JACOBS UNIVERSITY

Service Discovery in Resource Constrained Networks using Multicast DNS

Aleksandar Siljanovski
a.siljanovski@jacobs-university.de

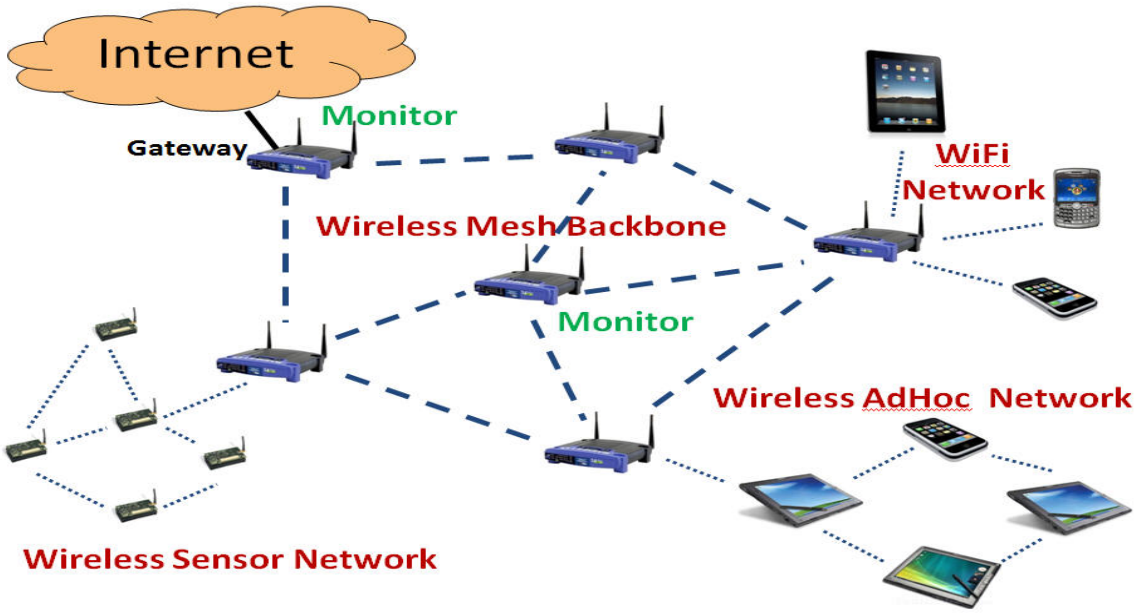
Anuj Sehgal
s.anuj@jacobs-university.de

Jürgen Schönwälder
j.schoenwaelder@jacobs-university.de

Outline

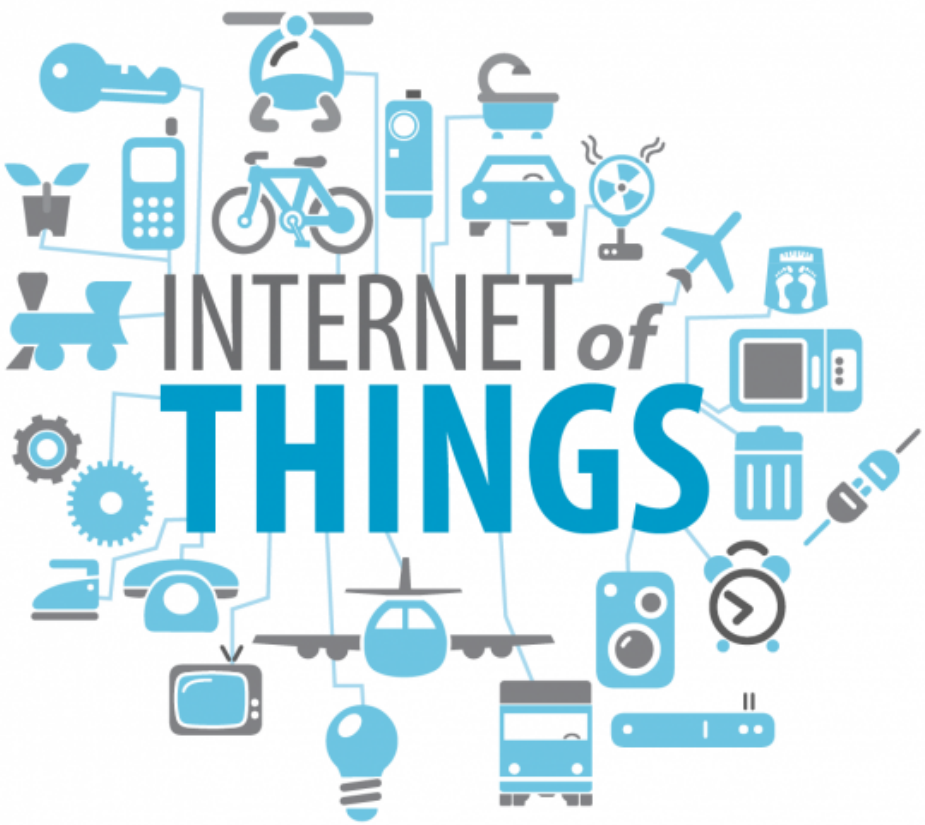
- Introduction
  - Background & motivation
  - Protocols for the IoT
- Our approach
  - mDNS / DNS-SD
  - Implementation
- Testing & Evaluation
- Conclusions

Resource Constrained Networks



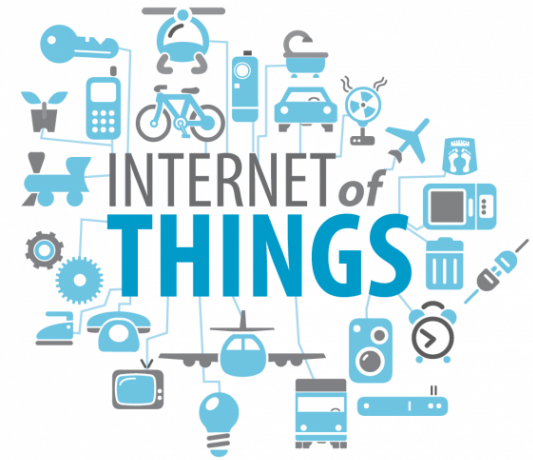
- **New(ish) class of networks**
  - Multiple class of devices
  - Most devices with *extremely* low computing resources (1-8 kB RAM/ROM, 2-16 MHz)
- **Low-power lossy channels**
  - 802.15.4, DECT, Bluetooth…
  - Small frame sizes – need for adaptation layers

Internet of Things



- **Large-scale deployment of objects**
  - Wireless & embedded sensors
  - RFID
  - Actuators
- **Interaction and cooperation amongst objects**
- **Varied application domains**
  - Logistics, transport, vehicles
  - Smart metering, AMI
  - Health, environment
- **End-to-end IP connectivity**

Internet of Things



Billions of Devices!
How do we manage them?
How to find them?

Varied Capabilities!
Describing services?
Discovering services?

- **Large-scale deployment of objects**
  - Wireless & embedded sensors
  - RFID
  - Actuators
- **Interaction and cooperation amongst objects**
- **Varied application domains**
  - Logistics, transport, vehicles
  - Smart metering, AMI
  - Health, environment
- **End-to-end IP connectivity**

Protocols for the Internet of Things

- **Foundations**
  - 6LoWPAN
  - RPL
- **Management**
  - SNMP
  - NetConf
  - RESTConf
- **Security**
  - DTLS
- **Application Layer**
  - CoAP

Wait a minute, no one said how to discover the devices or services!

- IETF CoRE working group decides to tackle service description problem
- Meantime, OMA creates LwM2M

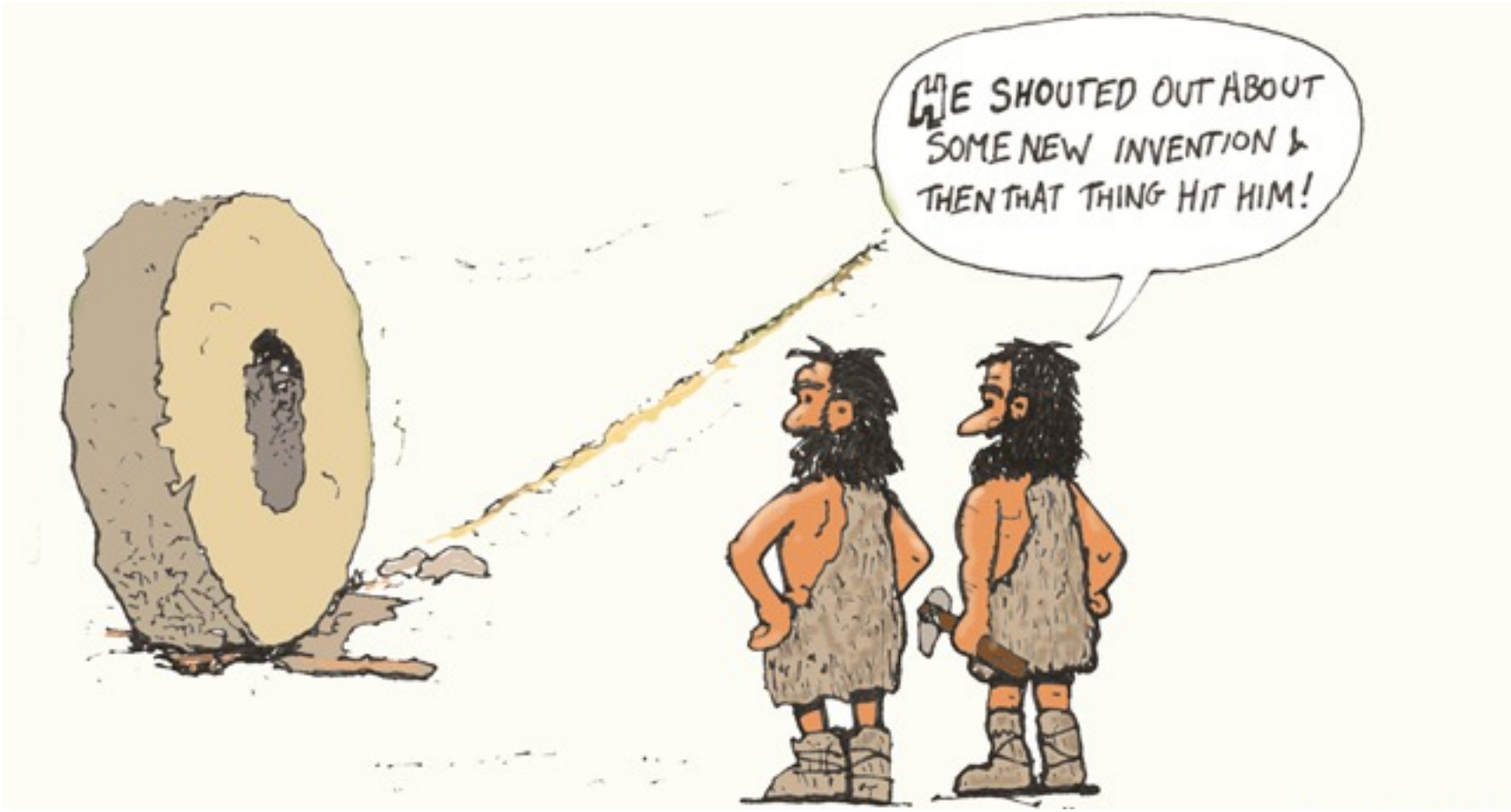
CoRE – CoAP

- Capabilities of each device described at well known URI

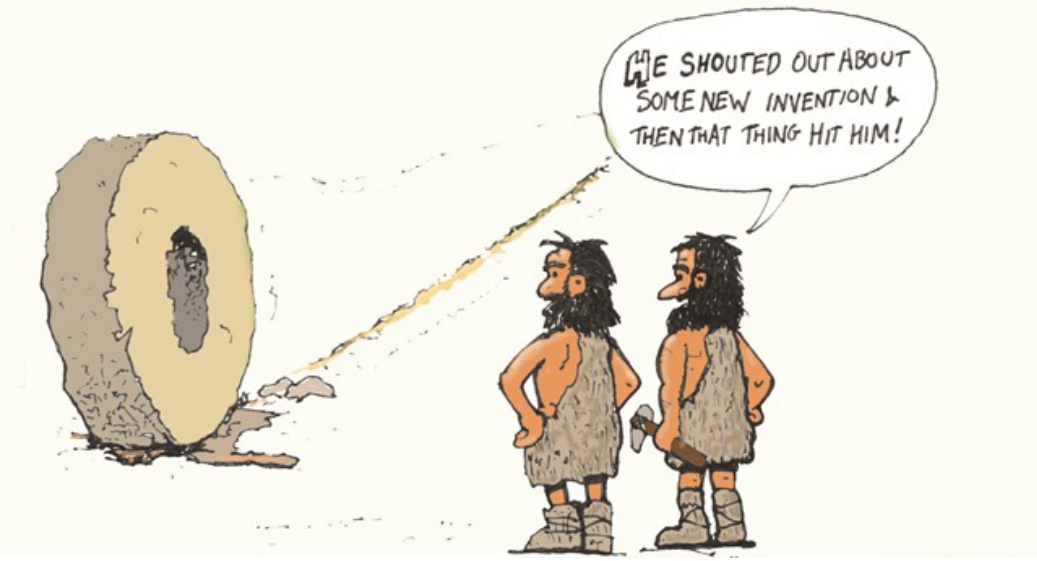
OMA LwM2M

- Adopt CoAP to manage embedded devices
- Create a registry of all devices

Are we wasting effort?



Are we wasting effort?



But we have end-to-end IP – why not reuse popular IP based methods?

A central registry – lots of state information!

DNS Service Discovery & Multicast DNS (Bonjour)

DNS Service Discovery (DNS-SD)

1. **Register, browse and resolve service names to DNS host names.**
2. **Every host (or DNS server) maintains its own registry.**
3. **The SRV record is used.**
   <Instance>.<Service>.<Domain>
4. **TXT record contains service specific information.**
5. **PTR queries for <Service>.Domain used to discover services.**

Search for HTTP servers at a domain

PTR query for *_http._tcp.google.com*

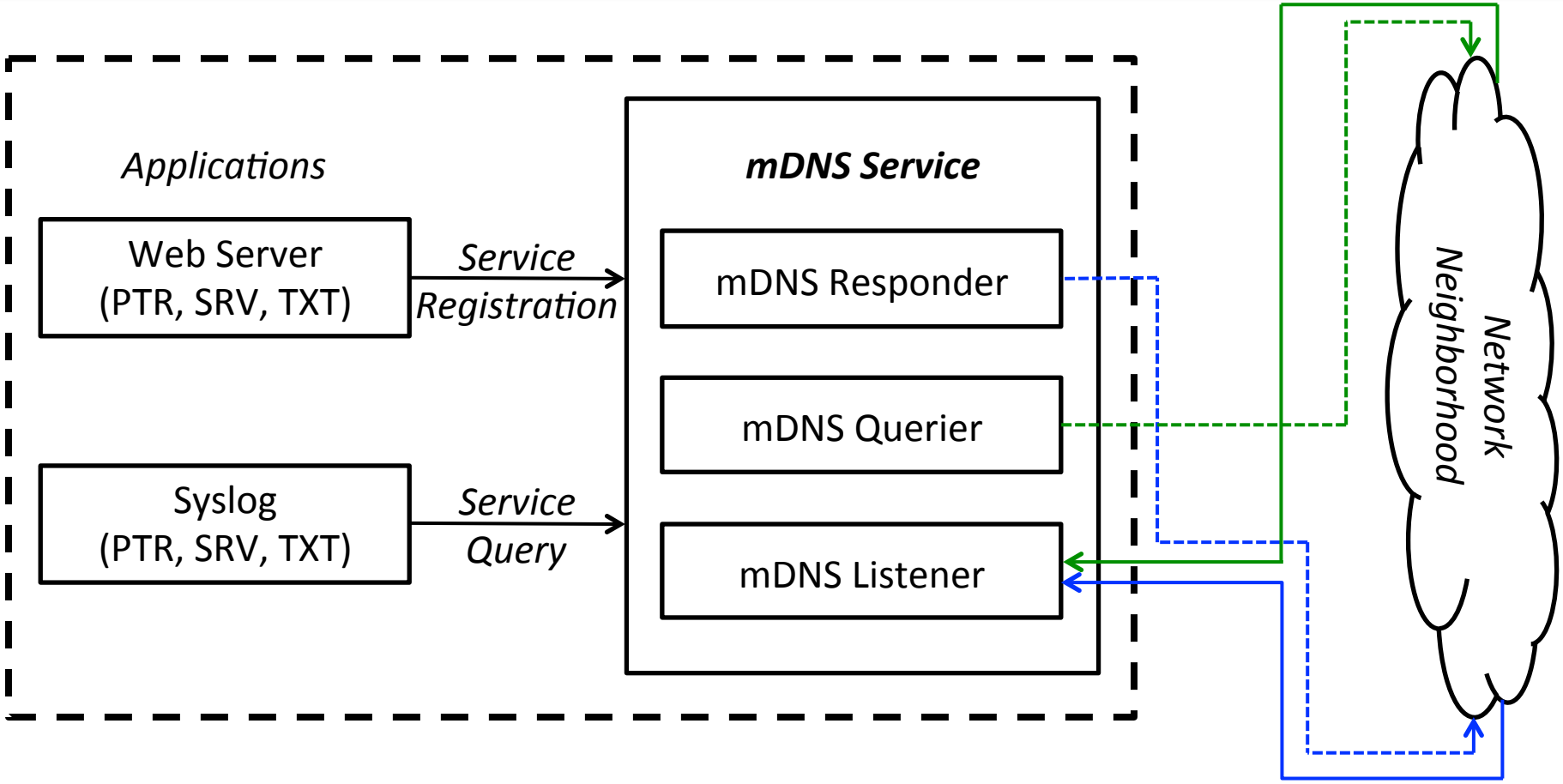
Multicast DNS (mDNS)

1. **Uses DNS over a multicast group ([FF02::FB]:5353).**
2. **No central DNS server.**
3. **Every node monitors group for queries.**
4. **Responses are broadcast to the group.**

Search for HTTP servers in local domain

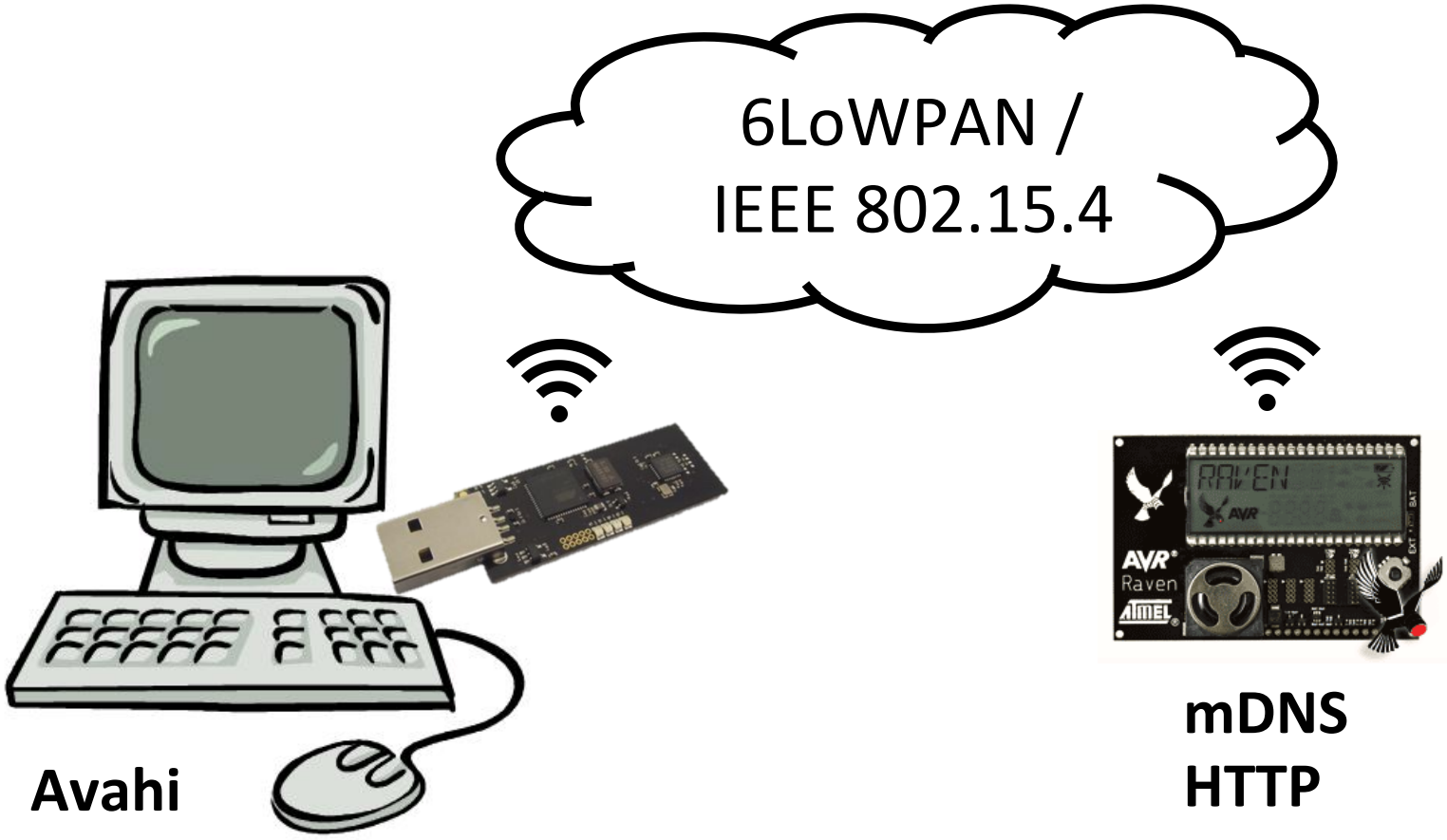
PTR query for *_http._tcp.local*

Implementation



- Developed for Contiki 2.5 OS
- Used the AVR Raven platform for development

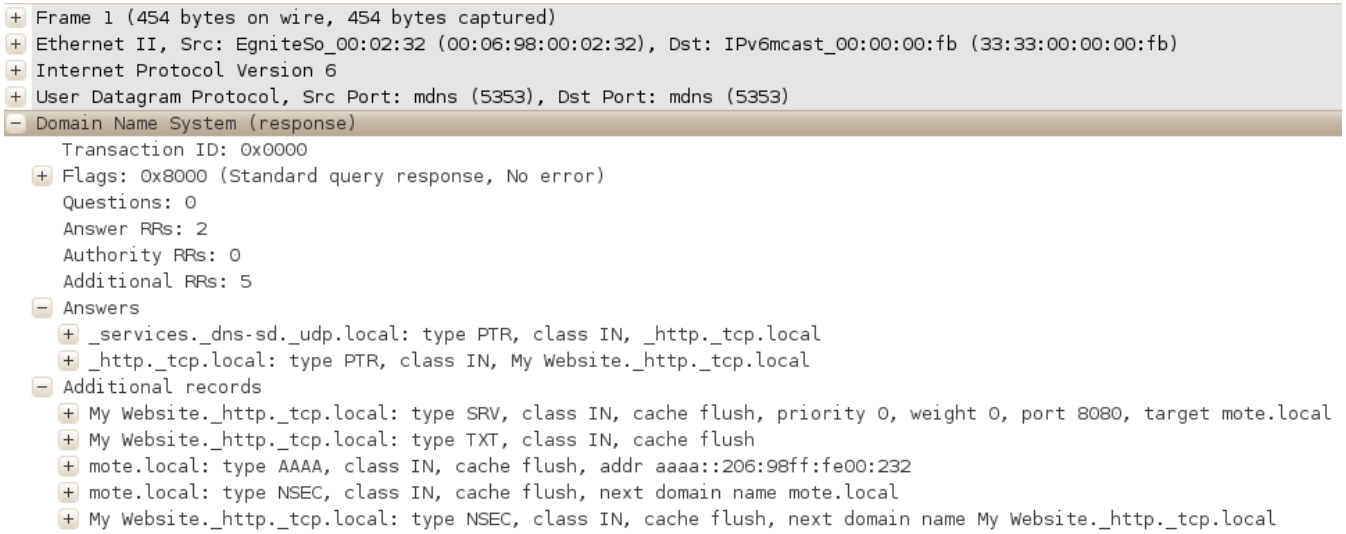
Evaluation



Avahi
Wireshark
6LoWPAN Routing

mDNS
HTTP
Syslog

Evaluation



```
+ Frame 1 (454 bytes on wire, 454 bytes captured)
+ Ethernet II, Src: EgniteSo_00:02:32 (00:06:98:00:02:32), Dst: IPv6mcast_00:00:00:fb (33:33:00:00:00:fb)
+ Internet Protocol Version 6
+ User Datagram Protocol, Src Port: mdns (5353), Dst Port: mdns (5353)
- Domain Name System (response)
    Transaction ID: 0x0000
  + Flags: 0x8000 (Standard query response, No error)
    Questions: 0
    Answer RRs: 2
    Authority RRs: 0
    Additional RRs: 5
  - Answers
    + _services._dns-sd._udp.local: type PTR, class IN, _http._tcp.local
    + _http._tcp.local: type PTR, class IN, My Website._http._tcp.local
  - Additional records
    + My Website._http._tcp.local: type SRV, class IN, cache flush, priority 0, weight 0, port 8080, target mote.local
    + My Website._http._tcp.local: type TXT, class IN, cache flush
    + mote.local: type AAAA, class IN, cache flush, addr aaaa::206:98ff:fe00:232
    + mote.local: type NSEC, class IN, cache flush, next domain name mote.local
    + My Website._http._tcp.local: type NSEC, class IN, cache flush, next domain name My Website._http._tcp.local
```

```
+ tap0 IPv6 instant-contiki [b2:dc:7c:ed:bd:f8]          Workstation          local
= tap0 IPv6 instant-contiki [b2:dc:7c:ed:bd:f8]          Workstation          local
   hostname = [instant-contiki.local]
   address = [fe80::b0dc:7cff:feed:bdf8]
   port = [9]
   txt = []
+ tap0 IPv6 My Website                                   Web Site             local
= tap0 IPv6 My Website                                   Web Site             local
   hostname = [mote.local]
   address = [aaaa::206:98ff:fe00:232]
   port = [8080]
   txt = ["path=/mywebsite"]
```

Evaluation

	mDNS (HTTP, Syslog)	mDNS (Incoming Queries)	Contiki
Flash	7.2 kB	3.6 kB	27.3 kB
RAM	0.2 kB	0.7 kB	10.7 kB

- **Relatively low resource usage on the target device.**
- **Integrates well with the existing Internet infrastructure.**

Room for further optimization

Conclusions

- **Resource constrained networks require service and device discovery**
- **CoAP and OMA LwM2M solve this problem, but not in an Internet standard way**
- **Maintaining registries at IoT scale is not optimal**
- **mDNS with DNS-SD provides a good Internet standard way of discovering devices and resources, even for resource constrained networks**

Thank you for your attention!
Questions?