BEYOND CONNECTIVITY: IOT PROGRAMMING & DATA MODELING

Shao-Wen Yang, Ph.D.
Staff Research Scientist
Intel Labs, Intel Corporation

Content may contain references, logos, trade or service marks that are the property of their respective owners
Executive Summary

• What are beyond connectivity?
  • Making *SENSE* of the data
  • Making *USE* of the data
DATA MODELING

Making *SENSE* of the data
These Differences Serve No One

- Manufacturers have to include ALL the right protocols
- Service providers have to choose a single ecosystem or build their own proprietary solution
- Customers have to choose a single ecosystem and can’t choose products “outside of the plan”
How should the IoT work?

Creation of new devices should scale at Internet speed

- New interfaces should take minutes to develop, not months

All ecosystems and devices should work together

- The device maker shouldn’t worry about being isolated by a technology choice

Verification should be simple

- A great idea shouldn’t be hampered by its origin or an unnecessarily lengthy process
THE CONSTRUCTIVE DEVICE DATA MODEL
(SCALES AT INTERNET SPEED)

• Choose a generic description strategy (e.g. RAML, JSON schemas)

• Start with physical properties (e.g. temperature, mass)

• All new devices are defined as collections of physical properties and previously defined devices (e.g. a thermostat is a collection of temperature, thermometer and switch)

• Abstract devices can also be defined (e.g. Clarke’s house, upstairs bedrooms)
THE DERIVED DEVICE DATA MODEL
(ALL ECOSYSTEMS WORK TOGETHER)

- ALL interoperable devices are defined exactly once in the common data model (CDM)
- Devices defined in other ecosystems (AllSeen, UPnP, etc.) are derived from devices in the common data model
- The definition of derived devices allows for differences in ecosystems (property names, variable types, range differences and conversions)
In operation, a shim layer (code stubs automatically generated from the device data model) provides for conversion between ecosystems.

Since all ecosystems derive from the common data model, there are at most two conversions.

The conversion can happen in a gateway, in the cloud or in end devices.
THE ONEIOTA TOOL
(VERIFICATION IS SIMPLE)

• A crowd-sourced Integrated Development Environment (IDE) for the Internet of Things device models (oneIoTa.org)

• RAML & JSON validated and syntax aware editors with shared editing

• Automatic support for derived models and multiple organizations

• Submission and approval process per organization
SIMPLE
"Write-once-deploy-everywhere"

RESILIENT
"Deploy-once-run-forever"

EFFICIENT

User Interface

Diagnosis & Prescriptive Analytics

Distributed Orchestrator

Global Mapper

Distributed Runtime

Connectivity & Communication

Design Consideration: Usability for end-users/installers

Programming Model
- **Syndrome**: IOT systems are hardly programmable
- **Root-cause**: Most IOT users are end-users
- **Gap**: Current programming models are not at the proper level of abstraction for IOT
- **New insights**: Abstraction layers for “write-once-deploy-everywhere” and “deploy-once-run-forever”

Anomaly Detection for Auto-Reconfiguration
- **Syndrome**: IOT systems are unreliable
- **Root-cause**: IOT devices are usually resource constrained
- **Gap**: The prior art is mostly centralized
- **New insights**: Exchanging models rather than continuous large volume data

Maximal Bipartite Matching

Missing Data Imputation
- **Syndrome**: IOT data can be intermittent
- **Root-cause**: IOT devices are diverse and may operate in harsh environments
- **Gap**: The prior art assumed a share variance
- **New insights**: Variance-aware collaborative filtering
PROGRAMMING MODELING

Making **USE** of the data
Programming Paradigm Shift from PC/Internet to IOT

• Help close the gaps between User Convenience and System Efficiency in IOT
• How to program IOT at scale? Write-once-run-everywhere?
• How to deploy IOT at scale? Deploy-once-run-forever?
Scope & Key Impact

- **Scope:**
  - **Why?** 20B devices are hardly possibly manageable by human operators.
  - **How?** Bridge the gaps between needs and reqs, and reqs and deployment.

- **What?** Abstraction makes services portable, reusable and sharable.
Methodology

Abstracted Programming Modeling

Design
• Use existing instances / recipes – write-once-deploy-anywhere
• Otherwise, drag-n-drop functional blocks and define relationships

Discovery
• Querying devices
• Determining their capability

Deployment
• Upload the code / logic
• Heartbeating

State of the Art

Design
• Drag-n-drop physical blocks
• Define relationships between physical devices

Discovery
• Querying devices
• Determining their capability

Deployment
• Upload the code / logic
• Heartbeating
Abstraction Layers
Software Defined Everything with Abstractions

Application A

Application B

LS1
LS1
IC2
IC2
IC1
IC1
IS1
IS1
Cloud
Cloud
GW1
GW1
PS1
PS1
LS1
LS2
IC1
IC2
IS1
Cloud
GW1
PS1

ZigBee HA
Light Sensing
Bluetooth Smart
Light Sensing
IP Camera
ZigBee HA
Infrared Sensing
Cloud
Gateway
ZigBee HA
Power Switch

Light Sensing
Presence Sensing
Analytics
Data Store
Light Actuating

Description of Application A

Description of Application B
Visual Interface as an Example
Concluding Remarks

• What are beyond connectivity?
  • Making *SENSE* of the data for *interoperability*
    • Defragmenting IOT data/device models
    • Abstracting IOT data into semantically meaningfully forms
  • Making *USE* of the data for *usability & reusability*
    • Improving IOT programmability
    • Enabling end-user programming

• Contact me at shao-wen.yang@intel.com