in Proc. of IEEE Photonics Society Summer Topicals, Newport Beach, CA, 20-22 July 2009

Time-Critical Networking – Extended Abstract for Invited Presentation

Edward A. Lee
University of California at Berkeley
eal@eecs.berkeley.edu

***Abstract*—This paper considers the networking needs of cyber-physical systems, which integrate computation and networking with physical dynamics. It argues that CPS demands networks where time is a semantic property, not just a quality factor.**

I. Cyber-Physical Systems

Cyber-Physical Systems (CPS) are integrations of computation with physical processes. Embedded computers and networks monitor and control the physical processes, usually with feedback loops where physical processes affect computations and vice versa. In the physical world, the passage of time is inexorable and concurrency is intrinsic. Neither of these properties is present in today's computing and networking abstractions.

Applications of CPS arguably have the potential to dwarf the 20-th century IT revolution. They include high confidence medical devices and systems, assisted living, traffic control and safety, advanced automotive systems, process control, energy conservation, environmental control, avionics, instrumentation, critical infrastructure control (electric power, water resources, and communications systems for example), distributed robotics (telepresence, telemedicine), defense systems, manufacturing, and smart structures. It is easy to envision new capabilities, such as distributed micro power generation coupled into the power grid, where timing precision and security issues loom large. Transportation systems could benefit considerably from better embedded intelligence in automobiles, which could improve safety and efficiency. Networked autonomous vehicles could dramatically enhance the effectiveness of our military and could offer substantially more effective disaster recovery techniques. Networked building control systems (such as HVAC and lighting) could significantly improve energy efficiency and demand variability, reducing our dependence on fossil fuels and our greenhouse gas emissions. In communications, cognitive radio could benefit enormously from distributed consensus about available bandwidth and from distributed control technologies. Financial networks could be dramatically changed by precision timing. Large scale services systems leveraging RFID and other technologies for tracking of goods and services could acquire the nature of distributed real-time control systems. Distributed real-time games that integrate sensors and actuators could change the (relatively passive) nature of on-line social interactions.

The economic impact of any of these applications would be huge. Networking technologies today, however, may unnecessarily impede progress towards these applications. They largely lack of temporal semantics, providing instead best effort techniques that make predictable and reliable real-time performance difficult. Many applications will not be achievable without substantial changes in the core abstractions.

The problem I address is that CPS requires networks with temporal semantics. The passage of time is a central feature in CPS — in fact, it is this key constraint that distinguishes these systems from distributed computing in general. Time is central to predicting, measuring, and controling properties of the physical world: given a (deterministic) physical model, the initial state, the inputs, and the amount of time elapsed, one can compute the current state of the plant. This principle provides the foundations of control theory. However, for current mainstream networking paradigms, timing behavior is highly variable and difficult to control. Engineers are stuck with a prototype-and-test style of design, which leads to brittle systems that do not easily evolve to handle small changes in operating conditions and hardware platforms.

But surely the "right time" is expecting too much, the reader may object. The physical world is neither precise nor reliable, so why should we demand this of networking systems? Instead, we must make systems robust and adaptive, building reliable systems out of unreliable components. While I agree that systems need to be designed to be robust, we should not discard the reliability we have. Electronics technology is astonishingly precise and reliable, more than any other human invention. We routinely deliver circuits that perform a logical function essentially perfectly, on time, billions of times per second, for years. Shouldn't we exploit this remarkable achievement?

We have been lulled into a false sense of confidence by the considerable successes of embedded software, for example in automotive, aviation, and robotics applications. But the potential is vastly greater; we have reached a tipping point, where computing and networking may be integrated into the vast majority of artifacts that humans make. However, as we move to more networked, more complex, and more intelligent applications, the problems are going to get worse. Embedded

This work was supported in part by the Center for Hybrid and Embedded Software Systems (CHESS) at UC Berkeley, which receives support from the National Science Foundation (NSF awards #0720882 (CSR-EHS: PRET) and #0720841 (CSR-CPS)), the U. S. Army Research Office (ARO #W911NF-07-2-0019), the U. S. Air Force Office of Scientific Research (MURI #FA9550-06-0312), the Air Force Research Lab (AFRL), the State of California Micro Program, and the following companies: Agilent, Bosch, Lockheed-Martin, National Instruments, Thales, and Toyota.

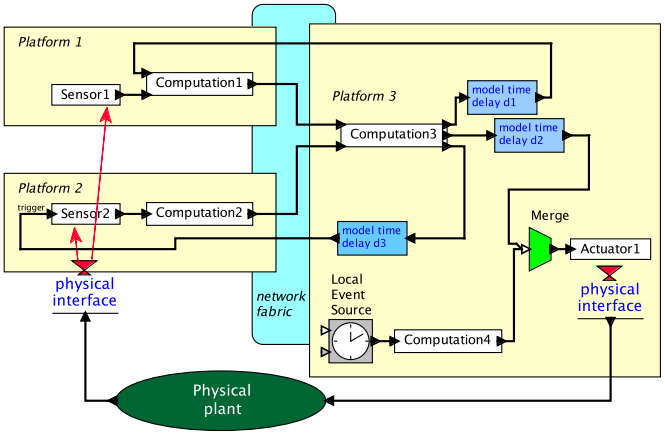

Fig. 1. Example structure of applications considered.

systems will no longer be black boxes, designed once and immutable in the field. Instead, they will be pieces of a larger system, a dance of electronics, networking, and physical processes.

Because of the lack of temporal semantics, today's distributed embedded applications do not use generic networking technology such as Ethernet and TCP/IP. In avionics and automotive applications, engineers use time-triggered architectures [3] and FlexRay, and in industrial automation and instrumentation systems, they use Foundation Fieldbus systems, and CAN busses, for example. More recent developments, are building temporal properties into generic networks. Synchronous Ethernet and Time-Triggered Ethernet are two such promising examples.

A new technology that promises to radically change the way distributed software for CPS is designed is time synchronization [2] such as IEEE 1588 [1], which can deliver wall-clock synchronization with bounded errors. In fact, an Ethernet PHY chip introduced in 2007 by National Semiconductor advertises a clock precision on the order 8 ns on a local-area network. Such high precision concurrence of clocks on a network is a game-changing phenomenon that enables a radically different approach to distributed software development. Unlike time-triggered networks, these techniques interplay well with non-time-sensitive uses of the network, including TCP/IP communication.

An interesting example supporting this point is General Electric's Mark^{TM}VIe Control Platform, which has integrated high-precision network time synchronization based on IEEE 1588 into its IO processors. To date, GE has manufactured in excess of 50,000 such units. This control platform is used for gas and steam turbine controls, wind turbines, hydro control, and other distributed control systems. A current challenge for such systems is to enable distributed micro power generation coupled into the power grid, where the complexity of the control system becomes much higher and its structure dynamic.

The structure of such applications is sketched in figure 1, which shows a small example with three networked compute platforms each with its own sensors and actuators. The actuators affect the data provided by the sensors through the physical plant. In an automation application, for example, the actuators could be motion control for high-speed printing presses, the sensors could detect disruptions, and the control algorithms could include rapid shutdown modes to prevent damage to the equipment in case of paper jams. Such shutdowns need to be tightly orchestrated across the entire system to prevent disasters. Similar situations are found in high-end instrumentation systems and in energy production and distribution.

Stankovic et al. [5] state "existing technology for RTES [real-time embedded systems] design does not effectively support the development of reliable and robust embedded systems," and discuss additional applications of CPS to sensor information systems for assisted living (SISAL), emergency response, and protecting critical infrastructures. They cite a need to "raise the level of programming abstraction," stating that "existing technology for RTES [real-time embedded systems] design does not effectively support the development of reliable and robust embedded systems." I argue that raising the level of abstraction is insufficient. We have to also fundamentally change the abstractions that are used. Timing cannot be effectively introduced at "higher levels of abstraction" if it is entirely absent from the lower levels of abstraction on which these are built.

We require *robust* and *predictable* designs with *repeatable* temporal dynamics (for a detailed discussion of the meanings of these terms, see [4,]). We must do this by building abstractions that appropriate reflect the realities of distributed systems. The result will be cyber-physical designs that can be much more extensively networked, can include more adaptive control logic, and can evolve over time, without suffering from the *brittleness* of today's designs, where small changes have big consequences. Timing properties must be built into the semantics of networking, rather than being considered after the fact as a quality factor.

References

[1] IEEE Instrumentation and Measurement Society. 1588: IEEE standard for a precision clock synchronization protocol for networked measurement and control systems. Standard specification, IEEE, November 8 2002.

[2] Svein Johannessen. Time synchronization in a local area network. *IEEE Control Systems Magazine*, pages 61–69, 2004.

[3] Hermann Kopetz and Gnter Bauer. The time-triggered architecture. *Proceedings of the IEEE*, 91(1):112–126, 2003.

[4] Edward A. Lee. Computing needs time. Technical Report UCB/EECS-2009-30, EECS Department, University of California, Berkeley, February 18 2009. To appear in *Communications of the ACM*, May, 2009.

[5] John A. Stankovic, Insup Lee, Aloysius Mok, and Raj Rajkumar. Opportunities and obligations for physical computing systems. *Computer*, pages 23–31, 2005.