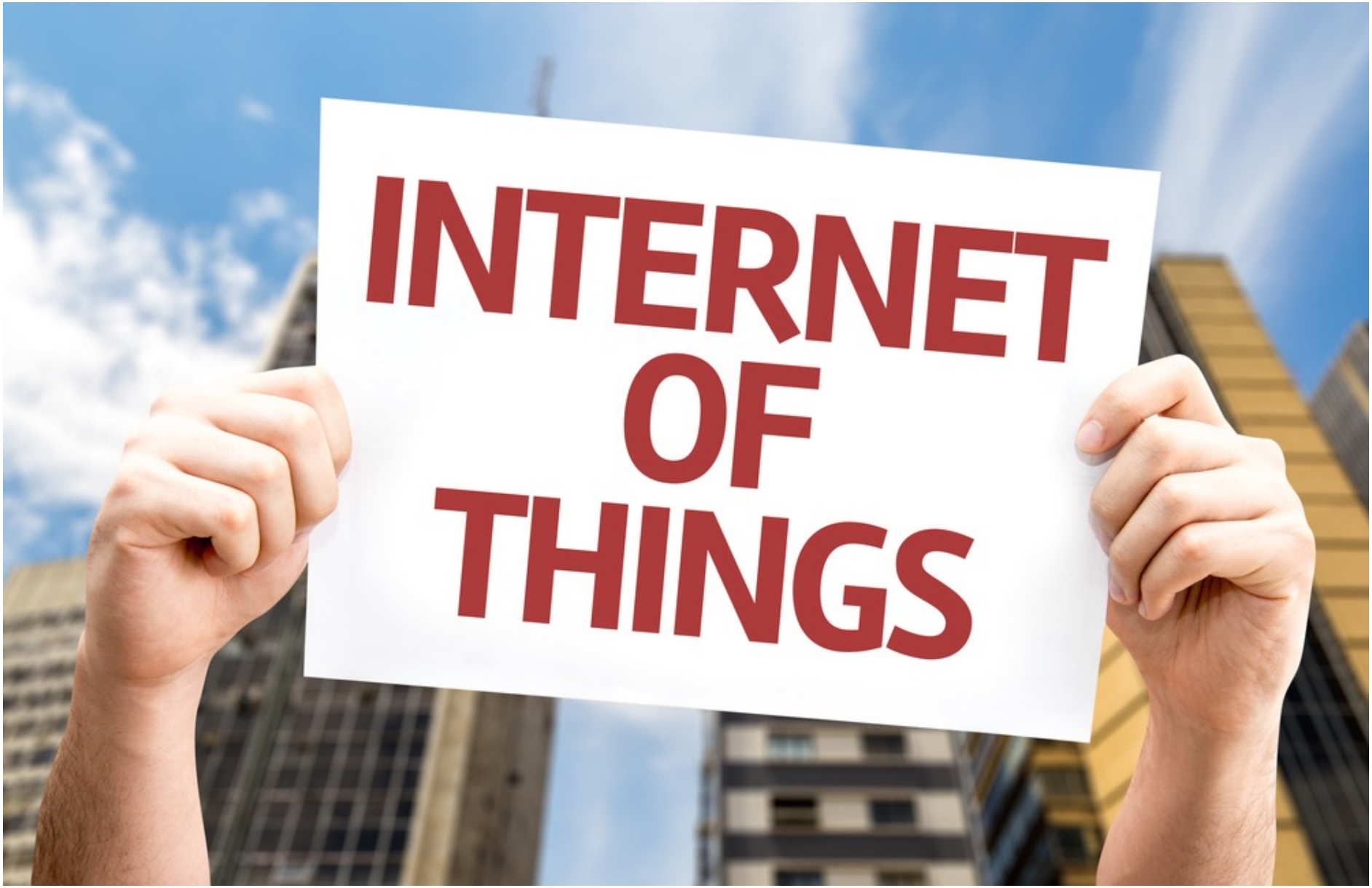
Internet-of-Things and Middleware

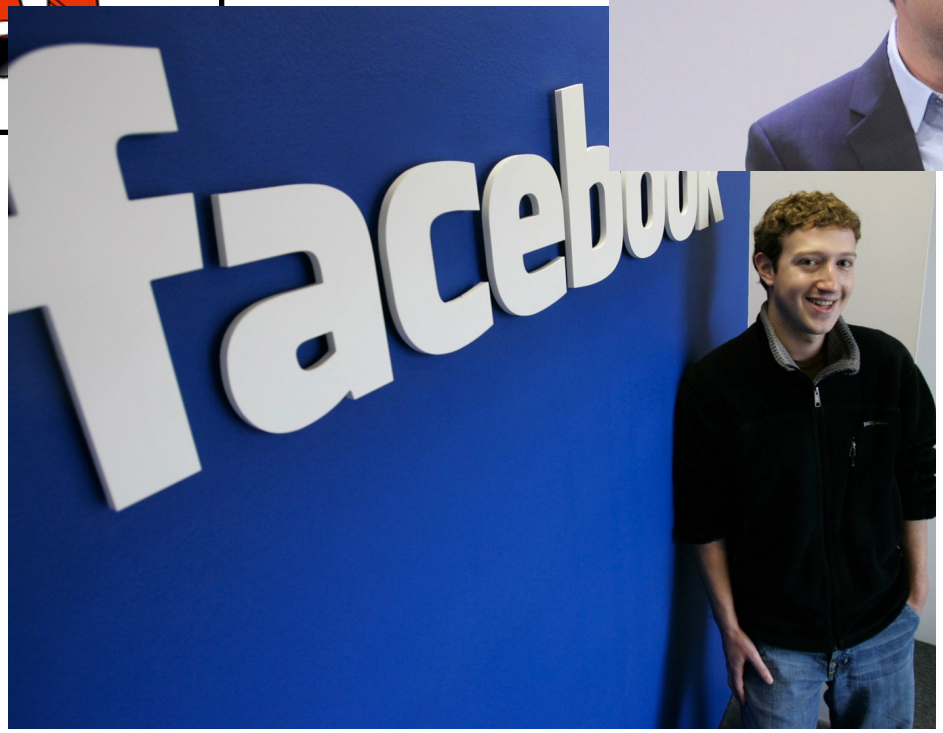
Chi-Sheng Shih

Outline

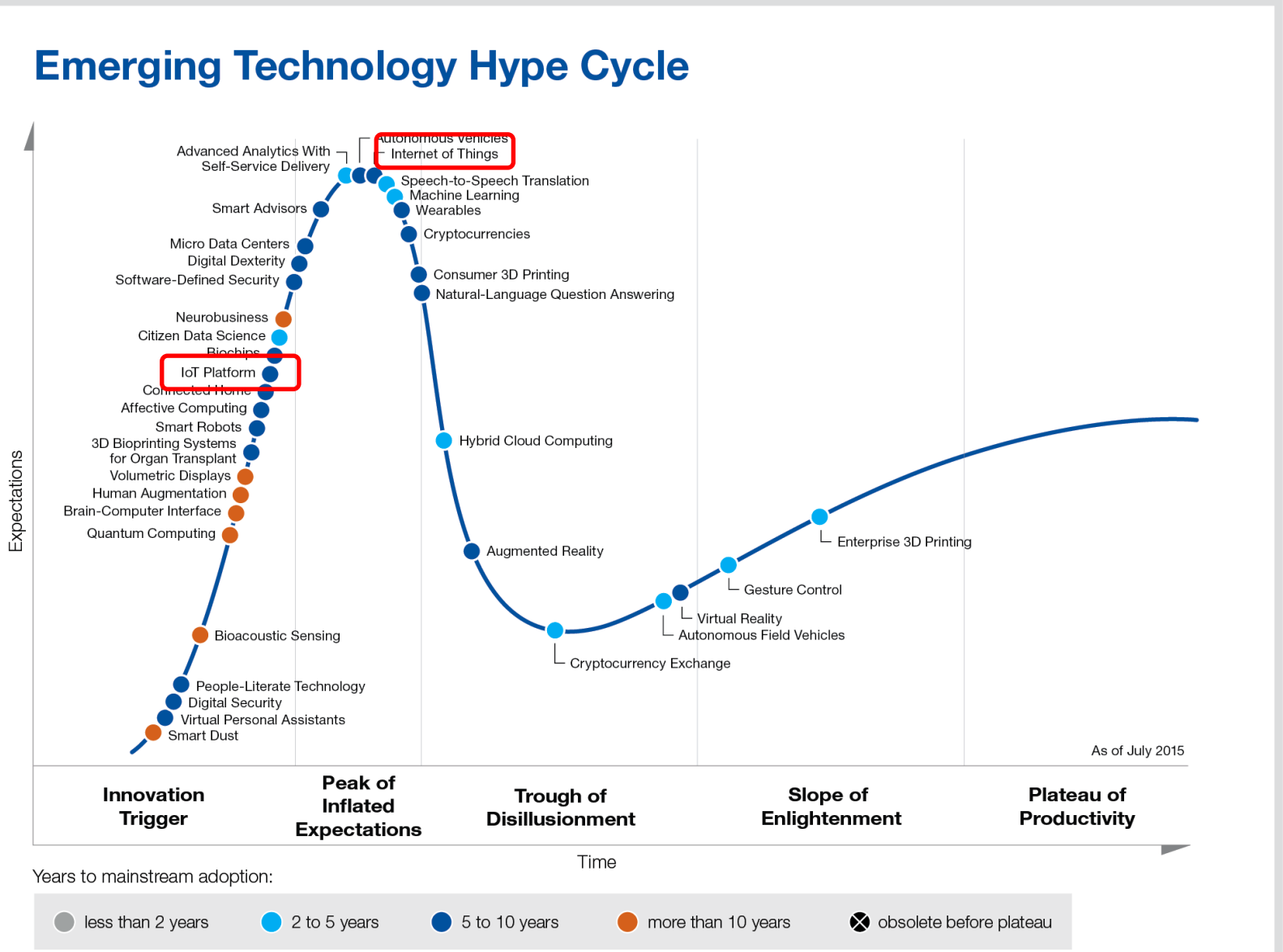
- Who's Who?
- Trend of Internet-of-Things
- History and Examples of Internet-of-Things
- Characteristics of Internet-of-Things
- Challenges for Internet-of-Things
- WuKong: Intelligent Middleware for IoT Systems


INTERNET
OF
THINGS

Who do you think the first one knowing your apartment is on fire?



Gartner Hype Cycle and IoT, July 2015



http://www.gartner.com/newsroom/id/3114217

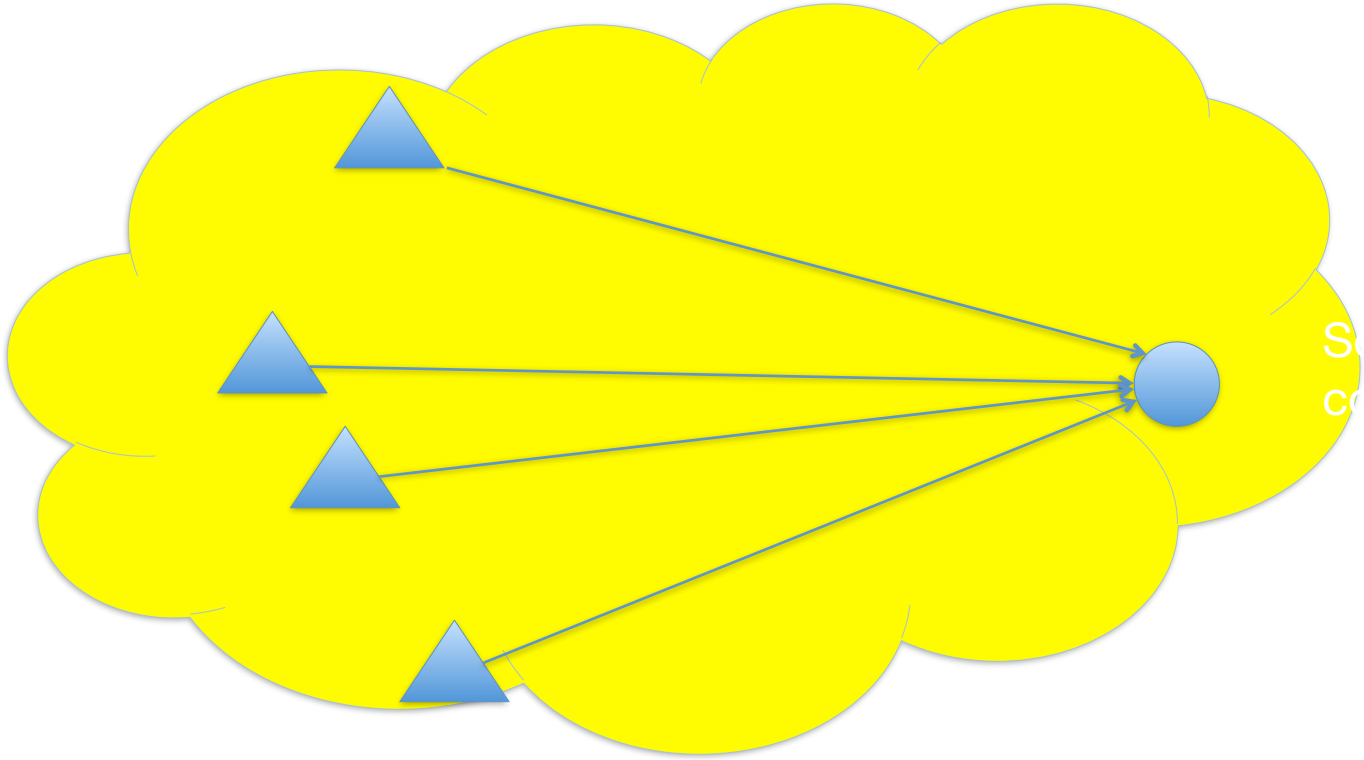
Smart Applications using IoT

Making Things Smart

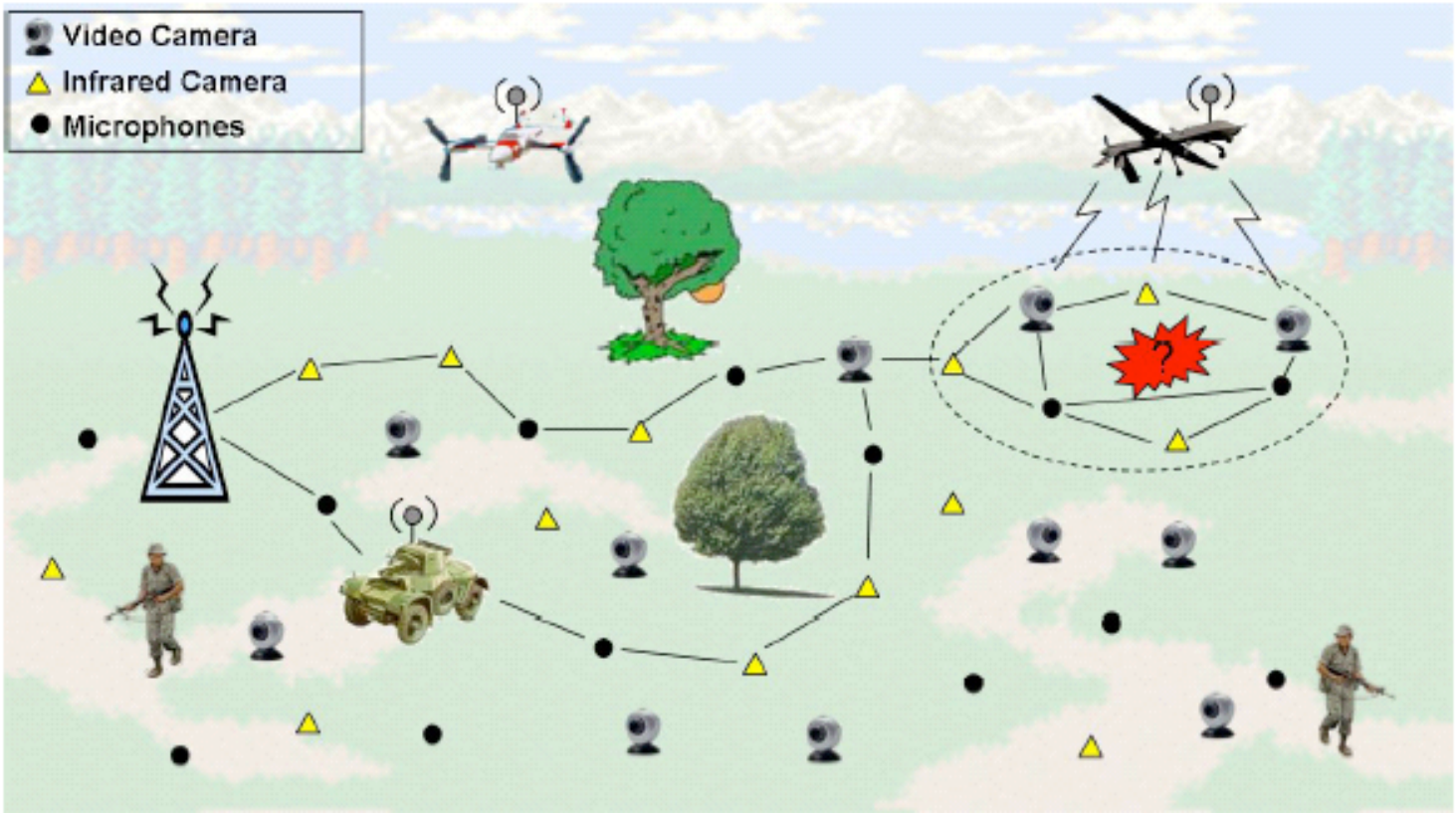
- Internet of Things (IoT) can be used to build many smart applications
  - Smart Wearable
  - Smart Living
  - Smart Home
  - Smart Building
  - Smart Factory
  - Smart Transportation
  - Smart Cities

Who’s who?

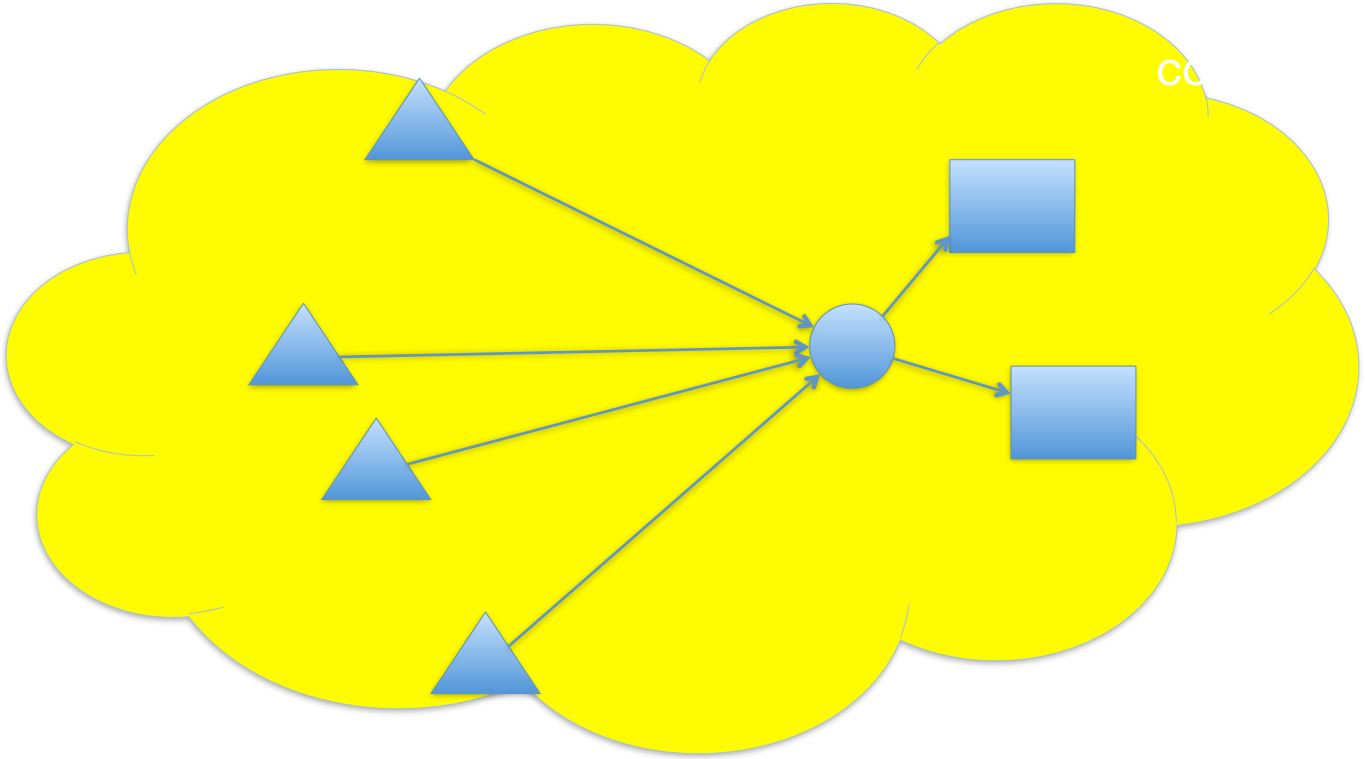
(Wireless) Sensor Networks



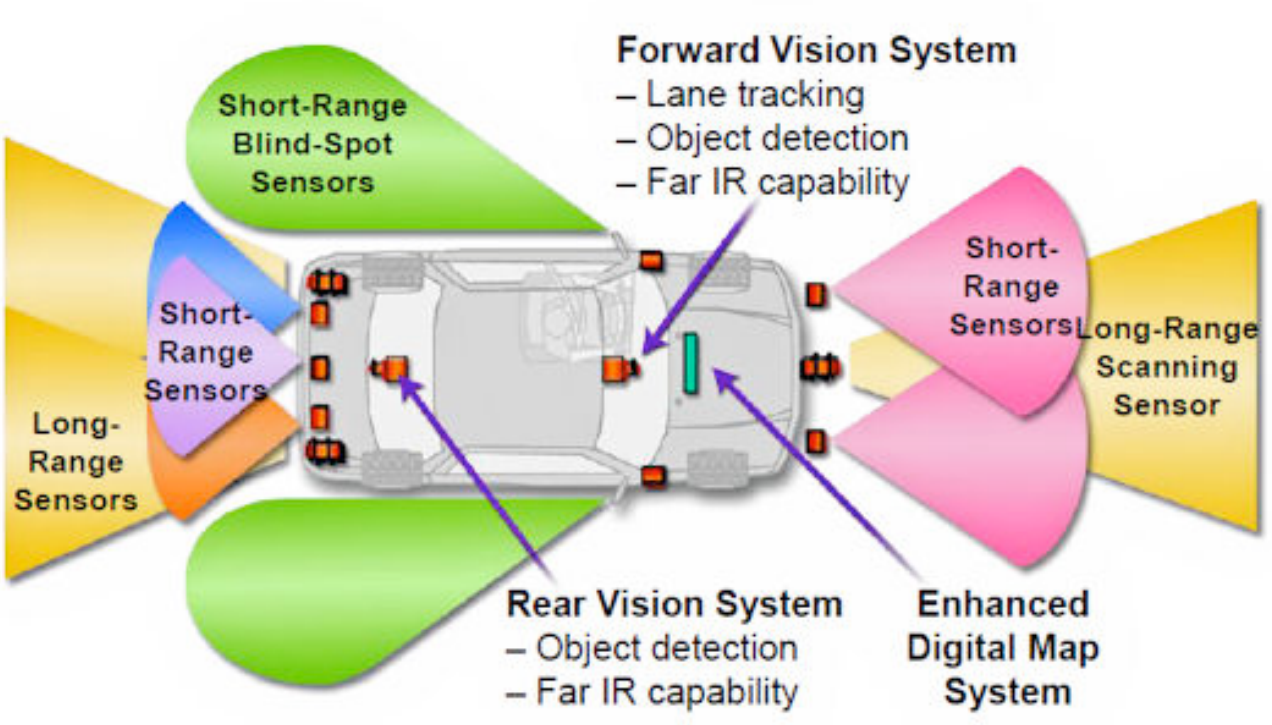
(Wireless) Sensor Networks



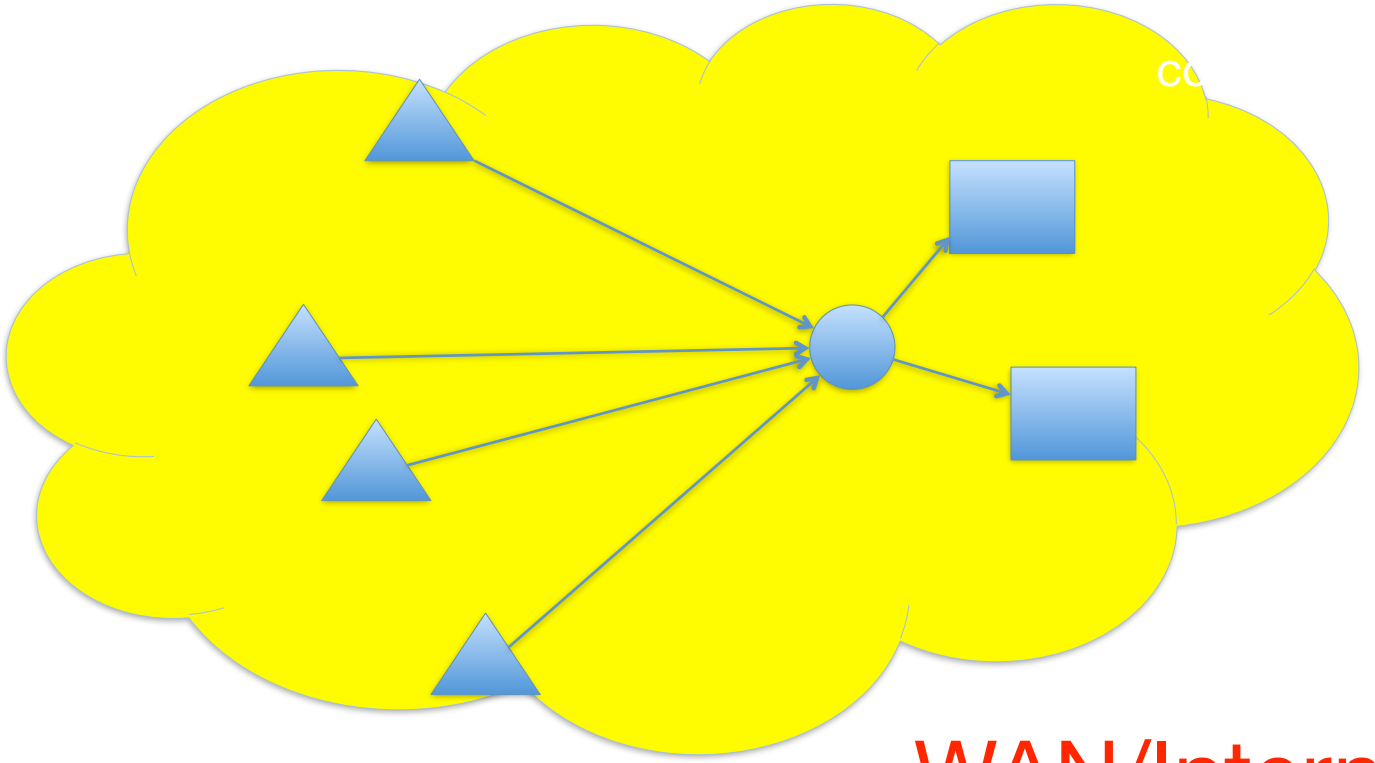
Cyber-Physical Systems



Cyber-Physical Systems

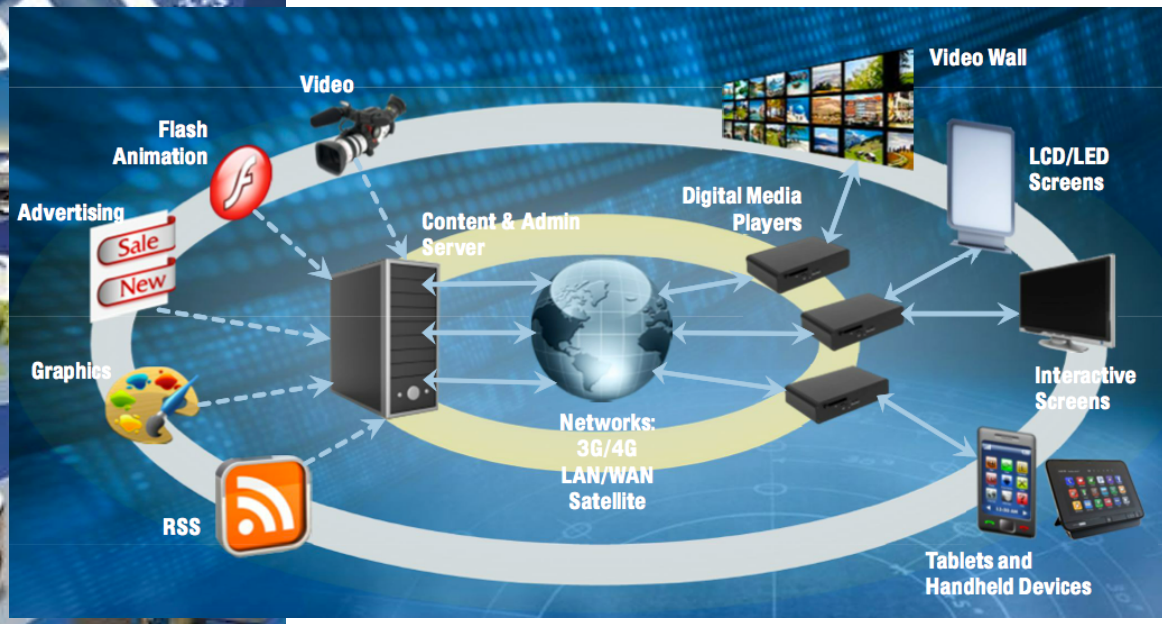


Internet of Things (IoT)

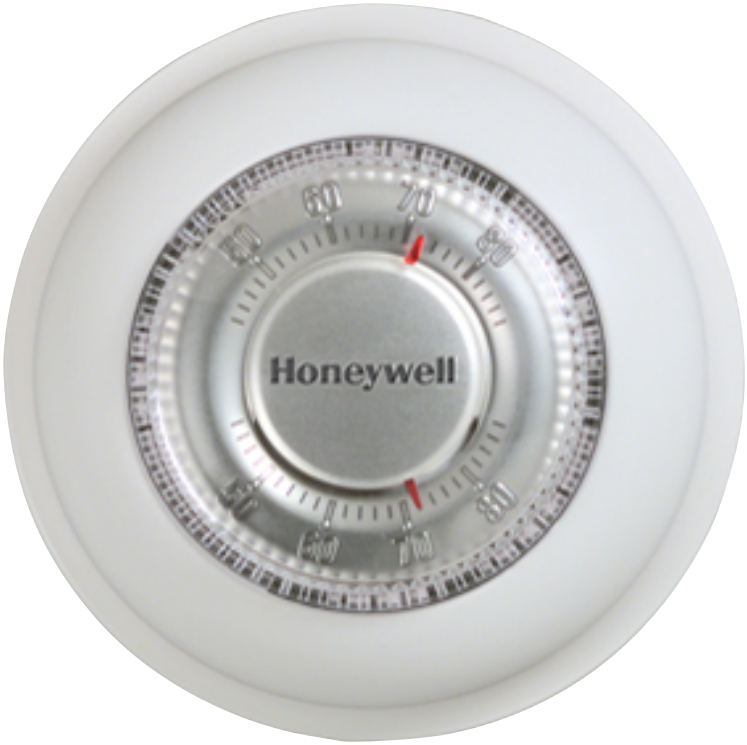


WAN/Internet

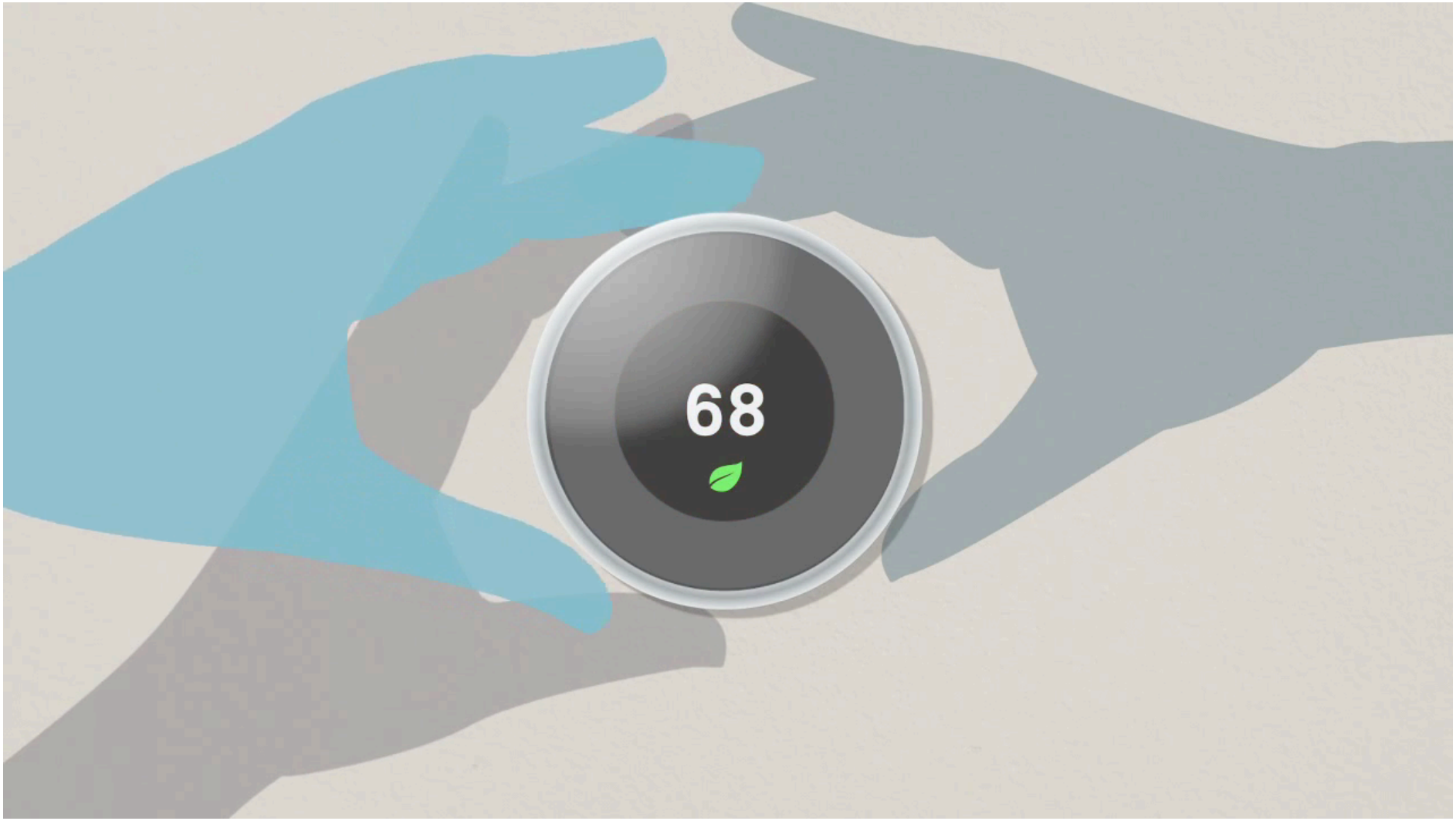
Internet of Things (IoT)



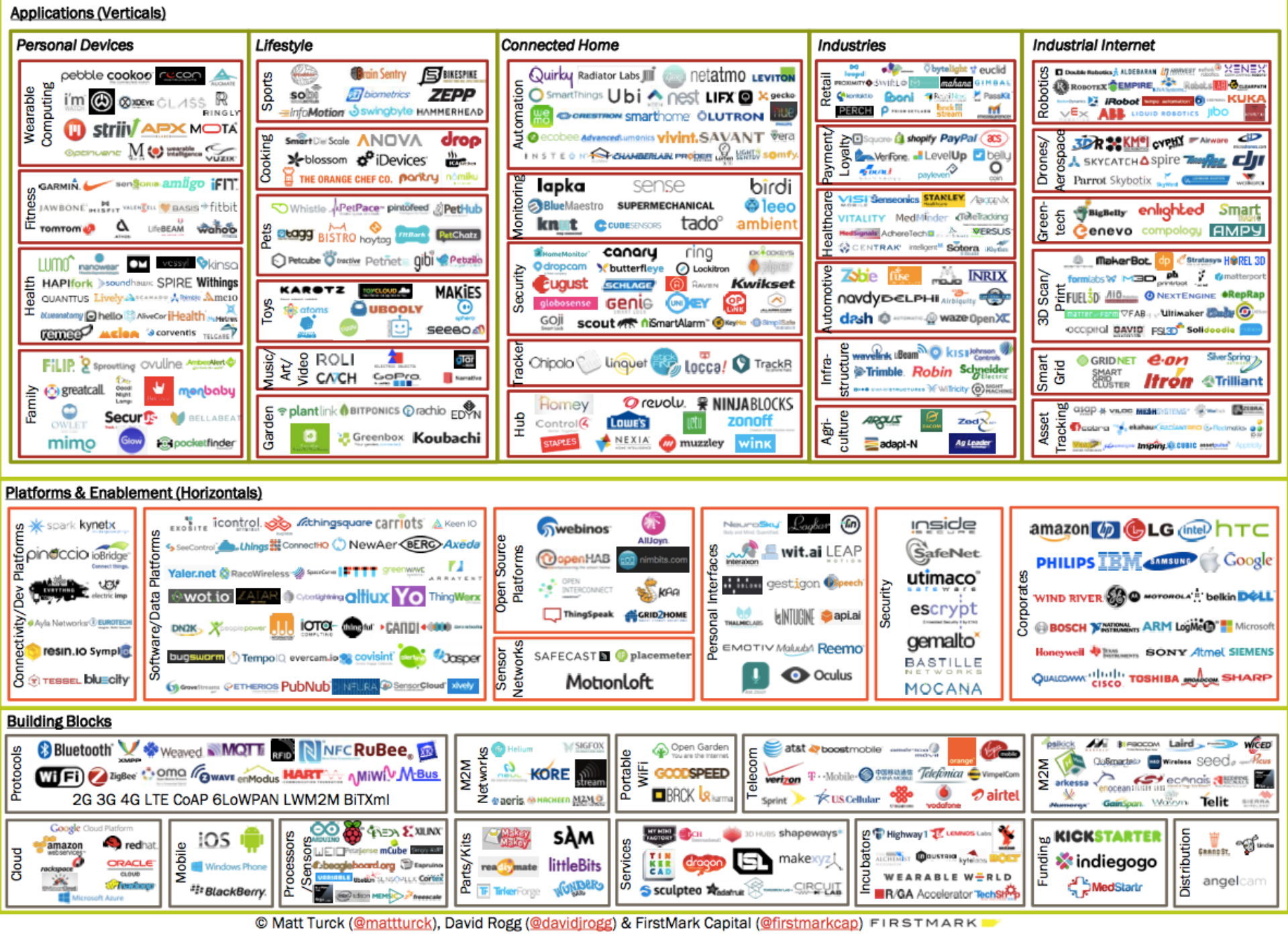
Machine-to-Machine Systems


Honeywell


nest
IN 25 MIN
72


68

物聯網，聯網物，還是聯物網

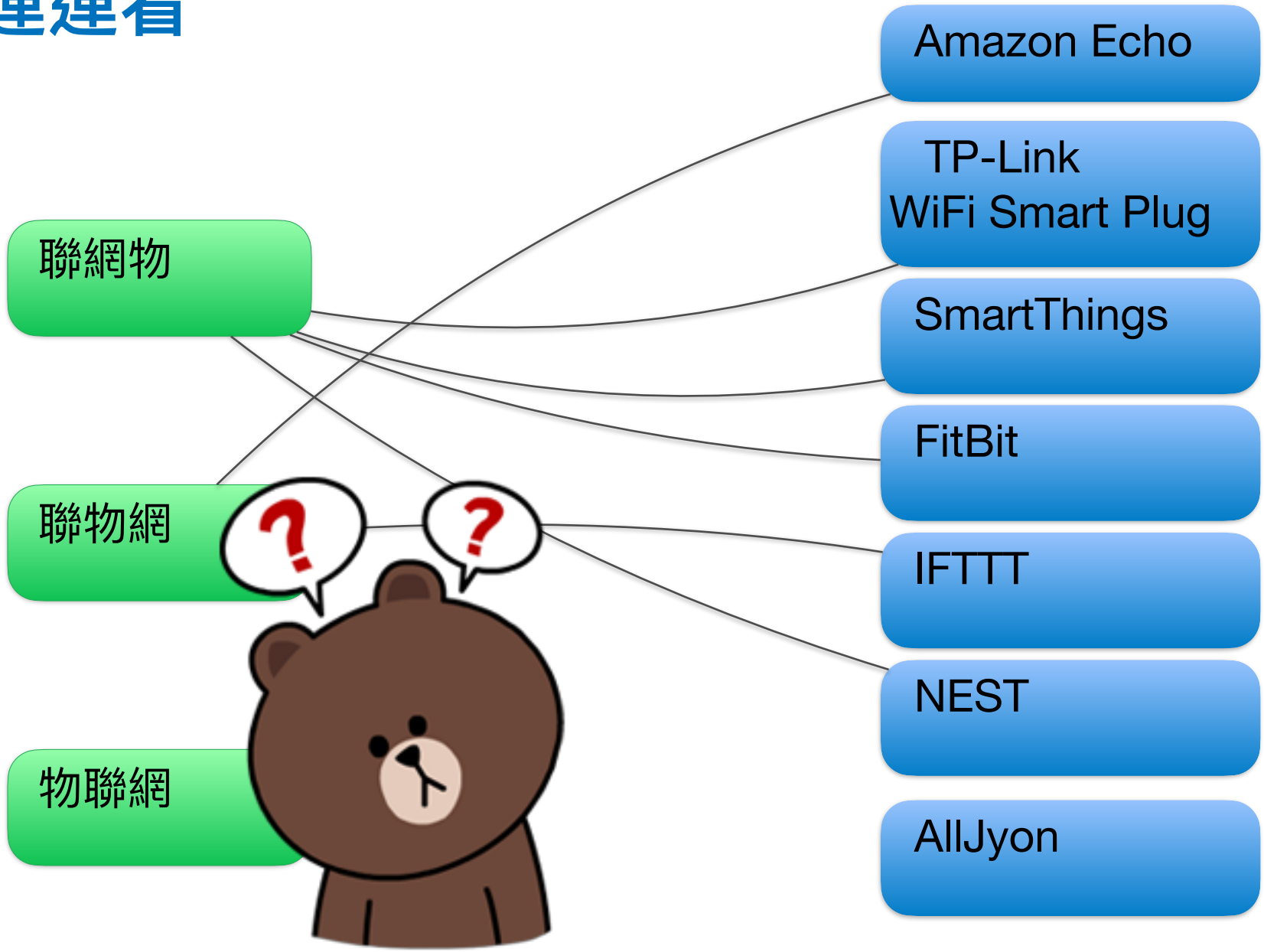


© Matt Turck (@mattturck), David Rogg (@davidjrogg) & FirstMark Capital (@firstmarkcap) FIRSTMARK

物聯網，聯網物還是聯物網?

- 聯網物:物能上網，卻只聯到服務商，物物不相連。
- 聯物網:物聯成網，卻網網不相連。
- 物聯網:物物相聯成網。

連連看


Amazon Echo
TP-Link
WiFi Smart Plug
SmartThings
FitBit
IFTTT
NEST
AllJyon
聯網物
聯物網
物聯網

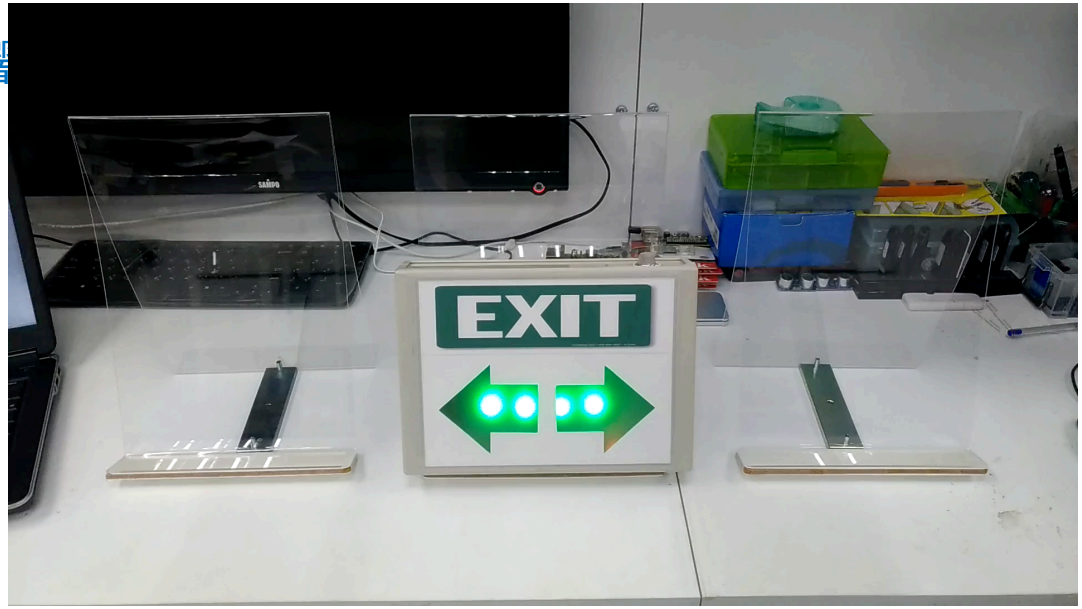
物聯網與你我的生活

- 早起要出門運動，才發現洗好的運動服還沒乾？
- 30分鐘後要出門了，才發現電鍋的時間錯誤，稀飯還沒煮？

- 這些都與日常生活相關，但是卻可有可無。
  - 不能慢跑，改做核心肌肉訓練。
  - 沒煮稀飯，巷口有早餐店。

攸關性命/商業的應用

- 車子要出門，發現昨晚充電現沒接好？
- 火車要發車了，才發現 50 公里外的軌道有障礙物需要排除？
- 電網要送電了，才發現 150 公里外電塔上的電線曲線夾角過大？
- 火災警報響



Challenges of IoT Development

Challenges

- IoT application is still hard to develop.
  - Look at many kickstart projects
- Hardware platforms are not well-supported.
- Programming IoT needs both HW and SW expertise (
  distributed computing, and cloud computing.
- Distributed device management is a nightmare.

Challenges of IoT systems

- Diverse Hardware Environment

  In future M2M systems, there will be many devices and platforms for hardware and software components.

- Evolving System Architecture

M2M systems are deployed to serve for many years; the hardware and software components will evolve over time.

- Dynamic User Needs

During the lifetime of machine-to-machine systems, users served by a system may change their needs, due to context, preference, lifestyle, etc.

- Service Composition Capability

New services may be flexibly composed by the system components to add new capabilities integrating existing and new system components to create a new architecture.

- Multiple Objective Optimization

The system middleware is to provide the best solution for multiple objectives such as comfort, energy consumption, reliability, and responsibility.

Current M2M development

- Many current M2M systems are developed
  - in a low-level language
  - from the node's perspective
  - "Current designs favor architectures where the WSN stack is highly application- or even deployment-specific, rather than application-agnostic as usual."*
- Reasons:
  - Resource constrained nodes (MSP430, ATmega)
  - Lack of OS support (Tiny OS)
  - Often developed by "WSN geeks"*
- Results:
  - Labour intensive, long development cycles
  - Error prone, developers need to consider error handling well
  - Static networks, difficult to adapt to changing conditions, or changing requirements

WuKong: Intelligent Platform for Machine-to-Machine

Wu-Kong : Project Goals

- The project is to build an *Intelligent Virtual Middleware* that can
  1. recognize and adapt to user context and needs;
  2. configure or transform devices into service components;
  3. deploy the most effective yet least expensive solutions;
  4. conduct all of the above capabilities dynamically and remotely.

- Wu-Kong promotes a new M2M programming paradigm:

  *developers should concentrate on high-level functional **flows**, instead of low-level sensor coding.*

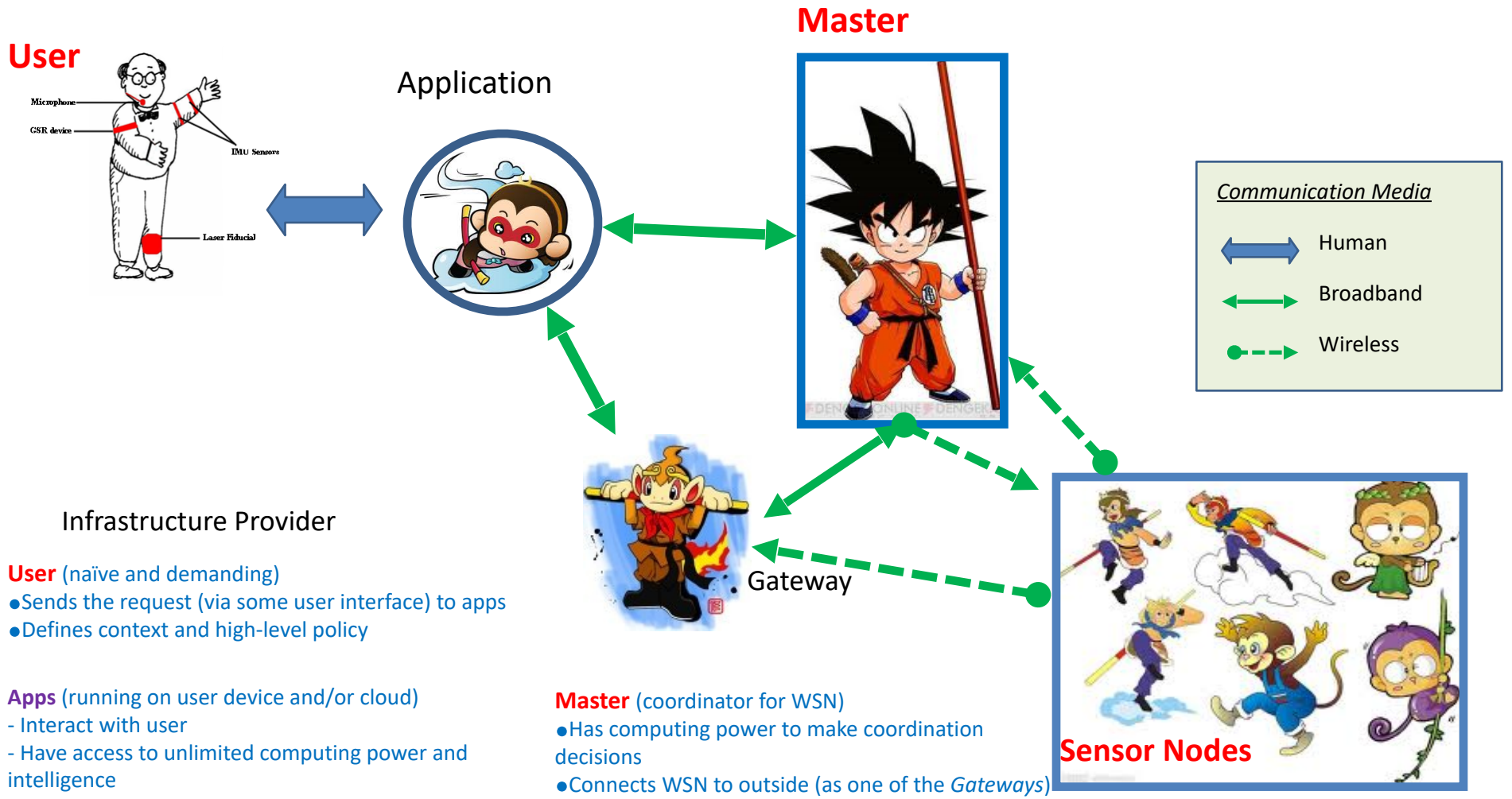
(The name Wu-Kong comes from a heroic character, also known as the *Monkey King*, in the Chinese epic novel *Journey to the West*,

- born from a stone, acquired super powers through teachings of many masters.
- was a trouble maker, but was captured and condemned under a mountain.
- became a loyal guard of a monk travelling to the western heaven.)

Roles in a WuKong System

- Human Entities
  - Application developer: builds programs using a flow editor
  - Device installer: builds *NanoKong* (tiny firmware+JVM), sets up device configuration, and deploys it on the network
  - M2M user: informs Master the user requirement and policy
- System Entities
  - Flow-based program (FBP) Editor: used by developer to create an M2M application
  - Network Base Station: used by installer for enabling device node connection to the network
  - WuKong Master: contacted by user before running an application, will discover devices on network, map FBP components to devices, and deploy the code (for functionality and communication link) to node

Wu-Kong : User's Perspective



User (naïve and demanding)
- Sends the request (via some user interface) to apps
- Defines context and high-level policy

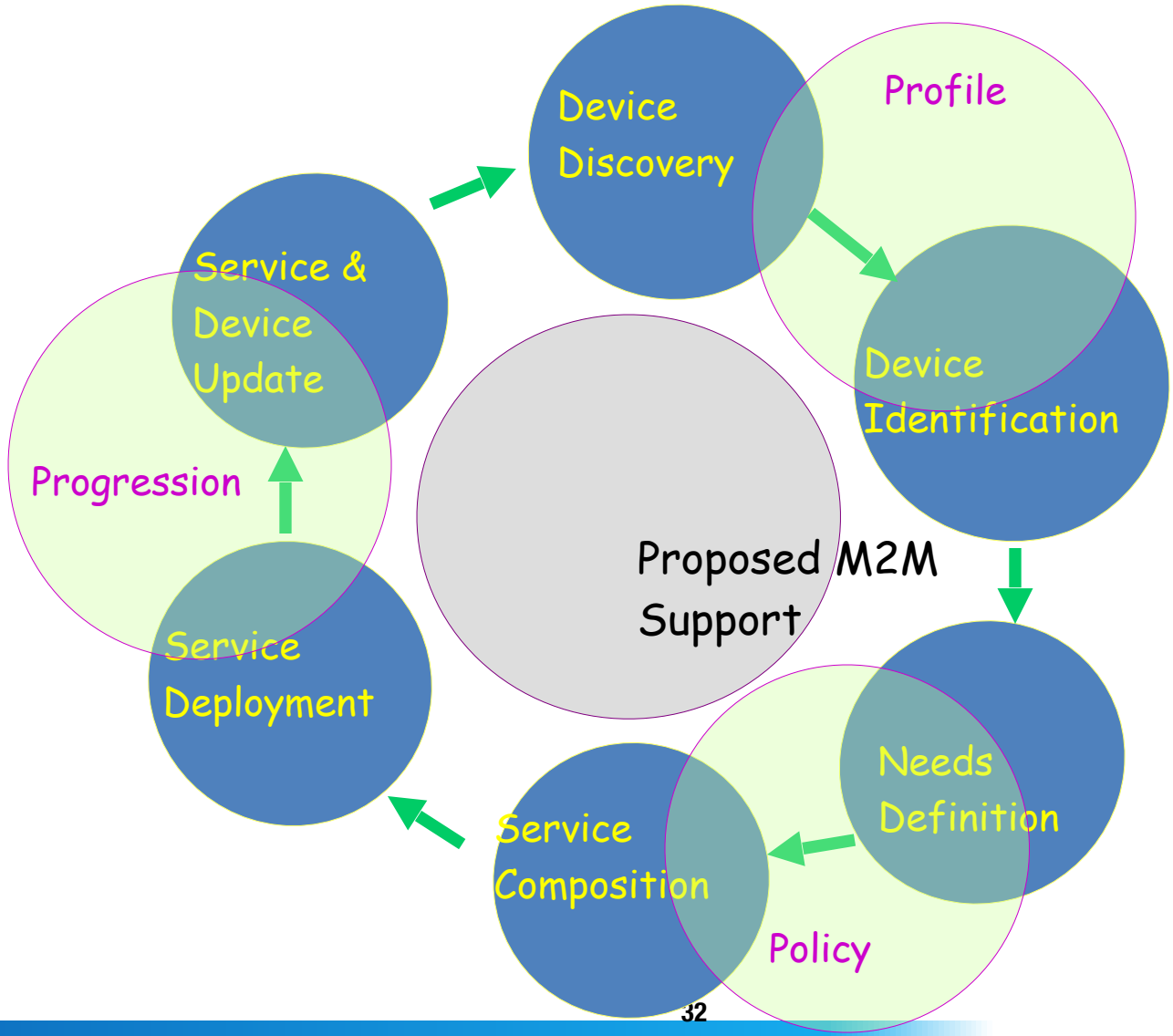
Apps (running on user device and/or cloud)
- Interact with user
- Have access to unlimited computing power and intelligence

Master (coordinator for WSN)
- Has computing power to make coordination decisions
- Connects WSN to outside (as one of the *Gateways*)

Gateway (Cohort for Master)
- provides extra computing/connection
- provides backup coordination

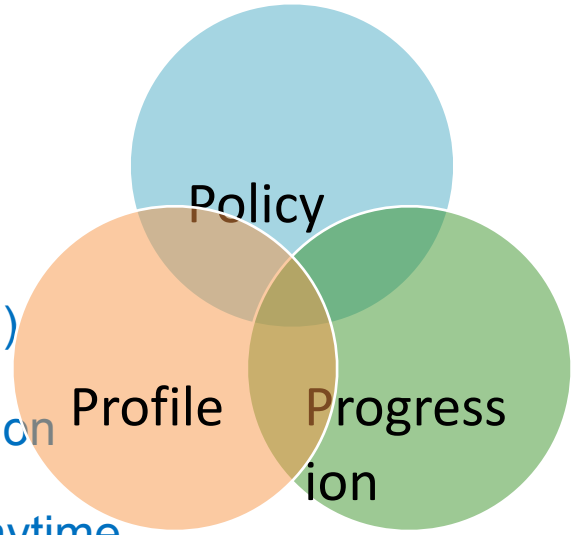
Nodes (sensor devices)
- Physical world sensing and actuating
- Need only limited computing power to sense/send data

A New Paradigm: Tri-Framework on M2M development cycles

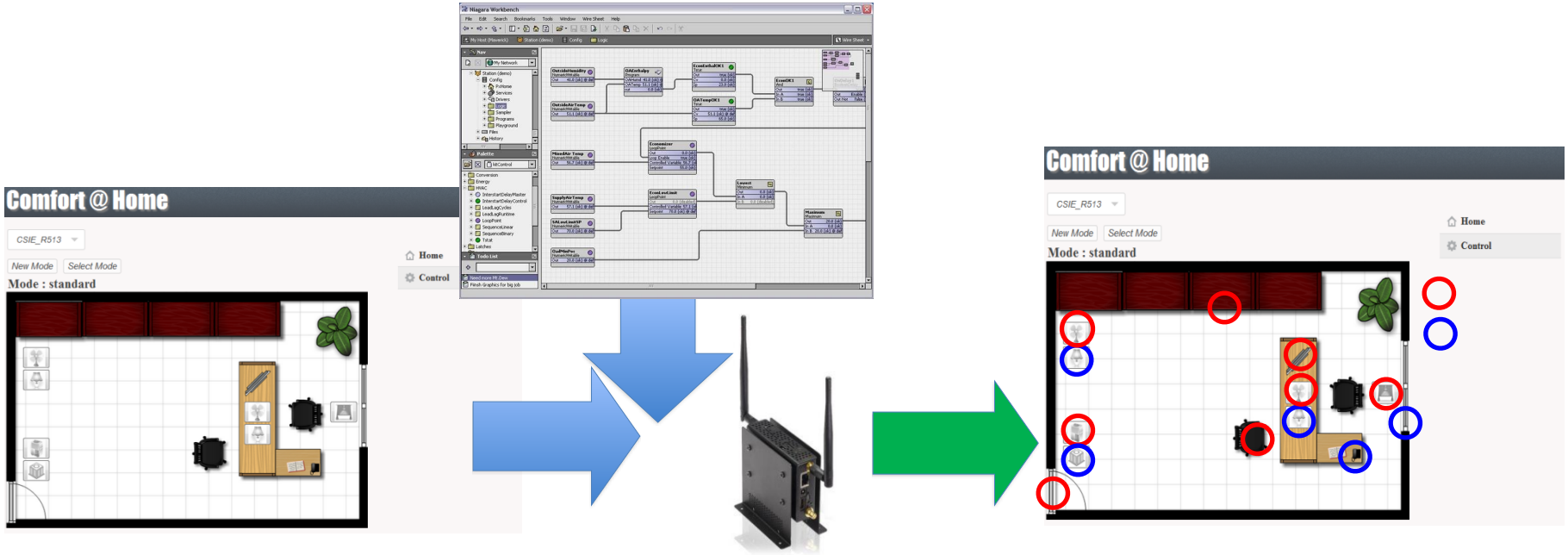


Wu-Kong : Software Perspective

- Tri-Framework approach
- Profile Framework (abstraction for heterogeneous nodes)
  - Device classes (capabilities): discover, share and control
  - Heterogeneous and virtual sensor sharing
- Policy Framework (adaption for M2M context)
  - Configuration decision and constraint optimization
  - Configuration <-> fault tolerance, security, trust
- Progression Framework (embedding intelligence in M2M)
  - Sensor to Master to cloud distribution and coordination
  - Tools for application access and control anywhere, anytime



Wu-Kong : Master's Perspective



Flow-Based Programming allows domain experts to easily put together M2M data and control flows by connecting functional components to produce results & actions.

Given a **user context**, Wu-Kong is to discover the sensors and devices in the target environment, select the most desirable sensors to use, and automatically decide mapping from components to devices,

From the flow-based program (FBP), **Intelligent Compiler** generates the actual (Java) code to be loaded on selected sensors and networks to provide networked services, and deploy codes as required in FBP.

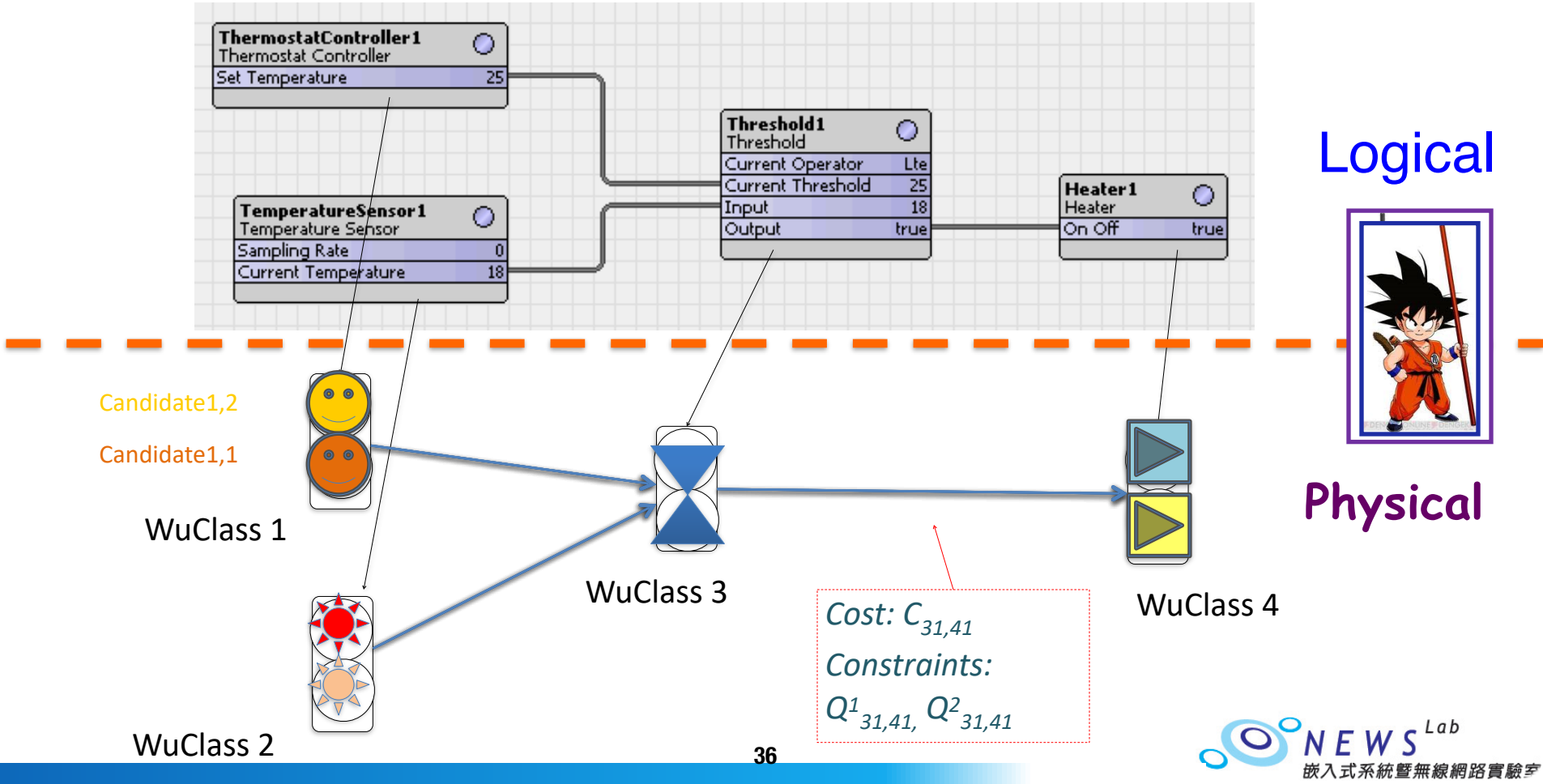
WuKong : Profile Framework

- To configure an M2M network, Master needs to determine what resources are available on a sensor device.
- Master-device protocol has three phases:
  - Determine what devices are on the network
  - Determine what those devices can do (profile)
  - Determine what those devices should do
- After Master has "discovered" devices and queried them, the profile framework on each device provides:
  - What resources are available on the device
  - A way to access and configure those resources

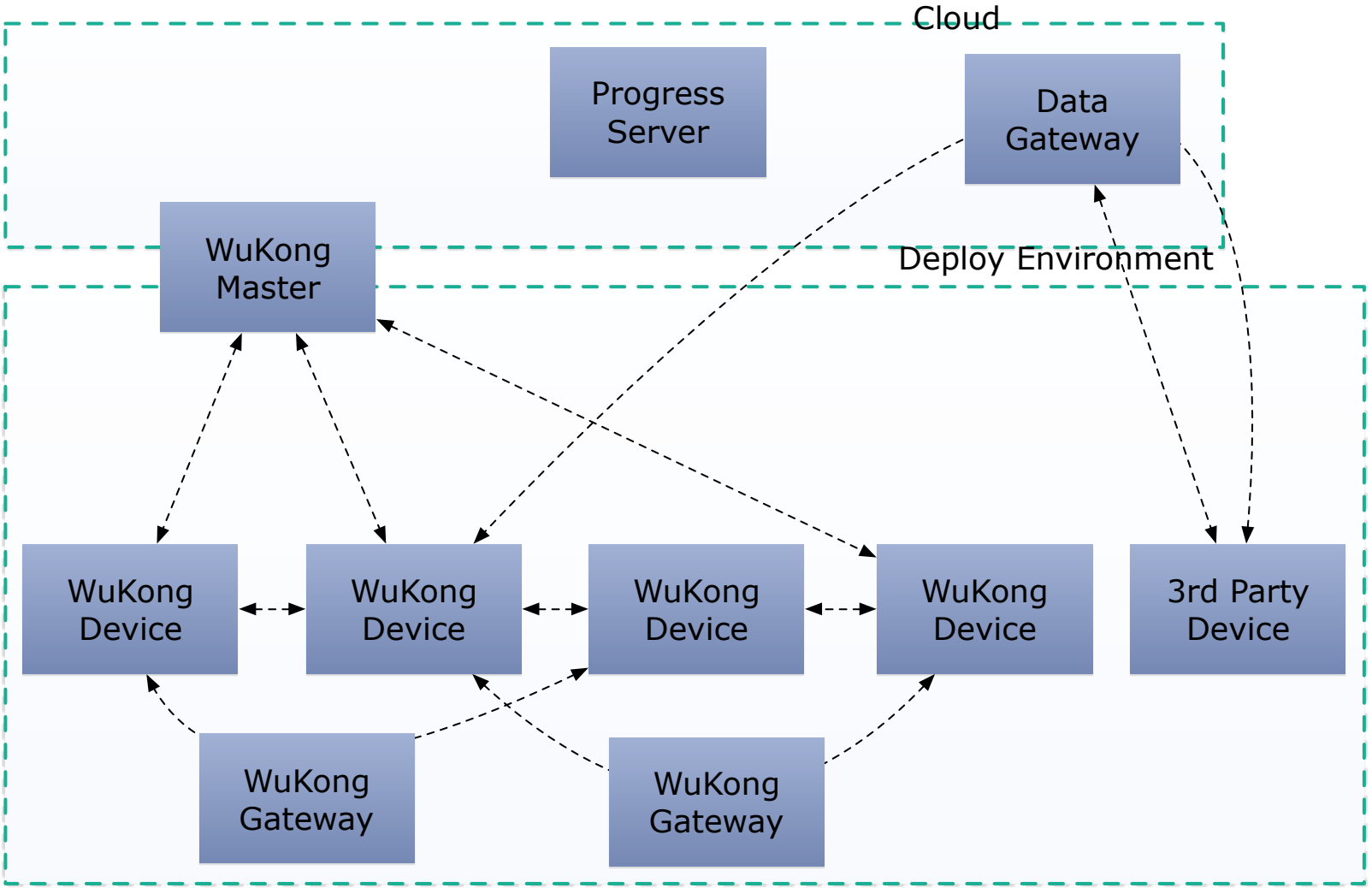
Wu-Kong : Master's Perspective

Given logical flows that have been defined, they must be mapped to some physical networks of nodes/sensors for execution.

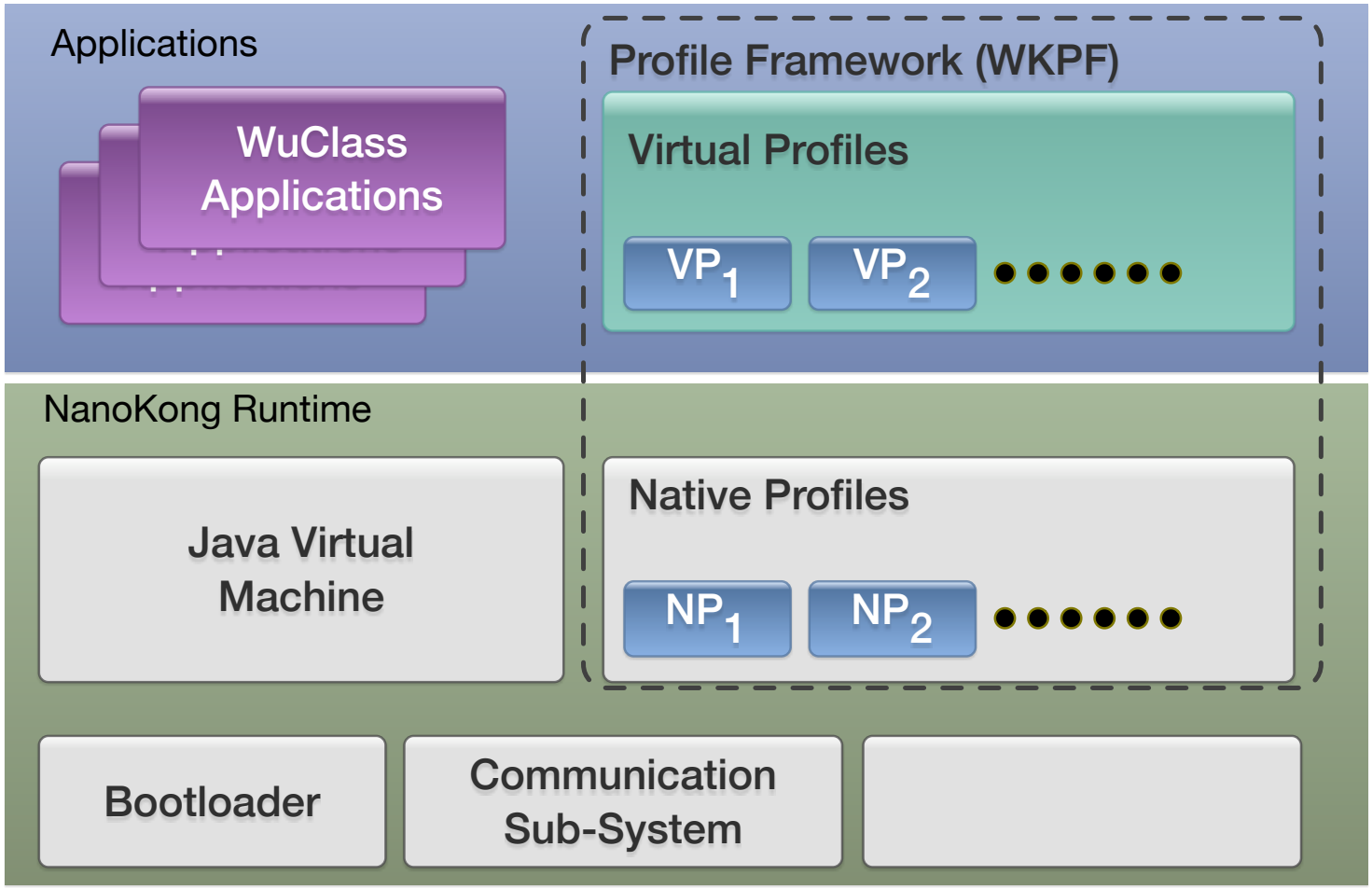
We have built *Master* for that.



Wu-Kong System Architecture



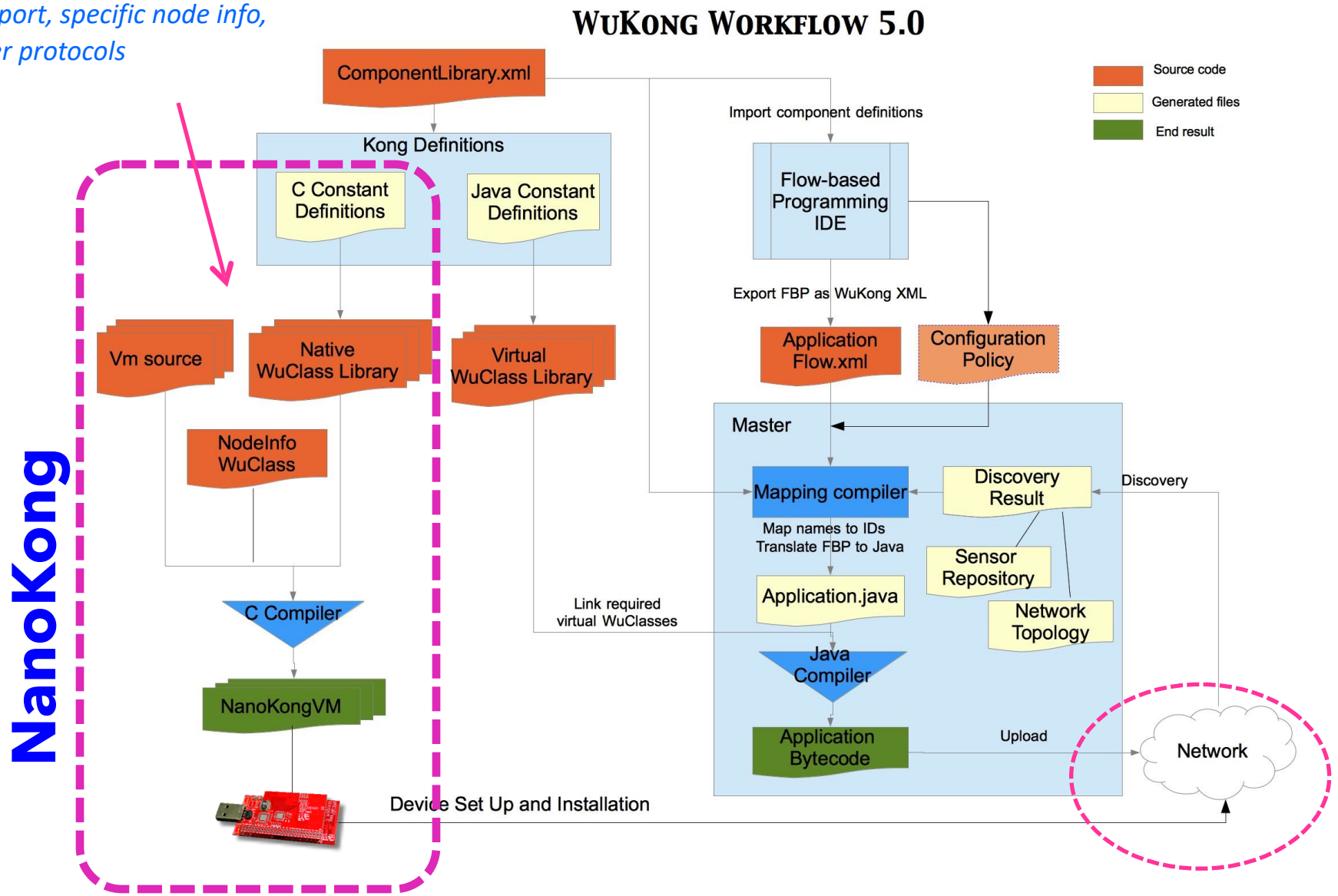
On Devices



Wu-Kong System Building Steps

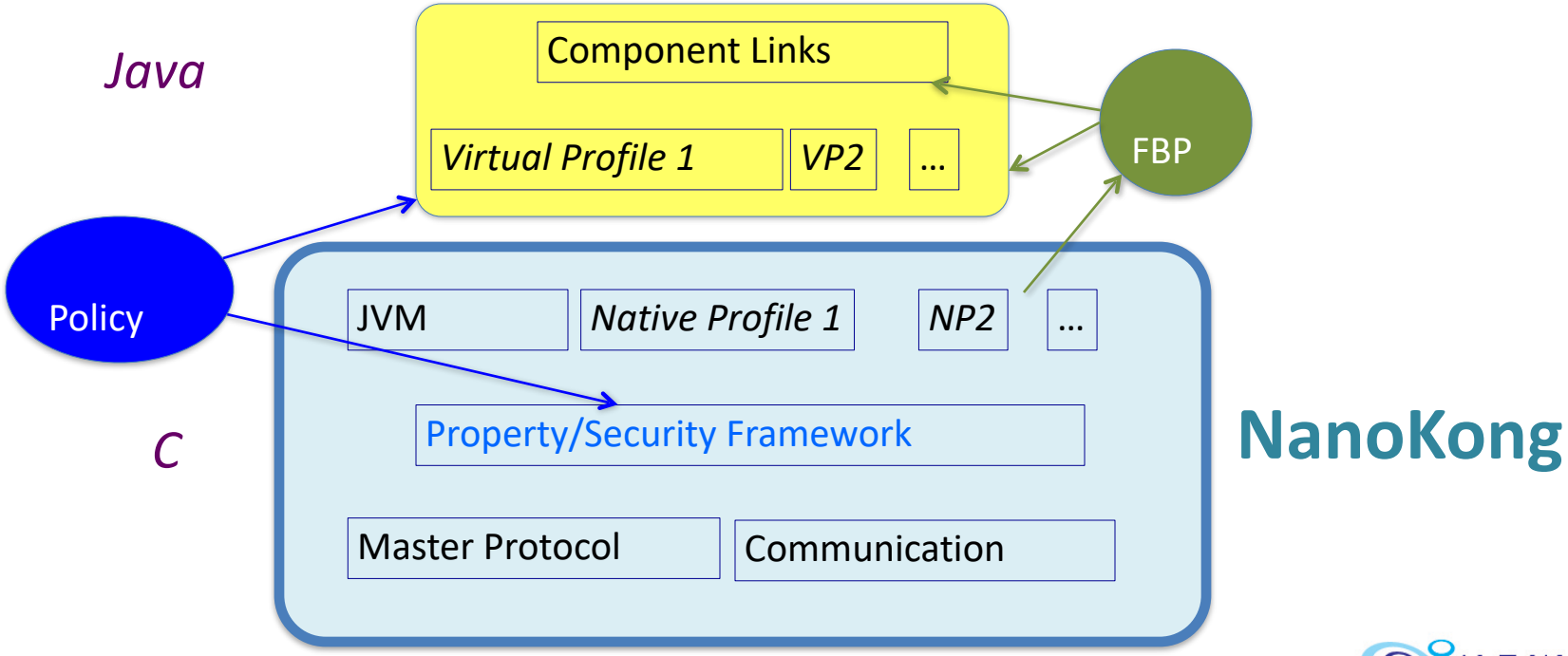
1. Build intelligent sensor devices using NanoKong
    1. Sensor classes (capabilities) are defined and selected for NanoKong
    2. JVM and device networking support are also included
2. Build flow-based programs using IDE
    1. Program components are selected from a component library and linked together
    2. IDE is used to build an application flow structure and export in XML
3. Build a working M2M using WuKong Master
    1. User sends the flow XML to Master to define the application flow
    2. Master discovers sensors remotely, maps them to flow components, and deploys the codes to all sensor nodes

Each sensor device is loaded with native support, specific node info, and Master protocols



Building NanoKong Platforms

- Every node must be statically pre-loaded with native profiles for sensor device drivers, communication support, as well as JVM.
- Functionality beyond the device's native hardware design can be added by uploading software function code on a device. Such a functionality is referred as an virtual profile (in Java code)



NanoKong VM code size (NanoVM)

Memory requirement: 25K Flash, 2K RAM

Type	Component	Size (bytes)
VM	*Core JVM*	*5604*
VM	*Communication (ZWave+Zigbee)*	*7564*
VM	*Profile Framework*	*4244*
VM	*Native Profiles*	*632*
VM	*AVR Specific IO*	*3630*
VM	*String operations/user IO*	*950*
VM	*Total*	*22624*
VM	*Total compiled*	*20824*
Application	*Home automation application (native)*	*413*
Application	*Native Threshold WuClass (C)*	*199*
Application	*Virtual Threshold WuClass(Java)*	*330*

NanoKong VM code size (Darjeeling)

Memory requirement: 75K Flash, 2K RAM

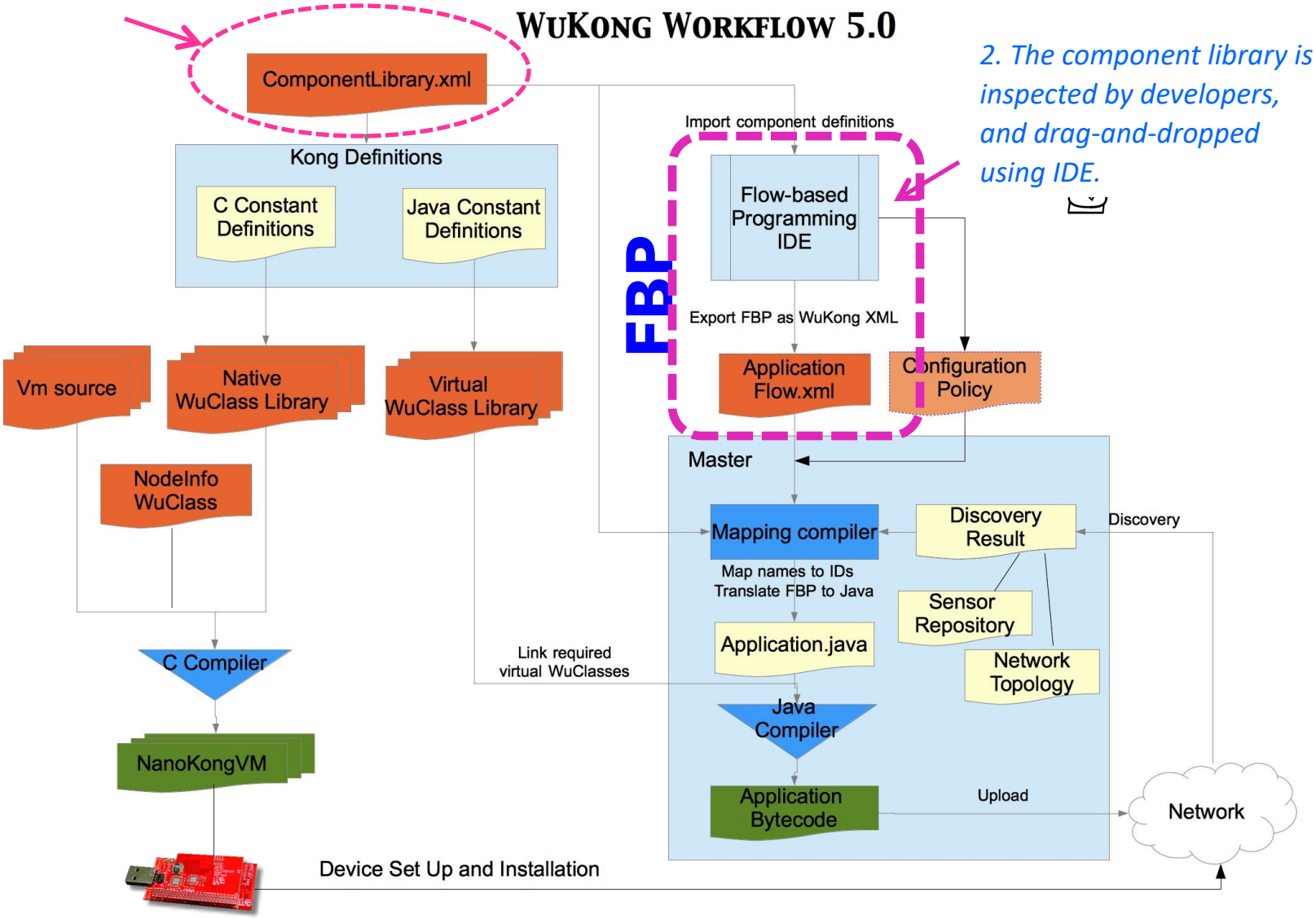
Type	Component	Size (bytes)
VM	*Core JVM*	*27770*
VM	*Communication (ZWave)*	*2965*
VM	*Profile Framework*	*8032*
VM	*Native Profiles*	*1379*
VM	*AVR Specific IO*	*3445*
VM	*String operations/user IO*	*14698*
VM	*Total C code*	*47736*
VM	*Total*	*74856*
Application	*Home automation application (native)*	*576*
Application	*Native Threshold WuClass (C)*	*199*
Application	*Virtual Threshold WuClass(Java)*	*330*

Wu-Kong System Building Steps

1. Build intelligent sensor devices using NanoKong
    1. Sensor classes (capabilities) are defined and selected for NanoKong
    2. JVM and device networking support are also included
2. Build flow-based programs using IDE
    1. Program components are selected from a component library and linked together
    2. IDE is used to build an application flow structure and export in XML
3. Build a working M2M using WuKong Master
    1. User sends the flow XML to Master to define the application flow
    2. Master discovers sensors remotely, maps them to flow components, and deploys the codes to all sensor nodes

1. We have created a component library that includes all native and virtual profiles.

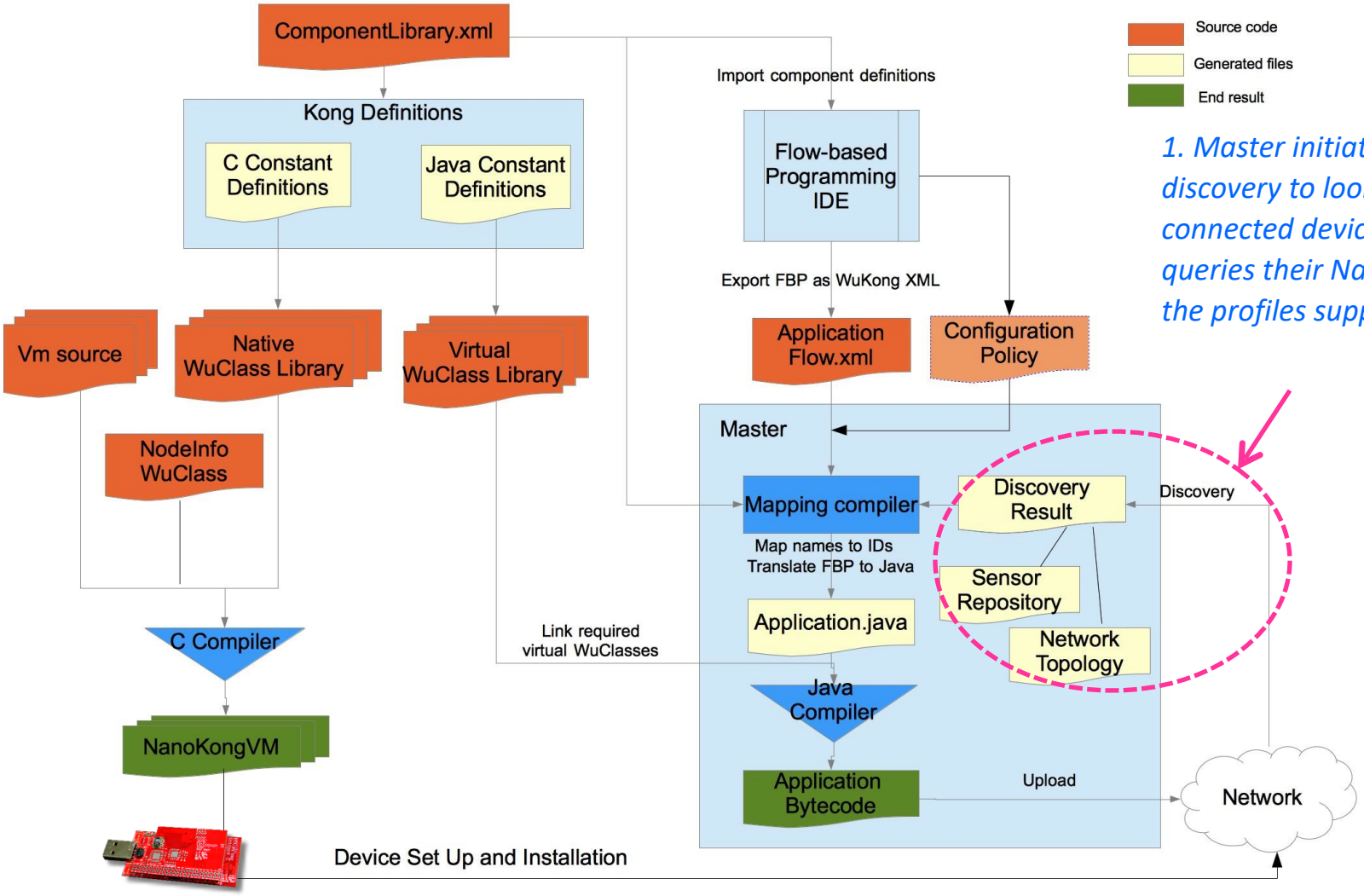
2. The component library is inspected by developers, and drag-and-dropped using IDE.



Wu-Kong System Building Steps

1. Build intelligent sensor devices using NanoKong
    1. Sensor classes (capabilities) are defined and selected for NanoKong
    2. JVM and device networking support are also included
2. Build flow-based programs using IDE
    1. Program components are selected from a component library and linked together
    2. IDE is used to build an application flow structure and export in XML
3. Build a working M2M using WuKong Master
    1. User sends the flow XML to Master to define the application flow
    2. Master discovers sensors remotely, maps them to flow components, and deploys the codes to all sensor nodes

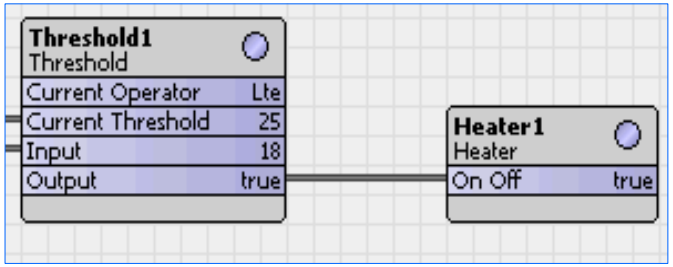
WuKong Workflow 5.0



1. Master initiates the discovery to look for all connected devices and queries their NanoKong for the profiles supported.

WuKong : Profile Framework

- A standard component library defines the components we use to create the FBP
  - Represents some kind of functionality
  - Either hardware or software
  - Defines
    - an interface as a set of properties
    - a method to run, either periodically, or when the input changes
  - FBP links connect component properties



- The profile framework takes care of
  - telling the master what components are available on a node
  - creating new instances of components and links between them
  - propagating changes between linked properties
  - invoking the component's method

WuKong : Profile Framework

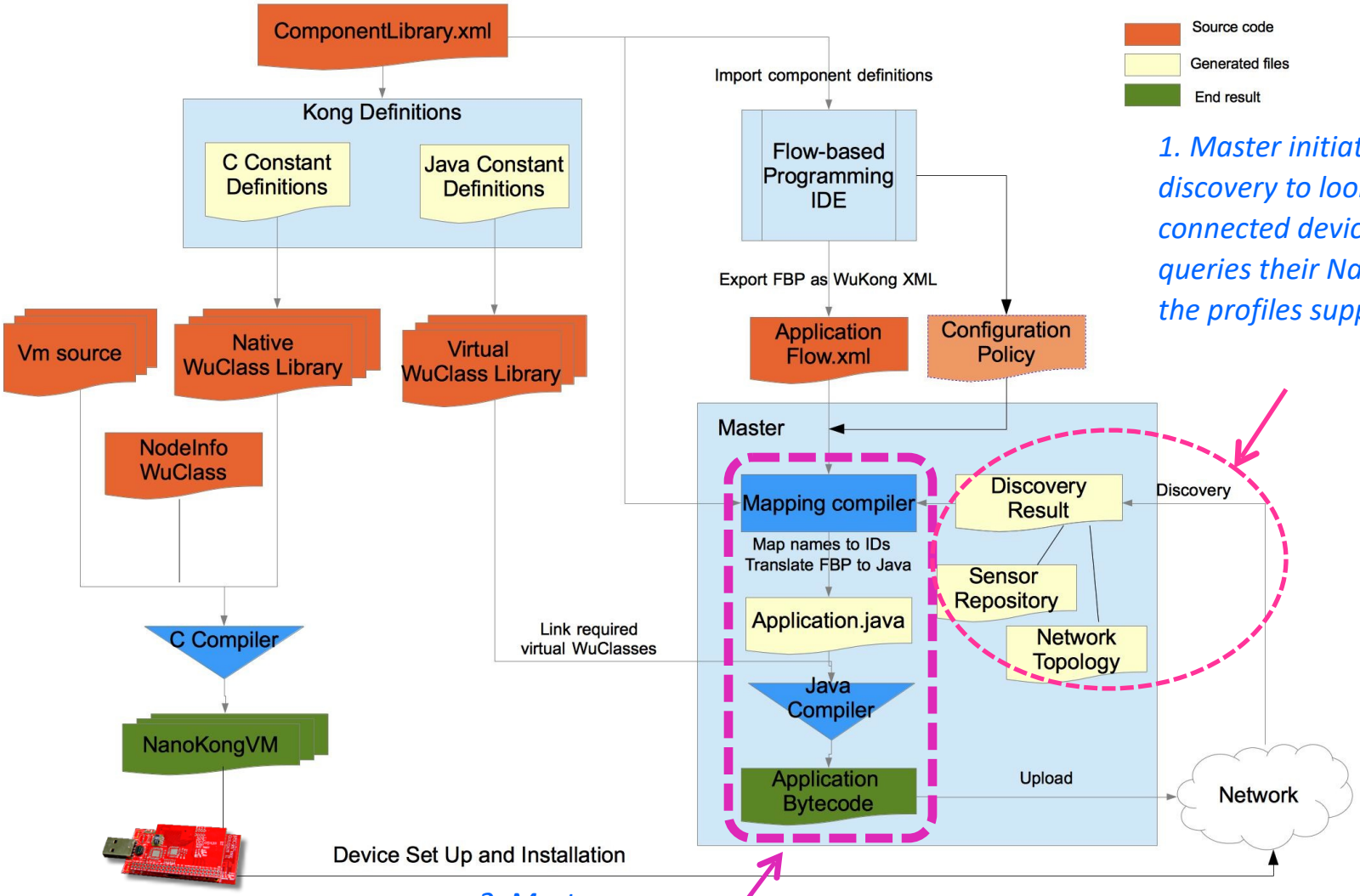
- Some examples
  - A light sensor (hardware)
  - A threshold (software)

```
<WuClass name="Light_Sensor" id="5" type="hard">
        <property name="current_value" datatype="short" access="readonly" />
        <property name="refresh_rate" datatype="refresh_rate" access="readwrite" default="10" />
</WuClass>

<WuClass name="Threshold" id="1" type="soft">
        <property name="operator" datatype="ThresholdOperator" access="readwrite" default="lt" />
        <property name="threshold" datatype="short" access="readwrite" default="30" />
        <property name="value" datatype="short" access="readwrite" default="20" />
        <property name="output" datatype="boolean" access="readonly" />
</WuClass>

<WuTypedef name="ThresholdOperator" type="enum">
        <enum value="LT"/>
        <enum value="GT"/>
        <enum value="LTE"/>
        <enum value="GTE"/>
</WuTypedef>
```

Native vs. Virtual Components

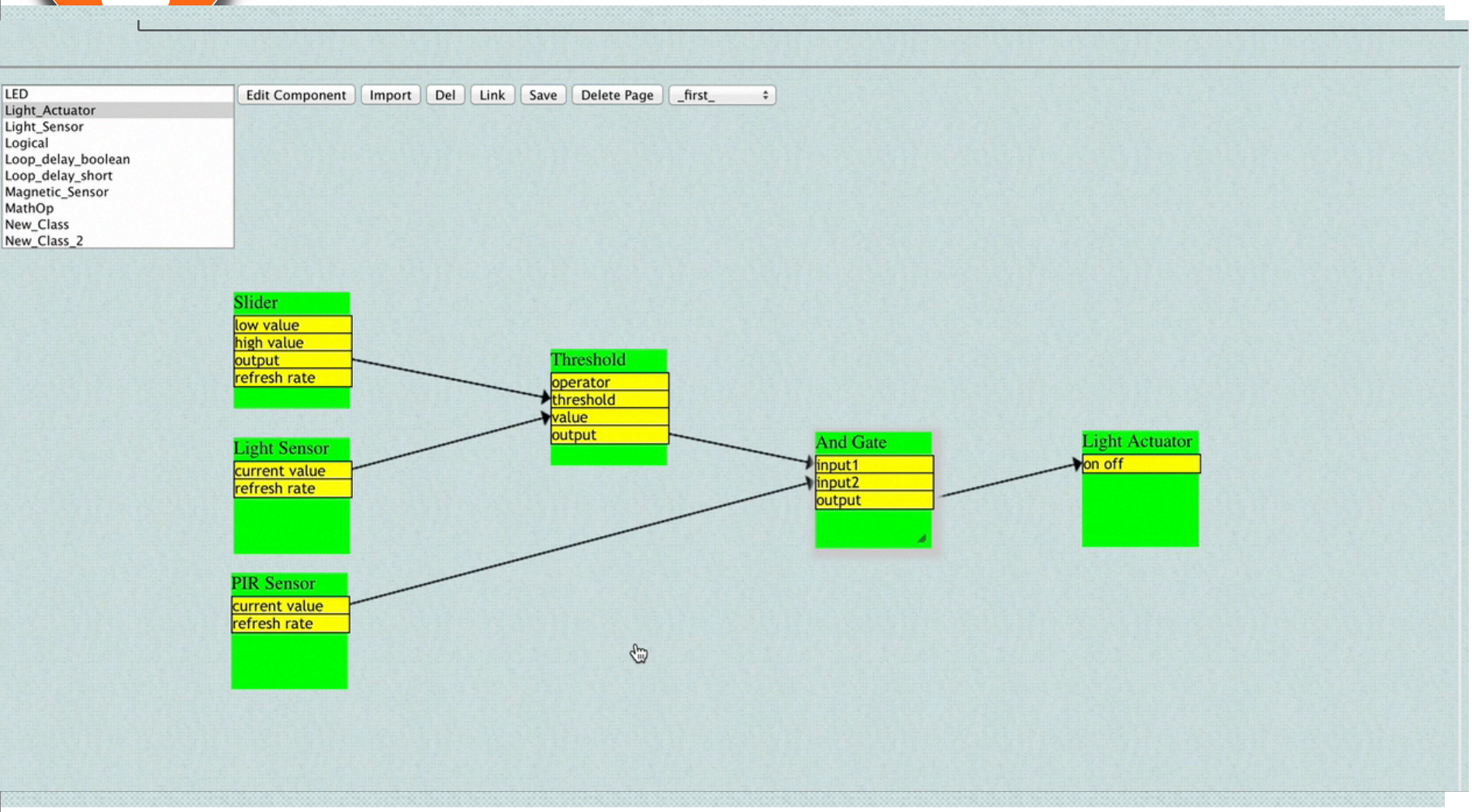
- Most device profiles are defined and pre-loaded to access a device's fixed native hardware
- Functionality beyond the device's native hardware design can be added by uploading software function code on a device.
- Such a functionality is referred as an "virtual" profile
  - This profile isn't (directly) tied to the node's hardware
  - It is implemented in software (Java bytecode)
- In this way virtual sensors can be implemented, e.g. combining data from different sensor sources into a new 'sensor'.

WuKong Workflow 5.0

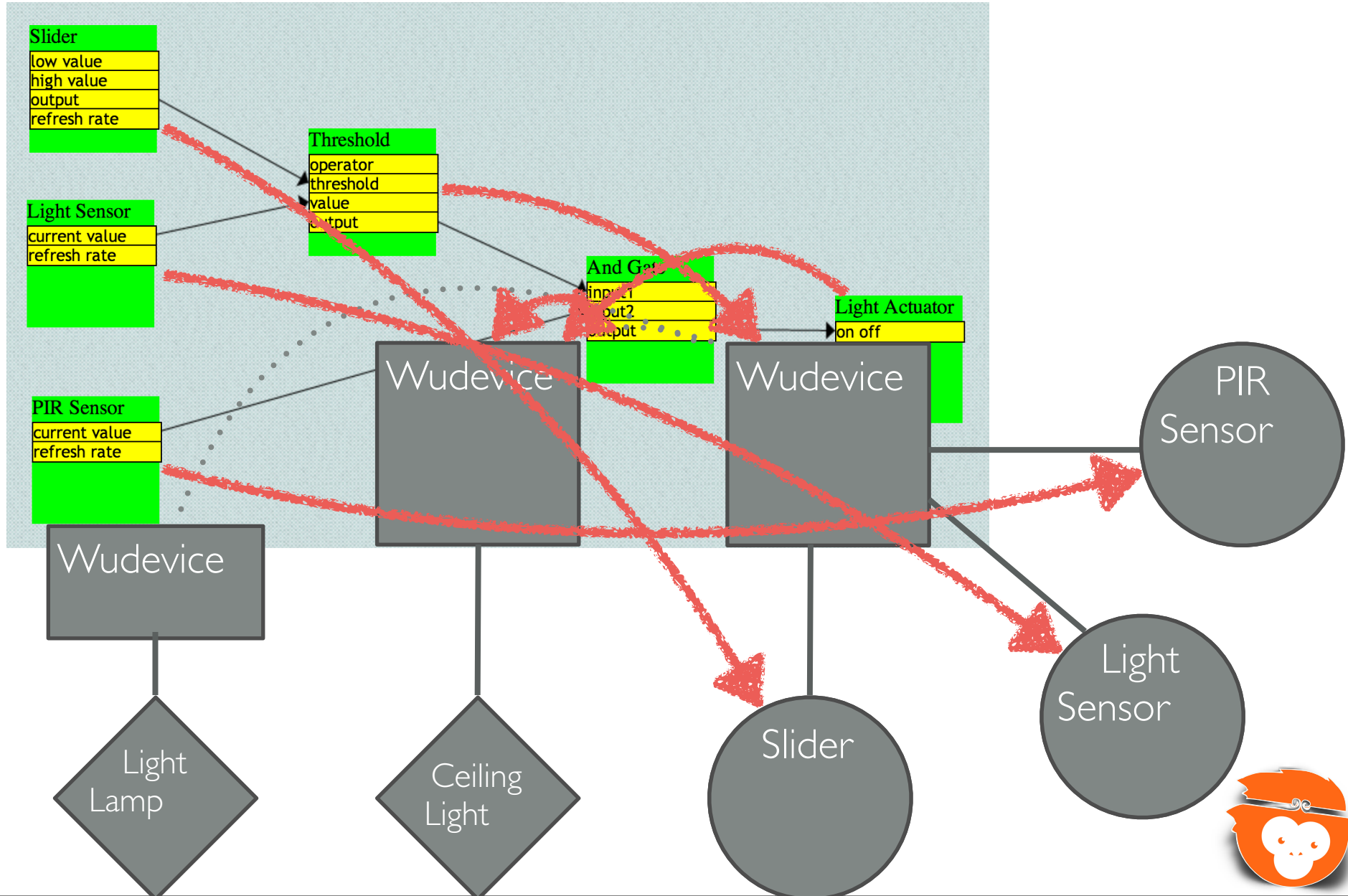


1. Master initiates the discovery to look for all connected devices and queries their NanoKong for the profiles supported.

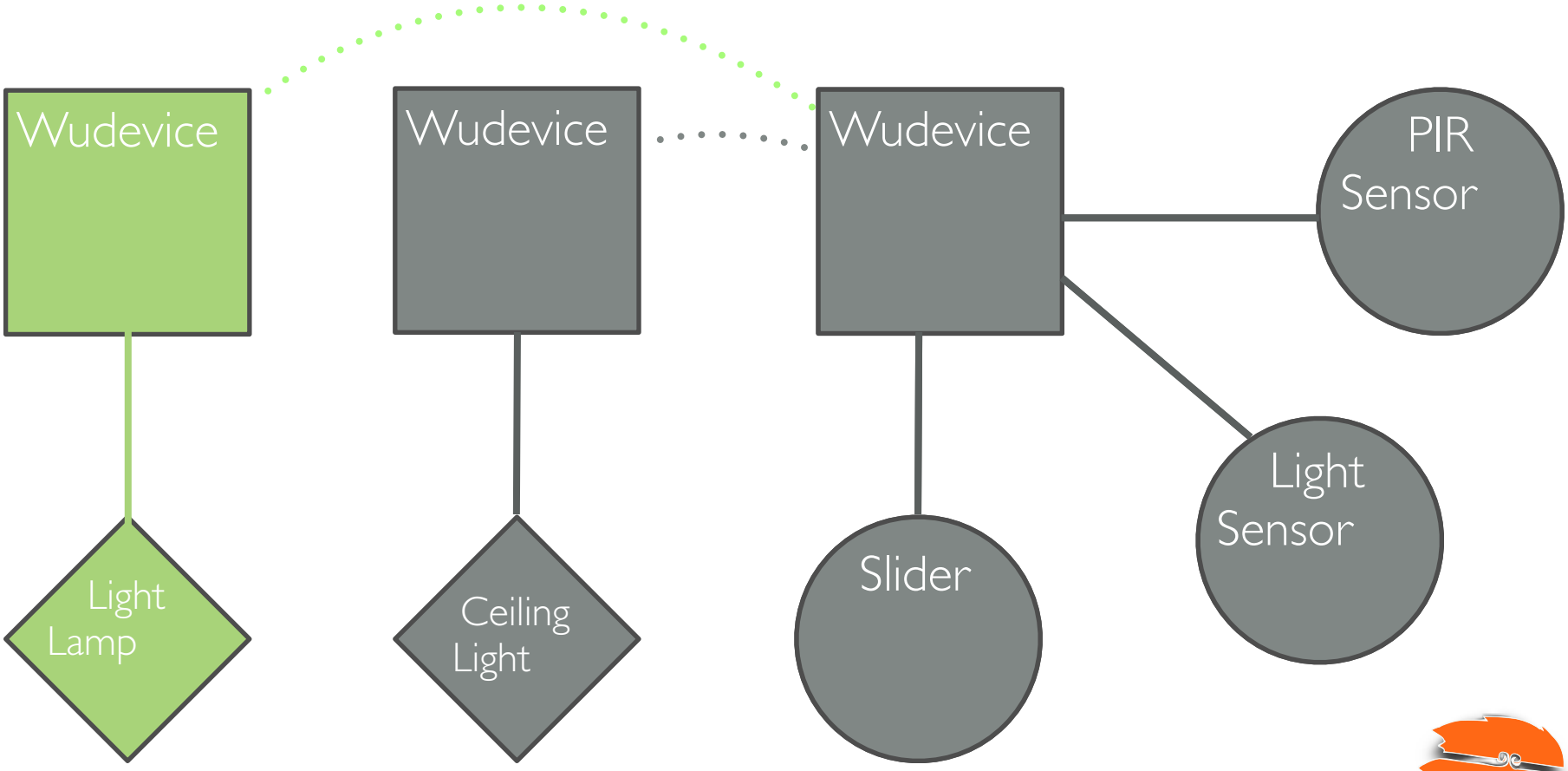
2. Master uses mapping algorithms to selects different devices for a FBP, and upload the codes


LED
Light_Actuator
Light_Sensor
Logical
Loop_delay_boolean
Loop_delay_short
Magnetic_Sensor
MathOp
New_Class
New_Class_2
Edit Component
Import
Del
Link
Save
Delete Page
first
Slider
low value
high value
output
refresh rate
Threshold
operator
threshold
value
output
Light Sensor
current value
refresh rate
And Gate
input1
input2
output
Light Actuator
on off
PIR Sensor
current value
refresh rate

Sensors, Actuators and WuDevices



Sensors, Actuators and WuDevices

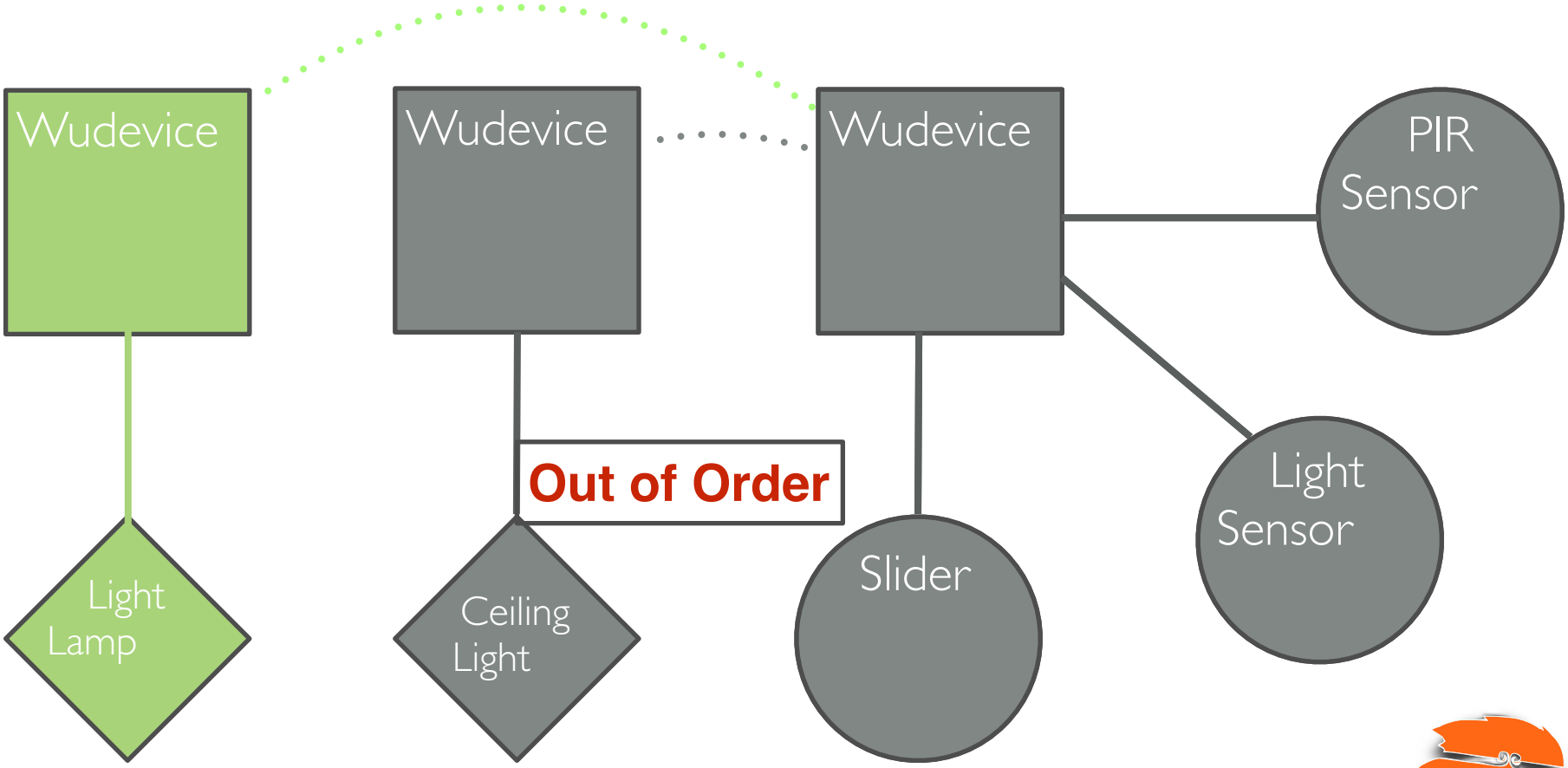


Health and idle

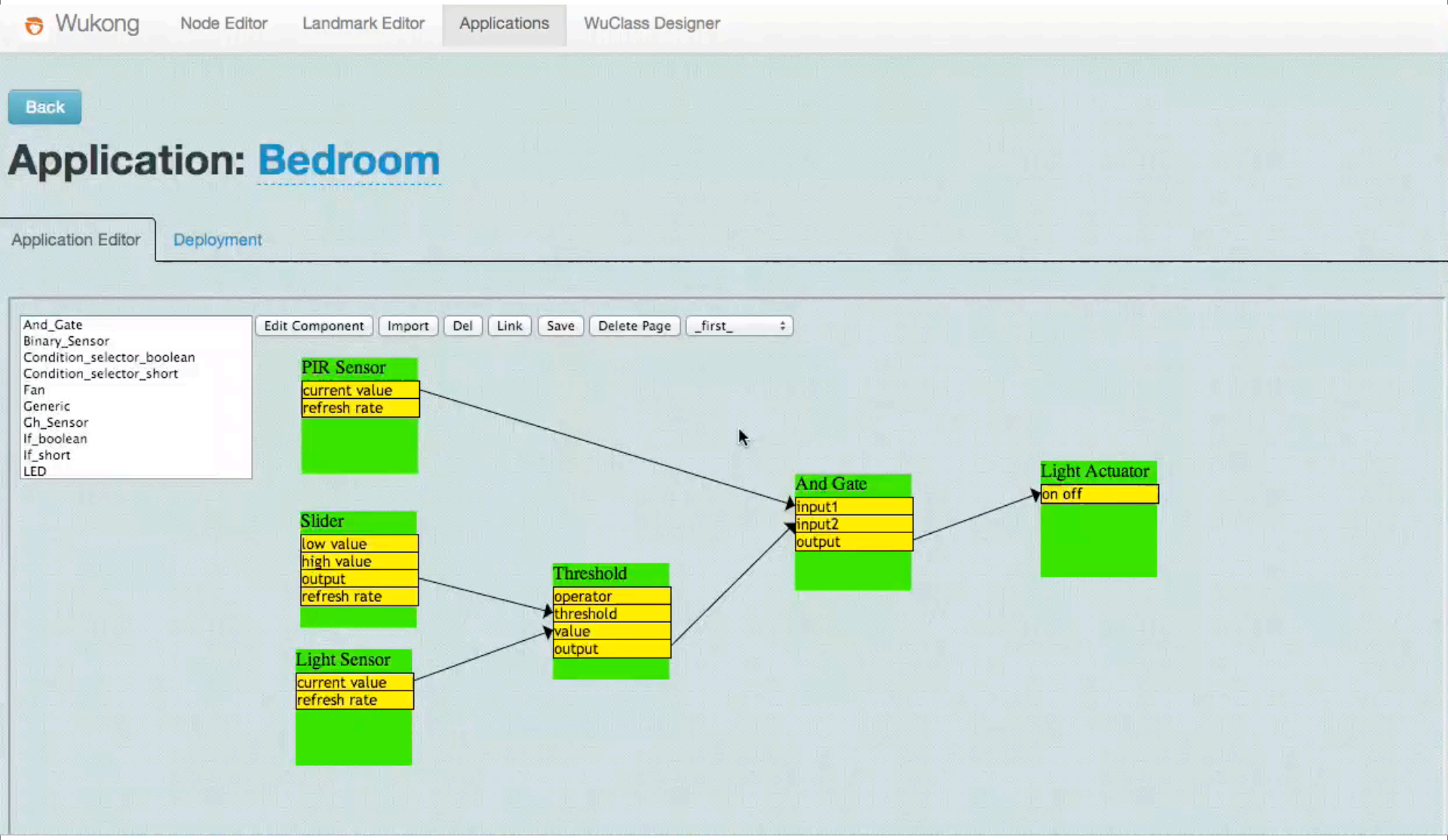
Self-reconfiguration to handle fault

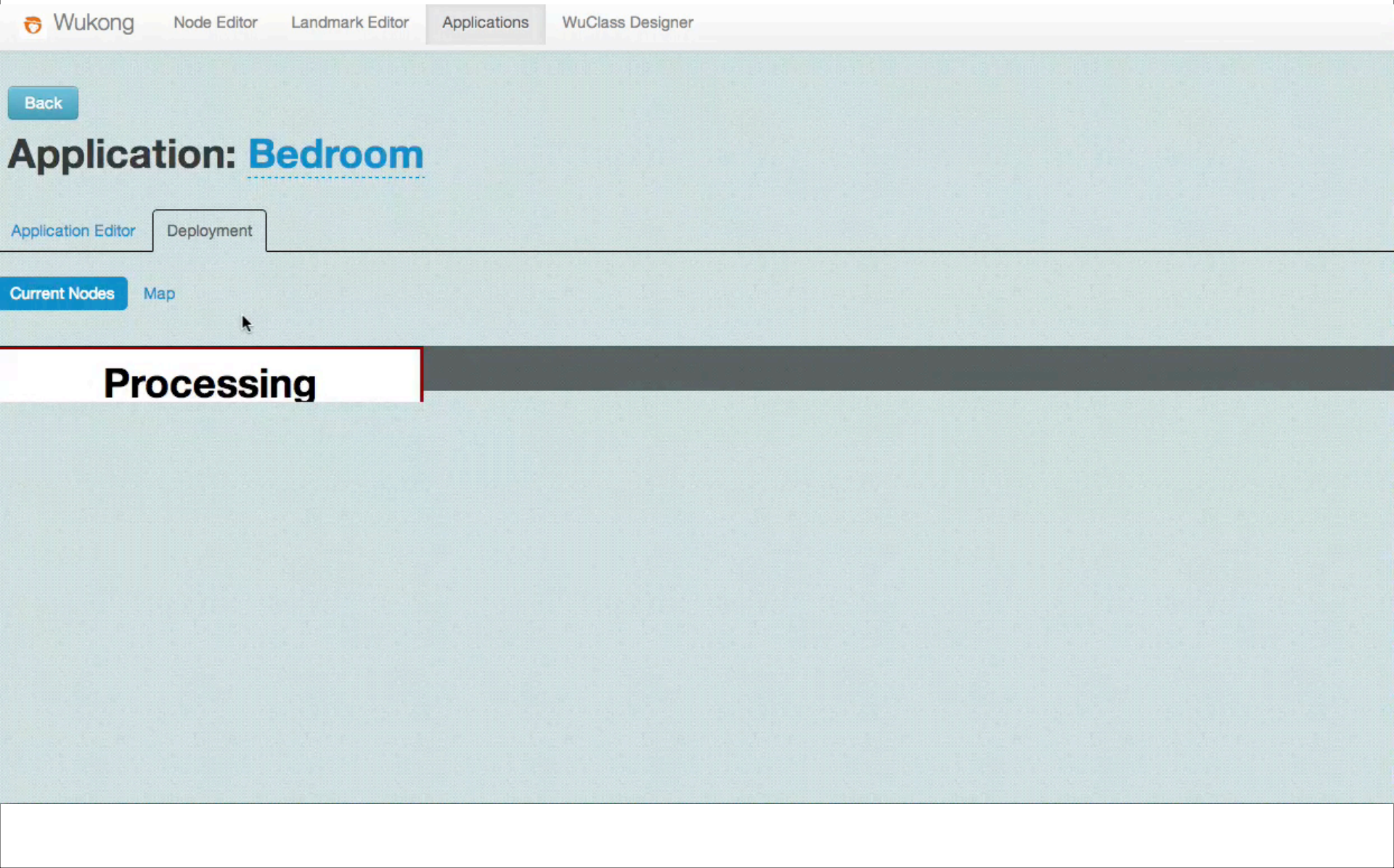
- Self-reconfiguration is a must-have feature for managing large-scale M2M systems.
- Fault Tolerance Policy defines how WuKong master handles the faults:
  - When the system is deployed, the Master checks if fault tolerance requirement can be fulfilled.
    - Example: to provide services and allow at most n simultaneous failures, the system should have at least n+1 candidate nodes to deploy the services.
  - During the run-time, the master handles the faults according to the given policy.
    - Example:
      - Policy: there should be at least one temperature sensor within the study room
      - Remapping temperature service from a failure node to a health node.

Sensors, Actuators and WuDevices



Health and idle


Wukong
Node Editor
Landmark Editor
Applications
WuClass Designer
Back
Application: Bedroom
Application Editor
Deployment
And_Gate
Binary_Sensor
Condition_selector_boolean
Condition_selector_short
Fan
Generic
Gh_Sensor
If_boolean
If_short
LED
Edit Component
Import
Del
Link
Save
Delete Page
first
PIR Sensor
current value
refresh rate
Slider
low value
high value
output
refresh rate
Light Sensor
current value
refresh rate
Threshold
operator
threshold
value
output
And Gate
input1
input2
output
Light Actuator
on off


Wukong
Node Editor
Landmark Editor
Applications
WuClass Designer
Back
Application: Bedroom
Application Editor
Deployment
Current Nodes
Map
Processing

Wukong | Node Editor | Landmark Editor | Applications | WuClass Designer

Back

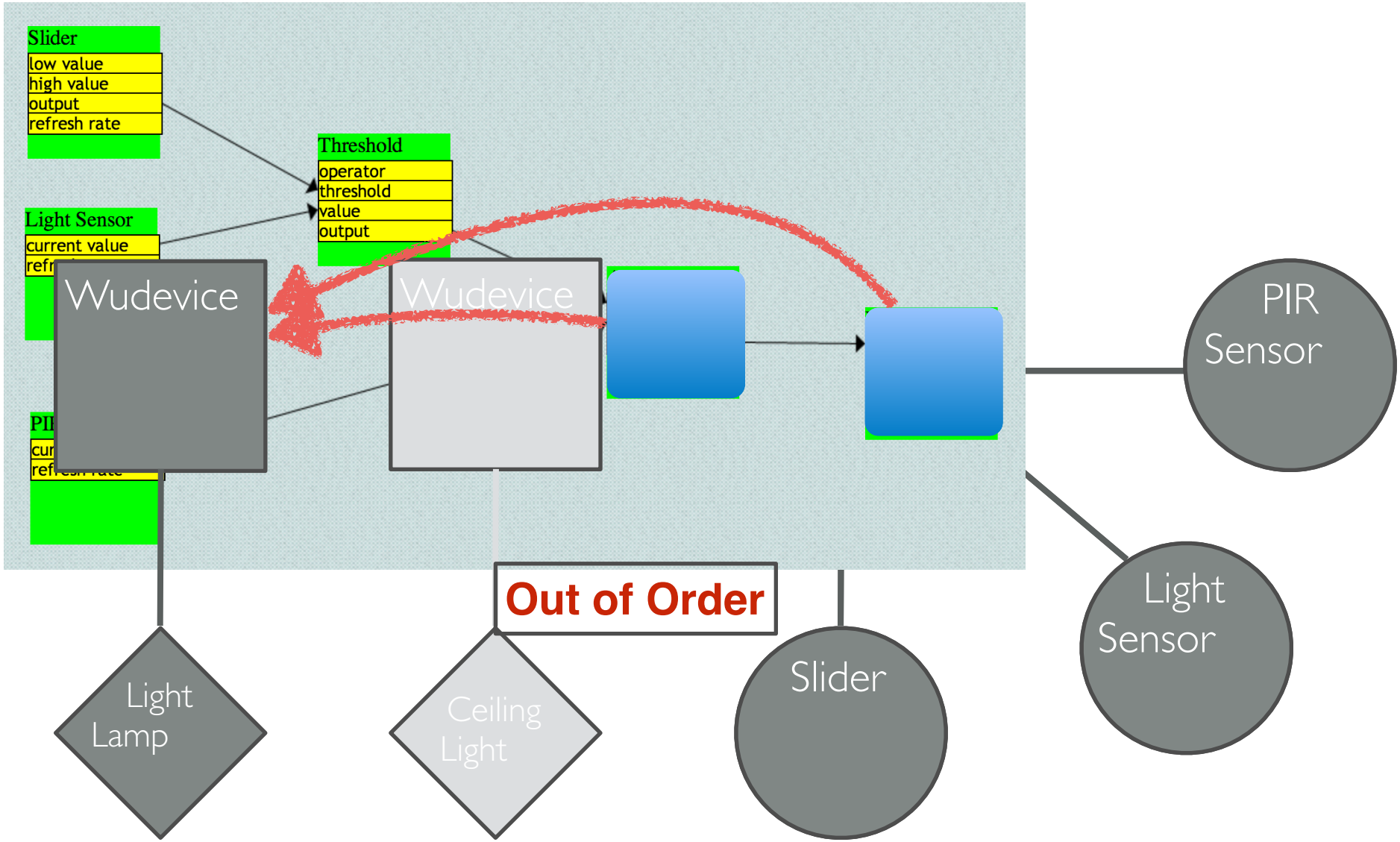
Application: Bedroom

Application Editor | Deployment

Current Nodes | Map | Deploy

#	WuObject Name	Node Id	Port Number
0	Slider	4	3
1	Light_Sensor	4	2
2	Threshold	4	5
3	(Virtual) And_Gate	2	13
4	PIR_Sensor	4	1
5	Light_Actuator	2	1

Sensors, Actuators and WuDevices



Up and running

Think Logical; Let Master Get Physical

- M2M Paradigm Shift: *Think Logical; Let Master Get Physical*
  - Sensor applications should be re-usable by different sensor devices and users
  - Users should have the freedom to choose the devices they want to use.
- WuKong provides an open system for developers to build and users to adopt new sensor applications.
- We want to make both user's and service provider's lives easier.
  - For users, we want to allow different types of applications, under various context and policies, to be easily configured and deployed on a set of sensors
  - For service providers, we want to make M2M operations simple and flexible to improve industry's operational cost