SHIOT: A Novel SDN-based Framework for the Heterogeneous Internet of Things

Hai-Anh Tran, Duc Tran, Linh-Giang Nguyen, Quoc-Trung Ha and Van Tong
School of Information and Communication Technology
Hanoi University of Science and Technology, Vietnam
E-mail: anhth@soict.hust.edu.vn

Abdelhamid Mellouk
Networks and Telecommunications Department
University of Paris-Est Creteil, France
E-mail: mellouk@u-pec.fr

Keywords: software defined networking, internet of things, Lagrange relaxation

Received: February 26, 2018

A new measurable and quantifiable world is created by the Internet of Things. However, the variety of IoT components, i.e., devices, access technologies and applications, which are deployed on the same core infrastructure with a common network policy have led to an unexpected issue of heterogeneity. Such issue directly dismisses the interoperability in the IoT, and hence, significantly decreasing the QoS of a given IoT service. In this paper, we develop a SDN-based framework, called SHIOT to address the above challenge. SHIOT relies on the ontology for examining the end-user requests and applies a SDN controller to classify flow scheduling over the task level. We also utilize the Lagrange relaxation theory to optimize the routing mechanism. Extensive experiments demonstrate that SHIOT is able to support stressed networks and offers a significant advantage over the traditional framework that is integrated without SDN.

Povzetek: Zasnovana je izboljšava heterogenega interneta stvari na osnovi SDN okvira in ontologij.

1 Introduction

The practical involvement of the Internet of Things (IoT) in our world is indisputable. The IoT contributes to the world's economy and improves the quality of life. There are 9 billion networking devices and the number is expected to reach 24 billion by 2020 [11]. The devices can be sensors, actuators, cameras, smartphones, which are placed in different access networks using different access technologies (e.g., bluetooth, wifi, zigbee, cellular, MANET). There are also a variety of IoT applications, implemented on top of the access networks. However, all these various elements, i.e., devices, access technologies, applications have to be built on a common network infrastructure with traditional equipments and communication protocols. The latter is not designed to support high-level of scalability, high amount of traffic and mobility in different IoT tasks. For this reason, it is required to develop diverse policies for the core network to adapt to the heterogeneous IoT. This challenge can not be accomplished without the help of a technology solution, namely Software Defined Networking (SDN) [12, 27].

SDN is considered as a network architecture that is able to enhance the flexibility of traditional core network. The idea of programmable networks facilitates network evolution, in which the forwarding devices are decoupled from the so-called control layer. This structure makes the behavior of the core network more adaptive to the quality of service, required by different access technologies, applications and devices. In addition, the centralized architecture in SDN gives the capability of collecting data and uses this information to improve network policies instead of manually transforming these high-level policies into low-level configuration commands.

In this paper, we proposed a SDN-based framework, named **SHIOT** (**S**dn-based framework for the **H**eterogeneous **IoT**) to tackle the heterogeneity issue in the IoT. The goal is to provide a transparent bridge between user-interface layer and other low-level layers in order to enhance the user convenience, while reducing the low-level complexity. To this end, we have developed an open ontology to analyze semantics of incoming requests. We also introduce a global optimization for routing layer of the core network by using a heuristic algorithm based on Lagrange relaxation theory. The characteristics of SDN help us to the above requirements without altering the untouchable core Internet network.

The paper is structured as follows: Section 2 presents the background and related works. Section 3 analyzes and describes in detail the SHIOT framework. Section 4 shows the experimental results. Finally, section 5 is dedicated to conclusions and future works.

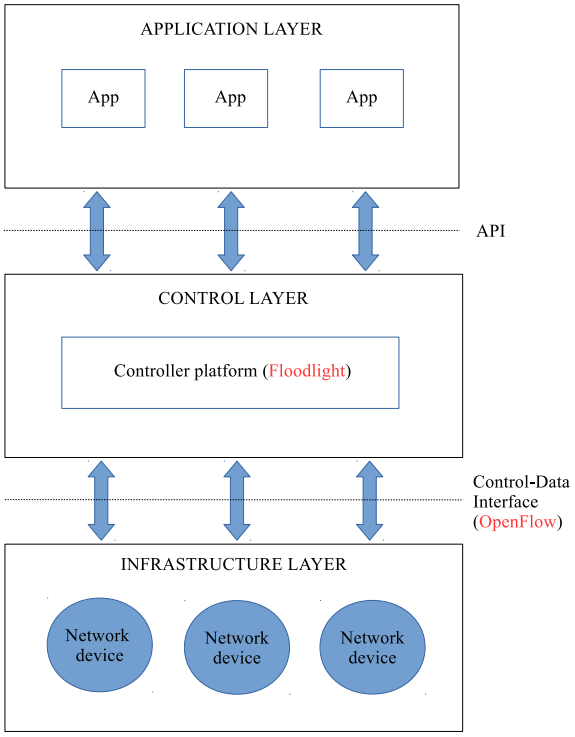

Figure 1: A general SDN architecture

2 Background and related works

2.1 Software defined networking

Over the past decade, the need for services that span multiple IoT application domains is growing in order to realize the efficiency gains, promised by the IoT. End-users, however, have to face the heterogeneity issue, which arises when a wide variety of devices, wireless communication solutions, and access technologies are implemented in the IoT. SDN is considered as a good solution to handle such issue because of its centralized and programmable controller that enables simple programmatic control of the network data-path.

The main idea of SDN is to separate the control and data planes. The control plane creates and modifies the forwarding rules, which are subsequently sent to network devices. The network devices (e.g., switches and routers) in turn, just forward packages based on the received rules. The controller is therefore considered as control logic precept to examine the overall network behavior. Using SDN-based controller, network administrators can easily program, manipulate and configure network protocols in a centralized way. Fig. 1 shows a general architecture of SDN, which consists of three main layers: Application layer, Control layer and Infrastructure layer.

In order to deploy a SDN architecture, it is essential to have an interface that ensures the communication between the data and control plane. Such interface is called Southbound Interface (SBI) and should be standardized. SBI defines a protocol to facilitate the diversity of network devices and controller softwares. There are variety of SBI protocols (e.g. ForCES [8]), but the most typical one seems to be OpenFlow [19]. An OpenFlow-enabled networking switch needs to maintain a forwarding flow table that has three types of information: rules, actions associated with each rule, and the statistics that count the number of packets and bytes for the flow. The OpenFlow-enabled switch aslo creates a secure channel to communicate with the SDN controller. We have chosen OpenFlow in the present work because it is capable of reducing management complexity, handling high bandwidth and implementing new policies when required. A detailed description on the advantages of OpenFlow can be found in [19].

2.2 SDN-enabled IoT: a review

The section gives a brief overview on the applicability of SDN in the IoT to improve the system performance and overcome the heterogeneity issue.

Huang et al. [13] developed an M2M based framework, which involves M2M nodes, a gateway to handle the nodes that does not support M2M protocol and a controller to carry out network management. Once the routing table is changed, the controller notifies such change and sends it to the different nodes. Therefore, the durability of the IoT network is improved. However, the authors only applied the same routing policy for all types of flows and did not consider their QoS.

Martinez-Julia and Skarmeta [18] utilized SDN that allows different objects from different networks to communicate with each other using IPv6. The IoT controller is also inserted to simplify the control operations of the various objects. After establising forwarding rules, such controller sends these rules to the SDN controller and other networking devices. Such mechanism enables the communication between the herotereous objects. Li et al. [17] and Omnes et al. [21] exploited SDN to contruct a framework that meets the requirement of the diversity and dynamics in the IoT. Vilalta et al. [28] developed an end-to-end orchestration for IoT services using an SDN/NFV-enabled edge node under SDN.

In [15], Jararweh et al. introduced an architecture model, namely SDIoF based on the combination of SDStore [5] and SDSec [2]. This architecture consists of three main components: 1) The physical layer where all the assets and hardware devices in the system reside. This layer is classified into several clusters such as sensor network cluster and database pool cluster; 2) The control layer acts as a middleware. It involves IoT controller, SDN controller, SDStore controller, and SDSec controller that are entirely software-based controllers to abstract the management operations from underling physical layer; 3) The application layer combines many fine-grained user applications, which simplify the end-user's accessing to the stored data through the Northbound APIs (N-APIs). Unfortunately, the authors in [15, 17, 21, 28] only gave the general and theoretical ideas. Their proposals have not been proven and assessed

through experiments.

Wei et al. [25] introduced a hash-based distributed strategy while integrating SDN into IoT in order to solve the problem of storage limitation of forwarding nodes. In their work, the multi-dimension selection method was utilized for finding the suitable storage. The hash space was formed by using the IoT data flow. However, the authors did not consider the QoS requirements related to the different IoT applications/services.

Chakrabarty et al. [3] developed the so-called Black SDN with the focus on various security issues. Black SDN was used to secure both the meta-data and payload within each layer of an IoT communication packet. The authors considered the SDN centralized controller as a trusted third party to ensure secure routing and optimize the system performance management. Olivier et al. [20] proposed a SDN-based architecture for the IoT that consists of multiple domains, where the SDN controllers are installed. The authors agued that such architecture is able to guarantee the security of the entire network. However,there is no simulation or evaluation to justify their architecture and outcomes. Sharma et al. [26] introduced DistBlockNet, a distributed secure SDN for the IoT, where blockchain technology was exploited to verify a version of the flow-rule table. However, in their experiments, they did not consider the average end-to-end delay, the most important QoS parameter that has a significant impact on the user experience while running an IoT service.

It is clear that SDN is an emerging architecture, which is able to facilitate network management in order to improve network performance and monitoring. However, there is a lack of evaluation and testing to assess its use in the IoT. This paper aims to address this question by developing SH-IOT framework to bridge the gap between end-users and the underlying layers. SHIOT will be discussed in detail in the following section.

3 The proposed SHIOT

This section presents SHIOT framework, a layered SDN controller that acts as a middleware to bring the transparency to users from the top layer to the bottom ones. The controller (Fig. 2) is developed using an open source platform called *Floodlight* [1]. We decided to use the Floodlight Open SDN Controller due to the fact that it is an enterprise-class, Apache-licensed, Java-based OpenFlow Controller. Floodlight is supported by a community of developers including a number of engineers from Big Switch Networks. Although the project is discontinued, its latest version is sufficient for our development.

Since the IoT scenario is very dynamic, we create a database, called *State info DB* to store on the fly the state information of the network such as its topology, the link state, the joining or leaving nodes, etc. The end-user's commands that determine what he/she needs are placed at the highest layer of abstraction. We also construct a web service that provides RESTful APIs, so that the end-user may use such service with any platform and on any device. The transparency, considered in this paper is defined as the level of independence from applications, devices and networks, which are used to accomplish the required tasks. For example, the user's command is to count the number of people in Room01. The controller analyzes this command and indentifies the relevant service. Such service can be either to capture the video in Room01 and count the number of people inside or to activate the counting sensor on the door of the room. The devices can be either the camera or counting sensors. In other words, the role of SDN controller is to map the devices to a specific application in order to perform the couting process. Lower layer then selects network and path to route the data flows. Finally, all these decisions are sent down to Communication layer, and installed on the selected devices. The operation of Communication layer has been processed by a network emulator, known as *mininet* [6] in our testbed.

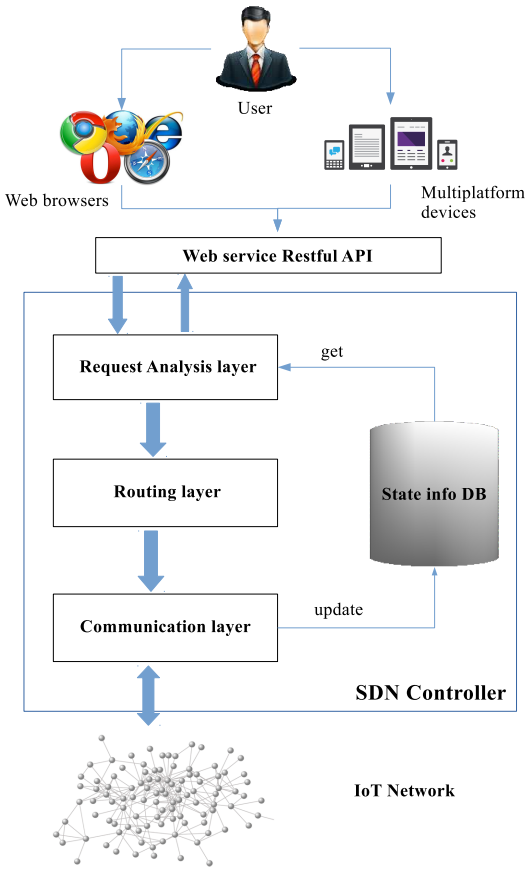

Figure 2: The proposed SHIOT framework architecture.

3.1 Request analysis layer

As mentioned above, there is always a challenge in the IoT due to the diversity in access networks and devices. Hence, the Request Analysis layer is used to provide an abstract

layer to end-users, so that the IoT may work independently from the underlying layers. In other words, the Request Analysis layer bridges the gap between user requests and underlying networking devices. The present work constructs an ontology and utilizes semantic technologies to describe the IoT context as well as the devices and their characteristics. We set our focus in the IoT that is deployed in E-healthcare system. Fig. 3 shows the constructed ontology, which involves three main classes as follows.

1. **Applications**: We considers five different healthcare applications:
   - Monitoring: This is used to capture and record the various healthcare indicators including the physiological (i.e., ECG, EMG, EEG), chemical (sweat, glucose, saliva), and optical (oximetry, the properties of tissues) metrics.
   - Therapeutic: The goal is to monitor the treatment of a given disease. This consists of medication (drug delivery patches), stimulation (chronic pain relief) and emergency (defibrillator).
   - Fitness and Wellness: The application aims to observe the motion and location indicators such as physical activity, calorie count, GPS information and indoor localization.
   - Behavioral: This application is used to maintain regular surveillance over the patient's activities (fall, sleep, exercise), emotions (anxiety, stress, depression) and diet (calorie intake, eating habits).
   - Rehabilitation: This application is used to monitor the rehabilitation of patients like speech (language development) and camera (technology for blinds).
2. **Devices**: There are two main types of devices, i.e., on-body contact sensors and peripheral non-contact sensors.
3. **Positions**: The positions where the devices locate include the laboratory, operating rooms, casualty rooms, consulting rooms, day rooms, emergency rooms, pharmacy, high dependency unit, maternity ward.

Based on the above classes in the ontology, the *Positions* and *Devices* are used to determine the sender and receiver nodes. We also assign a predefined maximum latency to each requested application in *Applications* class. This assignment is based on [14, 23, 22]. In order to validate the scalability of SHIOT, the current ontology includes 250 devices, deployed in 60 different rooms. However, such ontology can be extended and customized to be suitable for the use of any other organizations.

The delta delay Δ_{delay}, that was selected as the QoS metrics in this work, will be one of the three outputs, extracted from this layer. The two remaining outputs will be the sender and receiver nodes. Specifically, we construct two modules in the Request Analysis layer. As illustrated in Fig. 4, the *User expression analysis* module simply scans the expression text, sent from the various applications and detects keywords that are related to the constructed ontology. The *Ontology analysis* module in turn uses these keywords as input and relies on the *Ontology data* to determine the appropriate applications and the forwarding IoT devices. Finally, the outputs, i.e., Δ_{delay}, *the sender node* and *receiver node* will be sent to the Routing layer.

3.2 Routing layer

Based on the results, computed in the Request Analysis layer, the role of the Routing layer is to find an optimal path to route the data from the selected source to destination nodes. This is to ensure that the latency time is not larger than a predefined threshold Δ_{delay} and thus, optimizing the cost function.

To this end, we implemented a routing algorithm that is based on the concept of aggregate cost. The optimal cost is found using the Lagrange relaxation theory. Such algorithm offers a significant advantage because it can give a lower bound on the theoretically optimal solution. The experimental results showed that the difference between the obtained cost and the lower bound is naturally quite narrow. In addition, the proposed routing algorithm takes into account the trade-off between the end-to-end (E2E) delay and quality of the found path. Gary et al. [10] proved that a routing problem is NP-complete in the case where the number of QoS metrics that should be minimized is more than or equal to 2. Therefore, the paper tries to define a simpler problem instead of tackling the more complex problem. Particularly, the delay along the path should be acceptable, and the cost of the path should be as low as possible. The intuitive motivation of the routing task is to find a path that is minimal in terms of cost, provided that the delay is under a given bound. The delay bound is determined in the Request Analysis layer and the routing problem is formulated as a Delay Constrained Least Cost path problem (DCLC) [24].

The DCLC problem can be described as follows: A communication network is modeled as a directed and connected graph $G = (V, E)$, where E denotes a set of directed links and V represents a set of nodes (e.g., switches, routers), connected by directed links. Any node is reachable from any other node in this graph. Every directed link $e = (u, v) \in E$ has a delay $D(e)$ and a cost $C(e)$ associated with it. The link delay, i.e., $D(e)$ is measured when a packet is passing through link e. The link cost, i.e., $C(e)$ represents some other metrics, required to optimize such as the loss-rate, bandwidth, jitter, etc. The link cost is computed using Eq. 1 as

$$C(e) = w_1 * l_e + w_2 * b_e \quad (1)$$

where w_1 and w_2 are weights corresponding to the loss-rate (l_e) and bandwidth (b_e) metrics of link e, respectively.

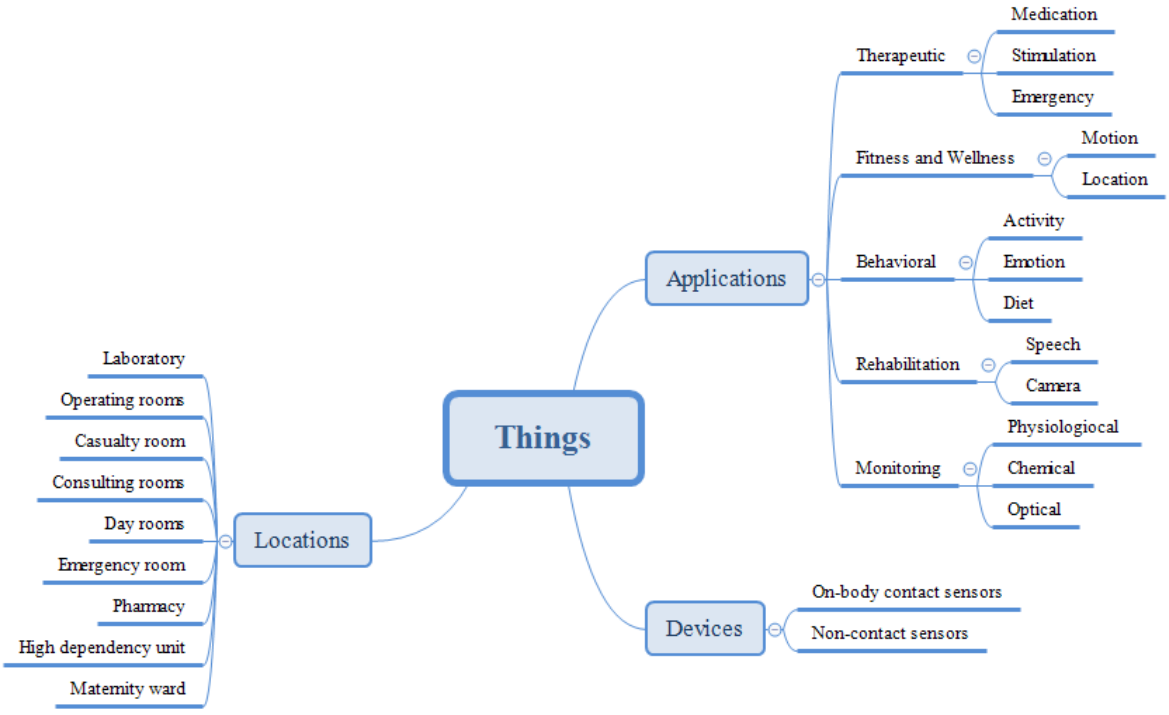

Figure 3: Ontology of the E-healthcare system

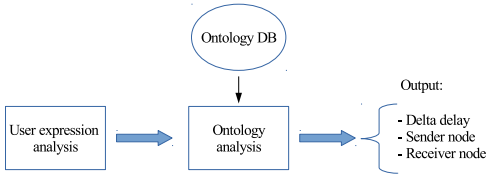

Figure 4: Request analysis module

The constrained minimization problem is represented as follow:

$$\min_{r \in R'(x,y)} \sum_{e \in r} C(e) \tag{2}$$

where $R'(x, y)$ is the set of routing paths r from source x to destination y for which the end-to-end delay is bounded by Δ_{delay}. Δ_{delay} is determined by the Resquest Analysis layer. We also have $R'(x, y) \subseteq R(x, y)$. A $r \in R(x, y)$ is in $R'(x, y)$ if and only if

$$\sum_{e \in r} D(e) \leq \Delta_{delay}$$

In order to solve the DCLC problem, we utilize a heuristic algorithm based on the Lagrange relaxation theory. This is considered as a common technique for determining lower bounds and finding solutions for this problem. The idea is based on the minimization of the modified cost function, where the cost and delay in terms of a parameter γ are combined to form an aggregate weight for each link as follows.

$$C_\gamma = C_{\text{old}} + \gamma.d \tag{3}$$

where d is the delay, C_{old} is the cost that is calculated using Eq. 1. For a given γ, the optimal path, denoted as r_γ, can be found by using Dijkstra's algorithm w.r.t. the cost c_γ, obtained in Eq. 3. If the total delay of the path r_γ, denoted as $D(r_\gamma)$, is equal or less than Δ_{delay}, it is the optimal solution. Otherwise, if $D(r_\gamma) > \Delta_{delay}$, the value of γ is increased in order to increase the influence of the delay factor in the cost function (see Eq. 3). The relationship between parameter γ, the cost and the delay of a given path can be illustrated by the following lemmas.

Lemma 1. *If* $0 \leq \gamma_1 \leq \gamma_2$ *then* $D(r_{\gamma 1}) \leq D(r_{\gamma 2})$ *and* $C(r_{\gamma 1}) \geq C(r_{\gamma 2})$.

Proof. See [9]. □

Lemma 1 shows that a larger γ will lead to a larger cost and a smaller delay. This implies that as long as the resulting shortest path does not violate the predefined delta delay, a smaller γ will definitely result to a better solution. The next lemma is used to find the smallest γ value (i.e., γ related to the shortest path that does not violate Δ_{delay}).

Lemma 2. *If* $\gamma_1 < \gamma_2$, $D(r_{\gamma 1}) \neq D(r_{\gamma 2})$, $\gamma' = \frac{C(r_{\gamma 1}) - C(r_{\gamma 2})}{D(r_{\gamma 2}) - D(r_{\gamma 1})}$, *then* $C(r_{\gamma 1}) \geq C(r_{\gamma'}) \geq C(r_{\gamma 2})$, $D(r_{\gamma 1}) \leq D(r_{\gamma'}) \leq D(r_{\gamma 2})$.

Proof. See [9]. □

Lemma 2 shows that with $\gamma' = \frac{C(r_{\gamma 1})-C(r_{\gamma 2})}{D(r_{\gamma 2})-D(r_{\gamma 1})}$, the shortest path $r_{\gamma'}$ must have a delay between the delays of $r_{\gamma 1}$ and $r_{\gamma 2}$, and the cost is also between the costs of these two paths. The above two lemmas imply that the *least cost path* r_c has to be first computed using Dijkstra's algorithm w.r.t. the cost. If its delay is not greater than Δ_{delay}, then it must be the optimal solution. Otherwise, the *least delay path* r_d, i.e., the path, found by using Dijkstra's algorithm w.r.t. the delay should be obtained. If its delay is greater than the Δ_{delay}, no optimal solution can be achieved. If none of the above conditions are true, the algorithm begins an iterative procedure. In each iteration, r_d is updated with a better solution having a lower delay and r_c is updated with a better solution having lower cost.

Algorithm 1 Lagrange relaxation-based routing algorithm

Require: $source$, $dest$, C, D, Δ_{delay}
Ensure: Optimal path
1: $r_c \leftarrow$ Dijkstra$(source, dest, C)$
2: **if** $D(r_c) \leq \Delta_{delay}$ **then return** r_c
3: **end if**
4: $r_d \leftarrow$ Dijkstra$(source, dest, D)$
5: **if** $D(r_d) > \Delta_{delay}$ **then return** "No solution"
6: **end if**
7: **while** true **do**
8: $\gamma := \frac{C(r_c)-C(r_d)}{D(r_d)-D(r_c)}$
9: $\mathcal{P} \leftarrow$ Dijkstra$(source, dest, C_\gamma)$
10: **if** $C_\gamma(\mathcal{P}) = C_\gamma(r_c)$ **then return** r_d
11: **else if** $D(\mathcal{P}) \leq \Delta_{delay}$ **then**
12: $r_d \leftarrow \mathcal{P}$
13: **else**
14: $r_c \leftarrow \mathcal{P}$
15: **end if**
16: **end while**

The heuristic algorithm (see Algorithm 1) is described in detail as follows: First, we utilize the original cost function (Eq. 1) and find the *least cost path* using Dijkstra's algorithm. If the delay of this path meets the delay requirement Δ_{delay}, it is the optimal path. Otherwise, we find the *least delay path* and examine whether the delay of this path is greater than Δ_{delay}. We may then decide to start the loop or to stop the algorithm as there is no optimal solution that can be found. The γ parameter is computed as

$$\gamma := \frac{C(r_c) - C(r_d)}{D(r_d) - D(r_c)}$$

Such parameter is updated after each iteration. The Dijkstra's algorithm is used w.r.t the new value of cost C_γ. If the cost value of the new path is found to be equal to the cost value of the *least cost path*, the optimal path should be the *least delay path*. If not, if the delay of the new path is found to be smaller than Δ_{delay}, the *least delay path* is updated as the new path. Otherwise, the *least cost path* is considered as the new path. The loop is repeated until the optimal path is found.

4 Experiments

4.1 Experimental setup

In order to conduct the experiments, we implemented a testbed, which is illustrated in the Fig. 5. The user is able to send requests to the E-healthcare system using Restful APIs from any platforms and devices. In the present work, we developed an Android application that automatically creates and sends the RESTful requests. A load balancing mechanism with PC coordinator is also implemented to support high rate of requests. Concretely, the coordinator applies the Round Robin algorithm to distribute the requests to three *Request Analysis PCs* (RA1-3). These three RA PCs analyze the incoming requests and search for the appropriate outputs in the *Ontology DB* database. In the SDN controller, these outputs are then fed to the routing algorithm that determines the appropriate way to control the simulated network. In this paper, we utilize *mininet* [6] to emulate the network topology, which involves a number of different nodes representing the Openflow-enabled switches and IoT devices.

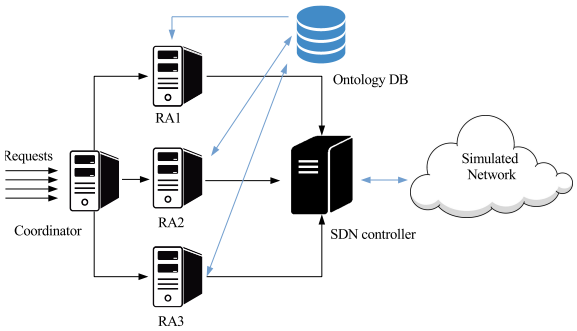

Figure 5: The testbed, implemented with load balancing mechanism

4.2 Analyzing the request analysis layer

First, we assess the scalability of the Request Analysis layer by varying the number of sending requests per second and evaluating the round trip time (RTT). In Fig. 6, it is obvious that the load balancing mechanism is able to support high request rate. It achieves a RTT of 2.2 seconds when the rate reaches 500 requests per second, while the RTT without load balancing mechanism is 26.8 seconds. This in turn, proves the scalability of the *Request Analysis layer*.

In order to evaluate the accuracy of the Request Analysis layer, we execute 10000 requests, and send them to five applications (2000 requests per application).

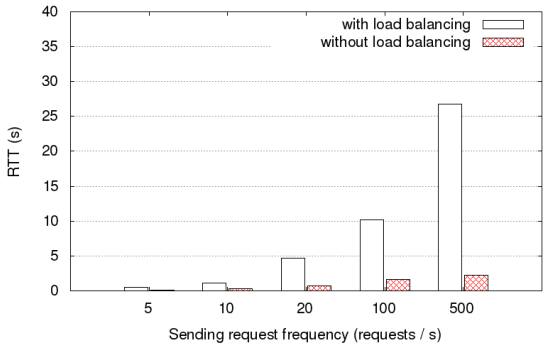

Figure 6: RTT corresponding to the Request Analysis layer, which is deployed with and without load balancing mechanism.

Applications	Number of user requests	Number of well-classified requests	Prop.
Monitoring	2000	1928	96.4 %
Therapeutic	2000	1987	99.3 %
Fitness and Wellness	2000	1979	98.9 %
Behavioral	2000	1893	94.6 %
Rehabilitation	2000	1912	95.6 %

Table 1: Accuracy related to the classification of user requests in the five applications

Table 1 shows that majority of user requests have been classified exactly for all five applications. The faults mostly come from the *Behavioral* application. This is due to the fact that the Request Analysis layer is based on the processing of text strings. The *Behavioral* application on the other hand, includes a variety of activity and emotion descriptions such as fall, sleep, exercise, anxiety, stress, depression, etc. Hence, it is more difficult to classify the user tasks. However, an accuracy of 94.6% is still acceptable for this kind of application.

4.3 Analyzing the routing layer

Concerning the performance of the routing layer, we implemented several other methods that also focus on the DCLC problem. These methods are as follows.

- The Constrained Bellman-Ford (**CBF**) routing algorithm [29], which is based on a breadth-first search that is able to update the lowest cost path in each visited node. CBF runs until either the highest constraint is exceeded or it cannot improve the paths anymore.
- The Multi-Criteria Routing algorithm (**MCR**) developed by Lee et al. in [16]: This routing algorithm is based on heuristics of ranked metrics in the network. A loop is repeated to determine the shortest path for each metric until the best path is found, or it fails for all metrics.
- The routing algorithm proposed by Cheng et al. [4] that combines the problems of finding the least cost and least delay paths by modifying the cost function. Such algorithm is abbreviated as MCF in the present work. It aims to compute a simple metric from multiple requirements using the weighted combination of the various QoS metrics.

The above mentioned algorithms are compared with the proposed algorithm (abbreviated as LARE in this paper) using the following measures:

- *Number of fails*, which is the average number of unreachable nodes, which occurs when the path cannot be found at a given delta delay Δ_{delay}. This measure aims at assessing the efficiency of a given algorithm in finding the destination nodes.
- *Delay*, which is the average delay time of the path from a source node to a reachable destination node.
- *Cost*, which is considered as the average cost of the paths from a source node to all reachable destination nodes.

Concerning the network topology, we first validate the routing algorithms and their functionality with a simple network including 17 nodes. We then carried out the experiments with NTT, a 37 node network topology [7] that is modeled using the exact characteristics of a real-world network. Finally, we construct by hand a huge network with 150 nodes (abbreviated as *150N* topology) to evaluate the scalability of the proposed framework.

To obtain the performance related to the various routing algorithms, this work varies the value of delta delay (Δ_{delay}) from 100 ms to 1000 ms in both the NTT and 150N topologies. As it can be seen from Figs. 7 and 8, when Δ_{delay} is smaller than 200 ms, the number of unreachable nodes is considerably high. This number decreases as the delay constraint is increased. All the paths are only found after 400 ms.

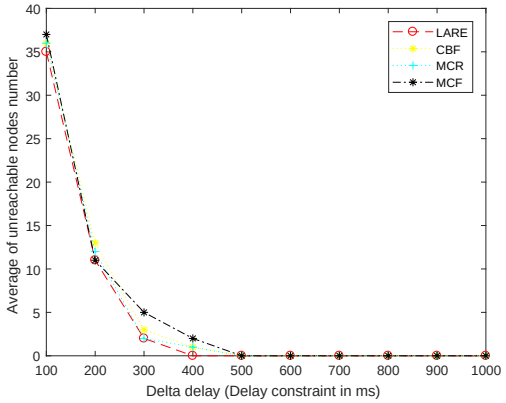

Figure 7: Average number of unreachable nodes in NTT topology

Figs. 9 and 10 show the average delay curves of the various algorithms in NTT and 150N topologies. As illustrated, with the low values of delta delay (ranging from 100 ms to

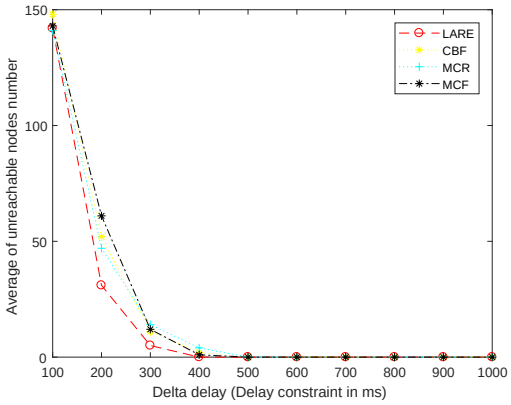

Figure 8: Average number of unreachable nodes in 150N topology

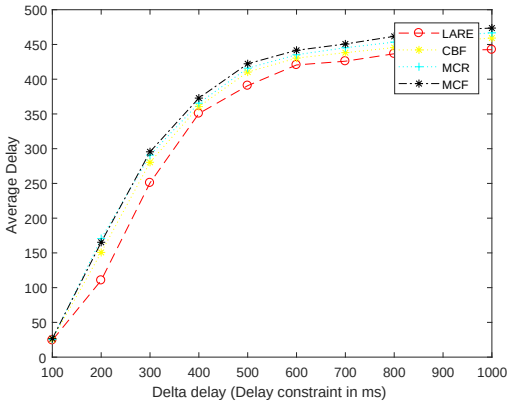

Figure 10: Average delay of the various routing algorithms in 150N topology

300 ms), the number of reachable nodes is small. The destination nodes are close to the source nodes, keeping the delay values at very low level. After 400 ms delta delay, when all the paths are found, the average delay becomes stable.

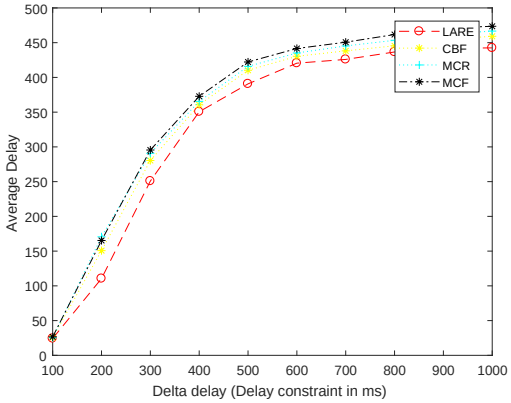

Figure 9: Average delay of the various routing algorithms in NTT topology

It is clear that the LARE algorithm provides the best result in terms of average delay in both the network topologies. The scalability of LARE has been proven in 150N (see Fig. 10), a large network having extremely high node density, where the performance gap between the proposed algorithm and other methods become more significant. At 1000 ms delta delay, LARE gives an average delay of 456 ms, while those of other methods are more than 660 ms. The MCF algorithm provides the highest average delay in both cases due to its difficulty to select an appropriate aggregate weights when combining the QoS metrics.

Figs. 11 and 12 shows the average cost of the paths found by the various algorithms in two network topologies. As explained above, at the beginning ($\Delta_{delay} < 300ms$) it is impossible to find paths to all destination nodes. Since the obtained paths are very short, the costs become relatively small. The algorithms find more paths as we increase the delta delay. When all destination nodes are found ($\Delta_{delay} \geq 300ms$), a higher delta delay would result in a lower cost. It is obvious that LARE algorithm gives almost similar outcomes to MCF in both the network topologies. This is exactly what we expected because MCF focuses on optimizing the cost value. MCR is the worst performer, since it only finds the best path for one metric, while ignoring the cost value. Especially, in the 150N topology (Fig. 12), LARE achieves an average cost of 19.5 at the delta delay of 1000 ms, while the MCR produces an average cost of 30.1.

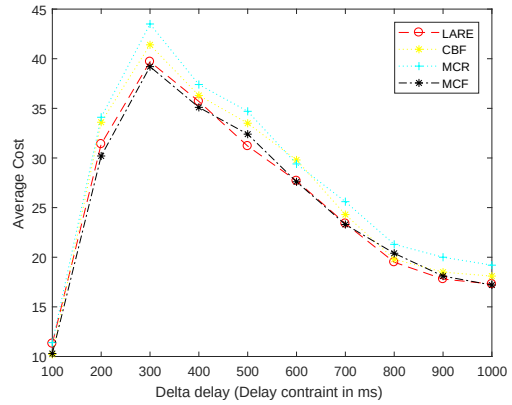

Figure 11: Average cost of the various routing algorithms in NTT topology

4.4 Analyzing the overall SHIOT framework

This section aims to validate the capability of supporting stressed network of the proposed framework and compared it with the traditional system, which relies on the simple best-effort policy and is implemented without SDN. Specifically, we try to stress the system by gradually in-

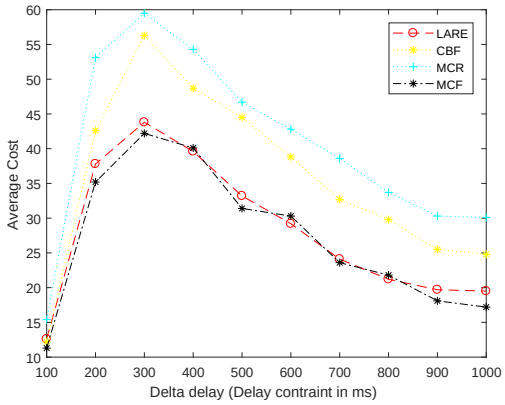

Figure 12: Average cost of the various routing algorithms in 150N topology

creasing the rate of sending requests from 100 to 2000 requests per second.

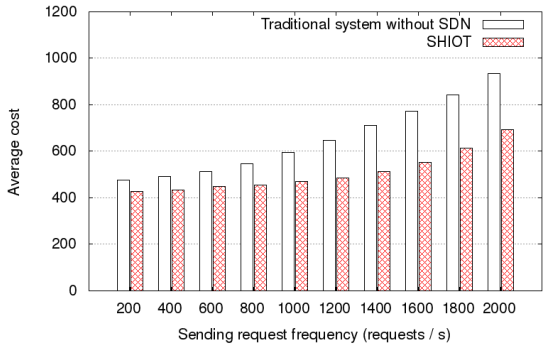

Figure 13: Capability of supporting stressed network of SHIOT and the traditional system in terms of delay.

We set the delta delay to the value of 1000 ms to ensure that the paths to all the destination nodes are found. As shown in Figs. 13 and 14, SHIOT is obviously better than the traditional couterpart in terms of delay and cost. Even in the most stressed state (i.e., 2000 requests per second), SHIOT is able to provide an average delay of 694 ms and a cost of 33, while those, achieved by the traditional system are 935ms and 69, respectively. The performance difference may be due to the simple policy (the best-effort policy) that is implemented in the core network of the traditional system. SHIOT, on the other hand, possess a layered architecture that is able to deal with the high rate of requests, sent from the various applications.

Finally, we evaluate the system performance while running the video application. Specifically, video flows are generated (video streaming) using an open source software, named VLC. Such flows are sent from one node to another in the simulated network. The experiment lasts five minutes. The delay and jitter are computed for each chosen path. In Table 2, we can see that SHIOT outperforms the conventional system. Its average end-to-end delay is about

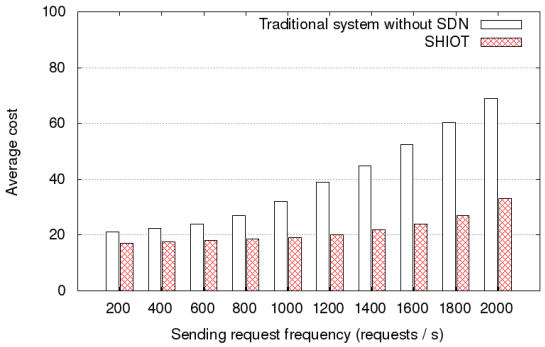

Figure 14: Capability of supporting stressed network of SHIOT and the traditional system in terms of cost.

3.5 s while that of the traditional system is 7.3 s. Similar observation can be obtained in terms of jitter. This is exactly what we expected because SHIOT has the ability to differentiate the different types of data flow (e.g. video, audio, information data).

5 Conclusions

In this paper, we have proposed a layered SDN framework, named SHIOT, to address the heterogeneity issue in the IoT. SHIOT is based on an open ontology to classify the incoming user requests. This framework also utilizes the Lagrange relaxation theory to find the optimal path in order to forward these requests to the destination nodes. In general, SHIOT can be considered as a remedy to bridge the gap between abstract high-level tasks and other low-level networks/devices. Experimental results showed that SHIOT yielded better performance when compared with the traditional system, that is deployed without SDN. It is also proved to be efficient and effective in handling tasks required by the various applications. Concerning the future works, we are in the process of evaluating SHIOT, which is implemented in the real devices (e.g. Openflow-enabled switches).

6 Acknowledgments

This work is supported by the Vietnamese Ministry of Science and Technology (MOST) under the Grant No. NDT.14.TW/16.

References

[1] The floodlight open sdn controller project. http://www.projectfloodlight.org/floodlight/.

[2] M. Al-Ayyoub, Y. Jararweh, E. Benkhelifa, M. Vouk, A. Rindos, et al. Sdsecurity: A software defined security experimental framework. In *Communication Workshop (ICCW), 2015 IEEE Inter-*

Time (s)	SHIOT		Traditional system	
	delay (s)	jitter (s)	delay (s)	jitter (s)
20	7.3	0.93	8.3	0.9
40	9.2	0.84	8.7	0.7
60	9.1	0.52	9.3	0.8
80	7.7	0.68	8.5	0.6
100	4.3	0.34	7.6	0.62
120	5.2	0.26	7.2	0.57
140	3.5	0.33	6.8	0.61
160	4.5	0.41	7.1	0.63
180	3.8	0.27	7.4	0.61
200	4.7	0.43	7.6	0.58
220	3.9	0.39	7.2	0.62
240	3.5	0.52	7.3	0.63
260	3.2	0.45	7.8	0.60
280	3.3	0.42	7.2	0.62
300	3.5	0.33	7.1	0.59

Table 2: Comparison of SHIOT and the traditional system, while running video streaming application

national Conference on, pages 1871–1876. IEEE, 2015. DOI: https://doi.org/10.1109/ICCW.2015.7247453.

[3] S. Chakrabarty, D. W. Engels, and S. Thathapudi. Black sdn for the internet of things. In *Mobile Ad Hoc and Sensor Systems (MASS), 2015 IEEE 12th International Conference on*, pages 190–198. IEEE, 2015. DOI: https://doi.org/10.1109/MASS.2015.100.

[4] S. Chen and K. Nahrstedt. On finding multi-constrained paths. In *Communications, 1998. ICC 98. Conference Record. 1998 IEEE International Conference on*, volume 2, pages 874–879. IEEE, 1998. DOI: https://doi.org/10.1109/ICC.1998.685137.

[5] A. Darabseh, M. Al-Ayyoub, Y. Jararweh, E. Benkhelifa, M. Vouk, and A. Rindos. Sdstorage: a software defined storage experimental framework. In *Cloud Engineering (IC2E), 2015 IEEE International Conference on*, pages 341–346. IEEE, 2015. DOI: https://doi.org/10.1109/IC2E.2015.60.

[6] R. L. S. de Oliveira, A. A. Shinoda, C. M. Schweitzer, and L. R. Prete. Using mininet for emulation and prototyping software-defined networks. In *Communications and Computing (COLCOM), 2014 IEEE Colombian Conference on*, pages 1–6. IEEE, 2014. DOI: https://doi.org/10.1109/ColComCon.2014.6860404.

[7] G. Di Caro, F. Ducatelle, and L. Gambardella. AntHocNet: an adaptive nature-inspired algorithm for routing in mobile ad hoc networks. *European Transactions on Telecommunications*, 16(5):443–455, 2005. DOI: https://doi.org/10.1002/ett.1062.

[8] A. Doria, J. H. Salim, R. Haas, H. Khosravi, W. Wang, L. Dong, R. Gopal, and J. Halpern. Forwarding and control element separation (forces) protocol specification. Technical report, 2010. DOI: https://doi.org/10.17487/RFC3746.

[9] G. Feng and C. Doulgeris. Fast algorithms for delay constrained leastcost unicast routing. *shortened version presented on INFORMS*, 2001.

[10] M. R. Gary and D. S. Johnson. Computers and intractability: A guide to the theory of np-completeness, 1979.

[11] J. Gubbi, R. Buyya, S. Marusic, and M. Palaniswami. Internet of things (iot): A vision, architectural elements, and future directions. *Future Generation Computer Systems*, 29(7):1645–1660, 2013. DOI: https://doi.org/10.1016/j.future.2013.01.010.

[12] F. Hu, Q. Hao, and K. Bao. A survey on software-defined network and openflow: from concept to implementation. *IEEE Communications Surveys & Tutorials*, 16(4):2181–2206, 2014. DOI: https://doi.org/10.1109/COMST.2014.2326417.

[13] H. Huang, J. Zhu, and L. Zhang. An sdn_based management framework for iot devices. In *Irish Signals & Systems Conference 2014 and 2014 China-Ireland International Conference on Information and Communications Technologies (ISSC 2014/CIICT 2014). 25th IET*, pages 175–179. IET, 2013. DOI: http://dx.doi.org/10.1049/cp.2014.0680.

[14] R. ITU-T. P. 862. *Perceptual evaluation of speech quality (PESQ): An objective method for end-to-end speech quality assessment of narrow-band telephone networks and speech codecs*, 2001.

[15] Y. Jararweh, M. Al-Ayyoub, E. Benkhelifa, M. Vouk, A. Rindos, et al. Sdiot: a software defined based internet of things framework. *Journal of Ambient Intelligence and Humanized Computing*, 6(4):453–461, 2015. DOI: https://doi.org/10.1007/s12652-015-0290-y.

[16] W. Lee and M. Hluchyj. Multi-criteria routing subject to resource and performance constraints. In *ATM Forum*, volume 94, page 0280, 1994.

[17] J. Li, E. Altman, and C. Touati. A general sdn-based iot framework with nvf implementation. *ZTE communications*, pages 1–11, 2015.

[18] P. Martinez-Julia and A. F. Skarmeta. Extending the internet of things to ipv6 with software defined networking. Technical report, Euchina-fire, 2014.

[19] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner. Openflow: enabling innovation in campus networks. *ACM SIGCOMM Computer Communication Review*, 38(2):69–74, 2008. DOI: https://doi.org/10.1145/1355734.1355746.

[20] F. Olivier, G. Carlos, and N. Florent. New security architecture for iot network. *Procedia Computer Science*, 52:1028–1033, 2015.

[21] N. Omnes, M. Bouillon, G. Fromentoux, and O. Le Grand. A programmable and virtualized network & it infrastructure for the internet of things: How can nfv & sdn help for facing the upcoming challenges. In *Intelligence in Next Generation Networks (ICIN), 2015 18th International Conference on*, pages 64–69. IEEE, 2015. DOI: https://doi.org/10.1109/ICIN.2015.7073808.

[22] I. Rec. Y. 1541: Network performance objectives for ip-based services. *International Telecommunication Union, ITU-T*, 2003.

[23] I. Recommendation. 910,"subjective video quality assessment methods for multimedia applications," recommendation itu-t p. 910. *ITU Telecom. Standardization Sector of ITU*, 1999.

[24] D. S. Reeves and H. F. Salama. A distributed algorithm for delay-constrained unicast routing. *IEEE/ACM Transactions on Networking (TON)*, 8(2):239–250, 2000. DOI: https://doi.org/10.1109/90.842145.

[25] W. Ren, Y. Sun, T.-Y. Wu, and M. S. Obaidat. A hash-based distributed storage strategy of flowtables in sdn-iot networks. In *GLOBECOM 2017-2017 IEEE Global Communications Conference*, pages 1–7. IEEE, 2017. DOI: https://doi.org/10.1109/GLOCOM.2017.8254507.

[26] P. K. Sharma, S. Singh, Y.-S. Jeong, and J. H. Park. Distblocknet: A distributed blockchains-based secure sdn architecture for iot networks. *IEEE Communications Magazine*, 55(9):78–85, 2017. DOI: https://doi.org/10.1109/MCOM.2017.1700041.

[27] V. R. Tadinada. Software defined networking: Redefining the future of internet in iot and cloud era. In *Future Internet of Things and Cloud (FiCloud), 2014 International Conference on*, pages 296–301. IEEE, 2014. DOI: https://doi.org/10.1109/FiCloud.2014.53.

[28] R. Vilalta, A. Mayoral, D. Pubill, R. Casellas, R. Martínez, J. Serra, C. Verikoukis, and R. Muñoz. End-to-end sdn orchestration of iot services using an sdn/nfv-enabled edge node. In *Optical Fiber Communication Conference*, pages W2A–42. Optical Society of America, 2016. DOI: https://doi.org/10.1364/OFC.2016.W2A.42.

[29] R. Widyono et al. *The design and evaluation of routing algorithms for real-time channels*. International Computer Science Institute Berkeley, 1994.