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

Middleware for IoT – m4iot@Middleware 2017

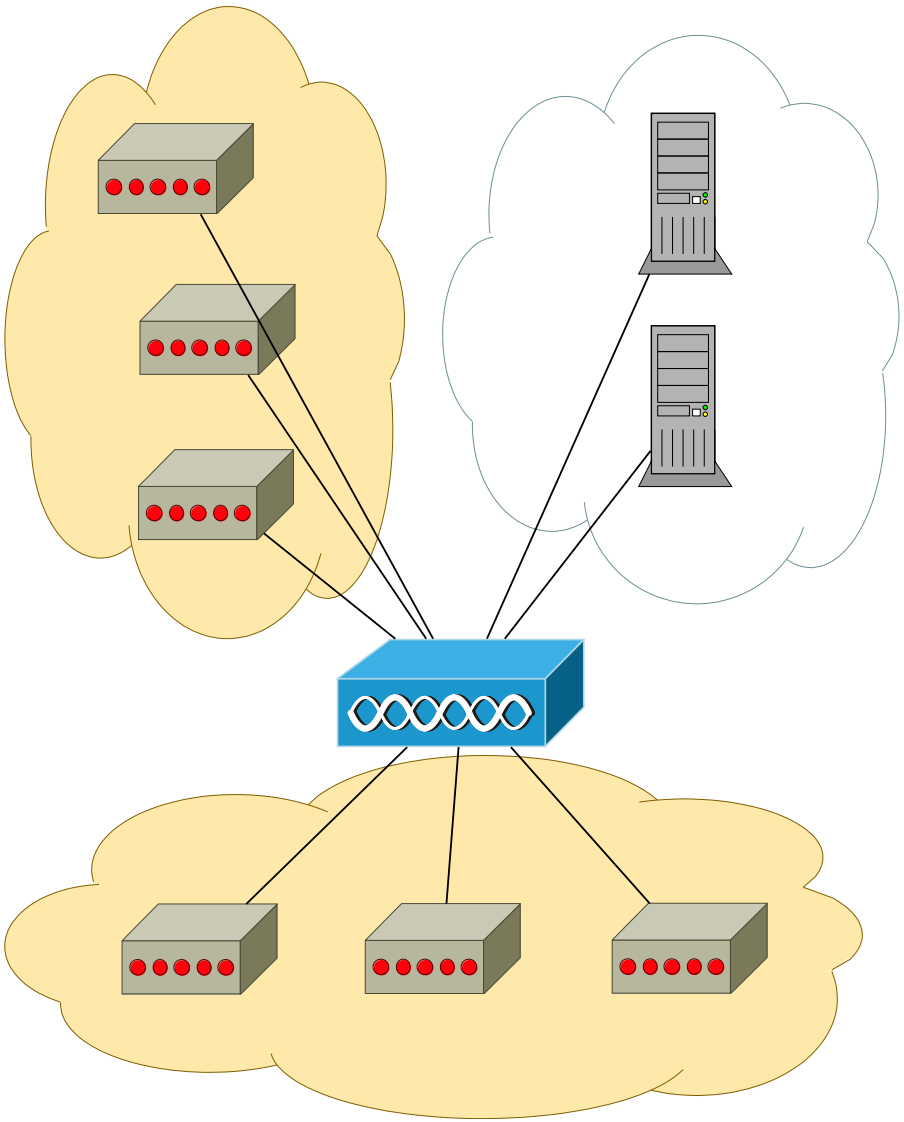
Julien Gascon-Samson, Mohammad Rafiuzzaman, Karthik Pattabiraman

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

Electrical and Computer Engineering

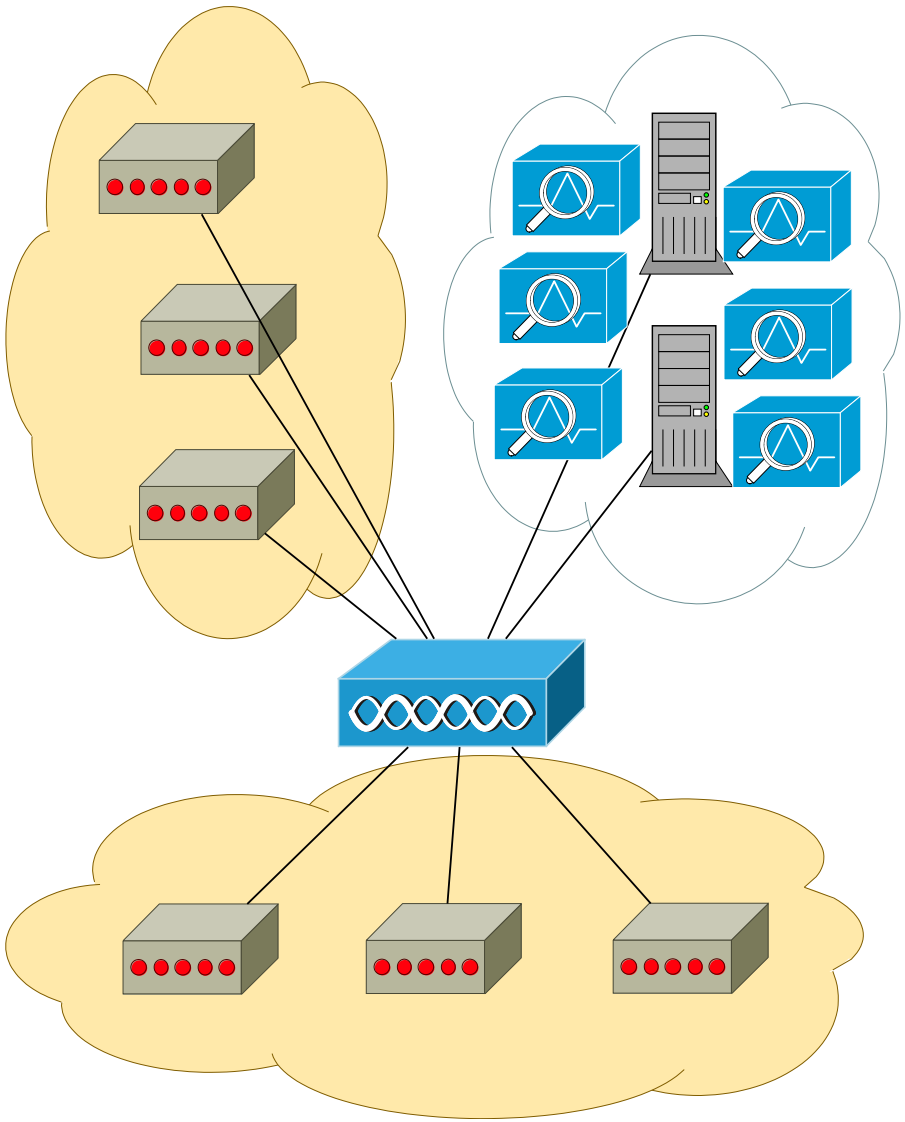
December 11, 2017

Motivation



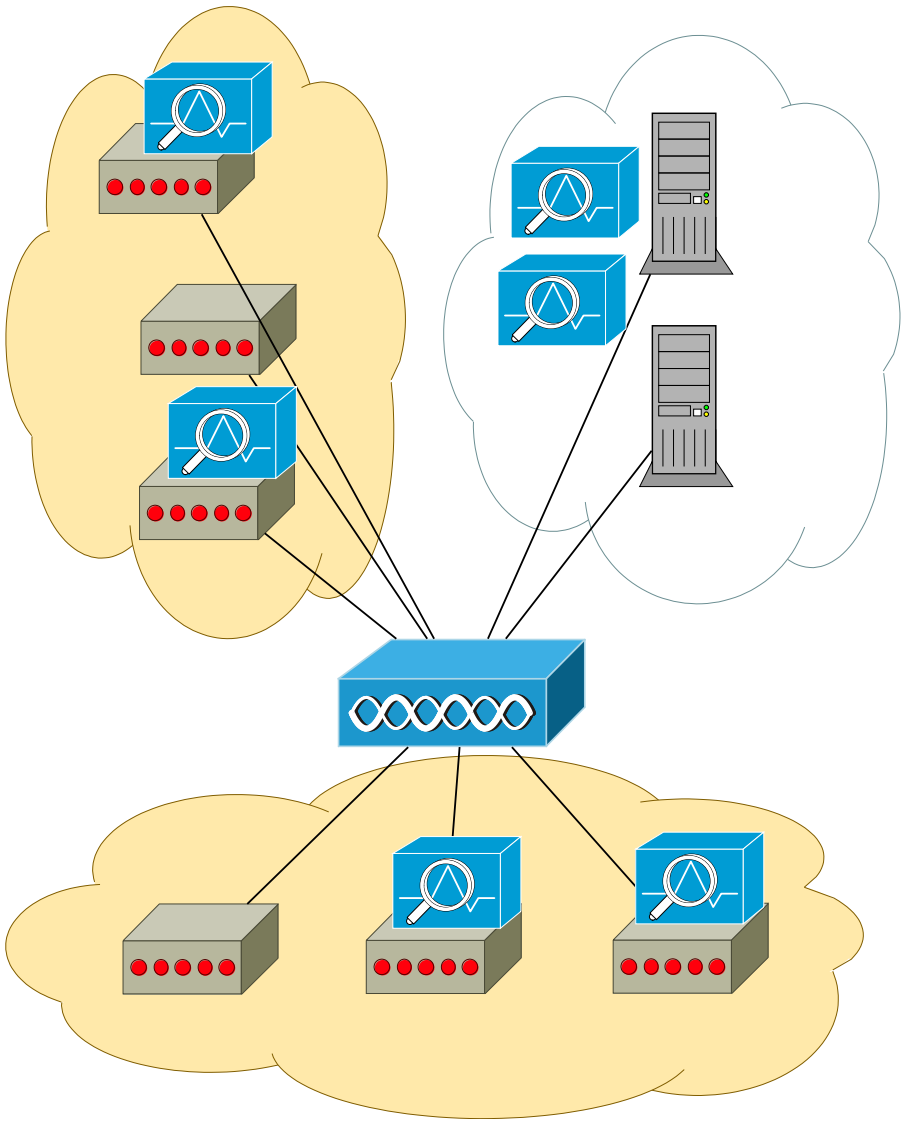
- World of IoT growing at a very fast pace!

Motivation



- World of IoT growing at a very fast pace!
- Traditionaly, processing was done in the cloud

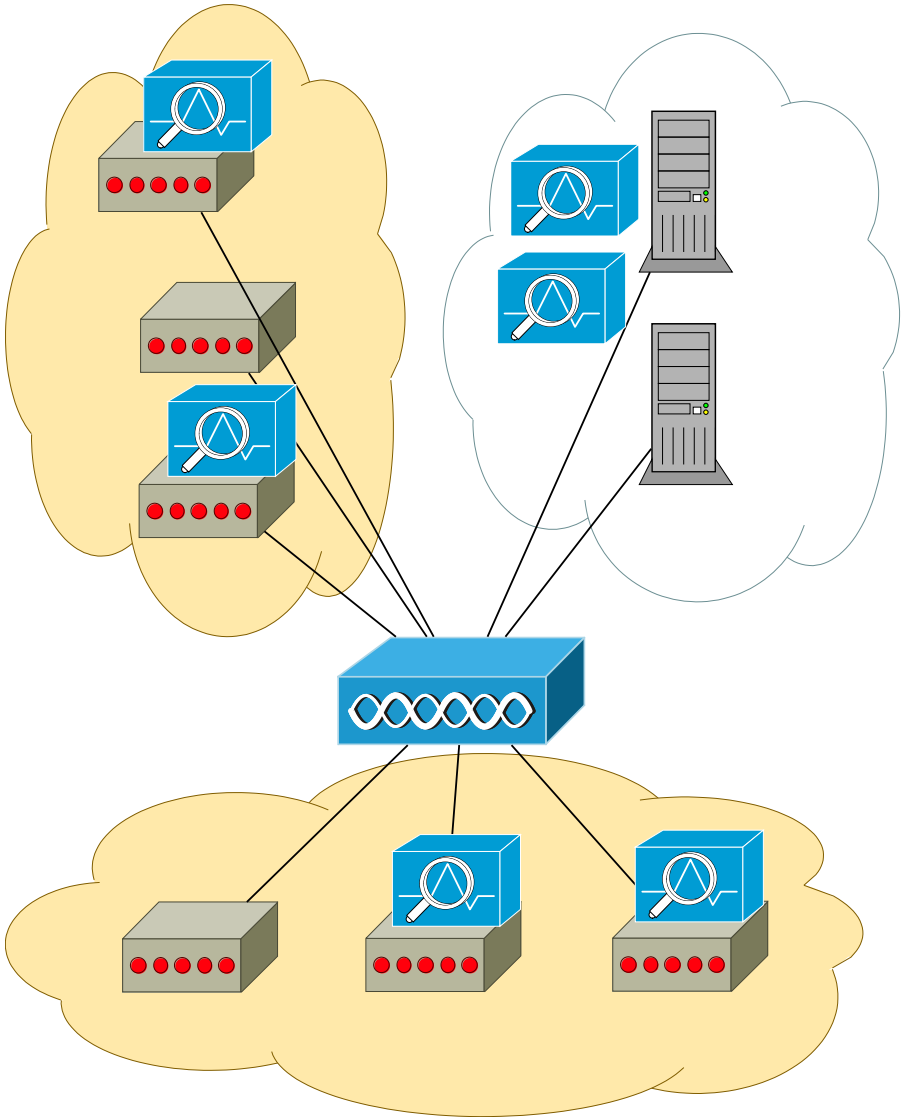
Motivation



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

Goals and Motivation

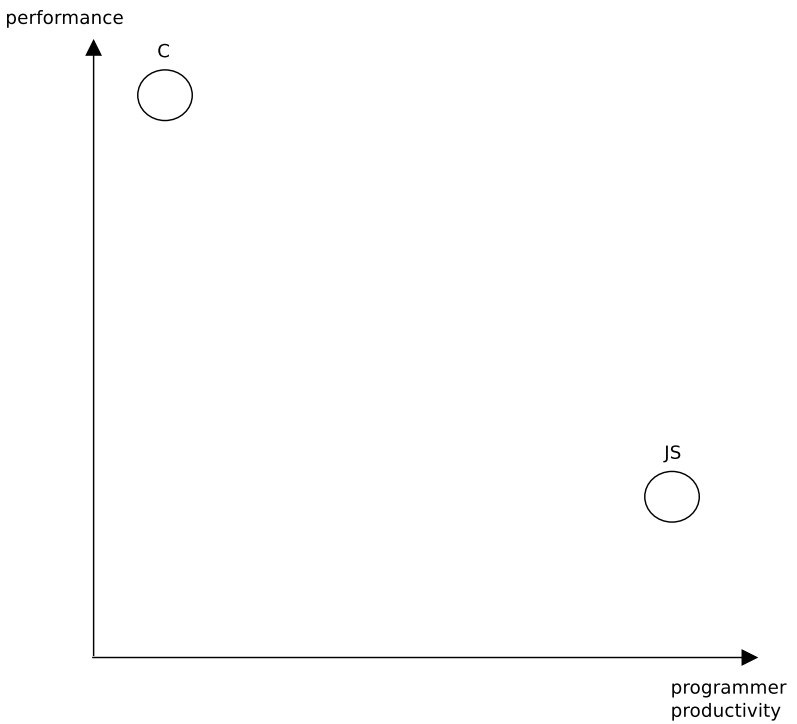
- ThingsJS: a framework for developing and deploying *high-level* applications on IoT devices (edge computing)



Goals and Motivation

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

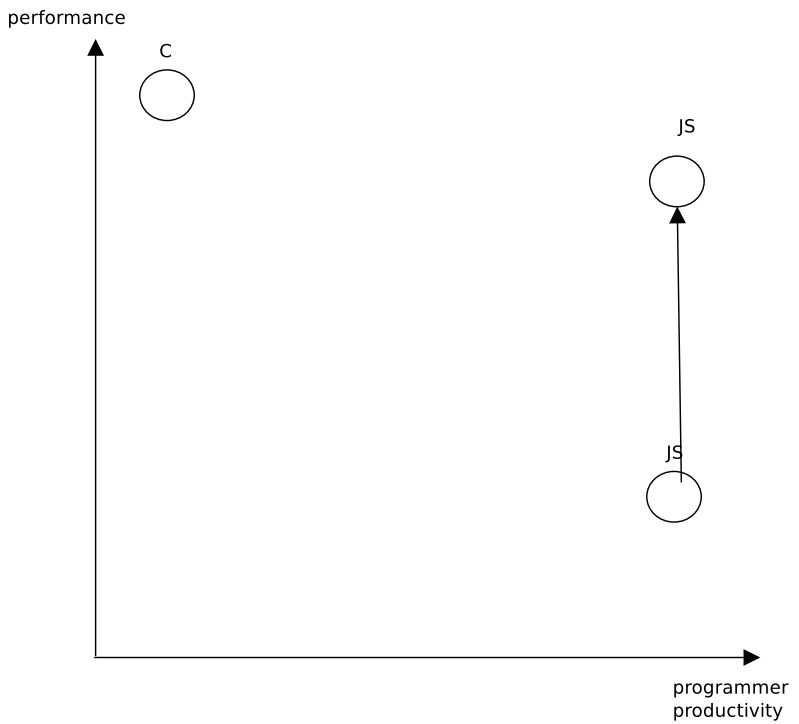
- Programmers are typically more productive in higher-level languages
- JavaScript: strong user base



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

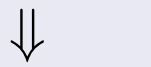
- Samsung IoT.js
- Intel XDK
- DukServer
- Smart.js
- Node.js on IoT devices

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



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

$$\Downarrow$$

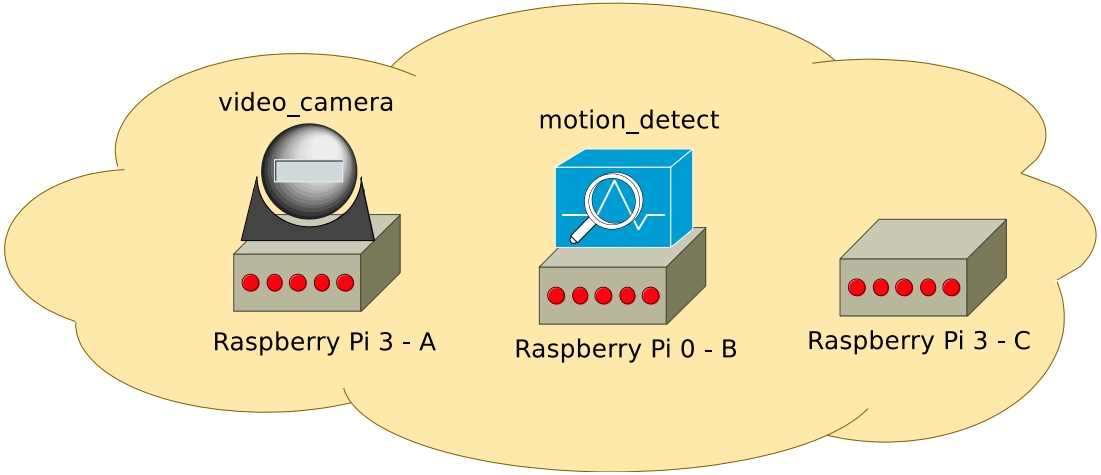
- What is the optimal mapping of components to devices?
- Significant work on scheduling applications in the cloud – idea of scheduling applications *in the edge* is relatively novel

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

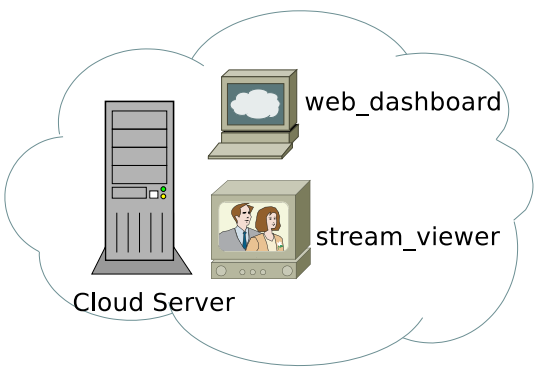


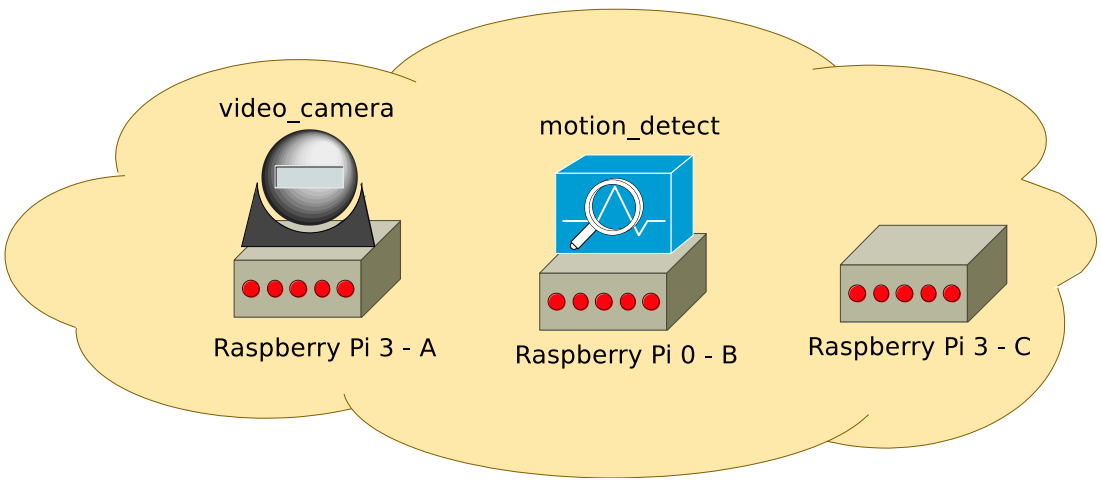
Goals and Motivation

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

Example: motion detection

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



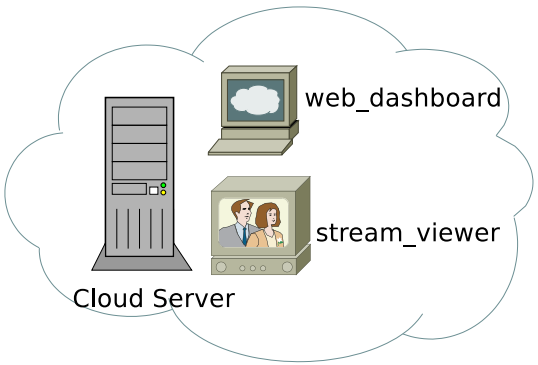


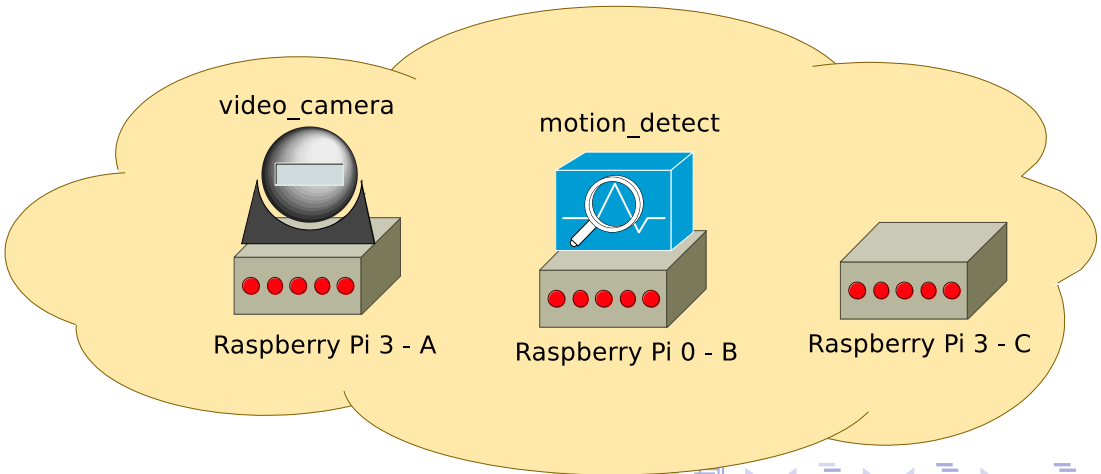
Goals and Motivation

- 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



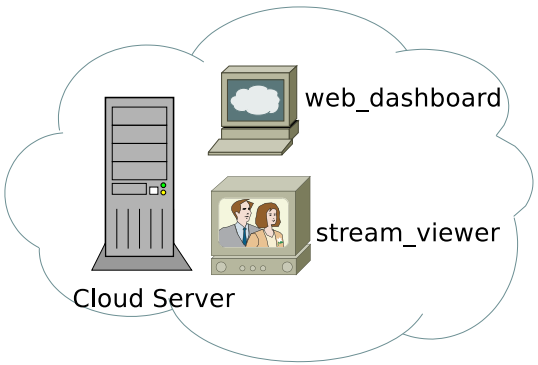


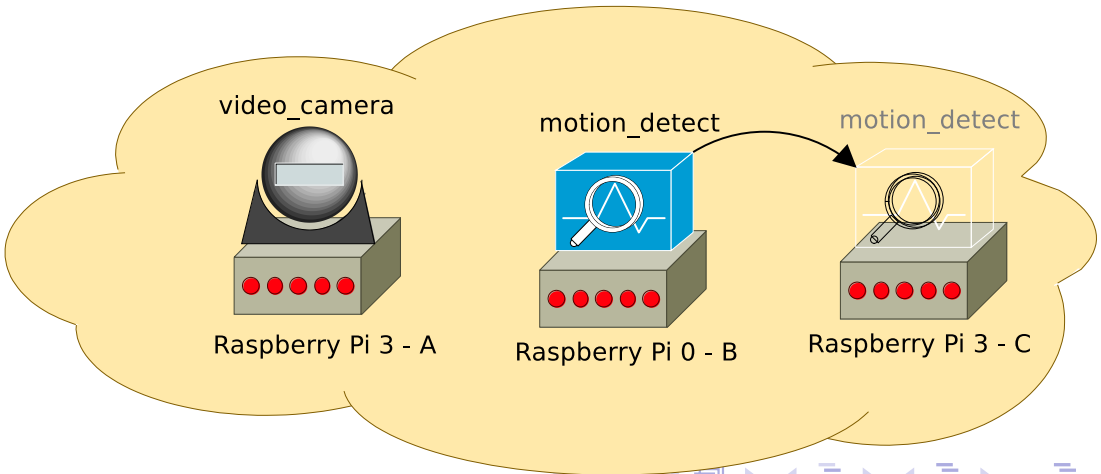
Goals and Motivation

- 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



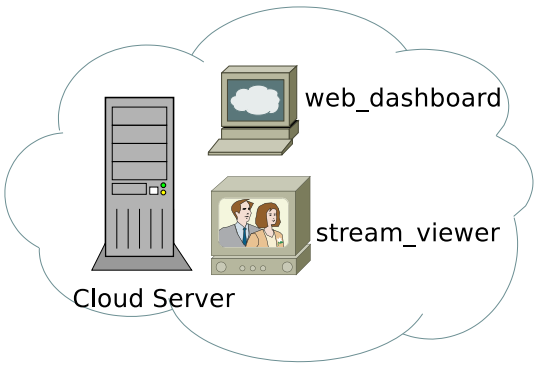


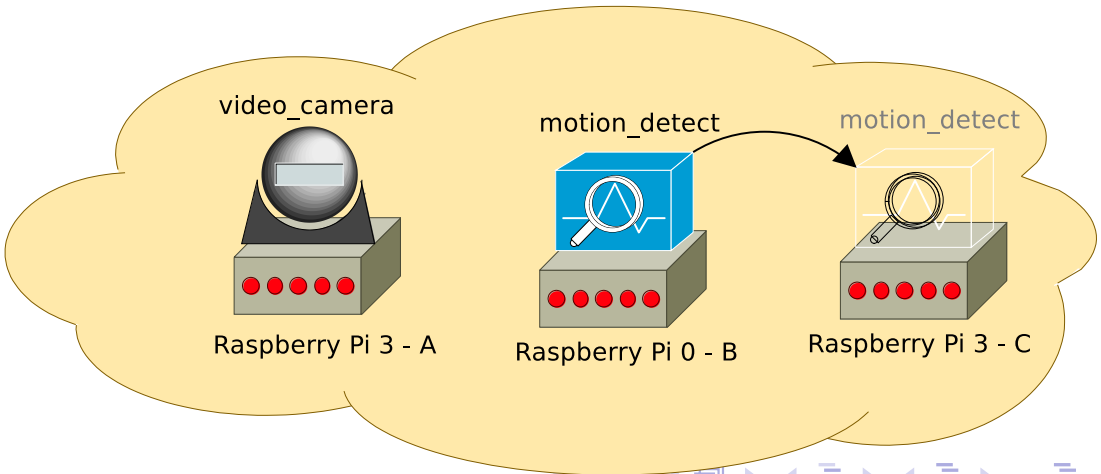
Goals and Motivation

- 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



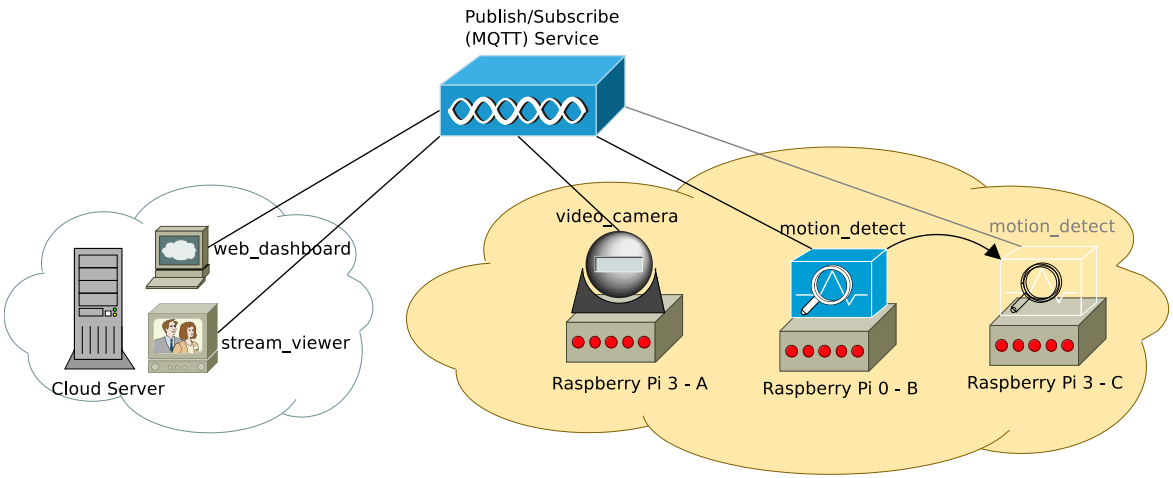


Goals and Motivation

- 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/Subcribe - MQTT

- Publish/subscribe maps well to IoT (MQTT: iso standard)

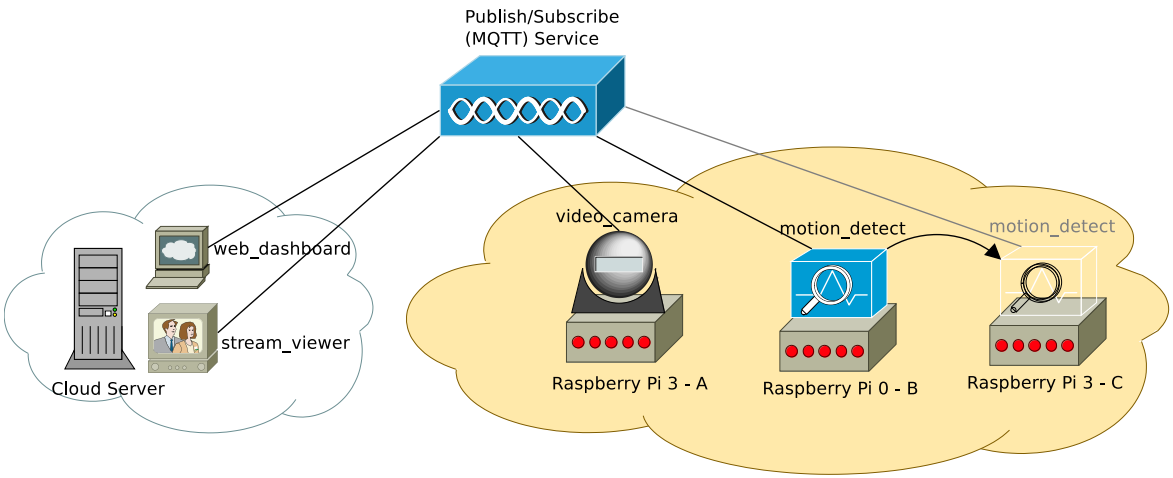


Goals and Motivation

- 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

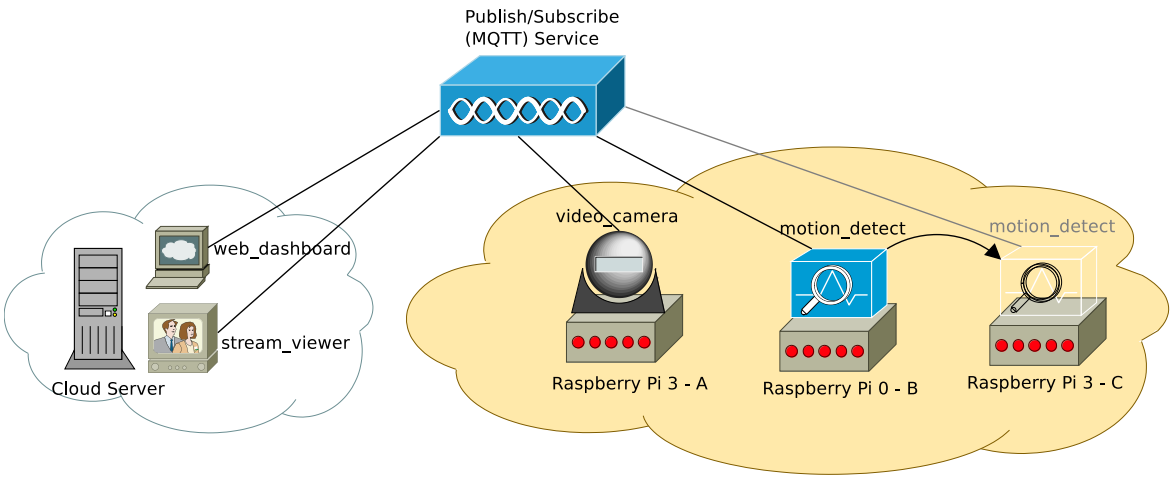


Goals and Motivation

- 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**

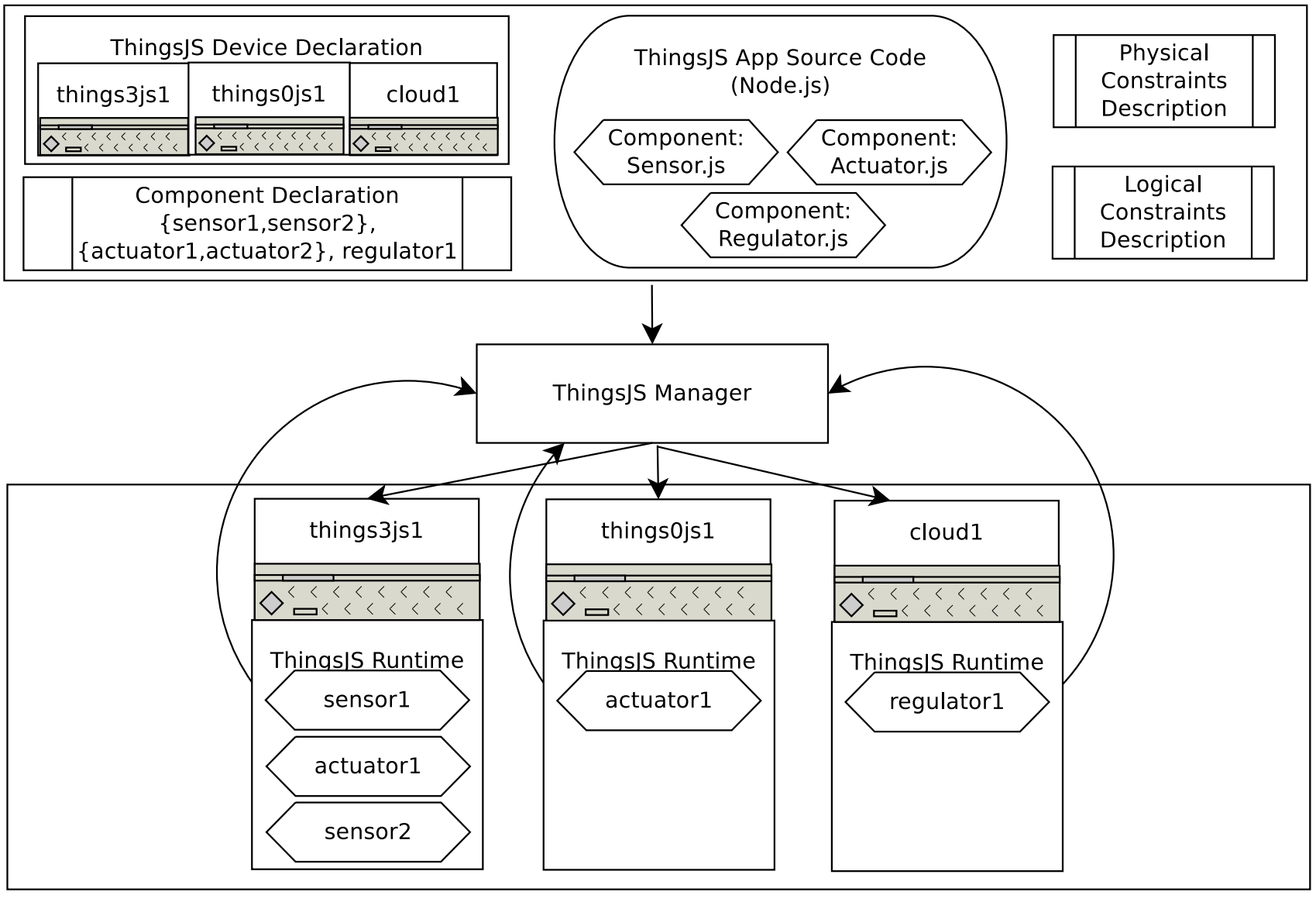
Publish/Subcribe - MQTT

- Publish/subscribe maps well to IoT (MQTT: iso standard)
- Significant work in cloud/p2p pub/sub
- Adapting pub/sub architectures for IoT

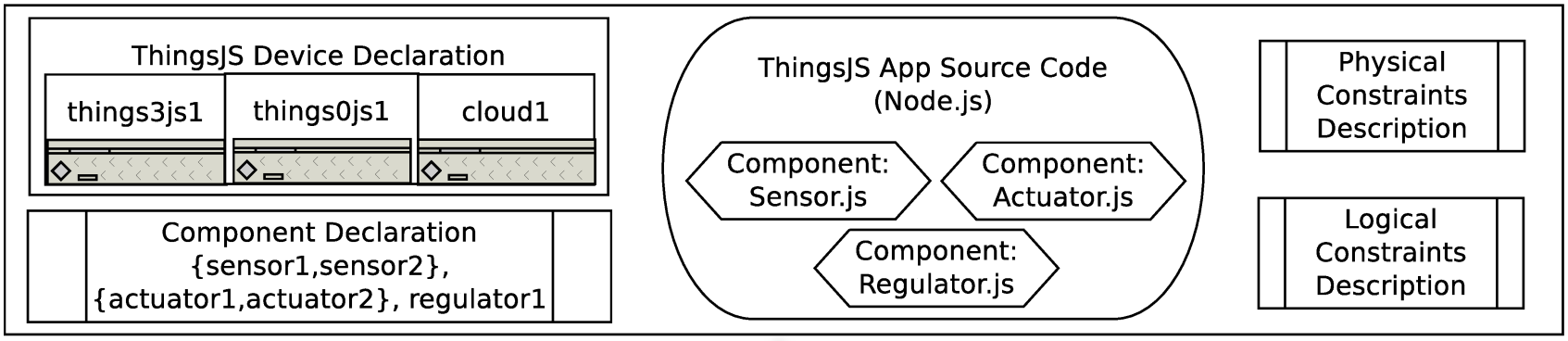


ThingsJS: IoT Runtime Middleware

ThingsJS


ThingsJS Device Declaration
things3js1
things0js1
cloud1
Component Declaration
{sensor1,sensor2},
{actuator1,actuator2}, regulator1
ThingsJS App Source Code
(Node.js)
Component:
Sensor.js
Component:
Actuator.js
Component:
Regulator.js
Physical
Constraints
Description
Logical
Constraints
Description
ThingsJS Manager
things3js1
things0js1
cloud1
ThingsJS Runtime
sensor1
actuator1
sensor2
ThingsJS Runtime
actuator1
ThingsJS Runtime
regulator1

ThingsJS Application



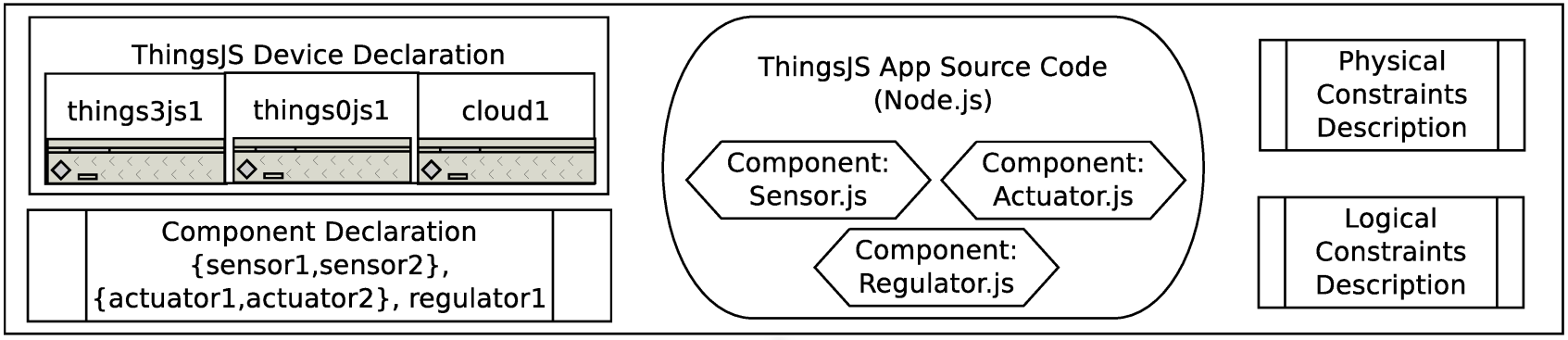
Source Code:

- 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:

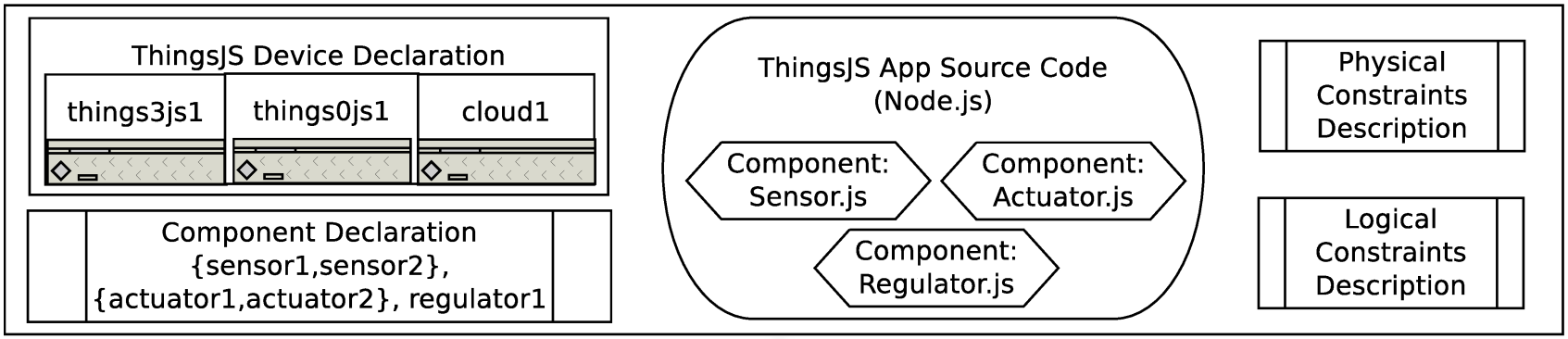
- High-Level Language (i.e., Javascript - Node.js)
- Code written in terms of "components"

Constraints:

- **Physical: device-related**
- Logical: component-related

- CPU (workload units)
- RAM
- Available incoming & outgoing bandwidth

ThingsJS Application



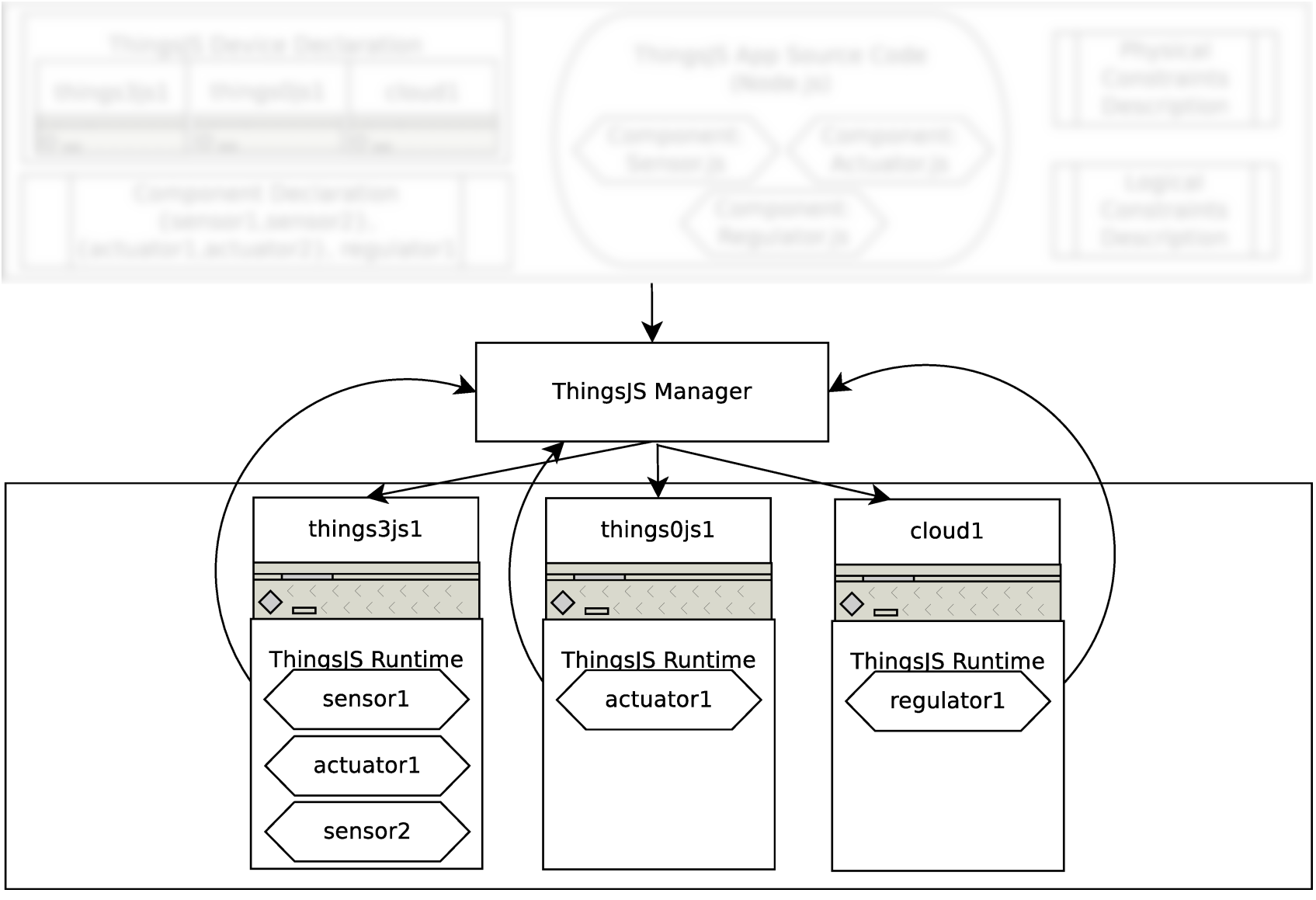
Source Code:

- 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

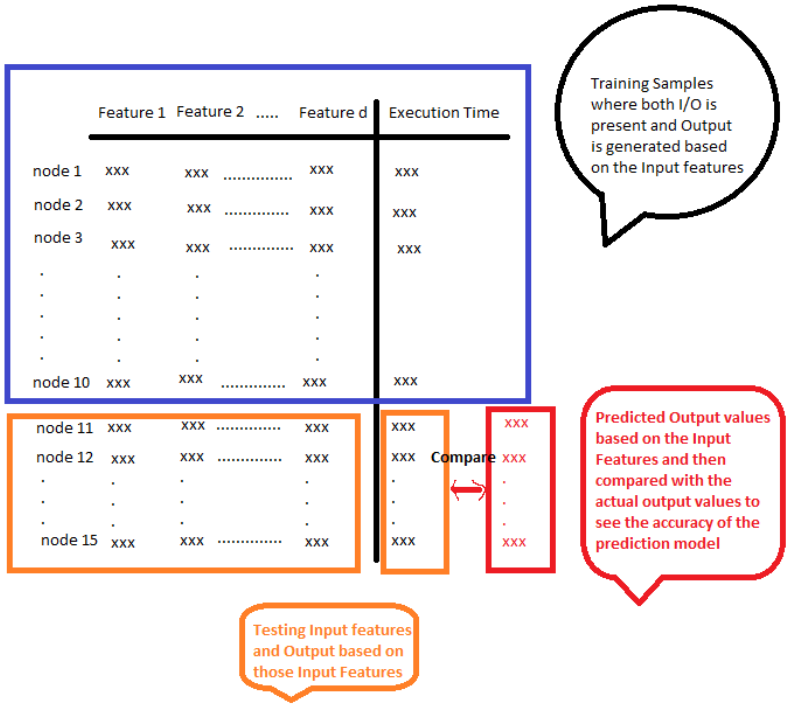
ThingsJS Middleware



Dynamic Scheduling

Scheduling Applications on Things (1)

1. **Predicting the workload of components**

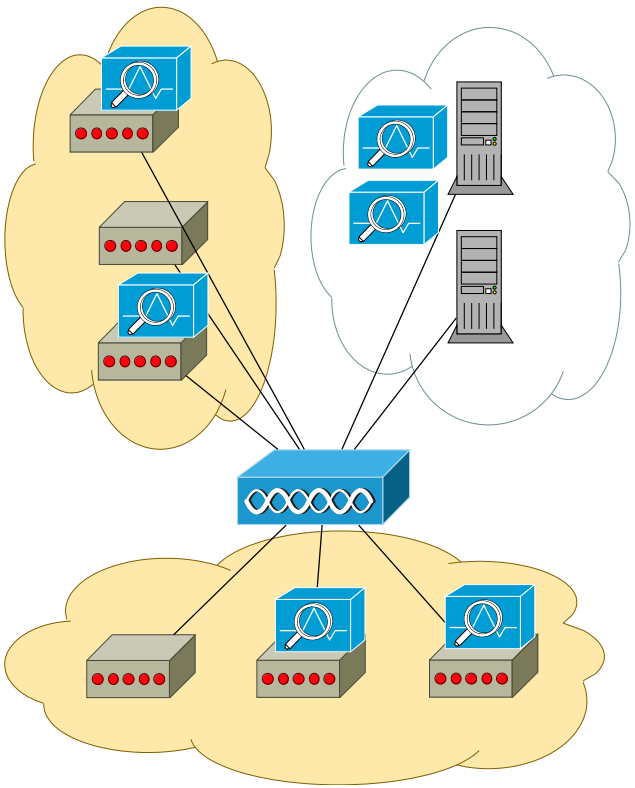


- Machine-learning
- Training:
  - Several devices, different load profiles
  - Monitoring performance (execution time)
  - Construction of a model
- Predicting:
  - Execution time on device
  - With a specific load level

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**

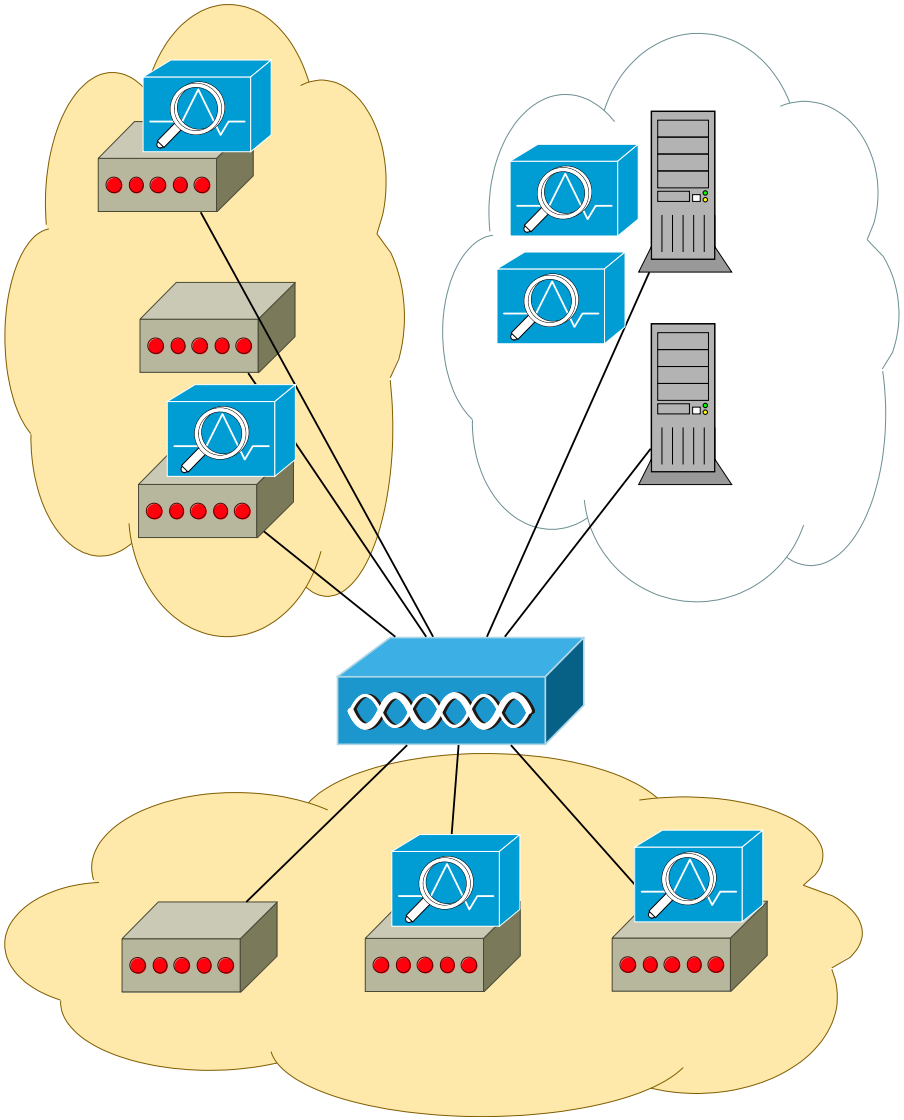


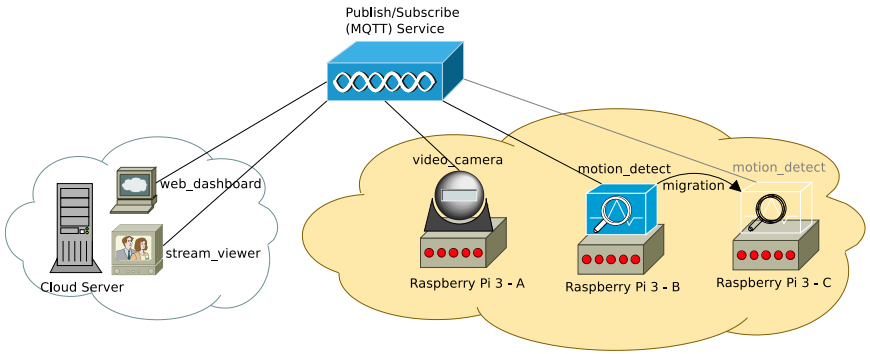
Given a set of constraints
...and the prediction model

- What is the *optimal* arrangement of components-to-devices?
- Respecting all constraints
- SMT Solver
- Most suitable *global* solution
- Rescheduling?

ThingsMigrate: Migrating JavaScript IoT Applications

Constraints





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

Challenges and Approach


```
1  function Counter(val) {
2      var value = val;
3
4      return function() {
5          value = value + 1;
6          // Can access parent function local
               variable
7          return value;
8      }
9  };
10
11 var f = Counter(5);
12 var g = Counter(2);
13
14 document.writeln( f() ); // Prints 6
15 document.writeln( g() ); // Prints 3
```

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

Challenges and Approach


```
1  function Counter(val) {
2     var value = val;
3
4     return function() {
5        value = value + 1;
6        // Can access parent function local
             variable
7        return value;
8     }
9  };
10
11 var f = Counter(5);
12 var g = Counter(2);
13
14 document.writeln( f() ); // Prints 6
15 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

Topic-Based Publish/Subscribe

- Easy decoupling of *content producers* from *content consumers*
- Abstraction of network-related considerations

```
1   // ...
2
3   // Connect
4   pubsub.connect(function() {
5
6       // Repeat every second
7       setInterval(function() {
8
9           // Read temperature from GPIO pin
10          var temperature = GPIO.readPin(12);
11
12          // Publish temperature
13          pubsub.publish("smartsensor/temperature", {
14              id: mySensorId,
15              temperature: temperature
16          });
17
18      }, 1000);
19
20  });
```

Topic-Based Publish/Subscribe

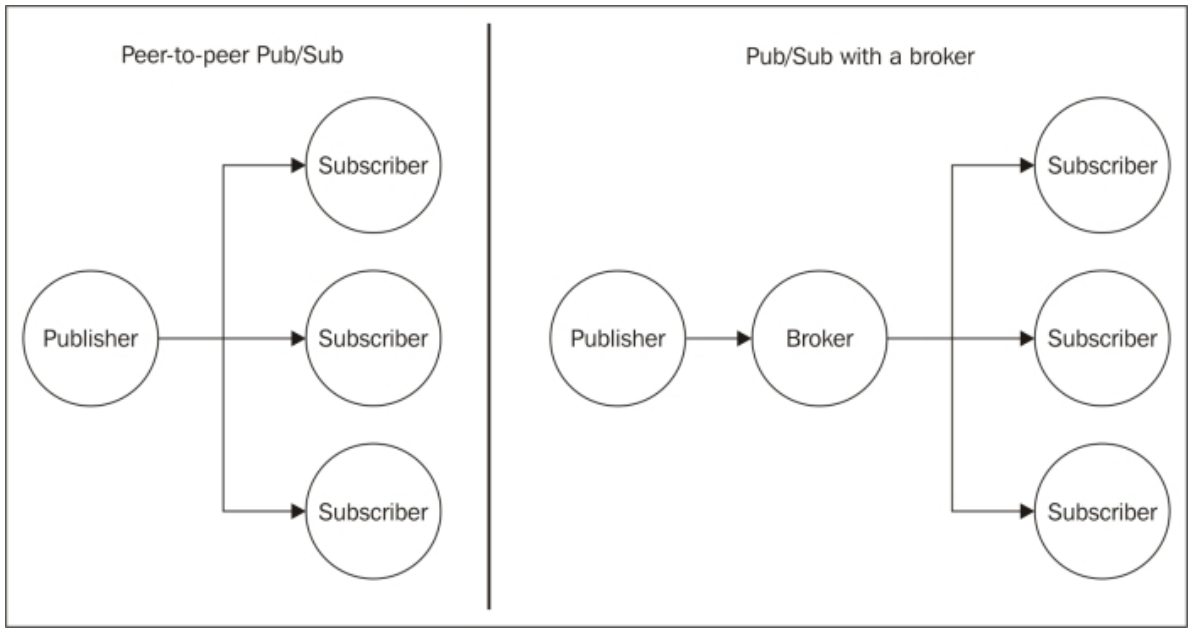
- Easy decoupling of *content producers* from *content consumers*
- Abstraction of network-related considerations

```
1   // ...
2
3   // Connect
4   pubsub.connect(function() {
5
6       // Subscribe to temperature messages
7       pubsub.subscribe("smartsensor/temperature", function(d) {
8
9           if (d.temperature > threshold) {
10              pubsub.publish("smartsensor/actuation", {
11                  id: d.id,
12                  powerVariation: -5
13              });
14          } else if (d.temperature < threshold) {
15              pubsub.publish("smartsensor/actuation", {
16                  id: d.id,
17                  powerVariation: 5
18              });
19          }
20
21      });
22  });
```

Pub/Sub Adaptability

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



Conclusion

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