A smart university platform for building energy monitoring and savings

Thanos G. Stavropoulos[a,b,*], George Koutitas[b], Dimitris Vrakas[a], Efstratios Kontopoulos[c] and Ioannis Vlahavas[a,b]

[a]*Department of Informatics, Aristotle University of Thessaloniki, University Campus, 54124 Thessaloniki, Greece*
[b]*School of Science and Technology, International Hellenic University, 14th km Thessaloniki – Moudania, 57001 Thessaloniki, Greece*
[c]*Information Technologies Institute, Centre of Research & Technology - Hellas, 6th km Charilaou-Thermi Rd, 57001 Thessaloniki, Greece*

Abstract. This paper presents a novel, integrated platform for energy monitoring, management and savings in the context of a Smart University Building. Namely, the proposed Smart International Hellenic University (IHU) platform integrates an intelligent, rule-based agent that enforces savings, while a variety of applications offers user interaction with the system and the means for monitoring and management. The application layer is built over a common Web Service middleware, incorporating semantic interoperability. Monitoring applications visualize raw and aggregated sensor readings, such as building energy disaggregation, environmental measurements and data center efficiency. Extensive monitoring capabilities allow users to take immediate action and devise policies towards energy-savings. Such policies are, then, autonomously enforced by the intelligent, hybrid agent, which is capable of both deliberative (long-term) and reactive (immediate) actions. The agent is also integrated with the OpenADR standard for receiving provider instructions in future Smart Grids. A pilot deployment of the agent with expert-formulated policies at the IHU premises, has managed to reduce the total daily consumption of a typical university office by approximately 16%.

Keywords: Smart grids, sensor networks, web services, semantic web, ambient intelligence

1. Introduction

The Smart Grid is the next generation of the power grid network, providing dynamic energy management and efficient resource allocation. The field originates from the convergence of the Energy and the Information and Communication Technologies (ICT) sectors. In essence, an ICT layer is built upon the existing energy layer of current power grid networks. Since the major consumer of electricity in the power grid network is currently the building sector, it is important to transform and integrate buildings into smart entities, capable of supporting such services.

The most prominent scientific fields concerning the role of ICT in the smart grid era include Machine-to-Machine communications (M2M), the Internet of Things (IoT), optimization and control theory. These fields, along with their underlying, specialized areas of study, form the foundation for smart grid services. Characteristic examples of such areas are the Advanced Metering Infrastructure (AMI) [1], Meter Data Management systems (MDM), Demand Response (DR) [2], Web 2.0 technologies, Artificial Intelligence (AI), etc. In [3], a methodology for integrating heterogeneous networks of smart meters and sensors is briefly analyzed. The paper gives a general description of the concept, presenting the key areas of investigation.

Meanwhile, Ambient Intelligence (AmI) is one of the eminent, concurrent computing paradigms, amongst Ubiquitous Computing (UbiComp) and Pervasive Computing (PerComp). According to these standards, ambient, non-intrusive devices are scattered

*Corresponding author. E-mail: athstavr@csd.auth.gr.

around the environment in order to sense and act in relation to context. Therefore, such systems are also called Context-Aware and typically rely on Semantic Web constructs i.e. ontologies, to model the context perceived by the system in a machine-interpretable way. So far, AmI systems have largely been based on Web Service technologies [4], such as WSDL[1], to model and exploit universal, platform-agnostic APIs for the various devices that need to be used in such settings. AmI state-of-the-art includes applications of limited scale, targeting domains such as smart offices [5], home automation and agriculture [6]. Much work targets ambient healthcare or Ambient Assisted Living – AAL [7] and the particular challenges posed [8].

The scope of this paper is to present the architecture and experimental results of the Smart IHU platform, which combines elements of all the aforementioned disciplines. Its novelty lies in the fact that it targets the smart building/grid sector with an all-in-one solution, by integrating heterogeneous device networks, multi-purpose applications and a rule-based agent. Specifically, it introduces the integration of existing middleware [9], ontologies and semantics [10] into an extended deployment, together with a novel hybrid intelligent agent and its real-world application with savings.

Smart IHU's AmI environment integrates various components in a three-layer approach, i.e. hardware, middleware and application. The system's hardware deployment consists of various heterogeneous and platform-dependent wireless networks of sensors, meters and actuators. The middleware layer exposes universal WSDL descriptions, which have been further enriched with annotations in SAWSDL. Incorporating semantics into the middleware enables knowledge retrieval and reasoning. These semantics are exploited by various applications on the upper layer of the platform. Desktop, mobile and web applications fulfill the needs of remote, historical or real-time monitoring and management for system administrators, students, staff and guests.

While the aforementioned components have previously been presented independently, their integration as a single platform enables a semantically-enhanced, rule-based agent to autonomously reason and act in order to manage the building. The agent's behavior is heavily based on defeasible logic, a non-monotonic logics formalism that provides the means for defining compact and intuitive rule sets [11]. An additional advantage of defeasible logics is their sophisticated conflict resolution mechanism implemented through a binary rule superiority relationship. While previous efforts have employed different agents for short-term actions and long-term policies, the current work integrates both behaviors into a single agent. By doing so, the need for conflict-resolution between agents is eliminated, at least for such scenarios.

Furthermore, in-depth monitoring provided by the underlying infrastructure and an extensive deployment presented in this work, has enabled the formulation and incorporation of energy-saving policies, for the specific building, ultimately achieving savings. The proposed architecture also integrates the Smart IHU agent with demand-response capabilities, in compliance to the OpenADR standard[2] [12].

The paper is structured as follows: Section II of the paper presents a general description of the system, while Section III gives a detailed explanation of the sensor and actuator infrastructure, which plays the role of the system's AMI. Section IV presents the system's middleware layer for heterogeneous sensor integration. Section V presents the system's application layer that hosts administrative and monitoring applications as well as an intelligent, autonomous agent. Section VI presents experimental energy-saving results achieved by the agent, while the final sections present a state-of-the-art comparison to the proposed work, conclusions and future work.

2. General Architecture

Towards efficient system integration together with unanimous and remote access to functions, a three-layer approach was followed, consisting, namely, of the sensor, middleware and application layers. The middle layer provides the necessary abstractions from device- and platform-specific languages according to the Service-Oriented Architecture's principles [13]. The sensor layer corresponds to data and command flow of the smart building. Data is related to the measured environmental and energy parameters (i.e. via sensors), while the command flow is related to binary switch on/off commands transmitted to the system's actuators of the sensor layer. The latter realize energy

1 WSDL W3C Recommendation: http://www.w3.org/TR/wsdl

2 OpenADR Alliance: http://www.openadr.org/

management under switch on/off policies on the appliances. The middleware layer corresponds to the integration layer of the heterogeneous network of the sensor layer, whereas the application layer provides the user interface along with services/applications for smart grid functionalities. In the uplink, the sensor layer provides data flow to the application layer, while in the downlink the sensor layer receives the command flow for energy management and automation. The overall system architecture is given in Figure 1. The figure presents the coverage range and connectivity (mesh and star) of the sensors with the gateways based on the theoretical investigation that implemented 2D ray tracing algorithm [14].

There are three different networks of smart meters/sensors/actuators in the building, each one described in detail in the next section. In total, there are five gateways in two different locations in the building used for coverage and operating as data aggregators. The gateways communicate with the central agent (PC unit), which provides an interface to the user, global exposure to internet but it also processes data for supporting automation and management.

The objective of the deployment is to provide functionalities related to AMI, MDM, demand response, mobile applications and analytics. The system is designed in a generic approach that can be easily extended to include additional services and functions. The services and functionalities are described in detail in Sections III-V.

3. Sensors and Actuators

The sensor or physical layer of the proposed architecture hosts a wide variety of wireless sensor and actuator networks. All hardware was selected under the scope of optimizing the tradeoff between necessary requirements, availability and affordability. First of all, the devices would have to operate over a large range and offer a variety of functions, in order to cover the scale and the diversity of parameters in the building. On the other hand, they should remain affordable and widely available in the market, especially in such quantities to cover the whole building. As the market is yet far from convergence in a common communication protocol and data format, the devices were not expected to interoperate out-of-the-box. On the contrary,

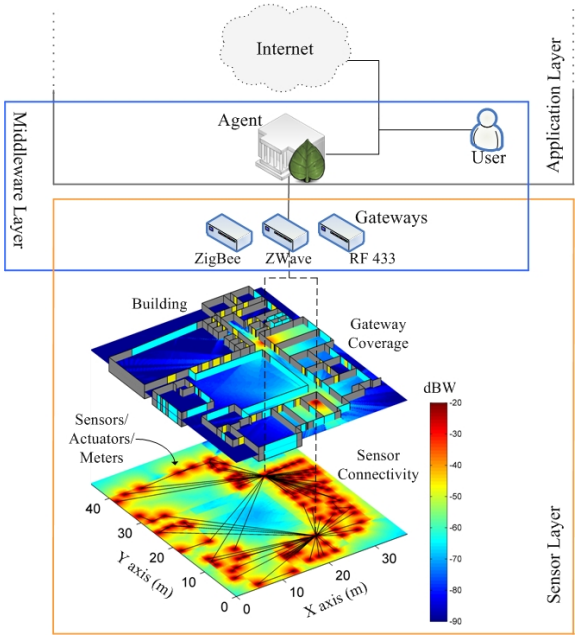

Fig 1. Overall three-layer architecture and WSN coverage in Smart IHU

the issue of interoperability is rather resolved on the middleware layer of the architecture.

The selection of deployed devices incidentally operates over wireless communications and comes from different manufacturers and suppliers. Most devices are also widely available in the market and affordable to retail customers, which adds to the system's realistic deployment and possible adaption to an industrial solution.

For the purpose of measuring and controlling energy consumption on appliance-level, the devices manufactured by Plugwise[3], which are already very popular amongst both retail and research [6, 15], were chosen. The fundamental device of the Plugwise product series is a sensor/actuator device that will be referred to as Smart Plug. Smart Plugs intersect between a wall socket and any electrical appliance, allowing users to measure and control its power supply. In detail, their sensor functions can measure the attached appliance's status (on or off) and its power supply (in W), while the actuator function can switch it on or off. A different variant of this product handles non-pluggable appliances, by intersecting power cables between source and appliance. The two plug types provide identical capabilities, so, to preserve generality, both will be referred to as Smart Plugs from now on.

3 Plugwise Online, http://www.plugwise.com

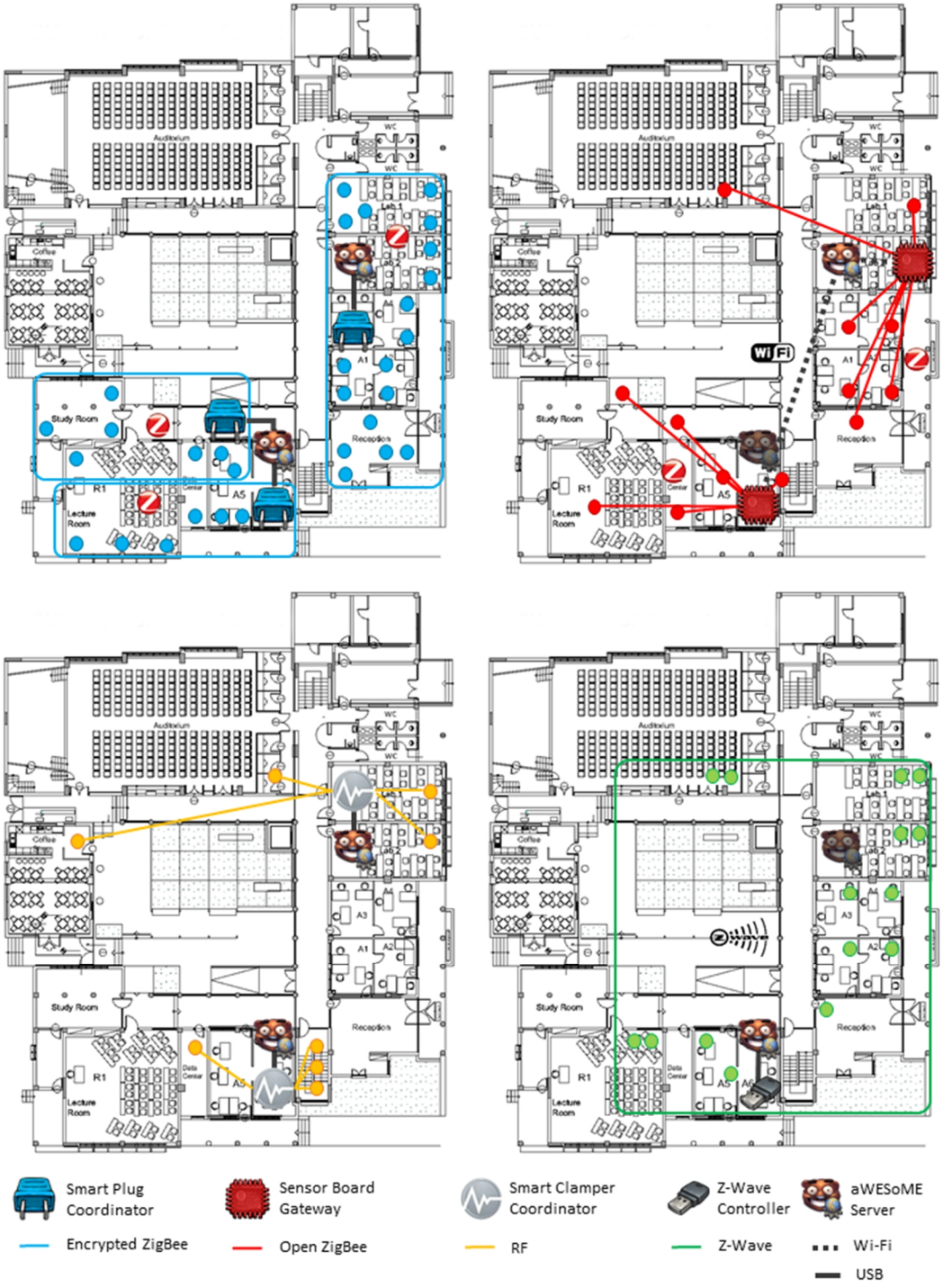

Fig 2. The platform's deployment at the ground floor of IHU's Building A.

In terms of communication, the Smart Plug platform follows the ZigBee wireless communication protocol, one of the optimally suitable protocols for smart home applications. The Plugs form encrypted ZigBee networks of up to thirty nodes in mesh topology. Each network is coordinated by an augmented Smart Plug, which collects and propagates all network communi-

Table 1. Hardware technical, communication and networking aspects

Device	Measured Parameter	Capabilities	Sampling Freq.	Protocol	Max nodes per net.	Range	Topology	Nodes/Networks Deployed
Smart Plug	Energy	Sensor/ Actuator	1s	Encrypted ZigBee	25	10m	Mesh	50/3
Sensor Board	Environment	Multi-Sensor	Any	Open ZigBee	10	10m	Mesh	20/2
Z-Wave	Environment/ Motion	Sensor	Any	Z-Wave	232	10m	Mesh	20/1
Smart Clamper	Energy	Sensor	8s	RF	10	10m	Star	8/1

cations to and from a USB stick PC interface. Consequently, PC users can monitor Plug data and invoke actuator operations. Smart Plugs are passive devices, which means that they have to be polled to perform an action or return a measured value in a bidirectional manner. Each network's range depends on how evenly the mesh network is distributed. Users are prompted to ensure a circular distribution of plugs around the coordinator to avoid long lines of hops, but rather provide short, alternative paths for all nodes. To adequately cover the building's appliances, approximately forty-five Smart Plugs, grouped in three Plug networks have been deployed. One network covers the eastern part of the building, whereas the other two overlapping networks cover the southern part.

While Smart Plugs are responsible for small-scale, per appliance measurements, large-scale power usage has to be monitored by a different set of devices. Especially in the case of a large building, it is rather impractical and inaccurate to monitor its total consumption just by using Smart Plugs. On the contrary, it is much easier and precise to monitor total consumption directly at its source i.e. the main power supply. Smart Clampers are affordable sensors that clip around main power supply cables, without intersecting them, and inductively measure the current. The selected Smart Clamper bundle, offered by Current Cost[4], additionally offers open XML-data format and support for multiple transmitters. Each Current Cost digital 433MHz SRD band transmitter can be attached to up to three Smart Clampers for the measurement of three-phase current. Data of up to ten transmitters is collected and displayed on a monitor/receiver that also provides a PC USB-interface. In our deployment, two three-phase main power supplies are measured by corresponding clampers. The sum of these two measurements returns, thus, the total consumption of the building. For disaggregation purposes, additional bundles were added to effectively disaggregate the building's energy. Namely, clampers and transmitters where placed to measure the Data Center's i.e. server room, the central cooling unit, the coffee shop, the auditorium and the PCs at the university's two computer labs.

Apart from energy, the proposed smart building applications need to monitor and correlate various environmental factors to energy, by placing a variety of environmental sensors in the building. The first bundle

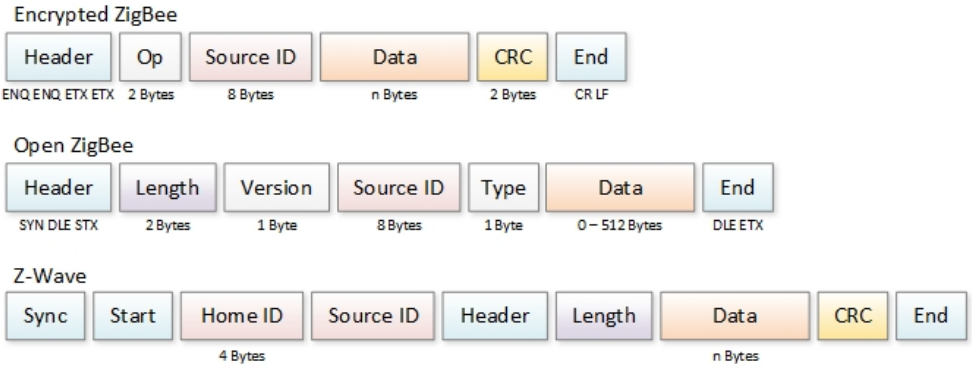

Fig 3. Data packet decomposition of three different communication protocols in the infrastructure.

4 Current Cost online, http://www.currentcost.com/

of sensors is manufactured by Prisma Electronics[5] and consists of two types of nodes. Quaxes are Sensor Boards that embed a microcontroller, a ZigBee module and various arrays of sensors. The microcontroller can be manually programmed to periodically parse data from sensor arrays and transmit them over ZigBee. Optionally, they can also act as routers for other Sensor Boards, forming a ZigBee mesh network. The second type of nodes are ZigBee gateways that collect Sensor Board data and transmit them over Ethernet or Wi-Fi to the LAN's PC clients. The devices are push-based, which means that they actively transmit data one-directionally.

For the purposes of our deployment, two networks of ten Sensor Boards each have been distributed across the building. The sensors attached to the boards measure temperature, luminance and humidity. Again, one network was assigned with the southern part and one with the eastern part of the building. The Sensor Boards were set in non-routing mode, so each network follows a star topology instead of mesh. This allows the modules to enter hibernation between transmissions, prolonging battery life. For the same purpose, their transmission interval has been set to ten minutes. Two Quax gateways collect data from each network and make it accessible from anywhere in the LAN.

Finally, to complement the variety of measured environmental data, various types of devices compliant to the Z-Wave protocol were integrated. The Z-Wave alliance constitutes one of the commercially established efforts for unifying data formats and communications between smart home automation devices at hardware/transport level. So far, the alliance has managed to provide a wide variety of sensors, actuators, network controllers/coordinators, remote controllers etc., achieving interoperability between numerous manufacturers. Transmissions are very similar to ZigBee in nature, as they form wireless mesh networks. In our deployment four CO_2 air level sensors, fourteen motion sensors and three smoke detectors were used, each coming from a different manufacturer. All data is gathered by a USB-Stick controller and PC-interface. A single mesh network of these devices was installed to cover the whole building. Table 1 compares different aspects of all device families in the proposed deployment, while Figure 2 shows the actual deployment and distribution of sensor and actuator networks on the building's ground floor.

5 Prisma Electronics online, http://www.prismaelectronics.eu

4. Web Service Middleware

The intermediate level of the proposed architecture hosts a middleware specifically tailored for Ambient Intelligence applications, based on the Service Oriented Architecture. The nature of these applications operating on top of a dynamic, real-time environment and heterogeneous devices poses several requirements. Service-orientation has been proved to be extremely suitable for such environments and has been used widely in literature [16], resolving application development issues. Namely, one benefit of Service-Orientation is being able to program on a high-level of abstraction, agnostically of specific platform and communication-protocol device programming. This also adds to extensibility, as new devices can be integrated via the authoring of new middleware-plugins which simply translate to new services. All services comply with a web-wide universal API, the W3C WSDL language, which syntactically defines service operations, resolving heterogeneity. The service descriptions are further described semantically using Semantic Annotations for WSDL (SAWSDL) rendering them machine-interpretable. To resolve the issues related to dynamicity in the environment, the Service-Oriented Architecture supports the notion of service provisioning. A Service Broker maintains a list of available services at any given time, which means that mobile providers that interleave the environment do not have to be handled explicitly during application programming.

After studying current state-of-the-art guidelines

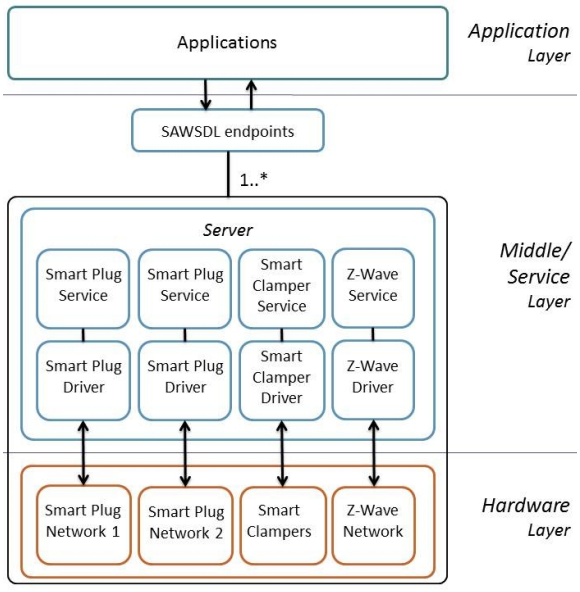

Fig 4. Smart IHU three-layer architecture, focusing on hardware and middleware packages

and existing implementations, aWESoME (a WEb Service MiddlewarE) was developed [9], tailored to the system's needs. The middleware is decomposed internally into three distinct layers: hardware integration (device driver modules), service integration (service endpoints) and semantic integration (SAWSDL endpoints), according to Figure 4. Additionally, it reuses and extends existing implementations as much as possible. Our motivation behind building a middleware from scratch has been based on defining semantics more efficiently and integrating hardware modules that existing middleware does not. However, some existing open-source libraries were indeed used internally in the hardware integration layer, to interface with part of the hardware.

4.1. Hardware Integration Layer

This layer consists of separate modules, each of which interfaces with a target device platform. The so-called driver modules can be considered as plug-ins developed for each device family to be integrated into the Smart IHU system. Figure 3 shows the main communication protocols handled by these modules.

Smart Plug Driver: The Smart Plug Driver, implemented in Java, is invoked every time a "get" or "set" operation needs to be performed over the Smart Plug network. Smart Plug operation is bidirectional, since the devices need to be polled to read a sensor value or to perform an actuator action. Thus, the Smart Plug Driver is rather a library than a module, invoked each and every time it is needed by the corresponding Smart Plug Service operations. To implement the library, the Smart Plug encrypted ZigBee protocol commands had to be reverse-engineered to implement the essential

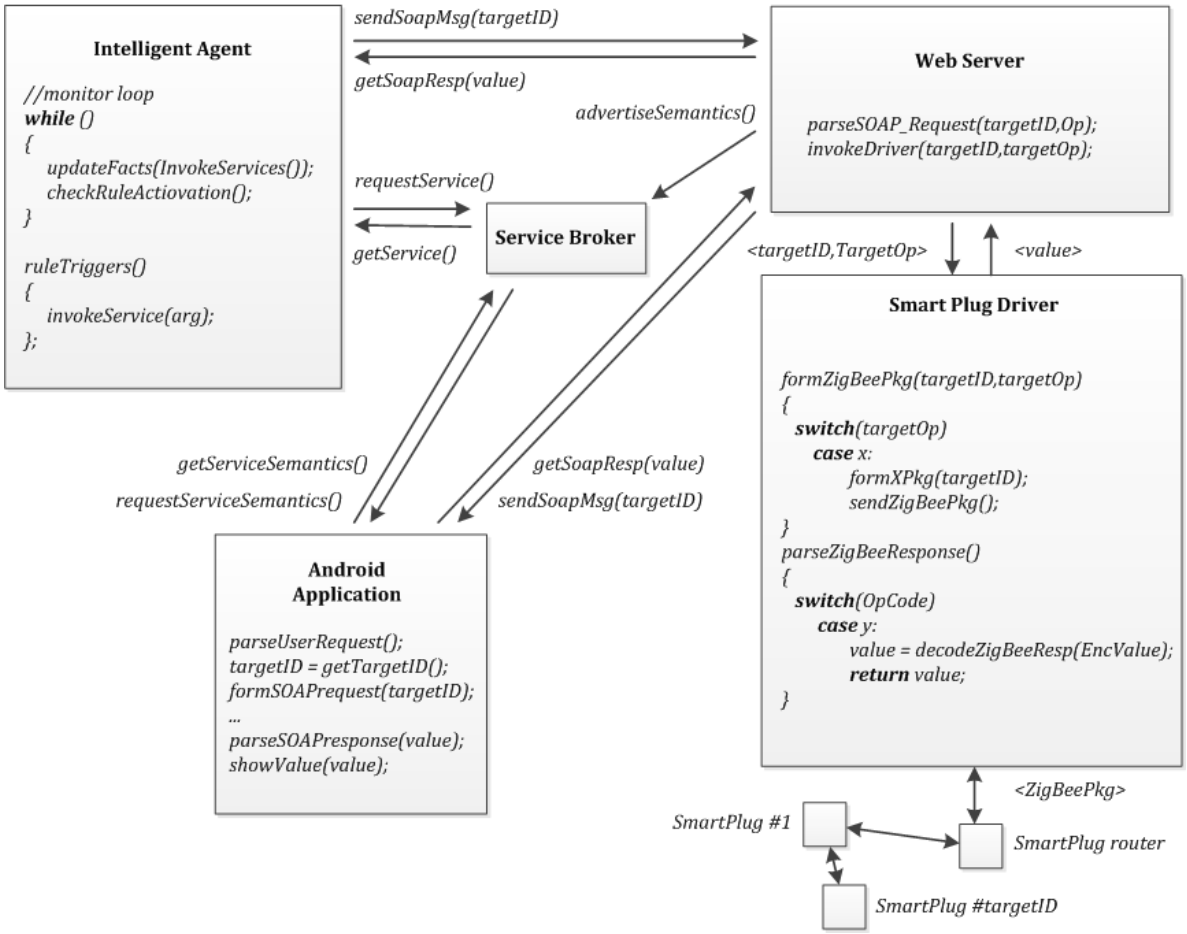

Fig 5. Smart IHU application scenario showing underlying folding and unfolding of messages propagated through all architectural layers

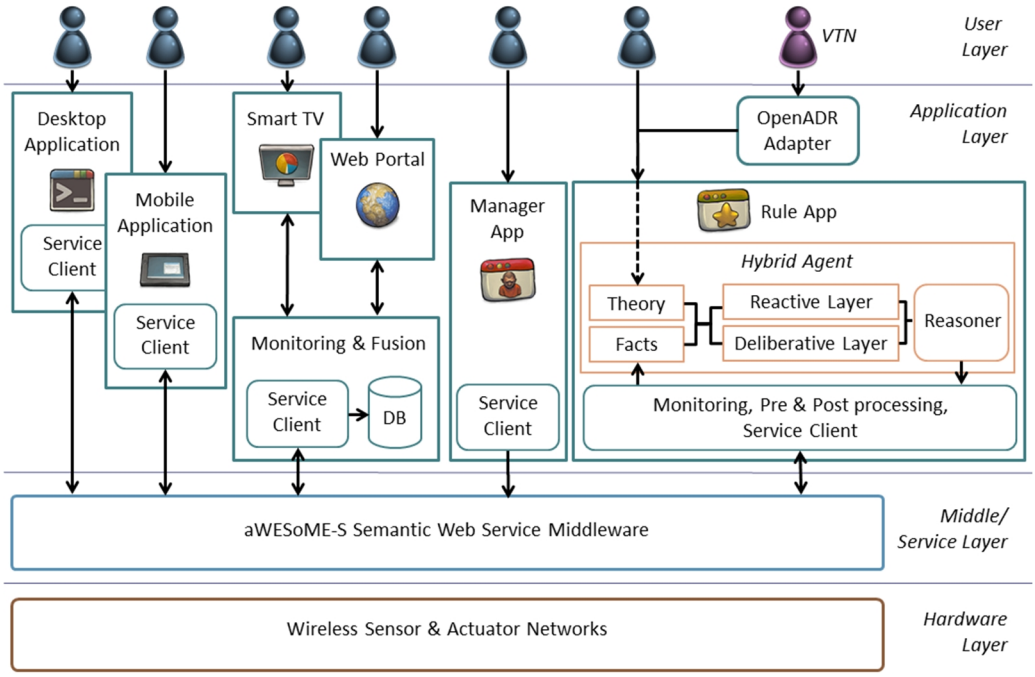

Fig 6. Smart IHU architecture of multiple applications, ranging from desktop, mobile, web, intelligent agents to expert systems

polling functions and decrypt response packages, according to Figure 3.

Sensor Board Driver: Unlike Smart Plugs, Sensor Board communication is one-directional. The Sensor Board Driver module is a C# daemon that constantly receives data, parses them according to the open package format provided by the manufacturer (Figure 3) and stores them for the Sensor Board Service to retrieve. The module also offers a GUI to allow administrators monitor the multiple gateways and nodes of the sensor board network.

Smart Clamper Driver: Quite similarly, the Smart Clamper Driver is a Java module that passively receives all Smart Clamper data, parses and stores them to be retrieved by the Smart Clamper Service. It also presents data on a GUI for monitoring and administration purposes.

Z-Wave Driver: Being the largest and most diverse family of devices, the Z-Wave Driver makes use of the open source Open Z-Wave library to handle potentially any Z-Wave device (Figure 3). To, again, provide administrators with a GUI, another open source solution was employed. zVirtualScenes is a GUI application project which already incorporates the Open Z-Wave library and was adapted to our environment. The Z-Wave driver provides a GUI for monitoring and managing a Z-Wave device network. Unlike other devices, Z-Wave nodes receive configuration parameters, such as sleep interval over the network, so this function is also provided on the administration GUI. It also presents all values received from connected nodes, and stores them for the Z-Wave Service to retrieve on-demand. Virtually any type of Z-Wave device is supported by the Open Z-Wave library and in turn by the Z-Wave Driver.

4.2. Web Service Layer

This layer hosts the Web Service APIs themselves, implemented as JAX-WS[6] web services, syntactically described in WSDL format and hosted on instances of the Glassfish server. According to the design methodology followed by all web service modules, each service is associated with a single device family bundle. In turn, each service provides as many operations as the corresponding hardware offers. This choice was not so straightforward, as, in general, even a single service can hold many operations encapsulating code to handle every device family. On the other hand, numerous single-operation services could provide a single function for each module. As hardware is physically separated in device networks of the same brand or

6 Java JAX-WS reference implementation: https://jax-ws.java.net/

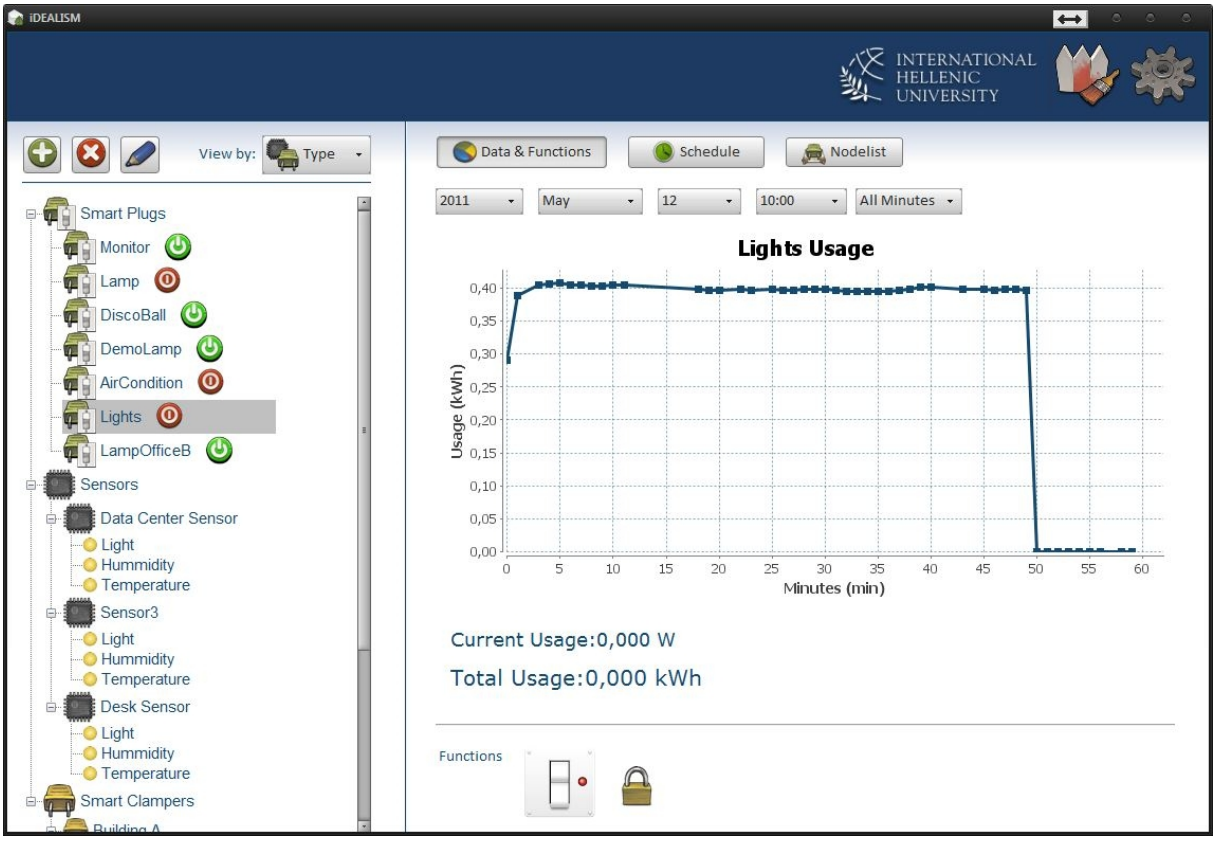

Fig 7. The iDEALISM administrative application, giving an overview of all installed devices, monitoring and management capabilities.

communication protocol, services were made to reflect that physical distinction, also matched essentially by the driver modules. Note that, since drivers already perform protocol unification of data coming from the devices, a single service could indeed handle more than one device type. Instead, this integration is allowed to be resolved on the semantic layer, in a more efficient and machine-interpretable way.

Smart Plug Service: The service provides sensor operations for polling and getting each Plug's on/off binary status, read power consumption (in 1 or 8 sec interval), and various hardware information, such as internal clock and firmware. It also provides actuator operations for switching them on or off. All operations require a target Plug ID as input.

Sensor Board Service: This service provides numerous sensor operations that receive a target board ID and return the value of measured temperature, humidity, luminance or the board's battery level.

Smart Clamper Service: The Smart Clamper Service basically provides one operation for receiving the target Clamper's ID and return its power measurement.

Z-Wave Service: Similarly, operations of this service accept a target node ID parameter and each returns the measured temperature, humidity, luminance, motion detection and CO_2 air concentration level.

4.3. Semantic Layer

The semantic layer of the architecture consists of two components, the ontological infrastructure and the semantically described service endpoints. The ontological infrastructure generally entails one or more ontologies that serve as a lexicon of interrelated concepts, meant to semantically describe entities. After a thorough review of state-of-the-art, the BOnSAI (Smart Building Ontology for Ambient Intelligence) ontology was designed, aimed at the effective description of Smart Building, Ambient Intelligence, Service and Sensor Network concepts [10]. An important aspect of ontology design is reuse. As the very aim of any ontology is to provide semantic interoperability outside the borders of a particular system implementation and across the Web community, it must be universally adopted. This has indeed been an issue in Semantic

Web technologies overall, since there has been no convergence to commonly-used models. Hence, BOnSAI extends leading existing ontologies found in each domain such as the Semantic Sensor Network (SSN) ontology and the OWL-S upper ontology for services.

Existing approaches handle knowledge storage differently. In some approaches, a subset of sensor data is stored in ontologies, so that it can be reasoned upon and queried [17]. Such data can accumulate to very large quantities in large-scale systems such as Smart Cities. In such cases, stream processing techniques used in Big Data have been combined with semantic models to optimize performance and resource management, as in [18]. As opposed to storing knowledge, the proposed method relies on the semantic annotations of services to handle semantics on-the-fly, applying rules and logic. The main ontology, BOnSAI, serves as a lexicon that describes terms and defines relationships, and not as a knowledge base. Thus, when performing reasoning on real-time context data, information is retrieved much faster coming directly from the sensors. The resulting SAWSDL files semantically describe input and output of services as well as the nature of operations themselves. All SAWSDL-defined annotations (the so-called *Model References*) originate from the BOnSAI ontology and are derived from specific needs of applications, which are going to exploit them (following usage-driven design).

Operations are annotated as either *SensorOperations* or *ActuatorOperations*. These annotations are currently used by rule-based software agents for both rule authoring and invocation. The safe assumption made here is that SensorOperations are suitable to serve as rule conditions (left-hand side), while ActuatorOperations are suitable for rule results (right-hand side). Hence, during the authoring of rules, available options are dynamically retrieved from available semantic service descriptions.

Inputs and Outputs are annotated according to their semantic type e.g. *sensor:ID*, *Temperature*, *Humidity*, *Power* etc. These annotations can serve as a basis for a wider range of clients, e.g. service discovery, matchmaking and composition algorithms. Namely, a software or human agent that is looking for a particular service can form semantic queries e.g. based on inputs and outputs, and in turn achieve automatic service composition. In order to focus on rule-based energy savings and management, service matchmaking and composition are left outside the scope of this work.

Another use of the ontology is to serve as a reference for the hierarchies and relationships of sensor data streams on semantic portals. One of its branches describes a hierarchy of the *sensor, multi-sensor, actuator* and *sensor-actuator* classes that help instantiate our particular device network. These devices are linked through the ObjectProperty *returnsParameter* with a range of *Parameter* class to clarify the nature of measured data. The *Parameter* class is further defined in subclasses *Energy, Power* and *EnvironmentalParameter,* such as *Temperature, Humidity, Luminance, CO_2 level* etc. Every device's physical location is defined through the ObjectProperty *hasLocation,* linking it to a *Location* instance. *Location* subclasses include *Building, Room* and *Floor*, which are further described through interconnecting properties. E.g. a *Building* can have many *Floors* and a *Floor* many *Rooms* (through *hasFloor, hasRoom*). These branches of the ontology enable a semantic indexing of data streams that can be queried based on combinations of device type, nature of measurements and location in the physical space. The index has been used building a semantic portal visualizing and aggregating data feeds in historic charts for monitoring.

5. Applications for Monitoring and Management

Based on the remote, universal web service API, the proposed architecture allows for multiple, diverse applications to coexist in the system. Figure 5 shows the information flow for different use case scenarios. A client application, e.g. an intelligent agent or a mobile client, first retrieves suitable services of interest, advertised on the service broker. It then invokes operations directly on the corresponding web servers, which in turn operate on the underlying sensor and actuator networks. Requests are enveloped in SOAP and translated to device protocols, e.g. ZigBee or Z-Wave and vice versa, for responses.

The complete ecosystem of applications built on top of the middleware is shown in Figure 6. According to the purpose they serve, the different applications developed so far can be categorized in monitoring, administration, automation and integration with external applications. Monitoring applications provide extensive insight into the building's and individual appliances' energy distribution. This has allowed experts to formulate tailored energy-saving policies, in order to autonomously reduce consumption. Meanwhile, administrative applications allow immediate interventions by authorized staff in case of emergencies. Data center monitoring applications provide insight into its unique properties. Rules and automation applications

come in the form of intelligent agents incorporating the aforementioned policies. Connection to external providers in the context of Smart Grids is provided via OpenADR integration. The following subsections examine each set of applications separately.

5.1. Monitoring and Administration

Although automations and intelligent agents already manipulate the infrastructure without human intervention, monitoring and administrative applications do play an important role in the system. Their purpose is twofold; for one, they allow human administrators to manually manipulate the devices, but most importantly, to observe and analyze historical data of the building's behavior. Consequently, they are able to take decisions and author new optimized policies to be applied. Secondly, monitoring applications allow simple users, such as the university's staff and students, visitors and web users globally to observe the infrastructure's behavior and status.

Monitoring applications target all popular platforms to ensure accessibility. iDEALISM is a desktop application for monitoring and management, developed in Java [9]. As shown in Figure 6, a service client handles all service calls, manipulating sensors and actuators. The application is meant to be used by administrators only, such as the Smart IHU, IT and security staff, enabling them to monitor all historical and real time data before taking action. Additionally, the application allows them to group devices together and get aggregated values to better assess the actual environmental and consumption status of the building. As shown in Figure 7, iDEALISM displays a list of all deployed devices on the left, per type, group or room. Upon selecting one of them, its historical data is shown on the top right (e.g. power, luminance, temperature etc.), while actuator functions are available on the bottom right (e.g. switch or lock).

The mobile application counterpart, named PlugDroid, offers manipulation capabilities, plus some added-value facilities powered by the smartphone hardware [9]. It has been implemented in the Java Android SDK, due to its openness and popularity, by incorporating a Web Service client to connect to the middleware. Users are able to create, edit and maintain lists of sensors and actuators, invoke functions with one touch and view data. The abundance of integrated smartphone sensors in Android devices presents an excellent basis for augmented-reality applications. The

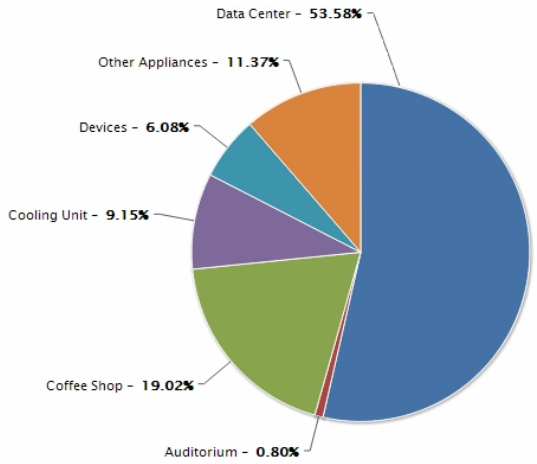

Fig 8. Building A energy consumption disaggregation for July 2014, enabled by fusion of Smart Clampers and Plugs

Smart IHU Android application currently takes advantage of the QR/barcode scanning function via the camera to offer such a feature. Specifically, mobile users can scan QR codes printed on various connected appliances or sensors in the building, interacting, in a way, with the physical environment. This action adds target appliances to the user's set of monitored devices. Thereafter, they are able to monitor and manage them remotely at any time, over the web.

Finally, the main web application, the Smart IHU portal[7], presents a view of historical sensor readings targeting students, guests and outside visitors abroad. The portal is one among many web applications built on top of a general-purpose monitoring subsystem. The Smart IHU portal offers open access for any web user to select and view the most vital sensor data, such as power consumption of the building or the data center, selected environmental measurements (e.g. CO_2 levels in the labs), waste metrics and more. Another web application taking advantage of the same underlying infrastructure is an on-site flat TV-screen at the university's reception, which displays current, a seven-day average and a three-day line chart of power consumption, together with weather information and the course schedule timetable for the day.

The monitoring subsystem that enables these applications contains a collective database of historical sensor data to be queried and displayed. Data is filled in by a persistent web service client. A dedicated module for sensor fusion within this subsystem provides aggregated data in a homogeneous manner. For instance, a common required task is the fusion of Smart

7 Smart IHU Portal http://smart.ihu.edu.gr/

Clamper and Smart Plug data. Both sensors measure power consumption, but in different sampling frequencies and scales. The fusion module provides universal queries for the disaggregation of daily total building energy consumption to Data Center, Coffee Shop, Cooling Unit, Auditorium (as measured by Clampers) and Devices (as measured by Plugs). As a result of this fusion, it is possible to monitor energy disaggregation results. E.g. Figure 8 shows daily average energy consumption distribution for July 2014, taken directly from the Smart IHU portal. Devices are further decomposed into graphs per school and appliance type, by fusion of various smart plug networks.

All in all, the monitoring process, through the combined usage of iDEALISM, PlugDroid and the web portal, according to user scope and situation, offers deep insight into the building's consumption and environmental conditions. In turn, it allows authoring more targeted and, thus, more effective energy saving policies. The applications also allow to assess the effectiveness of energy-saving measures taken, either by human administration or by autonomous policies, such as those employed in the experimental section.

The overall monitoring has shown, that the Data Center, i.e. the building's server room, is responsible for around 50% of the buildings yearly consumption. This massive energy consumption is mainly related to the cooling units of the datacenter – energy efficiency metrics green datacenter operation is discussed in the following section. Quite similarly, the coffee shop and cooling unit, which account for around 20% and 10% of consumption during summer, can only be altered through infrastructural changes, not optimizations. Meanwhile, devices monitored with smart plugs can be optimized as presented in the corresponding section, showing promising savings. Finally, around 10% of the consumption is still unaccounted for and yet to be discovered by installing additional sensors.

5.2. Datacenter Efficiency Monitoring

An important application of the system concerns the data center monitoring. Smart IHU support real time computation and representation of energy efficiency metrics, based on the green grid association [19, 20]. The deployed AMI used for metric computation is presented in Figure 9. The smart clamp network was installed at the central electricity installation of the data center to monitor energy consumption. The smart plugs were used for capturing power consumption of individual components (such as the servers), whereas the SNMP requests were used to capture parameters

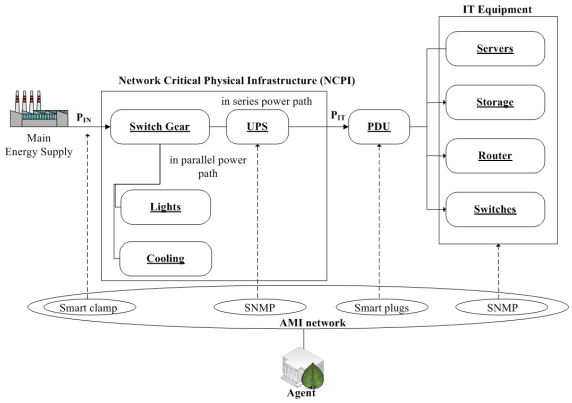

Fig 9. The monitored data center.

from the UPS, the servers and the routers. These parameters are used as input variables to the equations of metrics. The sampling rate of measured data was 5 minutes.

Smart IHU provides real-time computation and monitoring for the following metrics:

Power Usage Effectiveness (*PUE*) is defined as the ratio of the total facility input power over the power delivered to IT. Datacenter infrastructure Efficiency (DCiE) is the inverse of PUE and can be described in a mathematical form as

$$PUE = \frac{1}{DCiE} = \frac{P_{IN}}{P_{IT}}, 1 < PUE < \infty \qquad (1)$$

The PUE metric characterizes the performance or the power wasted in the network critical components of the datacenter (NCPI). The closer the PUE is to 1, the more efficiently the NCPI equipment operate. A study over twenty four datacenters has established an average PUE reference value of approximately 1.83 or 0.53 (53%) [21], whereas the average PUE of the measured data center for a period of one year is 2.1. This proves that 47% of the consumed energy is delivered to the IT and the rest is used for cooling and in general NCPI operations. This value is not constant over time and depends on the operation of the air conditioners and the load of the UPS.

Telecom metric (*Mbits/kWhr*) models the efficiency of the telecommunication equipment and, more precisely, the data transmission efficiency of the data center. In a mathematical form the metric is given by:

$$M_T = \frac{\sum_{i=1}^{k} b_i}{E_{IN}}, Mbits/kWh \qquad (2)$$

In the above equation k is the number of routers in the datacenter and b_i is the total number of bits coming out from the i-th router during the assessment window.

E_{IN} is the total consumed energy by the datacenter during the assessment window. The energy was computed by the power consumption values, taking into account the duration of the assessment window. The assessment window must be defined in such a way to allow the capture of datacenter's variation over time. The metric M_T can measure the underutilization of routers or redundant components in the system.

Server metric (*ssj_ops/kWhr*) models the efficiency of the computational part of the data center as a function of the average CPU utilization. It is correlated to the benchmark values of the servers named as *Specrate* and *Specpower*. The CPU utilization for each server in the datacenter is averaged over the assessment window of time T. The used metric models the server productivity (MSP) related to the total datacenter power consumption.

$$MSP = T\frac{\sum_{i=1}^{n} U_i S_i \left(\frac{CC_i}{CB_i}\right)}{E_{IN}}, ssj\_ops/kWh \qquad (3)$$

In the above formulation n is the number of servers, U_i is the average CPU utilization over T of server i, S_i is the SPECpower ssj_ops/sec at 100% server utilization of server i, CC_i is the nominal clock speed of the CPU of server i, CB_i is the clock speed of the CPU, used to establish B_i which is the rate benchmark result of server i. This metric models datacenter productivity and the correlation of the actual useful work to the maximum possible work if all servers were running at 100% utilization.

Datacenter operation was captured in terms of metrics described in (1-3). Table 2 presents the aggregated findings, while more detailed results can be found in [22]. The PUE metric presented an average value of 1.9. This reflects to energy losses of approximately 50% due to the operation of NCPI equipment. An immediate action to reduce PUE is to perform free cooling. The router equipment presented an average value of M_T=2Mbits/kWh fluctuating periodically according to traffic (actual employee activity in the building). This small value can be justified by the small number of university outbound traffic. Regarding the server utilization, an average value of MSP=100000 ssj_ops/kWh was measured. Underutilization was observed and an energy-efficient technique is to implement virtualization that will switch off CPU power during low traffic periods.

5.3. Intelligent Agent

While simple client applications manually manipulate actuators and recover sensor data, intelligent applications allow the automatic, smart administration of power consumption. One approach towards that end is the use of intelligent agents that incorporate knowledge, conduct decision making and manipulate the environment. To begin with, this section presents two different logic approaches explored in previous work of ours [11]. These approaches have then been combined into a single hybrid agent architecture and embedded into an application suitable for the university staff. The application was experimentally piloted in the university, demonstrating positive experimental results, as presented in the next section.

5.3.1. Productive and Defeasible Logics

Traditionally, agents employ various forms of reasoning following different principles of logics. For the purposes of energy saving in the proposed system, two different approaches have been explored.

An agent based on productive logic is responsible for reactive rules, where fast response is of the essence. The agent, named Wintermute, incorporates an instance of the JESS rule execution engine. Users are able to author and maintain automation policies using a GUI, implemented with JavaFx. The GUI is enhanced with semantic information, dynamically acquired from the descriptions of services that are online at the time. This is carried out based on the assumption that SensorOperations are suitable for rule condition-predicates and, likewise, ActuatorOperations are well-suited for action predicates. Hence, during rule authoring, users are presented with available options in a dropdown box.

The Wintermute agent interfaces with semantic web services to periodically obtain facts of interest and update its knowledge of the world's state as perceived via the sensors. These facts, along with authored policies in the agent's knowledge base, are used for the reasoning process which, in turn, triggers and fires rules that invoke semantic web services to manipulate appliances. Overall, this agent's functionality lies in the straightforward approach of monitoring values and

Table 2. Data Center Performance

Metric	Value	Actions needed
PUE	1.9	Freecooling
M_T(Mbits/kWh)	2	
MSP (ssj_ops/kWh)	100000	Virtualization

immediately firing up rules, at the expense of having to resolve conflicts during the rule authoring process.

For example a simple thermostat rule (*r1*), in JESS syntax appears as follows:

```
(defrule ruleThermostat
       (call GetTemperature 9BA14E ?x0)
       (test (< ?x0 20) )
=>
       (call SwitchOn CA7712)
)
```

where `9BA14E, CA7712` are the IDs for the thermal sensor and the heater's power actuator, respectively. In this example, a second rule *r2* that switches off the heater when no motion is detected is defined. This rule indirectly conflicts with the previous one, as their results contradict. This rule should preferably dominate over *r1,* hence, *r1* has to be redefined to include `(¬motion)` in its conditions. Proceeding with the rule set, a third energy-saving policy *r3* involves entering a so-called 'Saving Mode' where flexible, relatively excessive devices such as heating are switched off when crossing a power level threshold. Accordingly, for this rule to obtain maximum superiority, all subordinate rules *r1, r2* must explicitly enumerate the negation of all conflicting rules stated. Apparently, as the rule set grows, the left-hand side of all subordinate rules grows exponentially. For one, this issue introduces a huge hassle for the rule author to carefully review and maintain the rule base, which ultimately takes a non-intuitive form. Secondly, the method presumes that rule authors have complete access, knowledge and privileges over the rule base, which might not be the case in such an environment.

Therefore, a second agent paradigm is introduced, incorporating a defeasible logic rule engine (*SPINdle* [23]). Defeasible logics feature highly intuitive and compact rule-authoring, coupled with a significantly more flexible conflict resolution scheme, implemented through a binary rule superiority relationship. Hence, rules become short, intuitive declarations that are easy to maintain, while defining their priority handles conflict resolution. The rule creator can use a variety of user interfaces to author such sets, such as *SPINdle's Defeasible Logic Theory Editor*[8], or the much more flexible *S*^{2}*DRREd* (*Syntactic-Semantic Defeasible Reasoning Rule Editor*) [24]. In our approach, the superiority declarations were used for defining three different rule clusters: preferences, maintenance and emergency. These three rule groups provide different authorization levels to different users. Simple end-users only have authoring rights to the preference rule set, which has the lowest priority. Power users have access to maintenance and emergency rules of higher superiority, resolving the matter of privacy and security of the system. The following are examples of rules belonging to each of the three categories:

```
# preference rule
p1: tempLow(X) => switchOff(X,cooler)

# maintenance rule
m1: motion(X) => ¬switchOff(X,cooler)

# emergency rule
e1: alert(X) => switchOn(X,alarm)
```

Defeasible logic rules in our approach are represented in first-order logic (FOL); thus, e.g. rule p_1 reads as "*If the temperature inside a room is low, then switch off the cooler in this room*". Similar interpretations accompany the rest of the sample rules. Section 6 features rule implementations containing specific thresholds for temperature, luminance etc. According to what was described previously, rules belonging to each level have escalating priority (i.e. prevail) over rules belonging to the previous levels. This constitutes a representative demonstration of a key aspect of the flexible conflict resolution mechanism featured in defeasible logics. By introducing superiority relationships between pairs of defeasible (i.e. rules that can be retracted) rules (e.g. $m_1 > p_1$), one can easily introduce different levels of rule execution rights. Conflict resolution in defeasible logics also involves other mechanisms inherent to the logics, like different rule types (denoted by different arrow types) and conflicting literals, which, however, are out of the scope of this paper and are not further elaborated, but are tightly integrated into the implementation.

5.3.2. Hybrid Agent

While previous work has explored and applied the two agents operating in parallel, in a mutually exclusive manner [11], this work presents a hybrid agent combination. The reactive and deliberative agent both present merits useful in the Smart Building scenario. However, they could not co-exist without adaptation, due to conflicts when intervening on common resources, i.e. appliances; that is, because the agents

8 SPINdle Defeasible Logic Theory Editor: http://spin.nicta.org.au/spindle/tools.html

Fig 10. Manager App view, showing a list of devices in various rooms with On and Off capabilities

have no knowledge and conflict-resolution mechanism for deciding on the use of common resources.

The proposed hybrid agent combines aspects of both agents into one, by incorporating two distinct, horizontal layers of different behavior, as shown in Figure 6. The reactive layer implements fast, responsive behavior. It contains a rule-set for emergencies and maintenance urgencies, and invokes the monitor module to refresh relevant facts most frequently. The deliberative layer on the other hand, contains considerably more rules and updates relevant facts less often. Rule sets can again be executed in SPINdle. The agent takes advantage of semantics similarly, distinguishing between SensorOperations and ActuatorOperations. While the agent can handle any user-defined rule set, it uses semantics to map them to actual service invocations to retrieve facts and apply actions. For example, a condition of type Temperature is resolved to the corresponding SensorOperation of output Temperature.

5.3.3. Rule App and Manager App

Meanwhile, it can be argued that the agent alone is not enough for efficiently handling an open, real environment pilot deployment. Namely, due to the occurrence of emergencies or exceptional occasions during 24/7 operation, there might be a need to bypass the rule system. In general, the Smart IHU system should be robust, available and usable at all times, mainly enforcing the energy-saving policies. Offering such a mechanism for bypassing the rule system also helps users who are not familiarized with technology to feel more confident during the pilot. Hence, while users might configure and adjust the policies, they should still be able to manually manage the devices, possibly overriding one or more policies.

According to this requirement, two applications were designed and developed, to co-exist in the Smart IHU system, namely Manager and Rule App. For piloting, both applications can be configured to target a selection of IHU rooms, which enables different piloting scenarios; e.g. a single instance can control the whole building or distributed instances can control different parts. Note that in the latter scenario, agent communication is required for sharing common resources or co-operation towards a common goal. These scenarios are to be explored as future work.

The Smart IHU Manager App allows manual management and overrides policies. It initially retrieves associated rooms and appliances from the central building taxonomy. Users can then select appliances from a tree hierarchy and manually switch them on or

Fig 11. Rule App view, showing a list of available rooms, current measurements and active rules for room SciTech1

off through a Web Service client, overriding the policies. A view of its main user interface is given in Figure 10, where the user is allowed to manage the On or Off state of selectable devices in respective rooms listed on the left. In contrast to previously built administrative applications, Manager App functionality has been kept to a minimum, i.e. it excludes monitoring the status and statistics of devices, in order to maximize performance.

The Smart IHU Rule app, on the other hand, is considerably more complex. Its purpose is to automatically manage the infrastructure by continuously enforcing the policies. Initially, it also loads associated rooms and appliances from the central taxonomy, but also environmental and power sensors. Rule App incorporates an instance of the hybrid agent to which it streamlines sensor measurements as facts. The agent deliberates and enforces fired rules through the application's Web Service client. The GUI allows users to adjust the policies at all times, i.e. select rules, thresholds and monitoring intervals. During piloting, users are provided with two ways of adjusting the system to their needs in cases of conflict. Primarily, they can adjust thresholds of a given rule or even deactivate it using Rule App. In case this does not suffice, the user can override its decision by using Manager App and directly manage the setting. Figure 11 displays the application's user interface, from where the user can pick one of the available rooms. Consequently, he is presented with the current sensor measurements (on the top center) and a view of current rules and thresholds (on the bottom center).

The improvement that Rule App introduces over the previous rule-based approaches [11] is that it presents the underlying logic in a very user-friendly manner, suitable for robust piloting. Previously built agent applications were meant to be used by system experts and, thus, allowed to view and control the full scope of the rule sets. On the contrary, Rule App, which is designed for university staff, contains a predefined set of policies and a GUI to allow its configuration (Figure 11). Essentially, it masks a hybrid agent instance with the predefined rule-set, of given semantics. Users can only alter numerical threshold values and not types/semantics. The application's current rule set is thoroughly presented in Section VI. Also the hybrid agent itself can handle any user-defined rule set of both reactive and deliberative behavior.

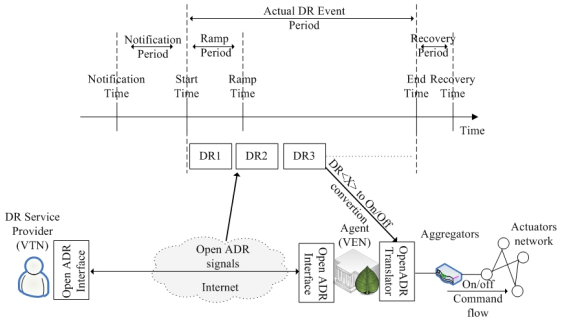

Fig 12. Smart IHU and OpenADR integration.

5.4. Integration with OpenADR

Recently developed standards, such as OpenADR [12] and Smart Energy Profile (SEP[9]) are used as a common language of communication among smart actuator networks to provide demand response services. Focusing on the OpenADR standard, pricing signals are broadcasted by a central server (the DR service provider) to end devices. The whole scheme is equivalent to server/client architecture. The server is the Virtual Top Node (VTN) whereas the smart end devices are the Virtual End Nodes (VEN). The OpenADR signal encapsulates information regarding the DR event start and end time and also DR signals that incorporate pricing or power/energy thresholds for discrete time steps. For the Smart IHU system, the required architecture to support OpenADR signals is presented in Figure 12 and includes: (a) an OpenADR interface that is integrated at the smart IHU agent of the building, and (b) an OpenADR translator, also attached at the agent that is responsible for translating the DR signals in the OpenADR packets to actual command flow in the smart actuator network of the building (AMI). In this architecture, the agent can consider the DR signals of the packet and according to the data captured by the AMI and the list of possible smart actuators to decide upon which appliances to switch on/off.

6. Experimental Results

The purpose of the experiment was to evaluate the monitoring and autonomous energy savings of the system, while ensuring its acceptability by end-users in a real-world setting. In this context, this work aims to evaluate realistic system usage of expert-written rules

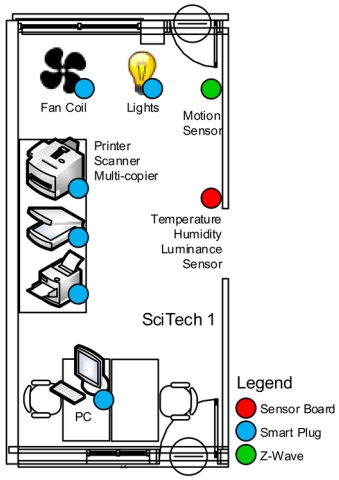

Fig 13. Deployment of sensors in SciTech1 for pilot experimentation

and not simulated demand-response signals from providers, since actual ones are currently not available. Therefore OpenADR signals are not sent during the pilot.

The proposed combination of Rule App, with the underlying hybrid agent, and Manager App were deployed and piloted at the IHU premises, in order to experimentally enforce energy-saving policies. The combined usage of iDEALISM and web portal has allowed to initially author policies and measure savings after the experiment. Namely, iDEALISM has provided per-appliance monitoring insight in respect to environmental conditions such as light and temperature, while web portal displays per-appliance consumption distribution and savings.

In detail, energy distribution within the building has already revealed that a large percentage of the daily energy consumption is due to factors not yet dealt with, e.g. the cooling unit, the data center and the coffee shop. These factors had to be excluded from the measurement of savings. Even so, previous work [11] has already established that experiments targeting the whole university at once are hard to assess. Therefore, the evaluation targets the controlled and monitored devices at a given office, namely the course office of the Science and Technology School. The sensor and actuator setup is shown in Figure 13. To guarantee that this method can be extended to the whole building in the future, it was ensured that the given office presents no exceptional properties, i.e. working hours and equipment, compared to the rest of the offices.

9 SEP: http://www.ti.com/lit/ml/slyt432/slyt432.pdf

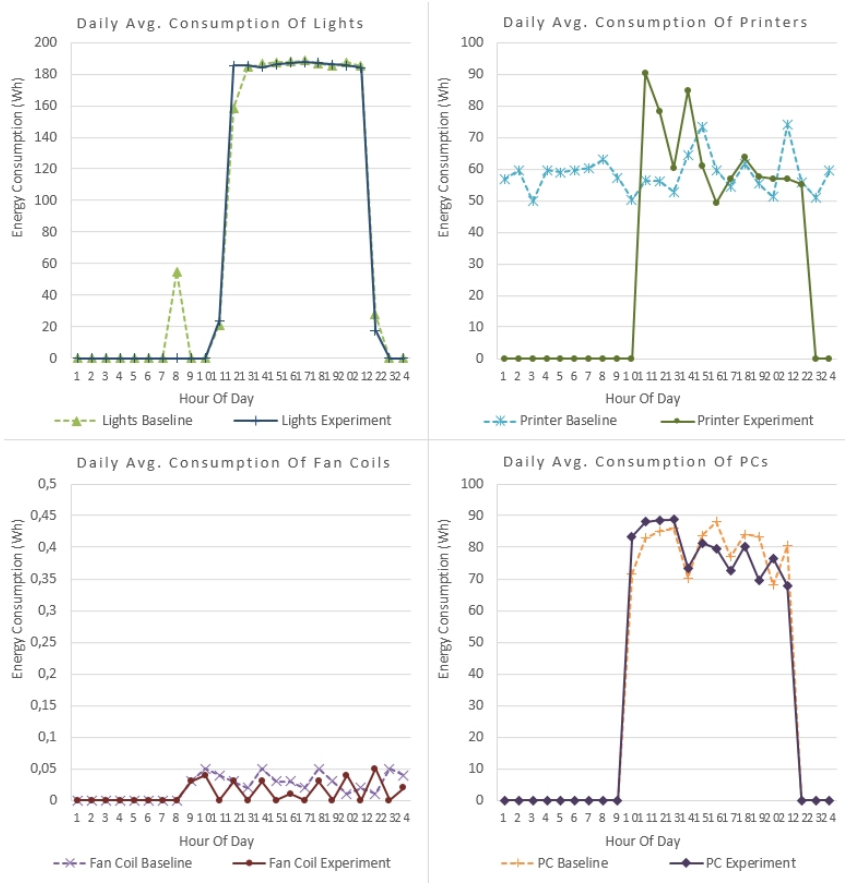

Fig 14. Daily average energy consumption per hour of devices during pilot testing compared to measured baselines

More fine-grained monitoring in respect to environmental conditions has revealed energy waste in respect to lighting and heating/cooling. In some offices, temperature sensors have measured extreme values, i.e. 30°C during winter and 20°C during summer, which reveals overusing heating and cooling equipment beyond comfort. Also, lighting is continuously on during working hours while environmental sensors report adequate natural light near the window. Taking user comfort into account, the temperature and luminance rule have been devised to allow a domain expert set a preference threshold for each measure. If the desired luminance threshold is reached, perhaps by natural light, the lighting is kept off.

```
p01: luminance(X, Y), Y >= 700 -> daytime(X)
p02: luminance(X, Y), Y < 700 -> ¬daytime(X)
p03: daytime(X) => switchOff(X, light)
p04: ¬daytime(X) => switchOn(X, light)
```

As described previously in subsection 5.3, rules are expressed in FOL. For instance, rule p_{01} reads as "*Whenever the luminance inside a room is above 700 cd/m^2, then the system should derive that it is daytime in the specific room*". Note that having a representation like daytime(X) (i.e. "it is daytime in room X") might seem redundant at first glance (shouldn't daytime be independent of rooms?); nevertheless, this formalism allows for more flexible behaviors, e.g. rooms in the basement may require different policies with respect to lighting. Heating and cooling now operates until reaching a certain threshold set by the expert, while without the agent, the central thermostat control of the building has overworked the appliances.

```
p05: temp(X, Y), Y < 20 -> tempLow(X)
p06: temp(X, Y), Y >= 20 -> ¬tempLow(X)
p07: tempLow(X) => switchOn(X, heater)
p08: ¬tempLow(X) => switchOff(X, heater)
```

Another observation was that printers and multifunctional machines are entering maintenance mode and consume substantial amounts of energy during night or day. Additionally, there have been occasions where users have left lights on overnight. This has led to defining a motion rule, which switches printers and lights off after the course of a duration threshold, to

prevent stand-by consumption and appliances left on by mistake.

```
p09: ¬motion(X, Y), Y > 30 -> roomEmpty(X)
p10: roomEmpty(X) => switchOff(X, printer)
p11: roomEmpty(X) => switchOff(X, light)
```

Finally, an additional rule, in accordance to Smart Grid principles, allows defining a total per room consumption threshold. Crossing the threshold switches off flexible appliances, which in our case are lights and heating/cooling. This rule can help satisfy provider demands (e.g. over OpenADR) to lower consumption upon request.

```
p12: consumption(X, Y), Y > 370 -> savingMode(X)
p13: savingMode(X) => switchOff(X, heater)
```

Threshold values for all rules have been set to reasonable values for lighting (700 cd/m^2), temperature (20^{o}C), and absence tolerance period for motion (30'). The personalized power threshold for the room was set to be able to use all basic appliances (370W). The full ruleset is available online in machine-readable format (see Appendix).

The duration of the experiment was two weeks, i.e. from 08-May-2014 to 21-May-2014. During this period, the office staff was able to access both the Manager and Rule App from a dedicated laptop, at all times. They could freely set thresholds, activate or deactivate policies or the Agent altogether. In order to measure savings, a baseline equal to the daily average consumption measured over an equal time period of two weeks was set. The baseline period would have to assimilate environmental conditions, such as weather and people's presence, as much as possible. Therefore, baseline consumption was measured as the daily average of two weeks before the experiment, i.e. between 24-Apr-2014 and 07-May-2014.

Arguably, the length of the experiment could be longer. However, extending its duration for more than a month, would likely decrease the safety to draw conclusions. This is due to (approximately) monthly variations in significant conditions such as: a) office occupancy due to alterations in event and lecture schedule and b) central cooling or heating usage due to altered weather conditions. Notably, the system was still operating and saving energy after the course of the experiment. But based on the above, it is not safe to compare and deduce energy savings over altered conditions. Given the variation in such significant factors, one would have to re-draw the baseline for each similar period and effectively reset the experiment. For this reason, it was ensured that the chosen period to draw the baseline and deduce savings involved stable

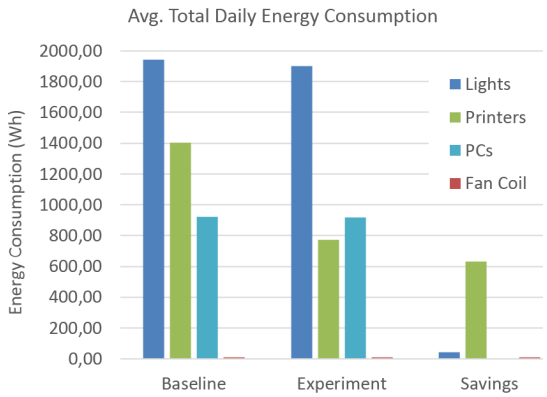

Fig 15. Average total daily energy consumption per device for measured baseline, pilot testing and their difference (savings)

weather conditions and (adequate) office occupancy. In the future, a more methodic and longer study could be set up in place, as the one in [25].

Figure 14 shows individual daily averages per hour of the day, for both time intervals and for various devices. Regarding consumption of lights, the motion rules (rules p_{09} and p_{11} presented above) managed to eliminate a morning peak during 6am – 8am. This peak was due to the cleaning crew leaving the lights on while interleaving rooms which accounts for around 1% of the office's daily consumption. Other than that, consumption of lights stayed unaffected, but the system did ensure that the lights stayed off during the night (when the office was unoccupied). Concerning printers, the respective motion rules (rules p_{09} and p_{10}) brought about notable savings, as they eliminated printer standby consumption during nighttime. Since the printer consumes nearly as much energy in standby mode as when it is used (at least in the particular office instance), savings during 0am – 6am inactivity account for around 15% of the room's consumption, resulting in a total of 16% savings.

Meanwhile, some rules did not manage to bring about any savings. Namely, the luminance rules (rules p_{01}-p_{04}), setting the lights on or off according to preference was left unused, as natural sunlight does not suffice in the office. Therefore, the lights had to be always on during working hours. Regardless of natural sunlight, the luminance rule could prove effective in future works, if the preference can be mapped to a dimmer value, resulting in some energy savings. The cooling/heating rules (rules p_{05}-p_{08}) were ineffective as well. For one, consumption of the fan coils is notably low, accounting for hardly 0.1% of the room's total. The cooling and heating aspect of the building in general will be targeted in future works as well. Namely, a different method is developed to aim at the

central cooling unit, responsible for a large percentage of the building's consumption. Finally, PCs were not targeted by any rule, due to their importance, but were measured and presented for the sake of completeness.

All in all, the system managed to save around 16% of the room's total daily energy consumption. Figure 15 shows the total daily average amounts of energy during the baseline and experiment interval. Their difference is shown in the "Savings" column, which helps assess the amount and the origin of savings. All savings originate entirely from the Rule App, as the Manager App was left unused, due to no exceptional occasions or emergencies occurring. This fact proves the non-intrusiveness and acceptability of the system, in the context of such an office. The measured savings seem quite optimistic, especially if applied to the whole building. Specifically, if the application is adopted by all offices, up to around 1% of the total building's consumption can be targeted (i.e. 16% savings of device consumption which accounts for 6% of the total building consumption in July, Figure 8).

7. State of the Art

The state of the art in the field of AmI, in general, includes many application domains such as health, Ambient Assisted Living, agriculture, multimedia and smart offices. Most of these approaches also employ the use of web services and semantics but follow the more complex top-down approaches of upper ontologies for services such as WSMO[10] or OWL-S[11] in [26, 27] or custom ontologies as in [16, 28]. On the contrary, the approach proposed in this work, employs the bottom-up lightweight and W3C recommended SAWSDL[12] standard for semantically annotating web service descriptions. Apart from being more recent than upper ontologies and W3C recommended, the SAWSDL approach allows the system to interact closer with the service implementations i.e. bindings. It also implements a custom universal middleware based on SAWSDL for lightweight semantic interoperability, outside the borders of the local environment.

Regarding middleware, other approaches target different environments and hardware. The system presented in [29] unifies various protocols such as X10, UPnP and Lonworks. Other middleware [30] provide higher level services, such as sensor fusion and user profiling. The cloud also plays an important role as part of some existing middleware. Some works use the cloud for facilitating the communication between devices [31] or for accessing cloud computing resources [32]. On the other hand, the middleware used in the proposed system is specifically tailored to target hardware, currently not supported by existing, open middleware. Additionally, it provides access to local sensor and actuator functions and does not require the use of the cloud for additional computing resources or communication. Also, the provided functions are rather primitive and do not require further analysis on middleware level. The semantic interpretation of those primitives is rather implemented in the application layer, when needed e.g. in the intelligent agent.

As far as rule-based smart environments are concerned, one approach introduced in [15] is a meta-language defined over JESS, to syntactically enhance the rule authoring process in ambient applications. However, this additional syntactic layer, named the Event-Control-Action (ECA) model, is far less flexible and extensible over the defeasible logic paradigm used in Smart IHU. Other similar approaches include SESAME-S [33], an all-in-a-box smart home prototype that uses ontologies and JESS reasoning to enforce rules. The approach in [34] also uses agents, web services and ontologies to store and reason on energy data. The main issue of both works is the lack of conflict resolution, enforcing the execution of all triggered rules in a serial manner and scalability. Contrasted, the proposed Smart IHU platform features a flexible and highly intuitive conflict resolution mechanism for rule execution, based on the non-monotonicity of the underlying defeasible logics. Additionally, Smart IHU does not store most information in ontological form, but rather relies on semantic annotations using SAWSDL, facilitating scalability, flexibility and extensibility to more clients. It also employs a hybrid combination of reactive and deliberative behavior based on two-layers of defeasible logic reasoning.

8. Conclusion and Future Work

Smart IHU is an integrated platform which combines principles from the fields of AmI, Semantic Web, Web Services, Intelligent Agents and the Smart Grid. Integration at its core exposes ambient sensor data and actuator functions through a middleware based on

10 WSMO W3C Submission: http://www.w3.org/Submission/WSMO/

11 OWL-S W3C Submission: http://www.w3.org/Submission/OWL-S/

12 SAWSDL W3C Recommendation: http://www.w3.org/TR/sawsdl/

Web Services. The system also provides semantic interoperability based on ontologies and SAWSDL. Applications have been designed to support a variety of use cases such as administration and extensive external monitoring of energy distribution. After thorough monitoring, energy-saving policies can be authored and assessed. For this purpose, they are integrated into an intelligent hybrid agent, of both reactive and deliberative behavior, which autonomously manages the building based on semantics and rules. A pilot system evaluation has shown up to 16% daily energy reduction in a university office. The agent also provides integration with future Smart Grids, as it is able to interpret and react to demands expressed in the OpenADR protocol.

Future plans include deploying the policies on the whole building in a distributed manner. Such an effort would require the agents to be able to communicate and interact with each other and such a framework has been investigated in [35]. Through this effort, the central cooling system is to be targeted by coordinating all distributed agents in an adaptive manner. The experiment will not only be extended in space, i.e. other rooms, but also in time. Conducting the experiment over a long time period, longer than a month, will require, as mentioned before, to establish new, per-month baselines according to the ever-changing weather and occupancy conditions. After those baselines have been established, and the system is piloted for more a year, it will be possible to safely estimate monthly energy savings of the platform on this specific building and setting.

Another goal is to extend the platform with more advanced context sensing capabilities. It is also interesting to expand Smart Grid integration at a semantic level. Currently, semantic services are used within the building, while external providers rely on OpenADR. On the contrary, integration with more standards, domain and information models and secure, semantic web services at grid level could provide far more interoperability, as in [36].

Finally, advancements in the fields of Computer Vision could allow the system for more fine-grained sensing of human activities performed in the premises. Also, more sensors can be investigated to provide more accurate presence detection and user profiling.

ACKNOWLEDGEMENT

The authors wish to thank the reviewers for the fruitful comments which have helped to improve this manuscript. They would also like to thank Mr. John Argyriou, Mr. Konstantinos Gottis, Mr. George Pilikidis and Mr. Thodoris Tsompanidis for their past contributions to the foundation of this work.

APPENDIX

The ruleset, available online along with instructions[13], illustrates the SPINdle-specific syntax for the rule base presented in Section VI. It contains a set of threshold values for luminance, temp, motion and power consumption (the same as specified in the paper) and a set of sample input values for demonstration purposes. The ruleset can be executed both offline, using SPINdle, or even online at[14].

REFERENCES

[1] L. Wenpeng, "Advanced metering infrastructure," *South. Power Syst. Technol.*, vol. 3, no. 2, pp. 6–10, 2009.

[2] M. H. Albadi and E. F. El-Saadany, "Demand response in electricity markets: An overview," in *IEEE Power Engineering Society General Meeting*, 2007, vol. 2007, pp. 1–5.

[3] T. G. Stavropoulos, A. Tsioliaridou, G. Koutitas, D. Vrakas, and I. Vlahavas, "System architecture for a smart university building," in *Artificial Neural Networks–ICANN 2010*, Springer, 2010, pp. 477–482.

[4] V. Issarny, D. Sacchetti, F. Tartanoglu, F. Sailhan, R. Chibout, N. Levy, and A. Talamona, "Developing ambient intelligence systems: A solution based on web services," *Autom. Softw. Eng.*, vol. 12, no. 1, pp. 101–137, 2005.

[5] C. Le Gal, J. Martin, A. Lux, and J. L. Crowley, "Smart office: Design of an intelligent environment," *IEEE Intell. Syst.*, vol. 16, no. 4, pp. 60–66, 2001.

[6] M. Eisenhauer, P. Rosengren, and P. Antolin, "Hydra: A development platform for integrating wireless devices and sensors into ambient intelligence systems," in *The Internet of Things*, Springer, 2010, pp. 367–373.

[7] T. Kleinberger, M. Becker, E. Ras, A. Holzinger, and P. Müller, "Ambient intelligence in assisted living: enable elderly people to handle future interfaces," in *Universal access in human-computer interaction. Ambient interaction*, Springer, 2007, pp. 103–112.

[8] H. Sun, V. De Florio, N. Gui, and C. Blondia, "Promises and challenges of ambient assisted living systems," in *Information Technology:*

13 Ruleset: http://lpis.csd.auth.gr/ontologies/ontolist.html#smartihuruleset

14 SPINdle demo: http://spin.nicta.org.au/spindle/demo.html

New Generations, 2009. ITNG'09. Sixth International Conference on, 2009, pp. 1201–1207.

[9] T. G. Stavropoulos, K. Gottis, D. Vrakas, and I. Vlahavas, "aWESoME: A web service middleware for ambient intelligence," *Expert Syst. Appl.*, vol. 40, no. 11, pp. 4380–4392, 2013.

[10] T. G. Stavropoulos, D. Vrakas, D. Vlachava, and N. Bassiliades, "BOnSAI: a smart building ontology for ambient intelligence," in *Proceedings of the 2nd International Conference on Web Intelligence, Mining and Semantics*, 2012, p. 30.

[11] T. G. Stavropoulos, E. Kontopoulos, N. Bassiliades, J. Argyriou, A. Bikakis, D. Vrakas, and I. Vlahavas, "Rule-based approaches for energy savings in an ambient intelligence environment," *Pervasive Mob. Comput.*

[12] C. McParland, "OpenADR open source toolkit: Developing open source software for the Smart Grid," in *Power and Energy Society General Meeting, 2011 IEEE*, 2011, pp. 1–7.

[13] T. Erl, "Service-Oriented Architecture," *Concepts Technol. Des.*, 2004.

[14] G. Koutitas, A. Karousos, and T. Brown, "3D propagation modeling from high elevation systems to indoor environments," in *Proc. IEEE Conf. on Electromagnetic Field Computation, Athens, Greece*, 2008.

[15] L. Daniele, P. D. Costa, and L. F. Pires, "Towards a rule-based approach for context-aware applications," in *Dependable and Adaptable Networks and Services*, Springer, 2007, pp. 33–43.

[16] V. Issarny, M. Caporuscio, and N. Georgantas, "A perspective on the future of middleware-based software engineering," in *2007 Future of Software Engineering*, 2007, pp. 244–258.

[17] V. Kumar, A. Fensel, G. Lazendic, and U. Lehner, "Semantic Policy-Based Data Management for Energy Efficient Smart Buildings," in *On the Move to Meaningful Internet Systems: OTM 2012 Workshops*, P. Herrero, H. Panetto, R. Meersman, and T. Dillon, Eds. Springer Berlin Heidelberg, 2012, pp. 272–281.

[18] Sylva Girtelschmid, Matthias Steinbauer, Vikash Kumar, Anna Fensel, and Gabriele Kotsis, "On the application of Big Data in future large-scale intelligent Smart City installations," *Int. J. Pervasive Comput. Commun.*, vol. 10, no. 2, pp. 168–182, May 2014.

[19] J. Haas, M. Monroe, J. Pflueger, D. J. POUCHET, E. P. SNELLING, A. Rawson, and A. F. RAWSON, "Proxy proposals for measuring data center productivity," *Green Grid*, 2009.

[20] G. Koutitas and P. Demestichas, "Challenges for energy efficiency in local and regional data centers," vol. 1, no. 1, pp. 1–32, 2010.

[21] L. A. Barroso, J. Clidaras, and U. Hölzle, "The datacenter as a computer: An introduction to the design of warehouse-scale machines," *Synth. Lect. Comput. Archit.*, vol. 8, no. 3, pp. 1–154, 2013.

[22] L. Chiaraviglio, R. Bruschi, A. Cianfrani, O. M. J. Ortiz, and G. Koutitas, "The TREND Meter: Monitoring the Energy Consumption of Networked Devices," *Int. J. Bus. Data Commun. Netw. IJBDCN*, vol. 9, no. 2, pp. 27–44, 2013.

[23] H.-P. Lam and G. Governatori, "The making of SPINdle," in *Rule Interchange and Applications*, Springer, 2009, pp. 315–322.

[24] E. Kontopoulos, T. Zetta, and N. Bassiliades, "Semantically-enhanced authoring of defeasible logic rule bases in the semantic web," in *Proceedings of the 2nd International Conference on Web Intelligence, Mining and Semantics*, 2012, p. 56.

[25] V. Kumar, A. Fensel, and P. Fröhlich, "Context based adaptation of semantic rules in smart buildings," in *Proceedings of International Conference on Information Integration and Web-based Applications & Services*, 2013, p. 719.

[26] G. Thomson, S. Bianco, S. B. Mokhtar, N. Georgantas, and V. Issarny, "Amigo aware services," in *Constructing Ambient Intelligence*, Springer, 2008, pp. 385–390.

[27] S. M. Iacob, J. P. A. Almeida, and M. E. Iacob, "Optimized dynamic semantic composition of services," in *Proceedings of the 2008 ACM symposium on Applied computing*, 2008, pp. 2286–2292.

[28] A. Paz-Lopez, G. Varela, J. A. Becerra, S. Vazquez-Rodriguez, and R. J. Duro, "Towards ubiquity in ambient intelligence: User-guided component mobility in the HI^3 architecture," *Sci. Comput. Program.*, vol. 78, no. 10, pp. 1971–1986, 2013.

[29] A. Capone, M. Barros, H. Hrasnica, and S. Tompros, "A new architecture for reduction of energy consumption of home appliances," in *TOWARDS eENVIRONMENT, European conference of the Czech Presidency of the Council of the EU*, 2009, pp. 1–8.

[30] A. De Paola, S. Gaglio, G. Lo Re, and M. Ortolani, "Sensor$_9$k: A testbed for designing and

experimenting with WSN-based ambient intelligence applications," *Pervasive Mob. Comput.*, vol. 8, no. 3, pp. 448–466, 2012.

[31] H.-W. Yeh, C.-H. Lu, Y.-C. Huang, T.-H. Yang, and L.-C. Fu, "Cloud-Enabled Adaptive Activity-Aware Energy-Saving System in a Dynamic Environment," in *Dependable, Autonomic and Secure Computing (DASC), 2011 IEEE Ninth International Conference on*, 2011, pp. 690–696.

[32] S.-Y. Yang, "Developing an energy-saving and case-based reasoning information agent with Web service and ontology techniques," *Expert Syst. Appl.*, vol. 40, no. 9, pp. 3351–3369, 2013.

[33] A. Fensel, S. Tomic, V. Kumar, M. Stefanovic, S. V. Aleshin, and D. O. Novikov, "Sesame-s: Semantic smart home system for energy efficiency," *Inform.-Spektrum*, vol. 36, no. 1, pp. 46–57, 2013.

[34] Z. Wang, R. Yang, and L. Wang, "Multi-agent control system with intelligent optimization for smart and energy-efficient buildings," in *IECON 2010-36th Annual Conference on IEEE Industrial Electronics Society*, 2010, pp. 1144–1149.

[35] T. G. Stavropoulos, E. Rigas, E. Kontopoulos, N. Bassiliades, D. Vrakas, and I. Vlahavas, "A Multi-Agent Coordination Framework for Smart Building Energy Management," in *3nd International Workshop on Artificial Intelligence Techniques for Power Systems and Energy Markets (IATEM) in conjunction with DEXA 2014*, Munich, Germany, 2014.

[36] S. Rohjans, *Semantic Service Integration for Smart Grids*, vol. 14. IOS Press, 2012.