Internet of Things
2016/2017

Discovering the Things

Johan Lukkien
with slides by Milosh Stolikj

John Carpenter, 1982
Guiding questions

• What does service discovery entail, and what are relevant criteria to compare technologies?
• Which service discovery techniques are relevant for IoT?
Agenda

• Understanding and motivating service discovery
• Overview and comparison of SD protocols on the Internet
• CoAP resource discovery
• mDNS as a basic or bootstrapping method
Example: Smart City Services

• Model:
  – mobile users moving around in the ambient infrastructure of a city

• Smart city services
  – available for use by these (mobile) users
  – beyond information services

Parking spaces in (x,y) region?

Parking space at (x1,y1)

......

Parking space at (xn,yn)
Digital services

• Services represent accessible functionality, possibly with a concept of locality

• Examples for users
  – just information: standard website of shop right there on the corner
  – parking spot location and status, plus the ability to reserve one;
  – access to local lighting actuators;
  – access to local air quality measurements.

• Examples for nodes
  – finding and using functionalities like sensors and actuators
Definition: service

- **Service**: a contractually specified overall functionality (semantics) of an entity.

- **Service quality**: extra-functional properties of a service (e.g. speed, reliability, ...).

- **Service interface (API)**: actions (“primitives”) and responses that make the service available; these responses can be autonomous (“events”, “call-backs”). In addition, a specification that
  - describes their effect on state variables and parameters, as well as their results;
  - describes rules as how and in what sequence to call them;
  - describes the functional and non-functional properties of sequences of calls.

(i.e., the interaction or access protocol)
Service discovery model

- The basic problem: how do two parties, that don’t know each other, meet?

- Three roles:
  - publisher, seeker, (optional) mediator or broker

- Concepts involved
  - service, interface, query, advertisement, seeker, repository
Service discovery process

- Tasks involved:
  - Advertisement of service & query, (propagation), matching, evaluation
Service discovery system architecture

- **Structure**
  - entities, their roles, responsibilities and dependencies
  - deployment view
    - which functionality on which device

- **Protocol**
  - the message exchanges between parties for the different stages of discovery
    - order and format

- **Syntax & semantics**
  - service description
  - service identification (interface specification / typing, access points)
  - queries, responses
  - registrations
- User devices discover parking spots
- Parking spots are autonomous
  - negotiation with users
  - report to a cloud service
Parking

- User devices discover parking server in the local net
- Parking spots are managed by this server

smart city (local network)  
internet access  
core internet (clouds)
User devices contact the cloud server
- either a known address within an app
- or a reference through a local discovery

Parking spots are dumb devices
Agenda

- Understanding and motivating service discovery
- Overview and comparison of SD protocols on the Internet
- CoAP resource discovery
- mDNS as a basic or bootstrapping method
Services in the Internet

• Need to *know* service interface
• Need to *find* the service access point
• Aspects
  – which type of service: this defines the protocol to access it (also called *application protocol*)
  – find which specific service instances are there
  – find on which device a selected service instance is to be accessed
  – which location to access inside this device

  • Example:
     – a Philips type XYZ light control service
       • exposes a SOAP or REST function to set a light level
     – l1.home.local, l2.home.local
     – IP address
     – HTTP/UDP:4309
     – local name space extension
Some examples

• Google!
  – arbitrary query maps to set of access points
  – focus on HTTP as access protocol
  – global indexing – may not be the best way
    • overkill: energy
    • accuracy (in time)
    • maintain locality: security and privacy
    • dependability and scalability

• SLP – service location protocol (IETF)
  – service discovery in LAN
  – with / without directory
    • query: lpr//((LOCATION==12th FLOOR)/
    • Returned URL: service:lpr://igore.wco.ftp.com:515/draft
      (a service using the lpr protocol, with queue draft at the given host)

actually, the only service discovered on the network is the DA
Service discovery system

- A service advertisement/discovery system is a service by itself
- However, its structure, its protocol, its bindings and its syntax should be known, be part of the infrastructure
  - e.g. DNS

- Requirements, in IoT context:
  - applicable to low and high resource devices
    - low resource: processing power, memory, energy, communication volume and standard, duty cycling
    - limited overhead in all these dimensions
  - location dependence
  - access control to the discovery process (who may enter it)
    - either at service level or at network access level
  - ‘IP compliant’: adopted in the IETF protocol suite
    - history shows, IP-based protocols survive
Comparison of protocols

• Compare existing / proposed protocols on the basis of ….
  – architecture: distributed(each device) / centralized / hierarchical
  – discovery & registration mechanisms: unicast / multicast / broadcast
  – overhead: protocol, (packet) size, processing, bandwidth
  – description: format
  – interoperability: refers to open standards, particularly IP, and shared semantics

• … and some properties
  – Localization: can location be referred to?
  – Mobility: is it dealt with, and how?
  – Proxy-support: can powerful devices act for less powerful?
  – Energy-awareness: is energy addressed as criterion?
  – Low-resource: suitable for low resource devices?
<table>
<thead>
<tr>
<th>Protocol</th>
<th>Architecture</th>
<th>Discovery mechanism</th>
<th>Registration mechanism</th>
<th>Overhead</th>
<th>Description</th>
<th>Interoperability</th>
</tr>
</thead>
<tbody>
<tr>
<td>mDNS/DNS-SD [3]</td>
<td>distributed</td>
<td>multicast</td>
<td>multicast</td>
<td>DNS packets</td>
<td>DNS-SD</td>
<td>yes</td>
</tr>
<tr>
<td>UPnP [12]</td>
<td>hierarchical</td>
<td>multicast or unicast</td>
<td>multicast</td>
<td>heavy transport protocol</td>
<td>URL</td>
<td>yes, gateways</td>
</tr>
<tr>
<td>Jini [11]</td>
<td>centralized</td>
<td>multicast or unicast</td>
<td>multicast or unicast</td>
<td>heavy protocol</td>
<td>Java</td>
<td>yes</td>
</tr>
<tr>
<td>SLP [35]</td>
<td>distributed or centralized</td>
<td>multicast or unicast</td>
<td>multicast or unicast</td>
<td>translation agents</td>
<td>string, XMLS</td>
<td>yes</td>
</tr>
<tr>
<td>SDP [19]</td>
<td>distributed</td>
<td>broadcast</td>
<td>proactive</td>
<td>SDP server discovery (unknown)</td>
<td>UUID</td>
<td>no</td>
</tr>
<tr>
<td>DEAPSpace [20]</td>
<td>distributed</td>
<td>broadcast</td>
<td>proactive</td>
<td>periodic advertisements</td>
<td>none</td>
<td>no</td>
</tr>
<tr>
<td>Konark [27]</td>
<td>distributed</td>
<td>multicast</td>
<td>multicast</td>
<td>periodic advertisements</td>
<td>XML</td>
<td>no</td>
</tr>
<tr>
<td>SANDMAN [21]</td>
<td>hierarchical</td>
<td>unicast</td>
<td>unicast</td>
<td>cluster maintenance, sleep announcements</td>
<td>none</td>
<td>no</td>
</tr>
<tr>
<td>Sleeper [28]</td>
<td>distributed</td>
<td>broadcast</td>
<td>broadcast</td>
<td>proxy election, periodic advertisements</td>
<td>taxonomy</td>
<td>no</td>
</tr>
<tr>
<td>Ozcelebi et al [23]</td>
<td>centralized</td>
<td>unicast</td>
<td>unicast</td>
<td>periodic advertisements (heartbeat)</td>
<td>none</td>
<td>no</td>
</tr>
<tr>
<td>Anwar et al [24]</td>
<td>hierarchical</td>
<td>unicast</td>
<td>unicast</td>
<td>cluster maintenance</td>
<td>ENUM/DNS</td>
<td>DNS</td>
</tr>
<tr>
<td>TRENDY [26]</td>
<td>centralized or hierarchical</td>
<td>multicast</td>
<td>multicast</td>
<td>periodic group updates, leader election</td>
<td>CoRE, URI</td>
<td>no</td>
</tr>
<tr>
<td>Protocol</td>
<td>Localization</td>
<td>Mobility</td>
<td>Proxy</td>
<td>Energy-aware</td>
<td>Low resource</td>
<td></td>
</tr>
<tr>
<td>-------------------</td>
<td>-------------------------------</td>
<td>----------------------------------------</td>
<td>-------</td>
<td>--------------</td>
<td>--------------</td>
<td></td>
</tr>
<tr>
<td>mDNS/DNS-SD [3]</td>
<td>through domain name</td>
<td>time-to-live expiry, service invalidation</td>
<td>yes</td>
<td>no</td>
<td>yes [8] [9]</td>
<td></td>
</tr>
<tr>
<td>UPnP [12]</td>
<td>through domain name</td>
<td>timeout expiry, periodic advertisements</td>
<td>no</td>
<td>no</td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td>Jini [11]</td>
<td>no</td>
<td>timeout expiry</td>
<td>yes</td>
<td>no</td>
<td>no</td>
<td></td>
</tr>
<tr>
<td>SLP [35]</td>
<td>yes [18]</td>
<td>timeout expiry (check)</td>
<td>yes</td>
<td>no</td>
<td>yes [17]</td>
<td></td>
</tr>
<tr>
<td>SDP [19]</td>
<td>yes (one hop)</td>
<td>no</td>
<td>no</td>
<td>yes</td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td>DEAPSpace [20]</td>
<td>yes (one hop)</td>
<td>periodic advertisements</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td>Konark [27]</td>
<td>yes, in description</td>
<td>timeout expiry, periodic advertisements</td>
<td>no</td>
<td>yes</td>
<td>unknown</td>
<td></td>
</tr>
<tr>
<td>SANDMAN [21]</td>
<td>yes, in clusters</td>
<td>moving clusters</td>
<td>cluster heads</td>
<td>yes</td>
<td>unknown</td>
<td></td>
</tr>
<tr>
<td>Sleeper [28]</td>
<td>yes, in description</td>
<td>timeout expiry, service invalidation</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
<td></td>
</tr>
<tr>
<td>OSAS [23]</td>
<td>no</td>
<td>timeout expiry</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td>Anwar et al [24]</td>
<td>yes, in identifier</td>
<td>timeout expiry</td>
<td>yes</td>
<td>no</td>
<td>unknown</td>
<td></td>
</tr>
<tr>
<td>TRENDY [26]</td>
<td>yes, description</td>
<td>periodic advertisements</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td></td>
</tr>
</tbody>
</table>
Agenda

- Understanding and motivating service discovery
- Overview and comparison of SD protocols on the Internet
- CoAP resource discovery
- mDNS as a basic or bootstrapping method
CoAP resource discovery

• CoAP services are really manipulations of resources
  – CoAP has integrated methods for registering and discovering resources
  – not considered a complete SD system, but comes pretty close

• CoAP resource discovery uses the ‘/.well-known/’ path prefix (RFC 5785)
  – intended to obtain host-specific information
    • example: access policies of a host, only required when you get there
  – admits IANA registration
  – application-specific suffix in this name space
    • ‘core’ for CoAP (core: working group)

• Example:
  – GET /.well-known/core
  – Response: set of links referring to resources (in core link format, see later)
CoRE Resource Discovery

From presentation by Z. Shelby

CoAP Client

CON [0xaf6] GET ./well-known/core

ACK [0xaf6] 2.05 Content "<light>..."

CoAP Server

</dev/bat>;obs;if="";rt="ipso:dev-bat";ct="0",
</dev/mdl>;if="";rt="ipso:dev-mdl";ct="0",
</dev/mfg>;if="";rt="ipso:dev-mfg";ct="0",
</pwr/0/rel>;obs;if="";rt="ipso:pwr-rel";ct="0",
</pwr/0/w>;obs;if="";rt="ipso:pwr-w";ct="0",
</sen/temp>;obs;if="";rt="ucum:Cel";ct="0"
Web linking

- RFC 5988: defines a web linking format for web resources
  - the current resource – as returned by an HTTP request – has a relation with another resource as specified by a ‘Link: ‘ field in the HTTP response
  - this is similar to a <link /> element in the HTML code of the requested resource
  - the format specifies a variety of attributes

- Example
  - Link: <http://example.com/TheBook/chapter2>; rel="previous"; title="previous chapter"
  - Meaning: the link given above has relation ‘previous’ with the requested resource, and the string ‘previous chapter’ can be used as a name
    - relations can be extensions (an uri)
    - relations can be registered (via IANA)
CoRE Link Format

- RFC 6690 defines the CoRE Link Format based on the Web Linking format
- The aim is to describe
  - resources and attributes
  - relationships between links
- A link therefore is a resource description

```
Link = link-value-list
link-value-list = [ link-value *[ "," link-value ]]
link-value = "<" URI-Reference "">" * ( "," link-param )
link-param = ( ( "rel" ="" relation-types )
  / ( "anchor" ="" DQUOTE URI-Reference DQUOTE )
  / ( "rev" ="" relation-types )
  / ( "hreflang" ="" Language-Tag )
  / ( "media" ="" ( MediaDesc
    / ( DQUOTE MediaDesc DQUOTE ) ) )
  / ( "title" ="" quoted-string )
  / ( "title**" ="" ext-value )
  / ( "type" ="" ( media-type / quoted-nt ) )
  / ( "rt" ="" relation-types )
  / ( "if" ="" relation-types )
  / ( "sz" ="" cardinal )
  / ( link-extension )
```

Partial syntax, see RFC 6690
Attributes

Name

- rt: resource type
- if: interface
- sz: size
- ct: content type of the resource

Example value

- outdoortemp (opaque)
- core.rd (resource directory)
- sensor (opaque)
- 25 (number)
- 40 (index in known table)

REQ: GET /.well-known/core
RES: 2.05 Content
</sensors/temp>:if="sensor",
</sensors/light>:if="sensor"

REQ: GET /sensors
RES: 2.05 Content
</sensors/temp>:rt="temperature-c":if="sensor",
</sensors/light>:rt="light-lux":if="sensor"
Handing off discovery

- CoAP resource discovery is extended with a concept of resource directory (draft-ietf-core-resource-directory-05, version now: 09)
  - a node (endpoint) can inquire the existence of a resource directory

  | ----- GET /.well-known/core?rt=core.rd* ------> |
  | <---- 2.05 Content "<rd>;rt="core.rd" ------> |

  - a node can register (POST) or remove (DELETE) resources

  | --- POST /rd?ep=node1 "</sensors..." ------> |
  | <---- 2.01 Created Location: /rd/4521 ------> |

- Notice extra attributes, e.g.: ep: endpoint, d: domain (DSN-SD), lt: lifetime
Further usage of the directory

- In the example the endpoint name is already known

- Further support for updating links
  - POST (or PATCH)

- Group definition

- Advanced lookup
Finding CoAP devices

• Inquire at multicast address All-CoAP-Nodes
  – e.g. a GET /.well-known/core from that address

• Use pre-set with a resource directory, e.g.,
  – the border router of a domain
  – included in bootstrap info

• Use another service discovery protocol……
Agenda

• Understanding and motivating service discovery
• Overview and comparison of SD protocols on the Internet
• CoAP resource discovery
• mDNS as a basic or bootstrapping method
Bootstrapping discovery

• There are many accepted service discovery technologies
  – see the tables
  – e.g. UPnP and CoAP for ad-hoc networking domain
    • resp. SSDP and CoAP resource directory
• Need a general means to obtain start points ("closure") regardless of the technology

• mDNS/DNS-SD: Distributed service discovery, designed initially for LANs
  – Transport protocol – mDNS (RFC 6762)
  – Description (syntax) – DNS-SD (RFC 6763)

• Supported on most platforms
  – Bonjour by Apple
  – Avahi by the Linux community
Recap: DNS, resolve internet names

(2010)
root zone:
20 generic top-level domains
248 country-code TLDs

Domain: subtree
Zone: delegated portion of a domain (though terms are often used interchangeably)
# Example database for domain cs.vu.nl

<table>
<thead>
<tr>
<th>Name</th>
<th>Record type</th>
<th>Record value</th>
</tr>
</thead>
</table>
| cs.vu.nl.       | SOA         | star.cs.vu.nl. hostmaster.cs.vu.nl.  
                         |                                                       | 2005092900 7200 3600 2419200 3600 |
| cs.vu.nl.       | TXT         | "Vrije Universiteit - Math. & Comp. Sc."        |
| cs.vu.nl.       | MX          | 1 mail.few.vu.nl.                                |
| cs.vu.nl.       | NS          | ns.vu.nl.                                         |
| cs.vu.nl.       | NS          | top.cs.vu.nl.                                     |
| cs.vu.nl.       | NS          | solo.cs.vu.nl.                                    |
| cs.vu.nl.       | NS          | star.cs.vu.nl.                                    |
| star.cs.vu.nl.  | A           | 130.37.24.6                                       |
| star.cs.vu.nl.  | A           | 192.31.231.42                                     |
| star.cs.vu.nl.  | MX          | 1 star.cs.vu.nl.                                  |
| star.cs.vu.nl.  | MX          | 666 zephyr.cs.vu.nl.                              |
| star.cs.vu.nl.  | HINFO       | "Sun" "Unix"                                      |
| zephyr.cs.vu.nl.| A           | 130.37.20.10                                      |
| zephyr.cs.vu.nl.| MX          | 1 zephyr.cs.vu.nl.                                |
| zephyr.cs.vu.nl.| MX          | 2 tornado.cs.vu.nl.                               |
| zephyr.cs.vu.nl.| HINFO       | "Sun" "Unix"                                      |
## Example (cnt’d)

<table>
<thead>
<tr>
<th>Domain</th>
<th>Type</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>ftp.cs.vu.nl</td>
<td>CNAME</td>
<td>soling.cs.vu.nl.</td>
</tr>
<tr>
<td><a href="http://www.cs.vu.nl">www.cs.vu.nl</a></td>
<td>CNAME</td>
<td>soling.cs.vu.nl.</td>
</tr>
<tr>
<td>soling.cs.vu.nl</td>
<td>A</td>
<td>130.37.20.20</td>
</tr>
<tr>
<td>soling.cs.vu.nl</td>
<td>MX</td>
<td>1 soling.cs.vu.nl.</td>
</tr>
<tr>
<td>soling.cs.vu.nl</td>
<td>MX</td>
<td>666 zephyr.cs.vu.nl.</td>
</tr>
<tr>
<td>soling.cs.vu.nl</td>
<td>HINFO</td>
<td>&quot;Sun&quot; &quot;Unix&quot;</td>
</tr>
<tr>
<td>vucsdas1.cs.vu.nl</td>
<td>PTR</td>
<td>0.198.37.130.in-addr.arpa.</td>
</tr>
<tr>
<td>vucsdas1.cs.vu.nl</td>
<td>A</td>
<td>130.37.198.0</td>
</tr>
<tr>
<td>inkt.cs.vu.nl</td>
<td>HINFO</td>
<td>&quot;OCE&quot; &quot;Proprietary&quot;</td>
</tr>
<tr>
<td>inkt.cs.vu.nl</td>
<td>A</td>
<td>192.168.4.3</td>
</tr>
<tr>
<td>pen.cs.vu.nl</td>
<td>HINFO</td>
<td>&quot;OCE&quot; &quot;Proprietary&quot;</td>
</tr>
<tr>
<td>pen.cs.vu.nl</td>
<td>A</td>
<td>192.168.4.2</td>
</tr>
<tr>
<td>localhost.cs.vu.nl</td>
<td>A</td>
<td>127.0.0.1</td>
</tr>
</tbody>
</table>
mDNS vs DNS

• DNS queries

PC1.local
aaaa::1

PC2.local
aaaa::2

DNS Query: who is PC2.local?

DNS Response: PC2.local is aaaa::2

Unicast

Multicast

Johan J. Lukkien, j.j.lukkien@tue.nl
TU/e Informatica, System Architecture and Networking
Recap: DNS name servers

- Name server serves a certain *zone*
  - authoritative or cached answer
  - authoritative server for a host: always maintains a record of that host
- Root servers: are authoritative for root and top-level domains
- Name servers are configured with addresses of root servers
- Name servers for a domain
  - know their child domains
  - are typically replicated: *primary* (initialized from file) and *secondary* server (synchronizes with primary)
- DNS is made available to applications as a middleware service
  - typically a library (*resolver*) that calls upon a configured name server
  - that server often uses *BIND (Berkeley Internet Name Daemon)*, named
mDNS vs DNS

- mDNS queries

PC1.local aaaa::1

Can also be multicast

mDNS Query: who is PC2.local?

mDNS Response: PC2.local is aaaa::2

PC2.local aaaa::2

mDNS Query: who is PC2.local?

PC3.local aaaa::3

Unicast

Multicast
Recap: DNS resource records in DNS-SD

<table>
<thead>
<tr>
<th>Type of record</th>
<th>Associated entity</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SOA</td>
<td>Zone</td>
<td>Holds information on the represented zone</td>
</tr>
<tr>
<td>A</td>
<td>Host</td>
<td>Contains an IP address of the host this node represents</td>
</tr>
<tr>
<td>MX</td>
<td>Domain</td>
<td>Refers to a mail server to handle mail addressed to this domain</td>
</tr>
<tr>
<td>SRV</td>
<td>Domain</td>
<td>Refers to a server handling a specific service</td>
</tr>
<tr>
<td>NS</td>
<td>Zone</td>
<td>Refers to a name server that implements the represented zone</td>
</tr>
<tr>
<td>CNAME</td>
<td>Node</td>
<td>Symbolic link with the primary name of the represented node</td>
</tr>
<tr>
<td>PTR</td>
<td>Host</td>
<td>Contains the canonical name of a host</td>
</tr>
<tr>
<td>HINFO</td>
<td>Host</td>
<td>Holds information on the host this node represents</td>
</tr>
<tr>
<td>TXT</td>
<td>Any kind</td>
<td>Contains any entity-specific information considered useful</td>
</tr>
</tbody>
</table>

- Used in DSN-SD to express services
  - PTR (std DNS: map IP to DNS name)
  - SRV (std DNS: service description)
  - TXT (std DNS: additional descriptive information)
DNS Based Service Naming

- Services described as “domain names”

\[ \text{<instance>}. \text{<service>}. \text{<domain>} \]

Ex. S1._coap._udp.local

- instance: unique name
- service: is actually type; can further use a subtype of this (_sub)
- domain: scope where resolution is valid. For mDNS: .local

- Re-use of existing DNS records
  - SRV and TXT for service description
    - together service and domain, TXT contains (key,value) pairs
    - response mapped to URI
  - PTR for browsing service types
  - A/AAAA for address resolution
mDNS/DNS-SD Combined (1)

Q: PTR _light._sub._coap._udp.local

PTR: _light._sub._coap._udp.local → l1._coap._udp.local

Q: PTR _light._sub._coap._udp.local

PTR: _light._sub._coap._udp.local → l2._coap._udp.local

PC1.local aaaa::1

sensor1.local aaaa::2

sensor2.local aaaa::3
mDNS/DNS-SD Combined (1)

Q: SRV l1._coap._udp.local

SRV: l1._coap._udp.local → sensor1.local:5678

TXT: l1._coap._udp.local → PATH="/light1"

PC1.local
aaaa::1

Q: SRV l1._coap._udp.local

sensor1.local
aaaa::2

sensor2.local
aaaa::3
mDNS/DNS-SD Combined (1)

Q: AAAA sensor1.local

A/AAAA: sensor1.local -> aaaa::2

PC1.local aaaa::1

Q: AAAA sensor1.local

sensor1.local aaaa::2

sensor2.local aaaa::3
Four problems

1. Many message exchanges
2. Large code base
3. Always-on endpoints
4. Expressiveness of queries

Discussed solutions are *proposals*
mDNS/DNS-SD Combined (2, proposal)

Q: PTR _light._sub._coap._udp.local

SRV: 11._coap._udp.local → sensor1.local:5678
TXT: 11._coap._udp.local → PATH="/light1"
PTR: _light._sub._coap._udp.local → 11._coap._udp.local
A/AAAA: sensor1.local -> aaaa::2

SRV: 12._coap._udp.local → sensor2.local:5678
TXT: 12._coap._udp.local → PATH="/light2"
PTR: _light._sub._coap._udp.local → 12._coap._udp.local
A/AAAA: sensor2.local -> aaaa::3

PC1.local aaaa::1

sensor1.local aaaa::2

sensor2.local aaaa::3
Problem 2: large code base

- This was resolved in our implementation – see table

<table>
<thead>
<tr>
<th>Implementation</th>
<th>Code</th>
<th>Memory</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bonjour by Apple</td>
<td>500KB</td>
<td>/</td>
</tr>
<tr>
<td>Ethernet Bonjour for Arduino</td>
<td>14KB</td>
<td>/</td>
</tr>
<tr>
<td>uBonjour for Contiki [8]</td>
<td>7.69KB</td>
<td>0.4KB</td>
</tr>
<tr>
<td>mDNS/DNS-SD for Contiki²</td>
<td>6.51KB</td>
<td>0.7KB</td>
</tr>
</tbody>
</table>

1 Based on the size mDNSResponder.exe on 64 bit platforms. Memory information is unavailable.
2 Available at https://github.com/mstolikj/contiki
Problem 3: service provider constantly on-line

• Solution: proxy support
  – Before a node sleeps, it hands-off service descriptions to a proxy server
  – The proxy responds to queries on behalf of the sleeping node

• Two approaches in the hand-off
  1. The sleeping node chooses the proxy server (active proxy)
  2. The sleeping node does not care who handles it, it is decided implicitly (passive proxy)
  3. Dynamic DNS is used for registration of resource records
Active proxy

Sleeping node
sensor1.local

Proxy server
proxy.local

Service client
client.local

Q: _sleep-proxy._udp.local
A: SRV, TXT, PTR, AAAA RRs
for the proxy server

DynDNS: register
SRV, TXT, PTR, AAAA, Lease
ACK

Radio off

Q: query for a proxy service
A: advertise the proxy service
use discovered service

Q: sensor1.local
A: AAAA sensor1.local
Passive proxy

Sleeping node
sensor1.local

Radio off

A: advertising all records
• (PROXY, 1) as (name,value) pair to request proxy
A: advertising response to accept to be proxy
• (PROXY,2) as (name, value) pair to confirm
• multicast: everyone knows proxy was established
• dynamic registration (DYNDNS) implicit

Proxy server
proxy.local

A: SRV, TXT, PTR, AAAA RRs
+ TXT+=PROXY=1, DC=1s

A: SRV, TXT, PTR, AAAA RRs
+ TXT+=PROXY=2

Service client
client.local

Q: sensor1.local

A: AAAA sensor1.local

Johan J. Lukkien, j.j.lukkien@tue.nl
TU/e Informatica, System Architecture and Networking
Problem 4: expressiveness of queries

- DNS-SD design:
  - discover all service instances in a logical domain
    - typically, a broadcast domain, or a DNS scope
  - assumes selection at query issuer (TXT records), or equivalence of services that respond to the same query

- Scalability concern: large number of responses when there are many, identical small services
  - reduce responses by more accurate specification of query
  - interpret TXT records
Proposal

• Context tags in PTR records

• Use boolean operators to specify queries as predicates
  – logical and, or, not

• Remain compatible in syntax
  – and: ‘*’, or: ‘.’, not: ‘-’

• Concerns:
  – reduce size: specify only the value part of (key,value) pairs
  – reduce processing: limit query complexity, in practical uses
Example

- Just one query
- Containing a precise predicate

### Property

<table>
<thead>
<tr>
<th></th>
<th>Light switch 1</th>
<th>Light switch 2</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Service type</strong></td>
<td>_light._sub._coap._udp.&lt;domain&gt;</td>
<td></td>
</tr>
<tr>
<td><strong>Location</strong></td>
<td>MetaForum, Floor 3, Office 1</td>
<td></td>
</tr>
<tr>
<td><strong>Color</strong></td>
<td>Blue</td>
<td>Red</td>
</tr>
<tr>
<td><strong>Resource path</strong></td>
<td>/light/switch1</td>
<td>/light/switch2</td>
</tr>
</tbody>
</table>
Guiding questions

• What does service discovery entail, and what are relevant criteria to compare technologies?
• Which service discovery techniques are relevant for IoT?