Joint Optimization of Virtual Function Migration and Rule Update in Software Defined NFV Networks

Jie Zhang*, Deze Zeng*, Lin Gu†, Hong Yao* and Muzhou Xiong*
*School of Computer Science, China University of Geosciences, Wuhan, China
†School of Computer Science and Technology, Huazhong University of Science and Technology, Wuhan, China

***Abstract*—Emerging technologies such as Software-Defined Networks (SDN) and Network Function Virtualization (NFV) promise to address cost reduction and flexibility in network operation while enabling innovative network service delivery. To catch up with the time-varying traffic demands, the network changes frequently. We should come up with a sequence of instructions to manipulate the starting network into the goal network, while preserving the network semantics correctness (e.g., freedom of loops, bandwidth guaranteeing). In this case, how to migrate the virtual network functions (VNF) and update the flow forwarding rules efficiently is an important and challenging problem. In this paper, we are motivated to address the migration of VNF and flow update rule problem with joint consideration of migration cost and update delay. The problem is first formulated into a mixed integer non-linear programming (MINLP). By linearizing and relaxing the MINLP, we then present a polynomial-time two-stage heuristic algorithm. The high efficiency of our algorithm is extensively validated by simulation based studies by the fact that it performs much closer to the optimal solution.**

I. Introduction

To simplify the network management and make efficient use of network resources, network function virtualization (NFV) and software defined networks (SDNs) have been recognized as the promising next-generation networking technologies. NFV is proposed to run the network functions as software instances on commodity servers, while SDN decouples the network control plane from the data plane. During network operation, network updates need to be implemented frequently for various reasons. For example, when traffic demand or network topology changes (e.g., network device failure) at runtime, it is necessary to dynamically adjust the network configuration to meet the predefined Quality-of-Service (QoS). In other cases, virtual network functions (VNF) may need to be migrated from one server to another to adapt to the network changes. A growing number of studies [7], [17], [18] have focused the problem of VFN placement and migration to preserve consistency of NFV state so as to ensure the network service correctness during migration. Meanwhile, the forwarding rules of flows shall also be updated accordingly to reserve the correctness of network service semantics.

Forwarding rule updating has been regarded as a common challenge faced in SDNs [1], [8]. In SDN, the forwarding is determined by per-flow rule that SDN switches applied to the packets of each flow. During the process of network update, the controller instructs switches to add, change or remove some rules. To avoid service disruption, both forwarding correctness (i.e., packet delivery) and policies (i.e., requirements on forwarding paths) have to be preserved throughout the update process. Moreover, the update strategy must be robust with respect to various factors such as non-deterministic processing time on the switch to install or modify rules, transmission latency between the controller and the switches. Therefore, upon the given *starting* network and *goal* network, we should come up with a sequence of network update instructions, to manipulate the starting network into the goal network, while preserving predefined network semantics correctness (e.g., freedom of loops, and bandwidth guaranteeing). Much effort has been devoted to network update in SDNs. Some studies focus on consistency preservation (e.g., congestion avoidance [2]–[4], forwarding correctness [5], [6] and policy preservation [1], [9], [12], [13]). Some others intend to update the network fast and efficiently [10], [11], [14]–[16]. We notice that all existing studies assume a given goal network. While, in NFV-enabled environment, it is possible to shape the goal network by flexibly migrating the VNFs according to the network conditions towards various objectives. However, none of them discuss the issues of VNF migration together with rule update.

In this paper, we are motivated to jointly investigate the VNF migration and rule update in software defined NFV networks. When we need to re-shape the network according to the runtime network conditions, the following issues shall be taken into consideration. Migrating a VNF from one server to another inevitably incurs migration cost, which is proportional to the distance between the source server and the destination server, and is non-ignorable. Different migration decisions not only lead to different migration cost but also different goal network topologies, which further influence the forwarding rule update decisions (i.e., rule update sequence) and hence the update delay. Therefore, it is essential to balance the network migration cost and update delay with joint consideration of VNF migration and rule update sequence.

Our main contributions are summarized as follows:

- We jointly address the VNF migration and forwarding rule update to balance the migration cost and network update delay. To our best knowledge, we are the first to take the goal network re-shaping into network update.
- We formally describe the problem into a mixed integer non-linear optimization programming (MINLP) problem.

978-1-5090-5019-2/17/$31.00 ©2017 IEEE

By linearizing and relaxing the MINLP, a polynomial-time two-stage heuristic algorithm is proposed.
- Through extensive simulations, the high efficiency of our heuristic algorithm is verified by the fact that it performs much close to the optimal solution.

The rest of the paper is organized as follows. Section II introduces the system model and states the problem to be studied in this paper. An MINLP formulation is presented to formally describe the problem in Section II-B. Section III proposes a two-phase heuristic algorithm based on and relaxation. The performance of our proposed algorithm is verified by simulations in Section IV. Finally, Section V concludes this paper.

II. System Model and Problem Formulation

In this section, we first present a general system model for our study. Then, based on the system model discussed above, we propose an MINLP model to formally describe the problem studied in this paper.

A. System Model

Without loss of generality, we consider a software defined NFV networks as a graph $G = (V, E)$, where V is the set of servers and switches, and E is the set of links between servers. Some VNFs have already been deployed on some servers and we denote the set of VNFs placed in server $v \in V$ as F_v. We assume that a set F of VNFs need to be migrated out of their current host server. $K = |F|$ defines the number of to-be-migrated VNFs. We use B_f to denote traffic demand of flow through function $f \in F$, and R_f to represent the number of flow entries that need to be installed per server to route flow on function f.

The old routing path of flow on function f is denoted by P_f, and the new path is denoted by P'_f. A path from a source server v_{sf} to a destination server v_{df} is defined as the list of traversed servers $\{v_{sf} = v_0, v_1, v_2, ..., v_k = v_{df}\}$, and the links connecting them, where $v_i \in V$ is the next hop server of v_{i-1} and $(v_{i-1}, v_i) \in E$. The paths are loop-free and each server has only one next hop. We use c_e to represent the capacity of link $e \in E$ and q_v to represent the flow table size of switch $v \in V$. The resource capacity on server $v \in V$ is denoted as C_v and the resource demand of function $f \in F$ is S_f. When a VNF $f \in F$ is migrated to server $v \in V$, certain migration cost d_f^v will be incurred. We assume that the migration cost is proportional to the distance between servers, which can be easily derived from G.

B. Problem Formulation

1) Migration Cost: As there are K to-be-migrated VNFs, at most K steps are needed to update the forwarding rules without violating the flow table size constraints. We define a binary variable x_{fv}^k to indicate whether f is migrated to server v in step $k \in [1, K]$. For each to-be-migrated VNF $f \in F$, it must be migrated to one server $v \in V$ with enough resource to accommodate it eventually. Therefore, we shall have:

$$\sum_{k=1}^{K} \sum_{v \in V} x_{fv}^k = 1, \forall f \in F, \tag{1}$$

and

$$\sum_{f \in F} x_{fv}^k \cdot S_f + \sum_{f' \in F_v} S_{f'} \leq C_v, \forall v \in V, k \in [1, K]. \tag{2}$$

2) Update Delay: To avoid congestion, part of flows will be migrated to new path during each update step. A variable $y_f^k, 0 \leq y_f^k \leq 1$ is defined to indicate the fraction of flows of VNF $f \in F$ updated from the old path P_f to the new one P'_f in step k. All flows of each migrated VNF must be updated eventually, i.e.,

$$\sum_{k=1}^{K} y_f^k = 1, \forall f \in F. \tag{3}$$

Moreover, the process of rule update is related to migration decision. Only when the corresponding function is migrated to the new server, it is possible that $y_f^k \geq 0$. That is,

$$y_f^k \leq \sum_{a=1}^{k} \sum_{v \in V} x_{fv}^a, \forall f \in F, k \in [1, K]. \tag{4}$$

We use binary variable $n_k, k \in [1, K]$ to indicate whether the rule update is finished at step k. That is, if there are some rules updated in step k, $n_k = 1$; otherwise, $n_k = 0$. Obviously, the value of n_k is determined by the value of y_f^k as $n_k = 1$ if and only if there exists a $y_f^k, \forall f \in F$ with value larger than 0; otherwise, $n_k = 0$. We can describe such relationship as

$$n_k \geq \sum_{f \in F} \frac{y_f^k}{K}, \forall k \in [1, K]. \tag{5}$$

Note that as long as the update is finished in step k, there will be no more update operations in the subsequent steps. Therefore, we have:

$$n_k \geq n_{k+1}, \forall k \in [1, K-1]. \tag{6}$$

Next, we analyze the link capacity constraints and flow table size constraints of the problem. Let l_e^k and r_v^k represent the traffic load on link e and the number of rules on switch v in step k, respectively. The initial traffic load l_e^0 on link e can be calculated by summing up all the traffic load of the flows routed through link e as $l_e^0 = \sum_{f \in F} B_f \alpha_{fe}$, where $\alpha_{fe} = 1$ if link $e \in P_f$; otherwise, $\alpha_{fe} = 0$. During rule update process, the old path and new path may coexist. We shall always guarantee that the total bandwidth requirement does not exceed the link capacity, i.e., congestion avoidance. The bandwidth requirement on a link e in step k can be calculated via adding up the new bandwidth requirement of new path and subtracting the requirement of the old path migrated out as

$$l_e^k = l_e^{k-1} + \sum_{f \in F} B_f y_f^k \alpha'_{fe} - \sum_{f \in F} B_f y_f^k \alpha_{fe}, \tag{7}$$

If $e \in P'_f$, $\alpha'_{fe} = 1$; otherwise, $\alpha'_{fe} = 0$. by which we can iteratively calculate the link bandwidth consumption on link e in step k as

$$l_e^k = l_e^0 + \sum_{a=1}^{k} \sum_{f \in F} B_f y_f^a \alpha'_{fe} - \sum_{a=1}^{k} \sum_{f \in F} B_f y_f^a \alpha_{fe} \quad (8)$$

Similarly, we can calculate the flow table size requirement on switch v in step k as follows. We first calculate the initial flow table size requirement as $r_0^v = \sum_{f \in F} R_f \gamma_{fv}$, where $\gamma_{fv} = 1$ if $v \in P_f$; otherwise, $\gamma_{fv} = 0$. To re-route a flow to its new path, one rule shall be added in the corresponding switches, regardless of the network traffic. Note that the old rule cannot be removed until the whole flow has been updated. We introduce two binary variables zo_{kf} and zn_{kf} to represent whether the old rules of function f can be removed and whether the new rules shall be added in step i, respectively, i.e.,

$$zo_{kf} = \begin{cases} 0, & \text{if } \ k = 1 \\ 0, & \text{if } \ k \in [2, K] \text{ and } \sum_{a=1}^{k-1} y_f^a < 1 \\ 1, & \text{if } \ k \in [2, K] \text{ and } \sum_{a=1}^{k-1} y_f^a = 1 \end{cases} \quad (9)$$

$$zn_{kf} = \begin{cases} 1, & \text{if } \ \sum_{a=1}^{k} y_f^a > 0 \\ 0, & \text{otherwise.} \end{cases}, \quad k \in [1, K]. \quad (10)$$

We can then derive the expression of r_i^v as,

$$r_v^k = r_v^0 + \sum_{f \in F} R_f zn_{kf} \gamma'_{fv} - \sum_{f \in F} R_f zo_{kf} \gamma_{fv}. \quad (11)$$

The expression of γ'_{fv} s is similar with that of γ_{fv}, except that P_f is replaced with P'_f.

3) Joint Optimization: As migrating f to server v incurs cost d_f^v, the overall migration cost therefore can be calculated as $\sum_{k=1}^{K} \sum_{v \in V} \sum_{f \in F} x_{fv}^k \cdot d_f^v$. As n_k indicates whether there is any rule update in step k, we can calculate the total number of steps, i.e., network update delay, to finish the rule update as $\sum_{k=1}^{K} n_k$. Our objective is to minimize the migration cost and the update delay, there exists certain tradeoff between the two issues. We introduce variable $\beta \in (0, 1)$ as a bias factor to balance the two issues. Taking all the constraints discussed above, we formulate the joint optimization problem into MINLP form as:

$$\min: \quad \beta \sum_{k=1}^{K} \sum_{v \in V} \sum_{f \in F} x_{fv}^k \cdot d_f^v + \sum_{k=1}^{K} n_k \quad (12)$$

$$\text{s.t.} \quad l_e^k \le c_e, \forall e \in E, k \in [1, K], \quad (12a)$$

$$r_v^k \le q_v, \forall v \in V, k \in [1, K], \quad (12b)$$

$$(1), (2), (3) - (6). \quad (12c)$$

III. Heuristic Algorithm Design

Generally, MINLP is regarded as NP-hard and is computationally prohibitive to obtain the optimal solution when the problem size is large. To this end, we propose a linearization and relaxation based heuristic algorithm in this section.

A. Problem Reformulation

The non-linear terms exist in (8) is a product of two variables (i.e., $\alpha_{fe} y_f^a$, $\alpha'_{fe} y_f^a$). We notice that we can equivalently linearize them by introducing auxiliary variables $p'^a_{fe} = \alpha_{fe} \cdot y_f^a$ and $p_{fe}^a = \alpha'_{fe} \cdot y_f^a$, respectively. Then we obtain:

$$\begin{cases} l_e^k = l_e^0 + \sum_{a=1}^{k} \sum_{f \in F} B_f p'^a_{fe} - \sum_{a=1}^{k} \sum_{f \in F} B_f p_{fe}^a \\ \sum_{e \in E} p'^a_{fe} = y_f^a, \quad \forall f \in F, a \in [1, K] \\ \sum_{e \in E} p_{fe}^a = y_f^a, \quad \forall f \in F, a \in [1, K] \\ p'^a_{fe}, p_{fe}^a \ge 0, \quad \forall e \in E, f \in F, a \in [1, K] \end{cases} \quad (13)$$

Similarity, we introduce two new variables $m_{fv}^{k\,\prime}$ and m_{fv}^k to substitute the corresponding terms in constraint (11). Then, we obtain:

$$\begin{cases} r_v^k = r_v^0 + \sum_{f \in F} R_f m'^k_{fv} - \sum_{f \in F} R_f m_{fv}^k \\ \sum_{v \in V} m'^k_{fv} = zn_f^k, \quad \forall f \in F, k \in [1, K] \\ \sum_{v \in V} m_{fv}^k = zo_f^k, \quad \forall f \in F, k \in [1, K] \\ m'^k_{fv}, m_{fv}^k \ge 0, \quad \forall v \in V, f \in F, k \in [1, K] \end{cases} \quad (14)$$

Finally, we can reformulate the original MINLP problem into an equivalent ILP problem as:

$$\min \quad \beta \sum_{k=1}^{K} \sum_{v \in V} \sum_{f \in F} x_{fv}^k \cdot d_f^v + \sum_{k=1}^{K} n_k \quad (15)$$

$$\text{s.t.} \quad l_e^0 + \sum_{a=1}^{k} \sum_{f \in F} B_f p'^a_{fe} - \sum_{a=1}^{k} \sum_{f \in F} B_f p_{fe}^a \le c_e \quad (15a)$$

$$r_v^0 + \sum_{f \in F} R_f m'^k_{fv} - \sum_{f \in F} R_f m_{fv}^k \le q_v \quad (15b)$$

$$(1), (2), (3) - (6), (13), (14).$$

B. Algorithm Design

After linearizing the original problem, we can then apply relaxation method to approximate the optimal solution in polynomial time. The proposed algorithm is shown in Algorithm. 1. The algorithm is divided into two phases. In the first phase, we relax the integer variable x_{fv}^k and n_k to obtain an LP problem, which can be solved in polynomial time to obtain fractional solutions as $\widetilde{x}$ and $\widetilde{n}$ (line 2-3). As the fractional solutions do not reflect the actual VNF migration and rule update decisions, in the second phase, we try to round them to the 0-1 solution. We define the set of un-updated virtual functions as F^u, which is initialized as F (line 5). In the beginning, we initialize each $x_{fv}^k = 0, \forall v \in V, k \in [1, K]$ (line 6). Intuitively, the one with higher value shall be with higher priority to be converted into 1. Therefore, we first try to set the maximum x_{fv}^k in $\widetilde{x}$ be 1 and the others still be 0 (line 8). Since the process of rule update is related to migration decision, we must ensure that the currently selected x_{fv}^k is feasible. We regard $x_{fv}^k = 1$ as a known condition, and put it into constraint (2) to check whether the (2) is satisfied or not

Algorithm 1 Relaxation-based VNF migration and rule update

1: **Phase 1: Solve the Relaxed Problem (12).**
2: Relax the integer variable x_{fv}^k and n_k
3: Obtain a fraction solution $\widetilde{x}$ and $\widetilde{n}$
4: **Phase 2: Round to 0-1 solution**
5: $F^u = F$
6: $\forall x_{fv}^k \leftarrow 0$
7: **while** $F^u \neq \emptyset$ **do**
8: Choose a maximum x_{fv}^k, $f \in F^u$, set $x_{fv}^k = 1$
9: Check the constraint (2)
10: **if** the currently selected x_{fv}^k is infeasible **then**
11: Set $x_{fv}^k = 0$
12: Choose next x_{fv}^k with smaller value
13: **else**
14: $F^u = F^u - f$
15: Choose a maximum n_k, set $n_k = 1$
16: Regard $n_k = 1$ as known, resolve the problem (12).
17: Obtain fractional solutions $\widetilde{x}$ and $\widetilde{n}$
18: **end if**
19: **end while**

(line 9). If the currently selected x_{fv}^k satisfies the condition, we can determine the value of n_k. Otherwise, a new larger x_{fv}^k will be selected (lines 11-12). Once the feasible values of x_{fv}^k is found, we similarly choose a maximum n_k from the fractional solution $\widetilde{n}$, and set it be 1. After that, we will regard $x_{fv}^k = 1$ and $n_k = 1$ as known, and update the problem (12). Therefore, we will obtain a new fractional solution $\widetilde{x}$ and $\widetilde{n}$ by solving the new model. The above process is repeated until every virtual function in F is migrated and all corresponding rules are updated (lines 14-17).

IV. Performance Evaluation

To validate the efficiency of our proposed algorithm, we have conducted extensive simulation based studies. In this section, we report our experiment results.

A. Experiment Settings

In order to show the advantage of our joint optimization on VNF migration and rule update, we compare our proposed algorithm ("Joint") against the one without joint optimization ("Un-joint"). At the same time, we also solve the ILP in (12) to obtain the optimal solution ("Optimal") using Gurobi optimizer. For all algorithms, we use regular topologies generated by NetworkX. The capacity of each link is set as 100. We investigate how our algorithm performs and how various parameters affect the migration cost and update delay by varying the settings of flow table size, flow number, and bias factor β in each experiment group, respectively. As our objective is to minimize and balance the the migration cost and update delay. We therefore evaluate the performance of all the algorithms in the metric of both migration cost and update delay. For each group of experiment, we run the simulation for 100 times to obtain the average value.

B. On the effect of flow table size

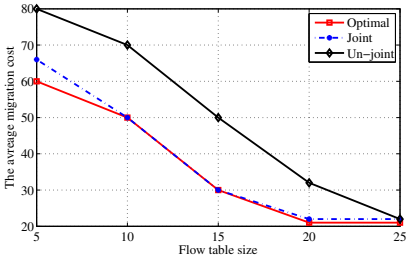
First, we show the migration cost and update steps under different settings of flow table size varying from 5 to 25. There are 20 service chains to update. Fig. 1 shows the evaluation results. From both Fig. 1(a) and Fig. 1(b), we can see that both the migration cost and the number of update steps (i.e., update delay) first decrease and finally converge with the increasing of flow table size. This is because when the flow table size is small, the number of forwarding rules that can be stored in the flow table is limited. This imposes limitation to both the migration as well as the rule update decisions and therefore more steps are required to update all rules without violating the flow table size constraints. With the increase of flow table size, more rules can be stored in the flow table and hence more locations become available to migrate and it is easy to update quickly. Finally, when the flow table size becomes large enough, e.g., from 20 to 25 in Fig. 1(a) and Fig. 1(b), further increasing the flow table size does not take any benefit any more as the flow table size already satisfies the VNF migration and rule update requirement. In addition, we can always observe from Fig. 1 that our joint-optimization algorithm outperforms the un-joint one, and performs much close to optimal under any flow table size.

C. On the effect of the number of VNFs

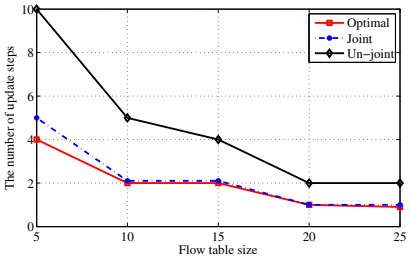
Then, we investigate the impact of the number of to-be-migrated VNFs varied from 10 to 50. For each switch, the flow table size is fixed as 15. The results are shown in Fig. 3. It is obvious that both the migration cost and the update delay of all three algorithms show as an increasing function of the number of VNFs. The reason behind such phenomenon is that as the number of functions increases, the resource requirements increase while all network resources (e.g., link capacity, flow table size, etc.) are capacity-limited. Without doubt that this shall lead to higher migration cost and more update steps to ensure that the capacity constraints are not violated. Nevertheless, we shall always observe the high efficiency of our algorithm as it performs close to the optimal solution and significantly outperforms the un-joint one.

D. On the Effect of Bias Factor β

As has been demonstrated, we use bias factor β to balance the tradeoff between the migration cost and the network update delay. To investigate its effect, we vary the value of β from 0 to 1 and report the results in Fig. 3, where we consider 20 VNFs to migrate and the flow table size is 15 on each switch. Concerning the migration cost, we can see that the migration cost shows as a decreasing function of β, as shown in Fig. 3(a). This is because when β is small, we emphasize more on the update delay. Consequently, we can see that the network update delay shows as an increasing function of β. When β is close to 1, we almost fully focus on the migration cost and do not care about the rule update delay.

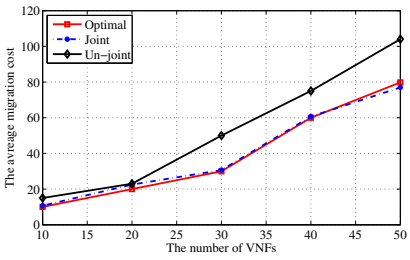


(a) Migration cost

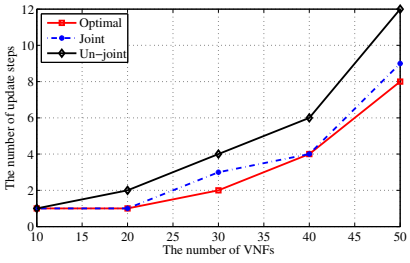


(b) Rule update delay

Fig. 1. The migration cost and update delay under different values of flow table size

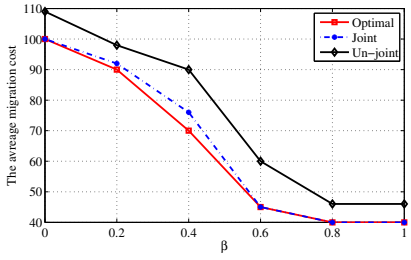


(a) Migration cost

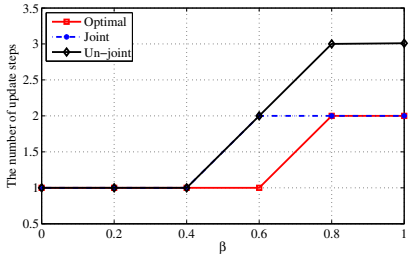


(b) Rule update delay

Fig. 2. The migration cost and update delay under different number of VNFs



(a) Migration cost



(b) Rule update delay

Fig. 3. The migration cost and update delay under different values of β

V. Conclusion

In this paper, we investigate the problem of VNF migration and forwarding rule update, aiming at minimizing the migration cost and rule update delay. We discover that the VNF migration has a deep influence on the rule update decision and therefore they two shall be considered in a joint manner. We formulate the joint optimization problem into an MINLP form. To address the computation complexity, we further propose a polynomial-time heuristic algorithm based on linearization and relaxation. By extensive simulations, we prove the correctness of our joint design philosophy and the high efficiency of our algorithm by the fact it performs much close to the optimal solution and significantly outperforms the un-joint algorithm.

VI. Acknowledgments

This research was supported by the NSF of China (Grant No. 61602199, 61402425, 61772480, 61673354, 61672474), the Provincial NSF of Hubei (Grant No. 2016CFB107).

References

[1] J. McClurg, H. Hojjat, P. Černý, and N. Foster, "Efficient Synthesis of Network Updates," in *Proc. of the 36th ACM PLDI*. ACM, 2015, pp. 196–207.

[2] C.Y. Hong, S. Kandula, R. Mahajan, M. Zhang, V. Gill, M. Nanduri, and R. Wattenhofer, "Achieving High Utilization With Software-Driven Wan," in *ACM SIGCOMM CCR*, vol. 43, no. 4. ACM, 2013, pp. 15–26.

[3] S. Ghorbani and M. Caesar, "Walk the Line: Consistent Network Updates With Bandwidth Guarantees," in *Proc. of the 1st Workshop on Hot Topics in Software Defined Networks*. ACM, 2012, pp. 67–72.

[4] H. H. Liu, X. Wu, M. Zhang, L. Yuan, R. Wattenhofer, and D. Maltz, "Zupdate: Updating Data Center Networks With Zero Loss," in *ACM SIGCOMM CCR*, vol. 43, no. 4. ACM, 2013, pp. 411–422.

[5] R. Mahajan and R. Wattenhofer, "On Consistent Updates in Software Defined Networks," in *Proc. of the 12th ACM Workshop on Hot Topics in Networks*. ACM, 2013, p. 20.

[6] S. Vissicchio, L. Cittadini, O. Bonaventure, G. G. Xie, and L. Vanbever, "On the Co-Existence of Distributed and Centralized Routing Control-Planes," in *Proc. of INFOCOM*. IEEE, 2015, pp. 469–477.

[7] J. Liu, Z. Jiang, N. Kato, O. Akashi, and A. Takahara, "Reliability Evaluation for NFV Deployment of Future Mobile Broadband Networks," *IEEE Wireless Communications Magazine*, vol. 23, no.3, pp. 90-96, Jun. 2016.

[8] J. Liu, S. Zhang, N. Kato, H. Ujikawa, and K. Suzuki, "Device-to-Device Communications for Enhancing Quality of Experience in Software Defined Multi-Tier LTE-A Networks," *IEEE Network Magazine*, vol. 29, no. 4, pp. 46-52, July 2015.

[9] A. Ludwig, M. Rost, D. Foucard, and S. Schmid, "Good Network Updates for Bad Packets: Waypoint Enforcement Beyond Destination-Based Routing Policies," in *Proc. of the 13th ACM Workshop on Hot Topics in Networks*. ACM, 2014, p. 15.

[10] Y. Liu, Y. Li, Y. Wang, A. V. Vasilakos, and J. Yuan, "Achieving Efficient and Fast Update for Multiple Flows in Software-Defined Networks," in *Proc. of the 2014 ACM SIGCOMM Workshop on Distributed Cloud Computing*. ACM, 2014, pp. 77–82.

[11] W. Zhou, D. K. Jin, J. Croft, M. Caesar, and P. B. Godfrey, "Enforcing Customizable Consistency Properties in Software-Defined Networks." in *Proc. of NSDI*, 2015, pp. 73–85.

[12] W. Wang, W. He, J. Su, and Y. Chen, "Cupid: Congestion-Free Consistent Data Plane Update in Software Defined Networks," in *Proc. of INFOCOM*. IEEE, 2016, pp. 1–9.

[13] S. Vissicchio and L. Cittadini, "Flip the (Flow) Table: Fast Lightweight Policy-preserving SDN Updates," in *Proc. of INFOCOM*. IEEE, 2016, pp. 1–9.

[14] X. Jin, H. H. Liu, R. Gandhi, S. Kandula, R. Mahajan, M. Zhang, J. Rexford, and R. Wattenhofer, "Dynamic Scheduling of Network Updates," in *ACM SIGCOMM CCR*, vol. 44, no. 4. ACM, 2014, pp. 539–550.

[15] X. Wen, B. Yang, Y. Chen, L. E. Li, K. Bu, P. Zheng, Y. Yang, and C. Hu, "Ruletris: Minimizing Rule Update Latency for TCAM-based SDN Switches," in *Proc. of ICDCS*. IEEE, 2016, pp. 179–188.

[16] A. Rasmussen, A. Kragelund, M. Berger, H. Wessing, and S. Ruepp, "TCAM-based High Speed Longest Prefix Matching with Fast Incremental Table Updates," in *Proc. of HPSR*, 2013, pp. 43–48.

[17] A. Gember-Jacobson, R. Viswanathan, C. Prakash, R. Grandl, J. Khalid, S. Das, and A. Akella, "OpenNF: Enabling Innovation in Network Function Control," in *ACM SIGCOMM CCR*, vol. 44, no. 4, pp. 163–174, 2015.

[18] S. Rajagopalan, D. Williams, H. Jamjoom, and A. Warfield, "Split/Merge: System Support for Elastic Execution in Virtual Middleboxes.," in *Proc. of NSDI*, vol. 13, pp. 227–240, 2013.