Joint Optimization of Latency Monitoring and Traffic Scheduling in Software Defined Heterogeneous Networks

Xu Zhang, Weigang Hou(✉), Lei Guo(✉), Siqi Wang, Qihan Zhang, Pengxing Guo, and Ruijia Li

College of Computer Science and Engineering, Northeastern University, Shenyang 110819, China
{houweigang,guolei}@cse.neu.edu.cn

Abstract. Since the current Internet is only able to provide best-effort services, the quality of service (QoS) for many emerging businesses cannot be well guaranteed. Meanwhile, due to the privatization of networks management, multi-vendor heterogeneous networks are difficult to provide end-to-end QoS assurance on demand. Therefore, heterogeneous devices from different vendors also bring new challenges to the flexible control of network equipment. Software defined network (SDN) is an emerging paradigm which separates the network's control logic from the underlying routers and switches. In this paper, we design a monitoring loop of link latency by using both LLDP and Echo probing modules. Then, a dynamic routing algorithm is proposed to select optimized transmission path based on the information of link latency. In addition, we develop a routing application assorted with the monitoring mechanism by extending the RYU controller. We implement our solution in a semi-practical SDN testbed. Finally, the overall feasibility and efficiency of the proposed solution are experimentally verified and evaluated.

Keywords: SDN · Latency monitoring · Traffic engineering · QoS
Heterogeneous network

1 Introduction

In the traditional IP network, all packets are switched by using best-effort way. Since best-effort service model reduces the overhead and the cost at the network layer without losing reliability and robustness, this architecture can perfectly provide data transmission in the case of conventional applications (voice, video, etc.). However, with the rapid development of the Internet, there have been a large number of new types of business, e.g., VoIP, HDTV, Multimedia service, and so on. This kind of traffic has stringent delay requirements which cannot be guaranteed in the best-effort Internet. Although the Internet Engineering Task Force (IETF) has proposed several quality of service (QoS) architecture such as Integrated Services (IntServ) [1] and Differentiated Services (DiffServ) [2], these

L. Wang et al. (Eds.): QShine 2017, LNICST 234, pp. 104–113, 2018.
https://doi.org/10.1007/978-3-319-78078-8_11

proposals are not very successful. They have not been deployed on a large scale commercial environment because they all need to make fundamental changes to the top layer of a distributed hop-by-hop routing architecture without a global view. Meanwhile, some solutions based on multi-protocol label switching (MPLS) [3] and border gateway protocol (BGP) [3] are explored to address the problem that is severely lacking on the information of the available network resources (e.g., network delay information) from an end-to-end perspective. However, these schemes do not have good re-configurability and adaptability. More particularly, with the expansion of network scale, multi-vendor devices coexist on the Internet [4]. Therefore, the network architecture of the current Internet is inflexible for the rapid development and deployment of network services supporting multiple application requirements.

Software defined network (SDN) [5] is one of the latest revolutions in the networking field, which allows network administrators to manage network services through the abstraction of underlying network functionality [6–8]. To efficiently apply QoS policies [9–12], it is important to obtain the latency information in a SDN-based packet network. In [13], authors presented a measurement scheme of the link latency, but they did not give a routing solution to schedule traffic. Note that, both the network measurement and the traffic engineering are important. If operators want to provide differentiated services for various users through the SDN solution, it is essential to design the monitoring mechanism and develop the traffic scheduling algorithm.

In this paper, we first focus on designing a monitoring mechanism of link delay by using both link layer discovery protocol (LLDP) monitoring module and Echo monitoring module. Then, we propose a dynamic shortest delay routing algorithm, i.e., SDRA, in order to meet the QoS requirements of different traffic. In addition, we develop a routing application assorted with both the monitoring mechanism and the SDRA traffic scheduling by extending the RYU controller. We implement our solution in a semi-practical SDN testbed. Finally, the overall feasibility and efficiency of the proposed solution are experimentally verified and evaluated. In the four-node four-link network (n4s4) topology, we measure the link latency detected by the delay monitoring module, and this demonstrates the feasibility of the monitoring mechanism. Moreover, the performance of the SDRA scheme under different traffic loads is also quantitatively evaluated based on the simulated NSFNET network in term of the end-to-end delay, compared with the minimum hop routing algorithm (MHRA). It is beneficial for us to verify the efficiency of our SDN-based delay solution.

The contributions of this work can be summarized as follows,

(1) We design a monitoring mechanism to measure the link delay in multi-vendor heterogeneous network. And experimental presentation demonstrates the feasibility of the monitoring mechanism.
(2) We propose a delay-aware SDRA algorithm to provide the end-to-end QoS guarantees based on a centralized network view. It is able to choose the optimized path according to the network status in real time.

(3) We develop a SDN-based shortest delay application by extending RYU controller. The efficiency of overall solutions is evaluated in a semi-practical SDN platform.

2 Problem Statement and Analysis

In this section, we first describe the network architecture and the key notations used for the problem formulation. Then, we introduce the system model. Finally, we formulate mathematically our problem in term of providing an end-to-end shortest delay guarantee.

2.1 Network Architecture

The SDN architecture can be depicted as a composition of three planes, as shown in Fig. 1. Each plane has its own specific functions. The data plane corresponds to the multi-vendor heterogeneous devices, which only are responsible for the data forwarding. Different from traditional networks, the control plane is stripped from the underlying forwarding device. The control plane controls all network devices and abstracts the underlying network resources by using the OpenFlow protocol. This is a key characteristic of SDN network, which makes it possible to allocate network resources of multi-vendor heterogeneous devices based on a centralized controller. The management plane is the set of applications that use algorithms or protocols to implement network control and distribute the operations logic to the underlying devices. For instance, the SDRA application maintains a routing policy to provide the end-to-end QoS service by utilizing the proposed solution.

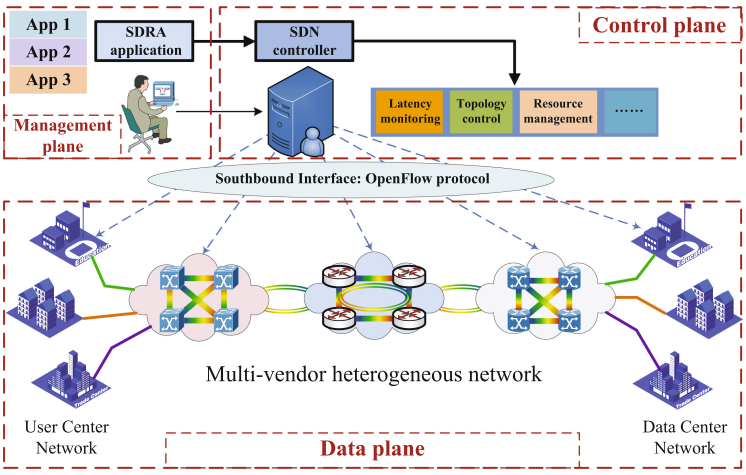

Fig. 1. Network architecture.

2.2 Notation Definitions

(1) u, v: index of OF-Switch, $u, v \in \{1, 2, 3, ..., N\}$.
(2) K: the number of shortest paths.
(3) P_K: the set of K shortest paths.
(4) P^k: the k^{th} shortest path from source node to destination node, where $k \in \{1, 2, 3, ..., K\}$.
(5) $B_{e(u,v)}$: the total bandwidth capacity of link $e(u, v)$.
(6) $B^f_{e(u,v)}$: the free bandwidth capacity of link $e(u, v)$.
(7) H: the maximum number of path length, and we regard that a path is unreachable if its length is more than H hops.
(8) T_c^{lldp}: a time cycle of LLDP process, and it is constant value.
(9) T_c^{echo}: a time cycle of Echo process, and it is constant value.
(10) $T_{e(u,v)}(t)$: the delay of link $e(u, v)$ at time t, and we assume that $T_{e(u,v)}(t) = T_{e(v,u)}(t)$.
(11) $T^{lldp}_{e(u,v)}(t)$: the delay of a LLDP process about link $e(u, v)$ at time t, where $\forall t \in \left\{t_1, t_2, t_3, ..., T_c^{lldp}\right\}$. The process is defined as: first, the controller sends Packet-Out message to OF-Switch u for guiding OF-Switch u forwarding LLDP packet. Then, LLDP packet is transmitted to OF-Switch v through link $e(u, v)$. Finally, OF-Switch v sends Packet-In message to the controller when it receives the LLDP packet.
(12) $T_u^{echo}(t)$: the delay of an Echo process about OF-Switch u at time t, where $\forall t \in \left\{t_1, t_2, t_3, ..., T_c^{echo}\right\}$. The process is that the controller sends Echo Request message to the OF-Switch u for checking the latency between the controller and OF-Switch u. Then, when the OF-Switch u receives the message, it immediately returns an Echo Reply message to the controller.
(13) $e(u, v)$: binary variable, taking 1 if that exists a link between node u and v $(u \neq v)$, and 0 otherwise.
(14) $\varphi^{P^k}_{e(u,v)}$: binary variable, taking 1 if the k^{th} shortest routing path includes link $e(u, v)$ and 0 otherwise.
(15) α_p: binary variable, taking 1 if the path p is selected as the working path and 0 otherwise.

2.3 System Model

We model the network of packet switching by a graph $G(V, E)$, in which $V = \{v_1, v_2, ..., v_N\}$ denotes the set of OpenFlow-enabled switches and $E = \{e_1, e_2, ..., e_M\}$ is the set of bidirectional edges between OF-Switches. N is the total number of OF-Switches in the network, while M is the total number of edges in the network. Note that, $e(u, v)$ taking 1 if that exists a link between node u and v $(u \neq v)$, and 0 otherwise. In a SDN-based packet network, it consists of a controller and multiple OF-Switches. All OF-Switches are responsible for forwarding data packet only. The control function of the network belongs to the upper controller, and OpenFlow protocol is utilized to communicate between the controller and OF-Switch. More specifically, the network is dynamic, and

the status information changes over time. Let $G'(V', E', t)$ represent the network graph at time t, where $\forall t \in \{t_1, t_2, t_3, ..., T_c\}$, and T_c denotes a cycle time of probing network status. Meanwhile, let $T_{e(u,v)}(t)$ denotes the delay of link $e(u, v)$ at time t, and we assume that it is invariant in a cycle T_c. Therefore, we are able to dynamically generate the graph $G'(V', E', t)$ when let $T_{e(u,v)}(t)$ be the weight of the edge in the packet network. In addition, every delay-sensitive service requirement is represented by a 5-tuple: $\langle s, d, x, b_x, t_x \rangle$, where s is a IP address of source host, d is a IP address of destination host, x represents a type index of the delay-sensitive traffic, b_x is the bandwidth requirement, and t_x is the maximum number of latency requirement level.

2.4 Problem Formulation

In reality, the status of packet switching network is varies with time. However, we can investigate the optimal delay problem at a particular time t. Our objective is to minimize the transmission delay. Mathematically, our problem can be formulated as follows (1).

$$Minimize: \sum_{k=1}^{K} \sum_{u=1}^{N} \sum_{v=1}^{N} \left(T_{e(u,v)}(t) \times \varphi_{e(u,v)}^{P^k} \right) \tag{1}$$

$$\sum_{u=1}^{N} \sum_{v=1}^{N} \left[e(u,v) \cdot \varphi_{e(u,v)}^{P^k} \cdot b_x \right] - \sum_{v=1}^{N} \sum_{u=1}^{N} \left[e(v,u) \cdot \varphi_{e(v,u)}^{P^k} \cdot b_x \right] = \begin{cases} b_x, & if\ u = s, \\ -b_x, & if\ u = d, \\ 0, & otherwise. \end{cases} \tag{2}$$

$$\varphi_{e(u,v)}^{P^k} \cdot b_x \leq B_{e(u,v)}^{f} \tag{3}$$

$$\sum_{u=1}^{N} \sum_{v=1}^{N} \varphi_{e(u,v)}^{P^k} \leq H \tag{4}$$

$$T_{e(u,v)}^{lldp}(t) \leq T_c^{lldp} \tag{5}$$

$$T_u^{echo}(t) \leq T_c^{echo} \tag{6}$$

$$\sum_{p \in P_K} \alpha_p = 1 \tag{7}$$

$$\frac{1}{2} \left[T_{e(u,v)}^{lldp}(t) + T_{e(v,u)}^{lldp}(t) - T_u^{echo}(t) - T_v^{echo}(t) \right] = T_{e(u,v)}(t) \tag{8}$$

The constraint (2) states that for all nodes of the network, the outgoing traffic should be equal to incoming traffic except for the source and the destination nodes. Equation (3) represents that the residual bandwidth of each link along the routing path should be higher than the bandwidth requirement of the traffic. Equation (4) limits the path length no longer than H hops. Equations (5) and (6) ensure the effectiveness of the probing time for the both LLDP and Echo processes. Equation (7) ensures that, for each service request, only one candidate path is selected as the working path. Equation (8) calculates the delay of the link according to the probing results from the monitoring module.

3 SDN-Based Delay Solution

3.1 Monitoring Mechanism

Our latency monitoring mechanism consists of the LLDP monitoring module and the Echo monitoring module. LLDP monitoring module is used to obtain the delay of the link discovery process, while Echo monitoring module is responsible for detecting the propagation latency between the controller and the switch. The link discovery process is achieved based on sending a specially crafted packet (i.e., LLDP data packet) through a link from the controller and back while the adjacent switch has no matched flow entry. The controller guides the switch 1 to send LLDP packet through a particular port via a Packet-Out message, and records the current timestamp. Then, the packet is transmitted from the switch 1 to the switch 2 along link 1→2. Since there is no flow entry to match the packet in the switch 2, switch 2 sends Packet-In message to the controller. The controller calculates the latency of LLDP process about link 1→2 when receiving the Packet-In message from switch 2. Similarly, we can get the latency of LLDP process about link 2→1. Let $T^{lldp}_{e(s1,s2)}$ and $T^{lldp}_{e(s2,s1)}$ denote the delay of the above two processes, respectively. In addition, we can also describe the process of Echo monitoring module. Firstly, the controller sends the Echo Request message encapsulated timestamp to the switch. The switch returns the Echo Reply message back to the controller when it receives the Echo Request message. Then, the Echo Monitoring module retrieves the timestamp from the Echo Reply message, and deduces the latency how long it takes for the packet to complete its journey between the controller and the switch. Note that, let T^{echo}_{s1} and T^{echo}_{s2} represent the propagation latency between the controller and the switch 1 and 2, respectively. Thus, we have above four variables. The latency of link S1↔S2 will be $T^{link}_{e(s1,s2)} = T^{link}_{e(s2,s1)} = 1/2 \times (T^{lldp}_{e(s1,s2)} + T^{lldp}_{e(s2,s1)} - T^{echo}_{s1} - T^{echo}_{s2})$, where we assume that the delay of link S1→S2 is equal to link S2→S1.

3.2 Delay-Aware Heuristic Algorithm

We illustrate the detailed procedure of the dynamic SDRA algorithm by using Fig. 2. First, we have constructed a virtual network graph with the edge weight that is the link delay. Based on the virtual network graph, Dijkstra's algorithm is used to calculate the best path from source node A to destination node F, which is A→C→D→F with cost 5. This path becomes the first shortest delay path. Meanwhile, node A becomes the spur node with a root path of itself. The edge A→C is removed because it coincides with the root path. Dijkstra's algorithm is used to compute the spur path again, which is A→B→D→F with a cost of 8. Then, the node C becomes the spur node, while A→C is the root path. We remove the edge C→D from the graph since it coincides with the root path. And, the spur path C→E→F can be obtained by using Dijkstra. The total path is the sum of root path and spur path, i.e., A→C→E→F with cost 7. Finally, we remove the edge D→F, and let node D become the spur node. Another potential path A→C→D→E→F is drawn with a cost of 8. Note that, A→C→E→F is

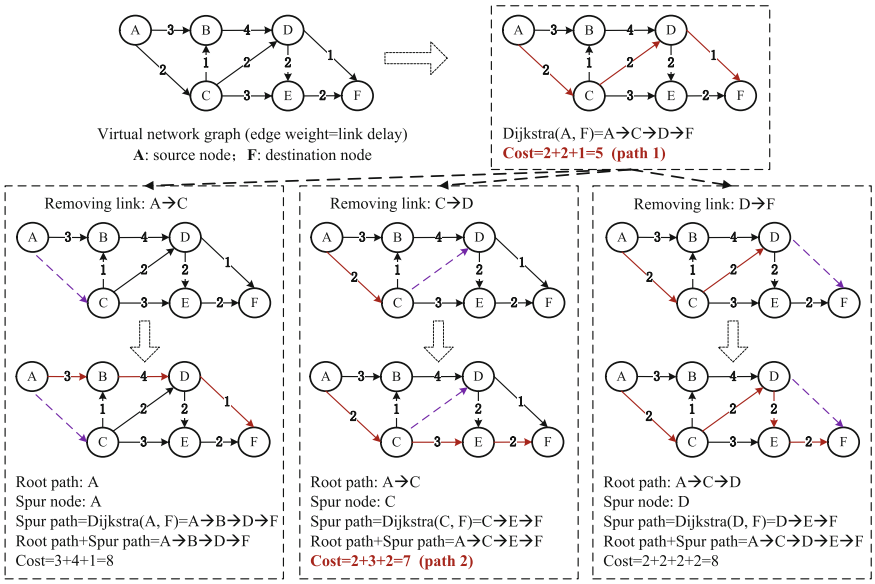

Fig. 2. Delay-aware heuristic algorithm.

chosen to become the second path because it has the lowest cost of 7. Repeat the above iterative process, so that we can dynamically calculate the k-shortest delay path between any two nodes. The purpose of multiple paths is to achieve the fault tolerance of the network, the detailed process can be found in our previous work [14,15].

The time complexity of the algorithm is dependent on the Dijkstra algorithm used in the computation of the spur paths. Dijkstra's algorithm has a worse case time complexity of $O(N^2)$, but using a Fibonacci heap it becomes $O(M + N \log N)$. Since our algorithm makes Kl calls to the Dijkstra in computing the spur paths, where l is the length of spur paths. In a network graph, the value of l is N at the worst case. Therefore, the total time complexity of our algorithm is approximate $O(KN(M + N \log N))$, which is polynomial.

4 Experimental Results and Discussions

To evaluate the feasibility and efficiency of the proposed solution, we first establish a SDN testing environment by utilizing the RYU controller (2 processors, 2 GB memory, and independent network adapter) and the Mininet (2 processors, 2 GB memory, and independent network adapter). The IP address of the controller is 192.168.100.100, whereas the 192.168.100.20 represents the IP address of the mininet.

Next, in our experiments, we focus on a 4-node 4-link 5-host network topology for verifying the feasibility of our SDRA solution. Meanwhile, we also define

a scenario to evaluate the delay parameters of the network. That is a simulation of the client-server model in the real network, i.e., the Host 1 (client) sends the traffic requests to Host 2 and 3 (servers) at the same time. Then, we record the latency of all links in the topology, shown in Fig. 3. In addition, Fig. 4 quantitatively shows the histogram of the end-to-end delay, processing delay and path delay, respectively. We can see that the end-to-end delay is about 15 milliseconds (ms), which consist of the processing delay of the controller, the forwarding delay of the switch and the transmission delay of path. The processing latency of the controller here is the time duration from receiving a traffic request to distributing Flow-Mod message. The path delay is the sum of the delays of all links that the traffic passes from client to server. Therefore, we thank to the introduction of SDN-based delay scheme, the network can achieve dynamic delay balance (e.g., service requests of the host 1→2 and the host 1→3 dynamically select different paths to implement shortest transmission delay).

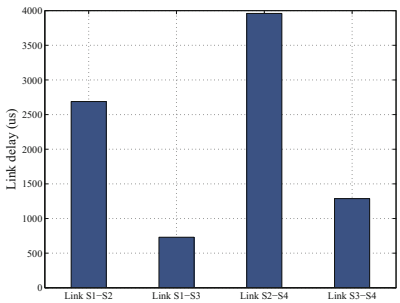

Fig. 3. The link latency.

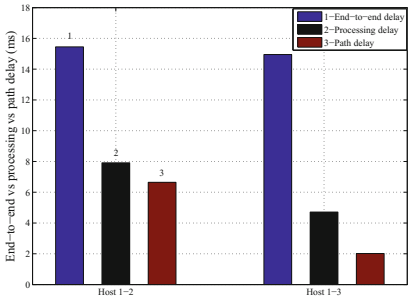

Fig. 4. End-to-end vs processing vs path.

Finally, we conduct a lot of dynamic experiments based on the NSFNET topology [8]. In this experiment, the traffic request is randomly generated between any two nodes. The MHRA algorithm is also compared with the proposed SDRA algorithm in term of both the maximum end-to-end delay and the maximum path delay. As shown in Figs. 5 and 6, the MHRA mechanism makes it possible to achieve a delay control similar to that of SDRA at light traffic loads. However, when the number of traffic request keeps increasing, MHRA increases significantly. This is because a lot of traffic congestion occurs on the shortest path. As a result, the MHRA deteriorates the quality of data transmission. Conversely, the SDRA can turn to the idle path to ease the network pressure and thus ensure the quality of service. With the increasing number of services, the advantage of the proposed SDRA will be more highlighted. The results indicate that the SDN-based delay solution can dynamically select the optimized path to reduce the delay of data transmission, which verifies its feasibility and efficiency.

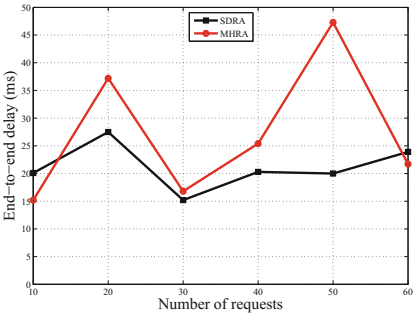

Fig. 5. The maximum end-to-end delay.

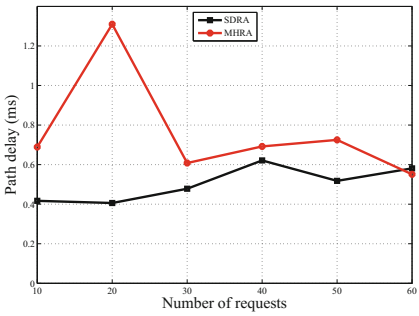

Fig. 6. The maximum path delay.

5 Conclusions

Traditional IP networks were complex and very hard to manage, hence it was difficult to configure the network based on predefined policies or very hard to reconfigure the network adaptive to faults and load variation. Meanwhile, multi-vendor heterogeneous networks also brought new challenges to the flexible control of network device for providing the end-to-end QoS service on demand. Therefore, in this paper, we investigated how to utilize SDN to guarantee the QoS of applications. We first formulated mathematically our problem in term of providing the end-to-end shortest delay. Then, we designed a monitoring mechanism of link latency by using both LLDP and Echo probing modules. Next, a dynamic routing algorithm was proposed to select optimized transmission path based on the information of link latency. More importantly, we developed a routing application that had the monitoring mechanism through extending the RYU controller, and our solution was achieved in a semi-practical SDN testbed. Finally, we quantitatively evaluated the performance of the overall system in terms of the end-to-end delay, the path delay, and the processing latency of the controller. The experimental results showed the system's overall feasibility and efficiency.

Acknowledgements. This work is supported by the National Nature Science Foundation of China under Grant 61401082, in part by the General Armament Department and Ministry of Education United Fund under Grant 6141A0224-003, in part by the Fundamental Research Funds for the Central Universities under Grant N161604004 and Grant N161608001, and in part by National Scholarship Foundation of China.

References

1. Braden, R., Clark, D., Shenker, S.: Integrated services in the internet architecture: an overview, June 1994
2. Blake, S., Black, D., Carlson, M., Davies, E., Wang, Z., Weiss, W.: An architecture for differentiated services, December 1998

3. Rosen, E., Rekhter, Y.: BGP/MPLS VPN, March 1999
4. Hou, W., Ning, Z., Guo, L., Chen, Z., Obaidat, M.: Novel framework of risk-aware virtual network embedding in optical data center networks. IEEE Syst. J. **PP**(99), 1–10 (2017). https://doi.org/10.1109/JSYST.2017.2673828
5. Kreutz, D., Ramos, F., Verissimo, P., Rothenberg, C., Azodolmolky, S., Uhlig, S.: Software-defined networking: a comprehensive survey. Proc. IEEE **103**, 14–76 (2015)
6. Hou, W., Ning, Z., Guo, L., Zhang, X.: Temporal, functional and spatial big data computing framework for large-scale smart grid. IEEE Trans. Emerg. Top. Comput. **PP**(99), 1–11 (2017). https://doi.org/10.1109/TETC.2017
7. Hou, W., Tian, G., Guo, L., Wang, X., Zhang, X., Ning, Z.: Cooperative mechanism for energy transportation and storage in internet of energy. IEEE Access **5**, 1363–1375 (2017). https://doi.org/10.1109/ACCESS.2017.2664981
8. Zhang, X., Guo, L., Hou, W., Wang, S., Zhang, Q., Guo, P., Li, R.: Experimental demonstration of an intelligent control plane with proactive spectrum defragmentation in SD-EONs. Opt. Express **25**(20), 24837–24852 (2017)
9. Ning, Z., Hu, X., Chen, Z., Zhou, M., Hu, B., Cheng, J., Obaidat, M.: A cooperative quality-aware service access system for social internet of vehicles. IEEE Internet Things J. (2017). https://doi.org/10.1109/JIOT.2017.2764259
10. Ning, Z., Xia, F., Ullah, N., Kong, X., Hu, X.: Vehicular social networks: enabling smart mobility. IEEE Commun. Mag. **55**(5), 49–55 (2017)
11. Hou, W., Ning, Z., Guo, L.: Green survivable collaborative edge computing in smart cities. IEEE Trans. Ind. Inform. (2018). https://doi.org/10.1109/TII.2018.2797922
12. Hou, W., Zhang, R., Qi, W., Lu, K., Wang, J., Guo, L.: A provident resource defragmentation framework for mobile cloud computing. IEEE Trans. Emerg. Top. Comput. **6**(1), 32–44 (2015). https://doi.org/10.1109/TETC.2015.2477778
13. Phemius, K., Bouet, M.: Monitoring latency with OpenFlow. In: International Conference on Network & Service Management, pp. 122–125 (2013)
14. Zhang, X., Hou, W., Guo, L., Wang, S., Sun, Y., Yang, X.: Failure recovery solutions using cognitive mechanisms for software defined optical networks. In: 2016 15th International Conference on Optical Communications and Networks, pp. 1–3 (2016)
15. Zhang, X., Guo, L., Hou, W., Zhang, Q., Wang, S.: Failure recovery solutions using cognitive mechanisms based on software defined optical network platform. Opt. Eng. **56**(1), 1–14 (2017)