ThingsJS: Towards a Flexible and Self-Adaptable Middleware for Dynamic and Heterogeneous IoT Environments

Middleware for IoT – m4iot@Middleware 2017

Julien Gascon-Samson, Mohammad Rafiuzzaman, Karthik Pattabiraman

University of British Columbia
Department of Electrical and Computer Engineering
Vancouver, Canada

December 11, 2017
World of IoT growing at a very fast pace!
World of IoT growing at a very fast pace!

Traditionally, processing was done in the cloud.
Motivation

- World of IoT growing at a very fast pace!
- Traditionally, processing was done in the cloud
- Emerging trend: running applications on the IoT devices themselves (edge)
  - Performance, costs, reliability
Goals and Motivation

- ThingsJS: a framework for developing and deploying *high-level* applications on IoT devices (edge computing)
Goals and Motivation

- **ThingsJS**: a framework for developing and deploying *high-level* applications on IoT devices (edge computing)
- Programmers are typically more productive in higher-level languages
- JavaScript: strong user base
Goals and Motivation

- ThingsJS: a framework for developing and deploying high-level applications on IoT devices (edge computing)
- Programmers are typically more productive in higher-level languages
- JavaScript: strong user base
Goals and Motivation

- **ThingsJS**: a framework for developing and deploying *high-level* applications on IoT devices (edge computing)

- Programmers are typically more productive in higher-level languages
- **JavaScript**: strong user base

**JavaScript VMs on IoT**
- Samsung IoT.js
- Intel XDK
- DukServer
- Smart.js
- Node.js on IoT devices
Goals and Motivation

Constraints

- IoT world is highly heterogeneous!
  - Different hardware platforms
  - OSes
  - Environments
- ThingsJS: Declarative language for expressing constraints
  - Over the devices
  - Over the applications

- Programmers are typically more productive in higher-level languages
- JavaScript: strong user base

JavaScript VMs on IoT

- Samsung IoT.js
- Intel XDK
- DukServer
- Smart.js
- Node.js on IoT devices
Goals and Motivation

ThingsJS: a framework for developing and deploying *high-level* applications on IoT devices (edge computing)

1. **Scheduling Applications on IoT Devices**

Scheduling

- Given a set of IoT applications (“components”)
- Given a set of constraints

Significant work on scheduling applications in the cloud – idea of scheduling applications in the edge is relatively novel
Motivation

ThingsJS

Scheduling

Code Migration

Communications

Goals and Motivation

- ThingsJS: a framework for developing and deploying high-level applications on IoT devices (edge computing)
  - **Scheduling Applications on IoT Devices**
  
  - **Scheduling**
    - Given a set of IoT applications ("components")
    - Given a set of constraints
      
        ↓

    - What is the optimal mapping of components to devices?
    - Significant work on scheduling applications in the cloud – idea of scheduling applications *in the edge* is relatively novel
Goals and Motivation

- ThingsJS: a framework for developing and deploying *high-level* applications on IoT devices (edge computing)

1. Scheduling Applications on IoT Devices

Example: motion detection

- `video_camera`: tied to hardware
- `motion_detect`: detects motion
Goals and Motivation

- **ThingsJS**: a framework for developing and deploying *high-level* applications on IoT devices (edge computing)

  1. **Scheduling Applications on IoT Devices**

  - **video_camera**: tied to hardware
  - **motion_detect**: detects motion
  - Some components can/should be run in the cloud

Example: motion detection
Goals and Motivation

- **ThingsJS**: a framework for developing and deploying *high-level* applications on IoT devices (edge computing)
  1. Scheduling Applications on IoT Devices
  2. **Migrating IoT Applications**

Migration

- Conditions change over time
- IoT devices are resource-constrained
Goals and Motivation

- **ThingsJS**: a framework for developing and deploying *high-level* applications on IoT devices (edge computing)
  1. Scheduling Applications on IoT Devices
  2. Migrating IoT Applications

**Migration**
- Conditions change over time
- IoT devices are resource-constrained
- It might be necessary to *migrate* components

Prior work: migrating web applications across browsers
Goals and Motivation

- **ThingsJS**: a framework for developing and deploying *high-level* applications on IoT devices (edge computing)
  1. Scheduling Applications on IoT Devices
  2. **Migrating IoT Applications**

**Migration**

- Conditions change over time
- IoT devices are resource-constrained
- It might be necessary to *migrate* components
- Prior work: migrating web applications across browsers
Goals and Motivation

- **ThingsJS**: a framework for developing and deploying *high-level* applications on IoT devices (edge computing)

1. Scheduling Applications on IoT Devices
2. Migrating IoT Applications
3. Optimizing the Communications

**Publish/Subscribe - MQTT**

- Publish/subscribe maps well to IoT (MQTT: ISO standard)
Goals and Motivation

- **ThingsJS**: a framework for developing and deploying high-level applications on IoT devices (edge computing)
  1. Scheduling Applications on IoT Devices
  2. Migrating IoT Applications
  3. Optimizing the Communications

**Publish/Subcribe - MQTT**
- Publish/subscribe maps well to IoT (MQTT: iso standard)
- Significant work in cloud/p2p pub/sub
Goals and Motivation

- **ThingsJS**: a framework for developing and deploying *high-level* applications on IoT devices (edge computing)
  1. Scheduling Applications on IoT Devices
  2. Migrating IoT Applications
  3. Optimizing the Communications

**Publish/Subscribe - MQTT**
- Publish/subscribe maps well to IoT (MQTT: ISO standard)
- Significant work in cloud/p2p pub/sub
- Adapting pub/sub architectures for IoT
Goals and Motivation

ThingsJS: IoT Runtime Middleware

Dynamic Scheduling

JavaScript Code Migration

Inter-Component Communications
ThingsJS
ThingsJS Application

Source Code:
- High-Level Language (i.e., Javascript - Node.js)
- Code written in terms of “components”

Constraints:
- Physical: device-related
- Logical: component-related
ThingsJS Application

Source Code:
- High-Level Language (i.e., Javascript - Node.js)
- Code written in terms of “components”

Constraints:
- Physical: device-related
- Logical: component-related

Constraints:
- CPU (workload units)
- RAM
- Available incoming & outgoing bandwidth
ThingsJS Application

Source Code:
- High-Level Language (i.e., Javascript - Node.js)
- Code written in terms of “components”

Constraints:
- Physical: device-related
- Logical: component-related

- Workload units
- RAM
- Incoming & outgoing bandwidth
- Inter-component constraints: latency, bandwidth
Dynamic Scheduling

1. Goals and Motivation
2. ThingsJS: IoT Runtime Middleware
3. Dynamic Scheduling
4. JavaScript Code Migration
5. Inter-Component Communications
1. Predicting the workload of components

   - Machine-learning
   - Training:
     - Several devices, different load profiles
     - Monitoring performance (execution time)
     - Construction of a model
   - Predicting:
     - Execution time on device
     - With a specific load level

2. Scheduling the placement of components to devices
Scheduling Applications on Things (2)

1. Predicting the workload of components
2. **Scheduling the placement of components to devices**

Given a set of constraints
...and the prediction model

- What is the *optimal* arrangement of components-to-devices?
- Respecting all constraints
- SMT Solver
- Most suitable *global* solution
- Rescheduling?
ThingsMigrate: Migrating JavaScript IoT Applications

1. Goals and Motivation
2. ThingsJS: IoT Runtime Middleware
3. Dynamic Scheduling
4. JavaScript Code Migration
5. Inter-Component Communications
Constraints

- Portability: heterogeneous devices, cloud (*cloud-edge* computing)
  - No modifications to VM
- *Stateful* applications
- Asynchronous nature of JS
Challenges and Approach

1. Closures
2. Timers
3. Asynchronous Model (Event-Based)

```javascript
function Counter(val) {
  var value = val;

  return function() {
    value = value + 1;
    // Can access parent function local variable
    return value;
  }
}

var f = Counter(5);
var g = Counter(2);
document.writeln(f()); // Prints 6
document.writeln(g()); // Prints 3
```
function Counter(val) {
    var value = val;

    return function() {
        value = value + 1;
        // Can access parent function local variable
        return value;
    }
}

var f = Counter(5);
var g = Counter(2);

document.writeln(f()); // Prints 6
document.writeln(g()); // Prints 3

1. Closures
2. Timers
3. Asynchronous Model (Event-Based)

1. Code Instrumentation
2. State Serialization
3. Code Reconstruction
Inter-Component Communications

1. Goals and Motivation

2. ThingsJS: IoT Runtime Middleware

3. Dynamic Scheduling

4. JavaScript Code Migration

5. Inter-Component Communications
Easy decoupling of *content producers* from *content consumers*

Abstraction of network-related considerations

```javascript
// ...

// Connect
pubsub.connect(function() {

    // Repeat every second
    setInterval(function() {

        // Read temperature from GPIO pin
        var temperature = GPIO.readPin(12);

        // Publish temperature
        pubsub.publish("smartsensor/temperature", {
            id: mySensorId,
            temperature: temperature
        });

    }, 1000);

});
```
**Easy decoupling of content producers from content consumers**

- Abstraction of network-related considerations

```javascript
// ...

// Connect
pubsub.connect(function() {

  // Subscribe to temperature messages
  pubsub.subscribe("smartsensor/temperature", function(d) {

    if (d.temperature > threshold) {
      pubsub.publish("smartsensor/actuation", {
        id: d.id,
        powerVariation: -5
      });
    }

    else if (d.temperature < threshold) {
      pubsub.publish("smartsensor/actuation", {
        id: d.id,
        powerVariation: 5
      });
    }

  });

})();
```
How should the pub/sub service be provided?

- In the cloud?
- Peer-to-peer (mesh)?
- Hybrid approaches?
  - Dynamic reconfiguration
- Other pub/sub paradigms: content-based, graph-based
Summary

- ThingsJS: IoT Runtime Middleware
- Publish/Subscribe: Inter-Component Communications
- Dynamic Scheduling
- Code Migration

Research Team:

- Professor Karthik Pattabiraman
- Julien Gascon-Samson, PhD – NSERC Post-Doctoral Fellow
- Kumseok Jung – Master’s Student
- Mohammad Rafiuzzaman – PhD Student

Resources:

- ThingsJS: http://thingsjs.juliengs.com
- GitHub: https://github.com/karthikp-ubc/ThingsJS