Temporal Issues in Cyber-Physical Systems

David Broman*†, Patricia Derler*, John C. Eidson*
*University of California, Berkeley, CA, USA
†Linköping University, Sweden
{broman, pd, eidson}@eecs.berkeley.edu

***Abstract*—This paper reviews the use of time, clocks, and clock synchronization protocols in cyber-physical systems (CPS). Recent advances in the area of timing suggest avenues of research and potential new application areas. We discuss how introducing timestamps and clocks can help overcome issues such as latency, jitter, and determining correct execution order. Furthermore, we show how system complexity can be reduced and distribution as well as parallelism can be done deterministically. We also point to recent work in raising time to first class citizen status in modeling and implementation. In particular, we describe design and execution environments of CPS and specialized hardware such as predictable timing architectures where time plays a key role.**

***Index Terms*—Timestamps, Clocks, Synchronization, Ordering, Simultaneity, Discrete event systems, Distributed control, Real-time systems, and PTIDES**

I. Introduction

This paper discusses temporal issues in the design and implementation of cyber-physical systems (CPS). Increasingly synchronized clocks and timestamps are being used to improve application performance in distributed CPS. We review the state of the art in the underlying clock and synchronization techniques and illustrate how this technology helps overcome issues such as latency, jitter, and determining correct execution order. Recent research in the areas of design and execution environments and specialized hardware such as predictable timing architectures raises time to first class citizen status in modeling and implementation. Finally we suggest avenues of future research and potential new application areas.

Section II of this paper briefly reviews the traditional use of time and clocks in CPS. Section III reviews the current state of the art in distributed timing systems and notes recent improvements and support available to designers of CPS. Section IV outlines existing and recent techniques for using timestamps and clocks to alleviate problems in ordering events and overcoming some issues related to latency, jitter and system complexity. Section V discusses recent results in raising timing issues to first class citizen status in design and execution environments used to create CPS. Section VI reviews analysis techniques used in designing CPS and recent efforts to create time-centric hardware support including the use of predictable timing architectures. Finally we summarize and suggest new avenues of research suggested by the discussion.

This work was supported in part by the iCyPhy Research Center (Industrial Cyber-Physical Systems, supported by IBM and United Technologies), and the Center for Hybrid and Embedded Software Systems (CHESS) at UC Berkeley (supported by the National Science Foundation, NSF awards #0720882 (CSR-EHS: PRET) and #0931843 (ActionWebs), the Naval Research Laboratory (NRL #N0013-12-1-G015), and the following companies: Bosch, National Instruments, and Toyota). The first author was funded by the Swedish Research Council #623-2011-955.

II. Traditional uses of time and clocks in CPS

A cyber-physical system consists of a digital computation portion, the cyber portion, and whatever the cyber portion interacts with, i.e., the physical portion. Here, physical is understood to cover not only artifacts modeled by the laws of physics but also artifacts modeled by the laws of chemistry, biology and increasingly man-made laws, e.g., telecommunication protocols.

For our purposes it is convenient to divide CPS into two categories, as the consideration of time and clocks is somewhat different in each. The first category, *data acquisition*, consists of systems where the primary focus is on observing the physical world. The second category, *control*, consists of systems incorporating closed loop control with reasonably stringent loop requirements. Of course, there are many systems that do not fit cleanly into one of these categories, e.g., supervisory control and data acquisition (SCADA) systems. However, we believe the role of timing in these systems can be understood based on the characteristics of timing in each of the two categories.

A. Data Acquisition Systems

A data acquisition CPS consists of one or more computers that acquire data from sensors that observe the physical world. Familiar examples are surveillance systems, test and measurement systems, and environmental monitoring systems.

The purpose of a data acquisition system is to acquire, record, and display data describing the physical world for eventual evaluation by humans or computers. The raw data is often correlated or otherwise subject to algorithmic processing prior to recording or display. However, any action taken based on the data is delayed in time (often due to a human in the loop), such that typically it is inappropriate to model it as a control system.

It is important that data acquired from several sensors be indexed in such a way as to permit meaningful analysis. For example, we may need to order data to 1) help establish causal relationships e.g., the flight data recorders for aircraft and data recorders on military test ranges, 2) to enable identification and tracking, e.g., surveillance, 3) to determine location e.g.,

trilateration for gunshot location, or 4) in a test and measurement system to ensure that the device under tests meets its specifications.

In almost all distributed CPS, data indexing is done using timestamps. The accuracy and precision of the timestamps is application dependent. For example, for gunshot detection, millisecond accuracy is sufficient, but for locating clandestine radio transmitters, sub-microsecond accuracy is required. In most cases, timestamps are required to be based on wall clock time, e.g., *coordinated universal time* (UTC) or *international atomic time* (TAI). In complex or extended physical systems with multiple computers this requires some sort of time distribution protocol as discussed in Section III.

Prior to the advent of digital computers, the *global navigation satellite system* (GNSS), and network-based time protocols, data acquisition systems typically recorded measurements on mechanical chart recorders, analog tape recorders or on film. The time of measurement was based either on the speed of the recording medium or in the case of film on recorded or photographed time codes such as IRIG-B[1] [1]. In the United States, wide area timing and reference to standard time made use of WWV[2] radio broadcasts [2]. With the advent of digital computers, the GNSS, and network-based time protocols, the timestamping of acquired data can easily be done with accuracies adequate for all but the most demanding applications. Timestamps are associated with data either upon receipt at a computer or, increasingly more common, at the source using local clocks synchronized over a network as discussed in Sections III and VI.

B. Control Systems

In contrast to data acquisition systems, a control CPS consists of one or more computers that 1) acquire data from sensors that observe the physical world, 2) execute a control strategy, and 3) instantiate changes in the physical world via actuators. Familiar examples are fly-by-wire aircraft control, automotive cruise control, HVAC (heating, ventilation, and air conditioning) systems for buildings, and the control of industrial processes and machinery. Prior to the advent of digital computers and networks, control systems were based on analog electronics, and mechanical, pneumatic, or hydraulic components.

Almost all modern control systems of appreciable size are based on digital computers and networks, and fall into two rough categories: safety critical and non-safety critical.

Safety critical systems are those where failure may result in loss of life, severe financial loss, or unacceptable inconvenience. In many cases, these systems are subject to government certification and required to be provably correct. Timing is invariably a major issue in the design of such systems which typically use a time-slotted control protocol to perform sensing, computation, networking, and actuation on a periodic schedule. This is enforced by a local clock and, in the case of a networked system, by local clocks synchronized over the network or by basing time on the slot boundaries in a *time division multiple access* (TDMA) network protocol, i.e., the time-triggered model of computation [3]. This fixed schedule provides a degree of composability and admits to formal verification procedures needed for certification [4]. This type of system plays a critical role in the ARINC [5] and SAE [6] standards for aircraft.

While *non-safety critical systems* can use the time-triggered approach, many require different execution strategies often due to the presence of highly asynchronous inputs. For example, a test and measurement system verifying the performance of a radar system requires careful temporal coordination between several instruments and the device being tested. However, the instrument system controllers invariably depend on a combination of code execution time and hardware triggers to determine system timing. Adjusting such systems is difficult and the result is not robust in the face of system evolution or replacement of devices or controllers with faster computers.

Alternative techniques for designing these systems are discussed in Sections V and VI.

C. Internet of Things

In recent years, mobile devices such as smart phones have proliferated to the extent that there are few places on earth without mobile coverage. Furthermore, these devices increasingly incorporate sensors. For example, the Samsung Galaxy S4 smart phone reportedly incorporates an accelerometer, a gyroscope, a hygrometer, a magnetometer, a light sensor, a thermometer, and a barometer as well as a camera. The existence of these devices in principle enables unprecedented capabilities for data acquisition and control. The paradigm of connecting such devices, usually termed the *Internet of things* (IoT), has spawned lots of research on the many technical and social issues involved [7], [8]. Many, if not all potential applications or services of the IoT, will require the explicit use of a global or local timebase. However, the implementation of such a timebase in an ever changing, mobile and largely wireless environment presents new challenges. Except possibly for localized applications, the simplest way to provide such a timebase is via the communications infrastructure, i.e., a combination of the Internet and mobile devices such as smart phones. Many wireless devices will have access to GNSS receivers which can provide sub-microsecond global time. However, GNSS has limitations as discussed in Section III-B. In particular, most mobile devices using GNSS will exhibit worse time accuracy than fixed installations due to their poorer antennae and the requirement to compute location as well as time in solving the equations based on received satellite signals. Fixed devices not only have better antennae but can have accurate position information available, e.g., from a survey or averaging spatial information over extended periods, thus increasing the accuracy of computing time from the equations.

[1]Inter-range instrumentation group (IRIG). IRIG-B is one of the time codes created by the TeleCommunications Working Group of the Inter-Range Instrumentation Group, the standards body of the Range Commanders Council of the US Department of Defense.

[2]WWV is the call sign of the United States National Institute of Standards and Technology's (NIST) shortwave radio station in Fort Collins, Colorado.

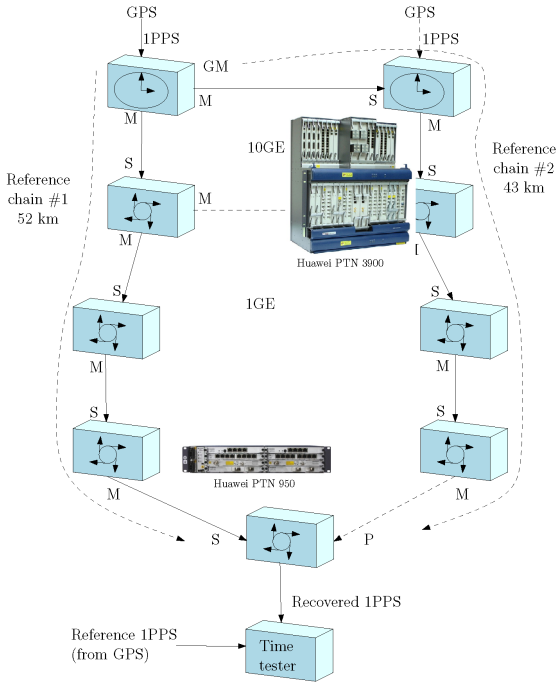

Fig. 1. Diagram of China Mobile field trial. Photos courtesy of Huawei.

Fortunately, the world's telecommunication providers, who will carry the bulk of the IoT traffic, are in the process of standardizing and deploying robust sub-microsecond time services for the operation of their own networks [9], [10]. They are therefore in a position to make time available to any connected device or application network. Figure 1 is a diagram of the topology of a field trial conducted by China Mobile that delivered better than 3μs time transfer over each of the 40-50km paths under normal traffic loads [10]. The figure also shows the type of equipment used in the trial.

III. State of the Art of Distributed Timing Systems

Today a device or system designer has a wide range of options for implementing clocks within devices and establishing a common sense of time among devices. These options result from advances during the last twenty years in oscillator technology, in GNSS, in network technology, and in time and frequency distribution protocols.

A. Oscillator Technology

In a centralized timing system, a remote device obtains the time by querying a central clock, a process that limits the accuracy to milliseconds due to communication latency fluctuations. As discussed in Section III-C, network-based time distribution protocols synchronize the local clocks of devices in a distributed system. The principal oscillator limits on obtainable synchronization accuracy and precision of clocks are the temperature stability and the noise characteristics of the local oscillators driving the clocks. In most CPS devices, the oscillators will be quartz crystal oscillators.

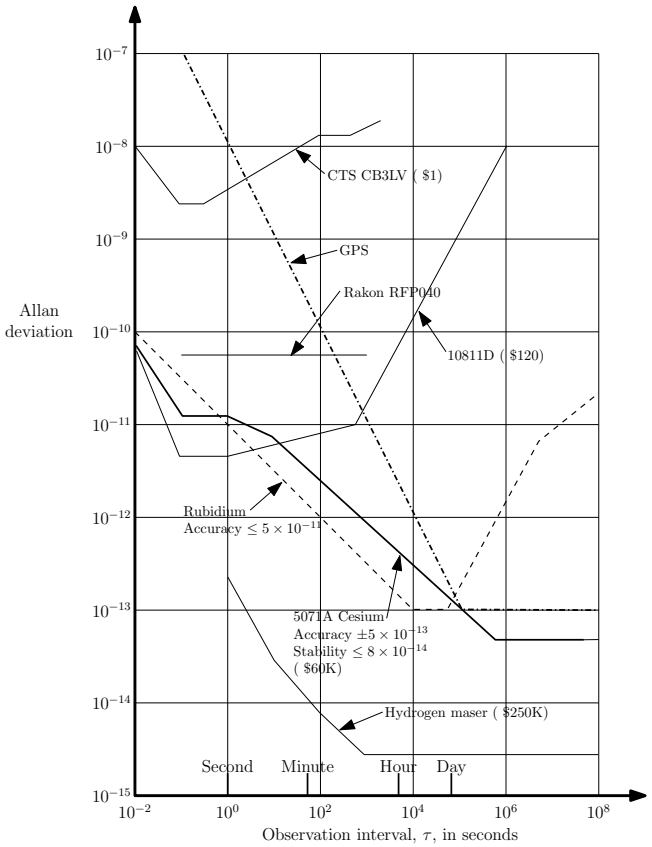

Fig. 2. Allan deviation of several oscillators.

The oscillation frequency of a quartz crystal is temperature dependent. Quartz crystal oscillators typically have a temperature fractional frequency dependence, $\Delta f/f$, from $\sim 10^{-4}$ for an uncompensated crystal to $< 4 \times 10^{-9}$ from 0°C to 70°C for an ovenized oscillator, OCXO, such as the HP 10811D. For example, a 1°C temperature rise over a 1s interval using an unsynchronized 100PPM oscillator results in a time error of 100μs over the 1s interval. Using an HP 10811D, the error is reduced to 4ns.

In addition to temperature dependence, $\Delta f/f$ is limited by noise processes within the crystal. This dependence is shown in Figure 2 for a variety of oscillators. The Allan deviation is a function of the observation interval, τ, with short observation times limited by white phase noise, long observation times limited by random walk frequency noise and the noise floor by flicker frequency noise [11]. The observation interval, τ, limits integration times whether for data averaging or synchronizing to another clock. Once the noise floor is reached, increases in the averaging time will decrease the temporal accuracy.

Figure 2 shows plots of $\Delta f/f$, the Allan deviation, for a typical uncompensated crystal, CTS CB3LV, and Rakon RFP040 and HP 10811D OCXOs. Uncompensated oscillators, e.g., CTS CB3LV, commonly used in computers cost around $1, while a high quality ovenized oscillator can cost more than $100 forcing designers to carefully consider application timing

requirements in selecting an appropriate oscillator. Recently miniature OCXOs such as the Rakon RFP040 [12] have reached the market with performance nearly equal to instrument grade oscillators such as the HP 10811D but with costs in the range of $10. For applications requiring long observation intervals a variety of atomic clocks are available. Atomic clocks depend on a quartz oscillator for short observation interval stability and on some atomic resonance for longer term stability. Atomic clocks most commonly used in CPS are the rubidium and the cesium clocks whose properties are shown in Figure 2. Rubidium clocks are found in great numbers in telecommunications base stations, which has driven the cost down to around $700, while the more expensive cesium clocks, ∼$60000, are found in lesser numbers in the core of telecommunications systems, military systems, and some scientific CPS.

Recently, so-called chip scale cesium atomic clocks have been introduced into the market [13]. These cesium-based oscillators currently have rubidium performance but in a $<17\text{cm}^3$, 35 grams, <120mW form factor. At a price of ∼$1500 they are used mostly in special circumstances such as oil exploration sensors. However, the cost will no doubt be greatly reduced as larger numbers are used. These oscillators promise much longer averaging times, up to an hour, and temperature stability $< 4 \times 10^{-10}$ from 0°C to 70°C which should eventually open up a new class of designs for CPS devices.

B. GNSS Technology

Over the past twenty years, satellite-based time and location services have found their way into numerous CPS, e.g., military applications, telecommunications base stations, surveillance and tracking. Today these services are obtainable world-wide from the *global positioning satellite* (GPS) system maintained by the United States Department of Defense and the *Glonass system* maintained by the Russian Defense Forces. The BeiDou system (formerly known as Compass) maintained by the Chinese government is currently available only in China and environs but is expected to provide global coverage in the next decade. Another regional system is the Indian Regional Navigational Satellite System. The Galileo system is a similar project of the European community but is not yet fully operational. Collectively these systems are termed Global Navigation Satellite Systems (GNSS).

GNSS can be used as a source of time for a CPS either as the source for a networked time distribution protocol or as a source for individual devices. In either case the GNSS receiver must have line of sight communication with a sufficient number of satellites. This is problematic in urban canyons and in the interior of buildings. As noted, all of the existing systems are controlled by a national government which has raised concern in some circles as to the availability during periods of international tension. Each of the GNSS systems, e.g., GPS, Glonass, derives its time from the national laboratory of the host country. The consistency of UTC time between these systems has been measured to be tens of nanoseconds at best [14]. GNSS systems are also subject to inadvertent or intentional degradation either by spoofing of satellite signals or simply raising the noise floor by jamming. These issues are a major issue for safety critical and public infrastructure systems as evidenced by discussions at recent telecommunications conferences [15]–[18].

GNSS timing accuracy over continental scale distance is unmatched by any other technology with the exception of *two-way satellite time transfer* (TWSTT) which is practical only for national laboratories. A GNSS receiver synchronizes a local clock based on signals from the satellites. The accuracy depends on the quality of the quartz oscillator and the averaging time as noted in the discussion of Figure 2 in Section III-A. A cheap crystal is unlikely to do better than 100ns. With a well-designed receiver and an oscillator permitting integration times of >100s, 100ns accuracy can be achieved. With an oscillator permitting integration times of 24 hours, ±10ns accuracy can be achieved [19].

C. Network Timing Distribution Protocols

As noted in Section II-A, time distribution prior to 1985 was typically via WWV, IRIG-B and proprietary methods. The *network time protocol* (NTP) became available around 1985 and today is the dominant network-based time distribution system and is implemented in essentially every PC in the world. NTP accuracy is on the order of milliseconds and is limited by network and operating system latency jitter. GPS has been available worldwide since 1994 and can achieve submicrosecond accuracy as noted in Section III-B.

In recent years two additional network-based time distribution protocols designed for CPS have been standardized. The first, SAE6802 [6], is designed for safety-critical systems and has been implemented in several recently designed aircraft. SAE6802 is designed for reasonably compact systems such as an aircraft and achieves sub-microsecond accuracy. SAE6802 partitions IEEE 802.3 bandwidth using time slots allocated to safety-critical and normal traffic thereby providing latency guarantees for safety-critical traffic and best effort for normal traffic.

The second major protocol is the *precision time protocol* (PTP), defined in standard IEEE 1588 [20]. IEEE 1588 specifies a master-slave synchronization hierarchy with the root, the grandmaster, determining the time scale for the system. The grandmaster in turn can be synchronized to GNSS or other sources of time traceable to international time standards. IEEE 1588 has achieved wide acceptance and implementation in the areas of telecommunications, data acquisition, industrial automation and power systems. Because it specifies specialized network bridges, so-called boundary and transparent clocks, that greatly reduce bridge timing jitter, IEEE 1588 can easily achieve 100ns synchronization over a local area network and with care <10ns [21], [22].

In addition, recent work at CERN has achieved 100ps accuracy and 8ps precision across three 5km hops of fiber optic cable using a combination of IEEE 1588 and layer 1

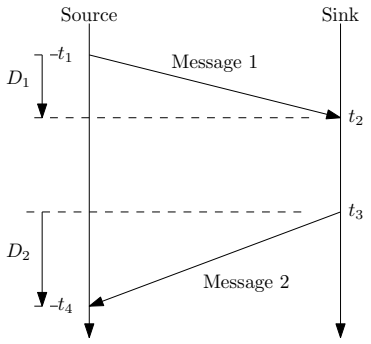

Fig. 3. Two way time transfer.

syntonization [23]. This work will likely be incorporated into the upcoming revision of IEEE 1588-2008.

Other than oscillator characteristics discussed in Section III-A, the principal accuracy limitation in two way time transfer techniques such as NTP and PTP is asymmetry in the communication paths. Two way time transfer protocols work via an exchange of signals which are timestamped as illustrated in Figure 3. The source and sink for the transfer process exchange signals which are timestamped based on the local clocks on transmission and receipt at shown. The asymmetry is the difference in the propagation times of the two messages, i.e., $D_1 - D_2$. In order to solve the equations describing this process it is necessary to assume that $D_1 = D_2$ i.e., that the path is symmetric. With this assumption the equations for computing the offset between the two clocks and the mean path delay are:

$$\mathit{offset} = [(t_2 - t_1) - (t_4 - t_3)]/2 \tag{1}$$

$$\mathit{delay} = [(t_2 - t_1) + (t_4 - t_3)]/2 \tag{2}$$

The error in the offset is one half of the asymmetry $D_1 - D_2$. Unfortunately the asymmetry cannot be measured using an exchange of signals but must be determined by some other means. There are three principal sources of this asymmetry:

- The transceivers between digital representations and the signals on the network media, i.e., the PHY[3], typically have different latencies on the transmit and receive paths. Depending on the media, the PHY asymmetry can be several tens of nanoseconds. These asymmetries are extremely difficult to measure and are best determined by the PHY manufacturer. A recent standard, IEEE 802.3bf, provides an abstract interface designed for IEEE 1588 based implementations that allows PHY manufacturers to provide these latency parameters [24].
- The communication medium itself may be asymmetric. For example the length of the forward and reverse fibers in an optical link, chromatic dispersion in a full duplex wave division multiplex fiber optic link, or the different twist rates in Category 5 twisted pair copper cable all introduce asymmetry. Asymmetry can be calibrated at installation. For some media it is possible to model asymmetry as a function of mean path delay which can be measured using two way protocols.
- In routed systems, such as the Internet, unless careful attention is paid to network design, it is possible for the forward and reverse paths to take different routes through the system. This can produce very large asymmetry. Even with identical routes the differences in network traffic on the forward and reverse paths may produce asymmetry unless on-path time support is provided, for example via boundary or transparent clocks when using IEEE 1588.

If the asymmetry is known then the appropriate correction can be made to the clock offset.

These protocols establish a global sense of precise time in a networked distributed system. Each node of the network has a local clock synchronized to its peers that may be used for a variety of purposes as outlined in next section.

IV. Explicit time techniques in CPS

Until recently the principal uses of time in CPS were found in safety-critical systems as the foundation for time triggered architectures [3] and in data acquisition and SCADA applications where data was timestamped upon receipt at a central processor. Barbara Liskov noted that NTP spurred interest in using global time to improve mainstream computer science algorithms and protocols by "Examining the messages to identify those that could be avoided by using timestamps" [25].

Given that accurate global time can now be implemented using the protocols discussed in Section III-C, it is appropriate to ask whether the techniques suggested by Liskov can be applied to CPS. We believe that this is the case and suggest the following examples as evidence.

A. Replacing Messages with Reasoning About Timestamps

Recently the Google "Spanner" project used NTP to enable better performance in a global database by "Transform(ing) commit order reasoning to timestamp order reasoning" [26]. While not a CPS, Spanner is an example of using timestamps to enforce consistent global state which is a common requirement in many CPS.

Rockwell reportedly uses timestamps in specifying motion control trajectories of components in high speed machinery to reduce the message traffic between controllers and actuation devices [27]. The actuator is provided with trajectory specification which can replace controller point-by-point command to an initialization command and less frequent minor trajectory corrections, thus saving bandwidth.

B. Reducing Complexity, Signal Conditioning, and Calibration Issues

In control systems with thousands of sensors and actuators, bringing dedicated analog wiring to a central point leads to complexity, signal conditioning, bandwidth, calibration and response time problems. Moving to a distributed system with

[3]PHY is the term often used to designate a device implementing the physical or lowest layer, i.e., layer 1, of the OSI seven layer computer network model.

Fig. 4. General Electric MarkVIe distributed measurement device. Photo courtesy of General Electric.

Fig. 5. Bridge vibration monitoring. Photo courtesy of Bruel & Kjaer.

local processing ameliorates many of these issues but requires global time to realize temporal control equivalent to the centralized architecture. General Electric took this approach in their MarkVIe control system for power plants and wind farms [28]. Figure 4 shows one of the MarkVIe distributed devices. This device communicates via Ethernet, synchronizes a local clock using IEEE 1588, and includes both a small computer as well as local signal conditioning for sensors and actuators. General Electric uses these devices to manage the several thousand sensors and actuators used in a typical thermal or nuclear power plants replacing long analog sensor cables with local processing and timestamping and using Ethernet communication to centralized supervisory control.

Bruel & Kjaer [29] use global time in data acquisition systems for vibration monitoring to improve accuracy, and to simplify wiring and calibration. For example, Figure 5 shows a bridge vibration monitoring application where a local data acquisition device captures and timestamps sensor data and communicates the data to a central analysis computer via Ethernet. Boeing has developed a similar application for data acquisition used in aircraft monitoring [30].

Similar efforts are being investigated in the power industry for substation automation and control of the grid [31], [32].

Certainly the most exciting example of the use of time-stamps and a distributed architecture to reduce complexity and

Fig. 6. CERN to Gran Sasso (CNGS) neutrino speed test. Photo courtesy of CERN.

address signal conditioning and calibration issues is the CERN White Rabbit project [23], [33]–[35]. Figure 6 illustrates a section of the accelerator that sends neutrinos from CERN to Gran Sasso, a distance of 732km in an experiment to measure the overall neutrino time of flight. The White Rabbit timing system was used to verify the timing of measurements crucial to determining that the neutrinos indeed traveled at less than the speed of light [23].

V. Design and Execution with Time as a First Class Citizen

In this section we discuss recent results in raising timing issues to first class citizen status in design and execution environments used to create CPS. In synchronous programming models [36], logic ticks are part of the language semantics. These ticks are used to order events and is not necessarily related to the actual physical time observed by the computing platform.

A. Time-Triggered Programming Models

Explicit modeling of time becomes a fundamental part in the *time-triggered* programming paradigm. In a time-triggered system, certain actions such as IO operations take place at fixed, periodic time instances. Other actions such as performing computations and updating internal states can happen at any time between IO operations. Time determinism is achieved by specifying the timing behavior of the software execution independent from the platform. At this point, time is considered purely logical and used to define an order on the computations. Functional correctness can be verified by executing the system in a way that obeys the order provided by the logical times. When deploying the software, the logical times are mapped to physical times and a check must be performed to ensure that the underlying hardware can meet the timing requirements. In such time-triggered programming models, software is typically separated into tasks. For each task, the release time, the time for reading inputs, and the time for writing outputs, is strictly defined. Time-triggered

programming models have been used in PLC designs and later formalized in languages such as Giotto [37], the hierarchical timing definition language HTL [38], the timing definition language TDL [39], and others. In general, time-triggered systems perform more computations than necessary since tasks are executed in every period, even when inputs to the task remain unchanged. One can think of optimizations to such systems by buffering inputs for the next period and only executing if inputs changed. However, there is still some overhead in the buffering and the reaction to input changes is delayed to the next period start.

B. PTIDES, an Event-Triggered Programming Model

PTIDES (Programming Temporally Integrated Distributed Event Systems) [40] is a programming model that uses time-stamps to perform deterministic distributed computations. All inputs from the environment are received by sensors which assign timestamps to each value. These timestamps are stored with the values throughout the computation path from sensors to actuators. Along the path, the timestamps are carefully modified to meet application and system timing requirements. First, the desired fixed delay between sensing and actuation is modeled. Such a delay specification is usually given by the control engineer who determines the controller performance based on the given time delay. Delay requirements between sensing and actuation can also come from the characteristics of the physical environment. For instance, one can estimate the time it takes between two events and use this information to perform an actuation on the second event. An example is given in [41], where a controller for a printing press determines the exact time to exchange paper rolls on the fly.

PTIDES is an event-triggered programming model that leverages discrete event simulation techniques for execution. In a PTIDES system, components communicate via times-tamped events. Static as well as dynamic analysis leveraging information about the structure, the inputs, and the state of other components in the system allow an efficient execution to be implemented. Unlike in typical *discrete event* (DE) simulations, the relation of timing specification in terms of logical time to physical time allows for out of timestamp order processing of unrelated events. Knowing which events are unrelated, allows not only out of timestamp order processing but also dynamic distribution over multiple cores and distribution over multiple platforms without changing the observable behavior at the actuator outputs. Some results on multicore execution of PTIDES programs are shown in Figure 7. The upper trace plots the designed events using the Ptolemy simulation platform as a PTIDES design environment. From this design, code was generated for execution on two different platforms one with a Renesas processor and the other with a multicore XMOS processor. The middle and bottom traces are measured results on the two platforms. Note that the event timing specified at design is reproduced in the execution platform to a sub-microsecond accuracy (although this is not visible on the timescale of the plots). The grey area indicates times when the processors were actually executing code and

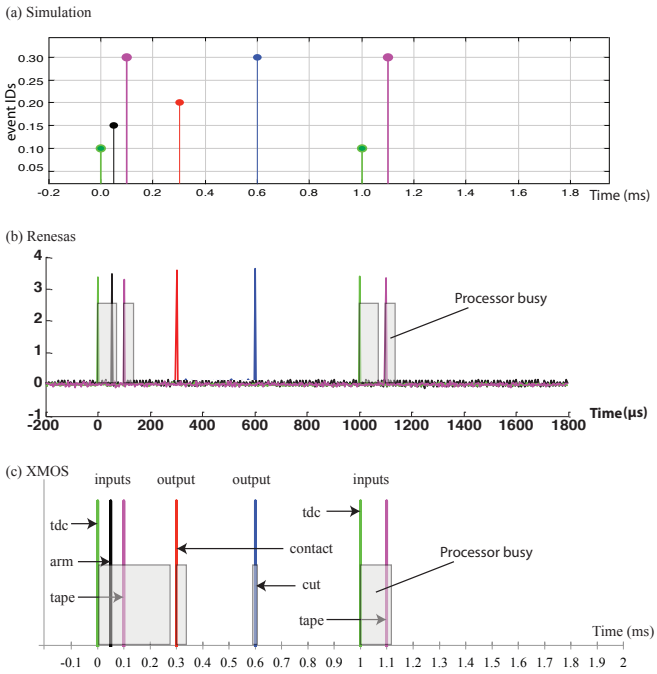

Fig. 7. Deterministic timing with different execution platforms.

clearly shows that while the time occupied in instruction set execution differed between the two processors, the external actions at actuators was identical. Although not shown in this figure, the code was compiled for single and multicore execution on the XMOS platform with no change in the external timing.

In a distributed PTIDES systems, the different platforms need to synchronize in order to reason about timestamps of events. Synchronization messages, such as null messages in Chandy and Misra's work [42], impose a great bandwidth overhead. However, by analyzing timestamps and given some knowledge about the sending system, one can compute how long to wait for new input and when input is safe to process. In particular when communication media is shared, reducing bandwidth is a desirable property.

The entire workflow around the PTIDES programming model including modeling, simulation, implementation, and analysis is presented in Figure 8. A modeling and simulation environment has been implemented in Ptolemy II [43], a framework for heterogeneous systems. This tool allows for exploring designs of PTIDES models with different distribution of functions to components, scheduling strategies and underlying architecture components. With this framework it is possible to research the interaction of PTIDES-based models with models of other system components such as plant models and networks. A code generation framework is outlined in [44] and an implementation of this approach has been performed by extending the Ptolemy code generation functionality. PTIDES systems have been implemented on bare iron using a light weight operating system, PtidyOS, that uses an *earliest deadline first* (EDF) scheduling algorithm

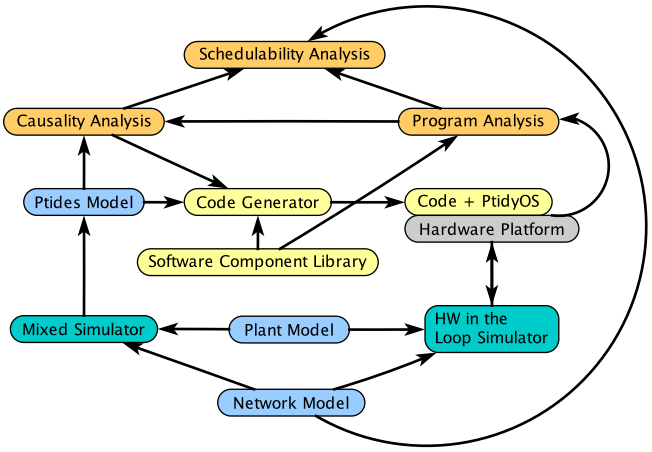

Fig. 8. PTIDES workflow.

to implement the PTIDES timing requirements [45]. Recent results show that the schedulability of PTIDES programs is decidable. More precisely, the schedulability problem can be reduced to a reachability problem for timed automata.

VI. Time-Critical Embedded Systems

The cyber part of CPS is inherently a *real-time system*; sensors and actuators are interacting with the physical environment at distinct point in time. These systems are often realized as *embedded systems*—cheap platforms with limited memory and computation power. In this section we review different aspects of implementing real-time applications on embedded systems.

A. Tasks, Deadlines, and Scheduling

A real-time system must react to external or internal timed events according to specified timing constraints. These constraints are often defined as *deadlines*. A real-time program fragment, typically referred to as a *task*, must finish executing before its deadline is reached. As a consequence, the time it takes for a real-time system to execute a task is not just a performance factor; it is also a correctness criterion.

Missed deadlines may have different consequences depending on the type of the real-time application. Real-time tasks are often divided into three categories [46]:

- *Hard tasks* must finish executing before its deadline, otherwise catastrophic consequences may occur. Example of such systems are sensory data acquisition, image processing, and control systems.
- *Firm tasks* are tasks where missed deadlines do no harm, but there is no use of the computation result after missing the deadline. Network and multimedia applications are examples where firm tasks may be used.
- *Soft tasks* are tasks that still have some utility even if the deadline is missed. System-user interactions are examples of soft real-time tasks.

If a single processor executes several concurrent tasks or if a multicore system executes more tasks than available cores then tasks must be *scheduled*. A *scheduling algorithm* determines at what time, in which order, and on which cores different tasks should be scheduled. Besides the timing constraints used in forming deadlines, tasks may also have precedence and resource constraints. A schedule is said to be *feasible* if all tasks can be scheduled to meet all constraints. If there exists at least one algorithm that can generate a feasible schedule, a set of tasks are *schedulable* [46].

Real-time scheduling theory is a mature but still very active area of research. Scheduling policies are commonly implemented in *real-time operating systems* (RTOS) and are fundamental for efficiently utilizing limited resources of embedded systems.

B. Timing Analysis

In real-time scheduling theory, the computation time of specific tasks are assumed to be known. In general, the computation time for a specific tasks is, however, not constant; the execution time depends highly on input data, system state, and timing behavior of the hardware.

Figure 9 depicts the distribution of execution times for a specific task. The upper bound is called *worst-case execution time* (WCET) and the lower bound *best-case execution time* (BCET). In general, it is very hard to compute exact numbers of WCET and BCET. Instead, the goal is to estimate *safe* upper and lower bounds of WCET and BCET, respectively. A safe estimate of the WCET means that the estimated number is equal to or larger than the real WCET. The upper bound of WCET should also be *tight*, meaning that the uncertainty between the real WCET and the estimated upper bound is as small as possible.

In industry, it is common to estimate the WCET by *measuring* the execution time for a set of test cases with different input data. Measured WCET cannot, however, be guaranteed to be safe in general [47]. The drawback with this approach is that measurements cannot be performed for all possible inputs and machine states—only a small subset can in practice be covered. As a consequence, the estimated values are given extra safety marginals, introducing overall pessimistic WCET bounds.

An alternative approach is to perform static timing analysis on a task's code. Static WCET analysis typically consists of three distinct phases: program flow analysis, microarchitectural analysis, and global bound analysis [48]. The *program*

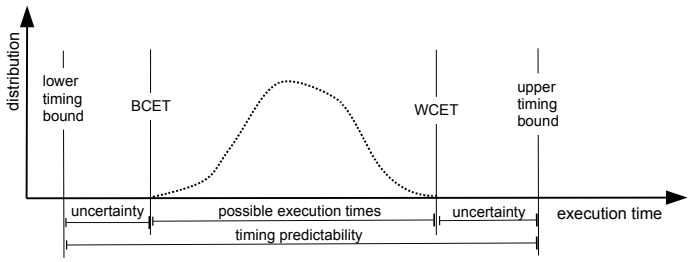

Fig. 9. Probability distribution of execution time.

flow analysis phase identifies loop bounds [49] and infeasible paths [50] of the program. Such constraints on the program's control flow are commonly referred to as *flow facts*. Some WCET tools perform this analysis automatically, either on the executable binary or on the source code, whereas other tools require users to specify these constraints manually. The *microarchitectural analysis* determines safe time bounds of individual basic blocks[4]. The microarchitectural analysis uses an abstract architectural model to estimate a safe upper bound for each basic block. However, caches and pipelines—essential for achieving good average case performance—make static timing analysis particularly challenging. Analysis techniques based on abstract interpretation [51] can be used for analyzing both caches [52] and pipelines. The strength of these methods highly depends on the complexity of the underlying hardware. Finally, the *global bound analysis* phase computes an upper bound of the WCET by combining the results from the two previous phases. This computation phase is typically solved as an integer linear programming (ILP) problem, using the *implicit path enumeration technique* [53].

Although significant progress of timing analysis has been achieved during the last two decades, several challenges remain. New methods for program flow analysis are needed to enable analysis of large complex programs. Complex hardware, including new multicore platforms with shared caches, introduces new analysis challenges.

C. Precision Timed Hardware

Timing analysis can be vastly simplified if the underlying hardware has *predictable timing* behavior. By predictable timing we mean that the size of the machine state is limited. The idea of *precision timed* (PRET) machines, a new era of processors with predictable timing, was first advocated by Edwards and Lee [54] in 2007.

One distinguished feature of a PRET machine is that the *instruction set architecture* (ISA) is augmented with timing instructions. For instance, PTARM [55] is an ARM-based processor developed at UC Berkeley. The ARM ISA is extended with several instructions for programming with real-time. The main purpose of extending the ISA with timing instructions is to enable *portability* of real-time systems.

Another desirable property of a real-time system is *testability*. Although formal verification of embedded systems is a very active area of research, testing is the predominate verification technique used by industry. However, if a system is executed several times with the same input data and returns different output for each execution, testing becomes extremely difficult. As a consequence, repeatability of both functional and timing behavior is vital for testability. A common way for PRET machines to achieve repeatability is to use thread-interleaved pipelines [56] (to remove pipeline hazards) and by replacing caches with scratchpad memories [57]. Other predictable processor exists, based on different design choices [58], [59].

[4]A basic block is a sequence of instructions ending with a branch instruction.

D. Precision Timed Infrastructure

Many modern modeling environments for CPS [60], such as Modelica [61], Simulink [62], Ptolemy II [43], and Modelyze [63], all include ways of precisely modeling and controlling time. Several of these environments can also compile the functional behavior of a model into C code, but few can give any guarantees about timing correctness. In enabling correct-by-construction of cyber-physical models, the challenge is to compile or synthesize a model's cyber parts so that the behavior of a simulated model and the real system coincide. The key challenge of this *model fidelity* problem is to guarantee the correctness of timing [64].

A recent research initiative, called *precision timed infrastructure* [65], addresses this problem. Besides PRET hardware, such an infrastructure should include both an *PRET intermediate language* and a *PRET compiler*. The purpose of an intermediate language is to act as an abstraction layer between the PRET hardware and the modeling language. The key advantage of this approach is that the intermediate language exposes timing constructs for expressing real-time, but hides machine dependent details of the platform. In particular, memory hierarchies and hardware threads must be abstracted away, so that the modeling environment's code generators do not need to consider platform dependent details. The benefit with this approach is twofold: Firstly, portability is improved; timing is now part of the language semantics and is not just an accidental consequence of implementation. Secondly, the PRET compiler—which has the intermediate language as source language and an PRET ISA as target language—can now perform timing dependent optimizations. In particular, scratchpad allocation schemes may be used to improve WCET for specified timing constraints.

VII. Conclusions and Future work

In this paper we have provided an overview of some of the more important issues and techniques related to timing issues in designing and implementing CPS. Oscillator and clock technology for CPS is well developed but further work is needed to reduce costs, particularly for future mobile applications with demanding temporal accuracy requirements. GNSS systems play an important role in widely distributed CPS but are vulnerable to a variety of failures and attacks. Time transfer over networks is readily available from NTP at millisecond accuracy and with recent protocols such as IEEE 1588 to sub-microsecond accuracy. However increasing the accuracy of network time transfer will require additional work to resolve asymmetry problems in a way that is cost effective for use in CPS. The design and implementation of embedded controllers for CPS that ensure correct timing remains a difficult issue. Compact safety-critical systems are reasonably well understood but the design of systems with highly asynchronous inputs, extensive spatial distribution, and systems of systems remains problematic although significant progress has been made in recent years. Additional work on design environments that allow better specification and

verification of timing requirements is needed as well as more work in controlling processor instruction execution times.

REFERENCES

[1] Timing Committee Telecommunications and Timing Group- Range Commanders Council, "IRIG Serial time code formats," September, 2004. [Online]. Available: http://www.irigb.com/pdf/wp-irig-200-04.pdf

[2] NIST, "Radio station WWV." [Online]. Available: http://www.nist.gov/pml/div688/grp40/

[3] H. Kopetz, *Real-Time Systems : Design Principles for Distributed Embedded Applications*. Springer, 1997.

[4] J. Rushby and W. Steiner, "TTA and PALS: Formally veried design patterns for distributed cyber-physical systems," in *30th IEEE/AIAA Digital Avionics Systems Conference (DASC)*, Seattle, WA, 2011.

[5] ARINC, "ARINC 653 family of standards," November, 2010. [Online]. Available: https://www.arinc.com/cf/store/

[6] As-2d2 Deterministic Ethernet And Unified Networking, "AS6820 Time-Triggered Ethernet," SAE, Standard Specification, November 1, 2011.

[7] L. Atzori, A. Iera, and G. Morabito, "The internet of things: A survey," *Computer Networks*, vol. 54, no. 15, pp. 2787–2805, 2010.

[8] M. Zorzi, A. Gluhak, S. Lange, and A. Bassi, "From today's intranet of things to a future internet of things: a wireless-and mobility-related view," *Wireless Communications, IEEE*, vol. 17, no. 6, pp. 44–51, 2010.

[9] ITU, "Network synchronization and time distribution performance," 2013. [Online]. Available: http://www.itu.int/en/ITU-T/studygroups/2013-2016/15/Pages/q13.aspx

[10] M. Ouellette, K. Ji, S. Liu, and H. Li, "Using IEEE 1588 and boundary clocks for clock synchronization in telecom networks," *Communications Magazine, IEEE*, vol. 49, no. 2, pp. 164–171, 2011.

[11] J. Vig, "Quartz crystal resonators and oscillators for frequency control and timing applications - a tutorial," April, 2012. [Online]. Available: http://www.ieee-uffc.org/frequency-control/learning-vig-tut.asp

[12] Rakon, "RFPO40 SMD Oven Controlled Crystal Oscillator," 2013. [Online]. Available: http://www.rakon.com/products/families/ocxo-ocso

[13] Symmetricom, "SA.45s CSAC," 2013. [Online]. Available: http://www.symmetricom.com/products/frequency-references/

[14] "Circular T," BUREAU INTERNATIONAL DES POIDS ET MESURES, Tech. Rep. 303, April 10 2013.

[15] A. Gupta, "Mitigation of GPS vulnerability using time transfer over microwave systems," *Telcordia -NIST - ATIS Workshop on Synchronization in Telecommunication Systems*, 2013.

[16] ——, "GPS spoofing: A brief survey of methods, effects and counter measures," *Telcordia -NIST - ATIS Workshop on Synchronization in Telecommunication Systems*, 2012.

[17] J. Merrill, "GPS vulnerability and backing up critical infrastructure," *Telcordia -NIST - ATIS Workshop on Synchronization in Telecommunication Systems*, 2012.

[18] C. Curry and G. Jolly, "GPS jamming quantifying the threat interference of GPS anti-jam techniques with accurate time determination," *Telcordia -NIST - ATIS Workshop on Synchronization in Telecommunication Systems*, 2013.

[19] M. Weiss, "One way GPS time transfer," 2013. [Online]. Available: http://tf.nist.gov/time/oneway.htm

[20] IEEE Instrumentation and Measurement Society, "1588: IEEE standard for a precision clock synchronization protocol for networked measurement and control systems," IEEE, Standard Specification, July 24, 2008.

[21] D. Vook, B. Hamilton, A. Fernandez, J. Burch, and V. Srikantam, "Update on high precision time synchronization," in *Proceedings of the 2005 Conference on IEEE-1588*, Zurich, CH, 2005.

[22] D. Mohl and M. Renz, "Improved synchronization behavior in highly cascaded networks," in *International IEEE Symposium on Precision Clock Synchronization for Measurement, Control and Communication*, Vienna, AT, 2007.

[23] M. Lipinski, T. Wlostowski, J. Serrano, P. Alvarez, J. David, G. Cobas, A. Rubini, and P. Moreira, "Performance results of the first white rabbit installation for cngs," in *International IEEE Symposium on Precision Clock Synchronization for Measurement, Control and Communication*, San Francisco, CA, 2012.

[24] IEEE Computer Society, "IEEE standard for information technology–local and metropolitan area networks– part 3: CSMA/CD access method and physical layer specifications amendment 7: Media access control (mac) service interface and management parameters to support time synchronization protocols," IEEE, Standard Specification, 2011.

[25] B. Liskov, "Practical uses of synchronized clocks in distributed systems," *Distributed Computing*, vol. 6, no. 4, pp. 211–219, 1993.

[26] J. Corbett, J. Dean, M. Epstein, A. Fikes, C. Frost, J. Furman, S. Ghemawat, A. Gubarev, C. Heiser, P. Hochschild, W. Hsieh, S. Kanthak, E. Kogan, H. Li, A. Lloyd, S. Melnik, D. Mwaura, D. Nagle, S. Quinlan, R. Rao, L. Rolig, Y. Saito, M. Szymaniak, C. Taylor, R. Wang, and D. Woodford, "Spanner: Google's Globally-Distributed Database," in *10th USENIX Symposium on Operating Systems Design and Implementation*, Hollywood, CA, 2012.

[27] K. Harris, "An application of IEEE 1588 to industrial automation," in *Precision Clock Synchronization for Measurement, Control and Communication, 2008. ISPCS 2008. IEEE International Symposium on*. IEEE, 2008, pp. 71–76.

[28] M. Shepard, D. Fowley, R. Jackson, and D. King, "Implementation of IEEE Std.-1588 in a Networked I/O Node," in *Proceedings of the 2003 Workshop on IEEE-1588, NIST publication NISTIR 7070*, Gaithersburg, MD, 2003. [Online]. Available: http://www.nist.gov/publication-portal.cfm

[29] Bruel & Kjaer, "Features and specifications," 2013. [Online]. Available: http://www.bksv.com/products/frontends/lanxi/features-and-specifications.aspx

[30] H. Mach, E. Grim, O. Holmeide, and C. Calley, "PTP enabled network for flight test data acquisition and recording," in *Precision Clock Synchronization for Measurement, Control and Communication, 2007. ISPCS 2007. IEEE International Symposium on*. IEEE, 2007, pp. 110–115.

[31] F. Steinhauser, C. Riesch, and M. Rudigier, "IEEE 1588 for time synchronization of devices in the electric power industry," in *Precision Clock Synchronization for Measurement Control and Communication (ISPCS), 2010 International IEEE Symposium on*. IEEE, 2010, pp. 1–6.

[32] M. Lixia, C. Muscas, and S. Sulis, "Application of IEEE 1588 to the measurement of synchrophasors in electric power systems," in *Precision Clock Synchronization for Measurement, Control and Communication, 2009. ISPCS 2009. International Symposium on*. IEEE, 2009, pp. 1–6.

[33] M. Antonello, B. Baibussinov, P. Benetti, F. Boffelli, E. Calligarich, N. Canci, S. Centro, A. Cesana, K. Cieslik, D. Cline *et al.*, "Precision measurement of the neutrino velocity with the ICARUS detector in the CNGS beam," *Journal of High Energy Physics*, vol. 2012, no. 11, pp. 1–21, 2012.

[34] P. Moreira, J. Serrano, T. Wlostowski, P. Loschmidt, and G. Gaderer, "White rabbit: Sub-nanosecond timing distribution over ethernet," in *Precision Clock Synchronization for Measurement, Control and Communication, 2009. ISPCS 2009. International Symposium on*. IEEE, 2009, pp. 1–5.

[35] G. Gaderer, P. Loschmidt, E. G. Cota, J. H. Lewis, J. Serrano, M. Cattin, P. Alvarez, P. M. Oliveira Fernandes Moreira, T. Wlostowski, J. Dedic, C. Prados, M. Kreider, R.Baer, S.Rauch, and T.Fleck, "The white rabbit project," in *Int. Conf. on Accelerator and Large Experimental Physics Control Systems*, Kobe, Japan, 2009.

[36] A. Benveniste and G. Berry, "The synchronous approach to reactive and real-time systems," *Proceedings of the IEEE*, vol. 79, no. 9, pp. 1270–1282, 1991.

[37] T. A. Henzinger, B. Horowitz, and C. M. Kirsch, "Giotto: A time-triggered language for embedded programming," in *EMSOFT 2001*, vol. LNCS 2211. Tahoe City, CA: Springer-Verlag, 2001, pp. 166–184.

[38] A. Ghosal, T. A. Henzinger, D. Iercan, C. M. Kirsch, and A. Sangiovanni-Vincentelli, "A hierarchical coordination language for interacting real-time tasks," in *Sixth Annual Conference on Embedded Software (EMSOFT)*. Seoul, Korea: ACM, 2006.

[39] W. Pree and J. Templ, "Modeling with the timing definition language (TDL)," in *Automotive Software Workshop San Diego (ASWSD) on Model-Driven Development of Reliable Automotive Services*, ser. LNCS. San Diego, CA: Springer, 2006.

[40] Y. Zhao, J. Liu, and E. A. Lee, "A programming model for time-synchronized distributed real-time systems," in *13th IEEE Real Time and Embedded Technology and Applications Symposium, 2007. RTAS '07*, April 2007, pp. 259 – 268. [Online]. Available: http://chess.eecs.berkeley.edu/pubs/325.html

[41] P. Derler, J. Eidson, S. Goose, E. A. Lee, and M. Zimmer, "Deterministic execution of ptides programs," EECS Department, University of California, Berkeley, Tech. Rep. UCB/EECS-2013-65, May 2013. [Online]. Available: http://www.eecs.berkeley.edu/Pubs/TechRpts/2013/EECS-2013-65.html
[42] K. M. Chandy and J. Misra, "Distributed simulation: A case study in design and verification of distributed programs," *IEEE Trans. on Software Engineering*, vol. 5, no. 5, pp. 440–452, 1979.
[43] J. Eker, J. W. Janneck, E. A. Lee, J. Liu, X. Liu, J. Ludvig, S. Neuendorffer, S. Sachs, and Y. Xiong, "Taming heterogeneity—the Ptolemy approach," *Proceedings of the IEEE*, vol. 91, no. 2, pp. 127–144, 2003. [Online]. Available: http://www.ptolemy.eecs.berkeley.edu/publications/papers/03/TamingHeterogeneity/
[44] P. Derler, J. Eidson, E. A. Lee, S. Matic, and M. Zimmer, "Model-based development of deterministic, event-driven, real-time distributed systems," in *International Workshop on Model-Based Design with a Focus on Extra-Functional Properties (MBDEFP)*, 2011. [Online]. Available: http://chess.eecs.berkeley.edu/pubs/895.html
[45] J. Zou, S. Matic, and E. Lee, "PtidyOS: A lightweight microkernel for Ptides real-time systems," in *Real-Time and Embedded Technology and Applications Symposium (RTAS)*. IEEE, April 2012.
[46] G. C. Buttazzo, *Hard real-time computing systems: predictable scheduling algorithms and applications*, 3rd ed. Springer, 2011.
[47] R. Wilhelm, J. Engblom, A. Ermedahl, N. Holsti, S. Thesing, D. Whalley, G. Bernat, C. Ferdinand, R. Heckmann, T. Mitra, F. Mueller, I. Puaut, P. Puschner, J. Staschulat, and P. Stenström, "The Worst-Case Execution-Time Problem - Overview of Methods and Survey of Tools," *ACM Transactions on Embedded Computing Systems*, vol. 7, pp. 36:1–36:53, May 2008.
[48] R. Wilhelm, D. Grund, J. Reineke, M. Schlickling, M. Pister, and C. Ferdinand, "Memory hierarchies, pipelines, and buses for future architectures in time-critical embedded systems," *IEEE Transactions on Computer Aided Design*, vol. 28, no. 7, pp. 966 – 978, 2009.
[49] J. Knoop, L. Kovcs, and J. Zwirchmayr, "Symbolic Loop Bound Computation for WCET Analysis," in *Perspectives of Systems Informatics*, ser. LNCS, E. Clarke, I. Virbitskaite, and A. Voronkov, Eds. Springer, 2012, vol. 7162, pp. 227–242.
[50] J. Gustafsson, A. Ermedahl, C. Sandberg, and B. Lisper, "Automatic derivation of loop bounds and infeasible paths for WCET analysis using abstract execution," in *Proceedings of the 27th IEEE International Real-Time Systems Symposium (RTSS'06)*. IEEE, 2006, pp. 57–66.
[51] P. Cousot and R. Cousot, "Abstract interpretation: a unified lattice model for static analysis of programs by construction or approximation of fixpoints," in *Proceedings of the 4th ACM SIGACT-SIGPLAN symposium on Principles of programming languages*, ser. POPL '77. New York, USA: ACM Press, 1977, pp. 238–252.
[52] C. Ferdinand and R. Wilhelm, "Efficient and precise cache behavior prediction for real-time systems," *Real-Time Systems*, vol. 17, no. 2, pp. 131–181, 1999.
[53] Y.-T. Li and S. Malik, "Performance analysis of embedded software using implicit path enumeration," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 16, no. 12, pp. 1477–1487, 1997.
[54] S. A. Edwards and E. A. Lee, "The case for the precision timed (pret) machine," in *Proceedings of the 44th annual conference on Design automation*, June 2007, pp. 264 – 265.
[55] I. Liu, J. Reineke, D. Broman, M. Zimmer, and E. A. Lee, "A PRET Microarchitecture Implementation with Repeatable Timing and Competitive Performance," in *Proceedings of the 30th IEEE International Conference on Computer Design (ICCD 2012)*. IEEE, 2012.
[56] E. Lee and D. Messerschmitt, "Pipeline interleaved programmable dsp's: Synchronous data flow programming," *Acoustics, Speech and Signal Processing, IEEE Transactions on*, vol. 35, no. 9, pp. 1334–1345, 1987.
[57] R. Banakar, S. Steinke, B. Lee, M. Balakrishnan, and P. Marwedel, "Scratchpad memory: A design alternative for cache on-chip memory in embedded systems," in *In Tenth International Symposium on Hardware/Software Codesign (CODES), Estes Park*. ACM, 2002, pp. 73–78.
[58] S. Andalam, P. Roop, and A. Girault, "Predictable multithreading of embedded applications using PRET-C," in *Proceedings of the 8th IEEE/ACM International Conference on Formal Methods and Models for Codesign (MEMOCODE)*. IEEE, 2010, pp. 159–168.
[59] M. Schoeberl, "A Java processor architecture for embedded real-time systems," *Journal of Systems Architecture*, vol. 54, no. 1-2, pp. 265 – 286, 2008.
[60] D. Broman, E. A. Lee, S. Tripakis, and M. Törngren, "Viewpoints, formalisms, languages, and tools for cyber-physical systems," in *Proceedings of the 6th International Workshop on Multi-Paradigm Modeling*, 2012.
[61] *Modelica—A Unified Object-Oriented Language for Physical Systems Modeling—Language Specification*, 2012, http://www.modelica.org.
[62] MathWorks, "The Mathworks - Simulink - Simulation and Model-Based Design," http://www.mathworks.com/products/simulink/ [Last accessed: May 8, 2013].
[63] D. Broman and J. G. Siek, "Modelyze: a gradually typed host language for embedding equation-based modeling languages," EECS Department, University of California, Berkeley, Tech. Rep. UCB/EECS-2012-173, June 2012.
[64] D. Broman, "High-confidence cyber-physical co-design," in *Proceedings of the Work-in-Progress (WiP) session of the 33rd IEEE Real-Time Systems Symposium (RTSS 2012)*, 2012, p. 12.
[65] D. Broman, M. Zimmer, Y. Kim, H. Kim, J. Cai, A. Shrivastava, S. A. Edwards, and E. A. Lee, "Precision Timed Infrastructure: Design Challenges," in *Proceedings of the Electronic System Level Synthesis Conference (to appear)*. IEEE, 2013.

David Broman is currently a visiting scholar at UC Berkeley, USA, working in the Ptolemy group at the Electrical Engineering & Computer Science department. He is an assistant professor at Linkping University in Sweden, where he also received his PhD in computer science in 2010. David's research interests include programming and modeling language theory, compiler technology, software engineering, and mathematical modeling and simulation of cyber-physical systems. He has worked five years within the software security industry, co-founded the EOOLT workshop series, and is member of the Modelica Association and the Modelica language design group.

Patricia Derler received her Ph.D. in Computer Science from the University of Salzburg, Austria and she did her undergraduate studies at the University of Hagenberg, Austria. After she graduated, she started a postdoctoral position at the University of California, Berkeley. She currently holds this position and is a member of the PTIDES group where she does research on the design and simulation of cyber-physical systems, deterministic models of computation and modeling of time in distributed systems.

John C. Eidson received his BS and MS degrees from Michigan State University and his PhD. Degree from Stanford University all in electrical engineering. He retired in 2009 after a career in the central research laboratories of Varian Associates, Hewlett-Packard and most recently, Agilent Technologies. He is a co-chair of the IEEE 1588 standards committee. He is a life fellow of the IEEE, the recipient of the 2007 Technical Award of the IEEE I&M Society, and a co-recipient of the 2007 Agilent Laboratories Barney Oliver Award for Innovation. He is currently a visiting scholar in the PTIDES group at the University of California at Berkeley and is interested in the application of synchronized clocks to cyber-physical systems.