Internet of Things
2017/2018

Frameworks

Johan Lukkien

John Carpenter, 1982
Guiding questions

• What is a programming framework for IoT and what are concerns?
• How to quickly analyze and understand a framework?
What problems does a framework need to solve?
Physical elements: devices and networks

- ‘Things’: low capacity devices
  - (T-S) sensors
  - (T-A) actuators
  - (T-I) identifier (special sensor)
- Infra structure:
  - (I-S) switches (layer 2 connectivity within a network technology)
  - (I-G) gateways
    - converting between two parties
    - different layers of the OSI stack
  - networks
- (S) Storage devices
  - e.g. SAN or NAS, Cloud storage
- (U) User devices: phones, tablets, desktops, laptops
- (E) Embedded devices (containing several functions)
- (F) ‘Fog’: high capacity devices in the vicinity of data generation
- (C) ‘Clouds’: massive storage and execution power
Mapping IoT Architecture elements to devices

balance functionals, extra-functionals and boundary conditions

- **Functional**
  - Sensing (event and state)
  - Actuation (event)
  - Application logic (incl. control)
  - Communication / translation
  - Storage
  - Data, Information (context, semantics, location, identity)
  - **Vertical Analytics**
  - **Horizontal Analytics**
  - Management (of application, of data), UI
  - (APIs for) services, advertisement, discovery

- **Extra functional**
  - **Dependability**
    - reliable, available
    - secure, private
    - safe
  - Performance, QoS
    - response time, latency, throughput
    - processing
    - timeliness
  - (Resource) management
    - program, update, extend
    - sharing, concurrent applications, scheduling
  - Interoperability
  - Mobility
  - Managerial domains, ownership

- **Boundary conditions**
  - Distributed systems
  - Given components
  - Given protocols
  - Network standards
  - Legal matters
  - (Design) Technology
    - languages, tools
  - ... all that is given
What problems does a framework need to solve?

Mapping IoT Architecture elements to devices

- Functional
  - Sensing (event and state)
  - Actuation (event)
  - Application logic (incl. control)
  - Communication / translation
  - Storage
- Data, Information (context, semantics, location, identity)
- Vertical Analytics
- Horizontal Analytics
- Management (of application, of data), UI
- (APIs for) services, advertisement, discovery

- Extra functional
  - Dependability
    - reliable, available
    - secure, private
    - safe
  - Performance, QoS
    - response time, latency, throughput
    - processing
  - (Resource) management
    - program, update, extend
    - sharing, concurrent applications, scheduling
  - Interoperability
  - Mobility
  - Managerial domains, ownership

- Boundary conditions
  - Distributed systems
  - Given components
  - Given protocols
  - Network standards
  - Legal matters
  - (Design) Technology
    - languages, tools

- Physical elements: devices and networks
  - ‘Things’: low capacity devices
    - (T-S) sensors
    - (T-A) actuators
    - (T-I) identifier (special sensor)
  - Infra structure:
    - (I-S) switches (layer 2 connectivity within a network technology)
    - (I-G) gateways
      - converting between two parties
      - different layers of the OSI stack
  - (S) Storage devices
    - e.g. SAN or NAS, Cloud storage
  - (U) User devices: phones, tablets, desktops, laptops
  - (E) Embedded devices (containing several functions)
  - (F) ‘Fog’: high capacity devices in the vicinity of data generation
  - (C) ‘Clouds’: massive storage and execution power

create functionalities and map them onto to devices
- services, applications

support extra-functional properties

within the given boundary conditions

Johan J. Lukkien, j.j.lukkien@tue.nl
TU/e Informatica, System Architecture and Networking
What are particular lifecycle challenges?

• Lifecycles are uncorrelated
  – equipment is bought and installed from different vendors

• Configuration and commissioning concerns:
  – a device must become a (trusted) member of an IoT system
    • bootstrapping, initialization, security keys
  – finding partners, and establishing connections between them
  – setting operational parameters

• Programming concerns:
  – application development / installation
    • applications rely on code executing in a heterogeneous distributed system, consisting of devices in different managerial domains
    • need to develop such codes, install these codes at appropriate places, establish connections, run coordination software somewhere
  – software installs and upgrades are most often vendor defined as opposed to owner defined
How to solve?

- A complete system approach
  - define an ‘ecosystem’
    - of components and services
    - of stakeholders
  - address all components explicitly and design all stakeholders in, e.g.
    - compliant hardware specification
    - app store and API
- Focus on specific applications and extend from that
  - NEST
  - HUE (missed opportunity)

- Solutions should reduce dependencies
  - connect components via the network rather than via a linked library
    - define interoperability at the networking level
    - separate node programming from application programming (HUE, NEST)
  - decouple, break dependencies
    - use styles that support this (e.g. publish & subscribe)
Framework (e.g., Leshan)

- A (programming, development) framework is a tool to develop a system according to a certain style or method.

- The framework supports the different views relevant to development:
  - deployment, physical, process

- The framework itself is also a software system and has views for its own development:
  - might be a bit confusing

A framework consists of
- a ‘static’ part:
  - programming model, data model
    - libraries, APIs
  - life cycle model
  - methods or tooling for development

- a ‘dynamic’ part:
  - a run-time system, or platform
    - entirely separate entity or a library
  - a set of services
    - provided by the platform
    - e.g. binding, installation
  - a process model

Johan J. Lukkien, j.j.lukkien@tue.nl
TU/e Informatica, System Architecture and Networking
Framework for discussing frameworks ;-)  

• What is the goal of the framework? Where is it useful for?  
• What are its elements?  

• Describe the static part (before execution):  
  – programming model  
  – life cycle model  
  – relevant development tooling  

• Describe the dynamic part (during execution):  
  – run-time system  
  – support services  
  – a process model
Guiding questions

• What is a programming framework for IoT and what are concerns?
• How to quickly analyze and understand a framework?