Minimizing Congestion Impairment of Network Update in SDN: A Flow-Based Solution

Chaozhun Wen, Peng Yang, Qiong Liu, Jingjing Luo, and Li Yu
School of Electronic Information and Communications, Huazhong University of Science and Technology, Wuhan, China
Email: {czwen, yangpeng}@hust.edu.cn, {liuqiong2018, luojingjing1989}@gmail.com, hustlyu@hust.edu.cn

***Abstract*—In software defined networks, network states are frequently updated by controllers. Unfortunately, due to resource and time constraints, there are scenarios in which transient congestion and packet loss are inevitable. In this regard, minimizing the packet loss ratio becomes crucial. Previous efforts on congestion-free updates suggest *link-based* solutions, which aim at minimizing the overloaded data volume on the bottleneck links. Observing the fact that *the least overloaded data volume on links still does not guarantee the least packet loss*, in this paper, we propose a *flow-based* update solution that directly minimizes the packet loss by jointly optimizing the congestion duration and rate limitation. Specifically, *congestion impairment* is defined to jointly accommodate the flow's importance and packet loss. Then, we present the FBU (Flow-Based Update problem), which minimizes the congestion impairment on a flow basis. To deal with the NP-hardness of this optimization problem, we propose `MIC`, which is an efficient two-phase heuristic algorithm based on the relationship between rate limitation, congestion duration and packet loss. Experimental results show that `MIC` can reduce up to 84% of packet loss compared to previous algorithms.**

I. Introduction

Software Defined Network (SDN) separates the control plane from the data plane, enables agile network management [1], rapid network updates [2] and improved resource allocation strategies [3]. Thanks to the timely global view on network states, network controller can continuously make centralized decisions for higher resource utilization [4]. As a result, network states are dynamically changed for various reasons, such as link failure, routing reconfiguration and hardware maintenance. In most cases, the network is designed to perform better after the update. However, transient link congestion is inevitable if the update process is not properly scheduled [5]. Since the update operations in SDN are not atomic, bottleneck links may be congested when newly scheduled flows arrive at certain links before previous flows leave, leading to packet loss on those overloaded links [6].

When the link capacity is inadequate or the time constraint is stringent, network congestion and packet loss are unavoidable [7][8]. There are extensive efforts seeking to minimize the packet loss ratio, resulting in a series of *link-based* algorithms [9], i.e., algorithms that seek to minimize the overloaded data volume on bottleneck *links* [10][11]. However, *these update schemes still have great difference in packet loss even they all managed to minimize the overloaded data volume on links*. The rationale is two-fold. Firstly, update schemes that lead to the same overloaded data volume have various *congestion duration*, i.e., the time period when a flow is congested. With the increase of the congestion duration, the flow's dropped packets also increases. Secondly, update schemes with the same overloaded volume also differ in the lost data volume of each flow traversing the bottleneck links. When a flow f_1 traversing n bottleneck links gives up x units of data, all those links can free up x units of capacity. On the contrary, if each bottleneck link only drops packets of its proprietary flows (Unlike f_1, these flows do not traverse other bottleneck links) to free up x units of capacity, the total packet loss will be nx. Thus with the same overloaded data volume on bottleneck links but different rate limitation for flows, link-based update schemes may still lead to various packet loss scenarios.

Link-based solutions cannot make the congestion duration and rate limitation in control, thus they cannot minimize the packet loss ratio inherently. To proactively optimize the rate limitation and congestion duration, we resort to *flow-based* update solution. As a by-product, flow-based strategy can distinguish flows with different importance and offer differentiated services. We introduce the importance weight of flows and define *congestion impairment* as the weighted sum of dropped packets of all flows.

Specifically, in this paper, flows are considered to be unsplittable, and network is updated by migrating target flows though a series intermediate stages [5][10][11]. For a update process with S intermediate stages, flows will be migrated by $S+1$ times. Because the cost time of transient congestion is fluctuating and cannot be known in advance, in this paper, the congestion duration of each flow is measured by the migration times with congestion. We present the FBU (Flow-Based Update) problem: given the link capacity and time constraints, set a few intermediate stages and find a set of flows to perform rate limitation with the aim of minimizing congestion impairment. The FBU problem is then formulated as a nonlinear optimization problem. Due to its NP-hardness, a two-phase heuristic algorithm `MIC` is proposed to address the problem. The algorithm decomposes the problem into two key phases, based on the observation that the congestion duration and rate limitation are the essential factors that affect packet loss. Each phase addresses one factor respectively. It is important to point that congestion-free update is a special case of our strategy, when the congestion impairment is depressed to zero. Experimental results show that our algorithm can reduce the congestion impairment up to 84% than other benchmark algorithms. The contribution of this paper can be summarized as follows:

978-1-5386-6119-2/18/$31.00 ©2018 IEEE

- Instead of existing link-based solutions that neglect the congestion duration and the rate limitation, we propose a flow-based SDN network update strategy. This strategy directly minimizes the congestion impairment by jointly optimizing above two essential factors.
- Due to the NP-hardness of the solution to the minimization problem, a two-phase heuristic algorithm is proposed. The first phase optimizes the congestion duration and the second phase sets optimal rate limitation.
- Simulation results show that the heuristic algorithm can reduce packet loss ratio by up to 84%, compared with existing link-based solutions. Meanwhile, this algorithm can protect important flows from losing packets.

The remainder of this paper is organized as follows. Section II describes the related work. Section III presents the motivation and update problem formulation. The two-phase heuristic algorithm is presented in Section IV. Simulation results are shown in Section V, and Section VI concludes this paper.

II. Related Work

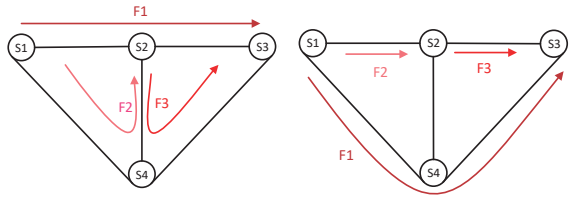
To avoid the temporary congestion during the update process, early efforts were dedicated to design congestion-free update algorithms. Hong *et al.* prove that if all links leave free capacity slack s, the network can be updated by $\lceil s-1 \rceil$ migration steps without congestion [6]. Liu *et al.* propose to find a series of middle stages so that there is no congestion when flows are migrated from one stage to the next [5]. Jin *et al.* introduce a dynamic network update schedule by constructing dependency graph where flows are migrated to target paths directly [12]. For splittable flows, the first polynomial-time algorithm deciding whether a congestion-free update scheme is possible has been proposed in [13]. The authors also prove that it is NP-hard to decide if all flows are unsplittable. Then, for un-splittable flows, Foerster gives the upper and lower complexity bounds for congestion-free update [14]. Other works were also dedicated to minimize the overloaded data volume ratio of links [10][11]. Most of these works target on *congestion-free* update, and the scenario where transient congestion is unavoidable is regarded as a special case. So those schemes are designed on a link basis, where the congestion impairment is not fully revealed.

A similar research direction is the deadlock break problem (DBP), which has been rarely studied in SDN [15]. Directly updating network from source state to target state will probably end up in a deadlock state, and DBP seeks to limit several flows' rate with the lowest dropped packets in order to break the deadlock and hence recover the update process. In this paper, we consider that flows can be migrated to intermediate paths and they can also be moved back and forth repeatedly. What's more, we focus on the impairment of congestion which jointly measures flows' importance and packet loss.

III. Motivation and Problem Formulation

A. Link-based Network Update

There are many scenarios where the congestion is unavoidable. In these scenarios, update schemes still have great



(a) Source State (b) Target State

Fig. 1. An example that transient congestion is unavoidable due to link capacity shortage.

difference in rate limitation and congestion duration, thus packet loss, even that they all get the lowest overloaded volume ratio. On the contrary, these two factors can be optimized on a flow basis, as shown by the following two examples.

1) Rate Limitation: Consider the example in Figure 1. There are four switches S_1, S_2, S_3 and S_4. The bandwidth capacity of each link is 10 units. F_1, F_2 and F_3 are three un-splittable flows of size 6 units. The source routing is illustrated in Fig. 1(a), which is in-efficient in terms of link utilization. The SDN controller then decides the target routing strategy as shown by Fig. 1(b). Since the switches cannot be updated at the same time, congestion-free update scheme is unavailable.

When the network is migrated from Fig. 1(a) to Fig. 1(b), the overloaded volume ratio of link (S_1, S_2), (S_2, S_3), (S_1, S_4) and (S_4, S_3) is 0.2. The three flows have to lose packets and with the different rate limitation, schemes are great different in packet loss. Generally, the higher volume of dropped packets F_2 has, the less packets F_1 will lose, and F_1 will compete for more bandwidth of link (S_2, S_3) or (S_4, S_3) with F_3, thus F_3 will lose more packets. The worst scheme is that both F_2 and F_3 lose 2 units, and F_1 does not lose any packets. As a result, there are 4 units lost in total. The best scheme is that only F_1 lose 2 units volume and neither F_2 nor F_3 will lose packets. So it is better to limit the size of key flow F_1 by 2 units proactively. This best scheme can be find from a flow-based perspective, which is outside the consideration of link-based algorithms.

2) Congestion Duration: The speed of network update is important as how quickly the workloads are changed directly determines the effectiveness of the update task [11]. Hence, there is always a time threshold that the network update duration cannot exceed. The time cost of each migration are different in different scenarios. Without loss of generality, for stage-based algorithms, the maximum number of intermediate stages is used as the measurement of update duration [10].

Figure 2 shows an example where the congestion is unavoidable because of the time constraint, where Fig. 2(a) is the source state and Fig. 2(d) is the target state. The constraints of flows and links are the same as that of Fig. 1. Without the time constraint, there is a congestion-free scheme: [Fig. 2(a) $\rightarrow$ Fig. 2(b) $\rightarrow$ Fig. 2(c) $\rightarrow$ Fig. 2 (d)], which needs 2 intermediate stages. With the time constraint that restricts the number of intermediate stages to 1, the congestion-free scheme is unavailable. There are three schemes with the same

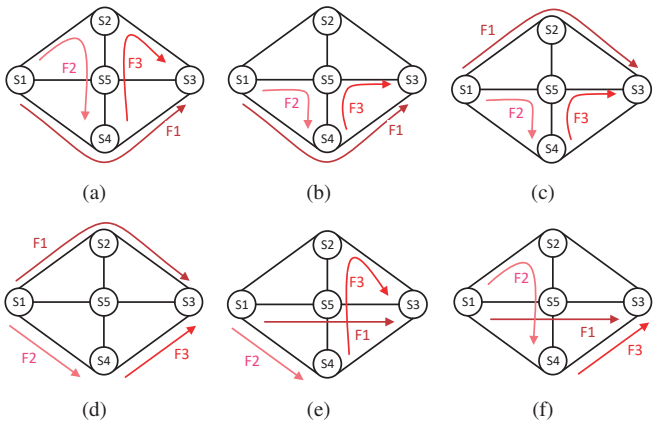


Fig. 2. An example that congestion is unavoidable for time limitation

link overloaded volume: [Fig. 2(a) → Fig. 2(e) → Fig. 2(d)], [Fig. 2(a) → Fig. 2(f) → Fig. 2(d)] and [Fig. 2(a) → Fig. 2(d)]. From the link perspective, the update scheme [Fig. 2(a) → Fig. 2(d)] is identical to the rest two solutions as their overloaded volume ratio is the same. But from the flow perspective, this scheme is significant. Only if the key flow F_1 is limited by 2 units, all the rest flows will not be impaired. As a result, this scheme leads to less packet loss because of shorter congestion duration.

B. Flow-based Update Problem Formulation

Before formulating the problem of jointly accommodating flows' rate limitation and congestion duration by a flow-based way, we firstly present the network model. We consider a SDN that is featured by a directed graph $G = (\mathcal{N}, \mathcal{E})$, where $\mathcal{N}$ is the set of switches and $\mathcal{E}$ is the set of links. $\forall e \in \mathcal{E}$, B_e is the bandwidth capacity of the link. S is the maximum number of intermediate stages related to the time limitation, and the set $\mathcal{S} = \{0, 1, 2, \ldots, S\}$ contains the index of the source state and intermediate stages. The source state is presented by index 0 and the target state is denoted by $S+1$. $\mathcal{F}$ is the set of flows in the network. $\forall f \in \mathcal{F}$, $\mathcal{P}_f$ is the set of all possible loop-free paths of flow f, which is computed by the SDN controller. W_f is the weight of flow f and the bandwidth demand of flow f is denoted by D_f.

Flow-Based Update (FBU) Problem: Suppose the parameters S, W_f, $\mathcal{P}_f$, D_f, source state and target state are known ahead. The importance weight of each flow is determined by the application priority, which is beyond the scope of this paper. The objective of the FBU problem is to choose a feasible path in set $\mathcal{P}_f$ at each intermediate stage for each flow f, and then select key flows that will be limited by a certain partition during each migration stage. The aim of these operations is to minimize the congestion impairment, measured as the weighted sum $\sum_{s=0}^{S}\sum_{f\in\mathcal{F}} W_f l_{s,f}$.

Based on the above definition, the FBU problem can be formulated as a nonlinear optimization program as problem (1), where $l_{s,f}$, $b^e_{s,f}$ and $a^p_{s,f}$ are decision variables. $l_{s,f}$ represents the limited volume of flow f when network is

$$\min \quad \sum_{s=0}^{S}\sum_{f\in\mathcal{F}} W_f l_{s,f}$$

$$\text{s.t.} \quad \sum_{f\in\mathcal{F}} D_f b^e_{s,f} - \sum_{f\in F} l_{s,f} b^e_{s,f} \le B_e, \forall s \in \mathcal{S}, \forall e \in \mathcal{E} \quad (1.1)$$

$$b^e_{s,f} = \max\Big(\sum_{p\in\mathcal{P}_f: e\in p} a^p_{s,f}, \sum_{p\in\mathcal{P}_f: e\in p} a^p_{s+1,f}\Big),$$
$$\forall s \in \mathcal{S}, \forall e \in \mathcal{E}, \forall f \in \mathcal{F} \quad (1.2)$$

$$\sum_{p\in\mathcal{P}_f} a^p_{s,f} = 1, \forall s \in \mathcal{S}\backslash\{0\}, \forall f \in \mathcal{F} \quad (1.3)$$

$$a^p_{s,f} = \{0, 1\}, \forall s \in \mathcal{S}\backslash\{0\}, \forall f \in \mathcal{F}, \forall p \in \mathcal{P}_f \quad (1.4)$$

$$l_{s,f} \le D_f, \forall s \in \mathcal{S}, \forall f \in \mathcal{F} \quad (1.5)$$

updated from stage s to $s+1$. $b^e_{s,f}$ indicates whether or not flow f passes edge e during the migration from stage s to $s+1$. When $a^p_{s,f} = 1$, it indicates that flow f chooses path p in stage s. Constraint (1.1) and (1.2) guarantee that there is no congestion. Constraint (1.3) is the flow demand constraint and (1.4) represents that flows are un-splittable. Constraint (1.5) indicates that the limit portion of each flow is less than the flow size. Because (1) can be induced to the same NP-hard optimization problem in [16], it is also NP-hard.

IV. Heuristic Algorithm

As the FBU problem is NP-hard, a two-phase heuristic algorithm is proposed in this section. Then the complexity of this algorithm is analyzed.

A. Two-phase Algorithm

Based on the analysis in Section III-A, the congestion duration and rate limitation are essential factors that determines packet loss. Hence, we propose a heuristic algorithm that solves the NP-hard problem by two phases, which address the congestion duration and rate limitation respectively.

1) Update Schedule: The first phase aims at obtaining the update schedule by jointly optimizing the congestion duration and the overloaded volume ratio. Because $b^e_{s,f}$ is no bigger than 1, optimization problem (1) can be relaxed to MIP (2). The objective function becomes the total overloaded volume of all migration steps, taking into account the congestion duration and the overloaded volume. $\sum_{f\in\mathcal{F}} l_{s,f}$ is the transient variable, representing the instant overloaded volume.

$$\min \quad \sum_{s=0}^{S}\sum_{f\in\mathcal{F}} W_f l_{s,f}$$

$$\text{s.t.} \quad \sum_{f\in\mathcal{F}} D_f b^e_{s,f} - \sum_{f\in\mathcal{F}} l_{s,f} \le B_e, \forall s \in \mathcal{S}, \forall e \in \mathcal{E} \quad (2.1)$$
$$(1.1)(1.2)(1.3)(1.4)$$

Because constraint (1.2) is nonlinear, a set of auxiliary variables $\{h^e_{s,f}\}$ are introduced. The constraint (2.1) and (1.2) are transformed to (3.1), (3.2) and (3.3). Constraint (1.4) is relaxed to (3.5), leading to the optimization problem (3) which can be solved by linear programming (LP). $h^e_{s,f}$, $l_{s,f}$ and $a^p_{s,f}$ are variables. The auxiliary variable $h^e_{s,f}$ presents the

maximum fraction of flow f in link e during the migration from stage s to $s+1$. For each s^* and f^*, the set $\{a^p_{s^*,f^*}\}$ can be obtained by solving the LP (3). Path p^* is chosen as the path of flow f^* in stage s^* when the $a^{p^*}_{s^*,f^*}$ is the maximum item of the set $\{a^p_{s^*,f^*}\}$.

$$\min \quad \sum_{s=0}^{S} \sum_{f \in \mathcal{F}} W_f l_{s,f}$$

$$\text{s.t.} \quad \sum_{f \in \mathcal{F}} D_f h^e_{s,f} - \sum_{f \in \mathcal{F}} l_{s,f} \leq B_e, \forall s \in \mathcal{S}, \forall e \in \mathcal{E} \quad (3.1)$$

$$\sum_{p \in \mathcal{P}_f : e \in p} a^p_{s,f} \leq h^e_{s,f}, \forall s \in \mathcal{S}, \forall f \in \mathcal{F}, \forall e \in \mathcal{E} \quad (3.2)$$

$$\sum_{p \in \mathcal{P}_f : e \in p} a^p_{s+1,f} \leq h^e_{s,f}, \forall s \in \mathcal{S}, \forall f \in \mathcal{F}, \forall e \in \mathcal{E} \quad (3.3)$$

$$\sum_{p \in \mathcal{P}_f} a^e_{s,f} = 1, \forall s \in \mathcal{S} \backslash \{0\}, \forall f \in \mathcal{F} \quad (3.4)$$

$$x^p_{s,f} \geq 0, \forall s \in \mathcal{S} \backslash \{0\}, \forall f \in \mathcal{F}, \forall p \in \mathcal{P}_f \quad (3.5)$$

$$l_{s,f} \leq D_f, \forall s \in \mathcal{S}, \forall f \in \mathcal{F} \quad (3.6)$$

2) Rate Limitation: The second phase is limiting key flows by certain volume during each migration of the update schedule. Once the update schedule is determined, rate limitation is necessary as long as there is congestion during the migration process of from stage s to $s+1$. Given the flow demand D_f, importance weight W_f, source routing and target routing, it is required to find optimal rate limitation scheme where the weighted sum $\sum_{f \in \mathcal{F}} W_f v_f$ is the highest (v_f is the allocated bandwidth for flow f).

This problem can be formulated as the LP (4). $K_{f,e}$ can only be 1 or 0. As long as flow f passes through edge e at source state or target state, $K_{f,e}$ is 1. Constraint (4.1) represents there is no congestion and constraint (4.2) indicates that the allocated bandwidth of each flow is not higher than the demand value. The proposed heuristic algorithm `MIC` is sketched in Algorithm 1.

$$\max \quad \sum_{s=0}^{S} \sum_{f \in \mathcal{F}} W_f v_f$$

$$\text{s.t.} \quad \sum_{f \in \mathcal{F}} v_f K_{f,e} \leq B_e, \forall e \in \mathcal{E} \quad (4.1)$$

$$v_f \leq D_f, \forall f \in \mathcal{F} \quad (4.2)$$

Algorithm 1 `MIC`: Minimizing the Impairment of Congestion

Input: The source and target state; S, W_f, D_f, $\mathcal{P}_f$;
Output: The routing at each intermediate stage; rate limitation for flows during each migration process;

1: Get solutions $\{a^p_{s,f}\}$ from the LP (3)
2: **for** $s^* = 1$ to S **do**
3: **for** each f^* in $\mathcal{F}$ **do**
4: Choose path p^* when the $a^{p^*}_{s^*,f^*}$ is the maximum item of set $\{a^p_{s*,f^*}\}$
5: Flow f^* is routed through path p^* at stage s^*
6: **end for**
7: **end for**
8: Set the source state as the stage 0
9: Set the target state as the stage $S+1$
10: **for** $s^* = 0$ to S **do**
11: **for** each f^* in $\mathcal{F}$ **do**
12: For LP(4), set the path of flow f^* at stage s^* as the source path of flow f^*
13: For LP(4), set the path of flow f^* at stage s^*+1 as the target path of flow f^*
14: **end for**
15: Solve LP (4) and the result is the rate limitation during the migration from stage s to $s+1$
16: **end for**

B. Complexity Analysis

For each update process, `MIC` needs to solve LP (3) once and LP (4) $S+1$ times. The LP (3) has $O(S*T+S*|\mathcal{F}|*|\mathcal{E}|)$ constraints and variables, where T is equal to $\sum_{f \in \mathcal{F}} |\mathcal{P}_f|$. The LP (4) only has $|\mathcal{E}| + |\mathcal{F}|$ constraints and $|\mathcal{F}|$ variables. Let M denotes $S*T+S*|\mathcal{F}|*|\mathcal{E}|$. The worst-case complexity of LP (3) is $O(M^3)$ by Karmarkar's algorithm [17]. Note that the complexity of line 2 $\sim$ 7 is $O(S*T)$, the whole complexity of `MIC` is $O(M^3)$.

V. Performance Evaluation

In this paper, we focus on the better performance (less packet loss and Qos) of the flow-based solution than link-based solution. For simplicity, we conduct simulation based on Microsoft's inter-datacenter WAN topology (Figure 3) which is widely used in related research area [10][12]. There are 8 switches and 14 links and the bandwidth capacity of each link is 60 Mbps. Five flows are simultaneously generated with the bandwidth demand of 25 Mbps and each of them sustains for 110 seconds. The source path, target path and importance weight of each flow are shown in Table I. Note that the update task will end up in a deadlock state, the congestion is unavoidable no matter how many intermediate stages are allowed.

The performance of packet loss during the update process is measured on platform Mininet 2.0[1], switches are configured to run OpenFlow v1.3[2]. The network is controlled by a RYU[3] based controller, and switch states are installed and updated via RYU REST API. Rate limitation is implemented by TC tools[4]. We use the *iperf3* to generate flows and measure the packet loss ratio of the whole update duration. The congestion during the update process is transient and hence the *shared bandwidth technology* of HTB (Hierarchy Token Bucket) is introduced to implement flexible rate limitation. Because the bandwidth is shared by all the flows, rate limitation only be triggered once the link is overloaded and key flows are not limited from the beginning to the end of the update process.

We compare the proposed `MIC` algorithm with the following

[1] http://mininet.org/
[2] https://www.opennetworking.org/images/stories/downloads/sdn-resources/onf-specifications/openflow/openflow-spec-v1.3.0.pdf
[3] https://osrg.github.io/ryu/
[4] http://tldp.org/HOWTO/Traffic-Control-HOWTO/intro.html

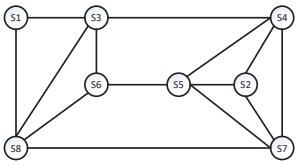


Fig. 3. Inter-data center topology of Microsoft

TABLE I
SOURCE AND TARGET PATHES OF FLOWS WITH IMPORTANCE WEIGHT

Flow	Source Path	Target Path	Weight
F_1	$S1 \rightarrow S3 \rightarrow S4 \rightarrow S2$	$S1 \rightarrow S8 \rightarrow S7 \rightarrow S2$	2
F_2	$S1 \rightarrow S3 \rightarrow S4 \rightarrow S5$	$S1 \rightarrow S8 \rightarrow S7 \rightarrow S5$	10
F_3	$S1 \rightarrow S8 \rightarrow S6$	$S1 \rightarrow S3 \rightarrow S6$	10
F_4	$S3 \rightarrow S8 \rightarrow S7$	$S3 \rightarrow S4 \rightarrow S7$	2
F_5	$S8 \rightarrow S7 \rightarrow S4$	$S8 \rightarrow S3 \rightarrow S4$	10

benchmark algorithms: RR: A link-based update algorithm proposed in [10], GI: A link-based greedy improvement based on RR aiming at reducing the maximum overloaded volume ratio of links [10], OPT: the optimal update scheme by exhaustive method. The OPT is used to measure the influence of system dynamics. As the update operations of different flows during a migration step are asynchronous, the result of each simulation can be different. Meanwhile, GI and RR are both influenced by algorithm randomness. Hence, each algorithm is performed 10 times and the average results are shown in our experiments.

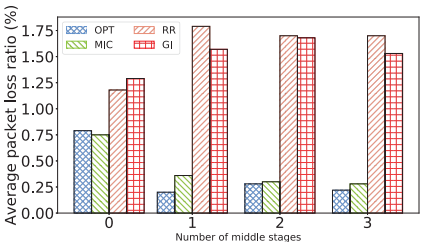
A. The Case of Weight-identical Flow

Firstly, we consider the case where all flows have the same weight so the congestion impairment is directly measured by the packet loss ratio. The packet loss ratio of the whole network and each flow are evaluated.

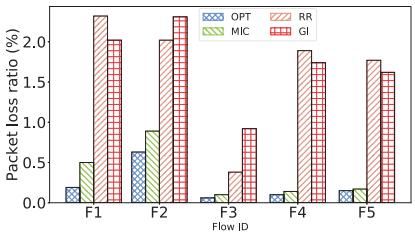
Figure 4(a) shows each algorithm's whole performance of packet loss ratio with different maximum number of intermediate stages. The x-axis represents the number of intermediate stages S, from 0 to 3. The y-axis represents the average packet loss ratio of the five flows. Although there is no intermediate stage when $S = 0$, MIC can limit rate based on the source and target state to reduce the impairment of congestion.

As shown in Figure 4(a), MIC has almost the same performance as OPT when $S = 0, 2, 3$. Although when $S = 1$, the packet loss ratio of MIC is higher than that of OPT, MIC can still reduce the ratio by 80% compared with RR. When $S = 0$, MIC reduces the ratio by 36% compared with RR, and when S is 2 and 3, the gain of MIC increased by 84%.

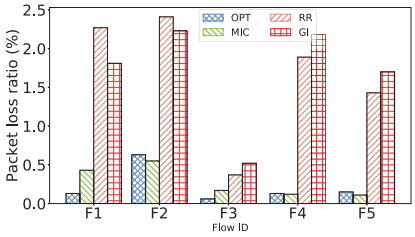
When $S = 0$, the drop ratio of RR and GI is much less than the case of $S = 1, 2, 3$. It proves our argument in Section I that link-based update algorithms do not take congestion duration into consideration. Although the overloaded volume ratio can be reduced with the increase of intermediate stages, the congestion duration is extended. On the contrary, MIC, a flow-based update algorithm, results in better schemes that make full use of intermediate stages and take into account the congestion duration. When $S = 1, 2, 3$, the packet loss ratio of MIC is less than half of the case when $S = 0$. It is important



(a) Average packet loss ratio of 5 flows



(b) Packet loss ratio of each flow when $S = 1$



(c) Packet loss ratio of each flow when $S = 3$

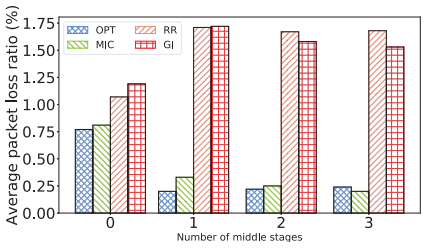
Fig. 4. Performance of four algorithms without importance difference

to point out that the GI does not necessarily perform better than RR. GI reduces the the maximum overloaded volume ratio, which is not closely related to the rate limitation and congestion duration, so the packet loss ratio is not reduced. There may be less overloaded ratio of links in schemes designed by GI, but more flows may be affected or the congestion duration may be longer.

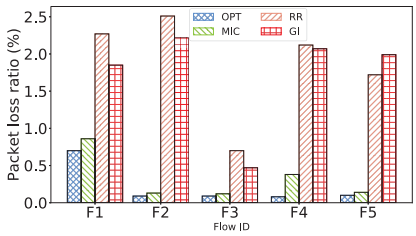
Because of the space limitation, we only show the packet loss ratio of each flow when $S = 1$ (Fig 4(b)) and $S = 3$ (Fig 4(c)). As these two figures present, the optimal scheme just needs to limit F_2. Both RR and GI influence all five flows. MIC has better performance than both RR and GI, but it is still not optimal. Comparing each flow's packet loss ratio of MIC with OPT, it is clear that MIC choose flow F_1 and F_2 to limit when $S = 1$ and limit flow F_1 , F_2 and F_3 when $S = 3$.

B. The Case of Weight-differentiated Flow

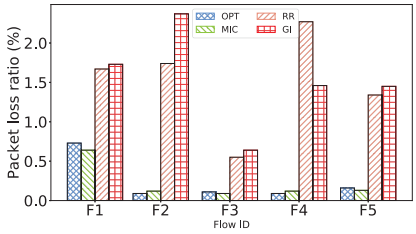
We evaluate the performance of those algorithms when each flow has the importance weight shown in Table I. The rate limitation of each flow presents those algorithms' ability to protect important flows from losing packets. Fig 5(a) gives the overall performance comparison of each algorithm. Fig 5(b) presents the details of rate limitation when $S = 1$ and the Fig 5(c) shows that when $S = 3$.



(a) Average packet loss ratio of 5 flows



(b) Packet loss ratio of each flow when $S = 1$



(c) Packet loss ratio of each flow when $S = 3$

Fig. 5. Performance of 4 algorithms with importance difference

As F_1 is less important than F_2, the optimal scheme chooses F_1 to limit (shown in Fig 5(b) and Fig 5(c)) instead of F_2 (shown in Fig 4(b) and Fig 4(c)). When $S = 1$, `MIC` also fully considers the importance weight although it cannot get the optimal performance compared with `OPT`. `MIC` chooses F_4 to limit instead of F_2, because F_4 is less important. When $S = 3$, `MIC` has the same performance as `OPT`. It only limits flow F_1 and protects flow F_2 and F_3 from losing packets.

Interestingly, the packet loss performance shown in Fig 5(a) is similar with Fig 4(a) although the importance factor is introduced. It is obvious for `RR` and `GI` because these two algorithms does not accommodate the weight difference of different flows. When flow priority is not considered, `MIC` and `OPT` actually generate multiple update schemes with the same packet loss ratio and some of these schemes are also optimal when the importance factor is introduced.

VI. Conclusions

In SDN, there are certain scenarios where the congestion is unavoidable during the update process. In this paper, we prove that schemes with the same overloaded volume can still have great difference in rate limitation and congestion duration, thus packet loss ratio. Unlike existing link-based solutions, we resort to the flow-based strategy, which jointly optimizes the rate limitation of flows and congestion duration. This strategy is formulated as an NP-hard optimization program. Then, we proposed a two-phase heuristic algorithm and simulation results show that this flow-based update algorithm not only reduces the packet loss ratio compared with link-based update algorithms, but also protects important flows from dropping packets, so it minimizes the congestion impairment. In the future, we will investigate dynamic flow-based network update strategy based on dependency graph where flows can be moved back and forth repeatedly.

Acknowledgement

This work was supported by the National Natural Science Foundation of China (NSFC) (No. 61871437).

References

[1] A. R. Curtis, J. C. Mogul, J. Tourrilhes, P. Yalagandula, P. Sharma, and S. Banerjee, "Devoflow:scaling flow management for high-performance networks," in *Proc. of ACM SIGCOMM*, Toronto, Canada, 2011, pp. 254–265.

[2] M. Reitblatt, N. Foster, J. Rexford, C. Schlesinger, and D. Walker, "Abstractions for network update," in *Proc. of ACM SIGCOMM*, New York, NY, USA,2012, pp. 323–334.

[3] P. Yang, N. Zhang, Y. Bi, L. Yu, and X. S. Shen, "Catalyzing cloud-fog interoperation in 5g wireless networks: An sdn approach," *IEEE Network*, vol. 31, no. 5, pp. 14–20, 2017.

[4] S. Jain, A. Kumar, S. Mandal, J. Ong, L. Poutievski, A. Singh, S. Venkata, J. Wanderer, J. Zhou, and M. Zhu, "B4: experience with a globally-deployed software defined wan," in *Proc. of ACM SIGCOMM*, Hong Kong, China, 2013, pp. 3–14.

[5] H. H. Liu, X. Wu, M. Zhang, L. Yuan, R. Wattenhofer, and D. Maltz, "zupdate: Updating data center networks with zero loss," in *Proc. of ACM SIGCOMM*, Hong Kong, China, 2013, pp. 411–422.

[6] C.-Y. Hong, S. Kandula, R. Mahajan, M. Zhang, V. Gill, M. Nanduri, and R. Wattenhofer, "Achieving high utilization with software-driven wan," in *Proc. of ACM SIGCOMM*, Hong Kong, China, 2013, pp. 15–26.

[7] B. G. Jozsa and M. Makai, "On the solution of reroute sequence planning problem in mpls networks," *Computer Networks*, vol. 42, no. 2, pp. 199–210, 2003.

[8] O. Klopfenstein, "Rerouting tunnels for mpls network resource optimization," *European Journal of Operational Research*, vol. 188, no. 1, pp. 293–312, 2008.

[9] K.-T. Foerster, S. Schmid, and S. Vissicchio, "Survey of consistent network updates," *arXiv preprint arXiv:1609.02305*, pp. 1–24, 2016.

[10] J. Zheng, H. Xu, G. Chen, and H. Dai, "Minimizing transient congestion during network update in data centers," in *Proc. of IEEE ICNP*, San Francisco, CA, USA, 2015, pp. 1–10.

[11] H. Xu, Z. Yu, X.-Y. Li, C. Qian, L. Huang, and T. Jung, "Real-time update with joint optimization of route selection and update scheduling for sdns," in *Proc. of IEEE ICNP*, Singapore, 2016, pp. 1–10.

[12] X. Jin, H. H. Liu, R. Gandhi, S. Kandula, R. Mahajan, M. Zhang, J. Rexford, and R. Wattenhofer, "Dynamic scheduling of network updates," in *Proc. of ACM SIGCOMM*, Chicago, Illinois, USA, 2014, pp. 539–550.

[13] S. Brandt, K.-T. Förster, and R. Wattenhofer, "On consistent migration of flows in sdns," in *Proc. of IEEE INFOCOM*, San Francisco, CA, USA, 2016, pp. 1–9.

[14] K.-T. Foerster, "On the consistent migration of unsplittable flows: Upper and lower complexity bounds," in *Proc. of IEEE Network Computing and Applications*, Cambridge, MA, USA, 2017, pp. 1–4.

[15] L. Maggi, P.-L. Poirion, and J. Leguay, "Reroute backward to better break deadlocks," in *Proc. of IEEE Cloud Networking*, Prague, Czech Republic, 2017, pp. 1–6.

[16] B. G. Jozsa, "Reroute sequence planning for protected traffic flows in gmpls networks," in *Proc. of IEEE ICC*, New York, NY, USA, USA, 2002, pp. 2702–2706.

[17] N. Karmarkar, "A new polynomial-time algorithm for linear programming," in *Proc. of ACM STOC*, New York, NY, USA, 1984, pp. 302–311.