CS4491-02 Fog Computing

IoT Architecture
Guiding questions

• What are relevant architecture concerns for IoT systems?

• What are different architectural options?

• What are typical architecture styles?

• What is (software, system) architecture anyway, and can we make sure we understand what is meant with all these terms?
- domains
  - home
  - mobile / outdoor (fields, ad-hoc)
  - office
  - industry
  - public (city)

- architecture, layered and deployment view
  - devices, things
  - functionality placement alternatives
  - data and control flow

- communication stack, protocols

- lifecycles
  - devices
  - services
  - applications

The IoT Architectural Framework, Design Issues and Application Domain, Gordana Garđas ević et al.
Involved Bodies, Companies and Frameworks

- ETSI (network)
  - European Telecommunication Standards Institute
- ITU (network)
  - International Telecommunication Union
- IETF (protocols)
  - Internet Engineering Task Force
- EU Projects (books, architectures):
  - IoT-A
  - SENSEI
- OSG (data, information)
  - Open Geospatial Consortium
- OMA
  - Open Mobile Alliance
- IEEE (protocols)
The war on standards

- Apple – HomeKit
- Google – NEST + Firebase
- Google – Home
- Google – Thread $\rightarrow$ OpenThread
- ARM – mbed
- Intel – Intel IoT Platform, IoTivity
- Samsung – Artik (AllJoyn, IoTivity)
- Microsoft – Azure IoT
- Cisco – Cisco IoT
- Amazon – AWS IoT
- Philips – HUE (just lighting)

- Threadgroup – Thread
- Open Connectivity Foundation – IoTivity, AllJoyn (absorbed)
Architectural description: overall picture

from existing system to description: analysis, reverse engineering, documentation

from stakeholders and requirements to system: architecture design, detailed design, implementation
Architecture

• An architecture (of a system) is
  – “The fundamental organization of a system embodied by its components [building blocks], their relationships to each other [connectors and interfaces, dependencies] and to the environment and the principles guiding its design [rationales for choices, rules & constraints for building blocks and connectors] and evolution”

  (IEEE Standard P1471 Recommended Practice for Architectural Description of Software-Intensive Systems)

• An architecture description is
  – a collection of models organized into views that examine a system from a certain viewpoint defined by the concern of a stakeholder
  
    – for understanding, analysis, communication, construction, documentation
  
    – ….for answering questions

• Views include structure and behavior (scenarios)
Example: deployment view

- Addresses concerns of **realization**, **performance** (throughput, latency), **availability**, **reliability**, etc., together with the process view.

- Models in the deployment view describe
  - Machines (processors, memories), networks, organization of interconnect at relevant levels of detail
    - including specifications, e.g. speeds, sizes
      - this part is also called: **physical view**
  - Mapping of components and functionality to machines
  - Flow of control and data (also relevant for **process view**)

(From: *Towards Horizontal Architecture for Autonomic M2M Service Networks*, Future Internet 2014, 6(2), 261-301)
Which views and models will we look at?

- Logical layering, relevant for all (technical) views
  - to give an overall impression, at different layers of abstraction
- Physical view
  - examining organizational alternatives, related to both functional and extra-functional aspects
- Development view
  - looking at devices: how is the software (and hardware) organized, how does this support the operational requirements deriving from use cases
- Process view
  - how is the operation and interoperation: protocols and architectural patterns; active entities in the system, flow of control
- Data view
  - concerns about data semantics, data protection, data flow and data processing
Deployment views for IoT

- Show devices and functionality (components) that goes to the devices

- Indicate flows between the components

- Which functionalities are relevant for IoT?

- Which devices or device types and other physical elements are there?
Physical elements: devices and networks

- ‘Things’: low capacity devices
  - (T-S) sensors
  - (T-A) actuators
  - (T-I) identifier (special sensor)

- Infrastructure:
  - (I-S) switches (layer 2 connectivity within a network technology)
  - (I-G) gateways
    - converting between two parties
    - different layers of the OSI stack
  - networks, e.g. (wireless) LANs, PANs

- (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

<table>
<thead>
<tr>
<th>Functional</th>
<th>Extra functional</th>
<th>Boundary conditions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sensing (event and state)</td>
<td>Dependability - reliable, available</td>
<td>Distributed systems</td>
</tr>
<tr>
<td>Actuation (event)</td>
<td>secure, private</td>
<td>Given components</td>
</tr>
<tr>
<td>Application logic (incl. control)</td>
<td>safe</td>
<td>Given protocols</td>
</tr>
<tr>
<td>Communication / translation</td>
<td>Performance, QoS - response time, latency, throughput</td>
<td>Network standards</td>
</tr>
<tr>
<td>Storage</td>
<td>processing</td>
<td>Legal matters</td>
</tr>
<tr>
<td>Data, Information (context, semantics, location, identity)</td>
<td>timeliness</td>
<td>(Design) Technology - languages, tools</td>
</tr>
<tr>
<td><strong>Vertical Analytics</strong></td>
<td>(Resource) management - program, update, extend</td>
<td>… all that is given</td>
</tr>
<tr>
<td><strong>Horizontal Analytics</strong></td>
<td>- sharing, concurrent applications, scheduling</td>
<td></td>
</tr>
<tr>
<td>Management (of application, of data), UI</td>
<td>Interoperability</td>
<td></td>
</tr>
<tr>
<td>(APIs for) services, advertisement, discovery</td>
<td>Mobility</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Managerial domains, ownership</td>
<td></td>
</tr>
</tbody>
</table>
Analytics

- Vertical analytics
  - data from a single unit
    - e.g. person, item, household, office
  - analysis results in knowledge about that single unit

- Horizontal analytics
  - data from many units
  - analysis results in knowledge about a population of units
  - can characterize classes of units, reference models, averages
  - average temperature chart won’t say much
IoT Communication Model

- **Request-response**: client send requests to the server and the server responds to the requests

- **Publish-subscribe**: when the broker receives data for a topic from publisher, it sends the data to all the subscribed consumers
  - Publishers: source of data
  - Brokers: manage topics
  - Consumers: subscribe to the topics

- **Push-pull**: data producers push the data to queues and all consumers pull the data from the queues
IoT Communication APIs

• **Rest-based Communication APIs:**
  • Representation State Transfer (REST) web APIs focus on a system’s resources and how resource states are addressed and transferred

• **Web-socket based Communication APIs:**
  • Allow bi-directional, full duplex communication between clients and servers
A Thermostat

- Device with integrated functionality
- Older ones are even *network unaware* (not connected to any network)
A distributed variant

• The gateway enables IP connections to sensors / actuators
• The application logic can be just a function in a user device
• Other devices can use the sensors as well
  – e.g. just show temperature
  – this sensor access could also be achieved by virtualizing functions in a connected thermostat
  • i.e., making a single box thermostat as in the previous case but connected

user
(home, office)

internet
provider

core internet
(clouds)
Long term storing for optimization

- Store data for analysis
  - storage and analytics could also be distributed again
- Show temperature profile
- Improve application behavior over time
  - learning patterns
  - using correlations (e.g. using weather info)
- Can also replace bottom with regular (connected) thermostat
- Remote UI, e.g. on phone
  - … but have to be home to see…

user  (home, office)  internet provider  core internet  (clouds)
Cloud storage for horizontal analytics and applications

New applications by combining many users
Data possibly crossing *managerial domains* (UI: contact C or U/F)

user (home, office) internet provider core internet (clouds)
Everything in the Cloud

User can examine her data everywhere
User has no (direct) control, neither data nor application

user (home, office)
internet provider
core internet (clouds)
Many alternatives

• More variations possible
  – extending the low capacity network across managerial domains
    • e.g. some smart meters create neighborhood meshes
• Note the use of ‘fog’: smart infrastructure devices that host applications and storage for aggregation
  – e.g. some Cisco routers
  – “fog computing”, “edge computing”
• Almost everything in one device
  – the smartphone

• Most important: architecture/deployment decisions and views are obtained from scenarios
  – previous examples: suggested functional scenarios
  – other scenarios: adding devices, install applications, establish associations
What are the options and alternatives?

- *Integration, distribution and virtualization*
  - *integration*: put functions together on one device
  - *distribution*: put functions on different devices
  - *virtualization*: decoupling of logical and physical representation (full abstraction of implementation details). Examples:
    - addressing an individual sensor in a device as if it were a separate device
    - let a gateway be just a software function in a larger server
    - address a distributed service as one
What are the options and alternatives?

• **Communication**
  – *indirect*: use a broker, store or proxy between communicating partners
    • **Broker**: a component that handles and translates calls (messages) between two or more parties, and that manages the binding between references and objects
    • **Proxy**: a component that acts on behalf of another component, implementing the same interface, and sometimes caching
  – *push or pull*
    • **Push**: control flow and data flow go in same direction (‘call-back’, ‘event driven’)
    • **Pull**: control flow and data flow go opposite (‘call’, ’polling’)


What are the options and alternatives?

• **Data: storage and handling (of data in flight and in rest)**
  – storage location: local or cloud: (not) leaving *managerial* domain
  – aggregation level (from raw data to a high level function), retention / history

• **Application & control logic:**
  – application *location*: local (same managerial domain); remote ("in the cloud")
  – centralized or distributed
What are guiding tactics in choices?

• *Choices in integration, distribution depend on:*
  – location, form factor, numbers, energy, cost
    • put sensor where it needs to be; may need to be battery operated
    • cannot afford many powerful devices
    • cannot power many devices
  – system complexity
    • easier to use the accelerometer (and other sensors) in your phone than to attach a bluetooth connected one
  – component availability
    • network technology gateways, sensor types, sensor boxes
  – software / framework availability

• *Virtualization improves:*
  – simplicity, generalization of components
    • e.g. making an internal sensor available as an IP end point
    • lights in ‘Markthal’ as IP endpoints admitting app development
  – flexibility, cost
What are guiding tactics in choices?

• **Direct or indirect communication:**
  – indirect: reduces dependencies between partners:
    • *dependency in time:* the need to be ‘on’ at the same time
      – admits intermittent connectivity
    • *and space:* the need to share any physical space resource directly
      – no need to share memory or a network connection
  – direct: reduces latency

• **Push or Pull communication:**
  – **Push:** reduces latency; needs administration at sender in order to know receiver; needs receiver to be on
  – **Pull:** obtain data when required; increases latency because of the extra request; needs sender to be on; sender does not need to know receiver; one extra interaction
  – tradeoff: implementing low latency with pull mode leads to excessive communication (polling) and bad scalability
What are guiding tactics in choices?

• *Data storage and handling choices, determined by:*
  – privacy concerns (privacy is reduced by storage)
  – the need for collecting evidence, post mortem analysis
  – cost concerns in business case: by giving away your data the service may come for free
  – the value of data: application needs, innovation
    • combining data from long time and many sources improves the service

• *Application and control logic choices:*
  – location of application logic: privacy concerns; overall business case
  – location of control logic: latency requirements
  – centralized/distributed: used framework, performance, complexity
Privacy, Safety, and Security

- **Privacy**: control over personal information
  - in particular: use information only within the intended context

- **Safety**: freedom from danger or risk on injury resulting from recognized but potentially hazardous events

- **Security**: regulating access to (electronic) assets according to some policy
  - *policy*: allowed and disallowed actions
  - *security mechanisms*: can be regarded as enforcing the policy

- Asset protection, privacy and safety result in particular *security policies*
  - *security for privacy and security for safety*
Summarizing

<table>
<thead>
<tr>
<th>flexibility</th>
<th>structural complexity</th>
<th>scalability</th>
<th>cost reduction</th>
<th>reduce dependency in space</th>
<th>reduce dependency in time</th>
<th>privacy</th>
<th>latency reduction</th>
<th>generate evidence</th>
</tr>
</thead>
<tbody>
<tr>
<td>virtualization</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>direct communication</td>
<td>0</td>
<td>0/-</td>
<td>-</td>
<td>0</td>
<td>-</td>
<td>-</td>
<td>0</td>
<td>+</td>
</tr>
<tr>
<td>indirect communication</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>0</td>
<td>+</td>
<td>+</td>
<td>0</td>
<td>-</td>
</tr>
<tr>
<td>push mode</td>
<td>0</td>
<td>+</td>
<td>+</td>
<td>0</td>
<td>0</td>
<td>R-S+</td>
<td>0</td>
<td>+</td>
</tr>
<tr>
<td>pull mode</td>
<td>0</td>
<td>0</td>
<td>+/-</td>
<td>0</td>
<td>0</td>
<td>R+S-</td>
<td>0</td>
<td>-</td>
</tr>
<tr>
<td>long term storage</td>
<td>+</td>
<td>0</td>
<td>-</td>
<td>-</td>
<td>+</td>
<td>+</td>
<td>-</td>
<td>0</td>
</tr>
<tr>
<td>crossing managerial domains</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>local application</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>cloud application</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>0</td>
</tr>
<tr>
<td>centralized operation</td>
<td>+</td>
<td>+</td>
<td>-</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>-</td>
</tr>
<tr>
<td>distributed operation</td>
<td>+</td>
<td>-</td>
<td>+</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>+</td>
<td>+</td>
</tr>
</tbody>
</table>

- +: improves
- -: makes worse
- 0: no effect

- centralized operation is bad for scalability, in general
  - distributed implementations help
- R+S-: Receiver+Sender- denotes one-sided dependency
- Note that cloud operation enables large innovations
Coming back to the architecture

- **model**: We saw general concepts occurring in IoT systems (e.g. device types, functionality), communication interactions (data and control flow) and typical scenarios (life cycles)

- **architecture**: We discussed a deployment view of an example, still abstracting from the implementation details
Reference architecture

- IoT is a generic term, referring to many possible instantiations

- The reference model defines generically what binds them together. It consists of
  - domain model: relevant concepts and relations in IoT
  - information model: ‘data structures’ of the domain model elements
  - functional model: (generic) operations
  - communication model: communication interactions between entities

- The reference architecture is an architecture description that takes this model as starting point.
  - it consists of a multi-view modeling of stakeholder concerns but at a generic level
  - it supports exploration of design decisions

- Both need instantiation in a concrete case

From M2M to the IoT, J.Holler et al., Academic Press 2014
Two generic layered views on IoT (1)

from whitepaper of CISCO:
Two generic layered views on IoT (2)

From M2M to the IoT, J.Holler et al., Academic Press 2014, based on IoT-A
Reference architecture

- Many organizations provide *partial* material (see previous layered views)

- European project IoT-a delivered a proposal, and a book
  - currently not followed up it seems

- European project OpenAIS provided a reference architecture for intelligent lighting
IoT (partial) domain model and instantiation

IoT domain model, from M2M to the IoT, J.Holler et al., Academic Press 2014, chapter 7
IoT domain model and instantiation

IoT domain model, from M2M to the IoT, J.Holler et al., Academic Press 2014, chapter 7
Concerns and aims for an IoT reference architecture

- Re-use (IoT) resources across application domains [create a platform]

- A set of support services that provide open service-oriented capabilities and can be used for application development and execution [platform services]

- Different abstraction levels that hide underlying complexities and heterogeneity

- Sensing and actuating can assume roles in different applications running over different managerial domains

- Trust, security, privacy, safety / scalability, performance / evolution, heterogeneous
- Include different service delivery models
- Simple integration, simple management
- Life cycle support: devices, applications and all components