CISTER
Research Centre in
Real-Time & Embedded
Computing Systems

Conference Paper

The Arrowhead Framework applied to Energy Management

Rafael Rocha*
Michele Albano*
Luis Lino Ferreira*
Flávio Relvas*
Luisa Matos

*CISTER Research Centre

CISTER-TR-180403

2018/06/13

The Arrowhead Framework applied to Energy Management

Rafael Rocha*, Michele Albano*, Luis Lino Ferreira*, Flávio Relvas*, Luisa Matos

*CISTER Research Centre

Polytechnic Institute of Porto (ISEP-IPP)

Rua Dr. António Bernardino de Almeida, 431

4200-072 Porto

Portugal

Tel.: +351.22.8340509, Fax: +351.22.8321159

E-mail: rtdrh@isep.ipp.pt, mialb@isep.ipp.pt, llf@isep.ipp.pt, flaviofrelvas@gmail.com

http://www.cister.isep.ipp.pt

Abstract

Energy management in buildings can provide massive benefits in financial and energy saving terms. It is possible to optimize energy usage with smart grid techniques, where the benefits are enhanced when the energy consumer can trade the energy on energy markets, since it forces energy providers to compete with each other on the energy price. However, two hurdles oppose this approach: the devices providing control over appliances do not interoperate with each other; and energy markets limit trading activities to large quantities of energy, thus impeding access for small consumers. This work considers using the FlexOffer (FO) concept to allow the consumer to express its energy needs, and FO-related mechanisms to aggregate energy requests into quantities relevant for energy markets. Moreover, the presented system, named FlexHousing, is based on the Arrowhead Framework - a framework that simplifies design and implementation of distributed applications by means of normalizing communication via services - and exploits its Service Oriented mechanisms to provide device interoperability. The implemented FlexHousing system uses multi-level FO aggregation to empower either the final user, for example the owner of an apartment, to manage its own energy by defining their flexibilities, or to offload this responsibility to an energy manager who takes care of all the apartments in a building or set of buildings.

The Arrowhead Framework applied to Energy Management

Rafael Rocha, Michele Albano,
Luis Lino Ferreira, Flávio Relvas
CISTER Research Center, ISEP
Polytechnic Institute of Porto
Rua Dr. António Bernardino de Almeida
4200-072 Porto, Portugal
Email: {rtdrh, mialb, llf, 1140826}@isep.ipp.pt

Luisa Matos
Virtual Power Solutions
Instituto Pedro Nunes
Rua Pedro Nunes - Edifcio D
3030-199 Coimbra
Portugal
Email: lmatos@vps.energy

***Abstract*—Energy management in buildings can provide massive benefits in financial and energy saving terms. It is possible to optimize energy usage with smart grid techniques, where the benefits are enhanced when the energy consumer can trade the energy on energy markets, since it forces energy providers to compete with each other on the energy price. However, two hurdles oppose this approach: the devices providing control over appliances do not interoperate with each other; and energy markets limit trading activities to large quantities of energy, thus impeding access for small consumers. This work considers using the FlexOffer (FO) concept to allow the consumer to express its energy needs, and FO-related mechanisms to aggregate energy requests into quantities relevant for energy markets. Moreover, the presented system, named FlexHousing, is based on the Arrowhead Framework – a framework that simplifies design and implementation of distributed applications by means of normalizing communication via services – and exploits its Service Oriented mechanisms to provide device interoperability. The implemented FlexHousing system uses multi-level FO aggregation to empower either the final user, for example the owner of an apartment, to manage its own energy by defining their flexibilities, or to offload this responsibility to an energy manager who takes care of all the apartments in a building or set of buildings.**

***Index Terms*—Arrowhead, Service Oriented, Prosumer, Smart Grid, Interoperability**

I. Introduction

Energy management in buildings can provide massive benefits, with regards to financial gains, pollution reduction, and total energy saving, since the energy consumed in buildings is one of the major factors in global energy expenditures. For example, it is estimated that energy consumption by the residential and commercial sectors cover together 39% of the total energy consumption in the US, and that most of that energy is consumed in buildings [1], [2].

The application of smart grid techniques to buildings, such as remote control of appliances and autonomous acquisition of energy, can provide dramatic savings [3], especially when energy is bought from energy markets, since this forces energy providers to compete with each other on the energy price. Current advances are contributing to the feasibility of this vision, since communication protocols are converging to a standardized vision of the last mile of the smart grid [4], and there is increased competition between energy utilities, leading to consumers' savings. On the other hand, two hurdles oppose the application of the smart grid concept to buildings: the devices providing control over appliances do not interoperate with each other; and energy markets limit their trading activities to large quantities of electricity, thus impeding access for small consumers.

Currently, the field of energy management in buildings is very active, and several competing solutions have already been devised so far. In fact, it suffices to point out the many research efforts that aim at innovating in this area, given the multiple surveys published on the topic [2], [5]–[7]. Therefore, this paper attempts at providing a novel solution for this active problem, by using different tools and concepts.

To cope with the interoperability issue, this work proposes to use the Arrowhead Framework [8] to streamline communication between the devices. In the Arrowhead vision, all interactions are mediated by means of *services*, which are consumed and produced by software *systems* executed on *devices*. The current project makes use of smart plugs that follow custom protocols, however, these can be controlled with a simple Service Oriented Architecture (SOA) interface, and past work [9] has already studied how to extend the Arrowhead Framework to custom protocols by means of adapters. In the current work, we employ two different smart plug types, one (Sonoff Pow) being controlled locally, the second (Virtual Power Solution) accessed through an external service provider.

To solve the second hurdle, this work uses the FlexOffer (FO) concept to allow the consumer to express its energy needs [10]. FOs aim to balance energy demand and response by synchronizing consumption with production. They permit exposing demand and supply loads with associated flexibilities in time and quantity for energy commerce, load leveling, and different use-cases. In other words, a FO specifies an amount of energy, a duration, an earliest begin time, a latest finish time, and a price, e.g., "I want 50 kWh over 3 hours between 5 PM and 12 PM, at a maximum price of 0.25€/ kWh". In order for the FO to be relevant for the Energy Market, the FO concept considers that FOs can be combined or aggregated together when they are on the same energy grid.

All FO-related systems are already Arrowhead-compliant. A system called Aggregator receives FOs from all the households, combines them, and sends a unique aggregated FO to the Market. Multiple Aggregators can be employed in sequence, thus creating tree-shaped topologies, and producing FOs that are large enough to be sent to the Market. When the Market reply arrives, the responsibility of each Aggregator is to redistribute the energy between the underlying Aggregators, until it is delivered to the households supervised by the bottom-layer Aggregators. Later on, the smart plugs enforce the energy consumption schedule sent back from the Aggregator, and collect data regarding energy consumption of the appliances.

The implemented system, named FlexHousing (FH), allows the application of the FO concept to buildings in real settings. The multi-level approach enabled by the Aggregator empowers either the final user, such as the owner of an apartment, to manage their own energy by defining their flexibilities, or to offload this responsibility to an energy manager who takes care of a whole building or set of buildings. Other Aggregators will take care of aggregating the FOs of multiple buildings until they reach a magnitude of interest for the Energy Market.

The rest of the paper is structured as follows: Section II provides background information, in particular regarding the FlexOffer concept and the Arrowhead platform, which is used to provide FlexOffer-related services; Section III describes the architecture of the FlexHousing system of systems; Section IV provides details about the implementation of the FlexHousing system; Section V is devoted to the test results obtained while experimenting with the implemented system; Section VI draws some conclusions on the topic at hand.

II. Background Information

A. FlexOffer Concept

A FlexOffer is a data structure for expressing flexibility in energy consumption (or production). A FO contains a number of slices, each representing a minimum and maximum consumption for a given time period. A set of slices is called a profile, and can be used to express the consumption profile for an appliance. A FO allows the profile to be shifted in time by specifying an earliest start time and a latest end time. This provides two types of flexibility to a FlexOffer: energy flexibility in the bounds of the slice, and time flexibility in the start and end time.

A Schedule can be attached to a FlexOffer. The schedule includes a start time and an assigned energy amount for each slice. Fig. 1 shows an example of a FO. The green area shows the flexible energy, the grey area is the minimum required energy. The red lines show the final schedule, sent by the Aggregator.

A FlexOffer is intended to be used on a Virtual Market of Energy. The actors on the market are energy sellers and buyers, and flexibility sellers and buyers.

The energy buyers are private homes or companies, which have devices that can be controlled in order to utilize their flexibility. This way, the energy buyers act as flexibility sellers,

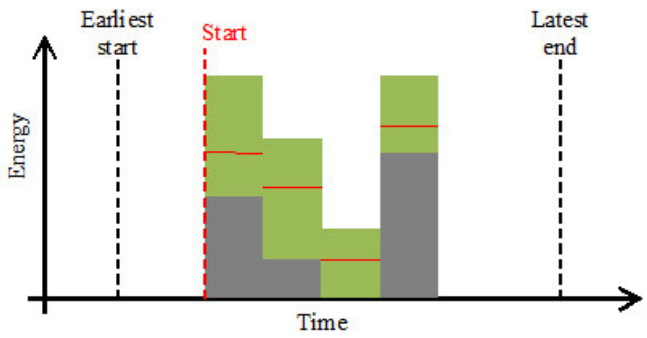

Fig. 1. Example of a FlexOffer

which generate FlexOffers based on the profiles and possible flexibility in their flexible resources. The FlexOffer is also associated to a default schedule that will be followed in case the FlexOffer is not sold. Along with the FlexOffer, the flexibility seller sends pricing information for the cost of deviating from its default schedule. This price can be calculated from local energy prices or loss in efficiency.

The energy sellers are energy producers that act as buyers with regards to flexibility, in order to shift energy consumption away from a possible grid overload. The pricing information is used to evaluate how much flexibility is worth buying. A new schedule is assigned based on the sold flexibility and the buyer compensates the seller based on the price. If a schedule is sold but is not followed, then the flexibility seller is penalized.

Since a single home or small company does not consume much energy compared to the capacity of the grid, the amount of flexibility offered is relatively small and therefore not very interesting to the buyers. For this reason, Aggregators are put in between the sellers and the market. The Aggregator receives several FlexOffers from different sellers and aggregates them together into larger FlexOffers, which are of interest to the buyers.

The FlexOffer concept was originally introduced in the MIRABEL project [10]. It has been further developed in the Arrowhead and TotalFlex projects [11]. Some research has been done to quantify the benefits of flexible resources on the energy grid [12].

B. The Arrowhead Framework Concepts

The FlexHousing system was developed on top of the Arrowhead Framework [8], which facilitates the development of service oriented distributed applications executed on top of Cyber Physical Systems.

The Arrowhead approach simplifies design and implementation of distributed applications by means of normalizing communication via services. All communications of a distributed application are mediated through services, exposed and consumed by systems. The systems of an Arrowhead distributed application are organized into a System of Systems (SoS), which is deployed as a local cloud – a bounded set of computational resources used by stakeholders to attain a goal.

A system can either provide and consume application services, and thus implement the functional requirements of a specific use case, or provide core services, which are diagonal to the use cases and provide support to non-functional requirements such as service registry, service discovery, quality of service, and security [13]. The set of systems that provide core services are delivered in the form of the Arrowhead Framework, and the core services deployed in the current SoS are described in Section III-A.

The rationale is that the systems of the SoS register themselves together with the list of services they produce or want to consume. The Orchestrator system collects all data regarding the SoS and matches systems and services to satisfy both the functional (which services are consumed) and non-functional (QoS, security, geographical location of the system producing the consumed service) requirements.

Supported by the Arrowhead Framework, the devices are able to set up a protected communication channel, enabling the execution of critical applications. Among them is the Virtual Market of Energy [11], which implements a service-oriented interface to trade energy and flexibility on one or multiple energy markets. The extension of the Arrowhead Framework to non-Arrowhead compliant components is done by means of adapters, which can act at different levels:

- Communication protocol: since Arrowhead is service-oriented, an adapter can be used to interact with components that, for example, exchange messages through publish/subscribe systems such as XMPP [14];
- Ontologies: Arrowhead considers devices, systems and services as actors in communication scenarios. Adapters and stubs should be used to hide a group of systems behind a unique Arrowhead system;
- Semantics: some frameworks associate timing characteristics to the communications, which assume semantics relevance. Since Arrowhead does not use this technique, an adapter may be necessary for mediating interactions;
- Syntax: the format of the messages can be different, thus a translation operation may be needed.

As it will be described in Section IV, the SoS presented in this paper interacts with custom smart plugs and sensors through a service-oriented interface, thus using adapters to perform a syntax translation between protocols.

III. System Architecture

The system's architecture, devised to support the FlexHousing concept, consists of a SoS (see Subsection II-B), and therefore is divided into a number of systems, as depicted in Fig. 2. In the Arrowhead sense, three of these, FlexHousing Middleware (FH Middleware), FlexHousing Front-End (FH Front-End) and FlexOffer Aggregator, are application systems, since they implement functionalities related to the business domain. All three systems interact at some point with the Arrowhead Framework, e.g., to perform service discovery to find the other systems. The FlexOffer Aggregator interacts with other Aggregators, and with the Virtual Market of Energy.

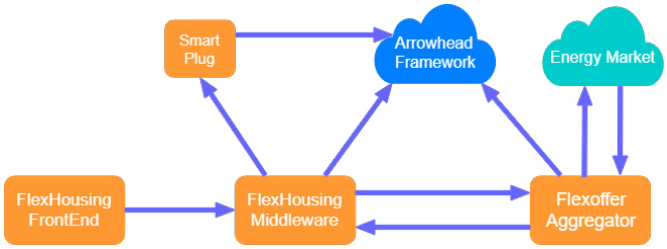

Fig. 2. Simplified architecture of the FlexHousing System of Systems.

Furthermore, the FH Middleware interacts with the house's smart plugs to manage energy consumption.

A. Arrowhead Core Services / Systems

The Arrowhead Framework provides services to support other systems regarding security, system and service registration, and service discovery and orchestration (see Subsection II-B). The local cloud providing FlexHousing services requires a number of basic core services that enable fundamental SOA properties like service registration, service discovery, authentication and authorization plus orchestration of SoS (see Fig. 3).

1) ServiceRegistry: The ServiceRegistry system allows a system to expose a service it is producing to the cloud, and allows consumer systems to discover services they wish to consume. It is used to ensure that all systems can find each other even if endpoints are dynamically changed. It supports a service registry functionality based on DNS and DNS Service Discovery (DNS-SD); since the Arrowhead Framework is a domain-based infrastructure. All Systems within the network that have services producing information to the network shall publish its producing service within the Service Registry by using the Service Discovery service. Within a system of systems, the Service Registry further supports system interoperability through its capability of searching for specific service producer features, i.e. an application service producer with a specific type of output. In short, it enables systems to publish their own application services and lookup others'.

2) Authorization: The Authorization system is responsible of controlling which service can be accessed by an authorized consumer. It consists of two service producers and one service consumer and it maintains a list of access rules to system resources (i.e. services). The Authorization Management service provides the possibility to manage the access rules for specific resources. The Authorization Control service provides the possibility of managing the access for an external service to a specific resource. The system uses the Service Discovery service to publish all its producing services within the Service Registry system.

3) Orchestration: The Orchestration system allows coordination (orchestration) of which producer will a certain consumer be able to employ/use. It is a central component of the Arrowhead Framework and in any SOA-based architecture. Orchestration is used to control how systems are deployed

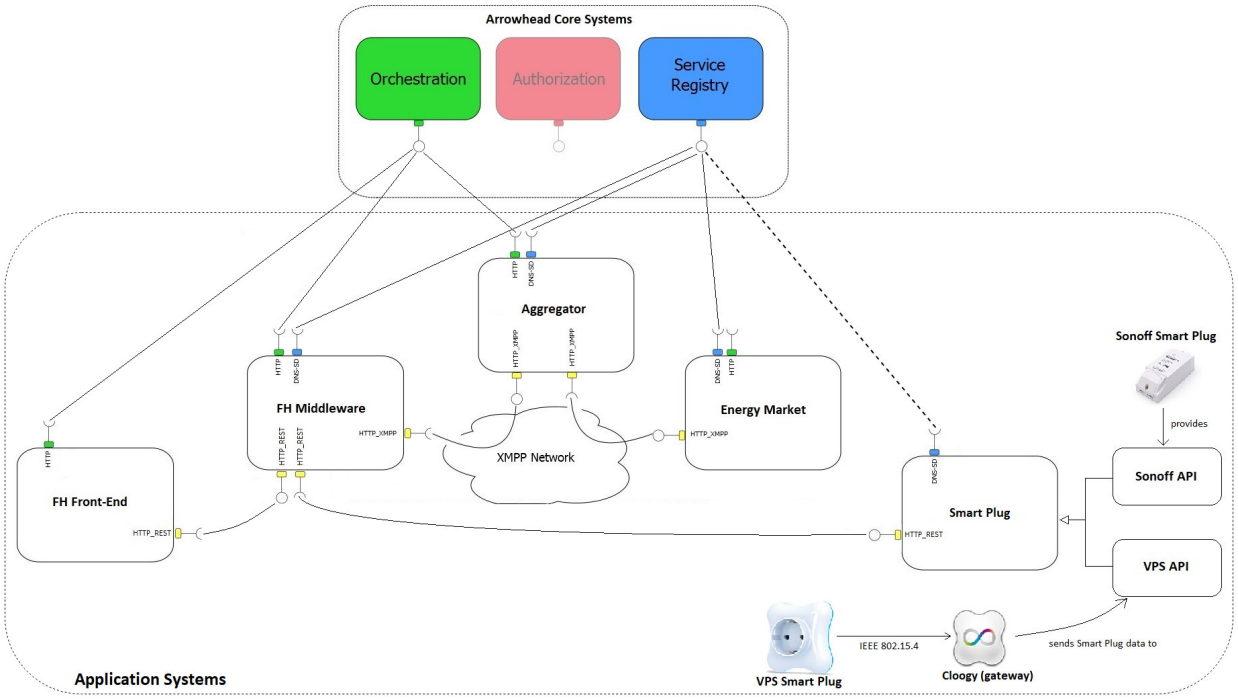

Fig. 3. Communication between the FH components, the third-party devices, the FlexOffer Aggregator, and the Energy Market, using Arrowhead Services.

and in what way should they be interconnected. Orchestration, in the context of SOA, can be viewed as the system that supports the establishment of all other systems through providing coordination, control and deployment services. It is utilized to dynamically allow the re-use of existing services and systems to create new services and functionalities. The application systems' services are initially seen as passive and being on standby. They are not connected at deployment or even during start-up of the system of systems. Their services can be managed to connect or be connected to others - to fulfill a specific need. The Arrowhead Framework currently supports REST-based Orchestration of services using, for example, REST or CoAP.

B. FlexHousing Middleware (FH Middleware)

The FH Middleware is responsible for the integration of the systems involved in the FlexHousing environment, and to facilitate the creation of FlexOffers. The FH Middleware focuses on automating the FO emission and appliance actuations, depending on the schedule that was retrieved from the Aggregator. Energy flow will be enabled depending on the Schedule that is active on a plug.

In particular, where FOs are concerned, the FH Middleware allows the FH Front-End to drive the creation of FOs, sends the FOs to the Aggregator via XMPP, receives energy schedules from the Aggregator, manages energy consumption based on the schedules from the Aggregators, and collects energy data which will then be used to verify the schedule's execution and improve the creation of future FOs.

In fact, the interactions between the FH Middleware and the user's appliances, portrayed as a *Smart Plug* in Fig. 3, can be mediated through a service-oriented interface provided by an external web service (e.g. VPS's REST API) or from the Smart Plug itself (e.g. a Sonoff's REST API), as seen in Fig. 4. Effectively, the FH Middleware is capable of retrieving data relative to actuation times and energy consumption, directly from non-Arrowhead compliant appliances (as long as these are connected to a Smart Plug) or from a service provider's REST API. This process is done through a software module developed for its respective smart plug brand and model. To include a new type of smart plug in the SoS, it is only necessary to develop an adapter to handle that smart plug's connection protocols and data format, while the rest of the process is managed by the FH Middleware.

The timeline of the provisioning of FOs to the Aggregator is studied to maximize the usage of energy markets. Effectively, the FH Middleware uses an heuristics to evaluate how much time it needs to send all configured FOs, and starts the process a number of minutes before midnight to be sure to complete it before the end of the day.

To simplify the deployment of the system, and the degree of interoperability, the FH Middleware makes use of the Arrowhead core systems. The FH Middleware registers itself on the ServiceRegistry system, to allow other systems, and in particular the FH Front-End (Subsection III-C), to discover it and use the services produced by the Orchestrator system to discover the Smart Plugs in the house of the user (Subsection III-F).

C. FlexHousing Front-End (FH Front-End)

The web-based application FH Front-End provides a means to engage with the FlexHousing environment, by displaying all

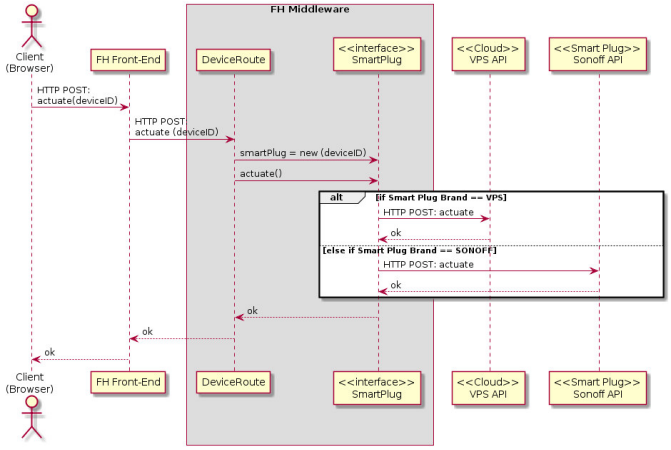

Fig. 4. Actuating a smart plug through the FlexHousing system

available features in the FH Middleware, which it discovers using the service provided by the Orchestrator system. This interface supports multiple users (and their respective roles, i.e. home owner, energy manager), allowing them to register a new smart plug, verify a room's or an appliance's current energy consumption, turn a smrt plug on/off, and create and send FOs for an appliance in the simplest way possible.

Regarding the registry of a new smart plug on the system, the user has to specify the smart plug's brand and model, so that the FH Middleware is able to know which protocols, targets, and APIs to use to ultimately connect to the smart plug.

When it comes to creating a FO, this process is divided into three steps: introducing basic details, choosing what kind of pattern to use, and creating the energy pattern for the FO. The first step requires the user to input the FlexOffer's name and the time period in which it must be applied. The second step requests the user to define the energy consumption pattern, which can be done either manually through the graphical interface, or automatically, using a model energy consumption pattern based on the user's energy profile.

By choosing to create it manually, in the third step the user can select the duration of the pattern and define the energy consumption for each 15 minute interval, by dragging the bars in the chart with the mouse.

If the user opts to create it automatically, the system itself defines the appliance's energy consumption pattern based on its past consumption data. To do so, certain algorithms from [15] were used to implement a way to identify energy patterns. However, the algorithm for identifying an energy pattern depends on what kind of appliance it is. If the smart plug is applied to, for instance, a wet-device – an appliance that has a certain daily routine (e.g. dishwashers, washing machines) – the system requests the user to input the hour when they typically turn it on. With that predicted activation hour, the system goes through every consumption entry and tries to perform a Pattern Sequence Matching (PSM). The PSM is used to predict values for various attributes of FOs, e.g., the number of time slices, energy profile, etc. On the other hand, if the appliance in question was a refrigerator, the wet-device process wouldn't work, since the refrigerator is constantly activating and deactivating throughout the day. Thus, for this case, the PSM algorithm was modified and new algorithms were developed to accommodate the situation. More specifically, the FH Middleware calculates the average value of these time segments of inactivity, determines the refrigerator's model energy profile with the PSM algorithm, and uses the two results so that it can ultimately create an energy pattern for a full day [16].

Regarding user roles, the FH Front-End allows for three types: the normal user who can visualize his energy consumption, the power user who can configure his own energy usage by setting up FlexOffers, and the energy manager who is entitled with the management of multiple houses in a complex.

Using the same options and features the energy manager sets up and customizes the FOs in the same way as the power user, after deciding which of the FH Middleware, such as apartment complexes, industrial facilities, or condominium, he is actually configuring by means of the same FH Front-End.

The authentication and authorization module for the manager is handled by the Arrowhead Framework, which also provides the addresses of the FH Middlewares that a specific FH Front-End can handle and control.

D. FlexOffer Aggregator

The Aggregator receives FOs from FH Middlewares, combines them with FOs from other sources into larger FOs, and then finally sends them to other Aggregators, or directly to the Virtual Market of Energy. Note that FOs need a specific magnitude before being able to be bid on at the Energy Market.

Afterwards, the Aggregator receives a response from the Virtual Market of Energy, which can be a refusal, or an energy schedule. In the latter case, the Aggregator disaggregates the response and sends the consumption schedule downwards to its respective FH Middleware. Many types of Aggregators may exist, some may be specific for a use case, such as the management of electrical motors, whereas others may be more generic. In addition, selecting the most adequate Aggregator also depends on the geographic region.

E. Energy Market

The Aggregators are able to schedule energy consumption by allocating it through a virtual market of energy, which communicates with appliances through FOs. The Energy Market allows to trade both energy and flexibility. (see Subsection II-A).

The Energy Market secures the balance in a logical subdomain within the grid, i.e. it ensures that consumption is equal to production. It utilizes the aggregated FOs from Aggregators for an internal energy balancing, and places FOs on the flexibility market for trading with energy producers. The Energy Market minimizes total costs by scheduling energy loads while respecting the constraints contained in the FOs (minimum/maximum power, earliest/latest start of energy

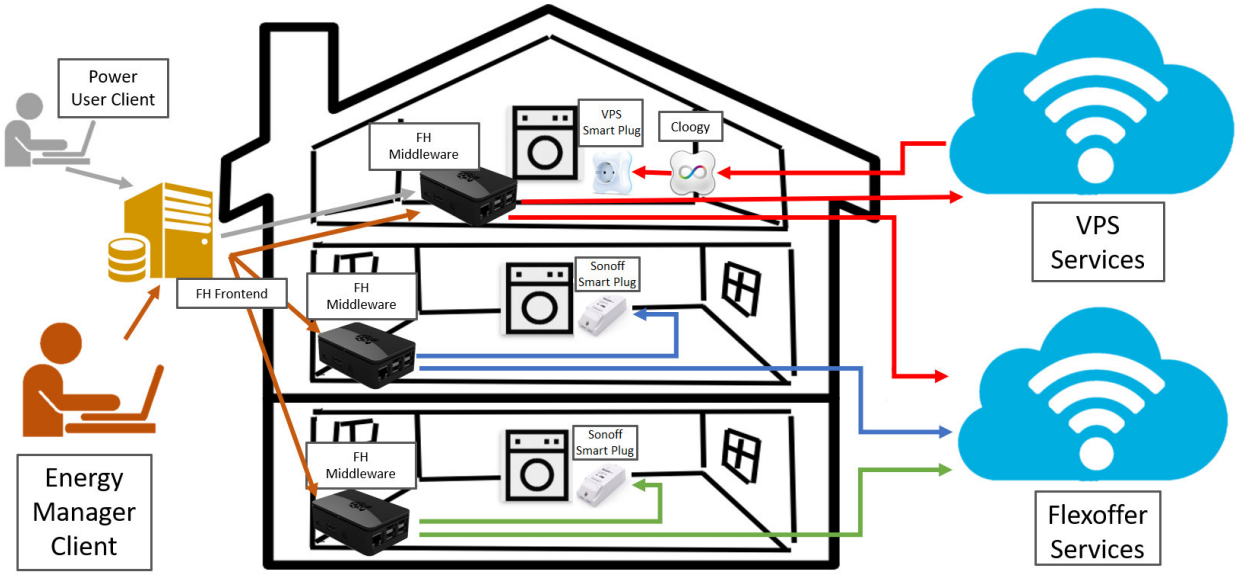

Fig. 5. Implementation of the FlexHousing SoS

consumption, etc.). In the case of the energy consumer, the net result of the interaction of the Aggregator with the Energy Market is to buy energy while selling the consumer's flexibility, attaining the goal of saving the consumer's money and supporting the energy producer in flattening off consumption peaks.

F. Smart Plugs in the User's Home

Common household appliances cannot be controlled remotely, and thus are not fit to FO compliance. One of the solutions to tackle this issue is to attach to the appliances a smart plug, which is placed between the appliance and the electrical outlet. The smart plug contains an energy switch actuator and a power meter sensor. The energy switch actuator allows to remotely switch on and off the appliance it is installed onto. The power meter sensor can collect data regarding energy consumption, to inform the user regarding the correct execution of FlexOffers and to allow for the creation of FlexOffers based on past consumption.

The FlexHousing SoS considers that devices are either smart plugs driving an appliance, or more complex devices that provide also smart plug functionalities, and in both cases they are called *smart plug* in this work. The interaction with the smart plug is through an API, capable of receiving requests and forwarding them back to the plug. The API can be exposed directly on the smart plug, or on an external service platform, and it is accessed in a uniform manner.

In the first case, the FH Middleware has direct access to the smart plug, for example through the local Wi-Fi network, without depending on external servers. In the second case, the topology usually features a gateway (in this case, the VPS Cloogy) in each user's home, which provides connectivity with the external service provider. Each time the smart plug collects data, it sends them to the external service provider through the gateway, where the data is then stored. Each time a system generates a request (see Fig. 4), be it to retrieve data or to control energy consumption, a message is sent to the external service provider. The provider then either answers directly, for example with data, or forwards it to the gateway, which routes it to each specific smart plug.

There are advantages and disadvantages for each approach. In particular, using an external service provider adds a dependency on a third-party when it comes to accessing the sensors' data. This can be an issue, for example, when the service provider's servers are down. On the other hand, this also allows data preprocessing and storage on the external service, thus offloading complexity from the application developer to a third party.

The FlexHousing platform enables both kinds of interaction, and it virtualizes the smart plug type, to ensure a more flexible and generic service access. In both cases, the appliance and its smart plug can be configured using custom techniques, for example statically on the FlexHousing application or through a web interface, or the smart plug can register itself using the Arrowhead ServiceRegistry core service, to make the smart plug discoverable to the FlexHousing platform.

IV. System Deployment

The FlexHousing SoS applies FO concepts to the real-life management of appliances based on FOs. The house is modeled as a set of smart plugs, organized into rooms (kitchen, living room, garage, etc) that pertain to buildings/houses. Each smart plug can measure and control energy consumption of an appliance, and it is either controlled using the VPS API and reachable through a Cloogy gateway, or it is a Sonoff.

A context diagram of the SoS of the pilot is depicted in Fig. 5. All the systems of the SoS are considered to interact with the Arrowhead Framework, and thus these interactions

TABLE I
SOME FH MIDDLEWARE REST API ROUTES

Method	URL	Result
POST	/FlexHousing/House/{HouseID}/Room/{RoomID}/Device	Registers a new smart plug in a specific house and room.
POST	/FlexHousing/House/{HouseID}/Device/{DeviceID}/Flexoffer	Creates a flexoffer for a specific smart plug and its appliance.
GET	/FlexHousing/House/{HouseID}/Device/{DeviceID}/Flexoffer	Returns a appliance's active flexoffer.
GET	/FlexHousing/House/{HouseID}/Device/{DeviceID}/Flexoffer/Schedule	Returns an appliance's active flexoffer's schedule.
DELETE	/FlexHousing/House/{HouseID}/Device/{DeviceID}/Flexoffer	Deletes an appliance's active flexoffer.
GET	/FlexHousing/House/{HouseID}/Flexoffer/GetAllActiveFlexoffers	Returns all active flexoffers.

are not represented. The users access the FH Front-End, which interacts with the FH Middleware only, since the latter contains all information for the management of the FOs and smart plugs. The FH Middleware interacts with the Aggregator's services.

To reach the smart plugs, the FH Middleware can either communicate with an external service provider (the VPS API, which is not Arrowhead-compliant) or contact directly the smart plug (Sonoff smart plug, not Arrowhead-compliant either). Thus, the FH Middleware must have adapters, to extend the reach of the Arrowhead local cloud to custom protocols. [9].

The VPS API is exposed by an external service provider and communicates with the Cloogy (in Fig. 5) to interact with the smart plugs attached to the appliance. The Sonoff smart plug exposes a REST interface. Both the VPS API and the Sonoff smart plug were integrated into the Arrowhead SoS by means of adapters, which leveraged on their service-oriented interfaces to provision a one-to-one mapping with the Arrowhead SoS.

In the following, the Arrowhead Framework is not described, but details can be found for example in [8].

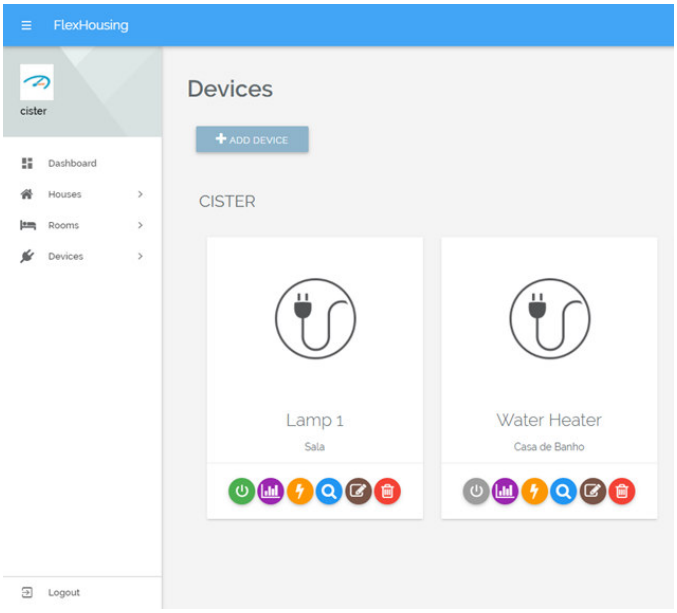

Fig. 6. Smart plugs section of the FH Front-End.

A. Implementation of the FH Middleware

The FH Middleware is built around the components required for communication. As such, it employs three different solutions, one for each communication path available.

For the interaction with the Aggregator, a Distributed Energy Resource (DER) agent was implemented [11], which is responsible for the emission of FOs using the XMPP protocols [14] used by the Aggregator, and the retrieval of Schedules through the same mechanism. The application was developed in Java, using Maven as a dependency manager to handle external modules and components. Apache Derby was selected for the system's database due to its easy integration with Java-based applications. For the services, a mix of Grizzly - for the HttpClient - and Jersey - for the service resources - was used.

There is no direct contact between the FH Middleware and the user, instead the FH Middleware exposes its services through a REST API (see Table I) which is then used by the FH Front-End. Those services are mostly CRUD actions around the rooms, smart plugs, FlexOffers and schedules.

For Aggregator purposes, the FH Middleware implements all the methods that allow the registration of the middleware on the Aggregator, the emissions of the FOs and the retrieval of the Schedules.

For the interaction with the VPS or Sonoff services, a HTTP client was implemented, which executes requests to query or interact with the smart plugs. As mentioned in Section III-B, the FH Middleware handles the data and communication with smart plugs from different manufacturers through an interface, independent of the underlying communication driver. This was done to separate the FH Middleware system from individual implementations for each smart plug, since each plug could differ greatly from others in how they handle data and other commands (e.g., while one service returns data in JSON, another one might return it in XML).

Even though most of the data is stored in the database, the FH Middleware keeps the most used/requested objects in cache to avoid excessive database queries. Any modification on the objects is reflected on the copy in cache but also in the database. This allows for the persistence of the data in the case of a power outage.

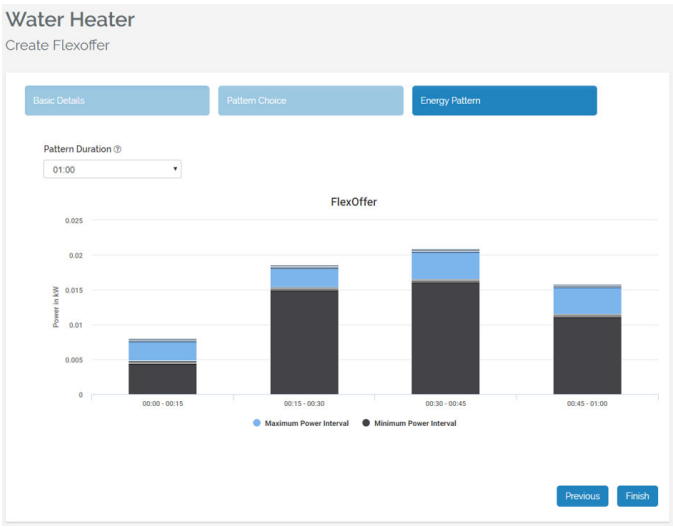

Fig. 7. Creation of a FlexOffer in FH Front-End: manually creating a consumption pattern.

B. Implementation of the FH Front-End

The web-based application FH Front-End provides a responsive, cross-browser compatible, graphical user interface. The FH Front-End is an MVC application, where its back-end is built in PHP, using Laravel 5.4 as its framework, while its front-end is built with HTML5, CSS3, and JavaScript.

The FH Front-End supports multiple users (and their respective roles, i.e. home owner, energy manager), allowing them to check a rooms or smart plugs current energy consumption, turn an appliance on/off through its smart plug, and create and send FlexOffers of an appliance in the simplest way possible. Therefore, the FH Front-End is composed of several sections: the overview page, the rooms section, and the smart plugs section.

The overview page provides insights on the total used energy, the number of FOs set up for each appliance, and other global statistics.

The rooms section allows to decide on which room, and later on on which smart plug, to work on to define involved FOs.

The smart plugs section has the same functionality as the rooms section, however it instead displays all available smart plugs from each house (see Fig. 6).

Fig. 7 shows the process of creating a FO manually, (see Subsection III-C). In the third step, the user can select the duration of the pattern (whilst respecting the time period defined in the previous step) and define the energy consumption for each 15-minute interval, by dragging the bars in the chart with the mouse. A similar interface is used to tune up a FO created using a pattern based on the appliance's past consumption data.

In the case of the Energy Manager role, the FH Front-End allows the same simplicity for the setup as the homeowner but without the constraints of having to constantly having to swap credentials.

TABLE II
SOME VPS REST API ROUTES

Method	URL	Result
POST	/api/actuations	Actuates Smart Plug.
GET	/api/consumptions/ from={Timestamp}& to={Timestamp}	Returns energy consumption values.
GET	/api/state/{SensorID}	Specifies if the appliance is on or off.

C. Aggregators and Virtual Market of Energy deployment

The setup at Aalborg University (AAU) in Aalborg, Denmark contains the communication infrastructure, the Aggregator, and the Energy Market implementation. All services are exposed through the Arrowhead framework.

The communication infrastructure consists of an XMPP server [14] to facilitate the HTTP-over-XMPP communication between the actors in the system. Aggregated FOs are sent to the Energy Market as selling bids, expressing flexibility being sold on the market.

The Energy Market receives selling bids from Aggregators or directly from energy consumers. It also receives buying bids from buyers, which are energy producers. At set intervals, the market will be cleared. The interval could be, for instance, fifteen minutes or daily. The clearing algorithm will match buyers and sellers such that highest buyers and lowest sellers will be favored.

D. VPS Services

As previously mentioned, VPS provides an external API (see Table II) for client systems to interact with their services, i.e., manage their VPS Smart Plugs. The API is service-oriented, and adheres to the RESTful principles of the HTTP protocol, thus providing a machine friendly, robust and predictable interface to the system functionalities.

The FH Middleware has a module to account for the location of each sensor and smart plug, and it uses the information to send messages to the correct component. Communication with the VPS Services is performed using a TCP/IP connection that hosts a HTTPS session, which exchanges data encoded using the JSON data format.

E. Sonoff Services

Sonoff smart plugs are cheap, generic, energy switches that, aside from switching the power on and off, and reading the current energy consumption, allow users to upload their own custom firmware on the switch's board. These boards are composed of a ESP8266 module (a low-cost Wi-Fi chip with full TCP/IP stack) to access the Wi-Fi network, and a HLW8012 current sensor to monitor the energy consumption.

A custom firmware was developed for the Sonoffs, for its inclusion into the FlexHousing SoS. The setup of a Sonoff smart plug requires an easy procedure, which can be executed by non-technical users. Once a Sonoff is connected to an appliance, and is plugged into an outlet, it will act as a Wi-Fi

TABLE III
SONOFF REST API ROUTES

Method	URL	Result
POST	/api/on	Turns Sonoff on.
POST	/api/off	Turns Sonoff off.
GET	/api/ consumptions	Returns current energy consumption values.
GET	/api/state	Specifies if the appliance is on or off.
POST	/api/config	Configures the Sonoff's settings.

access point, so that a computer or smartphone can connect to it. The user can configure the Sonoff by sending their home Wi-Fi's credentials to the smart plug through a POST request against a REST API exposed on the Sonoff. After that, the Sonoff smart plug will then be able to connect to the user's home Wi-Fi, to be accessed by the FH Middleware.

The Sonoffs provide, through a REST API (see Table III), their respective appliance's consumption values (see Listing 1), which updates every two seconds. The FH Middleware requests data periodically from the Sonoffs (in which, every message has a length of around 128 bytes), every five seconds, and store it into its database. Whenever the FH Front-End requests Sonoff data from the Middleware, the Middleware accesses its own database to deliver it.

```
{
    "consumption": "Active Power (W)=0,Voltage
        (V)=0,Current (A)=0.00,Apparent Power
        (VA)=0,Power Factor (%)=0.00",
    "name": "esp8266",
    "hardware": "esp8266",
    "connected": true
}
```

Listing 1. JSON snippet of the data returned from the Sonoff

V. TEST RESULTS

Some tests were executed on the implemented platform, to verify its correctness, and to ensure its performance characteristics.

A. Consumption patterns

The smart plugs allow to retrieve the consumption of any particular plug, independently if a FO was applied to it or not. In fact, as soon as a plug is registered in the SoS, it starts collecting data. Data is kept on the cloud of VPS services, accessible through GET requests.

Fig. 8 reports the result of a proof-of-concept data collection performed on a refrigerator. This test verified that the FlexHousing SoS is able to collect data with a reasonable granularity, to use them to build an energy consumption profile. The values were collected during a business day and during a weekend day, allowing to verify that human actions affect the energy consumption pattern for the appliance. In fact, during a business day the refrigerator is used on a regular basis, while during the weekend, especially on Sunday, the refrigerator is kept close, thus requiring less energy to maintain its cold temperature.

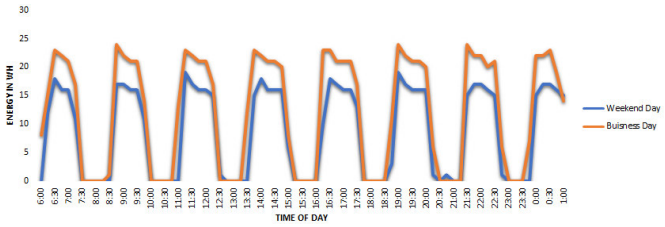

Fig. 8. Energy consumption pattern for a refrigerator

The energy usage matches with the theoretical consumption of the refrigerating cycle (Vapor-compression cycle) [17]. The periodicity is the same: between each peak, theres a period of no consumption, matching the idle state of the refrigerator. Aside from small statistical fluctuation in the data, the only difference between the days is the local maxima of each individual peak. While the weekend day averaged around $14.5Wh$, the business day averaged at $18.5Wh$.

B. Communications tests

These tests targeted the latency between the systems during communication. Since the latency inside the Arrowhead Framework was already targeted by analysis [8], this work focused on the latency for the orchestration process, and on the actuation over a smart plug. The testbed comprised systems deployed over 3 different devices. The first device hosted the Arrowhead core services, the second one is the FH Middleware which collects data periodically from Sonoffs and smart plugs, and the third device hosts a FH Front-End. The FH Middleware is previously registered on the Arrowhead ServiceRegistry.

This test executed 1000 experiments. Each experiment featured the FH Front-End using the Orchestration service exposed by an Orchestrator system to discover a FH Middleware, followed by 1000 service requests to the selected FH Middleware. Fig. 9 and Fig. 10 provide the Cumulative Distribution Functions for the delays for service orchestration and service consumption, and they provide a clear picture regarding the delay for the two operation, whose averages amount to $355.15ms$ and $7.30ms$ respectively.

Another test executed 30 request of actuation over a single plug controlled via the the VPS API (see Fig. 11). Two batches were performed, one by requesting the VPS API directly for the operation, and the other executed through the FH Middleware. The blue series represents the data collected when the request is directed to the VPS Services platform, and the orange series is related to mediating the interaction through the FH Middleware. The delay is less than $210ms$ and $270ms$ respectively, and it can be concluded that the FlexHousing internals do not impair the performance of the SoS with respect with the delay related to the access to the VPS API.

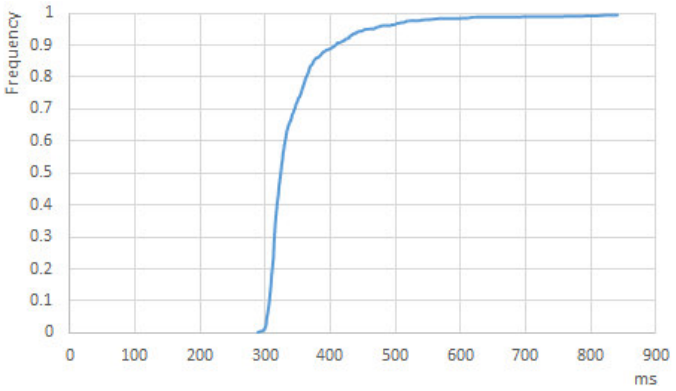

Fig. 9. Delay for orchestration process

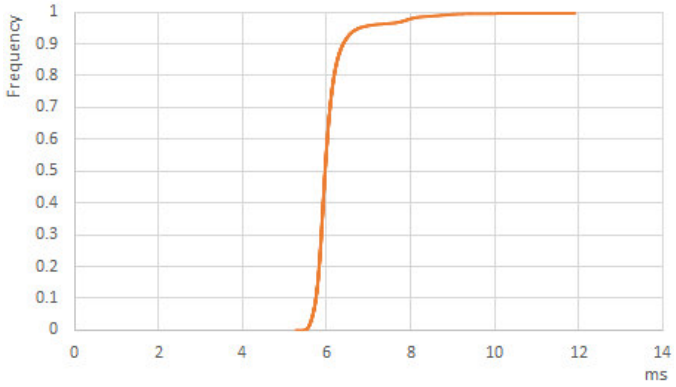

Fig. 10. Delay for service consumption

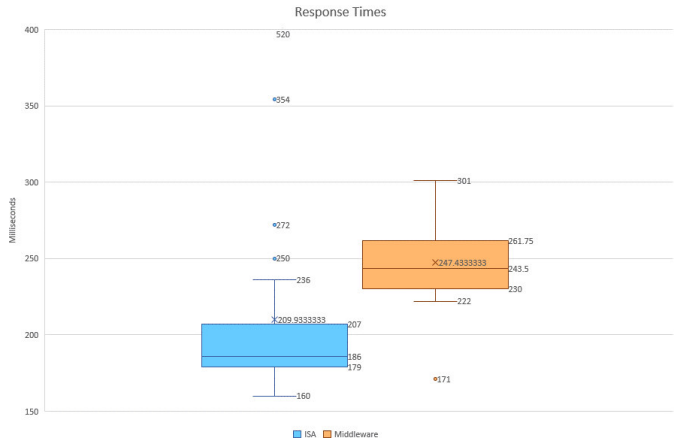

Fig. 11. Difference between response times from direct requests to the VPS API, as opposed to requests made through the FlexHousing Middleware.

VI. Conclusions

This paper presented the FlexHousing System of Systems, which is a platform for energy management in buildings. It is based on the Arrowhead Framework and on the FlexOffer concept, and it allows both the fine-grained management of energy by a power user that is knowledgeable regarding Energy Markets, or by the professional services of an energy manager.

The paper provided insights regarding how the systems are implemented, and some results regarding experimental tests.

In the future, the FlexHousing System of Systems will be extended to different smart plugs that obey to different interaction patterns, and data regarding the energy saved in real-world deployments will be collected.

Acknowledgments

This work was partially supported by National Funds through FCT/MEC (Portuguese Foundation for Science and Technology) within the CISTER Research Unit (CEC/04234); also by FCT/MEC and the EU ECSEL JU under the H2020 Framework Programme, within project ECSEL/0004/2014, JU grant nr. 662189 (MANTIS), also by EU ECSEL JU under the H2020 Framework Programme, JU grant nr. 737459 (Productive4.0 project).

References

[1] EIA, *Annual Energy Review 2015*, http://www.eia.gov/totalenergy/data/annual/

[2] Tuan Anh Nguyen, and Marco Aiello, *Energy intelligent buildings based on user activity: A survey.* Energy and buildings 56 (2013): 244-257.

[3] Thibaut Le Guilly, et al. *ENCOURAGEing results on ICT for energy efficient buildings.* IEEE 21st International Conference on Emerging Technologies and Factory Automation (ETFA). 2016.

[4] Michele Albano, Luis Lino Ferreira, and Luís Miguel Pinho, *Convergence of Smart Grid ICT architectures for the last mile.* IEEE Transactions on Industrial Informatics 11.1: 187-197. 2015.

[5] Pervez Hameed Shaikh, et al. *A review on optimized control systems for building energy and comfort management of smart sustainable buildings.* Renewable and Sustainable Energy Reviews 34: 409-429. 2014

[6] Alessandra De Paola, et al. *Intelligent management systems for energy efficiency in buildings: A survey.* ACM Computing Surveys (CSUR) 47.1: 13. 2014

[7] Douglas Harris. *A guide to energy management in buildings*. Routledge, 2016.

[8] Delsing, Jerker, et al., *The Arrowhead Framework architecture.*, chaper 3 of *IoT Automation: Arrowhead Framework*. CRC Press, 2017.

[9] L. L. Ferreira, M. Albano, and J. Delsing, *QoS-as-a-Service in the Local Cloud.*, IEEE 21st International Conference on Emerging Technologies and Factory Automation (ETFA), 2016.

[10] M. Boehm, et al. *Data management in the MIRABEL smart grid system*. In: Proceedings of the 2012 Joint EDBT/ICDT Workshops. EDBT-ICDT'12, ACM, 95-102. 2012.

[11] Luis Lino Ferreira, et al. *Arrowhead compliant virtual market of energy*, Emerging Technology and Factory Automation (ETFA), IEEE, 2014.

[12] Bijay Neupane, Torben Bach Pedersen, and Bo Thiesson. *Evaluating the value of flexibility in energy regulation markets.* Proceedings of the 2015 ACM Sixth International Conference on Future Energy Systems. ACM, 2015.

[13] M. Albano, P. M. Barbosa, J. Silva, R. Duarte, L. L. Ferreira, and J. Delsing, *Quality of service on the Arrowhead Framework.* In 13th International Workshop on Factory Communication Systems (WFCS), IEEE , pp. 1-8, 2017.

[14] P. Saint-Andre, K. Smith, and R. Troncon, *XMPP: The Definitive Guide*, O'Reilly, 2009

[15] B. Neupane, L. Siksnys, T. Pedersen. *Generation and Evaluation of Flex-Offers from Flexible Electrical Devices.* Proceedings of the Eighth International Conference on Future Energy Systems. ACM, 2017.

[16] R. Rocha. *Reengineering and development of IoT Systems for Home Automation.* BEng Thesis. CISTER, 2017. http://hdl.handle.net/10400.22/10966.

[17] Piotr Domanski, and David Didion. *Computer modeling of the vapor compression cycle with constant flow area expansion device.* Final Report National Bureau of Standards, Washington, DC. National Engineering Lab. 1983.