Cassowary: Middleware Platform for Context-Aware Smart Buildings with Software-Defined Sensor Networks

Pradeeban Kathiravelu
INESC-ID Lisboa
Instituto Superior Técnico,
Universidade de Lisboa
Lisbon, Portugal
pradeeban.kathiravelu@tecnico.ulisboa.pt

Leila Sharifi
INESC-ID Lisboa
Instituto Superior Técnico,
Universidade de Lisboa
Lisbon, Portugal
leila.sharifi@tecnico.ulisboa.pt

Luís Veiga
INESC-ID Lisboa
Instituto Superior Técnico,
Universidade de Lisboa
Lisbon, Portugal
luis.veiga@inesc-id.pt

Abstract

Smart devices sense the environment through their sensors and leverage the contextual information derived from the sensor readings to satisfy system requirements such as energy and carbon efficiency and user preferences. Smart buildings compose of smart devices, and local sensors of the devices and device controllers in a coordinated network. Software-Defined Networking (SDN) offers a centralized view of the entire networking data plane elements to a logically centralized controller. While smart buildings and ubiquitous computing are heavily researched, later advancements in networking are not exploited in achieving tenant-aware smart buildings.

This paper describes the research for the design, prototype implementation, and preliminary assessments of $Cassowary$, a middleware platform for **C**ontext-**A**ware **S**mart Buildings with **So**ft**wa**re-Defined Sensor Netwo**r**ks. By extending SDN paradigm and leveraging the message oriented middleware protocols to seamlessly connect the smart devices of the buildings to the centralized SDN controller, $Cassowary$ enables context-aware Software-Defined Smart Buildings.

Keywords

Smart Buildings, Context-Aware Applications, Software-Defined Networking (SDN), Internet of Things (IoT), Message Oriented Middleware (MOM), Ubiquitous computing, In-Memory Data Grid (IMDG), Software-Defined Sensor Networks (SDSN)

1. INTRODUCTION

The word ubiquitous is derived from a Latin root and means existing everywhere. Aligned with this meaning ubiquitous society envisions a society in which services are accessible from anywhere, at anytime, by anyone and anything. It is beyond person to person or person to object computing. Nowadays, by the advent of Internet of Things(IoT), which unites everyday things in a huge, pervasive platform [1]. Smart buildings are a major use case scenario of ubiquitous computing, integrating IoT elements including sensors, computing elements, and controlling algorithms into the buildings.

IoT applications compose a highly condensed network of devices within a small enclosed area, with an exponential growth in the connected devices. More and more devices are connected to the network, which traditionally used to be independent dumb devices. SDN has been proposed as an efficient approach in building IoT applications and enabling a seamless execution of them [38]. With the clear separation, and a global view of the entire network readily available, SDN enables composition of complex networks in a scale that was previously harder to achieve.

Initially developed targeting the networking domain, the SDN paradigm of the separation of control from execution of the logic has been generalized and extended to computing systems, by implementing them as Software-Defined Systems (SDS), including Software-Defined Storage [37], Software-Defined Data Centers (SDDC) [13], and Software-Defined Infrastructure [17]. Consisting of a programmable interface, control plane of software-defined systems provides the logic for the data plane to execute.

While there are researches and preliminary developments on smart and energy efficient smart buildings [35] with IoT applications, the existing approaches fail short in addressing the requirements at system level as well as providing control to the tenants of the building. They are either generic solutions that are agnostic to the dynamic nature and location of tenants, or too specific and focused to a given problem. Hence they lack of configurability for the building spaces shared by multiple tenants with varying preferences. An extensive and complete approach in managing building complexes with different user or tenant preferences should be researched and implemented. Moreover, SDN should be extended and exploited for the sensor network in the smart buildings, in a loosely coupled manner.

In order to construct a context-aware smart building, it is essential that the devices of the building should be equipped with a local sensor and controller. Exploiting SDN, a Software-Defined Sensor Network can be built to centrally coordinate and orchestrate the devices attached to the sensors. While sensors and local controllers are used in pervasive computing for smart devices, efficient software-defined smart buildings are yet to be researched. While there are research efforts on smart buildings and energy efficiency, there is a gap in the research space that can be addressed by extending and deploying a centralized controller to orchestrate the devices in the buildings, which themselves are extended to operate in a sensor network.

In this paper we present $Cassowary$, a middleware platform for smart buildings, leveraging SDN to connect the IoT devices to a logically centralized controller and messaging broker. $Cassowary$

extends the SDN paradigm for smart buildings, while leveraging message oriented middleware (MOM) to propagate the environment details sensed by the sensors of the heterogeneous devices. It designs and builds a sensor network, connecting each of the devices in the smart building. As major use cases, $Cassowary$ first functions as a tenant-aware energy provisioning framework for smart building complexes, leveraging the software-defined building concepts. It also satisfies user requirements partially, in multi-tenanted buildings.

In the upcoming sections, we further analyze the proposed $Cassowary$ middleware architecture for smart buildings. We continue to discuss the preliminary background information on the Internet of Things and smart buildings, in Section 2. Section 3 discusses the design and architecture of the proposed $Cassowary$ middleware framework, while discussing how SDN can be leveraged to centrally coordinate the sensor network of the smart devices in the building complex. Section 4 elaborates the preliminary implementation details of $Cassowary$ with a feasibility assessment. Finally, Section 5 drives us to the conclusion of this research discussing its current state and the possible future enhancements.

2. BACKGROUND AND RELATED WORK

In this section we outline the concepts that we need to explain the details of $Cassowary$ in the rest of this paper.

Internet of things (IoT). IoT is realized as loosely coupled smart objects, connected to be composed into a network [20]. Inspired by the success of radio-frequency identification (RFID) [40] to offer a representation of the objects in an interconnected network, further developments exploit RFID to implement IoT or Internet of Objects [42]. Loose coupling among the multiple heterogeneous devices is essential for a successful installation or deployment of an IoT scenario. Fog computing is a paradigm introduced by Cisco Systems [5] to support wireless data transfer via IoT.

MOSDEN is an IoT middleware, specifically targeting the mobile devices that generally have limited availability of resources [27]. People centric sensing, which defines how the system is perceived by the occupants of the building, offers a cost effective solution for a multi-tenanted building [14]. An IoT middleware platform should be scalable and capable of effectively propagating the contextual information and policies.

In-Memory Data Grids (IMDGs) in Ubiquitous Computing. [28] IMDGs such as Hazelcast [16], Infinispan [23], and GridGain [36] offer a larger resource pool with a unified view of computing resources, while executing on top of a physically distributed computer cluster. Infinispan has also been extended to execute on top of the Android mobile devices in addition to the desktop computers. Computers and smart devices can create an in-memory data grid cluster to share computing resources among themselves, while distributing the execution and data uniformly. Hence smaller devices can initiate larger execution, while being connected to the cluster, as the computing resources of all the contributing nodes will be contributed to the execution cluster. $MEDIator$ [18] is a data sharing synchronization platform built on top of an Infinispan cluster to offer access to data stored in heterogeneous devices and data stores, adhering to the principles of pervasive computing.

Message Oriented Middleware. Message Oriented Middleware [9] platforms can be leveraged for connecting sensors and controllers of the smart devices through messaging. By publishing the contextual information as certain topics to a logically centralized broker, while listening to relevant messages from the other devices by subscribing to the respective topics, a device can communicate with other devices to form a loosely coupled sensor network without actually forming a static link [31]. Hence such a virtual network formed by messaging protocols remain fault-tolerant as devices can join and leave the network in a seamless manner, without propagating failures upstream.

Many message oriented middleware protocols such as the Advanced Message Queuing Protocol (AMQP) [39], MQTT [21], Simple / Streaming Text Oriented Message Protocol (STOMP), OpenWire, or XMPP [29] exists, along with broker implementations that listens to the messaging publishers while letting the subscribers to subscribe and listen to their interested sub set of information, shared originally by the publishers. There have been research efforts on implementing communication across the IPv6-enabled pervasive devices through message oriented middleware [33]. Simple/Streaming Text Oriented Message Protocol (STOMP) has been used for enabling pervasive communication across the devices [33]. Many messaging brokers such as Apache ActiveMQ [30] and Apache QPid provide open source implementation of the messaging protocols.

Software-Defined Sensor Networks (SDSN). Software-defined wireless sensor networks have been built in research [22, 43, 15], by leveraging the OpenFlow [24] SDN controller. However, they do not leverage the sensor networks in developing an extensible tenant-aware smart building. Sensor OpenFlow [22] discusses the challenges in adopting SDN into Wireless Sensor Networks (WSN) and elaborates how the SDSN architecture would make the wireless networks more generic and less application-specific.

Software-Defined Buildings (SDB). Software-Defined Buildings is a term coined by the University of California at Berkeley [10], which intends to increase the programmability and reusability of the buildings' devices, with improved energy efficiency. Software-defined buildings define a software platform that increases the programmability and compatibility of variable appliances inside a building or a multi-building campus, which function as a building operating system (BOS) atop which the other firmware applications of the other appliances execute [10].

Building Application Stack (BAS) provides an API for the building applications to build on top of a modular architecture enabled by the building controller software platform, increasing the reusability of the building applications [34]. By introducing a building operating system and an API to build atop, Software-Defined Buildings operate as an effective development paradigm for smart buildings. Message oriented middleware platforms can be exploited to build generic SDSN systems for smart buildings, which can be extended by receiving user intents and system policies as input, and consider them in decision making.

Context-Aware Applications. Context-aware applications sense the environmental changes and act accordingly, such as providing heating, ventilating, and air-conditioning (HVAC) [12] in a power-efficient manner. In addition to such system-wide parameters, ubiquitous computing requires the buildings to respond to the user preferences in a seamless manner, to ensure user comfort with less power wastage [6]. Buildings are developed to be more power efficient, consisting of context-aware [4] smart devices.

Related Work. *Aware Home* [19] deploys a smart floor with a few tiles that are strategically located to collect footstep information of the occupants. The footstep information is further used in many use cases such as finding lost objects and supporting the elderly. Deploying autonomous intelligent agents for pervasive computing and smart buildings has been researched for its technical and social challenges [8]. When a building is occupied by multiple tenants, all the constraints and preferences of every tenant may not be satisfied. However, an efficient partial utility based approach will offer higher overall quality of service (QoS) to the tenants. There have been research efforts in reducing energy wastage in buildings shared by multiple tenants [3].

While In-Memory Data Grids are often used in research for distributed execution, their contribution for smart buildings needs further research. SDN and distributed shared memory frameworks such as Infinispan should be exploited to build a comprehensive platform for tenant-aware smart buildings. Leveraging message oriented middleware as the northbound user-facing API of SDN controllers is expected to offer more use cases for SDN, along with more loose coupling. Extending them to provide a Software-Defined Sensor Network for smart buildings is an interesting research question.

3. SOLUTION ARCHITECTURE

$Cassowary$ has been designed with a layered architecture consisting of two core layers. In the bottom is, network layer, which consists of the SDN controller and data plane. Above the network layer is, appliance layer, which is responsible for the integration and execution of the smart appliances. These layers are mostly logical views, than physical, as a device can often be represented in both layers. For example, a sensor, as a part of the device, is physically part of the appliance layer. However, it also is part of the sensor network that is composed through messaging, and hence can logically be considered a part of the network layer.

3.1 Deployment Architecture

The deployment consists of computing devices that contribute to a $Cassowary$ IMDG cluster. $Cassowary$ configurations such as profiles and policies are stored in Configuration Data Store, an in-memory data store. The cluster executes instances of a $Parser$ and a $Builder$. The $Parser$ parses the building/system profile, user/tenant profiles, and policies. The building/system profile is a single configuration file containing the properties specific to the building or the system, such as power consumption, HVAC, and lighting policies. User/tenant profiles are specific to each user/tenant in the building, mostly defining the personal preferences of the occupants of the building. Policies define how the $Builder$ computes the values of the different parameters of the deployment based on the profiles.

The computing devices are hosts that are connected to the SDN switches in the data plane. The SDN switches are coordinated by the SDN controller deployment. Control plane further consists of the message broker cluster and Event Listener cluster. SDN controller and Event Listener are loosely connected to the message broker through publish-subscribe (pub/sub) pattern [11] based on the messaging protocols such as AMQP. Event Listener further subscribes to the information published by the sensors of each of the smart appliance. Similarly, the controller in each of the devices subscribes to the relevant topics through the Event Listener. This indirectly lets the SDN controller to orchestrate the sensors and smart appliances in a network. Here, each sensor usually has a limited functionality, where the information sensed by each of the sensor is propagated to the other devices through the Event Listener to the appliances subscribing to the relevant topic. The topic can be anything ranging from temperature, noise level, visible light, wind speed, and relative humidity.

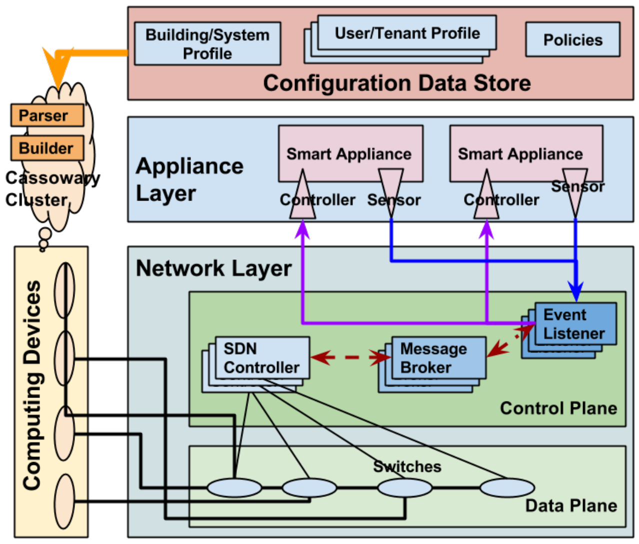

Figure 1: $Cassowary$ **Deployment Architecture**

3.2 Appliance Layer

Sharing data and execution flows across multiple smart devices, as proposed in our previous work [18] has been extended and used for the devices confined in a building. Multiple nodes including desktops, laptops, and smart Android devices running Infinispan [23] are leveraged and configured to create a smart cluster, known as a $Cassowary$ cluster. The $Cassowary$ cluster consists of the profiles and policies. It further executes the computations required to dynamically alter the system based on the context data, as well as the occupants in the building at any time.

A device controller of a smart equipment is a small firmware attached to the equipment that can control the equipment with limited storage and processing capacity. The equipments are connected to the computer cluster via their controllers as thin clients of the cluster. The cluster is made of computers and smart mobile devices such as mobile phones and tabs, which execute an in-memory data grid to provide a resource pool to the control panel. The computers in the cluster also function as surrogates to the devices in the proximity, while the cluster further provides processing and storage capacity to the controllers. The cluster is composed of computers and android devices, which are running Infinispan.

Features available to a few devices such as motion, temperature, or humidity detection can be shared across the cluster, which can be used by the other devices. For example, a motion detector connected to the lighting system of the building can be leveraged by the heating of the building, by utilizing the shared information in the cluster. Extending the software-defined sensor network, even the devices that do not have a sensor on their own can leverage the context information received from the other tenants and shared with the other interested devices by subscribing.

The cluster is configured across multiple buildings in the building complex. Named cache instances are used to represent each building and building profiles. These cache instances are independent from other cache instances, and hence provide a multi-tenanted cluster to represent the multiple buildings. Moreover, data related to the appliances in each building is stored in the instances located inside each building, leveraging the near caching provided by the underlying in-memory data grid platforms. This further reduces the communication overhead that may occur due to unnecessary communication between the buildings.

The buildings are occupied by tenants, which may also freely roam across different buildings. Tenant profiles are also represented, and building profiles as well as the profiles of the tenants occupying the building are considered in controlling the appliances in the building, or an enclosure in a smaller granularity, such as a room. Executions are passed on to the larger computing devices, as the computational resources are limited in the smart appliances. They merely operate as the sender and receiver of the context information.

Estimating the location of the occupants in a smart building has been previously researched, and there are many solutions based on ubiquitous computing. For the sake of simplicity, proximity of the tenants is estimated by their mobile devices (mostly mobile phones) in $Cassowary$, assuming a mobile device in most cases remain close to their owner. That means, the relevant RFID of the mobile devices can be used to identify those who are in the building, with a considerably lower rate of error, which may occur when the mobile devices are misplaced or kept away from the owner. Based on the policies, ideal values will be calculated in a distributed manner in the $Cassowary$ cluster. In many cases, the ideal value will be a compromise among the tenant profiles. The tenant proximity plays a significant role in deciding the impact of the specific tenant in an enclosed area such as a room. For example, the expected temperature in a room may be calculated as a weighted average of the preferred temperatures of the occupants, based on their location and distance from the air conditioner. A time average is considered to avoid jitter effect caused by walking tenants. Hence the tenants who remain in a single place will influence the estimates for the closest devices.

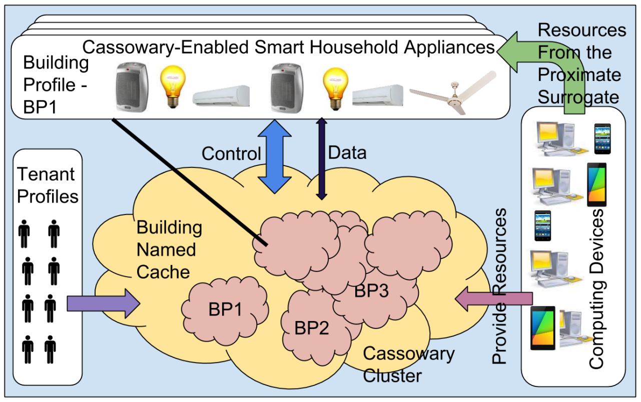

Figure 2: Appliance Layer Higher Level View

3.3 Network Layer

Figure 3 shows the network layer of a $Cassowary$ deployment. The core of the deployment is an SDN controller. While any SDN controller can work with the architecture, OpenDaylight has been chosen as it offers a modular architecture based on Apache Karaf [26] OSGi run time. The Model-Driven Service Abstraction Layer (MD-SAL) of OpenDaylight [25] further enables easy extension of the controller with more bindings. OpenDaylight controller by default consists of REST northbound bindings, through which the data tree, remote procedure calls (RPCs) [32], and notifications are exposed. Messaging4Transport is an OpenDaylight bundle that exposes the OpenDaylight data tree as messages in a messaging protocol. While AMQP has been used as the default messaging protocol, other messaging protocols can be implemented following the same design.

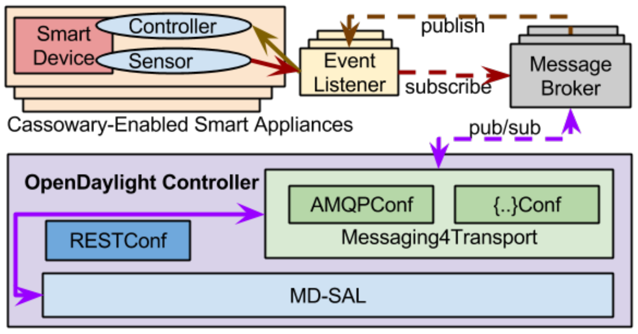

Figure 3: Network Layer Higher Level View

The data tree of OpenDaylight MD-SAL is leveraged to store the dynamic context data sensed by the sensors of the heterogeneous appliances. Each of the appliances subscribes to the relevant topics, while publishing the relevant information sensed by their sensor, to be written to the data tree. The controller is deployed physically distributed in a cluster, to avoid single point of failure.

The logical separation of appliance layer from network layer also enables a less complicate execution of $Cassowary$. The computations to decide the ideal values for the device parameters from the context information is executed inside the data grid which is deployed on top of the computing nodes. Each device is mapped as an agent in the calculations inside the data grid, where an agent executes independently as a cloudlet. Hence, each virtual agent represents a physical device in the building.

Integrating the device controller and sensors to the SDN controller enables quicker response to the dynamic changes in the context information. The SDN controller is often a super-computer or a cluster of high-end servers, where the in-memory data grid is composed of the resources contributed by utility computers. Hence, context-awareness is ensured by storing the dynamic data inside the data tree of the controller, instead of storing it along side the profile and policy information inside the data grid. On the other hand, more static information such as the tenant and system profiles, and policies are stored in the data grid, as they are mostly static information.

4. IMPLEMENTATION

A prototype of $Cassowary$ has been implemented as a middleware platform for software-defined smart buildings. OpenDaylight Beryllium development branch has been chosen as the SDN controller, which is the core of the $Cassowary$ deployment. AMQP is used as the messaging protocol, with Apache ActiveMQ-5.12.0 used as the broker. The messaging oriented middleware bindings for SDN has been developed [1] as an AMQP northbound binding for OpenDaylight controller.

Clustered deployments of ActiveMQ and OpenDaylight are used, in order to minimize the single point of failures while ensuring fault-tolerance. Infinispan-7.2.4.Final has been used as the in-memory data grid, deployed on top of multiple computing nodes. As a

[1] **`https://wiki.opendaylight.org/view/Messaging4Transport:Main`**

prototype implementation for early assessments, the smart buildings were simulated by extending CloudSim [7] cloud simulator by depicting the appliances by extending Cloudlets. Oracle Java 1.7.0_55 has been used as the development language of $Cassowary$.

4.1 Cassowary Minimalized Approach

Adopting household appliances as smart devices is a major challenge faced by ubiquitous computing attempts and approaches. There have been efforts on formalizing the calculation of the potential values for different parameters in a multi-tenanted building. Ergonomics of the thermal environment - Analytical determination and interpretation of thermal comfort using calculation of the Predicted Mean Vote (PMV) and Predicted Percentage Dissatisfied (PPD) indices and local thermal comfort criteria [2] looks into the HVAC aspects. $Cassowary$ simplifies, generalizes, and applies the formulae for different environmental properties.

$Cassowary$ deployment is intentionally kept minimal to ensure that the system can be deployed with the commonly available components and utility devices. The environment sensors in day to day life such as the ones in the air conditioners are exploited as the sensors. Sensors are in fact made optional in $Cassowary$ as the devices without sensors leverage the context information from the other sensors. The device controller alters the state of the appliance based on the context information. This can be achieved by extending or modifying the existing controllers/switches of the devices. An overlay network of the sensors from the devices is composed as a software-defined sensor network, which is further exploited for context-aware smart buildings.

Tenant Proximity. Distance of the tenant is an important measure in ensuring quality of service to the individual tenant. The distance is estimated by publishing messages to the controller, which will be propagated to the subscribed Event Listener instance connected to the devices. The distance of the tenants also is used to find the number of tenants in any enclosure, as the tenants farther than a specified distance can be ignored or neglected in the calculations for the specific device, as the device's impact will be negligible for the tenant, and the tenant preferences will be negligible to the device controller.

d_{min} is defined as the distance to the nearest tenant for the given enclosure. d_{min} is used to find whether the area can be considered to be empty or occupied. Pervasive displays [41] in the building can be dimmed or even hibernated when there is no tenant in the close proximity, to save power while avoiding unnecessary light and acoustic pollution.

Temperature Control. The temperature control of any room is associated with the occupied tenants in proximity to the heating or air conditioning (HVAC) system and their preferred or comfort level in temperature, as shown by Equation 1. This simplified equation is set as temperature policy in the policy configurations. More intuitive equations can be set, replacing the default policies.

$$T = \frac{\sum_{i=1}^{n}(\frac{T_i}{x_i})}{\sum_{i=1}^{n}(\frac{1}{x_i})} \tag{1}$$

Here,
T - Temperature chosen by the air conditioning system.
n - Number of tenants considered to be in close proximity.
T_i - Preferred temperature from the tenant profile of tenant i.
x_i - Distance of tenant i to the HVAC system.

Illumination Control. Light sensors may be attached to a few light sources, while other light sources such as regular light bulbs may just consume the information sensed and shared by the other light sensors. When the natural light is greater than a specific value, the light sources are turned off. Similarly when there is no tenants in a specific enclosed area, the area is dimmed. The brightness of the light sources in the building is defined by a function of the distance to the nearest tenant and the natural light such as the day light inside the enclosure, as shown by Equation 2.

$$L = F(d_{min}, L_s) \tag{2}$$

Here,
L - Light intensity to ensure.
L_s - Sensed external or natural light.

If the light source does not have a local light sensor, it will use an estimate from the nearest light sources, with the statistical information. Both the distance to the occupants and sensed external illumination are inversely popular to the illumination expected from the building's light sources. Hence, the Equation 2 can further be elaborated by Equation 3.

$$L = k * \frac{1}{d_{min}} * \frac{1}{L_s} \tag{3}$$

Here k is often a constant value defined by the system policies or derived from the historical or statistical information.

4.2 Feasibility Assessment

We evaluated $Cassowary$ for its functionalities, while estimating its performance using the middleware framework along with simulations for the building and sensor networks. Multiple scenarios have been modeled and verified for a simplified scenario. Here we will discuss a few simplified application scenarios of $Cassowary$, elaborating how quality of service (QoS) is guaranteed.

Table 1: Quality of Service Assessment of $Cassowary$

Feature	HVAC	Light Sources	TV/Displays
Energy Efficiency	✓	✓	✓
Tenant Comfort	✓	✓	N/A
Acoustic/Light Pollution	N/A	✓	✓

Table 1 indicates the overall quality of service achievements for each of the devices considered and properties. As assessed for HVAC, light sources, and TV or displays, $Cassowary$ covers the energy efficiency, tenant comfort, and prevention of acoustic and light pollution, wherever applicable.

$Cassowary$ ensures energy efficiency by automating the HVAC and light sources in an attempt to minimize the energy wastage, while also not compromising the tenant preferences as much as possible. Here, properties such as acoustic pollution are not applicable for HVAC, where tenant comfort is usually not associated with the displays. However, these properties can be changed from the properties files, or based on the statistical information derived from the $Cassowary$ cluster.

5. CONCLUSION AND FUTURE WORK

$Cassowary$ leverages OpenDaylight SDN controller and integrates the controller with message oriented middleware, in order to publish the sensed environment changes and subscribe to the relevant topics. By having a sensor and controller local to each device, $Cassowary$ offers a context-aware sensor network for smart buildings. By storing the user preferences in a tenant-aware manner in a distributed cache, $Cassowary$ functions as a middleware platform for multi-tenanted energy provisioning in the buildings.

Preliminary assessments on the $Cassowary$ prototype implementations and models indicate a potential for achieving software-defined smart buildings by leveraging the $Cassowary$ solution architecture. A full-fledged deployment over a real building complex is a future work. Upon building on a building with a sensor network, $Cassowary$ is expected to offer energy and carbon efficiency, while adhering to the principles of ubiquitous computing. Currently $Cassowary$ focuses on mostly a single building, where it can further be extended to coordinate building complexes composed of multiple buildings, to enable energy efficient building complexes and cities. This will also enable sharing of computing and energy resources across the buildings efficiently.

Acknowledgements: Special thanks to Ed Warnicke, Principal Engineer at Cisco and TSC Member of OpenDaylight for his continuous guidance in working with OpenDaylight MD-SAL. This work was supported by national funds through Fundação para a Ciência e a Tecnologia with references UID/CEC/50021/2013, PTDC/EEI-SCR/6945/2014.

6. REFERENCES

[1] Ubiquitous network society(itu), http://www.itu.int/world2006/forum/ubiquitous_network_society.html.

[2] Anonymus AC08024865. *Ergonomics of the thermal environment-Analytical determination and interpretation of thermal comfort using calculation of the PMV and PPD indices and local thermal comfort criteria*. ISO, 2005.

[3] CJ Axon, SJ Bright, Tim J Dixon, KB Janda, and M Kolokotroni. Building communities: reducing energy use in tenanted commercial property. *Building Research & Information*, 40(4):461–472, 2012.

[4] Matthias Baldauf, Schahram Dustdar, and Florian Rosenberg. A survey on context-aware systems. *International Journal of Ad Hoc and Ubiquitous Computing*, 2(4):263–277, 2007.

[5] Flavio Bonomi, Rodolfo Milito, Jiang Zhu, and Sateesh Addepalli. Fog computing and its role in the internet of things. In *Proceedings of the first edition of the MCC workshop on Mobile cloud computing*, pages 13–16. ACM, 2012.

[6] Bill Bordass, Ken Bromley, and Adrian Leaman. User and occupant controls in office buildings. In *International conference on building design, technology and occupant well-being in temperate climates, Brussels, Belgium*, pages 12–5, 1993.

[7] Rodrigo N Calheiros, Rajiv Ranjan, Anton Beloglazov, César AF De Rose, and Rajkumar Buyya. Cloudsim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms. *Software: Practice and Experience*, 41(1):23–50, 2011.

[8] Vic Callaghan, Graham Clarke, and Jeannette Chin. Some socio-technical aspects of intelligent buildings and pervasive computing research. *Intelligent Buildings International*, 1(1):56–74, 2009.

[9] Edward Curry. Message-oriented middleware. *Middleware for communications*, pages 1–28, 2004.

[10] Stephen Dawson-Haggerty, Jorge Ortiz, Jason Trager, David Culler, and Randy H Katz. Energy savings and the "software-defined" building. *Design & Test of Computers, IEEE*, 29(4):56–57, 2012.

[11] Patrick Th Eugster, Pascal A Felber, Rachid Guerraoui, and Anne-Marie Kermarrec. The many faces of publish/subscribe. *ACM Computing Surveys (CSUR)*, 35(2):114–131, 2003.

[12] Kwong Fai Fong, Victor Ian Hanby, and Tin-Tai Chow. Hvac system optimization for energy management by evolutionary programming. *Energy and Buildings*, 38(3):220–231, 2006.

[13] Aaron Gember, Prathmesh Prabhu, Zainab Ghadiyali, and Aditya Akella. Toward software-defined middlebox networking. In *Proceedings of the 11th ACM Workshop on Hot Topics in Networks*, pages 7–12. ACM, 2012.

[14] Jayavardhana Gubbi, Rajkumar Buyya, Slaven Marusic, and Marimuthu Palaniswami. Internet of things (iot): A vision, architectural elements, and future directions. *Future Generation Computer Systems*, 29(7):1645–1660, 2013.

[15] Zhi-jie Han and Wanli Ren. A novel wireless sensor networks structure based on the sdn. *International Journal of Distributed Sensor Networks*, 2014, 2014.

[16] Mat Johns. *Getting Started with Hazelcast*. Packt Publishing Ltd, 2013.

[17] G Kandiraju, Hubertus Franke, MD Williams, Malgorzata Steinder, and SM Black. Software defined infrastructures. *IBM Journal of Research and Development*, 58(2/3):2–1, 2014.

[18] Pradeeban Kathiravelu and Ashish Sharma. Mediator: A data sharing synchronization platform for heterogeneous medical image archives. In *Workshop on Connected Health at Big Data Era (BigCHat'15) , co-located with 21 st ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD 2015)*. ACM, 2015.

[19] Cory D Kidd, Robert Orr, Gregory D Abowd, Christopher G Atkeson, Irfan A Essa, Blair MacIntyre, Elizabeth Mynatt, Thad E Starner, and Wendy Newstetter. The aware home: A living laboratory for ubiquitous computing research. In *Cooperative buildings. Integrating information, organizations, and architecture*, pages 191–198. Springer, 1999.

[20] Gerd Kortuem, Fahim Kawsar, Daniel Fitton, and Vasughi Sundramoorthy. Smart objects as building blocks for the internet of things. *Internet Computing, IEEE*, 14(1):44–51, 2010.

[21] Dave Locke. Mq telemetry transport (mqtt) v3. 1 protocol specification. *IBM developerWorks Technical Library], available at http://www.ibm. com/developerworks/webservices/library/ws-mqtt/index. html*, 2010.

[22] Tie Luo, Hwee-Pink Tan, and Tony QS Quek. Sensor openflow: Enabling software-defined wireless sensor networks. *Communications Letters, IEEE*, 16(11):1896–1899, 2012.

[23] Francesco Marchioni. *Infinispan Data Grid Platform*. Packt Publishing Ltd, 2012.

[24] Nick McKeown, Tom Anderson, Hari Balakrishnan, Guru Parulkar, Larry Peterson, Jennifer Rexford, Scott Shenker, and Jonathan Turner. Openflow: enabling innovation in campus networks. *ACM SIGCOMM Computer Communication Review*, 38(2):69–74, 2008.

[25] Jan Medved, Robert Varga, Anton Tkacik, and Ken Gray. Opendaylight: Towards a model-driven sdn controller architecture. In *2014 IEEE 15th International Symposium on*, pages 1–6. IEEE, 2014.

[26] Achim Nierbeck, Jamie Goodyear, Johan Edstrom, and Heath Kesler. *Apache Karaf Cookbook*. Packt Publishing Ltd, 2014.

[27] Charith Perera, Prem Prakash Jayaraman, Arkady Zaslavsky, Dimitrios Georgakopoulos, and Peter Christen. Mosden: An internet of things middleware for resource constrained mobile devices. In *System Sciences (HICSS), 2014 47th Hawaii International Conference on*, pages 1053–1062. IEEE, 2014.

[28] Bhaskar Prasad Rimal, Eunmi Choi, and Ian Lumb. A taxonomy and survey of cloud computing systems. In *INC, IMS and IDC, 2009. NCM'09. Fifth International Joint Conference on*, pages 44–51. Ieee, 2009.

[29] Peter Saint-Andre. Extensible messaging and presence protocol (xmpp): Core. 2011.

[30] Bruce Snyder, Dejan Bosnanac, and Rob Davies. *ActiveMQ in action*. Manning, 2011.

[31] Eduardo Souto, Germano Guimarães, Glauco Vasconcelos, Mardoqueu Vieira, Nelson Rosa, and Carlos Ferraz. A message-oriented middleware for sensor networks. In *Proceedings of the 2nd workshop on Middleware for pervasive and ad-hoc computing*, pages 127–134. ACM, 2004.

[32] Raj Srinivasan. Rpc: Remote procedure call protocol specification version 2. 1995.

[33] Tomasz Szydło, Paweł Suder, and Jakub Bibro. Message-oriented communication for ipv6-enabled pervasive devices. *Computer Science*, 14(4)):667–667, 2013.

[34] Jay Taneja, Andrew Krioukov, Stephen Dawson-Haggerty, and David Culler. Enabling advanced environmental conditioning with a building application stack. In *Green Computing Conference (IGCC), 2013 International*, pages 1–10. IEEE, 2013.

[35] Paulo Filipe de Almeida Ferreira Tavares and António Manuel de Oliveira Gomes Martins. Energy efficient building design using sensitivity analysis—a case study. *Energy and Buildings*, 39(1):23–31, 2007.

[36] GridGain Team. Gridgain: In-memory computing platform. 2007.

[37] Eno Thereska, Hitesh Ballani, Greg O'Shea, Thomas Karagiannis, Antony Rowstron, Tom Talpey, Richard Black, and Timothy Zhu. Ioflow: A software-defined storage architecture. In *Proceedings of the Twenty-Fourth ACM Symposium on Operating Systems Principles*, pages 182–196. ACM, 2013.

[38] Angel Leonardo Valdivieso Caraguay, Alberto Benito Peral, Lorena Isabel Barona Lopez, and Luis Javier García Villalba. Sdn: Evolution and opportunities in the development iot applications. *International Journal of Distributed Sensor Networks*, 2014, 2014.

[39] Steve Vinoski. Advanced message queuing protocol. *IEEE Internet Computing*, (6):87–89, 2006.

[40] Roy Want. An introduction to rfid technology. *Pervasive Computing, IEEE*, 5(1):25–33, 2006.

[41] Mark Weiser. The computer for the 21st century. *Scientific american*, 265(3):94–104, 1991.

[42] Evan Welbourne, Leilani Battle, Greg Cole, Kyle Gould, Kyle Rector, Samuel Raymer, Magdalena Balazinska, and Gaetano Borriello. Building the internet of things using rfid: the rfid ecosystem experience. *Internet Computing, IEEE*, 13(3):48–55, 2009.

[43] Deze Zeng, Toshimasa Miyazaki, Song Guo, Tsuneo Tsukahara, Junji Kitamichi, and Teruaki Hayashi. Evolution of software-defined sensor networks. In *Mobile Ad-hoc and Sensor Networks (MSN), 2013 IEEE Ninth International Conference on*, pages 410–413. IEEE, 2013.