system design. Larger, more complex CPS can organize and assemble the canonical test components. The first step (in section 2) is to express/specify the timing requirements of a CPS. To that end, we formulate and organize the different timing constraints that apply to distributed CPS. The timing constraints are applied on events coming on signals that can be specified as singleton or repeating events. Singleton one refers to an event that is appeared in testing duration just once in contrast with repetitive events that there is not any limitation for the count of event occurrence. Among singleton events, there may be latency, simultaneity, and chronological requirements, while for repeating events, there may be frequency, phase, sporadic, and burst constraints. Next (in section 3), we outline a design of a testbed that can be used to test the CPS to check if all the timing constraints are being met or not in a systematic and correct manner to enable correct-by-construction (CbC) synthesis of the testbed. The testbed – like the distributed CPS it is trying to test – is also a distributed CPS, with each node (of the testbed) monitoring the required signals from the CPS node. Hardware timestamping and PTP synchronization of the clocks among the CPS components provides observations at the same timescale through vast geographies, without losing accuracy with time. We also discuss the key timing parameters of the testbed that will affect the time testing capability. The next section (section 4) discusses the specifications that must be met by the testbed, such that the testbed can validate the timing constraints. Finally, in section 5 we apply the CbC timing testbed to verify the timing constraints of two example distributed CPS.

2. TIMING REQUIREMENTS OF (DISTRIBUTED) CPS

In distributed systems, timing requirements often require all network nodes to have a common concept of time in order to establish the basis for comparison. All timing constraints are defined in terms of time interval between events. An event is defined by a tuple $< s, v, d >$, which refers to the occurrence time when the signal s crosses the voltage threshold v in the direction $d = \{rising,\ falling\}$. Time can be expressed as absolute (Coordinated Universal Time (UTC), Temps Atomique International (TAI)) or relative (s, ms), and depending on system sensitivity and inherent safety, business, or legal concerns, traceability would require a degree of accuracy from absolute time (e.g. milliseconds accuracy for safety issues in car). In order to define and analyze timing requirements, a formal means of expressing the constraints is needed. The following paragraphs describe the types of constraints we have considered in this work.

Latency Constraint (LC) describes the requirements on the time interval between two events. Latency constraints come in three types: minimum latency constraint (LCm), maximum latency constraint (LCM) and exact latency constraint (LCE) with an uncertainty tolerance of ϵ. A minimum latency constraint l between two events e_1 and e_2 can be expressed as $LCm(e_1, e_2, l, \epsilon)$. $LCm(e_1, e_2, l, \epsilon)$ states that, given that the occurrence time of event e_1 is t_1, and the occurrence time of event e_2 is t_2, the minimum latency constraint is met if $t_2 - t_1 \geq l - \epsilon$. Similarly the maximum latency constraint $LCM(e_1, e_2, l, \epsilon)$ is met, if $t_2 - t_1 \leq l + \epsilon$, and the exact latency constraint $LCE(e_1, e_2, l, \epsilon)$ is met, if $t_2 - t_1 = l \pm \epsilon$. Latency constraints are the most basic temporal requirements used in CPS applications. Note that the latency constraint may also be specified for two events in different nodes of a distributed CPS. For instance, if the maximum latency constraint is 5ms and and e_1 and e_2 happened at $t_1 = 2.1s$ and $t_2 = 7.3s$ respectively, then in the case of $\epsilon = 0.2ms$ the constraint is met since $7.3 - 2.1 \leq 5 + 0.2$ in contrast with $\epsilon = 0.1ms$, the constraint is not met.

Simultaneity Constraint specifies the requirement that two events happen at the same instant of time within a window of acceptable tolerance. A simultaneity constraint between two events e_1 and e_2 is expressed as $SC(e_1, e_2, \epsilon)$. Given the occurrence time of event e_1 is t_1, and the occurrence time of event e_2 is t_2, the simultaneity constraint is met if $|t_2 - t_1| < \epsilon$. The simultaneity constraint differs from the exact latency constraint (with the parameter $l = 0$) since the order of events e_1 and e_2 does not matter for simultaneity. Similar to latency constraint, events e_1 and e_2 can be specified for events that occur in individual nodes of a distributed CPS. Similarly, simultaneity constraint can be specified for more than two events. In that case, all events must happen within a tolerance window of time ϵ so as to be interpreted as simultaneous. For instance, the simultaneous image capturing application discussed in section 3 has a simultaneity constraint on capture time of multiple distributed cameras with $\epsilon = 100\mu s$.

Chronological Constraint describes the order or precedence of events. A chronological constraint between two events e_1 and e_2 can be expressed as $CC(e_1, e_2, \epsilon)$. Given the occurrence time of event e_1 is t_1 and, the occurrence time of event e_2 is t_2, the chronological constraint is met if $t_2 > t_1 + \epsilon$.

Frequency Constraint expresses a requirement on the occurrence frequency of an event. A frequency constraint for an event $e1$ can be expressed as $FC(e_1, f, \epsilon)$. Given repeating events e_k occurs at t_k for $k \in N$, frequency constraint is met if $\forall k, (t_{k+1} - t_k) = \frac{1}{f} \pm \epsilon$. Similar to the latency requirement, the frequency constraint has three types, minimum frequency constraint, maximum frequency constraint and exact frequency constraint that requires the time interval between every two consecutive events e_k and e_{k+1} to be less than, greater than, or equal to $f \pm \epsilon$. For example, if the desired frequency is $60Hz$ and the tolerance is $0.01Hz$, then for all frequencies between 59.99 and 61.01Hz, the requirement is met.

Phase Constraint describes the lag between two periodic events with the same frequency. A phase constraint between two periodic events e_1 and e_2 is expressed as $PC(e_1, e_2, p, \epsilon)$. Given that a periodic event e_1 occurs at t_k instants for $k \in N$ and another periodic event e_2 with the same period occurs at $t_k^{'}$ instants for $k \in N$, the maximum phase constraint $PCM(e_1, e_2, p, \epsilon)$ is met if the maximum phase difference $|t_k - t_k^{'}| \leq p \pm \epsilon$. The minimum phase constraint $PCm(e_1, e_2, p, \epsilon)$ is met if the minimum phase difference $|t_k - t_k^{'}| \geq p \pm \epsilon$. Finally the exact phase difference is met if $PCE(e_1, e_2, p, \epsilon)$ is met if the phase difference $|t_k - t_k^{'}| = p \pm \epsilon$.

Sporadic constraint expresses temporal requirements on those events that are repeating but are not necessarily periodic, so the length of the time interval between these events varies. Let Δt_{min} and Δt_{max} denote the min and max of those lengths. Recall that the time interval between events are bounded by the minimum and maximum latencies. A minimum sporadic constraint on e_1 is expressed as $SCm(e_1, \Delta t_{min}, \epsilon)$ and is met if the the latency constraint $LCm(e_1, e_2, \Delta t_{min}, \epsilon)$ is met for all e_2. Similarly, a maximum sporadic constraint $SCM(e_1, \Delta t_{max}, \epsilon)$ is met on an event e_1 if the latency constraint $LCM(e_1, e_2, \Delta t_{max}, \epsilon)$ is met for all e_2.

Burst constraint describes the occurrence of an event in a specified time interval. A burst constraint for an event e_1 is expressed as $BC(e_1, n, d, m, \epsilon)$. If an event e_1 occurs at t_k instants for $k \in N$.

The burst constraint is met if n occurrences of event e_1 happen in a time interval with duration d and there should be a recovery time m without any occurrence of e_1. For example, $BC(e_1, 3, 10, 5ms)$ is interpreted as a burst constraint on event $e1$ that should occur only 3 times in a $10s$ interval and should not happen for $5ms$ after the third occurrence.

3. DISTRIBUTED TESTBED TO EVALUATE TIMING BEHAVIOR OF (DISTRIBUTED) CPS

A systematic approach to test and verify the timing behavior of a distributed system is to monitor signals and events of the System Under Test (SUT) and timestamp events with a common testbed timebase within the specified precision and accuracy such that the required timing constraints over a distributed system can be properly verified. Figure 1 shows the structure of our distributed testbed. Just like the CPS under test, the testbed itself is also a distributed CPS. Each node of the CPS is monitored by a DAQ (Data Acquisition) platform – a programmable test and measurement device with analog and digital inputs and outputs. Each testbed node has two major components: i) Data Acquisition (DAQ) platform and ii) Operating System (OS). The DAQ platform of each node monitors the signals and timestamps the events of interest using a time-synchronized internal clock. The signals that should be monitored must be made observable by design – this is an aspect of Design For Testability (DFT). Each test and measurement node is synchronized to a local clock reference – a Global Navigation Satellite System (GNSS) traceable reference time source for example – and distributed using the Precision Time Protocol (PTP). The event timestamps are sent to the OS, which in turn, sends them all the workstations data, that loads them into a database. The database of time-stamped events enables automated analysis to determine if the timing constraints of the distributed CPS are met or not. Whether a testbed can validate a timing constraints or not, depends on the design parameters of the testbed. The most important design parameters of the distributed testbed that affect the errors in the timing measurements are described below.

Analog to Digital Converter (ADC) parameters The testbed monitors all signals by sampling them because they should be digitalized to use in cyber side. This is done by Analog to Digital Converters (ADCs) on the probes. The sampling rate of an ADC, f_s, is expressed as samples per second, or Hertz (Hz). In order to be able to monitor a signal correctly, the sampling rate must be sufficiently high to capture the fastest observable dynamics of interest in the signal. Suppose we intend to find out the time at which a signal rises above 3.4V. Figure 2 shows this signal, monitored with two different sampling rates. On the left with sampling rate of $f_s = 1kHz$, the threshold crossing time of the signal is detected as $t = 1ms$. However, on the right, with sampling rate of $f_s = 0.5kHz$, the threshold crossing time of the signal is detected as $t = 2ms$.

Since an ADC converts the voltage signal into digitized sampled events, the accuracy of measurement is also limited by the number of bits used to express the sampled value, $nbits_{ADC}$, and the voltage range of the ADC, VR_{ADC}. An n-bit ADC can represent 2^n values. A 12-bit ADC that measures the range of 0V to 5V has steps of $\approx 1mV$. The precision of the ADC is defined in terms of resolution of the ADC, or V_{ADC} can be calculated as: $V_{ADC} = \frac{VR_{ADC}}{2^{nbits_{ADC}}}$. The resolution of the ADC can affect the time at which the monitoring device detects an event on a signal. Figure 3 illustrates the conversion of an analog signal to digital samples with various resolutions, VR_{ADC} of 5V. If the user wants to detect the time when a signal rises above 4V, then in the left diagram, with $nbits_{ADC} = 12$, the time at which the threshold crossing is detected is $t = 3ms$, while in the right diagram, with $nbits_{ADC} = 11$, the time at which the threshold crossing is detected is $t = 4ms$.

Wiring a signal to a DAQ device adds a load to the CPS circuit under test, which causes a change in the shape of the monitored signal. For pure resistive loads, this change is a simple voltage drop while for general loads, the shape of the monitored signal is changed based on the equivalent resistance and reactance of the measuring device (including capacitance effect of the cables) and the SUT. As a result, based on the rate of change in the value of the signal, the measurements of the signal may be delayed or its amplitude may be attenuated.

Input impedance, Z_{in}, is defined as the CPS equivalent circuit from the terminal connected to the testbed. The test and measurement device must have a sufficiently high input impedance to minimize perturbation of the measurement process on the signal.

Figure 4.b shows the monitored signal perturbed by the loading effect of wiring the measurement device to the SUT. The threshold detection time of the original signal is before the threshold detection time of the monitored signal.

Clock Fractional Frequency Offset A clock's fractional frequency offset is defined as $f_{clock} = \frac{f_{inst} - f_0}{f_0}$, where f_{inst} is the instantaneous clock frequency, and f_0 is the nominal clock frequency. Thus, this is the unitless instantaneous fractional offset from the nominal frequency of an oscillator [4]. Environmental conditions such as voltage and temperature variations or mechanical vibrations, can affect the rate at which an oscillator runs. Typically, the fractional frequency offset of a clock, f_{clock}, is expressed in Parts Per Million (PPM), indicating the maximum amount of error in one million time units. Thus, the time error after an elapsed time $t_{elapsed}$ due to a fractional frequency offset of f_{clock} is $t_{elapsed} \times f_{clock}$. For instance, a clock with 5 PPM error, has $5\mu s$ error after 1 second, an error of about 0.5 s after a day, or about 2.5 minutes after a year.

Since all clocks deviate from each other, distributed clocks must be synchronized to a reference to have an agreement on time and have a unique and time notion. Synchronization protocols match the clock of a device to a reference clock. However, no synchronization protocol is perfect, and there is a synchronization error t_{sync}, that depends on several factors, including the number of bits used to represent the time, when the time stamping is done (e.g., in the hardware or in software), network jitter, network asymmetries delays, etc. [5]. The Network Time Protocol or NTP [6] can usually keep time synchronized to within tens of milliseconds over the public Internet ($t_{sync} \approx 10ms$). The Precision Time Protocol, PTP [7], can provide time synchronization over a LAN with sub-microsecond accuracy. PTP with the White Rabbit[8] extension used for the CERN Large Hadron Collider, can synchronize to sub-nanosecond accuracy. For CPS distributed over a wide area with high precision and accuracy needs, GNSS (Global Navigation Satellite Systems) can provide $100ns$ accuracy.

Another important parameter is the rate of synchronization, r_{sync}, which is the number of times per second (e.g.in units of Hz) that synchronization is performed. Every time we perform synchronization, the time offsets are within t_{sync} of each other. But from thereon, until the next synchronization, the clock times will move apart at the rate of f_{clock}, if the local clock uses the protocol to adjust its time but not its frequency. The worst-case clock offset, ϵ_{wcco}, while the system clock is synchronized via the time synchronization protocol in steady-state and while all other environmental conditions are stable, can be calculated as: $\epsilon_{wcco} = t_{sync} + \frac{f_{clock}}{r_{sync}}$. Note that the units are in time, since f_{clock} is unitless and the reciprocal of r_{sync} is in units of time.

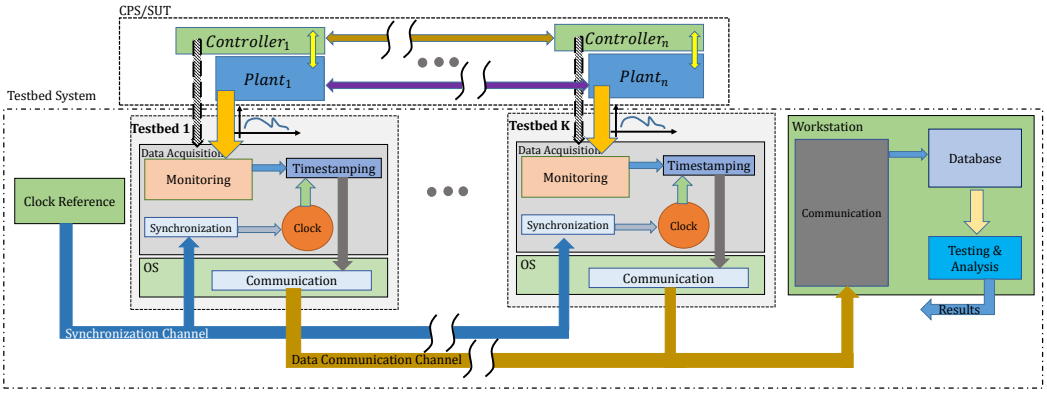

Figure 1: Time testing structure diagram. There are k distributed testbed nodes that communicate using a network link. All nodes are synchronized with a clock reference to have the same notion of time. There exists a database logging the time-stamped events from the distributed test and measurement system for further analysis.

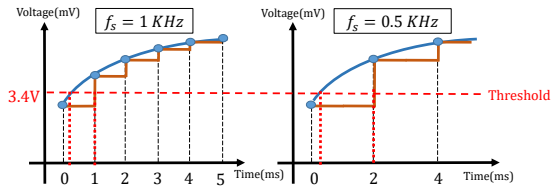

Figure 2: A digitized analog signal at two different sampling rates. Given a threshold of 3.4V, the threshold crossing time is detected at different times depending on the sampling rate.

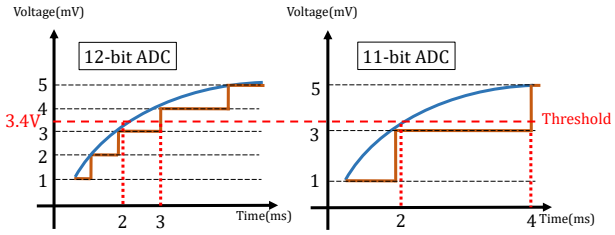

Figure 3: An analog signal sampled using two ADCs that have the same range (0V to 5V) and different resolutions. a) a 12-bit ADC is used. b) a 11-bit ADC is used. Threshold crossing time in the left figure is t_1, which is different in the right figure for the same signal.

4. TESTBED CAPABILITY ANALYSIS

In order to determine whether timing behavior is verifiable by a given testbed, it is important to understand the sources of timing measurement uncertainty, described as ϵ in the timing constraint specification.

Consider a distributed CPS, with an exact latency constraint $LCE(e_1, e_2, l, \epsilon)$ for events e_1 and e_2, where e_1 occurs on signal s_1 and e_2 on s_2 respectively. These events are detected at different nodes of the CPS. The latency constraint states that, given t_1 as the occurrence of e_1, the time at which e_2 occurs should be equal to $t_1 + l \pm \epsilon$. The testbed must capture the time at which an event occurs. However, the measured time will be erroneous. This can due to the several factors, including the sampling frequency f_s, the ADC resolution V_{ADC}, and the clock error, ϵ_{wcco}.

Consider an event described by the tuple $\langle s_1, v_t, rising \rangle$, marking

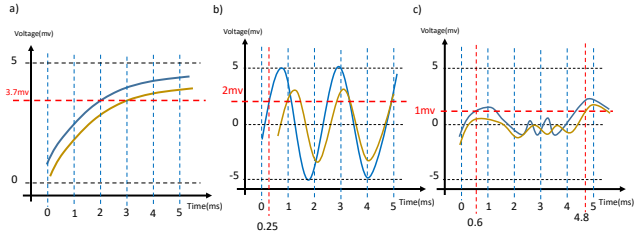

Figure 4: a) Voltage drop on a DC signal connected to a resistive load. b) Voltage drop and shift on an AC signal connected to a load that has both reactive and resistive components. c) Change in the shape of an arbitrary signal due to the loading effect.

the threshold v_t crossing of signal s_1 on a rising edge. Since the ADC output is a multiple of the supported resolution, the testbed may not be able to detect the exact point of the threshold crossing. Thus, the threshold value must be mapped to the nearest upper bound of the value. Since all sampled data are collected at known points in time (integer multiples of $\frac{1}{f_s}$), a threshold crossing is detected with a maximum error $\epsilon_{ADC} = \frac{1}{f_s}$. Figure 5 illustrates the worst-case error $\frac{1}{f_s}$ in an example.

Since all samples are timestamped using the local clock of the measurement system, clock synchronization error (ϵ_{wcco}) must be taken into account. Thus, the maximum time error between the actual event occurrence and the detected event occurrence is the sum of the ADC error and the clock synchronization error: $\epsilon_{total} \leq \epsilon_{wcco} + \epsilon_{ADC}$.

Since there will be at most ϵ_{total} error in both the measurements of e_1 and e_2, then the testbed can confidently verify whether the exact latency constraint is being met or not. Other types of constraints (e.g. simultaneity, frequency, phase, etc.) are also expressed with a temporal error tolerance and one can similarly reason and verify the temporal behavior.

5. CASE STUDIES

This section presents two case studies to showcase how a time testing framework is used to verify the timing behavior of distributed systems.

5.1 Simultaneous Image Capturing

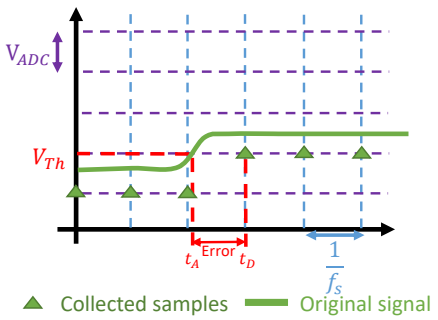

Figure 5: Worst-case error between actual occurrence time and detection time for an ADC with sampling frequency f_s and fixed threshold detection based on an integer multiple of the ADC resolution

3D image reconstruction based on multiple 2D images taken from different angles of a scene has application in many fields, including entertainment, military (geometric information extraction), autonomous driving, and sports (e.g., football match analysis). In 3D view reconstruction, the image processing algorithm detects and matches points in all pairs of images and then calculate the spatial position of the point based on cameras position and triangulation [9]. By computing the geometric position of all points, a 3D view of the scene is reconstructed. When capturing and processing a moving object in real time, exposure times, frame rates, and computation latencies must be precise and within a small error in order to achieve a successful reconstruction[10]. Since the motion of the object causes blur in the image, a short exposure time is desired. Depending on the object's color, size, shape, speed, distance from the camera, camera properties like resolution, exposure time, and the type of the algorithm used for detection, requires a degree of simultaneity between capture events on the respective cameras. Since there is no specific formula that accounts for the all parameters of camera, object and algorithm, we arbitrarily select a time to trigger the cameras. We took $100\mu s$ as the maximum delay between capture time of two cameras. We interpreted this timing requirement as a simultaneity constraint between trigger signals of cameras with maximum acceptable tolerance of $\epsilon = 100\mu s$ or $SC(cameraTrigger_1, cameraTrigger_2, 100\mu s)$.

We implemented the 3D view reconstruction application by taking a picture of a moving soccer ball using two cameras. We used the ArduCAM ESP8266 UNO boards which include a 2MP CMOS camera for image capturing, a built-in ESP8266 Module for wireless communication and some digital I/O. We utilized a rechargeable 3.7V 700mAh Li-Po battery as the power supply. A web-server is used to send the capture command to both cameras. Upon capturing, each ArduCAM board generates a trigger signal on one of the digital input/output (I/O) pins. This signal is used to show the delay between capture time instances between two cameras. Figure 6 shows the ArduCAM boards (towards the bottom) taking pictures of a rolling soccer ball and the testing platforms.

5.2 Power grid generator synchronization

An important and large-scale CPS is power distribution system. The future of the power grid will rely on distributed power generation using renewable energy resources [11]. So, generated power by the distributed resources should match in order to avoid short circuit. In such a system, a pair of generators connected to the same grid should generate a sinusoidal signal with frequency, $f = 60Hz$, with 0.1% tolerance[12]. Besides, the phase between two generator shouldn't

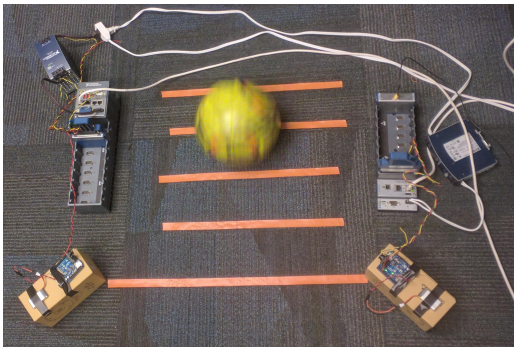

Figure 6: Two ArduCAM taking simultaneous images of a moving soccer ball. The shutter signal of the two cameras are monitored by two cRIO *NI-9067* that are synchronized by PTP over Ethernet.

be greater than 10 degree. So, timing constraint for this case study are a frequency constraint on both sinusoidal signals and a phase constraint between them. One can write the frequency constraint as $FC(e_{Master}, 60, 0.06)$ where acceptable tolerance for period, T, is $\frac{1}{60+0.06} < T < \frac{1}{60-0.06}$ ($16.65ms < T < 16.68ms$) so the tolerance error window is about $33\mu s$. For the second requirement, the phase between the signals should be at least $\pm 10°$. So, the phase constraint can be described by $PCE(e_{Master}, e_{Slave}, 0, 463\mu s)$. Acceptable tolerance for this requirement is $\epsilon = 463\mu s$ because a complete revolution (360°) is done in 16.67ms ($\frac{1}{60Hz} = 16.67ms$).

We implemented an experimental setup with two DC motors to represent two small generators. We installed two dials marked from 0° to 360° on the shaft of each motor. A hole was drilled in each dial at 0° and an optical detector is installed on the dial to detect when the angle is 0° (Figure 7). Two Arduino Mega2560 boards are utilized to control the speed of the motor using Pulse Width Modulation. Arduino boards are synchronized with each other using two wireless modules (NRF24L01+, 2.4GHz). First motor sends its data to the second motor and the second motor controls its speed as to match its phase with the first motor.

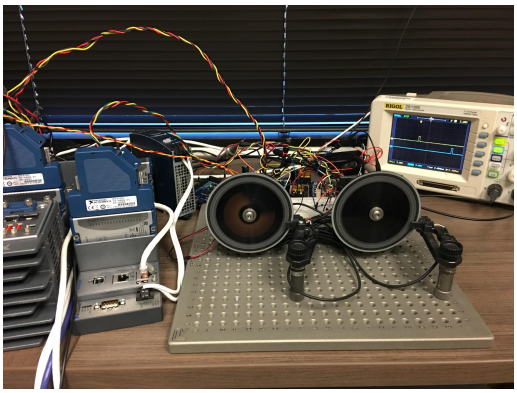

Figure 7: Two motors are controlled by two Arduino Mega 2560 boards that are synchronized, and the phase constraint is tested by two distributed NI-cRIO (9067 and 9035). The testing accuracy is checked by an oscilloscope.

6. TIME TESTING USING THE TESTBED

In order to test and verify the timing constraints of the experimental setups, testbed timing specifications (synchronization accuracy,

ADC sampling rate, ADC resolution, etc.) must exceed the CPS specifications as mentioned in section 4. We used two NI-cRIO (NI-9067 and NI-9035) platforms for testbed measurement and control which include on board FPGA with $40MHz$ clock frequency and $5ppm$ clock drift. FPGA clocks are synchronized using NI-TimeSync[13] that supports IEEE1588. Sampling rate of the data acquisition module is $20kHz$ and it utilizes a 12-bit ADC. Measurement devices are connected via the dedicated Ethernet network. Implementation of the IEEE 802.1AS profile of IEEE 1588/PTP (part of IEEE 802.1 Time Sensitive Networking (TSN) standards) uses hardware timestamping and compensation both in network elements and endpoints to minimize time synchronization errors. TSN generally provides both synchronization and also small and deterministic packet latency between testbed devices.

The clock drift of the measurement nodes is $5ppm$, where each node synchronizes every second via PTP with a precision of $100ns$ to the grandmaster. The worst-case clock time offset of each cRIO is $\frac{5\mu s}{1s} + 100ns = 5.1\mu s$. Since the voltage range of digital module is from $0V$ to $5V$ and it uses a 12-bit ADC, the ADC resolution V_{ADC} can be calculated as $\frac{5-0}{2^{12}} \approx 1mV$. For image capturing case study, the sum of ϵ_{wcco} and ϵ_{ADC} is less than the required accuracy of $100\mu s$. Output impedance of the ArduCAM boards is 470Ω and input impedance of the cRIO is $1M\Omega$. Therefore, the loading effect is very small and the testbed is suitable to verify the timing constraints. In the power grid case study, required accuracy is $33\mu s$ for frequency constraint and $463\mu s$ for phase constraint. The testbed has sufficient precision to verify the phase constraint since the sum of ϵ_{wcco} and ϵ_{ADC} is less than the required accuracy. Each of the two optical sensors (Omron EE-SX970-C1) has at maximum $25m\Omega$ output impedance and each of the two cRIOs has $1M\Omega$ input impedance parallel with a $50pF$ capacitor.

7. CONCLUSION

While correct timing behavior plays an important, sometimes critical, role in many Cyber-Physical Systems, both the expression of the timing constraints and their testing are often done in an ad-hoc manner for soft real-time systems and a complex and costly process for hard real-time systems. In an effort to explore an expressive, consistent formalism for the timing specification and time verification procedure, we outlined common temporal constraints that appear in CPS, and the design of our distributed testbed to verify the timing of a given CPS. Each node of our testbed monitors the required signals and captures the events on the signals using hardware timestamping. The nodes of the testbed are synchronized through a reference clock over PTP to within microsecond accuracy. We analyze the relationship between the specifications of the testbed to the timing requirements of the CPS to figure out if the testbed can confidently validate the timing constraints by measurements. The effectiveness of the testbed is tested on two distributed CPS, a synchronized camera system, and a generator synchronization application.

In future efforts, we envision the use and abstraction of the temporal constraint specification to enable formal analysis and automation of system verification. Future work will include a more comprehensive analysis of the timing constraints as well as development of improved timing measurement capabilities based on clock synchronization to a traceable reference and the availability of TSN devices.

Acknowledgments

The authors would like to thank the NIST colleagues, Dr. Spencer Breiner and Martin Burns for the technical reviews. This work was partially funded by the NSF grant CNS 1525855, and NIST grant 70NANB16H305.

Disclaimer: Certain commercial entities, equipment, or materials are identified in this document in order to describe the experimental design or to illustrate concepts. Such identification is not intended to imply recommendation or endorsement by the National Institute of Standards and Technology or the institutions of the other authors, nor is it intended to imply that the entities, materials, or equipment are necessarily the best available for the purpose.

References

[1] "IEEE/NIST Timing challenges in the Smart Grid Workshop 2017." http://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.1500-08.pdf. [Online; Jan 2017 NIST].

[2] E. A. Lee, "Cyber Physical Systems: Design Challenges," in *ISORC*, 2008.

[3] A. Shrivastava *et al.*, "Time in Cyber-Physical Systems," in *Proc. of CODES+ISSS*, 2016.

[4] "NIST Time and Frequency from A to Z Glossary." https://www.nist.gov/time-and-frequency-services/d. [Online; Accessed: 2017-03-21].

[5] J. C. Eidson and K. B. Stanton, "Timing in Cyber-Physical Systems: The Last Inch Problem," in *Precision Clock Synchronization for Measurement, Control, and Communication (ISPCS), 2015 IEEE International Symposium on*, pp. 19–24, IEEE, 2015.

[6] D. L. Mills, "RFC 1305: Network Time Protocol (Version 3) Specification," *Implementation and Analysis*, 1992.

[7] "IEEE Standard for a Precision Clock Synchronization Protocol for Networked Measurement and Control Systems," *IEEE Std 1588-2008 (Revision of IEEE Std 1588-2002)*, pp. 1–269, July 2008.

[8] M. Lipiński, T. Włostowski, J. Serrano, and P. Alvarez, "White Rabbit: A PTP Application for Robust Sub-nanosecond Synchronization," in *ISPCS*, pp. 25–30, IEEE, 2011.

[9] S. Paris, *Extraction of Three-dimensional Information from Images–Application to Computer Graphics*. PhD thesis, Université Joseph-Fourier-Grenoble I, 2004.

[10] M. K. Fard, M. Yazdi, and M. MasnadiShirazi, "A Block Matching Based Method for Moving Object Detection in Active Camera," in *IKT*, IEEE, 2013.

[11] N. Mohan and T. M. Undeland, *Power Electronics: Converters, Applications, and Design*. John Wiley & Sons, 2007.

[12] North American Electric Reliability Council, "Frequency Excursions." http://www.nerc.com/pa/RAPA/PA/Pages/FrequencyExcursions.aspx. [Online; accessed 17-March-2017].

[13] "NI Time Sync, Version 1.1." http://www.ni.com/pdf/manuals/373185a.pdf. [Online; 2010 National Instruments Corporation].