Toward a Definition of Internet of Things

Roberto Minerva, IEEE IoT Initiative Chair – TIMLab

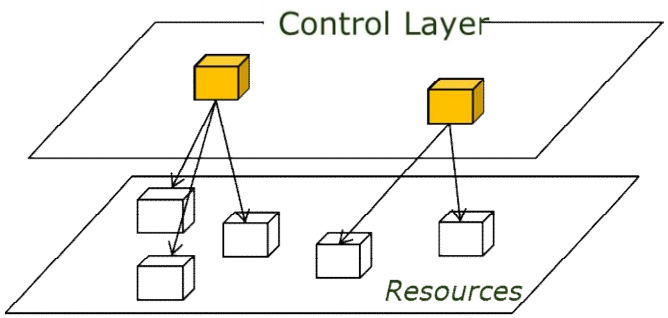
08 - 10 June 2016

IEEE Internet of Things

IEEE Advancing Technology for Humanity

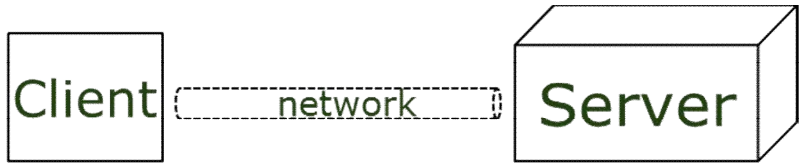
END OF PROTOCOLS, APIS AND MIDDLEWARE

How Smart Objects will communicate



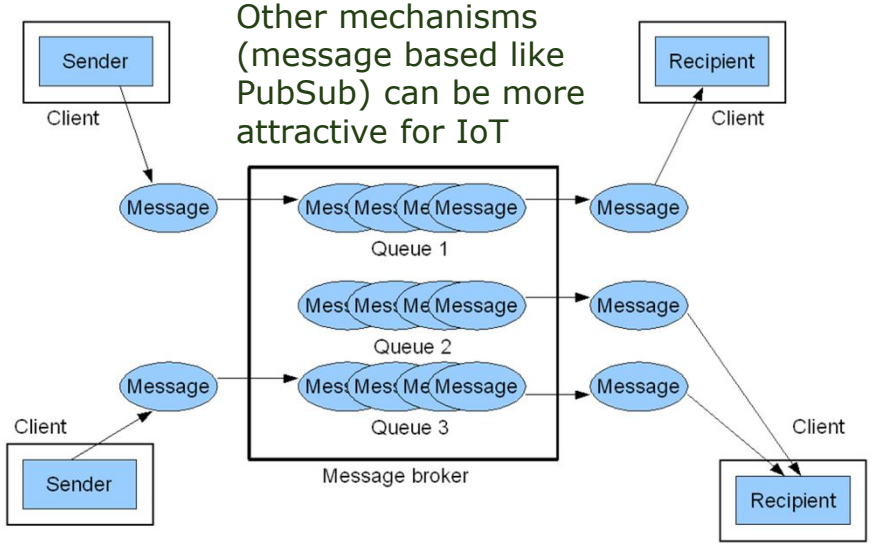
Physical Layer

Network Intelligence (e.g., IMS) is a hierarchical model based on the assumption that control has to be exerted by a few specialized control nodes



Client – Server model disregards the network aspects and can lead to a tragedy of commons (misuse of common networking resources)

Other mechanisms (message based like PubSub) can be more attractive for IoT



This is a reason for different IoT protocols …
Is it there a single communication paradigm for IoT ?

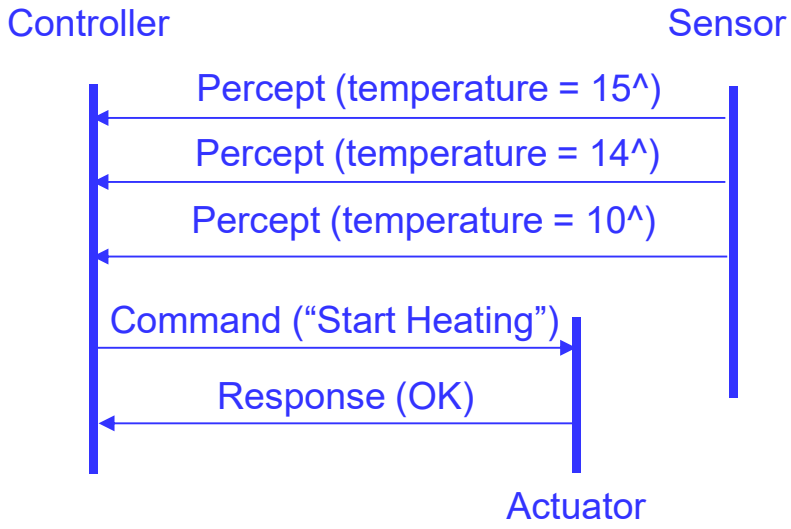
IEEE Internet of Things

for humanity

Interactions with Things (e.g., sensors)

Ideally:

- **Few Primitives as in M2M_XML**
  - **Percept**
  - **Command**
  - **Response**
  - **Exception**
  - **Property**
- **Simple Control: Events and Commands**
- **Simple Semantic**



Many protocols are currently used: SensorML, COAP, MQTT, … each one adhering to a communication paradigm. Another Protocol Battle ?

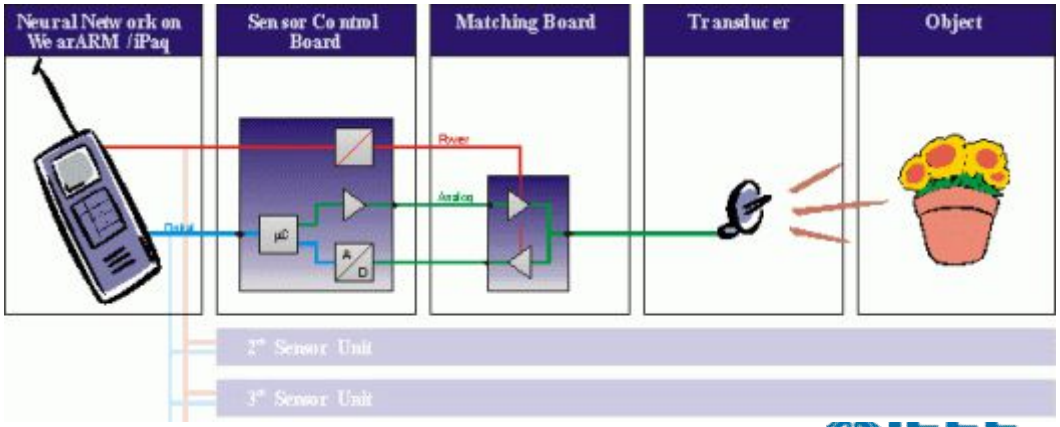
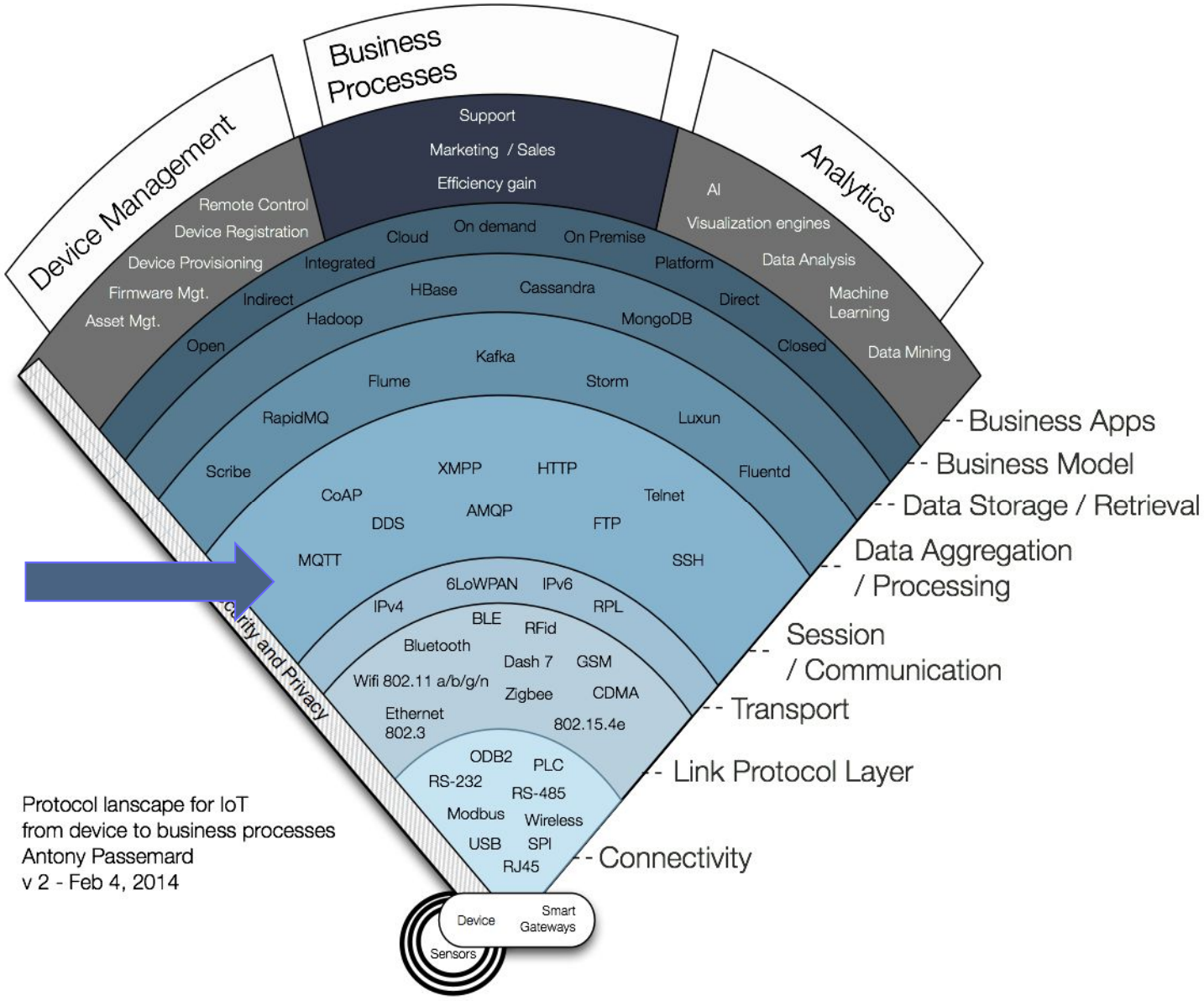


Figure from: http://www.wearable.ethz.ch/context_recognition.0.html

IoT Protocol Landscape

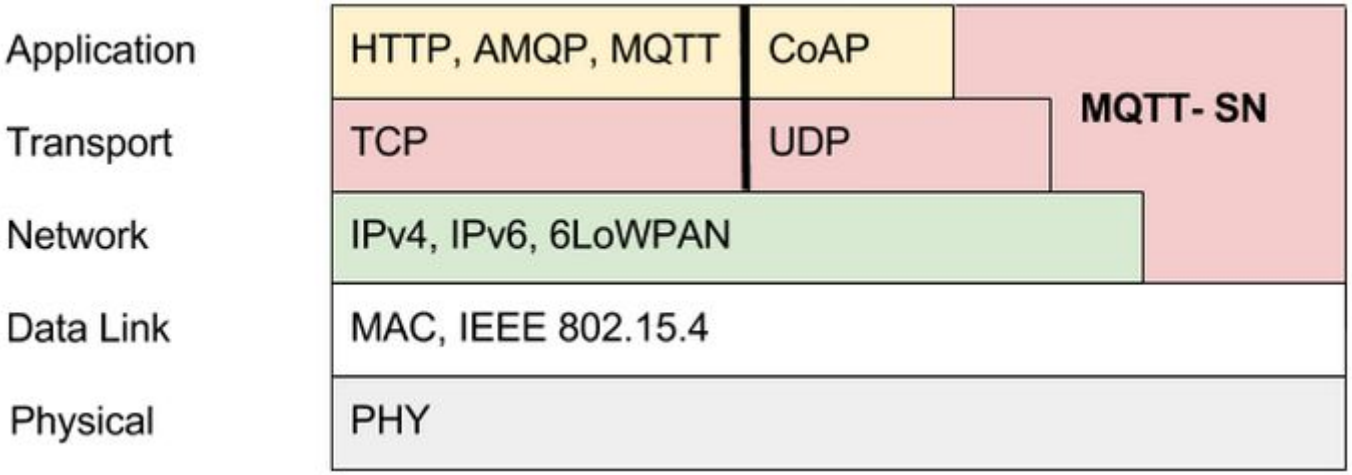


Protocol lanscape for IoT
from device to business processes
Antony Passemard
v 2 - Feb 4, 2014

https://entrepreneurshiptalk.wordpress.com/2014/01/29/the-internet-of-thing-protocol-stack-from-sensors-to-business-value/

IEEE Internet of Things

A Bit of Protocols

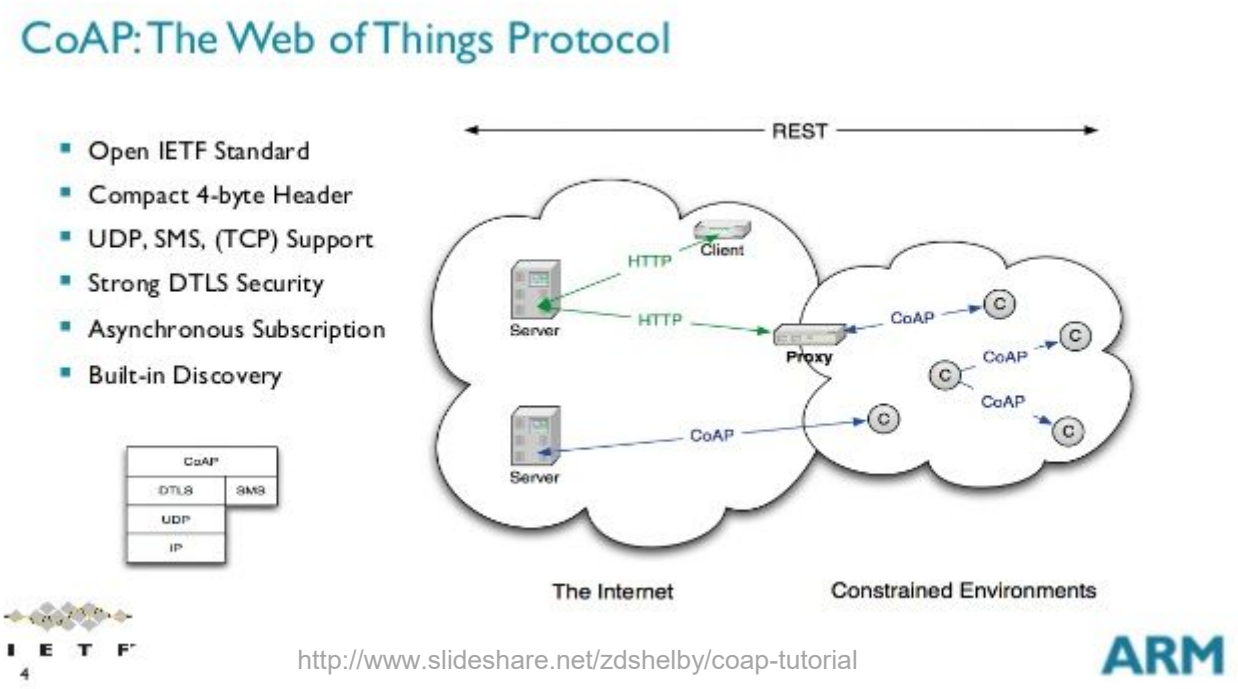


http://blog.zuehlke.com/en/iot-for-tiny-devices-lets-talk-mqtt-sn/

Constrained Application Protocol (CoAP) uses a REST-like approach offering resources with unique URI's for consumption by CoAP clients or the setting and update of such resources.

MQTT (formerly Message Queue Telemetry Transport) is a publish-subscribe based "light weight" messaging protocol for use on top of the TCP/IP protocol. It is designed for connections with remote locations where a "small code footprint" is required and/or network bandwidth is limited.

CoAP - Constrained Application Protocol



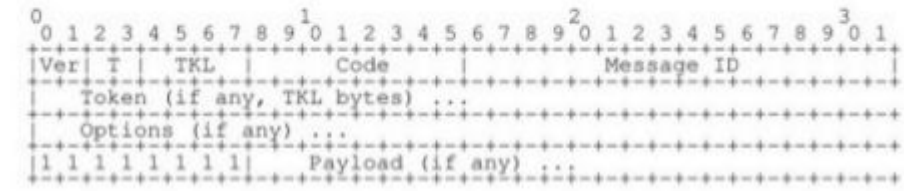
- Constrained Application Protocol (CoAP) is a software protocol to be used in simple electronics devices that allows them to communicate interactively over the Internet.
- It is targeted for small low power sensors, switches, valves and similar components that need to be controlled or supervised remotely, through standard Internet networks.
- CoAP is an application layer protocol that is intended for use in resource-constrained internet devices, such as WSN nodes.
- CoAP is designed to easily translate to HTTP for simplified integration with the web, while also meeting specialized requirements such as multicast support, very low overhead, and simplicity
- Source: Wikipedia

CoAP (2)

CoAP Features

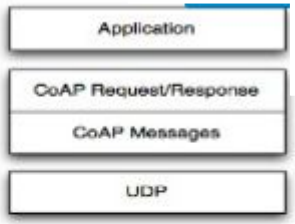
- Embedded web transfer protocol (coap://)
- Asynchronous transaction model
- UDP binding with reliability and multicast support
- GET, POST, PUT, DELETE methods
- URI support
- Small, simple 4 byte header
- DTLS based PSK, RPK and Certificate security
- Subset of MIME types and HTTP response codes
- Built-in discovery
- Optional observation and block transfer

Message Header (4 bytes)



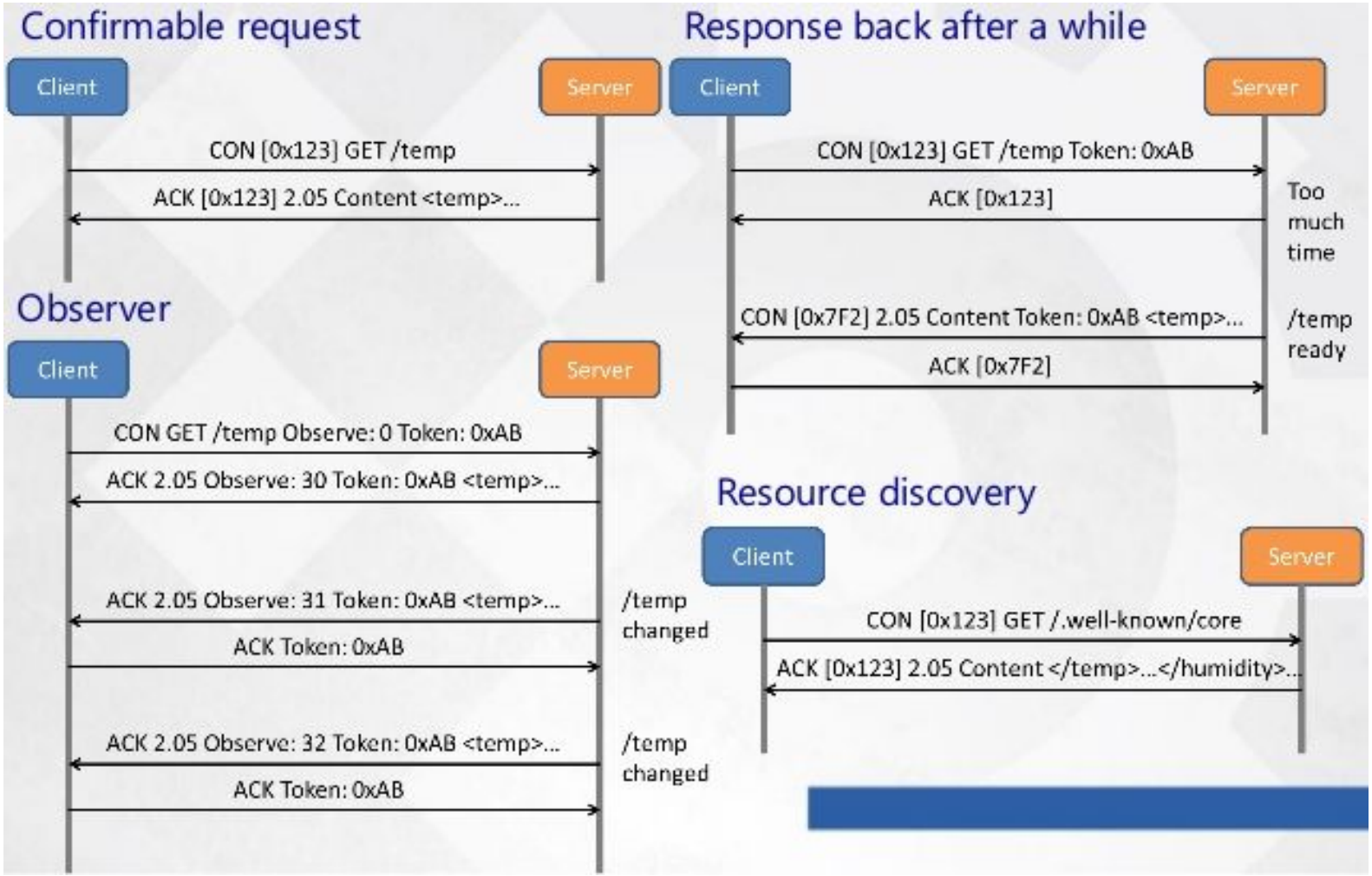
Ver - Version (1)
T - Message Type (Confirmable, Non-Confirmable, Acknowledgement, Reset)
TKL- Token Length, if any, the number of Token bytes after this header
Code - Request Method (1-10) or Response Code (40-255)
Message ID - 16-bit identifier for matching responses
Token - Optional response matching token

Transaction Model



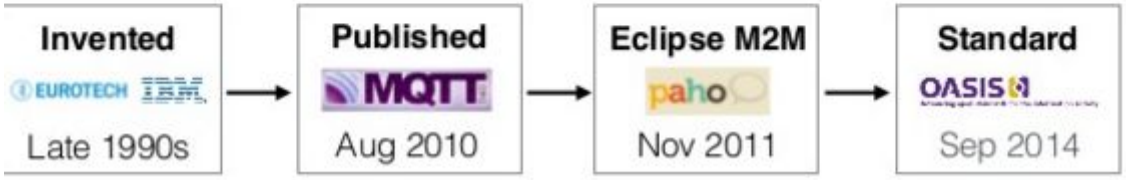
- Transport
  - CoAP currently defines:
  - UDP binding with DTLS security
  - CoAP over SMS or TCP possible
- Base Messaging
  - Simple message exchange between endpoints
  - Confirmable or Non-Confirmable Message answered by Acknowledgement or Reset Message
- REST Semantics
  - REST Request/Response piggybacked on CoAP Messages
  - Method, Response Code and Options (URI, content-type etc.)

COAP Services



Message Queuing Telemetry Transport

- **open** — open spec, standard — 40+ client implementations
- **lightweight** — minimal overhead — efficient format — tiny clients (kb)
- **reliable** — QoS for reliability on unreliable networks
- **simple** — 43-page spec — connect + publish + subscribe

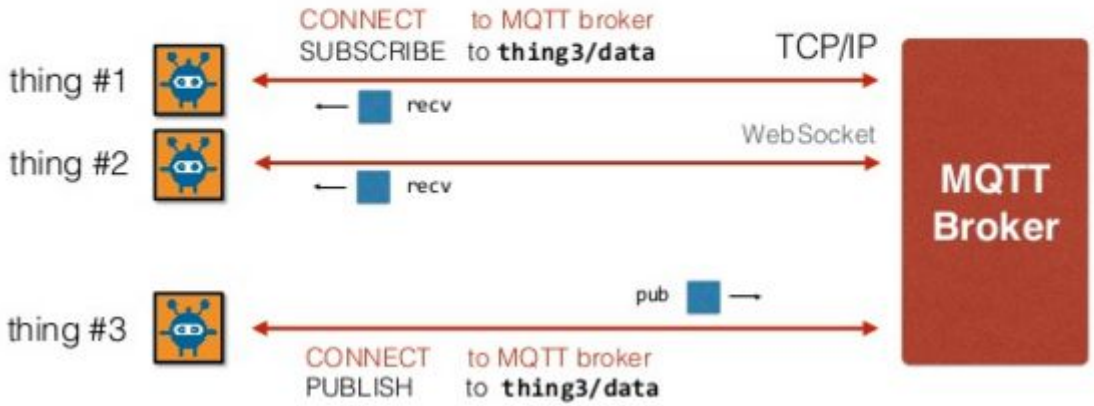


http://www.slideshare.net/BryanBoyd/mqtt-austin-api

MQTT (formerly Message Queuing Telemetry Transport) is a publish-subscribe based "light weight" messaging protocol for use on top of the TCP/IP protocol. It is designed for connections with remote locations where a "small code footprint" is required or the network bandwidth is limited. The publish-subscribe messaging pattern requires a message broker. The broker is responsible for distributing messages to interested clients based on the topic of a message

MQTT

bi-directional, async "push" communication



MQTT

simple to implement

Connect

Subscribe

Publish

Unsubscribe

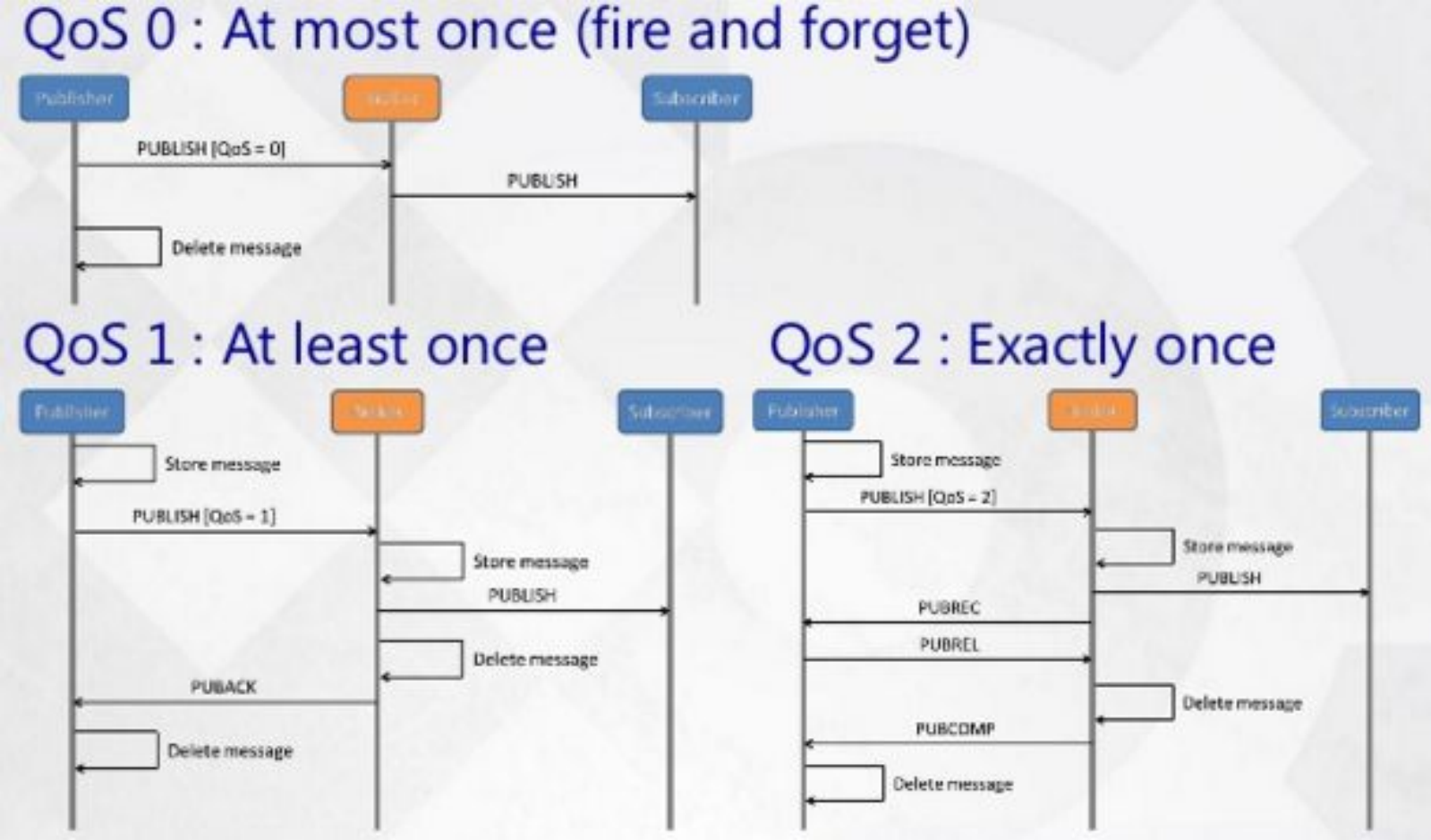
Disconnect

```
client = new Messaging.Client(hostname, port, clientId)
client.onMessageArrived = messageArrived;
client.onConnectionLost = connectionLost;
client.connect({ onSuccess: connectionSuccess });

function connectionSuccess() {
    client.subscribe("planets/earth");
    var msg = new Messaging.Message("Hello world!");
    msg.destinationName = "planets/earth";
    client.publish(msg);
}

function messageArrived(msg) {
    console.log(msg.payloadString);
    client.unsubscribe("planets/earth");
    client.disconnect();
}
```

MQTT / Quality of Service



http://www.slideshare.net/paolopat/mqtt-iot-protocols-comparison

MQTT vs. COAP Comparison

- MQTT and CoAP are both useful as IoT protocols, but have fundamental differences.
- MQTT is a many-to-many communication protocol for passing messages between multiple clients through a central broker. It decouples producer and consumer by letting clients publish and having the broker decide where to route and copy messages. While MQTT has some support for persistence, it does best as a communications bus for live data.
- CoAP is, primarily, a one-to-one protocol for transferring state information between client and server. While it has support for observing resources, CoAP is best suited to a state transfer model, not purely event based.
- MQTT clients make a long-lived outgoing TCP connection to a broker. This usually presents no problem for devices behind NAT. CoAP clients and servers both send and receive UDP packets. In NAT environments, tunnelling or port forwarding can be used to allow CoAP, or devices may first initiate a connection to the head-end.
- MQTT provides no support for labelling messages with types or other metadata to help clients understand it. MQTT messages can be used for any purpose, but all clients must know the message formats up-front to allow communication. CoAP, conversely, provides inbuilt support for content negotiation and discovery allowing devices to probe each other to find ways of exchanging data.
- Both protocols have pros and cons, choosing the right one depends on your application.

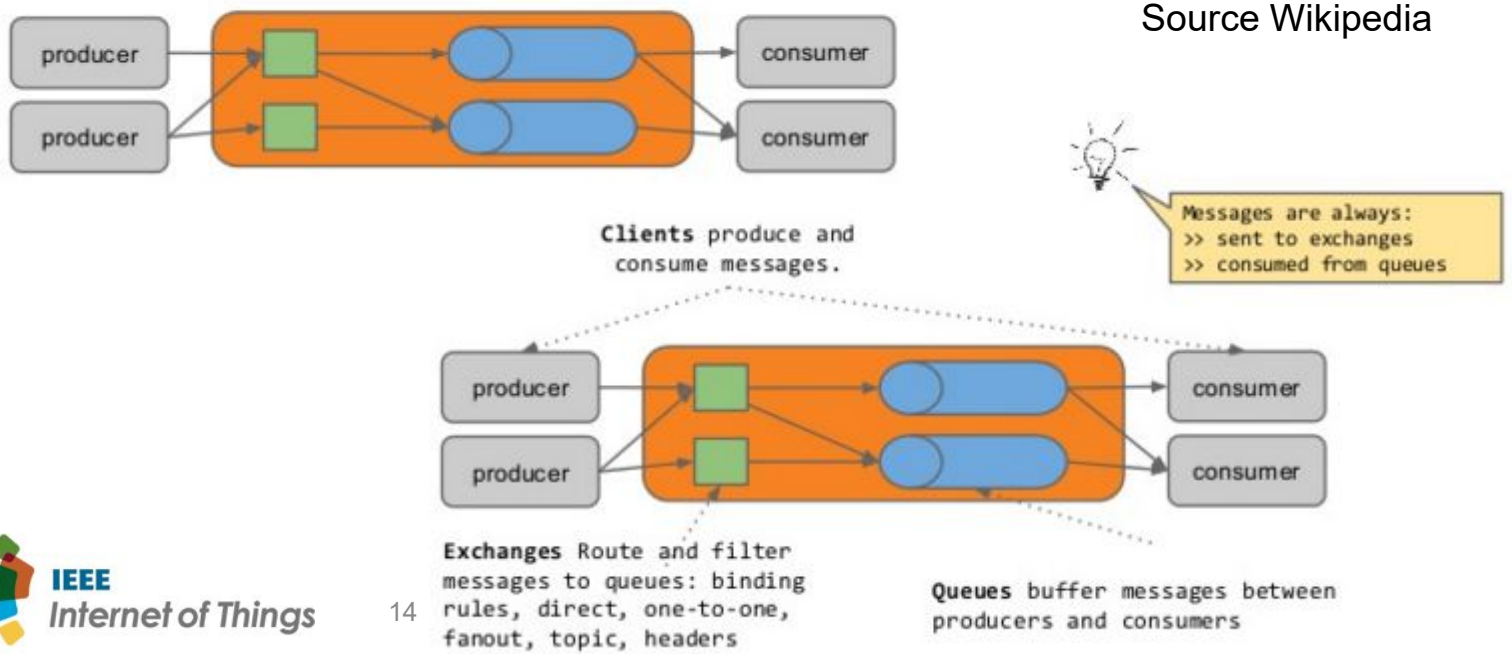
http://www.eclipse.org/community/eclipse_newsletter/2014/february/article2.php

Advanced Message Queuing Protocol (AMQP)

amqp is a message oriented wire-level protocol

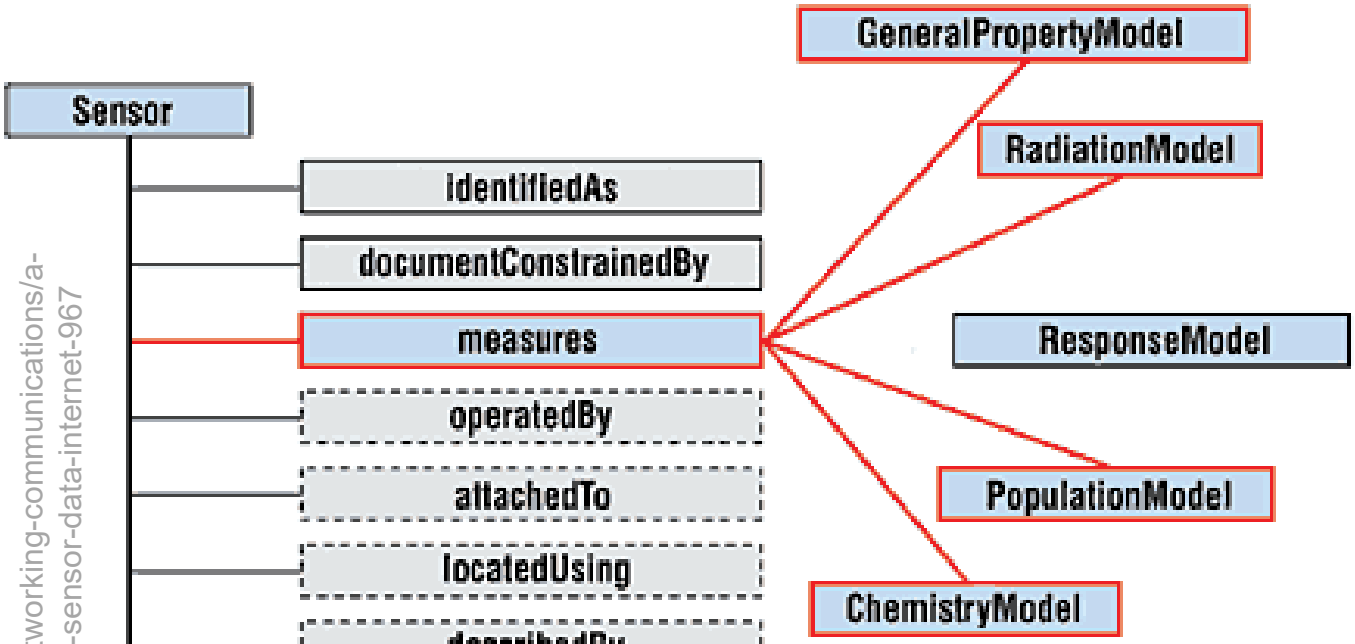
- message broker: receives and dispatches msgs using AMQP
- connection physical connection (eg: tcp/ip)
- channel: allows n clients over one connection

The Advanced Message Queuing Protocol (AMQP) is an open standard application layer protocol for message-oriented middleware. The defining features of AMQP are message orientation, queuing, routing (including point-to-point and publish-and-subscribe), reliability and security
Source Wikipedia



http://www.slideshare.net/javierarilos/rabbitmq-intromsgingpatterns

Sensor Model Language



http://www.sensorsmag.com/networking-communications/a-sensor-model-language-moving-sensor-data-internet-967

SensorML provides standard models and an XML encoding for describing sensors and measurement processes. SensorML can be used to describe a wide range of sensors,.
Functions supported include

- sensor discovery
- sensor geolocation
- processing of sensor observations
- a sensor programming mechanism
- subscription to sensor alerts

Source: wikipedia

Protocol Comparison

	DDS	AMQP	CoAP	MQTT	REST / HTTP	XMPP
TRANSPORT	UDP/IP (unicast + multicast) TCP/IP	TCP/IP	UDP/IP	TCP/IP	TCP/IP	TCP/IP
INTERACTION MODEL	Publish-and-Subscribe, Request-Reply	Point-to-Point Message Exchange	Request-Reply (REST)	Publish-and-Subscribe	Request-Reply	Point-to-Point Message Exchange
SCOPE	Device-to-Device Device-to-Cloud Cloud-to-Cloud	Device-to-Device Device-to-Cloud Cloud-to-Cloud	Device-to-Device	Device-to-Cloud Cloud-to-Cloud	Device-to-Cloud Cloud-to-Cloud	Device-to-Cloud Cloud-to-Cloud
AUTOMATIC DISCOVERY	✓	-	✓	-	-	-
CONTENT AWARENESS	Content-based Routing Queries	-	-	-	-	-
QoS	Extensive (20+)	Limited	Limited	Limited	-	-
INTEROPERABILITY LEVEL	Semantic	Structural	Semantic	Foundational	Semantic	Structural
SECURITY	TLS, DTLS, DDS Security	TLS + SASL	DTLS	TLS	HTTPS	TLS + SASL
DATA PRIORITIZATION	Transport Priorities	-	-	-	-	-
FAULT TOLERANCE	Decentralized	Implementation-Specific	Decentralized	Broker is SPoF	Server is SPoF	Server is SPoF

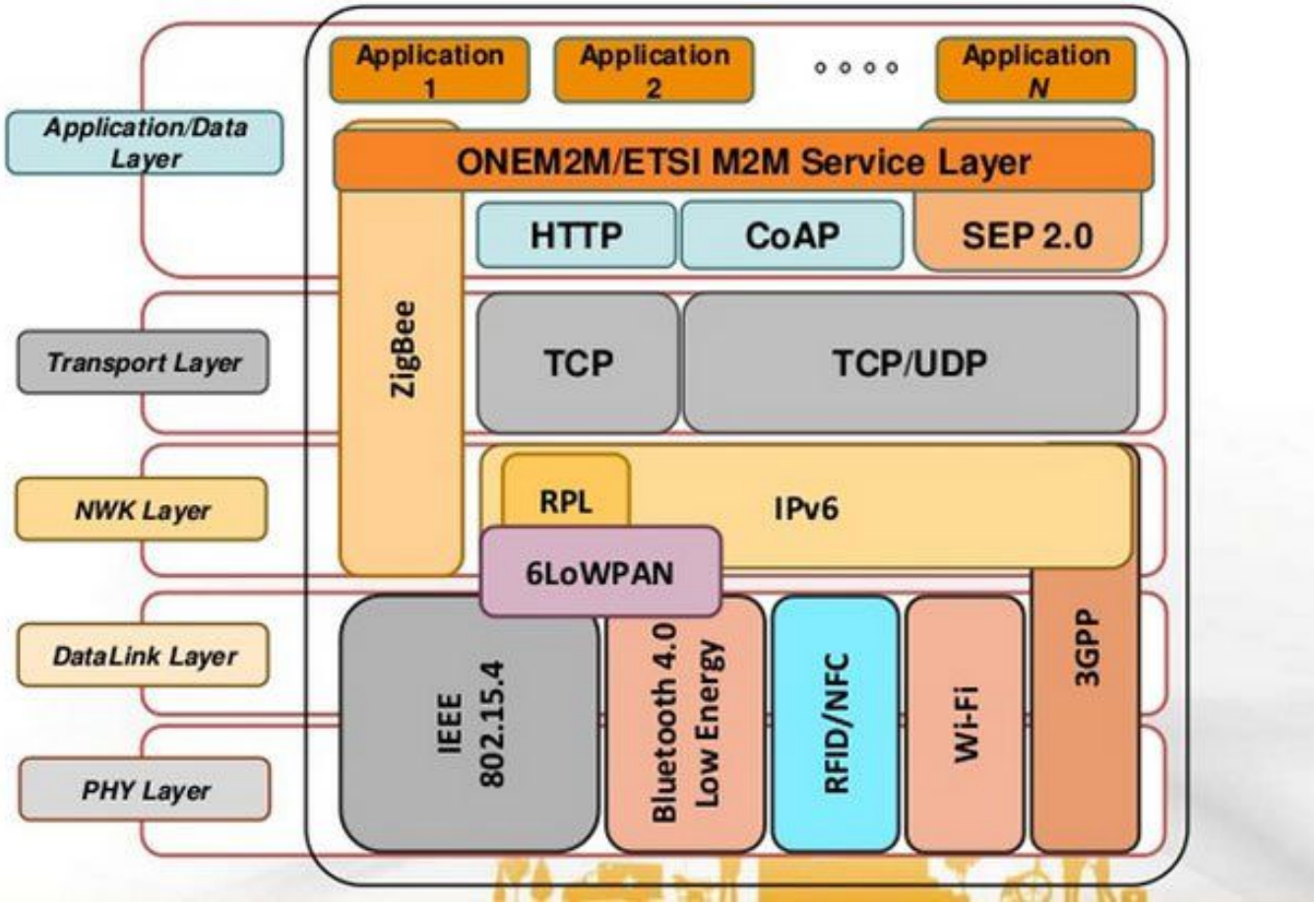
http://www.eetimes.com/author.asp?doc_id=1326169

Module 16: communications paradigms and IoT protocols

- What are some of the used communication paradigms in IoT ?
- What are some high level protocols used in IoT?
- Can you related a protocol to its communication paradigm?

FROM PROTOCOLS TO PLATFORMS

IoT Protocols Stack and initial Platform

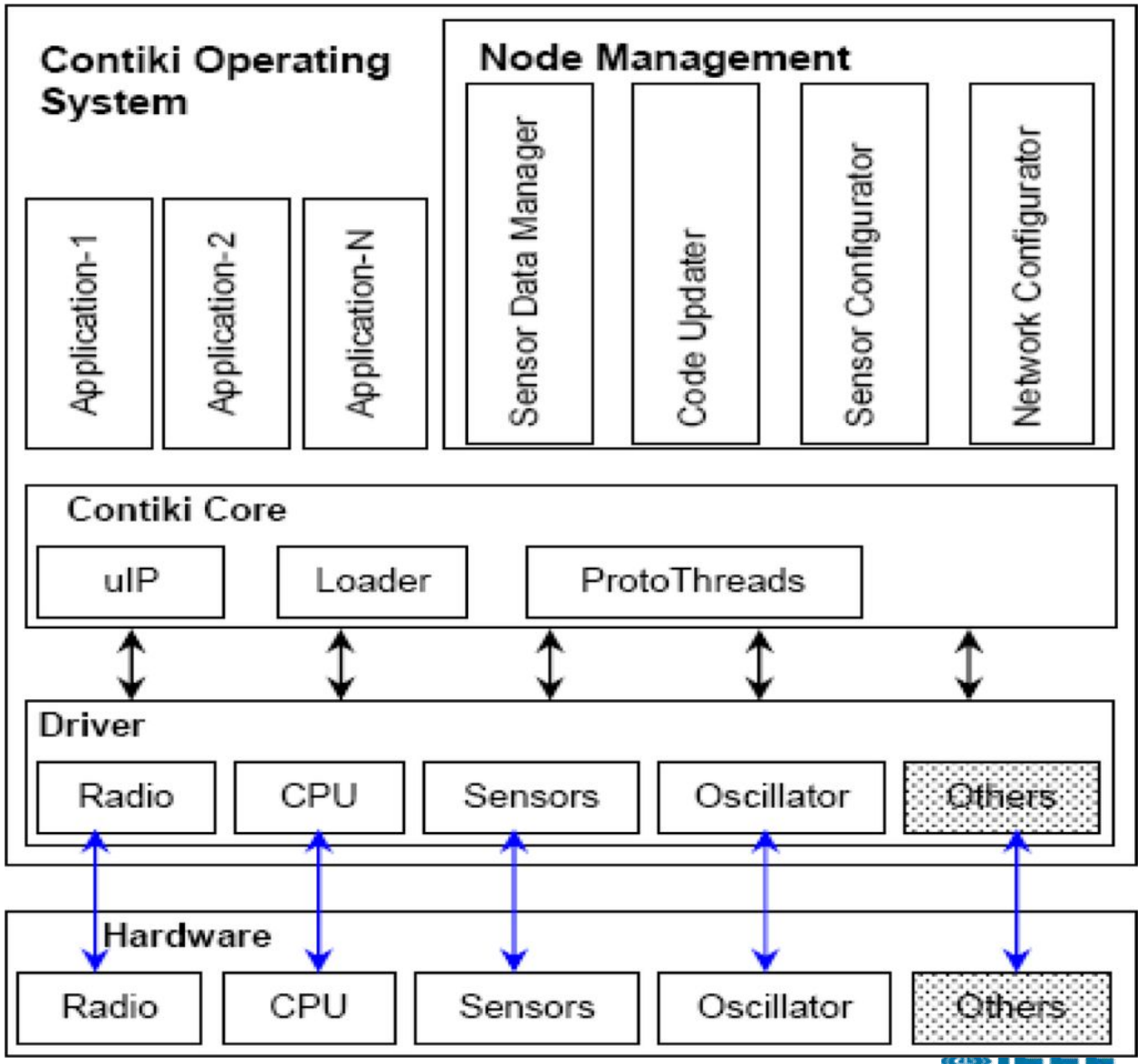


Available inhttp://postscapes.com/internet-of-things-protocols
Source: Butler project

Contiki

Contiki is an open source operating system for networked, memory-constrained systems with a particular focus on low-power wireless Internet of Things devices. Examples of where Contiki is used include street lighting systems, sound monitoring for smart cities, radiation monitoring systems, and alarm systems.

Despite providing multitasking and a built-in TCP/IP stack, Contiki only needs about 10 kilobytes of RAM and 30 kilobytes of ROM. A full system, complete with a graphical user interface, needs about 30 kilobytes of RAM



http://www.mdpi.com/1424-8220/11/6/5900?trendmd-shared=0
Sensors **2011**, *11*(6), 5900-5930; doi:10.3390/s110605900

Protothreads

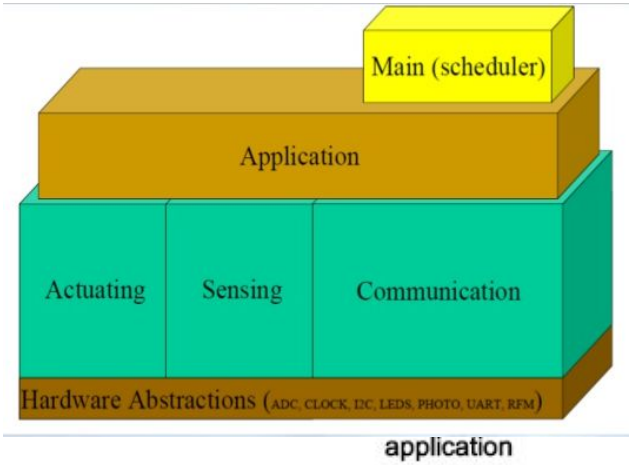
Protothreads are cooperatively scheduled. This means that a Contiki process must always explicitly yield control back to the kernel at regular intervals. Contiki processes may use a special protothread construct to block waiting for events while yielding control to the kernel between each event invocation.

- Protothreads – a new programming abstraction
  - For memory-constrained embedded systems
  - A design point between events and threads
  - Very simple, yet powerful idea
- Programming primitive: conditional blocking wait
  - PT_WAIT_UNTIL(*condition*)
  - Sequential flow of control
    - Programming language helps us: **if** and **while**
- Protothreads run on a single stack, like the event-driven model
  - Memory requirements (almost) same as for event-driven

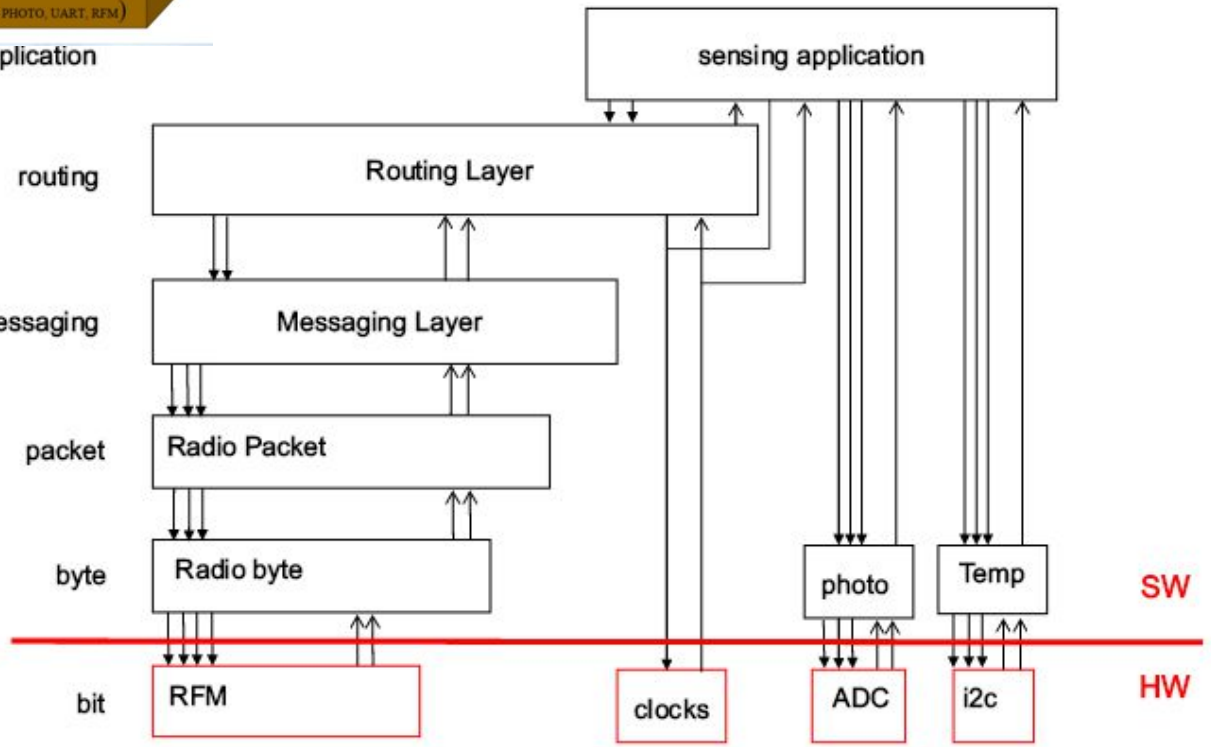
http://slideplayer.com/slide/2881229/

TinyOS

Sensor as a small computer → it needs an Operating System



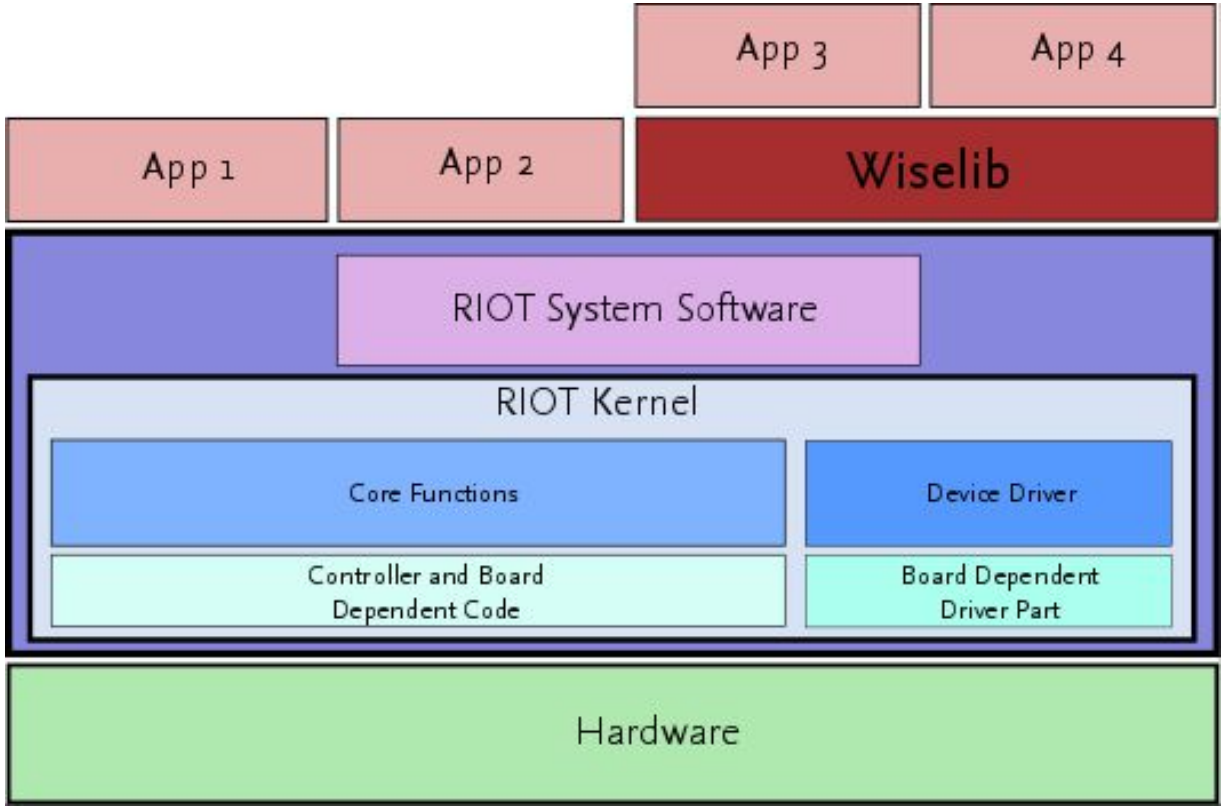
- Modular
  - Components provide simple functions by means of well defined interfaces
- Event-Based Model
- Completely not Blocking
- Tasks are not-preempitive and run in FIFO order
- It is power efficient and it makes the sensor sleep asap



http://www.slideshare.net/snecute/tinyos

RIoT

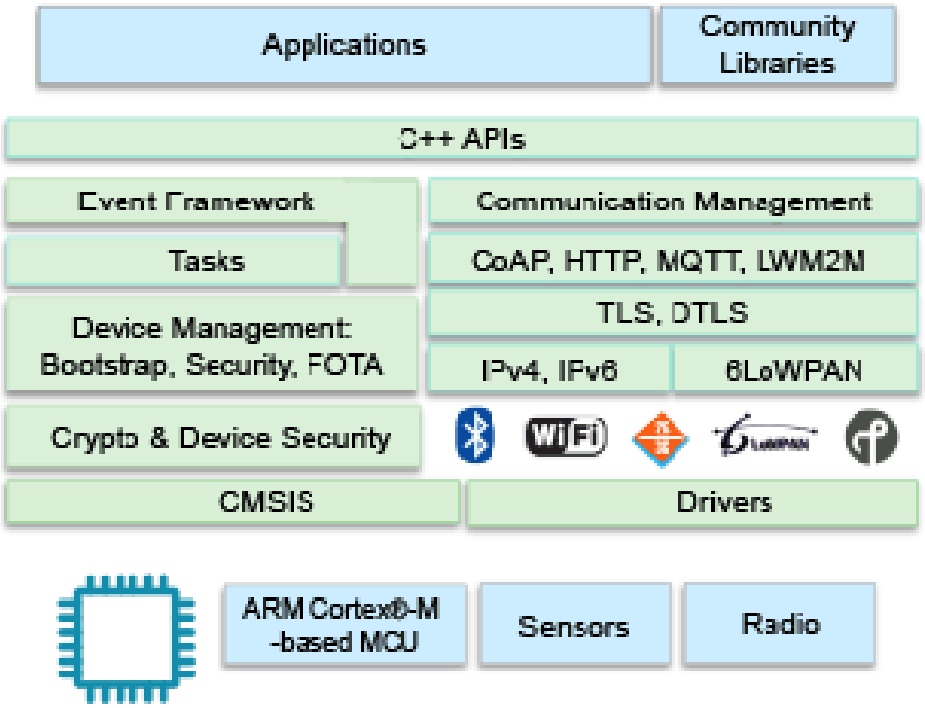
RIOT is a microkernel based architecture for distributed embedded systems with a strong focus on the Internet of Things Wireless Sensor Networks. The kernel is real-time capable and supports full multi-threading. Furthermore, it provides mutexes as well as synchronous and asynchronous message passing for interprocess communication. The scheduler is priority-based and uses a tickless timer system.



ARM Operating System for IoT

mbed OS Roadmap 2015

- Minimize time-to-market
- Low-power by design
- Complete security solution
- Top connectivity standards
- Built-in device management



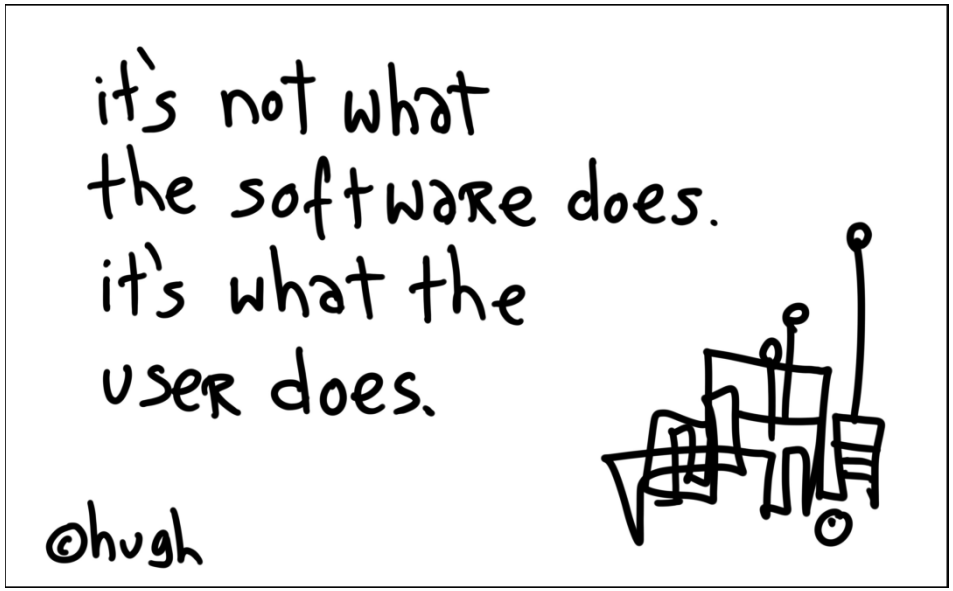
11

http://hexus.net/tech/news/software/75357-arm-looks-supercharge-internet-things-mbed-os/

Module 17: IoT Operating Systems

- What are the feature of an IoT operating system?
- Do you find similarities between the presented solutions? And what are the major differences?
- Can you relate these Operating System to Middleware? Are they the same?

- The software for (Vertical and Horizontal Middleware) IoT

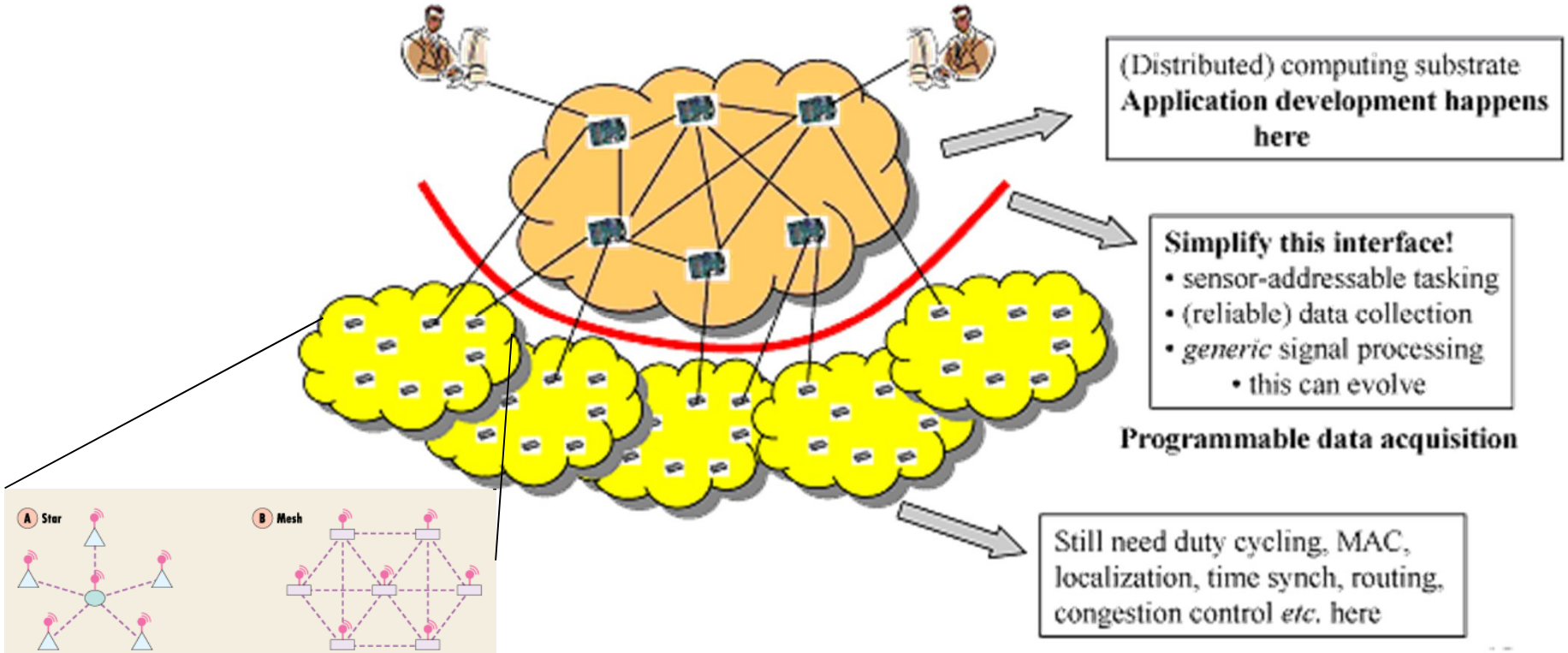


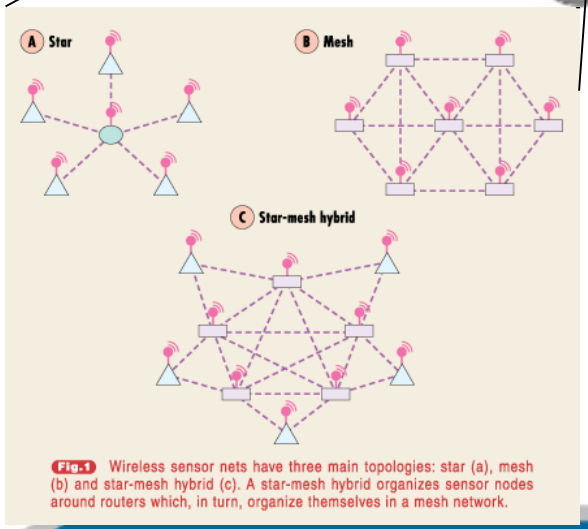
A further Step to Platforms

Name	Application/ Data Layer	Transport Layer	Network Layer	DataLink/MAC Layer	PHY Layer	OS	Wireless Communication	Programming Language
ALLSEEN ALLIANCE	Transport independent	TCP/IP, UDP, local UNIX transport	6LoWPAN, ZigBee	WiFi, WiFi-Direct, Ethernet, BLE, Serial/ SLIP	Powerline (PLC)	RTOS, Arduino, Linux, Android, iOS, OS X, Windows, OpenWRT, Unity game development	WiFi, GPRS, UMTS, Bluetooth, DECT, ZigBee, Z-Wave, ONE-NET, EnOcean, Infrared (Consumer IR), Insteon	C++, Objective C, C#, Java, JavaScript
OPEN INTERCONNECT	CoAP, JSON, CBOR, DTLS	UDP	IPv4, IPv6, 6LoWPAN	Bluetooth, BLE		Linux, Android, Tizen, Arduino	IEEE 802.11 WiFi, Bluetooth, BLE	C, C++, Java
eclipse	MQTT, CoAP, OMA-DM, OMA LWM2M					Linux, Windows, OS X, Solaris		C, C++, Java, JavaScript, Ada, ABAP, COBOL, Fortran, Haskell, Lasso, Lua, Natural, Perl, PHP, Prolog, Python, R, Ruby, Scala, Clojure, Groovy, Scheme, Erlang

http://www.slideshare.net/AllSeenAlliance/wearables-and-iot-strategy

Programming Wireless Sensor Networks





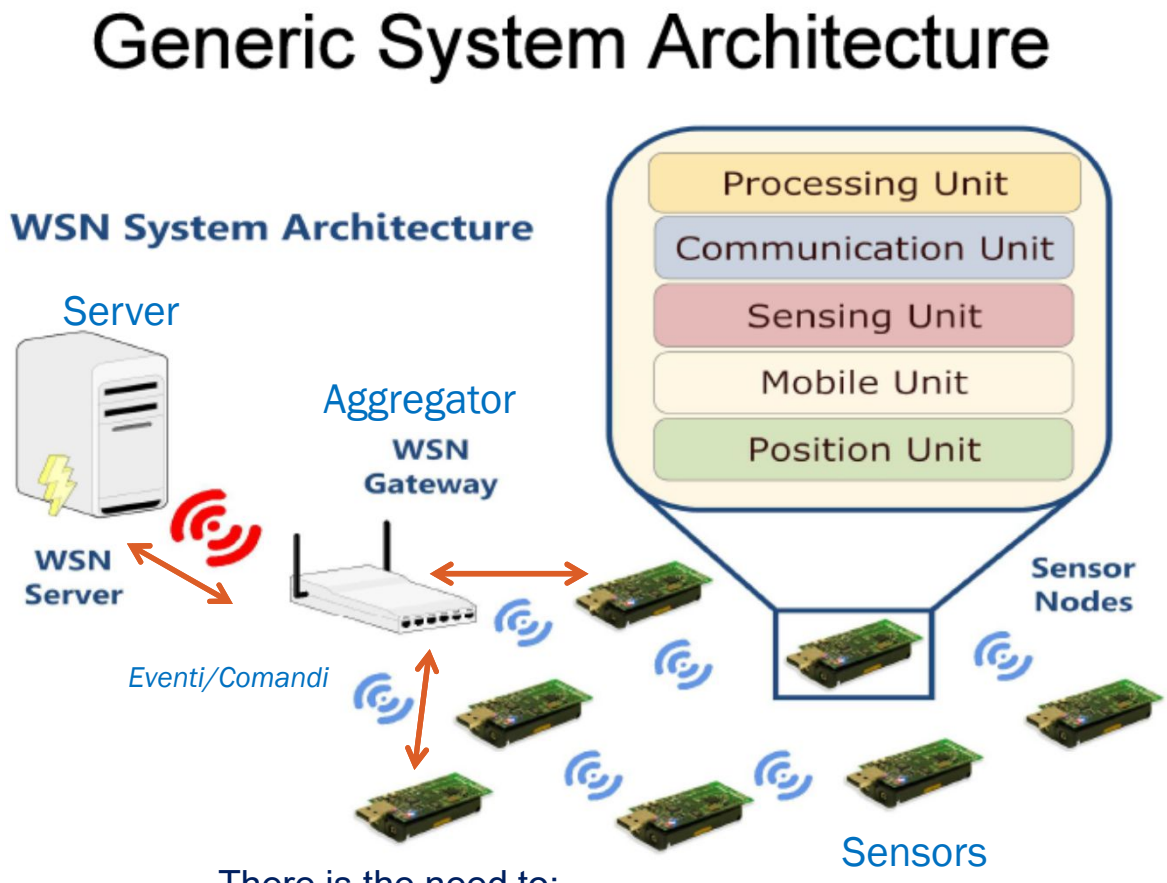
Each Node (a mote) is a small computer
A back-end platform for programmability

http://i.cmpnet.com/commsdesign/csd/2003/jul03/feat1-jul03-fig1.gif

http://research.cens.ucla.edu/projects/2006/Systems/Tenet/figure_1.gif

IEEE Advancing Technology for Humanity

The Typical IoT Setup

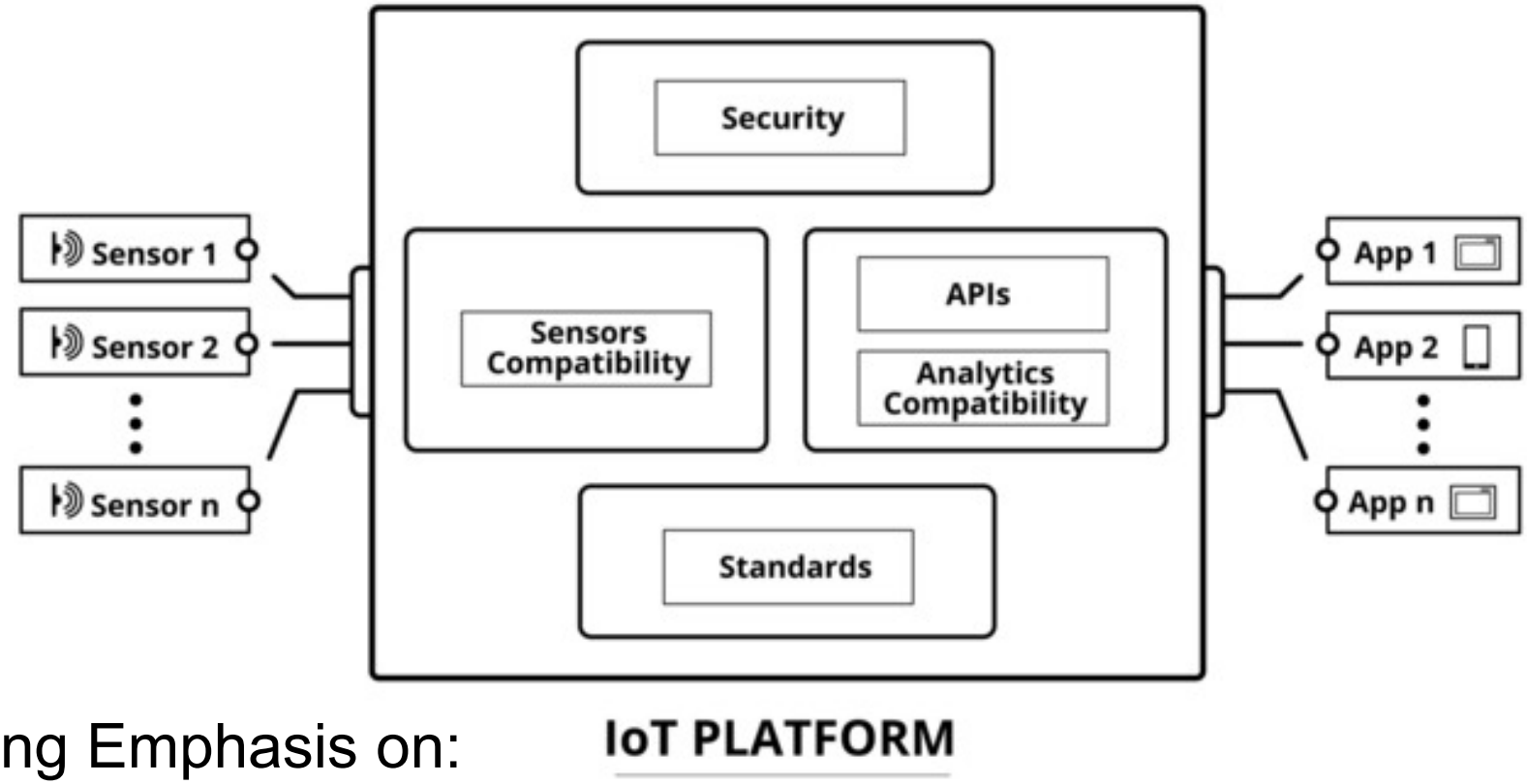


- Sensors, smart things will produce many data
- Proprietary solutions based on different standards or no standard at all
- Great Local Complexity
- Service Center closed and proprietary (Vertical Solutions)

There is the need to:

- Decouple the local solutions form the in-network one (from a technical and business perspective – Virtualization and standardization)
- Integrate data produced by sensors with the User Data Space

Towards IoT Platforms



Strong Emphasis on:

- Programmability
- Security
- Data Analytics

http://www.networkworld.com/article/2687240/internet-of-things/internet-of-things-roundtable-experts-discuss-what-to-look-for-in-iot-platforms.html

IoT Eclipse

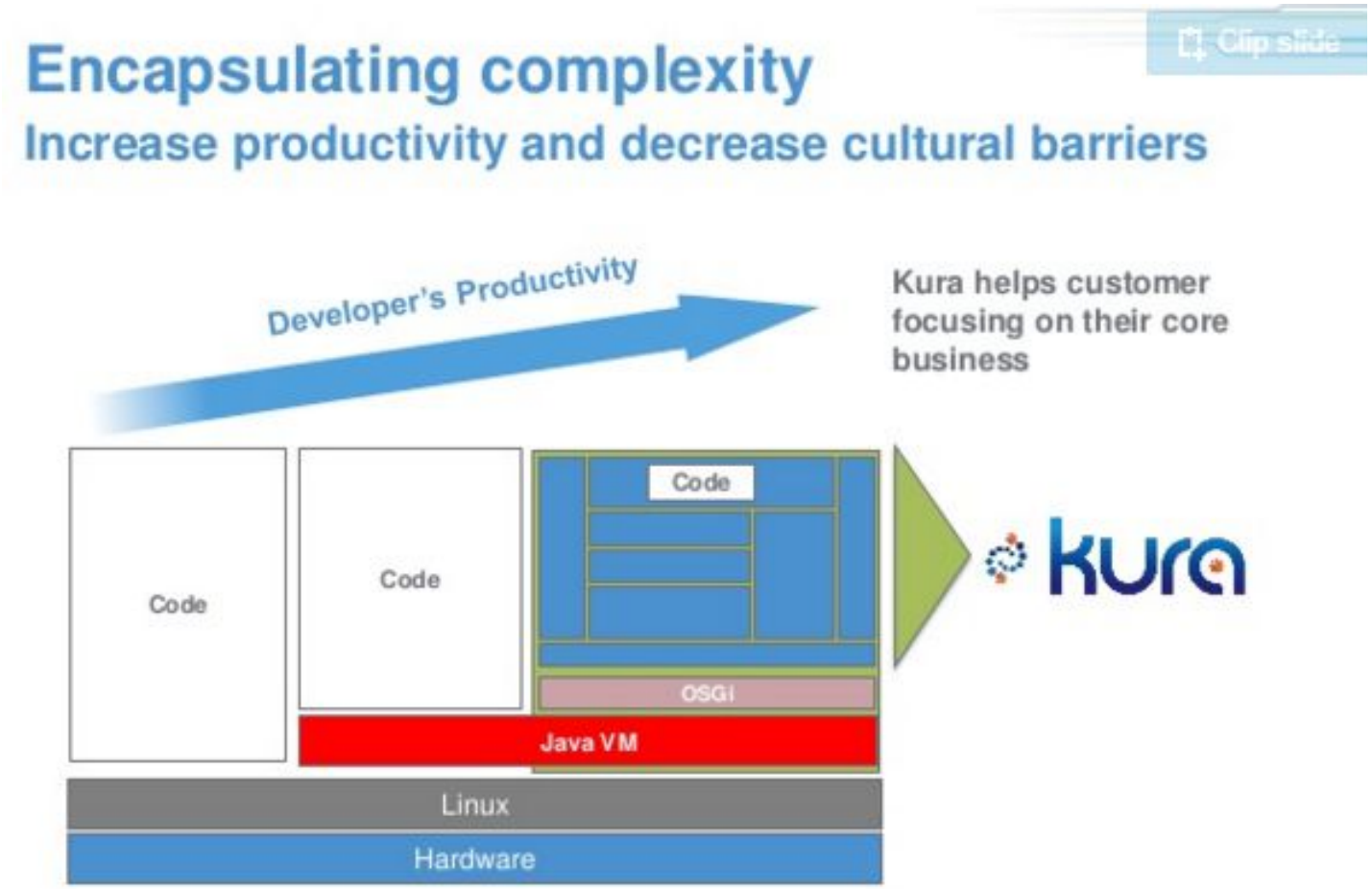
Clip slide

KURA is the open source Java and OSGi-based Application Framework for M2M Service Gateways in the Eclipse IOT Working Group.

Purpose
Simplify the design, deployment and remote management of embedded applications.

It provides
- Cohesive and integrated app environment
- Modular software components
- HW abstraction layer
- Field protocol libraries
- Cloud connectivity
- Remote app and device management
- Local app and device management
- Built-in Security
- Development tools

Programmable Architecture Kura

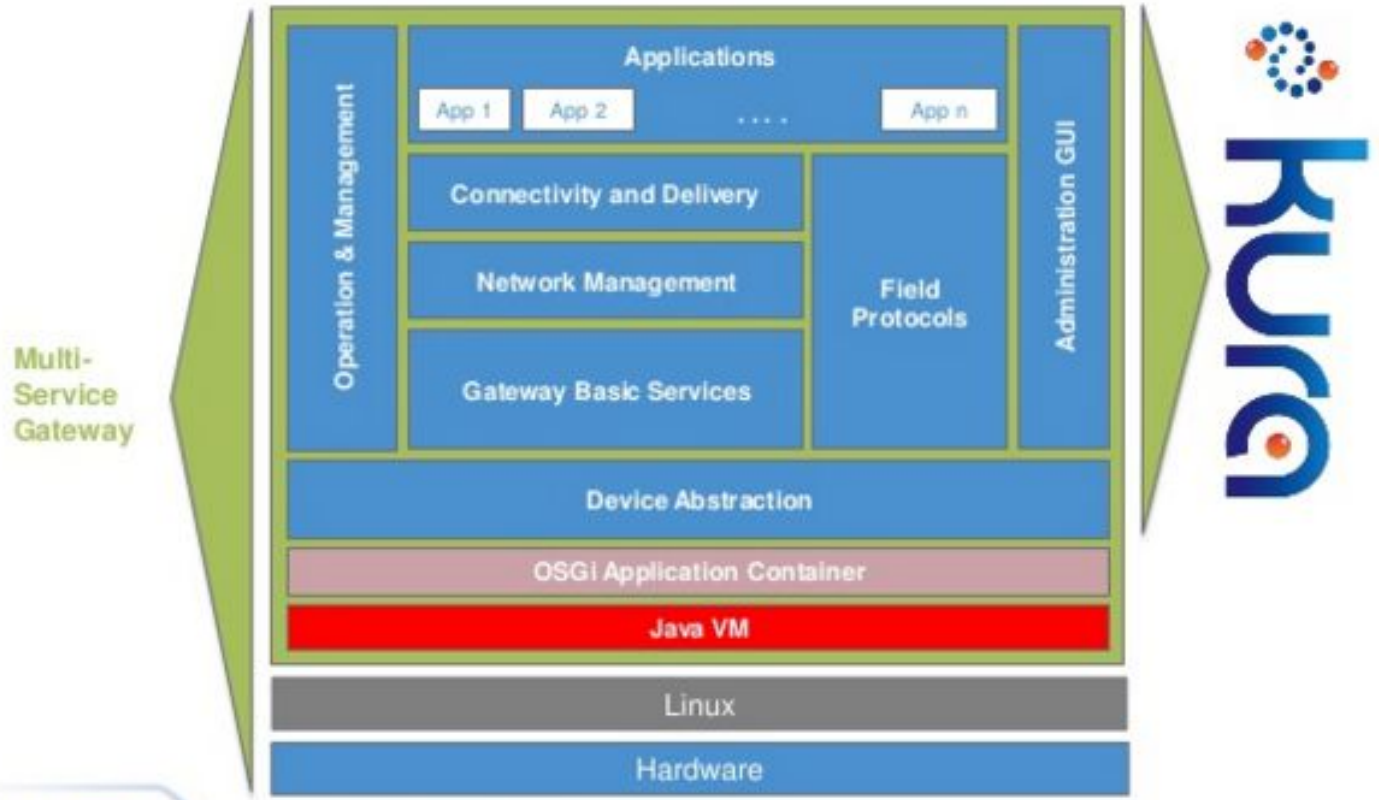


http://www.slideshare.net/Eurotechchannel/kuram2miotgateway

IoT Eclipse

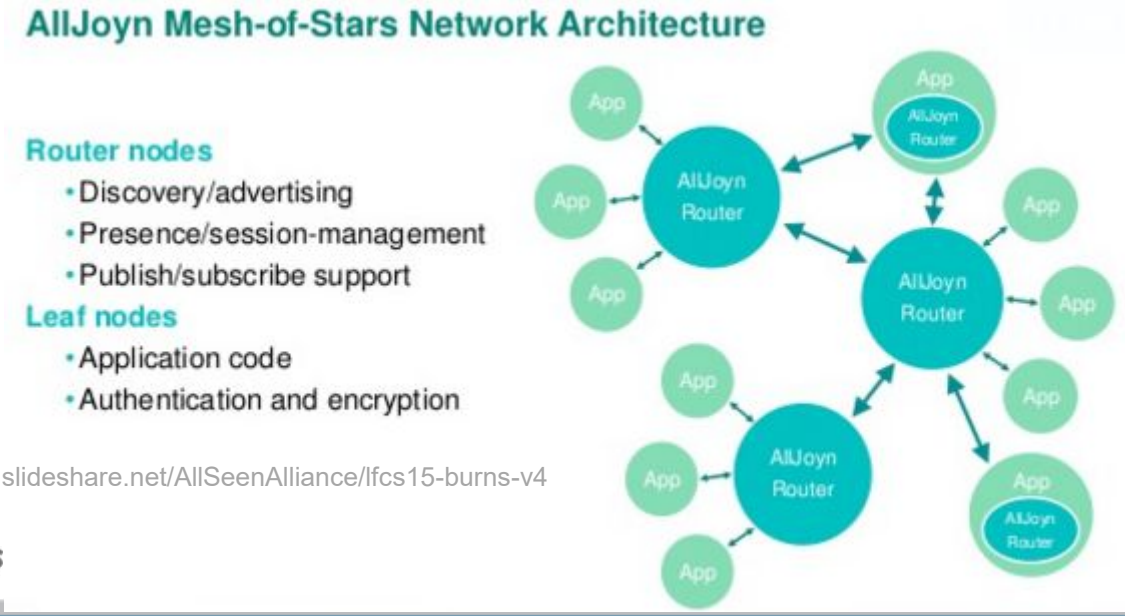
Functional Architecture

Decoupling functional layers



Allseen Architecture – based on alljoyn

From peer to peer communication to IoT architecture



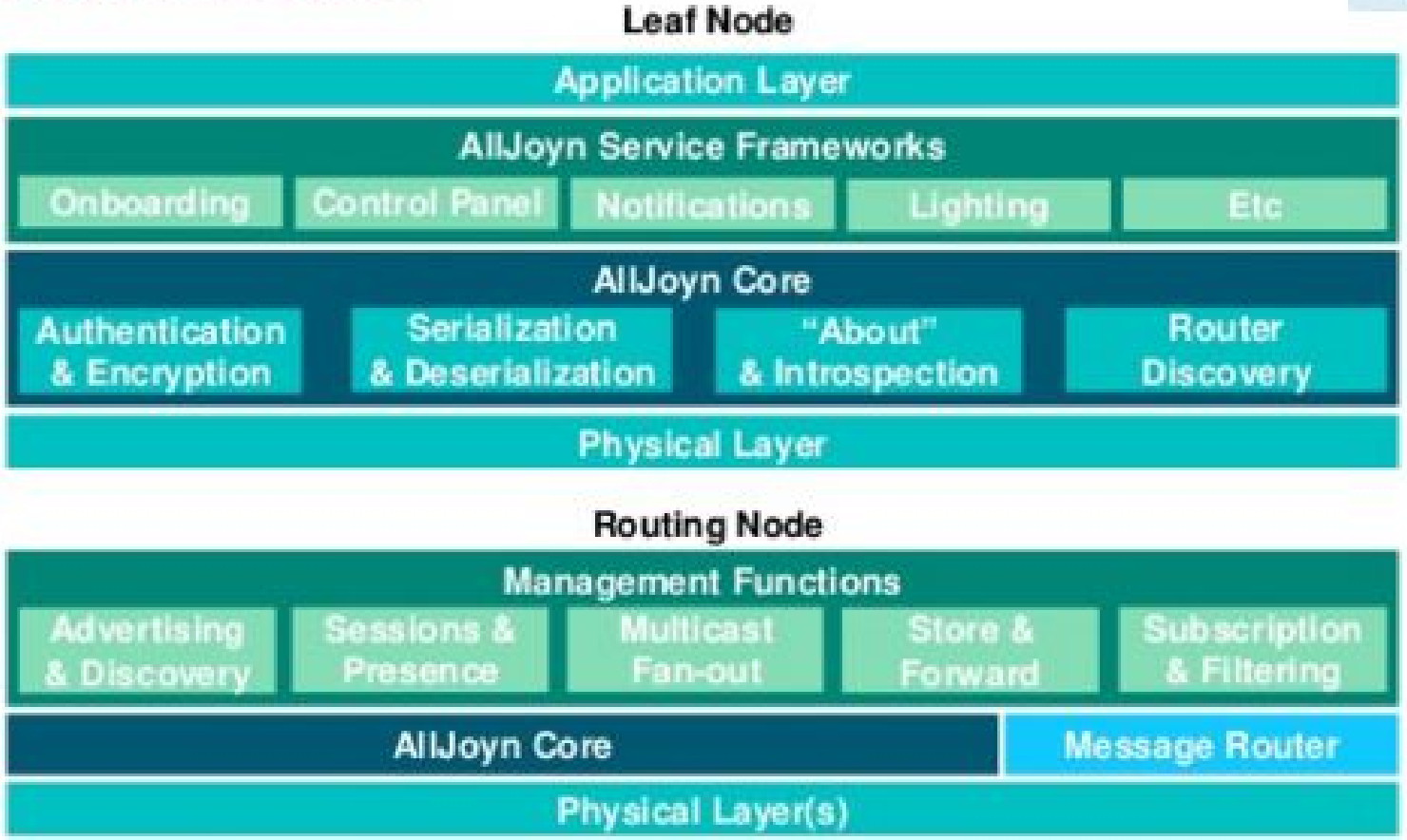
http://www.slideshare.net/AllSeenAlliance/lfcs15-burns-v4

IEEE Internet of Things

IEEE Advancing Technology for Humanity

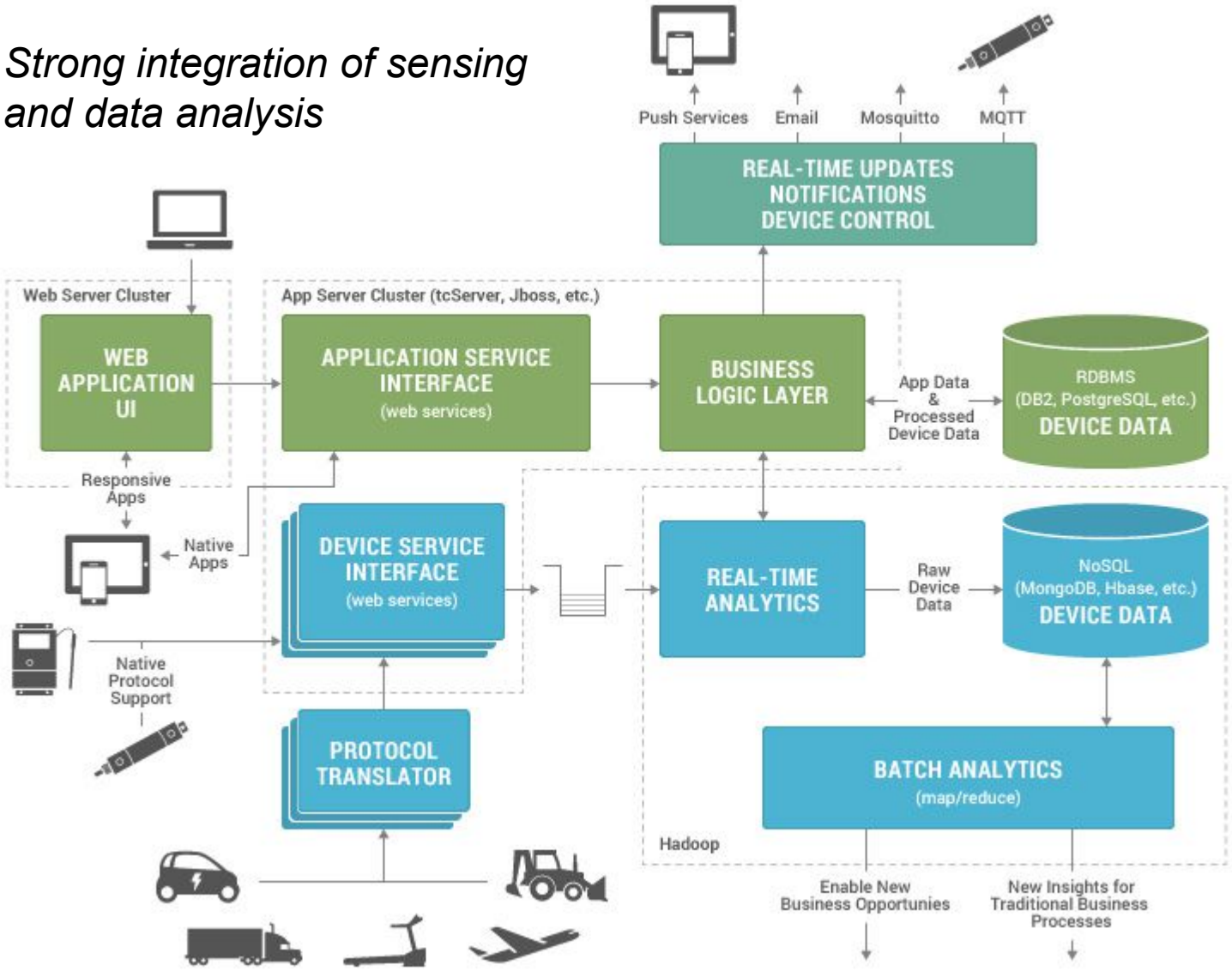
Allseen

Software Architecture



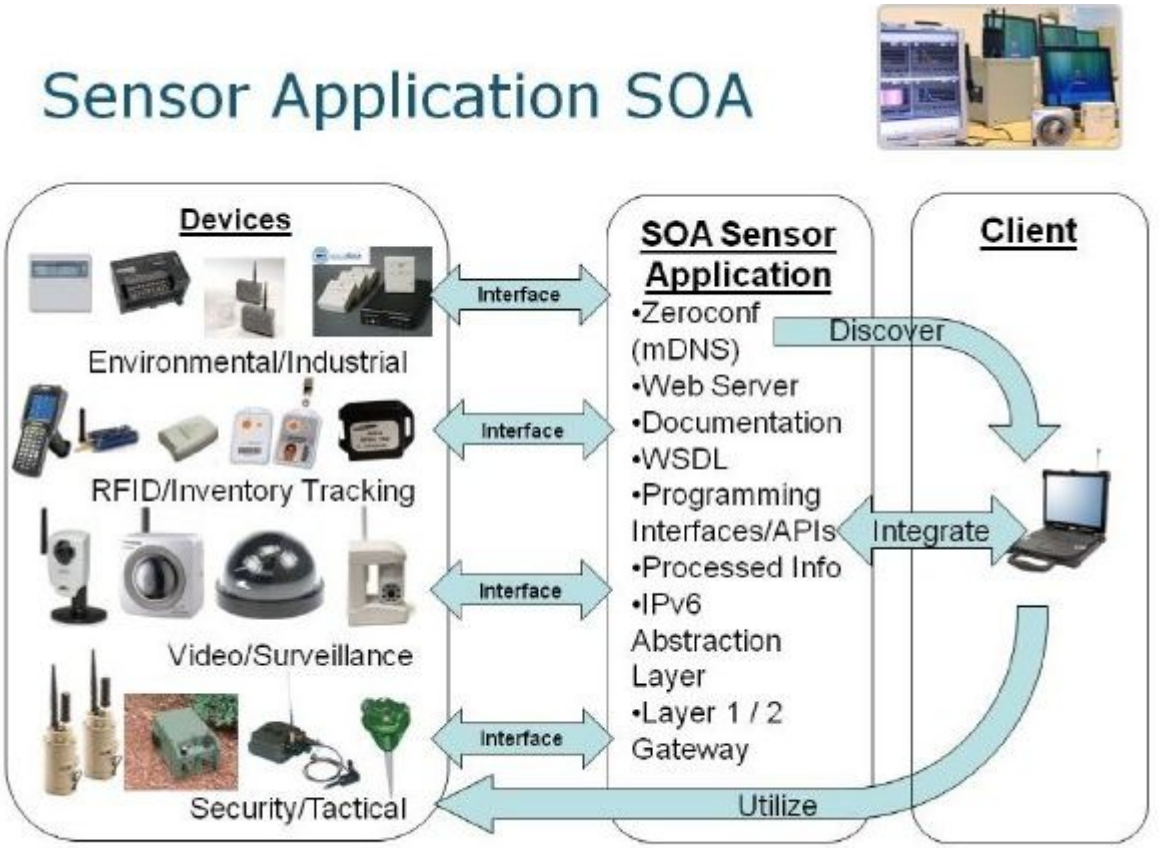
Towards IoT Platforms

Strong integration of sensing and data analysis



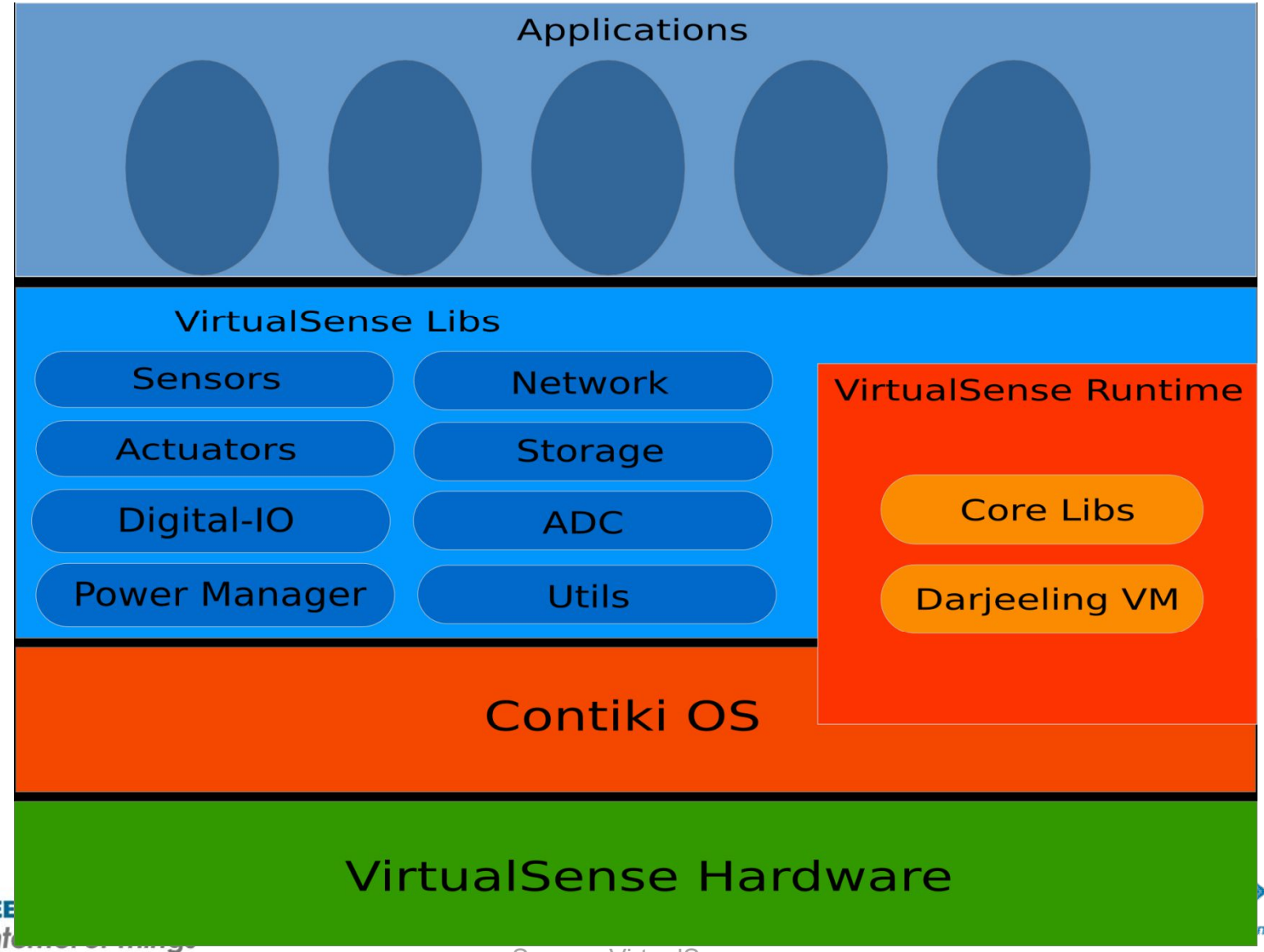
http://www.stagrp.com/internet-of-things/

Web Services and Sensors



http://www.commandinformation.com/blog/?p=73

Virtual Sense

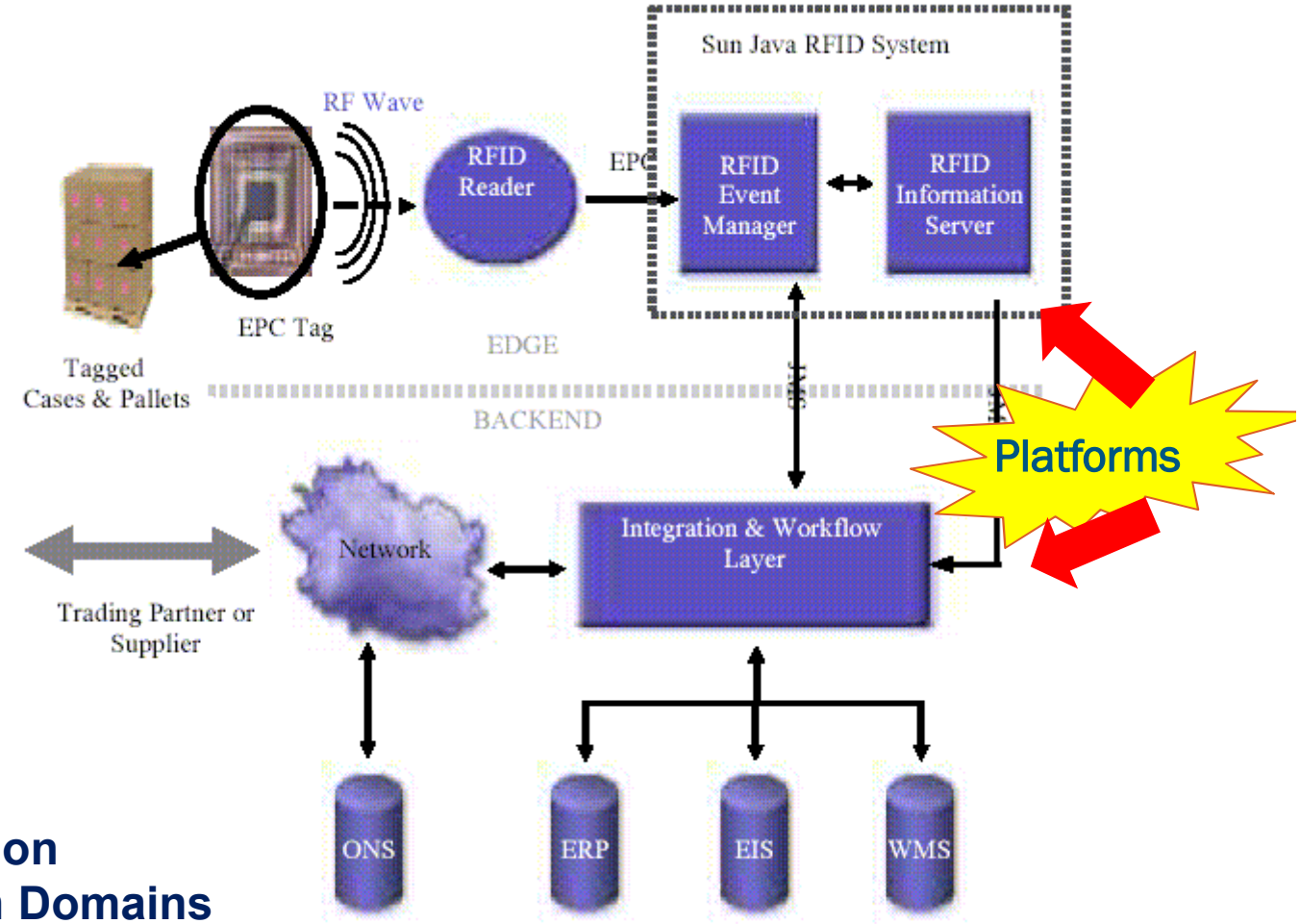


Source: VirtualSense

An example - EPC Global

Applications:

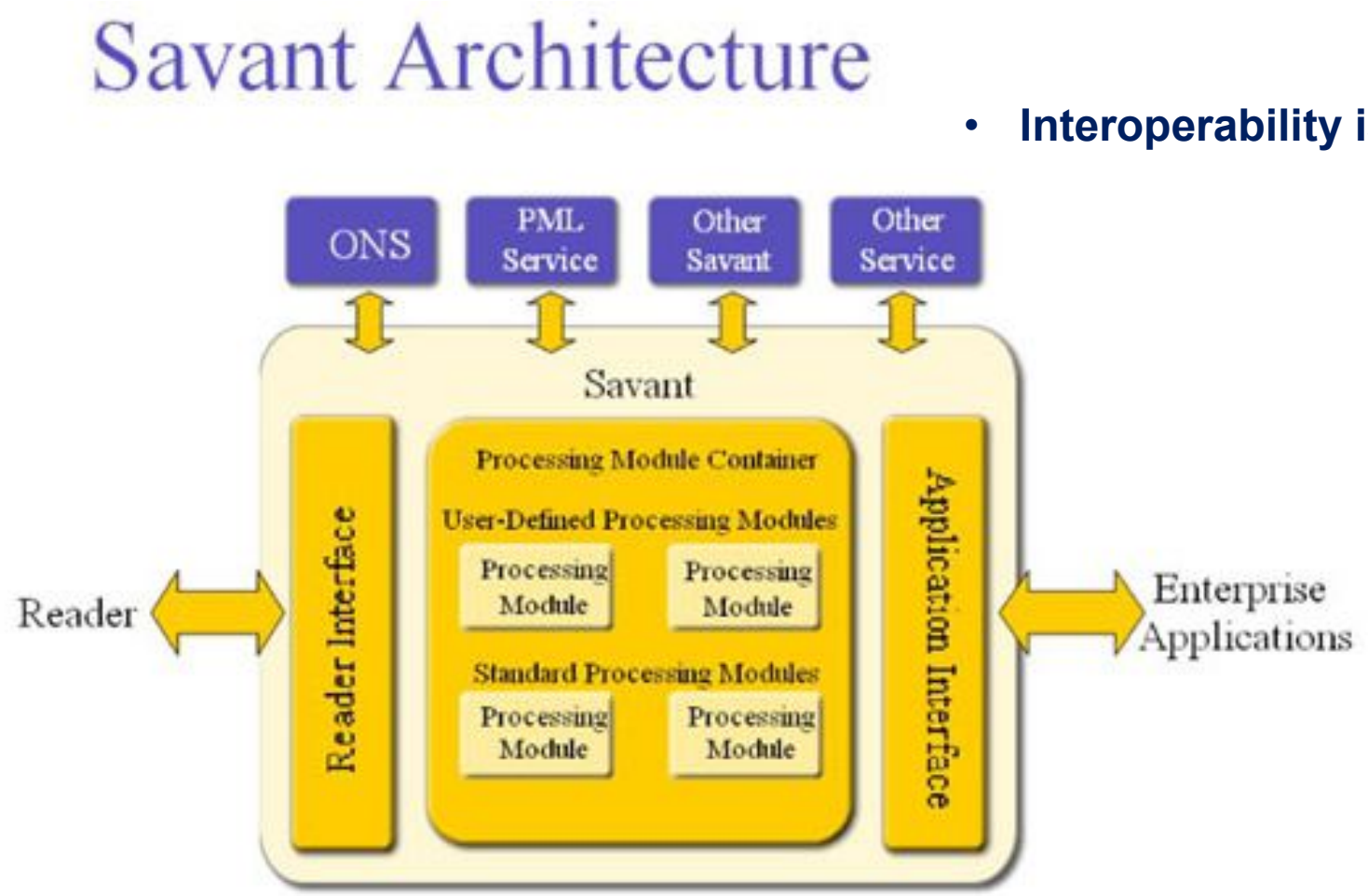
- **Logistics**
- **Supply chain**
- **Consumer Electronics**
- **Defense and Aerospace**
- **Media & Entertainment**
- **Automotive**
- **…**



- **Very focused solution**
- **Vertical Application Domains**

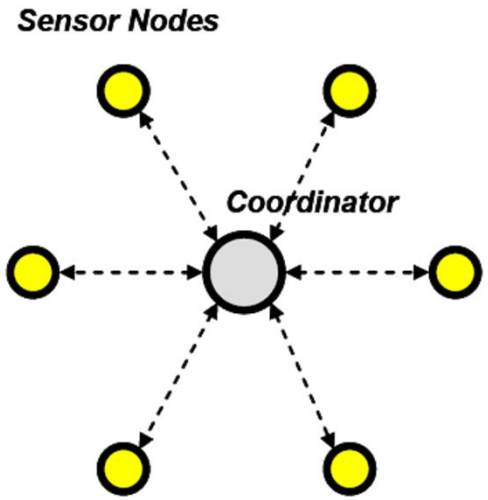
http://java.sun.com/developer/technicalArticles/Ecommerce/rfid/sjsrfid_epcnetwork.gif

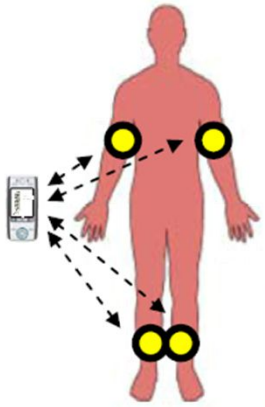
A Software Architecture for EPC Global

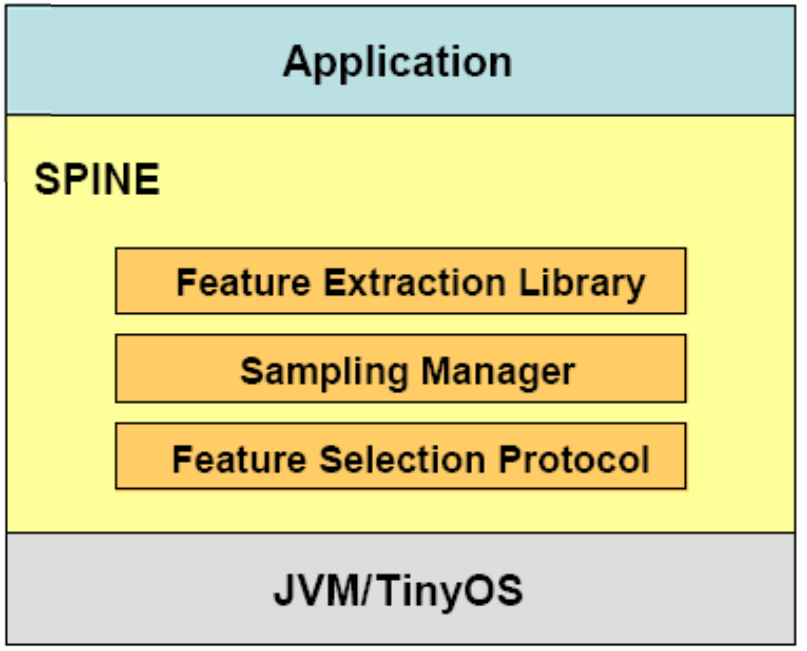


- **Interoperability issues**

Signal Processing in Node Environment (SPINE)







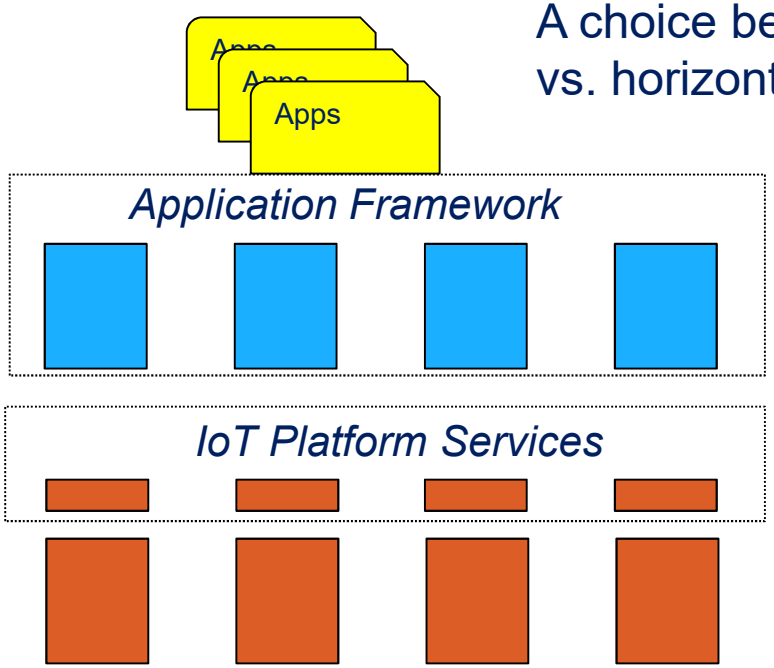
http://spine.tilab.com/papers/2007/WhitePaper.pdf

Module 18: IoT platforms

- What are the advantages of the platforms? How do they compare to Operating systems?
- Are they specific or have a general applicability ?
- List the IoT platform that seems to you more general

IOT MIDDLEWARE

The Software Issue: What Platform for IoT Networks (the middleware and softwarization challenge)



A choice between vertical and specialized platforms vs. horizontal and general purpose ones

A Framework for developing Applications

Distributed OS (comprising Local OSes)

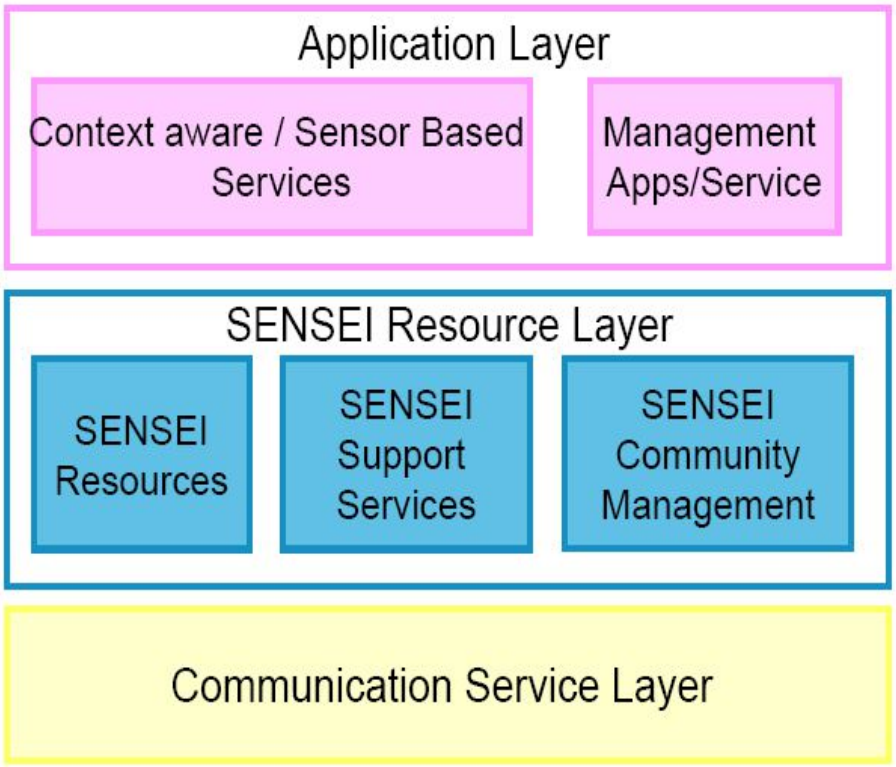
Sensors / Actuators/ Smart Objects

Sensor as a small computer ➔ it needs an Operating System providing for basic functions

Mobile Sensor API is an example of middleware service for Wireless Sensor Networks

Obviously there are many OSes for IoT (e.g., Contiki, …)
Many European Projects are working towards this vision

High level overview

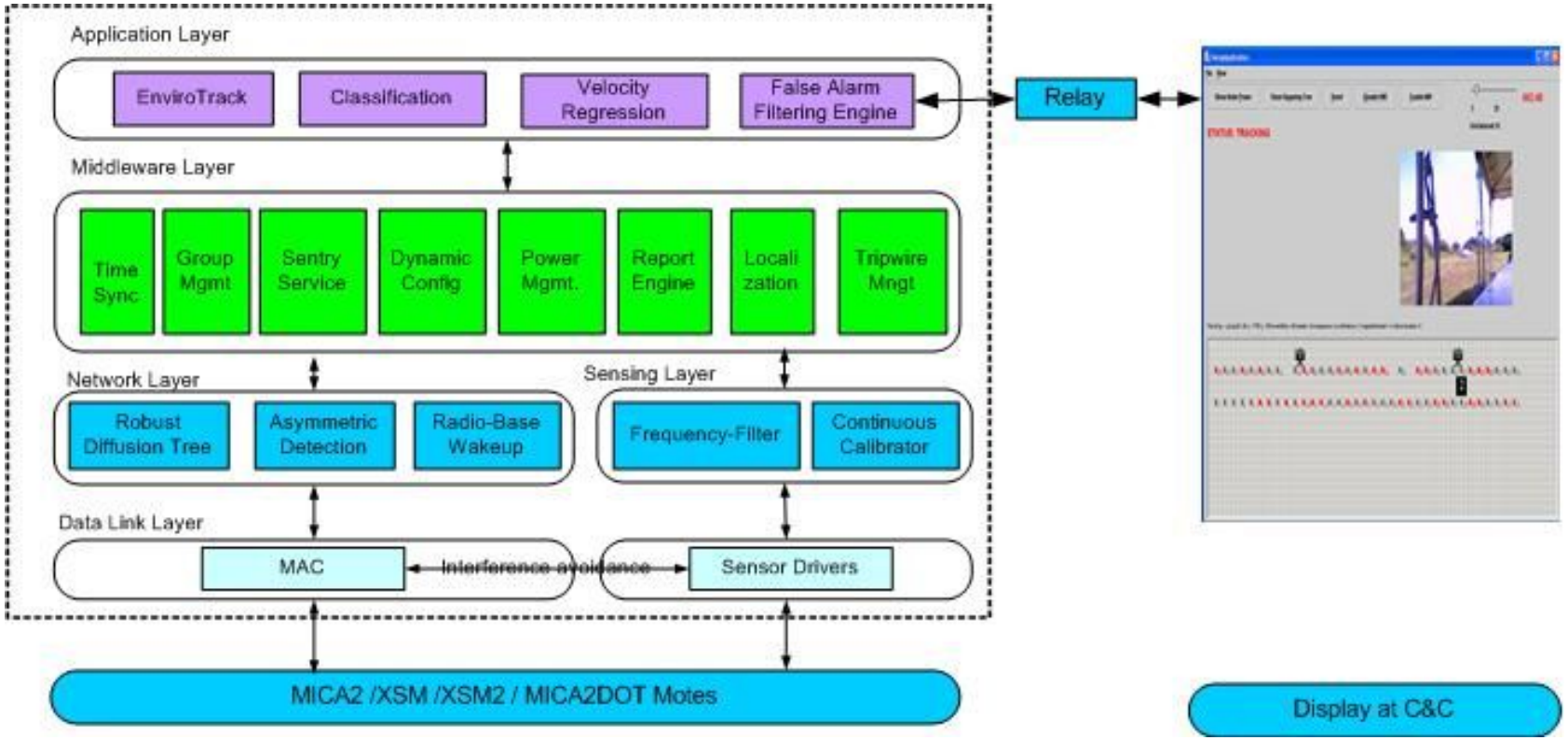


Layering as a fundamental architectural principle

SENSEI focus

A few Layering Principles emerging

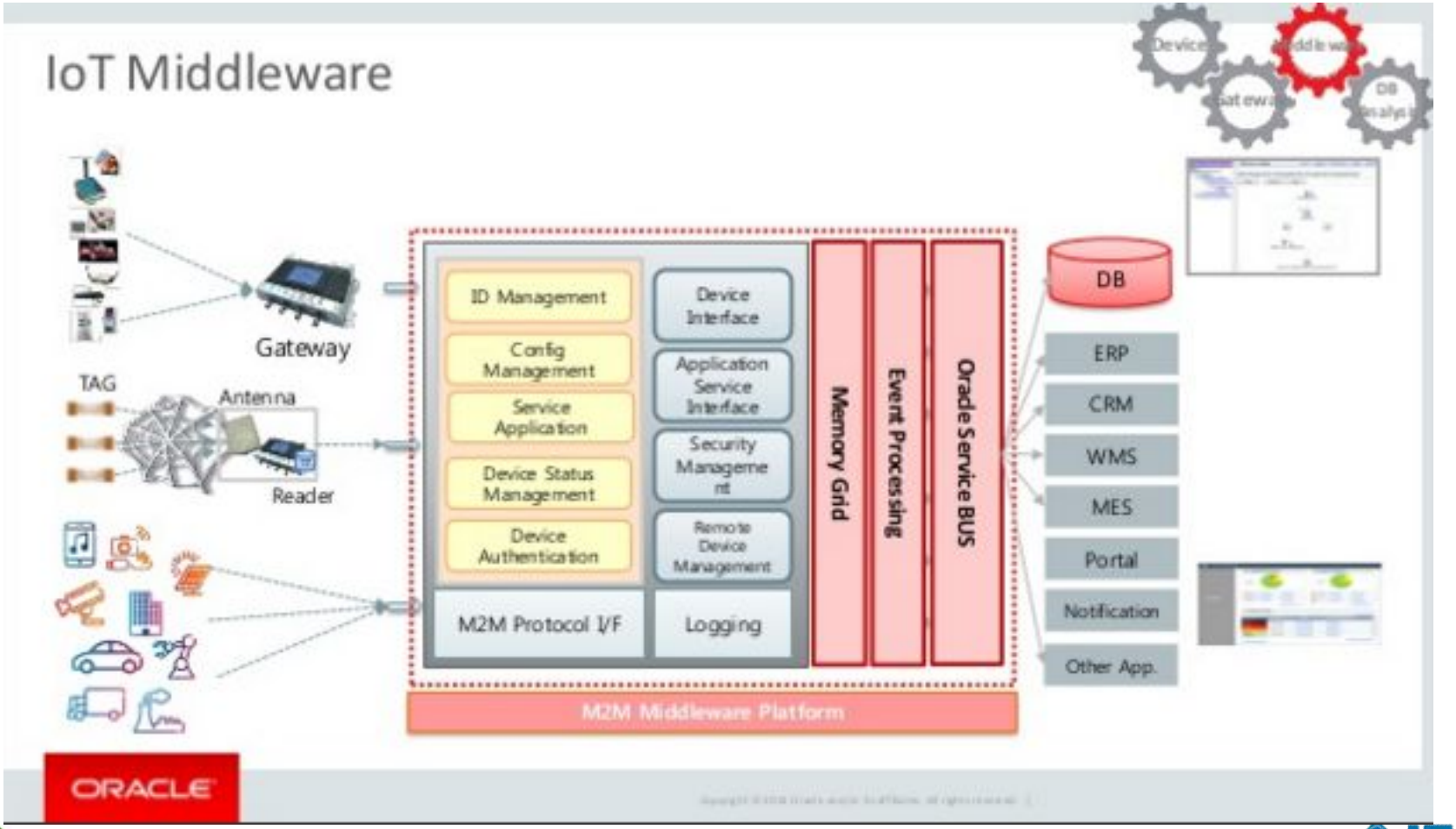
Middleware for IoT



https://www.cs.virginia.edu/wsn/vigilnet/

- Layering
- Components and APIs

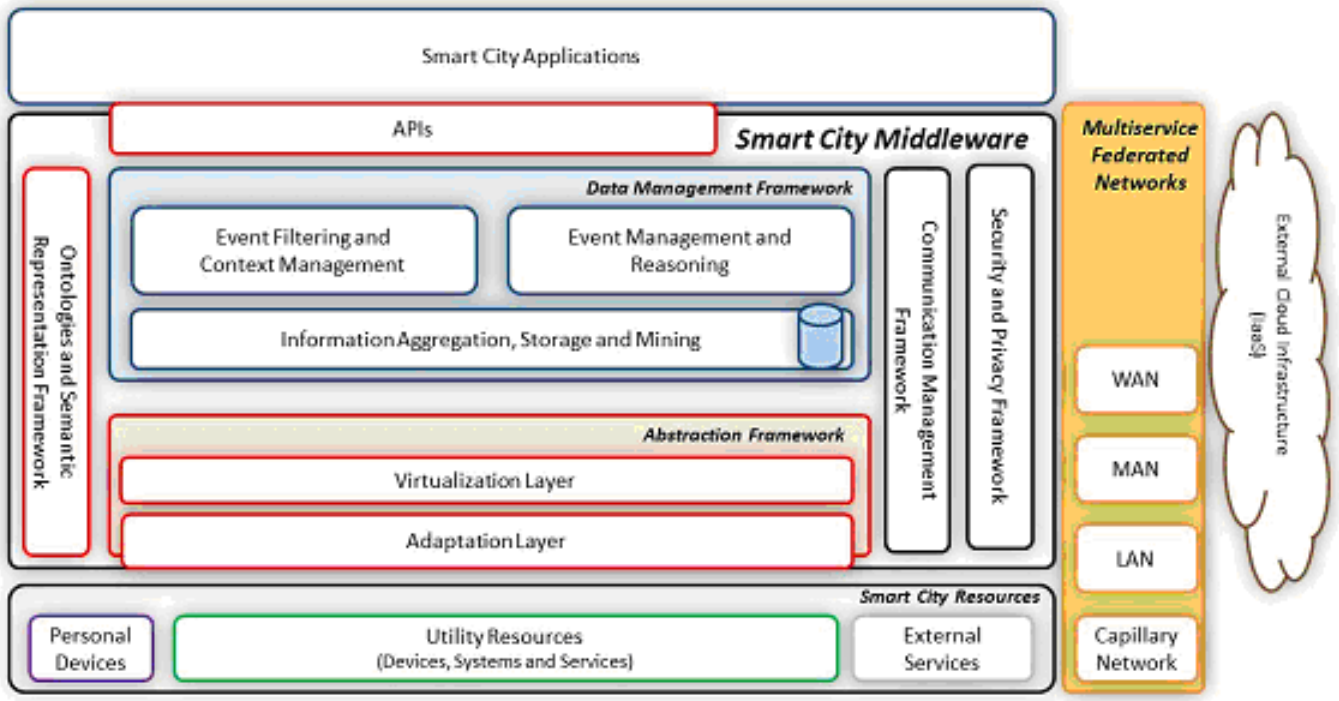
Oracle's view on Middleware for IOT



http://www.slideshare.net/junsukseo946/0-2-oracle

ALMANAC Smart City Platform architecture

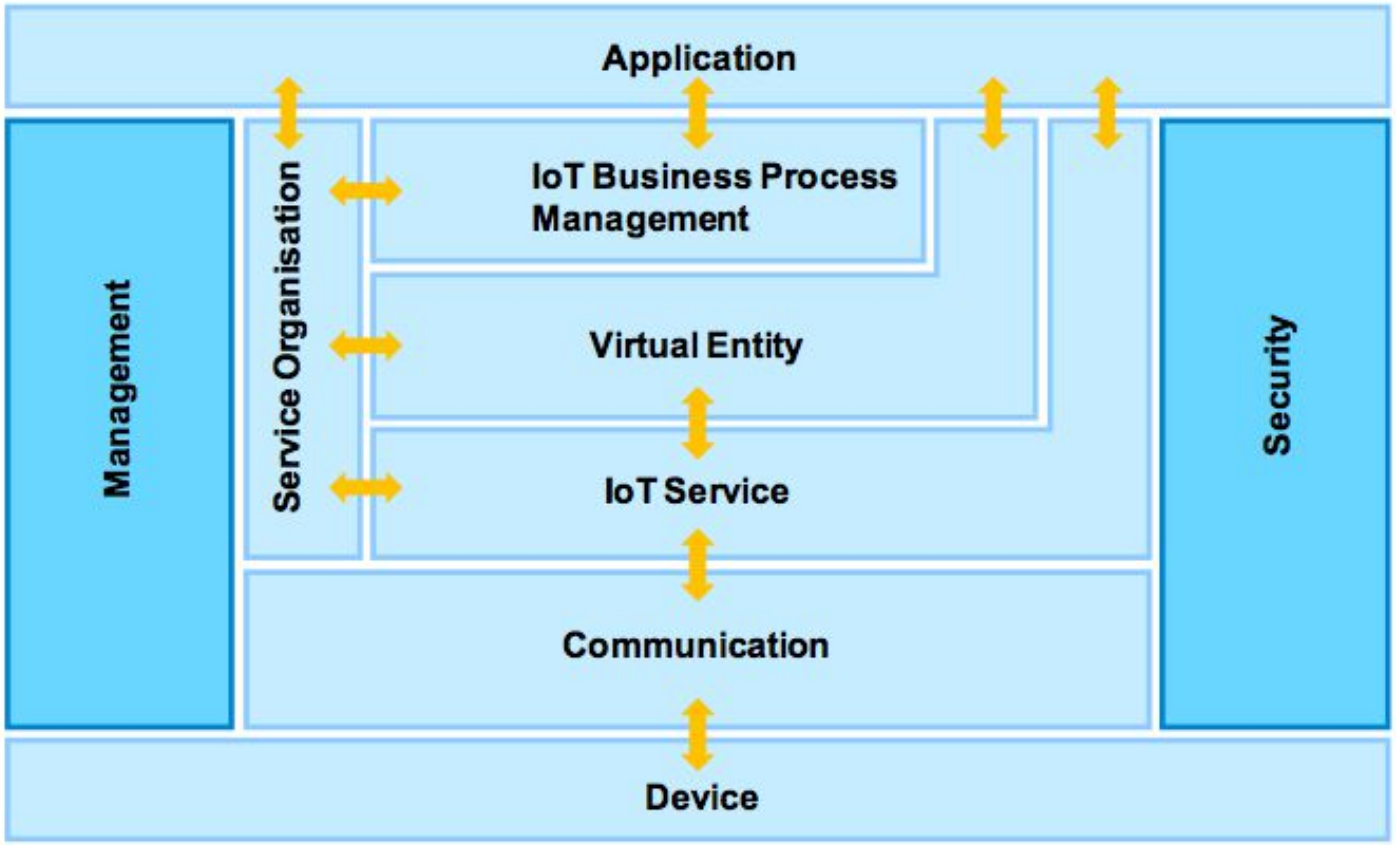
- Information aggregation
- Virtualization



http://www.in-jet.dk/en/articles.php?article_id=24

IEEE
Advancing Technology for Humanity

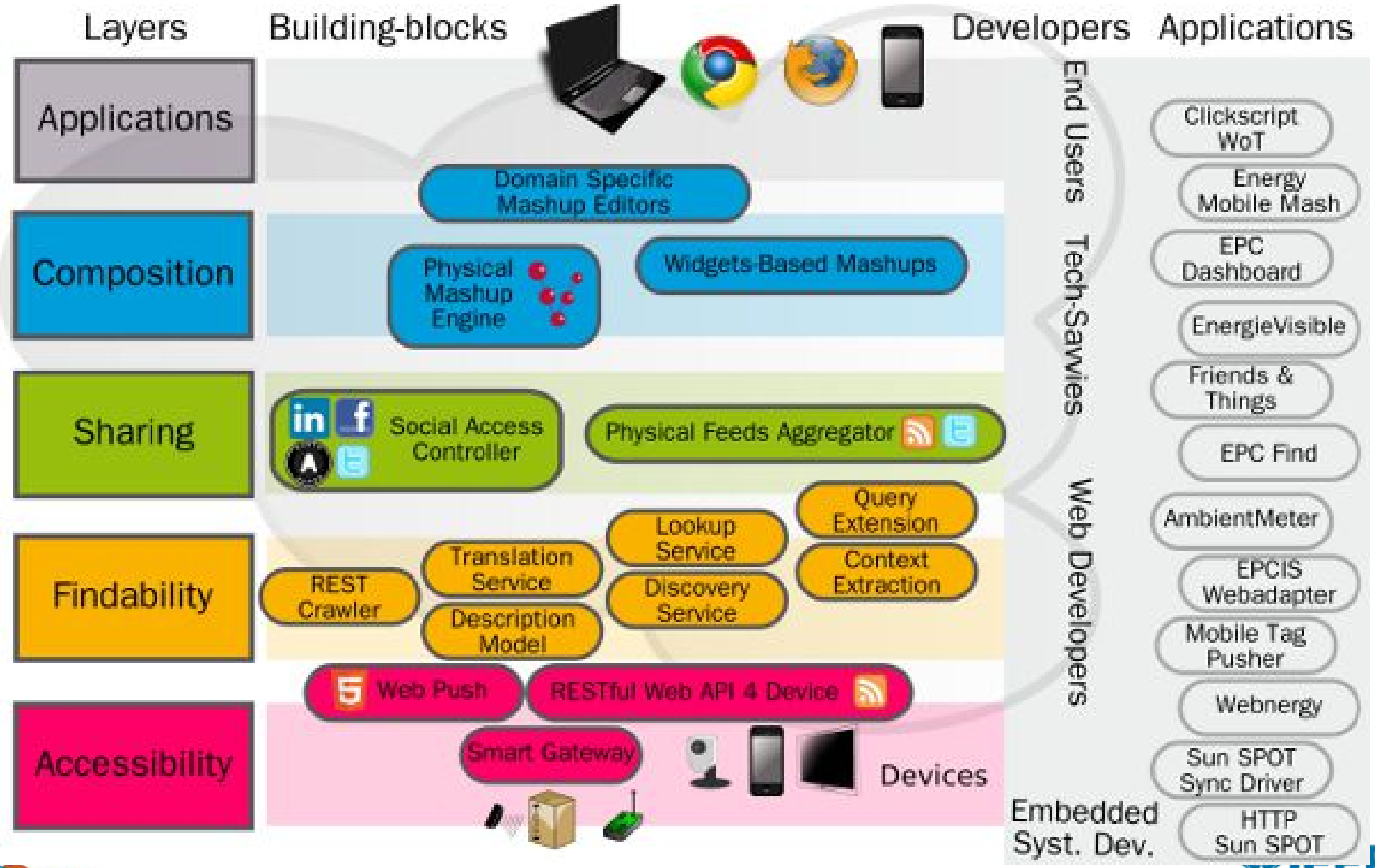
IoT-A Architecture



- Virtualization
- Management
- Security
- Orchestration

http://iot-datamodels.blogspot.it/2012/09/data-models-for-internet-of-things-5.html

Web of Things

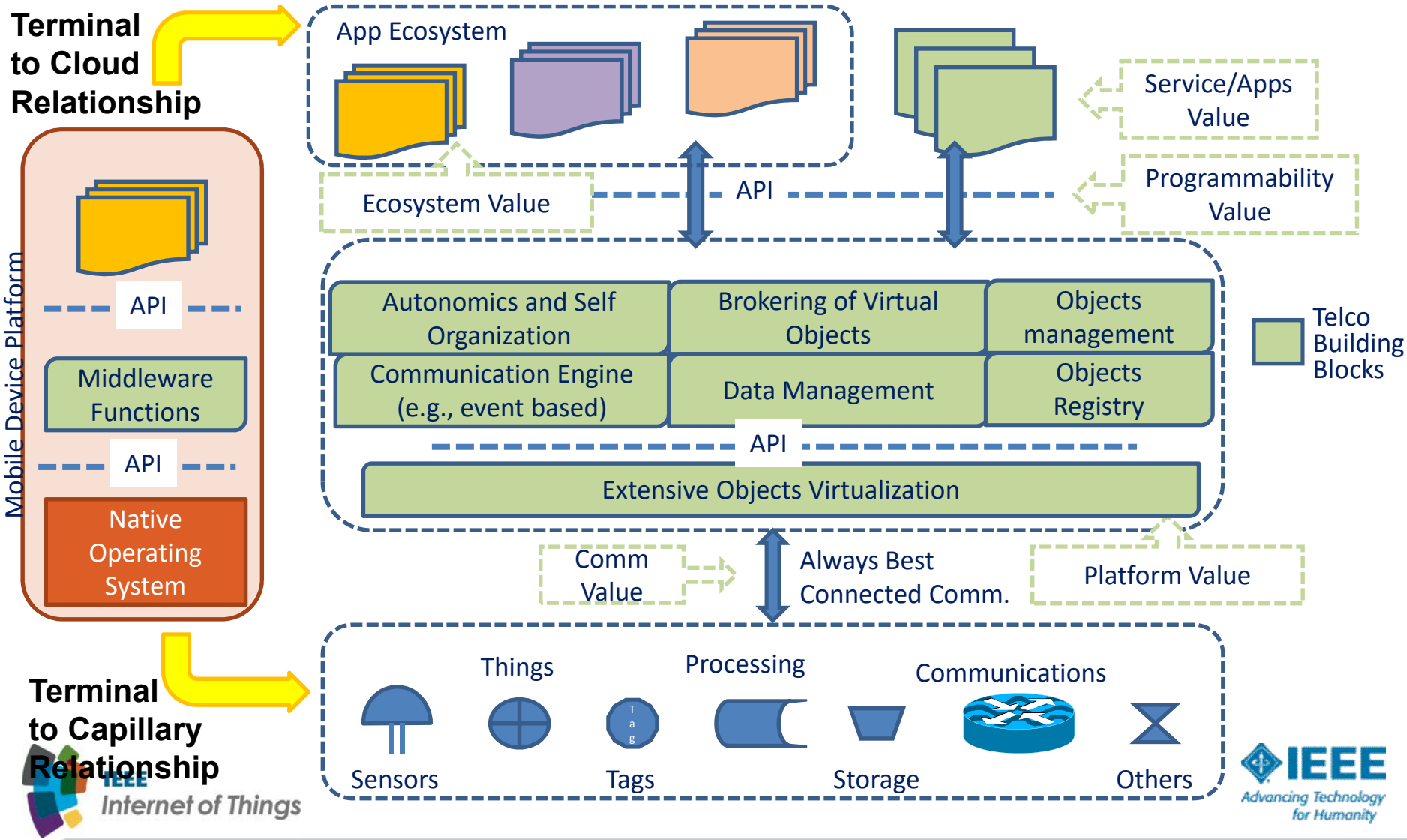


IEEE Internet of Things

http://webofthings.org/2011/12/01/phd-web-of-things-app-archi/

IEEE Advancing Technology for Humanity

An IoT Middleware View



Module 19: IoT Middleware platforms

- What are the services offered by relevant IoT middleware?
- What are their major differences?
- Can you relate the middleware platform to a specific communication paradigm?

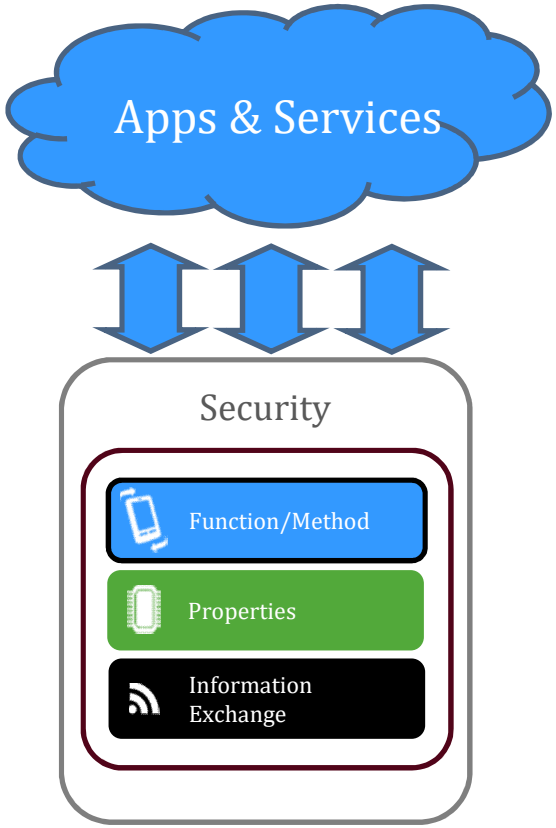
STANDARDS

IEEE P2413 Scope

- This standard defines an Architectural Framework for the IoT, including descriptions of various IoT domains, definitions of IoT domain abstractions, and identification of commonalities between different IoT domains.
- The Architectural Framework for IoT provides:
  - reference model that defines relationships among various IoT domains (e.g., transportation, healthcare, etc.) and common architecture elements
  - reference architecture that:
    - builds upon the reference model
    - defines basic architectural building blocks and their ability to be integrated into multi-tiered systems
    - addresses how to document and mitigate architecture divergence.
  - blueprint for data abstraction and the quality "quadruple" trust that includes protection, security, privacy, and safety.

IEEE P2413 Definitions

- The Group accepted the definition of the "Thing":

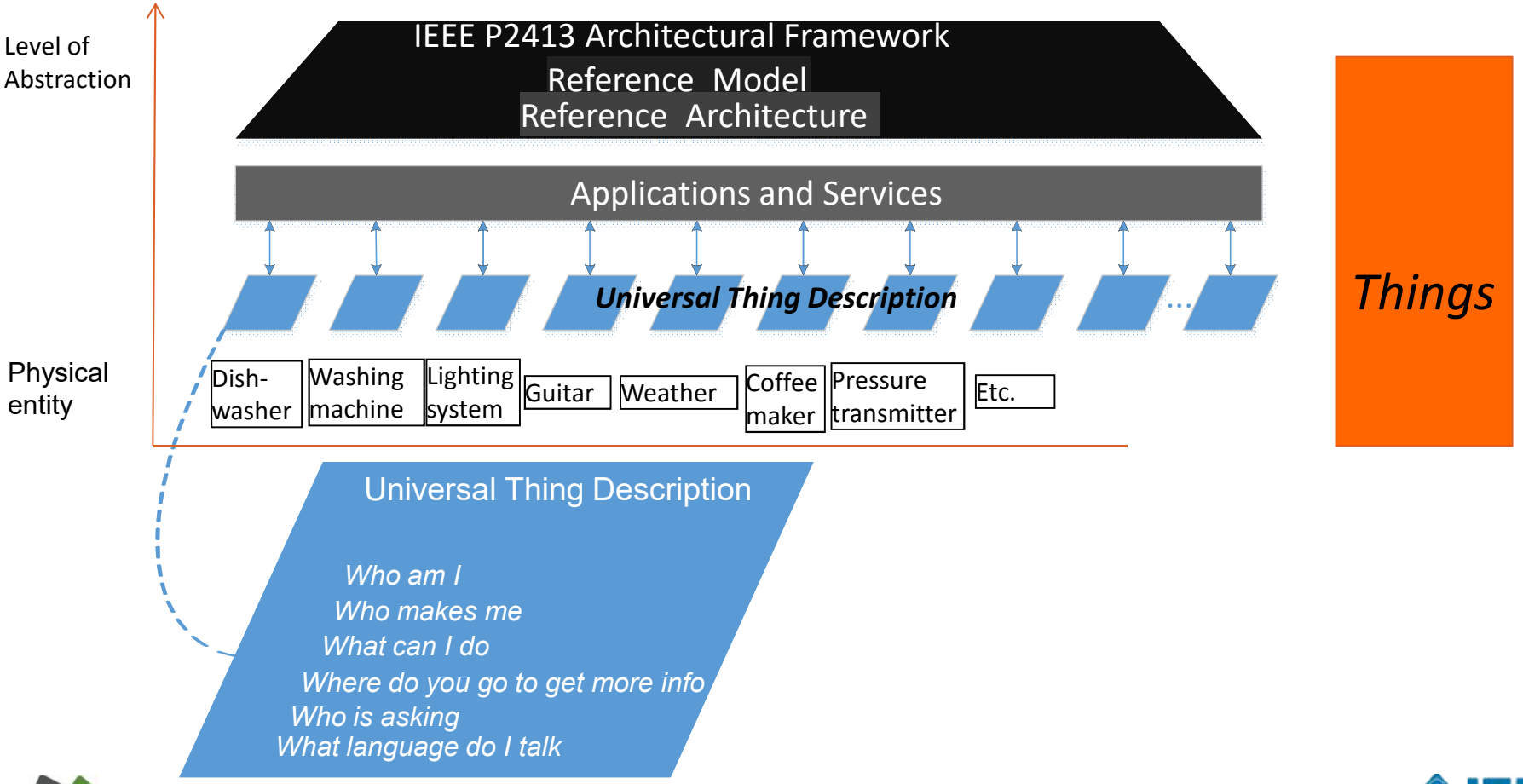


The "Thing"

Notes:

- Things, Apps, and Services can be integrated into what would be abstracted as a "Thing"
- Information exchange could be "horizontal" (subscribe/publish as an example) or vertical, or both
- Properties could be real or virtual

IEEE P2413 Levels of abstractions

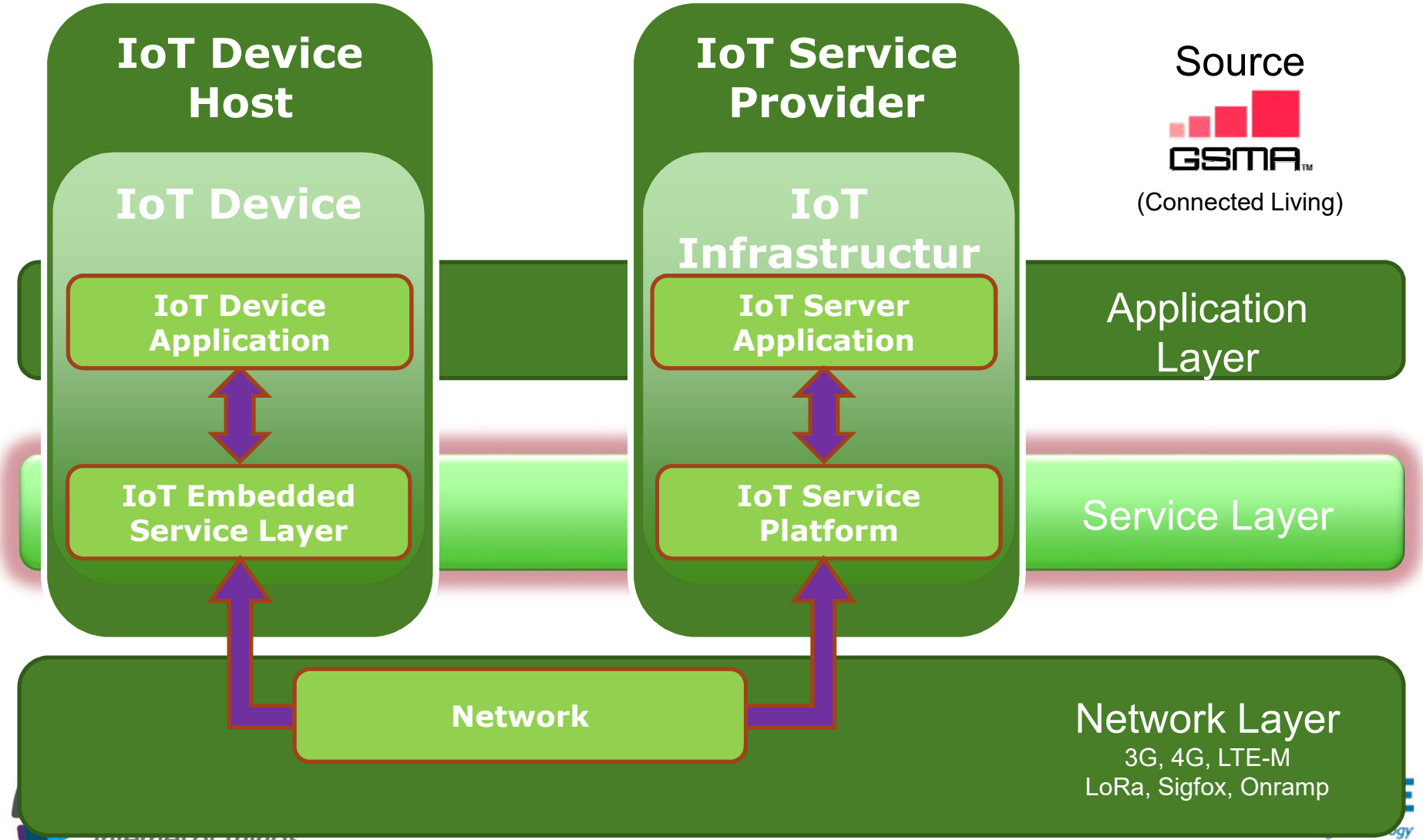


Scope & Objectives

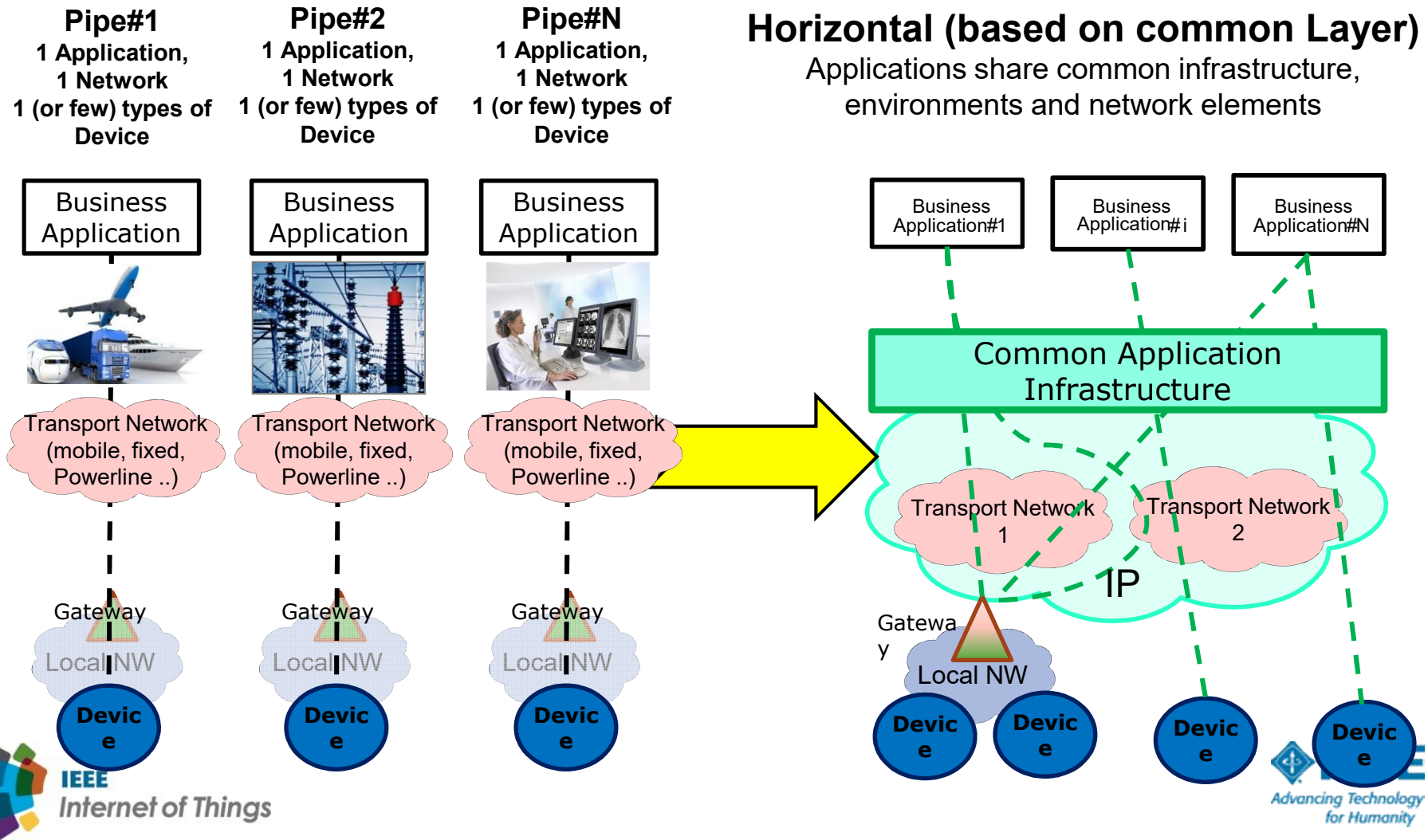
- **Common set of Service Layer capabilities**
- **Access independent view** of end-to-end services
- **Open /standard interfaces , APIs and protocols**
- Security, privacy, and charging aspects
- Reachability and discovery of applications
- **Interoperability**, including test and conformance specifications
- Identification and naming of devices and applications
- Management aspects (including remote management of entities

Next Slides: courtesy of Enrico Scarrone – Telecom Italia

oneM2M – The Service Layer



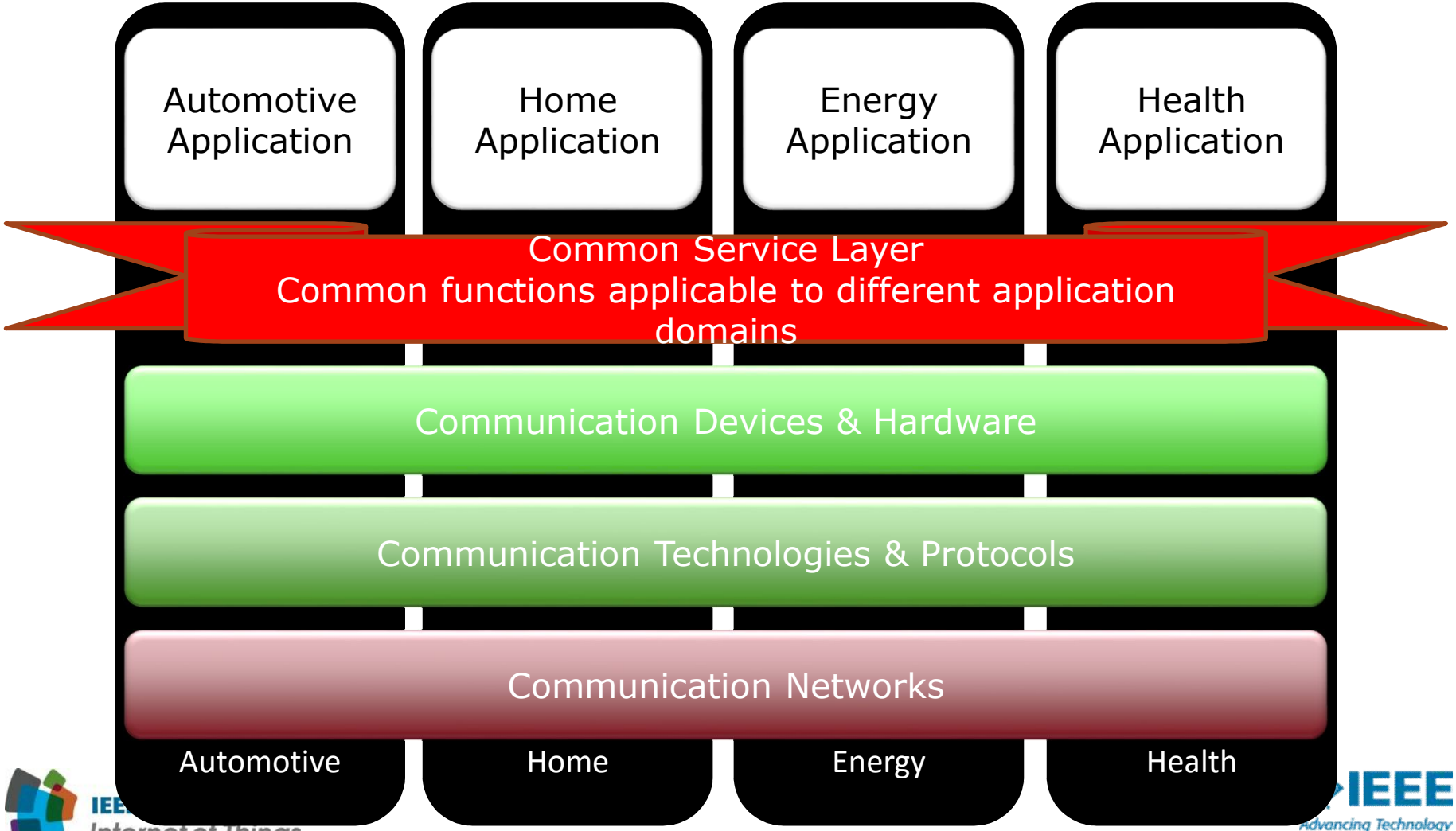
Break the silos and simplify the environment



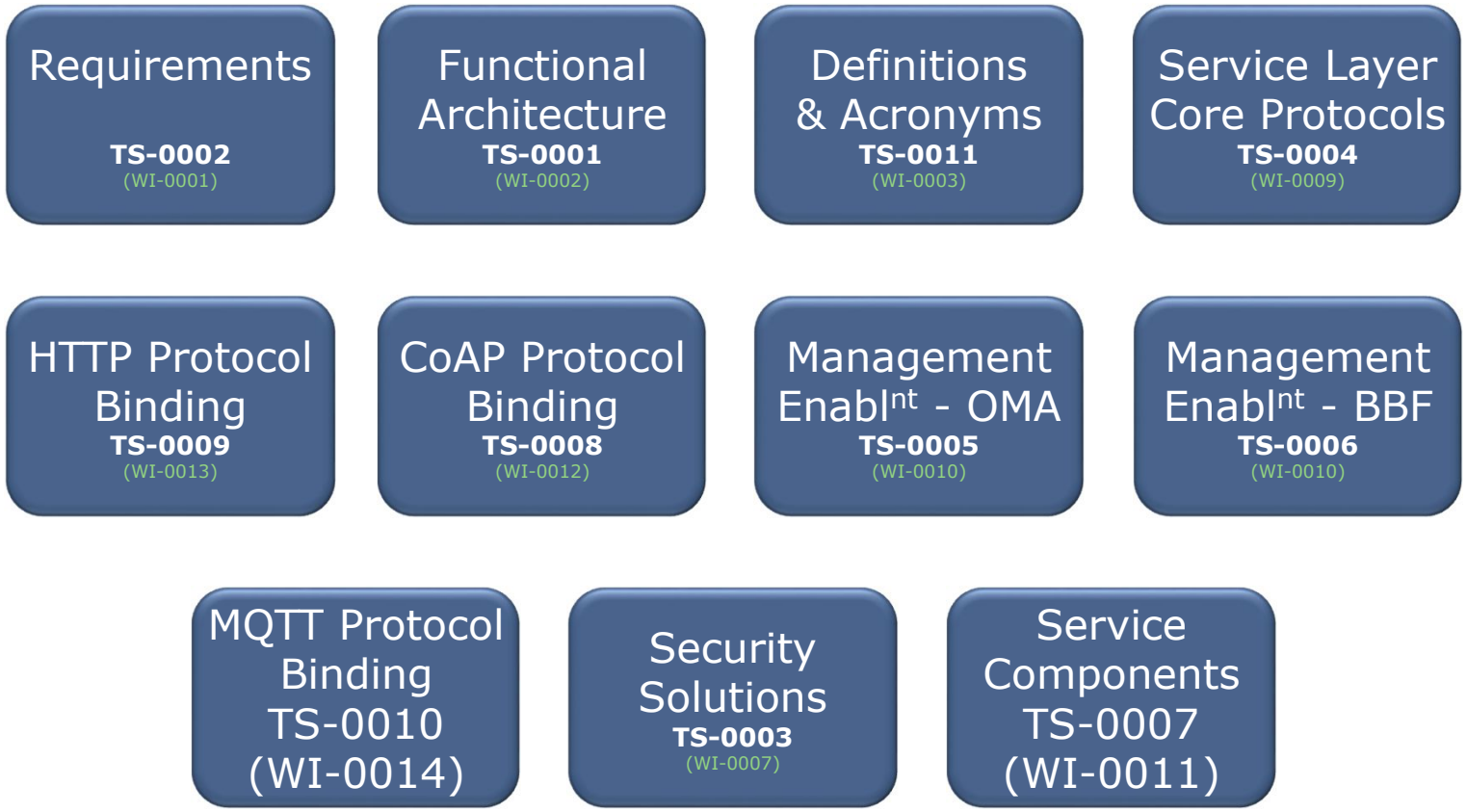
& provide the Interworking framework

- Simplification does not means one solution!
- Legacy technologies will continue to exist and needs to be integrated
- Specific technologies will be required in several sectors, for technical and commercial reasons
- TC SmartM2M solution is an interworking framework by means of a strict separation between communication and semantics aspects

oneM2M – The Service Layer



Main Technical Specifications



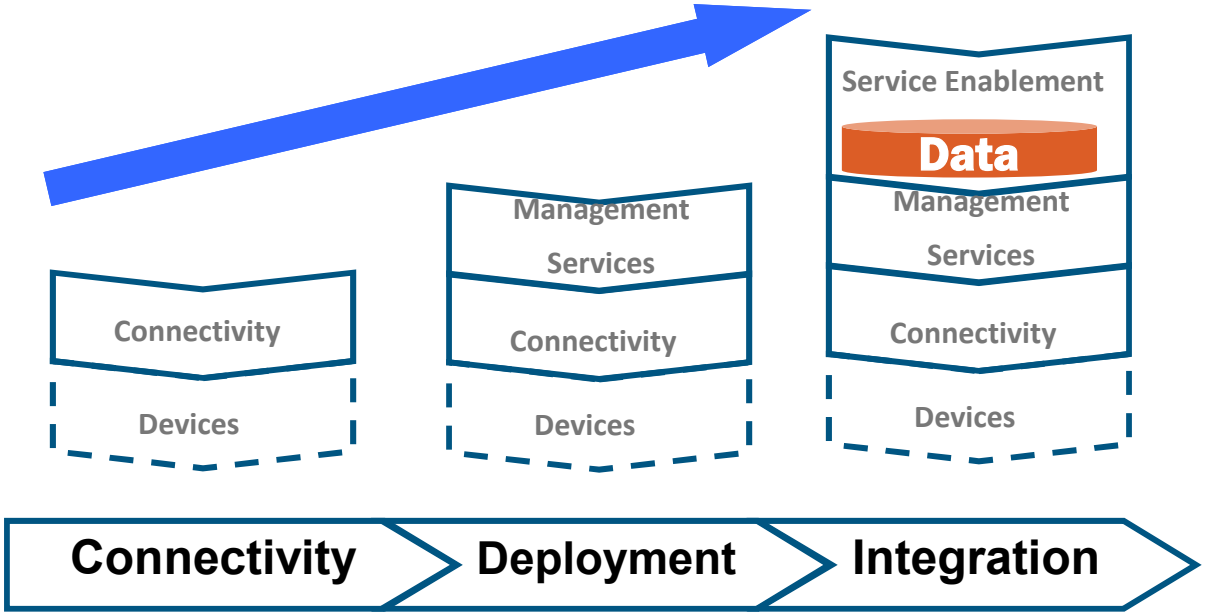
+ binding with web socket (under development)
+full testing suite (under development)

ftp://ftp.onem2m.org/Work Programme/

IEEE Internet of Things

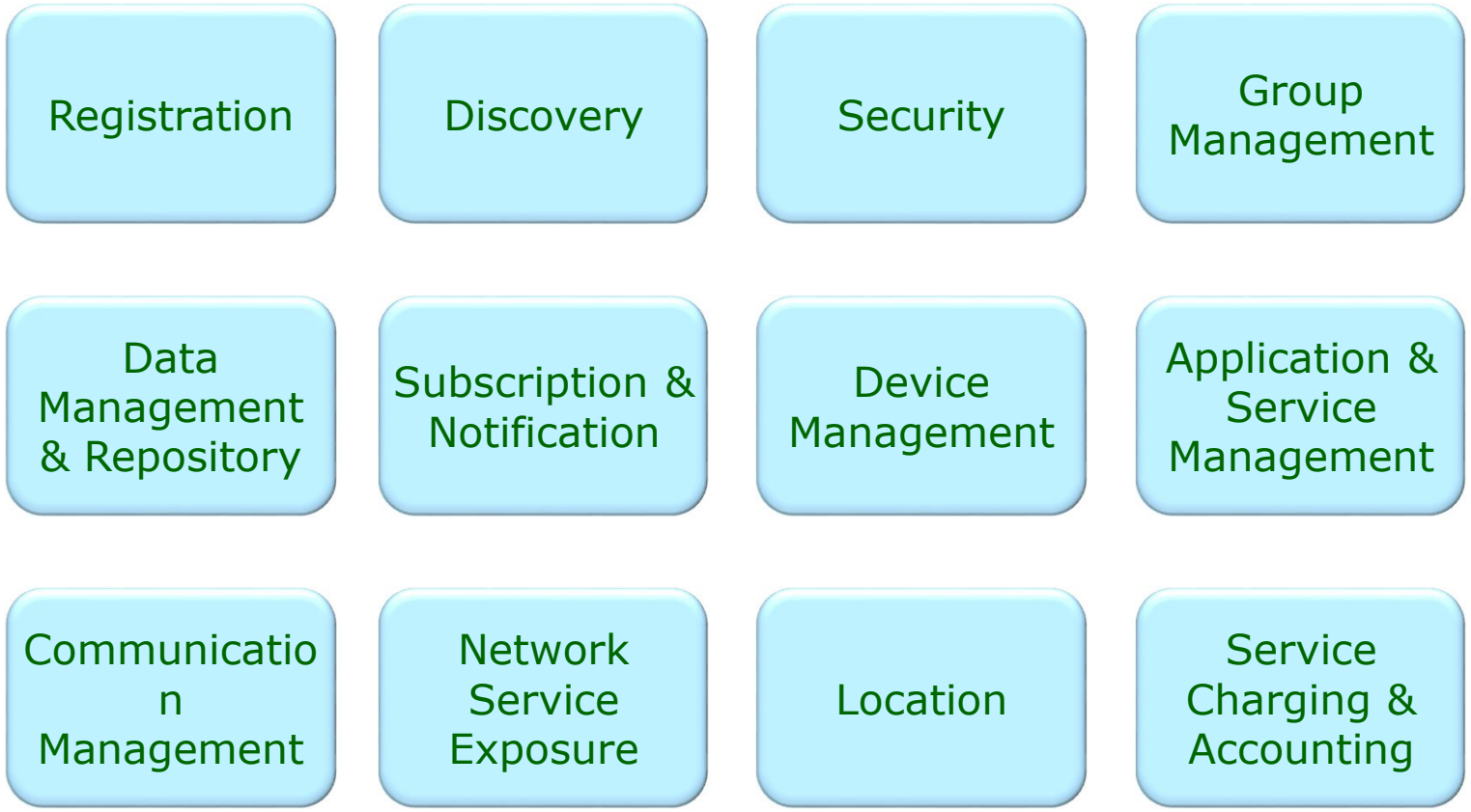
IEEE Advancing Technology for Humanity

oneM2M: It is all about information management and sharing



- OneM2M standard is based on a "Store and Share" resource based paradigm.
- The data may be made available in the platform to the other applications, interested application are notified by means of subscription
- Privacy is ensured by a strict Access Control Management, which relies on underlying network security, providing a secure light solution
- oneM2M is heavily reusing underlying network functionalities, including TR069 and OMA DM management, LCS, subscription management, QoS, Charging, etc.
- OneM2M release 1 has been released January 2015.

Common Service Functions



Module 20: Standardization

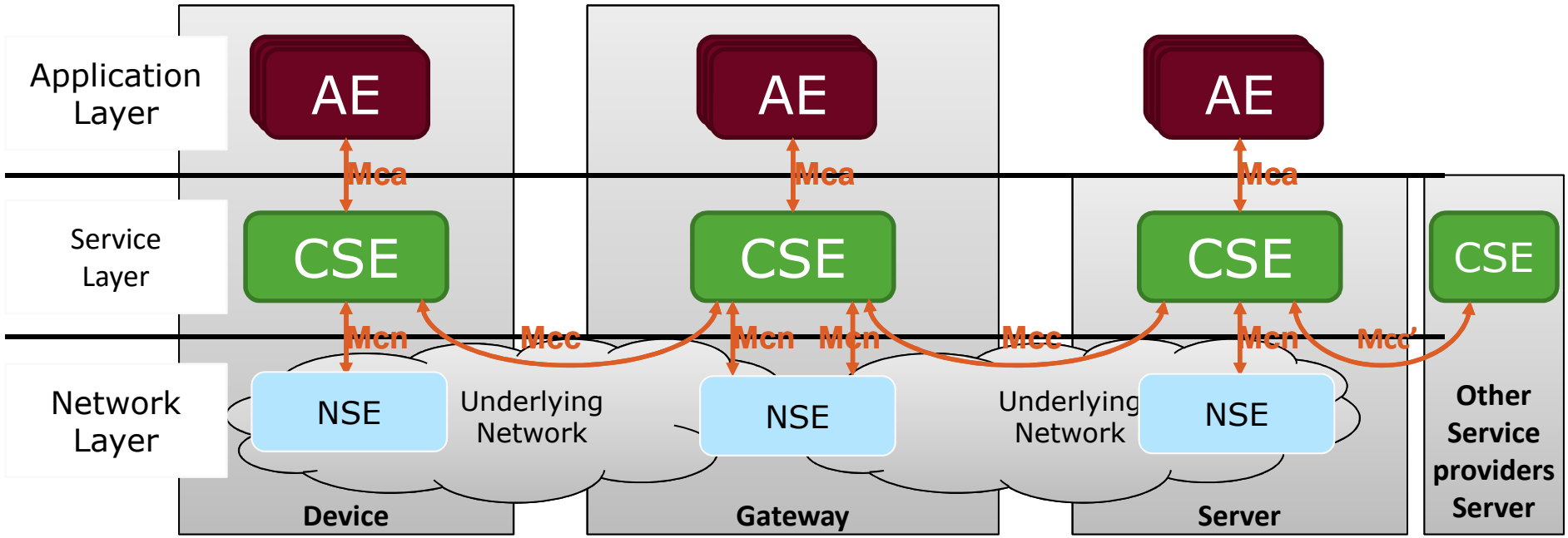
- What is an architectural framework and why is it impotant?
- What are the major functionalities (common services) to support in a IoT framework?

ONEM2M IN A NUTSHELL: PRINCIPLES, FUNCTIONS, ARCHITECTURE & API

IEEE
Internet of Things

IEEE
Advancing Technology for Humanity

oneM2M is Common API



Entities	AE (Application Entity), CSE (Common Services Entity), NSE (Network Service Entity
Reference Point	Mca, Mcn, Mcc and Mcc'

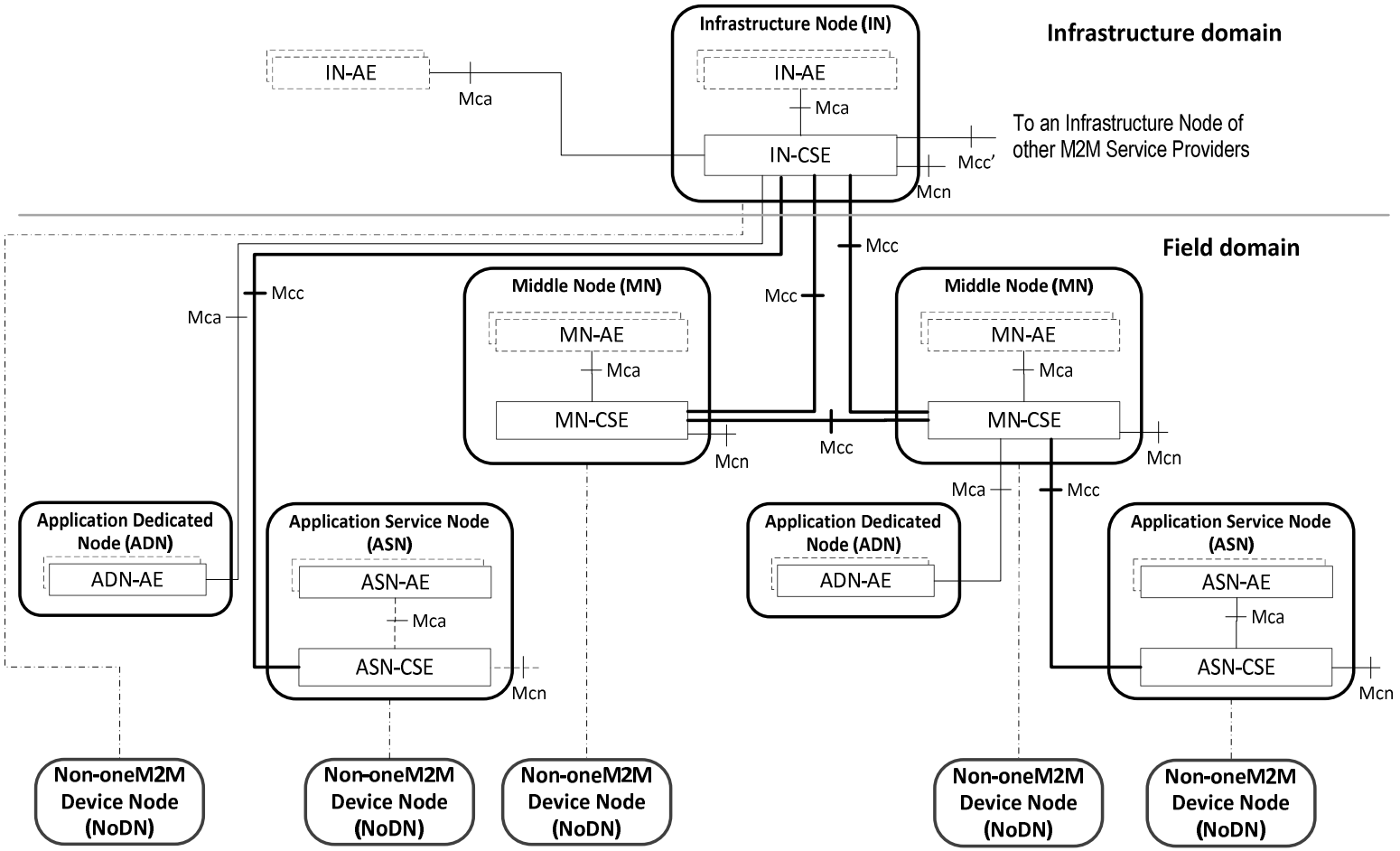
OneM2M architecture entities

- **AE:** Application Entity, containing the application logic of the M2M solution like home management functions, fleet management, blood sugar monitoring
- **CSE:** Common Service Entity containing a set of common service functions (CFE) that are common to a broad range of M2M environment (verticals). This is the main part of the oneM2M specification
- **CSF:** Common Service Functions included in a CSE, CSFs can be mandatory or optional, CSF can contain sub-functions (mandatory or optional)
- **NSE:** Network Service Entity, provides network services to the CSE, like device triggering, device management support, location services. These services are related to the underlying network capabilities

OneM2M architecture Reference points

- **Mca- Reference Points:** the interface point between the AE and the CSE, the Mca point provides the M2M applications access to the common services included in the CSE. The AE and CSE my be co-located in the same physical entity or not
- **Mcc- Reference Points:** This is the reference point between two CSEs. The Mcc reference point shall allow a CSE to use the services of another CSE in order to fulfil needed functionality. Accordingly, the Mcc reference point between two CSEs shall be supported over different M2M physical entities. The services offered via the Mcc reference point are dependent on the functionality supported by the CSEs
- **Mcn- Reference Points:** This is the reference point between a CSE and the Underlying Network Services Entity. The Mcn reference point shall allow a CSE to use the services (other than transport and connectivity services) provided by the Underlying Network Services Entity in order to fulfil the needed functionality.
- **Mcc‘- Reference Point:** interface between two M2M service providers, As similar as possible to the Mcc reference point. But due to the nature of inter-M2M Service Provider communications, some differences are anticipated.

Full architecture: allowed configurations



One or more AE

Zero or more AE

Link is out of scope

oneM2M in a nutshell

OneM2M is an IoT Interworking Framework

- Designed to interwork with legacy, proprietary and sector solution. This is based on a separation between protocol interworking and semantic interworking.
- Semantic interworking is already present in oneM2M Release 1, extention to full semantic support will be completed in oneM2M Release 2.

OneM2M is an IoT common Service Enablement layer

- Service independent
- Distruibuited (Devices, Gateways, Network servers)
- Application portability
- Flexible: AE can have a specific client or connect directly to gateways or network server
- Data can be stored in Devices, gateways or network server almost trasparently

oneM2M in a nutshell

Main Characteristics

- URI identification (and separation from IP addressing)
- IP based (irrelevant the version, IPv4 or IPV6)
- Network independent (but network aware!)
- REST approach
- Application protability
- Device and subscription management
- Accounting and charging
- HTTP/COAP/MQTT transport

Peculiary functions

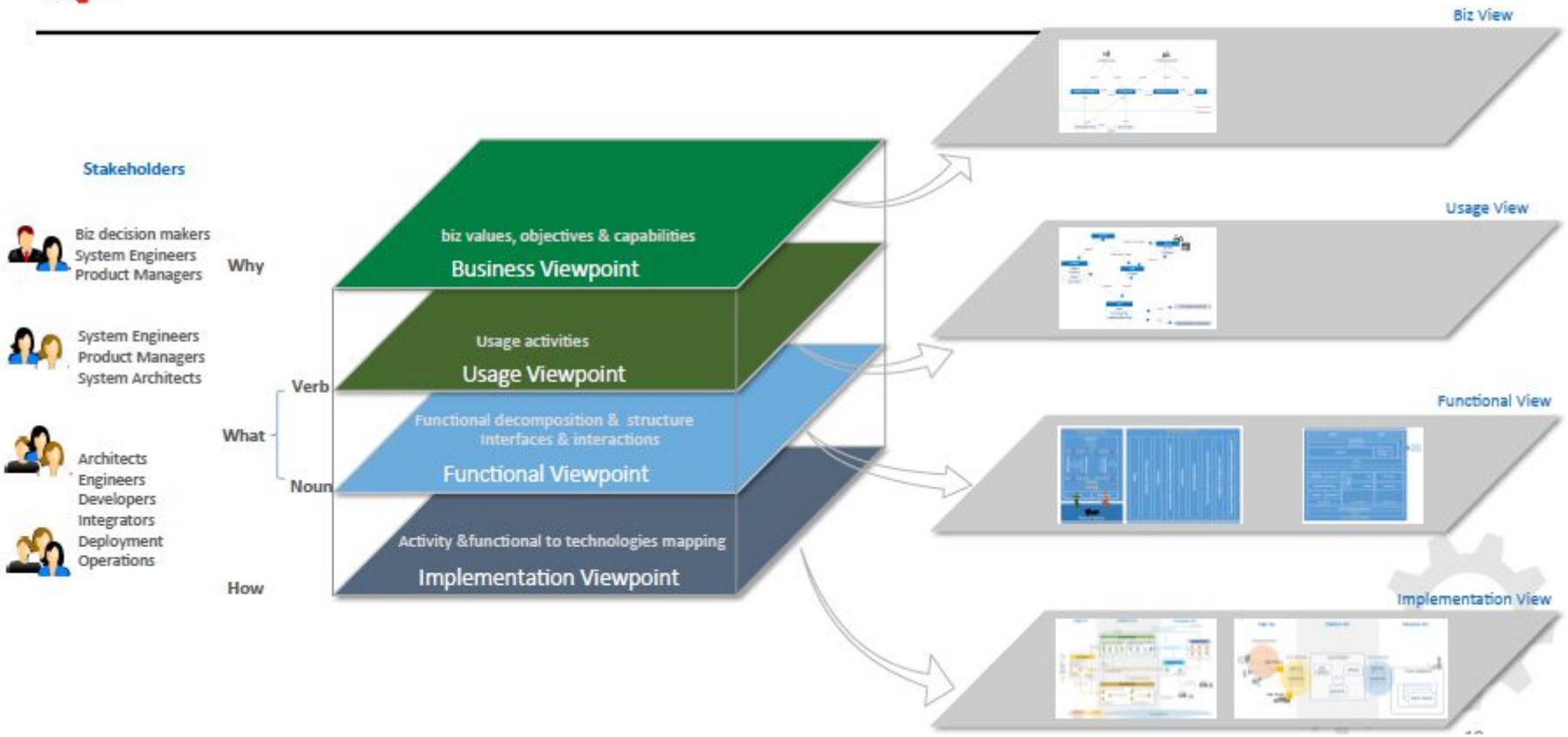
- Store and share paradigm
- Data management and historization
- Separation among Security and Privacy
- Flexible deployment (large, small, distributed, centralized)
- Network functionality re-use (Location, Device Management, Security, etc)

Module 21: oneM2M

- What are the functional entities and their reference points?
  - What is a functional architecture?
- What are the major characteristics of oneM2M ?

Industrial Internet Consortium

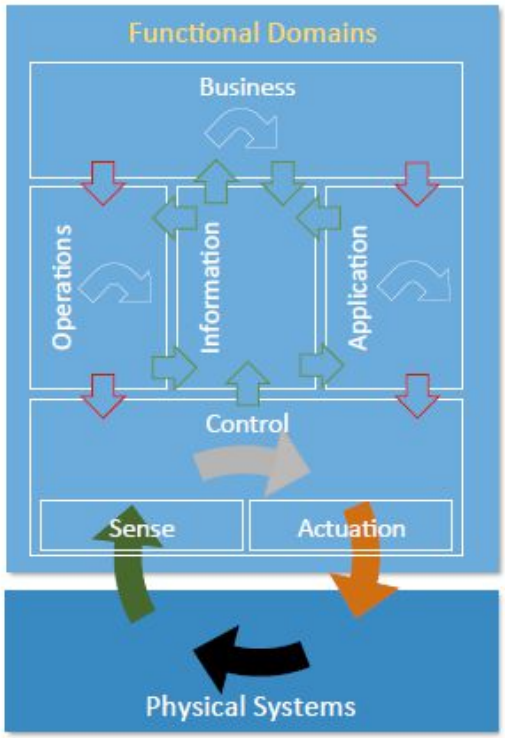
ISO/IEC/IEEE 42010:2011 - Architecture Description for IIC

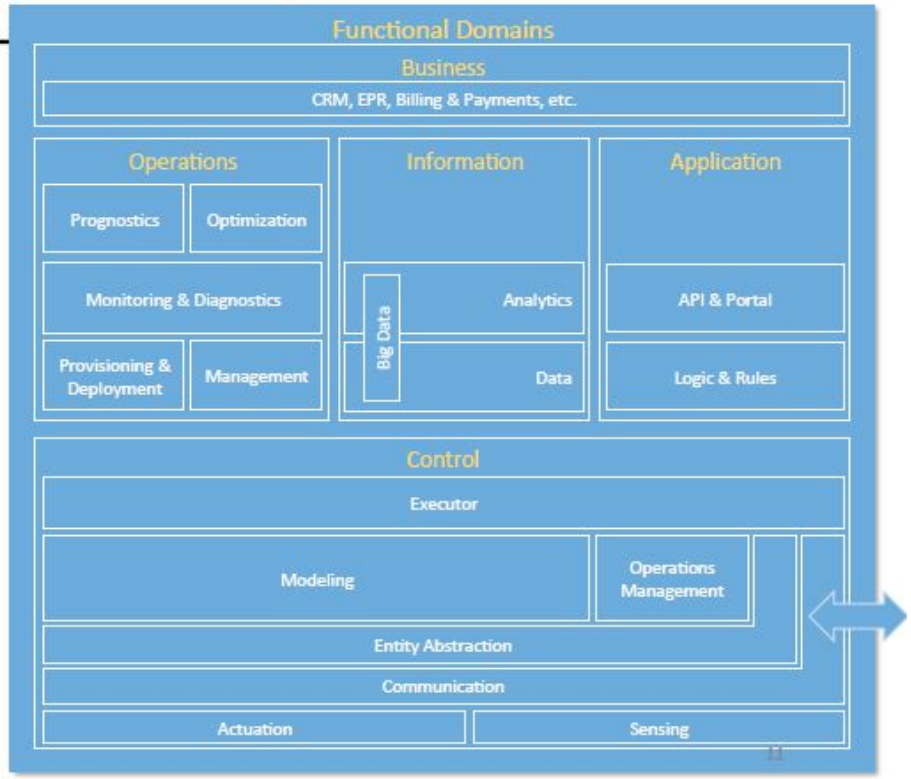


Sources: Architecture of Industrial Internet by S. Mellor
And IIC reference architecture http://www.iiconsortium.org/IIRA-1-7-ajs.pdf

IIC - Functional View

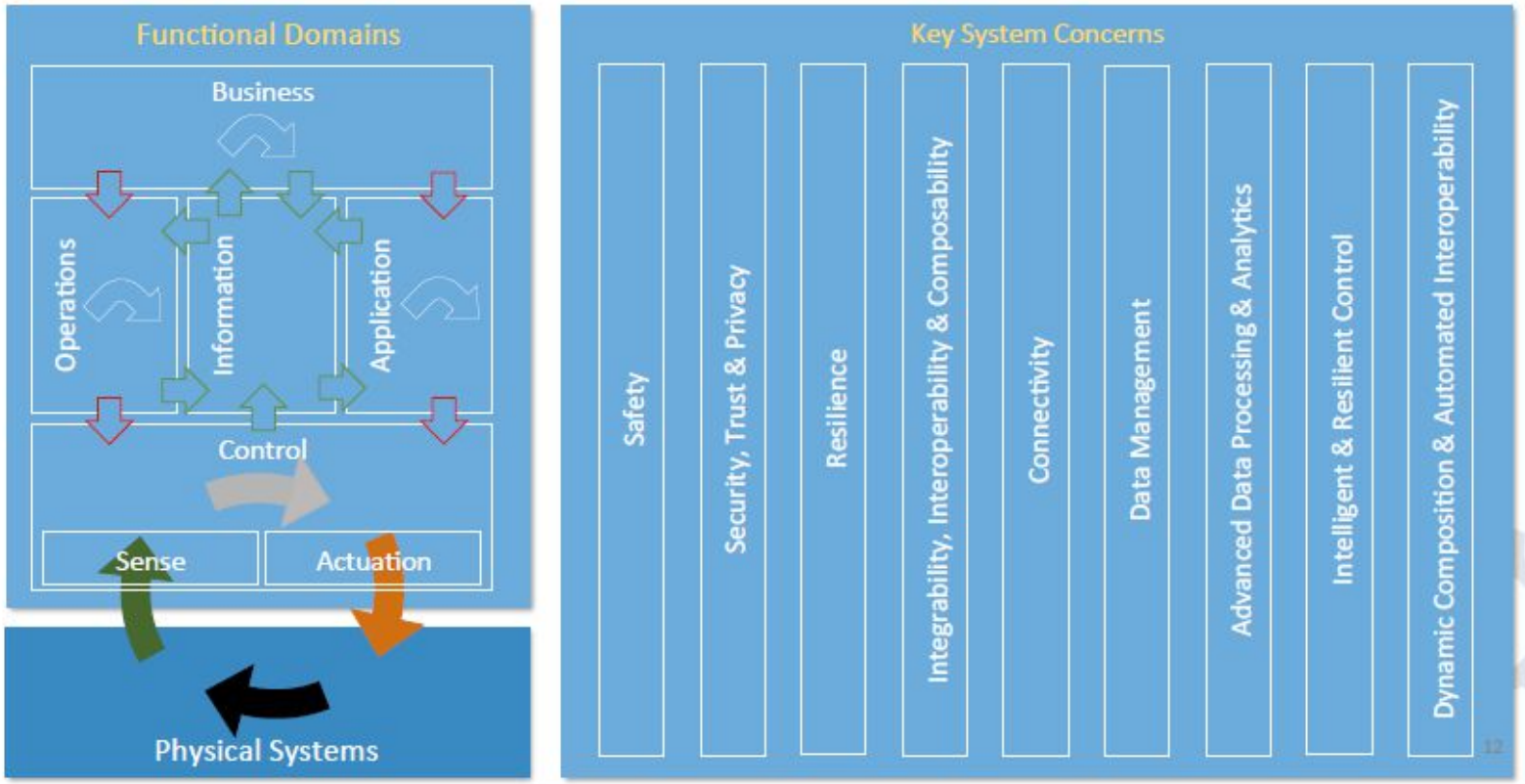
Functional Viewpoint – Domain decomposition





Key System Functionalities

Key System Concerns



Module 22: Industrial Internet Consortium

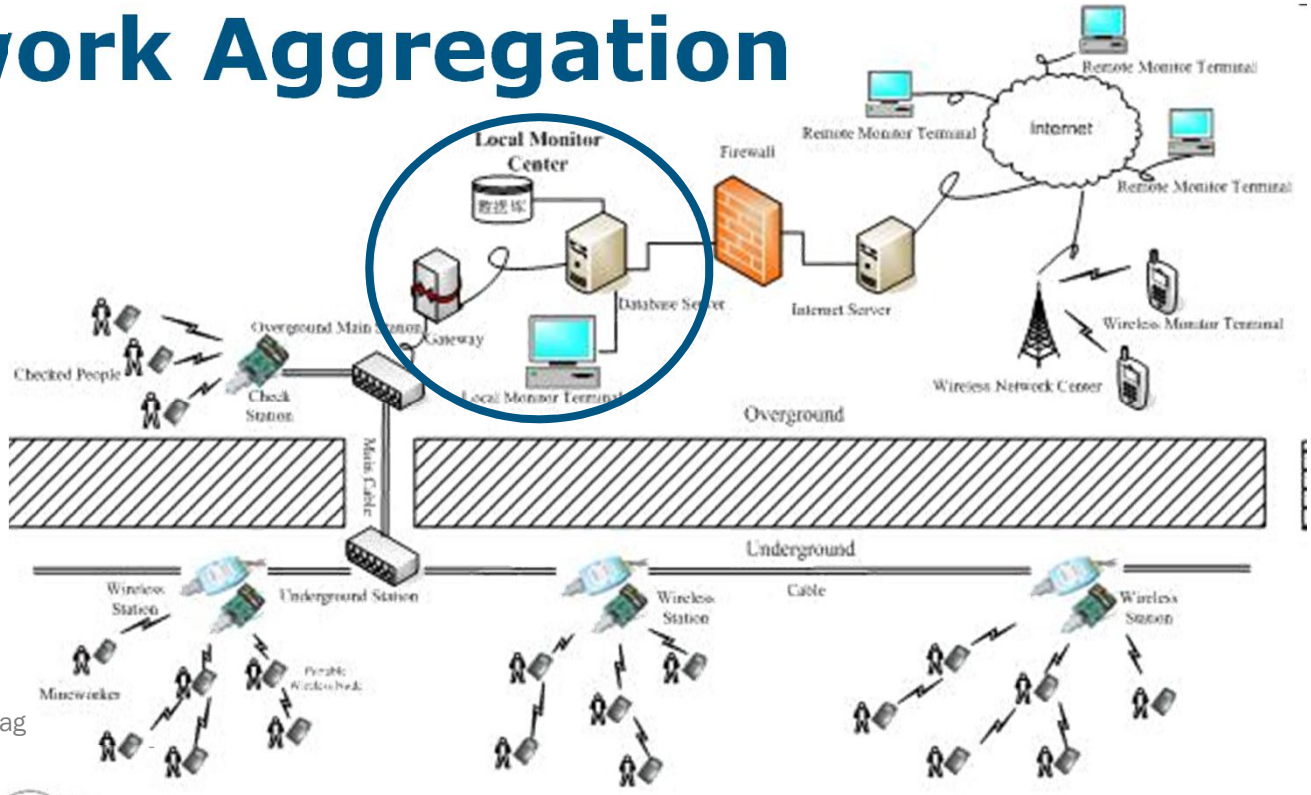
- What is the aim of the Industrial Internet Consortium?
- What are some major functionalities for supporting the Industrial Internet?

Agenda – Internet of Things

- The Context of IoT
- A Definition of IoT
- A few Challenges of IoT
- What Things are …
- Networks of Things
- Technologies of Communications
  - Access Technologies
  - Protocols
  - SW Platforms
  - Middleware
  - Standards
- ***IoT Challenges***
  - Identity, Data, and Ownership
  - Complex System
  - Business Issues
  - Social Issues
- Virtual Continuum
- IOT Scenarios
- The IEEE IoT Initiative

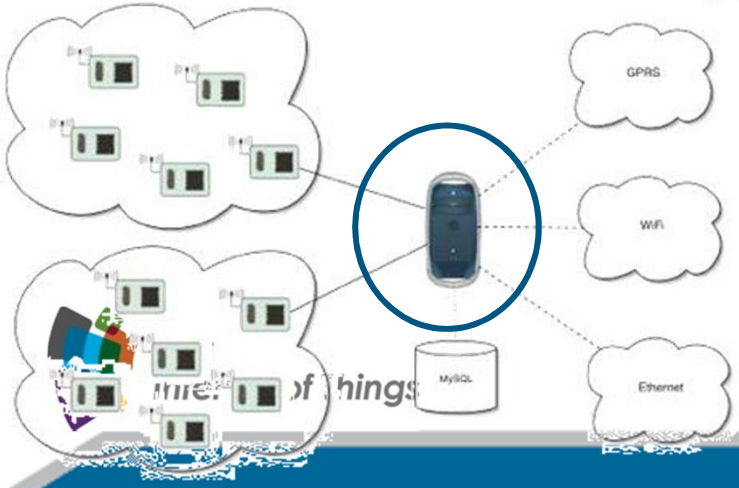
IoT, Identity and Data: a Conundrum

Where are the Data: Wireless Sensor Network Aggregation



http://www.mdpi.com/1424-8220/9/6/4845/ag

http://www.easinet.cn/photo/coalmine.jpg



Plenty of Data … distributed locally and remotely

Aggregation function and dispatching functions are fundamental

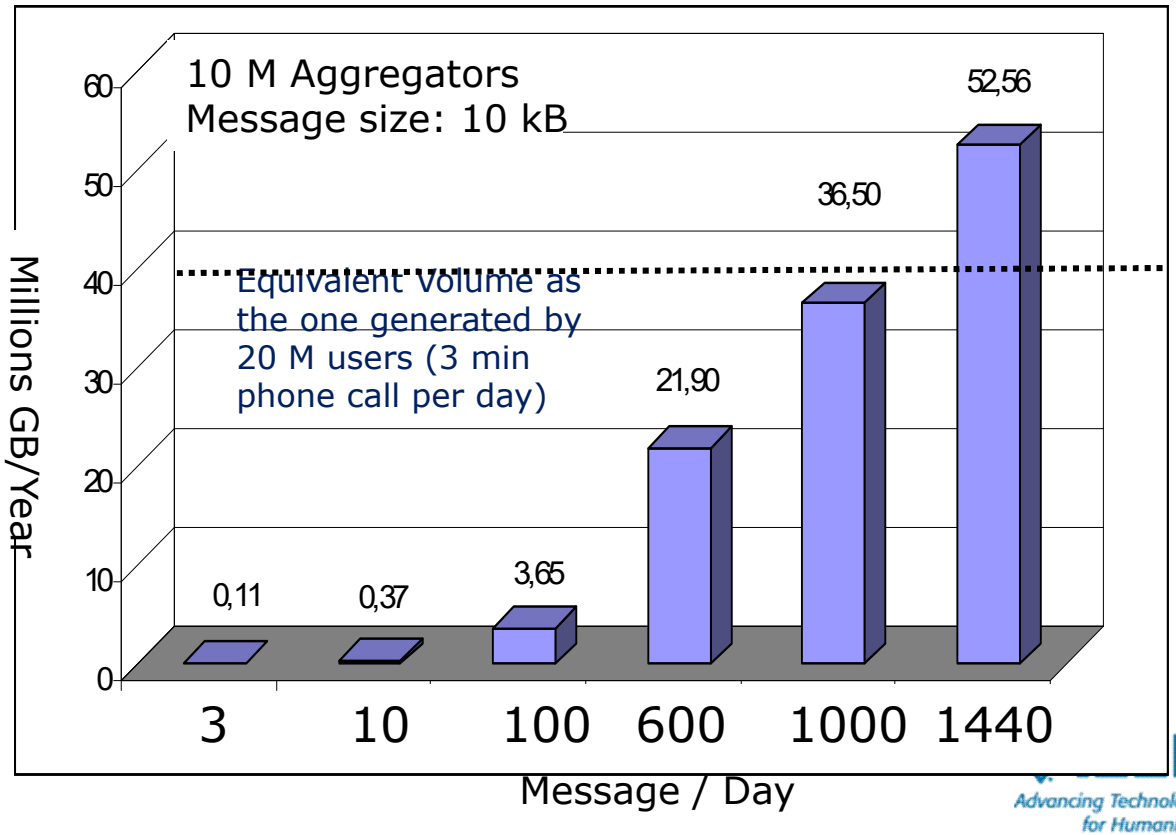
IEEE
Advancing Technology for Humanity

How Many Nodes, How Many Messages, How Much Bandwidth ?

The Bandwidth challenge …

Issue: low average traffic, but highly impulsive traffic (e.g., spikes of messages when containers ships enter in a harbor)

- Gateways/Aggregators will greatly reduce the number of messages forwarded on public networks
- Multimedia (video) will be the major cause for traffic
- Many objects/nodes will come with communications already paid for (i.e., embedded communications)
- Pure bit transport is not a big value for Operators



How much Data (and traffic)?

Table 3. Summary of Per-Device Usage Growth, MB per Month

Device Type	2014	2019
Nonsmartphone	22 MB/month	105 MB/month
M2M Module	70 MB/month	366 MB/month
Wearable Device	141 MB/month	479 MB/month
Smartphone	819 MB/month	3,981 MB/month
4G Smartphone	2,000 MB/month	5,458 MB/month
Tablet	2,076 MB/month	10,767 MB/month
4G Tablet	2,913 MB/month	12,314 MB/month
Laptop	2,641 MB/month	5,589 MB/month

Source: Cisco VNI Mobile, 2015

= 2.33 MB /Day

= 27 Byte /s

= 12.2 MB /Day

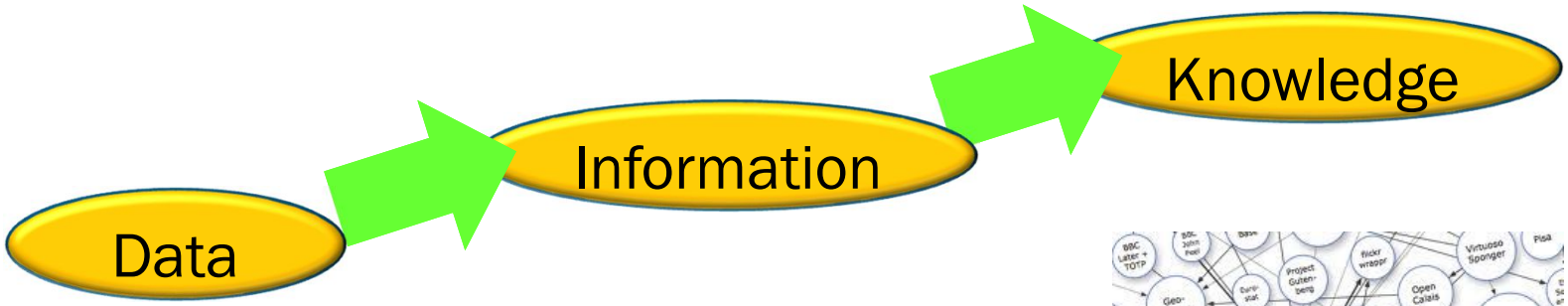
= 141 Byte /s

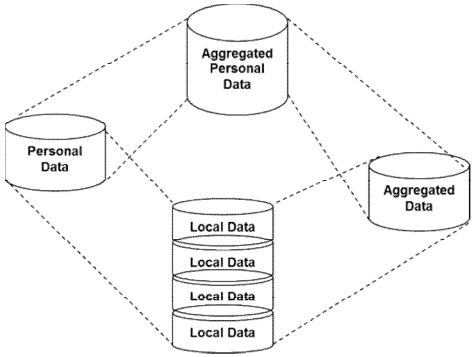
http://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/white_paper_c11-520862.html#Trend_3_Measuring_Mobile_IoE

Module 23: IoT and Connectivity

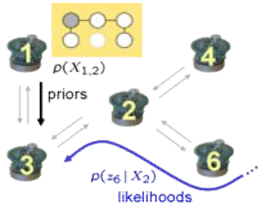
- Is connectivity a viable business model for IoT?
- Why ?
- What will the biggest source of traffic be in the future?

Where is the Value then ?

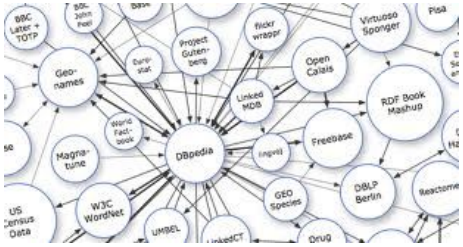




- **Aggregation**
- **Personalization**



- **Inference**



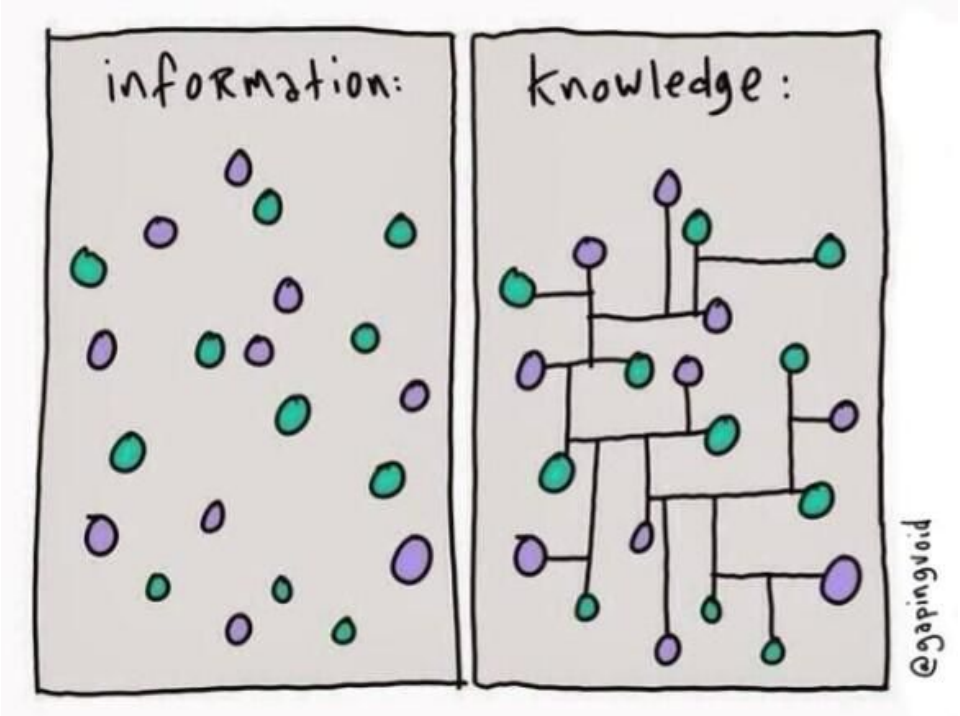
- **Relationships**

The Big Data Challenge …

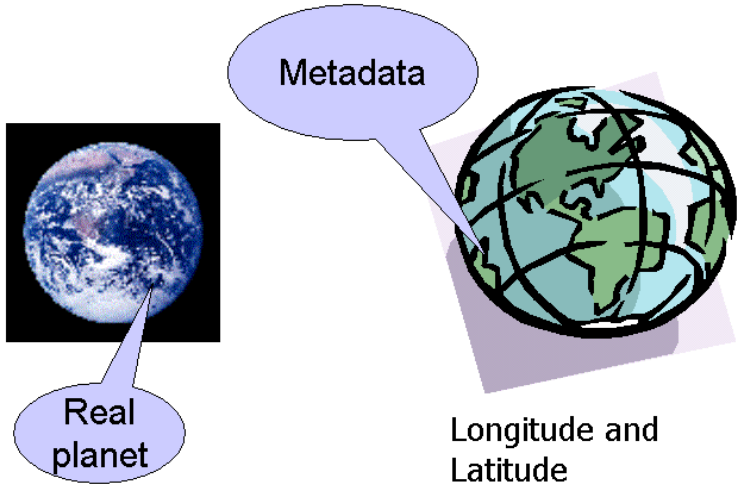
IEEE Internet of Things

IEEE Advancing Technology for Humanity

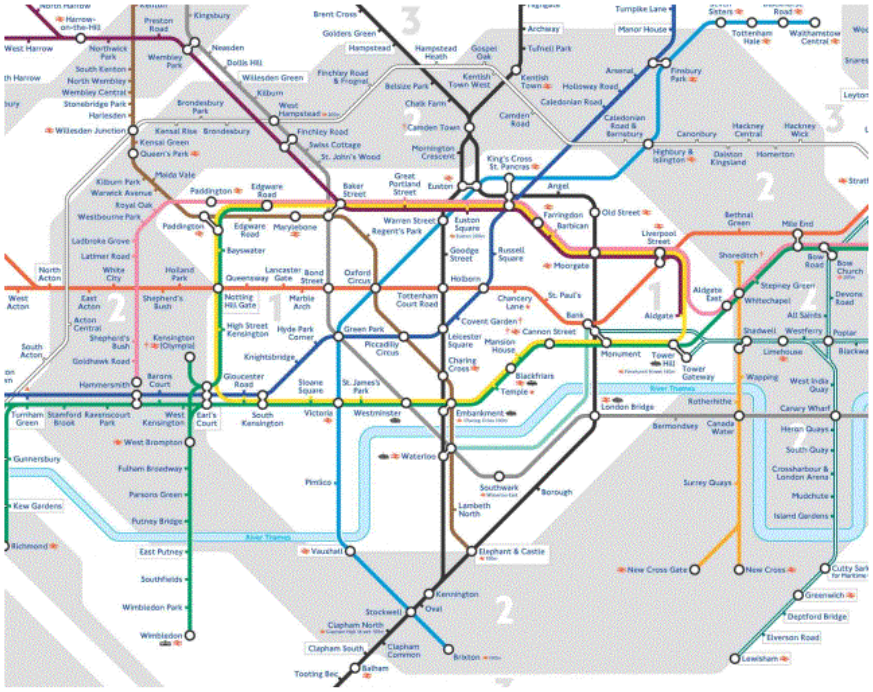
- The Data and Knowledge Path



Examples of (simple) metadata



http://www.kcoyle.net/meta_purpose.html

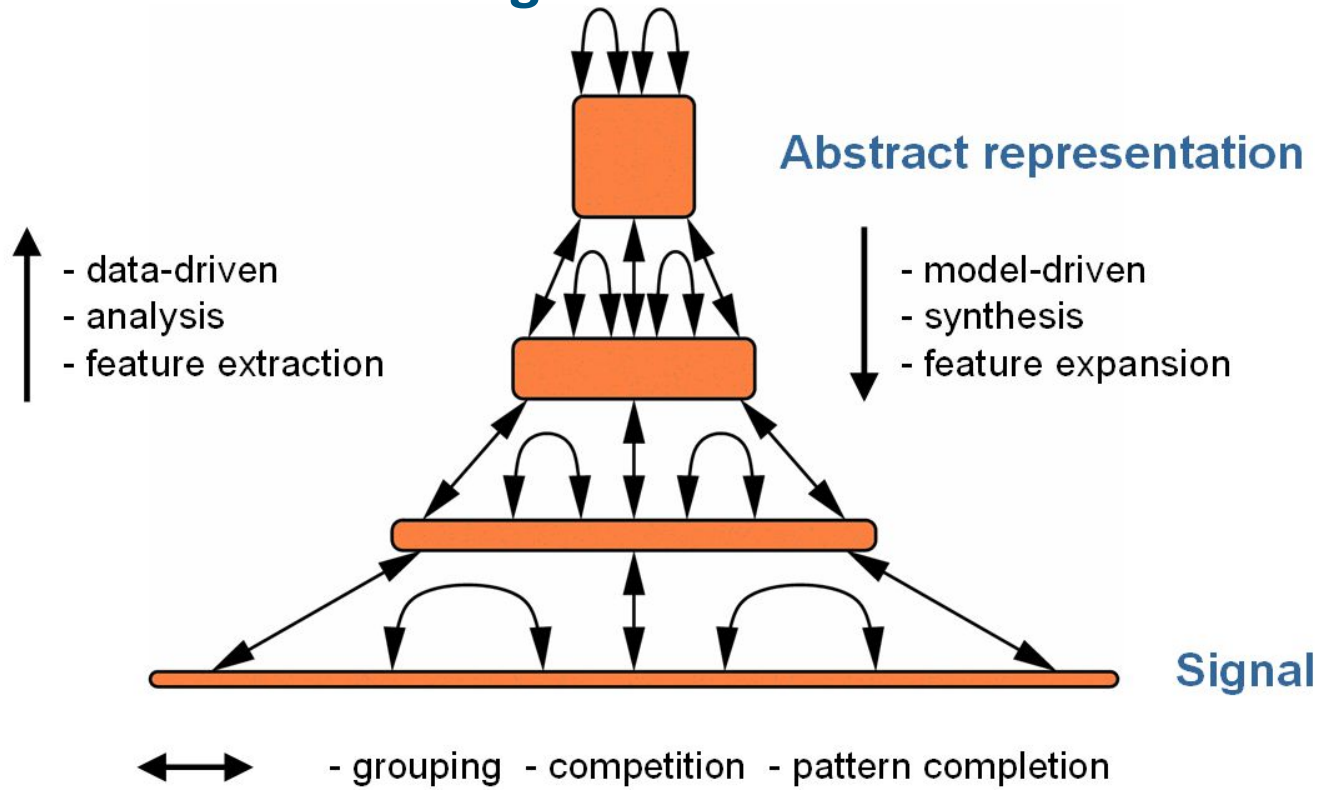


- Abstraction & richness of information
- New Usages

Data Mining

From Data → To Information

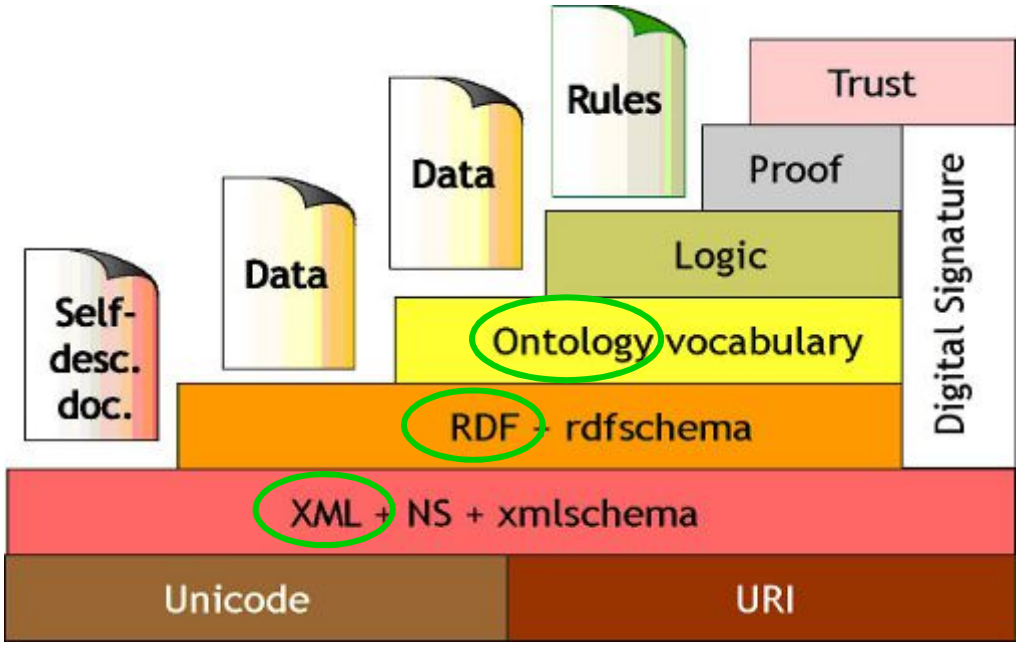
Extracting Value from Data



http://www.ais.uni-bonn.de/images/Neural_Abstraction_Pyramid.png

The Semantic Web

Representing Data and deriving Knowledge



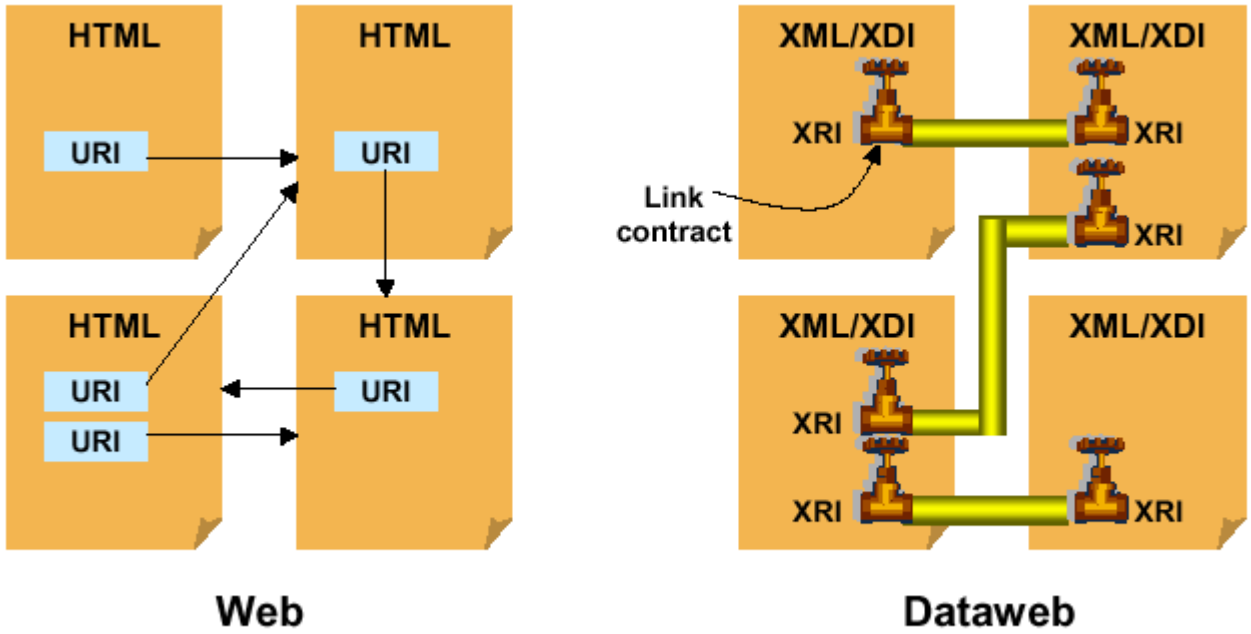
http://www.w3.org/2000/Talks/1206-xml2k-tbl/

However Big Data is all about non structured data

DataWeb

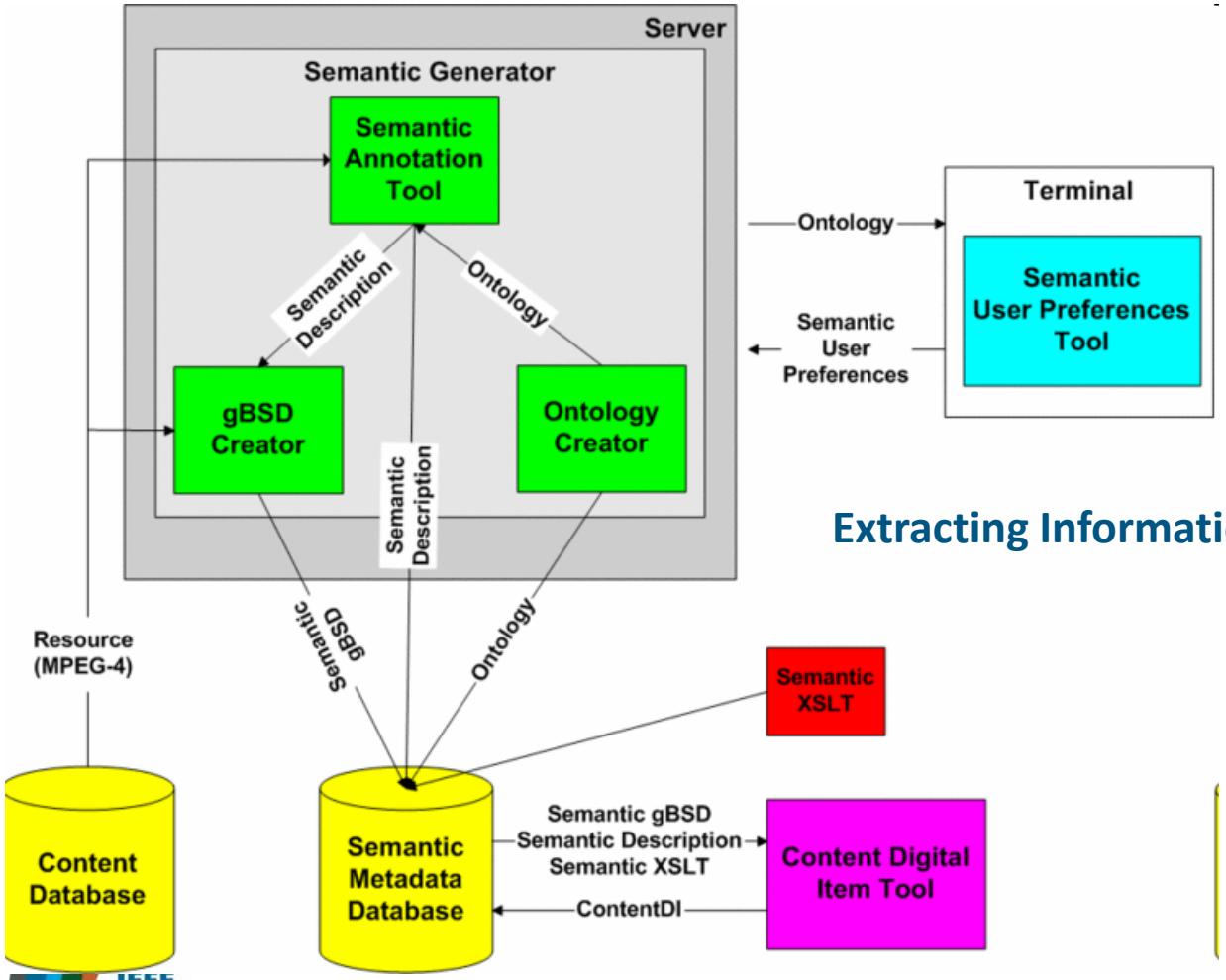
(ref.: The Dataweb: An Introduction to XDI - White Paper for the OASIS XDI Technical Committee)

Linking the data and protecting them



The goal of XDI is to enable data from any data source to be identified, exchanged, linked, and synchronized into a machine-readable dataweb using XML documents just as content from any content source can linked into the human-readable Web using HTML documents today.

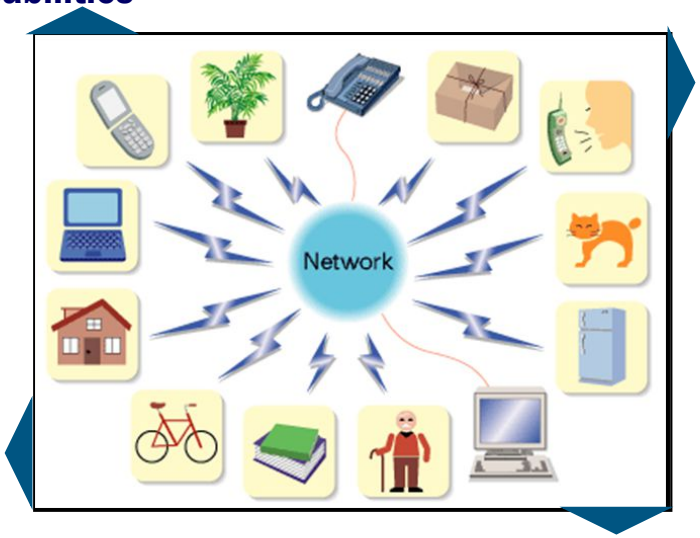
MetaData and Multimedia



Extracting Information from different types of data

Some Consequences

- Data have to be exchanged between different networks
  - There will be the need to have network capabilities



- Data will create complex relationships
  - Self-organizing systems
  - MetaData
  - Trust, Security and Privacy
  - New control Patterns
- Data will be everywhere
  - There is the need to transform data into useful information
- Security and Trustiness will become more and more important
  - How reliable are data? How secure?

Module 24: the data path

- Where value can be extracted from?
- What is metadata? What are some technologies for extracting information from data?