Node-RED

A Visual Tool for Building the Internet of Things

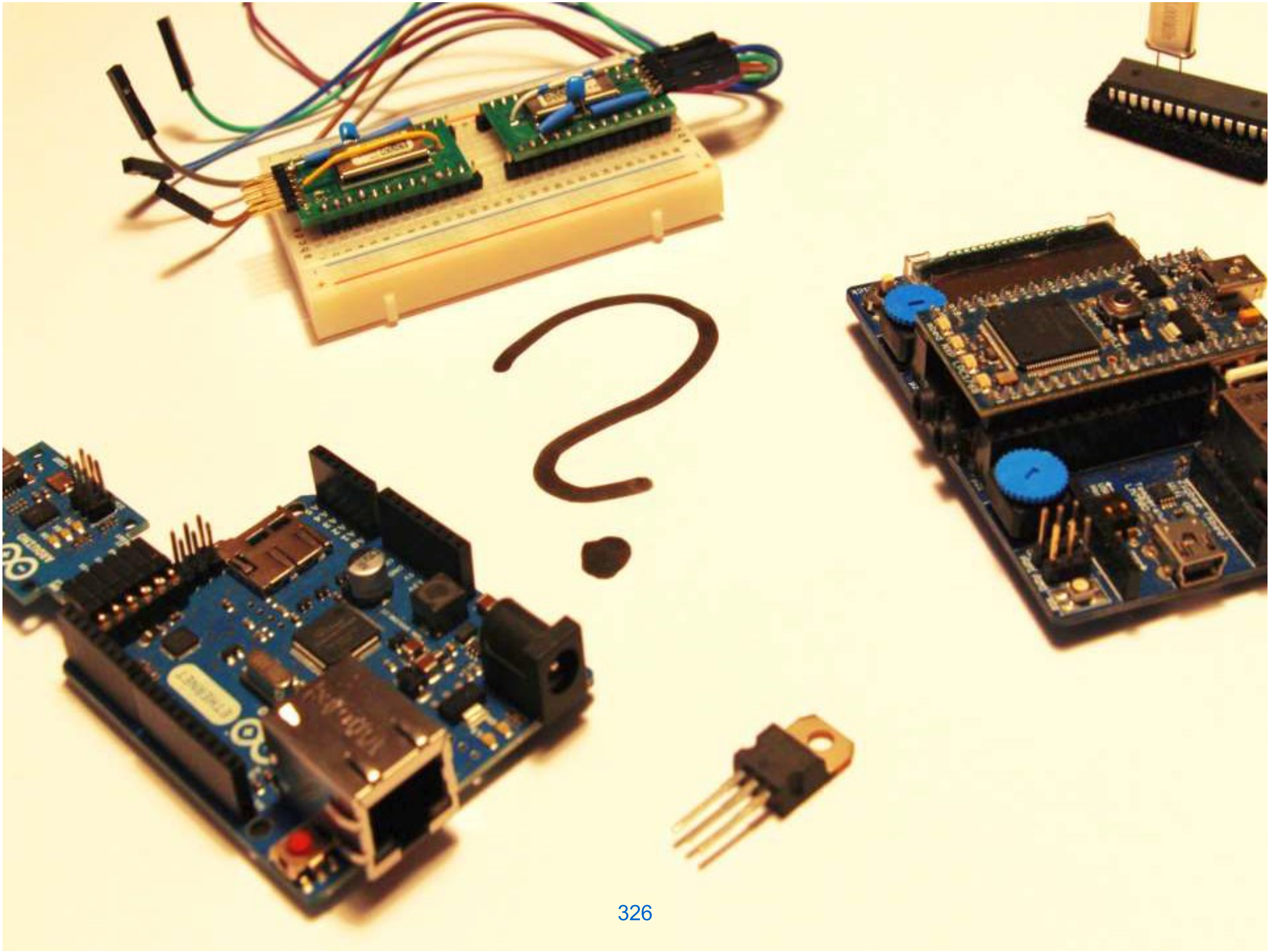
Live Self-Learning

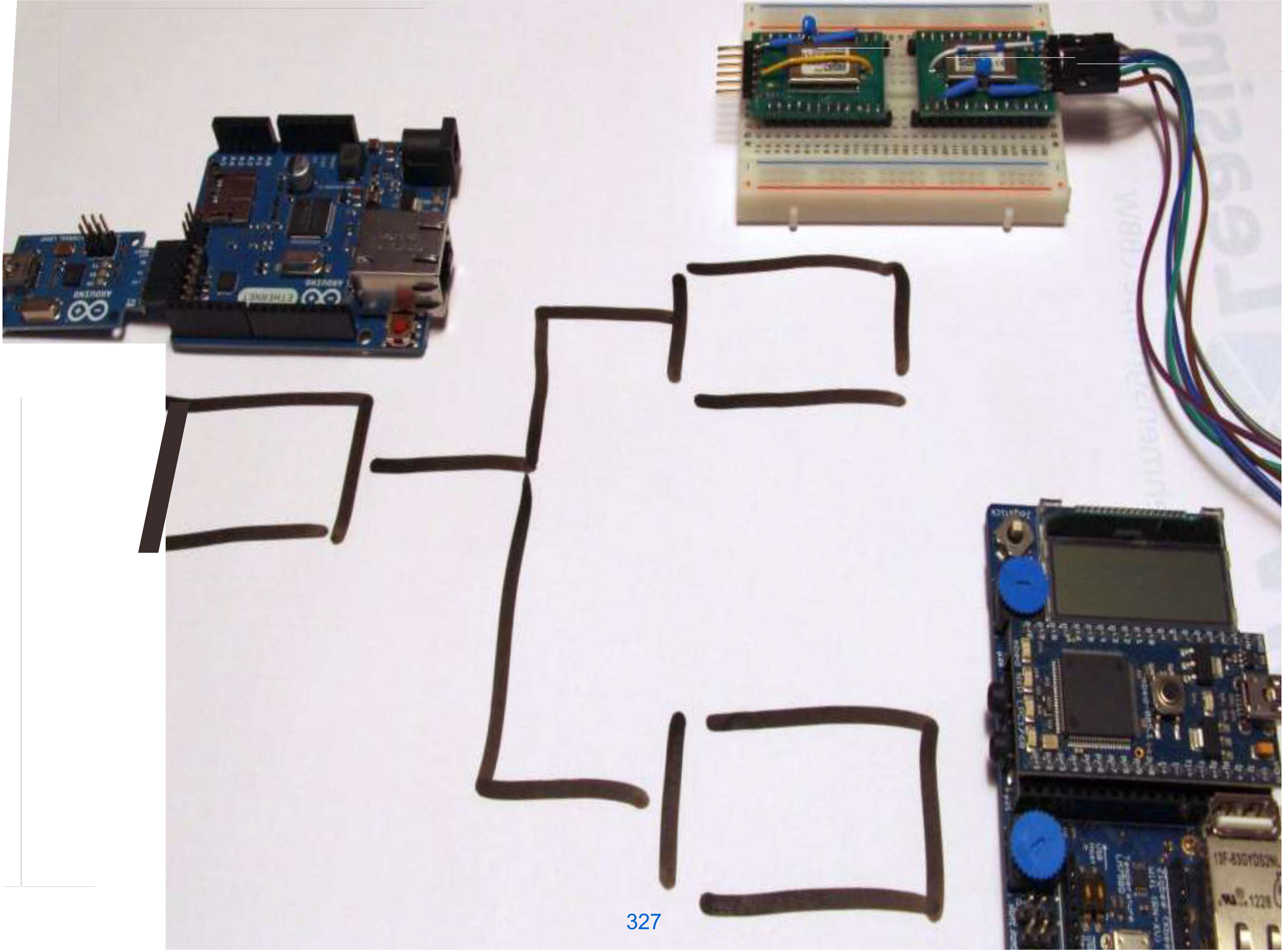
- NodeRED editor
    - https://users.sensetecnic.com/login?
    - User name: iot017
    - Pwd: iot2017
- UI: https://iot017.fred.sensetecnic.com/api/ui/#/0

Chapter Outline

- Motivation
- What is Node-RED?
- Architecture
- Basic Nodes
    - Input Nodes
    - Processing Nodes
    - Output Nodes
- Limitation of Node-RED
- Conclusions

beaglebone
Thank you for purchasing
PARKSIDE
RFduino





Why Node-RED

- We have
  - Processors for editing Words
  - Spreadsheets for working with Numbers
  - Powerpoint for arranging Pictures and Ideas
- But we don't have a simple tool for coordinating Events
  - Business events – status of processes, alerts from machines
  - Social events – tweets, alerts
  - IoT events – temperatures, weather, lights, doors
- Something that anyone can use to build situational applications
“Wouldn't it be neat if, when x happens it can tell me...
  - … and alert somebody...
  - … and kick off the xyz process...
  - … or just go ping !”

What is NodeRED

- NodeRED
  - Based on Node.js
  - Taking full advantage of its *event-driven*, *non-blocking* IO model
- Originally developed as an open source project at IBM in 2013
- A toolkit for „piecing“ together IoT applications by wiring together hardware devices, APIs and online services
  - An application composition tool
  - LEGO-lize application building
- It uses a visual programming approach
  - Easy to use
- Free logic engine
- http://nodered.org

What is NodeRED

- Nodes: predefined code blocks
  - Input nodes
  - Processing nodes
  - Output nodes
- Flows: A set of connected nodes to perform a task
- A rich library of nodes and flows
- Simple to extend to add new capabilities
- Web-based programming environment
  - Javascript

Node-RED

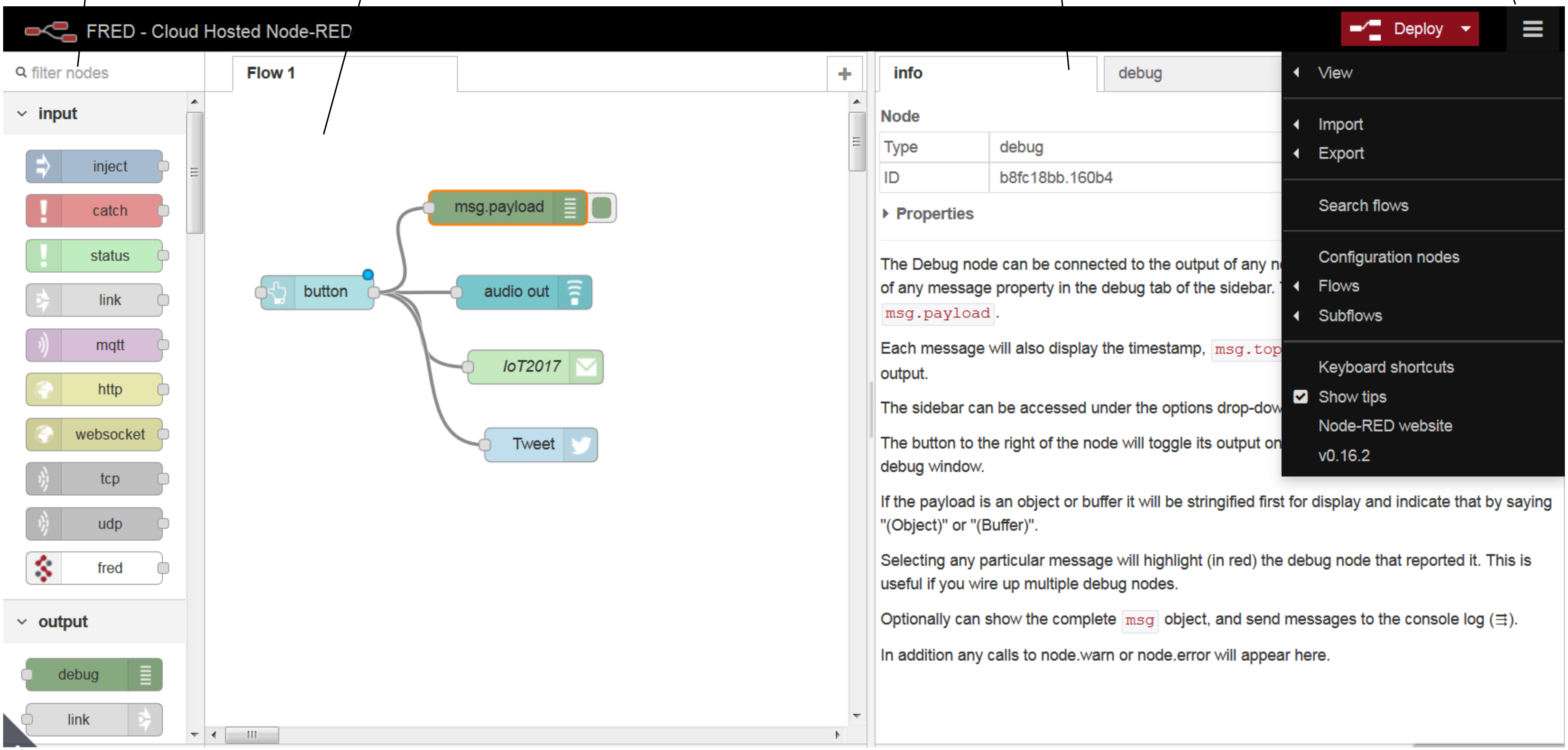
Info / Debug

Menu

Nodes

Flow definition

Deploy



Node

Type	debug
ID	b8fc18bb.160b4

Properties

The Debug node can be connected to the output of any n
of any message property in the debug tab of the sidebar. `
`msg.payload`.

Each message will also display the timestamp, `msg.top`
output.

The sidebar can be accessed under the options drop-dow

The button to the right of the node will toggle its output on
debug window.

If the payload is an object or buffer it will be stringified first for display and indicate that by saying "(Object)" or "(Buffer)".

Selecting any particular message will highlight (in red) the debug node that reported it. This is useful if you wire up multiple debug nodes.

Optionally can show the complete `msg` object, and send messages to the console log (☰).

In addition any calls to node.warn or node.error will appear here.

Node-RED Strengths

Node-RED's power comes from a combination of two factors:

- Node-RED has a flow-based programming model
  - Messages representing events flow between nodes, triggering processing that results in output.
  - The flow-based programming model maps well to typical IoT applications which are characterized by real-world events that trigger some sort of processing which in turn results in real-world actions.
- The set of built-in nodes
  - Node-RED offers developers *powerful building blocks* to allow them to quickly put together flows that accomplish a lot, without having to worry about the programming details.

Wiring the Internet of Things: Stakeholders and Requirements

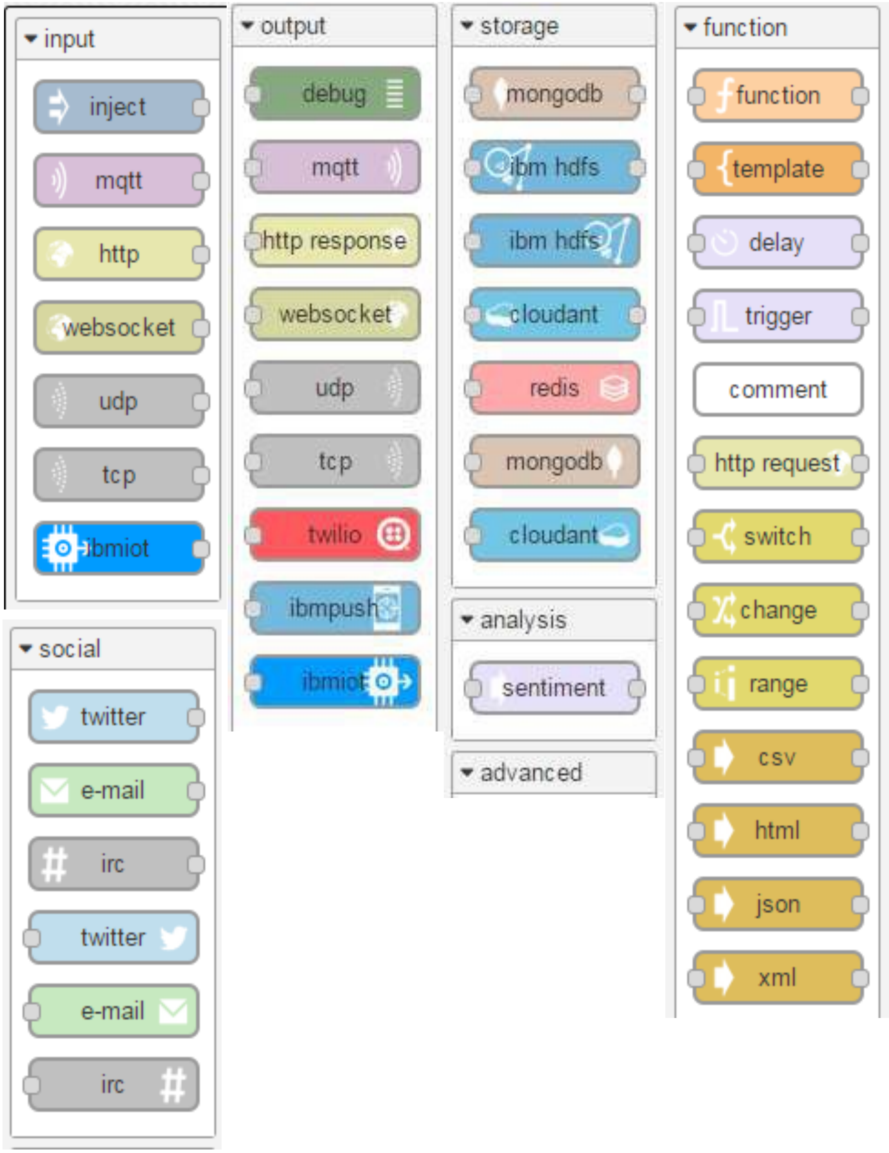
- New developers & education
  - Short learning curve
  - Easy to use
  - Low barrier to entry
- Community Developers
  - Open standards
  - Flexibility
  - Ability to share
- App Developers
  - Rapid prototyping
  - Easy to integrate with existing tools and applications
  - Easy to extend with richer/bespoke functionality
- Hackers
  - Runs on Raspberry Pi, Beaglebone,

  other low power devices.
  - Works with Arduino, etc

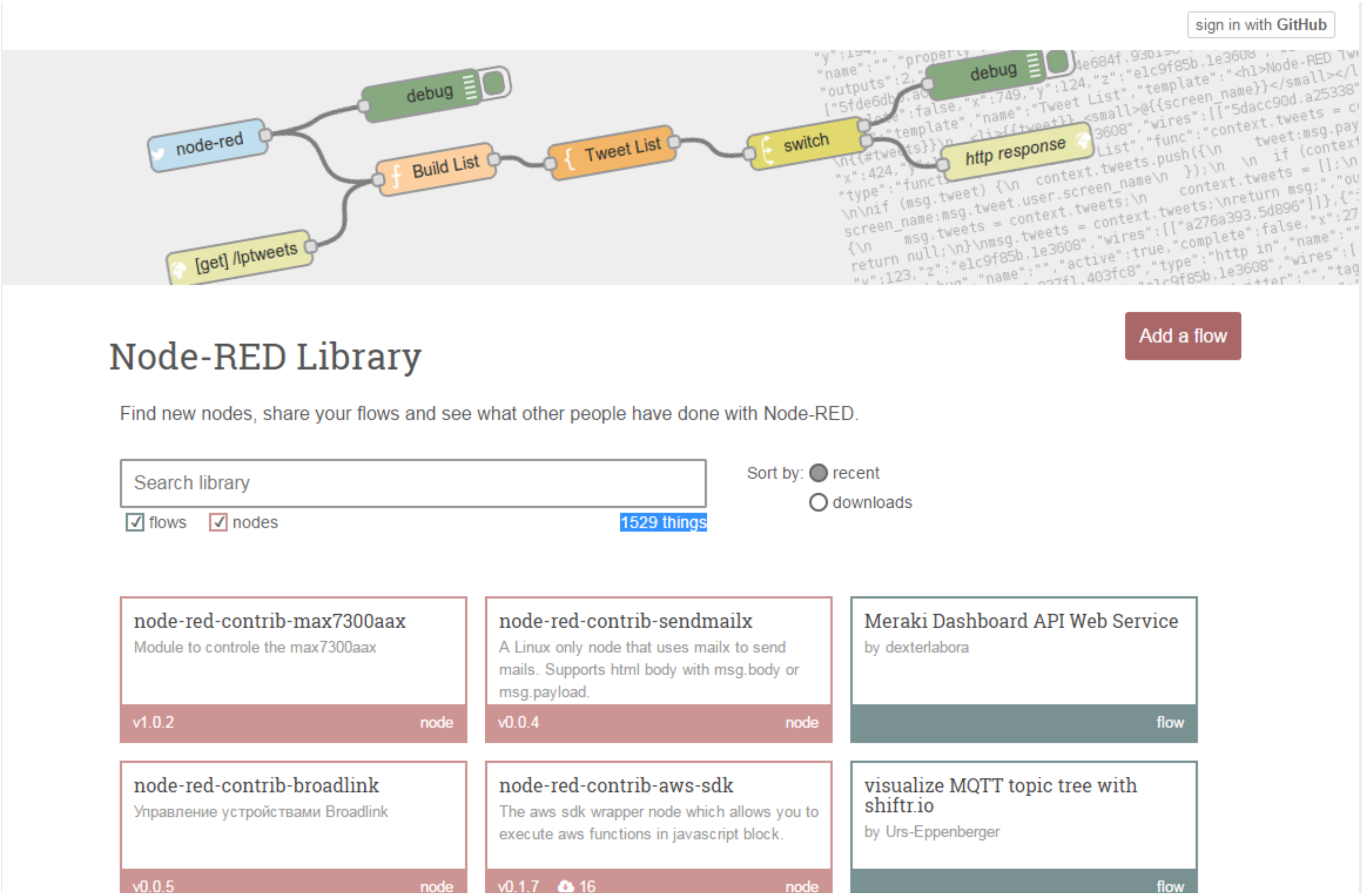
Nodes

Node-RED is already capable of connecting to many things, including:

- Local services:
  - Network sockets
  - Files
  - Serial ports
  - Execute local commands
  - Raspbery Pi / BeagleboneBlack GPIO pins
  - MongoDB
  - Redis
- Online services:
  - Twitter
  - IRC
  - XMPP Chat
  - RSS/ATOM
  - Email
- Processing functions:
  - User-defined functions, written in JavaScript
  - Sentiment analysis
  - XML to JavaScript handling



More Nodes



Node-RED Library

Add a flow

Find new nodes, share your flows and see what other people have done with Node-RED.

Search library

☑ flows ☑ nodes 1529 things

Sort by: ● recent
○ downloads

node-red-contrib-max7300aax
Module to controle the max7300aax
v1.0.2 node

node-red-contrib-sendmailx
A Linux only node that uses mailx to send mails. Supports html body with msg.body or msg.payload.
v0.0.4 node

Meraki Dashboard API Web Service
by dexterlabora
flow

node-red-contrib-broadlink
Управление устройствами Broadlink
v0.0.5 node

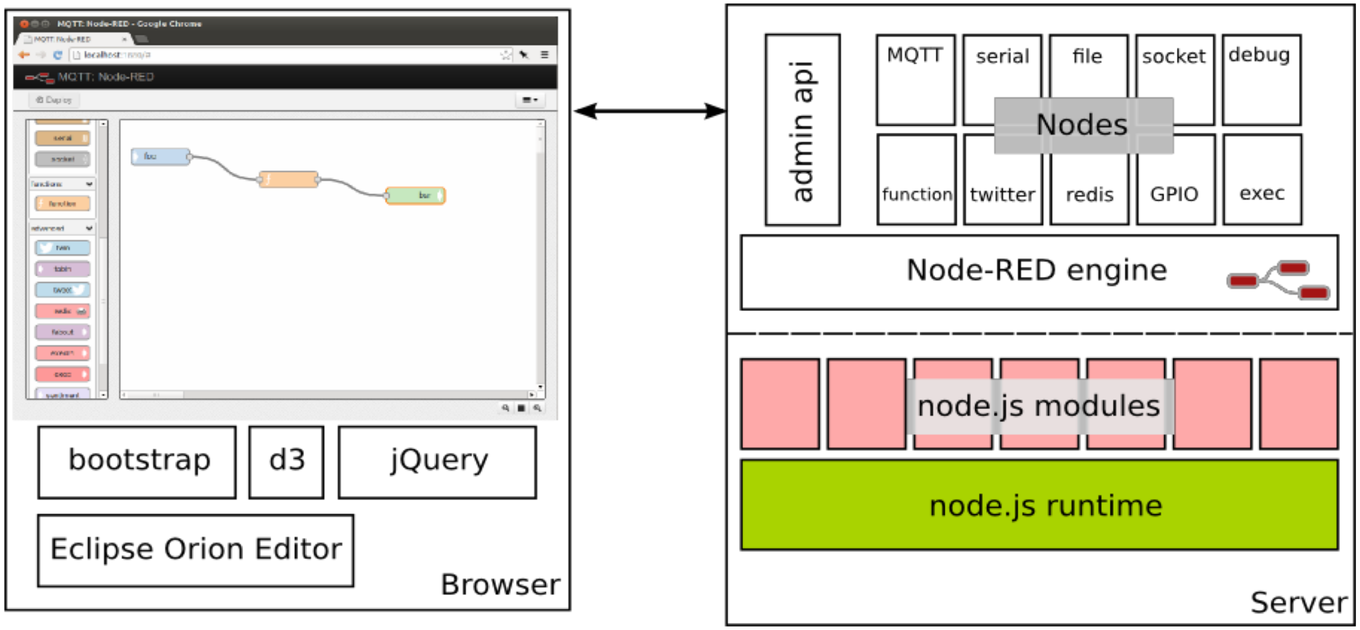
node-red-contrib-aws-sdk
The aws sdk wrapper node which allows you to execute aws functions in javascript block.
v0.1.7 16 node

visualize MQTT topic tree with shiftr.io
by Urs-Eppenberger
flow

Adding Palette Nodes

- http://flows.nodered.org
- sudo apt-get install npm
- cd ~/.node-red
- npm install node-red-{example node name}

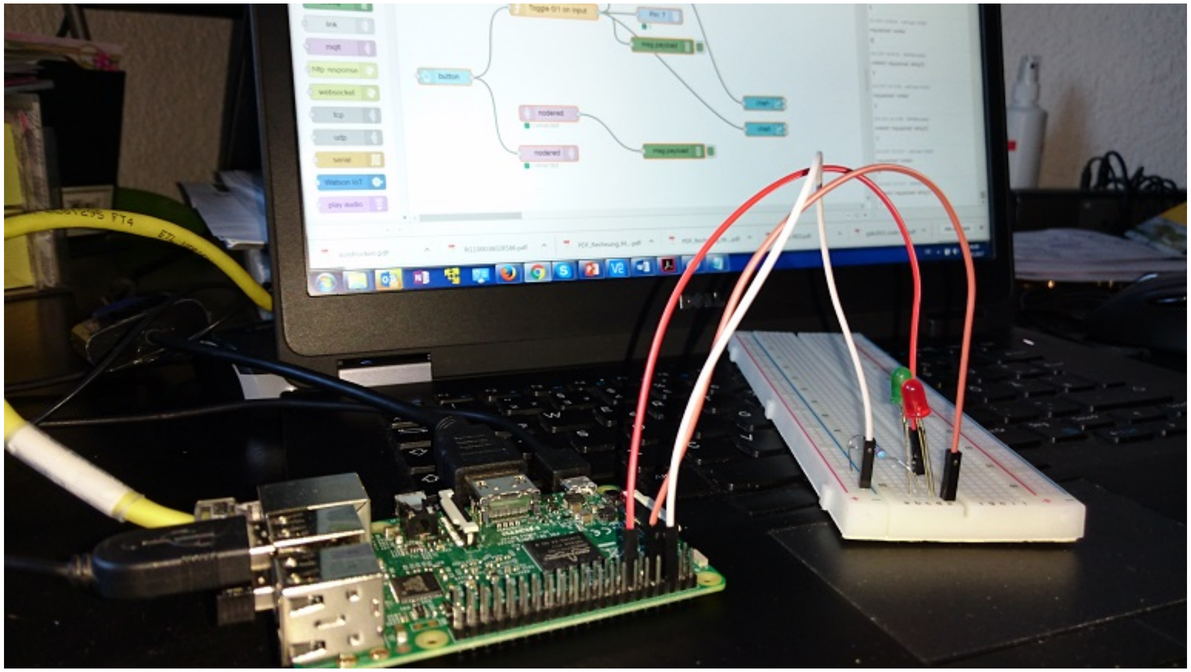
Architecture



- Runs on node.js
- Can exploit the 29,000+ open-source modules available via npm to add new functionality...
- Node-RED nodes provide integration with other systems. Each node is defined in their own pair of JavaScript and html files using a simple API and are dynamically loaded by the engine.
- Web interface can be secured or run headless.

Run Local, Even on Constrained Devices …

The lightweight runtime is ideal to run on Edge-of-Network devices, such as the Raspberry Pi.



The node library makes it easy to create simply, effective applications.

Here, the board lights LEDs.

Raspbian Jessie

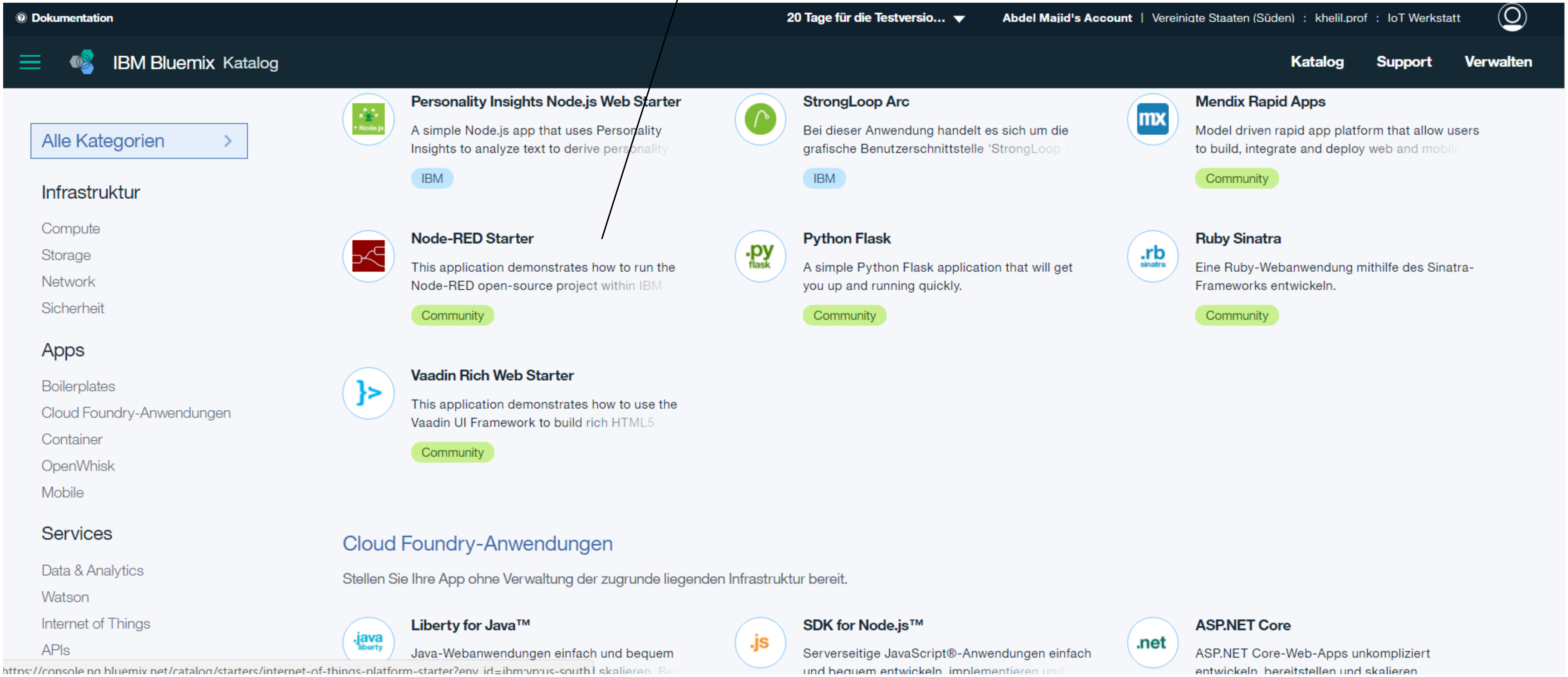
http://nodered.org/docs/hardware/raspberrypi.html

- Start: Desktop: Menu->Programming->Node-RED
    - Or: node-red-start
- Stop: node-red-stop

- Editor: http://ipaddress:1880
- GUI: http://ipaddress:1880/ui

… or in the Cloud

Node-RED editor



Dokumentation

20 Tage für die Testversio…

Abdel Majid's Account | Vereinigte Staaten (Süden) : khelil.prof : IoT Werkstatt

IBM Bluemix Katalog

Katalog Support Verwalten

Alle Kategorien

Infrastruktur

Compute
Storage
Network
Sicherheit

Apps

Boilerplates
Cloud Foundry-Anwendungen
Container
OpenWhisk
Mobile

Services

Data & Analytics
Watson
Internet of Things
APIs

Personality Insights Node.js Web Starter
A simple Node.js app that uses Personality Insights to analyze text to derive personality
IBM

StrongLoop Arc
Bei dieser Anwendung handelt es sich um die grafische Benutzerschnittstelle 'StrongLoop A
IBM

Mendix Rapid Apps
Model driven rapid app platform that allow users to build, integrate and deploy web and mobile
Community

Node-RED Starter
This application demonstrates how to run the Node-RED open-source project within IBM
Community

Python Flask
A simple Python Flask application that will get you up and running quickly.
Community

Ruby Sinatra
Eine Ruby-Webanwendung mithilfe des Sinatra-Frameworks entwickeln.
Community

Vaadin Rich Web Starter
This application demonstrates how to use the Vaadin UI Framework to build rich HTML5
Community

Cloud Foundry-Anwendungen

Stellen Sie Ihre App ohne Verwaltung der zugrunde liegenden Infrastruktur bereit.

Liberty for Java™
Java-Webanwendungen einfach und bequem

SDK for Node.js™
Serverseitige JavaScript®-Anwendungen einfach und bequem entwickeln, implementieren und

ASP.NET Core
ASP.NET Core-Web-Apps unkompliziert entwickeln, bereitstellen und skalieren

https://console.ng.bluemix.net/catalog/starters/internet-of-things-platform-starter?env_id=ibm:yp:us-south

Example: http://www.bluemix.net/

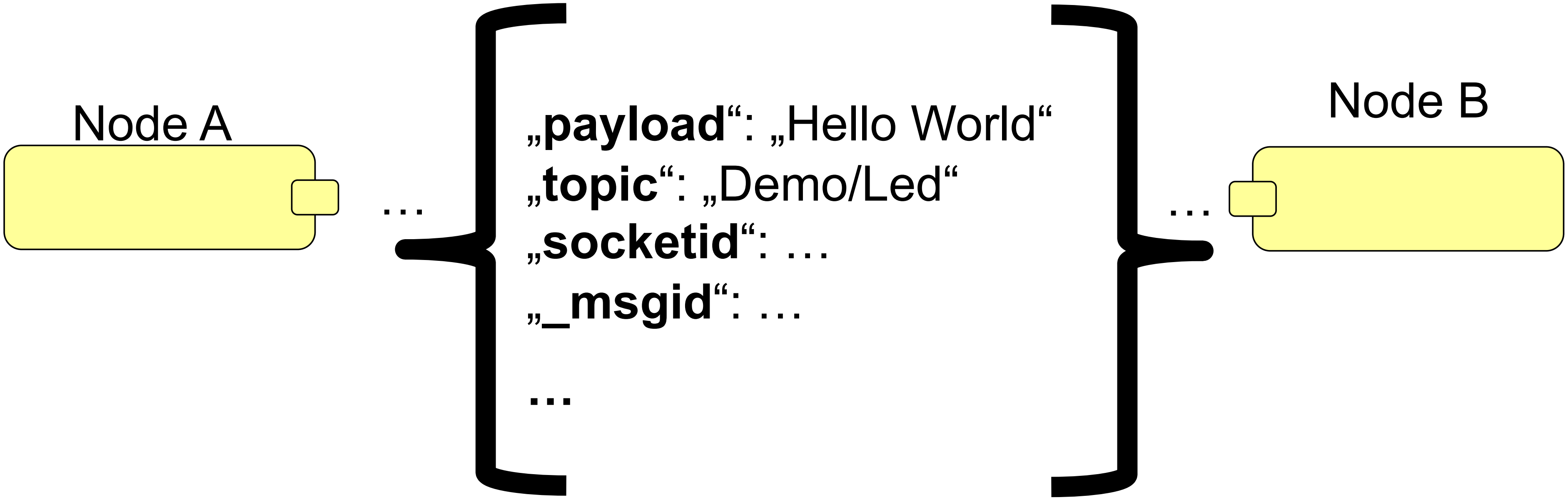
Popular Nodes (1)

- Inject Node
  - Allows manual triggering of flows
  - Can be scheduled to automatically inject at fixed intervals
- Debug Node
  - Shows message content, either just payload or entire object in the debug sidebar
- Function Node
  - Runs user-defined js against the message flowing past
- Logic Nodes
  - Comparisons, re-scaling, re-mapping

Popular Nodes (2)

- TCP/UDP Nodes
  - Connects out, or listens for incoming connections
- HTTP Nodes
  - Define http endpoints for incoming REQUESTs, or trigger GETs of urls in the middle of a flow
- MQTT Nodes
  - Define publishers or subscribers to a certain topic on a certain MQTT broker
- GPIO Nodes
  - Read and write from Raspberry Pi GPIO

Message Object



Node A

…

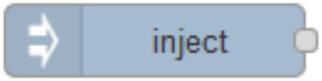
„**payload**“: „Hello World“
„**topic**“: „Demo/Led“
„**socketid**“: …
„**_msgid**“: …

…

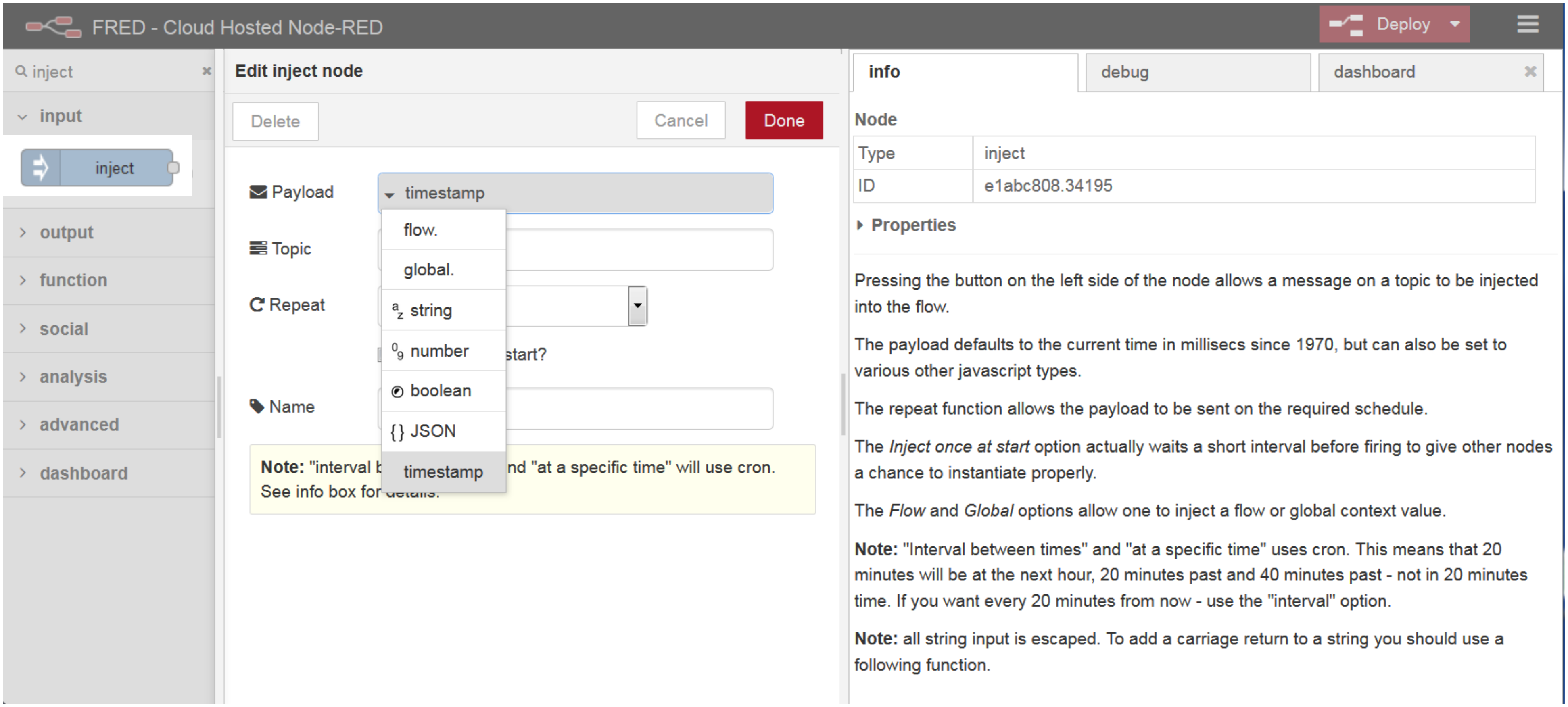
…

Node B

INJECT Node

- Input node 

- Allows you to inject messages into a flow, either by *clicking the button* on the node, or *setting a time interval* between injects.



Payload: timestamp

Topic

Repeat

start?

Name

Note: "interval b… …nd "at a specific time" will use cron. See info box for details.

Node

Type	inject
ID	e1abc808.34195

▸ Properties

Pressing the button on the left side of the node allows a message on a topic to be injected into the flow.

The payload defaults to the current time in millisecs since 1970, but can also be set to various other javascript types.

The repeat function allows the payload to be sent on the required schedule.

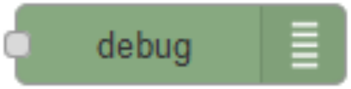
The *Inject once at start* option actually waits a short interval before firing to give other nodes a chance to instantiate properly.

The *Flow* and *Global* options allow one to inject a flow or global context value.

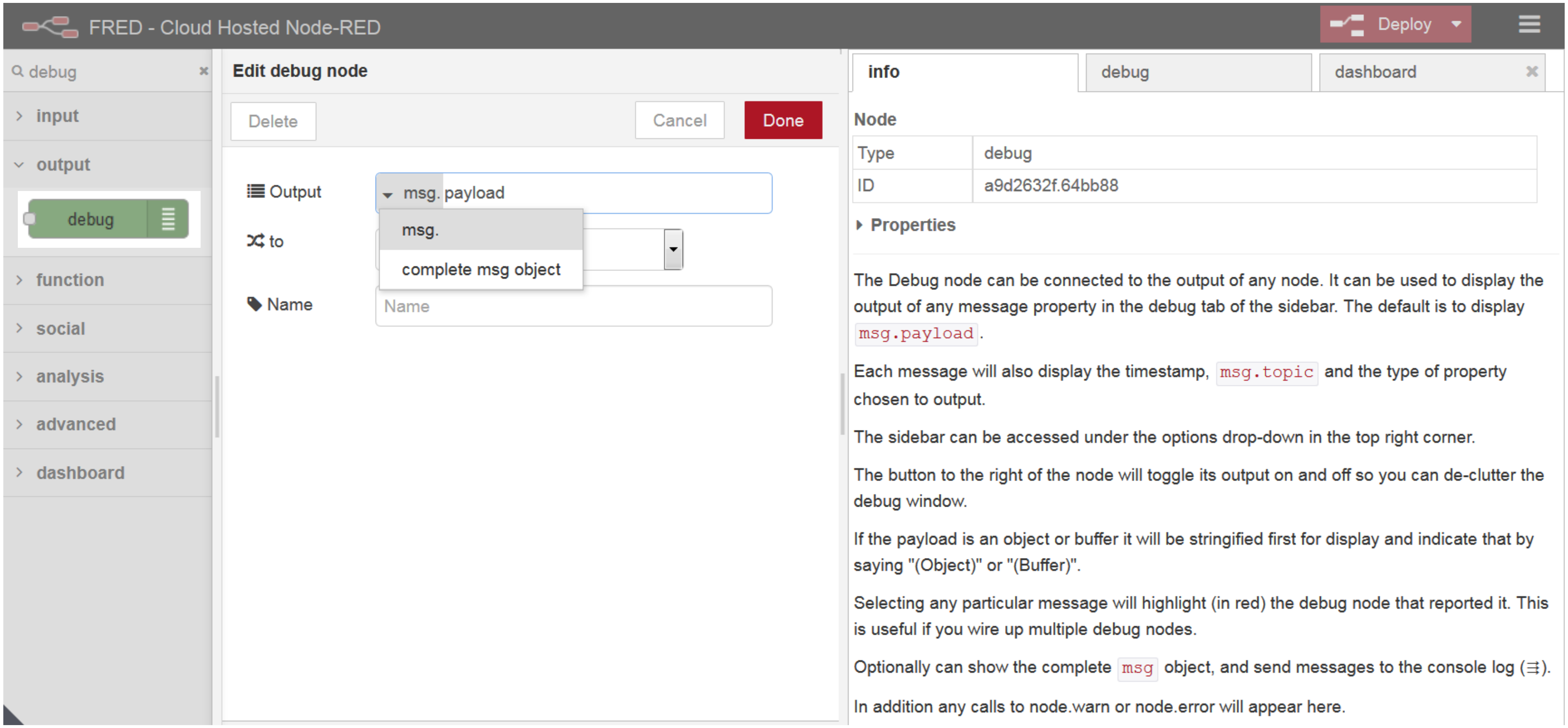
Note: "Interval between times" and "at a specific time" uses cron. This means that 20 minutes will be at the next hour, 20 minutes past and 40 minutes past - not in 20 minutes time. If you want every 20 minutes from now - use the "interval" option.

Note: all string input is escaped. To add a carriage return to a string you should use a following function.

DEBUG Node

- Output node 

- Causes any message to be displayed in the Debug sidebar. By default, it just displays the payload of the message, but it is possible to display the entire message object.



Edit debug node

Output: msg. payload

to

Name: Name

Node

Type	debug
ID	a9d2632f.64bb88

Properties

The Debug node can be connected to the output of any node. It can be used to display the output of any message property in the debug tab of the sidebar. The default is to display `msg.payload`.

Each message will also display the timestamp, `msg.topic` and the type of property chosen to output.

The sidebar can be accessed under the options drop-down in the top right corner.

The button to the right of the node will toggle its output on and off so you can de-clutter the debug window.

If the payload is an object or buffer it will be stringified first for display and indicate that by saying "(Object)" or "(Buffer)".

Selecting any particular message will highlight (in red) the debug node that reported it. This is useful if you wire up multiple debug nodes.

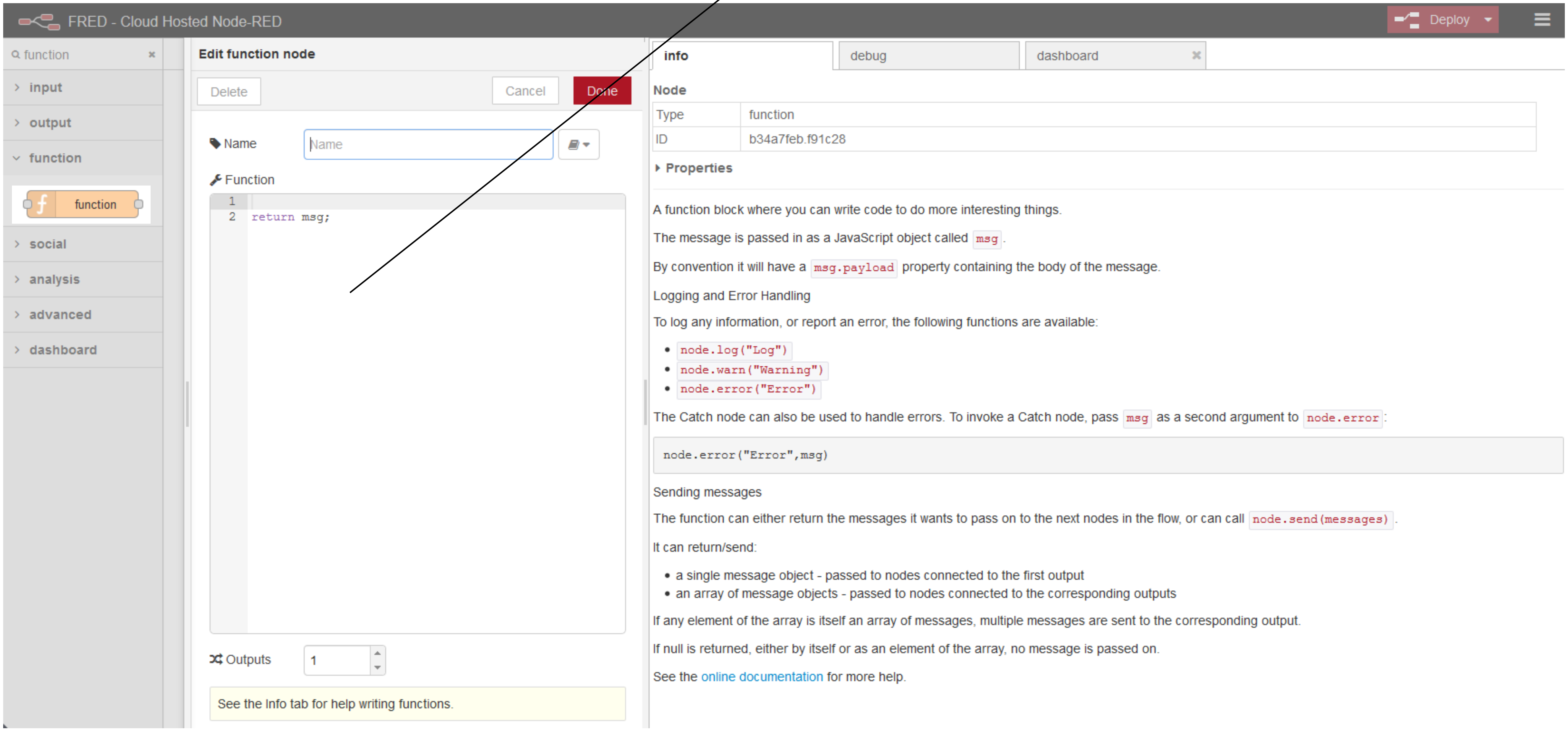
Optionally can show the complete `msg` object, and send messages to the console log (⇶).

In addition any calls to node.warn or node.error will appear here.

Function Node (1)

- Is a processing node

Write here your javaScript code



Name: Name

Function

```
return msg;
```

Outputs: 1

See the Info tab for help writing functions.

Node

Type	function
ID	b34a7feb.f91c28

Properties

A function block where you can write code to do more interesting things.

The message is passed in as a JavaScript object called `msg`.

By convention it will have a `msg.payload` property containing the body of the message.

Logging and Error Handling

To log any information, or report an error, the following functions are available:

- `node.log("Log")`
- `node.warn("Warning")`
- `node.error("Error")`

The Catch node can also be used to handle errors. To invoke a Catch node, pass `msg` as a second argument to `node.error`:

```
node.error("Error",msg)
```

Sending messages

The function can either return the messages it wants to pass on to the next nodes in the flow, or can call `node.send(messages)`.

It can return/send:

- a single message object - passed to nodes connected to the first output
- an array of message objects - passed to nodes connected to the corresponding outputs

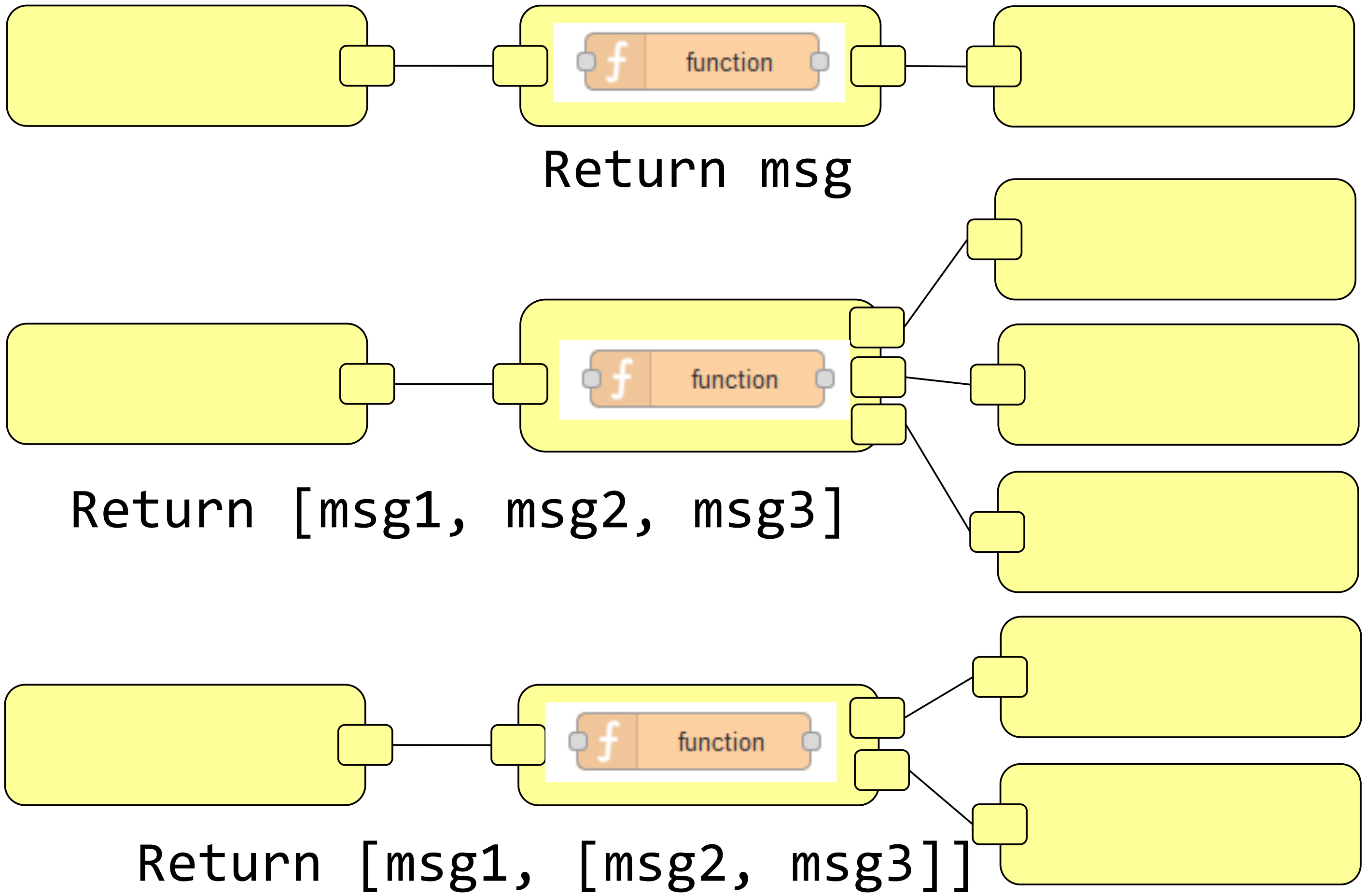
If any element of the array is itself an array of messages, multiple messages are sent to the corresponding output.

If null is returned, either by itself or as an element of the array, no message is passed on.

See the online documentation for more help.

Function Node (2)

- One or more outputs



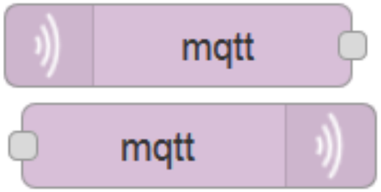
```
Return msg
```

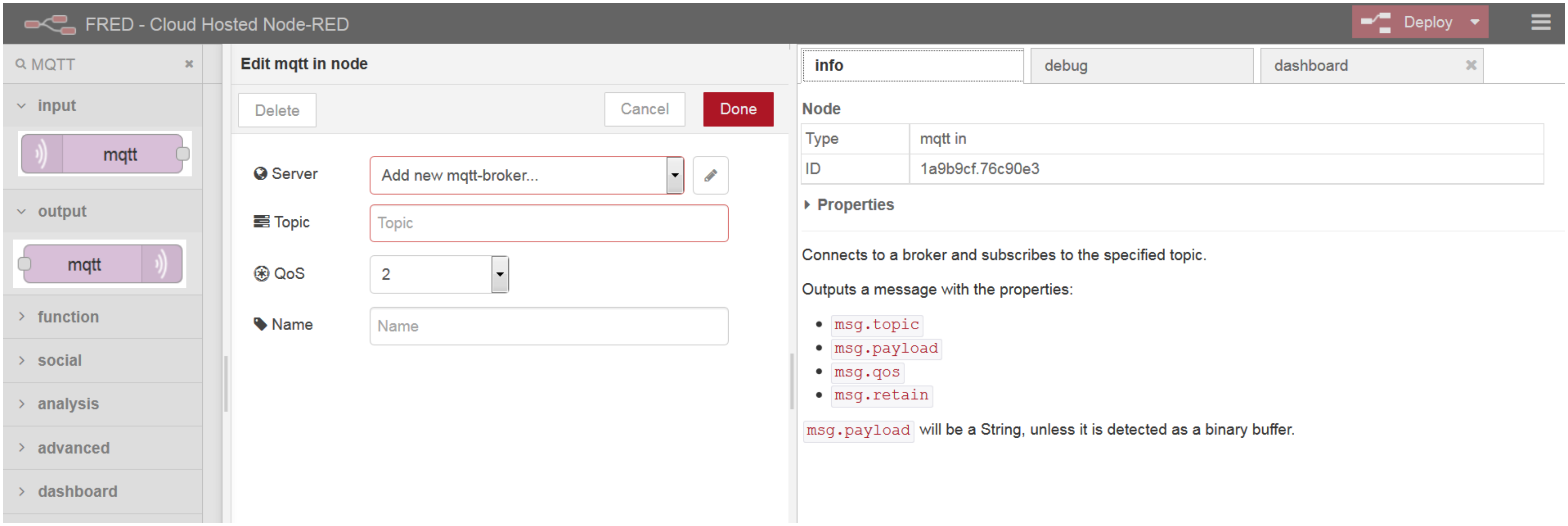
```
Return [msg1, msg2, msg3]
```

```
Return [msg1, [msg2, msg3]]
```

MQTT Nodes

- Input Node: Subscriber
- Output Node: Publisher





Edit mqtt in node

Server: Add new mqtt-broker...

Topic: Topic

QoS: 2

Name: Name

Node

Type	mqtt in
ID	1a9b9cf.76c90e3

Properties

Connects to a broker and subscribes to the specified topic.

Outputs a message with the properties:

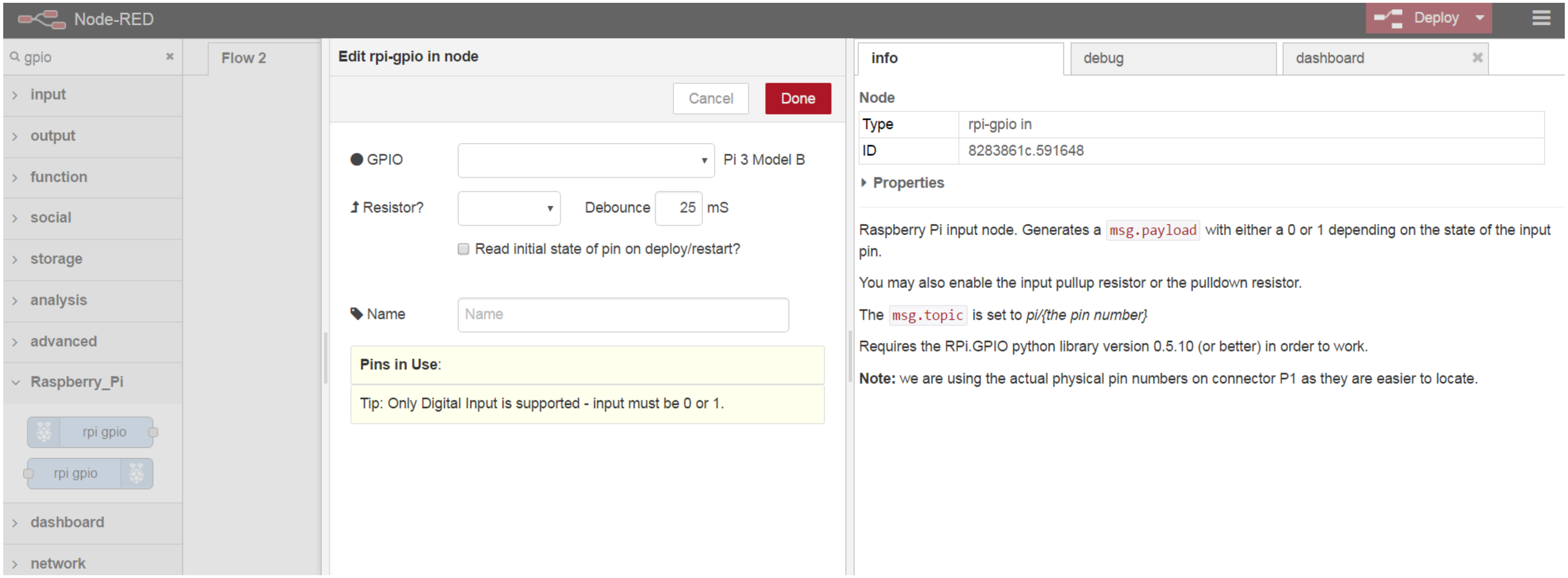
- `msg.topic`
- `msg.payload`
- `msg.qos`
- `msg.retain`

`msg.payload` will be a String, unless it is detected as a binary buffer.

GPIO Nodes

- Input Node: Read PIN
- Output Node: Write PIN

rpi gpio
rpi gpio



Edit rpi-gpio in node

● GPIO: Pi 3 Model B

Resistor? Debounce 25 mS

□ Read initial state of pin on deploy/restart?

Name: Name

Pins in Use:

Tip: Only Digital Input is supported - input must be 0 or 1.

Node

Type	rpi-gpio in
ID	8283861c.591648

▸ Properties

Raspberry Pi input node. Generates a `msg.payload` with either a 0 or 1 depending on the state of the input pin.

You may also enable the input pullup resistor or the pulldown resistor.

The `msg.topic` is set to *pi/{the pin number}*

Requires the RPi.GPIO python library version 0.5.10 (or better) in order to work.

Note: we are using the actual physical pin numbers on connector P1 as they are easier to locate.

SIMPLE DEMO

Limitations of Node-RED

Some situations where Node-RED may not be the first choice include:

- Complex multi-function IoT applications:
  - Node-RED excels at rapid application development
  - Sub-flows help to master complexity
  - However, when an application gets above a certain size, it becomes complex to visually program and manage through Node-RED.
- Flow-based programming has its weaknesses:
  - E.g., Node-RED is cumbersome when handling loops.
- Specific use cases:
  - Flow-based programming is a general purpose model and not targeted or optimized for specific needs, for example *Data Analytics* or *User Interface development*.

Conclusions

- Node-RED wires together building blocks, using a visual tool to rapidly create simple flows that actually carry out sophisticated real-world tasks.
- Node-RED is a rapid application development tool for the IoT
- Node-RED has evolved to being used for a variety of tasks, not just IoT programming; E.g., web apps, social media apps, back-office integration, IT task management..
- Node-RED has limitations for complex applications and GUI.

IoT Misc: Energy-efficiency, Cloud, Big Data, Interoperability, Security, etc

Chapter Outline

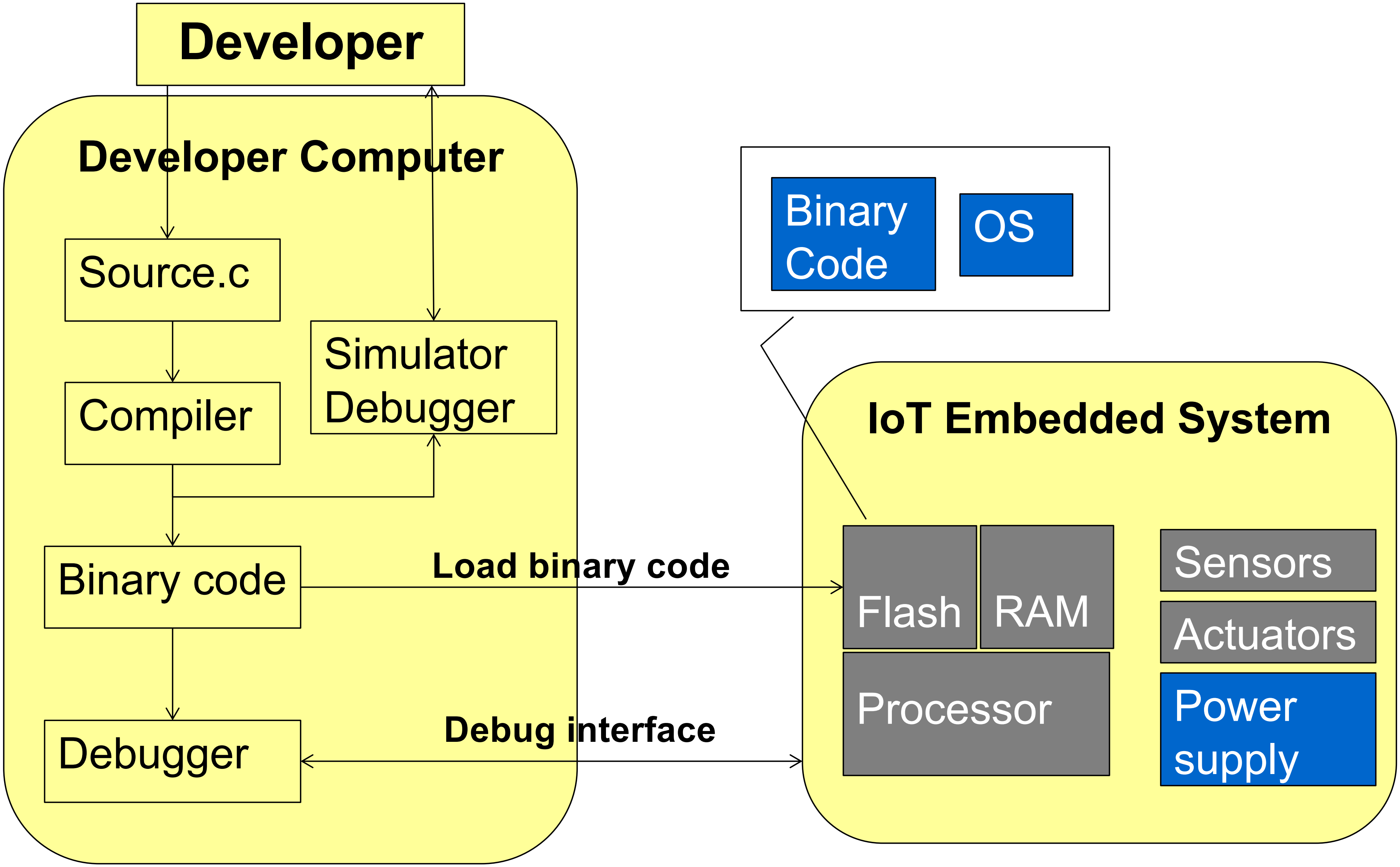
- Energy Efficiency
- IoT Cloud, IoT Big Data
- Interoperability

Energy Efficient Software Development for IoT

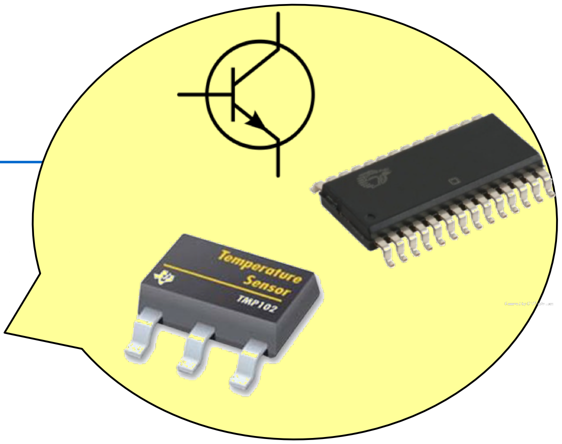
Outline & Objectives

- Motivation
- Energy efficiency
  - Large resources but cooling
  - Less resources but battery
- Environments
  - Battery-powered devices (offshore computation to edge or cloud)
  - Gateways (edge/fog)
  - Data centers (cloud)
- Tradeoff: {Latency & Energy} or {Space & Time}

IoT Software Development

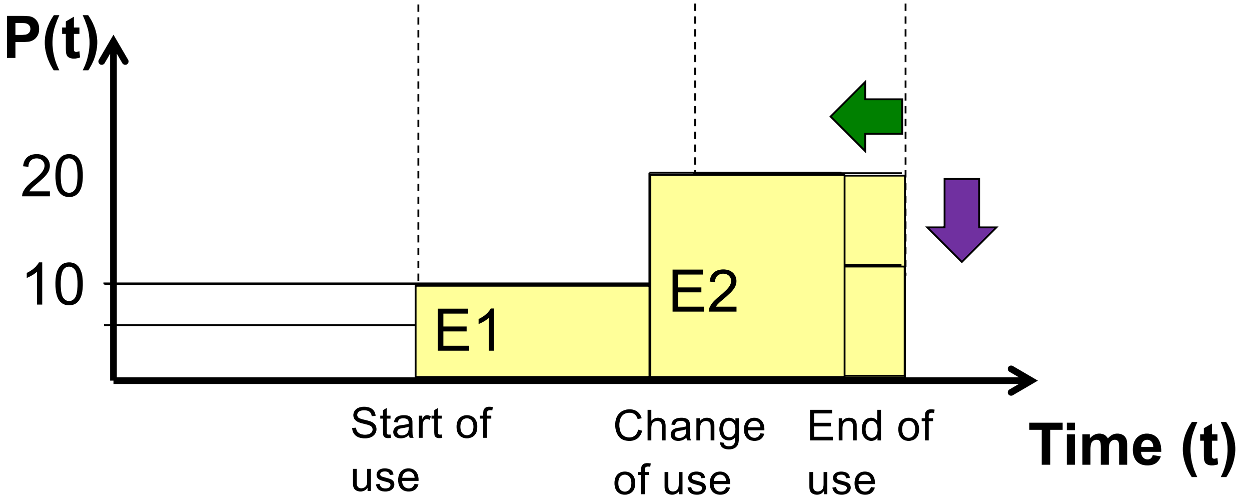


Power vs Energy



Energy Consumer	💡	💡	💡
Power (P in Watt)	20	20	40
Energy (E in Jule=Watt*Sec)	0	10	20

$$E = \int_{Start}^{End} P(t)dt$$



Need for both: Power Efficiency and Energy Efficiency

Power Philosophy

- Hardware (HW) dissipates energy …

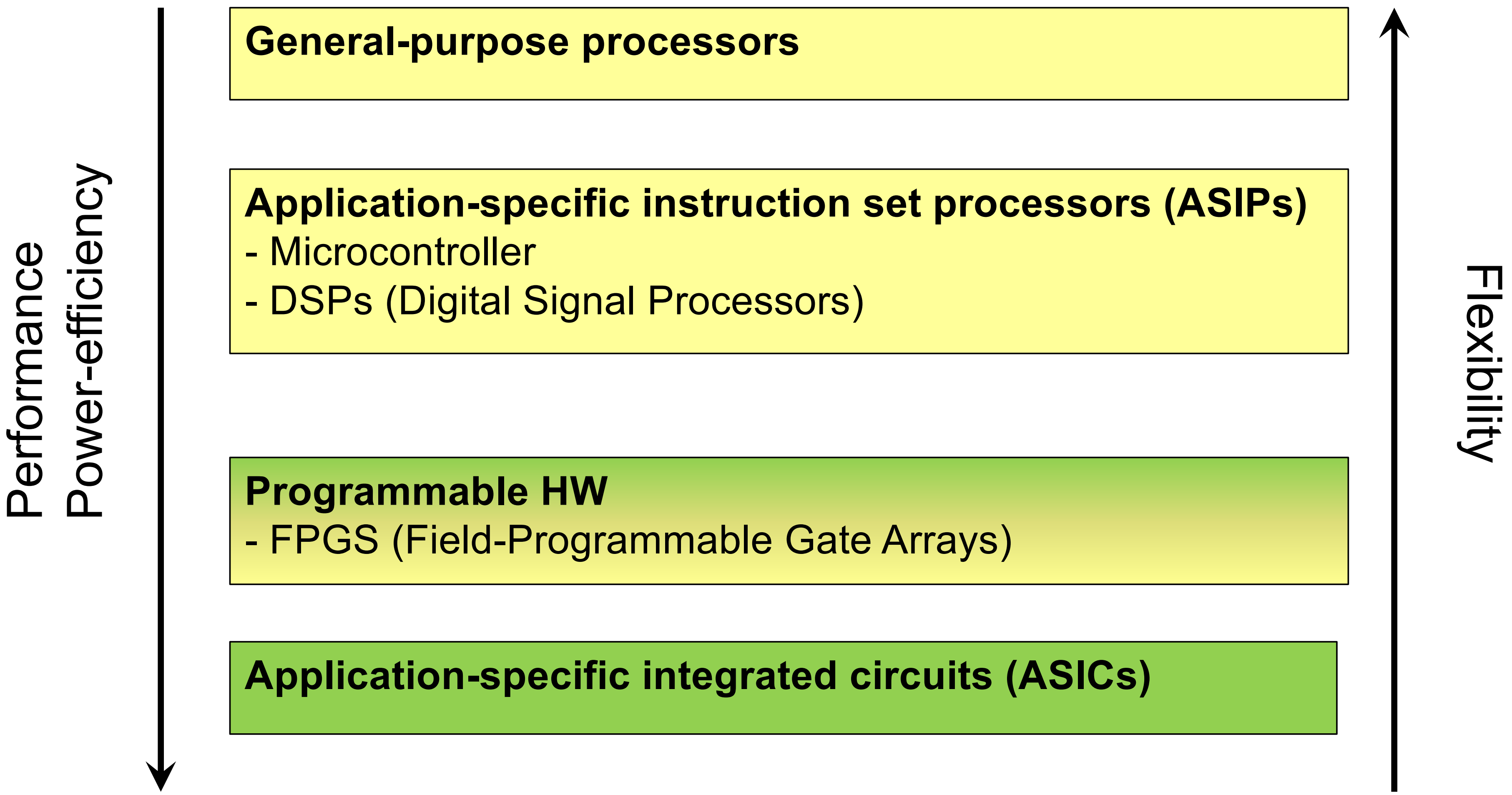
  … Software doesn‘t (but it tells Hardware to!)

→ Chose HW technology for best power efficiency

→ Use HW in dependency of required computing activities (zero activity = zero energy)

- Think System: It‘s how the „box“ performs, not its single components
  - Make OS/App/SW aware of the power and energy performance
  - Provide OS/App/SW options for controlling power efficiency
- Think Network of Systems: It‘s how the „networked boxes“ perform

HW-Level: The Power/Flexibility Conflict



Performance

Power-efficiency

General-purpose processors

Application-specific instruction set processors (ASIPs)
- Microcontroller
- DSPs (Digital Signal Processors)

Programmable HW
- FPGS (Field-Programmable Gate Arrays)

Application-specific integrated circuits (ASICs)

Flexibility

SW-Level – OS System Services

- During use
  - Switch off peripherals when they are not in use.
    - the best way to save energy with any electronic device is to simply switch it off.
    - facility is not as simple as it sounds, as some types of peripheral (e.g., a network interface) take a period of time to configure, or may continue transferring data after the SW has finished addressing it..

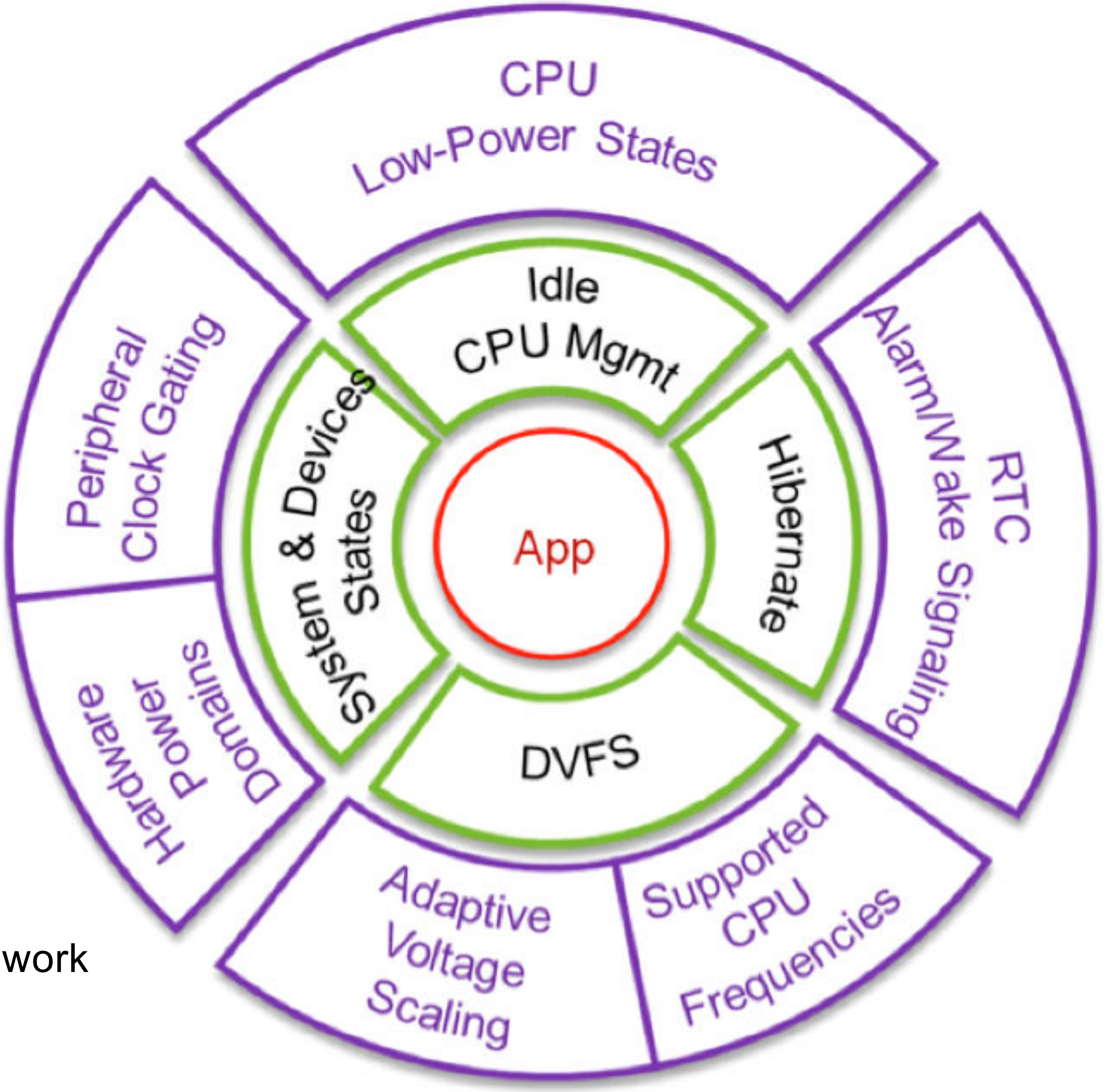
    → power-aware device driver
  - Adjust the frequency (f) and voltage (v) of the CPU according to the current performance requirements ("Dynamic Voltage and Frequency Scaling" - DVFS).

    $P \propto f * v^2$ ($\propto$ = is proportional to)
- Low power device modes
  - Standby, hibernate, etc

SW-Level – OS System Services

Example:

RTOS (Real-Time Operationg System)



🟪 HW power management

🟩 RTOS power management Framework

🟥 Application SW

SW-Level – Application Programming

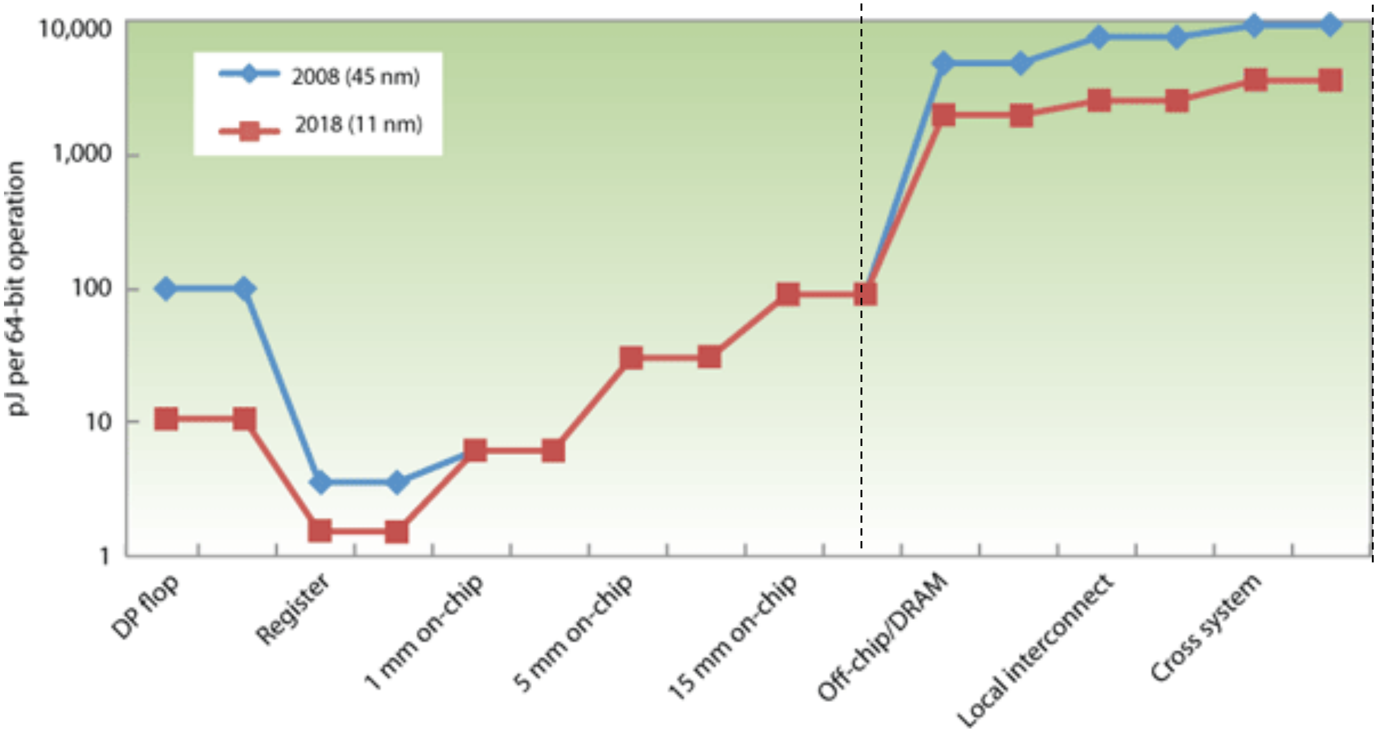
- Carefully analyze the application and define the “use cases”
- Meet user expectations
  - A wearable medical monitoring device would need to run for >18 hours on a single charge
  - A sensor node in a forest: a few years battery lifetime expected.
- Write energy-efficient code
  - Frugal code: Avoid unnecessary activities (max idle time, reduce the total number of instructions)
  - Exploit duty cycling (idle, sleep, listening, active ..)
  - Controlled degradation of user experience
  - Suppress/reject unnecessary data
  - Minimize movement of data

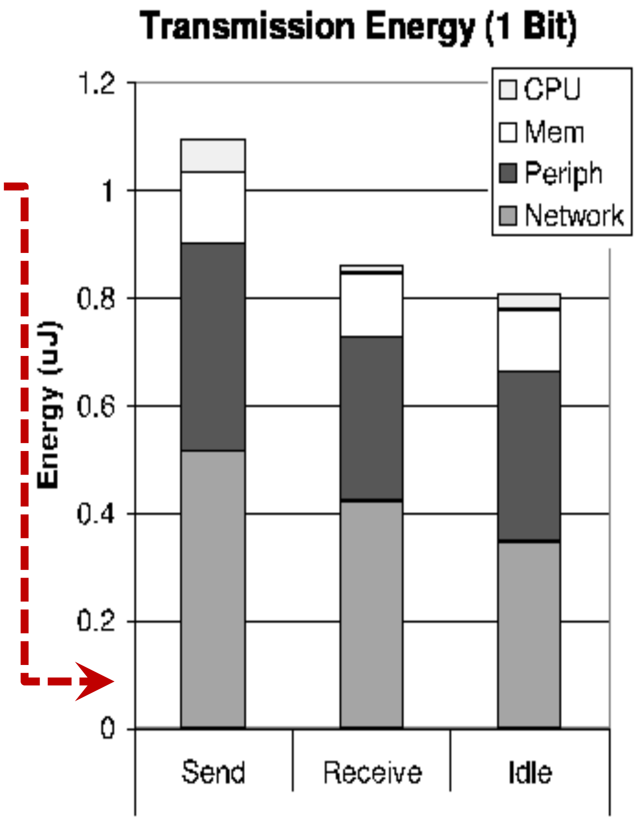
Data Movement Energy Overhead

On-chip

Off-chip

Via-network



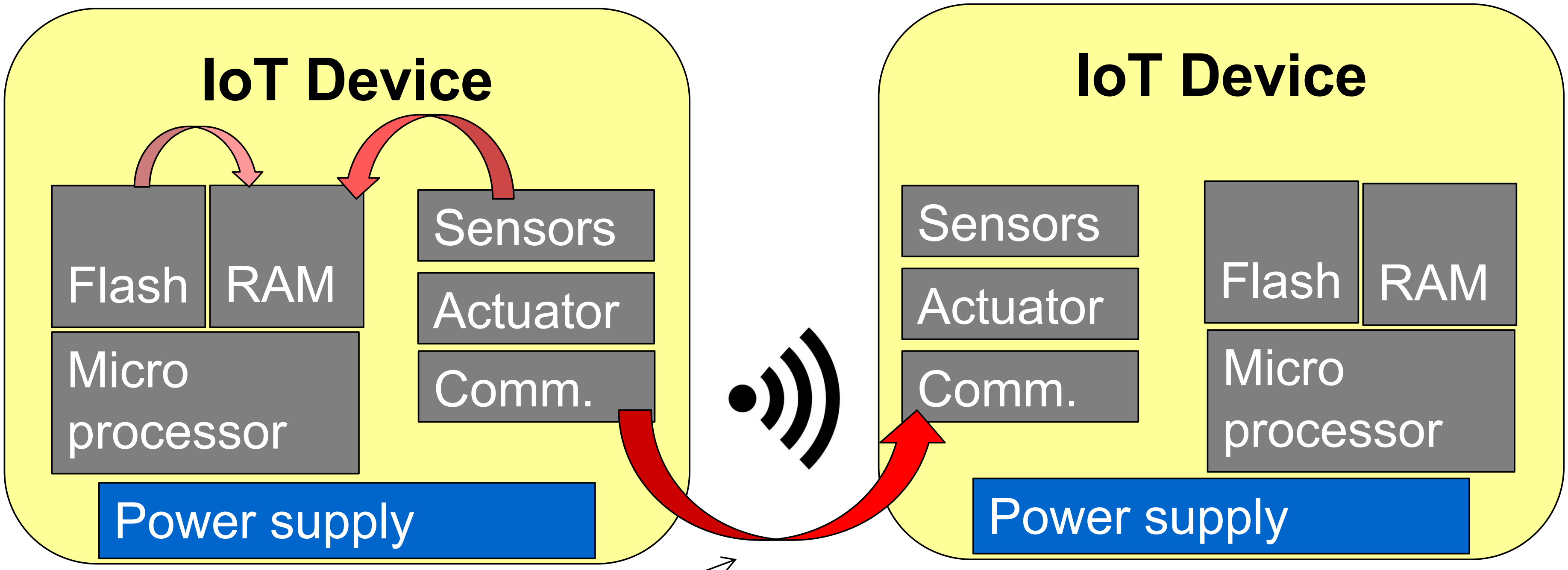


Wifi, 2003

Moving data consumes significant energy

SW-Level – Application + Middleware

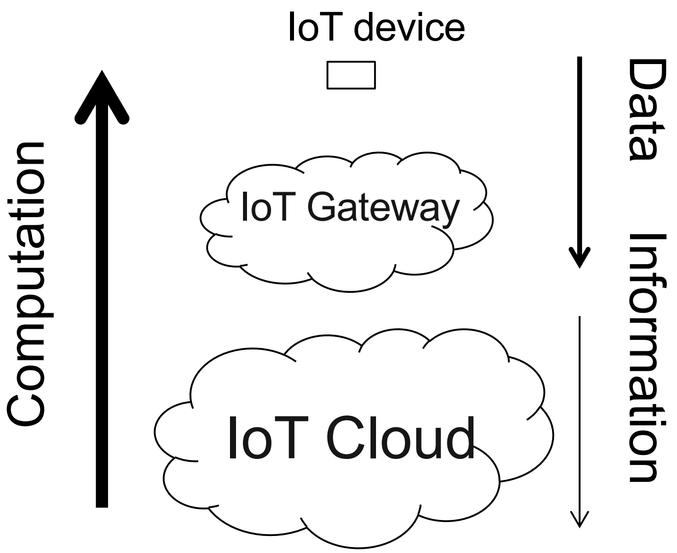
- Maximize data locality
- Bring processing to data



Design (distributed/middleware) algorithms to minimize data movement!

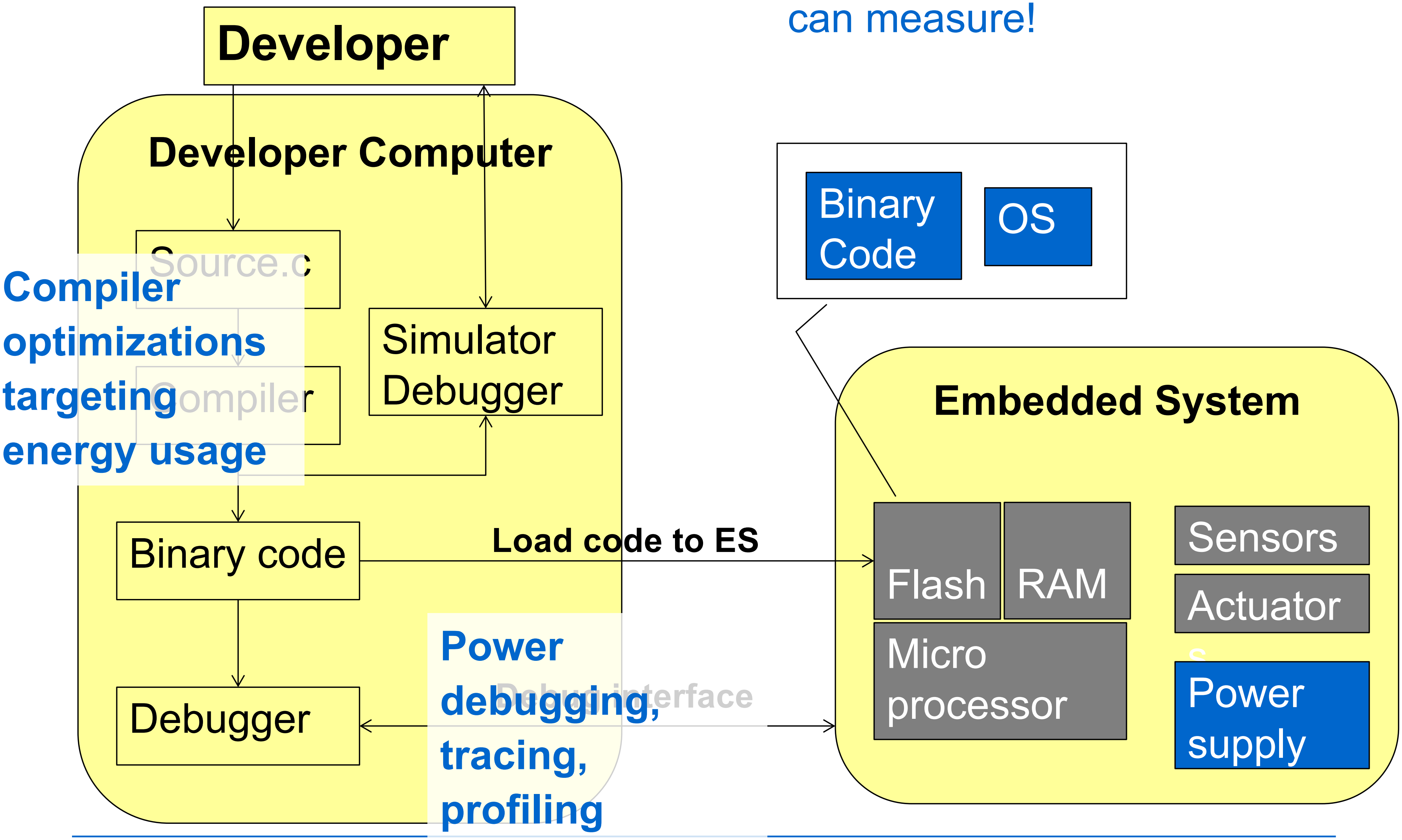
Reduce IoT Data Movement

- **Bring computation to data (IoT devices) rather than data to computation (cloud)**
- **Move Information rather than Data**



Tools for Developers

You can only improve what you can measure!



Compiler optimizations targeting energy usage

Power debugging, tracing, profiling

Principles of Energy-Efficient IoT

- System-level thinking
  - Cross network-layer
  - Cross abstraction layers
- HW-SW-MW-OS co-design
  - Architect HW & SW as efficiently as possible (reflecting the task)
    - Strive for no work → no power
- The arrangement of your data matters
  - Do not move data, move information
  - Process data locally

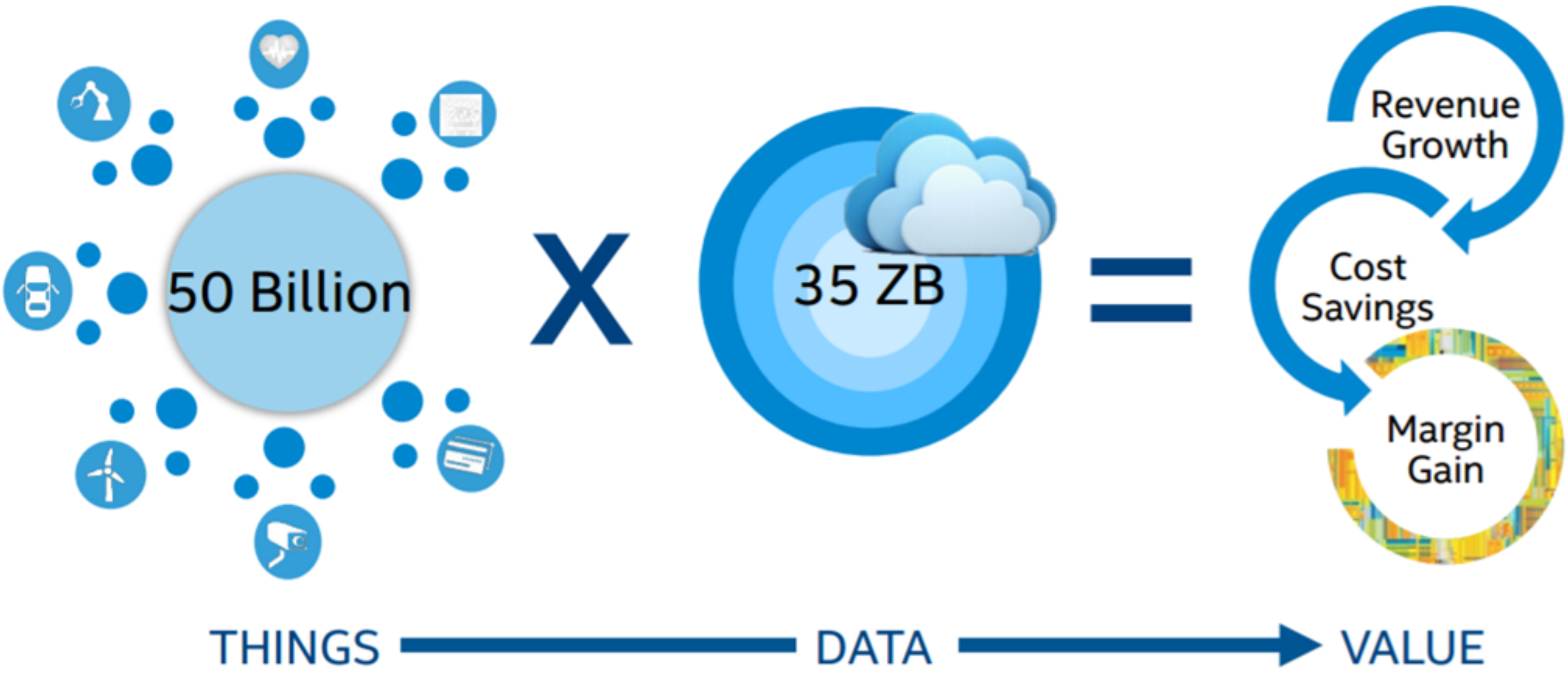
IoT Cloud and Big IoT Data

Challenges that Could Slow IoT Growth

- Security & Privacy
- Underutilized data
- Fragmentation of vertical markets→ Interoperability and standards
  - IT/OT and control/data integration
  - Legacy infrastructure

Towards Unprecedented Values

Internet of Things x Big Data = Unprecedented Value



50 Billion

X

35 ZB

=

Revenue Growth

Cost Savings

Margin Gain

THINGS

DATA

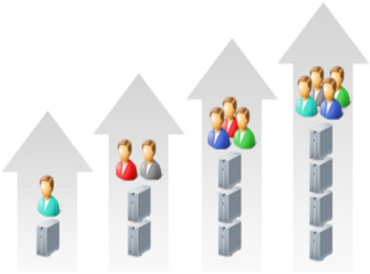
VALUE

Src: Intel, AMS Research, Gartner, IDC, McKinsey Global Institute

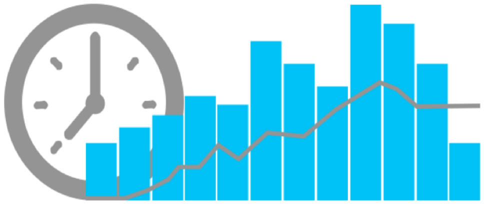
Convergence of IoT, Big Data and Cloud

- For IoT, **connectivity is just an enabler but the real value** of IoT is on **data** (business insight/data-driven economy)
- For Big Data, ***data collection*** is one of the main concern, and IoT can play an important roles for data collection and data sharing
- For Big Data, data is nothing without real business value insight
- Cloud offers ***Everything as a Service*** business model for IoT and big data.
- **IoT is a King, Big data is a Queen and Cloud is a Palace**

Key Requirements of IoT-Big Data Platform

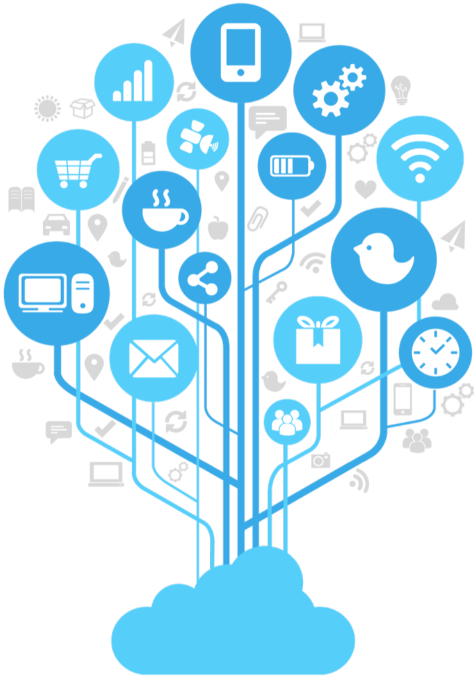


Scalable



Real-time

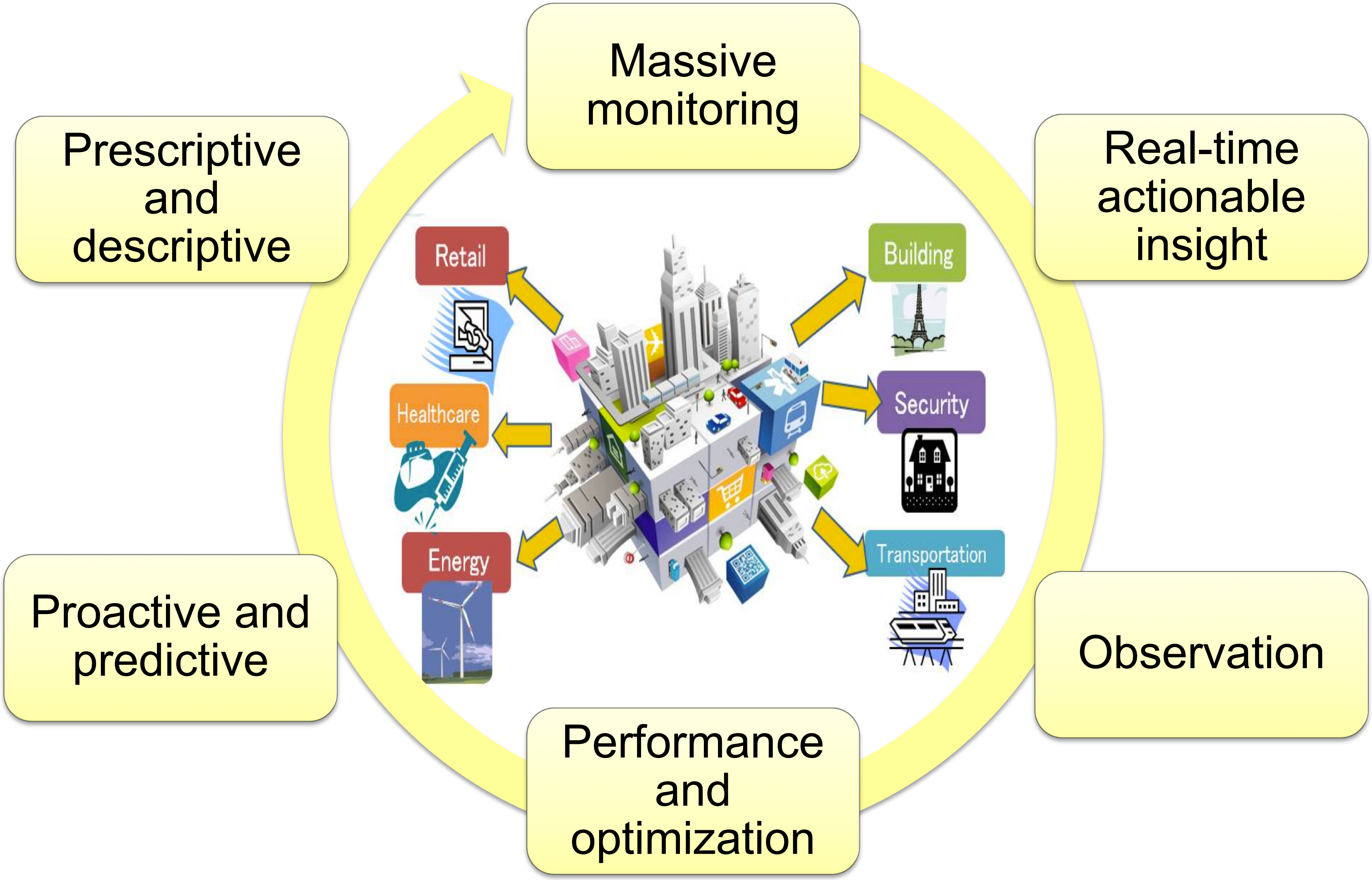
Security and privacy



Intelligent and dynamic

Distributed and decentralized

Cloud-based IoT Big Data Applications



Massive monitoring

Real-time actionable insight

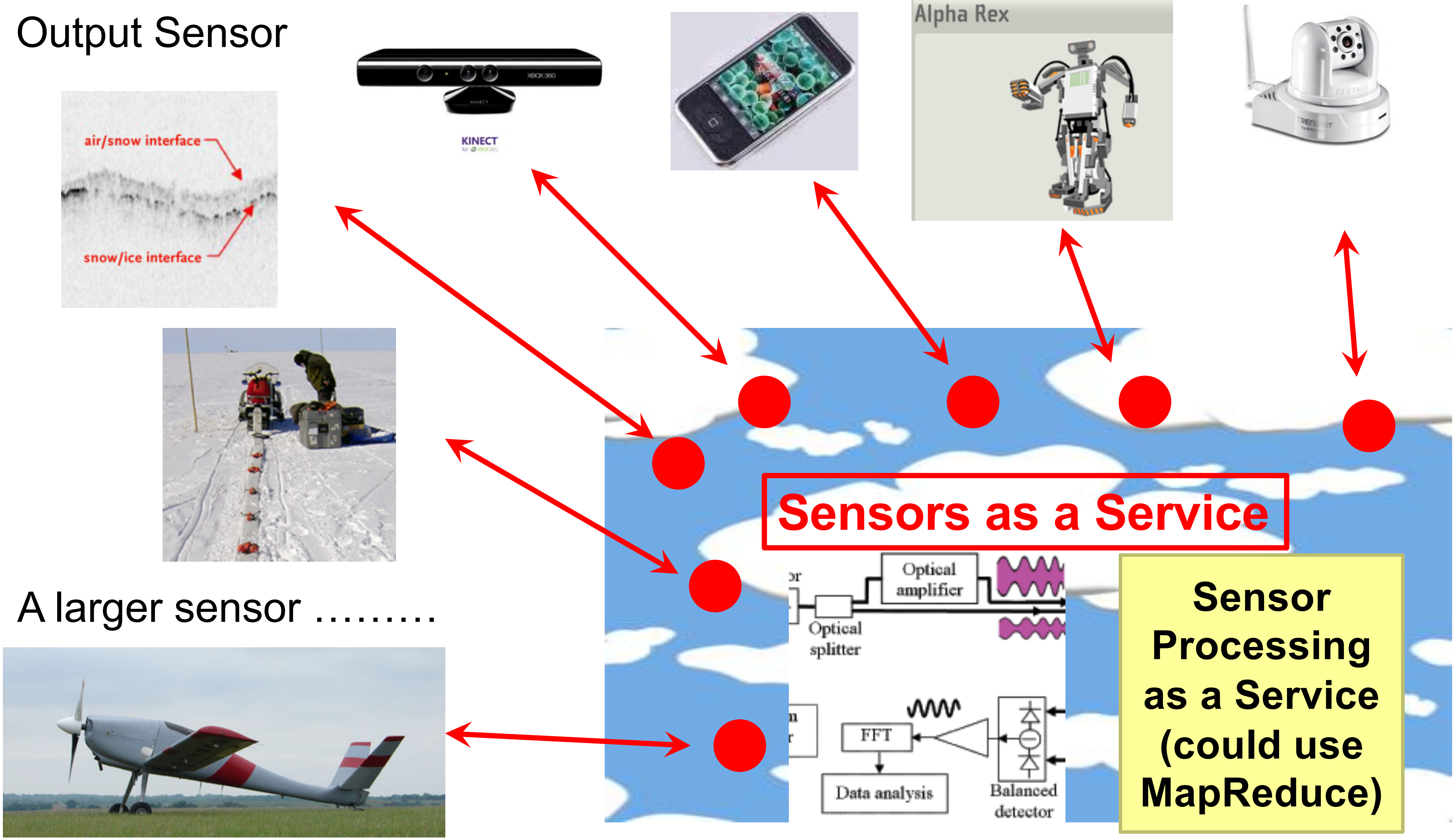
Observation

Performance and optimization

Proactive and predictive

Prescriptive and descriptive

Everything/Sensor as a Service



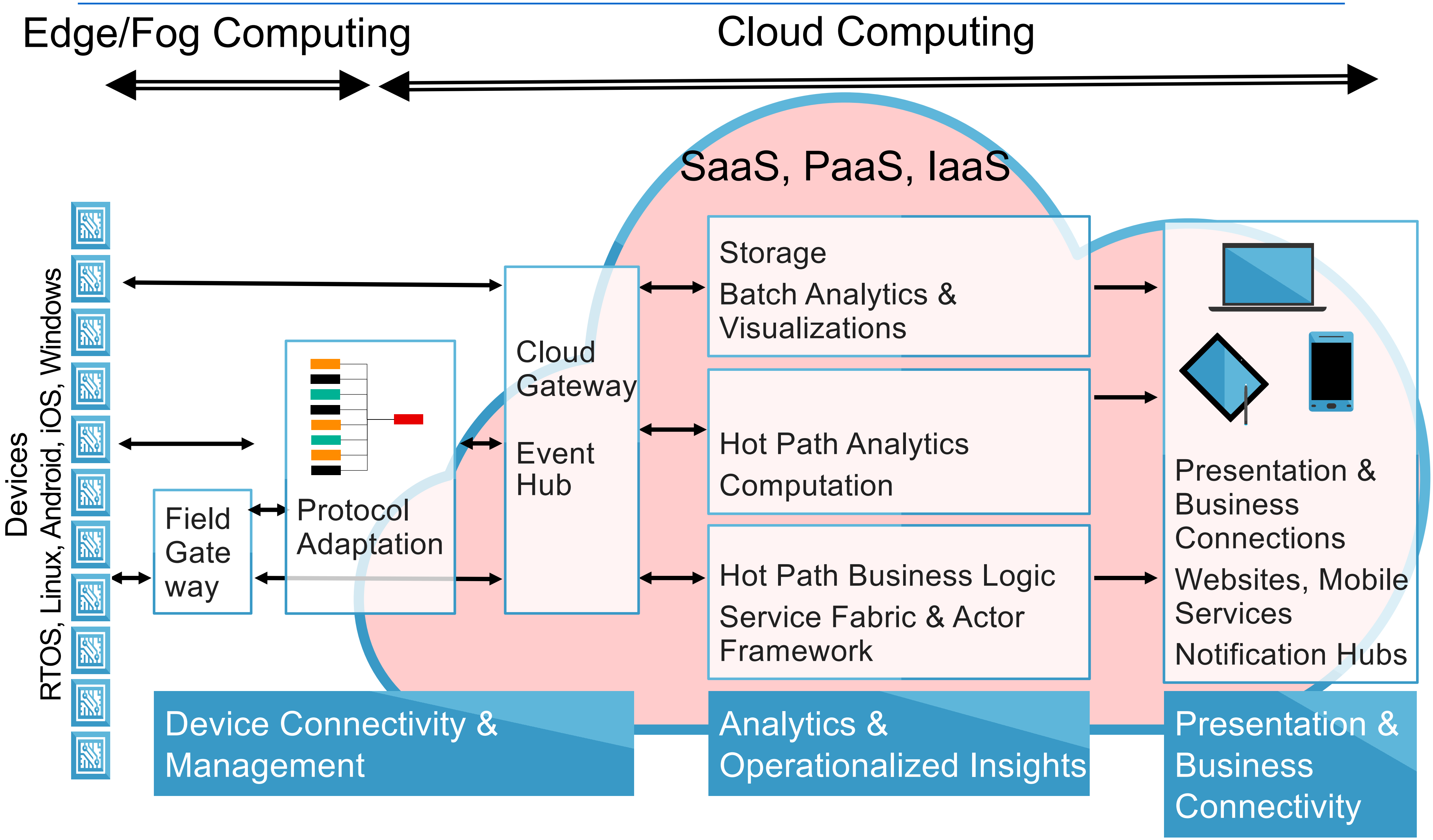
Output Sensor

A larger sensor ....

Sensors as a Service

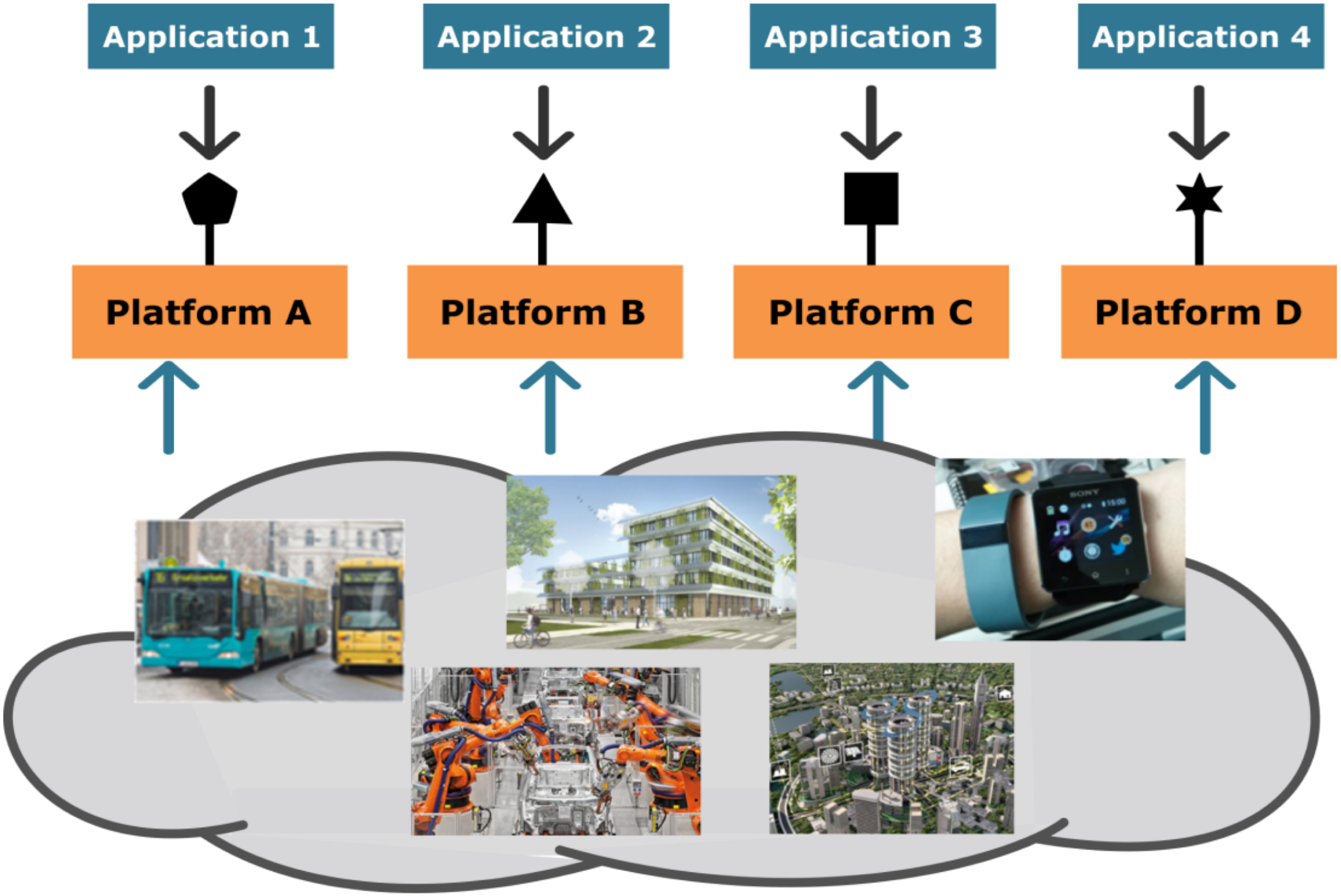
Sensor Processing as a Service (could use MapReduce)

IoT Device & Computing Patterns


Edge/Fog Computing
Cloud Computing
SaaS, PaaS, IaaS
Devices
RTOS, Linux, Android, iOS, Windows
Field Gate way
Protocol Adaptation
Cloud Gateway
Event Hub
Storage
Batch Analytics & Visualizations
Hot Path Analytics
Computation
Hot Path Business Logic
Service Fabric & Actor Framework
Presentation & Business Connections
Websites, Mobile Services
Notification Hubs
Device Connectivity & Management
Analytics & Operationalized Insights
Presentation & Business Connectivity

Interoperability to Break Silos Challenge: Semantic Interoperability

Current Challenge in IoT: Weak Interoperability



Src: H2020 BiG-IoT Project

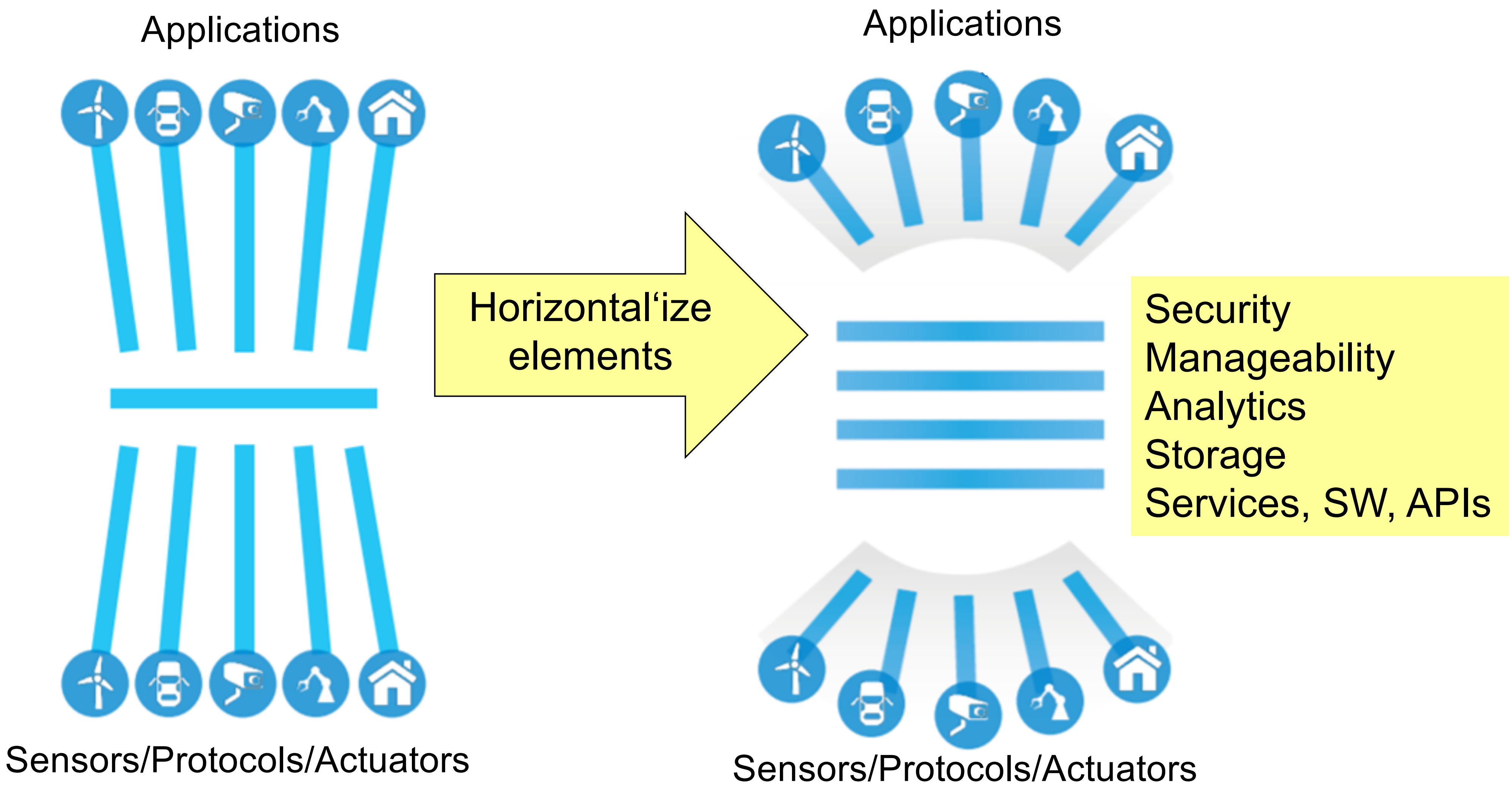
Coping with Weak Interoperability

- Fragmented value chains can kill innovations!
- The biggest challenges of IoT are (a) achieving interoperability between platforms & applications, and (b) creating standards & interfaces.

→ Cross-domain middleware is critical

→ Standardization activities are important for scaling IoT

Scaling IoT Through Interoperability



Applications

Applications

Horizontal‘ize elements

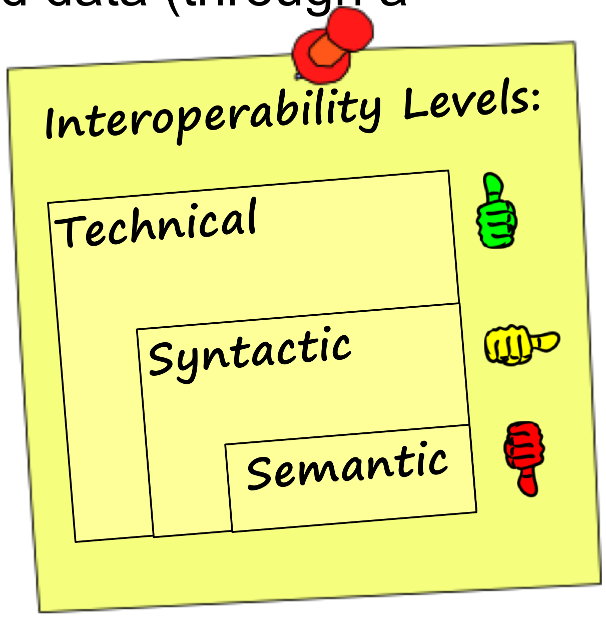
Security
Manageability
Analytics
Storage
Services, SW, APIs

Sensors/Protocols/Actuators

Sensors/Protocols/Actuators

What is Interoperability?

- Uniform move of data from one system to another, i.e., 2 or more systems can share data AND use it.
- Levels of Interoperability
  - Technical: Systems can communicate data to each other
  - Syntactic: A system can READ received data
  - Semantic: A system can UNDERSTAND received data (through a data model)



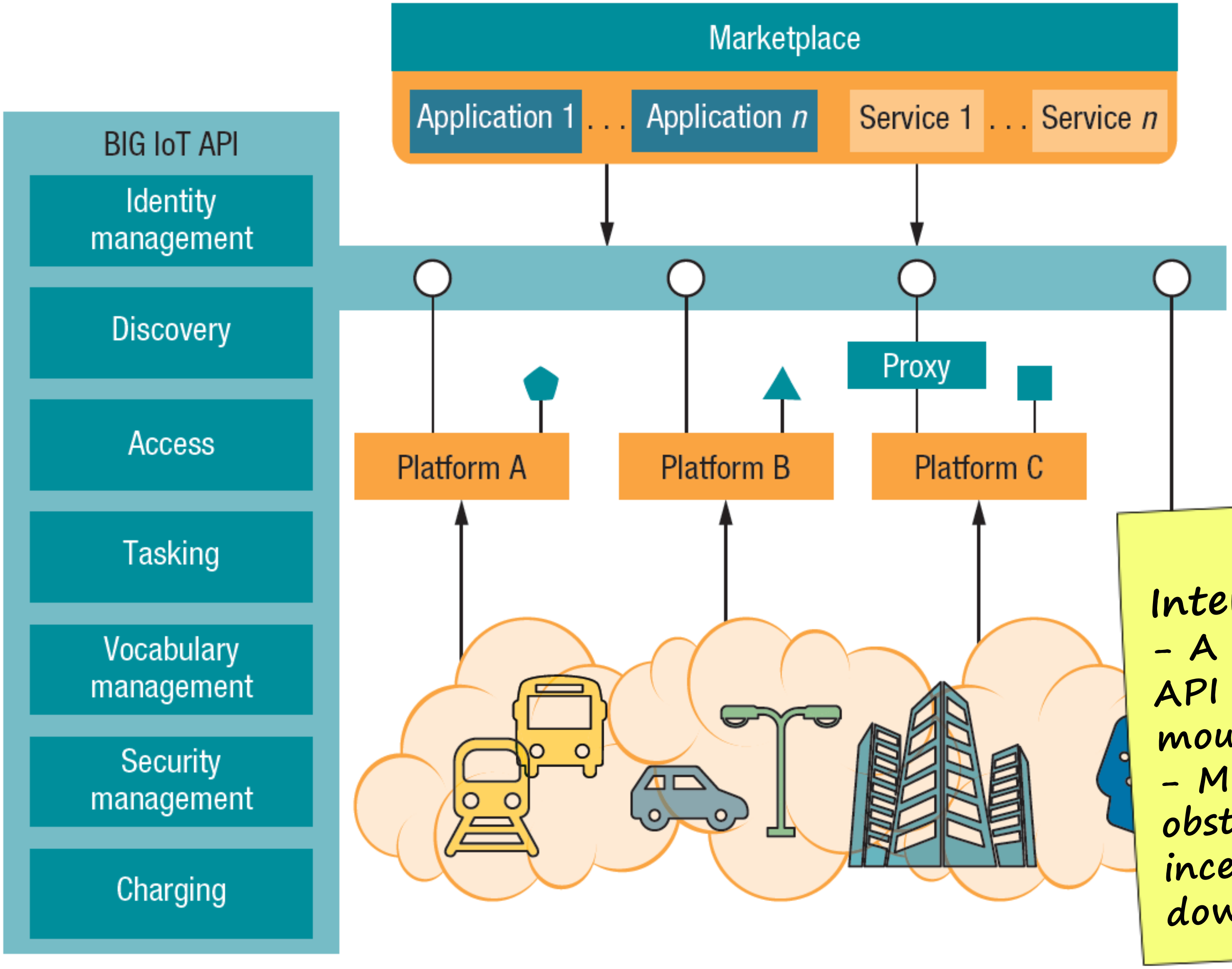
Interoperability Levels:

Technical

Syntactic

Semantic

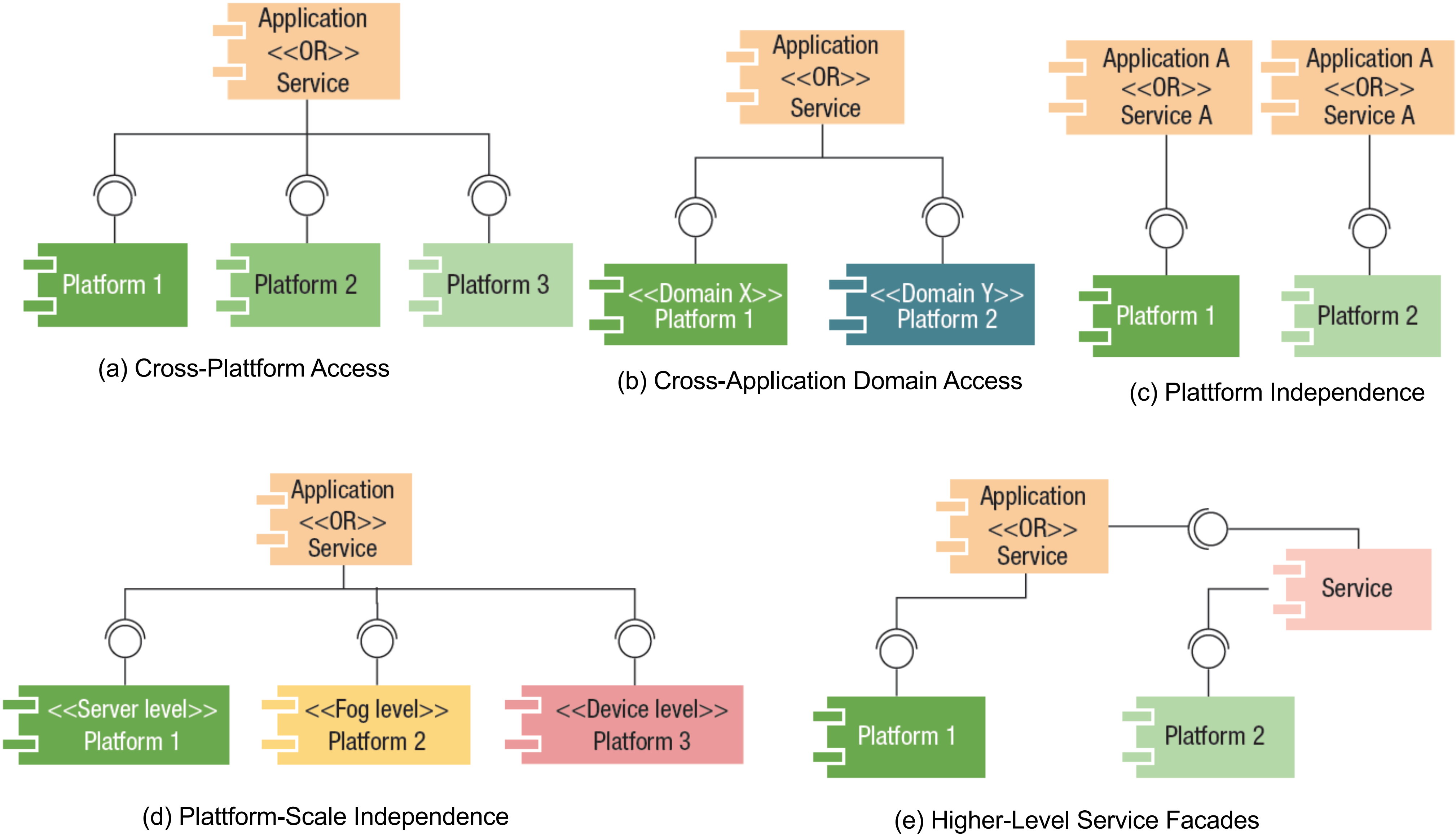
BIG-IoT Approach



Interoperability via:
- A single language + API („ears and mouths“)
- Marketplace: remove obstacles and create incentives to break down barriers

Src: H2020 BiG-IoT Project

Five interoperability patterns



(a) Cross-Plattform Access

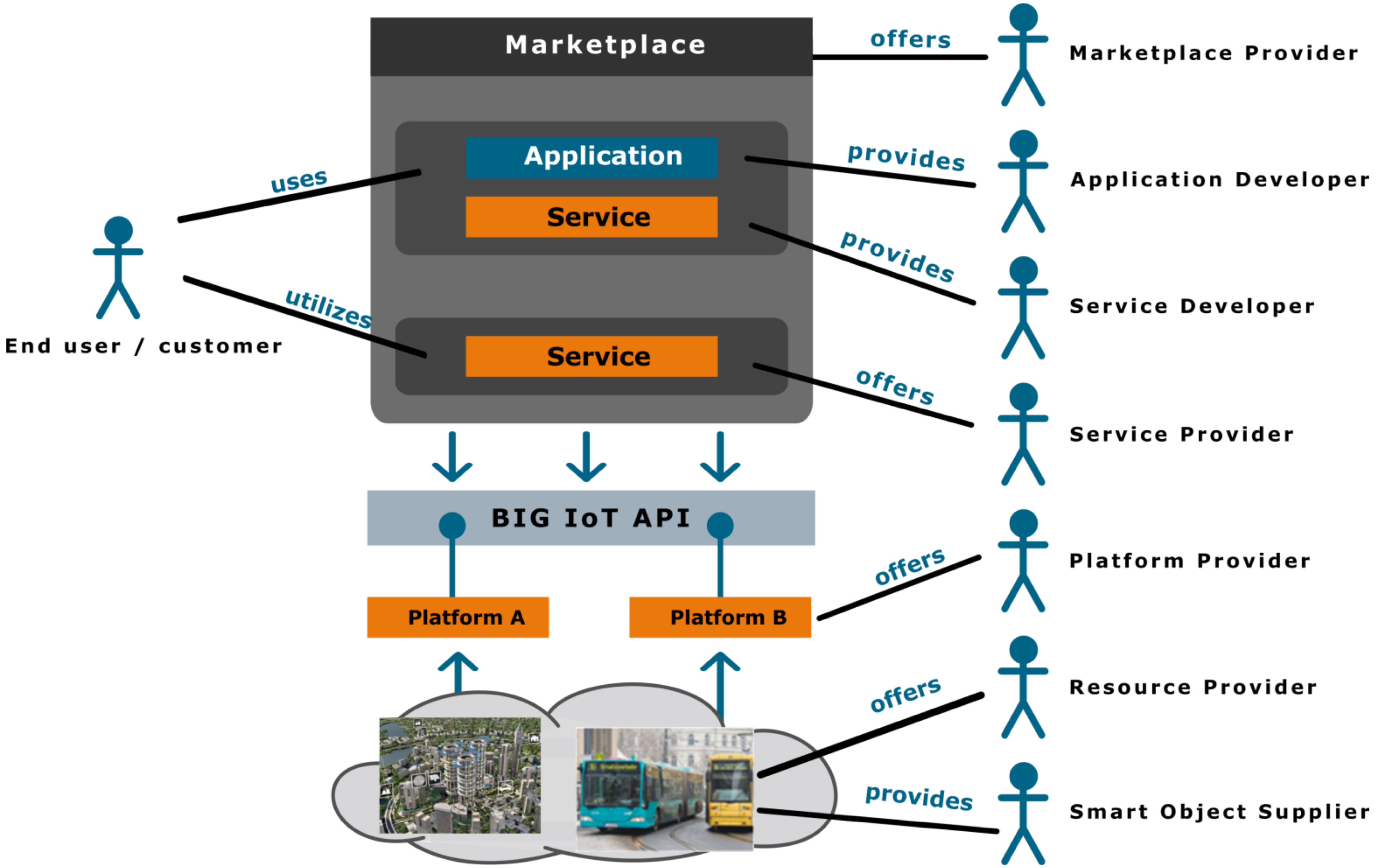
(b) Cross-Application Domain Access

(c) Plattform Independence

(d) Plattform-Scale Independence

(e) Higher-Level Service Facades

The IoT Ecosystem



Src: H2020 BiG-IoT Project

IoT Standardization to Foster Interoperability

- 7 SDO (ETSI, ATIS, TIA, CCSA, TTA, ARIB, TTC): **OneM2M** (since July 2012)
- ETSI: **M2M** service layer standard (published Jan 2012)
- Oasis **MQTT**
- IETF: **CORE** (Constrained RESTful Environments), ROLL, RPL, 6LoWPAN, CoAP
- OMA **LWM2N**
- **3GPP Machine Type Communication (MTC)**
- AllSeen Alliance, AllJoin standard

- OpenADR (Open Auto-Demand-Response) for smart grids

- IEEE 802.14.5, WirelessHART, ZigBee, DASH7, Bluetooth, UWB, ..
- LoRa Alliance, Sigfox UNB, ..

- Eclipse Open Source
- **Etc**

Summary

- IoT cuts across nearly every vertical sector
- Security & interoperability are primary concerns across the industry
- Protocol normalization enables developers to write applications that connect into legacy systems and protocols seamlessly
- API‘s are critical to scaling and developing IoT systems
- Turning Data into insights requires edge to cloud analytics

END

Thank you for your attention!