Internet of Things
2017/2018

The Things

Johan Lukkien

John Carpenter, 1982
Guiding questions

• What to think about things and how are they connected?

• What is the difference between IoT, WSN, M2M?

• What drives the development?

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 Gardas’evic et al.
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, 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
Resource limitations

- **Memory**: available flash (‘program code’) and ram
- **Processor**: Mhz, instruction set expressive power, address width, ability to manage its power
- **Energy**: available Joules and how they are replenished
- **Communication**: required transceive power, bps, complexity of protocols

- These are connected, mainly through energy
  - Ram requires power to retain state
  - Processor complexity and Mhz require energy
  - Small memory needs fewer address bits
  - Simpler network protocols and smaller bandwidths lead to lower power transceivers
RFC 7228: devices

• Three classes representing memory (hence processor) limitations

• **C0**: dependent on proxies for secure Internet inclusion
• **C1**: only low resource protocols
• **C2**: can run most Internet protocols
• (**C9**: phone, tablet, desktop)

<table>
<thead>
<tr>
<th>Name</th>
<th>data size (e.g., RAM)</th>
<th>code size (e.g., Flash)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Class 0, C0</td>
<td>&lt;&lt; 10 KiB</td>
<td>&lt;&lt; 100 KiB</td>
</tr>
<tr>
<td>Class 1, C1</td>
<td>~ 10 KiB</td>
<td>~ 100 KiB</td>
</tr>
<tr>
<td>Class 2, C2</td>
<td>~ 50 KiB</td>
<td>~ 250 KiB</td>
</tr>
</tbody>
</table>
Energy limitation and communication policies

<table>
<thead>
<tr>
<th>Name</th>
<th>Type of energy limitation</th>
<th>Example Power Source</th>
</tr>
</thead>
<tbody>
<tr>
<td>E0</td>
<td>Event energy-limited</td>
<td>Event-based harvesting</td>
</tr>
<tr>
<td>E1</td>
<td>Period energy-limited</td>
<td>Battery that is periodically recharged or replaced</td>
</tr>
<tr>
<td>E2</td>
<td>Lifetime energy-limited</td>
<td>Non-replaceable primary battery</td>
</tr>
<tr>
<td>E9</td>
<td>No direct quantitative limitations to available energy</td>
<td>Mains-powered</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Name</th>
<th>Strategy</th>
<th>Ability to communicate</th>
</tr>
</thead>
<tbody>
<tr>
<td>P0</td>
<td>Normally-off</td>
<td>Reattach when required</td>
</tr>
<tr>
<td>P1</td>
<td>Low-power</td>
<td>Appears connected, perhaps with high latency</td>
</tr>
<tr>
<td>P9</td>
<td>Always-on</td>
<td>Always connected</td>
</tr>
</tbody>
</table>

Table 4: Strategies of Using Power for Communication
## Some private taxonomy

<table>
<thead>
<tr>
<th></th>
<th>Flash</th>
<th>RAM</th>
<th>Address space</th>
<th>Processor (type)</th>
<th>OS</th>
<th>Energy</th>
<th>Operation</th>
<th>Actively reachable</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>small code memory</td>
<td>several bytes</td>
<td>&lt;= 8bits</td>
<td>~100Hz</td>
<td>no</td>
<td>External, or battery + wakeup</td>
<td>Externally activated, simple read/write</td>
<td>not designed for reachability via multi-hop</td>
<td>RFID tag, ISO 18000-6c</td>
</tr>
<tr>
<td>B</td>
<td>&lt;= 32K Few hundreds</td>
<td>&lt;=16 bits</td>
<td>~1Mhz TMS430</td>
<td>no, or simple executive</td>
<td>mechanical</td>
<td>mechanically activated, just generates some data</td>
<td>no; needs proxy</td>
<td>power switch</td>
<td></td>
</tr>
<tr>
<td>C</td>
<td>&lt;=32K Few hundreds</td>
<td>&lt;=16 bits</td>
<td>~1Mhz TMS430</td>
<td>Contiki, TinyOS</td>
<td>battery</td>
<td>simple, fixed external behavior, needs proxy, simple sensing</td>
<td>duty cycled, needs proxy</td>
<td>simple sensor mote</td>
<td></td>
</tr>
<tr>
<td>D</td>
<td>&lt;=32K ~10K</td>
<td>&lt;=16 bits</td>
<td>~1Mhz TMS430</td>
<td>Contiki, TinyOS</td>
<td>battery + recharge</td>
<td>capable of managing most constrained IP protocols, sensing, actuating, processing</td>
<td>self-managed on/off behavior</td>
<td>Crossbow</td>
<td></td>
</tr>
<tr>
<td>E</td>
<td>&lt;=256K ~32K</td>
<td>&lt;=32 bits</td>
<td>~1-10Mhz ARM</td>
<td>Contiki, TinyOS</td>
<td>battery + recharge, mains</td>
<td>complete IP endpoint behavior, limited storage</td>
<td>yes</td>
<td>Jennic mote</td>
<td></td>
</tr>
<tr>
<td>F</td>
<td>~GB ~500Mb</td>
<td>32 bits</td>
<td>~Ghz ARM</td>
<td>Linux</td>
<td>battery + recharge, mains</td>
<td>full fledged embedded computer system</td>
<td>yes</td>
<td>Raspberry PI</td>
<td></td>
</tr>
<tr>
<td>G</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>phones, laptops, servers</td>
</tr>
</tbody>
</table>

Johan J. Lukkien, j.j.lukkien@tue.nl
TU/e Informatica, System Architecture and Networking
# Some private taxonomy

<table>
<thead>
<tr>
<th>Category</th>
<th>Flash Memory</th>
<th>RAM Memory</th>
<th>Address Space</th>
<th>Processor (type)</th>
<th>OS</th>
<th>Energy</th>
<th>Operation</th>
<th>Actively reachable</th>
<th>Category</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>small code memory</td>
<td>several bytes</td>
<td>&lt;= 8bits</td>
<td>~100Hz</td>
<td>no</td>
<td>External, or battery + wakeup</td>
<td>Externally activated, simple read/write</td>
<td>not designed for reachability via multi-hop</td>
<td>C0,E0,P0</td>
</tr>
<tr>
<td>B</td>
<td>&lt;= 32K</td>
<td>Few hundreds</td>
<td>&lt;=16 bits</td>
<td>~1Mhz TMS430</td>
<td>no, or simple executive</td>
<td>mechanical</td>
<td>mechanically activated, just generates some data</td>
<td>no; needs proxy</td>
<td>C1,E0,P0</td>
</tr>
<tr>
<td>C</td>
<td>&lt;=32K</td>
<td>Few hundreds</td>
<td>&lt;=16 bits</td>
<td>~1Mhz TMS430</td>
<td>Contiki, TinyOS</td>
<td>battery</td>
<td>simple, fixed external behavior, needs proxy, simple sensing</td>
<td>duty cycled, needs proxy</td>
<td>C1, E2,P1</td>
</tr>
<tr>
<td>D</td>
<td>&lt;=32K</td>
<td>~10K</td>
<td>&lt;=16 bits</td>
<td>~1Mhz TMS430</td>
<td>Contiki, TinyOS</td>
<td>battery + recharge</td>
<td>capable of managing most constrained IP protocols, sensing, actuating, processing</td>
<td>self-managed on/off behavior</td>
<td>C1, E1,P1</td>
</tr>
<tr>
<td>E</td>
<td>&lt;=256K</td>
<td>~32K</td>
<td>&lt;=32 bits</td>
<td>~1-10Mhz ARM</td>
<td>Contiki, TinyOS</td>
<td>battery + recharge, mains</td>
<td>complete IP endpoint behavior, limited storage</td>
<td>yes</td>
<td>C2,E1/9,P1/9</td>
</tr>
<tr>
<td>F</td>
<td>~GB</td>
<td>~500Mb</td>
<td>32 bits</td>
<td>~Ghz ARM</td>
<td>Linux</td>
<td>battery + recharge, mains</td>
<td>full fledged embedded computer system</td>
<td>phones, laptops, servers</td>
<td>C9,E9,P9</td>
</tr>
<tr>
<td>G</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Example: A battery-less light switch

- The switch is pressed.

- The node *turns on* and sends a Route Request broadcast message for a known destination.
  - it boots an OS in the process!

- Using Route Reply, it finds the route to the luminaries.

- Using the discovered route, the node transmits the control signal (turn on/off) to the luminaries.

- The luminary node acknowledges the reception of the control signal.

- The switch node does multiple retries to transmit that control signal as long as the node stays on and until an ACK is received.

30mA @ 3.3V for 60ms

Running FreeRTOS and capable of transmitting compressed IP packets (6LoWPAN).

From: *6LoWPAN: IPv6 for Battery-less Building Networks*, MSc thesis of N.A. Abbasi, TU/e
Functionality of things

• ‘Things’ must be capable to perform the required sensing, actuation, computation, communication
  – functional requirements

• In addition, because they are many:
  – (secure) bootstrap, (secure) network association
    • upon (re)starting a device must load its code from a trusted source
    • it must join the correct network
  – secure communication
  – (secure) software update, over the network
    • updates are inevitable and must remain safe
  – … part of the life cycle

Johan J. Lukkien, j.j.lukkien@tue.nl
TU/e Informatica, System Architecture and Networking

19-Nov-17
Concerns and management

- can it join a network?
  - secure bootstrapping
- can it be configured?
  - adapting operational parameters
    - e.g. sensing, communication frequency
- can it be updated (over the air)?
  - new firmware, new services, new application components
- can it run IP?
  - serve as IP endpoint
- can it secure itself?
  - independent node

• A, B, C: need trusted partner (proxy)
• A, B: very little; C: limited

• From D onwards
  • From E onwards; D runs limited protocols
  • A, B, C: via trusted partner or specialized protocols for single interactions; D: limited
## Some private taxonomy

<table>
<thead>
<tr>
<th>Flash Memory</th>
<th>RAM</th>
<th>Address Space</th>
<th>Processor (type)</th>
<th>OS</th>
<th>Energy</th>
<th>Operation</th>
<th>Actively reachable</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>small code memory</td>
<td>several bytes</td>
<td>&lt;= 8bits</td>
<td>~100Hz</td>
<td>no</td>
<td>External, or battery + wakeup</td>
<td>Externally activated, simple read/write</td>
<td>not designed for reachability via multi-hop</td>
<td>RFID tag, ISO 18000-6c</td>
</tr>
<tr>
<td>&lt;= 32K</td>
<td>Few hundreds</td>
<td>&lt;=16 bits</td>
<td>~1Mhz TMS430</td>
<td>no, or simple executive</td>
<td>mechanical</td>
<td>mechanically activated, just generates some data</td>
<td>no; needs proxy</td>
<td>power switch</td>
</tr>
<tr>
<td>&lt;=32K</td>
<td>Few hundreds</td>
<td>&lt;=16 bits</td>
<td>~1Mhz TMS430</td>
<td>Contiki, TinyOS</td>
<td>battery</td>
<td>simple, fixed external behavior, needs proxy, simple sensing</td>
<td>duty cycled, needs proxy</td>
<td>simple sensor mote</td>
</tr>
<tr>
<td>&lt;=32K</td>
<td>&lt;10K</td>
<td>&lt;=16 bits</td>
<td>~1Mhz TMS430</td>
<td>Contiki, TinyOS</td>
<td>battery + recharge</td>
<td>capable of managing most constrained IP protocols, sensing, actuating, processing</td>
<td>self-managed on/off behavior</td>
<td>Crossbow</td>
</tr>
<tr>
<td>&lt;=256K</td>
<td>~32K</td>
<td>&lt;=32 bits</td>
<td>~1-10Mhz ARM</td>
<td>Contiki, TinyOS</td>
<td>battery + recharge, mains</td>
<td>complete IP endpoint behavior, limited storage</td>
<td>yes</td>
<td>Jennic mote</td>
</tr>
<tr>
<td>~GB</td>
<td>~500Mb</td>
<td>32 bits</td>
<td>~Ghz ARM</td>
<td>Linux</td>
<td>battery + recharge, mains</td>
<td>full fledged embedded computer system</td>
<td>yes</td>
<td>Rasberry PI</td>
</tr>
</tbody>
</table>

Johan J. Lukkien, j.j.lukkien@tue.nl
TU/e Informatica, System Architecture and Networking
What’s new with IoT?

• There are many things
  – #things / person >> 1 (50B in 2020)
  – hence, things need to talk to each other or to a database
    • about …?  
  – self-* properties, autonomy
    • self management, self healing, …
  – scalability, at access networks
    • many things sharing your wireless LAN
    • special infra structure outdoor

• Things have limitations
  – low processing power, memory, low capacity network
    • size IP packet comparable to available memory
  – sometimes battery operated
  – embedded: no UI

• Their numbers and far-reaching locations enable entirely new applications
  – large-scale data collection
  – data-based applications
  – manufacturers probing into the deployed systems

• Their scale and locations comes with complex concerns
  – device/data handling, ownership
  – security, safety, privacy, application reliability
    • at a compelling scale
  – application development, deployment, management
But we had already WSN?

**IoT**
- **System**
  - is platform: concurrent applications at endpoints
  - open, extensible, interoperable
- **Protocol**
  - IP (+ higher) to endpoints (..)
  - … on top of low resource networks
- **Applications**
  - use standard IP protocols
  - developed separately
- **Management**
  - IP management protocols
  - explicit, requires interfaces

**WSN**
- **System**
  - … is the application
- **Protocol**
  - application oriented
  - cross-layer optimization
- **Applications**
  - developed and optimized along with the entire system
- **Management**
  - implicit, part of the application
And M2M?

IoT
- System
  - is platform: concurrent applications at endpoints
  - open, extensible
- Protocol
  - IP (+ higher) to endpoints
  - ... on top of low resource networks
- Applications
  - use standard IP protocols
  - developed separately
- Management
  - IP management protocols
  - explicit, requires interfaces

M2M
- System
  - ... is the application
  - application-specific devices
  - closed
- Protocol
  - standardized, for low-resource networks
- Applications
  - classes
  - developed and optimized along with the entire system
- Management
  - explicit, built into protocols
Example: activity monitoring for stress analysis

Example and pictures from: From M2M to the IoT, J.Holler et al., Academic Press 2014

**IoT**
- Install devices with data generation services
- Combine all sources of information on the subject, shedding light on the entire situation
  - including stress *causes*

**M2M**
- Install devices and applications for the purpose of the application
  - typically, including the precise flow of control inside the system
- Accordingly, collect data, process, and give a stress level output
Data and information gathering

Data

- sensors, logging:
  - time traces and timed events
  - video sequences
- gathering:
  - first person: directly by or on the observed object
  - third person: observed from outside
- usage
  - long term storing
  - or until information has been retrieved
  - or until actuation

Information

- real-world entities represented as digital objects ("digital twins") identified by a key
  - in terms of attributes (e.g. location, color, structure, temperature, ...) evolving over time
  - evolution through timed events, actions
- relationship between those objects
- combining virtual objects about the same real-world entity
  - different contexts
tomato id-23104

room id-431

occupancy  temperature

table id-71209

virtual world  real world

1st person

1st person

3rd person

Johan J. Lukkien, j.j.lukkien@tue.nl
TU/e Informatica, System Architecture and Networking
Drivers for IoT & M2M

- **Required improvements in work efficiency, in environmental sustainability, in safety, security, health**
  - leads to increasing need to understand the physical environment…
    - hence, generate and analyze contextual data to increase understanding, take better decisions
  - …and to automate processes

- **Advances in ICT as enabler:**
  - in networking (penetration of IP), processing capabilities, storage
  - in established approaches, systems, services
    - e.g. Cloud Computing, Network Function Virtualization, Software Defined Networking
  - in established frameworks, to *commoditize* complex systems
    - the Internet of APIs

- **Cost reductions of components, networks, storage, etc.**
Guiding questions

• What to think about things and how are they connected?

• What is the difference between IoT, WSN, M2M?

• What drives the development?