Accessors
A Software Architecture for IoT

Edward A. Lee and Christopher Brooks
EECS, UC Berkeley

Tutorial on Tools for Real-Time Systems (TuToR)
In conjunction with the Real-Time Systems Symposium (RTSS)
December 5, 2017 – Paris, France
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.
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.
// 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.
Streaming requests:

This is a key property of accessors, a design pattern for IoT that embraces concurrency, asynchrony, and atomicity.
Principled Composition of Devices and Services

- A **Swarmlet host** is a platform for component-based design (analogous to a browser, but for things, not people).
- A **Swarmlet host** can instantiate **accessors**, which serve as local proxies for devices and services.
- **Accessors** can be provided by device vendors or third parties.
- **Accessors** enable interoperability of **independently designed** components.
We are not alone pursuing this approach

Notable efforts:

• Node Red (IBM)
• Calvin (Ericsson)

Demo: A Service

This model realizes a traffic light. It starts a web socket server and sends on it the time remaining until the next change in the light. If it receives a JSON input with a field "green" with value true, then it requests an immediate change if the light is red.

Traffic Light Model

Connection to the client

Retrieve the service’s IP address

Retrieve an accessor for the service

Customize the accessor with the IP

Tag to recognize the service

Publish the accessor for the service

Traffic Light Animation
Traffic Light Model

counter: 59

Cinit

- guard: true
- output: Cred=1; Cyel=0; Cgrn=0; count = redTime
- set: counter = redTime

Cred

- guard: Sec_isPresent && counter > 0
- output: count = counter
- set: counter = counter - 1

Cyel

- guard: Sec_isPresent && counter == 0
- output: Cred = 1; Cyel = 0;
- count = redTime
- set: counter = redTime

Cgrn

- guard: Sec_isPresent && counter == 0
- output: Cred = 0; Cgrn = 1;
- count = greenTime
- set: counter = greenTime

Sec

requestGreen

- guard: Sec_isPresent && counter > 0
- output: Cyel = 1; Cgrn=0;
- count = yellowTime
- set: counter = yellowTime
Demo – Augmented Reality

- **Script that controls selection of objects**
- **Mutable accessor that reifies accessor for selected object**
- **Accessor for local UI service (browser or other HTML5 UI)**
- **_accessor for device connected to the host**
- **Accessor for a service that recognizes objects in the field of view**
- **Composite script/accessor that asks the local edge computer for accessors for objects in the local environment**
- **Scripts that construct HTML + images for user interface**

TerraSwarm Research Center
Exactly the same model being used at the TerraSwarm annual meeting to access a variety of sensors and actuators.
Better User Interfaces

Audio spectrum  Radio spectrum

User interface is based on HTML 5. Supports extensive design.

Visualization interface by Beth Osyk
Tagless Odometry

Using object recognition, GPS, inertial sensors, WiFi fingerprinting, etc., we can hope to get rid of the tags.
Local Devices/Services
Demonstrated at TerraSwarm Meeting

- BLEE sensors via MQTT (local comm.)
- BLEE sensors via GDP (global comm.)
- Powerblade sensors via MQTT (local comm.)
- Powerblade sensors via GDP (global comm.)
- Hue lightbulb via HTTP (local comm.)
- Robot via ROS Bridge (local comm.)
- Audio service via WebSocket (local comm.)
- MessageBoard via HTTP (local comm.)
CapeCode: Swarmlet Development
Our Programming Framework for the IoT

CapeCode leverages Ptolemy II, which provides a visual editor, strong type system, and lots of other infrastructure.
Code generator produces JavaScript files that can run on a lightweight JavaScript host.
Deploying Swarmlets
Write Once, Run Everywhere?

Java 8 JavaScript - Nashorn

Ptolemy II

node.js

((o) Duktape

APACHE CORDOVA™
Accessor Architecture Version 1.0
http://accessors.org

Base Accessor

Accessor Interface

Accessor

Accessor Host
Nashorn + Ptolemy II

Accessor Host
Cordova

Accessor Host
Node.js

Accessor Host
Browser

Module API Definition
CommonJS + Text

Module Implementation
JavaScript + Java (Nashorn)

Module Implementation
JavaScript + iOS/Android

Module Implementation
JavaScript (Node.js)

Module Implementation
JavaScript

requires

extends

implements

requires

runs in

runs in

runs in
Key challenge: Many accessors require modules that cannot be supported in a browser due to security constraints.
Install and invoke:

```
svn co https://repo.eecs.berkeley.edu/svn-anon/projects/terraswarm/accessors/trunk/accessors
cd accessors/web.hosts/node
node nodeHostShell.js
```

Run the Node.js host:

```
Welcome to the Node swarmlet host (nsh). Type exit to exit, help for help.
nsh> var a = instantiate('a', 'test/TestGain')
Reading accessor at: /ptII/org/terraswarm/accessor/accessors/web/test/TestGain.js
Instantiated accessor a with class test/TestGain
undefined
nsh> a.initialize()
undefined
nsh> a.setParameter('gain', 4)
undefined
nsh> a.provideInput('input', 5)
undefined
nsh> a.react()
TestGain: inputHandler: input: 5 gain: 4
undefined
nsh> a.latestOutput('scaled')
20
nsh> exit
exit
```
Challenges

- Multi-platform strategy
  - Maintaining compatible modules across hosts
- Regression tests
  - Need dummy devices and services
- Brittle designs
  - APIs for remote services 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.
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 Sangiovanni-Vincetelli, Sanjit Seshia