iCyPhy

Accessors
A Software Architecture for IoT

Edward A. Lee

Robert S. Pepper Distinguished Professor

Invited Talk: Google, Mountain View, CA

SWARMLAB
UC BERKELEY

March 30, 2017

University of California at Berkeley

Cyber-Physical Systems
Focus on the Internet of *Important* Things

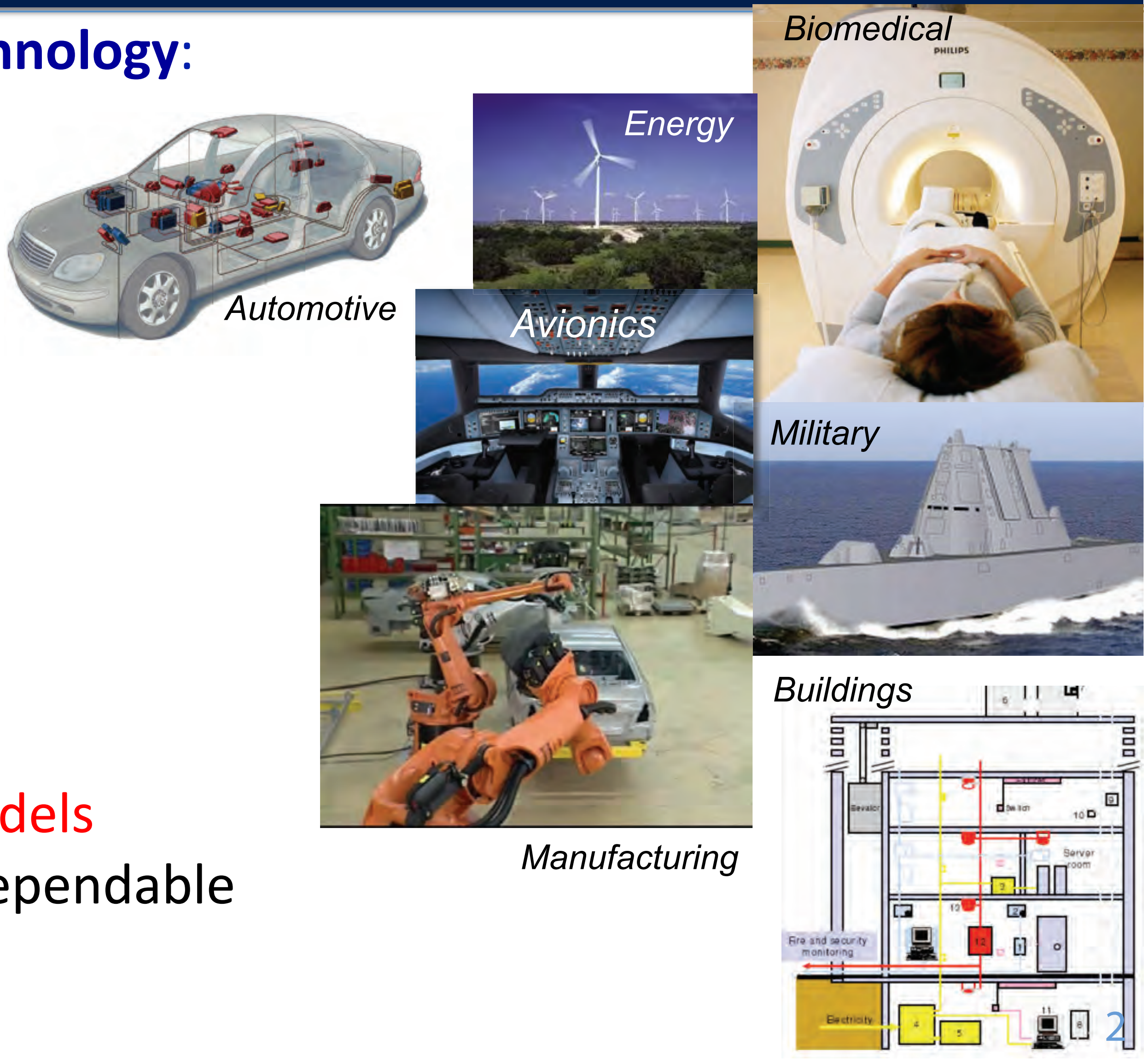
Not just information technology:

- Cyber + Physical
- Computation + Dynamics
- Security + Safety

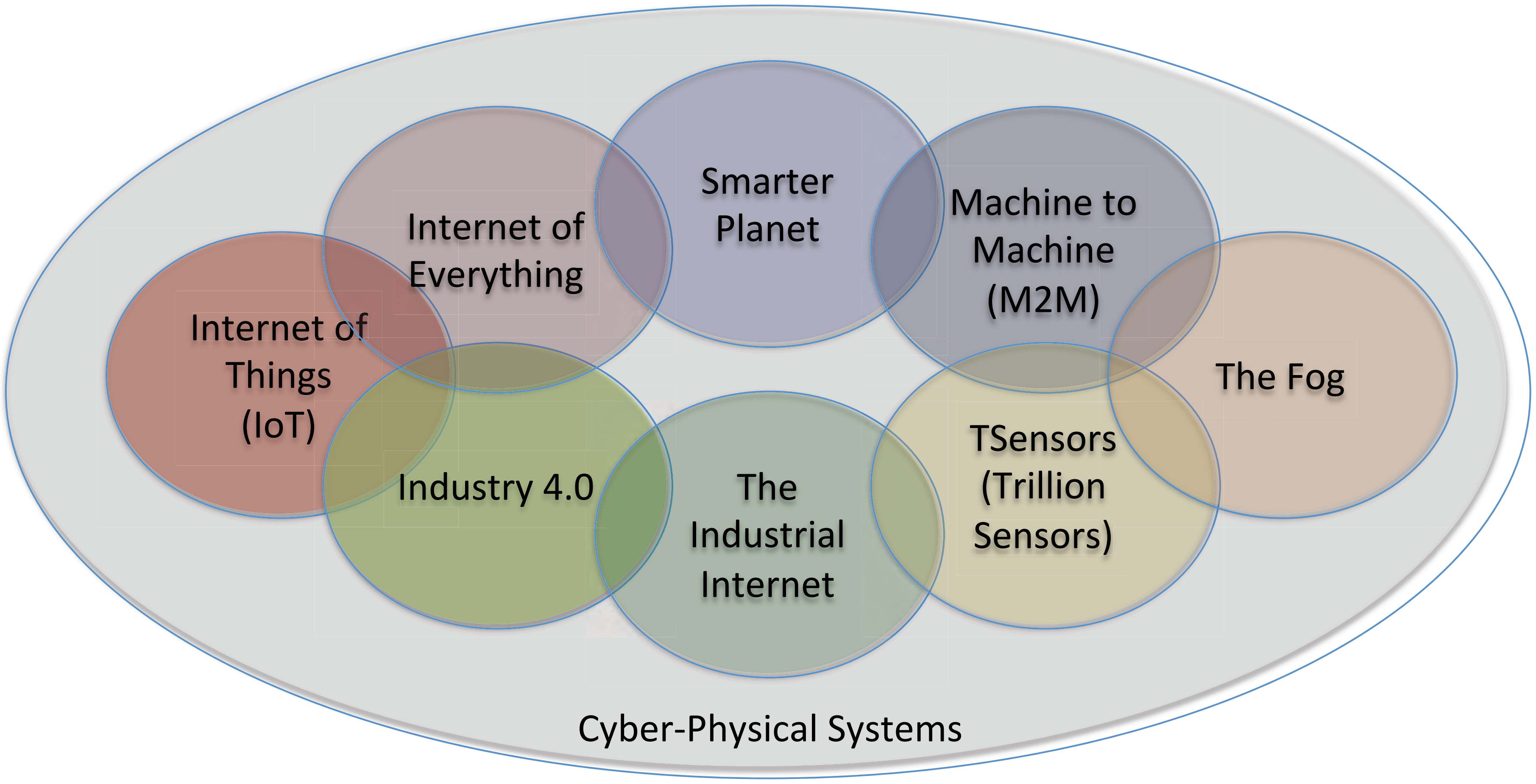
Properties:

- Highly dynamic
- Safety critical
- Uncertain environment
- Physically distributed
- Sporadic connectivity
- Resource constrained

We need engineering models and methodologies for dependable cyber-physical systems.



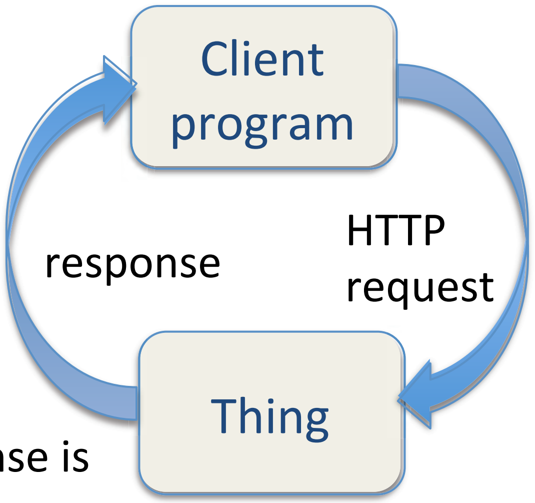
Note that this is not as new an area as some people assume…



A Common IoT Design Pattern: REST with AACs

A RESTful service [Fielding & Taylor 2002] is accessed using a design pattern common on the web that we call *Asynchronous Atomic Callbacks* (AAC) (also called the *Reactor Pattern*).

In the Web, AAC is widely used. It is central to many popular internet programming frameworks such as Node.js & Vert.x, and to CPS frameworks such as TinyOS.



Response is typically asynchronous to avoid blocking the client program.

URL encodes all state info (credentials, commands, etc.)

Response handler executes atomically.

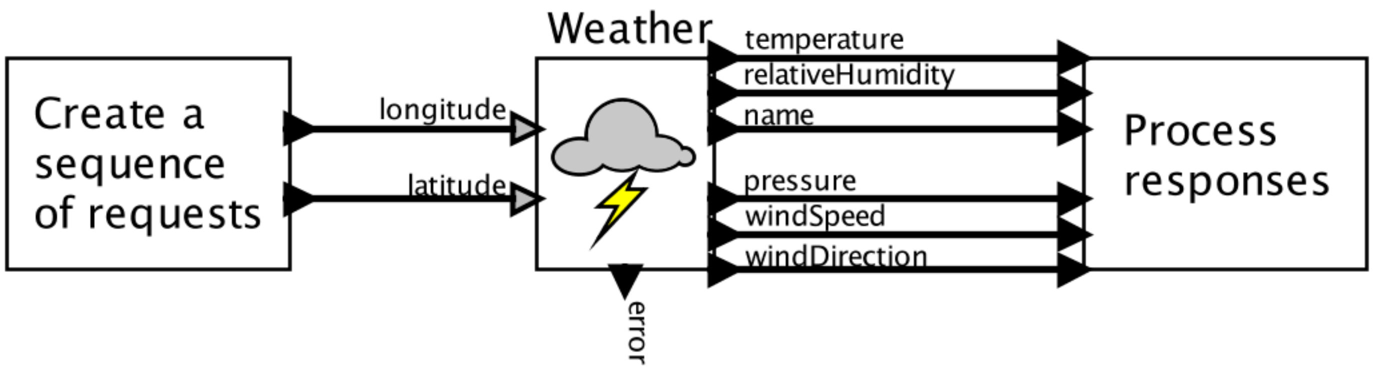
Example in JavaScript

```
// Import a module providing network services
var http = require("http”);
// Construct a URL encoding a request
var url = "http://foo.com/deviceID/...";
// Issue the request and provide a callback
http.get(url, function(response) {
    // ... handle the response ...
});
```

The callback function will be called atomically some time later when the server response arrives.

Another Common Design Pattern: *Actors*

Streaming requests:

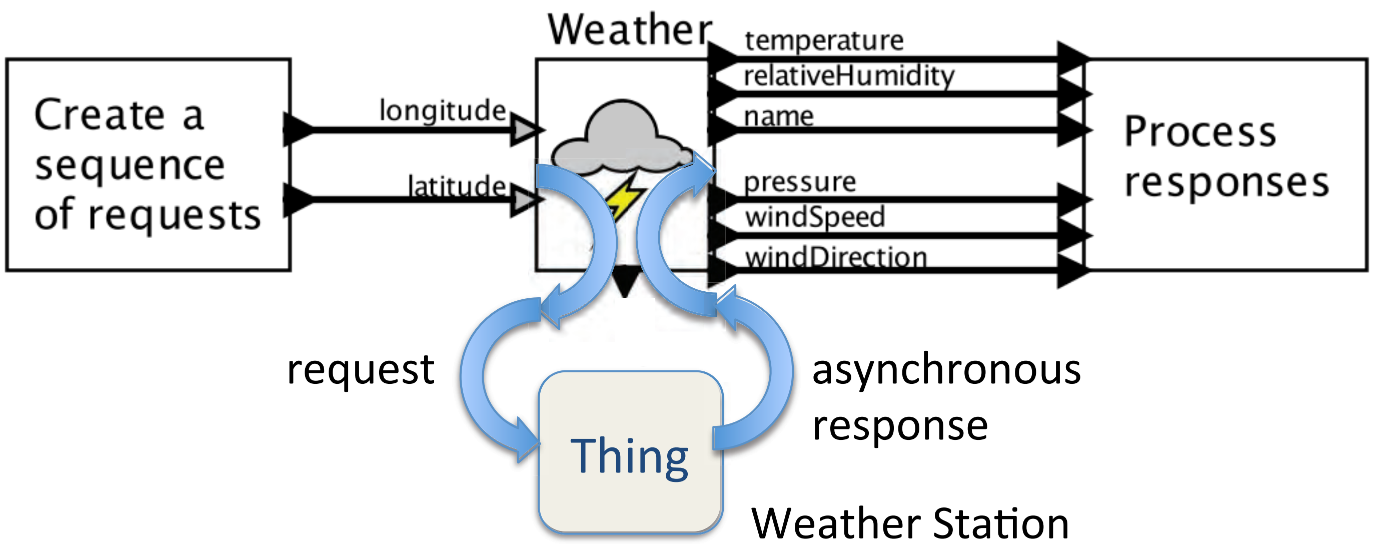


Sequence of requests for a service (a stream) triggers a sequence of responses.

Actors embrace concurrency and scale well.

Actors and AAC

Streaming requests:



This is the essence of accessors, a design pattern for IoT that embraces concurrency, asynchrony, and atomicity.

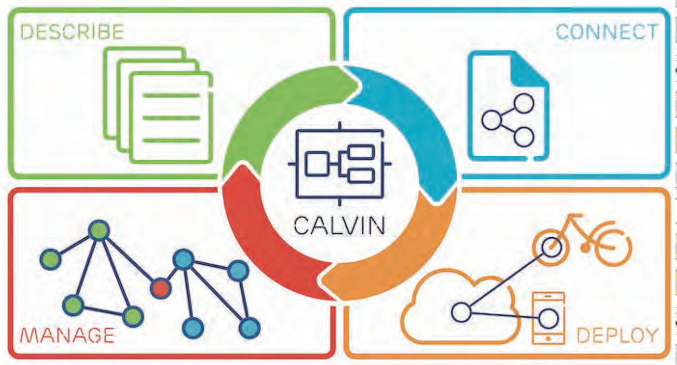
We are not alone pursuing this approach

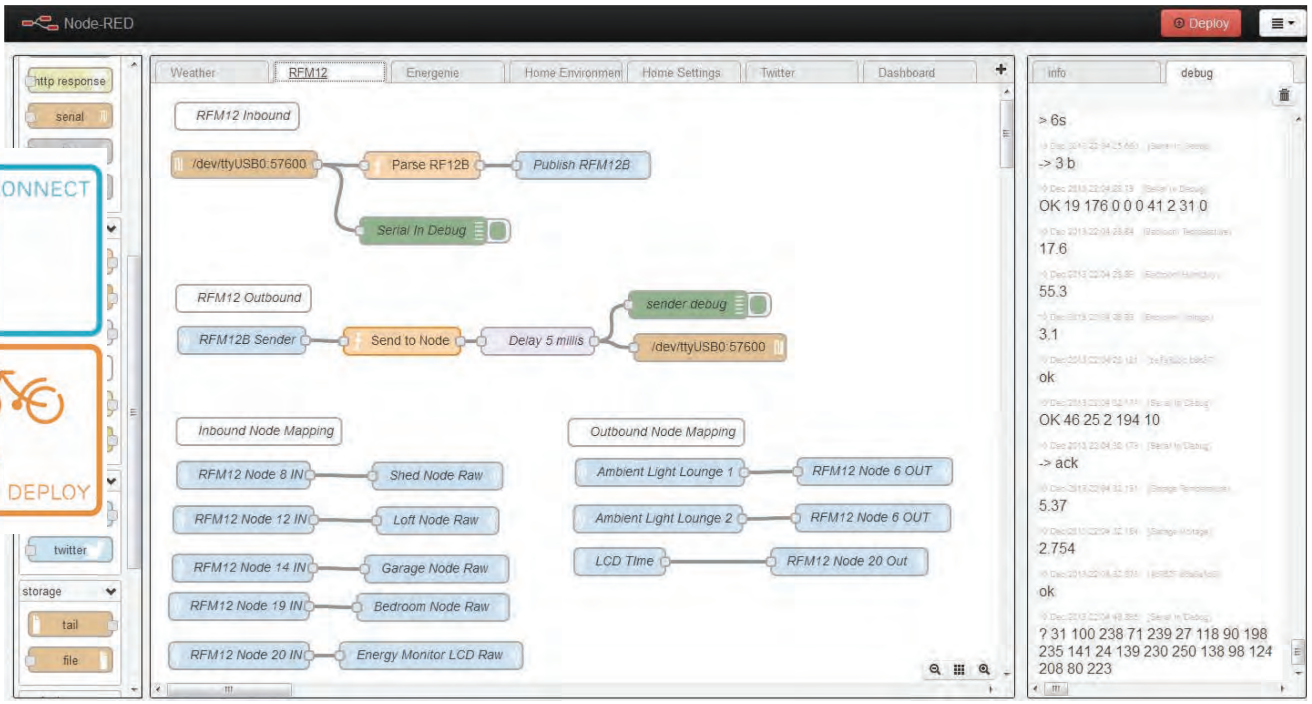
Notable efforts:

- Node Red (IBM)
- Calvin (Ericsson)

Node-RED

From: "Home Automation with Node Red, JeeNodes and Open Energy Monitor," Dom Bramley's Blog of Maximo and the 'Internet of Things', IBM Developer Works, Dec., 2013.

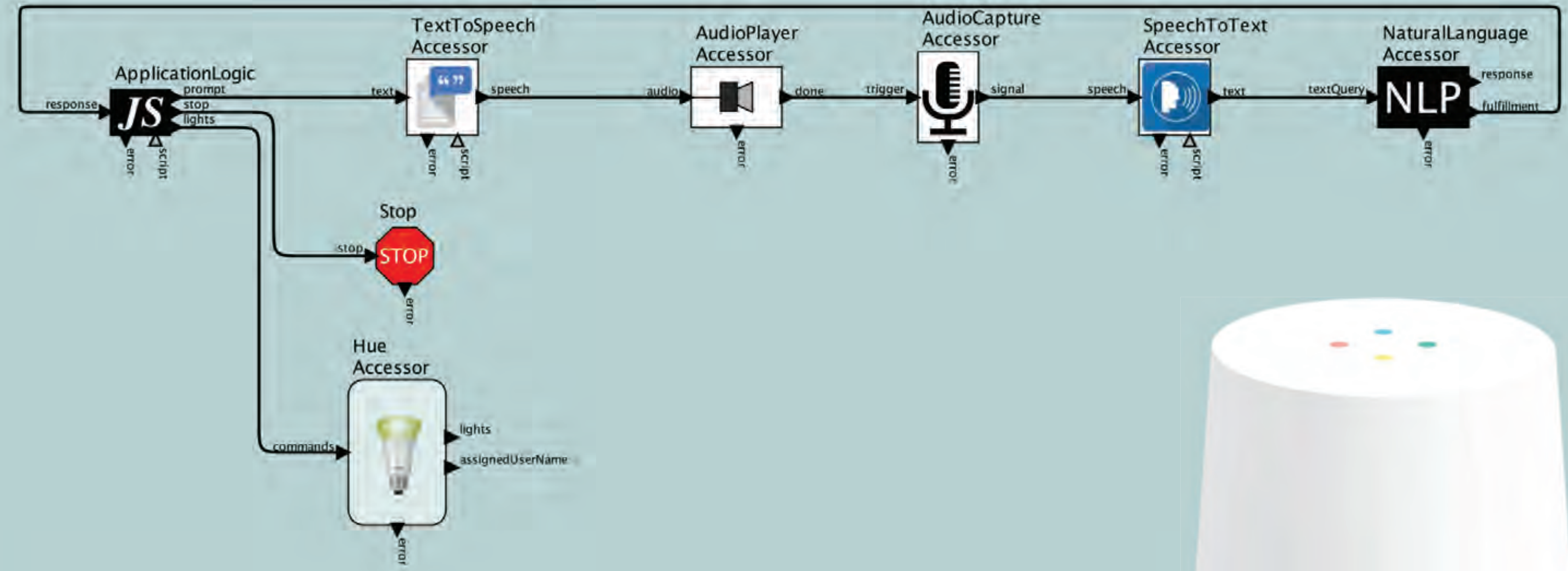




```
> 6s
-> 3 b
OK 19 176 0 0 0 41 2 31 0
17.6
55.3
3.1
ok
OK 46 25 2 194 10
-> ack
5.37
2.754
ok
? 31 100 238 71 239 27 118 90 198
235 141 24 139 230 250 138 98 124
208 80 223
```

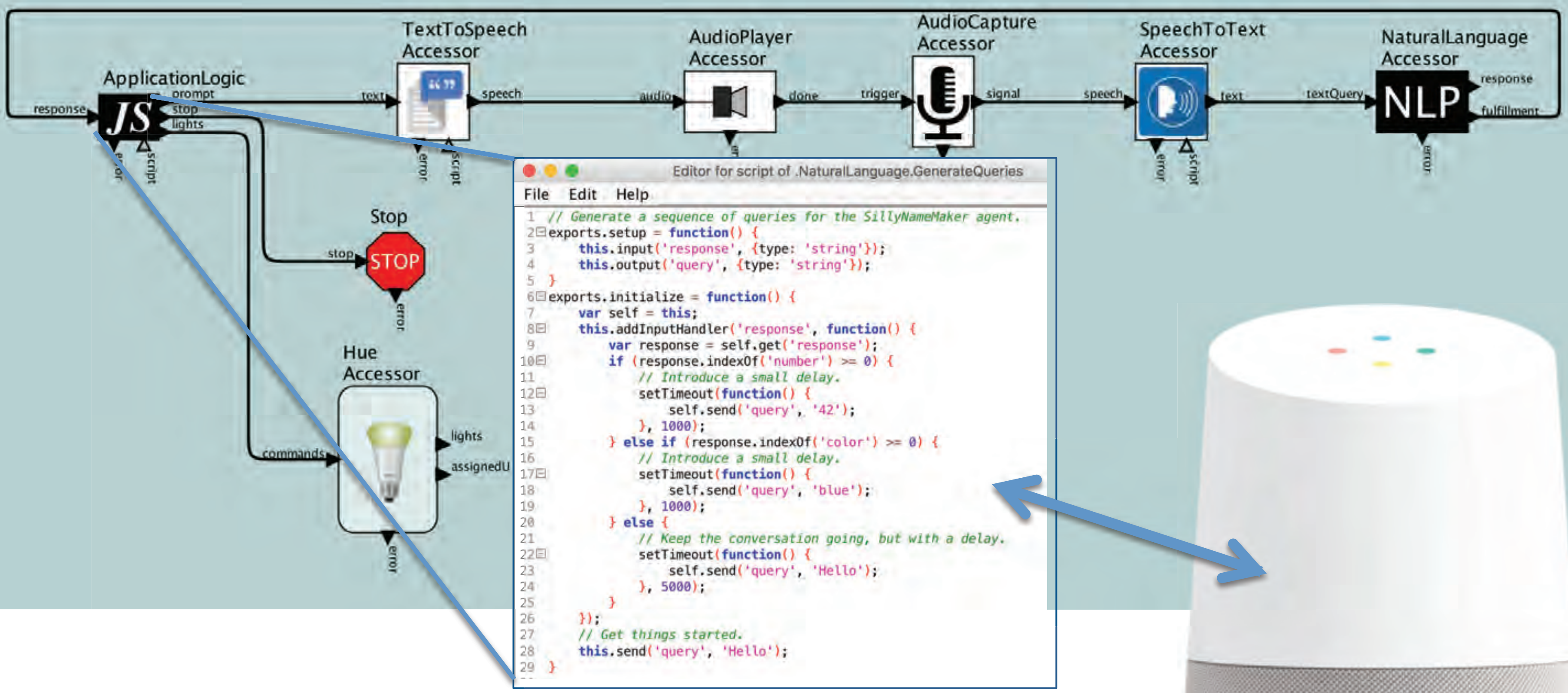
Lee, Berkeley

Swarmlet: Composition of Accessors



Consider a next generation digital assistant, an edge computer that can host swarmlets and functions as a smart gateway.

Swarmlet: Composition of Accessors



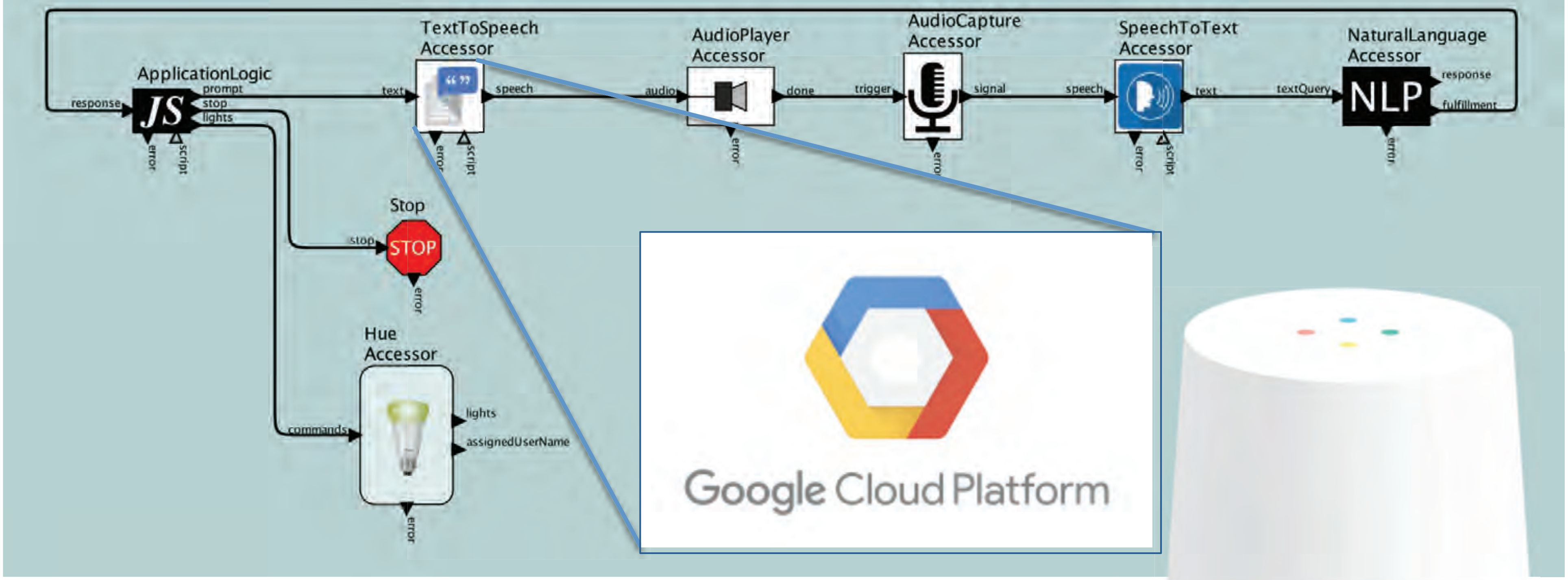
Editor for script of .NaturalLanguage.GenerateQueries

File Edit Help

```
// Generate a sequence of queries for the SillyNameMaker agent.
exports.setup = function() {
    this.input('response', {type: 'string'});
    this.output('query', {type: 'string'});
}
exports.initialize = function() {
    var self = this;
    this.addInputHandler('response', function() {
        var response = self.get('response');
        if (response.indexOf('number') >= 0) {
            // Introduce a small delay.
            setTimeout(function() {
                self.send('query', '42');
            }, 1000);
        } else if (response.indexOf('color') >= 0) {
            // Introduce a small delay.
            setTimeout(function() {
                self.send('query', 'blue');
            }, 1000);
        } else {
            // Keep the conversation going, but with a delay.
            setTimeout(function() {
                self.send('query', 'Hello');
            }, 5000);
        }
    });
    // Get things started.
    this.send('query', 'Hello');
}
```

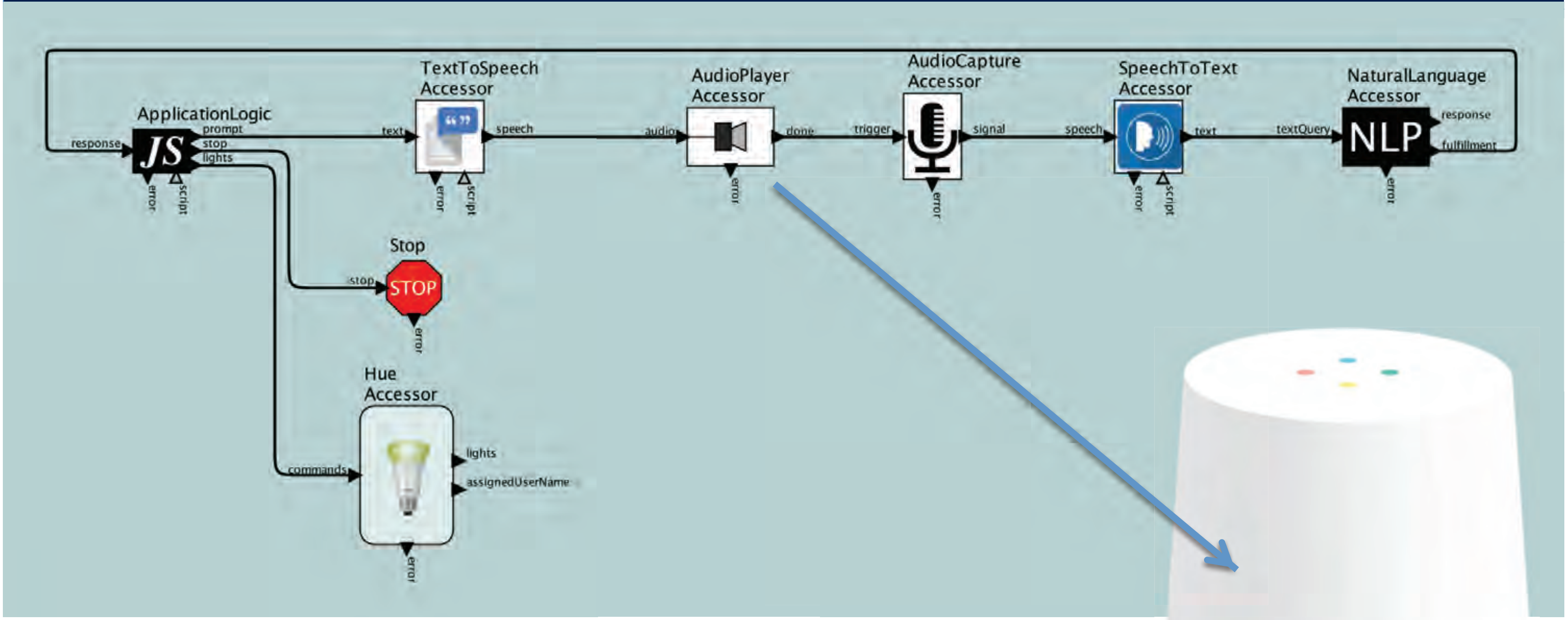
Scripted application logic that runs in a sandbox on the gateway.

Swarmlet: Composition of Accessors



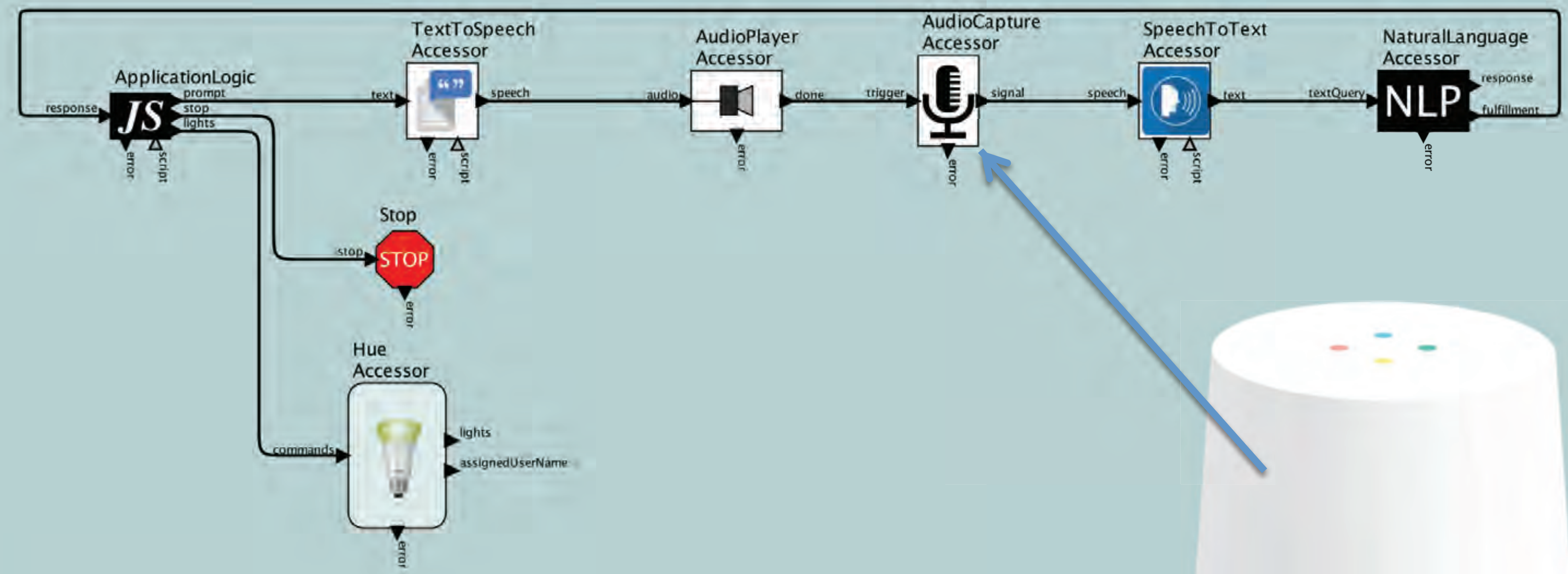
Application text converted to audio data via RESTful API.

Swarmlet: Composition of Accessors



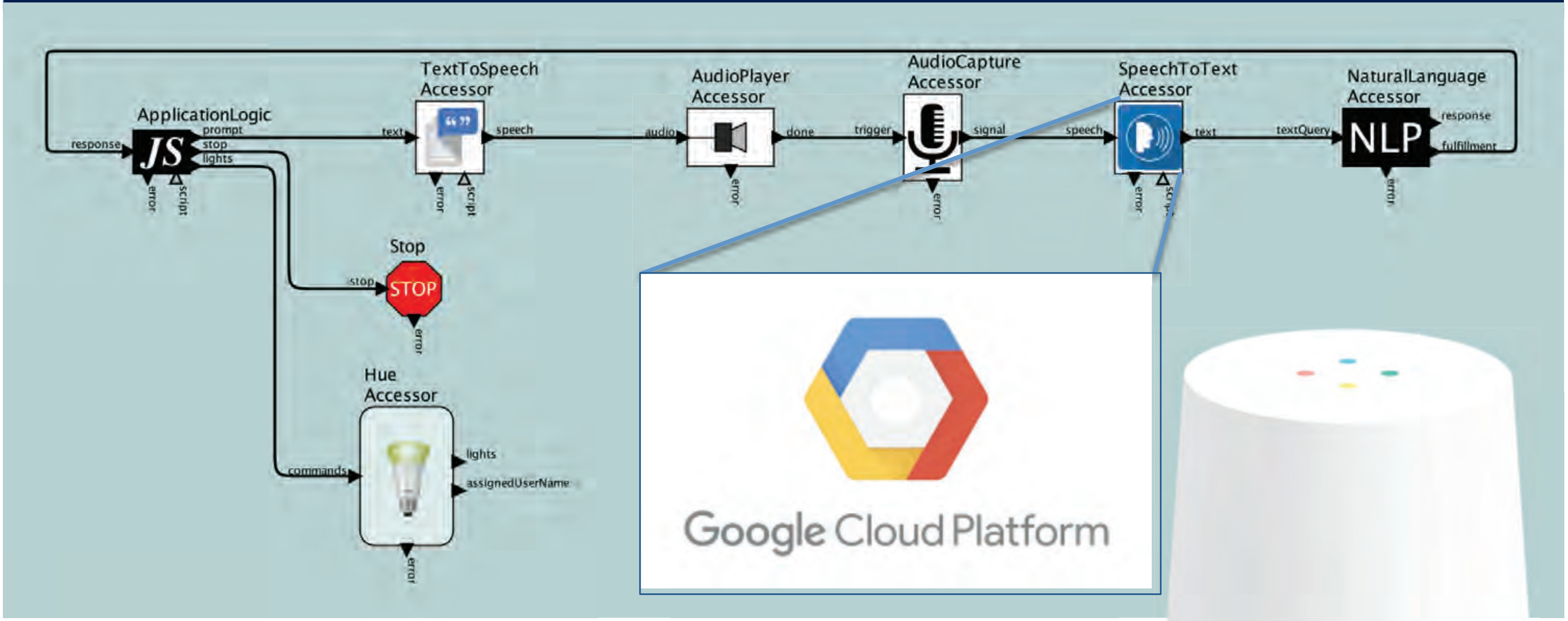
Accessor invokes audio API on the gateway to produce an audio prompt.

Swarmlet: Composition of Accessors



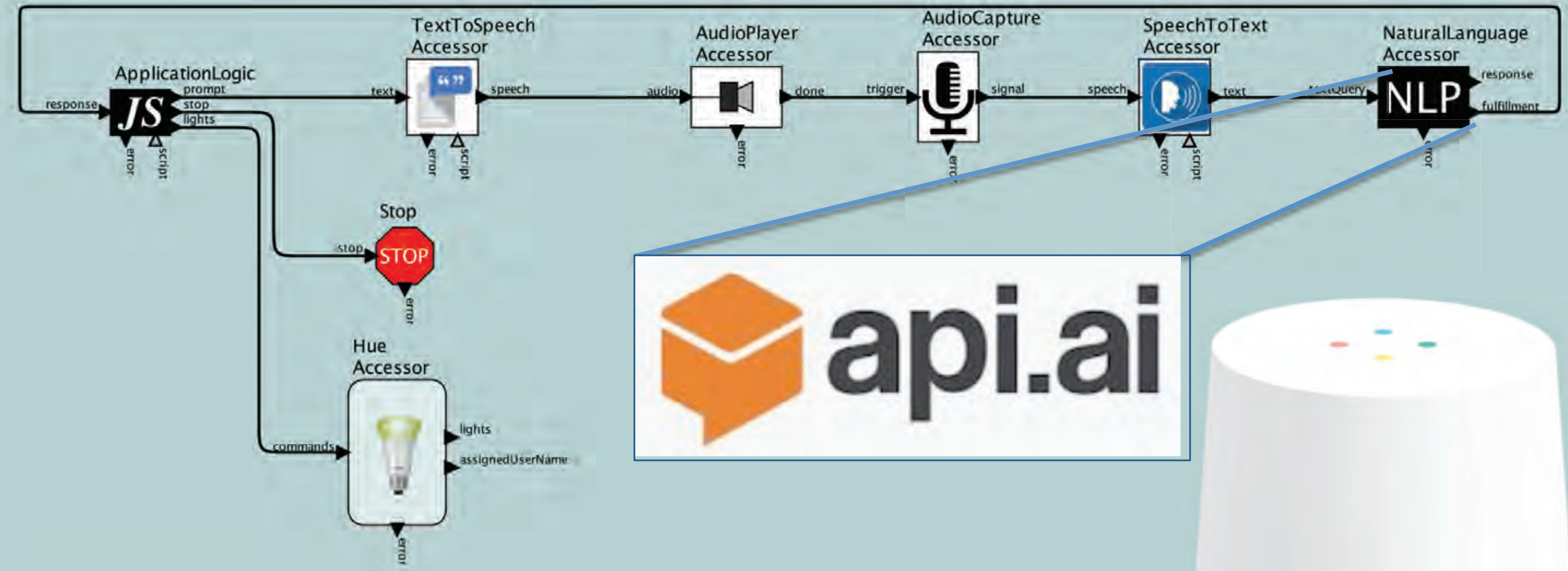
Accessor records audio until it detects a pause.

Swarmlet: Composition of Accessors



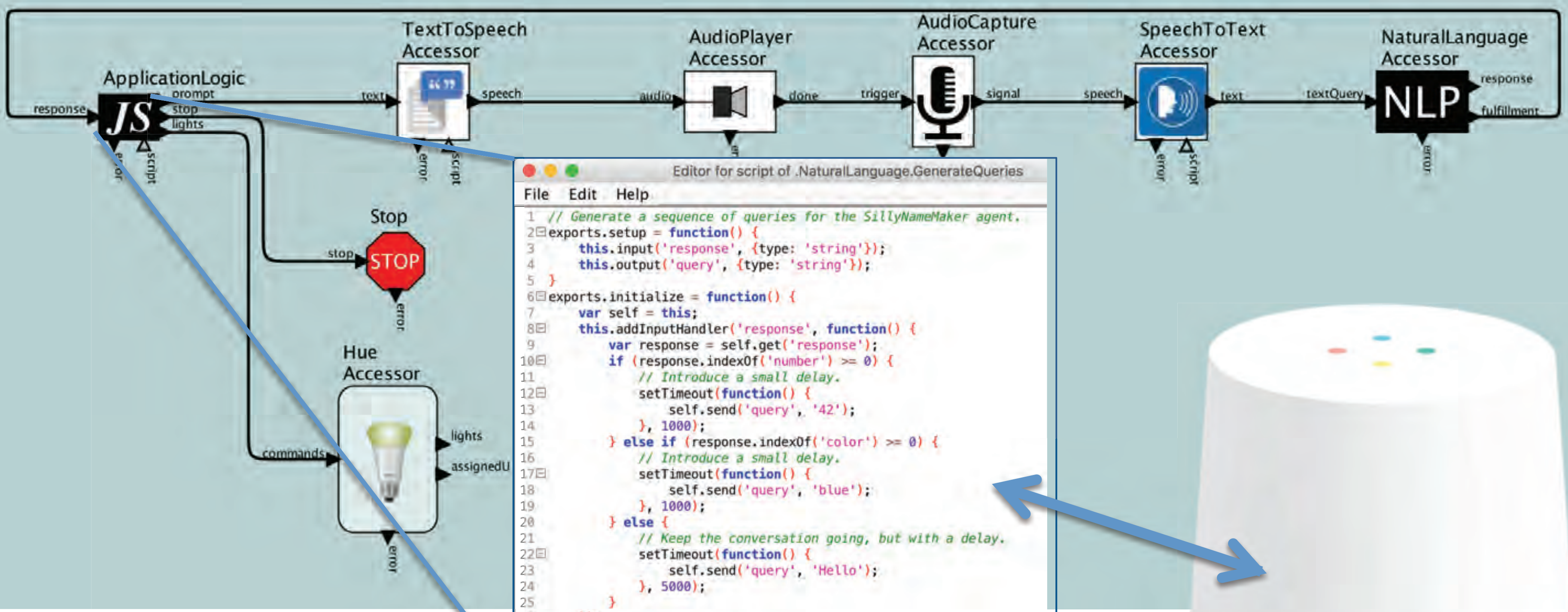
Audio data converted to text via RESTful API.

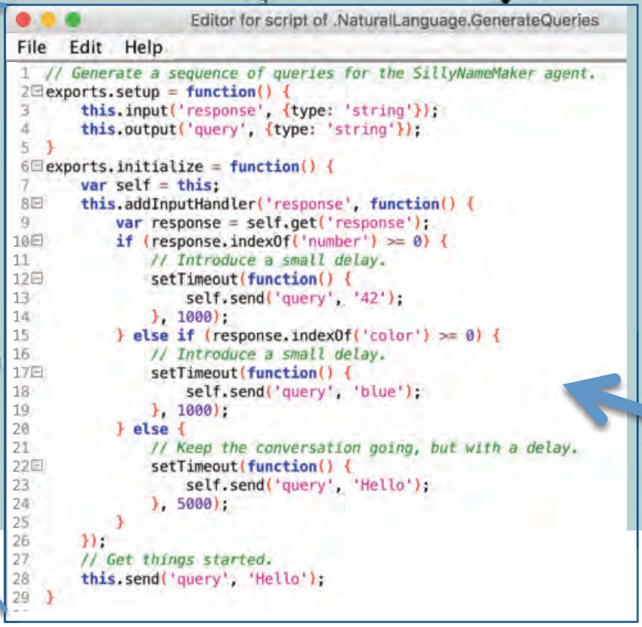
Swarmlet: Composition of Accessors



Natural language processing performed by an API.AI agent.

Swarmlet: Composition of Accessors

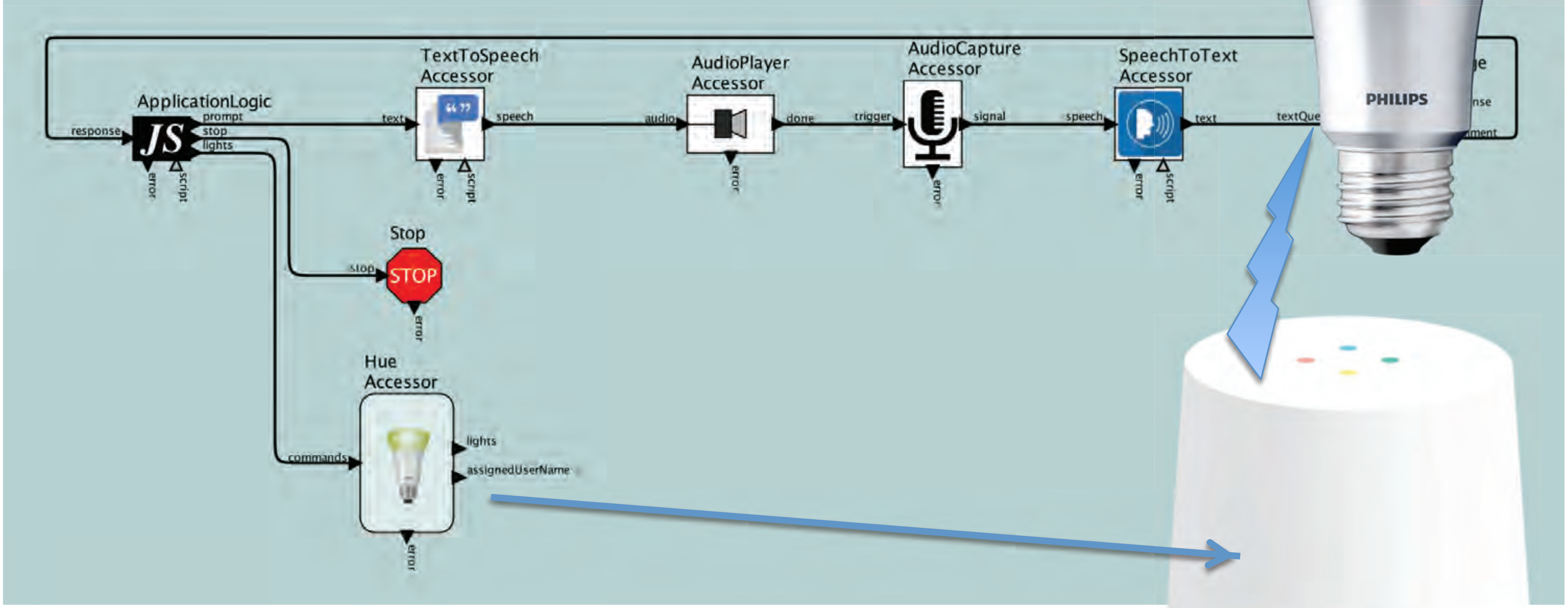




```
// Generate a sequence of queries for the SillyNameMaker agent.
exports.setup = function() {
    this.input('response', {type: 'string'});
    this.output('query', {type: 'string'});
}
exports.initialize = function() {
    var self = this;
    this.addInputHandler('response', function() {
        var response = self.get('response');
        if (response.indexOf('number') >= 0) {
            // Introduce a small delay.
            setTimeout(function() {
                self.send('query', '42');
            }, 1000);
        } else if (response.indexOf('color') >= 0) {
            // Introduce a small delay.
            setTimeout(function() {
                self.send('query', 'blue');
            }, 1000);
        } else {
            // Keep the conversation going, but with a delay.
            setTimeout(function() {
                self.send('query', 'Hello');
            }, 5000);
        }
    });
    // Get things started.
    this.send('query', 'Hello');
}
```

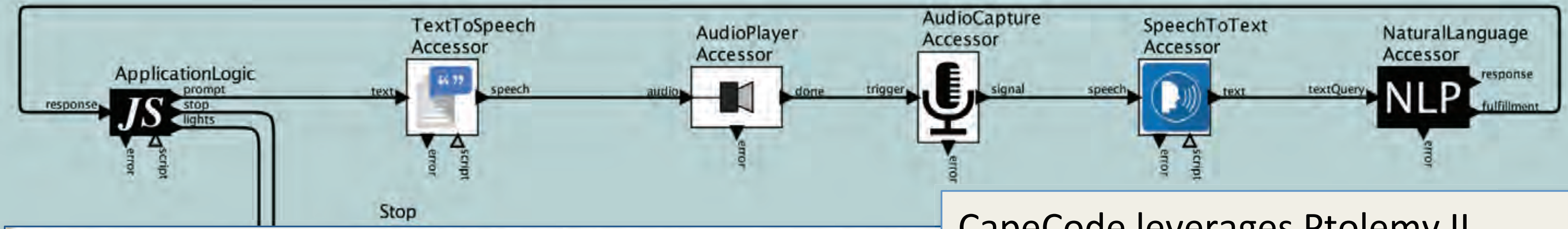
Application decides what to do with the response.

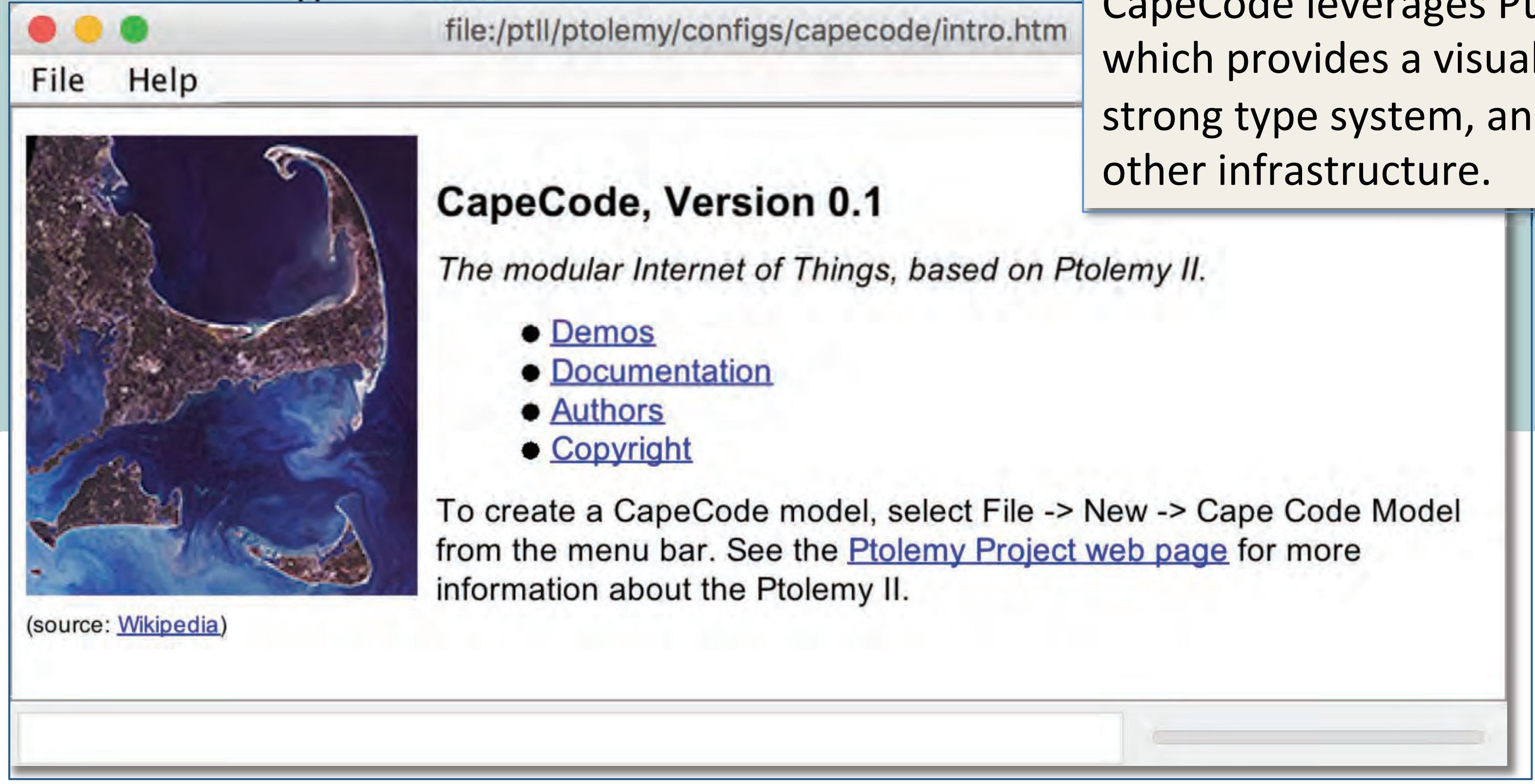
Swarmlet: Composition of Accessors



Accessor uses Zigbee radio on the gateway to control a light bulb.

CapeCode: Swarmlet Development
Our Programming Framework for the IoT





CapeCode, Version 0.1

The modular Internet of Things, based on Ptolemy II.

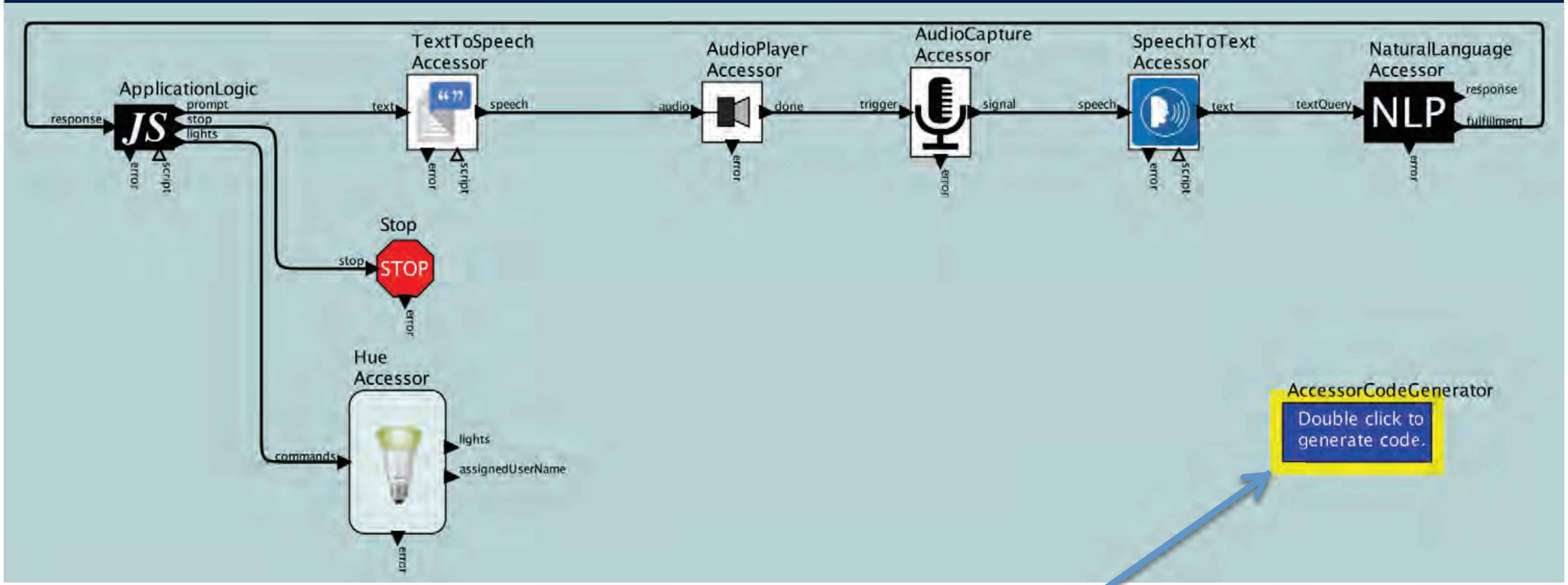
- Demos
- Documentation
- Authors
- Copyright

To create a CapeCode model, select File -> New -> Cape Code Model from the menu bar. See the Ptolemy Project web page for more information about the Ptolemy II.

(source: Wikipedia)

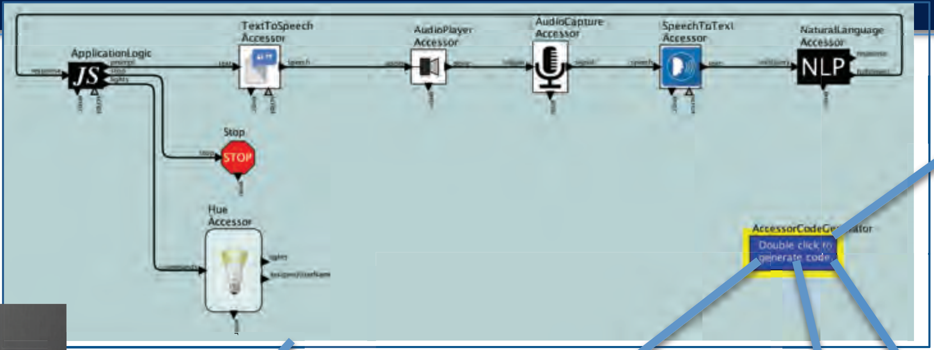
CapeCode leverages Ptolemy II, which provides a visual editor, strong type system, and lots of other infrastructure.

Deploying Swarmlets



Code generator produces JavaScript files that can run on a lightweight JavaScript host.

Deploying Swarmlets
Write Once, Run Everywhere?



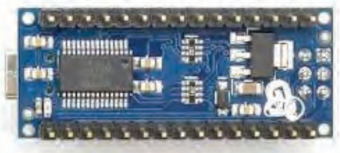
APACHE CORDOVA™

Java 8 JavaScript - Nashorn

Ptolemy II

node JS

((o) Duktape



Then Again…

We could instead use this technology:

What about the Internet of *Important* Things?

Challenges

- Brittle design
  - APIs change
  - Services and devices disappear
- Safety and security
  - Authorization (even without network connectivity)
- Privacy
  - Keep data local whenever possible
- Timing
  - Best-effort timing is not good enough
  - Cloud variability is too much for many applications.
- Regression tests
  - Very difficult to write
  - Need dummy devices and services

Challenges

- Brittle design
  - APIs change
  - Services and devices disappear

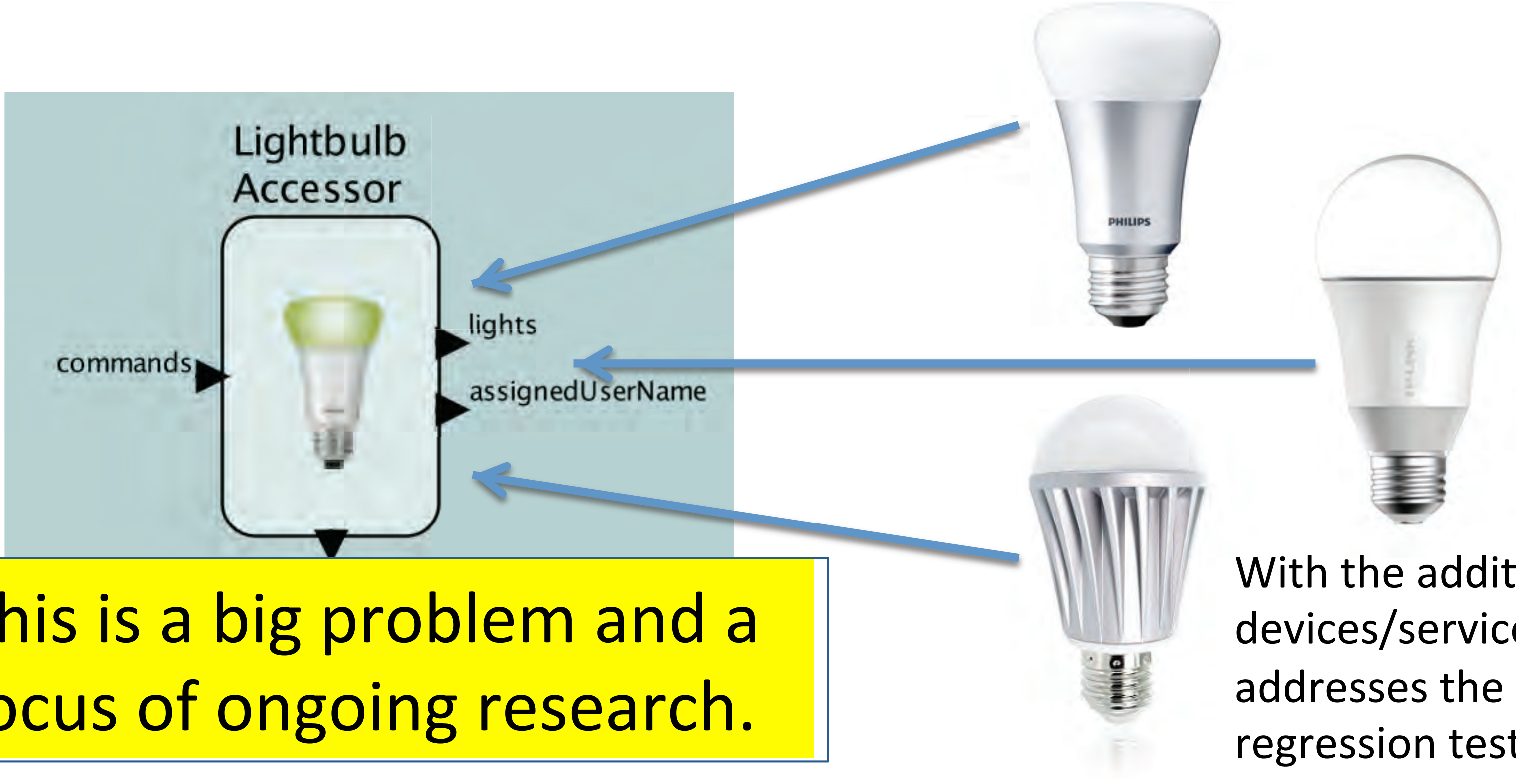
An attempt to run Google's own API.AI demo with sound input rather than text yielded this:

```
{... status: {
    code: 403,
    errorDetails: "API.AI speech recognition is going to be deprecated soon.
                   Use Google Cloud Speech API or other solutions.",
    errorType = "forbidden"}
...}
```

- Regression tests
  - Very difficult to write
  - Need dummy devices and services

Mutable Accessors

Accessors have well-defined interfaces. Mutable accessors can be reified with *discovered* services or (local) devices that match their interfaces.



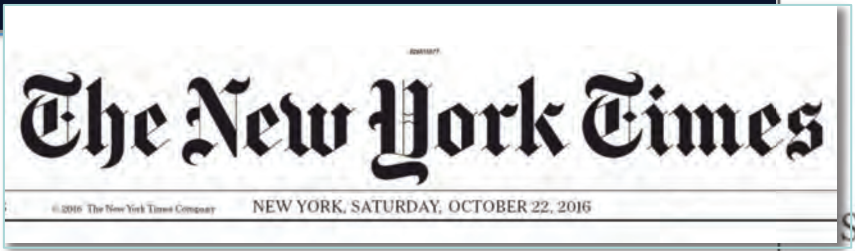
This is a big problem and a focus of ongoing research.

With the addition of dummy devices/services, this also addresses the problem of regression testing.

Challenges

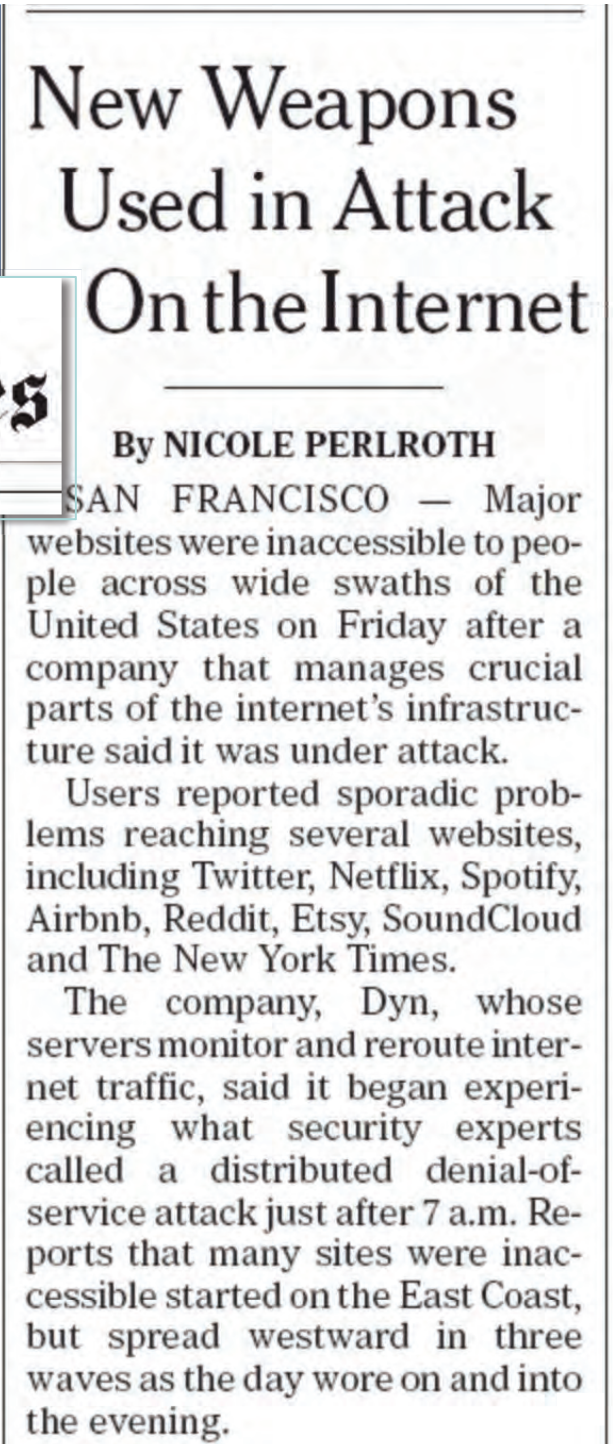
- Brittle design
  - APIs change
  - Services and devices disappear
- Safety and security
  - Authorization (even without network connectivity)
- Privacy
  - Keep data local whenever possible
- Timing
  - Best-effort timing is not good enough
  - Cloud variability is too much for many applications.
- Regression tests
  - Very difficult to write
  - Need dummy devices and services

IoT Devices Shut Down the Internet



The New York Times

© 2016 The New York Times Company NEW YORK, SATURDAY, OCTOBER 22, 2016



New Weapons Used in Attack On the Internet

By NICOLE PERLROTH

SAN FRANCISCO — Major websites were inaccessible to people across wide swaths of the United States on Friday after a company that manages crucial parts of the internet's infrastructure said it was under attack.

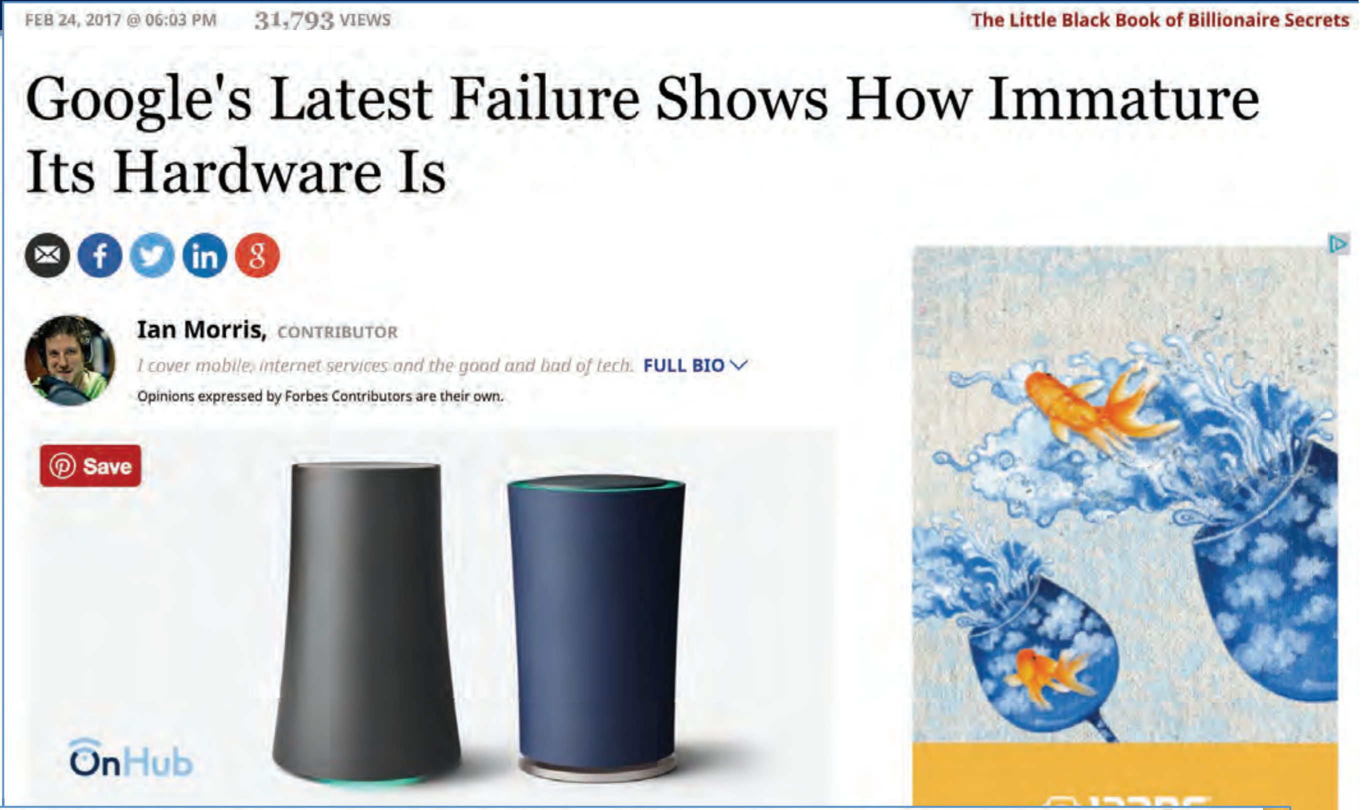
Users reported sporadic problems reaching several websites, including Twitter, Netflix, Spotify, Airbnb, Reddit, Etsy, SoundCloud and The New York Times.

The company, Dyn, whose servers monitor and reroute internet traffic, said it began experiencing what security experts called a distributed denial-of-service attack just after 7 a.m. Reports that many sites were inaccessible started on the East Coast, but spread westward in three waves as the day wore on and into the evening.

Lee, Berkeley

Cloud-based Authentication Shuts Down the Internet and Even Local Services!

Articles in Forbes and ZDNet, Feb. 24, 2017, on the failure of Google's OnHub smart router because of loss of cloud services.



FEB 24, 2017 @ 06:03 PM 31,793 VIEWS

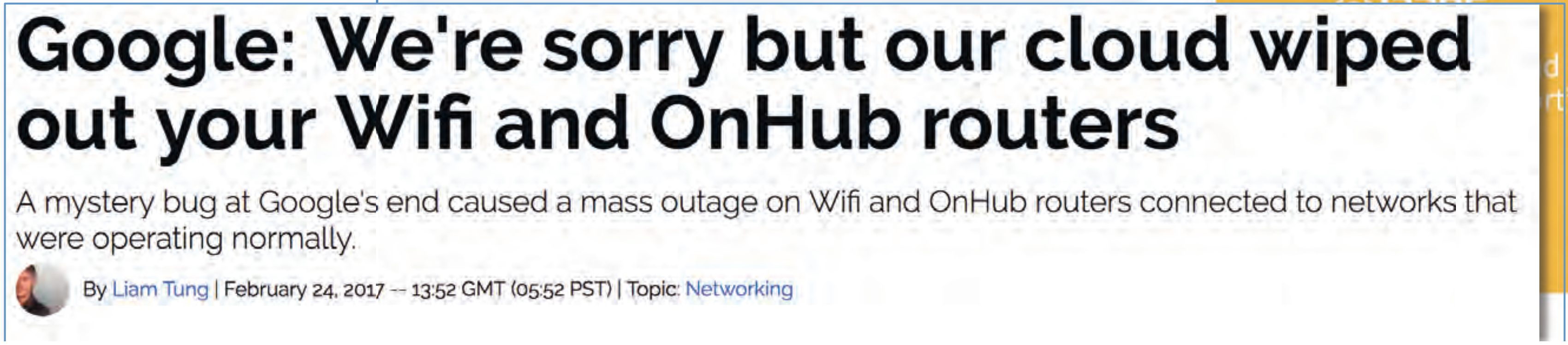
The Little Black Book of Billionaire Secrets

Google's Latest Failure Shows How Immature Its Hardware Is

Ian Morris, CONTRIBUTOR

I cover mobile, internet services and the good and bad of tech. FULL BIO

Opinions expressed by Forbes Contributors are their own.

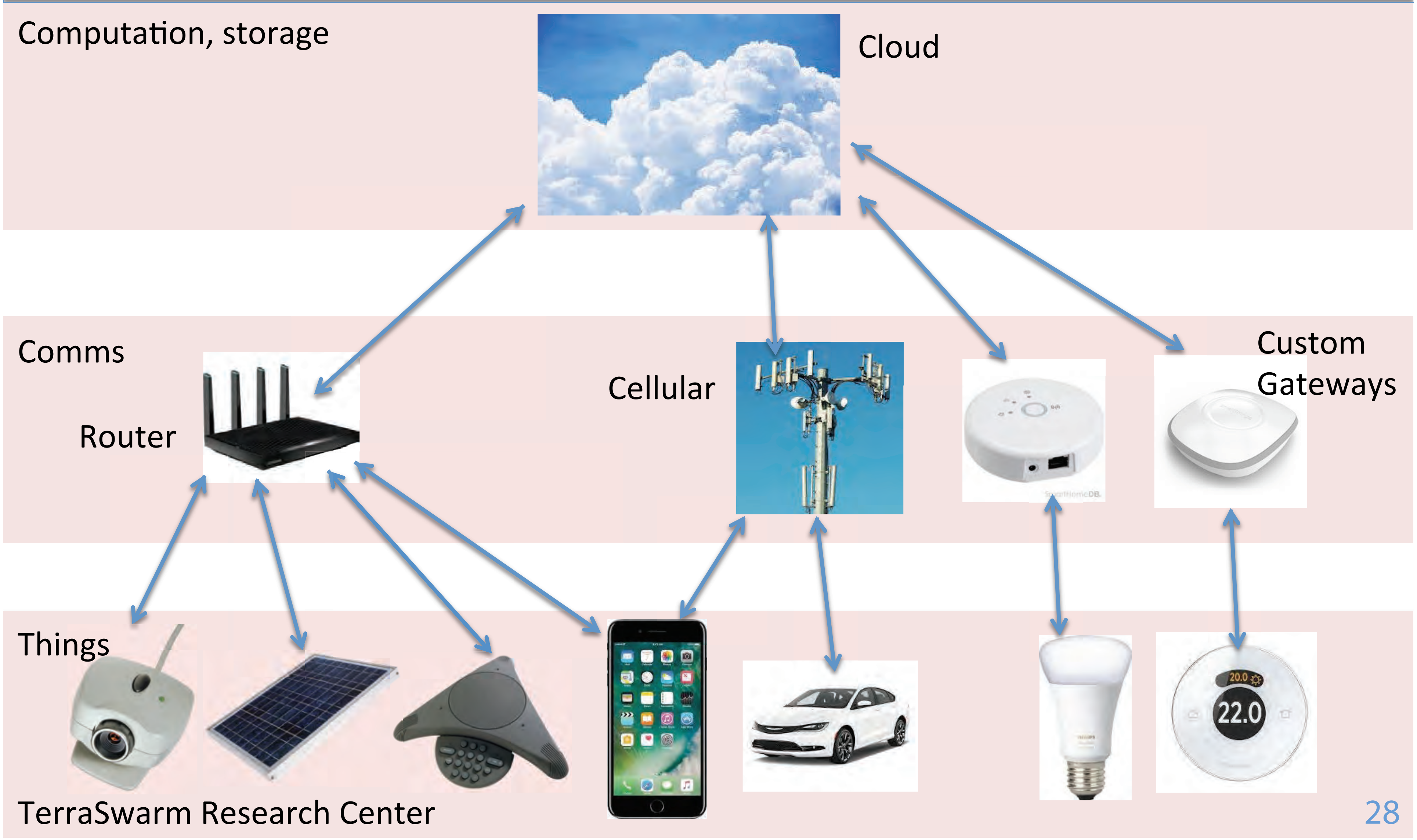


Google: We're sorry but our cloud wiped out your Wifi and OnHub routers

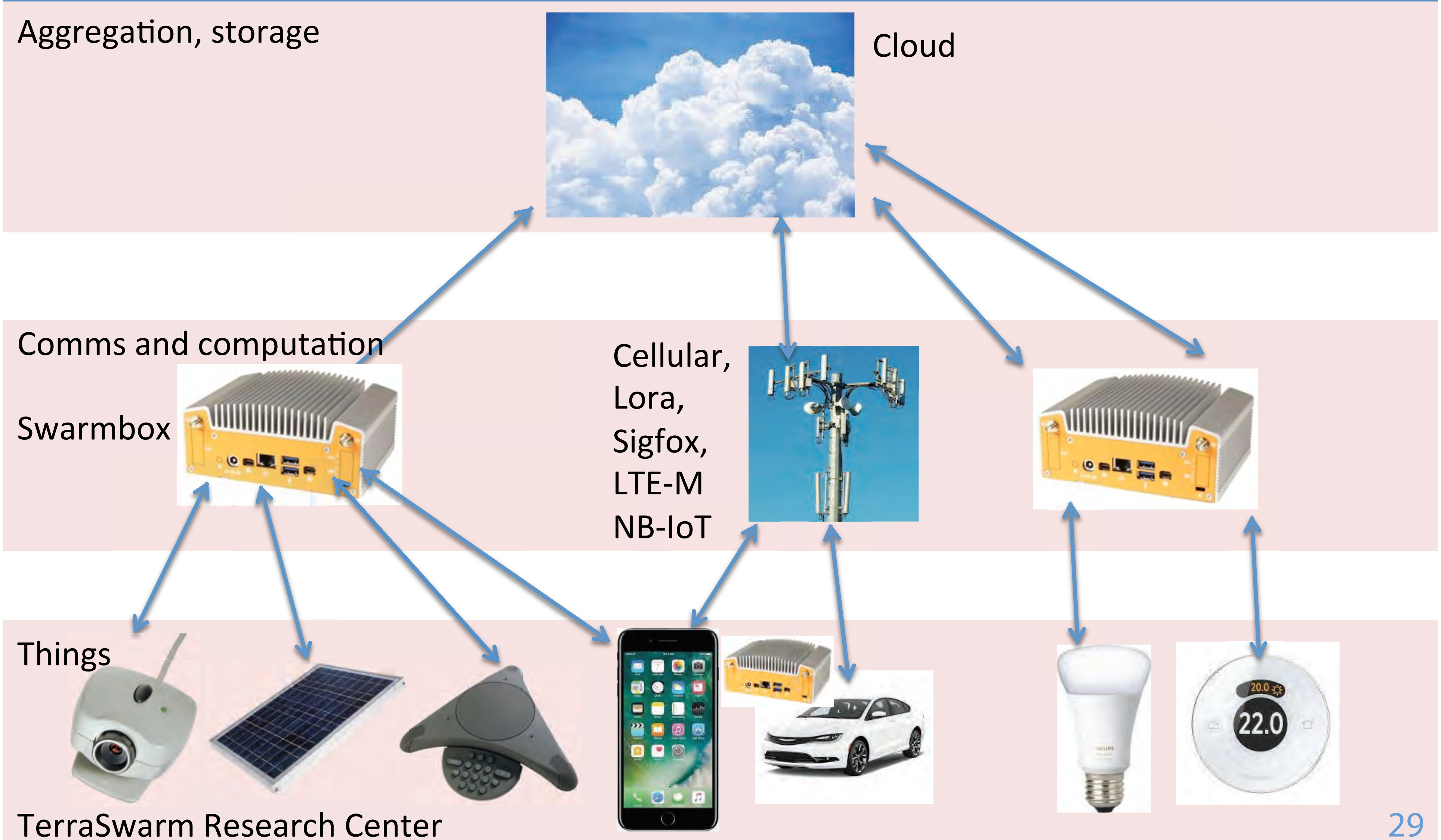
A mystery bug at Google's end caused a mass outage on Wifi and OnHub routers connected to networks that were operating normally.

By Liam Tung | February 24, 2017 -- 13:52 GMT (05:52 PST) | Topic: Networking

Cloud-Centric Services Today



Tomorrow: Edge Computing, Smart Gateways, Fog Computing, Swarmboxes, …

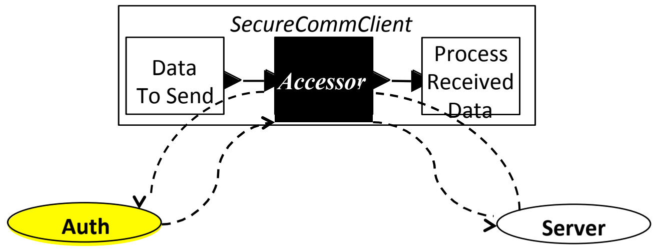


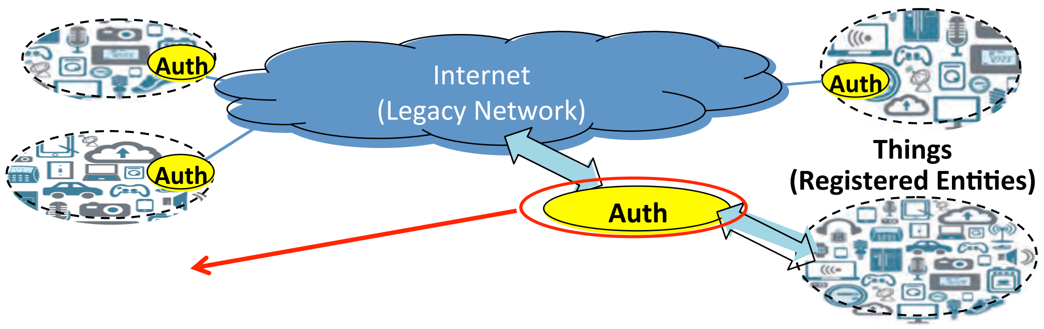
Fog-based Security
SST: Secure Swarm Toolkit

[Hokeun Kim]

Locally centralized, globally distributed authentication, authorization, and security.

- Open-source local authorization entity *Auth* as a gateway for authorization of the local "Things"
- Secure communication accessors for accessing local authorization service





The goal of the accessor is to make state-of-the-art security usable by nonexperts.

Local Auth operates even in the presence of Internet outages (e.g. Dyn attack, Nov. 2016).

See Kim, et al., "A Secure Network Architecture for the Internet of Things Based on Local Authorization Entities," FiCloud '16

Locally Centralized, Globally Distributed SST: Secure Swarm Toolkit

[Hokeun Kim]

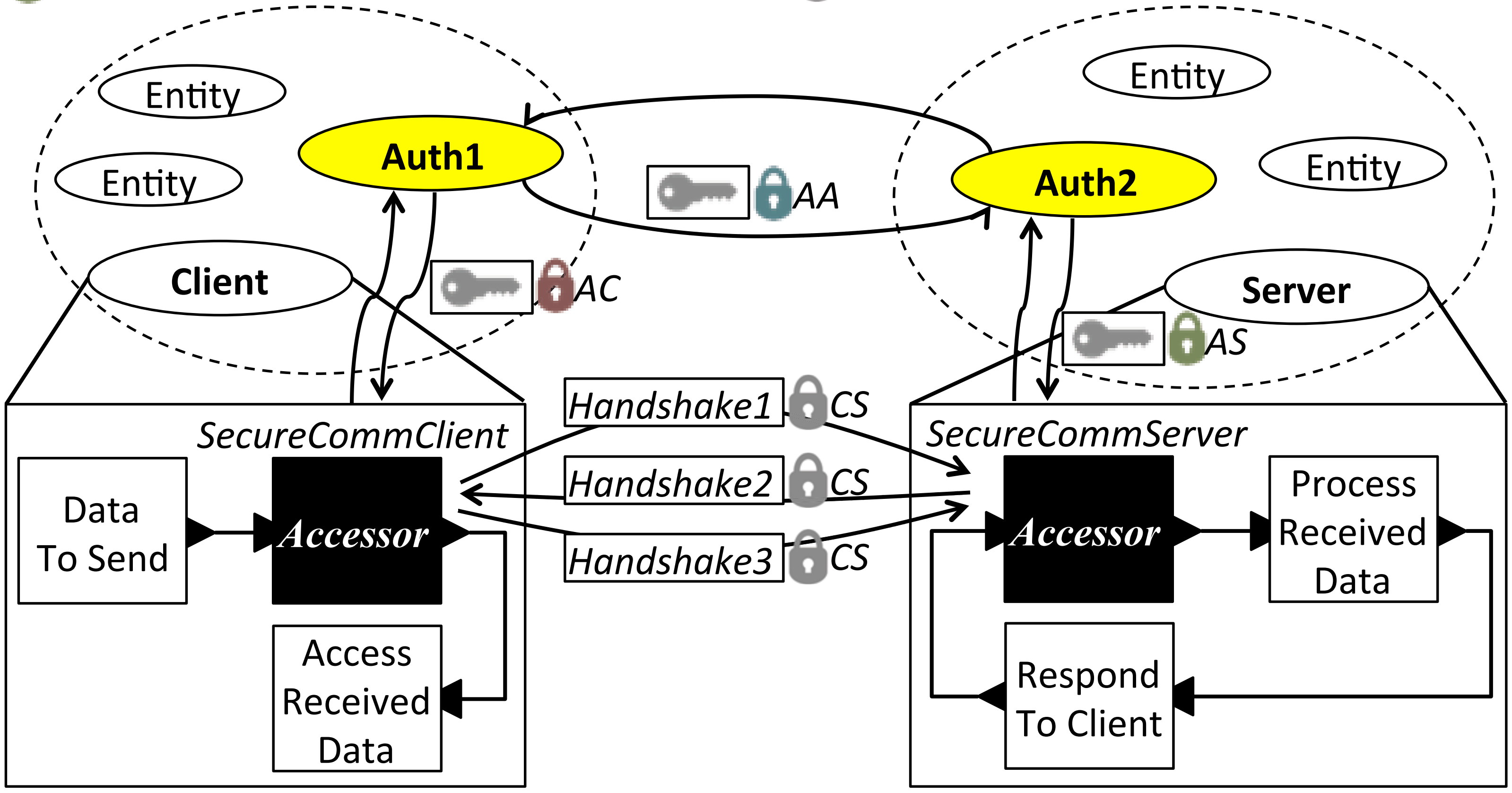
: Key for **C**lient & **S**erver

AC: Encrypted with Key for **A**uth1 & **C**lient

AA: Encrypted with Key for **A**uth1 & **A**uth2

AS: Encrypted with Key for **A**uth2 & **S**erver

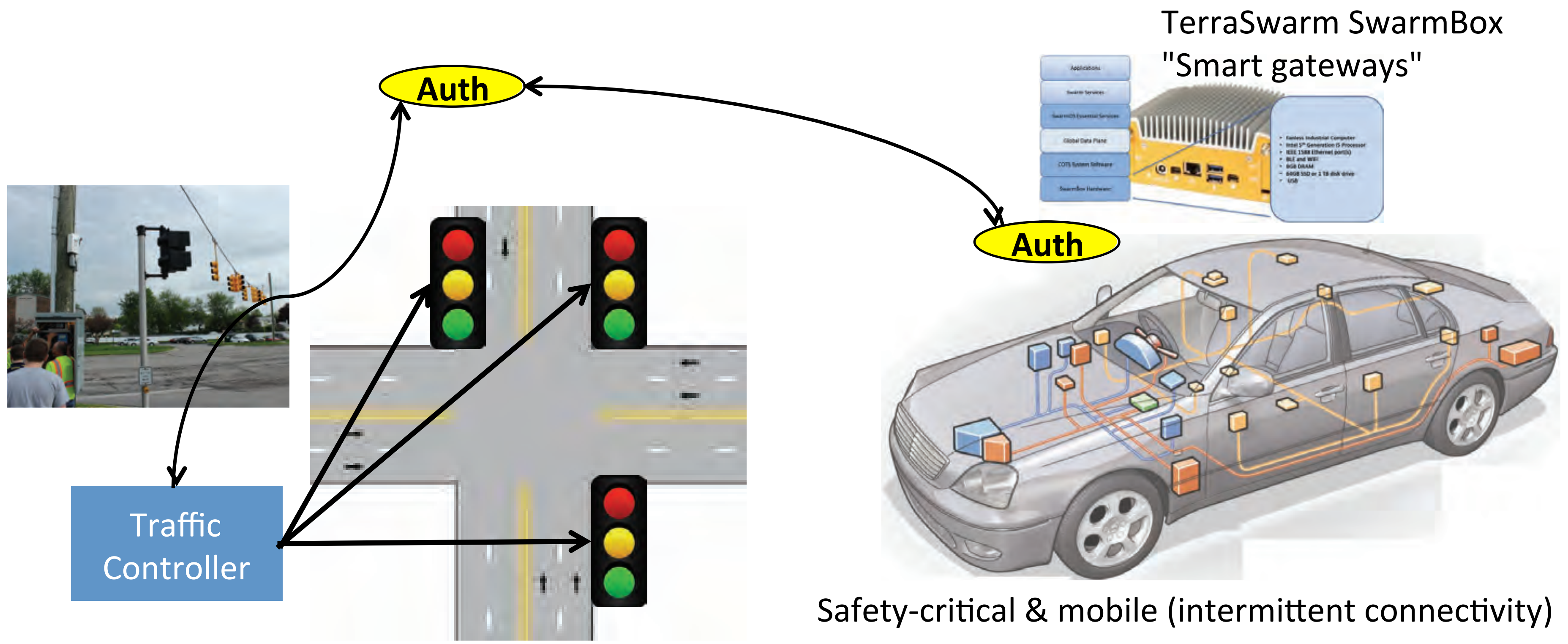
CS: Encrypted with Key for **C**lient & **S**lient



Application Example
SST: Secure Swarm Toolkit

[Hokeun Kim]

Traffic infrastructure & vehicles



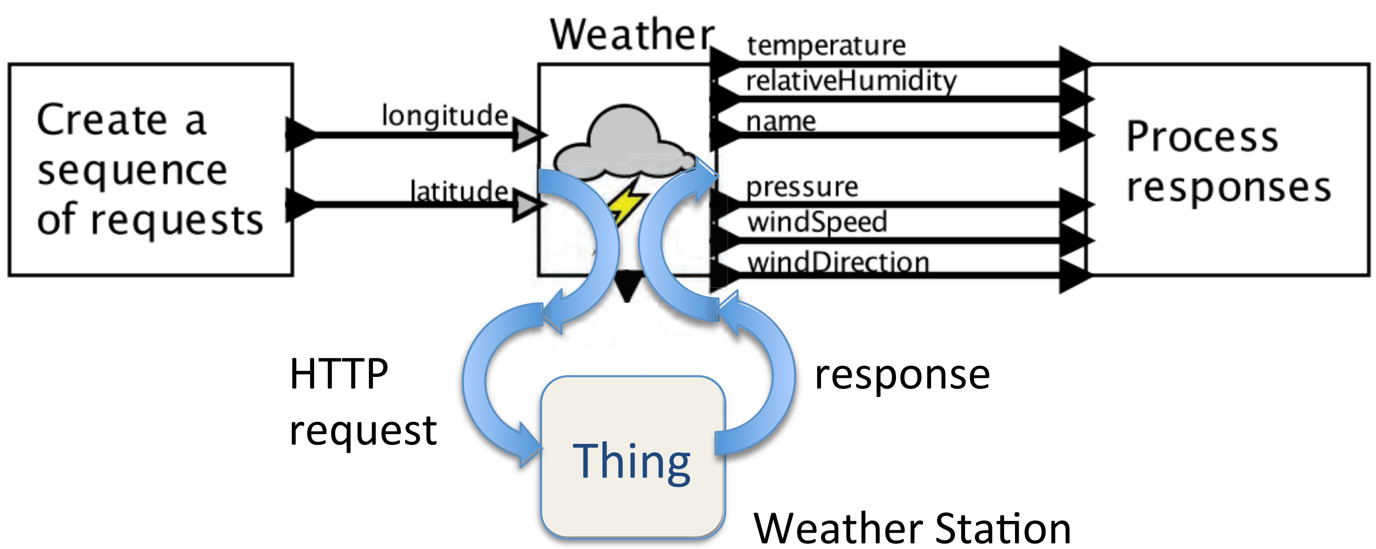
Safety-critical & mobile (intermittent connectivity)

Challenges

- Brittle design
  - APIs change
  - Services and devices disappear
- Safety and security
  - Authorization (even without network connectivity)
- Privacy
  - Keep data local whenever possible
- Timing
  - Best-effort timing is not good enough
  - Cloud variability is too much for many applications.
- Regression tests
  - Very difficult to write
  - Need dummy devices and services

Actors and AAC: Timing

Example of a potential problem:

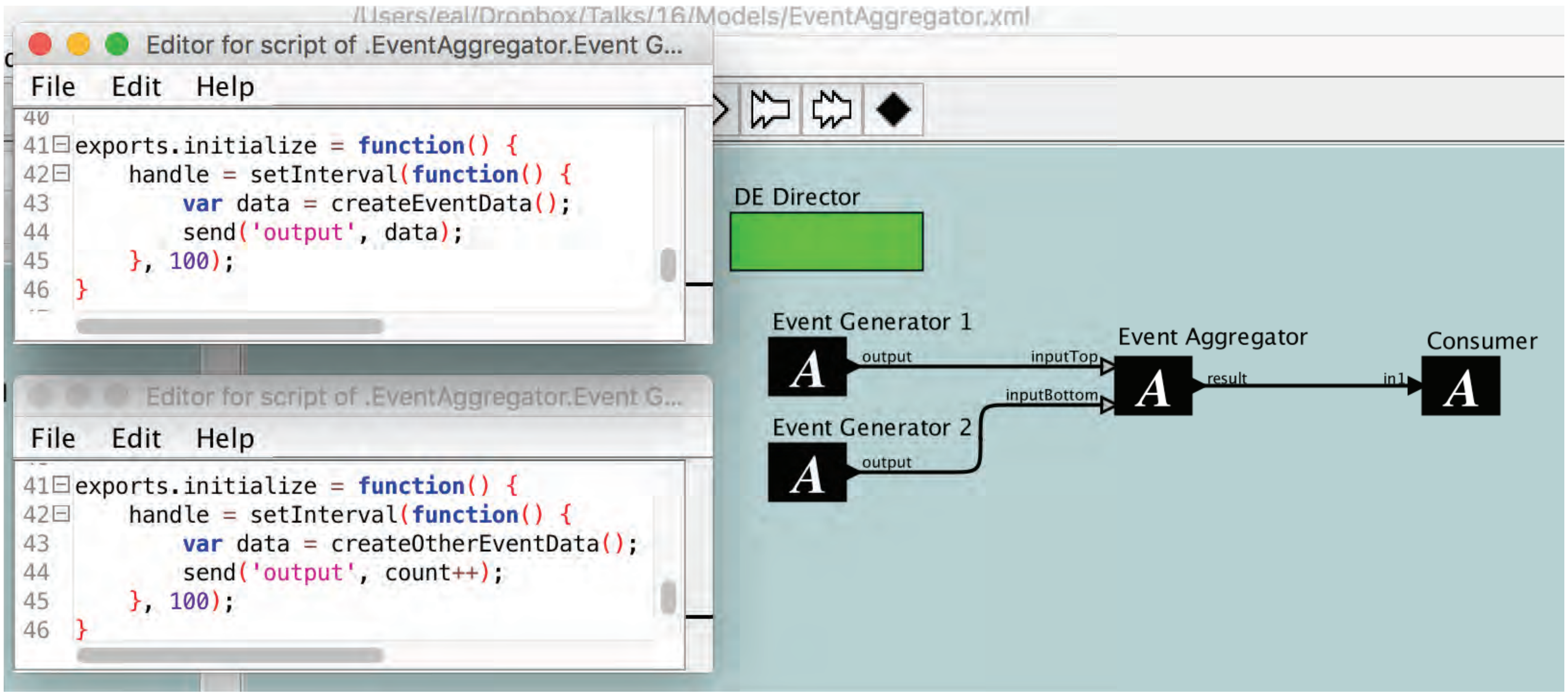


The responses may not come back in the same order as the requests!

This is a rudimentary timing problem.

Another Timing Problem

Coordinated timing:



/Users/eal/Dropbox/Talks/16/Models/EventAggregator.xml

Editor for script of .EventAggregator.Event G...

```
exports.initialize = function() {
    handle = setInterval(function() {
        var data = createEventData();
        send('output', data);
    }, 100);
}
```

Editor for script of .EventAggregator.Event G...

```
exports.initialize = function() {
    handle = setInterval(function() {
        var data = createOtherEventData();
        send('output', count++);
    }, 100);
}
```

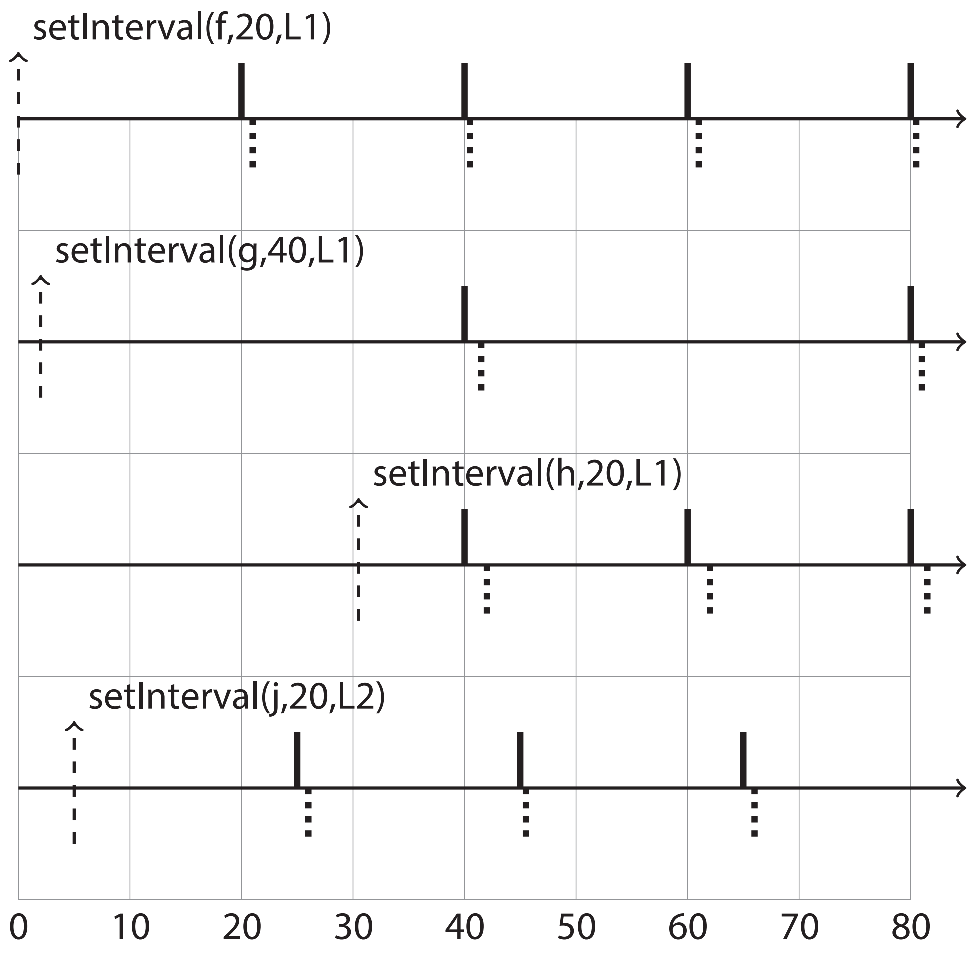
Timing Problems Loom Large in The Internet of *Important* Things (IoIT)

The order and timing of events matters a lot when interacting with physical processes.

The system at the right orchestrates hundreds of microcontrollers to deposit ink on paper flying through the printer at 100 km/h with micron precision.

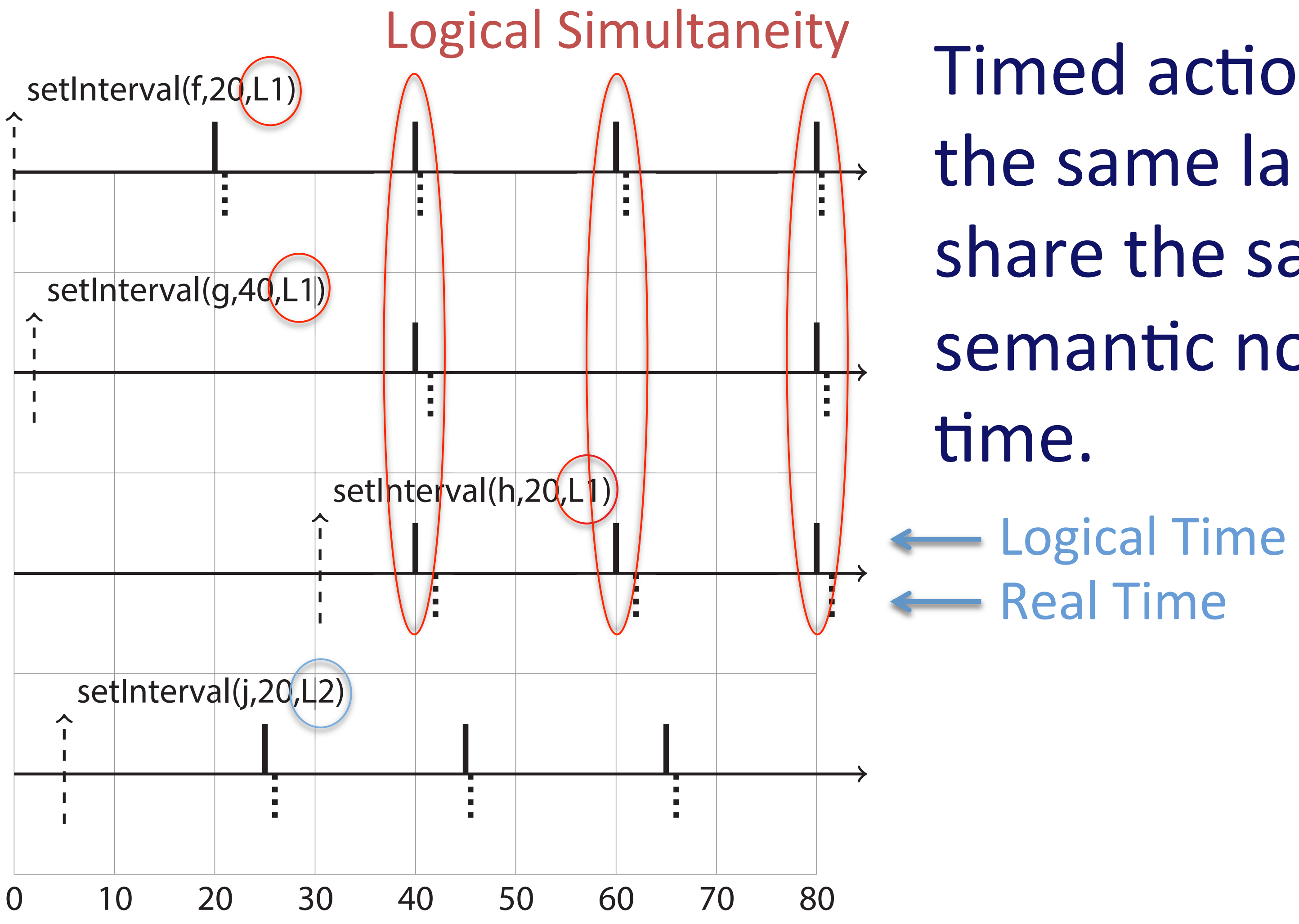
This Bosch Rexroth printing press is a cyber-physical factory using Ethernet and TCP/IP with high-precision clock synchronization (IEEE 1588) on an isolated LAN.

Labeled Clock Domains



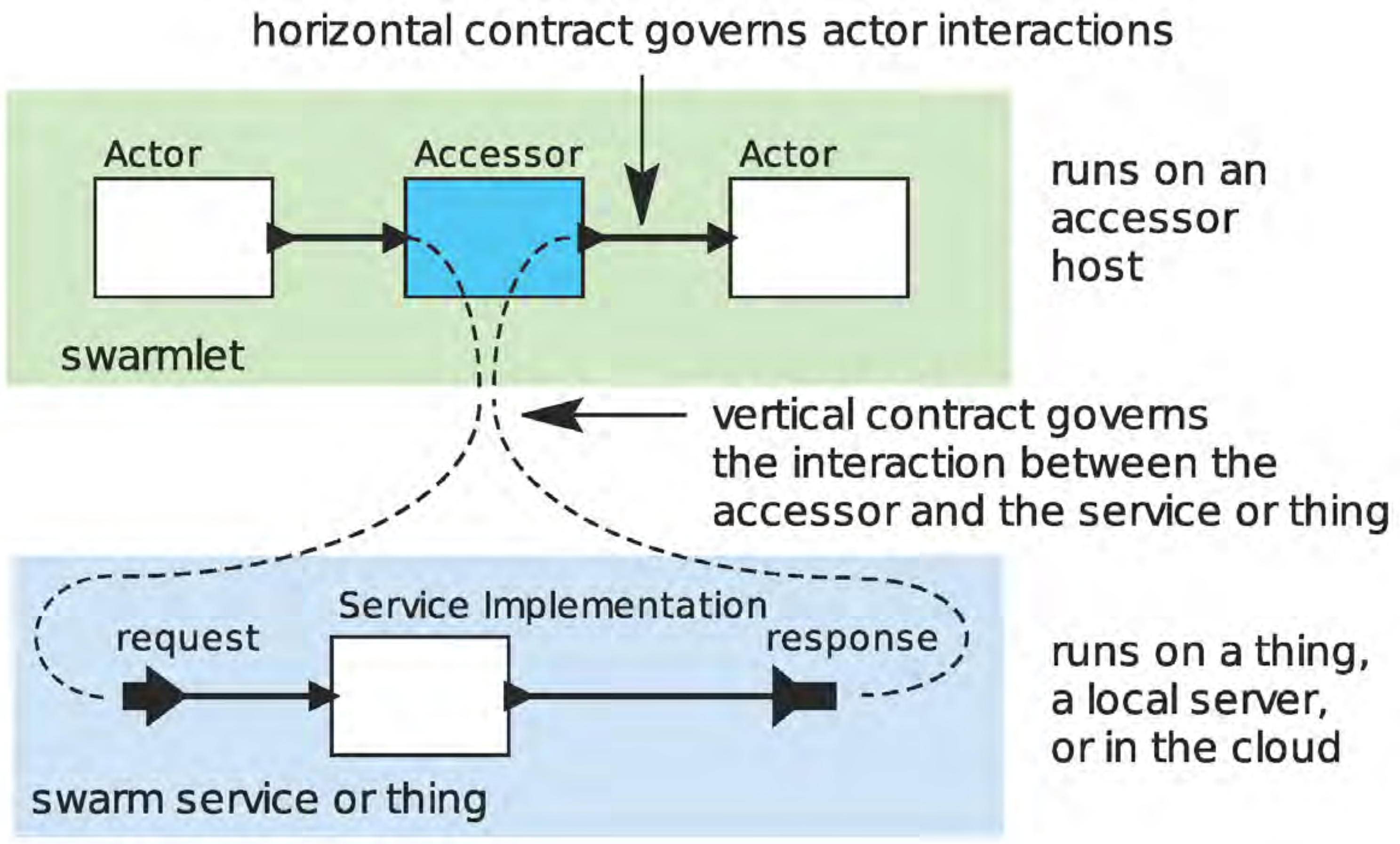
Our accessor framework introduces the idea of *labeled clock domains* with a semantic notion of simultaneity and deterministic ordering of events.

Labeled Clock Domains



Timed actions with the same label share the same semantic notion of time.

Focus on Interfaces

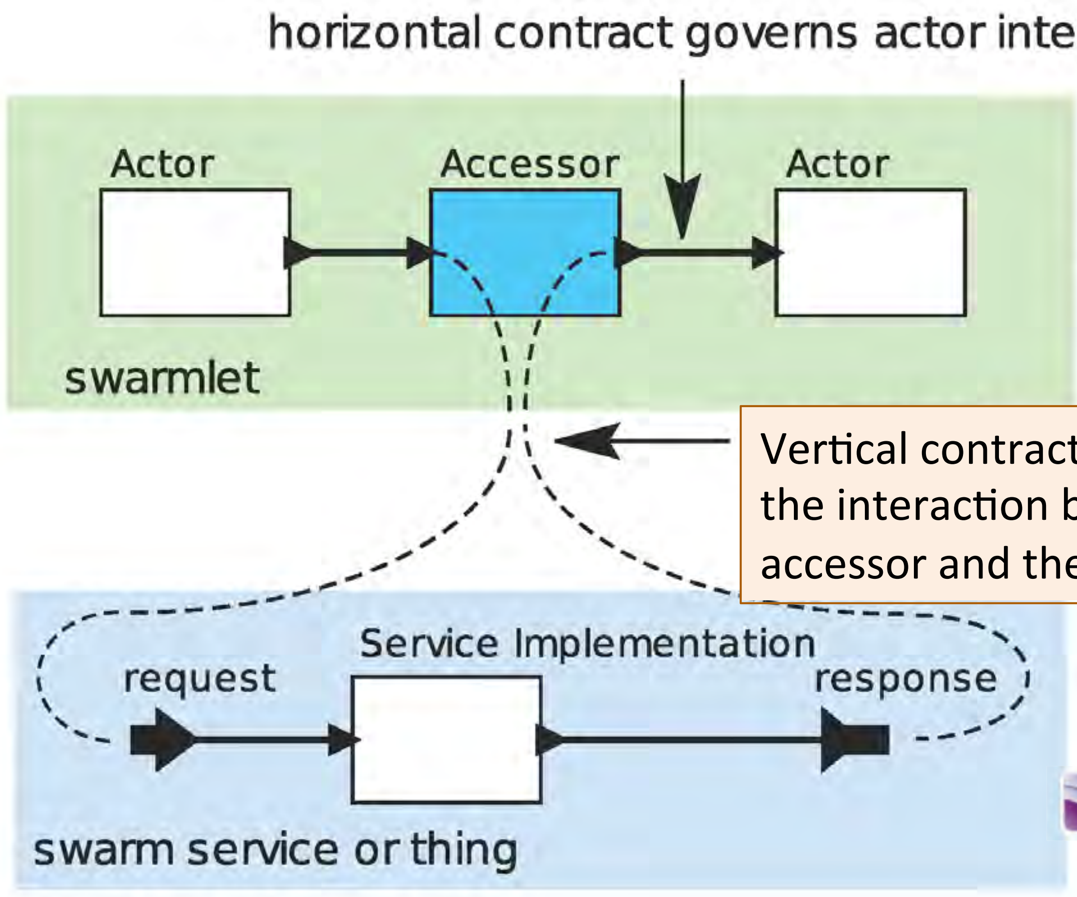

horizontal contract governs actor interactions
Actor
Accessor
Actor
swarmlet
runs on an accessor host
vertical contract governs the interaction between the accessor and the service or thing
Service Implementation
request
response
swarm service or thing
runs on a thing, a local server, or in the cloud

Standardization can occur with either the horizontal contract or the vertical contract.

E.g. asynchronous atomic callbacks (AAC).

Vertical Contract Standards

Focus on over-the-wire protocols

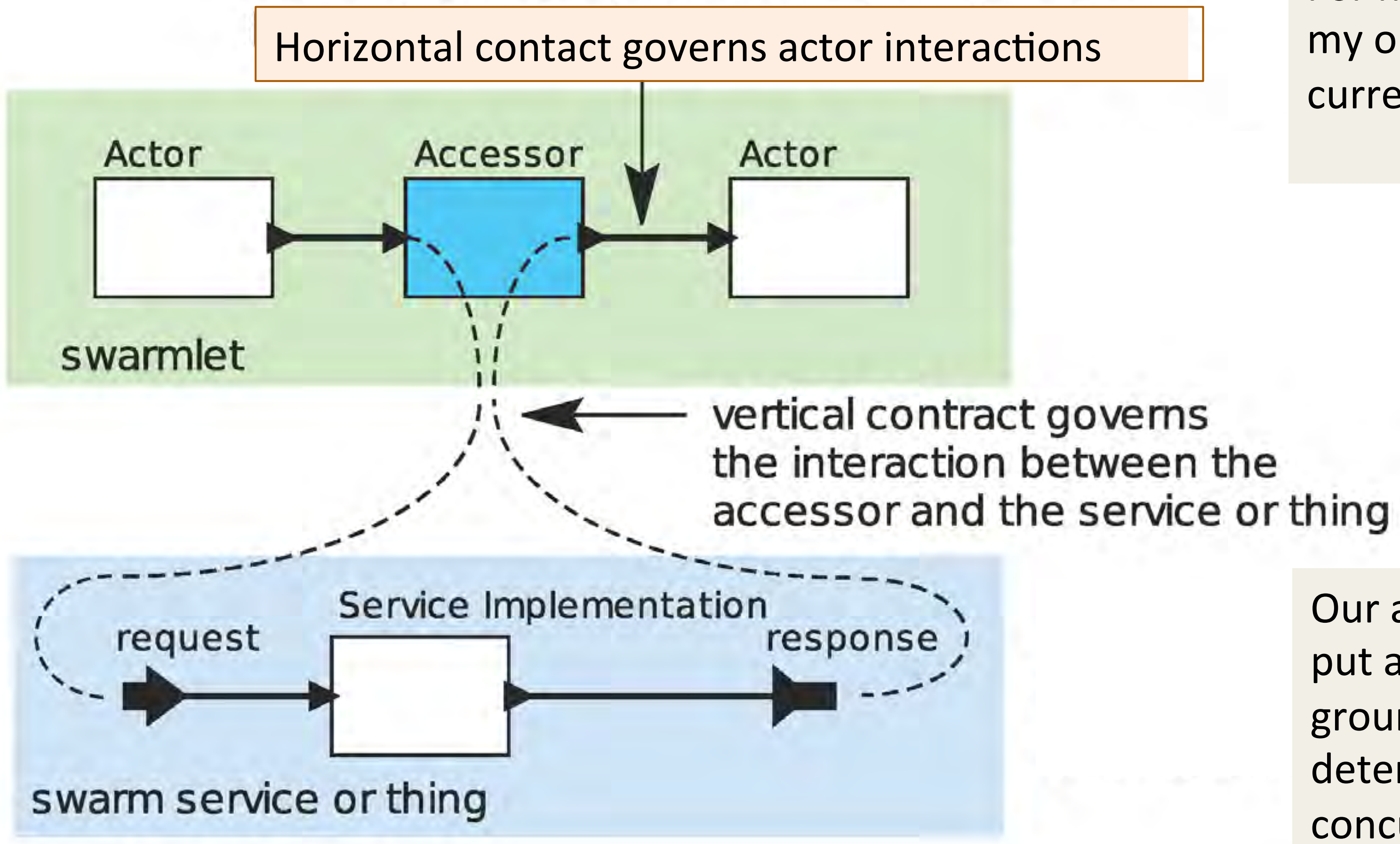


Vertical contract governs the interaction between the accessor and the service or thing.

- HTTP
- WebSockets
- CoAP
- XMPP
- MQTT
- UPnP
- DDS
- …

Horizontal Contract (Standards?)

Providing a *local proxy* for a *remote service*

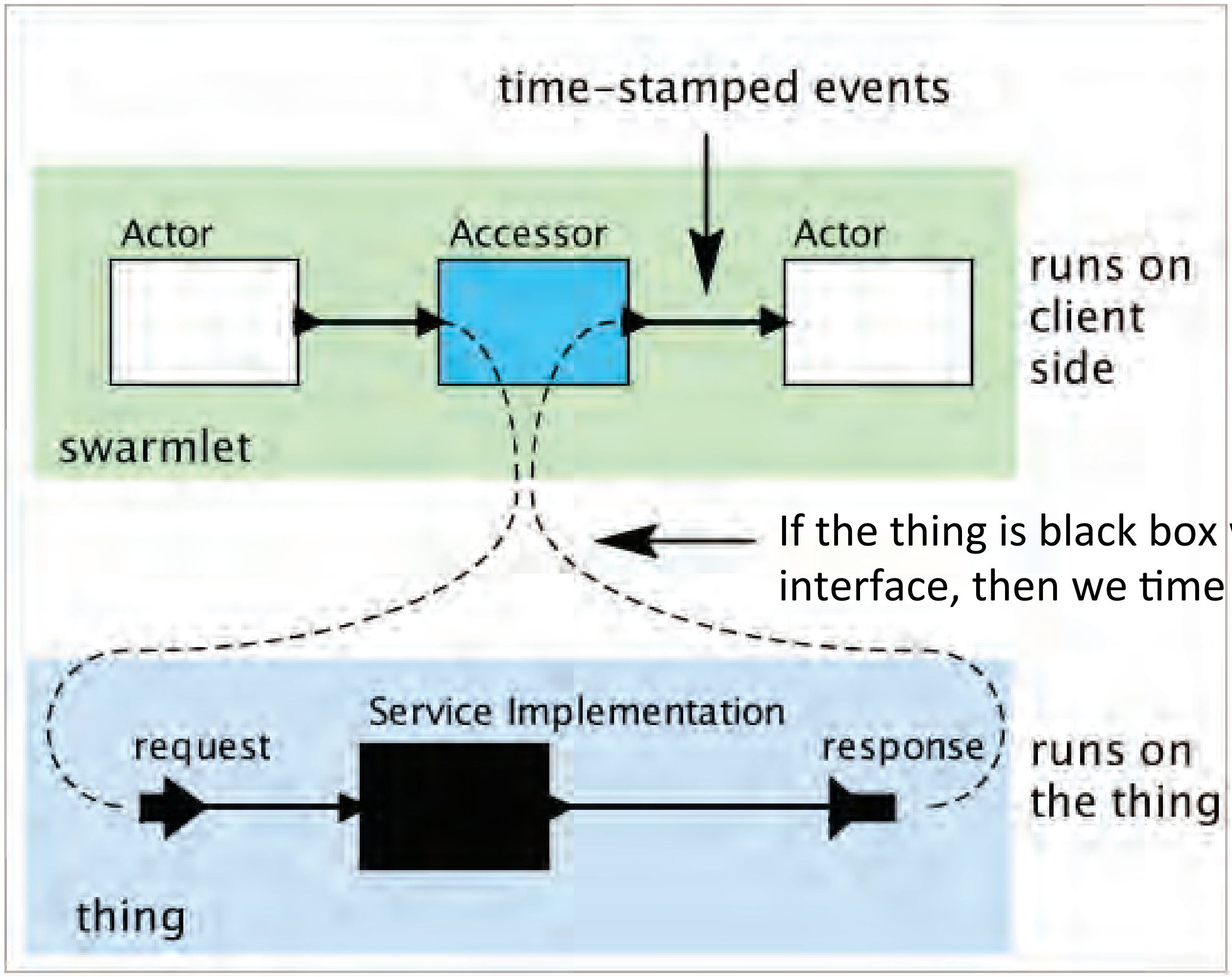

Horizontal contact governs actor interactions
Actor
Accessor
Actor
swarmlet
vertical contract governs the interaction between the accessor and the service or thing
Service Implementation
request
response
swarm service or thing

For horizontal contracts, my opinion is that current work is weak.

Our accessors work has put a stake in the ground that insists on deterministic concurrency models for composition of accessors.

Focus on the horizontal contract
Discrete Event MoC

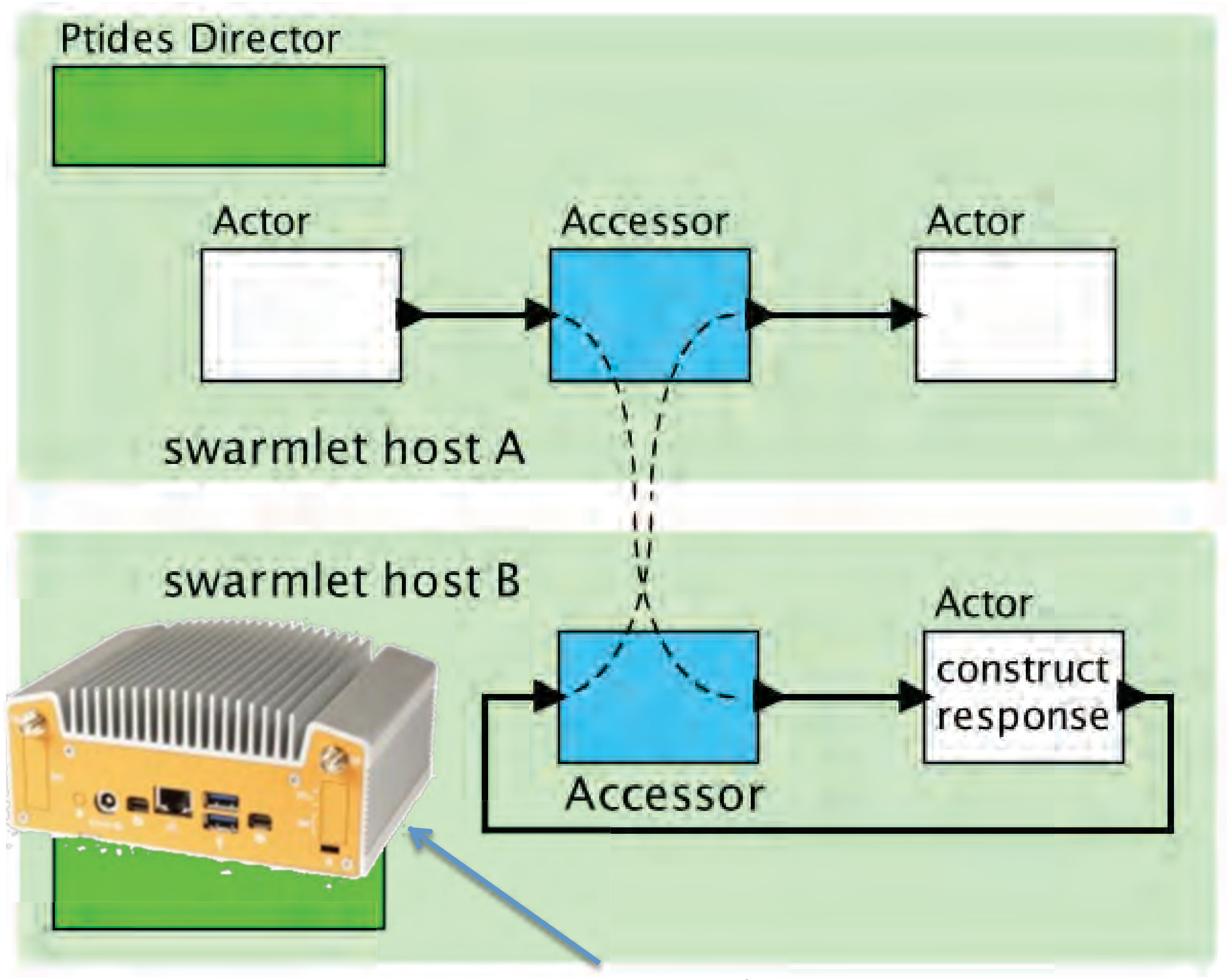
We use time-stamped events processed in time-stamp order, a discrete-event (DE) model of computation (MoC).



If the thing is black box with a RESTful interface, then we time stamp the response

… but if we can design the thing, we can do much better!

Distributed Swarmlets using Accessors

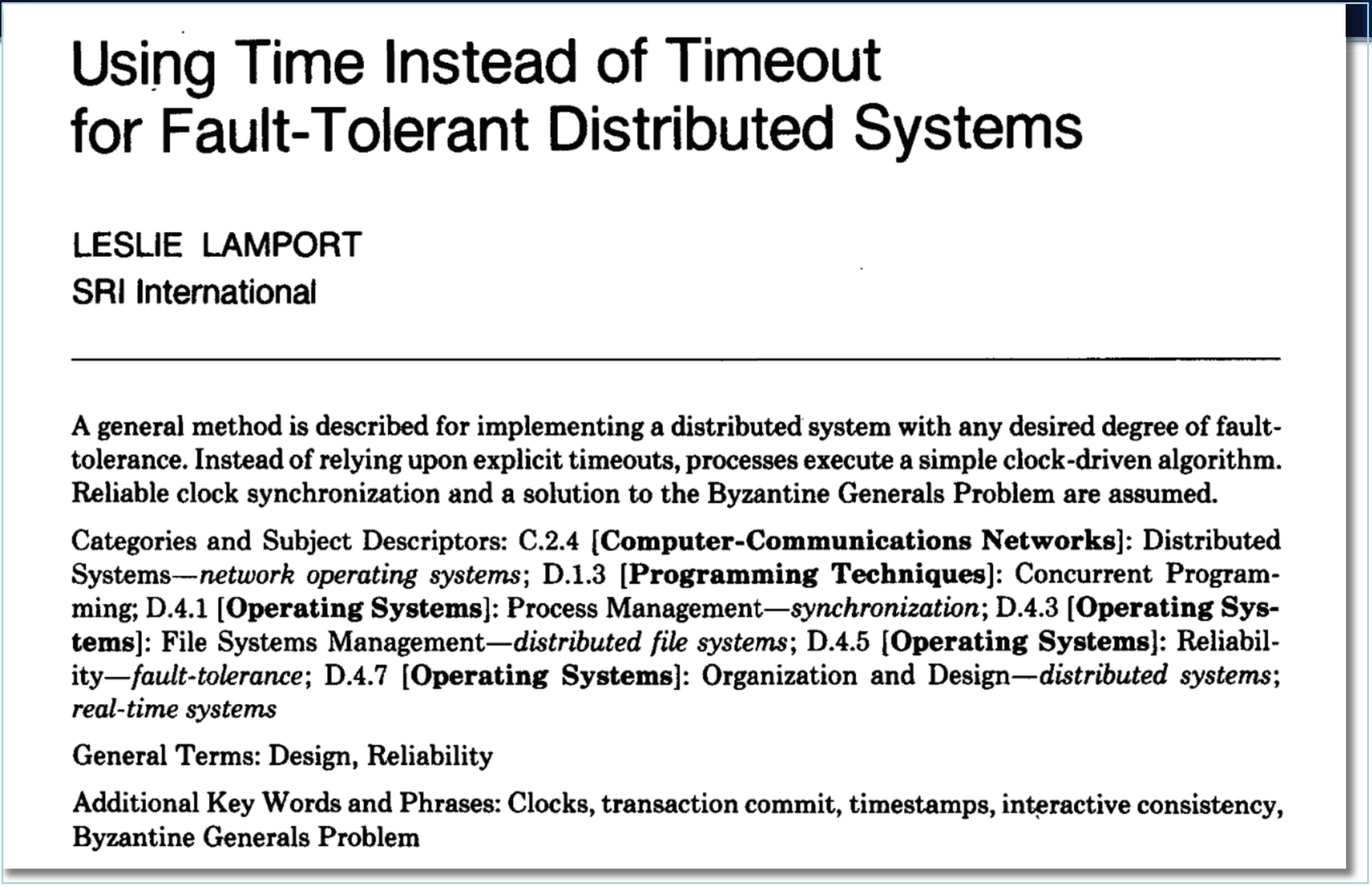


SwarmBox, Edge computer, Fog computer, …

Leveraging time stamps and synchronized clocks, we can achieve **deterministic** distributed MoCs.

Technique: PTIDES

Roots of the Idea



Using Time Instead of Timeout
for Fault-Tolerant Distributed Systems

LESLIE LAMPORT
SRI International

A general method is described for implementing a distributed system with any desired degree of fault-tolerance. Instead of relying upon explicit timeouts, processes execute a simple clock-driven algorithm. Reliable clock synchronization and a solution to the Byzantine Generals Problem are assumed.

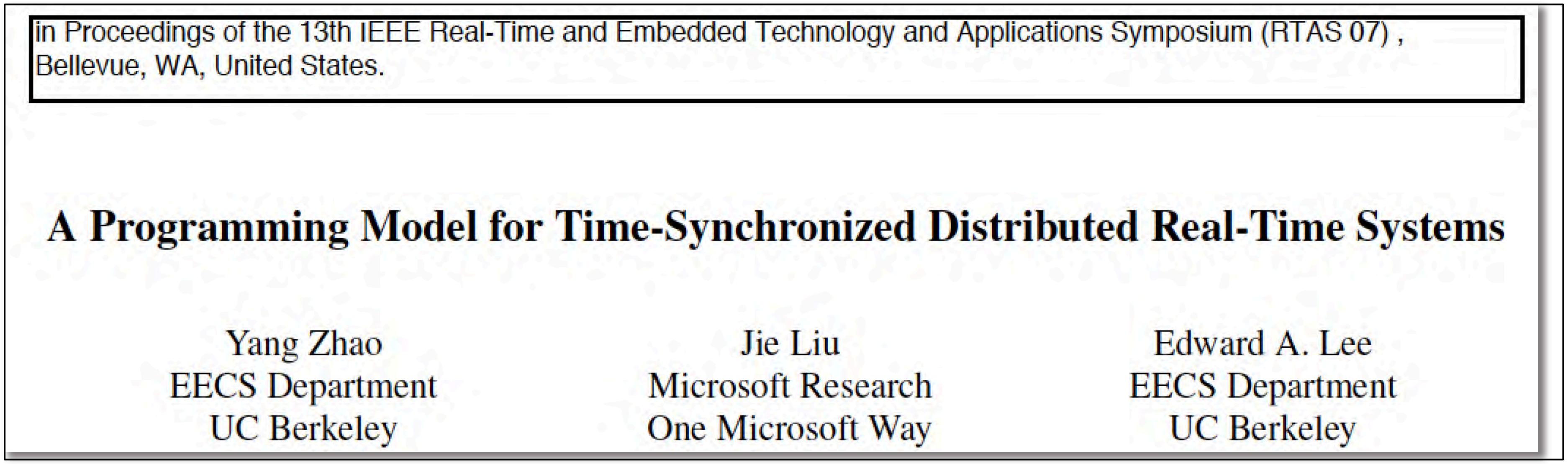
Categories and Subject Descriptors: C.2.4 [**Computer-Communications Networks**]: Distributed Systems—*network operating systems*; D.1.3 [**Programming Techniques**]: Concurrent Programming; D.4.1 [**Operating Systems**]: Process Management—*synchronization*; D.4.3 [**Operating Systems**]: File Systems Management—*distributed file systems*; D.4.5 [**Operating Systems**]: Reliability—*fault-tolerance*; D.4.7 [**Operating Systems**]: Organization and Design—*distributed systems*; *real-time systems*

General Terms: Design, Reliability

Additional Key Words and Phrases: Clocks, transaction commit, timestamps, interactive consistency, Byzantine Generals Problem

ACM Transactions on Programming Languages and Systems, 1984.

Ptides – A Robust Distributed DE MoC for IoIT Applications



in Proceedings of the 13th IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS 07) , Bellevue, WA, United States.

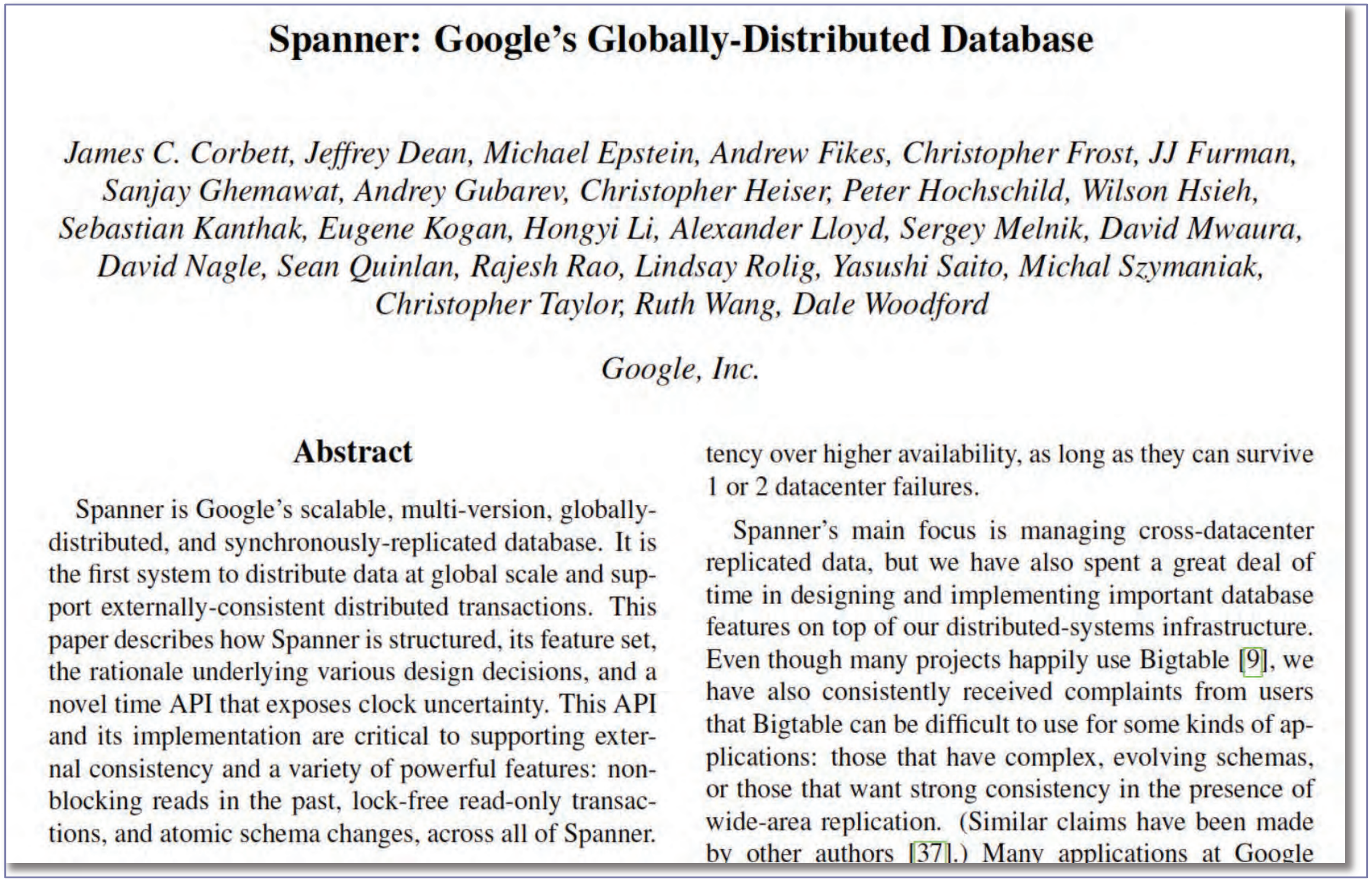
A Programming Model for Time-Synchronized Distributed Real-Time Systems

Yang Zhao, EECS Department, UC Berkeley
Jie Liu, Microsoft Research, One Microsoft Way
Edward A. Lee, EECS Department, UC Berkeley

Abstract: Discrete-event (DE) models are formal system specifications that have analyzable deterministic behaviors. Using a global, consistent notion of time, DE components communicate via time-stamped events. DE models have primarily been used in performance modeling and simulation, where time stamps are a modeling property bearing no relationship to real time during execution of the model. In this paper, we extend DE models with the capability of relating certain events to physical time…

Google Spanner – A Reinvention

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

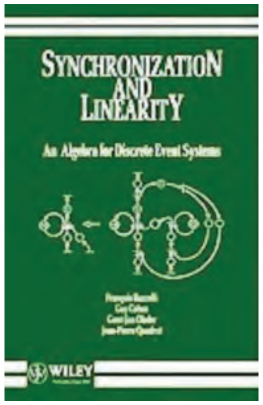
PTIDES: Discrete-Event Semantics + Synchronized Clocks + Sensors and Actuators

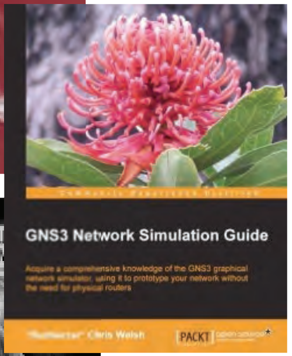
Time-stamped events that are processed in time-stamp order.

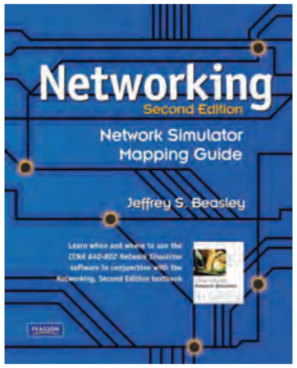
This MoC is widely used in simulation and HDLs.

Given time-stamped inputs, it is a deterministic concurrent MoC.

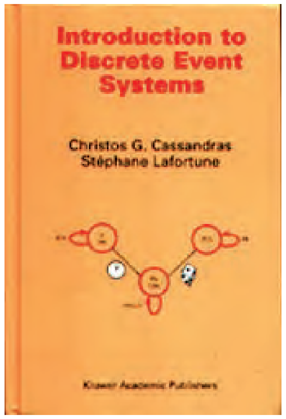
A few texts that use the DE MoC

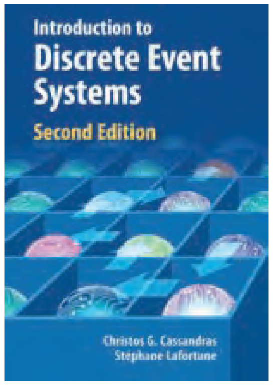

Synchronization and Linearity
An Algebra for Discrete Event Systems
WILEY

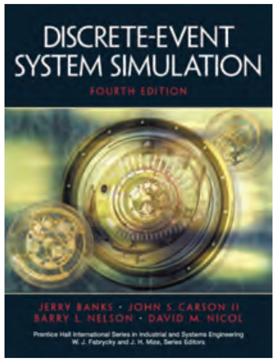




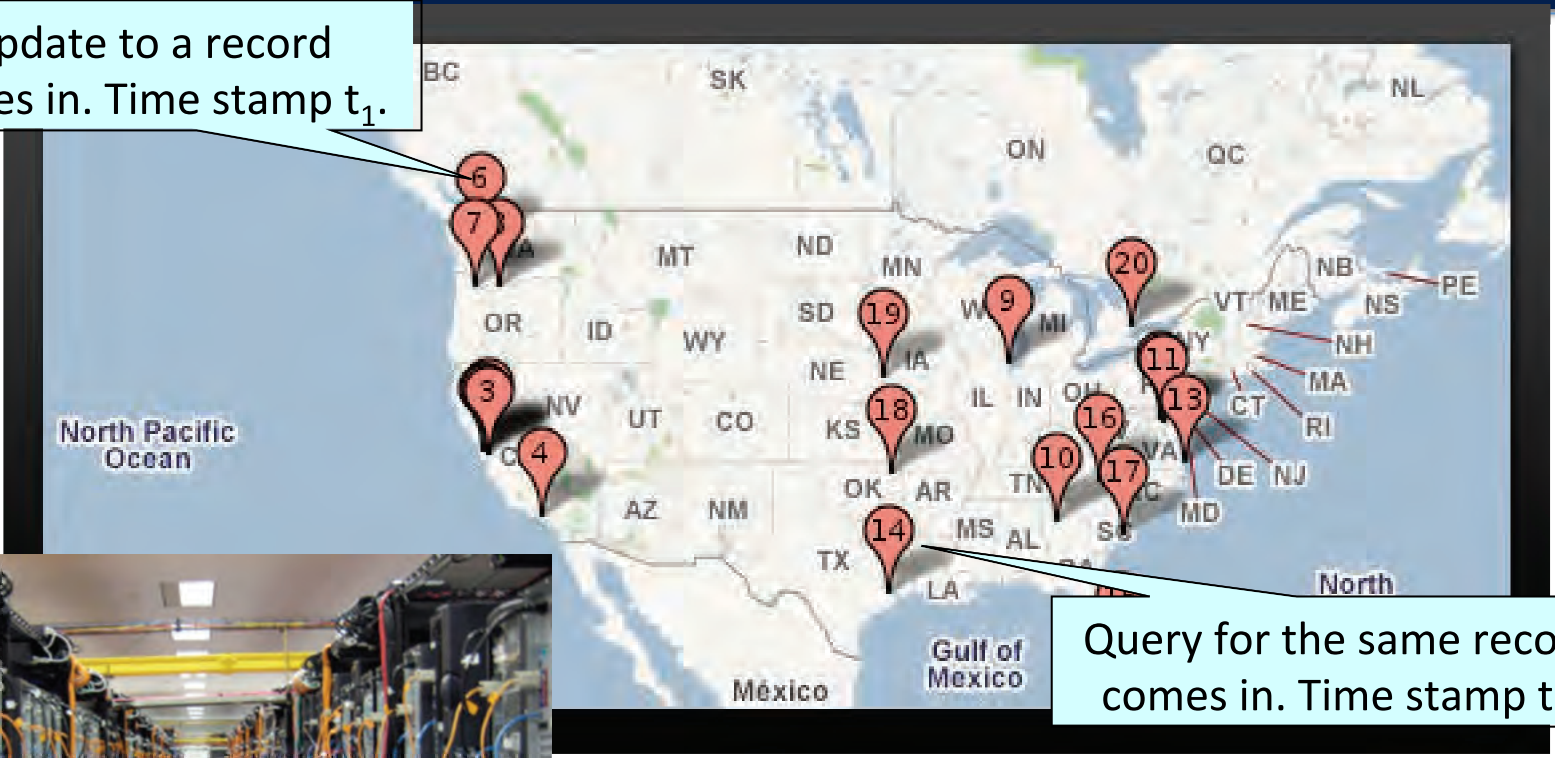
ADVANCES IN CAD FOR VLSI Volume 7
HARDWARE DESCRIPTION LANGUAGES


Introduction to Discrete Event Systems
Christos G. Cassandras
Stéphane Lafortune
Kluwer Academic Publishers




DISCRETE-EVENT SYSTEM SIMULATION
FOURTH EDITION
JERRY BANKS · JOHN S. CARSON II
BARRY L. NELSON · DAVID M. NICOL

Google Spanner – A Reinvention of PTIDES



Update to a record comes in. Time stamp t_1.

Query for the same record comes in. Time stamp t_2.

Distributed database with redundant storage and query handling across data centers.

Google Spanner – A Reinvention of PTIDES

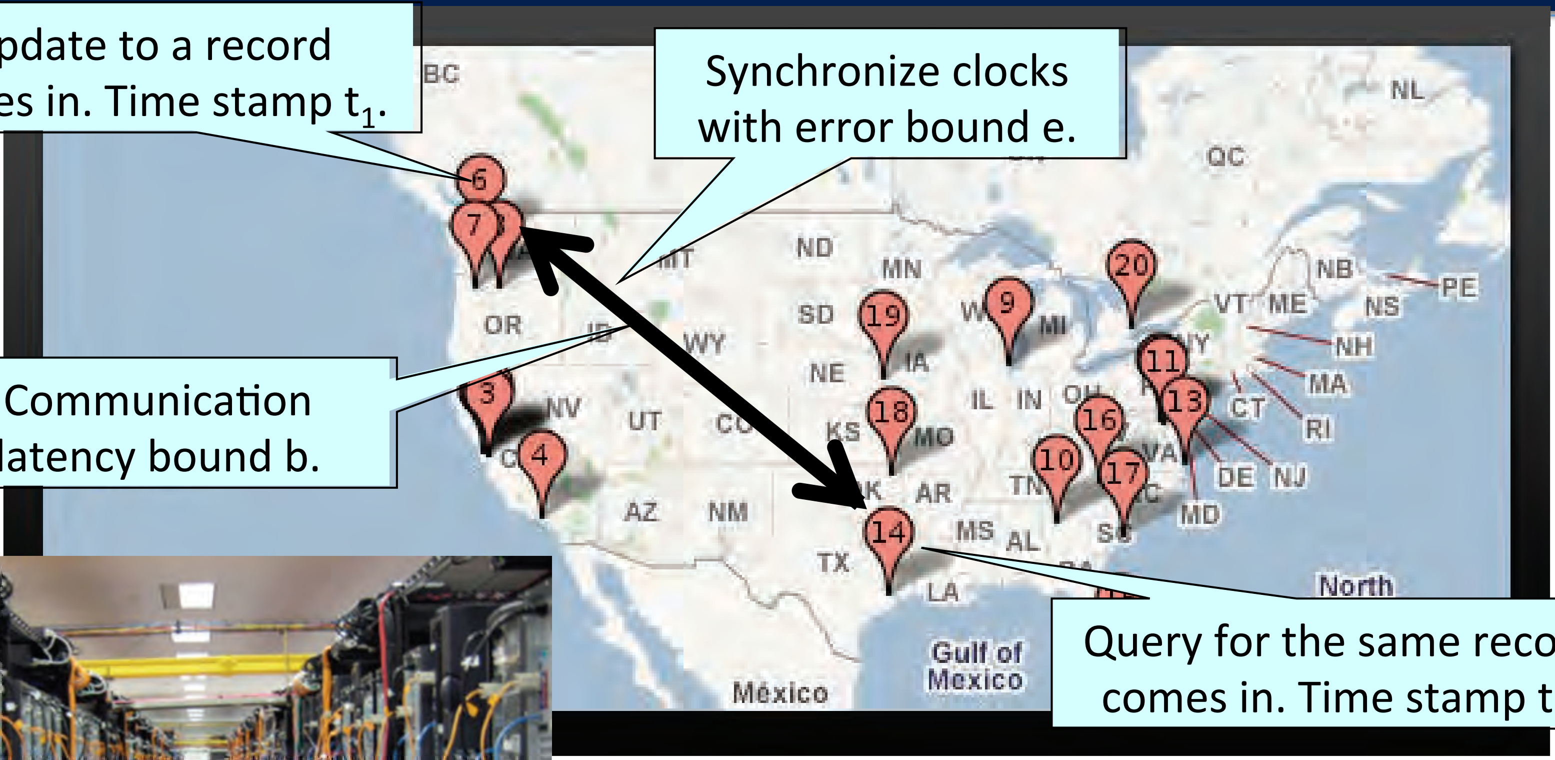
Update to a record comes in. Time stamp t_1.

BC
SK
NL
ON
QC
MT
ND
MN
NB
PE
OR
ID
WY
SD
MI
VT
ME
NS
NH
IA
NE
MA
North Pacific Ocean
NV
UT
CO
KS
MO
IL
IN
CT
RI
VA
DE
NJ
MD
AZ
NM
OK
AR
TN
MS
AL
TX
LA
North
Gulf of Mexico
México
3
4
6
7
9
10
11
13
14
16
17
18
19
20

Query for the same record comes in. Time stamp t_2.

If $t_2 < t_1$, the query response should be the pre-update value. Otherwise, it should be the post-update value.

Google Spanner: When to Respond?



Update to a record comes in. Time stamp t_1.

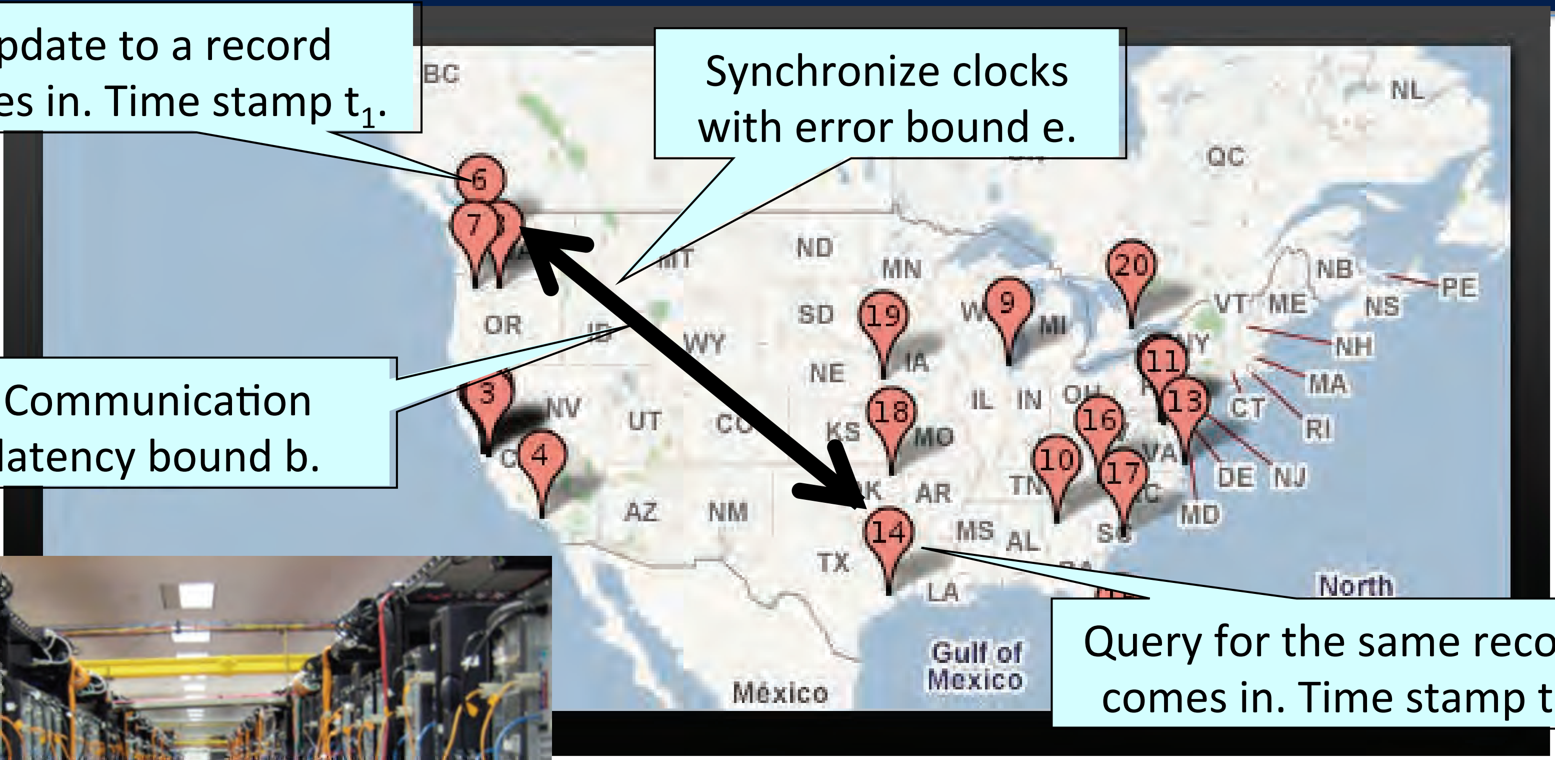
Synchronize clocks with error bound e.

Communication latency bound b.

Query for the same record comes in. Time stamp t_2.

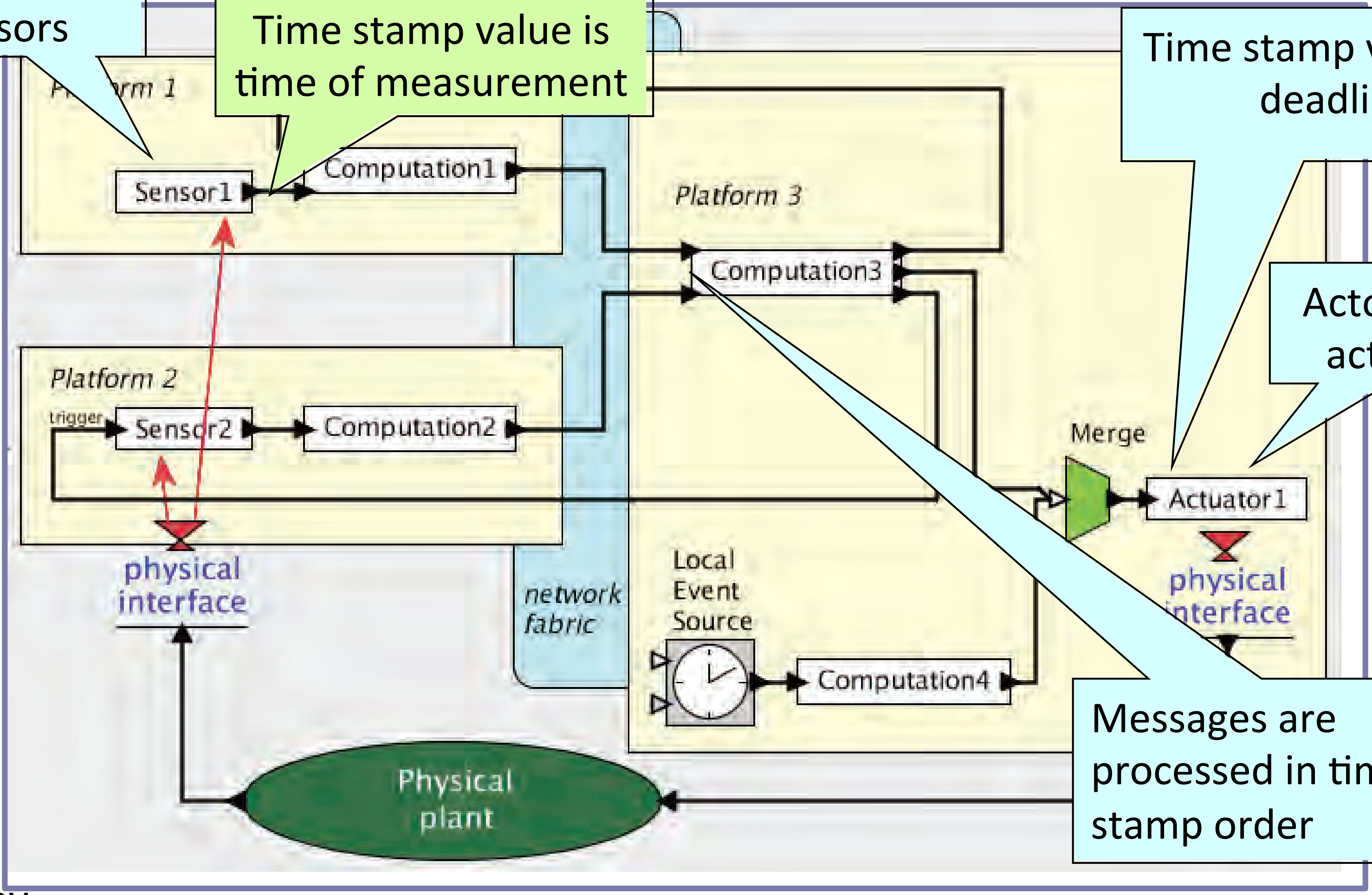
When the local clock time exceeds t_2 + e + b, issue the current record value as a response.

Google Spanner: Fault!


Update to a record comes in. Time stamp t_1.
Synchronize clocks with error bound e.
Communication latency bound b.
Query for the same record comes in. Time stamp t_2.
BC
NL
QC
ND
MN
MT
SD
WY
OR
ID
NE
IA
MI
IL
IN
OH
NV
UT
CO
KS
MO
AZ
NM
AR
TX
MS
AL
LA
TN
VT
ME
NB
NS
PE
NH
MA
CT
RI
DE
NJ
MD
Gulf of Mexico
México
North
6
7
3
4
19
18
14
9
20
11
13
16
10
17

If after sending a response, we receive a record update with time stamp $t_1 < t_2$ declare a fault. Spanner handles this with a transaction schema.

Ptides: Time stamps bind to real time at sensors and actuators



Actors wrap sensors

Time stamp value is time of measurement

Time stamp value is a deadline

Actors wrap actuators

Messages are processed in time-stamp order

Deterministic Distributed Real-Time

Assume bounds on:

- *clock synchronization error*
- *network latency*

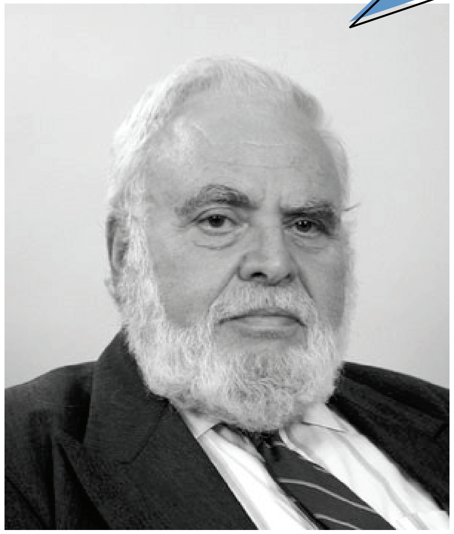
then ***events are processed in time-stamp order*** at every component. If in addition we assume

- *bounds on execution time*

then events are delivered to actuators on time.

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

So Many Assumptions?



All of the assumptions are achievable with today's technology, and are **requirements** anyway for hard-real-time systems. The Ptides model makes the requirements explicit.

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

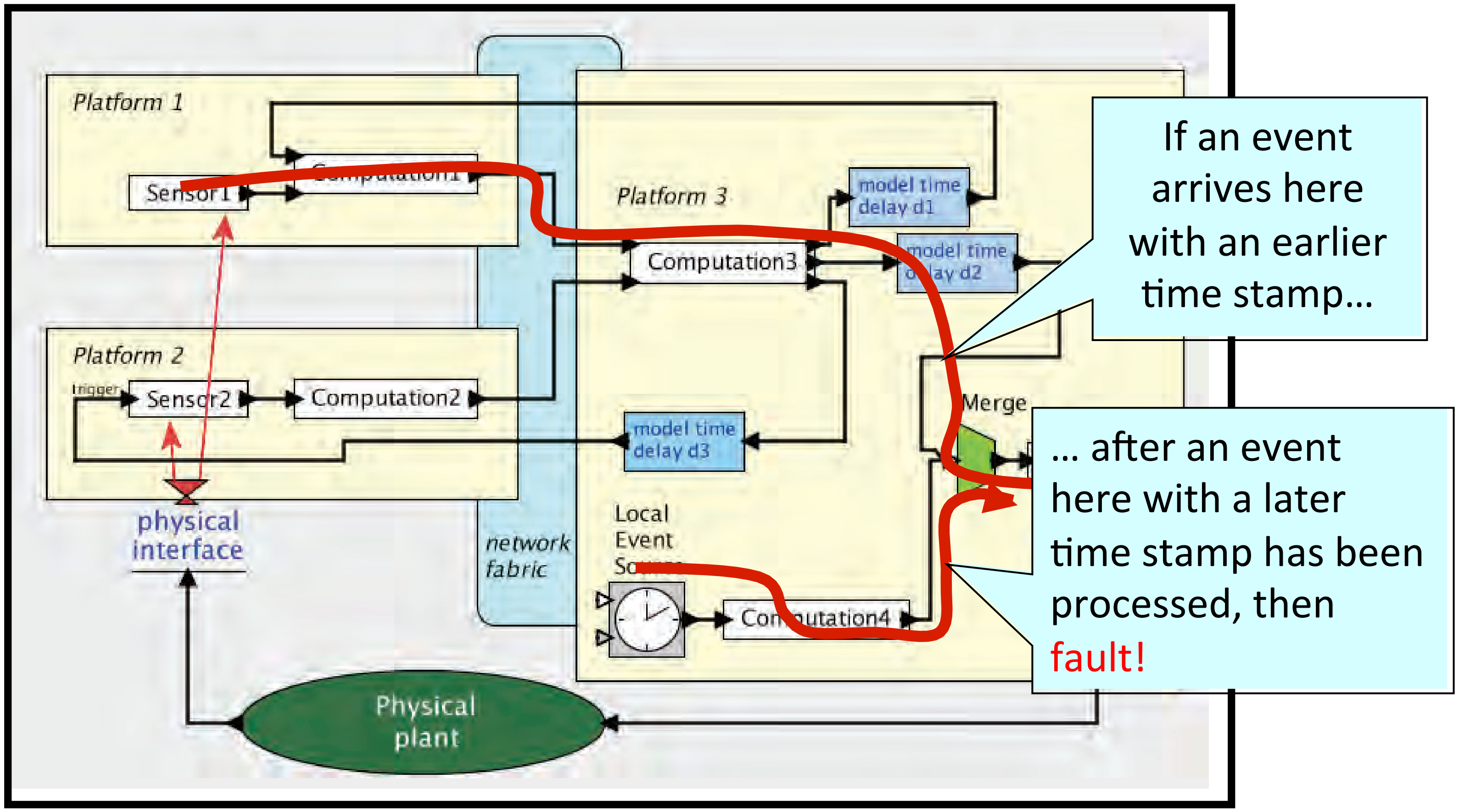
Lee, Berkeley

Non-Synchronized Clocks

Handling Faults

A fault manifests as out-of-order events.

Occurrence of a fault implies one or more of the assumptions was violated.



Conclusion https://accessors.org

- IoT is not so new. CPS is the essential problem.
- IoIT is a really interesting problem area.
- Modern concurrency models are useful:
  - Asynchronous atomic callbacks
  - Actors
- They can be combined (accessors)
- But for IoIT, they beg for more determinism
- PTIDES shows that deterministic models for distributed CPS applications are practical.

iCyPhy: The Home for this Research
Industrial Cyber-Physical Systems Center

ICyPhy is a university-industry partnership to pursue pre-competitive research on design, modeling, and analysis techniques for cyber-physical systems, with emphasis on industrial applications. Topics:

- Hardware and software architectures
- Model-based design for CPS
- Verification, validation, and certification
- Highly dynamic networked systems
- The Internet of things (IoT)
- Safety, privacy, and security
- Synthesis and learning
- Localization and location-aware services
- Learning and optimization
- Safety-critical systems
- Human-in-the-loop systems.
- Systems-of-systems design
- Semantics of timed systems

http://icyphy.org

Prabal Dutta, Edward Lee, Alberto Sangionvanni-Vincetelli, Sanjit Seshia