Toward Web Enhanced Building Automation Systems

Gérôme Bovet and Antonio Ridi and Jean Hennebert

Abstract The emerging concept of Smart Building relies on an intensive use of sensors and actuators and therefore appears, at first glance, to be a domain of predilection for the IoT. However, technology providers of building automation systems have been functioning, for a long time, with dedicated networks, communication protocols and APIs. Eventually, a mix of different technologies can even be present in a given building. IoT principles are now appearing in buildings as a way to simplify and standardise application development. Nevertheless, many issues remain due to this heterogeneity between existing installations and native IP devices that induces complexity and maintenance efforts of building management systems.

A key success factor for the IoT adoption in Smart Buildings is to provide a loosely-coupled Web protocol stack allowing interoperation between all devices present in a building. We review in this Chapter different strategies that are going in this direction. More specifically, we emphasise on several aspects issued from pervasive and ubiquitous computing like service discovery.

Finally, making the assumption of seamless access to sensor data through IoT paradigms, we provide an overview of some of the most exciting enabling applications that rely on intelligent data analysis and machine learning for energy saving in buildings.

Gérôme Bovet (1)(2), Antonio Ridi (2)(3), Jean Hennebert (2)(3)
(1) Telecom ParisTech, 46 rue Barrault, 75013 Paris, France

(2) University of Applied Sciences Western Switzerland, Bd de Pérolles 80, 1700 Fribourg, Switzerland

(3) University of Fribourg, Bd de Pérolles 90, 1700 Fribourg, Switzerland

1 Introduction

In the last decade, we have become more and more concerned by the environmental dimension resulting from our behaviours. Further than the ecological trend, the interests are also economics-centred due to the raising cost of the energy. Representing 20% to 40% of the global energy bill in Europe and USA, buildings are a major source of energy consumption, actually more important than industry and transportation [27]. In a building, half of the energy consumption comes from the Heating, Ventilation and Air Conditioning systems (HVAC), followed by lighting and other electrically operated devices. In offices, HVAC, lighting and electrical appliances together reach about 85% of the total energy consumption.

For these reasons, the reduction of building energy consumption has become an important objective. Many works are currently undertaken towards renovation and improvement of building insulation. The other facet is to leverage on energy efficiency that involves better usage of HVAC equipment taking into account local production and storage capacity as well as temporal occupation of the rooms. In simplified terms, the aim is to optimize the ratio between the energy savings and user comfort.

This objective requires a clear move from state-of-the-art conventional building automation systems to advanced information systems leveraging on (1) a variety of interconnected sensors and actuators, (2) a unified management of heating, lighting, local energy production or storage and (3) data modelling capacities to model room usage and predict user comfort perception.

Most automated buildings are currently working with dedicated building networks like KNX [3] or EnOcean [2]. These networks are specifically conceived for the purpose of building automation, including all layers of the OSI model starting from the physical to the application one. In such settings, a central Building Management System (BMS) is typically connected to the network and manages the equipments by implementing the operation rules of the building.

In many buildings, we observe the coexistence of different network technologies, often caused by the installation of new equipments answering specific physical constraints, for example wiring or power supply. The protocols are often relying on proprietary layers and this heterogeneity actually leads to two situations. In the first one, several BMS are coexisting and share the management of independent equipments, making difficult any global optimisation. In the second one, a unique but more complex and costly BMS is used where bridges to the different protocols are integrated. Without prejudging on the strategies of technology providers, we can reasonably converge to the fact that there is a lack of standardisation in building automation systems, at the network level and, probably more importantly at the application level. BMS could largely benefit of a common protocol stack compatible with any device.

Thanks to the miniaturization of electronics and increasing computing power, devices offering native IP connectivity are appearing. Sensor networks based on 6LoWPAN or Wi-Fi are nowadays in competition with classical building networks. Everyday objects are now able to connect to IPv4 and IPv6 networks, offering new

functionalities like machine-to-machine communications. This has led to the emergence of *Internet-of-Things* paradigms (IoT), a research field trying to find answers in how to connect objects to the Internet from the network point of view, i.e. covering the first four layers of the OSI model.

Going up to the application layer, the heterogeneity problem is even worse as there are currently no strong standards defining the *semantic* of the building resources, i.e. an expression of device and service capabilities. The paradigms of the Web-of-Things (WoT), which can be viewed as the natural extension on top of the Internet-of-Things, are here proposing to rely on web application standards. Arguably, these standards are more like a set of best practices than real standards. In the vision of IoT and WoT, any device embeds a Web server offering lightweight Web services for interaction. The Application Programming Interfaces (APIs) of the services often rely on RESTful principles, providing natural ways to embed the semantics in the communication protocol.

In this Chapter we will present the actual state-of-the-art in the field of IoT in smart buildings, putting into evidence remaining ongoing challenges. Some propositions overcoming open problematic will be discussed, especially regarding the integration of existing building automation systems in the core IoT and their respective discovery. Finally, making the assumption of a building where IoT paradigms are fully integrated, we will provide an overview of some enabling applications that rely on data analysis and self-learning algorithms for energy saving in buildings.

2 Integrating Building Automation Systems in the IoT

Many new or renovated buildings are nowadays equipped with automation networks. We can here mention office buildings, factories and even private households. The relative high investment costs have an impact on the payback period which is rather high, often around ten years. A sudden change of technology is therefore not conceivable. We envision here the IoT as adapting itself to existing installations and thus encouraging a smooth transition until building automation systems natively support it. Meanwhile, a mix of different technologies will probably coexist in buildings.

In this section, after reviewing existing building automation systems and technologies, we propose a Web-oriented protocol stack able to solve the heterogeneity problem between sub-systems. Concrete application scenarios will serve as basis of discussion.

2.1 Related Work

Historically, buildings are equipped with networks especially designed for automation purposes and offering services tailored to buildings. We can here mention sev-

eral technologies like BACnet, LonWorks, KNX and EnOcean. The physical mediums are typically not restricted to certain types, like KNX can support twisted pair, RF, power line and even Ethernet. Because of the custom protocols used for the transport and networks layers, it is not conceivable to shift the application layer to a more common network like IPv6. Some works have proposed to rely on multi-protocol devices [13]. The drawback of this approach resides in the integration cost of such devices that are, anyway, quite non-existent on the market. Another solution consists of providing gateways. As illustrated in Fig. 1, gateways can operate according to two modes, a *N-to-N* protocol mapping or a *N-to-1** approach mapping all sub-systems to only one central protocol. The *1** refers to a new protocol stack suited for IoT interactions. In the N-to-N case, gateways between BAS translate telegrams from the originating network to its destination. Those gateways have knowledge about each protocols composing the stacks of each network. Although this approach solves the heterogeneity across networks, it induces some limitations. First, it is possible that not all capabilities of a BAS can be mapped to another one, thus restricting functionalities. Secondly, this approach requires $\frac{n*(n-1)}{2}$ mappings between BAS, representing a considerable effort.

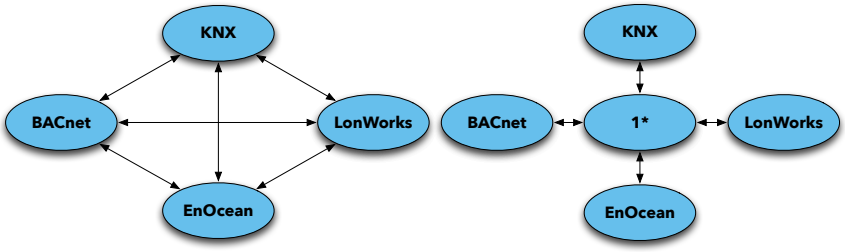

Fig. 1 N-to-N (left) and N-to-1* (right) approaches for protocol mapping in buildings.

In contrast to the N-to-N approach, the N-to-1* one considerably simplifies the number of gateways needed to n by introducing a common technology. The key challenge resides in the 1* technology where no standard is currently defined. Its components have still to be identified, according to Internet-of-Things constraints. A remaining decision has to be taken when integrating BAS into the IoT regarding the gateway position in the network. There are two extremes, either centralizing the services on a single node, or migrating them as close as possible to field devices. Centralizing the access at a backbone server brings some advantages in terms of maintenance, even if scalability problems may arise. Putting the services at the field level requires devices with more computational power but allows a direct native interaction between sensors and actuators with Web services over IP. More specifically, we can describe four different integrations styles, as illustrated in Fig. 2.

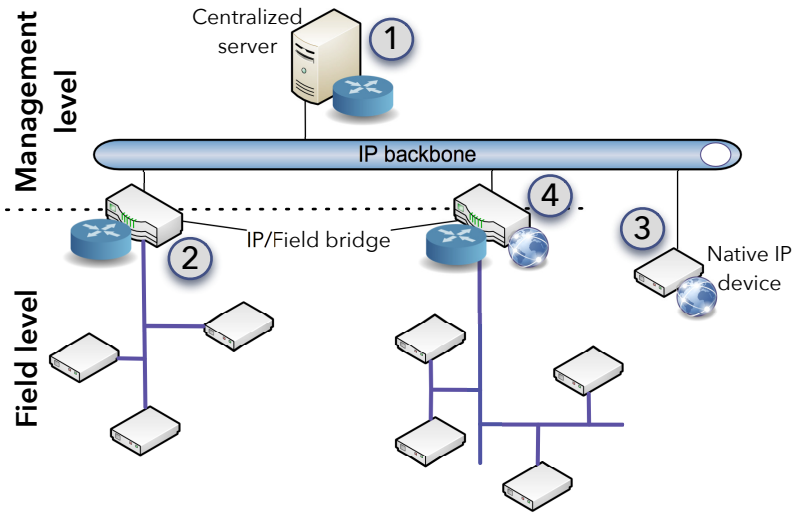

Fig. 2 Building automation systems integration styles

1. **On a centralized server:** this approach is currently the most used one where a server at the IP backbone handles all the interaction with the BAS. It allows an integration into enterprise systems.
2. **On a bridge:** by encapsulating field telegrams into IP packets one can interact with the BAS. Although being a more decentralized approach, the same problem regarding the application level remains open.
3. **On field devices:** IP enabled devices offering Web services for intercommunication with other participants of the network.
4. **Multi-stack bridges:** allow a more decentralized approach. By implementing the field stack and the IoT stack, mappings between Web services and endpoints are possible. By proceeding this way remote devices acting as clients are not aware that the final device resides on another type of network even having no IP connectivity.

A few research and proofs of concepts have already been proposed for IoT gateways. We can here cite the Universal Device Gateway developed in 2008 for enabling interoperability among heterogeneous communication protocols like ZigBee, X10 and KNX with an IPv6 compliant architecture [4]. More recently, a mapping from KNX endpoints, also known as datapoints, to oBIX objects has been proposed [26][24]. The oBIX framework was developed for facilitating data interchange with BAS, relying on a XML object model representing devices. Those endpoints are accessible over Web services. The oBIX approach is quite similar to the WS-* stack of classical SOAP Web services. The oBIX HTTP server waits for requests and then translates them to KNX telegrams. Here, depending on the type of request (GET or POST), the action performed will be either a query for an endpoint

status, or an update of a state. Although providing a common mapping, the proposed approach leaves certain issues open as for example the integration in existing installations.

In another work, the full paradigm of the Web-of-Things for smart buildings has been investigated in the context of KNX and EnOcean gateways in [7][6]. This work shows that Web resources can be mapped to Building Automation System (BAS) endpoints using RESTful APIs, which are lightweight Web services based on loose-coupling. In addition to this, a semantic for URLs pointing to resources is proposed by including the location of the device inside the DNS name of the resource, thus providing a clear way for identifying them.

Devices becoming more and more IoT aware, the question regarding the standardization of the application layer and semantics remains open. A concrete proposition of an application layer compatible with IoT's paradigms was proposed in [20] [19]. The main contribution resides on the proposition of IoTSyS, an entire protocol stack for integrating BAS in the Internet-of-Things. IoTSyS relies on technologies such as Constrained Application Protocol (CoAP), oBIX and EXI, as illustrated in Fig. 3. CoAP, which is a lightweight version of HTTP pursuing the RESTful concept, is used as application transport protocol. The advantage of CoAP over HTTP is to be optimized for constrained devices by having much less overhead. Then, at the application layer, the oBIX XML schema is used to describe endpoints and to define contracts according to device functionalities, such as, for example, push button, relay or temperature sensor. The contracts can be annotated by using some data models as sensorML or domoML allowing a better processing by machines. As XML payload is not suited for small devices or constrained devices, the EXI compressor is used, which allows to reduce the size of the exchanged data. In order to perform the best possible compression, a schema describing the contracts is used on both the server and the client.

Although the IoTSyS solution fulfils some requirements of the IoT, it induces one major drawback, which is being not loosely-coupled. Indeed, every client must have knowledge about the IoT contracts schema in order to be able to decompress the XML. This approach restricts the possible evolution of the protocol stack because of the difficulty to update the schema on existing devices. This situation would lead in multiple versions of contracts distributed across networks, resulting in incompatibilities. In addition, using a lightweight protocol (CoAP) while adding some complexity by relying on oBIX could be seen as a contradiction. Additionally, many applications are nowadays built upon Web technologies like HTML and AJAX following the Web 2.0 concept. Integrating specific technologies like oBIX and EXI is actually adding more complexity to those applications and require special knowledge from developers. Despite the fact that some semantics are needed at least for describing resources, we believe that existing Web technologies have to play a more important role.

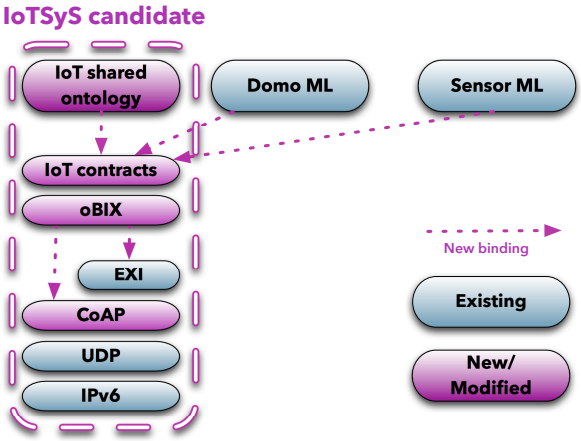

Fig. 3 Protocol stack for IoT building automation systems as proposed by [20]. Blocks that are introduced in this proposition or requiring some adaptations relative to their traditional use are presented in pink. Existing blocks requiring no adjustment are presented in blue. New bindings between blocks are shown with pink arrows.

2.2 A Web-Oriented Protocol Stack

The device accessibility can be decomposed into four sub-layers, which are resource identification, linking, resource description and operation on resources [16].

Resource identification - The location of a device, which is often an important information in BAS can be included in the DNS name describing the resource. For example *coap://temp.office05.00.c.company.org* would represent the temperature sensor in room 05, on the ground floor of building C of a company. The final endpoint can then be further accessed by specifying the resource or even a sub-resource. We can here illustrate this with a temperature sensor offering two contracts, one in Celsius *../temp/celsius* and the second one in Fahrenheit *../temp/fahrenheit*. We here reuse the concept of Web resource and URI for identifying unique device endpoints. In this approach, a traditional DNS architecture can be used. However we recommend to rely on the distributed approach of DNS, namely mDNS that is also compatible with classical DNS clients. The difference lies in the queries that are sent over multicast instead of unicast. This approach is more scalable and fault-tolerant as the knowledge about the DNS is distributed across nodes.

Linking - In the concept of WoT and Resource Oriented Architecture (ROA), an important aspect is the ability to discover resources by following links. A resource should provide links back to its parent and forward to its children. This allows to crawl knowledge about resource hierarchy. When applying this concept to the field of building automation, it comes out that devices should be linked to their location and endpoints. To achieve this, we propose to use an implicit and an explicit way. By decomposing the host part of the URI and discarding the first part, one can go

back in the building location hierarchy (e.g. coap://kitchen.ground.home.com -> coap://ground.home.com). This way is especially thought for humans. Machine-to-machine interaction could prefer the explicit way. There, each resource description, i.e. location, device or endpoint, must provide absolute URLs to its parent or children. Nonetheless, linking is closely related to resource description as the format and semantic used will influence how linking is provided.

Resource description - Discovering the capabilities of a resource is also a very important aspect in machine-to-machine communications. Not only the semantic but also the format is decisive to increase the chance of self-understanding. Regarding the format, a frequent suggestion is to use JSON instead of XML. First, JSON is more lightweight than XML, and that in terms of message size and parsing time [25], therefore better suited for devices with limited capabilities. Moreover, it can directly be parsed in JavaScript, allowing a faster integration into Web application and supporting the concept of mashups. Concerning the semantics, we here propose to rely on the concept of RDF and ontologies forming together the semantic Web. Further details about semantics and ontologies are given in Section 2.3.

Operation on resources - The last step is the execution of an operation on the resource. For devices with limited resources, the CoAP protocol is probably one of the best candidate. CoAP is a simplified version of HTTP, aiming at reducing the overhead size. Gateways translating HTTP to CoAP, and vice-versa, are available, exposing CoAP devices over the HTTP protocol. In a similar way as for HTTP, CoAP offers GET, POST, PUT and DELETE operations that have a standard semantic according to REST services, as for example:

- **GET** is used for retrieving the current state of a resource, e.g read the actual temperature.
- **PUT** is used to update the state of a resource, e.g. switch a power outlet.
- **POST** is used to create a new resource, e.g. register for event notification.
- **DELETE** is used to delete a resource, e.g. delete a threshold on a device.

Figure 4 summarizes the protocol stack for building automation systems relying on the Web-of-Things paradigm. The strengths of our WoTBAS (Web-of-Things Building Automation Stack) proposition reside in taking part of the best practices of the Web, being lightweight and loosely-coupled.

2.3 Semantics and Ontologies

Still a topic of research, semantics and ontologies are potential technologies to define common languages and representations facilitating machine exchange. The Web Service Description Language (WSDL) is currently the most used language for describing Web Services. WSDL is often combined with the Simple Object Access Protocol (SOAP) allowing the exchange of structured information about Web services. Although WSDL brings some standardization in terms of method prototyping by describing the attributes and return value, it is not intended for describing

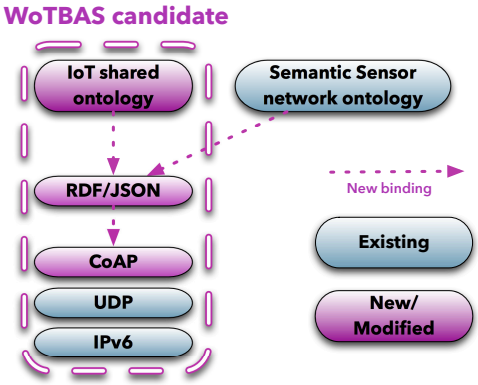

Fig. 4 Protocol stack for WoT building automation. Blocks that are introduced in this proposition or requiring some adaptations relative to their traditional use are presented in pink. Existing blocks requiring no adjustment are presented in blue. New bindings between blocks are shown with pink arrows.

the *semantic* of Web resources. For this reason, many XML schemas and other data models were developed trying to describe processes and physical environments [14]. Nowadays the RDF language is used together with ontologies [32]. Ontologies represent a vocabulary and association of definitions, while RDF is the common language to express the meta-data. Statements are expressed with *triples* composed of subject-predicate-object. For example the notion "The sensor is of type temperature" in RDF is represented with the following triple: a subject denoting "The sensor", a predicate "is of type" and an object denoting "temperature". In order to have standardized triples, the use of ontologies is strongly recommended. One can find various ontologies developed for specific contexts. The Semantic Sensor Network Ontology (SSN) is based around concepts of systems, processes and observations [15]. It supports the description of the physical and processing structure of sensors. However notions of units and locations are not part of it. One can include other ontologies like DOLCE Ultra Lite (DUL) [1]. Before defining the SSN ontology, the working group of the W3C performed an analysis of all existing data models representing sensors. The best practices of each data model were included in the specification of the SSN ontology. Even if the ontology is quite complete and allows to express a lot of properties, there is always the possibility to expand it with new classes and definitions. For example one could extend the sensor class in order to define a concrete sensor type with its own specific properties as for example temperature or presence. When further pushing this concept, one could imagine to create sub-classes of concrete sensor type representing a sensor model. This leads to an important question: should the ontology implement a low or a high level of abstraction? Having a low level of abstraction by defining sensor models in the ontology seems here not a good choice as it would result in a heavy-coupling between devices. Also, the ontology would be huge considering all types of sensors existing on the market. Addition-

ally the evolution of the ontology would be extremely limited, or requiring frequent updates. Such an approach would result in incompatibilities across devices of different versions. Keeping a high level of abstraction certainly barely complicates the machine intercommunication but allows much more scalability in terms of evolution. One strength of RDF is that one can expand the RDF description with its own properties that are not present in the followed ontology. Clients who do not know the non-standard triples will simply ignore them. The listing 1 shows an example of what could be the RDF description in Turtle format of a humidity endpoint on a temperature sensor using the SSN ontology.

Listing 1 Example of a humidity endpoint on a temperature sensor description using RDF with the Semantic Sensor Network Ontology

```
@prefix rdf: <http://www.w3.org/1999/02/22−rdf−syntax−ns#> .
@prefix ssn: <http://purl.oclc.org/NET/ssnx/ssn> .

<coap://temp.kitchen.home>
  rdf:type ssn:SensingDevice ;
  ssn:observes <http://purl.oclc.org/NET/muo/ucum/physical−quality/humidity> ;
  ssn:hasMeasurementCapability <coap://temp.kitchen.home/hum> .

<coap://temp.kitchen.home/hum>
  rdf:type ssn:MeasurementCapability ;
  ssn:hasMeasurementProperty <coap://temp.kitchen.home/hum/accuracy> ;
  ssn:hasMeasurementProperty <coap://temp.kitchen.home/hum/sensivity> .

<coap://temp.kitchen.home/hum/accuracy>
  rdf:type ssn:Accuracy ;
   rdf:value 1 .

<coap://temp.kitchen.home/hum/sensivity>
  rdf:type ssn:sensivity ;
  rdf:value 0.1 .
```

Another important concept of RDF is its associated query language SPARQL recognized as a key technology of the Semantic Web [33]. With SPARQL one can crawl over resources for retrieving only the interesting ones meeting some criteria. Queries are expressed with a dedicated language also expressing triple composed of conjunctions-disjunctions-patterns. The query can filter results according to literal, numeric data type properties, which makes it very similar to SQL. SPARQL offers four types of queries: LIKE for extracting raw values, CONSTRUCT for obtaining valid RDF, ASK for obtaining a simple true/false response on an endpoint, and finally the DESCRIBE for obtaining an RDF graph. In our context we will only use the SELECT type of query to return the URL of the resource. This information alone is sufficient as the client can then ask for the whole description by accessing the resource directly. However the client can also specify other attributes than the URL in the SELECT section of the query if needed. To illustrate what a SPARQL query for discovering devices could look like, we rely on a practical example as follows: what are the resources of type temperature with a value higher than 25° Celsius located in the kitchen? Listing 2 shows the resulting SPARQL query.

Listing 2 Example of a SPARQL query to look after temperature sensors with a value higher than 25° Celsius located in the kitchen

```
@prefix ssn: <http://purl.oclc.org/NET/ssnx/ssn> .
SELECT ?node WHERE {
  ?node a ssn:Sensor ;
    ssn:observes <http://purl.oclc.org/NET/muo/ucum/physical-quality/temperature> .
    dul:hasLocation "kitchen" ;
    rdf:value ?lv ;
   FILTER (?lv > 25) .
}
```

3 Discovery

Discovery of device and service is crucial in the IoT approach that rely on participating objects distributed all over the network. This case is also true for building automation. For example, we want the overall system able to discover if a temperature sensor located in a certain room is available. We can here distinguish two discovery strategies. Either by following links across resources, or by asking clients to provide their capabilities through requests. As the first one has already been discussed in the previous Sections, we will here focus on the latter.

3.1 Related Work

A key concept for Web-of-Things discovery is defined with CoAP through the use of the *./well-known/core* resource which will respond with all accessible services on the device [20, 19]. One major drawback of this concept is in the fact that previous knowledge about the device's DNS or IP address is required. Furthermore the response is limited to few predefined attributes giving little information about the services themselves. Regarding service discovery, well-spread protocols like UPnP's SSDP [12], SLP [17] or DNS-SD [9] have emerged in the last years. They are mostly implemented in operating systems and computer peripherals like printers or PDAs. They are based on a multicast approach allowing to discover the existence of devices according to user role and network scopes. DNS-SD, for example, extends the classical DNS or mDNS naming by recording services as DNS entries which can then be discovered by DNS query messages. Service discovery protocols that are specifically conceived for very constrained devices have also been proposed [22, 8]. Many of these protocols only consider static service attributes that are provided during service registration. In an ideal case, services should be discovered according to dynamic and contextual properties [23].

3.2 Building Automation System Requirements

From the perspective of discovery, building automation systems are significantly different to classical pervasive devices for which location services are most of the time sufficient. Actually, an extension from simple discovery to query is here required. Indeed the context of a device or, in our case, a resource is decisive during discovery. Many building management systems have to look for devices being in a certain state or context for regulation or alarming purposes. For example one BMS could search for temperature sensors on the ground floor having measured in the last five minutes a value higher than 25°C. This request must also consider devices that have not yet reported and are not known by the system. Relying on a plug-and-play approach allows to have highly dynamic systems requiring no previous knowledge of the available resources. We introduce here the concept of static and dynamic resources properties as illustrated in Fig. 5. If one wants to discover all the available devices in a given installation, a simple query specifying no properties would be sufficient. The next step would consist of retrieving the description for resources of interest. Working with constrained devices and low-power networks, the energy efficiency of the new layer is also key factor that has to be optimized in order to not affect life span of battery-operated devices.

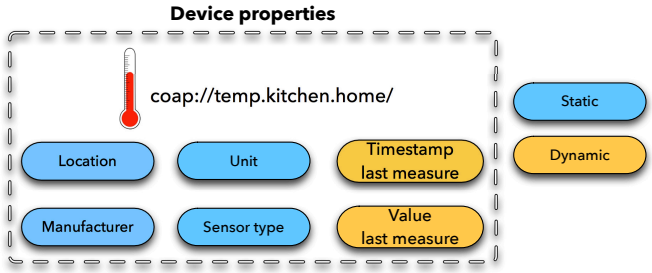

Fig. 5 Static and dynamic properties of a resource

Discovery in building automations systems should fulfil the following minimal requirements:

1. Be optimized for constrained devices
2. Allow a plug-and-play installation of new devices
3. Allow a discovery of the entire network
4. Allow a precise discovery and selection of devices according to some contextual parameters
5. Be scalable and fault tolerant

3.3 Architecture

In order to fulfil the above mentioned requirements, we have to make several choices regarding architecture and technologies. A potential answer can be proposed by dividing the problem into four sub-problems: infrastructure, application layer, data format and query engine.

Infrastructure - A first question is about the use of central or distributed directories of resources descriptions. Using repositories is more suitable for environments with thousands of services, where queries can be processed more efficiently. However a centralized approach has some drawbacks. The first one is about fault tolerance (see requirement 5 above). A central directory failure would result in the inability for clients to discover services. Further to this, the repository has to be frequently updated with context and device availability that would generate a lot of traffic. For these reasons, a de-centralized approach is probably recommendable, where every device will be a discovery endpoint, and thus guarantee a reliable availability. In addition, the traffic of messages is kept very low by using multicast techniques. Only one message is sent by the client, receiving unicast response from matching endpoints.

Application layer - For the application layer, we suggest to rely on CoAP for transporting the requests and responses. CoAP is indeed already present on devices implementing Web-of-Things stacks and more specifically building automation stacks. Additionally, CoAP provides a multicast communication in the *Groupcomm* draft, which can be used to address the entire network with only one packet [28]. With this method, devices offering discovery capability expand their interface with a new service responding to multicast requests. Leveraging on CoAP allows already to satisfy requirements 1 and 3 listed above. The use of multicast requests allows limiting the number of packets transiting over the network. The number of packets sent for discovering a resource with multicast can actually be computed with $N_p = n + 1$, with n representing the number of devices matching the query.

Query engine - The query engine is at the heart of the discovery process. It finds matches according to the properties specified by the client. Many systems for querying structured data exist. As explained in Section 2.3, SPARQL and RDF comes out to be a promising alternative. In this context, when receiving a discovery request, the SPARQL engine retrieves the current up-to-date description of each resource containing the static and dynamic property values. The next step consists of applying the query to the RDF document. In the case of matching results, they will be piggybacked within the response packet. If no resource meets the criteria, no response will be sent to the client. From a conceptual point of view, the coupling of the resource context with RDF and SPARQL makes the discovery process closer to a distributed query engine, which meets requirement number 4.

Data format - Following the same principle as for the building automation stack presented earlier, a reasonable proposition is to rely on JSON as exchange data format to minimise the network traffic. However, translating the SPARQL query expressions to JSON would add complexity and overhead as SPARQL is already free of tags and rather compact. Therefore, the original SPARQL format could be per-

fectly used for expressing the queries. On the other side, query responses are usually provided in XML format for which a translation to JSON could be recommended. Using SPARQL for queries and JSON for responses would allow minimizing the exchanged data and therefore comply with the requirement 1 stated earlier.

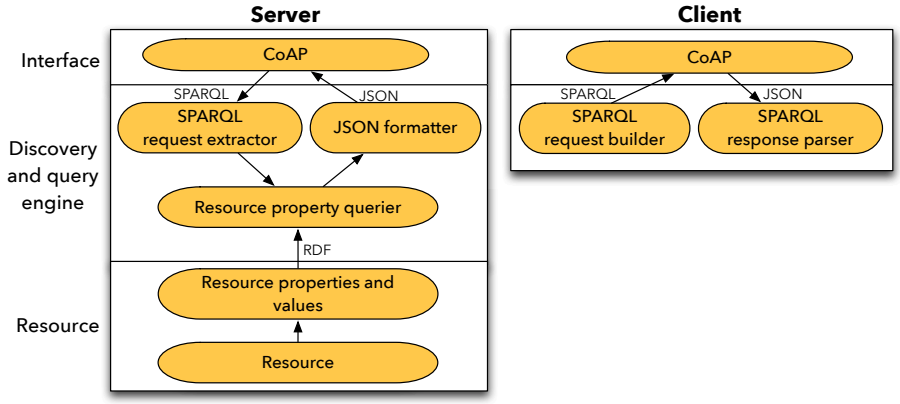

Fig. 6 Client and server discovery architecture

An implementation based on the proposed architecture for discovery is depicted in Fig. 6. The different modules can be grouped in interface, discovery and query engine, and finally the resource.

In more details, the flow of operations is the following. The CoAP server listens on a multicast socket with preconfigured port and IP address. Its role is to dispatch incoming discovery requests to the discovery and query engine. If matches are found, it will respond to the source of the request with the according payload data. Going one level down in the server, the discovery and query engine is responsible for evaluating requests and finding corresponding matches. It starts with a SPARQL request extractor that will take out the query from the CoAP messages and perform some validations regarding the format. Once the request is considered as well-formed, it is passed to the querier. This module will gather the RDF representations of all the resources present and available on the device. Once the collection complete, the SPARQL query is applied to each RDF representation of the collection. Each match is then stored in a collection of results. If the collection of results is empty after querying, the process can stop at this step. Otherwise, the collection is forwarded up to the JSON formatter. This module iterates through the results collection and formats SPARQL responses to JSON. The last step consists of responding to the client over the CoAP interface with the results.

We believe that having a modular approach as described above reduces the complexity of the discovery architecture and allows for future evolutions. Additionally, the different modules can rely on existing libraries offering the desired functionalities therefore reducing the implementation time.

4 Bridging Automation and Adaptation

So called *building automation systems* will perform more and more complex tasks, going way further than simple HVAC automation and light actuation. Future applications will rely on holistic optimisation leveraging on the access to data from the entire set of sensors. Applications will also use external data coming, for example, from weather web services, or even from neighbouring smart buildings, leading to the concept of smart neighbourhood or smart city. Regarding algorithmic, systems will have to incorporate more advanced data analytics technologies. Threshold based rules, often a priori set, will be replaced by complex parametric models implementing self-learning capacities and allowing for dynamic rules as a function of the context of use of the building. In this Section, we take a side step to analyse the system architectures that will, at the same time, leverage on the strengths of Internet and Web technologies, and on new emerging intelligent services, enabling to move from *building automation systems* to *building intelligence systems*.

4.1 Related Work

Classical building automation systems are composed of three tiers, as depicted on the left part of Fig. 7 [5]. At the bottom we find the *field level* consisting of the network topology and the attached sensors and actuators providing for measurements and actuations. The *automation layer* provides control logic for driving actuators, providing some kind of intelligence to the building. The purpose of the automation layer is to optimize the comfort inside the building by using rules of actuation typically based on predefined threshold values. At the top, the *management level* offers applications for configuration and data visualization. With this architecture, regulation algorithms are mostly centralized on a single node and rely on few historical data. In the case of large buildings, multiple computers are used with a repartition of the logic corresponding to parts of the building to avoid dependencies between the sub-systems.

Nowadays, an evolution of the automation can be observed such as the inclusion of genetic algorithms [18], artificial neural networks [21] and empirical models [34], among many others. Such models have the ability to capture through historical sensor data information about the physics of the building and therefore to elaborate automation rules that are more precise. New approaches are also attempting to optimize the energy consumption according to a modelling of the user behaviour (see Section 5).

Despite a lack of literature related to system architecture in buildings, we can reasonably argue that the classical three-tiers architecture currently proposed by industries is not well suited for future developments. Indeed, the arrival of IoT, inherently relying on distributed nodes, as well as the emergence of these new modeling strategies are advocating for new architectures that we present in the next Sections.

4.2 Evolution of Building System Architecture

We illustrate on the right part of Fig. 7 the perceived evolution of building system architectures. We first introduce an *adaptation level* that will dynamically feed the *automation level* with control logic, i.e. rules. In the IoT approach, the *management level* has also to be made available transversally as configuration, discovery and monitoring services must be made accessible to all levels.

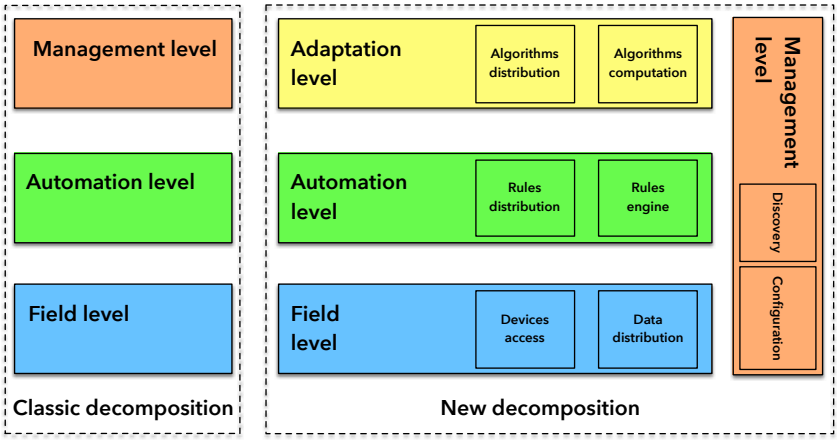

Fig. 7 Level based architecture of building automation systems. The left part shows the classic state-of-the-art decomposition. The right part shows the potential evolution of the architecture needed for future building applications.

4.3 Adaptation Layer

The current trend is to use intelligent data analysis technologies such as machine learning. Such algorithms need historical data from sensors to generate dynamic rules. Following IoT and WoT concepts, the data storage, as well as the automation and adaptation layers should be decentralized across the network and become ubiquitous. Consequently, historical data will be spread over the network and stored close to the devices. Algorithms and rules have also to be considered as Web resources in a similar way as for sensors and actuators. Figure 8 compares the repartition of roles for a classical building automation system (left) to the new WoT-enabled architecture (right). In this context, future works will have to be carried on to find solutions to minimize the transfer of data and the distribution of algorithms.

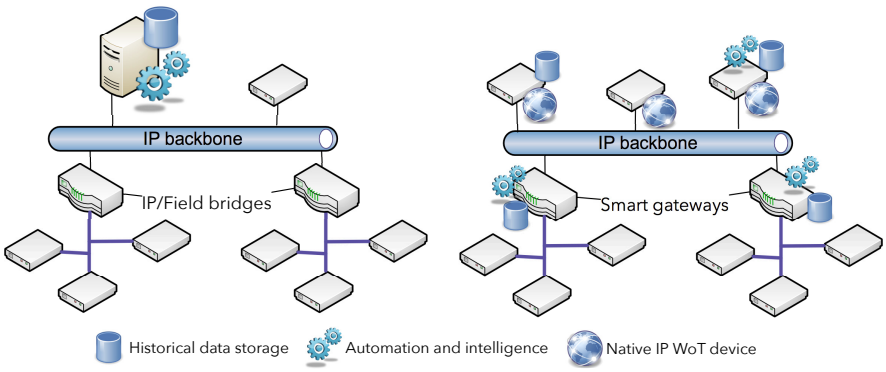

Fig. 8 Role distribution for a classical building automation system (right) and for a Web-of-Things architecture (right).

4.4 Automation Layer

The automation layer sends messages to the actuators to control the building equipments. The control is done according to rules which, in the classical approach, are most of the time static and a priori defined. In the new approach, rules will be generated by the adaptation layer, potentially evolving with time. A WoT compatible approach will be to distribute the rules over the network on devices with computational capabilities. We can already distinguish two types of automation rules that will probably be handled differently: event based and data oriented. In the first category, the rules will only be triggered as answers of building events. The second category of rules will require access to historical data such as, for example control loops. In the similar way as for the adaptation layer, future work will have to define strategies in order to reduce the traffic of data and to optimise the overall efficiency of the distributed system.

Although the adaptation and automation layers are quite similar from a WoT perspective, their purpose is not the same. For instance, buildings may not implement an adaptation layer and take part only of the automation one. This is actually a reasonable argument in favour of a separation of both layers.

5 Intelligent Data Analysis for Smart Environments

In this Section, we focus on providing further discussion on the *Adaptation Level* as described in Fig. 7. One could qualify an environment as being smart whenever it is able to make the right decision at the right moment and with a satisfying rate of success regarding the outcome of the decision. A recent trend to build smart systems is to rely on so-called *Intelligent Data Analysis* approaches. Such approaches are

often based on machine learning techniques able to infer prediction or classification models from large set of observation data. Different machine learning techniques can be employed depending on the application purpose, the computational capability and the desired accuracy rate. The main advantage of Machine Learning is to be found in the ability to discover the complex and sometimes unexpected correlations in heterogeneous input data. In this regards, the arrival of IoT and WoT is of course a key enabler for the use of Intelligent Data Analysis in Smart Buildings. A common application of machine learning in smart environment is the recognition of human activity. This application also encompass presence verification, intrusion detection and, to a larger extent, abnormal behaviour detection.

5.1 Evolution of Control Algorithms

Many algorithm strategies have been applied to smart buildings. A classification from the simplest algorithm with no or few adaptation, to the most complex approaches mostly relying on machine learning can be given as follows.

- **Fixed threshold based algorithms**. Such algorithms implement simple rules typically using fixed thresholds to control the equipments. These rules are often set without considering the real needs and dynamics of the users. HVAC systems are typically controlled depending on target temperature values conditioning the air by injecting heat or cold. The rules are sometimes using a schedule of predefined values to comply with known cyclic energy needs, typically day/night or seasonal cycles.
- **Physics based algorithms**. Such control algorithms are using mathematical models of the physics of the building. For example, the thermal inertia is computed and used to avoid undershoot or overshoot of temperatures. More sophisticated models will take into consideration the building orientation, its geometry, size of windows, among others for computing optimal blind control. Such models, while improving significantly the energy usage, are nevertheless targeting a comfort level for an "average user", disregarding the dynamics of the real use of the building. Furthermore, such systems are typically costly in terms of setup and need careful tunings at the commissioning phase.
- **Self adapting algorithms**. Machine learning technologies are typically used here to compute data-driven models for prediction of variables and classification of events. Such algorithms have the possibility to adapt continuously to building characteristics, building use, and environmental conditions.

We are here specifically interested into the third category of self adapting algorithms that have the most potential regarding smart applications. The arrival of IoT/WoT architectures are also enabling the use of more complex algorithms having to deal with heterogeneous and asynchronous data. In the scientific literature, many models can be found such as predictive models, artificial neural networks, fuzzy logic and many others. Recently, stochastic state based data-driven models

have also been proposed to capture the spatial and temporal statistics of building usages.

Self adapting algorithms present several advantages. First, they are independent of the physic of the building or, more precisely, the building characteristics are learned intrinsically from the observation data. Second, the optimization has the potential to become global instead of local, especially in the case of unified IoT/WoT architectures where all sensors and actuators are exposed from the sub-systems. Third, time-varying usage of buildings will be tracked as self-learning algorithms have the ability to incrementally adapt their parameters with new observation data. Such approaches allow an adaptation to real user, and not just an average user, thus minimizing the risk of rejection. Finally, new building configurations can be automatically learned, hence reducing setup costs. This last point has actually to be weighted by the fact that self adapting algorithms need some time to converge, creating potential loss of comfort during learning.

We have to point out that these advantages are still theoretical for smart buildings. Indeed, at the time of writing this text, we did not yet observe a smart buildings implementation relying fully on IoT/WoT and self-adapting algorithms. The vision is actually to go for a system where, typically, the installation of a new sensor would be automatically known from the rest of the system and incorporated into the algorithms as new piece of information for a better global settings of the equipments.

5.2 Adaptation Level Architecture

In this Section, we propose a global and unified architecture for data modeling in the context of smart building. In many configurations, physics based algorithms or self adapting algorithms can usually be considered as extensions of threshold based systems, where the threshold values are continuously set from the models. Actually, the fixed threshold based algorithms could be seen as rules being processed in the *automation level* as described in Fig. 7. Physic or self adapting algorithms would be dynamically creating the rules in the *adaptation level* and feeding them into the *automation level*. The algorithms of the adaptation level can operate either continuously or at predifined periods of time, for example everyday at midnight. The frequency depends on the needs of the application and on system capabilities in terms of memory and computing power. In this sense, the adaptation level can actually be seen as decoupled from the automation level implementing the on-line working system.

Figure 9 illustrates the generic life-cycle of intelligent data analysis. At the field level, the sensor nodes provide information captured from the environment. The raw observations are communicated to the adaptation level. The adaptation level then performs a feature extraction. The purpose of this operation is to compute useful characteristics from the raw observations. Typically, features consist of normalized or filtered raw values. More complex feature extraction can also be applied such as frequency analysis with Fourier transform or computation of relevant features

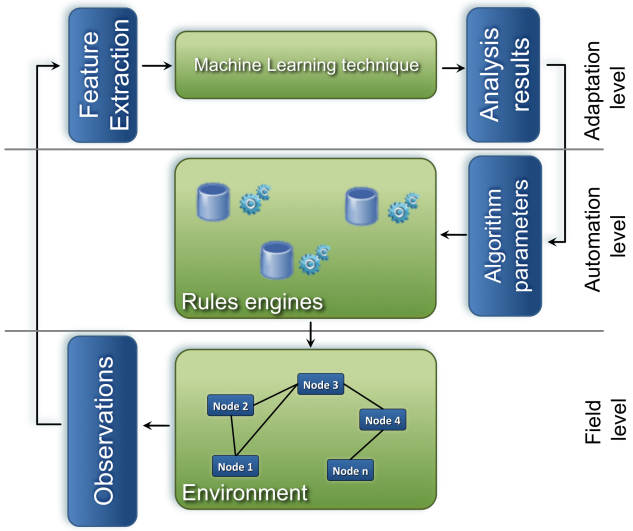

Fig. 9 Life-cycle of data processing with intelligent data analysis.

with Principal Component Analysis. The next step is to model the features for a specific objective such as prediction of future values or classification of events. At this point a specific machine learning technique is applied to compute the models. From a general perspective, the parameters of a mathematical model are computed or updated from exemplary feature data, i.e. data for which the expected output of the model is known. These data are refered to as *training data* or *ground truth*. Once the models are ready, they are used to compute on-line rules that are fed into the automation level. The execution of a rule in the automation level will finally send commands to the actuators of the field level.

5.3 Adaptation Level and State Modelling

In the case of smart buildings, the signals to be modelled are often time series representing the evolution of some observations measured at the level of a sensor, or at the level of a combination of sensors located in a room, or even, at the most general level, the whole set of sensors in a given building. These multi-level and time dimensions are suggesting that the signals can be modelled using state-based models such as Hidden Markov Models. Figure 10 illustrates this approach. Such modelling usually implies that a state is tied to a stability of the statistical properties of the signal. Typically, the task will be to discover the most probable state in which the sytem currently is. The state label will be used to generate rules then fed into the automation level as explained before. As depicted in this Figure, the discovered state at one given layer can be fed to the above layer as an extra input signal. For sake

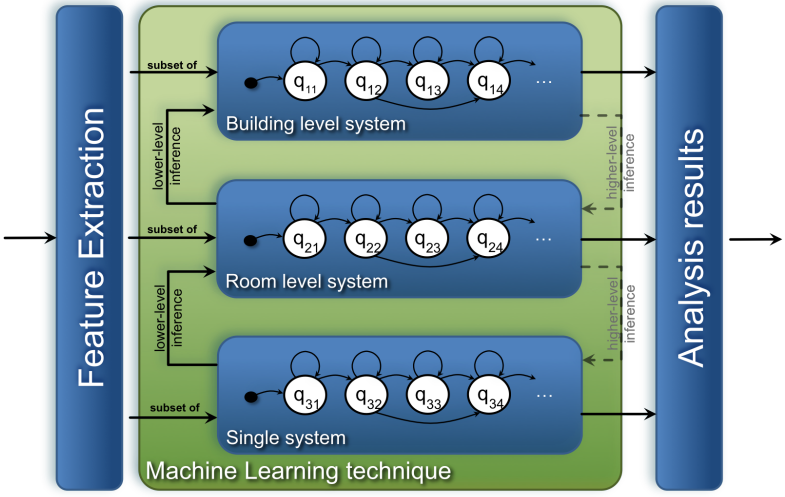

Fig. 10 Example of adaptation layer based on state modelling.

of simplicity, only three layers are shown, but more or less layers could be present depending on the context of the application. The generic principle is to include the information coming from the more granular to the less granular levels, from sensor information, to room information and up to floor and building knowledge.

5.4 Examples of Application

We describe here three applications that were implemented according to the architecture proposed in the previous section. Hidden Markov Models (HMMs) were used as modelling tool.

The first application is linked to single system modelling, i.e. the lowest level as illustrated in Fig. 10. A single system consisting of a smart plug is used to measure the electricity consumption of appliances at the frequency of 10^{-1} Hz. The target application is the automatic identification of electric appliances so that building residents can have detailed information on their electricity bill. The HMMs are trained using a publicly available database of electric signatures called *ACS-F1* for Appliance Consumption Signature Fribourg 1 [11]. The database consists of two acquisition sessions of one hour for 100 appliances spread uniformly into 10 categories. Two evaluation protocols are proposed with the database distribution. The first *intersession* protocol consists in attempting to recognize signatures from appliances already seen in the training phase [29]. The second protocol, called *unseen instance*, aims at recognizing signatures coming from appliances never seen in the

training phase. In this more difficult case, the system has to be able to generalize to new brands or models [30].

The second application aims at automatically recognizing activities at the level of a full floor in a residence. A set of presence and door sensors is spread in the residence and provides asynchronuous signals in case of movement and passage. The feature extraction consist here of a simple shifted window processing in order to reconstruct time synchronized observation vectors. HMMs are built with a topology where the states are related to the zones of the floor where the activities are expected. Seven HMMs are trained using the data, each one of them corresponding to a given activity to be recognized. The Viterbi algorithm is used for the training and testing phases. We evaluated the proposed models using the WSU CASAS dataset [10], achieving an accuracy of 98,9% using the leave-one-out technique on seven activities recorded for 8 months [31].

The third application aims at discovering a better definition of seasonal HVAC controls at the level of a building, i.e. the upper level as illustrated in Fig. 10. More specifically, the model is built to predict the change of seasons using the information coming from window openings, window blinds, external and internal temperatures and solar irradiation. The model is based on a HMM defined with three states, representing the heating, cooling and intermediate seasons. In this experiment, the training data is provided through past data of the activation of the HVAC system of the LESO-PB building in EPFL, Lausanne. For example, when the system is in heating mode, the associated state is known and the parameters of the models can be trained with the input data. In the testing phase, the Viterbi algorithm is used to determined the state label at a given time and compared to the ground truth. The results are showing a correct season identification with an accuracy up to 91% for the heating and cooling seasons, while the most difficult intermediate season shows a rate of 69% correct detection [31]. While this experiment is still a bit theoretical, it shows that a machine learning based modelling can capture the actual controls of a well-tuned HVAC system. It can be reasonably expected that the learned model can be re-used in a similar building configuration.

The three preceding applications show the feasibility of the proposed state-based modelling as algorithmic approach to implement the adaptation level of Fig. 10. A missing step which is not explored in this work, is the inference of relevant rules to be injected in the automation level.

6 Conclusion

Buildings automation systems have in the last years not followed the trend of modernization and are always relying on isolated networks. The emerging technologies of sensors and especially the Internet-of-Things can provide many advantages to those buildings. In this chapter we investigated the main issues that have to be solved for augmenting traditional building automation systems with IoT capabilities. The main challenge lies in the natural heterogeneity of building networks working on

different protocols. A successful homogenization of BAS can only be achieved by implementing a standard and open protocol stack. Multi-protocol gateways hiding the complexity of BAS by mapping devices capabilities to Web services highly simply the integration of existing networks. The emergence of IPv6 solves the address limitation of IPv4 and allows nowadays any device to be directly connected to IP networks. Trendy Web technologies like CoAP following the REST architectural style already contribute to the standardization process with their lightweight Web services. A key role in homogenisation is played by the acceptance of the application layer. Some propositions have already been done in this area converging to the use of data models for sensor properties. Until today none of them managed imposing itself as the de-facto standard. We can explain this by the heavy-coupling they impose using XML schemas that have to be known by every device. In our point of view only a loosely-coupled solution offering enough flexibility has a chance of being accepted by the scientific community and especially the industry. Here the contributions of the World Wide Web Consortium with RDF and ontologies have all their importance. Working in pervasive and ubiquitous environments requires approaching discovery of devices with another point of view. The need of a solution allowing to crawl devices with no prior knowledge appears inevitable. Distributing SPARQL queries over the network via multicast appears to be a clear response to this constraint. Building automation systems can benefit from advances made in the field of machine learning techniques. A new dimension of energy saving opens to IoT-enabled building management systems. Anticipating the users behaviour and actuating HVAC systems according to the buildings use has a real potential reducing the overall energy consumption. This target is only achievable by introducing a new level to the classical three-tiers decomposition of BAS.

IoT and WoT are expected to be key enablers for advanced building controls based on intelligent data analysis. In this direction, different machine learning techniques can potentially be used for managing a smart environment. The life-cycle of machine learning applications starts with an acquisition of raw data from the sensors distributed in the building, continues with an extraction of relevant features, follows with the identification or prediction of some events that are then, in turns used to modify the rules of the automation level of the building management system. The modification of the rules can take place continuously or off-line, at regular interval. A potential powerful modelling scheme can be found in state-based modelling such as Hidden Markov Models where the statistic of the features are captured according to states representing the piece-wise evolution of the building context. In the description of the adaptation level provided in this Chapter, we have left apart important practical considerations, for example the ones regarding the computing power, data storage and the energy consumption of the adaptation layer itself. The limitation of resources and lack of energetic availability in IoT installations limits de facto the complexity of the machine learning techniques that may be used.

Although building automation being quite a recent research field, certain directions still emerge. The near future will give us more insights about the practical feasibility and the acceptance of Web technologies in buildings. Moreover we can already distinguish two imminent key points that will limit the spreading of IoT

building automation systems. The first concern is about the energetic impact of such installations. The overall IoT approach should not consume more energy than traditional BAS and therefore counter the efforts done optimizing the building. Secondly, the very restricting open issue that has not been tackled is about security. One can easily imagine the consequences resulting in a misuse of the automation system. Cybercrime has become a major problem of today's information systems. Solutions limiting mashups between devices explicitly authorized and ensuring the privacy of historical data are essential before deploying IoT systems in buildings.

References

1. Dolce ultralite ontology. http://ontologydesignpatterns.org/wiki/Ontology%3aDOLCE%2BDnS_Ultralite.
2. Enocean. http://www.enocean.com/en/home/.
3. Knx association. http://www.knx.org/.
4. The universal device gateway. http://www.devicegateway.com/.
5. Iso 16484-2:2004 building automation and control systems (bacs). Technical report, International Organization for Standardization, 2004.
6. G. Bovet and J. Hennebert. Offering web-of-things connectivity to building networks. In *Proc. of the 2013 ACM international joint conference on Pervasive and ubiquitous computing*, 2013.
7. G. Bovet and J. Hennebert. Web-of-things gateway for knx networks. In *Proc. of the IEEE European Conference on Smart Objects, Systems and Technologies (Smart SysTech 2013)*, 2013.
8. T. Butt, I. Phillips, L. Guan, and G. Oikonomou. Trendy : An adaptive and context-aware service discovery protocol for 6lowpans. In *Proc. of the 3rd International Workshop on the Web of Things*, 2012.
9. S. Cheshire and M. Krochmal. Dns-based service discovery. Technical report, 2013.
10. D.J. Cook. Learning setting-generalized activity models for smart spaces. *IEEE Intelligent Systems*, 27:32–38, Jan 2012.
11. C. Gisler, A. Ridi, D. Zufferey, O. A. Khaled, and J. Hennebert. Appliance consumption signature database and recognition test protocols. In *Proceedings of the 8th Internation Workshop on Systems, Signal Processing and their Applications (Wosspa '13)*, pages 336–341, 2013.
12. Y. Goland, T. Cai, P. Leach, Y. Gu, and S. Albright. Simple service discovery protocol/1.0 operating without an arbiter. Technical report, IETF, 2000.
13. W. Granzer, W. Kastner, and C. Reinisch. Gateway-free integration of bacnet and knx using multi-protocol devices. In *Proc. of the 6th IEEE International Conference on Industrial Informatics (INDIN '08)*, 2008.
14. W3C Semantic Sensor Network Incubator Group. Review of sensor and observation ontologies. http://www.w3.org/2005/Incubator/ssn/wiki/Incubator_Report#Review_of_Sensor_and_Observation_ontologies.
15. W3C Semantic Sensor Network Incubator Group. Semantic sensor network ontology. http://www.w3.org/2005/Incubator/ssn/ssnx/ssn.
16. D. Guinard. *A Web of Things Application Architecture - Integrating the Real World into the Web*. PhD thesis, ETHZ, 2011.
17. E. Guttman, C. Perkins, J. Veizades, and M. Day. Service location protocol, version 2. Technical report, IETF, 1999.
18. W. Huang and H.N. Lam. Using genetic algorithms to optimize controller using genetic algorithms to optimize controller parameters for hvac systems. In *Energy and Buildings*, pages 277–282, 1997.

19. M. Jung, C. Reinisch, and W. Kastner. Integrating building automation systems and ipv6 in the internet of things. In *Proc. of the 6th International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing*, 2012.
20. M. Jung, J. Weidinger, C. Reinisch, W. Kastner, C. Crettaz, A. Olivieri, and Y. Bocchi. A transparent ipv6 multi-protocol gateway to integrate building automation systems in the internet of things. In *Proc. of the IEEE International Conference on Green Computing and Communications*, 2012.
21. A. Kanarachos and K. Geramanis. Multivariable control of single zone hydronic heating systems with neural networks. In *Energy Conversion and Management*, pages 1317–1336, 1998.
22. A. Kovacevic, J. Ansari, and P. Mahonen. Nanosd: A flexible service discovery protocol for dynamic and heterogeneous wireless sensor networks. In *Proc. of the 6th International Conference on Mobile Ad-hoc and Sensor Networks*, pages 14–19, 2010.
23. S. Mayer and D. Guinard. An extensible discovery service for smart things. In *Proc. of the 2nd International Workshop on the Web of Things (WoT 2011)*, 2011.
24. M. Neugschwandtner, G. Neugschwandtner, and W. Kastner. Web services in building automation: Mapping knx to obix. In *Proc. of the 5th IEEE International Conference on Industrial Informatics (INDIN '07)*, 2007.
25. N. Nurseitov, M. Paulson, R. Reynolds, and C. Izurieta. Comparison of json and xml data interchange formats: A case study. In *CAINE*, 2009.
26. oBIX 1.1 draft committe specification. The universal device gateway. `https://www.oasis-open.org/committees/download.php/38212/oBIX-1-1-spec-wd06.pdf`.
27. L. Perez-Lombard, J. Ortiz, and C. Pout. A review on buildings energy consumption information. *Energy and Buildings*, 40:394–398, 2008.
28. A. Rahman and E. Dijk. Group communication for coap. Technical report, IETF, 2013.
29. A. Ridi, C. Gisler, and J. Hennebert. Automatic identification of electrical appliances using smart plugs. In *Proceedings of the 8th Internation Workshop on Systems, Signal Processing and their Applications (Wosspa '13)*, pages 301–305, 2013.
30. A. Ridi, C. Gisler, and J. Hennebert. Unseen appliances identification. In *Proceedings of the 18th Iberoamerican Congress on Pattern Recognition (Ciarp '13), to appear*, 2013.
31. A. Ridi, N. Zakaridis, G. Bovet, N. Morel, and J. Hennebert. Towards reliable stochastic data-driven models applied to the energy saving in buildings. In *Proceedings of the International Conference on Cleantech for Smart Cities and Buildings (Cisbat '13)*, 2013.
32. W3C. Resource description framework. `http://www.w3.org/RDF/`.
33. W3C. Sparql query language for rdf. `http://www.w3.org/TR/rdf-sparql-query/`.
34. Y. Yao, Z. Lian, Z. Hou, and X. Zhou. Optimal operation of a large cooling system based on an empirical model. In *Applied Thermal Energy*, pages 2303–2321, 2004.