Leveraging Synchronized Clocks in Distributed Applications

Edward A. Lee

Robert S. Pepper Distinguished Professor
UC Berkeley

Swarm Lab Retreat

January 14, 2014.
Berkeley, CA

A Major Emerging Opportunity: Clock Synchronization

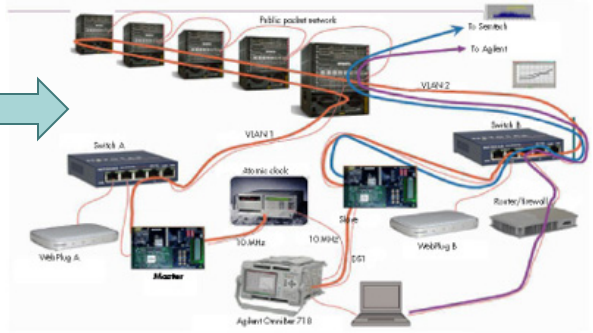
Clock synchronization is going to change the world (again)

Gregorian Calendar (BBC history)

1500s
days

Musée d'Orsay clock (Wikimedia Commons)

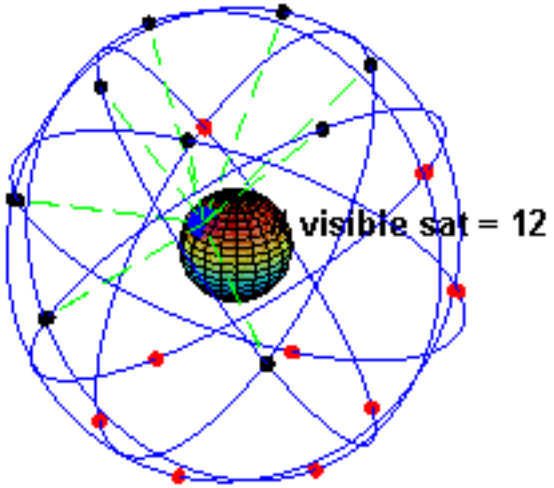
1800s
seconds



2005: first IEEE 1588 plugfest

2000s
nanoseconds

Global Positioning System



Provides ~100ns accuracy to devices with outdoor access.

Images: Wikimedia Commons

Precision Time Protocols (PTP) IEEE 1588 on Ethernet

Press Release October 1, 2007

National Semiconductor
The Sight & Sound of Information

NEWS RELEASE

For More Information Contact

Media Contact
Naomi Mitchell
National Semiconductor
(408) 721-2142
naomi.mitchell@nsc.com

Reader Information
Design Support Group
(800) 272-9959
www.national.com

Industry's First Ethernet Transceiver with IEEE 1588 PTP Hardware Support from National Semiconductor Delivers Outstanding Clock Accuracy

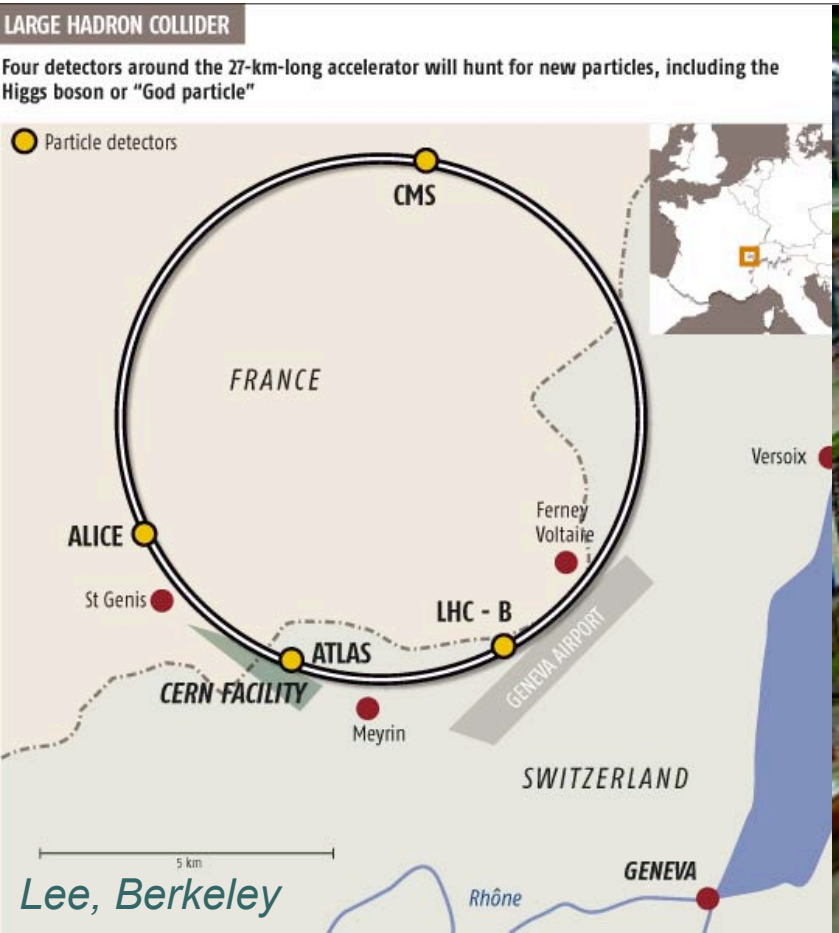
Using DP83640, Designers May Choose Any Microcontroller, FPGA or ASIC to Achieve 8- Nanosecond Precision with Maximum System Flexibility

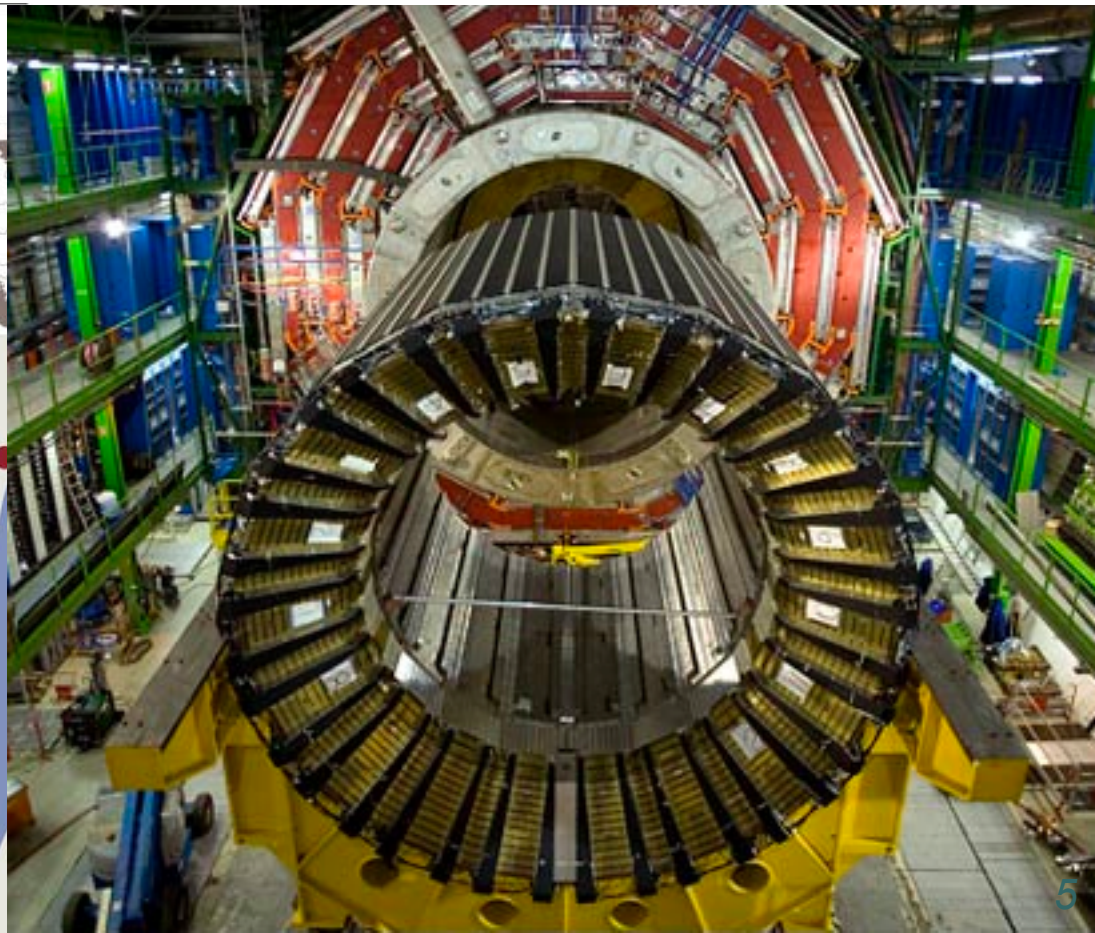
It is becoming routine for physical network interfaces (PHY) to provide hardware support for PTPs.

With this first generation PHY, clocks on a LAN agree on the current time of day to within 8ns, far more precise than GPS older techniques like NTP.

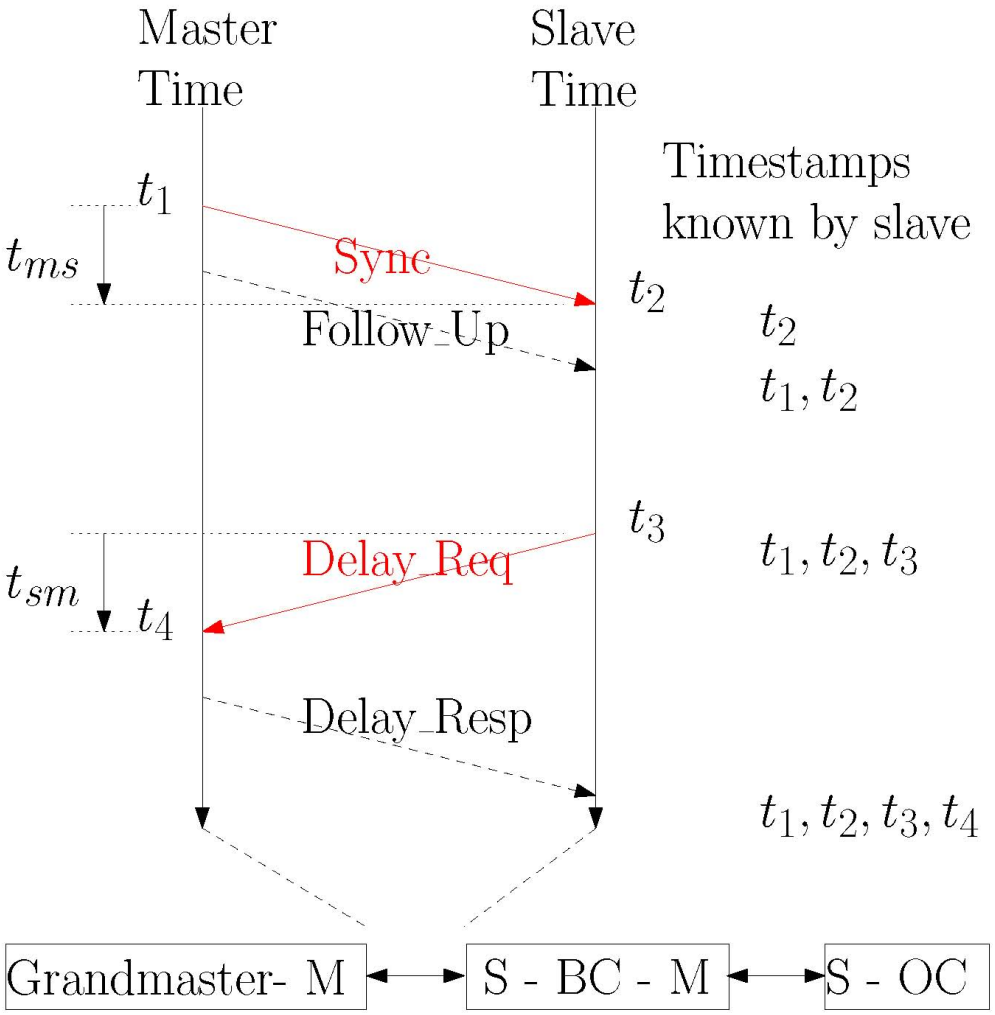
An Extreme Example: The Large Hadron Collider

The WhiteRabbit project at CERN is synchronizing the clocks of computers 10 km apart to within about 80 psec using a combination of GPS, IEEE 1588 PTP and synchronous ethernet.





How PTP Synchronization works



If link is symmetric:

$$Offset = t_{slave} - t_{master} = [(t_2 - t_1) - (t_4 - t_3)]/2 = [t_{ms} - t_{sm}]/2$$

$$Propagation\ time = [(t_2 - t_1) + (t_4 - t_3)]/2 = [t_{ms} + t_{sm}]/2$$

Source: John Eidson

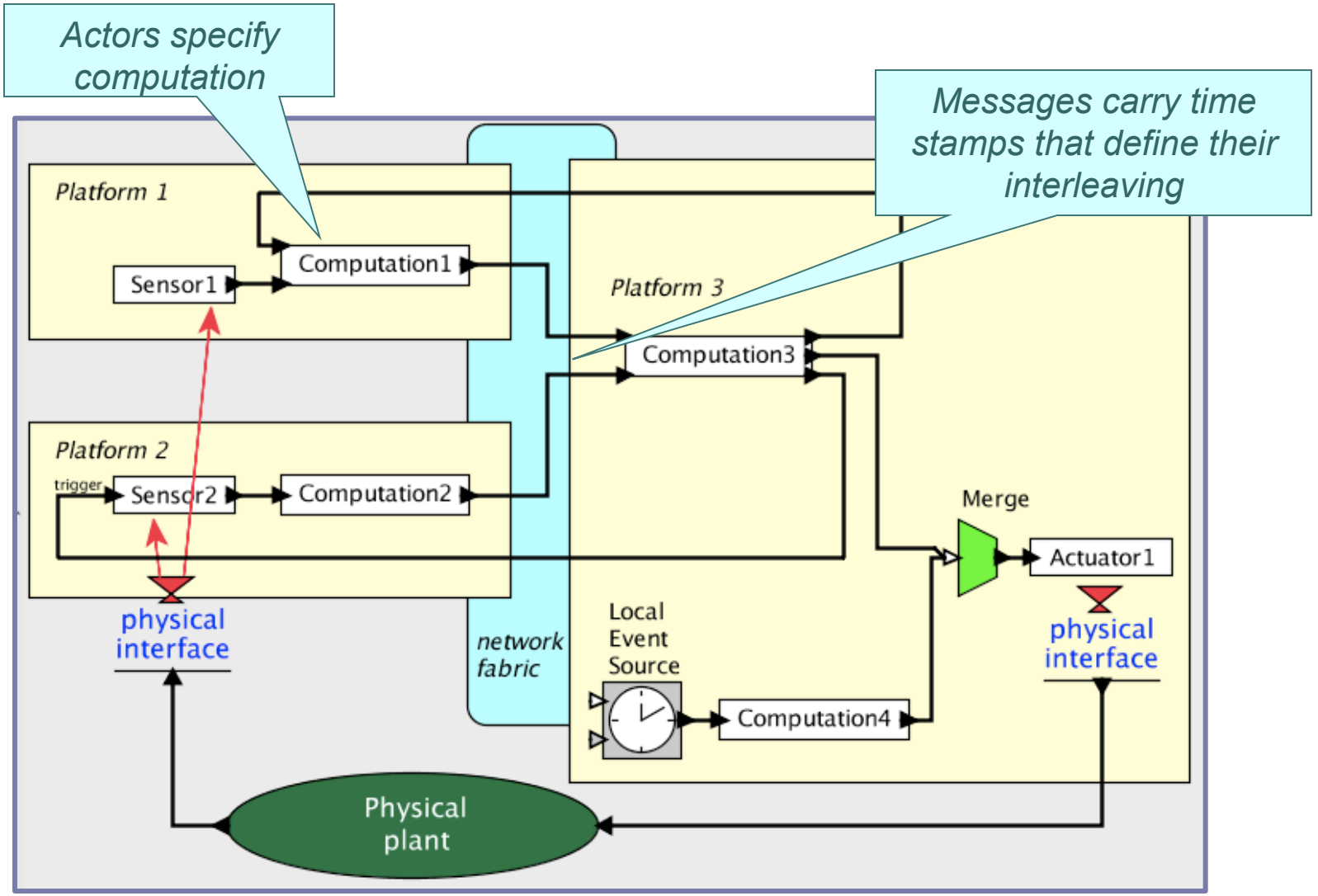
Clock Synchronization Enables:

- Energy efficiency
- Coordination, even without communication
- Security
- Resource management
- Determinism

… but I will skip this story in the interest of time…

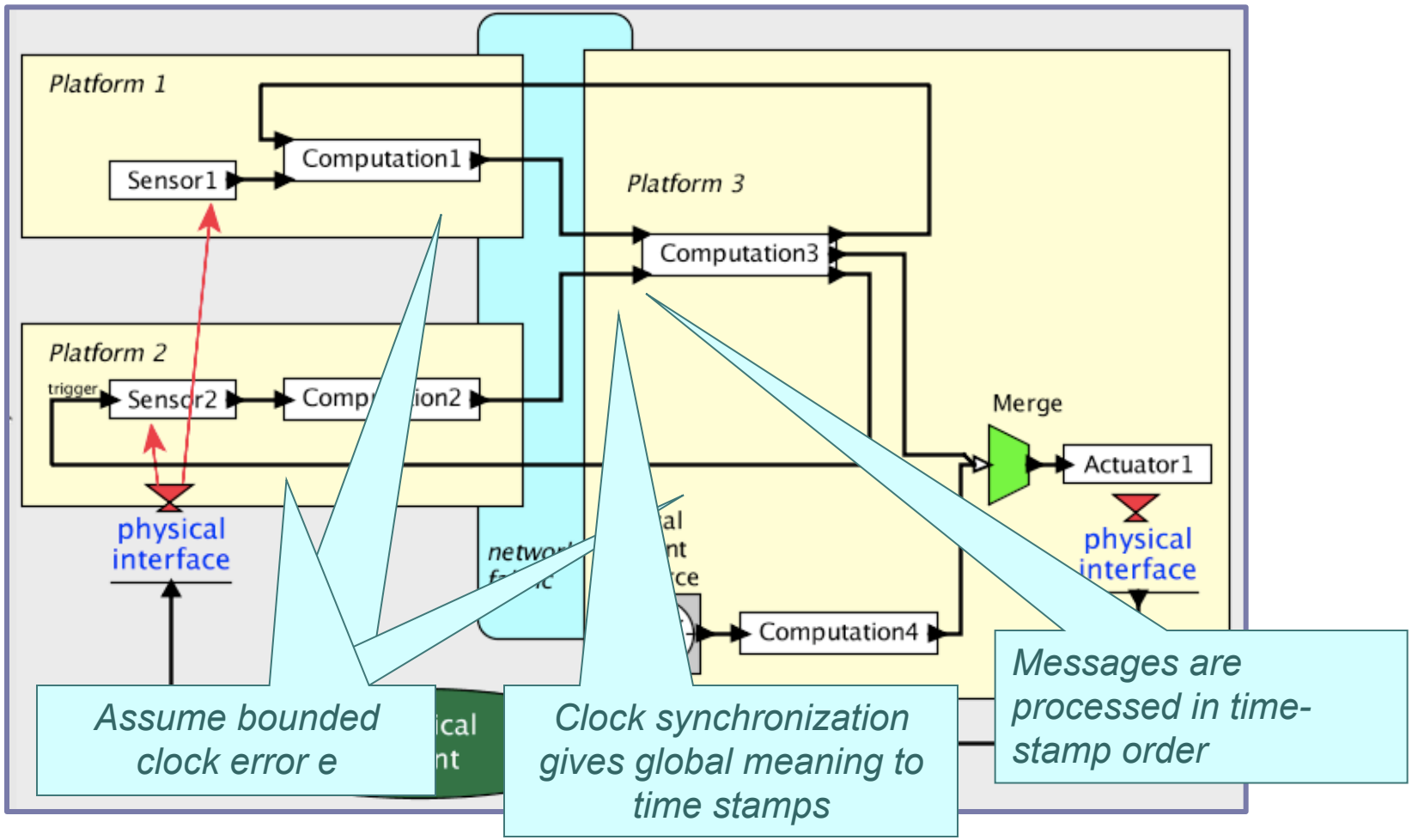
Ptides: Programming Temporally Integrated Distributed Embedded Systems

First step: Time-stamped messages.

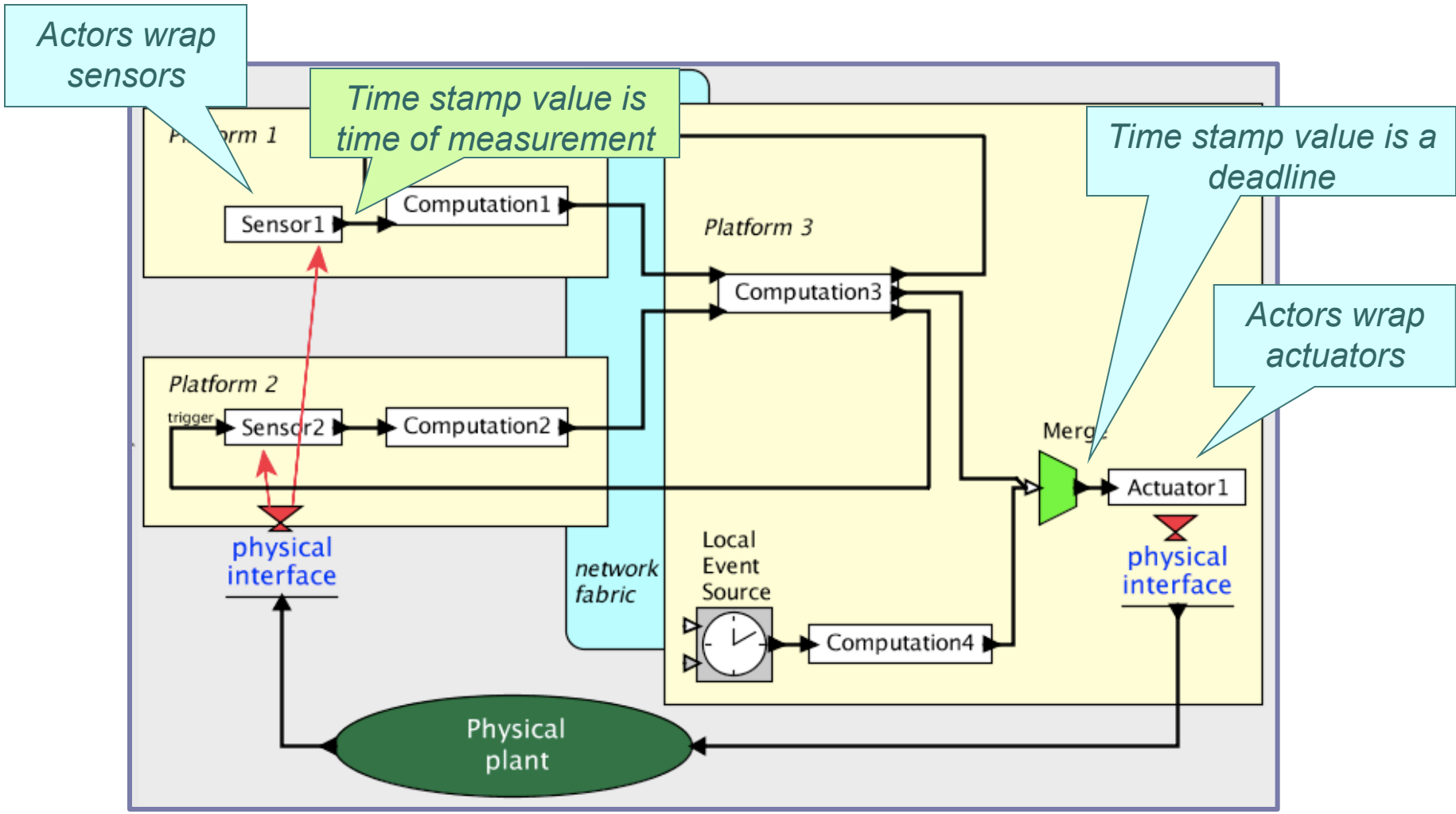


Ptides: Second step: Network clock synchronization

GPS, NTP, IEEE 1588, OpenWSN, time-triggered busses, … they all work. We just need to bound the clock synchronization error.

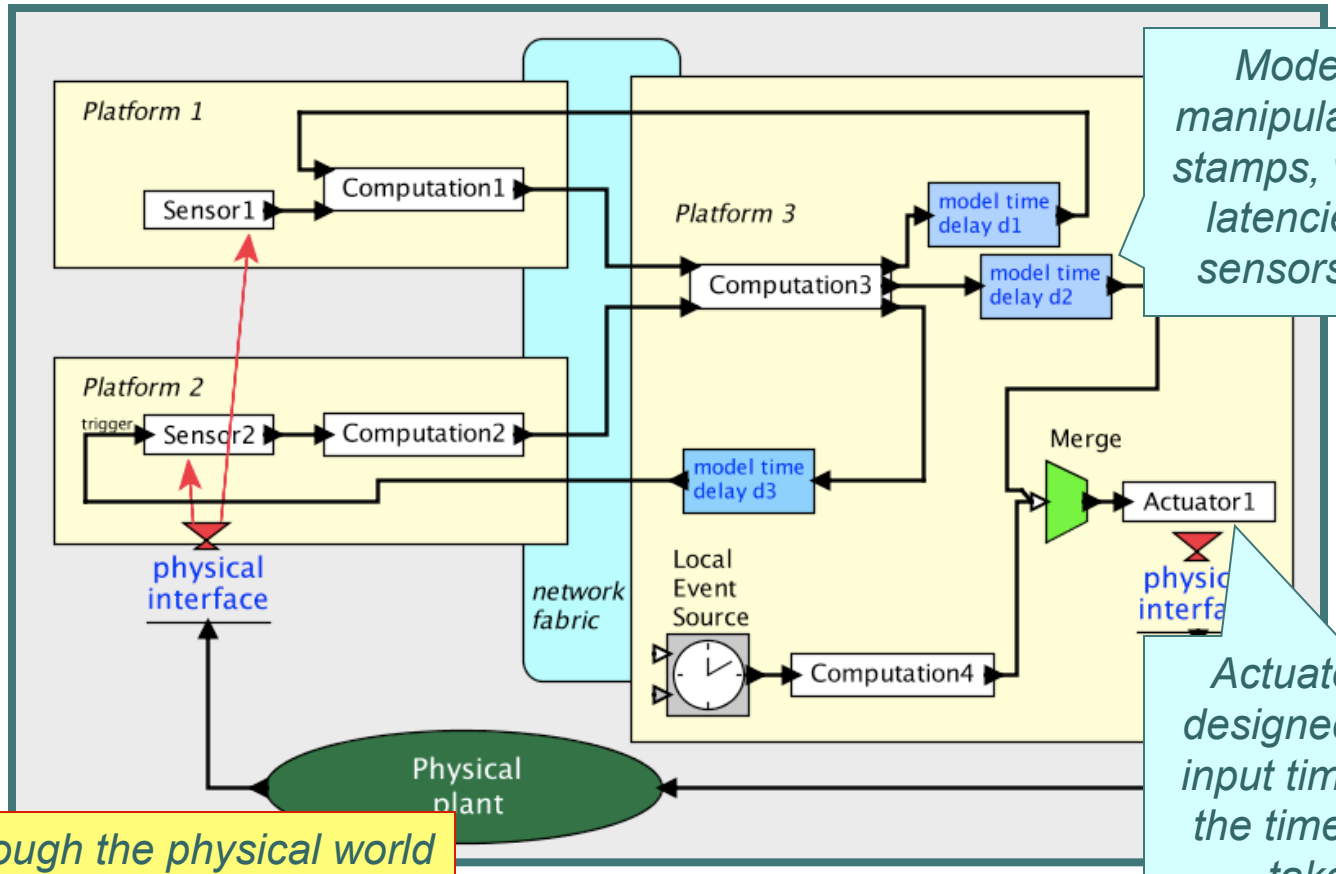


Ptides: Third step:
Bind time stamps to real time at sensors and actuators



Ptides: Fourth step: Specify latencies in the model

Global latencies between sensors and actuators become controllable, which enables analysis of system dynamics.



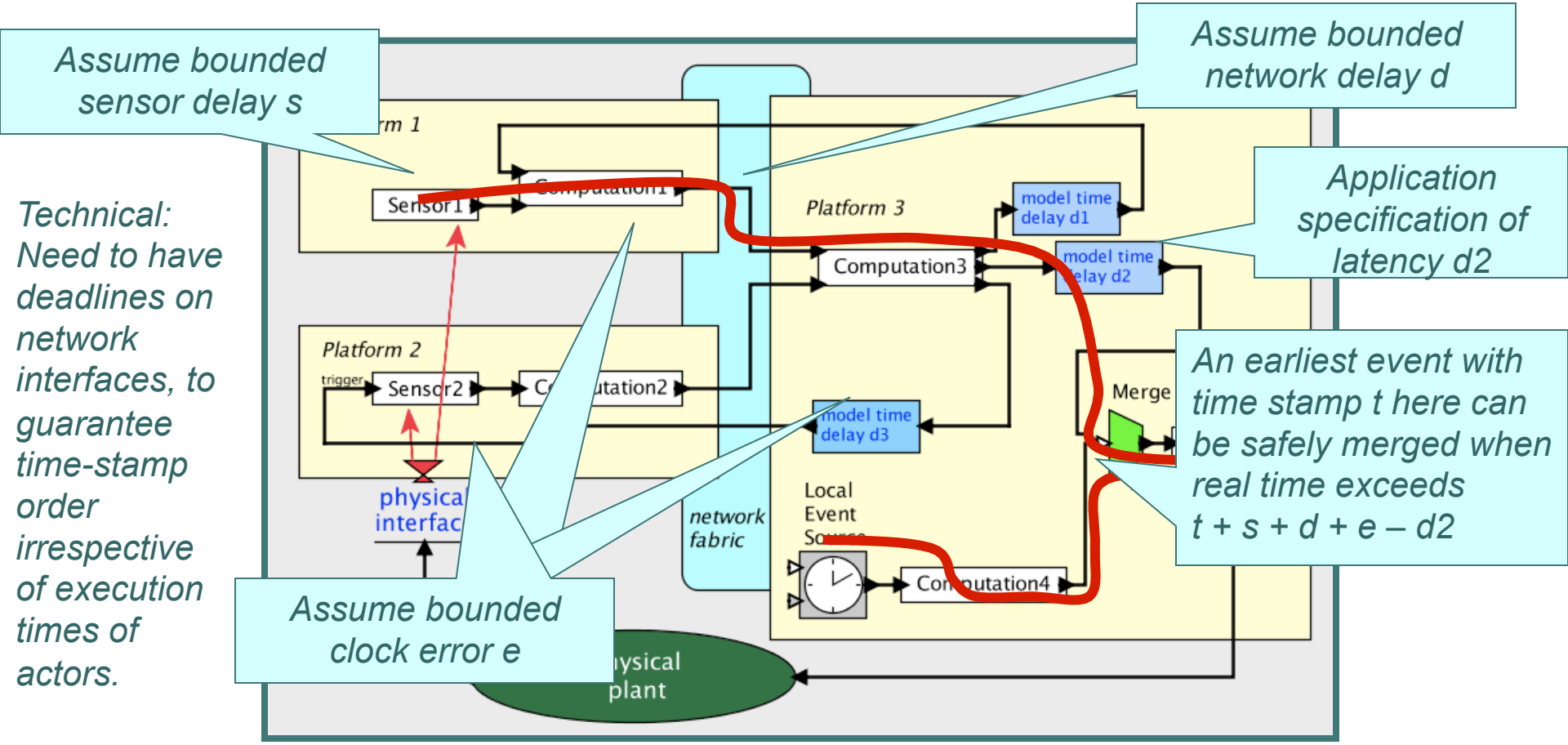
Model includes manipulations of time stamps, which control latencies between sensors and actors

Actuators may be designed to interpret input time stamps as the time at which to take action.

Feedback through the physical world

Ptides: Fifth step
Safe-to-process analysis (ensures determinacy)

Safe-to-process analysis guarantees that events are processed in time-stamp order, given some assumptions.



So Many Assumptions?

*All of the assumptions are achievable with today's technology, and in fact are **requirements** anyway for hard-real-time systems.*

The Ptides model makes the assumptions explicit.

*Violations of the assumptions are detectable as out-of-order events and can be treated as **faults**.*

Faults and Deadline Misses

Events cannot be processed according to DE semantics. Possible reactions:
- *Backtracking (transactions)*
- *Switch to degraded mode*
- *Drop events*
- *Reboot*

Faults occur if:

- Network latency exceeds expectations
- Clock synchronization error exceeds bound
- Sensor latency exceeds bound

Faults manifest as out-of-order time stamps.

Events have been processed according to DE semantics. Reactions:
- *Warning*
- *Degraded mode*
- *Drop action*

Deadline misses occur if:

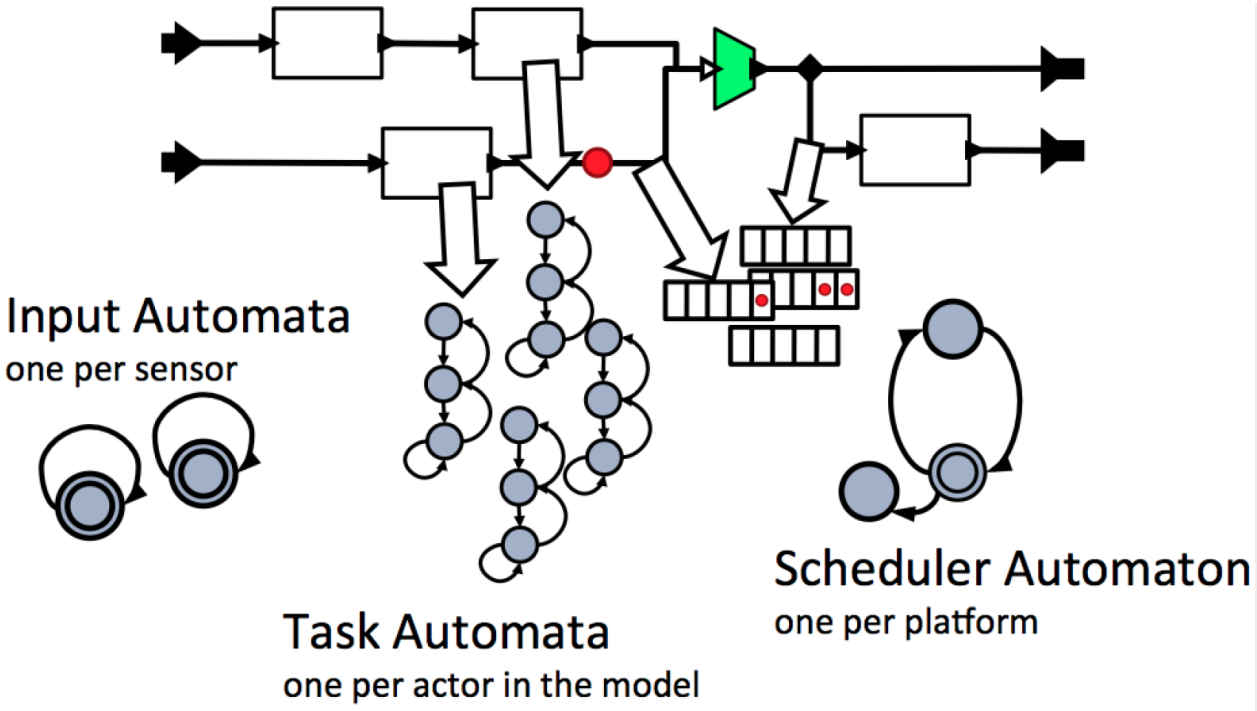
- Execution time exceeds expectations

Deadline misses are detected at actuators and network interfaces. ***They are not necessarily faults****!*

Ptides Schedulability Analysis

Determine *whether* deadlines can be met

The problem turns out to be decidable for a large class of models.



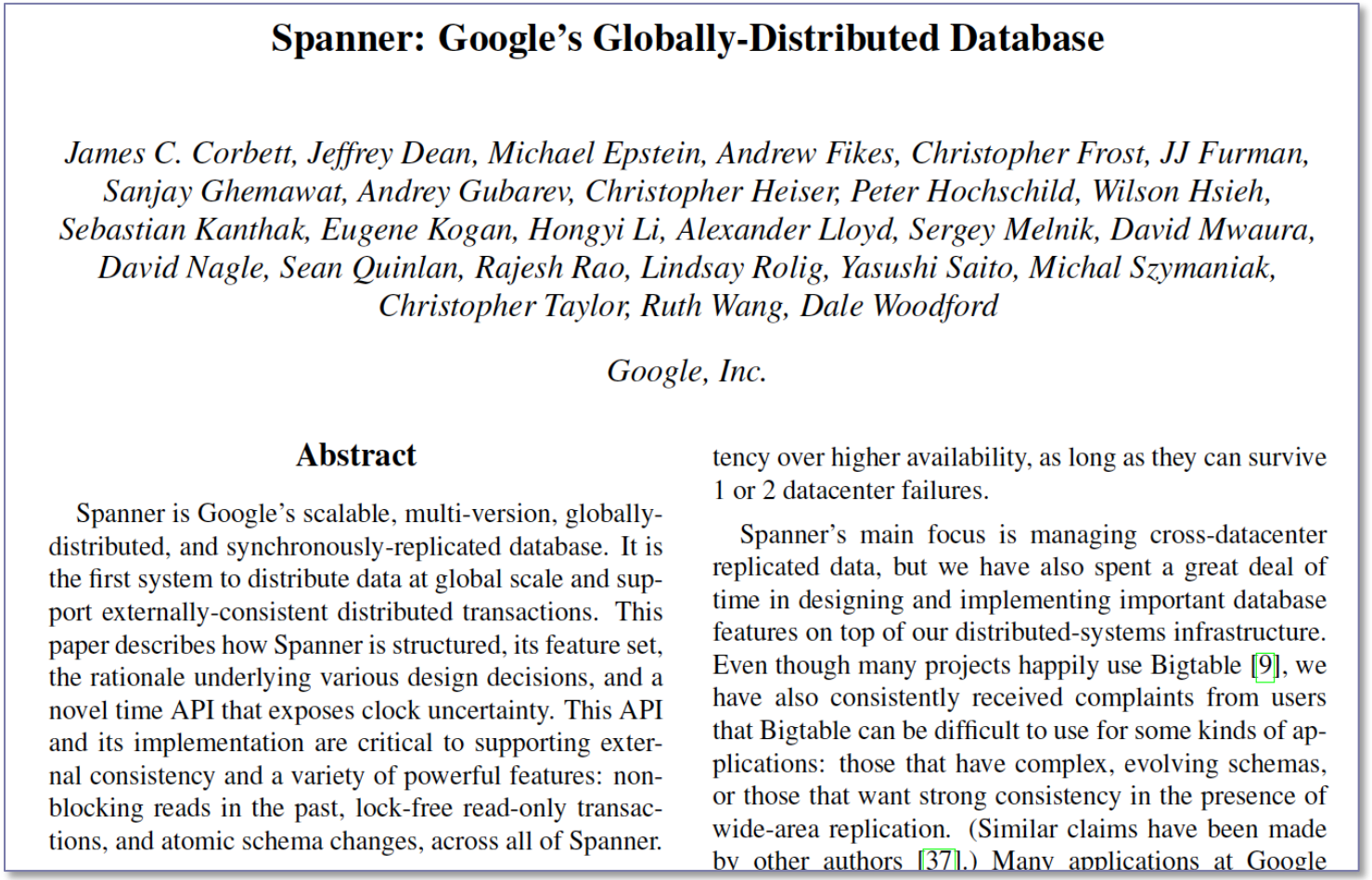
On the Schedulability of Real-Time Discrete-Event Systems*

Eleftherios Matsikoudis Christos Stergiou Edward A. Lee

EMSOFT 2013

Google Spanner

Google independently developed a very similar technique and applied it to distributed databases.



Spanner: Google's Globally-Distributed Database

James C. Corbett, Jeffrey Dean, Michael Epstein, Andrew Fikes, Christopher Frost, JJ Furman, Sanjay Ghemawat, Andrey Gubarev, Christopher Heiser, Peter Hochschild, Wilson Hsieh, Sebastian Kanthak, Eugene Kogan, Hongyi Li, Alexander Lloyd, Sergey Melnik, David Mwaura, David Nagle, Sean Quinlan, Rajesh Rao, Lindsay Rolig, Yasushi Saito, Michal Szymaniak, Christopher Taylor, Ruth Wang, Dale Woodford

Google, Inc.

Abstract

Spanner is Google's scalable, multi-version, globally-distributed, and synchronously-replicated database. It is the first system to distribute data at global scale and support externally-consistent distributed transactions. This paper describes how Spanner is structured, its feature set, the rationale underlying various design decisions, and a novel time API that exposes clock uncertainty. This API and its implementation are critical to supporting external consistency and a variety of powerful features: non-blocking reads in the past, lock-free read-only transactions, and atomic schema changes, across all of Spanner.

tency over higher availability, as long as they can survive 1 or 2 datacenter failures.

Spanner's main focus is managing cross-datacenter replicated data, but we have also spent a great deal of time in designing and implementing important database features on top of our distributed-systems infrastructure. Even though many projects happily use Bigtable [9], we have also consistently received complaints from users that Bigtable can be difficult to use for some kinds of applications: those that have complex, evolving schemas, or those that want strong consistency in the presence of wide-area replication. (Similar claims have been made by other authors [37].) Many applications at Google

Proceedings of OSDI 2012

Ptides is a Change in Philosophy

The implementation platform affects timing in a distributed real-time system.

Conventional approach: Specify functionality, implementation architecture, and mapping. Timing emerges from the combination.

Ptides approach: Specify temporal behavior. Then verify that it is met by a candidate implementation architecture.

Ptides offers a *deterministic*
model of computation
for distributed real-time systems.

What is the Value of Models?

You will never strike oil by drilling through the map!

Solomon Wolf Golomb on Modeling

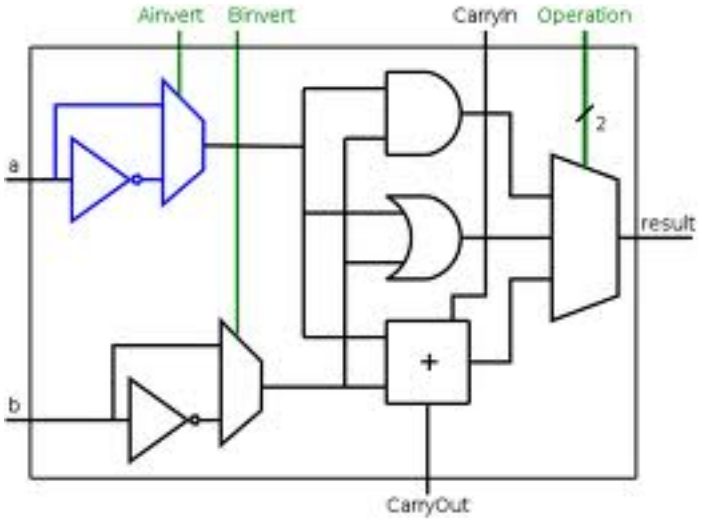
But this does not, in any way, diminish the value of a map!

Determinate Models

Physical System

Image: Wikimedia Commons

Model



Synchronous digital logic

Determinate Models

Physical System

Image: Wikimedia Commons

Model

```
/** Reset the output receivers, which are the inside receivers of
 *  the output ports of the container.
 *  @exception IllegalActionException If getting the receivers fails.
 */
private void _resetOutputReceivers() throws IllegalActionException {
    List<IOPort> outputs = ((Actor) getContainer()).outputPortList();
    for (IOPort output : outputs) {
        if (_debugging) {
            _debug("Resetting inside receivers of output port: "
                    + output.getName());
        }
        Receiver[][] receivers = output.getInsideReceivers();
        if (receivers != null) {
            for (int i = 0; i < receivers.length; i++) {
                if (receivers[i] != null) {
                    for (int j = 0; j < receivers[i].length; j++) {
                        if (receivers[i][j] instanceof FSMReceiver) {
                            receivers[i][j].reset();
                        }
                    }
                }
            }
        }
    }
}
```

Single-threaded imperative programs

Determinate Models

Physical System

Image: Wikimedia Commons

Model

```
module Timer:
input R, SEC;
output L, S;
Loop
  weak abort
    await 3 SEC;
    [
      sustain S
    ||
      await 5 SEC;
      sustain L
    ]
   when R;
  end
end module
```

[S. Edwards, Columbia U.]

Synchronous language programs

Determinate Models

Physical System

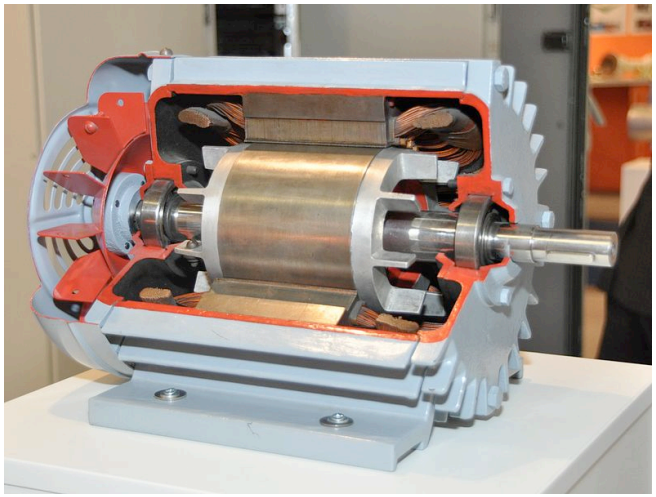


Image: Wikimedia Commons

Model

$$\dot{\mathbf{x}}(t) = \dot{\mathbf{x}}(0) + \frac{1}{M}\int_0^t \mathbf{F}(\tau)d\tau$$

Differential Equations

A Major Problem Today: Cyber-Physical Combinations are Nondeterminate

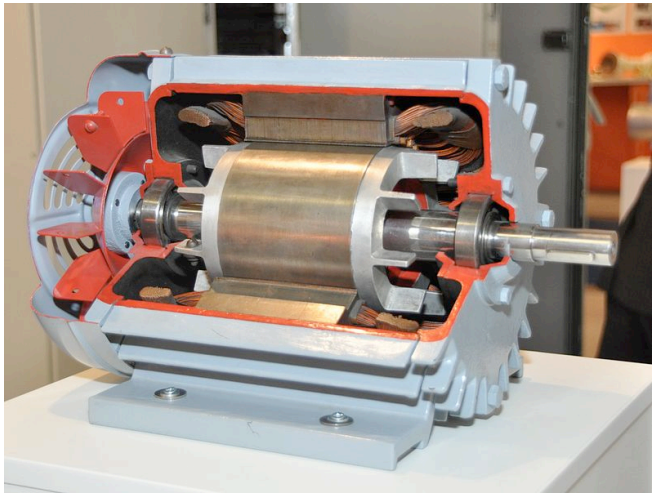


Image: Wikimedia Commons

```
/** Reset the output receivers, which are the inside receivers of
 *  the output ports of the container.
 *  @exception IllegalActionException If getting the receivers fails.
 */
private void _resetOutputReceivers() throws IllegalActionException {
    List<IOPort> outputs = ((Actor) getContainer()).outputPortList();
    for (IOPort output : outputs) {
        if (_debugging) {
            _debug("Resetting inside receivers of output port: "
                    + output.getName());
        }
        Receiver[][] receivers = output.getInsideReceivers();
        if (receivers != null) {
            for (int i = 0; i < receivers.length; i++) {
                if (receivers[i] != null) {
                    for (int j = 0; j < receivers[i].length; j++) {
                        if (receivers[i][j] instanceof FSMReceiver) {
                            receivers[i][j].reset();
                        }
                    }
                }
            }
        }
    }
}
```

$$\dot{\mathbf{x}}(t) = \dot{\mathbf{x}}(0) + \frac{1}{M}\int_0^t \mathbf{F}(\tau)d\tau$$

Ptides offers a *deterministic*
model of computation
for distributed real-time systems.

http://chess.eecs.berkeley.edu/ptides

Acknowledgements

- Patricia Derler
- John Eidson
- Slobodan Matic
- Christos Stergiou
- Yang Zhao
- Jia Zou

Plus: The entire Ptolemy II Pteam