Inspiring from SDN to Efficiently Deploy IoT Applications in the Cloud/Fog/IoT ecosystem

Nada Chendeb, Nazim Agoulmine, Mohammad El-Assaad

Inspiring from SDN to Efficiently Deploy IoT Applications in the Cloud/Fog/IoT ecosystem

Nada Chendeb[1], Nazim Agoulmine[2], and Mohammad El-Assaad[1,2]

[1] Lebanese University, Faculty of Engineering
nchendeb@ul.edu.lb, mdalassaad@gmail.com
[2] University of Evry Val d'Essonne - Paris Saclay University
nazim.agoulmine@ibisc.univ-evry.fr

Abstract

Billion of devices are connected to the internet nowadays, more are coming in the future. Cisco assumes that 50 billion devices will be connected by 2020. It is quite difficult to manage and control the exchange of the huge amount of data generated from those connected devices. Thus the need for a new more intelligent Internet of Things (IoT) architecture for large scale networks. Based on the above needs and challenges of the IoT, Software Defined Networking (SDN) seems to be an excellent key solution. Thus, using the concepts of SDN, we aim to reduce the complexity of traditional IoT networks, this means that recent IoT networks will be more programmable, managed and controlled by software. In this work, we provide an architectural model combining SDN and IoT, and a mathematical formulation of the controller placement problem as a linear integer program optimization. At the end, we present some simulation results to show the advantages of this integration.

1 Introduction

Connected physical devices can communicate and send information to each other over the Internet, they can be remotely monitored, programmed and controlled. IoT applications running on the top of such infrastructure, have numerous applications in verticals such as commercial, industrial, health, etc. IoT can also be very heterogeneous in terms of functionalities and capacities. The number of deployed IoT objects is increasing significantly as a consequence the generated data from these objects is increasing significantly constituting the so called Big Data. The characteristics of diversity, dynamics of such environment associated with the generated Big Data introduce hard challenges in order to design efficient and cost-effective IoT architecture solutions. This architecture should rely on not only efficient computing infrastructure but also communication networks infrastructures that should be more dynamic, efficient and more scalable to meet up the introduced challenges (performance, cost, adaptability, etc.). We believe that SDN is a promising technology that could help IoT architecture to address these challenges. In this paper, we propose a novel architecture called Software Defined-Internet of Things (SD-IoT) architecture that aims the convergence of the two concepts. We also present a mathematical formulation to optimize the placement of SDN controllers in the IoT infrastructure. We then highlight how the proposed architecture and optimization formulation permits to improve the performances of the converged system. The rest of this paper is organized as follows: Section 2 presents an overview of IoT and SDN technologies. It also presents the current state of the art related to the integration/convergence of the SDN and the IoT. In Section 3, we propose a novel architecture that integrates IoT and SDN concepts. In section 4, we present our mathematical formulation of the controller placement problem in the form of an optimization algorithm and then some simulation results are presented. Finally a conclusion and future works are given at the end of this paper.

2 IoT and SDN

2.1 Challenges of IoT Applications Deployment

The classical approach to deploy IoT applications is to entirely deploy them in Cloud data centers. This approach makes the design of the applications simple but makes it difficult to fulfill all the non functional requirements such response time, and data rate. In this paper, we propose to leverage IoT architecture with SDN concepts to allow a better programmable instrumentation of the infrastructure based on the requirements of IoT applications at any time. This architecture is called Software Defined - Internet of Things Architecture (SD-IoT). With this approach, we envision that IoT applications could interact directly with infrastructure in order to request resources and/or deploy its own components.

2.2 Overview of IoT and SDN

IoT architectures have been defined in order to introduce the necessary flexibility to interconnect various devices. From the conducted state-of-art we identify several models. The basic model is a 3-layer consisting of Applications, Networks, and Perception Layers. Recent literature introduces more abstraction models, however, some other models have been proposed to add more abstractions to the IoT architecture.
Network devices are usually from different providers exhibiting proprietary and heterogeneous control functions. Control plane and data plane are both packed inside the networking devices. This packaging increases the complexity and reduces the flexibility. SDN is a solution that allows to overcome these limitation providing a software-oriented control plane (control) that is decoupled from the data (forwarding) plane. SDN introduces a three layer architecture: (a) Data plane or Device layer, (b) Control plane and (c) Applications layer. The application layer expresses the customer's needs, and communicates with the controller via northbound API. The Control plane layer is considered as the brain of the network. The controller communicates with the physical devices in the data plane via southbound APIs, e.g. OpenFlow protocol. There exist many software controllers in the market such as Ryu, Opendaylight (ODL)[3], Floodlight, NOX[4], etc.

2.3 State of art inter-operating SDN with IoT

The heterogeneity and the complexity of IoT devices will require a new IoT architecture with SDN capabilities to manage them and to improve the performance of the whole network. If the networks are not prepared, the flood of IoT where a lot of traffic is generated could leave the network paralysed [6].

Authors in [2] propose a framework for managing connected devices and configuring the network dynamically based on SDN. SDN is used to solve the problem of continuous changes in the network by decoupling the control plane from the data plane. The approach resembles routing table used in routers. The routing information is stored centrally. When the topology needs to be changed, the SDN controller pushes the new routing information to the nodes. Thus, it can reconfigure the system without redeploying.

In [1], authors propose an SDN-based architecture to manage the diversity of devices and networks. Their solution is based on the utilization of dockers running on the connected devices. In this architecture, IoT devices running Docker communicate together via SDN based switches and they are monitored by an SDN controller. Authors implemented their approach to validate

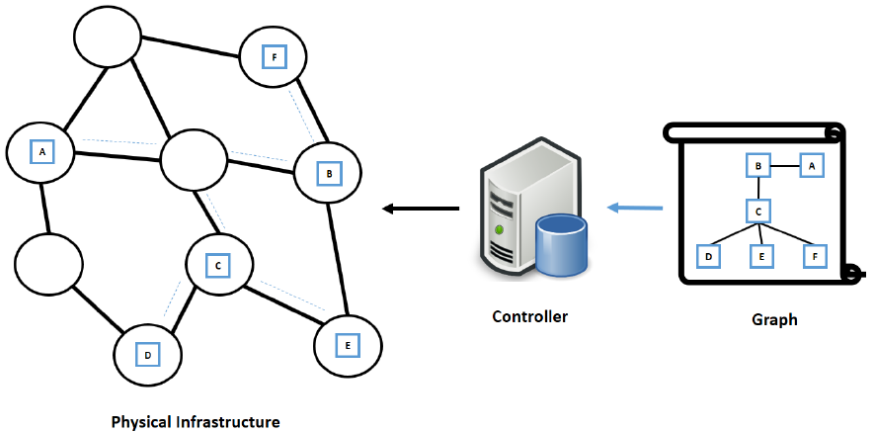

Figure 1: SDN inspired new IoT architecture

the possibility to overcome the heterogeneity of the devices as far as they are able to support containerization.

Authors in [5] propose an SDN based approach in Smart Cities. They developed a testbed highlighting how programmable SDN infrastructure can be used to significantly simplify the on-boarding and provisioning of end-to-end IoT services in a context of multi-tenant networks.

In another contribution [7], authors aim to fuse three concepts in a global IoT architecture: Mobile Edge Computing (MEC), SDN and Network Function Virtualization (NFV) to show how they can achieve better MEC employment and extend the MEC concept to provide Software-Defined Fog-enabled IoT gateways (SDF-Gateways).

Authors in [6] present a simple and general SDN-IoT architecture with NFV implementation to address the challenges of the IoT. The data layer comprising of SDN switches is not responsible of the decision making of the information being received/transmitted by/to the sensing layer rather they leave the decision making to the control layer through a southbound APIs such as OpenFlow. One of the key objectives of the proposed architecture is to replace the traditional gateways with SDN-gateways.

3 Our Proposed Software Defined IoT Architecture

SDN creates a high level of abstraction between the application layer and the physical layer. This centralized concept allows the controller to deal with any IoT application as a graph. The role of the controller is to guarantee the instantiating of this graph in the physical infrastructure while respecting the application requirements. Figure 1 illustrates our idea. The IoT Application specification is transformed into a graph specification of interconnected application components to submit to the controller to deploy, along with a set of requirements in terms of processing and communication resources. The controller will be in charge of the management of the deployment and execution of the application components in the physical infrastructure (the sensors and the network devices). The controller provides the required level of abstraction between the requested IoT application and the real implementation of the application components and communication links in the physical infrastructure.

We suppose that SDN concept applies also to the IoT devices themselves i.e. they are able to

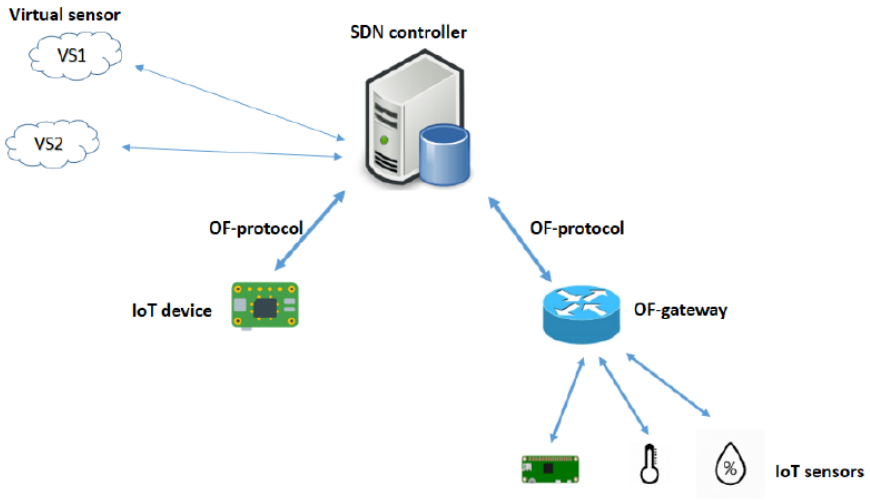

Figure 2: SD-IoT Devices connected to the controller

communicate with the controller via the OpenFlow protocol. In addition, to take into account the possibility that some IoT devices may not support the OpenFlow protocol, a gateway can help these devices to communicate with the controller by translating proprietary protocol into OpenFlow. The controller can directly update the flow table for those devices to trigger or stop the transmission of some particular data. In this way, the controller could have global control on the IoT devices and manage the generated traffic with a very coarse granularity. Figure 2 illustrates a network where IoT devices with SDN capabilities are connected directly to the controller while others are connected via an OpenFlow gateway (OF-Gateway). With this approach, the infrastructure is now decoupled from any specific IoT application. The client of the requested IoT application doesn't need anymore to worry about whether the infrastructure will be able to support the non functional requirements of its IoT application but these requirements will be taken into account by the infrastructure in a programmatic way. The physical infrastructure will be virtually shared between different IoT applications allowing for a better resource utilization.

4 Analytical Model of the Controller Placement Problem

In this section, we address the problem of controllers placement in the IoT infrastructure. The objective is to find the minimum number of controllers so that the response time of controllers is less or equal to a given delay bound d. The response time of the controller is an important QoS parameter. It is affected by two factors: (a) The round trip time delay between the controller and a node and (b) The sojourn time in the controller which depends on the service capacity and the load of the controller.

4.1 System Model

The considered system model is a set of OpenFlow switches, gateways and IoT devices deployed in a particular geographical area. The problem is to determine how many controllers should be

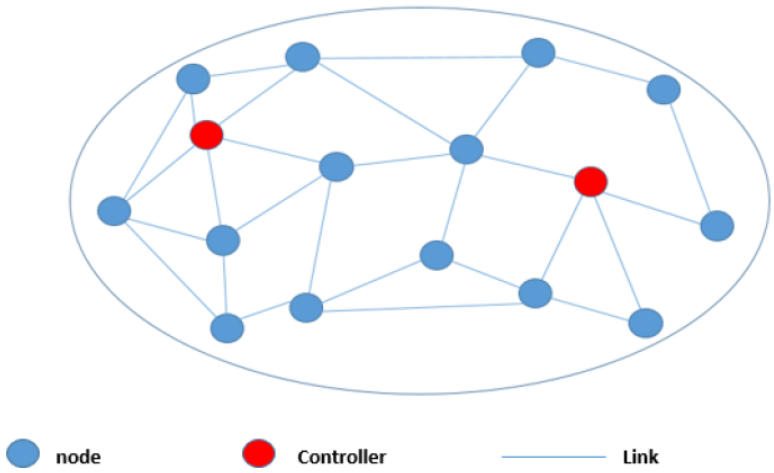

Figure 3: System Model

deployed and where so the response time constraint of the controllers is respected. We assume that a controller can be co-located with any node and share the same channel links with this node. Figure 3 illustrates our system model. The network is modeled as a graph G(V,E) where V represents the set of nodes and E represents the network links between nodes. Each link is associated with a network delay. The number of Nodes is N in the shared infrastructure. So we have V =v_1,v_2,...,v_N and v_i, $1 \leq i \leq N$, denotes the i^{th} node.

4.2 Response Time and Delay Formulation

- Network Delay Model: The network delay denoted as D_{ij}^N is the sum of the transmission delay and the propagation delay in the network from one node to the associated controller. $D_{ij}^N = D_{ij}^T + D_{ij}^P$ where D^P is the propagation delay and D^T is the transmission delay.
- Controller Response Time Model: Each controller can be modeled as an M/M/1 queuing model. Suppose packets are generated according to a Poisson process. Suppose also that all the controllers have a fixed service rate μ. Let S_j denotes the set of nodes associated to the controller j and λ_j denotes the total packets arrival rate to the controller in site j, which can be calculated as follows:

$$\lambda_j = \sum_{i \in S_j} \lambda_j^i \tag{1}$$

where λ_j^i is the request rate of node i to the controller j [packets/s]. But arrival rate of nodes is independent of the controller site itself, and we can consider that:

$$\lambda_j = \sum_{i \in S_j} \lambda^i \tag{2}$$

According to the queuing theory, the expected mean response time R_j of controller in site j can be calculated as follows:

$$R_j = \frac{1}{\mu - \lambda_j} = \frac{1}{\mu - \sum_{i \in S_j} \lambda^i} \tag{3}$$

4.3 Placement Problem Formulation

The optimization problem is an NP-hard problem, which is very difficult to solve when the size of the network becomes very large. We propose instead a heuristic algorithm based on iterative approach and Dijkstra's algorithm for the least cost path problem. This algorithm aims to actively increment the number of controllers in the network by satisfying all the required constraints for all nodes in the network. The proposed algorithm is illustrated in Algorithm 1:

Algorithm 1 Controllers Placement Algorithm

Input $G = (V, E, M), \delta, \mu$
Output X, Y
$R \leftarrow V$
$C \leftarrow V$
D = Dijkstra(V, E)
while $R \neq \emptyset$ **do**
    for $j \in C$ **do**
        $S_j \leftarrow \emptyset$
        $D_j \leftarrow 0$
        $\bar{t_j} \leftarrow 0$
        while $\bar{t_j} \leq \delta$ **do**
            $i \leftarrow NearestNode(V, j, D, S_j)$
            $D_j \leftarrow D_j + 2D_{ij}^N$
            $\bar{t_j} \leftarrow \frac{D_j}{|S_j|+1} + \frac{1}{\mu-(\sum_{n \in S_j} \lambda^n + \lambda^i)}$
            if $\bar{t_j} \leq \delta$ **then**
                $S_j \leftarrow S_j \cup i$
            end if
        end while
    end for
    $j = Argmax_j \mid S_j \mid$
    $y_j = 1$
    $C \leftarrow C - j$
    for each node i within S_j **do**
        $x_{ij} = 1$
        $R \leftarrow R - i$
    end for
end while
Return X,Y

The network is represented by $G = (V, E, M)$ where V is the set of nodes $V = (v_1, v_2; ..., v_N)$, E is the link's delay between the nodes, $E = (e_{11}, e_{12}, ..., e_{NN})$ and M represents the arrival rate packet-in requests to controller by each node. $M = (\lambda^1, \lambda^2, ..., \lambda^N)$. μ is the service rate of each controller, and δ is the delay boundary that the average response time of each controller should not exceed. G, μ and δ are the inputs of the algorithm. R represents all the nodes that are not yet associated to any controller, and C represents the candidates sites for controllers. Initially, we consider in our algorithm that all the nodes are not yet associated to any controller and $R = V$ and all the nodes could be considered as possible candidate sites i.e. $C = V$.
We start our algorithm by finding the shortest path from any node to any other node in the network using Dijkstra's algorithm. Then the algorithm enters in a loop until all the nodes are

associated to at least one controller. The algorithm calculates all the possible candidate sites, and the set of nodes that can be connected to them, starting always with the node that is the nearest one to the controller. At each iteration, the algorithm computes the average response time to verify that is still less than the threshold δ. The function $Nearest_Node(V, j, D, S_j)$ selects the node i that has the minimal cost path to the controller j. S_j is the resulting associated nodes to the controller located in site j. Then, the algorithm calculates and checks the average response time $\bar{t_j}$. After finding the possible associated nodes to each controller, the algorithm chooses the one that has the maximum number of nodes associated to it, using the function $\text{Argmax}_j \mid S_j \mid$, and updates the output variable Y by setting $y_j = 1$ to indicate that there is a controller on the site j. The algorithm removes the chosen controller from the possible candidates list C, and updates the output X to indicate the nodes that are now associated to this controller in site j by setting $x_{ij} = 1$.

4.4 Simulation and Performance Evaluation

In order to evaluate the performance of the system, we aim to study the effect of δ. We generated a network of N=120 nodes, the links between the nodes are randomly generated following uniformly distributed pseudo random integers between $mindelay$ and $maxdelay$. We fixed μ to a high value enough to cancel the effect of the diversity in nodes' requests in the network (λ_i) i.e. each controller has high performance capacity (very small service time) and is able to process all the requests in time (i.e. $\mu \gg \lambda$). This is because we need to study only the effect of Network delay.

In the first scenario, δ is set to $mindelay$, and therefore the number of required controllers is 120, each node shall host its own controller to satisfy the delay constraint. In scenario 3, the value of δ is very high ($maxdelay$) and therefore one controller will be sufficient to satisfy the constraints of all nodes. In scenario 2, the value of δ is $0.4 * maxdelay$, and the number of needed controllers is 5. Figure 4 highlights the results of our algorithm. This figure shows that when δ is very low, each node will host its own controller. While increasing δ, the number of required controllers decreases and converges towards only one.

5 Conclusion and Future Works

In this work, we have proposed an SD-IoT architecture where SDN controllers are in charge of instantiating and controlling applications components in the infrastructure. We have proposed an algorithm to the controller placement problem in this SD-IoT architecture aiming at satisfying the delay constraints between IoT devices and the controllers. We formulated the problem, modeled the average response time of a controller taking into account the network delay and the average service time in the controller. We have performed some simulation to highlight the effect of δ on the number of controllers to deploy. In future works, we aims to perform more simulation to study the effect of other parameters such the sampling rate of the IoT devices and the service time of the controller that depends on its architecture and price.

Acknowledgments

This research was partially funded by the IBISC laboratory during an internship, it was also supported by the Lebanese University and CNRS Lebanon.

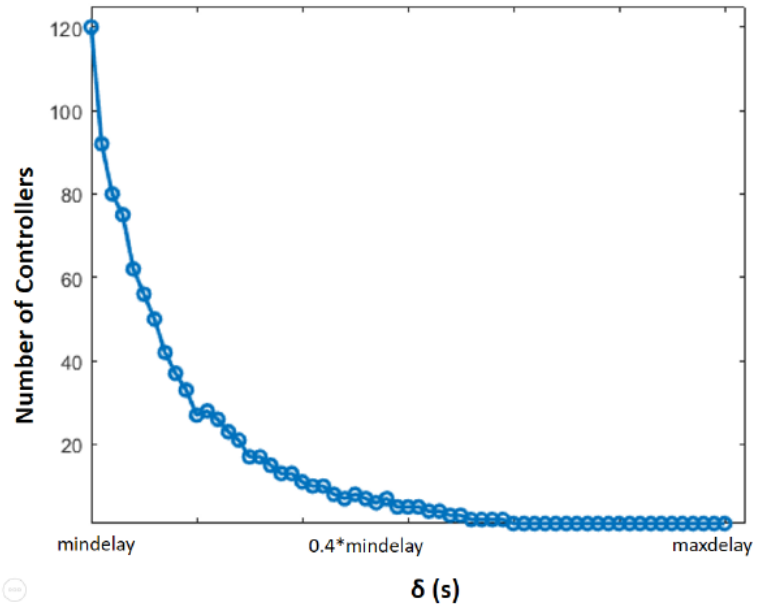

Figure 4: Number of controllers vs δ

References

[1] I. Bedhief, M. Kassar, and T. Aguili. Sdn-based architecture challenging the iot heterogeneity. In *2016 3rd Smart Cloud Networks Systems (SCNS)*, pages 1–3, Dec 2016.

[2] H. Huang, J. Zhu, and L. Zhang. An sdn based management framework for iot devices. In *2014 China-Ireland International Conference on Information and Communications Technologies (CIICT 2014)*, pages 175–179, June 2014.

[3] Adrian Lara, Anisha Kolasani, and Byrav Ramamurthy. Network innovation using openflow: A survey. *Communications Surveys and Tutorials, IEEE*, 16:493–512, 03 2014.

[4] B. A. A. Nunes, M. Mendonca, X. Nguyen, K. Obraczka, and T. Turletti. A survey of software-defined networking: Past, present, and future of programmable networks. *IEEE Communications Surveys and Tutorials*, 16(3):1617–1634, 03 2014.

[5] L. Ogrodowczyk, B. Belter, and M. LeClerc. Iot ecosystem over programmable sdn infrastructure for smart city applications. In *2016 Fifth European Workshop on Software-Defined Networks (EWSDN)*, pages 49–51, Oct 2016.

[6] M. Ojo, D. Adami, and S. Giordano. A sdn-iot architecture with nfv implementation. In *2016 IEEE Globecom Workshops (GC Wkshps)*, pages 1–6, Dec 2016.

[7] O. Salman, I. Elhajj, A. Kayssi, and A. Chehab. Edge computing enabling the internet of things. In *2015 IEEE 2nd World Forum on Internet of Things (WF-IoT)*, pages 603–608, Dec 2015.