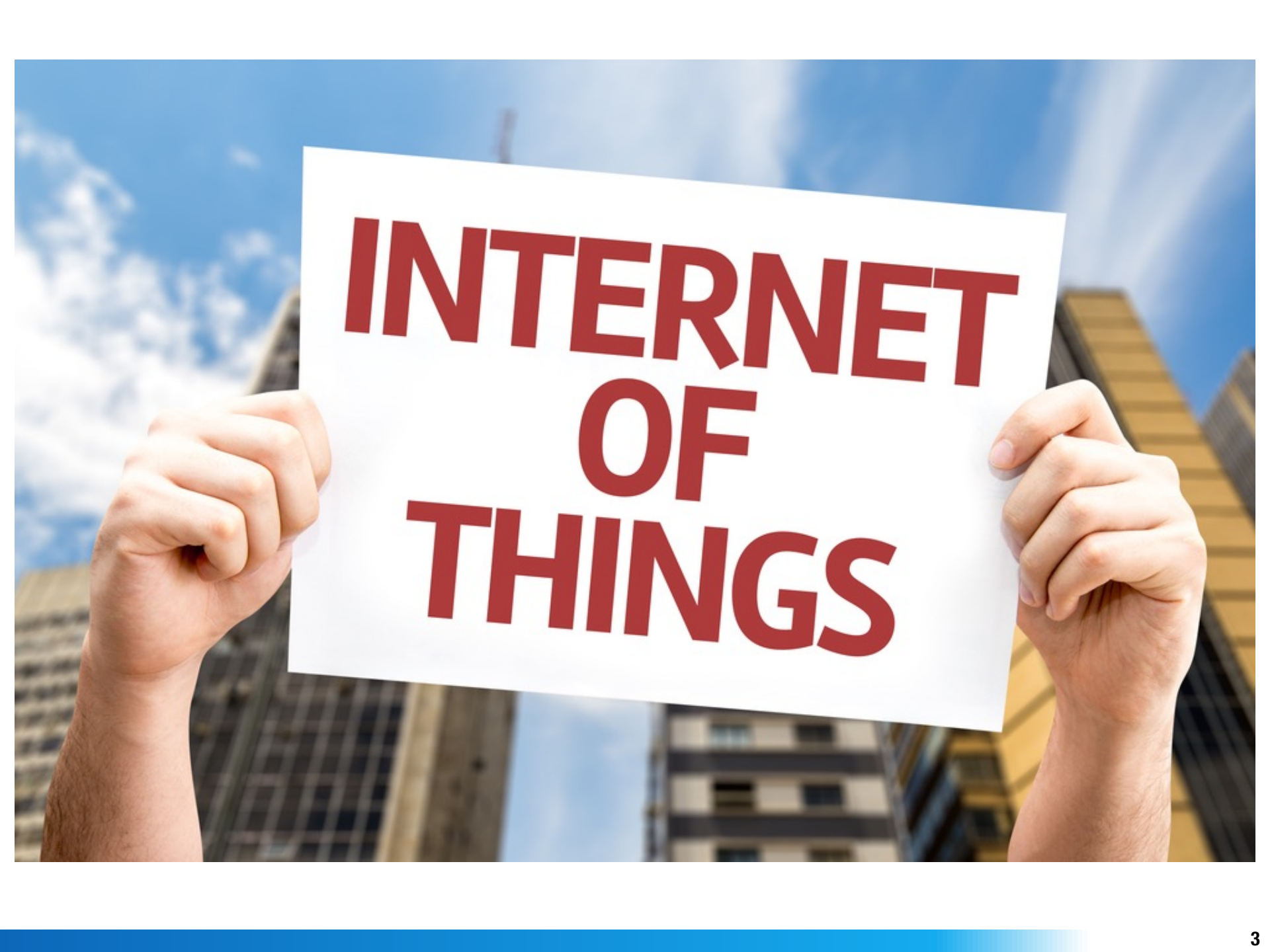


Internet-of-Things and Middleware

► 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

A person's hands are holding a white rectangular sign with the words "INTERNET OF THINGS" printed in a bold, red, sans-serif font. The sign is held up against a background of a city skyline with several tall buildings under a bright blue sky with scattered white clouds. The perspective is looking up at the sign and the buildings.

**INTERNET
OF
THINGS**

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

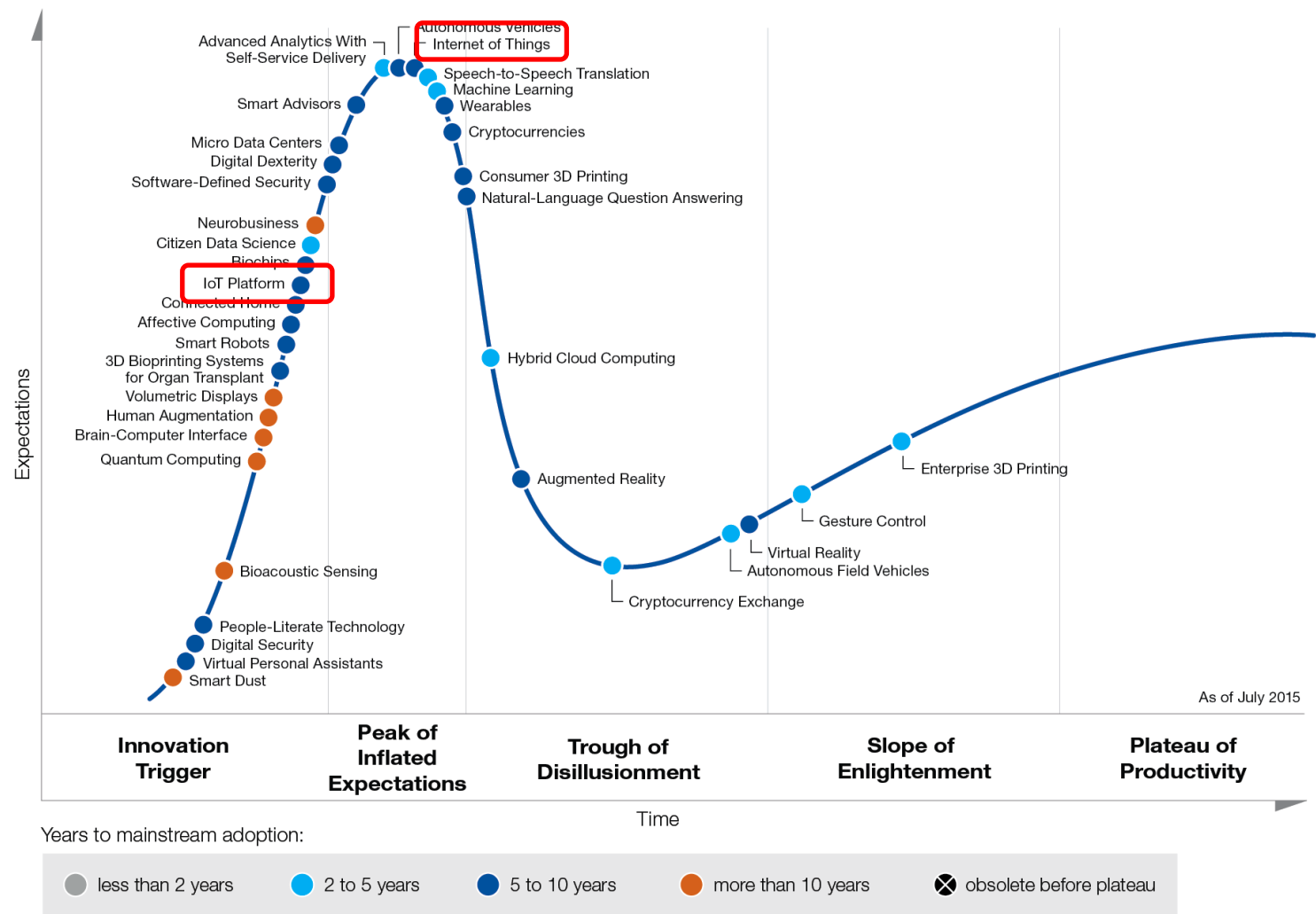


I THINK MY NEST SMOKE
ALARM IS GOING OFF.
GOOGLE ADWORDS JUST
PITCHED ME A FIRE
EXTINGUISHER AND AN OFFER
FOR TEMPORARY HOUSING.



Gartner Hype Cycle and IoT, July 2015

Emerging Technology Hype Cycle



Smart Applications using IoT

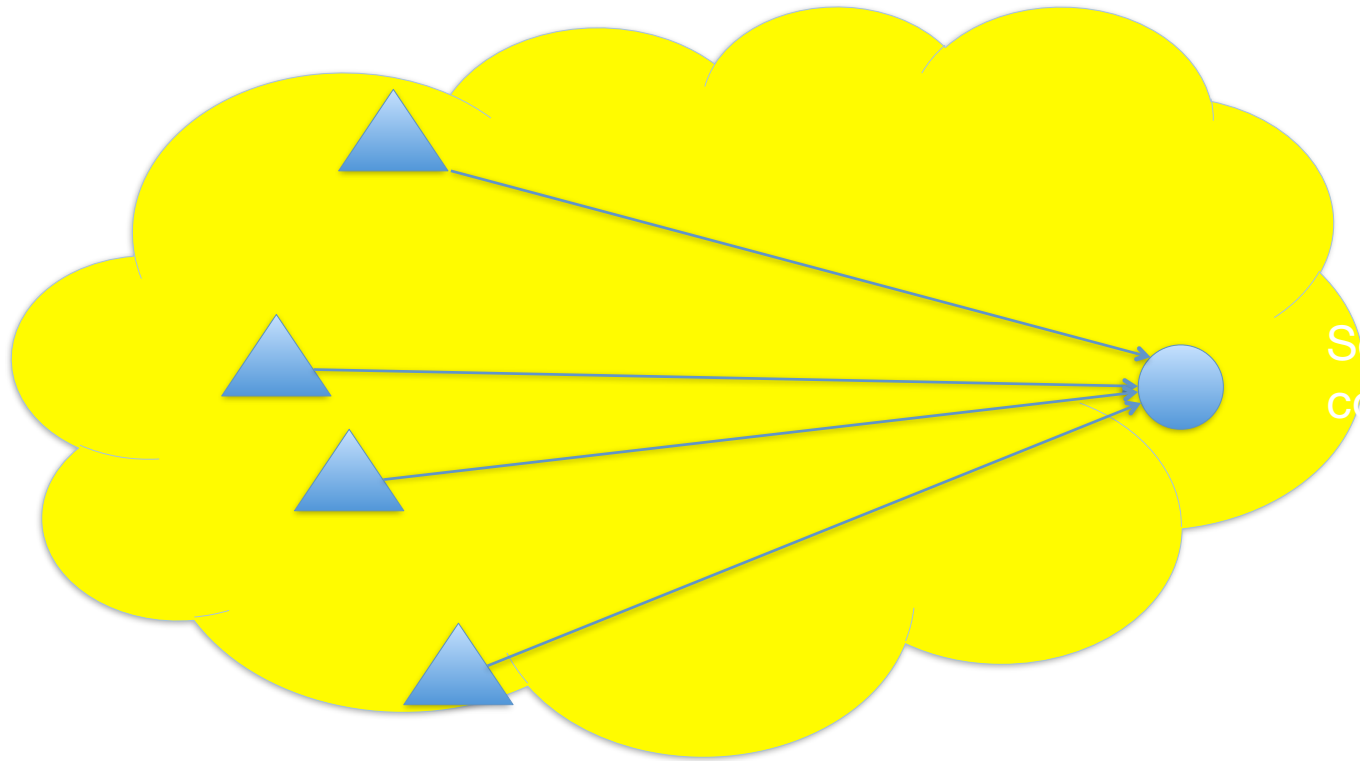
Making Things Smart

- ▶ 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

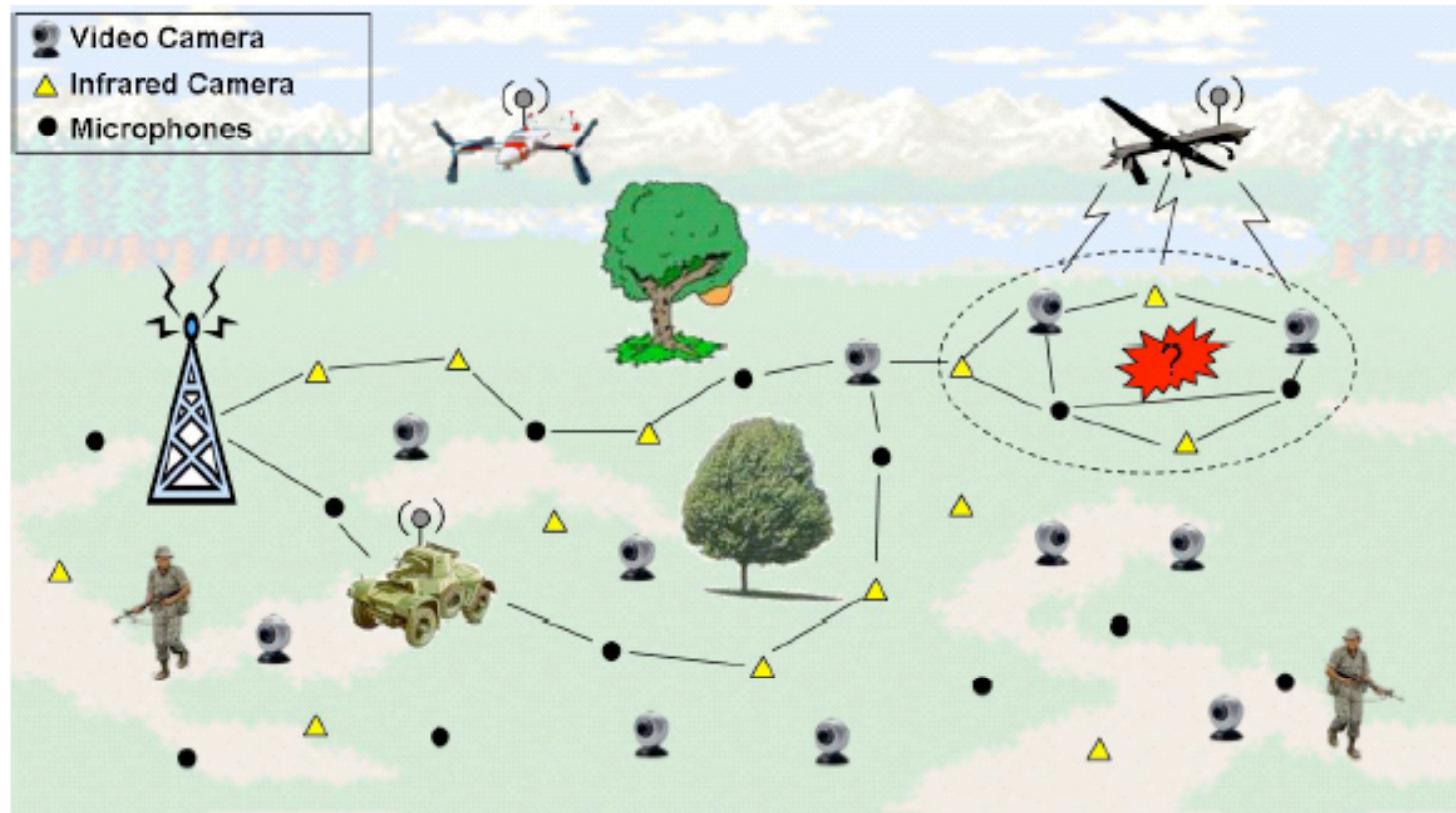


Who's who?

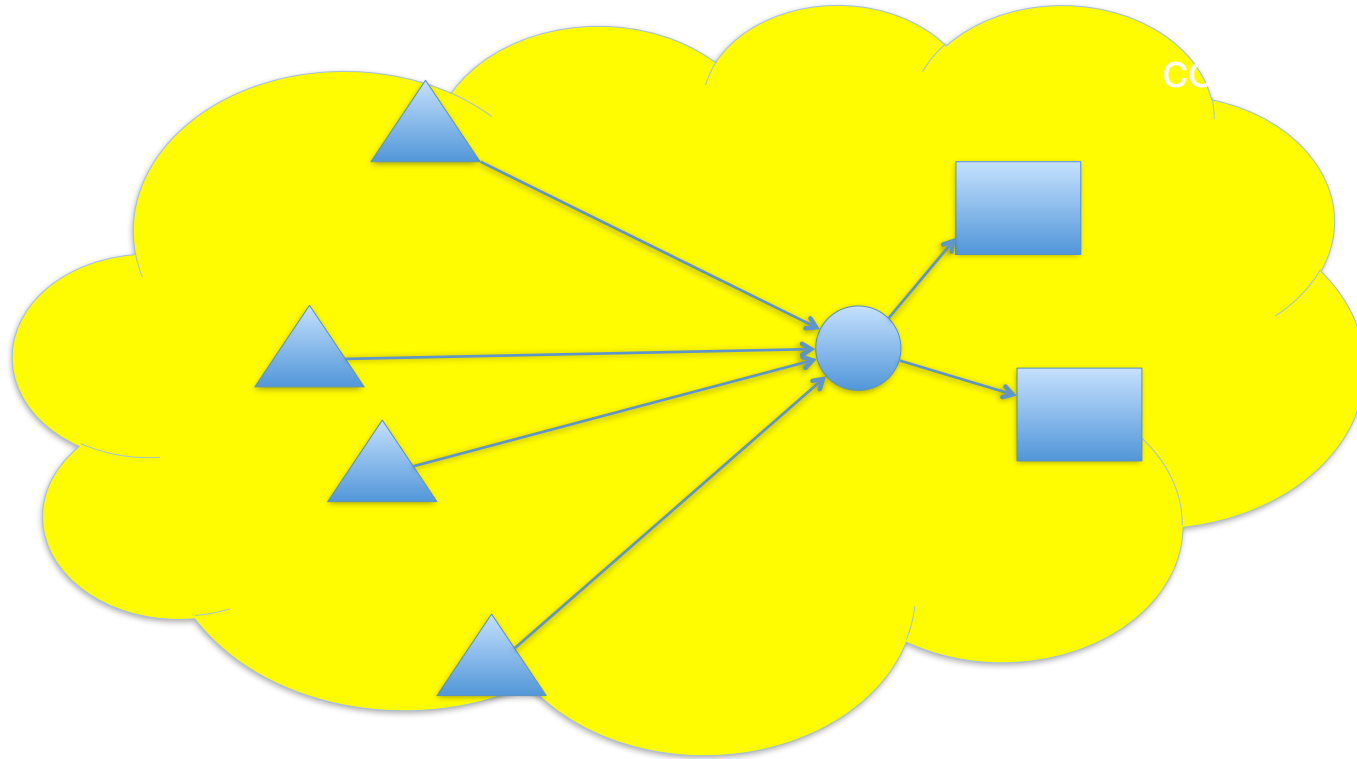
(Wireless) Sensor Networks



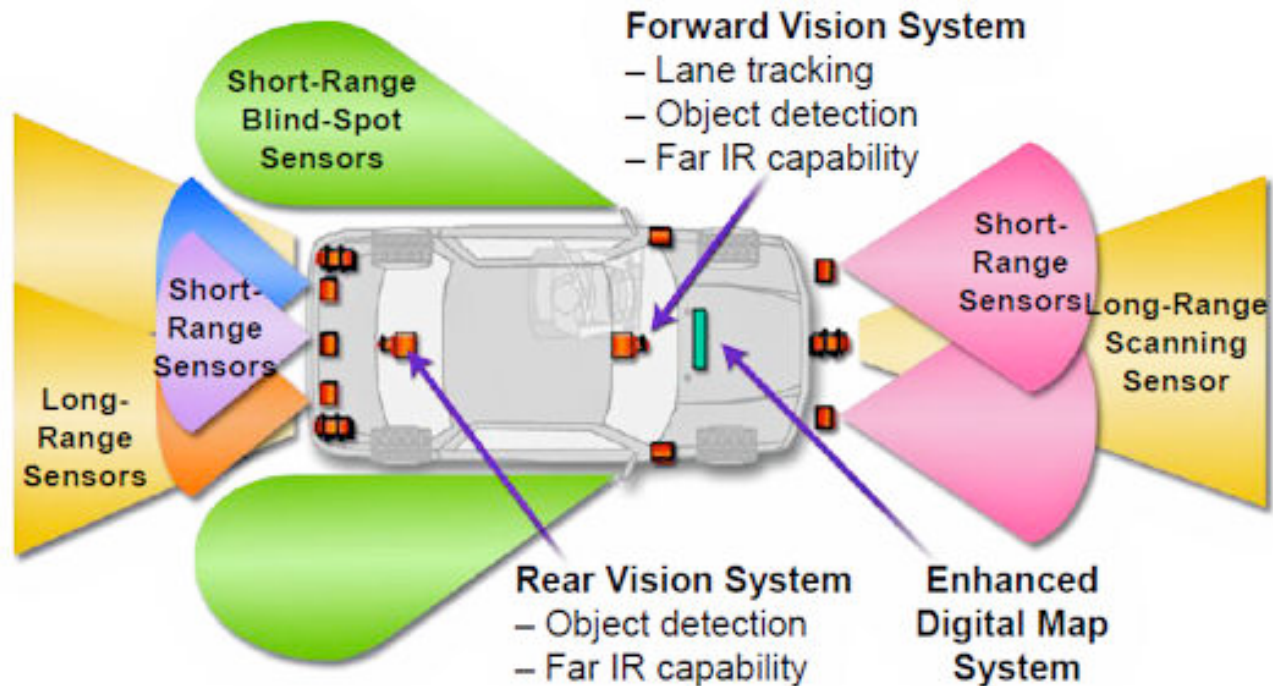
(Wireless) Sensor Networks



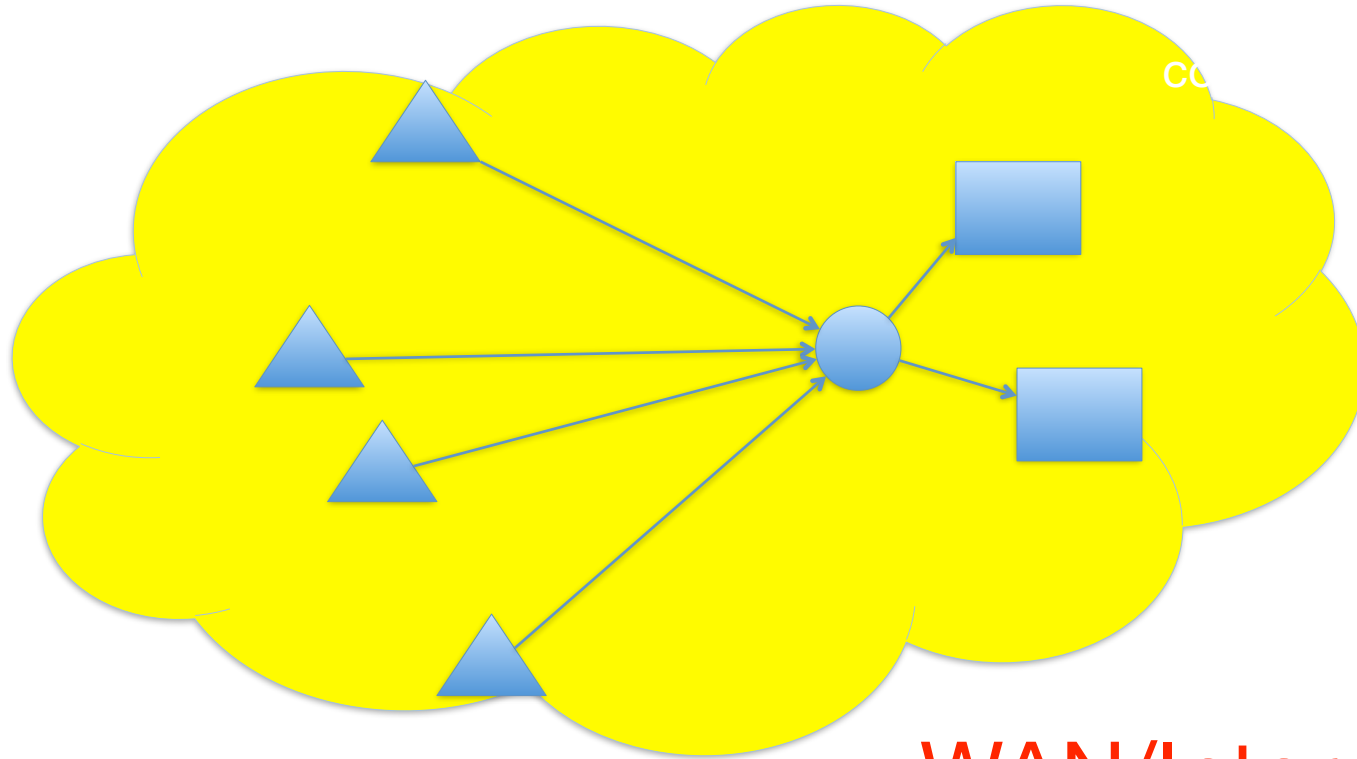
Cyber-Physical Systems



Cyber-Physical Systems



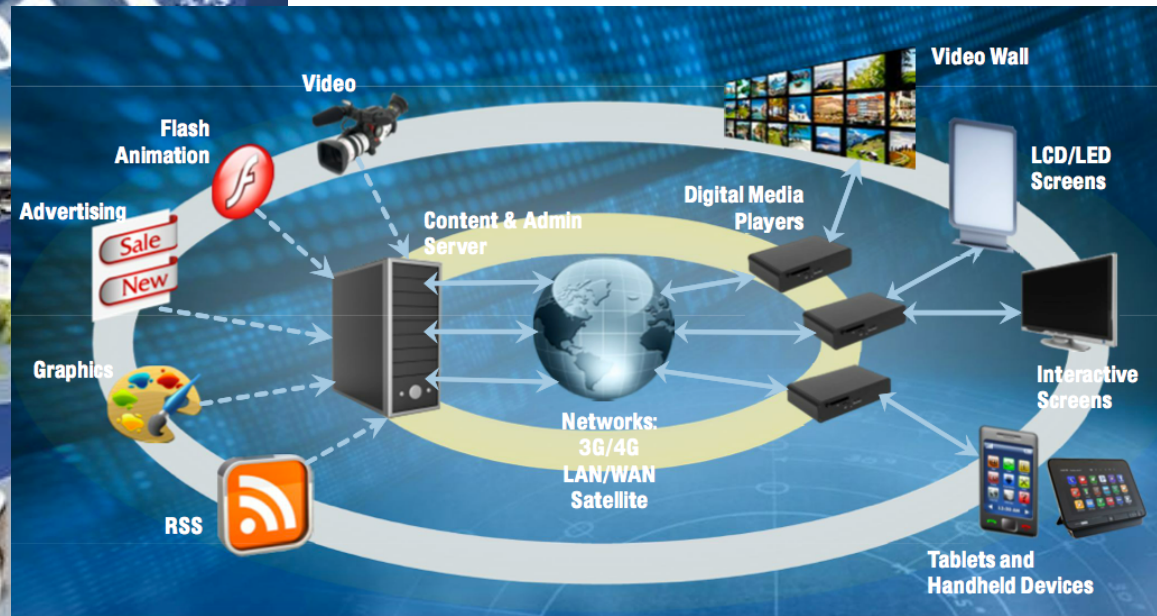
Internet of Things (IoT)



WAN/Internet

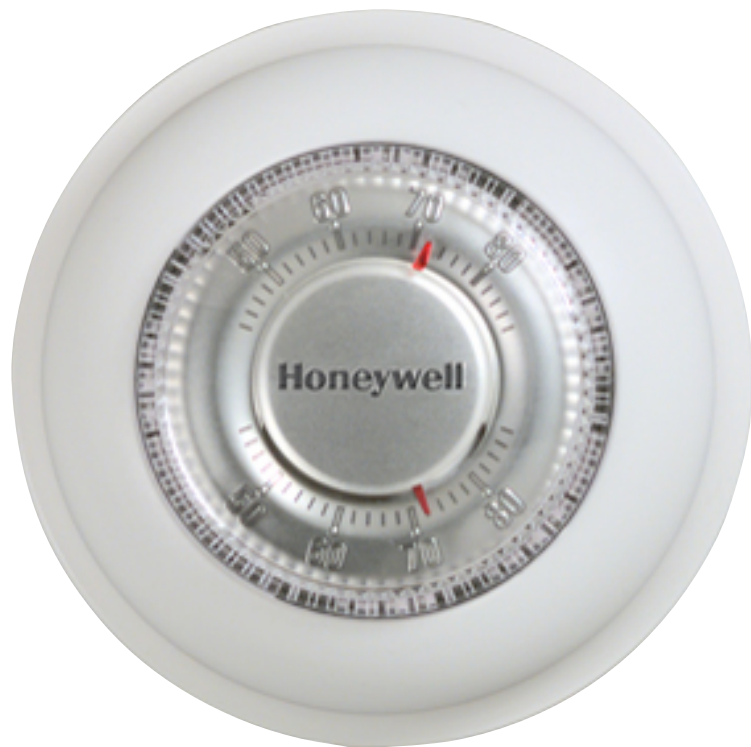


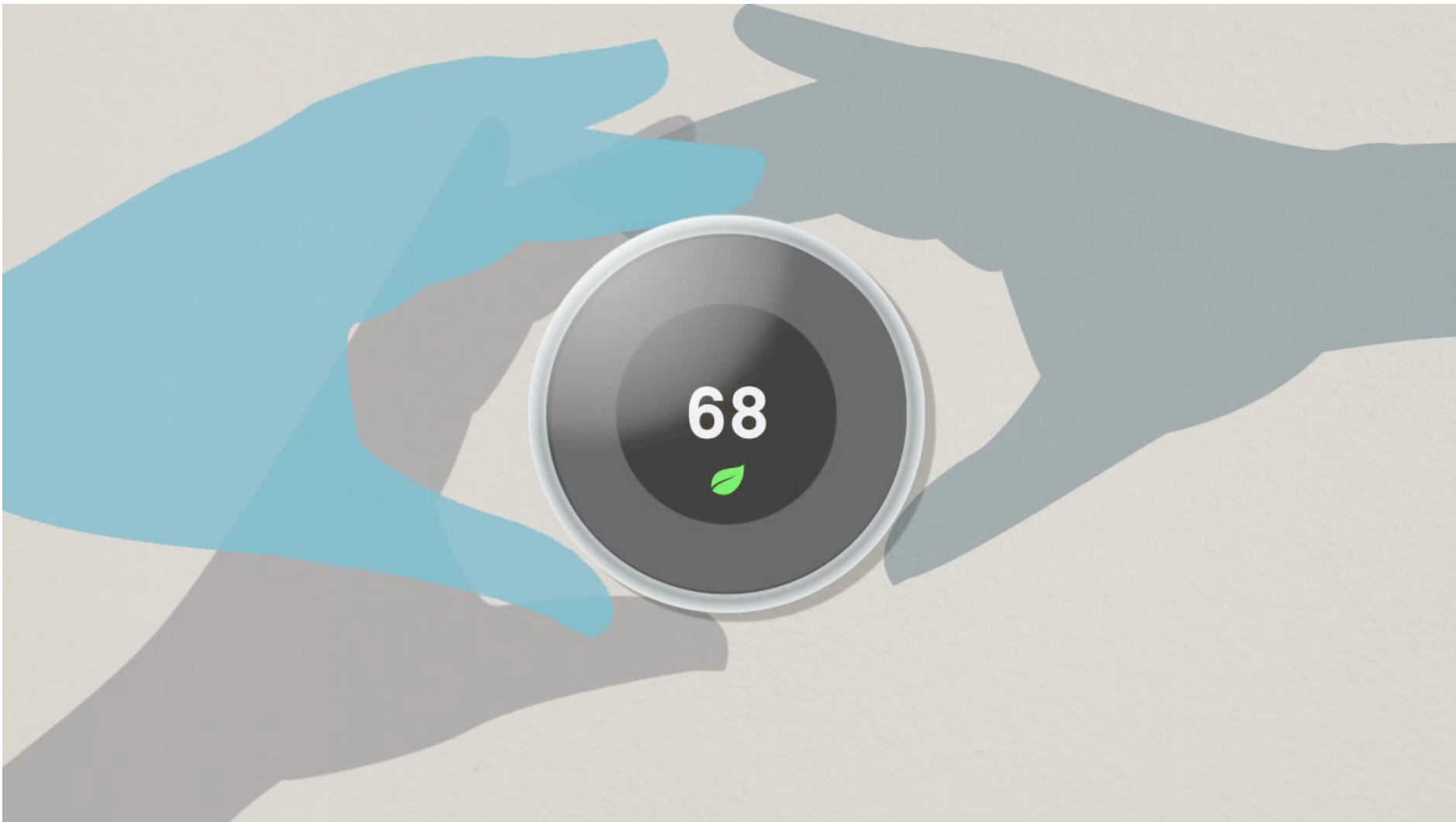
Internet of Things (IoT)



Machine-to-Machine Systems







物聯網，聯網物，還是聯物網

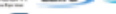
Applications (Verticals)

Personal Devices	Lifestyle	Connected Home	Industries	Industrial Internet
Wearable Computing pebble cookoo rufon i'm GLASS RINGLY striiv APX MOTI OpenSense M wearable intelligence VUZEX	Sports SOBI Brain Sentry BIKESPIKE InfoMotion swingbyte HAMMERHEAD	Automation Quirky Radiator Labs netatmo LEVITON SmartThings Ubi nest LIFX gecko ecobee Advanced.umenics vivint.SAVANT vera INSTON CHAMBERLAIN PROPER LIGHTIFY somfy	Retail bytelight euclid proximity SAVIELO malaga GIMBAL boni Roshni PassX PERCH PAPER DETAILER book stream	Robotics Double Robotics ALDEBARAN HYPERVECT XENEX ROBOTEX EMPIRE ROBOLOX KUKA iRobot omni automation KUKA VEX ABB LIQUID ROBOTICS Jibo
Fitness GARMIN senSoria amigo iFIT JAWBONE HE HISSFIT VALENTI BASIS fitbit tomrTom Atrios LifeBEAM wahoo	Cooking Smart Die Scale ANOVA drop blossom iDevices THE ORANGE CHIEF CO. pontry n.miku	Monitoring lapka sense birdi BlueMaestro SUPERMECHANICAL leeo knot CUBESENSORS tado° ambient	Payment/Loyalty Square shopify PayPal ACS VerFone LevelUp belly payleven con	Drones/Aerospace 3DR KIM CYPHY Airware SKYCATCH spire Parrot Skybotix
Health LUMO nanosensor vcsay kinsio HAPIfork soundhawk SPIRE Withings QUANTUS Lively LIFECARABO tenses Amc10 blueantony hello AliveCor iHealth remee aelon corventis TELCARE	Pets Whistle PetPace pintofeed PetHub etaggy BISTRO hoytag PetChats Petcube Inactive Petnet gibi Petzlo	Security HomeMonitor canary ring dropcam butterflyeye Lockitron August SCHLAGE RAVEN Kwikset globosense genie Ue OP LINK Goji scout nSmartAlarm KeyPlex SmartScale	Healthcare VISI Senseonics STANLEY VITALITY MedMinder GoleTracking MedSignale AchereTech VERSUS CENTRAK Intelligent Sotera RayGen	Green-tech BigBetty enlightened Smart Genevo compology AMPY
Family FILIP sprouting ovuline AmberAlert greatcall OWLET Secur BELLBEAT mimo Glow pocketfinder	Toys KAROTZ UBOOLY MAKIES atoms seeBO	Tracker Chipolo linquet locca! TrackR	Automotive Zobe fite INRIX navdyDELPHI Airbiquity dash waze OpenXC	3D Scan/Print MakerBot dp Stratasys HOREL 3D formlabs M30 ph matterport FUEL3D AIO NEXTENGINE eRepRap ULTIMAKER
	Music/Art/Video ROLI CATCH GoPro Narrative	Hub Homey Control LOWE'S zonoff STAPLES NEXIA muzzley wink	Infra-structure waveLink iBeam kisi Johnson Controls Trimble Robin Schneider CH2M HILL WITINITY	Smart Grid GRIDNET e-on SilverSpring SMART GRID CLUSTER Itron Trilliant
	Garden plantlink BITPONICS radio EDYN Greenbox Koubachi		Agri-culture adapt-N Zed	Asset Tracking asap vloc NESH SYSTEMS Kalamata ekahau RACALPACIFIC Inspiro CUBIC

Platforms & Enablement (Horizontal)

Connectivity/Dev Platforms	Software/Data Platforms	Open Source	Personal Interfaces	Security	Corporates
scark kynetix pinoccio ioBridge electric Imp Ayla Networks EUROTECH resin.io Symplic TESSEL bluecity	EXOSITE icontrol thingsquare carriers Keen IO SeeControl Uhings ConnectHO NewAer BERG Axeda Yaler.net RacoWireless SpaceCar IFTTT greenWAVE wot io ZALAR Cyberlighting altflux Yo ThingWorx DN2K Xecopower Joo IOTOL thingful CANDI bugswarm Tempolq evercam.io covisint Jasper GroveSystems ETHERIOS PubNub INFLURA SensorCloud xively	webinos AIJoin openHAB nimbix.com OPEN INTERCONNECT KFAI Thingspeak GRID2HOME SAFECAST placemeter Motionloft	NeuroSky Rapsheet In Interaction wit.ai LEAP gestigon Speech TRAILBLAZE LINTUONE aplai EMOTIV Malsuda Reemo Oculus	inside SafeNet utimaco escrypt gemalto BASTILLE NETWORKS MOCANA	amazon hp LG intel htc PHILIPS IBM SAMSUNG Google WIND RIVER MOTOROLA belkin DELL BOSCH NATIONAL INSTRUMENTS ARM LogMeIn Microsoft Honeywell TRANS INSTRUMENTS SONY Atmel SIEMENS QUALCOMM CISCO TOSHIBA SHARP

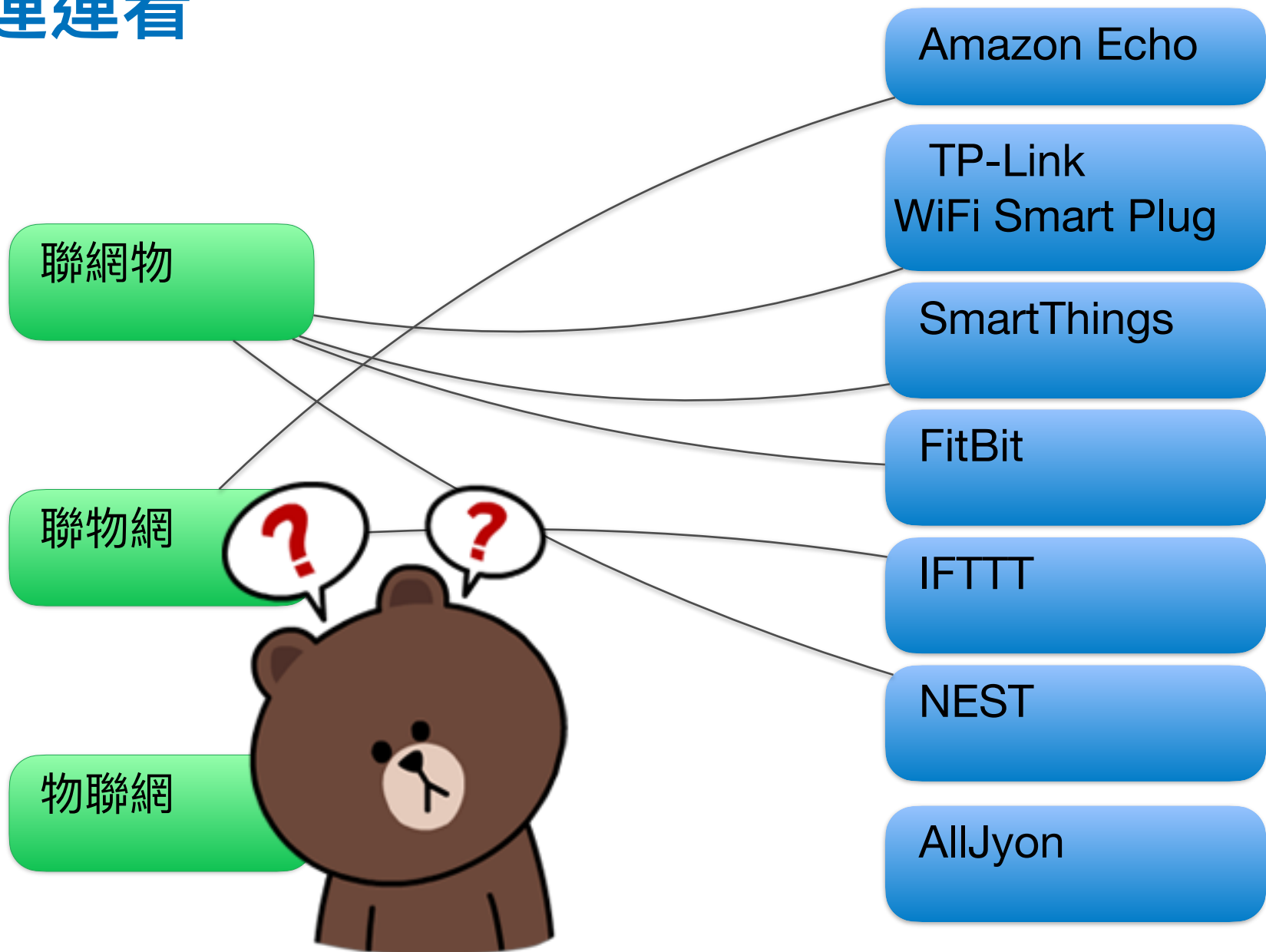
Building Blocks

Protocols  2G 3G 4G LTE CoAP 6LoWPAN LWM2M BITXmI	M2M Networks 	Portable WiFi 	Telecom 	M2M 			
Cloud 	Mobile 	Processors/Sensors 	Parts/Kits 	Services 	Incubators 	Funding 	Distribution 

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



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

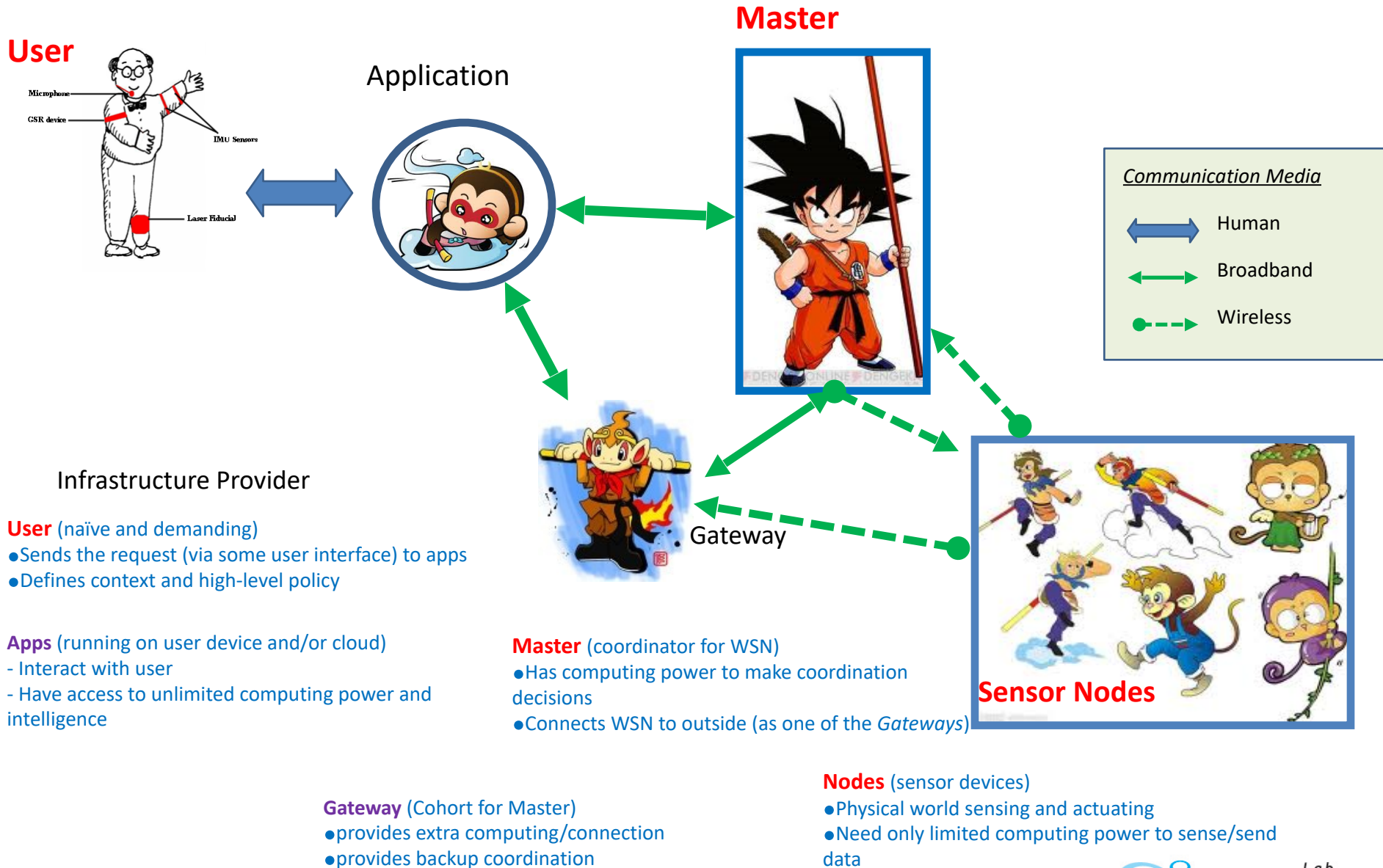
▶ Human Entities

- 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

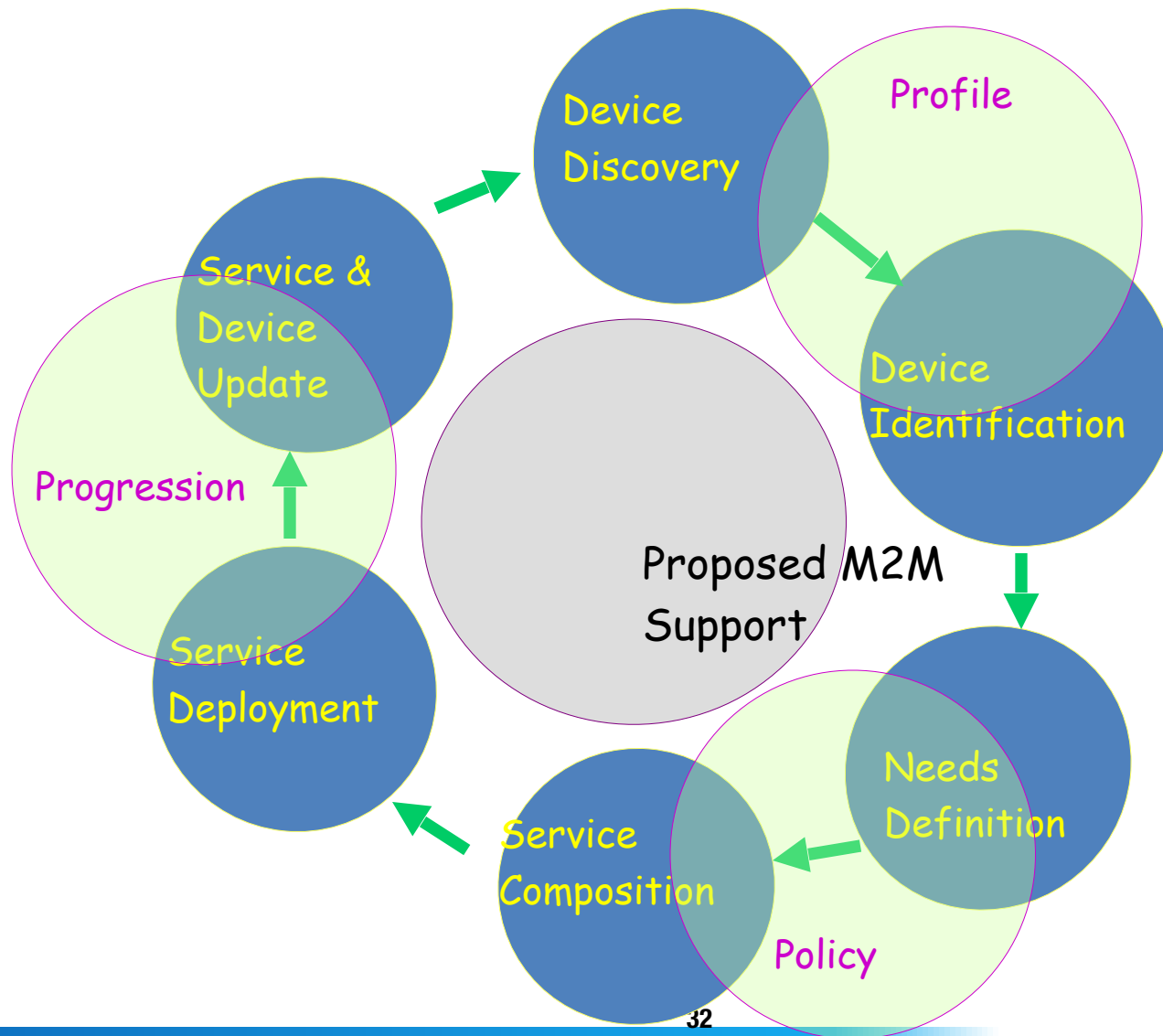
▶ System Entities

- 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

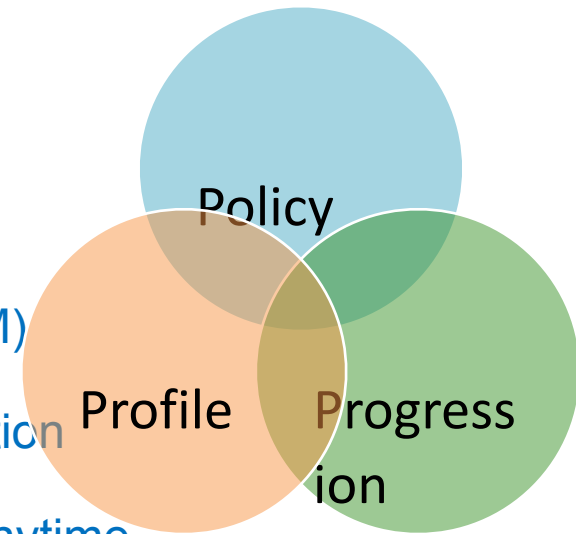


A New Paradigm: Tri-Framework on M2M development cycles

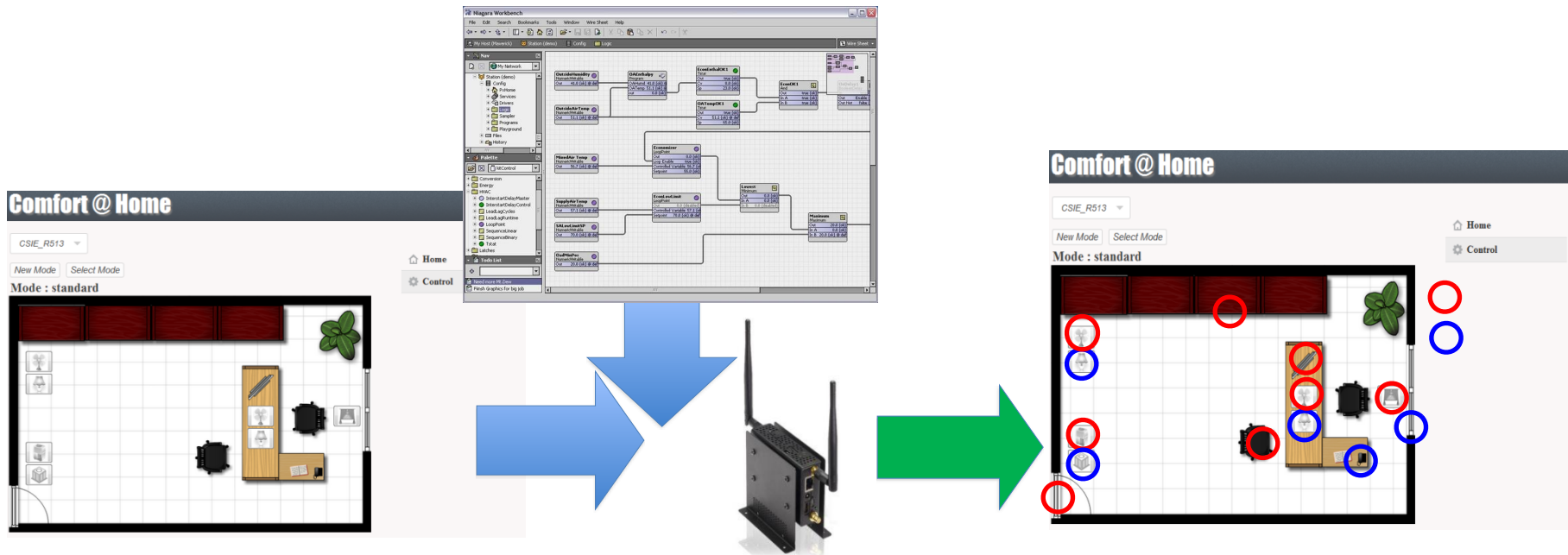


Wu-Kong : Software Perspective

- ▶ 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.

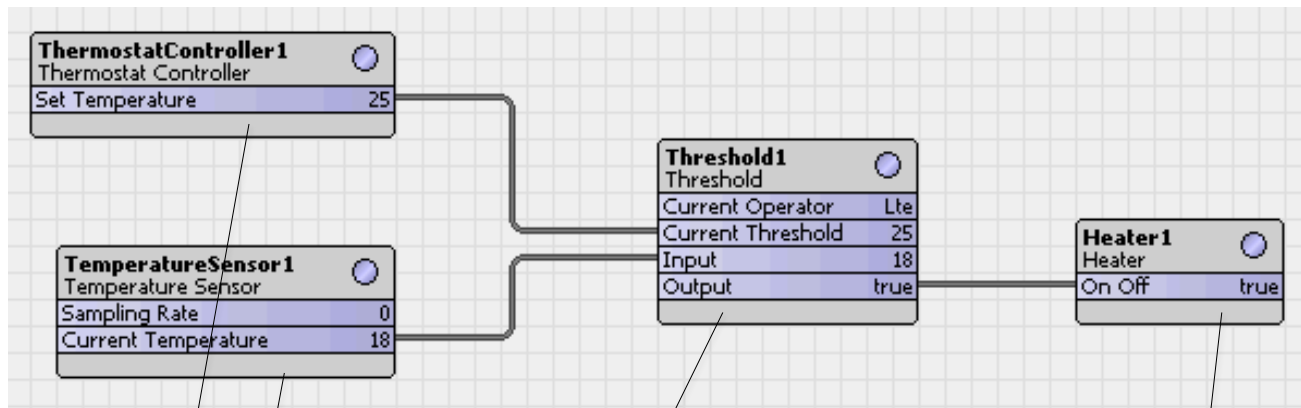
WuKong : Profile Framework

- ▶ 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.



Logical



Physical

Candidate1,2

Candidate1,1

WuClass 1

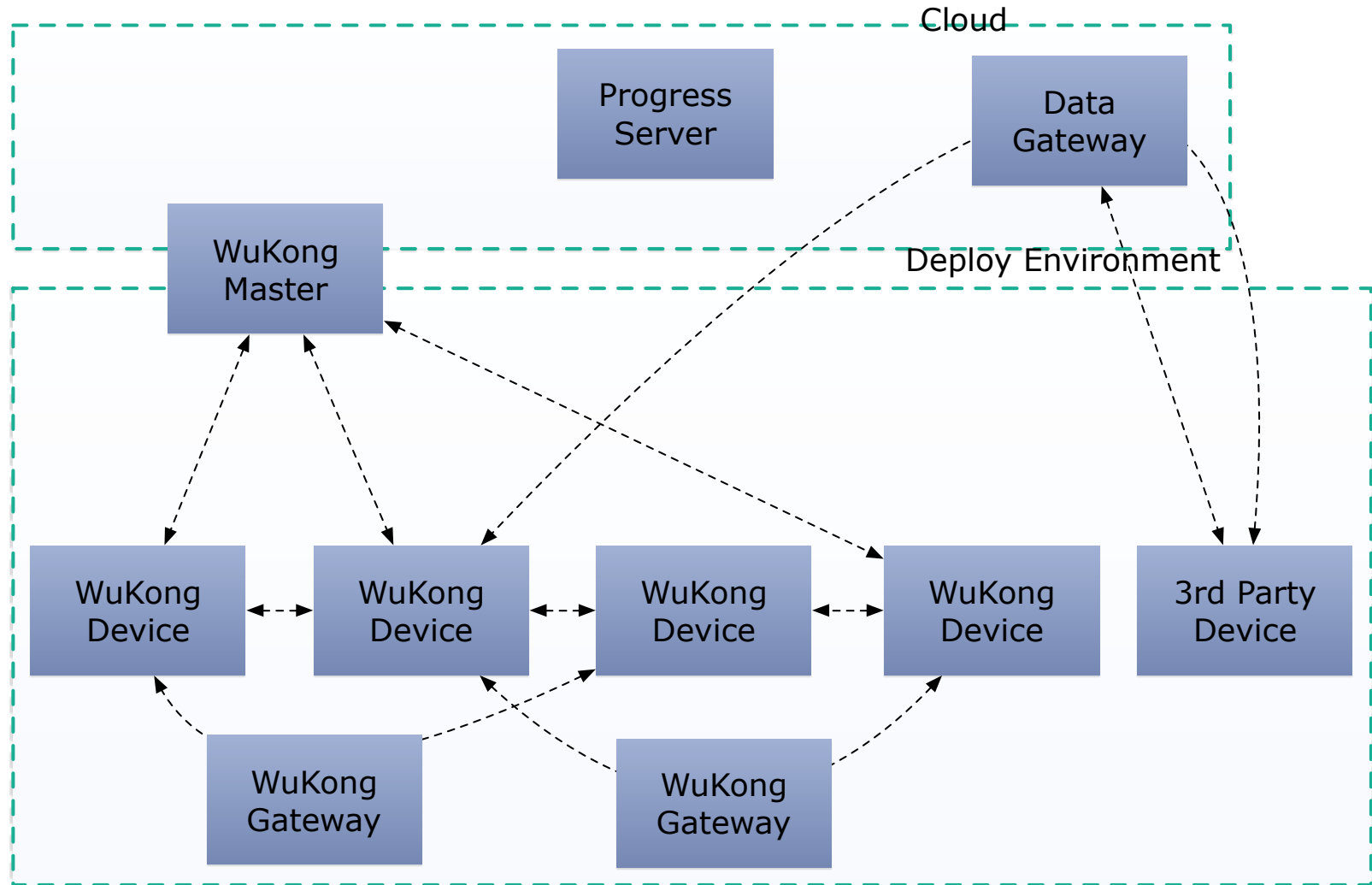
WuClass 3

WuClass 4

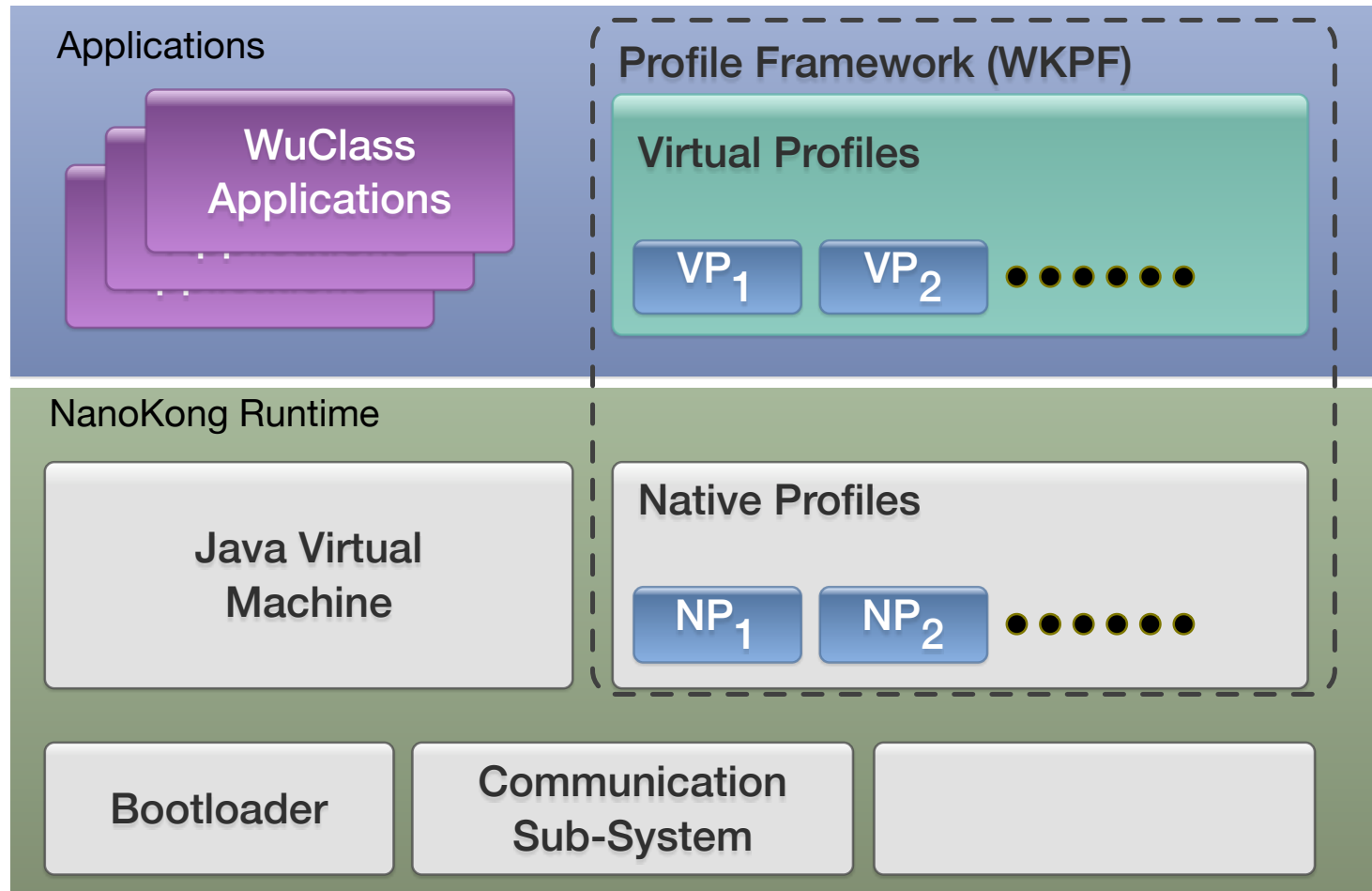
WuClass 2

Cost: $C_{31,41}$
Constraints:
 $Q^1_{31,41}, Q^2_{31,41}$

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

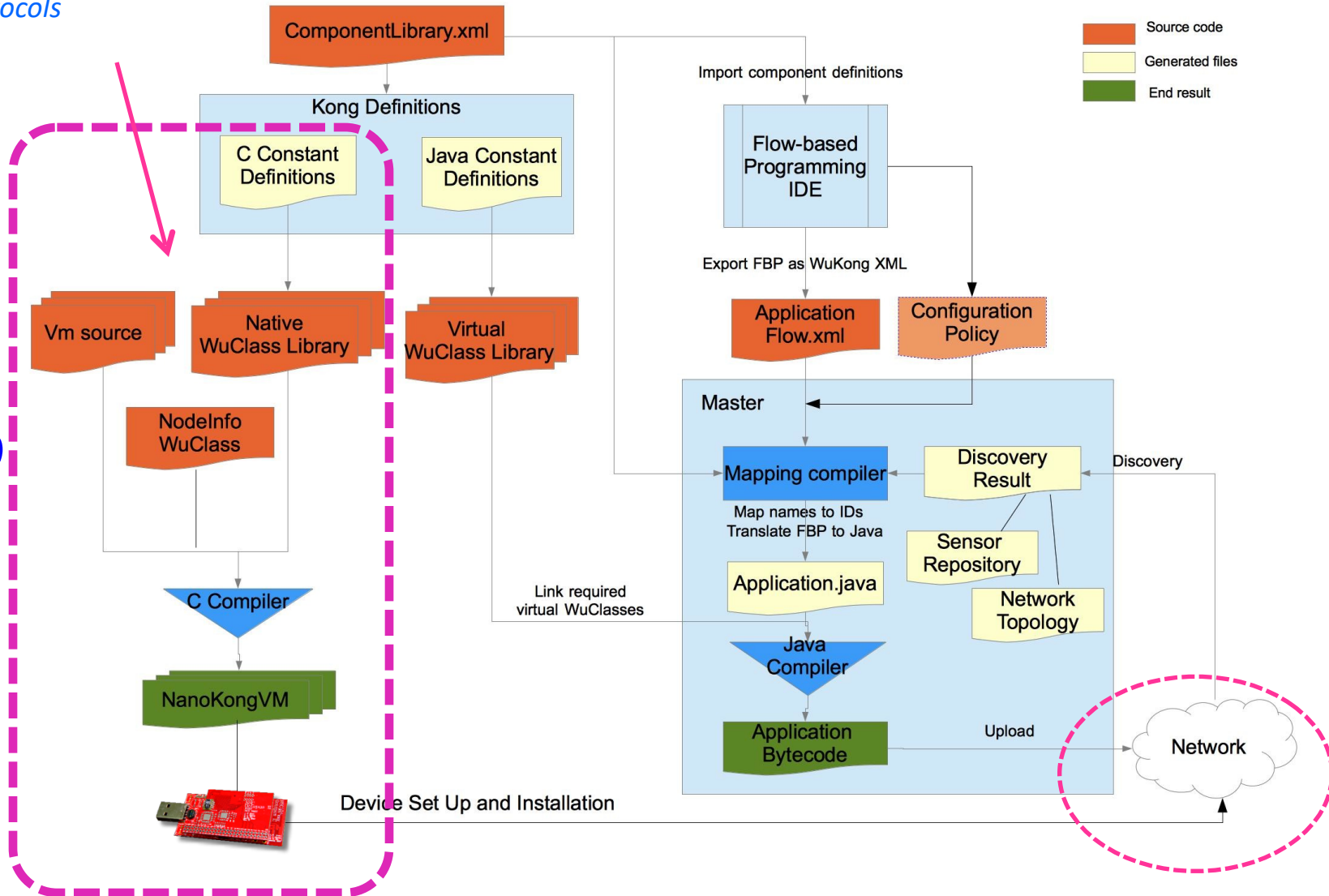
3. Build a working M2M using WuKong Master

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

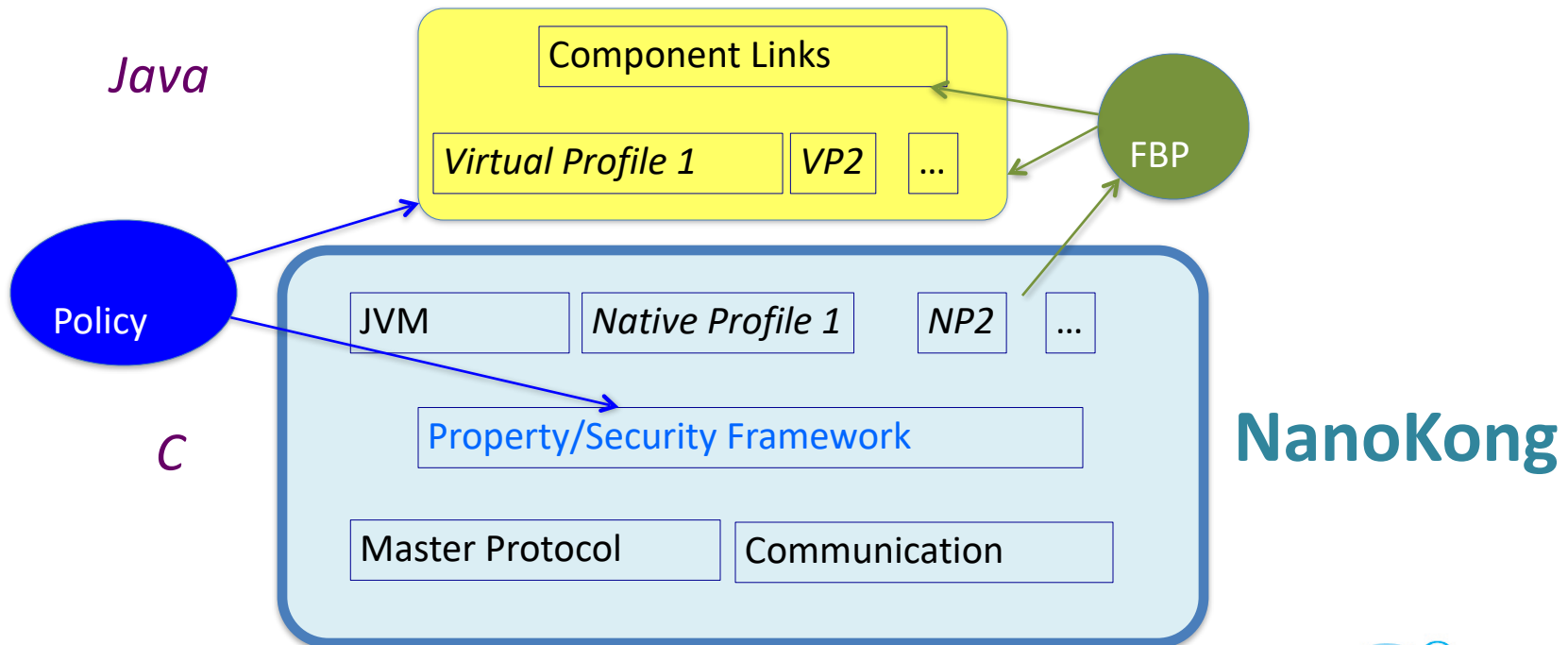
WuKONG WORKFLOW 5.0

NanoKong



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

NANOKONG VM CODE SIZE (DARJEELING)

Memory requirement: 75K Flash, 2K RAM

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

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

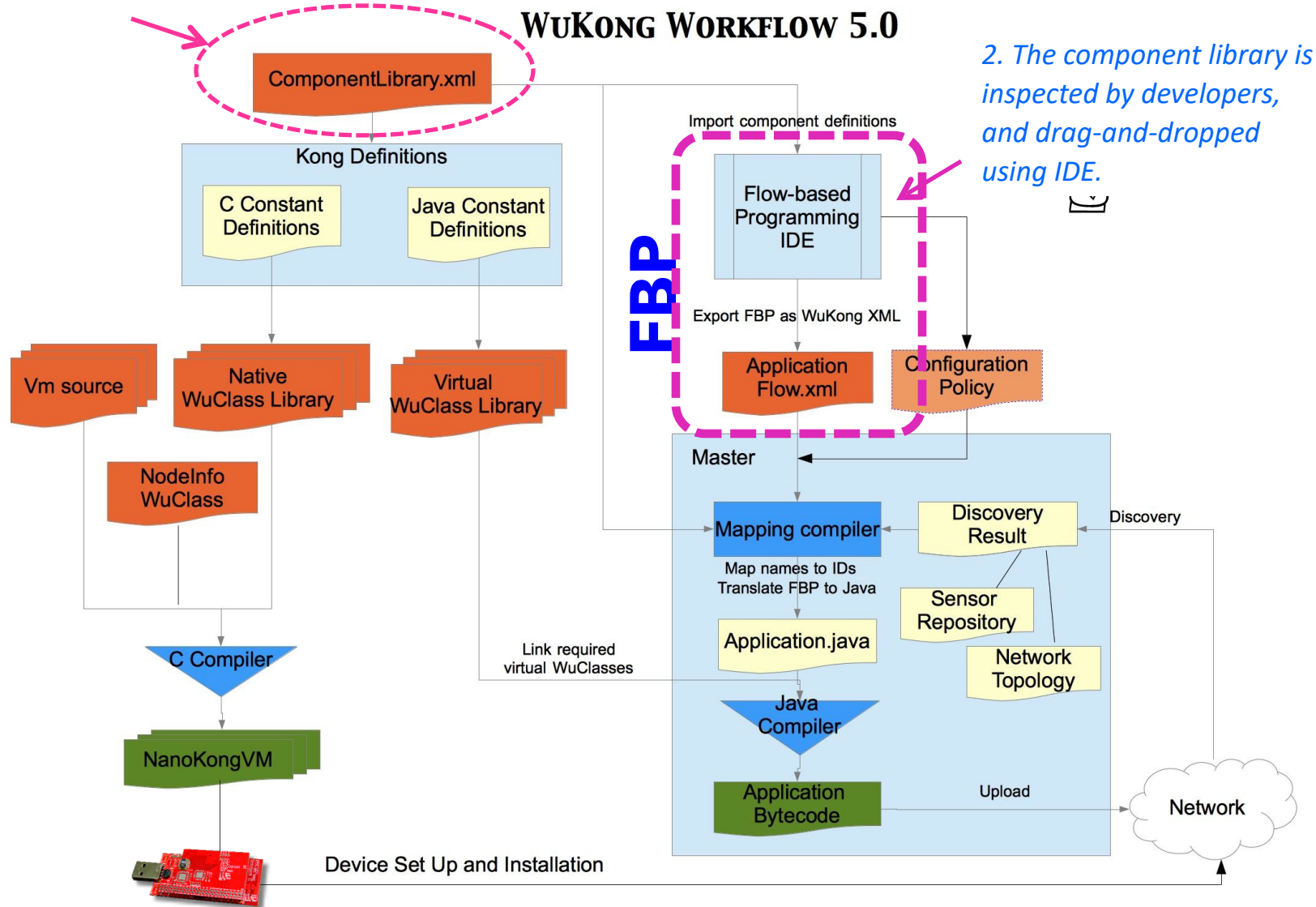
2. Build flow-based programs using IDE

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

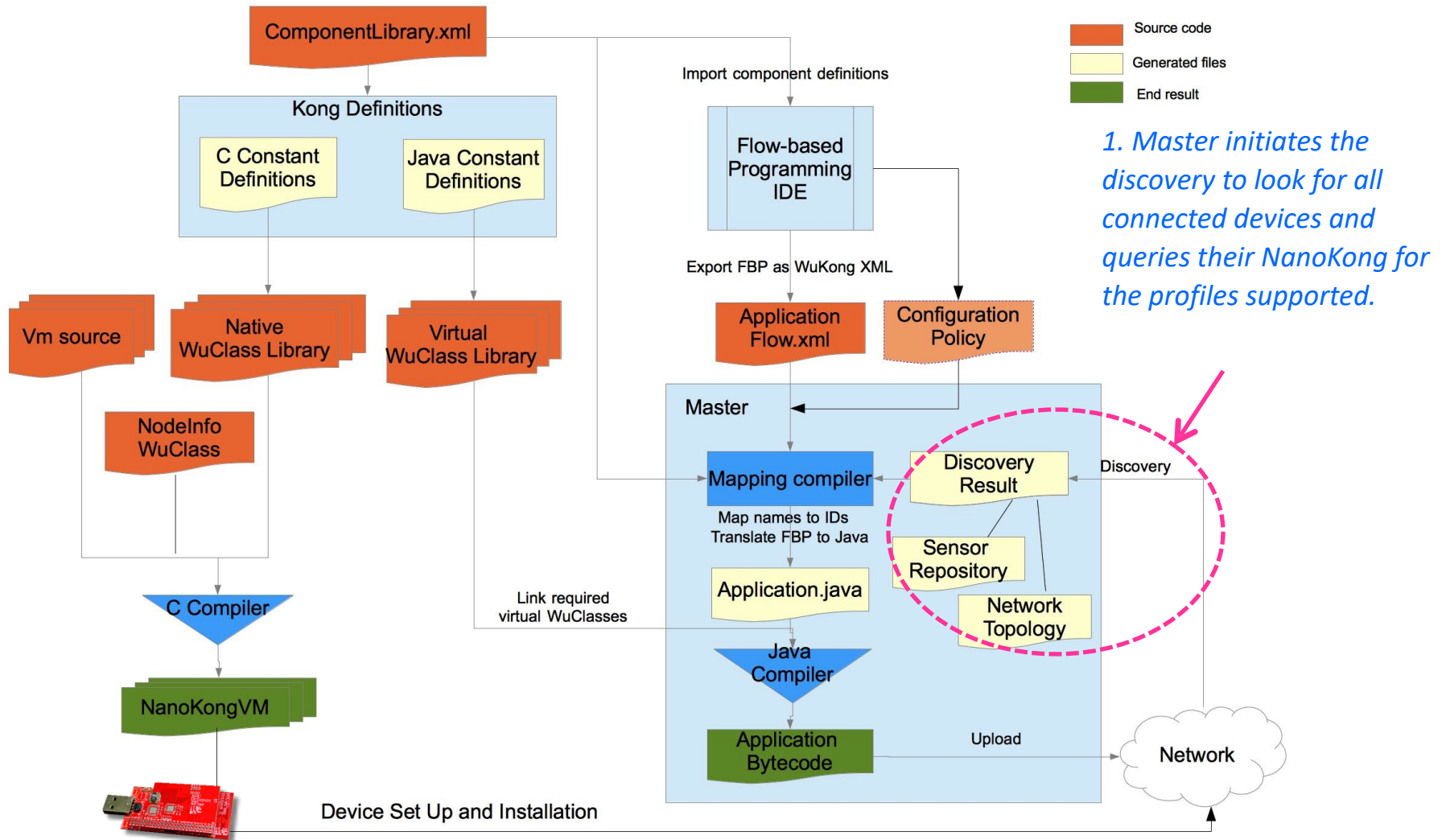
2. Build flow-based programs using IDE

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

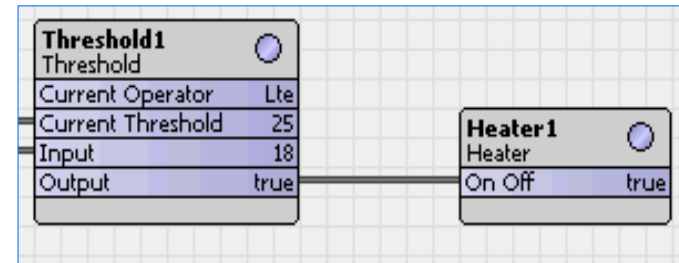
1. User sends the flow XML to Master to define the application flow
2. Master discovers sensors remotely, maps them to flow components, and deploys the codes to all sensor nodes

WUKONG WORKFLOW 5.0



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>

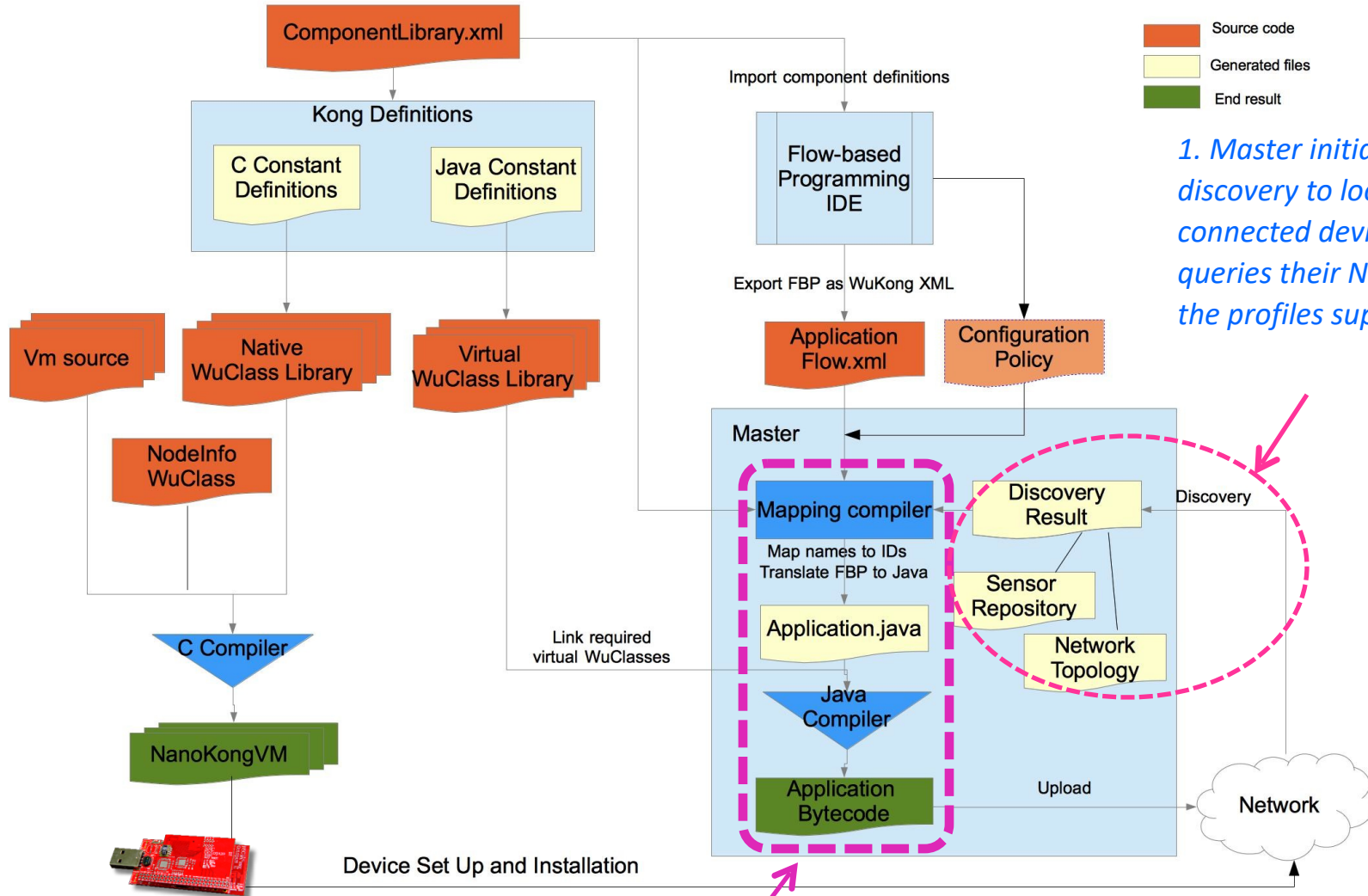
<WuClass name="Threshold" id="1" type="soft">
  <property name="operator" datatype="ThresholdOperator" access="readwrite" default="lt" />
  <property name="threshold" datatype="short" access="readwrite" default="30" />
  <property name="value" datatype="short" access="readwrite" default="20" />
  <property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

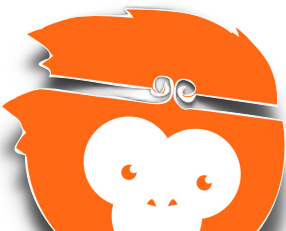
Native vs. Virtual Components

- 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

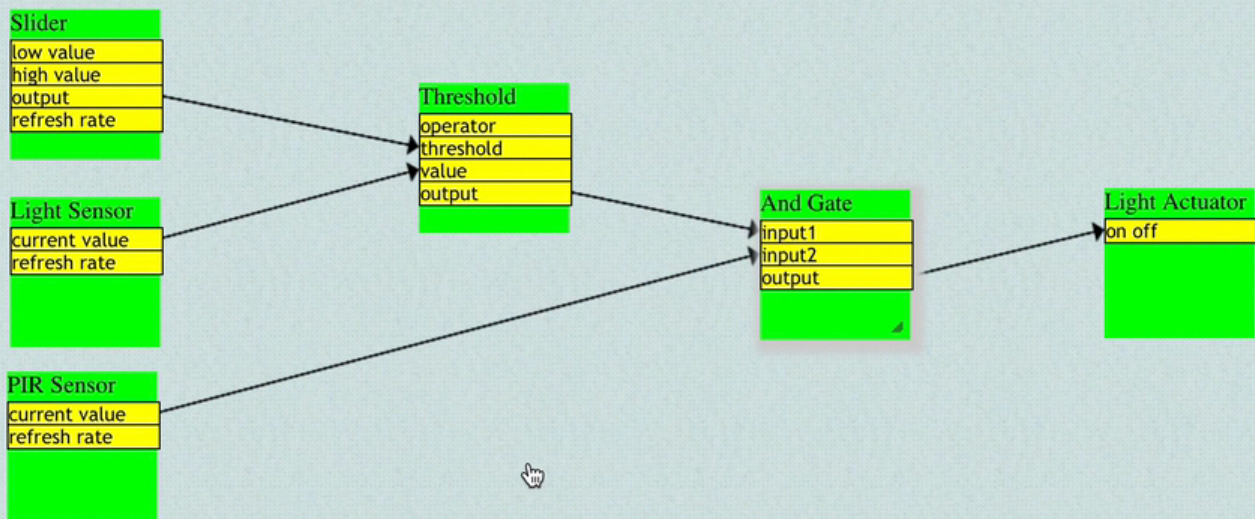


2. Master uses mapping algorithms to select different devices for a FBP, and upload the codes

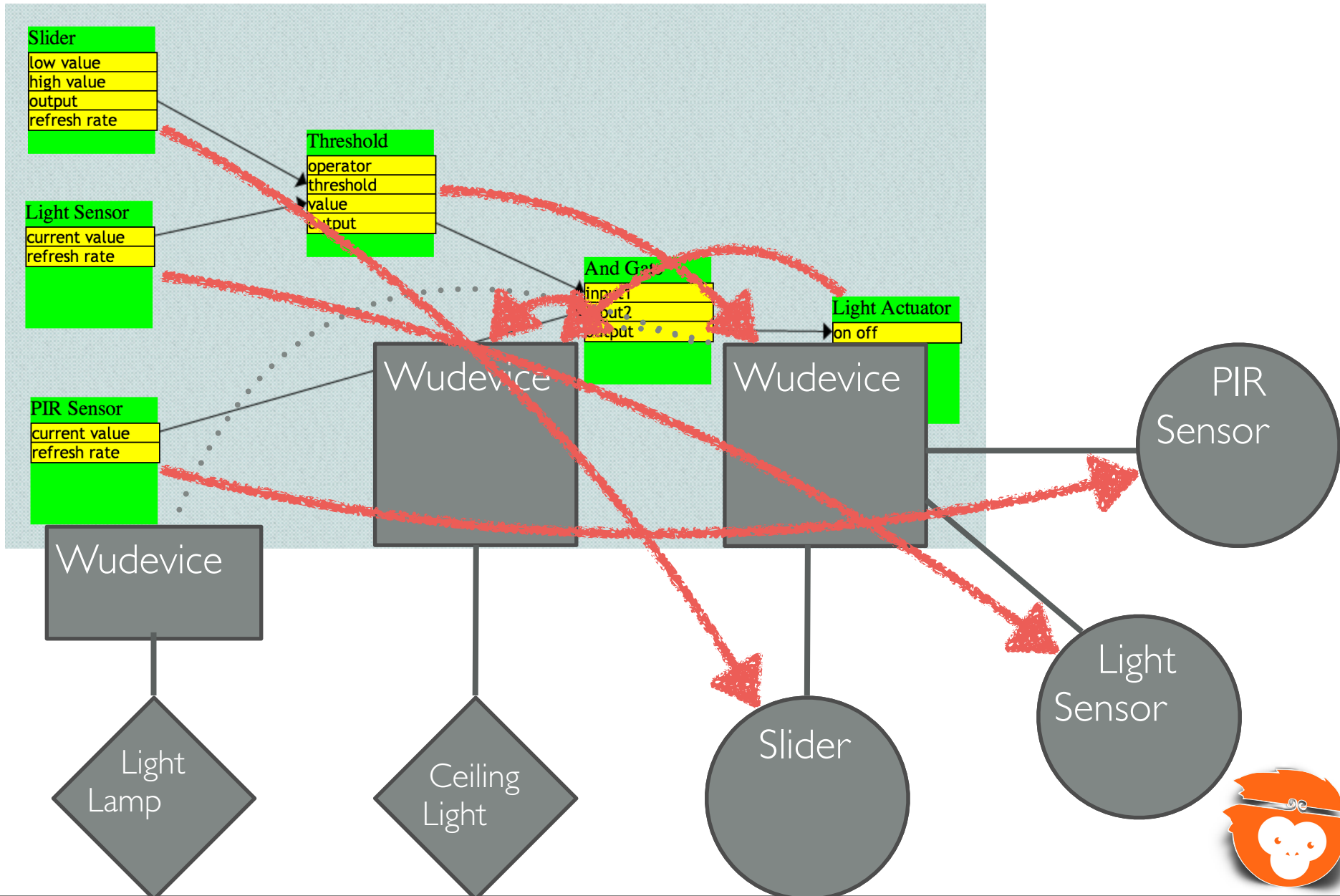


- LED
- Light_Actuator
- Light_Sensor
- Logical
- Loop_delay_boolean
- Loop_delay_short
- Magnetic_Sensor
- MathOp
- New_Class
- New_Class_2

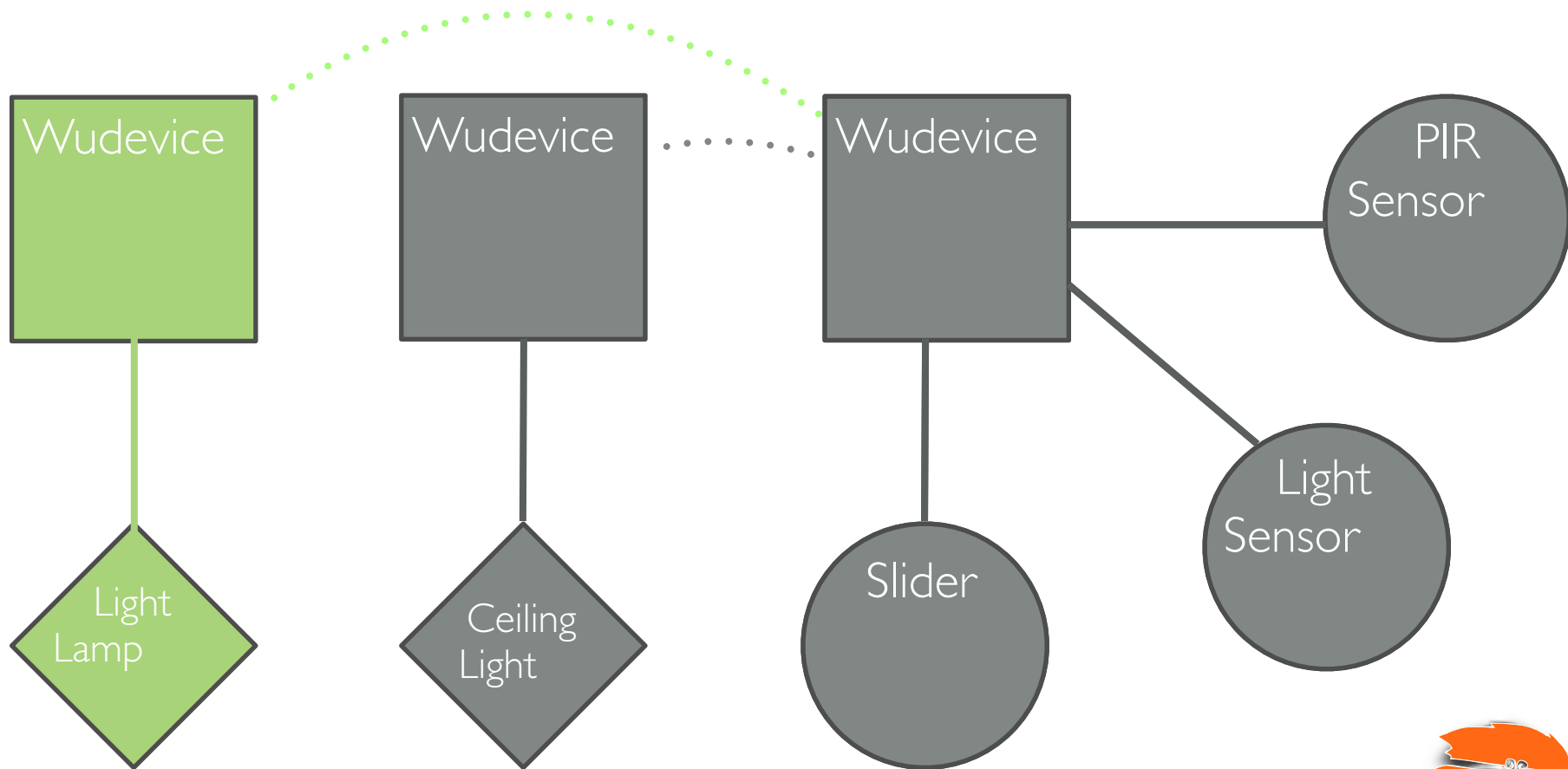
Edit Component Import Del Link Save Delete Page _first_ ⌵



Sensors, Actuators and WuDevices



Sensors, Actuators and WuDevices



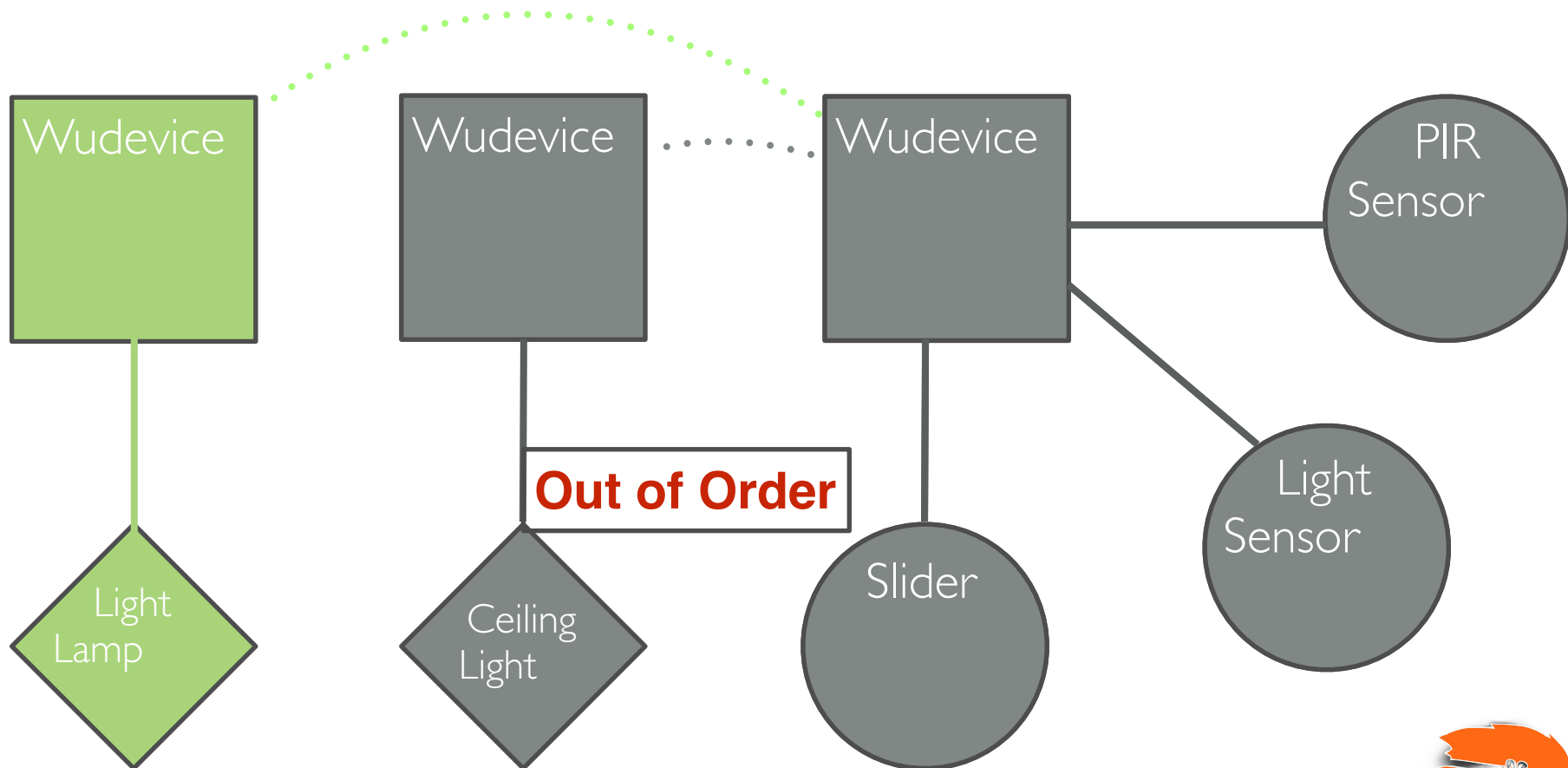
Health and idle



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



[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

Back

Application: Bedroom

Application Editor

Deployment

Current Nodes

Map

Processing

Back

Application: Bedroom

Application Editor

Deployment

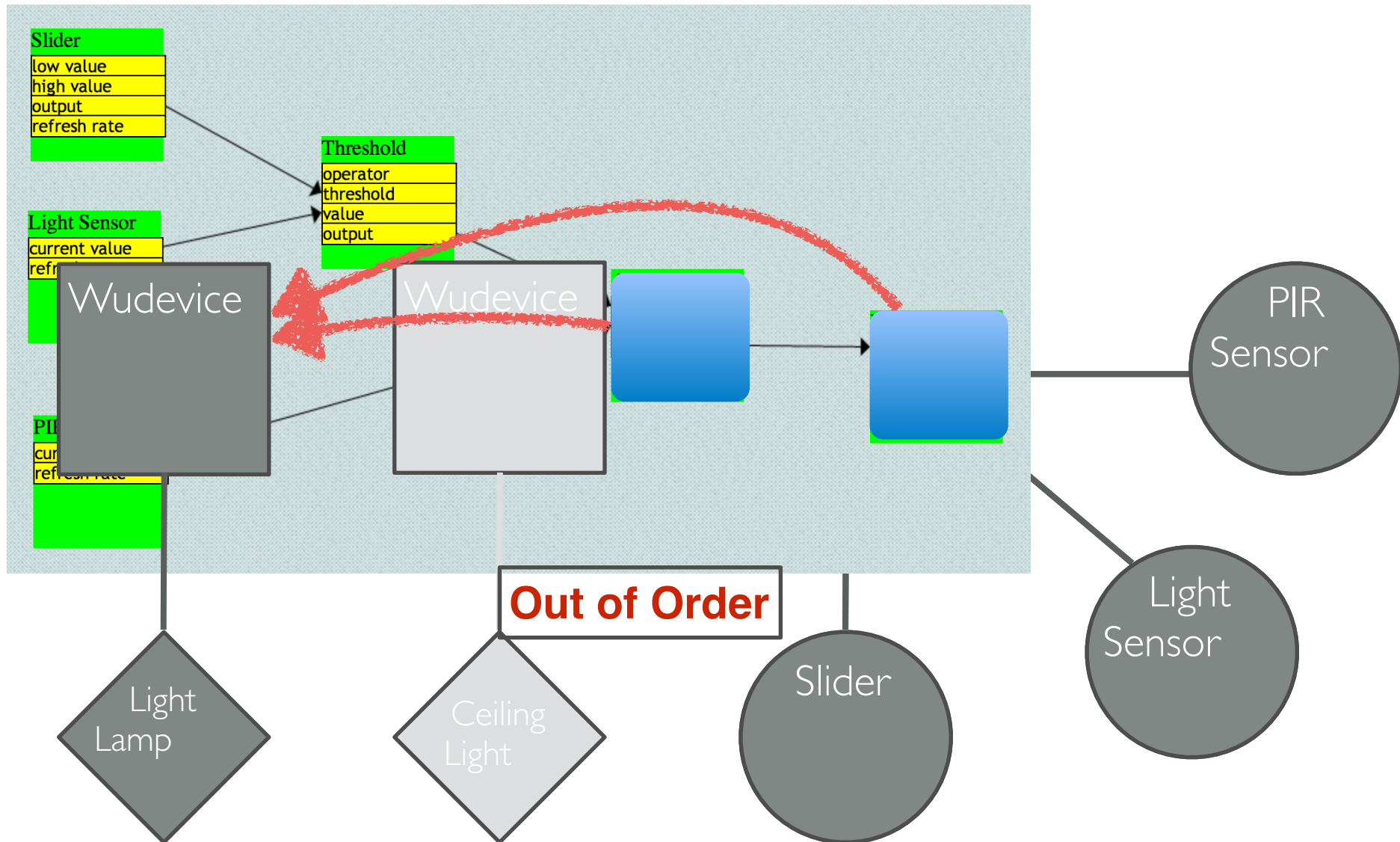
Current Nodes

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