IoT infrastructure

State of art

Aghiles DJOUDI

Paris-Est University

March 18, 2019

Plan

Context

Introduction

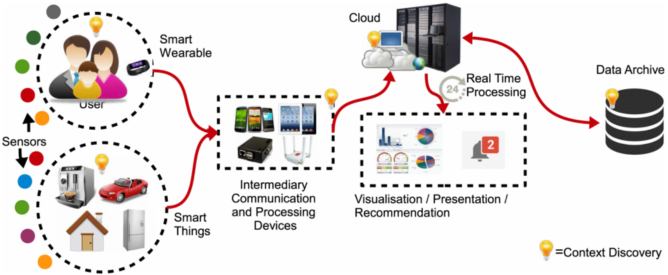

Figure 1: The IoT Platform

- Connect sensors to the gateway.
- Connect the gateway to the infrastructure.
- Store & Analyze sensors data.

Problematic

Introduction

Figure 2: The IoT problematics

- How to communicate sensors efficiently
  - IEEE 802.15.4, 6LowPAN
  - Throughput, Delay, Jitter, Loss rate and Availability.
- How to communicate sensors with the infrastructure efficiently
  - LPWAN, LoraWan
  - Interoperability ?
- How to extract knowledge from sensors data.
  - Data mining: Classification, Clustering
  - Deep learning: Machine learning

Plan

Plan

Application protocol	DDS	CoAP	AMQP	MQTT	MQTT-SN	XMPP	HTTP
Service discovery	mDNS			DNS-SD			
Network layer	RPL						
Link layer	IEEE 802.15.4						
Physical layer	EPCglobal			IEEE 802.15.4		Z-Wave	

Table 1: Standardization efforts that support the IoT

State of the art
Standardization

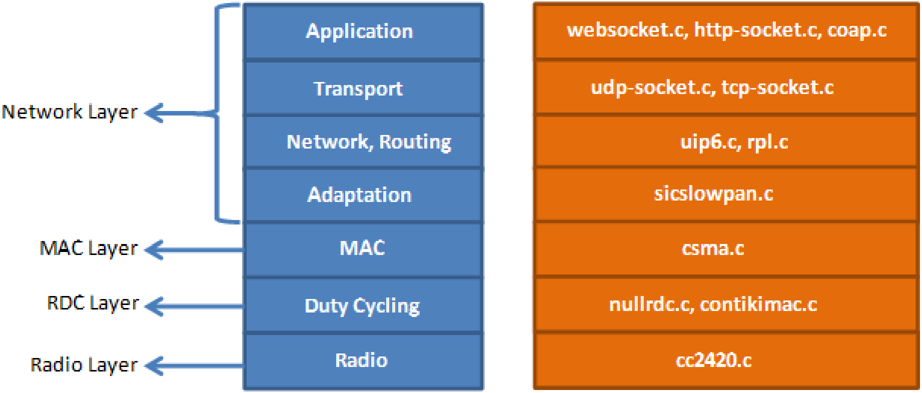

Figure 3: Rime Stack

State of the art

Standardization

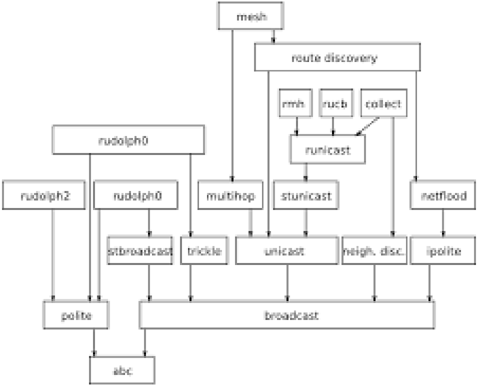

Figure 4: Uip Stack

State of the art
Standardization

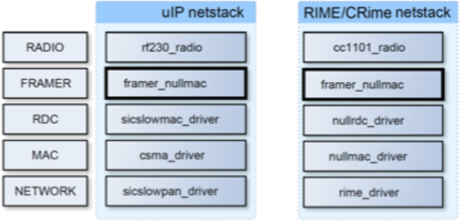

Figure 5: rime VS uip

Plan

CoAP application

State of the art

- Constrained Application Protocol
- The IETF Constrained RESTful Environments
- CoAP is bound to UDP
- CoAP can be divided into two sub-layers
  - messaging sub-layer
  - request/response sub-layer
    a) Confirmable.
    b) Non-confirmable.
    c) Piggybacked responses.
    d) Separate response
- CoAP, as in HTTP, uses methods such as:
  - GET, PUT, POST and DELETE to
  - Achieve, Create, Retrieve, Update and Delete
  - Ex: the GET method can be used by a server to inquire the clients temperature

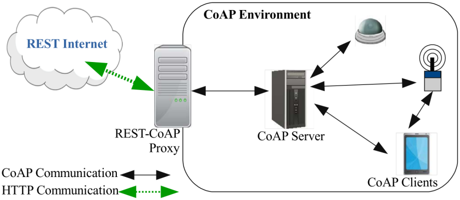

Figure 6: CoAP application

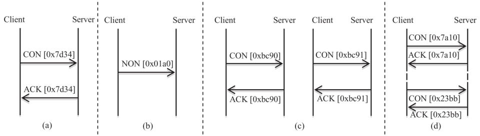

Figure 7: CoAP message types

CoAP application

State of the art

0 1	2 3	4-7	8-15	16-31
Ver	T	OC	CODE	Message ID
Token				
Options				
Payload				

Table 2: CoAP message format.

Ver: is the version of CoAP

T: is the type of Transaction

OC: is Option count

Code: represents the request method (1-10) or response code (40-255).

- ➟ Ex: the code for GET, POST, PUT, and DELETE is 1, 2, 3, and 4, respectively.

Message ID: is a unique identifier for matching the response.

MQTT application

State of the art

- Message Queue Telemetry Transport
- Andy Stanford-Clark of IBM and Arlen Nipper of Arcom
  - Standardized in 2013 at OASIS
- MQTT uses the publish/subscribe pattern to provide transition flexibility and simplicity of implementation
- MQTT is built on top of the TCP protocol
- MQTT delivers messages through three levels of QoS
- Specifications
  - MQTT v3.1 and MQTT-SN (MQTT-S or V1.2)
  - MQTT v3.1 adds broker support for indexing topic names
- The publisher acts as a generator of interesting data.

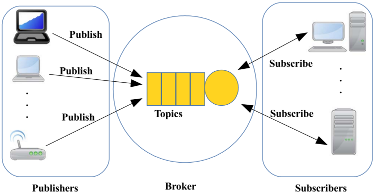

Figure 8: MQTT application

MQTT application

State of the art

0-3	4	5 6	7
Message type	DUP	QoS level	Retain
Remaining length			
Variable length header			
Variable length message payload			

Table 3: MQTT message format.

Message type: CONNECT (1), CONNACK (2), PUBLISH (3), SUBSCRIBE (8) and so on

DUP flag: indicates that the massage is duplicated

QoS Level: identify the three levels of QoS for delivery assurance of Publish messages

Retain field: retain the last received Publish message and submit it to new subscribers as a first message

XMPP application

State of the art

- Extensible Messaging and Presence Protocol
- Developed by the Jabber open source community
- An IETF instant messaging standard used for:
  - multi-party chatting, voice and telepresence
- Connects a client to a server using a XML stanzas
- An XML stanza is divided into 3 components:
  - message: fills the subject and body fields
  - presence: notifies customers of status updates
  - iq (info/query): pairs message senders and receivers
- Message stanzas identify:
  - the source (from) and destination (to) addresses
  - types, and IDs of XMPP entities

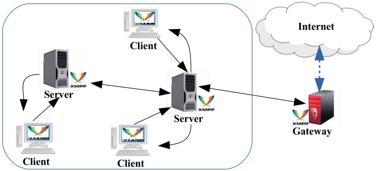

Figure 9: XMPP application

```
<stream>
  <presence>
    <show/>
  </presence>
  <message to='x'>
    <body/>
  </message>
  <iq to='y'>
    <query/>
  </iq>
</stream>
```

Figure 10: XML stanza

AMQP application

State of the art

- Advanced Message Queuing Protocol
- Communications are handled by two main components
  - exchanges: route the messages to appropriate queues.
  - message queues: Messages can be stored in message queues and then be sent to receivers
- It also supports the publish/subscribe communications.
- It defines a layer of messaging on top of its transport layer.
- AMQP defines two types of messages
  - bare massages: supplied by the sender
  - annotated messages: seen at the receiver
- The header in this format conveys the delivery parameters:
  - durability, priority, time to live, first acquirer & delivery count.
- AMQP frame format

Size: the frame size.

DOFF: the position of the body inside the frame.

Type: the format and purpose of the frame.

- ✳ Ex: 0x00 show that the frame is an AMQP frame
- ✳ Ex: 0x01 represents a SASL frame.

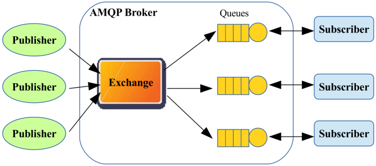

Figure 11: AMQP application

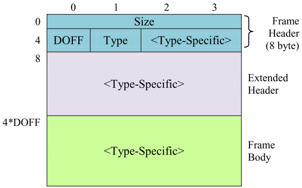

Figure 12: AMQP frame format

DDS application
State of the art

- Data Distribution Service
- Developed by Object Management Group (OMG)
- Supports 23 QoS policies:
  - like security, urgency, priority, durability, reliability, etc
- Relies on a broker-less architecture
  - uses multicasting to bring excellent Quality of Service
  - real-time constraints
- DDS architecture defines two layers:

DLRL Data-Local Reconstruction Layer
- serves as the interface to the DCPS functionalities

DCPS Data-Centric Publish/Subscribe
- delivering the information to the subscribers

- 5 entities are involved with the data flow in the DCPS layer:
  - Publisher:disseminates data
  - DataWriter: used by app to interact with the publisher
  - Subscriber: receives published data and delivers them to app
  - DataReader: employed by Subscriber to access received data
  - Topic: relate DataWriters to DataReaders

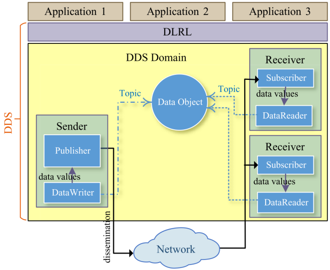

Figure 13: DDS application

IoT cloud platforms

State of the art

Platform	COAP	XMPP	MQTT
Arkessa			✓
Axeda			
Etherios			
LittleBits			
NanoService	✓		
Nimbits		✓	
Ninja blocks			
OnePlateformv	✓	✓	
RealTime.io			
SensorCloud			
SmartThings			
TempoDB			
ThingWorx			✓
Xively			✓
Ubidots			✓

Table 4: IoT cloud platforms and their characteristics

Plan

MDNS

State of the art

- No need for manual reconfiguration or extra administration
- It is able to run without infrastructure
- It is able to continue working if failure happens.
- It inquires names by sending an IP multicast message to all the nodes in the local domain
  - Clients asks devices that have the given name to reply back
  - the target machine receives its name and multicasts its IP @
  - Devices update their cache with the given name and IP @

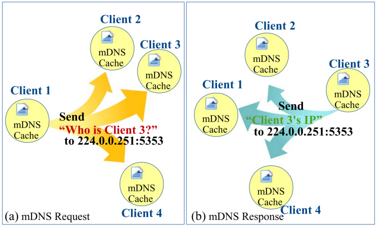

Figure 14: MDNS

DNS-SD

State of the art

- Requires zero configuration aids to connect machine
- It uses mDNS to send DNS packets to specific multicast addresses through UDP
- There are two main steps to process Service Discovery:
  - finding host names of required services such as printers
  - pairing IP addresses with their host names using mDNS
- Advantages
  - IoT needs an architecture without dependency on a configuration mechanism
  - smart devices can join the platform or leave it without affecting the behavior of the whole system
- Drawbacks
  - Need for caching DNS entries

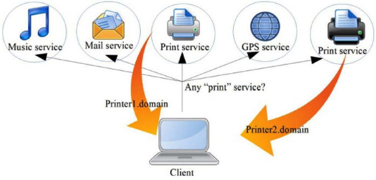

Figure 15: DNS-SD

IoT operating systems

State of the art

OS	Architecture	Multi threading	Scheduling	Dynamic Memory	Memory protection	Network Stack	Virtualization and Completion
Contiki/Contiki-ng	Modular	✓	Interrupts execute w.r.t.	✓	✗	uIP Rime	Serialized Access
MANTIS	Modular	✗	Priority classes	✓	✗	At Kernel COMM layer	Semaphores.
Nano-RK	Layered	✓	Monotonic harmonized	✗	✗	Socket abstraction	Serialized access semaphores
LiteOS	Monolithic	✓	Round Robin	✓	✓	File	Synchronization primitives

Table 5: Common operating systems used in IoT environment [1]

Plan

IEEE 802.15.4 topologies

State of the art

- Standard topologies to form IEEE 802.15.4e networks are

  Star contains at least one FFD and some RFDs

  Mesh contains a PAN coordinator and other nodes communicate with each other

  Cluster consists of a PAN coordinator, a cluster head and normal nodes.

- The IEEE 802.15.4e standard supports 2 types of network nodes

  FFD Full function device: serve as a coordinator
  - It is responsible for creation, control and maintenance of the net
  - It store a routing table in their memory and implement a full MAC

  RFD Reduced function devices: simple nodes with restricted resources
  - They can only communicate with a coordinator
  - They are limited to a star topology

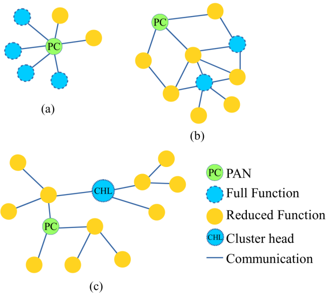

Figure 16: IEEE 802.15.4 topologies. (a) Star. (b) Mesh. (c) Cluster-tree.

Plan

Results

State of the art

Phy protocol	Radio band (MHz)	MAC address	Data rate (bps)	Scalability
IEEE 802.15.4	868/915/2400	TDMA, CSMA/CA	20/40/250 K	65K nodes
BLE	2400	TDMA	1024K	5917 slaves
EPCglobal	860-960	ALOHA	varies 5-640K	-
Z-Wave	868/908/2400	CSMA/CA	40K	232 nodes

Table 6: IoT cloud platforms and their characteristics [1]

Results

Comparison with related work

Characteristics	6LoWPAN	LoRaWAN	SigFox
Frequency band **(MHz)**	902-929 868-868.6	902-928 863-870 and 434	902 868
Number of channels **(channels for MHz)**	0016 for 2400 0010 for 915 0001 for 868.3	80 for 915 10 for 868 and 780	25
Channel bandwidth **(MHz)**	0005 for 2400 0002 for 915 0600 for 868.3	0.125 and 0.50 for 915 0.125 and 0.25 for 868 and 780	0.0001-0.0012
Maximum data rate **(kbps for MHz)**	0250 for 2400 0040 for 915 0020 for 868.3	0.00098-0.0219 for 915 0.250-0.05 for 868 and 780	0.1-0.6
Channel coding **(dBm for MHz)**	-085 for 2400 -092 for 915 -092 for 868.3	-137	-137
Protocol data unit (bytes)	6+127	x + (19 to 250)	12+ (0 to 12)
Channel coding	Direct	CSS	Ultra
Transmission range	10-100 m	5-15 km	10-50 km
Battery lifetime	1-2 years	<10 years	<10 years

Table 7: LPWan Characteristics [2]

Plan

Conclusion

Routing protocol	Control Cost	Link Cost	Node Cost
OSPF/IS-IS	✗	✓	✗
OLSRv2	?	✓	✓
RIP	✓	?	✗
DSR	✓	✗	✗
RPL	✓	✓	✓

Table 8: Routing protocols comparison **_rpl2_**

Application protocol	Rest-Full	Trans-port	Pub-lish/Sub-scribe	Request/Re-sponse	Secu-rity	QoS	Header size (Byte)
COAP	✓	UDP	✓	✓	DTLS	✓	4
MQTT	✗	TCP	✓	✗	SSL	✓	2
MQTT-SN	✗	TCP	✓	✗	SSL	✓	2
XMPP	✗	TCP	✓	✓	SSL	✗	-
AMQP	✗	TCP	✓	✗	SSL	✓	8
DDS	✗	UDP TCP	✓	✗	SSL DTLS	✓	-
HTTP	✓	TCP	✗	✓	SSL	✗	-

Table 9: Application protocols comparison

Conclusion

Routing protocol	Control Cost	Link Cost	Node Cost
OSPF/IS-IS	✗	✓	✗
OLSRv2	?	✓	✓
RIP	✓	?	✗
DSR	✓	✗	✗
RPL	✓	✓	✓

Table 8: Routing protocols comparison _**rpl2**_

Application protocol	Rest-Full	Trans-port	Pub-lish/Sub-scribe	Request/Re-sponse	Secu-rity	QoS	Header size (Byte)
COAP	✓	UDP	✓	✓	DTLS	✓	4
MQTT	✗	TCP	✓	✗	SSL	✓	2
MQTT-SN	✗	TCP	✓	✗	SSL	✓	2
XMPP	✗	TCP	✓	✓	SSL	✗	-
AMQP	✗	TCP	✓	✗	SSL	✓	8
DDS	✗	UDP TCP	✓	✗	SSL DTLS	✓	-
HTTP	✓	TCP	✗	✓	SSL	✗	-

Table 9: Application protocols comparison

Thank you !

References

[1] A. Al-Fuqaha, M. Guizani, M. Mohammadi, M. Aledhari, and M. Ayyash, " *Internet of Things: A Survey on Enabling Technologies, Protocols, and Applications," *IEEE Communications Surveys & Tutorials*, vol. 17, no. 4, pp. 2347–2376, 24–2015, 01947.

[2] H. A. A. Al-Kashoash and A. H. Kemp, " Comparison of 6LoWPAN and LPWAN for the Internet of Things," *Australian Journal of Electrical and Electronics Engineering*, vol. 13, no. 4, pp. 268–274, Oct. 2016, 00007.