A Prototype for Dynamic Provisioning of QoS-oriented Virtualized Network Functions in the Internet of Things

Clovis Anicet Ouedraogo, El-Fadel Bonfoh, Samir Medjiah, Christophe Chassot, Sami Yangui

A Prototype for Dynamic Provisioning of QoS-oriented Virtualized Network Functions in the Internet of Things

Clovis Anicet Ouedraogo*, El-Fadel Bonfoh*, Samir Medjiah*†, Christophe Chassot*‡ and Sami Yangui*‡
Univ. Toulouse, * CNRS-LAAS, † UPS, ‡ INSA, F-31400 Toulouse, France
{*FirstName.LastName*}@laas.fr

***Abstract*—Maturity of virtualization techniques and the success of rich cloud services have pushed the horizon of a new sophisticated and connected IoT systems over the Internet. However, provisioning, including managing, such systems is still static, and consequently, costly and time consuming. It is poorly suited to the dynamic criterion of such systems. In this paper, self-adaptive management procedures are proposed towards QoS maintaining at the middleware level of IoT systems. The design is based on concepts such as Network Function Virtualization (NFV) and its generalization, Software-defined Networks (SDN) and edge computing. A realistic use case is implemented for illustration and validation purposes. Prototype execution shows the self-adaptive capabilities of the approach to maintain operative QoS while facing unpredicted events such as fire detection.**

***Index Terms*—ANF, edge computing, IoT, QoS, NFV, SDN**

I. Introduction

According to the standardization bodies [1] [2], the IoT contextual model consists of four strata: Device, Network, Middleware, and Application. This model is depicted in Fig.1. IoT Devices collect and send data to the running Applications through the Network facilities. Middleware entities are often needed to support and manage the heterogeneous capabilities and/or requirements of Applications and Devices (e.g. different communication protocols, different data formats). More specifically and based on both ETSI SmartM2M [1] and oneM2M [2] specifications, the Middleware stratum consists of two types of entities, i.e. an IoT Server and a set of Gateways. The Gateways are the building blocks of the Middleware strata. They implement Network Functions (NFs) that enable devices to connect and communicate with each other and with applications by translating different protocols, formatting data to the appropriate shape and so on. The IoT Server provides the medium for IP-level communication between the Gateways and the underlying IoT Devices. Such contextual model is enabling a plethora of novel and various end-user applications. Home automation, healthcare and goods tracking are among the examples. Efficiency in resources usage, scalability, elasticity and easy provisioning are the key requirements for these applications. Furthermore, a critical challenge for such an architecture is to make up and running Quality of Service (QoS) such as operative latency. On one hand, IoT-based Applications are often latency-sensitive, demanding in terms of service availability and needs to react quickly and flexibly to changing in the operating environments [3] (e.g. detecting a fire in smart home, braking a vehicle when facing an unanticipated obstacle). On the other hand, according to the IoT contextual model, these applications are quite far from data sources.

978-1-5386-4633-5/18/$31.00 ©2018 IEEE

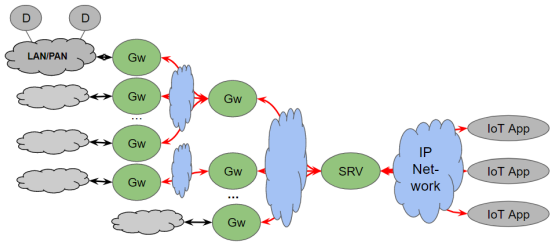

Fig. 1. IoT contextual model.

All the interactions are performed via the Internet. This may cause non-deterministic and tardy delays due to the latency and the network workload variations. Furthermore, the intermediate Network and Middleware strata [3] [4] are made up of static and pre-provisioned silos. This makes their adaptation and reconfiguration at runtime time consuming and costly in terms of delays. Consequently, existing Network and Middleware facilities often fail to dynamically adjust and meet the evolving IoT applications requirements, causing frequent QoS degradation during execution. This work focusses on designing and prototyping procedures that could support the dynamic QoS management of the involved network and gateway facilities in IoT. The proposed approach is based on emerging concepts such as Network Functions Virtualization (NFV), Software-defined networks (SDN) and edge computing. The next section introduces these concepts and discusses the proposed design. Section III describes the prototype architecture and the demo scenario. Section IV concludes the paper.

II. Proposed Approach

This section first introduces the key concepts of the proposed approach. This is followed by the discussion of the high-level architecture.

A. Key Concepts

Notably, concepts such as Network Function Virtualization (NFV) and Software-defined Networks (SDN) and edge computing might aid in addressing the previously described limitations. NFV [5] aims at decoupling network resources from underlying hardware. These resources can be considered at different levels of the communication stack (L3-L7 equipment). NFV defines them in standalone pieces of software that could be rapidly deployed and executed on any generic hardware. More generally, the NFV concept can be envisioned to dynamically deploy NFs in a seamless way (i.e. NFs are inserted into an ongoing communication between two communicating entities). Moreover, these NFs can be packaged into different formats ranging from classical virtualization containers (e.g. virtual machines, application containers) to software modules of an application, a.k.a Applicative Network Functions (ANF). SDN [6] decouples network control and data plane and makes the routing between the NFs by dynamically programming the network resources. When it comes to edge computing [7], it enables streamlining the traffic flow from IoT devices and provide local computing capabilities that could host and execute applications and network functions close (in terms of latency) to the IoT devices. Specifically, provisioning the network entities (including gateways) as Virtualized Network Functions (VNFs) will enable their agile and cost-effective operating. This will allow and ease automation and evolving of the network (e.g. adaptation, resizing) to maintain operative QoS during runtime. On the other side, SDN will ensure the dynamic reconfiguration of the control and the routing planes when making the network evolving during runtime. Finally, close edge nodes will enable hosting and executing the NFs close to the data sources if needed. The ultimate goal is to meet the latency-sensitive requirement of the IoT applications.

B. Proposed Architecture

The proposed approach involves designing, developing and demonstrating generic architectural and behavioral models for self-adaptive management (within the General Controller depicted in Fig. 2) of QoS-oriented network functions (NF) at the different levels of the system (referred as Managed Entity in Fig. 2):

- Taking advantage of the technological opportunities associated with dynamic deployment of virtual or applicative network functions (VNF vs. ANF), i.e. following an NFV-like approach (i.e. VNFs), or based on a modular software architecture (i.e. ANF), together with programmable networks such as SDN networks,
- Taking into account the heterogeneity in terms of runtime environments (Cloud/Fog/Edge), NF packaging formats (VM, Container, SW module), and the programmability (e.g. SDN capacity) of the networks being deployed,
- Ensuring the consistency of the configuration and reconfiguration choices made for each level through appropriate theoretical tools.

Basically, the General Controller (GC) is in charge of sensing the appropriate elements of the Managed Entity (e.g. gateway/server resources consumption, gateway/server performances, etc.), and to make them apply the adaptation actions (e.g. ANF/VNF dynamic deployment) that allow maintaining the application level QoS at the expected level. To do that, the GC interacts with both a VNF orchestrator (such as the ETSI open source MANO [8]) and an ANF orchestrator (such as Puppet[1]), that respectively allow provisioning VNF/ANF in edge nodes. To perform these actions seamlessly for the IoT application, the GC also interacts with an SDN controller (such as Floodlight[2]), that allows configuring the SDN switches of the underlying network. To implement the autonomous feature of its internal behavior, the GC may be based on the IBM autonomic computing paradigm [9] that makes appear the Monitoring, Analysis, Planification and Execution steps of the MAPE-K loop.

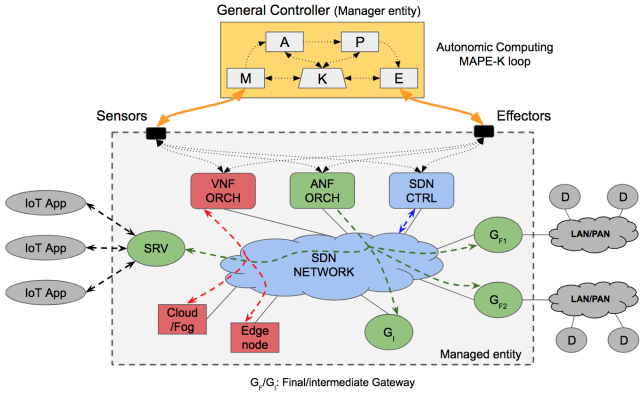

Fig. 2. High-level architecture.

III. Prototype Overview

A prototype was designed, implemented and evaluated for validation purpose. The prototype of the IoT-based business application and its features are first introduced. This is followed by the demo scenario description. Finally, the demo objectives, and the expected lessons learned are discussed.

A. Wildfire Surveillance and Monitoring System

Performed prototype implements an automatic wildfire surveillance and monitoring use case. It is a connected IoT-based system. It enables early fire spotting by: (i) monitoring temperature over the forest using temperature sensors and (ii) detecting suspicious smoke with surveillance cameras. In case of fire, alerts with live streaming over the fire zone are delivered to wildfire management authorities. Wildland fires are increasing in frequency, duration and intensity worldwide. In response to these trends, the use of automatic and connected wildfire surveillance are proliferating over risk area. Unlike human-only monitored ways, such systems simultaneously monitor wider areas and immediately send rapid alarms and streams through the network. Early detection is enabled using pattern-recognition software to detect smoke. Moreover, such

[1]Puppet, https://puppet.com/
[2]Project Floodlight, http://www.projectfloodlight.org/

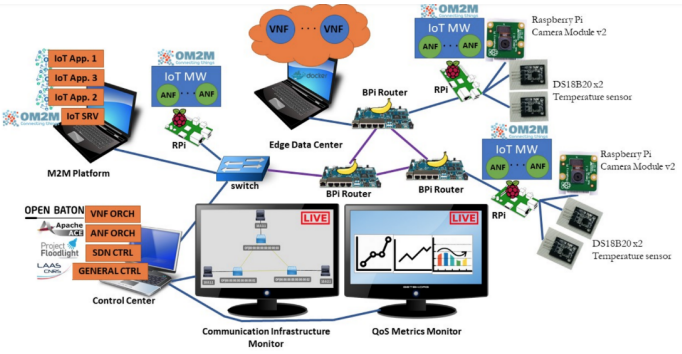

Fig. 3. Demo setup overview.

systems enable recording videos, which can be analyzed later to discern fire source (e.g. campfires, arson), as well as, its evolution on the local flora. However, one of the major limitation of these systems remains in the intermediate network. Indeed, these systems are usually deployed in non-urban areas. This means that the used networking facilities are often limited and not that capable. All the more so as such applications are very demanding in terms of latency and bandwidth, especially during fire detection events. Obviously, this is not always sustainable considering wildland zones where wired Internet is poor and telco satellites coverage is often limited. This often resulted in major failures of such systems during critical period of time. Investigation highlighted that most of these crashes were due to communication infrastructure overloading over the area. More generally, SLAs, such as 150 msec round-trip latency for multimedia services, become difficult to meet during the fire event when the very limited network facilities are likely to be overloaded. The rest of the section shows that the proposed prototype enables maintaining operative QoS thanks to the use of NFV, SDN and edge resources.

B. Demo Setup and Description

The planned demo setup is depicted in Fig.3. In the east-bound, DS18B20 temperature sensors and Raspberry Pi Module V2 cameras will be used for the prospective fire spotting. In the westbound, the several IoT applications are deployed over M2M-enabled server. The intermediate networking domain is made up the following devices:

- Raspberry Pi nodes hosting the required getaways behind the used IoT devices. These modular gateways may host ANFs,
- Dell Laptop implementing an edge node close to the data sources and hosting the required VNFs,
- D-Link regular network switch,
- Banana Pi routers with Open vSwitch software acting as SDN switches.

In addition to these, an additional laptop implementing the prototype control plane hosts and executes the Floodlight SDN controller, the OpenBaton VNF orchestration, the Apache Ace ANF orchestrator, and the General Controller introduced in the Section II. Finally, two live dashboards will be shown to the audience. The first monitor displays the live evolution of the IoT applications QoS through an unexpected event such as fire detection and streaming camera activation. The second monitor displays the instantaneous and autonomous changes of the network topology and configuration triggered by the control center to maintain the QoS in the operative range. The reader should note that some software will be used during the demo to simulate datasets and events such as fire outbreak. (Un)Deployment of VNFs in the edge node, dynamic reconfiguration of the SDN switches, increase of the bandwidth are among the prospective changes that audience can witness during the demo.

C. Expected Lessons Learned

Thanks to the live display boards, the demo will enable audience to assess the dynamic and self-adaptive mechanisms supported by the prototype to support and maintain floating QoS. More generally, the prototype will show the feasibility of such approach and validate the gain from the use of emerging concepts such as NFV/SDN and edge computing for such research issues. Provisioning dynamically network functions as virtualized entities (e.g. VNFs, ANFs) could be performed at all layers (i.e. application, middleware, transport, network) and brings agility. Finally, it is also expected from this demo to trigger relevant discussions between attendees around economic and commercial potential of such approach, keeping in mind the tradeoff between dynamicity and agility from one side and the virtualization, as well as, orchestration overhead on the other side.

IV. Conclusion

This paper motivates and describes a communication infrastructure prototype that uses NFV, SDN and edge computing to handle and manage floating QoS in dynamic and demanding contexts such as IoT systems. The prototype objectives and expected results are discussed. The related demo implements a realistic wildfire surveillance and monitoring use case.

References

[1] ETSI, "Machine-to-Machine communications (M2M); Functional architecture" *ETSI*, ETSI TS 102 690, 2013. [Online]. Available: http://www.etsi.org. [Accessed: Jan. 25, 2017].

[2] OneM2M, "oneM2M functional architecture," *OneM2M*, oneM2M-TS-0001, 2014. [Online]. Available: http://www.onem2m.org. [Accessed: Jan. 25, 2017].

[3] I. Lee, and K. Lee, "The Internet of Things (IoT) : Applications, investments, and challenges for enterprises," Business Horizons, vol. 58, no 4, pp. 431-440, 2015.

[4] F. Li, M. Vogler, M. CLAEENS and S. Dustdar, "Efficient and scalable IoT service delivery on cloud," In IEEE International Conference on Cloud Computing (CLOUD), 2013, pp. 740-747.

[5] B. Han, V. Gopalakrishnan, L. Ji, and S. Lee, "Network function virtualization: Challenges and opportunities for innovations," IEEE Communications Magazine, vol. 53, no 2, pp.90-97, 2015.

[6] H. Kim, and N. Feamster, "Improving network management with software defined networking," IEEE Communications Magazine, vol. 51, no 2, pp. 114-119, 2013.

[7] W. Shi, J. Cao, Q. Zhang, Y. Li, and L. Xu, "Edge computing: Vision and challenges," IEEE Internet of Things Journal, vol. 3, no 5, pp. 637-646, 2016.

[8] ETSI, OSM, "Open Source MANO," OSM home page, 2016.

[9] J. O. Kephart and D. M. Chess, "The vision of autonomic computing," Computer, vol. 36, no 11, pp. 41-50, 2003.