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.2018.2812718, IEEE Internet of Things Journal

SDN-enabled Traffic-aware Load Balancing for M2M Networks

Yu-Jia Chen*, Li-Chun Wang*, Meng-Chieh Chen*, Pin-Man Huang* and Pei-Jung Chung†
*Department of Electrical & Computer Engineering, National Chiao Tung University, Hsinchu, Taiwan,
†Delta Electronics Co., Ltd., Taiwan

***Abstract*— This paper proposes a traffic-aware load balancing scheme for machine-to-machine (M2M) networks using software-defined networking (SDN). Load balancing techniques are essential for M2M networks to relieve the heavy loading caused by bursty traffic. Leveraging the capability of SDN to monitor and control the network, the proposed load balancing scheme can satisfy different quality of service (QoS) requirements through traffic identification and rerouting. Experimental results show that the proposed scheme can reduce service response time up to 50% compared to the non-SDN load balancing scheme.**

***Index Terms*— Load balancing; machine-to-machine (M2M) networks; software-defined networking.**

I. Introduction

Machine-to-machine (M2M) networks can connect devices via the Internet for new services, such as smart home and industrial automation [1]. However, M2M networks need to resolve the congestion issue of massive devices, especially in the case with bursty traffic. Secondly, meeting various quality of service (QoS) requirements for different M2M services is also a challenging task [2].

Load balancing and overload relief techniques are crucial to guarantee QoS in M2M networks [3], [4]. Fig. 1 illustrates an M2M system with a medium load balancer used in OpenMTC [5]. When processing sensor data, a cloud server (i.e., serving node) increases its CPU workload [6]. If the CPU workload exceeds a predefined threshold, say 60%, an aggregator will create new virtual machines (VMs) as the serving nodes. In this case, this kind of load balancer becomes the service bottleneck, and raises the concern of the single point of failure (SPOF). Because the route from the CPU-based load balancer to the server is static, network congestion may occur [7]. Finally, the policy in this kind of load balancer is not easy to be scaled to the case of multiple balancers.

Software-defined networking (SDN) can provide the flexibility of programming the network by changing the policies of rules in its centralized software-defined controller with a global view of the network status. Therefore, SDN possesses the ability of load balancing [8]–[12]. By dynamically managing incoming traffic to different available routes, SDN can avoid service bottleneck and SPOF. By reconfiguring the flow routing rule in its controller, the load balancing policy can be easily updated in an SDN system.

In the literature, most of the existing SDN-enabled load balancing techniques are mainly designed to support high-quality multimedia streaming [13]. However, SDN-enabled load balancing technique for M2M system has been rarely

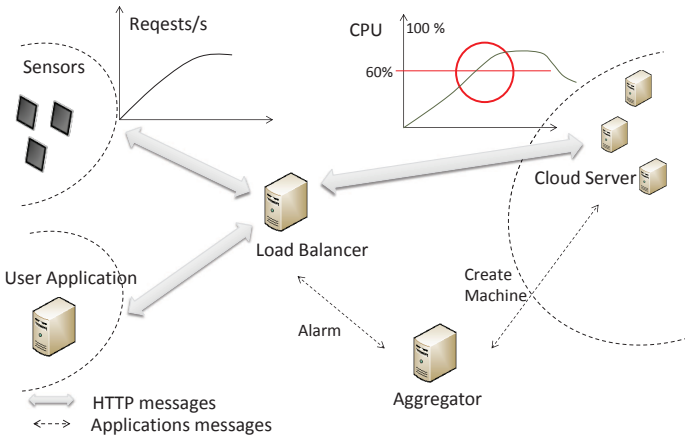

Fig. 1. An M2M system with a medium load balancer.

seen in the literature. Massive M2M traffic with various QoS requirements is usually with high overhead in the SDN control plane, including monitoring and reconfiguring the network forwarding policies. Moreover, the bursty M2M traffic can increase the overhead in the data plane due to frequent traffic rerouting.

The objective of this paper is to develop a proactive traffic-aware load balancing (TALB) technique for M2M networks by exploiting the traffic-aware feature in SDN, In our previous work [14], we developed a load balancing system framework that can take the advantage of the distributed virtual switches to serve as the load balancer. The detailed algorithm and the performance results of TALB have not been presented yet. The proposed load balancing scheme contains three key steps:

- Identify M2M traffic flows in SDN switch according to its packet headers.
- Route the identified traffic to an appropriate server based on flow tables.
- Update flow tables when the estimated delay performance exceeds the QoS threshold.

In principle, the proposed TALB mechanism exploits the features of instant traffic identification and dynamic traffic rerouting in SDN. Moreover, network forwarding policies are updated only when the server or network loading parameters exceed the QoS violation threshold, thereby reducing the computing and communication overhead in the control plane, We perform some SDN experiments, and our results show that the proposed scheme can reduce service response time up to

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.2018.2812718, IEEE Internet of Things Journal

50% compared to the existing load balancing scheme.

The rest of this paper is organized as follows. In Section II, we describe the background of SDN, M2M networks, and load balancing techniques. We identify the load balancing issues in M2M networks and discuss the potential of SDN in Section III. The proposed TALB mechanism is presented in Section IV. The experimental results and concluding remarks are given in Section V and Section VI, respectively.

II. Background

A. Software-defined Networking (SDN)

SDN brings the benefits of flexible reconfiguration of network settings by separating control plane and data plane of the network [15]. In SDN, network devices focus on forwarding packets, while the intelligence of the control logic is implemented in the controller. The centralized controller can collect the network condition information (e.g., network topology, link usage) and configure the forwarding table in the switches. Each switch forwards the incoming packets by flow table which can be defined by the controller.

OpenFlow is an open standard and solution for implementing SDN, which defines the communication protocol between the controller and switches [16]. OpenFlow switches can modify the packet header according to the flow table. Furthermore, network statistic such as link utilization can be reported by OpenFlow switches for network management.

B. M2M Networks

M2M enables communication between devices such as mobile phones, computers, light, refrigerators, and other sensors without the assistance of humans. The devices in M2M networks usually report sensing data such as temperature, acceleration, or their status. The M2M data are usually sent by low power transmitters, such as Bluetooth or RFID. An M2M gateway can collect the M2M data and send the data to the cloud server using HTTP representational state transfer (REST) interfaces [17]. The cloud server can analyze the M2M data and manage the devices. For example, the cloud server can turn on/off the light for power saving or allow a user who is traveling to see the view of the camera at home. Clearly, such two M2M services demand different bandwidth and response time requirements.

C. Load Balancing

Load balancing techniques aim to balance the load between the serving nodes so that the service response time can be reduced. Based on the system loading such as the average CPU workload, a load balancer can decide which server handles the service requests using a predefined schedule or a round-robin (RR) method. When the servers are all overloaded or most of the servers are idle, the system can dynamically boot up a server to service more requests or shutdown a server to improve resource efficiency.

In the literature, load balancing schemes can be classified into two categories: reactive [18], [19] and proactive [20], [21] load balancing.

- The reactive methods decide where the serving node should be placed by comparing the imbalance level of the resource utilization with the given thresholds. The authors of [18] considered the unique feature of time-varying and overutilized resources in the servers. The proposed load balancing method dynamically assigns different weights to different resources according to their usage intensity and is shown to constantly retain the load balanced state. An adaptive load balancing method was proposed in [19] to equalize the gateways loads taking network conditions and sensor data generation rates into accounts. However, the reactive methods require extra delay to respond the load imbalances due to the dynamic load changes.
- The proactive methods make load balancing decisions based on the predictions of resource utilizations [20]. The authors of [21] proposed a stochastic load balancing scheme which incorporates the resource demand prediction with stochastic characterization. Although the proactive methods can reduce the service response time, the prediction-based load balancing may be inefficient for the highly dynamic load of M2M traffic due to the inaccuracy of resource prediction.

In M2M networks, load balancing techniques are essential to relieve the heavy loads caused by bursty traffic. The authors of [22] investigated the performance bottleneck issue when a medium load balancing server was introduced into a large-scale M2M service system as in [5]. In recent years, SDN has shown to efficiently assign client requests to multiple servers in M2M networks. A new M2M architecture using SDN for a flexible and dynamic network control was proposed in [23]. However, such SDN-based load balancing method incurs extra overhead caused by frequent controller process for updating flow tables [24]. This phenomenon is especially severe for M2M networks with a massive number of requests with different QoS requirements. To relieve the controller loading and increase the speed of switch forwarding, the authors of [11] proactively installed wildcard rules with a timeout that triggers the switch to delete the rule after a fixed time interval. Such a scheme does not consider link loading and cannot apply to multiple services.

III. Load Balancing in M2M Networks

Now we discuss the characteristics of M2M traffic and their resulting load balancing issues. Then we present how SDN can solve these load balancing issues in M2M networks.

A. M2M Traffic VS. Human Traffic

Traditional load balancing schemes are mainly designed for human-to-human (H2H) communications. However, characteristics of M2M traffic are different from the H2H traffic. The differences of M2M traffic and H2H traffic are expressed as follows.

- M2M traffic has smaller payload with more frequent access compared to H2H traffic. Typically, an M2M device continuously reports its current status to the application server.

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.2018.2812718, IEEE Internet of Things Journal

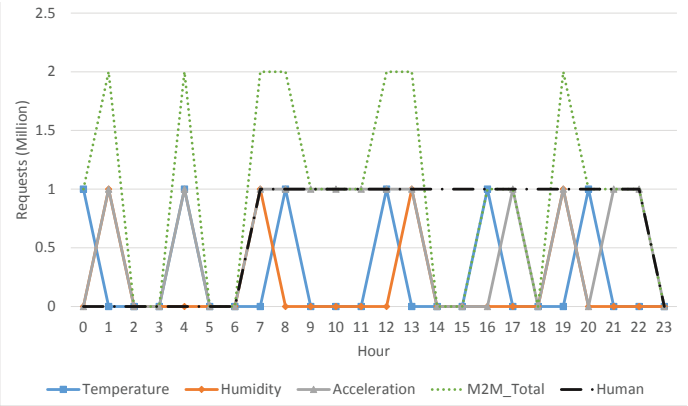

Fig. 2. Comparison of traffic pattern between M2M traffic and H2H traffic.

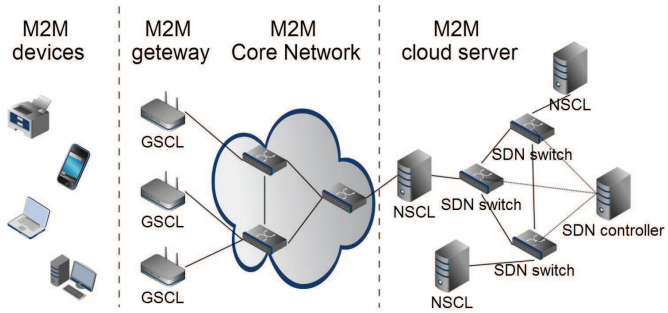

Fig. 3. Architecture of SDN-enabled M2M networks.

- M2M traffic is more uniformly generated throughout the day as compared to the H2H traffic. This because machines can work any time but human has a regular work cycle.

We conduct the experiments by using 10 Arduino microcontroller boards, each of which is equipped with multiple onboard sensors. These M2M devices are randomly deployed in our server room to collect surroundings data including temperature, humidity, and acceleration. The temperature and humidity data are recorded in different regular periods while the acceleration data is recorded when the data value exceeds a predefined threshold. For comparison, the H2H traffic is collected by monitoring the Internet activities of a smartphone user. In Section V, we will describe the network settings of our M2M testbed in more detail. Fig. 2 illustrates the request patterns of collected three M2M traffic and one H2H traffic. We can observe that the sum of the M2M traffic (i.e., M2M_Total curve) has more variation than the H2H traffic, increasing the difficulty of designing load balancing policies [17]. Thus, it is more difficult to schedule network and computing resources for M2M traffic compared to H2H traffic. Also, network congestion is one primary concern due to the bursts of M2M traffic at unpredictable times.

B. Load Balancing Issues

Figure 3 illustrates the architecture of SDN-enabled M2M networks. M2M requests from end devices to the cloud server can be divided into three stages. In the first stage, the end devices send these requests to the gateway which is referred as gateway service capability layer (GSCL) [5]. GSCL can directly forward these requests or collect more requests and then integrate into a batch process, depending on the service type. In the second stage, GSCL sends these requests to the cloud gateway which is referred as network service capability layer (NSCL). NSCL is the cloud server of M2M services and is responsible to handle these M2M requests. In the final stage, the NSCL processes these M2M requests or schedules other NSCL in cloud datacenter to handle it. NSCL can be deployed dynamically to satisfy the QoS requirements of various M2M services. Since M2M services depend on the NSCL to handle requests, how to achieve load balancing of these cloud servers is a critical issue. Conventional load balancing method places a medium server as a load balancer to schedule M2M requests to multiple cloud servers. However, such method faces the service bottleneck and SPOF problem. Furthermore, since M2M services are often communication-intensive, the links between NSCLs may suffer from network congestion.

C. SDN's Potential for Improving Load Balancing

With the global view of networks, SDN can resolve the aforementioned performance issues in M2M networks. The SDN controller can monitor the unpredictable M2M traffic of each link and control data forwarding easily, thereby avoiding network congestion. Besides, the SDN switch can be designed as a load balancer in which M2M requests can be distributed to different NSCLs by changing the packet header of the requests. In this paper, we call it SDN-based load balancer. Because any switch in the network can work as a load balancer, there is no SPOF issue for the SDN-based load balancer. Moreover, since the SDN controller can flexibly update the forwarding rules for each traffic flow, SDN can satisfy different response time requirements of different M2M services.

D. Objective

Although previous works have identified the potentials of SDN to improve load balancing in M2M networks, it is still an open issue to achieve different QoS requirements of multiple M2M services with the minimum overhead in the SDN controller. In this paper, we propose an SDN-enabled TALB mechanism for M2M networks. The novelty of the proposed scheme is mainly contributed by the elaborately designed link establishment algorithm which consists of traffic identification and traffic rerouting. To reduce the overhead in the SDN controller, flow tables in the proposed TALB mechanism are updated only when the server or network loading parameters exceed the QoS violation threshold. In Section V, we will demonstrate that the proposed scheme can provide server and network load balancing by conducting experiments with real sensor devices. Also, we will show that the proposed method can apply to different types of M2M requests including computation-intensive requests and communication-intensive requests. Table I compares the existing load balancing techniques, where the signs "○" and "×" mean that the proposed model "does" and "does not" consider the corresponding feature, respectively.

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.2018.2812718, IEEE Internet of Things Journal

TABLE I

COMPARISON OF THE PREVIOUS WORKS AND OUR PROPOSED SCHEME IN LOAD BALANCING.

	[5]	[11]	[23]	Ours
Balance server loading	O	O	O	O
Balance network loading	X	X	X	O
Avoid QoS violation	X	O	X	O
Apply to different types of M2M requests	X	X	O	O

From the standard aspect, our design is compliant with European telecommunications standards institute (ETSI) M2M functional architecture [25] including the GSCL and NSCL. For example, all the devices are mutually authenticated at the NSCL. Meanwhile, the delay requirements of an M2M service can be specified. Note that the integration of SDN is not specified in the current ETSI M2M standard. This work can be viewed as extensions to the M2M standard that leads to distributed NSCL operations by integrating SDN into M2M networks.

IV. TRAFFIC-AWARE LOAD BALANCING (TALB)

In this section, we detail the proposed SDN-based load balancing scheme for M2M networks.

A. Traffic-aware Load Balancing Mechanism

Figure 4 shows the information flows of TALB mechanism, which consists of four stages. In the first stage, the delay requirement of an M2M service is specified. We differentiate the M2M service according to the flag information in the packet header. Type of service (ToS) is a commonly used field in the TCP/IP packet header to differentiate the traffic types [26]. In the second stage, we monitor the network state by SDN and estimate the workload of NSCL server by the VM monitor. In the third stage, the delay performance is measured according to the workload of NSCL. If QoS violation occurs, the system will either find an available path and server combination or add a new NSCL server to satisfy the delay requirement. Otherwise, the system will check the utilization efficiency [27] of the NSCLs to decide whether shutdown NSCL or not. In the final stage, we configure the network setting by updating the flow entries in the SDN switches. For example, the packet headers of incoming packets, such as the MAC address and the IP address of the destination, are modified so that the packet can be forwarded to the low workload NSCL with low usage link. Note that in TALB mechanism we reconfigure the flow tables only when the server or network loading exceeds the predefined threshold.

Algorithm 1 shows the proposed TALB algorithm with complete procedures of path and server combination. To meet the delay requirement, one promising approach is to model the path and server determination problem as a constrained shortest path problem (CSP), which can be solved by Lagrangian relaxation based aggregated cost (LARAC) algorithm [16]. However, the complexity of the LARAC algorithm is exponential to both the number of links and servers [28]. To reduce the computational cost of the SDN controller, we adopt the simple depth-first search and then measure the response time R of the path. The first path and server combination which can satisfy the delay requirement is used. We will discuss the response time measurement in Section V.

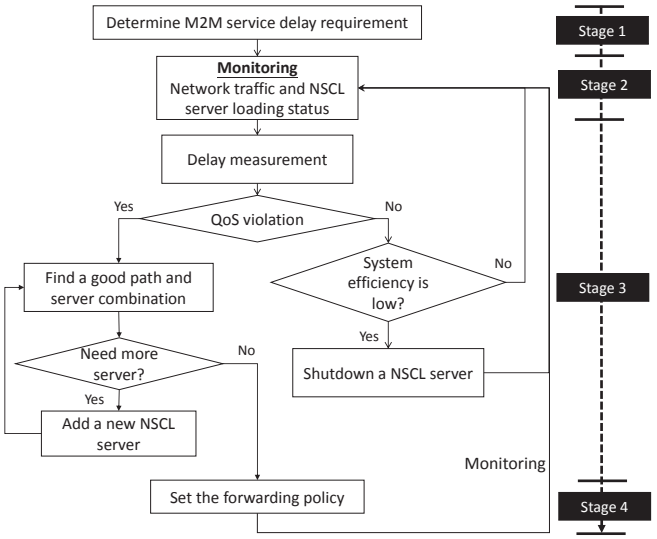

Fig. 4. System flow of the proposed traffic-aware SDN-based load balancing mechanism.

TABLE II

NOTATIONS IN THIS PAPER

Notations	Description
D	Delay requirement
$NSCLs$	NSCL pool
N_i	i^{th} NSCL
$INFO_i$	IP and MAC address table of N_i
S_i	Status of N_i, which is ready, booting or down
C_i	CPU workload of N_i
P_i	Assigned path for N_i
L	Link status table
D_i	Measured delay of N_i
E	Efficiency of the system based on CPU workload
B	Background traffic load
R	Response time

Figure 5 shows the system framework of the proposed TALB scheme with OpenFlow networks. The four components of the TALB scheme are network monitor, server monitor, traffic-aware load balancing module, and routing engine. Network monitor collects the information of network topology and link usage statistic from the OpenFlow switches. Next, server monitor obtains the workload information of NSCL servers via the open source Libvirt application programming interface (API) [29], including the CPU workload, memory usage, and disk status. Then the traffic-aware load balancing module evaluates the delay performance of the entire service by the collected network and workload information. With the delay performance evaluation, the traffic-aware load balancing mechanism can determine the optimal network configuration (e.g., routing path) to satisfy the delay requirement. Finally, the routing engine sets the network routing policy by updating the flow entries of the SDN switches.

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.2018.2812718, IEEE Internet of Things Journal

Algorithm 1 Traffic-aware Load Balancing Algorithm

for each $N_i \in NSCLs$ **do**
  initialize the IP and MAC address table of N_i;
  initialize the CPU workload C_i of N_i;
  initialize the assigned path P_i of N_i;
  initialize the network link status L;
end for
while true **do**
  for each $N_i \in NSCLs$ **do**
    update C_i by Libvirt API
    update L by SDN controller
  end for
  for each S_i is ready **do**
    measure D_i;
    if $D_i \geq D$ **then**
      loop
        if a path p found by depth-first search **then**
          for each N_i where S_i is ready **do**
            compute R with p and N_i
            if $R \leq D$ **then**
              the combination is found
            end if
          end for
        end if
      end loop
      if the combination is found **then**
        set the forwarding table of SDN switches
        update P_i
      else
        boot up a new NSCL by Libvirt API
      end if
    end if
  end for
  compute E of the system
  if E is low **then**
    shutdown a NSCL by Libvirt API
    update S_i of N_i
  end if
end while

B. M2M Traffic Rerouting using SDN

We highlight that our design separates the data forwarding and control logic of load balancing with the help of SDN. Instead of passing through a medium load balancer, all the M2M requests are forwarded by multiple SDN switches to achieve distributed load migration. On the other hand, a centralized controller with global network information makes the load balancing decision (as provided in Algorithm 1) via configuring the flow tables in the SDN switches. Fig. 6 shows an illustrative example of rerouting two classes of M2M traffic in the SDN switches. The flow table in the SDN switches consists of match rules and actions, which provides the forwarding policies for each flow with different delay requirements. Note that the priority and its corresponding ToS value of each M2M traffic are determined when the M2M device is authenticated at the NSCL. We incorporate the ToS

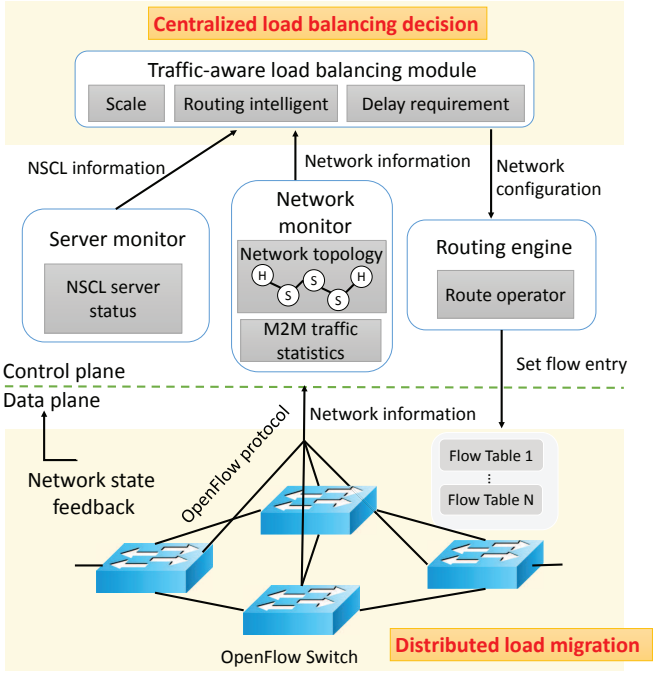

Fig. 5. System framework of the proposed traffic-aware SDN-based load balancing.

field of the packet header into the flow tables. By checking the ToS field in the packet header, each M2M traffic flow can be identified. As such, the SDN switch can redirect different incoming traffic to different paths in order to balance link utilization. Also, the traffic flow can thus be forwarded to the M2M servers over all possible paths in the entire network, as compared to the fixed forwarding path in the non-SDN load balancing scheme. It is worth mentioning that the proposed SDN-based load balancing scheme can be scaled to serve more types of traffic by simply storing more flow entries in the SDN switches.

Although we focus on one-way traffic in this paper, the proposed SDN-based solution can handle the two-way traffic flows. In our design, the flow rerouting mechanism follows the current SDN protocol. Also, the M2M device is required to register with the NSCL server before any application data can be transferred. Hence, with the registration information from the NSCL server, the M2M application server can deliver messages to the corresponding M2M devices.

C. Feedback Protocol for VM Utilization Enhancement

In practice, a new VM-based NSCL server might not be fully utilized since the VM boot up time is nontrivial [30]. To this end, we design the feedback protocols to inform the load balancer of the boot status of the new NSCL server. As shown in Fig. 7, the load balancer receives the network status from SDN switches and the CPU workload from the VM hypervisor [31] by Libvirt API. If QoS violation occurs, the load balancer will initiate a new NSCL. When the VM is booted and the service is ready, the load balancer will be notified. With the proposed protocol, the load balancer can know which NSCL

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.2018.2812718, IEEE Internet of Things Journal

Priority	Match	Action
0	port=2, VLAN=-1, src=00:01 :00:00:00:0e, dest=00:01:00:00:00:10, src=1.0.1.3, dest=1.0.1.2	Output 5
0	port=5, VLAN=-1, src=00:01:00:00:00:10, dest=00:01:00:00:00:0e, src=1.0.1.2, dest=1.0.1.3	Output 2
32767	port=4, src=00:01:00:00:00:0f, dest=00:01:00:00:00:0e, ethertype=0x0800, src=1.0.1.1, dest=1.0.1.3, TOS=12	Output 1
32767	port=1, src=00:01:00:00:00:0e, dest=00:01:00:00:00:01, ethertype=0x0800, src=1.0.1.3, dest=1.0.1.1	Output 1

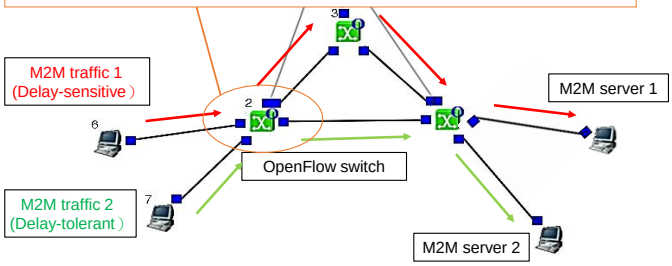

Fig. 6. An illustrative example of M2M traffic rerouting with SDN switches.

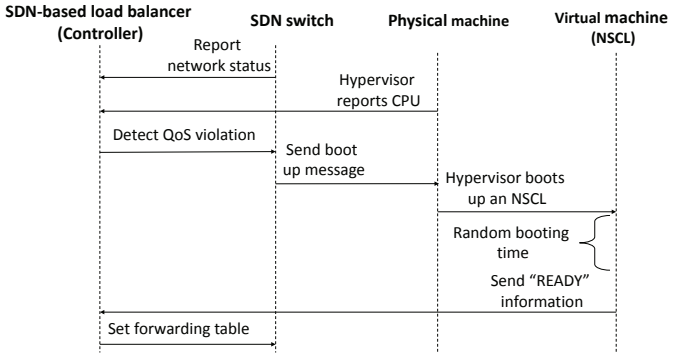

Fig. 7. The designed feedback protocol for enhancing VM utilization.

server is ready to handle requests and instantly forward the incoming requests to this new NSCL.

V. Experimental Results

To demonstrate the effectiveness of the proposed SDN-based load balancing scheme, we conduct experiments on the real testbed. We also assess the feasibility of M2M traffic rerouting with M2M devices connected through Ethernet interfaces and wireless connections.

A. Testbed Scenario

For our performance evaluation, we use Arduino Yun as a representative of a broad family of sensor devices berceuse of network programmability. As shown in Fig. 8, the roles of M2M devices in our testbed are classified as follows.

- *Sensor node:* Sensor node is composed of a set of sensors including temperature, humidity, and acceleration on a microcontroller board with a WiFi module. The sensor node performs sensing operation and communicates with the relay node using WiFi.
- *Relay node:* Relay node acts as a cluster head that receives messages from the sensor nodes and forwards the messages to the M2M gateway. The relay node is equipped with a Long Term Evolution (LTE) USB dongle that can transmit data to the M2M gateway through cellular links.

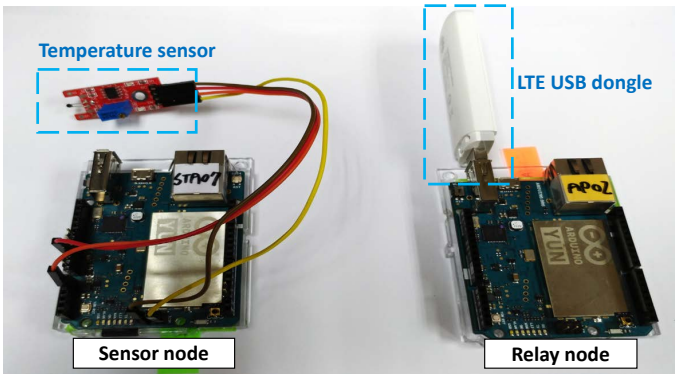

Fig. 8. Prototypes of sensor node used in our testbed.

Also, we use femtocell to act as a gateway to offload data traffic of the M2M networks from cellular networks [32]. For the SDN implementation, we adopt one Ryu SDN controller [33] and two Pica8 P3297 SDN switches [34], in which OpenFlow 1.0 and PicOS [35] are adopted, respectively.

Figure 9 illustrates our experiment environment, consisting of temperature sensors, SDN switches and LTE femtocell network devices. In our experiment setup, we randomly deploy 10 sensor nodes in our server room. A selected sensor node keeps sending sensed data every second, while the other nodes generate background traffic to create bandwidth contention. One of the four NSCL servers stores data into the database after receiving the data from the selected node while the other NSCL servers drop all the packets. Both the network interface cards (NIC) and the network cables have 1 Gbps capacity.

B. Response Time Measurement and Evaluation

To evaluate the delay performance of the M2M network, we perform experiments to establish the relation between the CPU workload, link usage, and response time. Fig. 10 shows the relation between CPU workload and the response time under different network status. By using linear regression, we obtain the relation between the response time R and CPU workload C with different background traffic load B, as shown in (1). One can observe that the background traffic load B highly affects the service response time. For example, when $B = 40$ Mbps, R increases tremendously even with small addition of CPU workload. This observation demonstrates the necessity of considering link usage to design the load balancing policy. With the response time evaluation, the TALB module can decide the path and server combination as shown in the 18th step of Algorithm 1.

C. Response Time Comparison between Normal and SDN-based Load Balancer

Figure 11 compares the response time between the open source load balancer NGINX [36] and the proposed SDN-based load balancer with $B = 20$ Mbps. It is observed that the proposed SDN-based load balancer can reduce up to 50%

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.2018.2812718, IEEE Internet of Things Journal

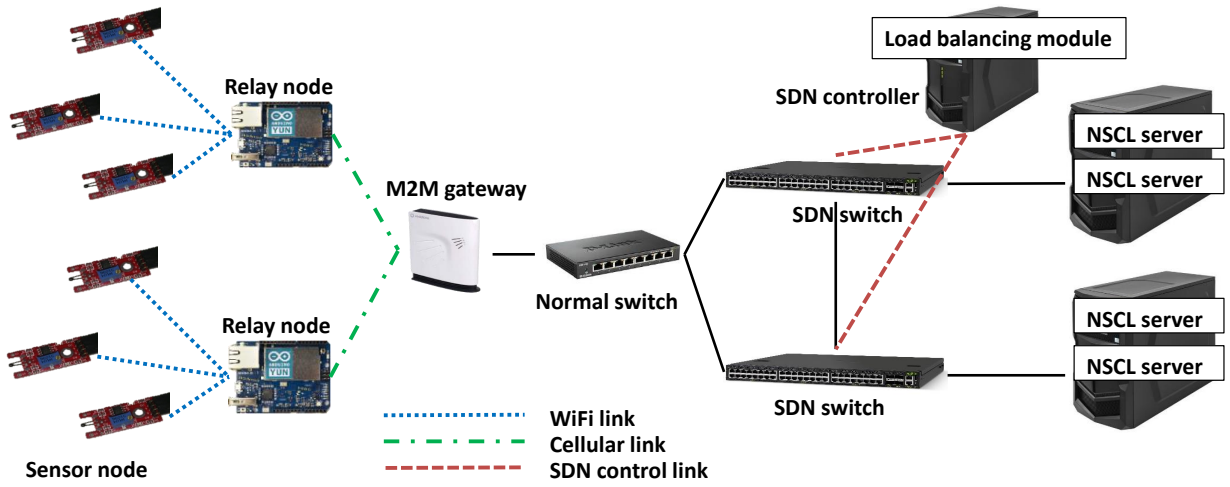

Fig. 9. Experimental testbed for the proposed load balancing scheme using temperature sensors, SDN switches, and LTE femtocells.

$$\mathrm{R} = \begin{cases} 0.0005 \times C^3 - 0.0567 \times C^2 + 2.0504 \times C - 4.0155 & , \quad \mathrm{B} = 0 \ \ \mathrm{Mbps} \\ -0.0006 \times C^3 + 0.0638 \times C^2 - 0.2334 \times C + 20.082 & , \quad \mathrm{B} = 20 \ \ \mathrm{Mbps} \\ 0.0022 \times C^3 - 0.2627 \times C^2 + 10.752 \times C - 16.459 & , \quad \mathrm{B} = 40 \ \ \mathrm{Mbps} \end{cases} \tag{1}$$

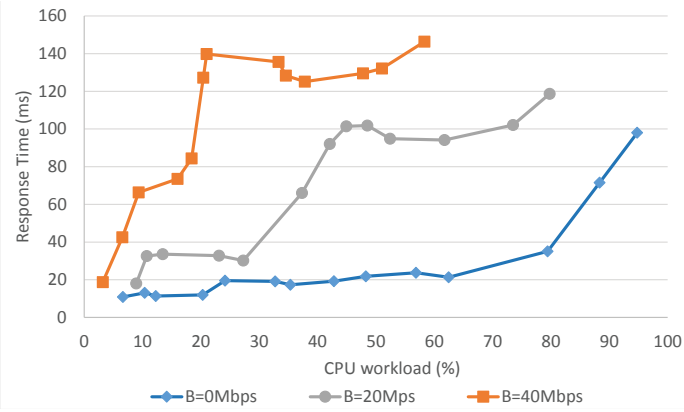

Fig. 10. Relation between response time and CPU workload under different background traffic load B.

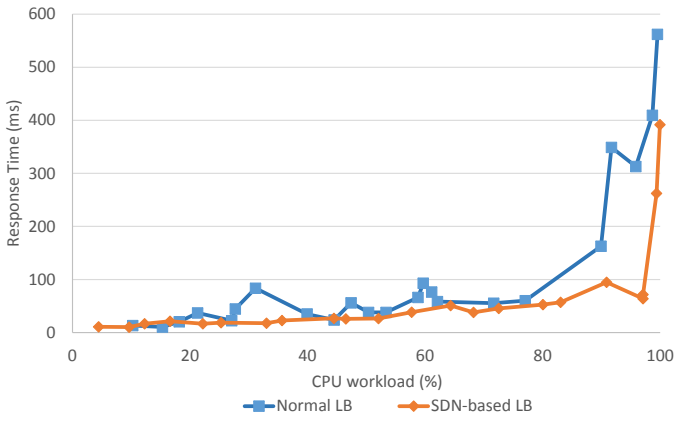

Fig. 11. Response time comparison between normal load balancer and SDN-based load balancer under different CPU workloads.

of response time compared to NGINX. Moreover, the response time of the proposed scheme has smaller variance compared to the normal load balancer. This is because the proposed scheme considers the impact of link usage when performing load balancing.

D. VM Utilization Improvement

Figure 12 demonstrates how the VM utilization can be improved when server scaling up. In the two deployed NSCLs, only NSCL1 is booted initially. The proposed SDN-based load balancer detects a delay violation at 00:25, thereby booting up a new NSCL (NSCL2). The CPU workload of NSCL2 approaches 100% due to the booting process. At 00:37, the NSCL is ready to handle requests and notifies the load balancer via READY message. Then the load balancer can set configuration to forward the incoming requests to the NSCL. It is shown that the idle time of the booted NSCL is shorter than two seconds. We note that the CPU workloads of the two NSCLs alternate with each other. This is because the proposed scheme performs traffic migration only when delay violation occurs, instead of balancing the workloads between the servers.

E. Dynamic Network Reconfiguration Frequency

Figure 13 shows the configuration frequency of the proposed TALB scheme and the typical SDN load-balancing approach which only considers the impact of CPU workload. If the response time increases, the proposed load balancer sets a new configuration to the switches at the time indicated with green lines. However, the typical SDN load-balancing approach is sensitive to workload changes. We observe that the TALB scheme can reduce the configuration times by 65% in comparison to the typical SDN load-balancing approach, resulting in much less overhead for the SDN control plane.

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.2018.2812718, IEEE Internet of Things Journal

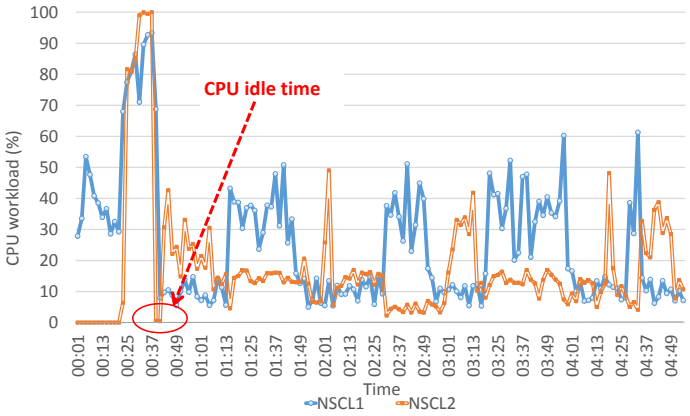

Fig. 12. CPU workload of the NSCLs in a scaling process.

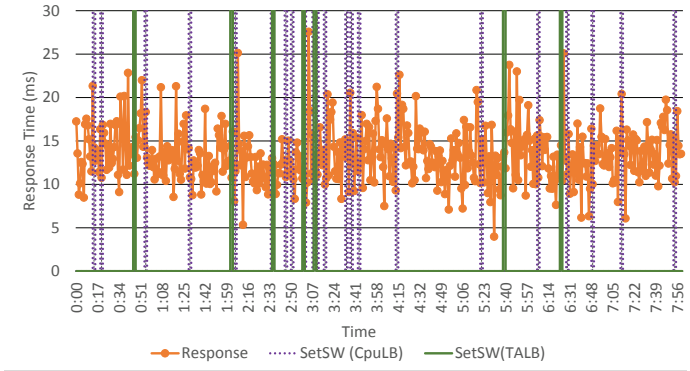

Fig. 13. Network configuration frequency in different SDN-based load balancing schemes.

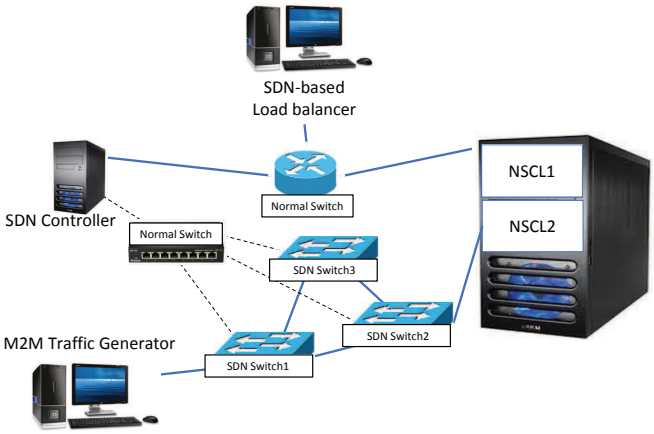

Fig. 14. M2M cloud testbed for SDN-based load balancing techniques.

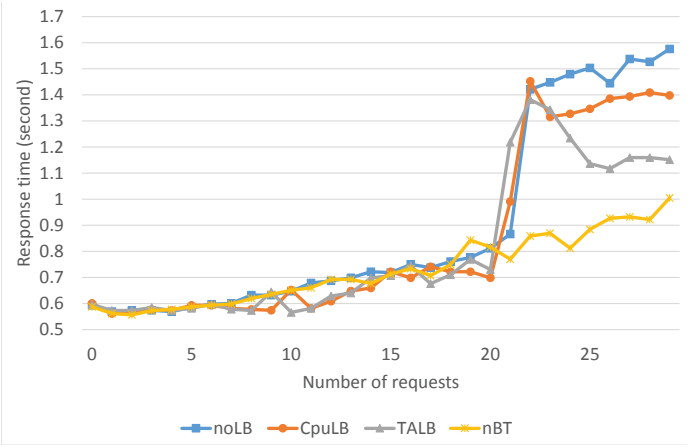

Fig. 15. Response time comparison of different load balancing schemes with computation-intensive requests.

F. Response Time Comparison for Different Load Balancing Policies with Bursty Arrivals

Here we compare the response time performance with different types of M2M requests. To simulate the congested network, we use Iperf traffic tool [37] to generate M2M traffic requests. The following two types of M2M requests are considered in our experiments.

- *Computation-intensive request:* One typical operation for M2M devices is to offload high computational tasks to the cloud servers. In this case, we implement a test application process to perform bubble sorting which demands high computational cost. We set the traffic arrival rate of computation-intensive request to 10 flows/sec in which each flow generates 1 Mb of traffic.
- *Communication-intensive request:* Most of the sensing applications generate communication-intensive requests in which high traffic rates with bursty arrivals can lead to network congestion. In this case, the traffic arrival rate is set in the range from 100 flows/sec to 1,000 flows/sec in which each flow generates 1 Mb of traffic.

Figure 14 shows the considered experimental environment, where two VMs with Libvirt API are deployed on a physical machine and three SDN switches are connected to the controller by a normal switch. For comparison, we consider the other two load balancing schemes: no load balancing (noLB) and CPU-based load balancing (CpuLB) [5]. For the CpuLB scheme, CPU workload threshold is set to 80% because the response time performance becomes worse when CPU workload exceeds 80% as shown in Fig. 11. For the TALB scheme, the response time requirement is set to one second. The response time is measured by considering the time interval from the instant when the traffic is generated until the instant when the response is received. We test 30 continuous requests with a burst traffic occurring at the 20^{th} request. The video demonstration of this load balancing application is available online [38].

Figure 15 shows the response time in comparison with computation-intensive requests. The burst traffic causes severe degradation in terms of response time at the 20^{th} request. As seen from the figure, the CPU-based load balancing scheme can reduce the response time by 10% compared to no load balancing scheme. The proposed TALB scheme can further reduce 30% of response time. We note that the no burst traffic (nBT) case can be regarded as the optimal performance in the considered scenario.

Figure 16 shows the response in time comparison with communication-intensive requests. It is shown that the CPU-based load balancing cannot improve response time in the case of communication-intensive requests. On the contrary,

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.2018.2812718, IEEE Internet of Things Journal

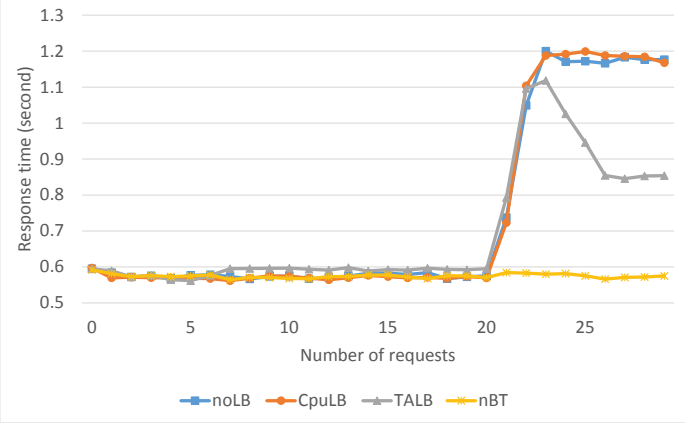

Fig. 16. Response time comparison of different load balancing schemes with communication-intensive requests.

our scheme can reduce 50% of the response time compared to no load balancing scheme. The performance gap between the no burst traffic with our scheme results from the unavoidable delay caused by switch processing. Hence, the proposed SDN-based TALB scheme can effectively reduce the response time, especially in the communication-intensive M2M services.

VI. Conclusions

We have investigated the potentials of SDN switches as a load balancing technique in M2M networks. A traffic-aware load balancing scheme is developed to satisfy different QoS requirements of M2M traffic by exploiting the features of instant traffic identification and dynamic traffic rerouting in SDN. Our experimental results show that the proposed SDN-based load balancer has better response time performance than the traditional load balancer. Although we focus on load balancing technique for M2M networks, the proposed scheme can be applied to Internet of things (IoT) networks in which human and machines communicate through IP-based networks with their data delivered to a cloud. In this case, SDN can provide benefits to load balancing by introducing flow-based networking to different types of traffic. In the future, an interesting research topic is to incorporate IoT security in the proposed traffic-aware load balancing scheme.

References

[1] C. L. Hu, H. T. Huang, C. L. Lin, N. H. M. Anh, Y. Y. Su, and P. C. Liu, "Design and implementation of media content sharing services in home-based IoT networks," in *IEEE International Conference on Parallel and Distributed Systems*, 2013.

[2] Y. Cao, T. Jiang, and Z. Han, "A survey of emerging M2M systems: Context, task, and objective," *IEEE Internet of Things Journal*, vol. 3, no. 6, pp. 1246–1258, 2016.

[3] A. H. Tsai, L. C. Wang, J. H. Huang, and T. M. Lin, "Overload control for machine type communications with femtocells," in *IEEE Vehicular Technology Conference (VTC Fall)*, 2012.

[4] W. Twayej, M. Khan, and H. S. Al-Raweshidy, "Network performance evaluation of M2M with self organizing cluster head to sink mapping," *IEEE Sensors Journal*, vol. 17, no. 15, pp. 4962–4974, 2017.

[5] M. Corici, H. Coskun, A. Elmangoush, A. Kurniawan, T. Mao, T. Magedanz, and S. Wahle, "OpenMTC: Prototyping machine type communication in carrier grade operator networks," in *IEEE Globecom Workshops (GC Wkshps)*, 2012.

[6] H. Zhang, Y. Xiao, S. Bu, D. Niyato, F. R. Yu, and Z. Han, "Computing resource allocation in three-tier IoT fog networks: A joint optimization approach combining stackelberg game and matching," *IEEE Internet of Things Journal*, 2017.

[7] H. Huang, S. Guo, J. Wu, and J. Li, "Joint middlebox selection and routing for software-defined networking," in *IEEE International Conference on Communications (ICC)*, 2016.

[8] M. Koerner and O. Kao, "Multiple service load-balancing with OpenFlow," in *IEEE 13th International Conference on High Performance Switching and Routing*, 2012.

[9] A. Al-Najjar, S. Layeghy, and M. Portmann, "Pushing SDN to the end-host, network load balancing using OpenFlow," in *IEEE International Conference on Pervasive Computing and Communication Workshops*, 2016.

[10] W.-H. Liao, S.-C. Kuai, and C.-H. Lu, "Dynamic load-balancing mechanism for software-defined networking," in *IEEE International Conference on Networking and Network Applications*, 2016.

[11] T.-L. Lin, C.-H. Kuo, H.-Y. Chang, W.-K. Chang, and Y.-Y. Lin, "A parameterized wildcard method based on SDN for server load balancing," in *IEEE International Conference on Networking and Network Applications*, 2016.

[12] V. G. Nguyen, A. Brunstrom, K. J. Grinnemo, and J. Taheri, "SDN/NFV-based mobile packet core network architectures: A survey," *IEEE Communications Surveys Tutorials*, vol. 19, no. 3, pp. 1567–1602, 2017.

[13] M. Mu, M. Broadbent, A. Farshad, N. Hart, D. Hutchison, Q. Ni, and N. Race, "A scalable user fairness model for adaptive video streaming over SDN-assisted future networks," *IEEE Journal on Selected Areas in Communications*, vol. 34, no. 8, pp. 2168–2184, 2016.

[14] Y.-J. Chen, Y.-H. Shen, and L.-C. Wang, "Traffic-aware load balancing for M2M networks using SDN," in *IEEE International Conference on Cloud Computing Technology and Science*, 2014.

[15] B. R. Al-Kaseem and H. S. Al-Raweshidy, "SD-NFV as an energy efficient approach for M2M networks using cloud-based 6LoWPAN testbed," *IEEE Internet of Things Journal*, vol. PP, no. 99, pp. 1–1, 2017.

[16] Y.-J. Chen, F.-Y. Lin, L.-C. Wang, and B.-S. Lin, "A dynamic security traversal mechanism for providing deterministic delay guarantee in SDN," in *IEEE 15th International Symposium on a World of Wireless, Mobile and Multimedia Networks*, 2014.

[17] J. Kim, J. Lee, J. Kim, and J. Yun, "M2M service platforms: survey, issues, and enabling technologies," *IEEE Communications Surveys & Tutorials*, vol. 16, no. 1, pp. 61–76, 2014.

[18] L. Chen, H. Shen, and K. Sapra, "RIAL: Resource intensity aware load balancing in clouds," in *IEEE Conference on Computer Communications*, 2014.

[19] Y. Miao, S. Vural, Z. Sun, and N. Wang, "A unified solution for gateway and in-network traffic load balancing in multihop data collection scenarios," *IEEE Systems Journal*, vol. 10, no. 3, pp. 1251–1262, 2016.

[20] V. Nae, R. Prodan, and T. Fahringer, "Cost-efficient hosting and load balancing of massively multiplayer online games," in *IEEE/ACM International Conference on Grid Computing (GRID)*, 2010.

[21] L. Yu, L. Chen, Z. Cai, H. Shen, Y. Liang, and Y. Pan, "Stochastic load balancing for virtual resource management in datacenters," *IEEE Transactions on Cloud Computing*, 2016.

[22] S. Kitagami, Y. Kaneko, and T. Suganuma, "Method of autonomic load balancing for long polling in M2M service system," in *International Conference on Advanced Information Networking and Applications Workshops*, 2012.

[23] A. A. Corici, R. Shrestha, G. Carella, A. Elmangoush, R. Steinke, and T. Magedanz, "A solution for provisioning reliable M2M infrastructures using SDN and device management," in *International Conference on Information and Communication Technology*, 2015.

[24] A. Bradai, K. Singh, T. Ahmed, and T. Rasheed, "Cellular software defined networking: A framework," *IEEE communications magazine*, vol. 53, no. 6, pp. 36–43, 2015.

[25] Y. Cao, T. Jiang, and Z. Han, "A survey of emerging M2M systems: Context, task, and objective," *IEEE Internet of Things Journal*, vol. 3, no. 6, pp. 1246–1258, 2016.

[26] C. Hue, Y.-J. Chen, and L.-C. Wang, "Traffic-aware networking for video streaming service using SDN," in *IEEE 34th International Performance Computing and Communications Conference*, 2015.

[27] R. Poddar, A. Vishnoi, and V. Mann, "HAVEN: Holistic load balancing and auto scaling in the cloud," in *IEEE International Conference on Communication Systems and Networks*, 2015.

[28] Y. J. Chen, L. C. Wang, F. Y. Lin, and B. S. Lin, "Deterministic quality of service guarantee for dynamic service chaining in software defined

networking," *IEEE Transactions on Network and Service Management*, vol. PP, no. 99, pp. 1–1, 2017.
[29] "Libvirt Team. libvirt: The virtualization api," http://libvirt.org.
[30] M. Mao and M. Humphrey, "A performance study on the VM startup time in the cloud," in *IEEE Fifth International Conference on Cloud Computing*, 2012.
[31] R. Cziva, S. Jouët, D. Stapleton, F. P. Tso, and D. P. Pezaros, "SDN-based virtual machine management for cloud data centers," *IEEE Transactions on Network and Service Management*, vol. 13, no. 2, pp. 212–225, 2016.
[32] W. Nasrin and J. Xie, "Signaling cost analysis for handoff decision algorithms in femtocell networks," in *IEEE International Conference on Communications (ICC)*, 2017.
[33] "Ryu SDN Framework," accessed: Nov. 22, 2017. [Online]. http://osrg.github.io/ryu/.
[34] "P-3297 DatasheetPica8," accessed: Nov. 28, 2017. [Online]. http://www.pica8.com/.
[35] "Pica8 Operating System," accessed: Nov. 28, 2017. [Online]. http://www.pica8.com/products/picos/.
[36] A. H. Ngu, M. Gutierrez, V. Metsis, S. Nepal, and Q. Z. Sheng, "IoT middleware: A survey on issues and enabling technologies," *IEEE Internet of Things Journal*, vol. 4, no. 1, pp. 1–20, 2017.
[37] Y. Qiao, X. Wang, G. Fang, and B. Lee, "Doopnet: An emulator for network performance analysis of Hadoop clusters using Docker and Mininet," in *IEEE Symposium on Computers and Communication*, 2016.
[38] "Demo Video of The Designed Load Balancing Application," accessed: Nov. 28, 2017. [Online]. http://youtu.be/6VHJicmsi24/.

Yu-Jia Chen received the B.S. degree and Ph.D. degree in electrical engineering from National Chiao Tung University, Taiwan, in 2010 and 2016, respectively. He is currently a postdoctoral fellow in National Chiao Tung University. His research interests include network coding for secure storage in cloud datacenters, software defined networks (SDN), and sensors-assisted applications for mobile cloud computing. Dr. Chen has published 22 conference papers and 4 journal papers. He is holding three US patent and three ROC patent.

Li-Chun Wang (M'96 – SM'06 – F'11) received the B.S. degree from National Chiao Tung University, Taiwan, R.O.C. in 1986, the M.S. degree from National Taiwan University in 1988, and the Ms. Sci. and Ph. D. degrees from the Georgia Institute of Technology , Atlanta, in 1995, and 1996, respectively, all in electrical engineering.

From 1990 to 1992, he was with the Telecommunications Laboratories of Chunghwa Telecom Co. In 1995, he was affiliated with Bell Northern Research of Northern Telecom, Inc., Richardson, TX. From 1996 to 2000, he was with AT&T Laboratories, where he was a Senior Technical Staff Member in the Wireless Communications Research Department. Since August 2000, he has joined the Department of Electrical and Computer Engineering of National Chiao Tung University in Taiwan and is the current Chairman of the same department. His current research interests are in the areas of radio resource management and cross-layer optimization techniques for wireless systems, heterogeneous wireless network design, and cloud computing for mobile applications.

Dr. Wang won the Distinguished Research Award of National Science Council, Taiwan in 2012, and was elected to the IEEE Fellow grade in 2011 for his contributions to cellular architectures and radio resource management in wireless networks. He was a co-recipient(with Gordon L. Stuber and Chin-Tau Lea) of the 1997 IEEE Jack Neubauer Best Paper Award for his paper "Architecture Design, Frequency Planning, and Performance Analysis for a Microcell/Macrocell Overlaying System," IEEE Transactions on Vehicular Technology, vol. 46, no. 4, pp. 836-848, 1997. He has published over 200 journal and international conference papers. He served as an Associate Editor for the IEEE Trans. on Wireless Communications from 2001 to 2005, the Guest Editor of Special Issue on "Mobile Computing and Networking" for IEEE Journal on Selected Areas in Communications in 2005, "Radio Resource Management and Protocol Engineering in Future Broadband Networks" for IEEE Wireless Communications Magazine in 2006, and "Networking Challenges in Cloud Computing Systems and Applications," for IEEE Journal on Selected Areas in Communications in 2013, respectively. He is holding 10 US patents.

Meng-Chieh Chen received the B.S. degree in Communication Engineering from National Central University, Taiwan, in 2016. He is currently working toward the M.S. degree with the Institute of Network Engineering, National Chiao Tung University, Hsinchu, Taiwan. His current research interests include Industrial Internet of Things, Machine to Machine Protocol, Group-based Approach in Internet of Things.

Pin-Man Huang received the B.S. degree in Communication Engineering from National Central University, Taiwan, in 2017. She is currently working toward the M.S. degree with the Institute of Communications Engineering, National Chiao Tung University, Hsinchu, Taiwan. Her current research interests include Internet of Things and Software-defined network.

Pei-Jung Chung received Dr.-Ing. in 2002 from Ruhr-Universitat Bochum, Germany with distinction. From 2002 to 2004 she held a post-doctoral position at Carnegie Mellon University and University of Michigan, Ann Arbor, USA, respectively. From 2004 to 2006 she was Assistant Professor with National Chiao Tung University, Hsin Chu, Taiwan. From 2006 to 2013, she was Lecture at the Institute for Digital Communications, School of Engineering, the University of Edinburgh, UK. Since 2013 she is with Delta Electronics, Inc. , Taipei, Taiwan. She is senior member of IEEE and associate member of IEEE Signal Processing Society Sensor Array Multichannel (SAM) Technical Committee. Her research interests include array processing, wireless MIMO systems, distributed processing in wireless sensor networks and their applications in industrial manufacturing.