Proceedings of 10th IOE Graduate Conference
Peer Reviewed
ISSN: 2350-8914 (Online), 2350-8906 (Print)
Year: 2021 Month: October Volume: 10

Congestion Minimization in SDN using Two-phase Heuristic Algorithm

Tirtha Raj Adhikari [a], Shashidhar Ram Joshi [b]

[a, b] *Department of Electronics and Computer Engineering, IOE, Tribhuvan University, Nepal*
Corresponding Email: [a] tirthaadhikari93@gmail.com , [b] srjoshi@ioe.edu.np

Abstract
Network update is unavoidable in any legacy as well as SDN frameworks. Undesirable and operational events like hardware maintenance, link failure, node failure etc trigger network updates. During the update process although initial and final state may remain consistent, different intermediate links might suffer from inconsistencies. Number of researches have been successful to prevent inconsistencies which are congestion free but time consuming. Network updates are frequent and need to be handled in a timely manner allowing minimum amount of congestion.
We propose Two-phase Heuristic Algorithm in a case of SDN network update mechanism to allow minimum amount of congestion to get fast network update and achieve maximum utilization of link resources. Main technique of this particular research is firstly to find the key flow in the network at different flow paths routed from source,intermediate stages and destinations and to rate limit the key flow allowing maximum flow in the link to attain its maximum capacity. The network is simulated in Mininet emulator and the flow rules were modified in the forwarding devices by RYU controller using OpenFlow protocol.The Abilene topology having 11 nodes and 14 links is used. The results showed that if key flow in the network is rate limited then other flows will have to loose lesser packets resulting low packet loss in total. If we insert optimum number of intermediate stages during network update from initial to final stage, the amount of packet loss decreases. Also, our algorithm is used to the case such that important flows can be preserved to some extent.

Keywords
SDN, Network Update, Flow Based, Two-phase Heuristic, Abilene, Congestion Minimization

1. Introduction

Network performance is determined by computational resources and bandwidth, which are frequently constrained by high operational costs, frequent node congestion, and uneven traffic loads. In any SDN implemented network, as a result of various events like reconfiguration, need of switch update, link failure, traffic variations, hardware maintenance, modifying ACL etc. there exist requirement of network update often. [1]. This modification of network states is termed as network update. Switch updates takes place individually and asynchronously, ensuing serious congestion problems. SDN is a novel networking architecture that allows for easier network management, faster network updates, and better resource allocation schemes by abstracting the control plane of SDN from its data plane. To lower network congestion, network controllers use a preventive strategy to make centralized decisions including managing flow tables and resource utilization [2]. Computer networks are no longer limited to LANs and WANs, due to advances in information and technology. Data-center networks, need to meet stringent standards for correctness, availability, and performance, while they must also be highly flexible, allowing for quick modifications in scenarios such as new policies, network problems, or increased traffic. In the network area, the current problem is to achieve update of network in such instances in a timely and consistent way. However, transient congestion and packet loss are inevitable during the network update process because of resource and time constraints. Datacenter network require frequent network adjustments. SDN being a modern technique to network system administration, configuration, and execution, the development of updating mechanisms with low congestion and little packet loss is a popular research topic these days.

2. Related Work

To limit the transient congestion in SDN, one may have to consider many definite properties, e.g., interface limit and calculation times. Primary goal of congestion minimization is to avoid the bandwidth violation. No matter whether the flow is utilizing the rules before or after the update process, the total amount of flow size must follow capacity constraint of particular link. The paper [3] analyses the physical design problems during internet backbone network construction. A problem is formulated using cost, performance and the actual survival and also considering important design factors.

Due to dynamic nature of SDN, network updates are more frequent and crucial in SDN architecture than in traditional networks. During the network update, the controller must meet the performance, security, and packet-processing tasks. Since, the controller and forwarding plane are separated in SDN, open-flow devices react slowly to new rules. Such conditions in network update leads to inconsistency.The important work of Mizrahi and Moses [4] demonstrates that flow swapping is required for network optimization, and techniques are required to preserve consistency during network update in SDN.This concept underpins emerging SDN approaches for reducing transient congestion. Contemporary algorithmic approaches use slack capacity or dependency graphs to explain partial moves.

Reiblatt et al. [5], in their foundational work, present two abstractions for updating : per-packet and per-flow consistency. In two-phase commit, flows are identified with new and old rules, posing a difficulty for flow ordering. It ensures that the flow is handled by the old configuration prior to the update or updated new configuration, not both. It also lowers the likelihood of loops during updates. Amiri et al. [6] introduced a congestion-free reconfiguration strategy for flows of a specific demand in the network from the current path to the relevant final path. Changing the forwarding rules at the nodes allows for reconfiguration from the current path to the new path while keeping the attributes. It presents a congestion-free updating plan for unsplittable flows at the trade-off of increased complexity. The approach of SWAN [7] makes extensive use of network capacity, even when traffic volume varies drastically. SWAN leaves 'scratch capacity' on links that will not be used until the time of update. Likewise, zUpdate [2] tries to reduce congestion in switches by using Equal-Cost Multi-path Routing (ECMP) to uniformly distribute traffic among all next hops, allowing redundant paths to be fully utilized. The only requirement for zUpdate is that the operator provide the final configuration without focusing to the update details.SWAN [7] and zUpdate [2] propose finding a congestion-free updating scheme in network updates. The update plan is splited into different phases, each of which requires altering flow tables on a set of switches, with the feature that no congestion will occur regardless of the sequence or time of the updates.It is impossible to update the network with zero congestion if all of the links are filled. With the premise that all links have free capacity (scratch capacity), congestion-free updates are attainable in [1/s]-1 updates. Scratch capacity might range from 0% to 50 %. For example, if all links have 10% free capacity, the congestion-free updates will require 9 updates. Each update step includes a set of changes to forwarding rules of switches, with the quality that no congestion will occur irrespective of the sequence or time of the updates. In addition, rather than wasting scratch capacity, it could be used for background traffic. It assures that non-background traffic is not interrupted during transactions, and that background traffic congestion is bounded. Ultimately, the link capacity is being used well. Hence, in the cost of the volume of scratch capacity, a congestion-free update is possible. Increased scratch capacity leads to faster network upgrades since updates are done in fewer steps. However, this lowers the amount of non-background traffic in the network. Also if the background traffic demand is low, capacity is wasted. Jin [8] and Mahajan [9] pioneered the concept of dynamic and dependency graphs in their study. According to various run time conditions of switches in the network, dynamic and dependency graphs are utilized to find a fast congestion-free update plan. Individual graph update dependencies are built, with updates being sent out greedily once the corresponding preconditions are met. Flows are rate-limited to ensure progress when this greedy traversal of the dependency graph results in a deadlock. Jin et al [8], in their research work, propose a fast and consistent network update scenario in SDN based on real-time network and switch characteristics. It depicts a consistency-related dependency graph of nodes and dependencies, where nodes represent rule updates and network resources, and edges reflect dependencies between them. Then, relying on the run-time variations in the update speeds of different switches, it proactively schedules these

updates to ensure consistent updates. Mahajan [9] proposes two network methods for consistency and speed plan in his research. A consistency plan is a directed acyclic graph with nodes representing rule updates and edges representing dependencies. The speed plan is determined by the amount of time it takes for individual switches to implement updates and the distance between the controller and the switches. The SDN platform's primary goal is to optimize the available resources. However, there is inefficient utilization of available link capacity in the network update strategy using scratch capacity.Gandhi [10] explains that flows have a number of migration choices. The paper calculates many options and optimizes the dependency network for the optimum path and a few intermediate steps. The seminal work by Brandt et al [11] gives us a new idea that new flow pathways should not be part of the problem input, but should instead be computed in conjunction with the migration plan. This concept leads to a faster migration approach and the resolution of more problems.These studies concentrate on maximizing link capacity utilization, but they do not ensure the availability of consistent updates, the optimum update scheme, or the time required for network updates.

Zheng et al [12] proposed a congestion-minimizing network update in their pioneering paper. Congestion-free updates have the drawback of not being able to fully utilize network capacity and is time-consuming because they require a series of LP solutions.During network updates, it introduces two new concepts: the bounded congestion update problem (BCUP) and the minimum congestion update problem (MCUP). MCUP seeks to identify the routing for all intermediate stages such that transient congestion is reduced, while BCUP aims to determine the least number of intermediate stages.They have simulated and analysed rounding algorithm, greedy algorithm and heuristic algorithm on 8-pod fat-tree topology.

The important research work done by Hertiana et al [13] explored at a flow-based routing system in abilene topology which provide a path based on bandwidth demand. The purpose of this work is to provide acceptable QoS while also avoiding network congestion.The goal of this technique is to discover a good path by choosing one of the paths with the most residual bandwidth. This flow based routing is compared with OSPF routing.

The pioneer research of Zheng et al [14] prove the hardness of the minimal congestion update problem (MCUP) and analyse its hardness.They have used two topologies: fat-tree DCN topology and Microsoft's inter-datacenter WAN topology. To discover the update sequence, they offer an approximation technique and a greedy improvement algorithm.

In the seminal work of Wen et al [1] flow based network update and minimization of congestion impairment is discussed and compared with existing link based update strategies. The network update problem is shown as NP-hard and two-phase heuristic algorithm is proposed, formulated and implemented on Microsoft's inter-datacenter WAN topology.

3. Methodology

3.1 Network Update Problem

A directed graph $G = (N, E)$ can be used to describe any network, in which N symbolizes the set of switches and E denotes the set of edges/links. The bandwidth capacity of the links is represented by B_c. F represents set of flows, for all f belongs to F, P_f denotes the set of all possible loop-free paths of flow f, W_f denotes the weight of the flow f and B_f represents bandwidth demand of flow f. M is assumed as the maximum number of source state, target state and intermediate stages. In the set, $M = \{0, 1, 2, 3, ..., m, m+1\}$, the initial stage is represented by index 0, the final stage is represented by index $m+1$ and subset $\{1, 2, 3..., m\}$ represents the intermediate stages.

Notation	Description
N	The set of switches
E	The set of edges
G	The directed network graph G = *(N,E)*
B_c	Bandwidth capacity of link
M	Maximum number of intermediate stages
F	The set of flows
Pf	The set of all possible loop free paths of flow f
W_f	Weight of flow f
B_f	Bandwidth demand of flow f
$l_{m,f}$	Limited volume of flow f when network is updated from stage m to $m+1$
$b^e_{m,f}$	Indicator, whether or not flow f passes edge e during the migration from stage m to $m+1$
$\alpha^p_{m,f}$	Indicator, whether flow f chooses path p in stage m or not
$lr^e_{m,f}$	Represents maximum fraction of flow f in link e during migration from stage m to $m+1$

Figure 1: Key notations of the network model

Flow conservation, demand satisfaction, and capacity constraints must all be considered when designing a network model as a directed graph. The following conditions must be met for the network model mentioned above:

$$\sum_{e \in out(m)} F(e) = \sum_{e \in in(m)} F(e), \forall m \in M0, m+1 \quad (1)$$

$$\sum_{e \in E} F(e) = B_f, \forall e \in E$$

$$F(e) \leq B_c, \forall e \in E$$

Furthermore, to avoid policy inconsistency, the two-phase commit protocol provided in [5] is effective for retaining packet coherence. In other words, each packet is forwarded by either the old routing prior to the update or the new routing after the update, but not together.

The initial state and final states are initially known, while intermediate states $\{1, 2, ..., m\}$ needs to be determined. Each flow f is associated with bandwidth demand B_f, routed through a possible path $p \in P(f)$ between its source and destination. The path set $P(f)$ is pre-computed by the SDN controller such that all paths are loop free for flow f. The flow weight W_f is determined by the preference application, which is beyond the scope of this paper. Parameters like: M, W_f, P_f, B_f and source and target state are already defined. For each flow f, it is required to find a feasible path in set P_f at each intermediate stage, and then select the key flows that will be limited by a significant proportion at each migration step. The primary goal of this operation is to minimize congestion, which is expressed as a weighted sum:

$$\sum_{m=0}^{M} \sum_{f \in F} W_f l_{m,f} \tag{2}$$

Hence, the network update problem can be formulated as a nonlinear optimization program as below:

$$minimize \sum_{m=0}^{M} \sum_{f \in F} W_f l_{m,f} \tag{3}$$

subject-to:

$$\sum_{f \in F} B_f b^e_{m,f} - \sum_{f \in F} l_{m,f} b^e_{m,f} \leq B_c, \forall m \in M, \forall e \in E$$

$$b^e_{m,f} = max(\sum_{p \in P: e \in p} a^p_{m,f}, \sum_{p \in P: e \in p} a^p_{m+1,f}), \forall m \in M, \forall e \in E, \forall f \in F$$

$$\sum_{p \in P_f} a^p_{m,f} = 1, \forall m \in M\{0\}, \forall f \in F$$

$$a^p_{m,f} \geq 0, \forall m \in M\{0\}, \forall f \in F, \forall p \in P_f$$

$$l_{m,f} \leq B_f, \forall m \in M, \forall f \in F$$

The problem represented by equation 3 must follow congestion less constraint, flow demand constraint, capacity constraint and non-splittable flow constraints where $l_{m,f}$, $b^e_{m,f}$ and $a^p_{m,f}$ are decision variables. $l_{m,f}$ represents the limited volume of flow f when the network is updated from stage m to $m+1$. $b^e_{m,f}$ indicates whether or not flow f passes edge e during the migration from stage m to $m+1$. $a^p_{m,f}$ determines weather flow f chooses path p in stage m or not. $a^p_{m,f}$ =1 when flow f chooses path p in stage m.

3.2 Two-phase Heuristic Algorithm

Since the formulated problem is NP-hard [14]. According to present data science, optimal solutions for NP-hard problems cannot be guaranteed in a reasonable amount of time for problem instances of practical sizes. [15]. Exact algorithms are not ideal in terms of practical usability due to their long running time. The exact solution of problem instances with massive number of nodes can take many hours, and there are also problem instances with practical sizes that can't be resolved due to memory constraints. Hence, a two-phase heuristic approach is proposed, which provides the desired efficiency while also limiting the running time. The algorithm is divided into two phases which address congestion duration and rate limitation individually.

3.2.1 Phase-I: Update Schedule

The phase-I tries to acquire the update schedule by combining the congestion duration and the overloaded volume ratio. Because, $b^e_{m,f}$ is no bigger than 1, optimization problem results to MIP. The total overloaded volume of all migration steps becomes the objective function, taking into account the congestion duration and the overloaded volume
The update schedule problem is illustrated as a LP problem as below:

$$minimize \sum_{m=0}^{M} \sum_{f \in F} W_f l_{m,f} \tag{4}$$

subject-to:

$$\sum_{f \in F} D_f h^e_{m,f} - \sum_{f \in F} l_{m,f} \leq B_c, \forall m \in M, \forall e \in E$$

$$\sum_{p \in P_f: e \in p} a^p_{m,f} \leq h^e_{m,f}, \forall m \in M, \forall e \in E, \forall f \in F$$

$$\sum_{p \in P_f: e \in p} a^p_{m+1,f} \leq h^e_{m,f}, \forall m \in M, \forall e \in E, \forall f \in F$$

$$\sum_{p \in P_f} a^p_{m,f} = 1, \forall m \in M\{0\}, \forall f \in F$$

$$a^p_{m,f} \geq 0, \forall m \in M\{0\}, \forall f \in F, \forall p \in P_f$$

$$l_{m,f} \leq B_f, \forall m \in M, \forall f \in F$$

This problem of Update Schedule can be illustrated in the form of algorithm as below:

Algorithm 1: Update Schedule Algorithm

Input : Source state, Target state, M, W_f, P_f, B_f

Output : Routing at each intermediate state, Key flow f_*

Get Solutions $a^p_{m,f}$ from LP 4;
for m^* *1 to M* **do**
  for *each* f^* *in F* **do**
    path p^* path p* when the $a^{p^*}_{m^*,f^*}$ is the maximum item of set $a^{p^*}_{m^*,f^*}$
    Flow f^* is routed through path p^* at stage m^*
  end
end

3.2.2 Phase-II: Rate limitation

During each migration of the update schedule, Phase-II involves limiting of key flows by a some volume. As soon as the update schedule is calculated, the rate limitation is required as long as there occur congestion in each migration step from stage s to $s+1$. With provided flow demand, weight, source routing and target routing , we need to determine the optimal rate limitation technique such that the weighted sum represented in **equation** 5 is highest.

$$\sum_{f \in F} W_f v_f \tag{5}$$

The rate limitation problem is formulated as LP as below:

$$maximize \sum_{f \in F} W_f v_f \tag{6}$$

subject-to:

$$\sum_{f \in F} v_f K_{f,e} \leq B_c, \forall e \in E$$

$$v_f \leq B_f, \forall f \in F$$

The decision variable $K_{f,e}$ which determines weather flow f passes through edge e at initial state or final state can only be 1 or 0. v_f is the assigned bandwidth for flow f. As long as flow f passes through edge e at initial or final state, $K_{f,e}$ is 1. The allocated bandwidth of each f should not higher than demand value to meet condition for no congestion. Also, the allocated bandwidth v_f of each flow f should not be higher than demand value B_f.This problem of rate limitation can be illustrated in the form of algorithm as **Algorithm 2**.

Algorithm 2: Rate Limitation Algorithm

Input : Source state and Target state, M, W_f, P_f, B_f

Output : Rate limitation for flows during each migration

Get Key flow f^* from Algorithm 1;
Set source stage as stage 0
Set target stage as $M+1$
for $m^* = 1$ *1 to M* **do**
  for *each* f^* *in F* **do**
    LP equation 6, set path of flow f^* at stage m^* as the source path of flow f^*
    LP equation 6, set path of flow f^* at stage m^*+1 as the source path of flow f^*
  end
  Solve LP 6 and result is the rate limitation during migration from stage m to m+1
end

4. Experimental Setup

The test network topology is implemented under Linux, using Mininet software to emulate the network, the RYU controller to add the flow tables in the Openflow switches. Python programming language is used to write the algorithm scripts and iperf3 tool is used for analysing network performance. The Abilene topology consisting of 11 open-flow switches and 14 interconnected links between them as shown in figure 2 is used for simulation. We set 70 Mbps bandwidth for each of those links.

We assume 5 simultaneous flows between 10 hosts as illustrated in table 6.2. We take link bandwidth to be 70 Mbps and flow bandwidth as 25Mbps and run each iteration for 1 minute. We take five iterations and take their average to make our result more accurate.

Flow	Initial Path	Final Path	Host Pair	Flow Weight
F1	S1→ S2 → S3→ S4→ S8	S1→ S2 → S3→ S9 → S8	h1-h6	2
F2	S2→ S3 → S4→ S8	S2→ S3 → S9→ S8	h3-h7	10
F3	S1→ S2 → S3→ S9	S1→ S11 → S10 → S9	h2-h9	2
F4	S3 → S4→ S8	S3→ S9 → S8	h4-h8	10
F5	S4 → S3→ S9	S4→ S8 → S9	h5-h10	10

Table 1: Flow Set for the Abilene topology

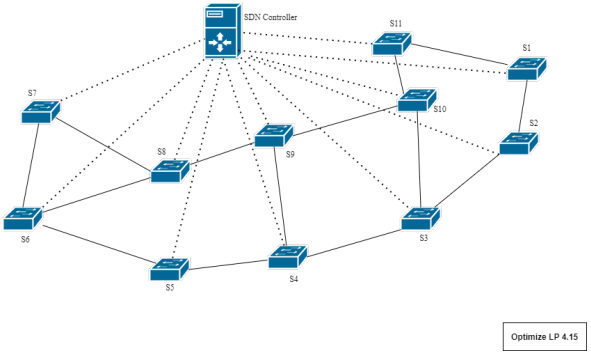


Figure 2: Abilene Topology

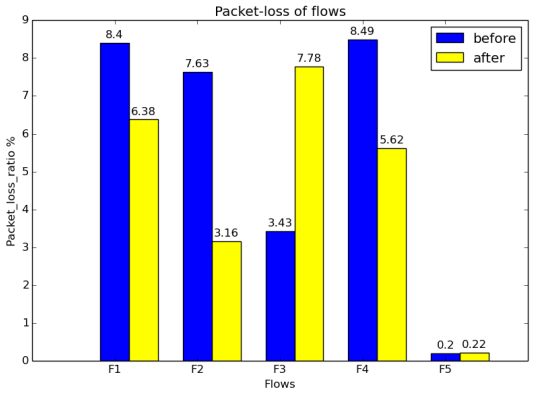


Figure 3: Comparative packet loss ratio of flows before and after the run of algorithm

5. Results and Discussion

This section discusses the simulation results obtained and compare the different scenarios. To minimize the probable margin of errors, each measurement has been executed several times and an average is taken.Those results are presented as graphical and/or tabular form for easier observation of the results.

5.1 Weight-Identical Flows

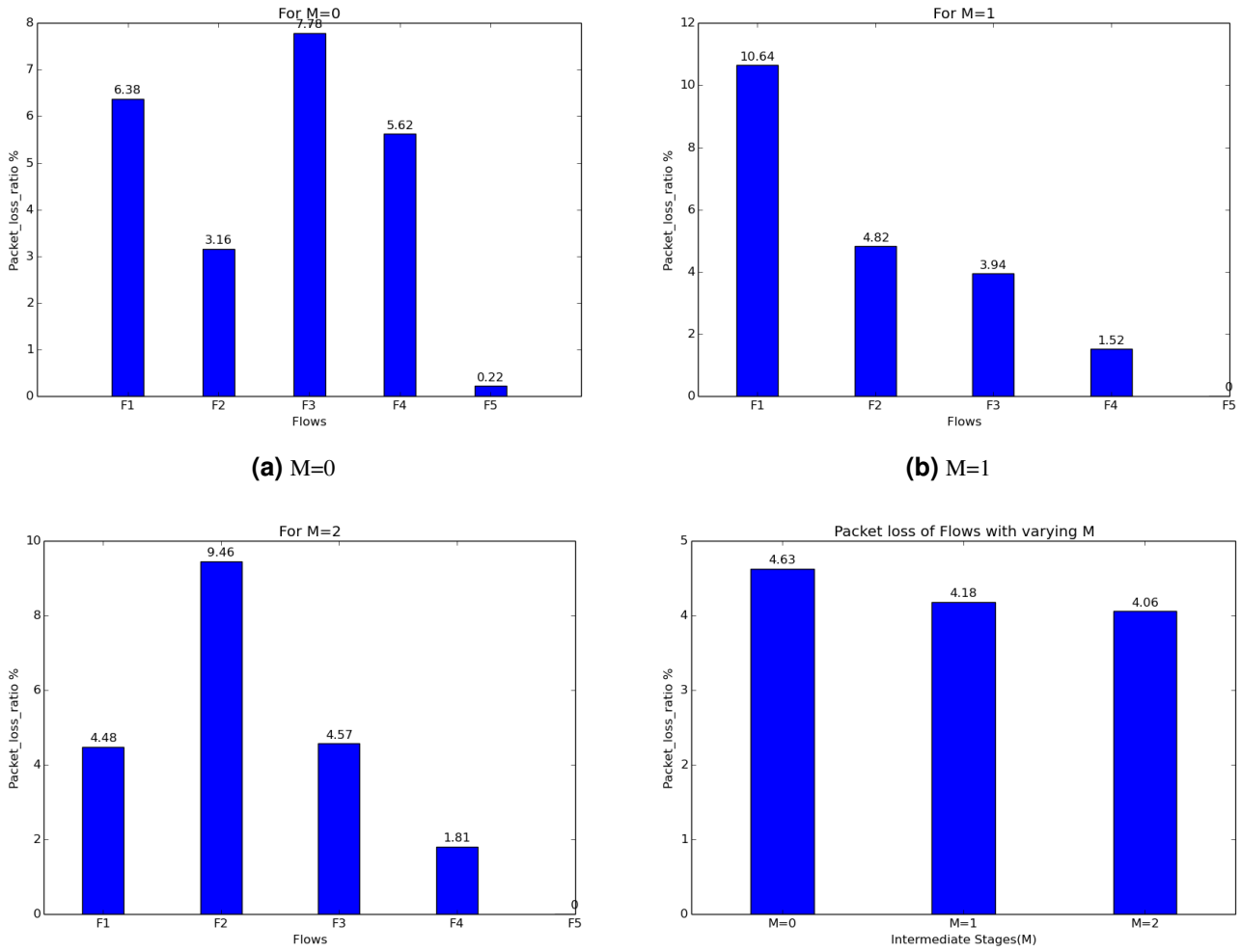
Initially, we take the scenario where all the flows have the same weight of 10 unit. Then packet loss ratio of the network and each individual 5 flows are calculated up to two intermediate stages. In the figure The x-axis represents flows before and after the run of algorithm. The y-axis show the averaged packet loss ratio (%) of the five flows before and after the run of algorithm.

Even without inserting intermediate stages between the initial and final stages, we can see a significant improvement in packet loss of flows after the algorithm is executed. After the run of algorithm, the key flow seems to be the F3 having higher loss and average packet loss ratio decreases from 6.5%(average before) to 4.63 % (average after). The comparative bar chart diagram is shown in figure 3.

The packet loss of flows at various intermediate stages is depicted in the figure 4.When M=0, flow F3 has the key loss and flow F1, F2 and F4 also has comparatively higher loss F5. Flow F5 has the negligible packet loss ratio because it has no overlapping flows in the associated link. Similarly, when M=1 and M=2 the key loss is in the flow F1 and F2 respectively leaving other flows with lesser packet loss ratios. As can be seen, the key flow with a higher packet loss ratio may change based on the intermediate stages. However, the pattern of having a lower packet loss ratio continues even in two intermediate stages.

5.2 Weight-Differential Flows

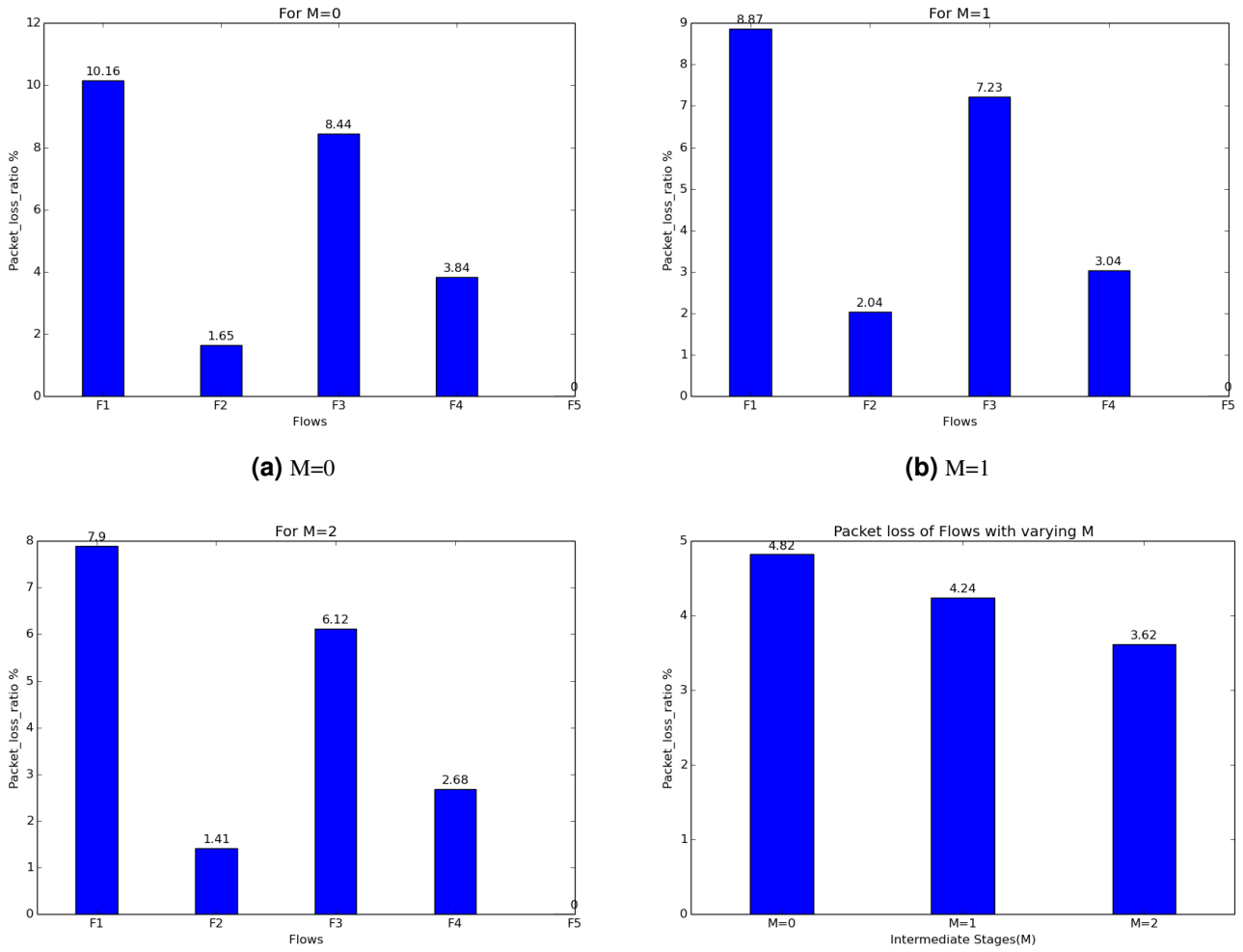
The performance of the network for flows with varying importance weight as shown in the table 1 is also evaluated. Packet loss ratios of flows at different intermediate stages and their average is shown in figure 5. In this scenario the key flow is F1 for all intermediate stages M=0,1 and 2. Flow F5 has no any loss because it do not share flow path with other flows.Flow F2 and F4 has comparatively lower loss than F1 and F3 which proves the role of flow importance in prioritizing the flows. Hence,we can conclude that those flows having higher importance have the lower packet loss ratios and also the packet loss ratio decreases with the insertion of intermediate stages when migrating from initial to final stage. From the figure we can observe that average packet



(a) M=0 **(b)** M=1

(c) M=2 **(d)** Average of all M

Figure 4: Packet loss ratios of identical weight flows at different intermediate stages M



(a) M=0

(b) M=1

(c) M=2

(d) Average of all M

Figure 5: Packet loss ratios of differential weight flows at different intermediate stages M

loss ratio of weight identical flows and weight importance flows seem similar though importance factor is introduced.

5.3 Update Time

In order to visualize the time delay caused by the insertion of intermediate stages, we calculate the time difference between the start and end of the update process. The results, as shown in the figure 6, reveal that as the number of intermediate states in the update process grows, so does the time it takes to complete the update for both identical-weight and differential-weight situations.

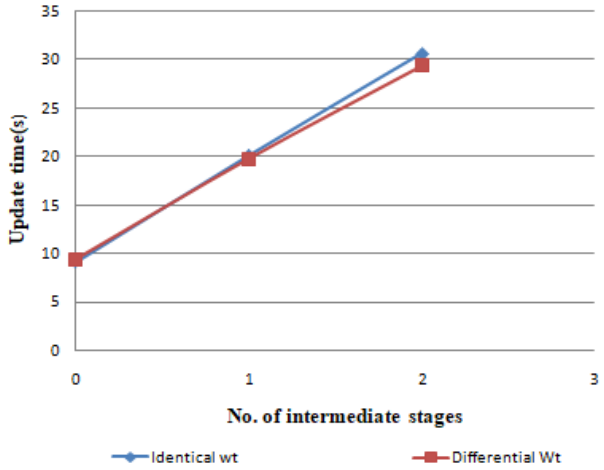


Figure 6: Update time at different intermediate stages M

6. Conclusion and Future Works

The Two-phase heuristic algorithm minimizes the congestion during network update while also conserving important flows. In both identical-weight as well as differential-weight flow cases, inserting the number of intermediate stages while moving from the initial to the final state, lowers the packet loss ratio. On the other hand, along with the increase in number of intermediate stages, the time to achieve update also increases. Also, important flows can be preserved to some extent.

To minimize the unavoidable transient congestion during network update, we proposed and tested a Two-phase Heuristic algorithm that comprises two distinct phases: update schedule and rate limitation. Update schedule phase aims to find the key flow in the network by analysing the flows in all paths of source state and intermediate stages, while rate limitation phase limits the key flow such that overall packet loss can be minimized. The Two-phase Heuristic algorithm not only minimizes packet loss ratios but also preserves essential flows, as per the results. The key flow was limited in weight-identical flows, but the priority wise limitation of flows occurred in weight-differential flows.

One of the problem that this research can be extended to is determining the number of intermediate phases. Inserting many number of intermediate stages may result in a higher packet loss ratio. So, research could focus on determining the best time and optimum number of intermediate stages for a given bounded congestion which completes and automates this technique. Updating the SDN network based on user's desire makes update process flexible and more customized. Another study could be pursued in the future is the search for a dynamic flow-based network updating mechanism based on dependency graphs allowing flows to be transferred back and forth frequently.

Acknowledgments

The authors would like to express their gratitude to the Department of Electronics and Computer Engineering, Pulchowk Campus, IOE, TU, for all of their help and assistance during this research.

References

[1] Chaozhun Wen, Peng Yang, Qiong Liu, Jingjing Luo, and Li Yu. Minimizing congestion impairment of network update in sdn: A flow-based solution. In *2018 10th International Conference on Wireless Communications and Signal Processing (WCSP)*, pages 1–6, 2018.

[2] Hongqiang Liu, Xin wu, Ming Zhang, Lihua Yuan, Roger Wattenhofer, and David Maltz. zupdate: Updating data center networks with zero loss. volume 43, 08 2013.

[3] Xuezhou Ma, Sangmin Kim, and Khaled Harfoush. Towards realistic physical topology models for internet backbone networks. In *2009 6th International Symposium on High Capacity Optical Networks and Enabling Technologies (HONET)*, pages 36–42, 2009.

[4] Tal Mizrahi and Yoram Moses. On the necessity of time-based updates in SDN. In *Open Networking Summit 2014 (ONS 2014)*, Santa Clara, CA, March 2014. USENIX Association.

[5] Mark Reitblatt, Nate Foster, Jennifer Rexford, Cole Schlesinger, and David Walker. Abstractions for network update. *SIGCOMM Comput. Commun. Rev.*, 42(4):323–334, August 2012.

[6] S. A. Amiri, S. Dudycz, S. Schmid, and S. Wiederrecht. Congestion-free rerouting of flows on dags. In *ICALP*, 2018.

[7] Chi-Yao Hong, Srikanth Kandula, Ratul Mahajan, Ming Zhang, Vijay Gill, Mohan Nanduri, and Roger Wattenhofer. Achieving high utilization with software-driven wan. In *Proceedings of the ACM SIGCOMM 2013 Conference on SIGCOMM*, SIGCOMM '13, page 15–26, New York, NY, USA, 2013. Association for Computing Machinery.

[8] Xin Jin, Hongqiang Harry Liu, Rohan Gandhi, Srikanth Kandula, Ratul Mahajan, Ming Zhang, Jennifer Rexford, and Roger Wattenhofer. Dynamic scheduling of network updates. *SIGCOMM Comput. Commun. Rev.*, 44(4):539–550, August 2014.

[9] Ratul Mahajan and Roger Wattenhofer. On consistent updates in software defined networks. In *Proceedings of the Twelfth ACM Workshop on Hot Topics in Networks*, HotNets-XII, New York, NY, USA, 2013. Association for Computing Machinery.

[10] Rohan Gandhi, Ori Rottenstreich, and Xin Jin. Catalyst: Unlocking the power of choice to speed up network updates. In *Proceedings of the 13th International Conference on emerging Networking EXperiments and Technologies, CoNEXT 2017, Incheon, Republic of Korea, December 12 - 15, 2017*, pages 276–282. ACM, 2017.

[11] Sebastian Brandt, Klaus-Tycho Foerster, and Roger Wattenhofer. Augmenting flows for the consistent migration of multi-commodity single-destination flows in sdns. *Pervasive and Mobile Computing*, 36, 04 2017.

[12] Jiaqi Zheng, Hong Xu, Guihai Chen, Haipeng Dai, and Jie Wu. Congestion-minimizing network update in data centers. *IEEE Transactions on Services Computing*, 12(5):800–812, 2019.

[13] Sofia Naning Hertiana, Hendrawan, and Adit Kurniawan. Performance analysis of flow-based routing in software-defined networking. In *2016 22nd Asia-Pacific Conference on Communications (APCC)*, pages 579–585, 2016.

[14] Jiaqi Zheng, Hong Xu, Guihai Chen, and Haipeng Dai. Minimizing transient congestion during network update in data centers. In *2015 IEEE 23rd International Conference on Network Protocols (ICNP)*, pages 1–10, 2015.

[15] Bin Chen and Guangri Quan. Np-hard problems of learning from examples. In *2008 Fifth International Conference on Fuzzy Systems and Knowledge Discovery*, volume 2, pages 182–186, 2008.