Managing QoS in Smart Buildings through Software Defined Network and Usage Control

Fabio Martinelli, Christina Michailidou, Paolo Mori, Andrea Saracino
Istituto di Informatica e Telematica, Consiglio Nazionale delle Ricerche, Pisa, Italy,
name.surname@iit.cnr.it

***Abstract*—This work presents a framework for applying QoS in a network of a Smart Building environment, exploiting Software Defined Networks (SDN) and Usage Control (UCON) policy enforcement. The proposed framework will be presented in a plausible use case of a Smart Building where the available Internet connection provided by an Internet Service Provider will be distributed both to tenants and the devices responsible for the management and the safety of the building, taking into account different levels of QoS. Network traffic and bandwidth are thus managed by dynamically exploiting Usage Control framework to enforce a set of management, security and safety policies, aimed at ensuring the appropriate QoS for the provided services according to both tenants Service Level Agreements (SLAs) and current context. A set of performance experiments is reported to demonstrate the viability of the proposed approach.**

I. Introduction

Smart Buildings are an upcoming application of pervasive and mobile computing [17]. They consist of a number of sensors and actuators all connected to the same network, exchanging data and information continuously which are used for managing, in a smart way, the common areas and the apartments of a condominium. Light control systems, temperature and humidity sensors, security and energy management are some of the services which a smart building provides to its tenants to simplify their daily activities. Typically, SmartBuildings already provide Internet access as a service to house tenants, managing with the same connection smart devices in both the house and the common areas of the building. In such a way, Smart Buildings offer a "plug and play "solution for tenants.

For ensuring both a high level of user's experience and an accurate and fast transmission of the sensors' data, the network of a smart building should be highly adjustable and fast responding to any changes observed either in the demand of the resources or based on the current context. For example, in case of an emergency such as a fire in the building, it is more than necessary for the sensors to be able to communicate their data with a maximum speed to the Smart Building Manager and eventually to emergency services (e.g. firefighters). This can be realized by temporarily minimizing the bandwidth allocated to the tenants and giving the highest priority to all traffic coming from the devices responsible for the safety of the building, such as the temperature sensors, the surveillance cameras, the moving sensors, which should be able to transmit and receive data with a determined Quality of Service (QoS). In normal situation instead, most of the traffic should be dedicated to ensure Internet access to the tenants, providing a specific QoS agreed beforehand and that must be ensured independently from the network workload.

To provide a fully adaptive and configurable network, Software-Defined networks (SDN) [12] came to give a solution. By decoupling the control and the data plane SDN provide the required flexibility, being able to generate ad-hoc software network channels (*flows*), with configurable bandwidth, via a programmable interface. Thanks to SDNs is thus possible to easily define semantic subnetworks dedicated to specific traffic types, assigning them high or low bandwidth according to the needs of the specific context. However, even if SDNs have basic mechanisms for defining access policies, the dynamicity and complexity required by a Smart Building environment requires automatic mechanisms for issuing network channel assignment and revocation, able to adapt to the context changes.

In this paper we propose an SDN-based framework responsible for managing the QoS in the network of a Smart Building, empowered through the Usage Control (UCON) paradigm to enable dynamic assignment and revocation of flows based on the current context. The UCON paradigm [15] is an improvement of XACML-based access control which enables the definition and enforcement of complex policies on resources usage, also considering conditions with attributes whose value changes over time. In the proposed solution we assume that the total bandwidth which is available in a smart building can be distributed both to the tenants and to the interconnected devices based on a set of security policies. Essentialy, the different QoS levels will be applied through the enforcement of these policies based on the general state of the building and the context of the access to the network. This paper will at first describe the Smart Building environment that we are considering, presenting the traffic types and semantic networks considered. Then, a set of policies is presented together with the description of the proposed system architecture and workflows. Hence, a set of performance experiments is reported, showing a minimal overhead that does not affect the user experience, nor cause criticalities in case of emergency.

The rest of the paper is organized as follows. In Section II, a background is presented on the topics of Software-Defined networks, Usage control model and smart buildings. Section III is devoted to the presentation of the proposed model and the established communication among UCON and the SDN controller. The implementation of the model alongside with some policy examples are given in Section IV. In Section V we present experimental results. Finally, Section VI reports the related work and Section VII concludes briefly.

II. Background

A. Software-Defined Networks

Software-Defined networking is a cutting edge technology in the network field and it is gaining more and more attention both from the industry and the academia. Leading information technology companies such as Google [6] and Microsoft have already adopted this new network paradigm. The main characteristic of SDN is the separation of network's control and data plane. In the traditional networks, these two layers are connected to each other, which makes the management of the system a tedious and complex task [4]. The SDN architecture on the other hand, with the decoupling of the control logic from the forward devices, allows the system administrators to dynamically control the network's operations, without having to configure each one of the devices individually. Thus, the management of the network and the policy enforcement are simplified [11], leading to more efficient and robust systems.

The role of each SDN level regarding the functionality of the network is briefly analyzed below.

Data Plane: is the layer which includes the forward devices of the network. These devices can be either hardware-based such as switches and routers or software-based such as virtual switches. The main responsibility of the devices is to follow the instructions coming from the controller and handle the packets according to them. Typical actions that the devices can perform on a packet are forward or drop the packet, send it back to the controller, or modify fields of its header.

Control Plane: is the layer which includes the controller of the network. Controller is often characterized as the intelligence of the network, since it is the one that has a holistic view of the components and the connections and is responsible for traffic monitoring, routing decisions and in general for the configuration of the network. A list of some of the existing controllers and their characteristics can be found in [12].

Application Plane: includes a range of applications both for the management of the network and for security

Northbound Interfaces: form the communication channel between the controller and the applications of the SDN network which run on top of the controller. So far, there is not a standard widely used for the northbound interfaces. In most cases, controllers also propose and provide documentation on their own northbound APIs.

Southbound Interfaces: form the communication channel between the controller and the forwarding devices. The most widely adopted southbound API for SDN is OpenFlow [14].

B. Usage Control

The Usage Control (UCON) model [16] extends traditional access control models introducing *mutable attributes* and new decision factors besides *authorizations*, *obligations* and *conditions*. Mutable attributes represent features of subjects, resources, and environment that change their values over time. [15]. Since mutable attributes change their values during the usage of an object, the usage control model allows to define policies which are evaluated before *(pre-decision)* and continuously during the access to the object (*ongoing-decision*).

This paper takes into account Usage Control systems based on the XACML reference architecture, with particular reference to the one we presented in [5], [13]. In the XACML reference architecture, the Policy Enforcement Points (PEPs) embedded in the controlled system intercept the execution of security relevant operations, and they invoke the Context Handler (CH), which is the frontend of the Usage Control system. The *Policy Information Points (PIPs)* are the components invoked by the CH to retrieve the attributes required by the Policy Decision Point (PDP) for the execution of the decision process. Attributes are managed by Attribute Managers (AMs), which provide the interfaces to retrieve their current values. Each specific scenario where the Usage Control system is exploited requires its own set of AMs to manage the attributes required for the policy evaluation. Hence, PIPs are properly configured in order to be able to query the specific AMs adopted in the scenario of interest for retrieving and updating attributes. The phases of the Usage Control decision process are regulated by the interactions between the PEP and the Usage Control systems as follows (derived from [19]):

TryAccess: is the *pre-decision* phase, which begins when the Tryaccess message is sent by the PEP to the Usage Control system because a subject requests to execute the access. The TryAccess phase finishes when the Usage Control system sends the response to the PEP. The possible responses are: PERMIT, to allow the access, or DENY;

StartAccess: is the first part of the *ongoing-decision* phase, which begins when the StartAccess message is sent by the PEP to the Usage Control system because the access has just started, and finishes when the policy has been evaluated and the response has been sent back to the PEP;

RevokeAccess and *EndAccess*: is the usage termination happening when the PEP releases the resource since not needed anymore (*EndAccess*), or forcefully terminating the session when an attribute change cause a policy mismatch (*RevokeAccess*).

III. Smart Building Control Framework

The system we are considering is a Smart Building, owned and managed by a *Smart Building Manager* who rents or sells houses to *House Tenants*, already furnished, with smart devices connected to the Internet and/or amongst them. Each house comes with an Internet connection, accessible via WiFi or LAN cables. The house Internet connection service level (bandwidth) is customizable and is agreed between the Smart Building Manager and the Tenant (for the rest of this paper *House Tenant* will be referred to simply as Tenant), through the establishment of a Service Level Agreement. Hence, the Smart Building Manager is the only actual subscriber for an Internet connection to an *Internet Service Provider*. Having at his disposal the provided bandwidth from the ISP he then redistributes the bandwidth among the Tenants, based on the subscribed SLAs, and exploits part of the bandwidth to provide Internet connection to the smart devices which manage the common services in the Smart Building (e.g. elevators, video surveillance, fire and gas alarms). Finally, we suppose that in the houses some devices and services are connected to a separate network for providing security and safety services (e.g. utilities management, video surveillance and intrusion management), which are managed by the Smart Building Manager. A logical view of this system is shown in Figure1.

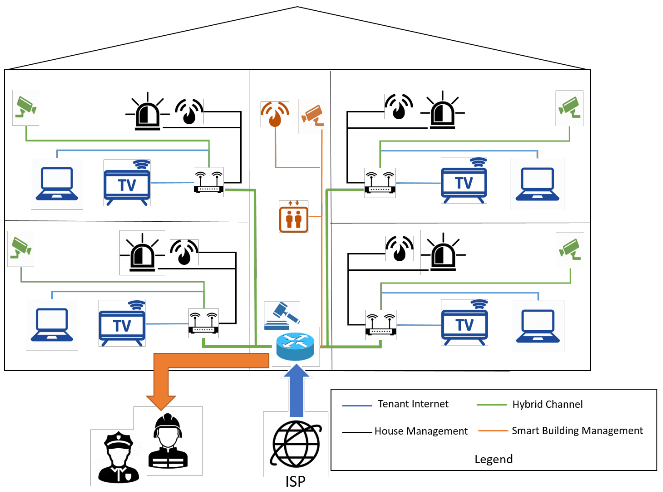

Fig. 1: High level architecture of the smart building

The proposed architecture considers a Smart Building where an SDN system is managing 9 types of traffic, by forwarding on demand the exchanging packets through the appropriate channels dedicated to manage both the Internet connections of the Tenants and the smart devices used for the management of the building. In particular, the considered channels, associated with the aforementioned 9 traffic types are:

Tenant Internet Traffic: (i) Premium High Speed Streaming, (ii) Multimedia Streaming, (iii) Browsing and Emails;

House Management Traffic: (i) Heating and Air Conditioning, (ii) Video Surveillance, (iii) Utilities and Consumptions

Building Management Traffic: (i) Alarm and Fire management, (ii) Lights and Elevators, (iii) Video Surveillance

As shown, these 9 traffic types are divided in three semantic groups to be used for a better management and for more expressive policies definition. Each semantic group might have one or more SDN channels dedicated to it. For example, each house is reached by at least an SDN channel for providing Internet access to the Tenant and reaching home devices. As shown, we consider three types of Internet traffic (blue connections in Figure 1) with different QoS, namely a (i) Premium High Speed Streaming, which ensures a very high QoS by allocating a large bandwidth for UHD multimedia streaming (4K quality), (ii) a medium speed connection for real time web applications, music and video streaming (up to Full-HD) and (iii) a low speed basic connection for web browsing and email access. The *availability* of these networks are regulated by the SLA between the Tenant and the Smart Building Manager. The second traffic group is dedicated to house management, connecting to the Smart Building Manager devices such as heating and air conditioning system, and sensors such as smoke detectors, or surveillance cameras. In particular, as shown in Figure 1 cameras can be used both by Tenants and by the Smart Building Manager, who redirects the video streaming to emergency services (e.g. firefighters, police, or private surveillance) in case a fire or an intrusion attempt is detected in a house. On the contrary, in normal situation, only the Tenant should have access to the streaming. The third group of traffic is instead dedicated to the Smart Building Management, i.e. it controls the network traffic of sensors and devices present in the common areas, such as elevators, cameras and smoke detectors. We suppose that this complex environment is handled completely via software, by exploiting Software Defined Networks to create ad hoc flows and managing dynamically the amount of bandwidth allocated to each flow. Unfortunately, the level of control enabled by SDN access control mechanisms is not sufficient to manage correctly the smart building environment described before. In particular, the correct management of emergency situation, together with the requirement of ensuring the agreed QoS to Tenants, managing dynamically the bandwidth and workload has the following requirements: (i) Consideration of complex conditions with attributes whose value change over time; (ii) Being able to revoke the availability of a flow after it has been granted, in case the context conditions change. To address these requirements, we integrate in the SDN controller the UCON framework presented in [5].

Figure 2 depicts the logical architecture of the framework. As previously stated in a SDN network switches act only as forward devices, following specific instructions from the controller. Thus, when a packet arrives at a switch and the switch has no installed instructions on how to handle it (PACKET_IN), it forwards this packet back to the controller and waits for the response. At this point, the proposed architecture invokes UCON framework so as to evaluate the request and enforce the given security policies. The evaluation of the request will be performed based not only on static attributes, such as the source and destination IP address (most common rule of a Firewall) but also on attributes which may change during the access period, modifying the context of access or violating the policy and requiring thus actions to be taken. The continuous monitoring of the attributes through UCON and the capability of dynamically revoking sessions which no longer satisfy the policy, gives the possibility of handling a complex network environment and reacting to unforeseen events on time, protecting the resources and the data.

As shown, to implement the proposed architecture, UCS is physically integrated in the same device as the SDN controller, i.e. in this study in the same Virtual Machine where the controller's software is running. As depicted PEP is integrated in the controller layer, meaning that no Northbound API was developed for the interaction of the two components and PEP actually extends the code of the controller making thus the communication faster. Hence, whenever the controller receives

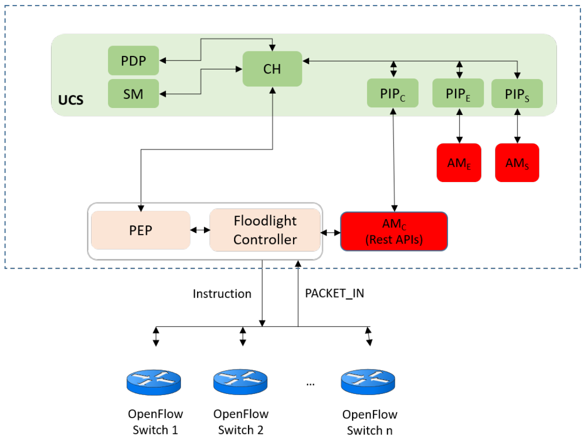

Fig. 2: UCON implementation in SDN.

a PACKET_IN message from the data plane, it immediately triggers the PEP and the evaluation of the request starts. The rest of the UCS components remain untouched, with the CH responsible for handling the interaction amongst them, the SM responsible for keeping track of the active sessions and the PDP responsible for the evaluation of the request based on the given policy. Worth noting is that the PIPs which appear in Figure 2 can be considered as abstract and represent three different groups of PIPs reading values from: i) the controller (PIP_C), ii) the environment (PIP_E) and iii) the subject (PIP_S). The functionality and the role of each component is explained in more details through the following sequence diagrams.

Session establishment: Figure 3 describes the session establishment phase. As mentioned above, the procedure initiates when the switch receives a PACKET_IN message, which is automatically forwarded to the controller. The controller notifies the PEP, which translates the request into a TryAccess message and sends it to the CH, alongside with the security policy. Consequently, the CH retrieves the necessary attributes from the PIPs and sends them with the policy to the PDP for the evaluation. In step 10 the final response of the PDP is given to the PEP through the CH.

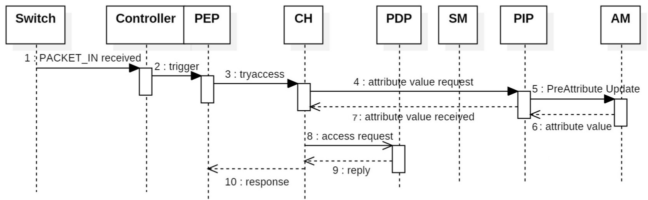

Fig. 3: Session establishment.

Permit of access: Steps 1-7 of Figure 4 shows the sequence diagram in case of a permit of access. If the reply of the PDP is *Permit*, then the SM creates the session and the CH informs the PEP for the positive decision. The PEP afterward, sends an affirmative answer to the controller, which finally installs a *forward* flow to the appropriate switch. Therefore, the packets coming from this requester will be forwarded to the destination, taking also into account the existing policy.

Denial of access: Given that the request is evaluated by the PDP as *Deny*, the PEP is informed through the CH. Consequently, PEP informs the SDN controller and a *drop* flow is installed in the appropriate switch. Hence, all the packets coming from this requester will not be delivered in the destination address.

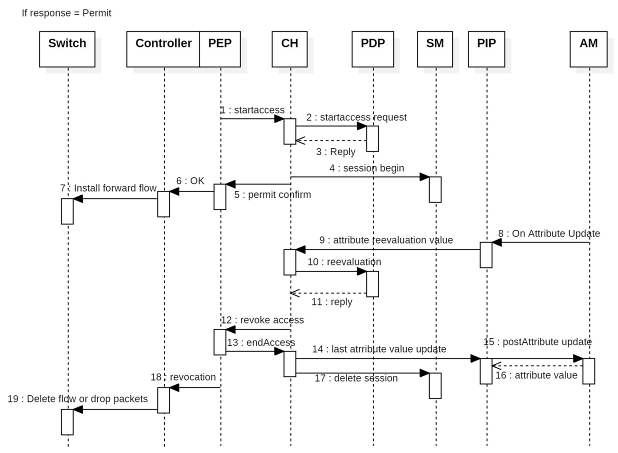

Fig. 4: Permit of access and attribute change.

Attribute Change: Steps 8-19 of Figure 4 describe the behavior of the model in case of a change of an attribute value. The AM will inform the PIP when a change of a value will occur and afterward the PIP will trigger the CH which will ask a re-evaluation from the PDP. If the reply is *Permit* the access continues normally. Otherwise, if the reply is *Deny*, a revoke of access will take place. Hence, the CH will send a revoke access message to the PEP which will also inform the SDN controller about the revocation. The SM will delete the active session and finally the controller will send a *delete* flow to the switch. By this way, flow entries regarding this session will be removed from the switch and replaced with *drop* flows, so as the packets from this requester will no longer be delivered to the destination. It is worth noting at this point that, after a revocation the system is configured to periodically send a TryAccess message (new packets to the switches), so as to trigger a new evaluation. Hence, in case the state of the environment has changed (i.e there is no longer an emergency) or the policies are again satisfied the re-destribution of the bandwidth will take place by installing the appropriate flows in the switches and restoring the network to its normal state.

IV. Implementation and Policy Examples

As described in Section III the chosen application environment for the implementation of a UCON-SDN network is a smart building. The selected controller for the implementation of the SDN network is the Floodlight OpenFlow SDN Controller [1]. Floodlight is a high-performance and widely used

[1]http://www.projectfloodlight.org/floodlight/

controller in the research community, which provides a number of easy to extend modules and many helpful features for conducting experiments. For generating the necessary traffic in the network Mininet[2] was chosen because it gives the possibility of creating a realistic and reliable network, which can be customized to meet the goals of each scenario.

In order to implement the aforementioned scenario and conduct the necessary experiments a topology consisted of three switches all of them connected to a router was chosen. Each one of the switches is dedicated to handle the traffic from one of the 3 traffic groups described above and it is not able to communicate (exchange traffic) with the other switches of the topology. The router is responsible for providing the Internet connection. For applying the QoS scenarios the OpenFlow queues were exploited, which are used for rate-limiting packets. Thus having installed a variety of queues gives to the network administrator the possibility to prioritize or limit traffic based on the given policies. The implemented topology is shown in Figure 5. As depicted for each port of the switches (also for the ports which are connected to the router) there is the possibility to forward the traffic through one of the three installed queues.

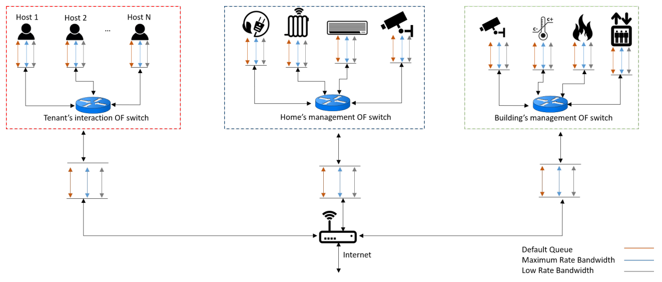
Host 1
Host 2
...
Host N
Tenant's interaction OF switch
Home's management OF switch
Building's management OF switch
Internet
Default Queue
Maximum Rate Bandwidth
Low Rate Bandwidth

Fig. 5: Topology.

For the generation of the traffic a python script was developed and run in the Mininet VM. This script is responsible for creating four Open vSwitches[3] (one is acting as router), which run the OpenFlow 1.3 protocol, assigning a number of hosts in the switches and creating the necessary links amongst them. Once the topology has been created, the script also starts generating ICMP traffic, by sending ping requests from one host to the others hosts of the network. Furthermore a second python script was run in the Mininet VM, in order to install in each switch 3 different OpenFlow queues as described above. Hence, we can define UCON policies, which based on the state of the network will dictate which queue must be used. Following, two example of policies are given.

Policy 1: Management of SLAs QoS. Let's consider for simplicity a smart building with two Tenants that have with the Smart Building Manager two different SLAs, namely a premium SLA and a low cost SLA. Premium SLA ensures access at any time at the Premium High Speed Streaming flow and at the other two flows of the first semantic group. The low cost SLA instead, only ensures the access to the Browsing and Email flow. The Multimedia Streaming is instead provided at best effort. For simplicity, without lacking in generality, we suppose that the bandwidth required by the Premium High Speed Streaming is not enough to support Multimedia Streamings. Hence, the low cost user can use the Multimedia Streaming flow till there is enough bandwidth to allow both this traffic and any traffic required by the premium user, and there are not pending requests for Premium High Speed Streaming. The needed attributes are the following: (i) `Available bandwidth` extracted from network stats, (ii) `High Speed Streaming Requests` and (iii) `SLA of Requester` checked by querying a MySQL database containing the information of each tenant. The SLA PIP is thus implemented through a jdbc connector and performs a query of the SLA attribute in the Tenants table. The correspondent policy can thus be expressed as: "IF `SLA of Requester` is `Low Cost` AND `High Speed Streaming Requests` is 0 AND `Available bandwidth` is GREATER THAN `threshold` THEN PERMIT. DENY otherwise ". This policy can be translated easily in U-XACML, thus enforced by the UCON framework. The XACML policy is not reported here due to space limitation. For what concerns the enforcement, as the low cost tenant attempts to open a Multimedia Streaming flow by sending the first packet, the control plane will forward the request through the PEP to the UCON issuing a tryAccess followed by a startAccess. The PIPs will collect the SLA level of the user, the available bandwidth and the eventual presence of pending or ongoing Premium High Speed Streaming connections, the policy is thus evaluated and the flow installation is eventually granted if the conditions match the policy. After the flow has been created, a context change, such as the request for a Premium High Speed Streaming from the premium tenant, will trigger the policy re-evaluation and consecutive revocation of the flow. Thus, the control plane will uninstall through the PEP the flow for the low cost user, forbidding the creation of a new one till the conditions are not matching the policy.

Policy 2: Emergency Management. Any user is allowed to use the bandwidth as described in the subscribed SLA unless an emergency such as a fire in the building is detected. The rationale behind this policy is that in case of an emergency, a high speed channel should be dedicated to the communication with firefighters, who might use the Smart Building Management network and the House Management Traffic to acquire high definition pictures from surveillance cameras and to remotely control the elevators. The needed attributes are: (i) `Available bandwidth`; (ii) `Fire Alarm Status` a boolean value which is provided by a sensor. The fire alarm sensor, in particular will write on a file the current reading. The PIP used to query this value is thus a file watcher triggered each time the file is updated. The file content can be either TRUE, i.e. a fire has been detected, or FALSE.); (iii) `SLA of Requester` and (iv) `Flow Type`: One of the three flow types belonging to the first semantic group. This attribute is automatically sent by the PEP in the request. The correspondent policy can thus be expressed as: "IF `SLA of Requester` is `Premium`

[2]http://mininet.org/
[3]https://www.openvswitch.org/

Tenant AND Flow Type is ANY OF Premium High Speed Streaming or Multimedia Streaming or Browsing and Emails AND Fire Alarm Status is FALSE, then PERMIT. ”“IF SLA of Requester is Low Cost Tenant AND Flow Type is ANY OF Multimedia Streaming or Browsing and Emails AND Available bandwidth is GREATER THAN threshold AND Fire Alarm Status is FALSE, then PERMIT. DENY otherwise ”As shown, this policy is more complex, showing three rules that are combined with the *First Applicable* combination algorithm [15], i.e. the first rule in this order that is valid, defines the system decision. For the enforcement, the workflow is similar to the one described in the previous example.

V. Experimental Evaluation

The testbed consists of two virtual machines (VMs), running on a host machine equipped with an Intel i7-7500U with 2 cores enabled. The first VM is used to run the Floodlight controller and the UCON framework and is equipped with 6GB DDR4 RAM, while the second VM is used to run Mininet and is equipped with 1GB DDR4 RAM. Both of the VMs have installed as OS Ubuntu 16.04 64-bit.

In order to evaluate the proposed model, two sets of experiments were conducted. The first experiment, whose results are shown in Figure 6, evaluates the time required to execute a complete evaluation of a PACKET_IN packet. The evaluation time starts upon the receiving of such a packet from the controller until the flow is successfully installed to the switch. If the given answer to PEP from the CH is *permit*, then a *forward* flow is installed and the communication amongst the hosts initiates. Otherwise, if the answer is *deny*, a *drop* flow is installed and the switch drops the packets, blocking thus the communication.

TABLE I: Overhead introduced by UCON

N. of attr.	TryAccess	StartAccess	Flow Install
2	1.8 s	1.4 s	1.5 s
10	2.6 s	2.2 s	1.5 s
40	2.9 s	2.7 s	1.5 s

For this experiment we used five different policies, increasing each time the number of attributes which UCON will evaluate and we measured the elapsed time in milliseconds. For each policy, the experiments have been repeated a number of time sufficient to obtain a low standard deviation, computing thus average, minimum and maximum times reported in the graph of Figure 6. It is worth mentioning that the policies are written in such a way so as to make sure that the value of all attributes has to be evaluated. The results show that, as the number of attributes increases, there is also a slightly increase of the evaluation time, which is something expectable considering that the PDP adds overhead for evaluating more attributes. As shown, the time required for installing a flow, from request time to completed installation ranges from 4.7 s to 7.2 s, depending on the number of attributes to be evaluated. Without UCON, the time needed to install a flow ranges from 1 s to 2 s. Details on the overhead are reported in Table I.

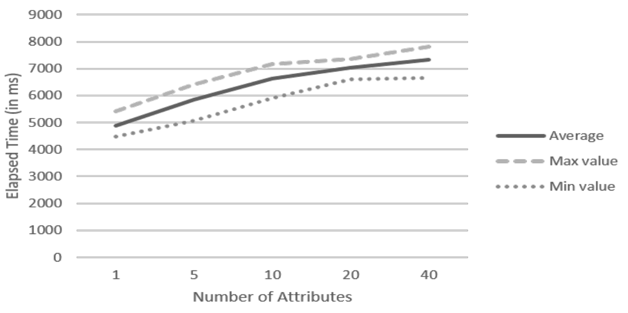

Fig. 6: Evaluation Time of a *PACKET_IN*.

The first two columns represent the actual overhead generated by the UCON for performing the two access requests, which is of the same order of the time needed to install the flow, thus should not introduce an observable delay able to modify the user experience. The goal of the second experiment

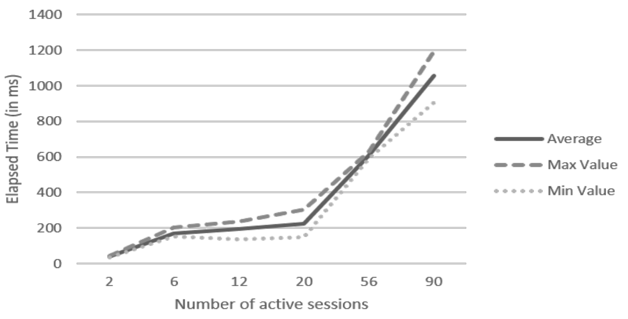

Fig. 7: Revocation time.

is to evaluate the required time for revoking an active session. For this experiment the evaluation time is measured from the moment when the PEP receives the revocation response until the appropriate flow is installed to the switch and the communication between the relevant hosts is terminated. The performance analysis was utilized by considering an increasing number of active sessions and policies with one and five attributes. Each time we run the scenario three times, taking as above the average, the minimum and the maximum elapsed time. Figure 7 depicts the results for this experiment. We observe that as the number of active sessions grows so is the time needed for the revocation of all these sessions, upon a violation of the policy. This difference is expectable since from one hand UCON framework has to re-evaluate more sessions and on the other the controller has to instruct the switch to delete more flows.

VI. Related Work

In [8] the authors propose a communication architecture for a smart home which considers both the security factor, using an authentication system and a role based access control, and the QoS assigning a defined data rate for each application. With respect to this work, our proposal combining UCON and SDN provides a continuous evaluation of the right of access, with a more configurable network interface. In [10] a bandwidth allocation framework is designed. This architecture aims to provide an ISP with the possibility to optimize the internal and external traffic of a smart home. Even though in this study an SDN network is used, the authors do not consider the security factor and the continuous enforcement of policies. In [1] the authors propose security frameworks especially for smart environments, while in [7] an access control scheme which is oriented for devices belonging to a smart environment is proposed. However, the studies mentioned above do not consider the continuous enforcing of the security policies and the dynamical implementation of QoS scenarios based both on these. In the smart buildings environments there are also a number of studies such as the one of [3] where an access control framework is introduced. Although, this solution does not provide nor context awareness neither a fine-grained access since one have either full access or not at all.

A study which considers the UCON model is [9], where the authors propose authorization models for ubiquitous computing environment. However, the authors do not consider an SDN enabled network and QoS scenarios. A QoS aware solution for smart buildings is presented in [18]. The authors identified IoT number of scenarios of IoT applications and their requirements in 5G networks and proposed a QoS mechanism with heterogeneous IoT devices. Although, they do not consider an access control model and moreover they take into account only the devices connected to the network and not the users and their characteristics. An authentication framework, which utilizes the different wearable and embedded devices used in ubiquitous computing environments is proposed in [1]. This framework is build over Kerberos and although it offers an authentication mechanism it does not provide a continuous evaluation based also on other attributes of the subject, the object and the environment. Finally, an architecture of authenticating sensors and context receivers in a smart space is presented in [2], which however also lacks to provide continuity of evaluation.

VII. Conclusion and Future Work

A Smart Building environment, being a representative example of pervasive and mobile computing, is required to provide to its Tenants the assurance that their information will be protected, their access to the building's network will be performed with a certain provides QoS and that all necessary countermeasures will be taken upon an emergency event. To this end, this study presents a framework for applying QoS in a Smart Building, exploiting the SDN technology and UCON. The experiments show that the overhead introduced by the proposed framework is in an acceptable limit. As future work we plan to apply the proposed framework to a more extended experimental testbed, considering multiple switches and a bigger number of hosts.

Acknowledgments

This work has been partially funded by EU Funded project H2020 NeCS, GA #675320 and H2020 C3ISP, GA #700294 and EIT Digital Trusted Cloud and Internet of Things.

References

[1] J. Al-Muhtadi, A. Ranganathan, R. Campbell, and M. D. Mickunas. A flexible, privacy-preserving authentication framework for ubiquitous computing environments. In *Proceedings of the 22nd International Conference on Distributed Computing Systems Workshops*, pages 771–776, 2002.

[2] S. Al-Rabiaah and J. Al-Muhtadi. Consec: Context-aware security framework for smart spaces. In *Proceedings of the 6th International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing*, pages 580–584, July 2012.

[3] S. Bandara, T. Yashiro, N. Koshizuka, and K. Sakamura. Access control framework for api-enabled devices in smart buildings. In *Proceedings of the 22nd Asia-Pacific Conference on Communications*, pages 210–217.

[4] T. Benson, A. Akella, and D. Maltz. Unraveling the Complexity of Network Management. In *Proceedings of the 6th USENIX Symposium on Networked Systems Design and Implementation*, pages 335–348, Berkeley,CA, USA, 2009.

[5] E. Carniani, D. D'Arenzo, A. Lazouski, F. Martinelli, and P. Mori. Usage control on cloud systems. *Future Generation Comp. Syst.*, 63:37–55, 2016.

[6] Yap Kok-Kiong et. al. Taking the Edge off with Espresso: Scale, Reliability and Programmability for Global Internet Peering. In *Proceedings of the Conference of the ACM Special Interest Group on Data Communication*, pages 432–445, New York,USA, 2017.

[7] K. Fysarakis, C. Konstantourakis, K. Rantos, C. Manifavas, and I. Papaefstathiou. Wsacd - A usable access control framework for smart home devices. In *Proceedings of Information Security Theory and Practice - 9th IFIP WG 11.2 International Conference,Heraklion, Greece, August 24-25*, pages 120–133, 2015.

[8] M. Hager, S. Schellenberg, J. Seitz, S. Mann, and G. Schorcht. Secure and qos-aware communications for smart home services. In *Proceedings of the 35th International Conference on Telecommunications and Signal Processing (TSP)*, pages 11–17, 2012.

[9] W. Hua, Z. Yanchun, and C. Jinli. Access control management for ubiquitous computing. *Future Generation Computer Systems*, 24(8):870 – 878, 2008.

[10] Hung-Chin Jang, Chi-Wei Huang, and Fu-Ku Yeh. Design a bandwidth allocation framework for sdn based smart home. In *2016 IEEE 7th Annual Information Technology, Electronics and Mobile Communication Conference (IEMCON)*, pages 1–6, Oct 2016.

[11] H. Kim and N. Feamster. Improving network management with software defined networking. *IEEE Communications Magazine*, 51(2):114–119, February 2013.

[12] D. Kreutz, F. M. V. Ramos, P. E. Verssimo, C. E. Rothenberg, S. Azodolmolky, and S. Uhlig. Software-defined networking: A comprehensive survey. *Proceedings of the IEEE*, 103(1):14–76, Jan 2015.

[13] A. Lazouski, G. Mancini, F. Martinelli, and P. Mori. Usage control in cloud systems. In *The 7th International Conference for Internet Technology And Secured Transactions,(ICITST-2012)*, pages 202–207.

[14] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner. Openflow: Enabling innovation in campus networks. *SIGCOMM Comput. Commun. Rev.*, 38(2):69–74, March 2008.

[15] J. Park, X. Zhang, and R. Sandhu. Attribute mutability in usage control. In *Research Directions in Data and Applications Security XVIII, IFIP TC11/WG 11.3 Eighteenth Annual Conference on Data and Applications Security*, pages 15–29, 2004.

[16] A. Pretschner, M. Hilty, and D.A. Basin. Distributed usage control. *Communications of the ACM*, 49(9):39–44, 2006.

[17] M. Satyanarayanan. Pervasive computing: vision and challenges. *IEEE Personal Communications*, 8(4):10–17, Aug 2001.

[18] N. ul Hasan, W. Ejaz, I. Baig, M. Zghaibeh, and A. Anpalagan. Qos-aware channel assignment for iot-enabled smart building in 5g systems. In *2016 Eighth International Conference on Ubiquitous and Future Networks (ICUFN)*, pages 924–928, July 2016.

[19] Xinwen Zhang, Francesco Parisi-Presicce, Ravi Sandhu, and Jaehong Park. Formal model and policy specification of usage control. *ACM Transactions on Information and System Security*, 8(4):351–387, 2005.