Accepted Manuscript

A novel QoS-enabled load scheduling algorithm based on reinforcement learning in software-defined energy internet

Chao Qiu, Shaohua Cui, Haipeng Yao, Fangmin Xu, F. Richard Yu, Chenglin Zhao

PII: S0167-739X(18)30308-X
DOI: https://doi.org/10.1016/j.future.2018.09.023
Reference: FUTURE 4458

To appear in: *Future Generation Computer Systems*

Received date : 12 February 2018
Revised date : 1 August 2018
Accepted date : 6 September 2018

Please cite this article as: C. Qiu, et al., A novel QoS-enabled load scheduling algorithm based on reinforcement learning in software-defined energy internet, *Future Generation Computer Systems* (2018), https://doi.org/10.1016/j.future.2018.09.023

This is a PDF file of an unedited manuscript that has been accepted for publication. As a service to our customers we are providing this early version of the manuscript. The manuscript will undergo copyediting, typesetting, and review of the resulting proof before it is published in its final form. Please note that during the production process errors may be discovered which could affect the content, and all legal disclaimers that apply to the journal pertain.

ACCEPTED MANUSCRIPT

A Novel QoS-Enabled Load Scheduling Algorithm Based on Reinforcement Learning in Software-Defined Energy Internet

Chao Qiu*, Shaohua Cui†, Haipeng Yao‡, Fangmin Xu*, F. Richard Yu§, *Fellow, IEEE*, and Chenglin Zhao*
* Key Lab. of Univ. Wireless Comm., Beijing Univ. of Posts and Telecom., Beijing, P.R. China
Email: zjkchouchao@bupt.edu.cn clzhao@bupt.edu.cn xufm@bupt.edu.cn
† China petroleum technology & development corporation
Email: cuish@cptdc.cnpc.com.cn
‡ State Key Lab. of Networking and Switching Tech., Beijing Univ. of Posts and Telecom., P.R. China
Email: yaohaipeng@bupt.edu.cn
§ Depart. of Systems and Computer Eng., Carleton Univ., Ottawa, ON, Canada
Email: Richard.Yu@Carleton.ca

***Abstract*—Recently, smart grid and Energy Internet (EI) are proposed to solve energy crisis and global warming, where improved communication mechanisms are important. Software-defined networking (SDN) has been used in smart grid for real-time monitoring and communicating, which requires steady web-environment with no packet loss and less time delay. With the explosion of network scales, the idea of multiple controllers has been proposed, where the problem of load scheduling needs to be solved. However, some traditional load scheduling algorithms have inferior robustness under the complicated environments in smart grid, and inferior time efficiency without pre-strategy, which are hard to meet the requirement of smart grid. Therefore, we present a novel controller mind (CM) framework to implement automatic management among multiple controllers. Specially, in order to solve the problem of complexity and pre-strategy in the system, we propose a novel Quality of Service (QoS) enabled load scheduling algorithm based on reinforcement learning in this paper. Simulation results show the effectiveness of our proposed scheme in the aspects of load variation and time efficiency.**

***Index Terms*—Reinforcement learning, software-defined networking, load scheduling, Quality of Service (QoS), energy Internet, smart grid.**

I. Introduction

ENERGE resources crisis and global warming have become two global concerns [1]. As reasonable solutions, smart grid [2] and Energy Internet (EI) [3] are seen as the new generation of energy provision paradigm, where improved communication mechanisms are important to enable end-to-end communication. Software-defined networking (SDN) [4] is seen as a promising paradigm shift to reshape future network architecture, as well as smart grid and EI, called software-defined EI (SDEI). Using SDN enables to improve smart grid and EI by providing an abstraction of underlying network resources, forming global view for applications from upper layers, and decoupling infrastructures and control plane to enhance the flexibility and reliability of the system [5]. Noteworthy, the control plane is considered as the brain of SDN [6]. With the explosion of network scales and network traffic, overload in a single controller is one of the most intractable issues [7]. There is a growing consensus that the control plane should be designed as a multiple controllers plane to constitute a logically centralized but physically distributed model [8]–[10]. So far, the issues of multiple controllers have been studied in literature. Except for addressing the consistency problem of global view among distributed control plane, another key issue is how to schedule loads among multiple controllers so as to mitigate the risk of overloads and failures in one single controller.

On the other hand, the most important application of SDN in smart grid is real-time monitoring and communicating. It follows that these applications require steady web-environment with no packet loss and less time delay to keep high accuracy and real time capability [11].

Traditionally, load scheduling algorithms make load scheduling decisions after the overload problems have happened [12]. In general, the traditional algorithms have three steps, including collecting load information, making load scheduling decisions, and sending load scheduling commands to the corresponding controllers. For example, the work in [13], load scheduling decision is made after the problem of overload. In addition, current CPU usage, current memory usage, current hard disk usage, and weight coefficient need to be exchanged among controllers when new load scheduling decision is made, which occupies lots of extra time so as to decrease time efficiency.

Recently, Machine learning (ML) has emerged as a novel

ACCEPTED MANUSCRIPT

technique, which can be used to tackle the above challenges. Reinforcement Learning (RL) is regarded as an important branch of ML to solve complex control problems. RL is quite different from the traditional management and operation methods. It develops computer models by training datasets, which resolves the problems without being explicitly programmed to, but learning from environments.

In this paper, we present a novel controller mind (CM) framework in distributed SDEI to implement automatic management among multiple controllers. Specifically, in order to solve the problem of complexity and pre-strategy in the system, we propose a novel Quality of Service (QoS) enabled load scheduling algorithm based on RL. The distinct features of this paper are as follows.

- It is the first time to bring reinforcement learning approach in the cooperation among controllers.
- We propose a CM framework to solve the problems of cooperation among multiple controllers automatically and intelligently.
- We formulate QoS-enabled load scheduling problem as an optimization problem.
- In order to solve the optimization problem, we propose a RL approach in this paper. We describe this problem as a Markov decision process, defining state space, action space, and reward function. We train historical data to learn load scheduling strategy in advance and offline.
- Simulation results with different system parameters are presented to show the effectiveness of our proposed scheme. It is illustrated that the performance of SDEI can be significantly improved with the proposed RL-based controller mind in the aspects of load variation and time efficiency.

The rest of this paper is organized as follows. Section II presents some related works, some motivations and enablers about this issue. In Section III, we give system description, followed by system model in Section IV. Section V presents the overview of reinforcement learning and formulates the problem. Simulation results are given and discussed in Section VI. Finally, conclusions and future works are presented in Section VII.

II. Related Work

Recently, the idea of logically centralized but physically distributed control plane is more and more popular. On the one hand, the problem of consistency among multiple controllers is addressed in some works, such as HyperFlow [14], Onix [15], ONOS [16], DISCO [17], Kandoo [18] and Balanceflow [19].

On the other hand, the issue of distributed controllers is how to allocate loads among controllers, as so to mitigate the risk of overload and failure. This problem has been studied in many works. It can be divided into three categories, namely *centralized decision*, *distributed decision* and *hybrid decision*.

1) Centralized Decision: ElastiCon in [20] treated multiple controllers as a controller resource pool and had a load balancer to expand or shrink the controller pool as needed, so as to address the problem of load imbalance. Sherwood *et al.* proposed FlowVisor in [21]. It was a network visualization layer between the control plane and the data plane to centrally manage controller resource pool. Following this research, Yutai *et al.* in [13] used the similar architecture to implement the weight coefficients of load balancing strategy.

Additionally, some researches have employed a centralized manager. *Pratyaastha* in [22] used a centralized manager to partition SDN application and referred to the dependencies between applications and switches. Chu *et al.* in [23] proposed a coordinator whose responsibility was to maintain the load information table of global controllers. Based on this information table, the coordinator decided how to balance the load among controllers.

In the above works, the operations of load adaptation are all taken after overload has happened. What is worse, lots of steps need to be taken before the final load adaptation decision is executed, such as collecting load measurements, determining actions, and sending load decisions to the corresponding controllers. With such passive and lagging load adaptation, it is difficult to meet the requirements of high time efficiency in SDEI.

2) Distributed Decision: In order to solve the problems in centralized decision schemes, some distributed decision load balancing schemes have been proposed. DALB was proposed in [24], where controllers made decisions locally, and the overloaded controller proactively collected others' load information before making decision. Following this research, Yu *et al.* in [25] also allowed controllers to make decisions locally, but periodical reporting its load information to all others.

From the perspective of time efficiency, the load balancing decisions are also made after the problem of overload has happened. And masses of signaling is exchanged. Therefore, the time efficiency in distributed decision scheme is also not satisfactory.

3) Hybrid Decision: Yao *et al.* in [26] proposed a Hybridflow architecture, which consisted of several cluster controllers and one super controller. In lower overload, Hybridflow allowed cluster controllers to make balancing decisions locally, and in higher overload, super controller made balancing decisions globally with the help of obtained load information. Additionally, in order to reduce waiting-time in super controller, Hader *et al.* [27] defined a ClusterVector (CV) to contain addresses and load status of controllers.

Similarly, the hybrid decision is also a passive and lagging load adaptation method, which has inferior time efficiency.

As we can see that no matter which methods, they all need

ACCEPTED MANUSCRIPT

three or more steps and lots of signaling interactions to adjust to load scheduling after overload has happened. It seems that improving the time efficiency of load scheduling among controllers is barely possible to be solved by traditional methods. However, some novel researches have offered the other solutions where machine learning is employed. Apiletti *et al.* proposed a self-learning network analyzer in [28]. It was a generic and self-tuning tool to dig the knowledge from network traffic measurements, so as to control the network configuration automatically. Batu *et al.* in [29] used machine learning tools to proactively control what to cache at base stations. Fan *et al.* in [30] proposed a bandwidth control algorithm to increase the throughput of cellular network by exploring users data and network data. Albert *et al.* considered a Knowledge-Defined Networking [31] which took the full advantages of centralized management from SDN and the analysis capacity from artificial intelligence to solve several problems, including routing in an overlay network, resource control in an NFV scenario, knowledge extraction from network logs, and short or long-term network planning. Liu *et al.* in [32] studied a deep learning based content popularity prediction system. By this system, cache control strategy was improved. Moreover, Cui *et al.* in [33] learned a stochastic learning based control scheme in MIMO. They used this online learning algorithm to control dynamic clustering and power allocation.

Thus we can see more and more researches have begun to leverage big data analysis and machine learning tools to solve some complex control problems.

Inspired by the time efficiency problem of load adaptation among controllers and some researches that utilize machine learning to solve complex control problem, we consider to use machine learning to control and manage among multiple controllers. There are some enablers to apply machine learning in distributed SDEI, including

- As the brain of SDEI, the control plane almost has all data of the network, which provides training datasets for machine learning.
- Controllers have powerful computing capacities, which is necessary in machine learning.
- Due to the logically centralized control in the control plane, the offline learning results can be executed quickly.

In conclusion, it is necessary and achievable to introduce learning methods into the control plane. In this paper, we propose a reinforcement learning approach to solve the problem of load scheduling among controllers for the first time. Specially, we use a Q-learning approach, which is a typical algorithm in reinforcement learning. Considering the slow convergence rate of Q-learning and the periodicity of users' behaviors, we train the historical load data offline, so as to make load scheduling decision ahead of time.

III. System Description

In this section, we first give a brief overviews of energy Internet and software-defined energy Internet. Then the CM framework in SDEI is presented.

A. Energy Internet

With energy crisis and limitation around world, how to use renewable energy has attracted lots of attentions, ranging from government and industry to academia. Here, the development of renewable energy, as well as information and communication technologies (ICTs) are two key enablers of energy Internet [34]. Thus, energy Internet can be seen as an energy-utilizing system, combining distributed renewable energy with the advanced ICTs, which is known as the version of smart grids 2.0 [35].

Specially, ICTs provide a viable way to use the control capability of smart grid and allow distributed energy to access to the backbone grid in EI [36]. Here, smart grid is used to collect and operate the information about the behaviors of users and suppliers to improve the sustainability and reliability of energy.

B. Software-Defined Energy Internet

With traditional TCP/IP protocol, energy Internet has achieved great success. However, many challenges have emerged with the increasing number of smart connected devices in smart grid. It is hard for such rigid and static Internet to meet the demands of flexibility, agility, and ubiquitous accessibility.

In order to solve this embarrassment, there is a consensus to establish the future energy Internet architecture. SDN is seen as one of the most promising paradigms [37]. It is an approach to implement the network that separates the control plane and the data plane, abstracts the underlying infrastructures, and simplifies the network management by introducing the ability of programming.

Some works have employed SDN in energy Internet and smart grid. For example, in order to support secure communications, the authors in [38] learned a SDN-enabled multi-attribute secure architecture for smart grid in IIoT environment. Moreover, the authors in [39] proposed a software defined advanced metering infrastructure (AMI) communication architecture. By this architecture, the problem of global load-balanced routing was solved.

Based on these works, we consider a software-defined energy Internet. However, before the wide adoption of SDEI, there are some problems remaining to be solved. The most intractable one is the scalability and reliability of control plane in SDEI. It can be anticipated that a logically centralized, but physically distributed control plane is necessary. Thus, we propose a controller mind framework in distributed SDEI to implement automatic management among multiple controllers.

ACCEPTED MANUSCRIPT

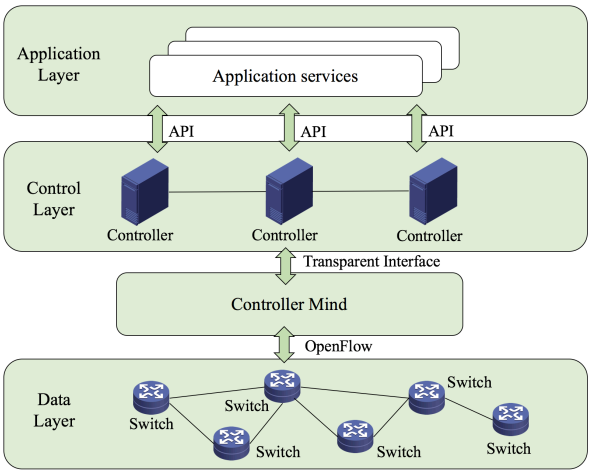

Fig. 1: A framework of controller mind in SDEI.

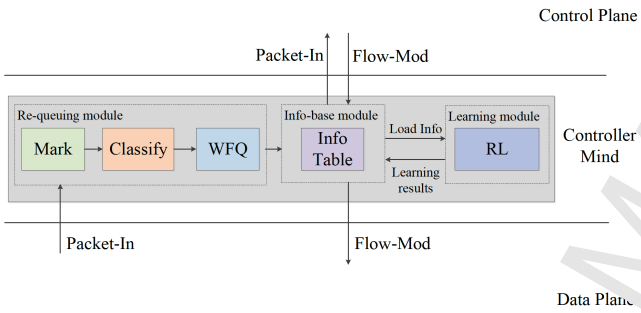

Fig. 2: The detailed CM framework of QoS-enabled load scheduling.

C. Controller Mind framework

Based on traditional SDN architecture, we propose CM framework bridging the control plane and the data plane transparently, as shown in Fig. 1.

The detailed structure of CM framework is given in Fig. 2. CM takes the responsibility of re-queuing incoming *Packet-in* flows to guarantee QoS and forwarding them to the appropriate controllers based on the results of reinforcement learning. Therefore, the CM framework consists of three modules, including *re-queuing module*, *info-base module* and *learning module*.

1) Re-Queuing Module: We assume that there are two types of traffic flows, namely QoS flows with high-priority and best-effort flows with low-priority. QoS flows include some traffic about the application of real-time monitoring in SDEI, and best-effort flows include some traffic from other applications with the low-level requirement of real-time capability. When these traffic flows are encapsulated into *Packet-in* messages and sent to CM, based on source/destination MAC address, IP address, and TCP/UDP port in the packet headers, the re-queuing module marks and classifies the incoming *Packet-in* messages as QoS flows and best-effort flows [40], then re-queues them by the method shown in Section IV-A.

2) Info-Table Module: It receives re-queuing *Packet-in* messages from *re-queuing module* sends them to controllers based on the learning results from the *learning module*, receives *Flow-mod* message from the control plane, and sends *Flow-mod* messages to the data plane. The corresponding changes of the loads in each controller are aware by sending *Packet-in* messages, and receiving *Flow-mod* messages, which are all recorded by this module. On the one hand, these load records are the training datasets of *learning module*, on the other hand, by this mechanism, the frequent signaling interactions that are used to obtain the current load information are avoided, compared with the traditional schemes shown in Section II.

3) Learning Module: Based on the historical load records from the *info-table module* and reinforcement learning algorithm, *learning module* trains the data offline, obtains the learning results, and sends to *info-table module*. The reinforcement learning algorithm, i.e., Q-learning, is executed in this module, and the detail of the algorithm will be shown in Section V.

IV. System Model

In this section, we present re-queuing model, followed by workload model.

A. Re-Queuing Model

If we use *FIFO* (First In First Out) model, some delay-sensitive flows can't be treated fairly. Therefore, when arriving at CM, *Packet-in* messages would be classified into QoS messages or best-effort messages by extracting their headers, such as source/destination MAC address, IP address and TCP/UDP port. The architecture of classification and re-queuing at CM is shown in Fig. 3.

Here, we use a weight fair queuing(WFQ) [41] algorithm in re-queuing model. In WFQ, we give weight coefficient w_i to classify queue i. Each classified queue sends the messages based on weight coefficient w_i. When the queue is empty, skip and access the next queue. Hence, the average service time of queue i is $\frac{w_i}{\sum w_j}$, where $\sum w_j$ is the sum of weight coefficients of all non-empty queues.

B. Workload Model

With the rapid development of commercial deployment, the performance of SDN controller is more and more important. Global SDN certified testing center (SDNCTC) [42] develops a SDN controller performance test tool, called *OFsuite_Performance*, and releases a RYU controller performance test report [43]. In this report, when the arriving rate of *Packet-in* messages is lower, the flow response time is linear with the *Packet-in* messages arriving rate, and the

ACCEPTED MANUSCRIPT

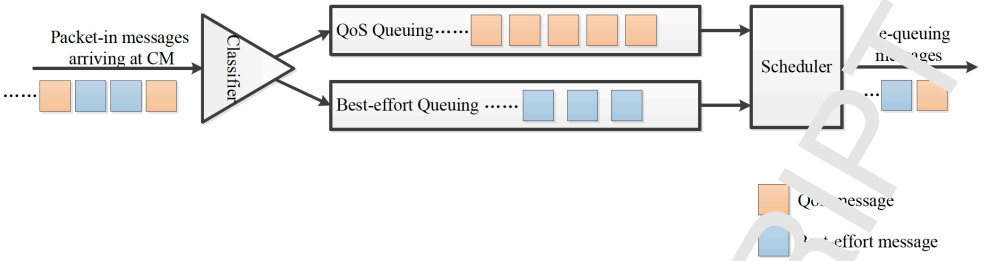

Fig. 3: Classification and re-queuing at CM

acceptable maximum arriving rate of *Packet-in* messages is related to the controller's own performance parameters. Hence, from the test result in the report, we obtain the relationship between the response time and the number of *Packet-in* messages in (1),

$$\tau = \rho N_p + \beta, N_p \leq max, \tag{1}$$

where τ represents the response time, and N_p is the number of *Packet-in* messages, ρ and β are the parameters related to the performance of each controller. max is the maximum number of $Packet-in$ messages in the controller. This equation is an empirical expression that is shown in RYU controller performance test report. Although it has not ever been used in previously published work, we can have this relationship according to the real test works.

Meanwhile, Zhang *et al.* in [44] provided the relationship between the response time and the servers' load status as shown in (2),

$$\tau = \theta^{l_s}, \tag{2}$$

where l_s is the load status of servers. θ is a parameter related to server's performance. Controllers are always deployed in servers, so in this paper we use the same load status model in controllers as servers.

From (1) and (2), we can deduce (3), which explains the relationship between the load status and the number of *Packet-in* messages in the controller.

$$l_s = log_\theta(\rho N_p + \beta), N_p \leq max. \tag{3}$$

When $N_p = max$, the load status is 100%. Thus using (3), we have $log_\theta(\rho max + \beta) = 1$, and there is a necessary relationship between the parameters, which is $\rho max + \beta = \theta$. Obviously, when $N_p > max$, the load status is also 100%. Thus we have (4),

$$l_s = \begin{cases} log_\theta(\rho N_p + \beta) & N_p \leq max \\ 100\% & N_p > max \end{cases}, \tag{4}$$

where $\rho max + \beta = \theta$

In the following problem formulation and simulation, we use (4) as the workload model. Since the number of *Packet-in* messages N_p is recorded by the info-table module, we use (4) to transfer the number of *Packet-in* messages to the load status of controllers to formulate and simulate the problem in the following sections.

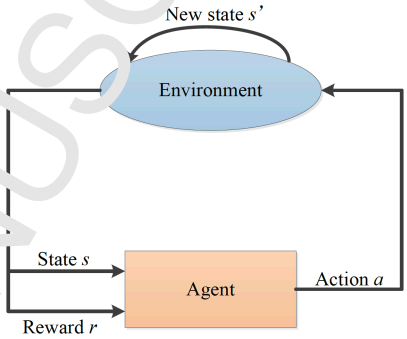

Fig. 4: The agent-environment interaction model of reinforcement learning.

V. Problem Formulation

In this section, we first describe a brief review of reinforcement learning. Then we formulate the load scheduling problem as a Q-learning process, which starts with the optimization problem formulation.

A. Reinforcement Learning

Fig. 4 shows the agent-environment interaction model of reinforcement learning. At the decision epoch, the agent obtains environment state s and corresponding reward r. According to the given policy, the agent selects action a to work on the current environment, which makes state s turn into new state s'. Then at the next decision epoch, the agent and environment interact in the same way.

The agent selects the action based on the policy function which defines the behavior at a given moment. A stationary random policy is defined as π: $S \times A \rightarrow [0, 1]$, where $\pi[s, a]$ represents the probability of selecting action a under state s, S is state space, and A is action space.

There are two value functions to represent the feedbacks from each decision, namely state value function $V^\pi(s)$ and action-state value function $Q^\pi(s, a)$. $V^\pi(s)$ means the

ACCEPTED MANUSCRIPT

expected total rewards based on policy π in state s, and it can be represented as:

$$V^{\pi}(s) = E^{\pi}[\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} | s_t = s], \quad (5)$$

where $E^{\pi}[*]$ denotes the mathematical expectation under state transferring probability $P(s, a, s')$ and policy π . r_t is immediate reward at time t. $\gamma \in (0, 1]$ denotes the discount factor to trade-off the importance of immediate reward and long-term reward.

Additionally, action-state value function $Q^{\pi}(s, a)$ represents the expected total rewards based on policy π in state-action pair (s,a), and it can be represented as:

$$Q^{\pi}(s, a) = E^{\pi}[\sum_{t=0}^{\infty} \gamma^k r_{t+k+1} | s_t = s, a_t = a]. \quad (6)$$

And there is a relationship between $V^{\pi}(s)$ and $Q^{\pi}(s, a)$. For a certain policy π, $V^{\pi}(s) = Q^{\pi}(s, \pi(s))$. For a stochastic policy, $V^{\pi}(s) = \sum_{a \in A} \pi(s, a) Q^{\pi}(s, a)$. Hence, $Q^{\pi}(s, a)$ can be expressed by $V^{\pi}(s)$ as follows:

$$Q^{\pi}(s, a) = R(s, a) + \gamma \sum_{s' \in S} P(s, a, s') V^{\pi}(s'), \quad (7)$$

where $R(s, a) = \sum_{s' \in S} P(s, a, s') R(s, a, s')$ means the expected reward of selecting action a under state s. $R(s, a, s') = E[r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + ... | s_t = s, a = a_t = \pi(s_t), s_{t+1} = s']$ denotes the expected reward of selecting action a to transfer the state from s to s'.

B. Reinforcement Learning Formulation

In order to obtain the optimal policy, it is necessary to define state space, action space and reward function in Q-learning model. Before this, we will give the optimization problem formulation in the RL based QoS-enabled load scheduling problem.

1) Optimization Problem Formulation: Our target is to find out the optimal scheme to allocate *Packet-in* messages from the data plane to the control plane with the minimum waiting time of QoS flows and the acceptable packet loss rate of best-effort flows. The minimization problem can be formulated as the weighted sum as follows.

$$min \quad k_1 \sum_{t=0}^{T} \sum_{i=1}^{N_1} TQ_i^1(t) + k_2 \sum_{t=0}^{T} \sum_{k=1}^{N_2} PLQ_k^2(t), \quad (8)$$

$$subject\ to$$

$$PLQ_i^1(t) = 0, \qquad \forall i = 1, 2, ...N_1, \quad (9)$$

where we assume there are T time slots during the whole system, which starts when the first *Packet-in* message comes and terminates when the last *Packet-in* message departs. Let $t \in \{0, 1, 2, ..., T-1\}$ denote the time instant. $TQ_i^1(t)$ is the waiting time of QoS flows i at time instant t. $PLQ_i^1(t)$ and $PLQ_k^2(t)$ are the packet loss rate of QoS flows i and best-effort flows k at time instant t, respectively. N_1 and N_2 are the total number of QoS flows and best-effort flows, respectively. k_1 and k_2 are the scale factors, and $k_1 + k_2 = 1$.

In the above optimization problem, the constraint (9) guarantees that the QoS flows have no packet loss.

Notably, one of the optimal target in (8) is to minimize the packet loss rate of best-effort flows, which is equivalently substituted by the load variation among all controllers in the remainder of this paper. Because best-effort messages have the low priority, i.e., when the messages loss happens, it is probable that best-effort messages are discarded. Thus, the lower load variation leads to the lower packet loss rate of best-effort messages directly.

2) State Space: In order to reduce the load variation among all controllers and the waiting time of QoS flows, we propose the definition of state space as follows:

$$S = \{s = [Q_{incom}, l_c, q_c], | Q_{incom} \in Q_{level}, l_c \in L_c, q_c \in Q_c\}, \quad (10)$$

where

$$Q_{level} = \{1, 2\}, \quad (11)$$

$$L_c = \{[l_{c_1}, l_{c_2}, \cdots, l_{c_k}, \cdots, l_{c_N}]\}, \quad (12)$$

$$Q_c = \{[Q_{c_1 1}, Q_{c_2 1}, \cdots, Q_{c_k 1}, \cdots, Q_{c_N 1}]\}, \quad (13)$$

where N is the total number of controllers in the system, and c_k means the kth controller. Q_{level} means the different QoS levels of flows in this system. When $Q_{level} = 1$, this flow is QoS flow with the high-priority. When $Q_{level} = 2$, this flow is best-effort flow with the low-priority. L_c means the set of the load status of all controllers, and l_{c_k} denotes the load status of controller c_k which is calculated by (4) and the number of $Packet-in$ messages is recorded by the info-table module. Q_c means the set of the number of QoS flows in all controllers, and $Q_{c_k 1}$ denotes the number of QoS flows in controller c_k, which is recorded by the info-table module.

3) Action Space: In the system, the agent has to decide how to allocate $Packet - in$ message among multiple controllers. Thus, the action space A of RL can be defined as follows,

$$A = \{a_{c_1}, a_{c_2}, \cdots, a_{c_k}, \cdots, a_{c_N}\}, \quad (14)$$

where a_{c_k} represents the allocation control between the current $Packet - in$ message and controller c_k. if $a_{c_k} = 1$, the current $Packet - in$ message is assigned to controller c_k. if $a_{c_k} = 0$, the current $Packet - in$ message is not assigned to the controller c_k. Note that $\sum_{k=1}^{N} a_{c_k} = 1$, which guarantees that the current $Packet - in$ message only has one assigned controller.

ACCEPTED MANUSCRIPT

4) Reward Function: We define numerical reward r that the agent obtains from taking action a at state s. We have two targets as shown in (8), including minimizing the load variation and the waiting time of QoS flows. Accordingly, there are two parts in reward function r, including the standard deviation of all controllers and the number of messages whose QoS levels exceed the incoming message, respectively.

The lower standard deviation means the better load balancing. Since bigger reward is taken in Q-learning, we use the negative standard deviation to represent the load variation in reward function, which is denoted in the first part of reward function r.

Since all controllers in the system are QoS-enabled, which means *Packet-in* messages will re-queue after arriving at all controllers to make sure QoS flows to be processed with the high priority. Thus, the waiting time of incoming QoS flows is only related to the number of QoS flows before them. The fewer QoS flows lead to the less waiting time, which is denoted in the second part of reward function r.

In summary, reward function r can be expressed as follows:

$$\begin{aligned} r = & - k_2 std(l_{c_1}, l_{c_2}, \cdots, l_{c_N}) \\ & - k_1 \sum_{i=1}^{N_{c_k}} bool(Q_{incom} \geq Q_{queue}(i)), \end{aligned} \tag{15}$$

where N_{c_k} is the number of $Packet-in$ messages in controller c_k. Q_{incom} represents the QoS level of incoming message, and $Q_{queue}(i)$ means the QoS level of the ith messages in controller c_k. *bool* denotes the Boolean operator. For example, when the QoS level of the incoming message is 1 and the current messages' QoS line on controller is 11222, the result of $\sum_{i=1}^{N_{c_k}} bool(Q_{incom} \geq Q_{queue}(i))$ is 2. In the other controller, its current messages' QoS line is 11122, the result of boolean operation is 3. If only consider the waiting time of incoming message, the agent is more possible to allocate this message to the first controller because of the less waiting time. k_1 and k_2 are the scale factors which are the same as (3).

5) Method to Learn the Optimal Strategy: Q-learning is a typical algorithm of reinforcement learning, and we use Q-learning to learn the optimal strategy in this paper, where action-state value function $Q(s,a)$ is selected as the estimation function, rather than state value function $V(s)$. The basic idea of Q-learning is to evaluate $Q(s,a)$ by a temporal difference method, which is denoted as follows:

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha(r_t + \gamma \max_a Q(s_{t+1}, a) - Q(s_t, a_t)), \tag{16}$$

where α denotes the learning efficiency. In Q-learning, each $Q(s,a)$ is put into $Q-$table.

At first, Q-learning initializes the $Q-$table. Then at state s_t, the agent determines action a_t according to ε-greedy policy, and obtains the experience knowledge as well as the training samples $(s_t, a_t, s_{t+1}, a_{t+1})$. Meanwhile, the agent uses (16) to update $Q(s_t, a_t)$ and $Q-$table. When meeting with the goal state, the agent terminates one loop iteration. Then Q-learning continues a new loop iteration from the initial state until the end of learning. The algorithm performed on each step is shown in Algorithm 1.

Algorithm 1 Q-learning

1: Initialize $Q-$table, and set parameters α, γ and k_0;
2: **for** $k = 1 : k_0$ **do**
3: Select a initial state s_t randomly
4: **while** $s_t != s_{goal}$ **do**
5: Select action a_t based on ε-greedy policy, and obtain immediate reward r_t and next state s_{t+1}
6: $Q(s_t, a_t) \leftarrow Q(s_t, a_t) + (r_t + \gamma \max_a Q(s_{t+1}, a) - Q(s_t, a_t))$
7: $s_t \leftarrow s_{t+1}$
8: **end while**
9: **end for**

VI. Simulation Results and Discussions

In this section, we use computer simulation to evaluate the performance of RL based QoS-enabled load scheduling. Firstly, we describe simulation settings, then present the simulation results.

A. Simulation Settings

1) Network Topology: We choose the same topology as the one in [13], which has three controllers in the control plane, and several switches in the data plane. Thus, $N = 3$.

2) Parameter Settings: The different seeds are employed in the simulation, and performances are average to estimate the performance of our proposed scheme. We utilize the queuing theory to model the arrival, processing and departure of *Packet-in* messages. Here, the arrival of the *Packet-in* messages is based on a Poisson distribution with the parameter of λ, indicated by the arriving rate of *Packet-in* messages. The processing time of each controller is based on the negative exponential distribution with the parameter of μ, indicated by the performance of controllers. And we assume that all controllers have the same performance, i.e., the same μ. Summarily, the values of all parameters in the simulation are summarized in TABLE.I.

For performance comparison, four schemes are simulated:

- RL based QoS-enabled load scheduling scheme proposed in this paper and we call it as RL in the remainder of this section.
- Dynamic weight based QoS-enabled load scheduling scheme in the work of [13] and we call it as DW in the remainder of this section.
- QoS-enabled scheme, and this scheme does not take consideration of the load scheduling. We call it as QS in the remainder of this section.

ACCEPTED MANUSCRIPT

- Mini-connected load scheduling scheme, and this scheme does not consider the QoS. We call it as MN in the remainder of this section.

B. Performance Evaluation Results

Fig. 5 shows the relationship between the load variation and the *Packet-in* messages arriving rates of different schemes when the proportion of QoS messages is 75%. With the increase of the arriving rate, the load variation is increasing. The reason is that, as the arriving rates increasing, more messages accumulate in controllers, which obviously results in the larger load variation. In any case of arriving rate, QS's load variation is much bigger than others', because QS scheme only considers the priority of messages and fails to take the load balancing into consideration, some controllers are overloaded and others are idle, which leads to the biggest load variation. Taking the load balancing into consideration, the other three schemes' load variations are much smaller. Relatively speaking, DW's load variation is bigger. The reason is that, the adjustment of the load does not happen at each step in DW. Only when overloaded, the dynamic weight load balancing is triggered. But in MN, when each message arrives, it is assigned to the controller with the least load status, which is equivalent to adjust the load distribution scheme in each step. So MN is better than DM. But in any case of the arriving rates, RL's load variation is very close to the MN's curve. Even the RL's load variation is smaller than the MN's in some cases. Because by the offline learning of the historical data, RL performs the optimal load distribution globally, which results in the best load scheduling effect.

Fig. 6 displays the relationship between the waiting time of QoS messages and *Packet-in* messages arriving rates of different schemes when the proportion of QoS messages is 75%. For QS scheme, although it considers the priority of messages to let the messages with the high priority be processed firstly, no load balancing mechanism also results

TABLE I: Parameters setting in the simulation

Parameter	*Value*	*Description*
max	20	The maximum number of $Packet-in$ messages controllers
ρ	0.5	The performance parameter of the controller
β	5	The performance parameter of the controller
θ	15	The performance parameter of the controller
N	3	The number of total controllers in the control plane
k_1	1.6	One scale factor
k_2	0.4	Another scale factor
ε	0.9	The greed factor
γ	0.65	The discount factor
Initialized α	1	The learning efficiency
μ	16	Service rate of each controller

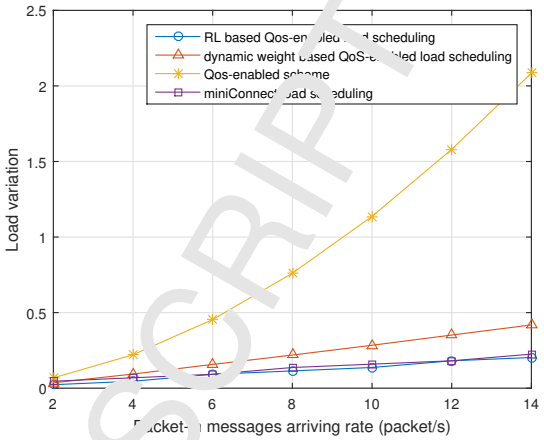

Fig. 5: Load variation versus *Packet-in* messages arriving rates of RL, DM, QS and MN.

in more waiting time of QoS messages. In the low arriving rates case, DW's waiting time is less than MN's. The reason is that, when the arriving rate is lower, it is unlikely to be overloaded in controllers, but MN scheme needs to get the load status of controllers by exchanging the signaling to adjust the load distribution in each step, which results in the additional time delay. DW scheme isn't triggered in the low load status, so it does not lead to the time delay of the signaling exchange and has relatively smaller time delay, compared with the MN scheme. With the increase of arriving rates and messages accumulating in controllers, MN and DW schemes also exchange the signaling frequently, but MN has the better load balancing performance, as shown in Fig. 5, it also has the better time efficiency, compared with DW. And for RL scheme, because the allocated scheme has been learned offline and in advance, and it is no need for RL scheme to exchange the signaling at all. So in the lower arriving rates, RL scheme has no additional time delay. In the higher arriving rates, RL scheme has a little time delay because of the increasing of messages. Overall, RL scheme has the best time efficiency.

Fig. 7 presents the load variation when the proportion of QoS messages changes at the arriving rate of 8 packet/s. Because the arrival rate is constant, the load variation has no relationship with the proportion of QoS messages. But we can draw the similar conclusion as Fig. 5, which is that RL scheme has the best load variation.

Fig. 8 shows the relationship between the waiting time of QoS messages and the proportion of QoS messages at the arriving rate of 8 packet/s. For QS, with the growth of QoS messages, the waiting time increases linearly, because it only considers the priority of messages but no load balancing. In the lower proportion case, DW's waiting time is less than MN's. The reason is that, only when overload happens, DW scheme is triggered. And MN scheme happens in each

ACCEPTED MANUSCRIPT

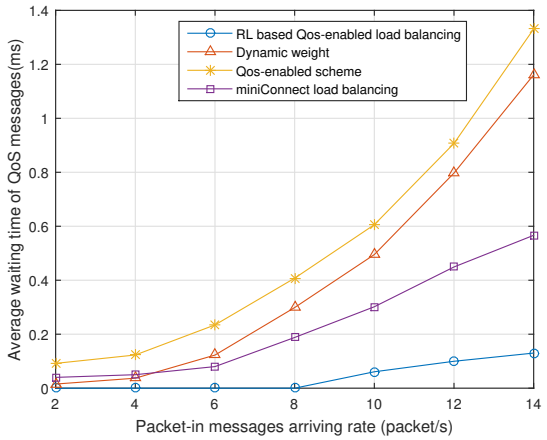

Fig. 6: Average waiting time of QoS messages versus *Packet-in* messages arriving rates of RL, DM, QS and MN.

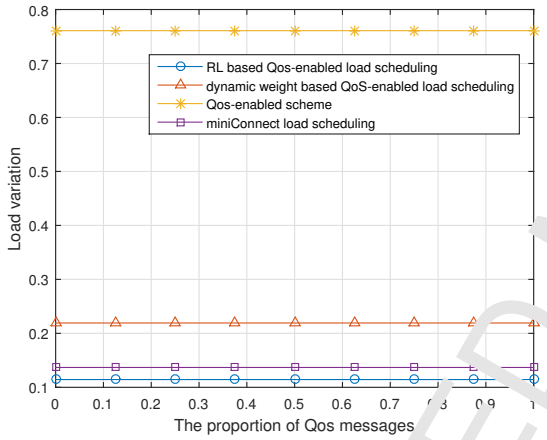

Fig. 7: Load variation versus the proportion of QoS messages in RL, DM, QS and MN.

decision epoch. Under the current arriving rate, it is unlikely to be overloaded. So DW has relatively smaller time delay, compared with MN in the lower proportion. The increase of proportion leads to the growth of time delay directly. MN has the better load balancing as shown in Fig. 7, which results in the better time efficiency in the higher proportion. RL enables to learn the allocated scheme in advance and offline with no signaling exchanging. So when QoS messages are smaller, RL has no time delay completely. And with the growth of QoS messages, RL has a litter time delay and the best time efficiency.

VII. Conclusion and Future Works

In this paper, we proposed a controller mind (CM) framework to manage multiple controllers automatically and intelligently in SDEI, so as to keep the high accuracy in the

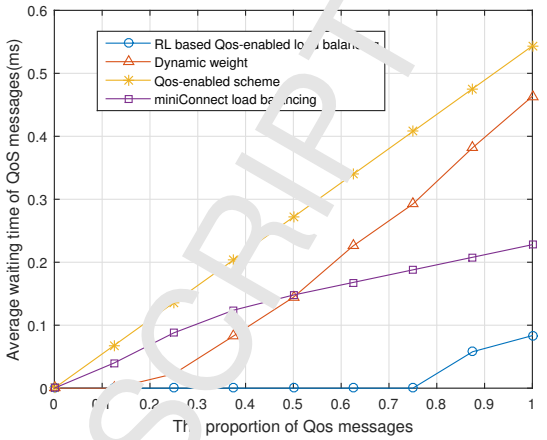

Fig. 8: Average waiting time of QoS messages versus the proportion of QoS messages in RL, DM, QS and MN.

real-time monitoring of smart grid. Specifically, we solved the QoS-enabled load scheduling by reinforcement learning, defined the learning agent, action space, state space, and reward function, as well leveraged the historical data to learn the load scheduling scheme offline and ahead of time, so as to realize the automatic management among multiple controllers. We simulated the performance of CM framework compared with three traditional schemes. Simulation results showed that the reinforcement learning based scheme had the best load balancing and time efficiency, which solved the problems of traditional load balancing schemes. However, the QoS-enabled load scheduling scheme learns from the historical data, so it has the lower robustness to the burst traffic. Once the burst traffic happens, state space in our scheme fails to describe all situations and also needs the longer time to learn the new allocation scheme. During this period, the load variation and time efficiency are severely affected. Future work is in progress to address these challenges.

Acknowledgment

The project is supported by the Key Program of the National Natural Science Foundation of China (Grant No 61431008), and BUPT Excellent Ph.D. Students Foundation (Grant No 2015010100).

References

[1] W. Zhong, K. Xie, Y. Liu, C. Yang, and S. Xie, "Topology-aware vehicle-to-grid energy trading for active distribution systems," *IEEE Trans. on Smart Grid*, 2018.

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

[3] W. Zhong, K. Xie, Y. Liu, C. Yang, and S. Xie, "Auction mechanisms for energy trading in multi-energy systems," *IEEE Trans. on Industrial Infor.*, vol. 14, no. 4, pp. 1511–1521, 2018.

ACCEPTED MANUSCRIPT

[4] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner, "Openflow: enabling innovation in campus networks," *ACM SIGCOMM Comp. Comm. Review*, vol. 38, no. 2, pp. 69–74, 2008.

[5] 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.

[6] M. Karakus and A. Durresi, "A survey: Control plane scalability issues and approaches in software-defined networking (SDN)," *Comp. Net.*, vol. 112, pp. 279–293, 2017.

[7] C. Qiu, C. Zhao, F. Xu, and T. Yang, "Sleeping mode of multi-controller in green software-defined networking," *EURASIP Journal on Wireless Comm. and Net.*, vol. 2016, no. 1, pp. 282–296, Jan. 2016.

[8] E. Ng, Z. Cai, and A. Cox, "Maestro: A system for scalable openflow control," *Rice University, Houston, TX, USA, TSEN Maestro-Techn. Rep, TR10-08*, 2010.

[9] S. Schmid and J. Suomela, "Exploiting locality in distributed sdn control," in *Proc. Conf. ACM SIGCOMM Workshop on Hot Topics in Software Defined Net., Hong Kong, China*, Aug. 2013, pp. 121–126.

[10] M. Canini, P. Kuznetsov, D. Levin, and S. Schmid, "A distributed and robust sdn control plane for transactional network updates," in *Proc. Conf. IEEE INFOCOM'15, Hong Kong, China*, Apr. 2015, pp. 190–198.

[11] S. Civanlar, M. Parlakisik, A. M. Tekalp, B. Gorkemli, B. Kaytaz, and E. Onem, "A qos-enabled openflow environment for scalable video streaming," in *Pro. Conf. GLOBECOM'10 Workshops, Miami, USA*, Dec. 2010, pp. 351–356.

[12] S. Xie, W. Zhong, K. Xie, R. Yu, and Y. Zhang, "Fair energy scheduling for vehicle-to-grid networks using adaptive dynamic programming," *IEEE Trans. on Neural Net. and Learning Sys.*, vol. 27, no. 8, pp. 1697–1707, 2016.

[13] N. T. Hai and D.-S. Kim, "Efficient load balancing for multi-controller in SDN-based mission-critical networks," in *Pro. Conf. Industrial Informatics'16, Poitiers, France*, Jul, 2016, pp. 420–425.

[14] A. Tootoonchian and Y. Ganjali, "HyperFlow: A distributed control plane for OpenFlow," in *Pro. Conf. Internet Net. Management, Berkeley, USA*, 2010, pp. 3–3.

[15] T. Koponen, M. Casado, N. Gude, J. Stribling, L. Poutievski, M. Zhu, R. Ramanathan, Y. Iwata, H. Inoue, T. Hama, and others, "Onix: A Distributed Control Platform for Large-scale Production Networks." in *Pro. Conf. Operating Sys. Design and Implementation, Vancouver, Canada*, vol. 10, Oct. 2010, pp. 1–6.

[16] P. Berde, M. Gerola, J. Hart, Y. Higuchi, M. Kobayashi, T. Koide, B. Lantz, B. O'Connor, P. Radoslavov, W. Snow *et al.*, "ONOS: towards an open, distributed SDN OS," in *Pro. Conf. Hot topics in Software Defined Net., Chicago, USA*, Aug. 2014, pp. 1–6.

[17] K. Phemius, M. Bouet, and J. Leguay, "Disco: Distributed multi-domain sdn controllers," in *Pro. Conf. Net. Operations and Management, Krakow, Poland*, May. 2014, pp. 1–4.

[18] S. Hassas Yeganeh and Y. Ganjali, "Kandoo: a framework for efficient and scalable offloading of control applications," in *Pro. Conf. Hot Topics in Software Defined Net., Helsinki, Finland*, Aug. 2012, pp. 19–24.

[19] Y. Hu, W. Wang, X. Gong, X. Que, and S. Cheng, "Balanceflow: controller load balancing for openflow networks," in *Pro. Conf. Cloud Comp. and Intelligent Sys. '12, Hangzhou, China*, vol. 2, Oct. 2012, pp. 780–785.

[20] A. Dixit, F. Hao, S. Mukherjee, T. V. Lakshman, and R. Kompella, "Towards an elastic distributed SDN controller," in *ACM SIGCOMM Computer Communication Review*, vol. 43, Oct. 2013, pp. 7–12.

[21] R. Sherwood, G. Gibb, K.-K. Yap, G. Appenzeller, M. Casado, N. McKeown, and G. Parulkar, "Flowvisor: A network virtualization layer," *OpenFlow Switch Consortium, Tech. Rep*, vol. 1, pp. 132–146, 2009.

[22] A. Krishnamurthy, S. P. Chandrabose, and A. Gember-Jacobson, "Pratyaastha: an efficient elastic distributed sdn control plane," in *Pro. Conf. Hot Topics in Software Defined Net., New York, USA*, Aug. 2014, pp. 133–138.

[23] C. Liang, R. Kawashima, and H. Matsuo, "Scalable and crash-tolerant load balancing based on switch migration for multiple open flow controllers," in *Pro. Conf. Comp. and Net. '14, Shizuoka, Japan*, Dec. 2014, pp. 171–177.

[24] Y. Zhou, M. Zhu, L. Xiao, L. Ruan, W. Duan, D. Li, R. Liu, and M. Zhu, "A load balancing strategy of SDN controller based on distributed decision," in *Pro. Conf. Trust, Security and Privacy in Comp. and Comm '14, Beijing, China*, Sep. 2014, pp. 851–856.

[25] J. Yu, Y. Wang, K. Pei, S. Zhang, and J. Li, "A load balancing mechanism for multiple SDN controllers based on load informing strategy," in *Pro. Conf. Net. Operations and Management '16, Kanazawa, Japan*, Oct. 2016, pp. 1–4.

[26] H. Yao, C. Qiu, C. Zhao, and L. Shi, "A multicontroller load balancing approach in software-defined wireless networks," *International Journal of Distributed Sensor Networks*, vol. 11, no. 10, pp. 41–49, 2015.

[27] H. Sufiev and Y. Haddad, "A dynamic load balancing architecture for SDN," in *Pro. Conf. Science of Electrical Engineering, Eilat, Israel*, Nov. 2016, pp. 1–3.

[28] D. Apiletti, E. Baralis, T. Cerquitelli, P. Garza, D. Giordano, M. Mellia, and L. Venturini, "Selina: a self-learning insightful network analyzer," *IEEE Trans. on Net. and Service Management*, vol. 13, no. 3, pp. 696–710, Aug. 2016.

[29] E. Başkaya, M. Bennis, E. Zeydan, M. A. Kader, I. A. Karatepe, A. S. Er, and M. Debbah, "Big data meets telcos: A proactive caching perspective," *Journal of Comm. and Net.*, vol. 17, no. 6, pp. 549–557, 2015.

[30] B. Fan, S. Leng, and K. Yang, "A dynamic bandwidth allocation algorithm in mobile networks with big data of users and networks," *IEEE Net.*, vol. 30, no. 1, pp. 6–10, 2016.

[31] J. Mestres, A. Rodriguez-Natal, J. Carner, P. Barlet-Ros, E. Alarcón, M. Solé, V. Muntés-Mulero, D. Meyer, S. Barkai, M. J. Hibbett *et al.*, "Knowledge-defined networking," *ACM SIGCOMM Computer Communication Review*, vol. 47, no. 3, pp. 2–10, Jul. 2017.

[32] W.-X. Liu, J. Zhang, Z.-W. Liang, L.-X. Peng, and J. Cai, "Content popularity prediction and caching for ICN: A deep learning approach with SDN," *IEEE Access*, vol. 6, pp. 5075–5089, 2018.

[33] Y. Cui, Q. Huang, and V. K. Lau, "Queue-aware dynamic clustering and power allocation for network MIMO systems via distributed stochastic learning," *IEEE Trans. on Signal Processing*, vol. 59, no. 3, pp. 1229–1238, 2011.

[34] J. Qi and D. Wu, "Green energy management of the energy internet based on service composition quality," *IEEE Access*, 2018.

[35] Y. Zhang, R. Yu, M. Nekovee, Y. Liu, S. Xie, and S. Gjessing, "Cognitive machine-to-machine communications: visions and potentials for the smart grid," *IEEE Net.*, vol. 26, no. 3, 2012.

[36] S. Maharjan, Q. Zhu, Y. Zhang, S. Gjessing, and T. Basar, "Dependable demand response management in the smart grid: A stackelberg game approach," *IEEE Trans. on Smart Grid*, vol. 4, no. 1, pp. 120–132, 2013.

[37] W. Xia, Y. Wen, C. H. Foh, D. Niyato, and H. Xie, "A survey on software-defined networking," *IEEE Comm. Surveys & Tutorials*, vol. 17, no. 1, pp. 27–51, 2015.

[38] R. Chaudhary, G. S. Aujla, S. Garg, N. Kumar, and J. J. Rodrigues, "SDN-enabled multi-attribute-based secure communication for smart grid in IIoT environment," *IEEE Trans. on Industrial Infor.*, vol. 14, no. 6, pp. 2629–2640, 2018.

[39] A. Montazerolghaem, M. H. Yaghmaee, and A. Leon-Garcia, "OpenAMI: Software-defined AMI load balancing," *IEEE Internet of Things Journal*, vol. 5, no. 1, pp. 206–218, 2018.

[40] Y. Zhang, R. Yu, S. Xie, W. Yao, Y. Xiao, and M. Guizani, "Home M2M networks: architectures, standards, and QoS improvement," *IEEE Comm Mag.*, vol. 49, no. 4, 2011.

[41] C. Li, S. Tsao, M. C. Chen, Y. Sun, and Y. Huang, "Proportional delay differentiation service based on weighted fair queuing," in *Pro. Conf. Comp. Comm. and Net., Las Vegas, USA*, Oct. 2000, pp. 418–423.

[42] SDNCTC, http://www.sdnctc.com/.

[43] http://www.sdnctc.com/download/resource_download/id/6.

[44] Q. Zhang, A. Riska, W. Sun, E. Smirni, and G. Ciardo, "Workload-aware load balancing for clustered web servers," *IEEE Trans. on Parallel and Distributed Sys.*, vol. 16, no. 3, pp. 219–233, 2005.

ACCEPTED MANUSCRIPT

ACCEPTED MANUSCRIPT

Chao Qiu received the B.S. degree from China Agricultural University, Beijing, China in 2013 in communication engineering. She is currently pursuing the Ph.D. degree with the School of Information and Communication Engineering, Beijing University of Posts and Telecommunications. Her current research interests include big data, software defined networking.

Shaohua Cui received the Bachelor's degree in Department of material science and engineering from University of Science & Technology Beijing (USTB) in 2010, and Ph.D degree in Materials Science and Engineering from Beijing University of Science and Technology. At present, she serves at China Petroleum Technology & Development Corporation. Her research is focused on petroleum equipment and new energy materials.

Haipeng Yao is a lecturer in Beijing University of Posts and Telecommunications. Haipeng Yao received his Ph.D. in the Department of Telecommunication Engineering at University of Beijing University of Posts and Telecommunications in 2008, and received his M.S. degree in Wireless Communications from Beijing University of Posts and Telecommunications in 2006. His main research interests are in the area of Big Data, future internet architecture, cognitive radio networks, and optimization of protocols and architectures for broadband wireless networks.

Fangmin Xu received the M.S. and Ph.D. degrees in Communication Engineering from Beijing University of Posts and Telecommunication(BUPT), China in 2003 and 2008, respectively. He is currently associate professor in the School of Information and Communicaiton Engineering, BUPT, China. From 2008 to 2014, he was with Samsung Electronics where he actively contributed to 3GPP LTE LTE-A and IEEE 802.16m. He is the author of 2 books, 20 peer-reviewed international research papers, 50 standard contributions and the inventor of 15 issued/pending patents among which 4 have been adopted in the specifications of 4G (3GPP LTE/LTE-A and IEEE 802.16m) standards. His research interests include advance technologies in wireless networks, especially Internet of things(IoT) field.

F. RICHARD YU (S00-M04-SM08) received the PhD degree in electrical engineering from the University of British Columbia (UBC) in 2003. From 2002 to 2006, he was with Ericsson (in Lund, Sweden) and a start-up in California, USA. He joined Carleton University in 2007, where he is currently an Associate Professor. He received the IEEE Outstanding Leadership Award in 2013, Carleton Research Achievement Award in 2012, the Ontario Early Researcher Award (formerly Premiers Research Excellence Award) in 2011, the Excellent Contribution Award at IEEE/IFIP TrustCom 2010, the Leadership Opportunity Fund Award from Canada Foundation of Innovation in 2009 and the Best Paper Awards at IEEE ICC 2014, Globecom 2012, IEEE/IFIP TrustCom 2009 and Intl Conference on Networking 2005. His research interests include cross-layer/cross-system design, security, green IT and QoS provisioning in wireless-based systems.

He serves on the editorial boards of several journals, including Co-Editorin-Chief for Ad Hoc & Sensor Wireless Networks, Lead Series Editor for IEEE Transactions on Vehicular Technology, IEEE Communications Surveys & Tutorials, EURASIP Journal on Wireless Communications Networking, Wiley Journal on Security and Communication Networks, and International Journal of Wireless Communications and Networking. He has served as the Technical Program Committee (TPC) Co-Chair of numerous conferences. Dr. Yu is a registered Professional Engineer in the province of Ontario, Canada

Chenglin Zhao received the Bachelor's degree in radio-technology from Tianjin University in 1986, and the Master's degree in circuits and systems from Beijing University of Posts and Telecommunications (BUPT) in 1993, and the Ph.D. degree in communication and information system from Beijing University of Posts and Telecommunications, in 1997. At present, he serves as a Professor in Beijing University of Posts and Telecommunications, Beijing, China. His research is focused on emerging technologies of short-range wireless communication, cognitive radios, 60GHz millimeter-wave communications.