Pervasive Systems

Ioannis Chatzigiannakis

Sapienza University of Rome
Department of Computer, Control, and Management Engineering (DIAG)

Lecture 21:
Web of Things

Application using Wireless Sensor Networks

Categorization of Applications

1. based on Sensor Types – (i.e. thermal, visual, seismic, acoustic, radar, magnetic, etc.) in order to monitor a wide variety of conditions
2. based on Control Center Types
   - fixed control center
   - mobile control center
   - multiple control centers
3. based on Notification Strategy – depending on the nature of the application
   - *Periodic Sensing* – constantly monitor the physical environment and periodically report
   - *Event driven* – monitor the environment and send reports only when certain events are realized
   - *Query based* – respond to queries made by a supervising control center

Application using Wireless Sensor Networks

Goal & Key Features

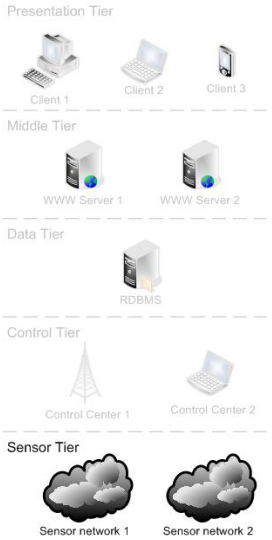
- provide a generic and modular application environment for developing and managing applications that are based on wireless sensor networks.
- provide a range of services that allow to create customized applications with minimum implementation effort that are easy to administrate
- move beyond the "networking-centric" view of sensor network research
- focus on how the end user (administrator, control center supervisor, etc.) will visualize and interact with the system

jWebDust – Generic Application Environment for Wireless Sensor Networks

jWebDust's Architecture: Key Points

- jWebDust differentiates the system into two main groups:
  1. the networked sensor devices (from now on referred as *sensors*) that operate using TinyOS
  2. the rest of the network (e.g. control centers, database server, etc.) that is capable of executing Java code
- Both system groups use an open architecture implementing the emerging component-based architecture.
- The standardized component interface and the exchange of data over broadly used protocols provide increased portability.

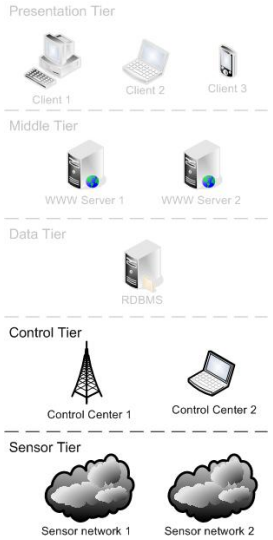
jWebDust's Architecture: N-tier application model



Sensor Tier

- the foundation of any application based on wireless sensor networks
- each of these scattered sensors has the capability to collect data and route data back to the control center
- sensors form **one or more** sensor networks
- each network can have devices with *heterogeneous* characteristics
- devices operate under the Tiny OS

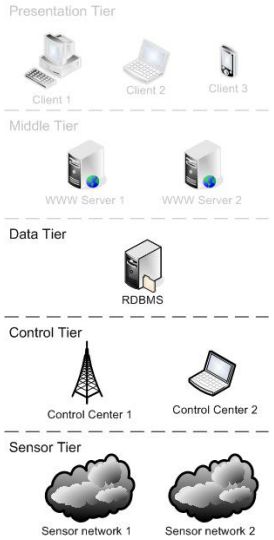
jWebDust's Architecture: N-tier application model



Control Tier

- consists of the control center(s) where the sensor tier reports to
  1. Control centers can operate even when there is no connection to the data tier for sustained periods of time.
     - Buffering of data from the sensor tier
     - Polling for queries from the data tier
  2. Support of multiple WSNs in a way that they are seen as a single virtual sensor network.

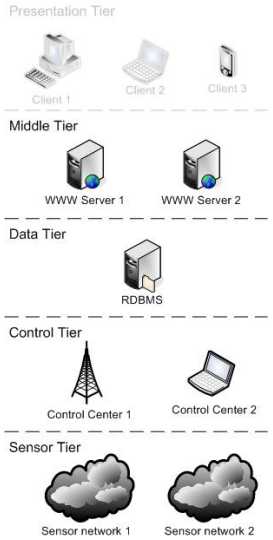
jWebDust's Architecture: N-tier application model



Data Tier

- responsible for storing info received from WSN and queries to WSN
- database schema consists of 10 tables organized in three categories:
  1. **Mote related tables** that are used to store information regarding the technical characteristics of the sensors
  2. **Query related tables** that keep track of active and historic queries
  3. **Sensor readings table** that stores the information received from the sensor network

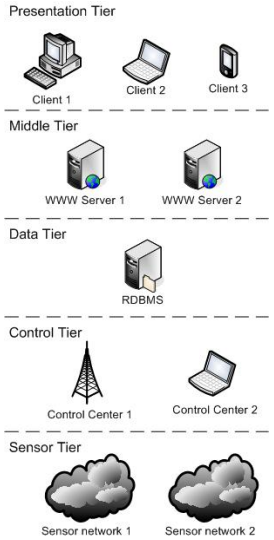
jWebDust's Architecture: N-tier application model



Middle Tier

- responsible for processing data, statistics and generating query data
- responsible for delivering structures and data to the presentation tier
  1. The middle tier is made up from components
  2. These components can be considered as applications that run on a server without a face (servlets)
  3. They provide an easy-to-use interface for developers to use the underlying tiers

jWebDust's Architecture: N-tier application model



Presentation Tier

- interface of the system to the final user
  1. Web-based interface uses portlets
  2. The end users are allowed to choose among the available client solution
  3. The open architecture allows new components to be introduced at later stages
  4. current version is based on rich clients

Sensors Discovery

Zero Configuration Protocol:
Every sensor is able to register itself and its distinct features

1. Discovery and registration of every sensor is achieved using a simple protocol
2. When a sensor is powered on, it sends out a message to the control center containing the sensor's ID, its' type and a list of available sensors
3. Ability to make queries concerning specific sensors of the sensors in the network
4. Support of heterogeneous WSNs

Sensor Network Communication Protocol

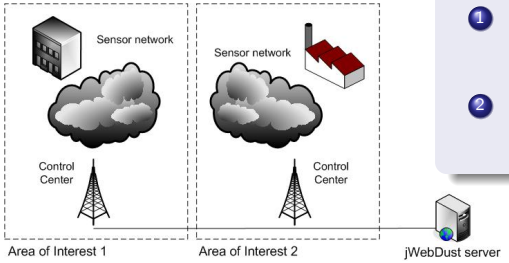
- Communication protocol is based on the multi-hop tree-based routing protocol included in the TinyOS distribution, called `MultiHopRouter`
- We extend it towards reducing the energy dissipation
  1. we implement an extension that uses a mechanism for varying the transmission range
  2. based on the ability of TinyOS to adjust the transmission range of the sensors
  3. main idea is to periodically check whether a satisfactory number of nearby sensors is active
  4. sensors can decide on whether to modify their transmission range in order to overcome network connectivity issues or not

Sensor Query and Data Dissemination Protocol

- We categorize all possible queries using two criteria
  1. the sensors they are targeted to
     - node-specific
     - attribute-based
  2. the way data regarding the queries is reported to the control center
     - periodic sensing
     - event-driven
- Sensor queries are combinations of these categories
- one packet might not be sufficient for sending the readings from all the requested sensors, more than one packets can be sent by sensors answering a query back to the sink
- The protocol supports the possibility of cancelling a query that is active

Virtual Sensor Network

- jWebDust can manage multiple wireless sensor networks
- each WSN has a different control center, under a common jWebDust installation
- *abstracts* the actual network topology $\Longrightarrow$ user can control the sensors as if they were deployed under a single, unified, sensor network.



Benefits

1. significantly reduces the overhead of administering multiple networks
2. allows the integration of totally heterogeneous sensor networks

jWebDust – Remote Building Monitoring

AEOLUS Integrated Project IST-015964
Algorithmic Principles for Building Efficient Overlay Computers

- 1 UDRLS network: 11 MOTEIV
- 5 CTI networks: 8 MICA2, 10 MICA, 20 TELOSB, 60 SUN SPOT, 60 iSense, 5 camera sensors
- 1 UPATRAS network: 5 MICA2, 1 camera sensor
- 1 UNIGE network: 5 MICAz, 60 iSense
- 1 UOI network: 10 MICA2
- 1 UPC network: 30 SUN SPOT
- 1 INRIA network: 3 TELOSB

jWebDust – Remote Building Monitoring

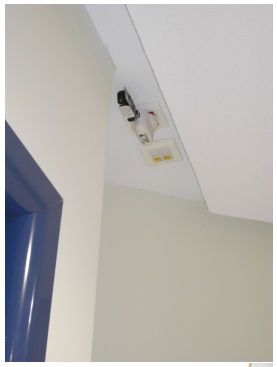
- We wish to develop a "*proof of concept*" application
  - Monitor indoor environments (temperature, light, motion)
  - Multiple places of interest at different locations
  - Monitor all locations (and data) through a single interface
  - Support different types of users with different access levels
- Monitoring Historic Sites for Cultural Heritage Preservation
  - Sensors are battery operated, Passive only
  - Sensors monitor humidity and temperature
  - Sensors cannot be easily accessed

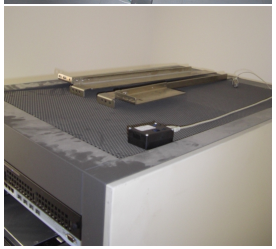
jWebDust – Remote Building Monitoring

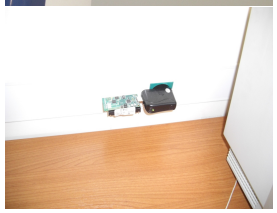
jWebDust – Remote Building Monitoring

- We wish to develop a "*proof of concept*" application
  - Monitor indoor environments (temperature, light, humidity)
  - Control appliances's power consumption
  - Multiple places of interest at different locations
  - Monitor & control all locations (and data) through a single interface
  - Support different types of users with different access levels
- Controlling Buildings for Community Energy Saving
  - Sensors monitor brightness, humidity and temperature
  - Actuators control office lights
  - Actuators control office ventilation
  - Sensors and Actuators are always on (connected to power plug)

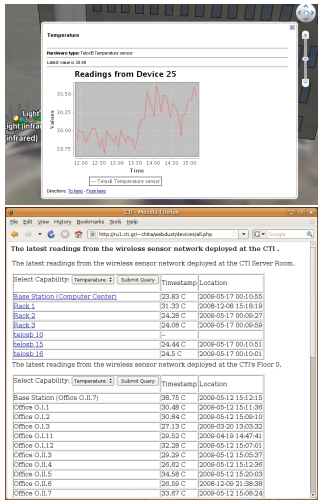
jWebDust – Remote Building Monitoring



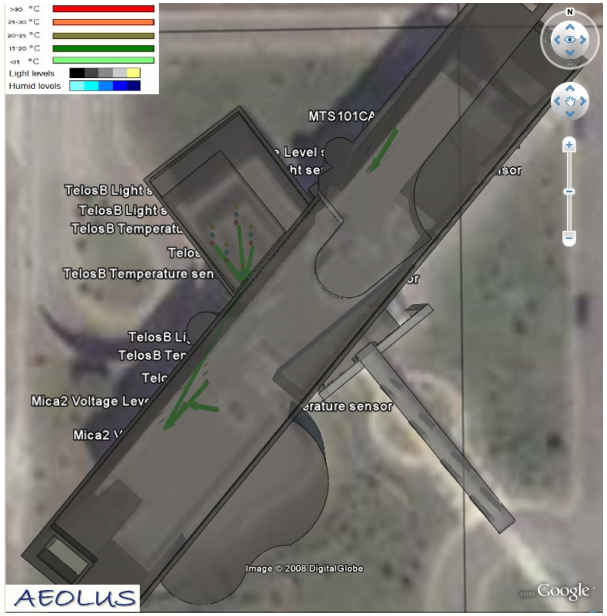


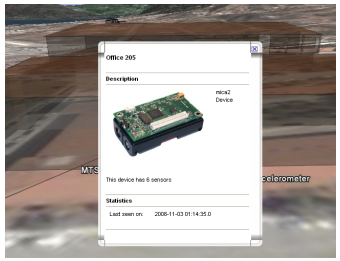


jWebDust – Remote Building Monitoring

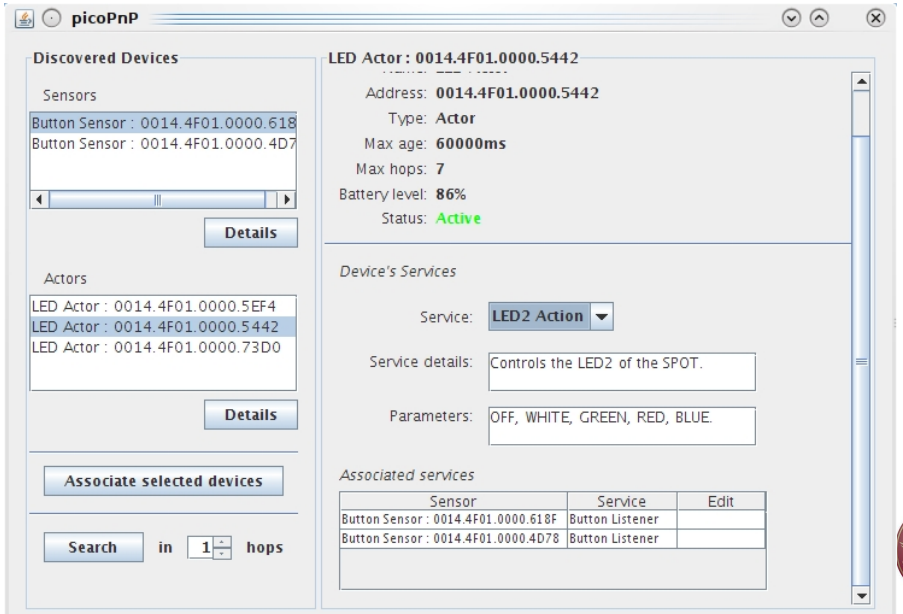


jWebDust – Remote Building Monitoring

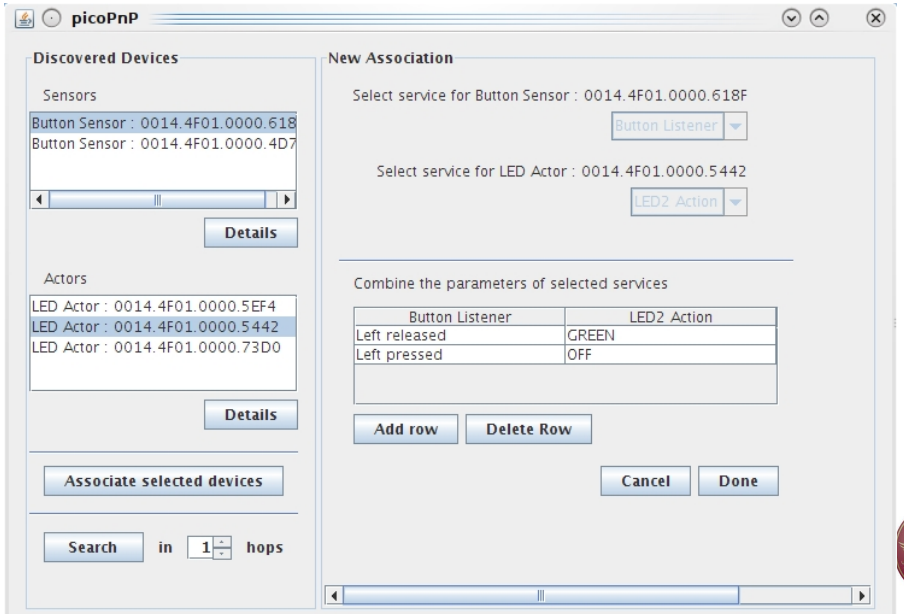




jWebDust – Remote Building Monitoring



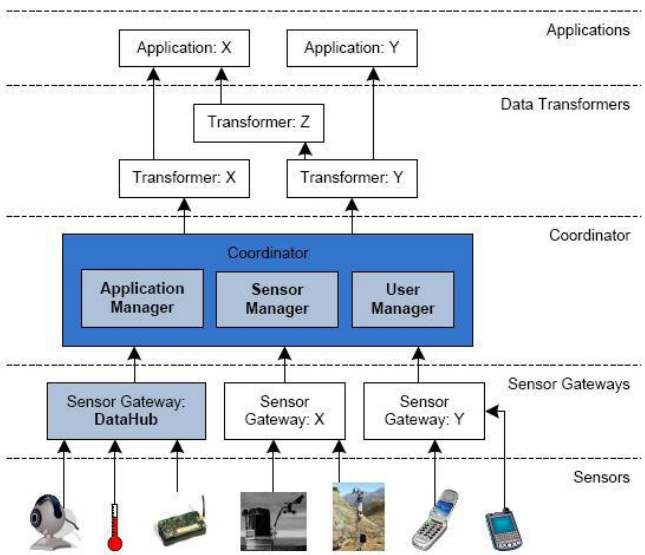
jWebDust – Remote Building Monitoring



SenseWeb : Key Points

- An infrastructure for shared sensing
- Peer-produced results greatly surpass monolithic architectures
- Based on sensor data produced by all sorts of networks
- Offer simplicity and generality
- Multi-tier architecture
- Web services

SenseWeb Architecture Overview



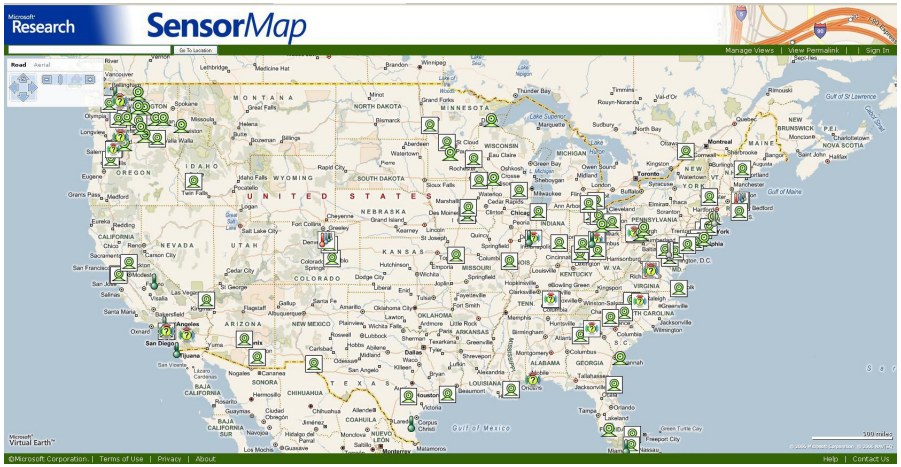
SenseWeb Architecture: Coordinator

- Central access point for:
  - Underlying sensor networks
  - Applications that use sensor data / manage WSNs
- SenseDB: storing, data and query aggregation, etc.
- Web services – Microsoft servers

SenseWeb Architecture: Sensor, gateways, proxies

- Underlying layers use provided web services to access the coordinators
- Implementations readily available for some WSN platforms
- Data transformers extract information out of sensor data – e.g., count free parking spaces from cameras, or contour maps creation
- Applications use coordinators and data transformers to produce results
  - SensorMap, a mashup application over Microsoft Virtual Earth
  - Swiss Experiment: monitor Swiss Alpic terrains

SensorMap Application based on SenseWeb



Windows Sensor and Location API

- Enables your applications to adapt to their current environment
- Location sensors (e.g., GPS)
  - your applications and gadgets can know exactly where they are, enabling them to provide more locally relevant content and functionality.
- Ambient light sensors
  - can allow your computer to automatically adjust your screen's brightness based on the current lighting conditions
- All other sorts of sensors

Modeling – Describing Sensor Networks

- Advertising sensor networks and nodes
- Advertising nodes capabilities
- Providing management and query functions
- Publish/subscribe functionality
- Formal methods - Standards

Sensor Web Enablement (SWE)

- Initiated from the Sensorwebs project (NASA)
- Set of protocols and services
- SensorML – Sensor Model Language
  - Very analytic model
  - Can describe almost every sensor
  - Complicated / overkill
- Interesting notions
  - Everything is a procedure
  - Inputs, procedures, outputs
  - Functional model of the sensor

Global Sensor Networks: Design Goals

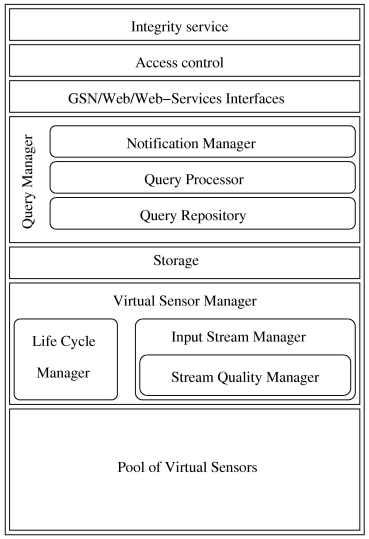
- Simplicity
  - minimal set of powerful abstractions
  - declarative specification of sensor networks and data streams
  - SQL-based query processing
- Adaptivity
  - low effort to add new types of sensor networks
  - dynamic (re-) configuration during run-time
- Scalability
  - large numbers of data producers and consumers
  - distributed query processing
  - distributed discovery of sensor networks
  - peer-to-peer architecture
- Light-weight implementation
  - no excessive hardware requirements
  - standard network connectivity

Central abstraction: Virtual Sensors

- A virtual sensor can be any kind of data producer
  - a real sensor, a wireless camera, a desktop computer, etc.
- Abstract from implementation details
  - physical sensors
  - a combination of other virtual sensors
  - 1 virtual sensor = n input data streams + processing + 1 output data stream
- Specification
  - metadata (identification, data, type, location)
  - structure and properties of input and output streams
  - declarative SQL-based specification of the data stream processing
  - functional properties related to stream quality management, persistency, error handling, life-cycle management, and physical deployment.

```
<virtual-sensor name="room-monitor" priority="11">
   <addressing>
      <predicate key="geographical">BC143</predicate>
      <predicate key="usage">room monitoring</predicate>
   </addressing>
   <life-cycle pool-size="10" />
   <output-structure>
      <field name="image" type="binary:jpeg" />
      <field name="temp"  type="int" />
   </output-structure>
   <storage permanent="true" history-size="10h" />
   <input-streams>
      <input-stream name="cam">
         <stream-source alias="cam"  storage-size="1" disconnect-buffer-size="10">
            <address wrapper="remote">
               <predicate key="geographical">BC143</predicate>
               <predicate key="type">Camera</predicate>
            </address>
            <query>select * from WRAPPER</query>
         </stream-source>
         <stream-source alias="temperature1"
                        storage-size="1m"
                        disconnect-buffer-size="10">
            <address wrapper="remote">
               <predicate key="type">temperature</predicate>
               <predicate key="geographical">BC143-N</predicate>
            </address>
            <query>select AVG(temp1) as T1 from WRAPPER</query>
         </stream-source>
      </input-stream>
   </input-streams>
</virtual-sensor>
```

GSN architecture



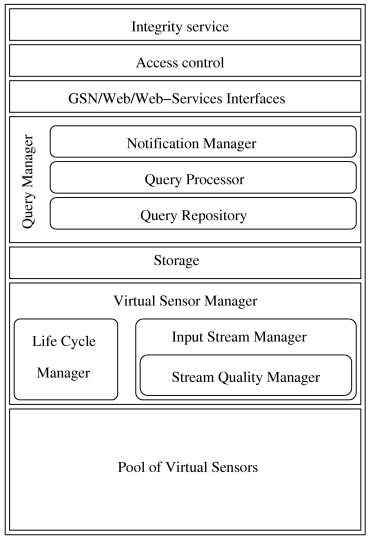
Node

- Each GSN node hosts a number of virtual sensors

Virtual Sensor Manager

- provides access to the virtual sensors
- manages the delivery of sensor data

GSN architecture



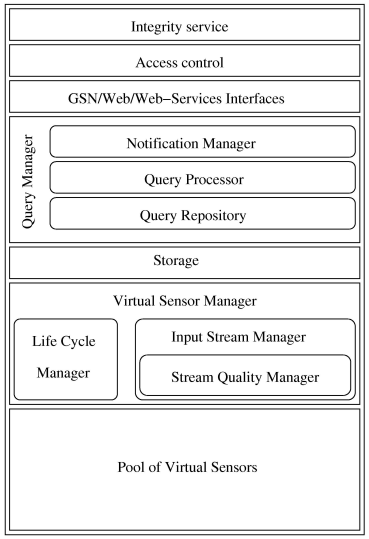
Life-cycle manager

- provides and manages the resources provided to a virtual sensor
- manages the interactions with a virtual sensor
- ensures stream quality
- manages the life-cycle of sensors

Storage Manager

- persistent storage for data streams

GSN architecture



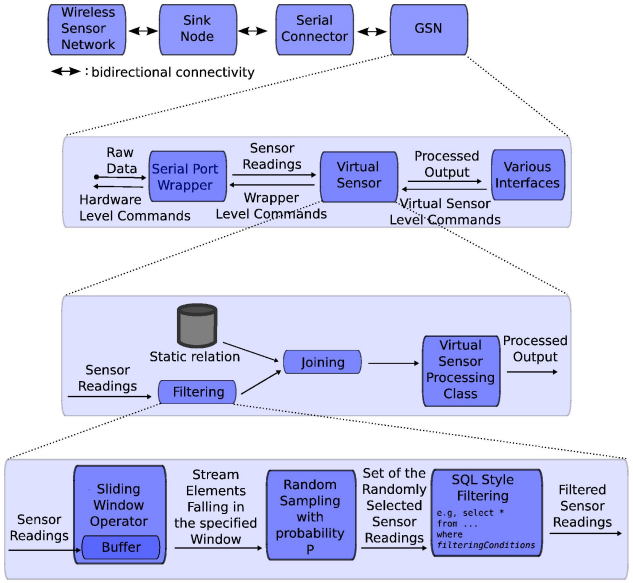
Query manager

- manages active queries
- query processing
- delivery of events and query results to registered, local or remote consumers

Top layers

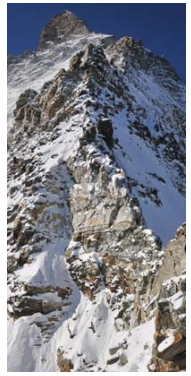
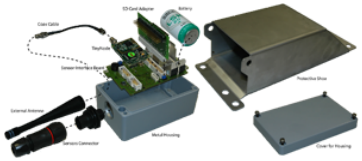
- access, access control, and integrity

Local sensor data processing

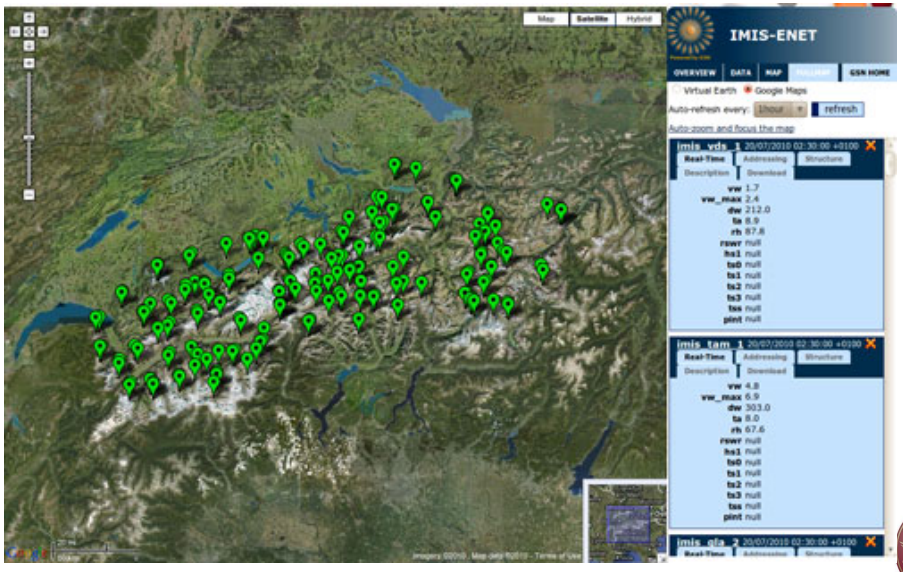


PermaSense – Sensing in High Alpine Environments

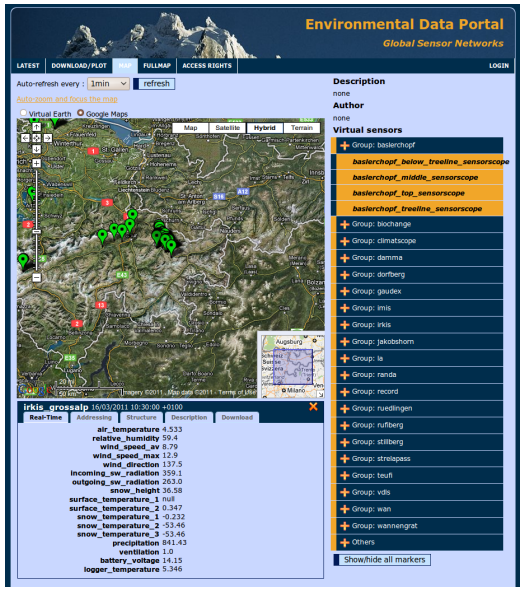


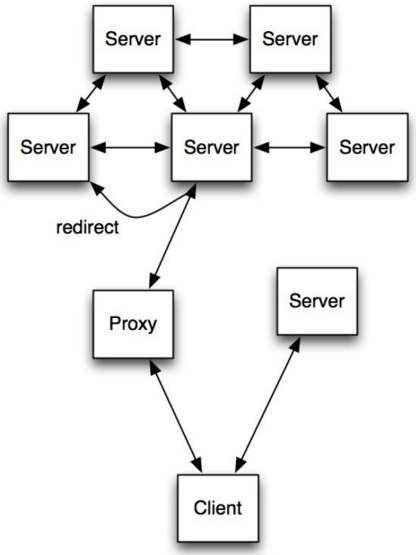

PermaSense – Sensing in High Alpine Environments



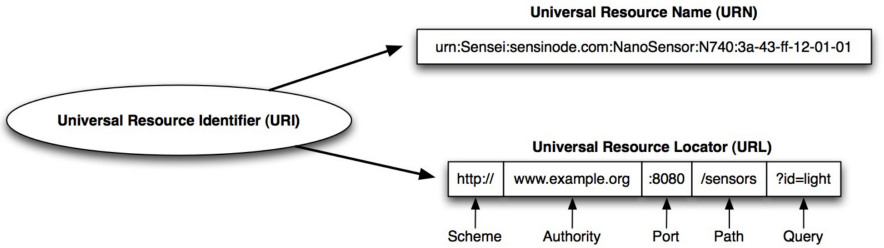
PermaSense – Sensing in High Alpine Environments



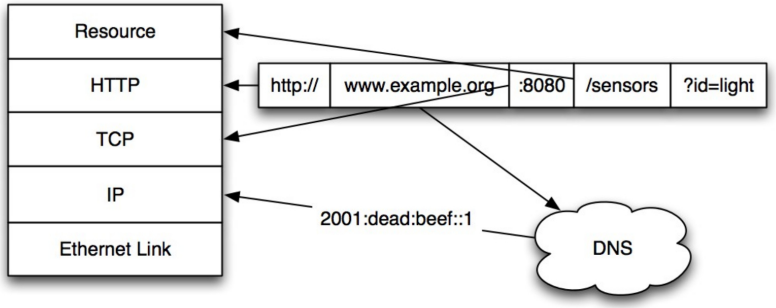
The Web and REST



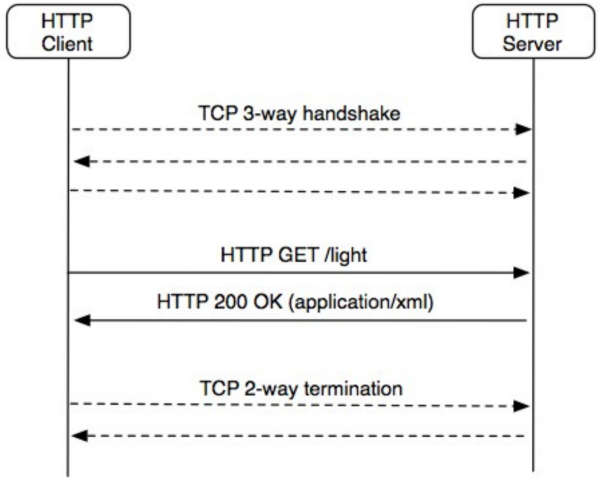
Web Naming



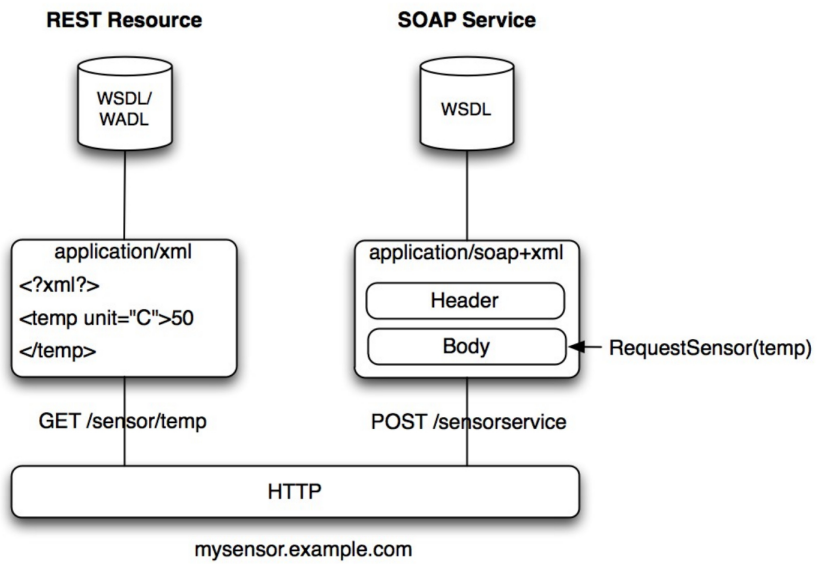
URL Resolution



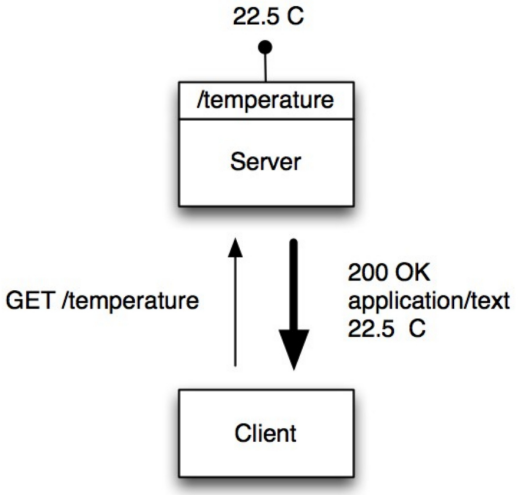
An HTTP Request



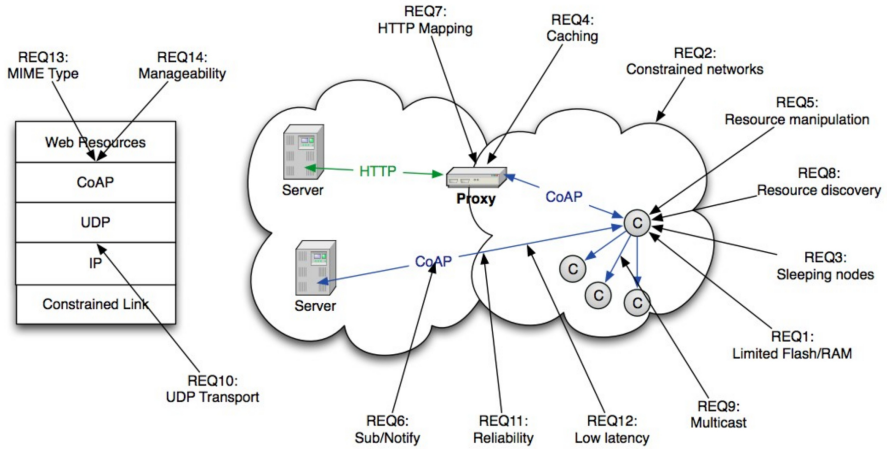
Web Paradigms



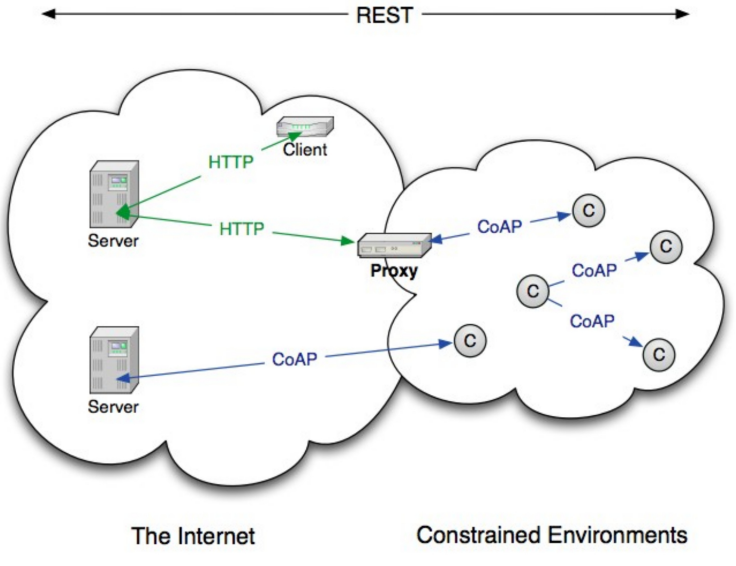
A REST Request



CoAP Design Requirements



The CoAP Architecture



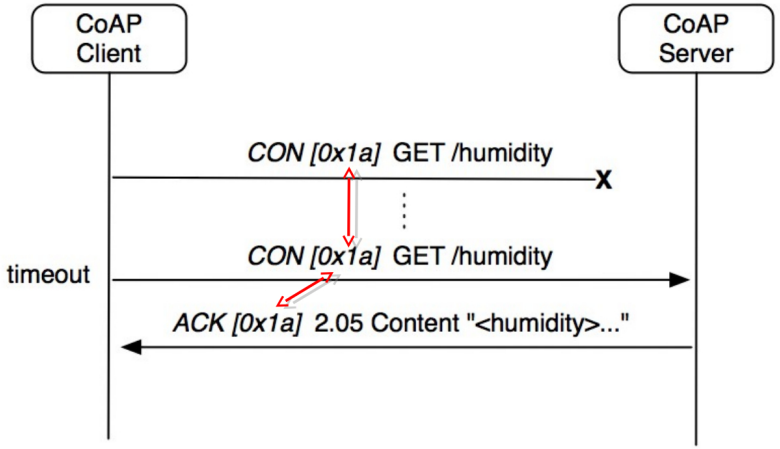
The CoAP Protocol

- A very efficient RESTful protocol
- Ideal for constrained devices and networks
- Specialized for M2M applications
- Easy to proxy to/from HTTP
- Does not replace HTTP
- Is not a cut-down HTTP version
- Not just for resource-constrained networks

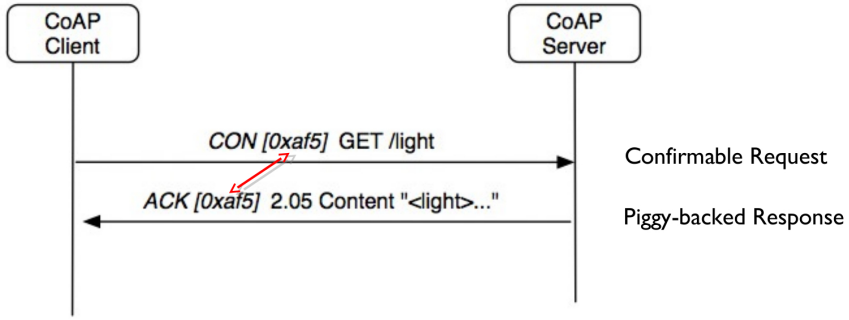
CoAP Features

- 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

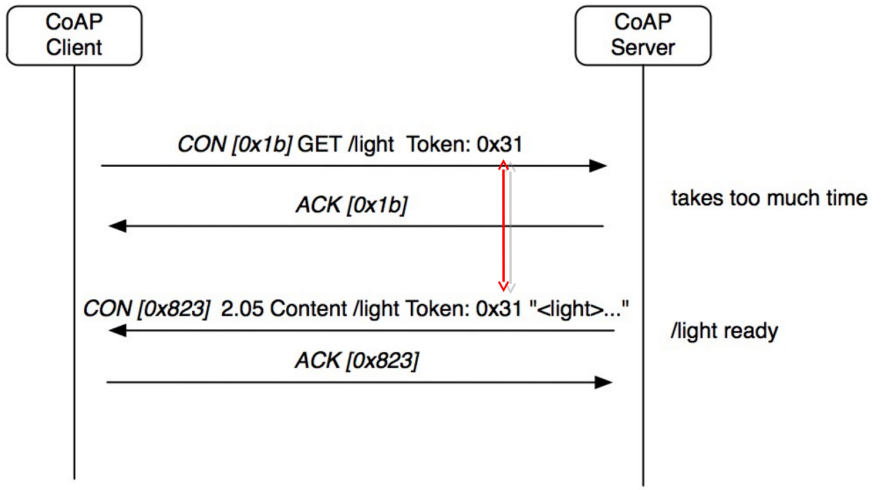
Request Example



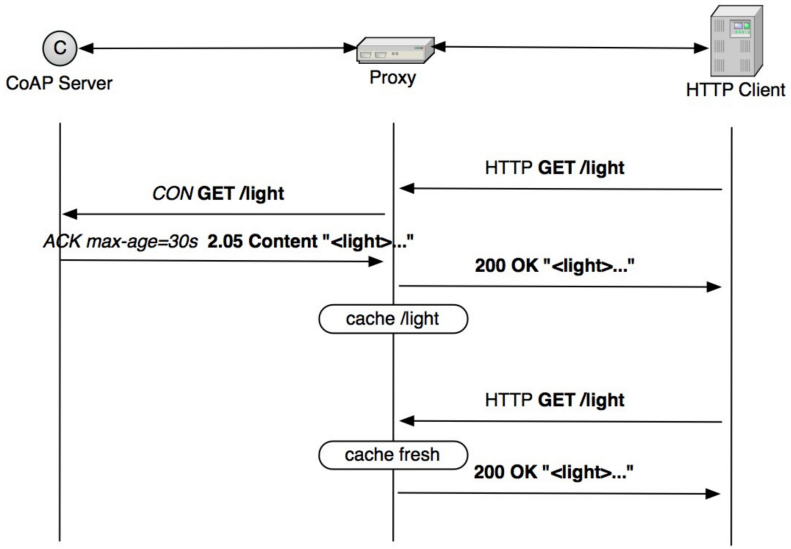
Dealing with Packet Loss



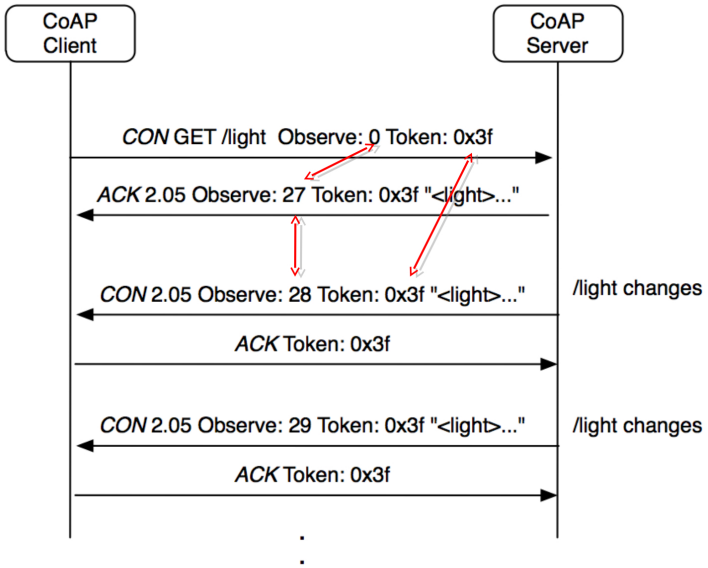
Separate Response



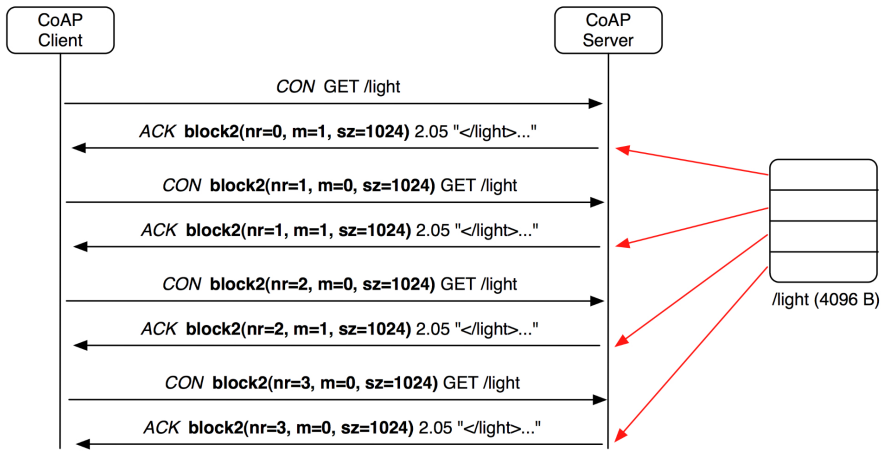
Proxying and caching



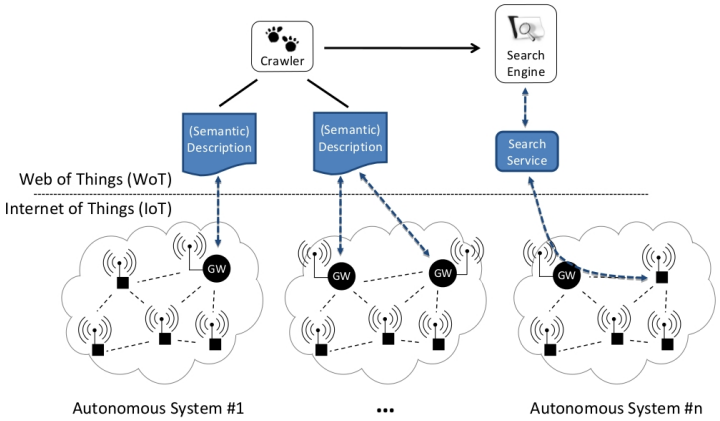
Observation



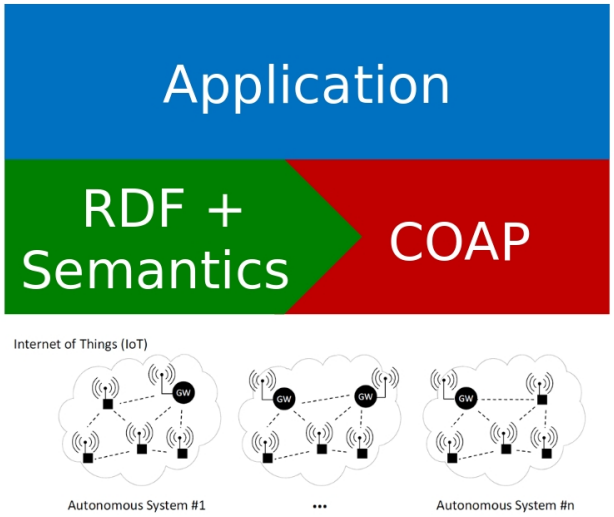
Block transfer



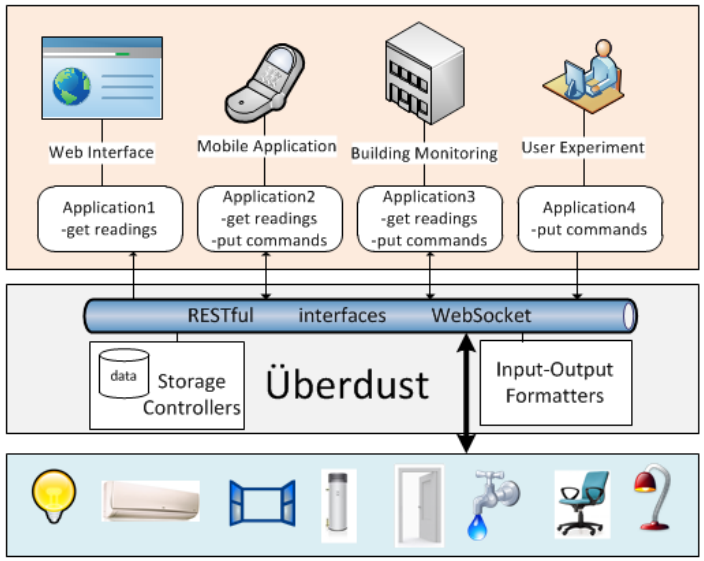
- Transparent & simplified network access
- Address the needs of a wide variety of IoT applications,
- Follow open standards,
- Light-weight data models at IoT device level.



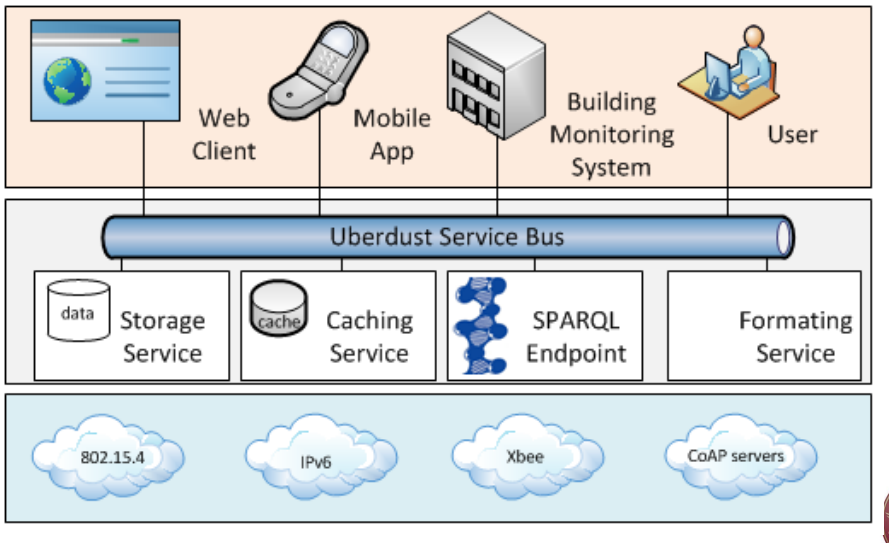
Application := Data + Services



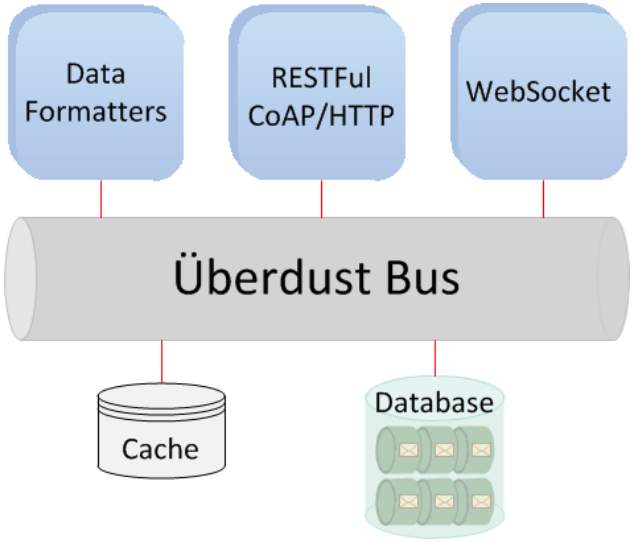
Web Integration Layer



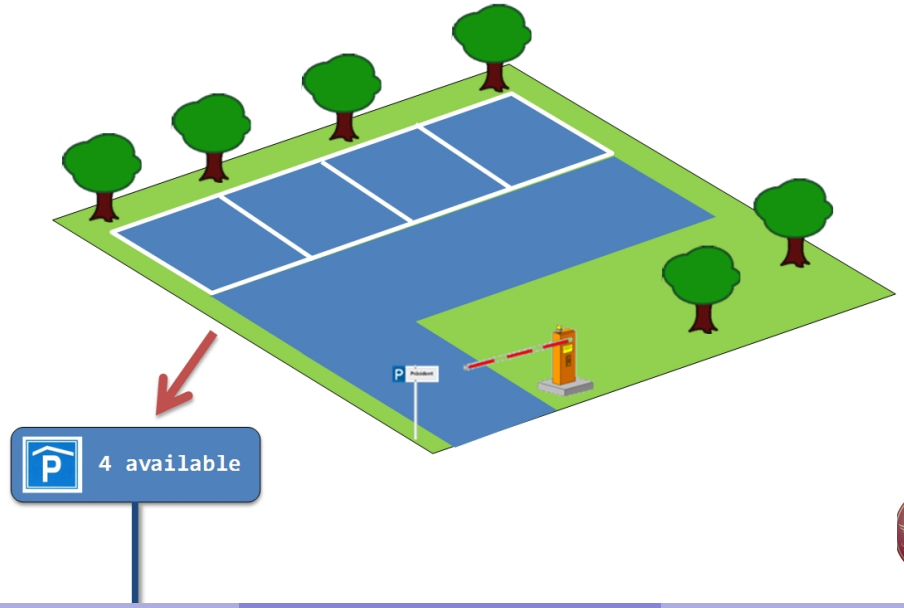
Web Integration Layer



Web Integration Layer



Smart Parking



Smart Parking

Existing deployments using IoT technologies:

- Custom-tailored IoT firmware,
- Nodes forward data to a well-known base station,
- Base station sends data to display at fixed IP.

Smart Parking

New Approach:

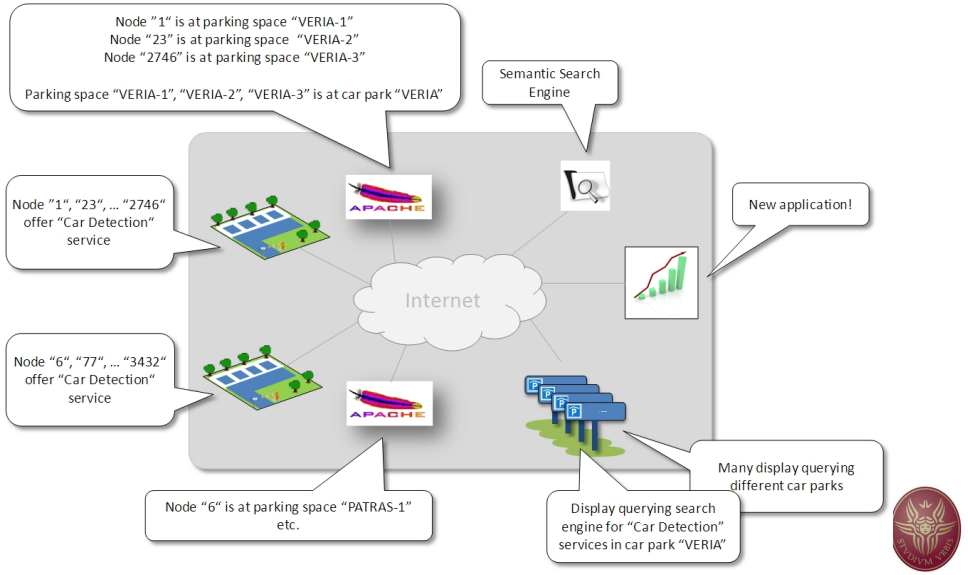
- Generic applications on nodes and base stations (not custom-tailored for a specific application)
- New applications can be written without requiring any modifications to nodes or gateways
- Additional nodes can be added at application run-time w/o changes to the application
- Fully decentralized

Smart Parking

- Nodes
  - Transmit self-description to base station
  - I am node “23” offering service “car detection”
  - Base station publishes this on semantic web page
- Car park operator provides semantic web page
  - Node “23” is on parking “Space 1”
  - “Space 1” is located in car park “VERIA”
- Search engine crawls these web pages
- Display
  - Searches for “car detection” services in car park “VERIA-1”
  - Queries the resulting services and displays processed result

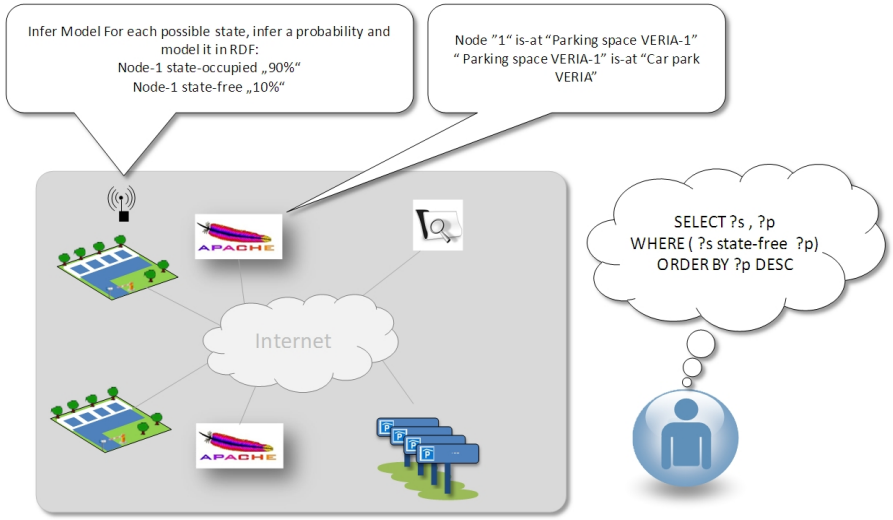
Web Access to WSN | Internet of Things | Web of Things
Semantic-Service Provisioning for the IoT

Smart Parking



Web Access to WSN | Internet of Things | Web of Things
Semantic-Service Provisioning for the IoT

Smart Parking

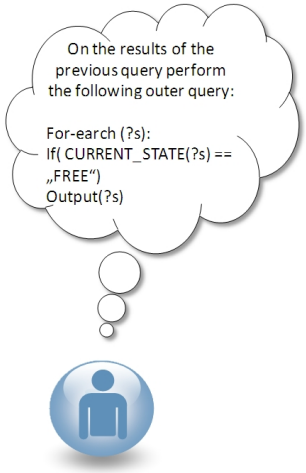


Web Access to WSN | Internet of Things | Web of Things
Semantic-Service Provisioning for the IoT

Smart Parking

?s	?p
VERIA-2	95%
VERIA-1	90%
VERIA-3	10%

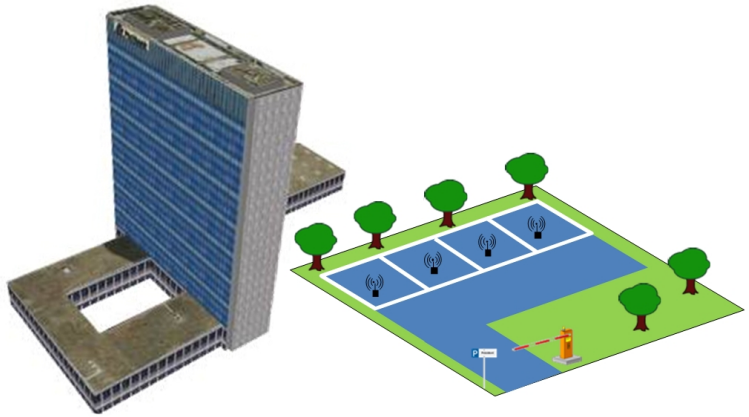
Smart Parking



?s	?p
VERIA-2	95%
VERIA-1	90%
VERIA-3	10%

Web Access to WSN | Internet of Things | Web of Things
Semantic-Service Provisioning for the IoT

Smart Parking



Car Park empty → Switch off all air conditioning in the building