SDN-Based Service Delivery in Smart Environments

Lucas Mendes Ribeiro Arbiza, Liane Margarida Rockenbach Tarouco, Leandro Márcio Bertholdo, and Lisandro Zambenedetti Granville

Abstract Internet of Things scenarios demand adaptability to hold the heterogeneity of systems and devices employed; gateway-based solutions are a common answer to the issues of smart environments. The development of software for gateways, using a middleware to handle the different devices demands, is possible in our working scenario, but the maintenance cost of such a solution is high because of software development constraints imposed by hardware and system limitations of the gateway. This paper depicts an SDN approach for smart environments resulted from a refactoring of a previous middleware proposal. Through an instantiation of the refactored middleware in a home network to deliver a health monitoring service the benefits are demonstrated; the benefits regard the digital representation of the physical realm, deployment and maintenance of services, and management of devices and networks.

1 Introduction

The rapid growth of population impose to cities a major challenge to their sustainability, as well as a threat to the infrastructure of main services provided to the population. A solution is expected to show up with the advent of smart cities. As discussed in [1], more than 150 cities can be documented around the world as smart.

Smart cities require IT services to capture, integrate, analyze, plan, inform, and act intelligently on city activities, resulting in a better place to live. This implies services to make life easier for people and businesses. A smart community is a community that carries out conscious efforts to use information technology to trans-

Lucas Arbiza · Liane Tarouco · Leandro Bertholdo · Lisandro Granville
Informatics Institute, Federal University of Rio Grande do Sul (UFRGS), Porto Alegre, Brazil.
e-mail: {lmrarbiza,granville}@inf.ufrgs.br,{berthold,liane}@penta.ufrgs.br

form significantly and fundamentally the way of life and work within its territory, instead of following an incremental way.

From this perspective, a smart city refers to a physical environment in which communication and information technology, and sensor systems that are part of it, disappear as they become embedded into physical objects and the environments in which people live, travel, and work [2]. Smart cities should use smart computing technologies to build critical infrastructure components and services of a city more intelligent, interconnected, and efficient [3].

Homes have becoming increasingly smarter embedding many devices able to sense their surroundings; usually those devices are part of a solution for home automation, security, healthcare, among other purposes. Smart solutions employed in citizens homes may be part of broad system, such as a smart city solution for intelligent use and distribution of energy. Smartness in home environments bring some issues for network management, security and orchestration due to the demands to handle the heterogeneity of smart devices when they are employed in services delivering.

Software-Defined Networking (SDN) is a paradigm where network intelligence and control are taken from forwarding devices and are deployed in central controllers where network logic behavior is defined by software, developed or customized to fit the needs of each network environment. SDN provides to the network the flexibility of software, allowing the development of features not available in network hardware; using SDN enables the development or the use of applications for network orchestration and management, suitable to deal with heterogeneity of smart environments.

This paper presents the use of a SDN-based middleware [4] in a health monitoring environment aiming to achieve a simpler and easier to manage solution to monitor patients in their own homes, when compared to a previous middleware where SDN was not used. In the same scenario, we also exploit the use of SDN resources for network management for smart environments. The presented approach also enables the delivering of multiple services for smart environments, sharing the same network infrastructure.

The rest of this paper is organized as follow. Section 2 presents some concepts of smart environment, focusing specially in healthcare. Section 3 details what is SDN paradigm, how it works, and its resources. In section 4 we discuss how SDN can be used to empower smart environments, we present the SDN-based middleware used detailing its architecture, implementation and workflow. Section 5 demonstrates the use of the SDN-based middleware for health monitoring and benefits achieved by using the SDN approach. In section 6 we present our final considerations.

2 Smart Environments

Smart cities are composed of smart devices. Currently, a large number of smart objects and different types of devices are interconnected and communicate via the

Internet Protocol, which creates a worldwide ubiquitous and pervasive network referred to as the Internet of Things (IoT) [5, 6]. With the inception of IoT, the Internet is further extended to connect things, such as power meters, heartbeat monitors, temperature meters, and many powerful operations, such as health care units, green energy services, and smart farming utilities, that can be made available to people for enhanced quality of life.

IoT is becoming the Internet of Everything (IoE) [7]. Most of smart devices, used in the context of Internet of Things (IoT), do not employ generic/open standards when communicating. One of the challenges in this context derives from the need to find a form of automated communication and to integrate the various devices and protocols, making it possible to obtain information about the scenario being monitored.

Data collected by devices on a given environment are combined to provide services (*e.g.*, smart homes, and healthcare) [8]). The combination can also be made through mashups [9], where data from different sources and using various resources are handled to make possible creating a vision of a whole.

3 Software-Defined Networks

SDN is mainly known for decoupling network control plane from the forwarding devices, usually combined in the same device, such as routers. Decoupling enables the network logic to be defined at the software level and to implement features that may not be available in the network hardware in use [10]. In general, the network logic behavior is defined configuring every network device individually, using vendor specific syntax and limited to the features available according to the licenses acquired. SDN allows to centralize network intelligence in a controller that sends forwarding rules, called *flow entries*, to the switches and routers defining logic behavior of the network [11].

Figure 1 illustrates the three layers SDN architecture: 1) Infrastructure layer: comprised of forwarding devices (switches and routers) enabled with an SDN standard, such as OpenFlow; 2) Control layer: one or more devices running a controller software enabling the communication between application and infrastructure layers; and 3) Application layer: one or more applications by which network intelligence is implemented; this layer makes use of the control layer to configure forwarding devices. Control layer communicates with infrastructure layer through OpenFlow messages or other SDN standard.

When a forwarding device, also called OpenFlow switch, receives a packet the switch looks up in its *flow tables* for a *flow entry* matching the received packet. If none of the existing *flow entries* match the packet, the switch sends a *packet-in* to the controller. *packet-in* is processed at the application layer and a *flow entry* is sent to the switch through the controller containing forwarding rules for the packet.

The controller has an entire view of the network; it is aware of every switch connected to it and of every device connected to the switches. That privileged view is a

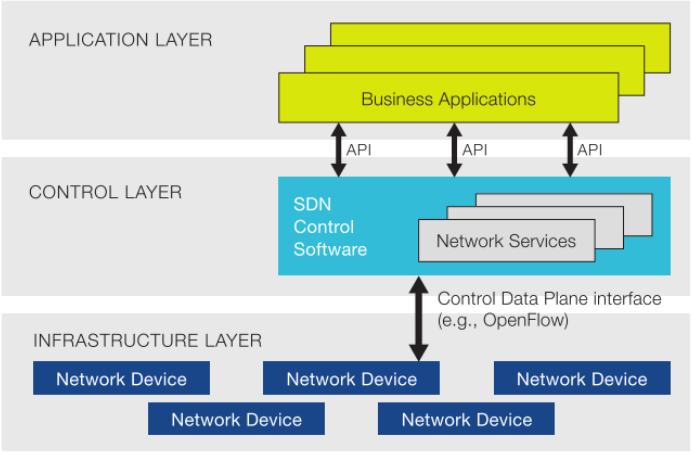

Fig. 1 Software-Defined Networking Architecture [10].

valuable resource exploited in some works that propose SDN-based approaches in different network related fields, for example: network virtualization: SDN is used to create isolated slices in the network over the same physical hardware [12, 13, 14]; routing: routes established by BGP and OSPF are reflected to low cost switches not enabled with routing protocols [15]; QoS: network paths are dynamically allocated assuring the best performance and availability for differentiated traffic [16]; mobility: aiming to prevent Wi-Fi handover from occurring when users are moving the traffic is sent to more than one access point simultaneously [17]; network management: SDN is used combined with SNMP and others protocols and security and authentication mechanisms existing in a campus, events arising from different sources and network policies are translated to *flow entries* to be installed on the forwarding devices [18].

Taking network control to the application level empowers a low cost network with capabilities not available in the existing hardware, as seen in [15]. By using SDN, network administrators do not dependent on features provided by vendors anymore, since new features can be developed using high level abstraction programming languages and management solutions may be combined with control applications to improve network intelligence, management, and automation.

4 Empowering Smart Environments with SDN

Recently, SDN started to be exploited in smart environments. In research efforts such as [19, 20], the authors made use of SDN in home networks, as we do in our work as well. In [19], the authors provide a simplified interface by which non technical users are able to configure their home networks properly. Those users

usually expose themselves to security and privacy risks just because they do not know technical information required to configure most of access points. OpenFlow is used to translate network settings configured by users to *flow entries* to be installed in the SDN empowered access point. In [20], SDN is used to slice the network in isolated virtual networks allowing different service providers to share the same network infrastructure to deliver their services at users' homes. Each provider have control over its slice to deliver the service as needed.

Based on the works mentioned above, in [4] we used SDN to refactor an IoT middleware designed to enable health monitoring of patients with chronic illnesses in their own homes [21]. In an environment empowered with SDN, we are able to provide a wider view of a sensed environment. In [22], IoT is defined as a digital representation of the physical realm built from data collected by devices enabled to sense the environment they are in. Although off-the-self sensing devices usually employ a vertical communication called *silos* [23, 24, 25], in this case retrieved data are sent to proprietary servers and cannot be retrieved directly from devices. OpenFlow provides resources by which one can build a representation of an environment based on the communication of all connected devices in a home network. SDN also benefits IoT in network orchestration and management. The following subsections present our approach to achieve our goals in IoT environments through SDN.

4.1 Architecture and Workflow

The architecture depicted in figure 2 is split into three layers: the access points, the controller called Derailleur, and the application called ThingsFlow. APs act as OpenFlow switches forwarding packets according to the *flow entries* received. Derailleur controller listens to AP connections; when an AP establishes a connection, Derailleur uses OpenFlow messages to retrieve information from the AP to build an abstraction as a switch object. Each AP may be identified and accessed individually. Derailleur owns and manages switches objects that are created or destroyed reflecting APs status. The controller triggers events to be handled by the application when APs connect, disconnect, or send OpenFlow messages. ThingsFlow is the application where network intelligence is implemented. Each AP may provide a different set of services, so ThingsFlow installs in a given AP only the *flow tables* of services that AP provides. The ThingsFlow also collects counters from APs and make it available to the services, thus to be used for different purposes, for example, management and services features.

4.2 Implementation Details

APs run OpenWRT firmware, a Linux-based operating system for embedded devices, built containing the virtual switch Open vSwitch which provides OpenFlow

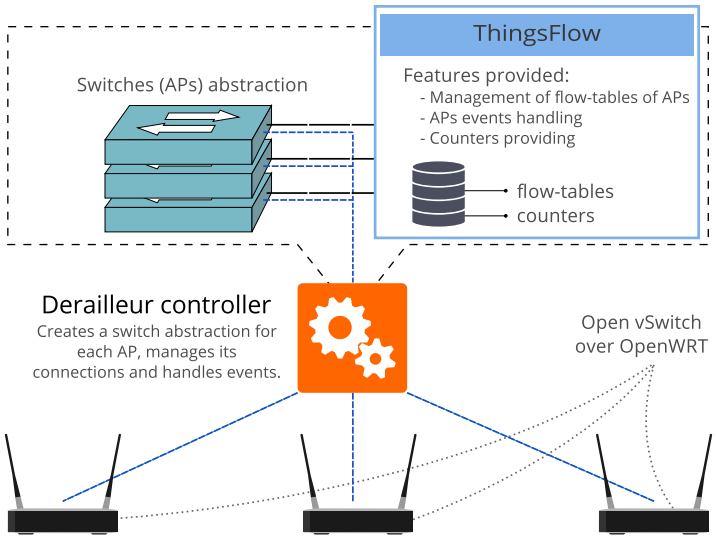

Fig. 2 Architecture overview and components roles [4].

capabilities to APs. APs can be replaced by any other hardware and operating system able to run Open vSwitch.

The Derailleur controller was developed in C++ using libfluid [26], the winner of Open Networking Foundation OpenFlow Driver Competition. libfluid is composed of two libraries: *libfluid_base*, that provides server features such as listening loop and events handling; and *libfluid_msg*, that provides mechanisms to build and parse OpenFlow messages.

ThingsFlow was developed inheriting an abstract application class provided by Derailleur. Through the abstract application class, the controller shares and provides access to the switches objects to ThingsFlow.

5 SND-based Healthcare

To demonstrate the benefits achieved employing our SDN-based approach in a smart environment, we use the same health monitoring example that we were working when we designed the previous middleware, in the REMOA[1] project. REMOA is a project that targets home solutions for care/telemonitoring of patients with chronic illnesses. The project also encompasses the design and implementation of a middleware to address issues found in monitoring devices used in the project, enabling

[1] *Rede Cidadã de Monitoramento do Ambiente Baseado no Conceito de Internet das Coisas* (Citizen Network for Environment Monitoring Based on the Concept of Internet of Things)

interoperability and security needed in the context of IoT for healthcare. The issues addressed are the following:

- Interoperability: used devices do not reply requests; they just send the data they read to vendor servers employing proprietary standards to communicate, what characterizes a *silo*. Settings, such as destination server or protocols to be used when transmitting data, cannot be changed.
- Security and privacy: data transmission is neither encrypted nor authenticated.
- Management: traditional management mechanisms, such as those based on ICMP or SNMP, cannot be used because of the sleeping scheduled mechanism employed by devices to save battery.
- Data structure: proprietary standards are also used to structure the data sent. It forces every data received to be parsed according to vendor standards. Usually vendors of devices designed for end-users do not provide the documentation required for parsing. The scheme used to parse the messages results from the analysis of the communication of the devices.

Figure 3 illustrates two different views of the same environment at the same moment. The hypertensive patient has fallen asleep while was watching TV; the patient is late with blood pressure measurement. Movements of patient are not captured by presence sensor because do no exceed thresholds. The left side illustrates the view provided by the REMOA middleware, where digital representation of the environment is based only on data provided by monitoring devices. The right side illustrates a view built combining data from monitoring devices and all other devices transmitting at that moment.

Monitoring based on health devices and presence sensor would trigger an emergence event if data from OpenFlow counters were not taken into account. The view

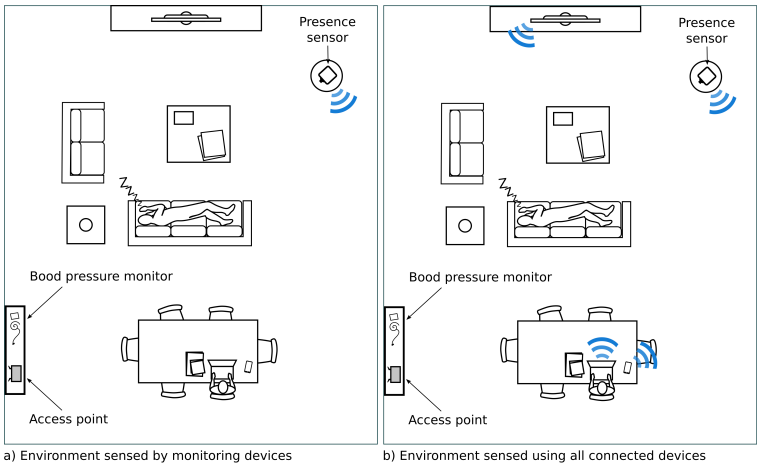

Fig. 3 Comparison between digital representations of the same environment with and without using OpenFlow counters.

illustrated in the right side of the figure is built using OpenFlow counters from *flow entries* installed on APs to forward traffic of devices used in the house of the monitored patient. These counters allow monitoring staff to know that the patient is not alone because a smartphone and a notebook belonging to one of the residents are in use at that moment. It is also possible to know that the TV is on. Monitoring staff can suppose that the patient is watching TV, making short movements and forgetting to do measurement. The action to be taken could be a phone call to remember the patient about the missing measurements.

In [21], the management of health monitoring devices was based in SNMP. Information about devices communication were stored in a Management Information Base (MIB) in the AP by a middleware module. In the SDN-based approach, this information is provided by OpenFlow counters, collected from APs by ThingsFlow, that makes counters available to the service provider that implements management mechanisms suitable for each monitoring device. Counters are combined with other data sources, such as the battery level transmitted by monitoring devices along collected data. Another important benefit achieved is the scalability regarding deployment of APs and monitoring devices. The configuration of the network behavior in the patient's residence is completely automated by OpenFlow; ThingsFlow provides the suitable *flow entries* for each AP. Through the orchestration of the network provided by SDN the network complexity was moved from APs to remote servers allowing developers to use of any development resources available in services features development instead of getting limited by hardware and system constraints of the APs. OpenFlow also rewrites packets headers to prevent health data of patients from being sent to proprietary serves; those packets are so forwarded to the monitoring server where patient data are processed.

6 Final Considerations

Designing a network solution for an smart environment is not an easy task because of the different demands of the smart devices. Solutions are usually based on local gateways that not only are in charge of related communication tasks but also must provide security, compatibility, and network management. To keep a gateway-based solution simple and less expensive, it is necessary to use an approach that enables the flexibility and functionality demanded, without high complexity and cost. In this sense, transferring more complex functions to a remote server and leaving the gateway to take care of only the task of switching and routing is the proposed solution presented in this paper.

Moving network control to the application level empowers a low cost network with capabilities usually not available in the gateway hardware. By using SDN, network administrators are neither dependent nor limited to the features provided by gateway vendors anymore. New features can be developed using high level abstraction programming languages and management solutions can be combined with control applications to improve network intelligence and automation.

The environment presented in this work provides health care service at home level to patients with chronic illness, but can also be used in other smart city environments as well. This work demonstrated how SDN can be used to build a wider view of an smart environment combining data from sensing devices with counters of network communication of all connected devices. SDN is also a good choice for smart environments because it enables the development of more appropriate management mechanisms and improves flexibility in network orchestration to fit the diversity of smart devices. As future work, we will deploy more complex scenarios with a wider range of services, aiming to exploit deeper SDN resources and its benefits for this kind of environment.

References

[1] L. Anthopoulos, P. Fitsilis, in *Advanced Communication Technology (ICACT), 2014 16th International Conference on* (2014), pp. 190–195. DOI 10.1109/ICACT.2014.6778947

[2] A. Steventon, S. Wright (eds.), *Intelligent Spaces: The Application of Pervasive ICT*, 1st edn. (Springer, London, 2006). DOI 10.1007/978-1-84628-429-8

[3] D. Washburn, U. Sindhu, S. Balaouras, R.A. Dines, N. Hayes, L.E. Nelson, Helping CIOs Understand "Smart City" Initiatives. Tech. rep., Forrester Research Inc. (2010)

[4] L.M.R. Arbiza, L.M. Bertholdo, C.R.d.P. Santos, L.G. Granville, L.M.R. Tarouco, in *30th ACM/SIGAPP Symposium On Applied Computing* (ACM, Salamanca, Spain, 2015), pp. 640–645. DOI 10.1145/2695664.2695861

[5] J. Gubbi, R. Buyya, S. Marusic, M. Palaniswami, Future Gener. Comput. Syst. **29**(7), 1645 (2013). DOI 10.1016/j.future.2013.01.010

[6] I. Mashal, O. Alsaryrah, T.Y. Chung, C.Z. Yang, W.H. Kuo, D.P. Agrawal, Ad Hoc Networks **28**(0), 68 (2015). DOI http://dx.doi.org/10.1016/j.adhoc.2014.12.006

[7] K.S. Yeo, M. Chian, T. Ng, D.A. Tuan, in *Integrated Circuits (ISIC), 2014 14th International Symposium on* (2014), pp. 568–571. DOI 10.1109/ISICIR.2014.7029523

[8] R. Blasco, A. Marco, R. Casas, D. Cirujano, R. Picking, Sensors **14**(1), 1629 (2014). DOI 10.3390/s140101629

[9] C.R.P.d. Santos, R.S. Bezerra, J.M. Ceron, L.Z. Granville, L.M.R. Tarouco, IEEE Communications Magazine **48**(12), 112 (2010)

[10] Software-Defined Networking: The New Norm for Networks (2012). URL `https://www.opennetworking.org/images/stories/downloads/sdn-resources/white-papers/wp-sdn-newnorm.pdf`

[11] J.A. Wickboldt, W.P.d. Jesus, P.H. Iolani, C.B. Both, J. Rochol, L.Z. Granville, IEEE Communications Magazine **53**(1), 278 (2015)

[12] N. Mckeown, T. Anderson, L. Peterson, J. Rexford, S. Shenker, S. Louis, (2008)

[13] R. Sherwood, G. Gibb, K.K. Yap, G. Appenzeller, M. Casado, N. McKeown, G. Parulkar, (2009). URL http://archive.openflow.org/downloads/technicalreports/openflow-tr-2009-1-flowvisor.pdf

[14] R.B. Rafael Pereira Esteves, Lisandro Zambenedetti Granville, IEEE Communications Magazine **51**(7), 80 (2013)

[15] RouteFlow Project. URL https://sites.google.com/site/routeflow/home

[16] H.E. Egilmez, S.T. Dane, K.T. Bagci, A.M. Tekalp, in *Signal & Information Processing Association Annual Summit and Conference (APSIPA ASC), 2012 Asia-Pacific*. Koc Univ., Istanbul, Turkey (IEEE, 2012), pp. 1–8

[17] K.K. Yap, M. Kobayashi, R. Sherwood, T.Y. Huang, M. Chan, N. Handigol, N. McKeown, ACM SIGCOMM Computer Communication Review **40**(1), 125 (2010)

[18] H. Kim, N. Feamster, Communications Magazine, IEEE **51**(2), 114 (2013). DOI 10.1109/MCOM.2013.6461195

[19] M. Chetty, N. Feamster, SIGCOMM Comput. Commun. Rev. **42**(3), 54 (2012). DOI 10.1145/2317307.2317318

[20] Y. Yiakoumis, K.K. Yap, S. Katti, G. Parulkar, N. McKeown, in *Proceedings of the 2Nd ACM SIGCOMM Workshop on Home Networks* (ACM, New York, NY, USA, 2011), HomeNets '11, pp. 1–6. DOI 10.1145/2018567.2018569

[21] L.M.R. Tarouco, L.M. Bertholdo, L.Z. Granville, L.M.R. Arbiza, F. Carbone, M. Marotta, J.J.C. de Santanna, in *IEEE International Conference on Communications, International Workshop on Mobile Consumer Health Care Networks, Systems and Services* (Ottawa, 2012), pp. 6121–6125. DOI 10.1109/ICC.2012.6364830

[22] D. Miorandi, S. Sicari, F.D. Pellegrini, I. Chlamtac, Ad Hoc Networks **10**(7), 1497 (2012). DOI 10.1016/j.adhoc.2012.02.016

[23] IOT-A: Internet of Things Architecture. URL http://www.iot-a.eu/public

[24] G. Wu, S. Talwar, K. Johnsson, N. Himayat, K.D. Johnson, Communications Magazine, IEEE **49**(4), 36 (2011). DOI 10.1109/MCOM.2011.5741144

[25] L. Coetzee, J. Eksteen, in *IST-Africa Conference Proceedings, 2011* (2011), pp. 1–9

[26] libfluid - The ONF OpenFlow driver. http://opennetworkingfoundation.github.io/libfluid/index.html