OpenAMI: Software-Defined AMI Load Balancing

Ahmadreza Montazerolghaem, *Student Member, IEEE,* Mohammad Hossein Yaghmaee, *Senior Member, IEEE,* and Alberto Leon-Garcia, *Fellow, IEEE*

***Abstract*—The Advanced Metering Infrastructure (AMI) is one of the main services of Smart Grid (SG), which collects data from smart meters (SMs) and sends them to utility company Meter Data Management Systems (MDMSs) via a communication network. In the next generation AMI, both the number of SMs and the meter sampling frequency will dramatically increase, thus creating a huge traffic load which should be efficiently routed and balanced across the communication network and MDMSs. This paper initially formulates the global load-balanced routing problem in the AMI communication network as an Integer Linear Programming (ILP) model, which is NP-hard. Then, to overcome this drawback, it is decomposed into two subproblems and a novel Software Defined Network (SDN)-based AMI communication network is proposed called OpenAMI. This paper also extends the OpenAMI for the cloud computing environment in which some virtual MDMSs are available. OpenAMI is implemented on a real test bed, which includes Open vSwitch, Floodlight controller, and OpenStack, and its performance is evaluated by extensive experiments and scenarios. Based on the results, OpenAMI achieves low end-to-end delay and a high delivery ratio by balancing the load on the entire AMI network.**

***Index Terms*—Industrial Internet of Things (IIoT), Large-scale AMI, Load balancing for AMI, Communication infrastructure, Resource management, SDN.**

I. Introduction

INDUSTRIAL INTERNET OF THINGS (IIOT) is a new communication paradigm that enables real-time monitoring, which provides a controlling mechanism for industrial domains. In the context of SG, the vision of deploying IIoT relies on using reliable communication technologies by employing standard protocols to perform end-to-end communication between the centre and smart entities [1]. One of the primary services in SG is AMI. It is an integrated system of SMs, communications networks, and data management systems that enables two-way communication between utilities and customers [2]. AMI utilizes the SG communication infrastructure to transfer metering data as well as customer consumption-related information [3], [4].

In the SG networks, Demand Response is utilized to balance total demand with the amount of supply. Smart consumers can make decisions autonomously about how and when to use electricity. By developing the Internet of Things (IoT) technology, it is possible to transfer customers power consumption information to the utilities through the existing communication infrastructure such as AMI network, and develop a demand side management program to control and schedule the customers appliances. In demand response approaches, customers change their power consumption depending on the energy price. By utilizing demand response approaches, it is possible to reduce or shift energy consumption from peak hours to period of less demand. AMI communication infrastructure can be utilized to transfer customer energy consumption. AMI not only can transfer SMs data, but also facilitates utilities to perform demand response programs [5]–[7]. In [8] an AMI infrastructure is introduced which checks the customers energy consumption and controls the electric energy used with the demand response techniques. In [9], a micro-grid system consisting of some energy sources and AMI infrastructure has been implemented. Based on the price of energy market, the customers power load are scheduled. The consumption information and the energy price is communicated through the existing AMI network.

So, in AMI networks, a critical issue is establishing a reliable, scalable, and secure communication network that can meet Quality of Service (QoS) requirements [10]. As investigated in [11], QoS is an essential component of the overall architecture in SG. Real-time transferring of metering data from the customer side (Smart Meter) to the utility's MDMS needs a QoS-support communication infrastructure able to guarantee low end-to-end delay and a high packet delivery ratio [12], [13].

On the other hand, AMI service is rapidly spreading and many utility companies prefer to develop this service in order to remotely gather consumption information from the customer side. In the large-scale AMI, there are many SMs located in different regions which should transfer metering and related consumption information to the MDMS through some intermediate nodes, such as concentrators or switches [14], [15]. The essence of routing mechanisms in the AMI network may cause an inefficient use of network resources [16]. Also, the current AMI communication network is limited to small-scale local regions which do not satisfy the mentioned QoS requirements for the next generation large-scale AMI. Thus, for the efficient management of this high-volume data, it is extremely critical to utilize new communication technologies.

Recently, introduced Software Defined Networking (SDN) is a major trend in the telecommunication industry that can enhance the SG networks [17]. In SDN, the control and data planes are separated and logically centralized using the OpenFlow protocol [18]. It can meet the requirements of the AMI communication network [19]. In this case, each network switch simply forwards the traffic and enforces policy according to instructions received from the controller. This makes the network programmable in a way that promises to be more flexible, scalable, and secure than that of traditional networks [18]. SDN has attracted attention for developing different SG applications which require a higher degree of network awareness [19]–[25].

A. Montazerolghaem and MH. Yaghmaee are with the Department of Computer Engineering, Ferdowsi University of Mashhad, Mashhad, Iran.
E-mail: Ahmadreza.montazerolghaem@stu.um.ac.ir, yaghmaee@ieee.org
Alberto Leon-Garcia is with the Department of Electrical and Computer Engineering, University of Toronto, Toronto, Canada.
E-mail: alberto.leongarcia@utoronto.ca

A. Motivations

Why do we need a scalable AMI communication infrastructure? Knowing that the AMI infrastructure is a key service in SG, it is vital to design and develop a high performance and scalable communication infrastructure with quality of service support. Recently, the AMI has been utilized for supporting new services in the energy market, such as managing household appliance usage based on current tariffs and electricity rates [5]. In addition, the AMI is rapidly developing and spreading in different geographical locations. Moreover, for the future large-scale AMI, it will be necessary to collect and transfer energy consumption information from each customer appliance [7]. Therefore, this will create a huge traffic load which should be efficiently routed and balanced across the network and MDMSs.

It should also be noted that in many AMI systems, data is collected from SMs every 15 minutes. Although this is a significant improvement (compared with the traditional way that only records the meter data once a month), it is far from enough in achieving the full vision of a SG [26].

To summarize, with the overall modern SG road map, both the number of SMs and the sampling frequency on a meter will increase dramatically. Consequently, a huge amount of data will go through the MDMS, which imposes a great challenge on the scalability of the traditional AMI communication network [26].

B. Contributions

The main contributions of our work can be summarized as follows:

- Theoretical aspect

In order to provide the communication requirements of the AMI network, a reliable, secure, and scalable communication protocol should be utilized. This paper proposes using the Session Initiation Protocol (SIP) to transfer AMI packets between the customer side and utility company. As will be explained later, the SIP protocol has many unique benefits for use in the AMI network. SIP is a mature protocol that is consistent with all current standards and architectures for the AMI and fulfills many of the device communication requirements. Therefore, a SIP-based SM and MDMS are proposed to serve as the communication interface to the AMI network.

On the other hand, the future large-scale AMI will merge with demand response and energy management systems. The volume of the traffic load transferred by the future large-scale AMI network is rapidly increasing. This may cause overload and congestion for the AMI intermediate nodes and MDMS servers. As a result, optimized traffic engineering and overload control should be applied. This paper presents an optimization problem to route the traffic on the AMI communication infrastructure and balance the load on the entire network. It is proven that when there are limited resources, the mentioned problem is NP-hard. A decomposition approach is proposed to solve this issue.

- Implementation

We present OpenAMI which is an SDN-based platform for the future large-scale AMI. Regarding the above mentioned decomposition approach, OpenAMI consists of different components, including *flow monitoring*, *admission control*, *MDMS load estimation*, *MDMS and path selection*. We also extend OpenAMI for the cloud computing environment in which some virtual MDMSs are available. In this case, OpenAMI benefits from resource scaling advantages. By dynamic resource scaling, the overload condition of the virtual MDMS can be mitigated. OpenAMI has been implemented on a real test bed. By extensive experiments and scenarios, its performance has been evaluated.

C. Organization

In what follows we begin reviewing the related work in Section II and then defining the problem in Section III. Then, the proposed architecture and model are illustrated in Section IV. Next, Section V presents implementation and performance evaluation results that confirm the superior performance of the proposed model. Finally, Section VI gives the concluding marks of the paper.

II. Related Work

This section discusses state-of-the-art research that uses SDN technology in SG. Application of SDN technology in SG has received considerable attention over the past few years.

[1] proposes a SDN platform based on IIoT to support resiliency by reacting immediately whenever a failure occurs. This renders real-time monitoring techniques of SG network possible. [17] presents the potential of SDN for strengthening the resilience of SG, even under catastrophic circumstances. As mentioned in [19], ease of configuration and management, cross-domain content-based networking, virtualization, and isolation are some of the opportunities offered by SDN in SG networks. Molina et al. [20] propose to integrate SDN in IEC-61850-based substation automation systems. In [21], Sydney et al. use SDN to provide an automatic fail-over method for SG networks. In addition, [22] focuses on how SDN can supply resilience to distribution substations with self-recovery. Considering the requirements of SG, Dorsch et al. [24] establish a test bed based on SDN for communications in IEC-61850. [25] investigates the use of SDN in heterogeneous SG so as to create a self-configuration infrastructure and implement it based on dissimilar technologies, such as IEEE 802.11 and Ethernet. In [27], the authors analyze local as well as central mechanisms for fast failover in software-defined SG communication networks.

Overall, despite the findings of these studies, the benefit of SDN for the AMI communication network remain largely unexplored. Moreover, the AMI load balancing problem is also an open and challenging issue because of the resource limitations of a large-scale AMI.

In the following, we introduce four of the state of the art in load balancing which are close enough to our work and in the performance evaluation section we compare the results of our proposed mechanism with these four papers ([28]–[31]). It is important to note that none of these methods are based on SDN.

In [28], a load balanced routing is introduced in the path identified by Hybrid Wireless Mesh Protocol (HWMP) for AMI wireless mesh networks. This mechanism uses airtime link metric as path selection parameter to avoid congestion. In such

cases, links with larger airtime link metric are avoided by the path selection process. They are, however, inevitably allowed in some unavoidable conditions.

In [29], a session-aware admission control algorithm (SAMbA) is proposed. SAMbA builds a minimized set of sessions among applications and Intelligent Electronic Devices (IEDs) to adjust the load. SAMbA comprises two components: the SAMbA-Portal and the SAMbA-Session Controller that deploys a session-based admission control approach.

In [30], the authors attempt to prevent server overload by balancing the load among available servers using an implicit mechanism called History Weighted Average Response time (HWAR). In this mechanism, each server has a corresponding window in a load balancer. The contents of each window are the history of server's response time, which is used to estimate the load being currently processed on servers.

In [31], a Transaction Least-Work-Left (TLWL) algorithm routes a new request to the server with the least load using a load balancer and set of counters. The counters specify the weighted sum of the transactions assigned to each server. A new request is passed to the server with the lowest counter.

III. Problem Definition

Assume that an AMI communication network can be modeled as graph $G = (V, E)$, where V represents the set of switches and E represents the set of links. Let $|V| = n$ and $|E| = m$, the number of switches and the number of links, respectively. This graph connects k SMs to w MDMSs (Fig. 1). The information of all MDMSs is synchronized to a central MDMS that, without loss of generality, we can connive it. Each SM either directly or through a concentrator connects to a network switch. Each link $e \in E$ has a cost c^e and consumes r^e units of available resources. The resource consumption of each switch and MDMS is also denoted by r^s and r^m, respectively. δ^m, δ^s, and δ^e represent the remaining resources of each MDMS, switch, and link, respectively. Costs and resources are assumed to be non-negative. The goal is to find the least cost (shortest) path from source node s (smart meter) to target node t (the most appropriate MDMS) that satisfies the resource limits of the MDMSs, switches and links. In other words, the objective of the problem is to balance the load of the entire network, including link load balancing, switch load balancing, and MDMS load balancing, in order to efficiently use resources and consequently achieve low end-to-end delay and a high delivery ratio. Generally c^e is considered as a hop count. Therefore finding a path which satisfies the above constraints is equal to finding a path with minimum hop counts in consideration of limited resources. If resource restrictions are not satisfied, then an overload occurs.

Prior to proposing the approach, we prove that the problem of *global load-balanced routing in the AMI communication network* is an ILP problem and is, therefore, NP-hard.

Proposition 1. *The global load-balanced routing problem in an AMI communication network with limited resources is an ILP problem.*

Proof: Let P be the set of all routes from s to t. For any path $p \in P$, a binary variable u_p is introduced, and c_p and r_p are used to denote the total cost and resource consumption of p, respectively. The constrained resource problem is:

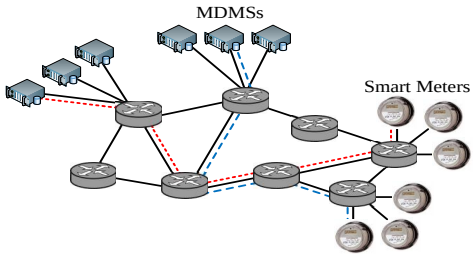


Figure 1. AMI communication network

$$\textbf{\textit{minimize}} \sum\nolimits_p c_p u_p \tag{1}$$

subject to:

$$\sum\nolimits_p u_p = 1, \tag{2}$$

$$\sum\nolimits_{m \in p} r_p^m u_p \leq \delta_p^m, \ \ \forall p \in P \tag{3}$$

$$\sum\nolimits_{s \in p} r_p^s u_p \leq \delta_p^s, \ \ \forall p \in P \tag{4}$$

$$\sum\nolimits_{e \in p} r_p^e u_p \leq \delta_p^e, \ \ \forall p \in P \tag{5}$$

Variables: $\quad u_p \in \{0, 1\}$

that is, finding a route p which the resource consumption is bounded by δ_p and the cost is minimized. Constraint (2) guarantees that exactly one path is selected and constraints (3-5) ensure that this path is feasible, i.e. the total resource consumption of the path satisfies the resource constraints. Also, the objective function ensures that the cheapest feasible path is chosen. The binary variable u_p renders the model an ILP which is generally NP-hard and cannot be solved in polynomial time. ■

To overcome this drawback and reduce complexity, the problem is decomposed into two subproblems (*MDMS selection subproblem* and *Path selection subproblem*), which are solved in two phases. Since MDMS resources are more critical than those of switches and links, the constraints of MDMS resources are first met, and then an appropriate path with respect to the selected MDMS is found. In this regard, a SDN-based framework is proposed for the AMI communication network which considers subproblems as SDN applications in the application plane. This framework benefits from SDN's ability to provide a global view of the entire network.

IV. Proposed Architecture and Model

A. AMI Communication Protocol

In the large-scale AMI, SMs should be able to send a large amount of collected data to the MDMSs and to receive operational commands. This, however, may lead to overload. Therefore, a standard and highly reliable communication protocol is required for transferring this high volume of data. The AMI's current communication protocol is limited to small-scale local regions, which do not yet meet the demanding communication requirements of the next generation AMI. There are a number of core requirements for the large-scale AMI communication protocol, including scalability, reliability,

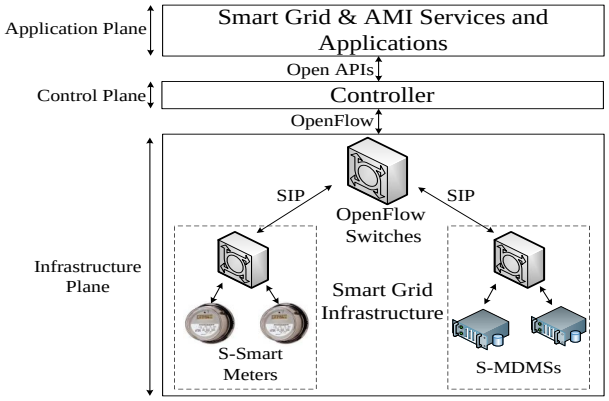


Figure 2. The proposed architecture of the SDN-based AMI communication network (OpenAMI)

flexibility, routability, and security [10]. Given our observation that all of these requirements are already presented in the Session Initiation Protocol (SIP) [32], we propose to use SIP as the AMI communication protocol.

SIP was originally developed as an IETF (Internet Engineering Task Force) protocol for use in the next generation telephony, initially for voice calls over an IP infrastructure but also with the intention of being general enough for adoption in many different environments. SIP is a mature protocol and has proven to be reliable, scalable, and secure. It can transmit any type of data and uses a wide variety of messages, such as `Invite` and `Bye`, to initiate and terminate a connection between endpoints.

In conclusion, SIP can operate very well in a peer-to-peer AMI network in which there is communication between SMs and MDMSs. In this regard, `Register`, `Invite`, `Ack`, and `Bye` messages are used to register SMs, initiate a new AMI connection, exchange data, and terminate an AMI connection, respectively. In addition, each SM (and MDMS) is attached with an embedded SIP module to serve as the communication interface to the network infrastructure. Together, the SM, MDMS, and embedded SIP module form an SIP-based SM (S-SM) and an SIP-based MDMS (S-MDMS).

B. AMI Communication Network

In this paper we introduce a novel SDN-based AMI communication network (OpenAMI), as shown in Fig. 2. The *infrastructure plane* includes S-SMs, S-MDMSs, and OpenFlow switches. The data generated by S-SMs, such as demand response information, will be routed to the proper S-MDMS via OpenFlow switches and the SIP protocol. Meanwhile, the other direction of data flow, such as control information or price broadcasting, can be also sent from S-MDMSs to S-SMs via these switches and SIP messages. Since the AMI consists of a large number of S-SMs that generate a large amount of data, a centralized *control plane* is required for effective resource management. By using OpenFlow messages, the controller also collects up-to-date switch network state information, such as the available capacity of switches and links. It requests various statistics from the switch network by sending `FEATURE-REQUEST` messages and, in return, the switch network sends `FEATURE-REPLY` messages containing the requested statistics[1]. In addition, the network topology is obtained using the Link Layer Discovery Protocol (LLDP). A variety of AMI services and applications can be applied in the OpenAMI *application plane*. Communication between the planes is conducted with OpenAPIs, such as the OpenFlow protocol.

With OpenFlow's help, the OpenAMI controller has a global view of AMI resources. OpenAMI also takes advantage of SIP. Thus, in terms of control and management, it provides excellent scalability and flexibility to large-scale AMIs.

C. OpenAMI for Global Load-balanced Routing

This paper proposes that the OpenAMI solve the global load-balanced routing problem, as shown in Fig. 3.

Upon receiving an AMI message (encapsulated in a SIP message), the OpenFlow switches encapsulate it into a `Packet-In` message and then send it to the controller for determining the S-MDMS and path. The *Flow Monitoring* application then categorizes these messages in to three queues and stores the Connection-ID of the messages; in this way, the various connection messages can be distinguished from each other. The task of the *Admission Control* application is deciding whether to admit or drop the new AMI connection requests (`Invite` messages), in regard to the total capacity of S-MDMSs. The *S-MDMS Load Estimation* application uses response time (connection establishment delay) as a criterion of each S-MDMS' load. The response time is the period from the forwarding of `Invite` to the receiving of `200 Ok`. The *S-MDMS Selection* application selects the S-MDMS with the least response time for serving admitted `Invite` messages. The task of the *Path Selection* application is to find the shortest path to the selected S-MDMS which meets the constraints of switch and link resources. Note that S-MDMS and path selection are conducted for `Invite` messages and the rest of that connection's messages follow it. Finally, the *Rules Handler* application has the duty of creating rules. This causes the controller to send OpenFlow rules to switches through *Flow-Mod* messages. In the following, all of the mentioned applications are explained in detail.

1) Flow Monitoring: The proposed *Flow Monitoring* application includes a Deep Packet Inspection (DPI) module which is able to detect and classify messages. By inspecting messages, the DPI module classifies them into three classes, namely *Invite*, *200 Ok*, and *ACK & etc*. Moreover, the connection information, including ID, is extracted from the messages and stored.

2) Admission Control: The *Admission Control* application is implemented based on a finite state machine, as shown in Fig. 4. Suppose that the capacity of each S-MDMS is $\mathbb{c}$ and the total capacity of S-MDMSs is $\mathbb{C}$. $\mathbb{c}$ means that how many connections in a time unit can be processed by an S-MDMS. Also, the total number of all S-MDMS active connections in the time unit is shown by $\mathbb{N}$. In fact, $\mathbb{N}$ is a counter which increases by one number with the admission of an `Invite` message. When receiving the `Bye` message of that connection,

[1]These messaging mechanisms are described in detail in OpenFlow specification [OpenFlow switch specification - Online Available: http://archive.openflow.org/wp/documents/]

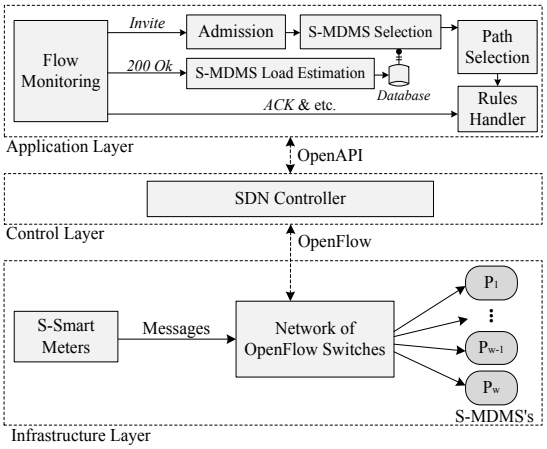


Figure 3. OpenAMI with developed applications for load balancing

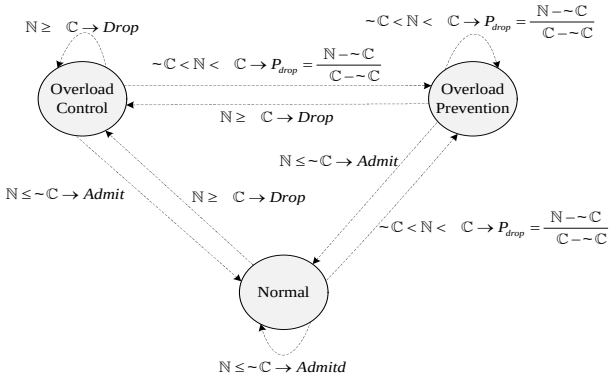


Figure 4. The finite state machine of the *Admission Control* application

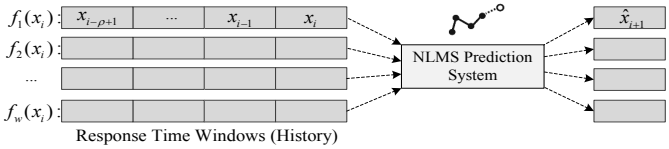


Figure 5. The overview of the *MDMS Load Estimation* application

it understands that the process of that connection is finished by S-MDMS and so it consequently decreases by one number.

$\mathbb{C}$ and $\mathbb{N}$ are the inputs of this state machine; its output is one of three actions: *admit*, *drop*, or *drop with a certain probability*. Moreover, this state machine has three states: *normal*, *overload prevention*, and *overload control*.

When $\mathbb{N}$ is less than $\alpha\mathbb{C}$, the state is normal and all of the newly received AMI connection requests are admitted. If $\mathbb{N}$ is more than $\alpha\mathbb{C}$ but less than $\beta\mathbb{C}$, the state is in overload prevention. In this situation, S-MDMSs are not overloaded. However, to avoid an overload, some percent of the requests, with the probability of $\frac{\mathbb{N}-\alpha\mathbb{C}}{\beta\mathbb{C}-\alpha\mathbb{C}}$, are dropped. In the case that $\mathbb{N}$ is more than $\beta\mathbb{C}$, an overload is imminent. Consequently, all of the received requests are dropped in order to prevent saturation of S-MDMS resources. Therefore, this mechanism is a proactive method of managing S-MDMS resources.

The α and β ($0 \leq \alpha < \beta \leq 1$) coefficients represent the degree of rigor for dealing with the S-MDMS overload. This means that, if one gives them a small value, the use of resources will be less and this is also true for the number of connections. Meanwhile, with the fast drop of requests, an overload will never take place. However, if one has larger values for α and β, more requests are admitted and consequently more S-MDMS resources are used. Nevertheless, the probability of overloading, due to the lack of resources, will increase. Therefore, fine-tuning coefficients for a trade-off between efficiency and resources is vital.

3) S-MDMS Load Estimation: OpenFlow does not provide any information about the status of each S-MDMS. Hence, in order to select the best S-MDMS for assigning its admitted request, OpenAMI should be able to estimate the future status of each S-MDMS. To this end, we use the history of each S-MDMS response times, because it can serve a good measure of the S-MDMS' future status.

The *S-MDMS Load Estimation* application employs a separate window for each S-MDMS (Fig. 5). Each window contains the response time history of the S-MDMS over time. The size of the window is ρ. x_i is the value of the i^{th} response time and f is the vector of the response time history. The aim is to obtain an estimation of x_{i+1} ($\widehat{x}_{i+1}$) in regard to f and a prediction system. One of the best prediction systems that provides a trade-off between accuracy, complexity, and responsiveness is Normalized Least Mean Square (NLMS). That is why this filter is applied as the prediction system in the *MDMS Load Estimation* application. Details will be provided in the following section.

Given a vector of ρ observations of x_i, $f(x_i) = [x_i, x_{i-1}, \ldots x_{i-\rho+1}]$, generates the estimation $\widehat{x}_{i+1}$ of the value x_{i+1}. The NLMS filter coefficients are time differing and are tuned on the basis of the feedback information taken by error ε_i, that $\varepsilon_i = x_{i+1} - \widehat{x}_{i+1}$. The vector of filter coefficients with h_i is specified. The values of h adjust dynamically in order to decrease the Mean Square Error. The NLMS performs the following: 1- Initializes the coefficient h_0; 2- For each new data, the filter h_i is updated based on the recursive equation:

$$h_{i+1} = h_i + \mu \frac{\varepsilon_i f(x_i)}{\| x_i \|^2} \tag{6}$$

where $\| x_i \|^2 = f(x_i) f^T(x_i)$ and μ is a fixed parameter called step size. Pursuant to [33], NLMS converges so long as $0 < \mu < 2$. At time i, the values x_{i+1}, hence ε_i, are not known. Therefore, the value ε_{i-1} is used instead and the one step NLMS predictor update equation becomes:

$$h_{i+1} = h_i + \mu \frac{\varepsilon_{i-1} f(x_{i-1})}{\| f(x_{i-1}) \|^2} \tag{7}$$

Algorithm 1 presents the NLMS algorithm formulation for our system, where $f(x_i)$ and $\widehat{x}_{i+1}$ are the input and output of the predictor, respectively.

4) S-MDMS Selection: The up-to-date output of the *S-MDMS Load Estimation* application always exists in a database (see Fig. 3). This database includes w pairs of <S-MDMS, $\widehat{x}_{i+1}$> and represents the up-to-date value of $\widehat{x}_{i+1}$ for each S-MDMS.

By searching the database, the *S-MDMS Selection* application chooses the pair with the minimum $\widehat{x}_{i+1}$. In other words, among the w S-MDMS's, the S-MDMS with the minimum

Algorithm 1: NLMS for *S-MDMS Load Estimation*

Parameters: ρ =filter order, μ =step size
Initialize: $h_0 = 0$
for $i = 1, 2, ...$ **do**
  $f(x_i) = [x_i, x_{i-1}, \dots x_{i-\rho+1}]$
  $h_i = h_{i-1} + \mu \frac{\varepsilon_{i-1} f(x_{i-1})}{\| f(x_{i-1}) \|^2}$
  $\| f(x_i) \|^2 = f(x_i) f^T(x_i)$
  $\varepsilon_i = x_i - \widehat{x}_i$
  $\widehat{x}_{i+1} = h_i f^T(x_i)$
end

$\widehat{x}_{i+1}$ is selected. Because the minimum $\widehat{x}_{i+1}$ indicates the S-MDMS with the least load, it is assigned to the admitted AMI connection request. Finally, the controller has a list which includes <S-MDMS$_\xi$, connection$_\zeta$> pairs, thus indicating that S-MDMS$_\xi$ has been selected for handling the connection$_\zeta$.

5) Path Selection: The *Path Selection* application solves the *path selection subproblem*. This subproblem seeks the shortest path between S-SM and the selected S-MDMS in consideration of the limitation of resources of switches and links. In other words, this subproblem is to minimize the cost c of path p from S-SM to the selected S-MDMS, while keeping r_p^s and r_p^e under the given constraints δ_p^s and δ_p^e, respectively. As previously mentioned, by using OpenFlow messages, the controller is able to collect the values of δ^s and δ^e and consequently $\delta_p^s = \sum_{s \in p} \delta^s$ and $\delta_p^e = \sum_{e \in p} \delta^e$. To describe this formally, the *path selection subproblem* is looking for an optimal path:

$$p^* = argmin\{c_p : p \in P \text{ and } r_p^s \leq \delta_p^s,\ r_p^e \leq \delta_p^e\} \tag{8}$$

We propose using the Lagrangian Relaxation Based Aggregated Cost (LARAC) [34] method, which is a polynomial time algorithm that efficiently finds a satisfactory path. LARAC is based on *Lagrange relaxation*. Lagrange relaxation is a usual method for determining lower bounds and finding solutions for this problem. The *Path Selection* application runs the LARAC algorithm to solve the subproblem for a given source S-SM and destination S-MDMS. Then, the controller updates the switches' flow tables accordingly. Hence, the routes are dynamically set. The following explains how the mentioned subproblem is solved with the LARAC algorithm. Before that, Eq. (8) is reformulated in the form of Eq. (9):

$$p^* = argmin\{c(p) : p \in P \text{ and } \vartheta(p) \leq \Delta_p\} \tag{9}$$

$c(p)$ represents the cost of entire path p, which is the same as c_p. The total consumed resources and total remaining resources of path p (including switches and links) are shown by $\vartheta(p)$ and Δ_p, respectively.

LARAC is based on the heuristic of minimizing the $c_\lambda := c + \lambda\vartheta$ modified cost function. c_λ denotes the aggregated cost. For a given fixed λ, one can easily calculate the minimal path (p_λ). If $\lambda = o$ and $\vartheta(p_\lambda) \leq \Delta_p$, then an optimal solution for the original problem can be found as well. If $\vartheta(p_\lambda) > \Delta_p$, λ must increase to augment the dominance of resources in the modified cost function. Thus, λ is increased while the optimal solution of c_λ suits resource requirements. For a given λ, define $L(\lambda)$ as:

$$L(\lambda) = min\{c_\lambda(p) : p \in P\} - \lambda\Delta_p \tag{10}$$

Then $L(\lambda)$ is a lower bound to problem (9) for any $\lambda \geq 0$ (proof in [34]). Note that $min\{c_\lambda(p) : p \in P\}$ is the same as the minimum aggregated cost of a path with respect to a given value of λ. This can be easily arrived at by applying Dijkstra's algorithm with the aggregated cost. It should be remembered that the path which has a minimum aggregated cost with respect to a given λ is denoted as p_λ. Then $L(\lambda) = c_\lambda(p_\lambda) - \lambda\Delta_p$ and the duality of Eq. (10) can be offered in the following form:

$$L^* = max\{L(\lambda) : \lambda \geq 0\} \tag{11}$$

The problem of maximizing $L(\lambda)$ is called the Lagrangian dual problem. The value of λ that achieves the maximum $L(\lambda)$ is indicated by λ^*. Note that L^*, the optimum value of Eq. (11), is a lower bound on the optimum cost of the path solving the corresponding problem. The main issue in solving Eq. (11) is how to search for the optimal λ and determining the criterion for stopping the search. One such efficient search procedure is the LARAC algorithm provided as follows (Algorithm 2).

Algorithm 2: LARAC for *Path Selection*

Procedure LARAC (s,t,c,ϑ,Δ)
$p_c = Dijkstra(s, t, c)$
if $\vartheta(p_c) \leq \Delta_p$ **then**
  return p_c
end
$p_\vartheta = Dijkstra(s, t, \vartheta)$
if $\vartheta(p_\vartheta) > \Delta_p$ **then**
  return *There is no solution*
end
for $i = 1, 2, ...$ **do**
  $\lambda = \frac{c(p_c) - c(p_\vartheta)}{\vartheta(p_\vartheta) - \vartheta(p_c)}$
  $r = Dijkstra(s, t, c_\lambda)$
  if $c_\lambda(r) = c_\lambda(p_c)$ **then**
    return p_ϑ
  end
  if $\vartheta(r) \leq \Delta_p$ **then**
    $p_\vartheta = r$
  end
  else $p_c = r$
end
where $Dijkstra(s, t, c)$ returns a c-minimal path between the source node s and destination node t.

In the first step, the algorithm computes the shortest path for cost. If the path found meets the resource constraint, this is the optimal path. Otherwise, the algorithm stores the path as the latest infeasible path, called p_c. Then, the algorithm specifies the shortest path for resources, denoted as p_ϑ. If p_ϑ is infeasible, there is no solution in this instance.

In the second step, set $\lambda = \frac{c(p_c) - c(p_\vartheta)}{\vartheta(p_\vartheta) - \vartheta(p_c)}$. With this value of λ, one can find a new c_λ-minimal path r. If $c_\lambda(r) = c_\lambda(p_c) = c_\lambda(p_\vartheta)$, then the optimal λ is obtained pursuant to claim 5 of [34]. Otherwise, set r is the new p_c or p_ϑ depending on whether r is infeasible or feasible.

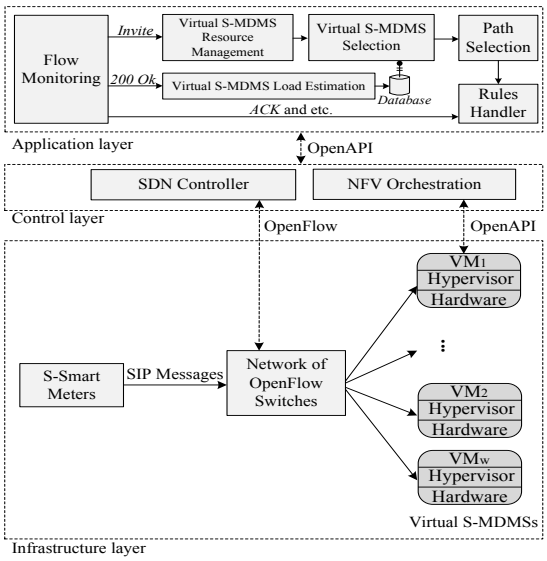


Figure 6. OpenAMI for virtual S-MDMSs scaling

D. OpenAMI for Virtual S-MDMS Resource Scaling

Network Function Virtualization (NFV) can contribute to the SDN and partly overcome resource constraints through the virtualization of network devices and functions [35].

In this section, OpenAMI is extended so as to benefit from the advantages of virtualization in the scaling of S-MDMS resources. Resource scaling is a technique for adjusting resources in regard to demands that increase the scalability of the network. As shown in Fig. 6, in the proposed system all networking resources (such as switches) are under the control of SDN controller, while all other resources (such as virtual S-MDMSs) are under the control of NVF orchestration (NFVO). More specially, the SDN controller is responsible for computing the forwarding tables for all switches. On the other hand, the NFVO is responsible for managing all computing and storage resources. It keeps all the necessary information about virtual machines and physical hosts. In the infrastructure layer, hypervisors run on physical servers to support Virtual Machines (VMs) that implement S-MDMSs. As a result, customizable and programmable virtualized S-MDMSs running as software within VMs are provided. In the control plane, the NVFO manages the virtual S-MDMSs and the SDN controller manages the switch network. In the application plane, the *Admission Control* application changes to the *Virtual S-MDMS Resource Management* application. The task of the proposed *Virtual S-MDMS Resource Management* application is to make decisions about the scaling of virtual S-MDMSs in regard to the incoming load (Fig. 7). Therefore, if $\mathbb{N} \leq \alpha \mathbb{C}$, the state is in underload and the virtual S-MDMS with the minimum response time is scaled down. If $\mathbb{N} \geq \beta \mathbb{C}$, the state is in overload and the virtual S-MDMS with the maximum response time is scaled up. In addition, if $\alpha \mathbb{C} < \mathbb{N} < \beta \mathbb{C}$, the state is in normal load and no action is performed. As the rest of the applications are similar to the previous plan, it is unnecessary to re-explain them.

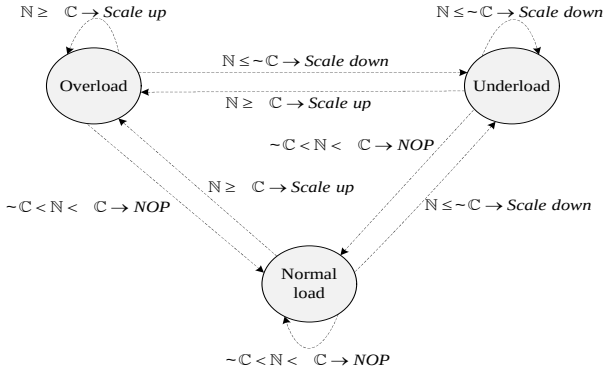


Figure 7. The finite state machine of the *Virtual S-MDMS Resource Management* application

V. Implementation and Performance Evaluation

To evaluate the performance of the proposed approach, the topology in Fig. 8 is employed. This topology includes 2 S-MDMSs (P1 and P2), 7 OpenFlow switches (S1 to S7), and 1 controller (OpenAMI controller). The bandwidth of each link is 10 Mbps. Here we use *Open vSwitch v2.4.1*, *Floodlight v1.2*, and *Kamailio v4.3.6* to implement the OpenFlow switches, controller, and S-MDMSs, respectively. The applications are implemented as a module running atop the controller. The *nDPI* engine is used to implement the DPI module. The open source *SIPp* software is also utilized to implement the S-SMs and inject the traffic. *OProfile* software measures the CPU and memory usage. We generate the background traffic by having *iperf* send packets at a fixed rate. Each experiment is run three times and the average is taken as the result. α and β are considered 0.65 and 0.95, respectively. ρ and μ are chosen to be 30 and 0.8, since they produce a satisfactory performance based on the analysis of the results. We run several tests to tune this parameter. A photograph of our implementation setup is given in Fig. 9.

A. The First Experiment: Constant Offered Load

The first experiment includes two scenarios with different background traffics. In Scenario 1, the background traffic of both S-MDMSs is equal at 50 requests per second (rps); however, in Scenario 2, the background traffic of P1 and P2 are 100 rps and 50 rps, respectively. Then, a constant offered load of 300 rps is injected in to the network for 100 seconds. In this

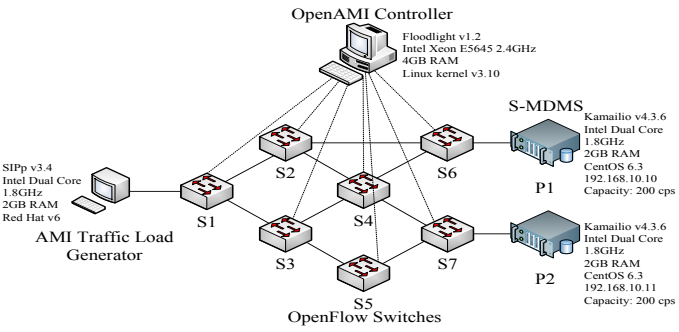


Figure 8. The test bed of OpenAMI

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/JIOT.2017.2778006, IEEE Internet of Things Journal

Table I
OPENAMI CONTROLLER PERFORMANCE

Evaluation Criteria	Scenario	Time (second) 10	20	30	40	50	60	70	80	90	100
Throughput of OpenAMI controller (fps) ∼	1	297	299	293	300	295	300	295	293	294	294
	2	299	296	300	294	295	296	294	297	298	296
Response time of OpenAMI controller (s) ∼	1	0.0063	0.0072	0.0055	0.0086	0.0064	0.0092	0.0061	0.0057	0.0046	0.0074
	2	0.0051	0.0092	0.0063	0.0044	000.89	0.0056	0.0067	0.0076	0.0083	0.0064
CPU consumption of OpenAMI controller (%) ∼	1	33.60	32.10	30.50	25.54	22.67	25.96	28.41	26.11	23.25	25.32
	2	33.62	27.94	28.22	27.89	34.31	25.27	32.86	31.41	26.93	26.52
Memory consumption of OpenAMI controller (%) ∼	1	26.87	20.40	21.22	22.84	28.16	23.78	19.18	26.52	23.37	26.79
	2	26.64	28.49	30.97	23.22	27.85	26.92	26.31	26.05	25.09	26.47

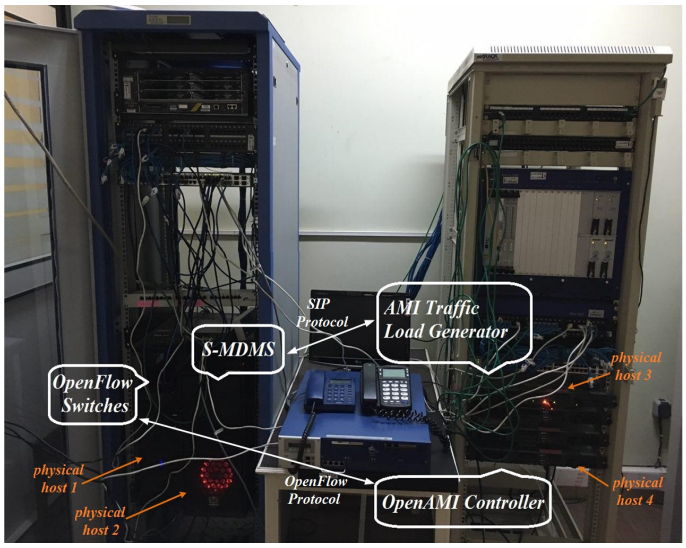


Figure 9. The implementation setup of OpenAMI

experiment, three mechanisms are used in order to implement the *S-MDMS Selection* application: the proposed method (based on response time), round-robin, and random.

Fig. 10 presents the total requests that are successfully delivered to S-MDMSs (Delivered Requests or DR), End-to-End Delay (EED), and the utilized CPU of S-MDMSs (due to space limitations, memory results are not provided). In both scenarios, the DR of the proposed method are almost equal and closer to the offered load (Fig. 10 (a) and (d)). However, the DR of the other two mechanisms worsened in Scenario 2. The reason for this is that the background traffic of the S-MDMSs is different in Scenario 2. This causes the load distribution of these two mechanisms to work blindly. Consequently, the EED and utilization of resources increase (Fig. 10 (b), (c), (e) and (f)). The resource utilization of both S-MDMSs in the proposed method is almost equal, representing an informed and fair load distribution.

Fig. 11 presents more results about the proposed method. The DR of each S-MDMS is illustrated in Fig. 11 (a). In Scenario 1, the DR of both S-MDMSs are almost equal (∼150 rps). In Scenario 2, the DR of P2 (∼199 rps) is almost two times that of P1 (∼98 rps). This is because the background traffic of P2 is equal to half of the background traffic of P1 in Scenario 2; consequently, more loads are allocated to P2.

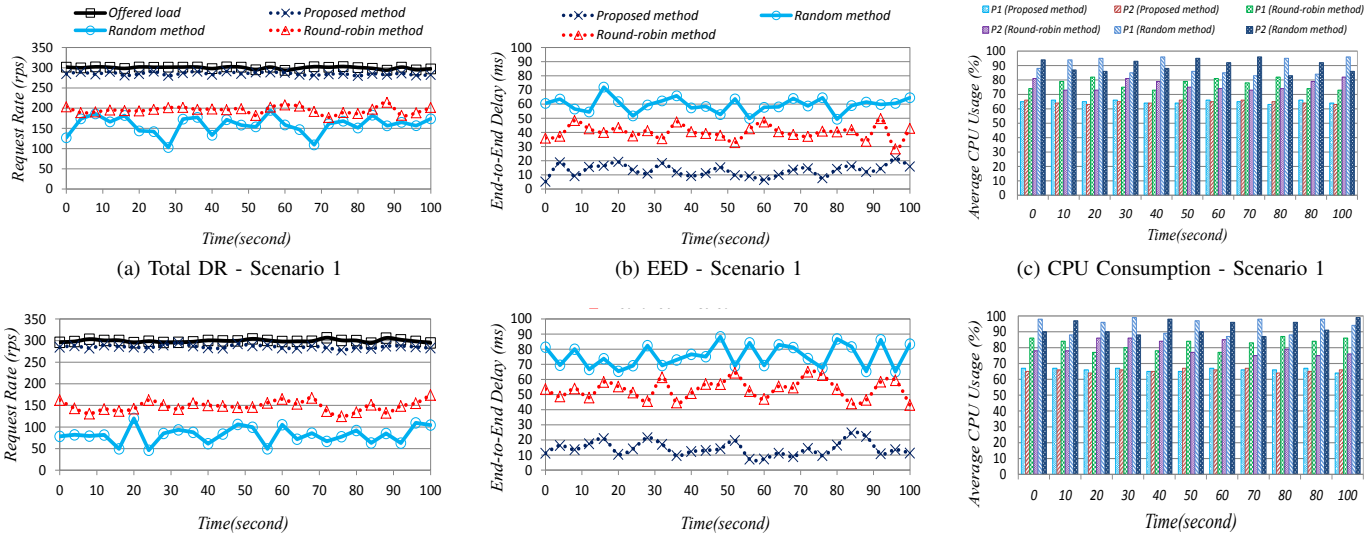
Fig. 11 (b) and (c) demonstrates the average link utilization and average CPU consumption of the switches. In essence, these figures show that the best paths have been selected for forwarding the request message to P1 and P2. The path <S1, S2, S6> is utilized for forwarding requests to P1 and the paths <S1, S3, S4, S7> and <S1, S3, S5, S7> are for forwarding requests to P2. These paths are the shortest existing paths that also observe the resource limitation. Therefore, the load has been well balanced among the links (and switches) and their resources have been fairly used. For example, the utilization of link <S1, S3> is almost two times that of links <S3, S4> and <S3, S5>; Moreover, the CPU usage of S3 is twice that of switches S4 or S5. In Scenario 2, note that the paths to P2 (<S1, S3, S5, S7> and <S1, S3, S4, S7>)is more congested compared with those of P1 (<S1, S2, S6>).

Table I shows the throughput, average response time, and resource utilization of the OpenAMI controller. The controller's throughput is the number of serviced flows per time unit. The average response time of the controller is the period between forwarding the `Packet-In` message from a switch to the time of receiving the `Flow-Mod` message from the controller. The results are almost equal in both scenarios. The average throughput of the controller is about 296 fps and its average response time is about 6ms. Therefore, OpenAMI is able to achieve a high throughput with low delay. The resource utilization of the controller indicates that the designed applications do not force extra overhead on the controller. Consequently, the controller is not a bottleneck.

B. Second Experiment: Variable Offered Load

The performance of OpenAMI with a variable offered load is assessed in this experiment. As observed in Table II, the offered load starts at 150 rps and gradually goes up to 450 cps (to the 300th second). Then, with a sudden drop at the 300th second, it drops down to 150 rps and, at the 400th second, it ramps back to 450 rps with a sudden spurt. In the last 100 seconds, the offered stabilizes at 300 rps.

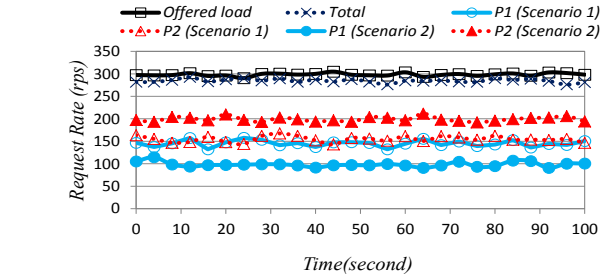
Regarding the considered α and β values, in the first and second 100 seconds, the state of the *Admission Control* application is normal ($\mathbb{N} \leq \alpha\mathbb{C}$) along with that of overload prevention ($\alpha\mathbb{C} < \mathbb{N} < \beta\mathbb{C}$), respectively. In these 200 seconds, the total DR of the S-MDMSs is close to that of the offered load. At the 200th second, the state of the *Admission Control* application changes to overload control ($\mathbb{N} \geq \beta\mathbb{C}$). In this state almost all resources of the S-MDMSs are used. Consequently, overcapacity requests are dropped and the rejection rate increases (note that the total capacity of the S-MDMSs is 400 rps). With the end of the overload at the 300th second, OpenAMI has again succeeded to bring the DR close to the offered load. Unlike the gradual increase of the offered load in the first 300 seconds, an



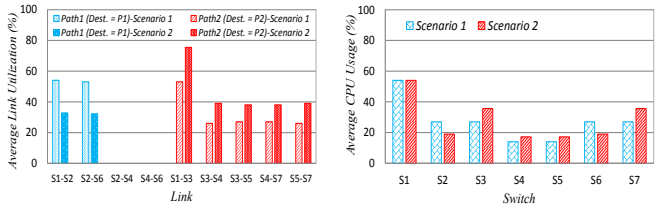
(a) Total DR - Scenario 1 (b) EED - Scenario 1 (c) CPU Consumption - Scenario 1

(d) Total DR - Scenario 2 (e) EED - Scenario 2 (f) CPU Consumption - Scenario 2

Figure 10. OpenAMI performance with a constant offered load



(a) DR of Esch S-MDMS



(b) Links Utilizations (c) CPU Consumption of Switches

Figure 11. Performance of the proposed method in the first experiment

Table II
OPENAMI PERFORMANCE WITH VARIABLE OFFERED LOAD

Time (s)	0-100	100-200	200-300	300-400	400-500	500-600
Offered load (rps) ∼	150	300	450	150	450	300
Total DR of S-MDMSs (rps) ∼	146	296	**385**	147	**386**	297
Rejection rate (%) ∼	2.66	1.33	**14.44**	2	**14.22**	1
Avg. CPU usage of P1 (%)	32	66	**98**	34	**100**	64
Avg. CPU usage of P2 (%)	34	64	**100**	32	**97**	66
Avg. memory usage of P1 (%)	31	60	**98**	30	**99**	60
Avg. memory usage of P2 (%)	30	59	**99**	31	**98**	58

immediate congestion takes place at the 400th second. Again, OpenAMI has been able to properly use the capacity of S-MDMS. Immediate congestion occurs when a large number of S-SMs simultaneously make connections and a heavy load is forced on the network (for example, after the elimination of a malfunction). As a result, The stability of OpenAMI is indicated by a high DR in sudden fluctuations of the offered load.

Table III
VM FLAVOR SPECIFICATIONS

Flavor	Memory (MB)	vCPUs	Disk (GB)
m1.small	2048	1	20
m1.medium	4096	2	40
m1.large	8192	4	80
m1.xlarge	16384	8	169

C. Third Experiment: S-MDMS Virtualization

In the time periods $[200, 300]$sec and $[400, 500]$sec in the previous experiment (when the incoming load is greater than the S-MDMSs resources), OpenAMI can achieve a DR close to the offered load with S-MDMS virtualization (by scaling up the resources of the virtual S-MDMSs). Moreover, during $[300, 400]$sec, one can release the extra resources in spite of having achieved a DR close to the offered load (by scaling down the resources of the virtual S-MDMSs). To provide virtual S-MDMSs in the test bed and to manage them by NVFO, the *OpenStack* software platform is employed. OpenStack consists of components responsible for the establishment and management of VMs of which the most important one is *Nova*. Each of the virtual S-MDMSs in this experiment can have one of the four flavors shown in Table III. Small is the primary flavor of S-MDMSs. By running the following command, each virtual S-MDMS can be resized:

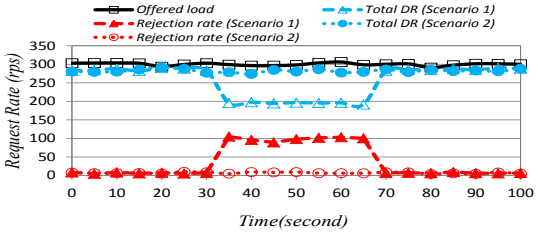
```
nova resize <VM Instance Name> <new flavor>
```

For example, by running the command `nova resize P1 m1.medium`, the P1 flavor can be changed to medium.

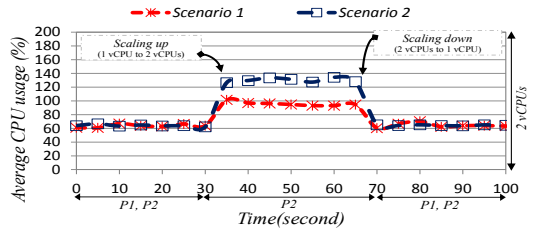
Table IV shows the DR, rejection rate, and flavor of each of the S-MDMSs over time. Unlike the second experiment, this time our method obtains a DR close to the offered load for the entire 600 seconds. This is because P2 is scaled up at the 200th and 400th second and scaled down at the 300th and 500th second.

Table IV
OPENAMI PERFORMANCE WITH VIRTUAL S-MDMSS

Time (s)	0-100	100-200	200-300	300-400	400-500	500-600
Offered load (rps) ∼	150	300	450	150	450	300
Total DR of S-MDMSs (rps) ∼	147	295	447	148	446	298
Rejection rate (%) ∼	2	1.66	0.66	1.33	0.88	0.66
P1 flavor	small	small	small	small	small	small
P2 flavor	small	small	**medium**	small	**medium**	small

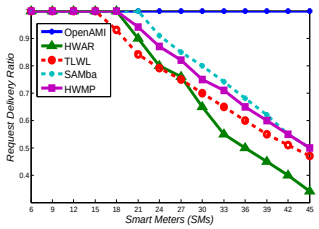


(a) Total DR

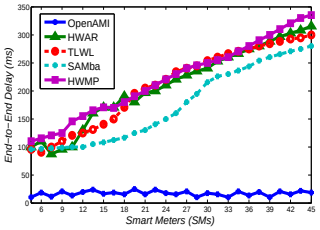


(b) CPU Consumption

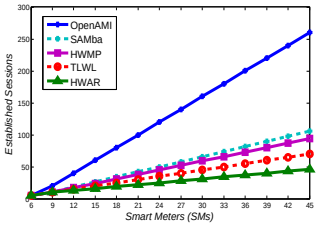
Figure 12. OpenAMI performance with the S-MDMS failure



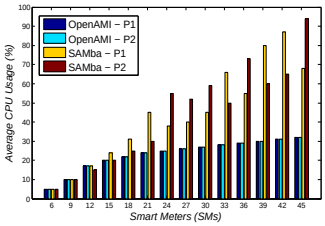
(a) Request delivery ratio



(b) End-to-end delay (ms)



(c) Established sessions



(d) CPU Consumption

Figure 13. Comparison with other algorithms

D. Fourth Experiment: S-MDMS Failure

In addition to immediate congestion, the sudden failure of one of the S-MDMSs is another reason why S-MDMSs may face overload. The results of this experiment are presented in Fig. 12. P1 faced failure at the 30th second and goes back to service again at the 70th second. Two scenarios are tested:

- **Scenario 1**: neither of the two S-MDMSs is virtual,
- **Scenario 2**: both of the S-MDMSs are virtual.

Up to the 30th second, the average DR of both scenarios is 295 rps. With the failure of P1 at the 30th second, only P2 keeps on providing service. In this situation, OpenAMI in Scenario 1 intends to prevent the DR drop by using the maximum capacity of P2 (200 rps). For this reason, resources utilization of P2 increases. However, the average DR decreases down to about 194 rps and the rejection rate increases up to 98 rps. In Scenario 2, the required conditions for scaling up P2 at the 30th second are provided. By doing so, DR remains without a drop and at about 295 rps. With the reactivation of P1 at the 70th second and by returning to the normal state, the load is distributed between both S-MDMSs. Besides, DR maximizes and resource utilization of both S-MDMSs becomes equal.

E. Fifth Experiment: Comparison with Other Algorithms

In this section, we compare OpenAMI with TLWL, HWAR, HWMP and SAMba. For this end, we increase the offered load (and inevitably the SMs to 45). The algorithms are compared with respect to their request delivery ratio, end-to-end delay, established sessions and resource consumption. The results are provided in Fig. 13. Part (a) of this figure demonstrates that OpenAMI has been able to achieve a request delivery ratio of one regardless of the number of SMs, whereas this decreases for the other methods. This is the result of the integrated architecture of OpenAMI, which has provided a global view of the entire network and global load balancing. The other advantage of OpenAMI is that load balancing in this method is performed on the links, middle nods and MDMSs. This is in contrast to the other methods, where balancing is only performed over the load between MDMSs or links.

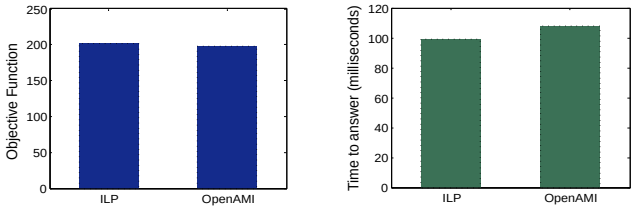
Among the SAMba, HWMP, HWAR, and TLWL methods, the SAMba has the highest request delivery ratio, since, as previously mentioned, this method is aware of the *session*, while the TLWL and HWAR methods only seek to estimate MDMSs load (without considering the number of sessions or the capacity of links and intermediate nodes). In other words, as stated in [29], the SAMba uses the Minimized Session Set Algorithm (MSSA), which functions fairly cleverly. HWMP seeks to distribute load among the links by using airtime link metric. So, it does not succeed in getting the request delivery ratio like the SAMba.

Fig. 13 (b) shows the EED. It is clear from the figure that EED in OpenAMI is 15ms on average, which is significantly shorter than the others. We can also deduce from Fig. 13 (a) and (b) that by increasing the number of SMs, the other methods would face request delivery rate reduction (or EED elevation).

Fig. 13 (c) illustrates the number of established sessions. It is evident that in OpenAMI, the number of established sessions increases with the growth of SMs. This indicates that OpenAMI has truly achieved a request delivery ratio of one. Moreover,

Table V
COMPARISON BETWEEN OPENAMI AND ILP MODEL 1

		$k=50, w=10$ $n=8, m=11$	$k=60, w=15$ $n=10, m=15$	$k=70, w=20$ $n=12, m=19$	$k=80, w=25$ $n=14, m=23$
Time to answer (ms)	ILP	99	985	3921	10087
	OpenAMI	108	156	198	243
Objective Function ($\sum(c_p u_p)$)	ILP	201	324	422	566
	OpenAMI	198	319	418	561



(a) Objective Function (b) Time (ms)

Figure 14. Objective function and time to answer of OpenAMI and ILP

it indicates that all AMI sessions that were supposed to be established between SMs and MDMSs were initiated without imposing any additional delay.

Fig. 13 (d) demonstrates the resource consumption of P1 and P2 in OpenAMI and SAMba. It is evident that resource consumption of P1 and P2 is equal in OpenAMI. This further indicates that load has been fairly distributed in the network and between the MDMSs. This is in contrast to SAMba, where resource consumption is significantly different for P1 and P2 (especially after 20 SMs).

F. Sixth Experiment: ILP Solution Assessment

In this section, we intend to assess the optimal solution (ILP model 1) for limited cases (limited S-SMs and S-MDMSs as sources and destinations) of the problem. Given that the problem is NP-hard, the optimal solution cannot be achieved in polynomial time. Hence, increasing the size of the problem exponentially increases the time of ILP model 1.

For this test, k and w are set to 50 and 10, respectively, which are connected based on the topology in Fig. 1. In this manner, n and m are 8 and 11. Moreover, δ^m, δ^s and δ^e are 100, 90 and 80, respectively. Besides, $\mathbb{C}$ is set to 500 and the remaining configuration is the same as that of section 5 (α and β are chosen as 0.65 and 0.95, respectively. Moreover, ρ and μ are chosen as 30 and 0.8). Finally, to solve ILP model 1 we used CPLEX solver [36]. The results are depicted in Fig. 14 and it is clear that OpenAMI is able to closely resemble the result of ILP.

In Table V, the solution value and time is provided for various inputs for the two methods. As is evident in the table, response time significantly increases for ILP method, whereas OpenAMI has a slow growth (in fact OpenAMI is a *heuristic* method to solve global load-balanced routing problem in an AMI communication network).

VI. CONCLUSION AND FUTURE WORK

The growing number of SMs together with their constant increase in data rate impose a serious problem on the traditional AMI network in SG. Therefore, a main challenge is to build a scalable communication architecture to route and balance the huge amount of data generated by those SMs. This paper initially introduces the global load-balanced routing problem in the AMI communication network, which is NP-hard due to limitations of MDMSs and network resources. A novel SDN-based AMI communication network (OpenAMI) is then proposed. Moreover, an extension of OpenAMI for a virtualization environment based on NFV technology is presented. OpenAMI is implemented on a real test bed which includes Open vSwitch, Floodlight controller, and OpenStack; its performance is evaluated by extensive experiments and scenarios. The results show that the proposed architecture has low resource overhead and satisfactory performance. In addition, it can take advantage of flexible scale-out design for application deployment.

An important notion which can be considered as a future work is to adopt a more advanced policy of MDMS load estimation and resource scaling. Furthermore, the application of OpenAMI for providing various SG services is a promising new research direction.

REFERENCES

[1] S. Al-Rubaye, E. Kadhum, Q. Ni, and A. Anpalagan, "Industrial internet of things driven by sdn platform for smart grid resiliency," *IEEE Internet of Things Journal*, 2017.

[2] V. C. Gungor, D. Sahin, T. Kocak, S. Ergut, C. Buccella, C. Cecati, and G. P. Hancke, "Smart grid technologies: Communication technologies and standards," *IEEE transactions on Industrial informatics*, vol. 7, no. 4, pp. 529–539, 2011.

[3] J. Gao, Y. Xiao, J. Liu, W. Liang, and C. P. Chen, "A survey of communication/networking in smart grids," *Future Generation Computer Systems*, vol. 28, no. 2, pp. 391–404, 2012.

[4] V. C. Gungor, D. Sahin, and e. Kocak, "A survey on smart grid potential applications and communication requirements," *IEEE Transactions on Industrial Informatics*, vol. 9, no. 1, pp. 28–42, 2013.

[5] F. Benzi, N. Anglani, E. Bassi, and L. Frosini, "Electricity smart meters interfacing the households," *IEEE Transactions on Industrial Electronics*, vol. 58, no. 10, pp. 4487–4494, 2011.

[6] J. Haase, J. M. Molina, and D. Dietrich, "Power-aware system design of wireless sensor networks: Power estimation and power profiling strategies," *IEEE Transactions on Industrial Informatics*, vol. 7, no. 4, pp. 601–613, 2011.

[7] P. Palensky and D. Dietrich, "Demand side management: Demand response, intelligent energy systems, and smart loads," *IEEE transactions on industrial informatics*, vol. 7, no. 3, pp. 381–388, 2011.

[8] I.-H. Choi and J.-H. Lee, "Development of smart controller with demand response for ami connection," in *Control Automation and Systems (ICCAS), 2010 International Conference on*. IEEE, 2010, pp. 752–755.

[9] V. Fusco, G. K. Venayagamoorthy, S. Squartini, and F. Piazza, "Smart ami based demand-response management in a micro-grid environment," in *Power Systems Conference (PSC), 2016 Clemson University*. IEEE, 2016, pp. 1–8.

[10] R. R. Mohassel, A. Fung, F. Mohammadi, and K. Raahemifar, "A survey on advanced metering infrastructure," *International Journal of Electrical Power & Energy Systems*, vol. 63, pp. 473–484, 2014.

[11] H. Li and W. Zhang, "Qos routing in smart grid," in *2010 IEEE Global Telecommunications Conference GLOBECOM 2010*, Dec 2010, pp. 1–6.

[12] R. H. Khan and J. Y. Khan, "A comprehensive review of the application characteristics and traffic requirements of a smart grid communications network," *Computer Networks*, vol. 57, no. 3, pp. 825–845, 2013.

[13] N. Saputro, K. Akkaya, and S. Uludag, "A survey of routing protocols for smart grid communications," *Computer Networks*, vol. 56, no. 11, pp. 2742–2771, 2012.

[14] E. Ancillotti, R. Bruno, and M. Conti, "The role of communication systems in smart grids: Architectures, technical solutions and research challenges," *Computer Communications*, vol. 36, no. 17, pp. 1665–1697, 2013.

[15] Y. Yan, Y. Qian, H. Sharif, and D. Tipper, "A survey on smart grid communication infrastructures: Motivations, requirements and challenges," *IEEE communications surveys & tutorials*, vol. 15, no. 1, pp. 5–20, 2013.

[16] J. Park, Y. Lim, S.-J. Moon, and H.-K. Kim, "A scalable load-balancing scheme for advanced metering infrastructure network," in *Proceedings of the 2012 ACM Research in Applied Computation Symposium*. ACM, 2012, pp. 383–388.

[17] X. Dong, H. Lin, R. Tan, R. K. Iyer, and Z. Kalbarczyk, "Software-defined networking for smart grid resilience: Opportunities and challenges," in *Proceedings of the 1st ACM Workshop on Cyber-Physical System Security*. ACM, 2015, pp. 61–68.

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

[19] J. Zhang, B.-C. Seet, T.-T. Lie, and C. H. Foh, "Opportunities for software-defined networking in smart grid," in *Information, Communications and Signal Processing (ICICS) 2013 9th International Conference on*. IEEE, 2013, pp. 1–5.

[20] E. Molina, E. Jacob, J. Matias, N. Moreira, and A. Astarloa, "Using software defined networking to manage and control iec 61850-based systems," *Computers & Electrical Engineering*, vol. 43, pp. 142–154, 2015.

[21] A. Sydney, D. S. Ochs, C. Scoglio, D. Gruenbacher, and R. Miller, "Using geni for experimental evaluation of software defined networking in smart grids," *Computer Networks*, vol. 63, pp. 5–16, 2014.

[22] A. Aydeger, K. Akkaya, M. H. Cintuglu, A. S. Uluagac, and O. Mohammed, "Software defined networking for resilient communications in smart grid active distribution networks," in *Communications (ICC), 2016 IEEE International Conference on*. IEEE, 2016, pp. 1–6.

[23] S. Wang and X. Huang, "Aggregation points planning for software-defined network based smart grid communications," in *Computer Communications, IEEE INFOCOM 2016-The 35th Annual IEEE International Conference on*. IEEE, 2016, pp. 1–9.

[24] N. Dorsch, F. Kurtz, H. Georg, C. Hägerling, and C. Wietfeld, "Software-defined networking for smart grid communications: Applications, challenges and advantages," in *Smart Grid Communications (SmartGridComm), 2014 IEEE International Conference on*. IEEE, 2014, pp. 422–427.

[25] S. Rinaldi, P. Ferrari, D. Brandão, and S. Sulis, "Software defined networking applied to the heterogeneous infrastructure of smart grid," in *Factory Communication Systems (WFCS), 2015 IEEE World Conference on*. IEEE, 2015, pp. 1–4.

[26] J. Zhou, R. Q. Hu, and Y. Qian, "Scalable distributed communication architectures to support advanced metering infrastructure in smart grid," *IEEE Transactions on Parallel and Distributed Systems*, vol. 23, no. 9, pp. 1632–1642, 2012.

[27] N. Dorsch, F. Kurtz, F. Girke, and C. Wietfeld, "Enhanced fast failover for software-defined smart grid communication networks," in *Global Communications Conference (GLOBECOM), 2016 IEEE*. IEEE, 2016, pp. 1–6.

[28] A. Robertsingh, D. Devaraj, and R. Narmathabanu, "Development and analysis of wireless mesh networks with load-balancing for ami in smart grid," in *Computing and Network Communications (CoCoNet), 2015 International Conference on*. IEEE, 2015, pp. 106–111.

[29] H. Silva, A. Neto, E. Cerqueira, and H. Silva, "Samba: A session aware multicast based architecture for cost-efficient smart grid applications," in *Communications (ICC), 2015 IEEE International Conference on*. IEEE, 2015, pp. 226–231.

[30] A. Montazerolghaem, S. Shekofteh, M. Yaghmaee, M. Naghibzadeh *et al.*, "A load scheduler for sip proxy servers: design, implementation and evaluation of a history weighted window approach," *International Journal of Communication Systems*, vol. 30, no. 3, 2017.

[31] H. Jiang, A. Iyengar, E. Nahum, W. Segmuller, A. N. Tantawi, and C. P. Wright, "Design, implementation, and performance of a load balancer for sip server clusters," *IEEE/ACM Transactions on Networking (TON)*, vol. 20, no. 4, pp. 1190–1202, 2012.

[32] J. Rosenberg, H. Schulzrinne, G. Camarillo, A. Johnston, J. Peterson, R. Sparks, M. Handley, and E. Schooler, "Sip: session initiation protocol," Tech. Rep., 2002.

[33] S. S. Haykin, *Adaptive filter theory*. Pearson Education India, 2008.

[34] A. Juttner, B. Szviatovski, I. Mecs, and Z. Rajko, "Lagrange relaxation based method for the qos routing problem," in *Proceedings IEEE INFOCOM 2001. Conference on Computer Communications. Twentieth Annual Joint Conference of the IEEE Computer and Communications Society (Cat. No.01CH37213)*, vol. 2, 2001, pp. 859–868 vol.2.

[35] R. Jain and S. Paul, "Network virtualization and software defined networking for cloud computing: a survey," *IEEE Communications Magazine*, vol. 51, no. 11, pp. 24–31, 2013.

[36] I. I. CPLEX, "V12. 1: Users manual for cplex," *International Business Machines Corporation*, vol. 46, no. 53, p. 157, 2009.