FlowMan: QoS-Aware Dynamic Data Flow Management in Software-Defined Networks

Ayan Mondal, *Student Member, IEEE*, and Sudip Misra, *Senior Member, IEEE*

***Abstract*—In this paper, we study the problem of data flow management in the presence of heterogeneous flows — elephant and mice flows[1] in software-defined networks (SDNs). The researchers did not consider the presence of heterogeneous flows in SDN in the existing literature. Moreover, we argue that the optimal data flow management in the presence of heterogeneous flows is NP-hard. Hence, we propose a game theory-based heterogeneous data flow management scheme, named FlowMan. In FlowMan, initially, we use a generalized Nash bargaining game to obtain a sub-optimal problem, which is NP-complete in nature. By solving it, we get the Pareto optimal solution for data-rate associated with each switch. Thereafter, we use a heuristic method to decide the flow-association with the switches, distributedly, which, in turn, helps to get a Pareto optimal solution. Extensive simulation results show that FlowMan is capable of ensuring quality-of-service (QoS) for data flow management in the presence of heterogeneous flows. In particular, FlowMan is capable of reducing network delay by 77.8–98.7% while ensuring 24.6–47.8% increase in network throughput compared to the existing schemes such as FlowStat and CURE. Additionally, FlowMan ensures that per-flow delay is reduced by 27.7% with balanced load distribution among the SDN switches.**

***Index Terms*—Load Balancing, Software-Defined Networks, Nash Bargaining game, IoT, Heterogeneous Flow.**

I. Introduction

With the advent of Internet of Things (IoT), huge amount of data traffic, i.e., *big-data*, is generated by the various IoT-devices, which results in significant difficulties in managing this data. Under these circumstances, software-defined networks (SDNs) are getting attention from the research community for supporting new services and applications. Essentially, SDN aims to divide the network functionalities and thus, follows a two-plane architecture, comprising of *control* and *data planes*. The control plane handles network control. On the other hand, the data plane manages the packet processing and forwarding tasks [2]. In the control plane, the application programming interfaces (APIs) are of two types such as northbound and southbound. Using the northbound APIs, the applications directly interact with the SDN controller. On the other hand, the interaction between the controller and the switches is supported by the southbound APIs. Additionally, the data traffic flows are handled by these APIs according to the flow-rules installed in the ternary content-addressable memory (TCAM) space. These flow-rules are, in turn, installed in the switches by the controller, which has a logically centralized view of the network. Therefore, SDN is considered to be one of the most promising architectures for supporting numerous applications generated from the heterogeneous IoT-devices [3].

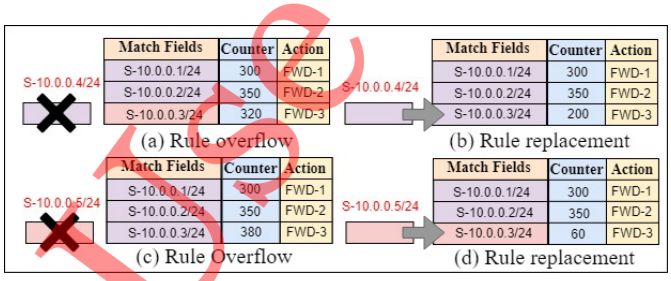

Fig. 1: Issues in Heterogeneous Flow Management in SDN

In SDN, data flow[2] management and flow-rule placement in the switches are two significant problems. In the existing literature, few works [2], [4], [5] focused on proposing schemes and architectures for addressing these issues. However, in these works, researchers considered each flow to be homogeneous in nature [6]. This assumption is no longer practical in the modern world, in which, due to the digitization of everything, we encounter different types of data traffic with high and low volume, i.e., elephant and mice flows, respectively, in the network. In such a scenario, due to the unbalanced nature of data traffic, the switches handling the elephant flows incur high delay, whereas the switches handling the mice flows incur low throughput. As shown in Figure 1, the incoming flows can be discarded due to overflow or can replace the existing flows, which results in high delay or low throughput, respectively. This, in turn, degrades the overall performance of the SDN. In the existing literature, researchers overlooked this unbalanced traffic problem in the presence of heterogeneous flows generated from the IoT devices. Therefore, in order to ensure high quality-of-service (QoS), i.e., high throughput and low delay, there is a need to design efficient flow management techniques for SDN to ensure high throughput and low delay, while considering the presence of heterogeneous IoT traffic.

Motivating Scenario: We consider an SDN with IoT devices and switches connected using access points (APs). These heterogeneous IoT devices are capable of running different applications such as Internet browsing, Email, VoIP calls, and video streaming [7]. As per Federal Communications Commission (FCC) [7], video streaming applications in IoT devices consume almost 10^4 times more bandwidth than

The authors are with the Department of Computer Science and Engineering, Indian Institute of Technology Kharagpur, India (Email: ayanmondal@iitkgp.ac.in; smisra@sit.iitkgp.ernet.in).

[1]The data flow is divided into two categories based on the associated data-rate [1]. The flows with high data-rate are termed as elephant flows, while those with low data-rate are termed as mice flows.

[2]We define data flow as a stream of data packets

normal applications such as browsing and calls. Therefore, we argue that the flows generated from these video streaming applications are the elephant flows, whereas the other flows are mice flows. This necessitates the design of a dynamic heterogeneous flow management scheme in SDN.

In this paper, we introduce a QoS-aware stochastic data flow management scheme, named *FlowMan*, for SDN in the presence of heterogeneous flows, i.e., elephant and mice flows. We use a *generalized Nash bargaining game* to decide the Pareto optimal datarate to be allocated to each SDN switch, while considering the heterogeneous flows within one-hop network. Here, using a generalized Nash bargaining game, we achieve an intermediate Pareto optimal throughput for the switches. Thereafter, using a distributed heuristic method, i.e., a method for solving the bounded Knapsack problem, we decide the switch and flow-rule association for ensuring optimal data flow management in SDN. The specific contributions of this paper are listed as follows:

1) We present a QoS-aware data flow management scheme, FlowMan, for ensuring high QoS, i.e., maximizing the throughput and minimizing the delay, of SDN in the presence of heterogeneous flows.

2) We argue that dynamic data flow management in the presence of heterogeneous flows, which can be mapped to zero-one knapsack problem, is an NP-hard problem. Therefore, we use the generalized Nash bargaining game to obtain a sub-optimal problem which can be mapped to the bounded Knapsack problem, an NP-complete problem.

3) We present an algorithm for FlowMan. The first part of the algorithm deals with the optimal data rate selection by using the bisection method. On the other hand, the second part of the algorithm focuses on the optimal mapping of data flows to the appropriate switches for efficient data flow management.

II. Related Works

In the past few years, many research works on the different aspects of SDN emanated, viz., [8]–[11]. The existing literature are divided into two categories — (a) data traffic management and (b) flow-rule management.

Some of the SDN management schemes proposed in the existing literature are discussed here. Huang *et al.* [8] proposed a rule multiplexing scheme to reduce the usage of TCAM memory while maintaining QoS constraints. The authors formulated a joint optimization problem while considering route engineering and rule placement. In another work, Sadeh *et al.* [12] proposed a scheme, named Bit Matcher', to reduce the TCAM memory usage for a given set of flow-rules. Rottenstreich *et al.* [9] proposed a traffic splitting scheme for switches while considering the heterogeneity of the network paths or servers and the limited capacity of the flow-tables.

Agarwal *et al.* [13] studied the traffic handling in SDN. The authors showed that having a centralized view of the network, the controller is capable of reducing the delay and packet loss in data traffic. Tseng *et al.* [14] studied the path stability in hybrid SDN. The authors calculated the routes locally to reduce computational complexity, thereafter, using a centralized scheme to re-evaluate the routes for gaining stability. Moradi

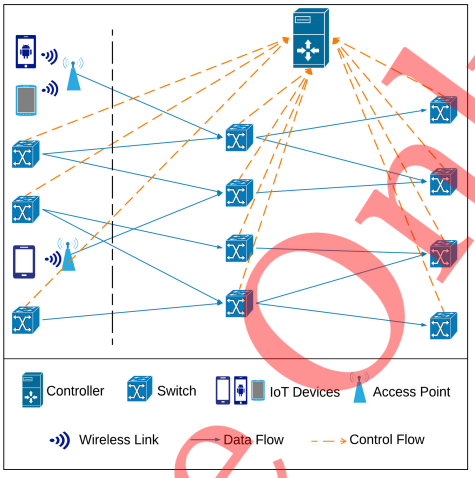

Fig. 2: Schematic Diagram of SDN Architecture

et al. [11] proposed an efficient traffic engineering scheme, named DRAGON, for SDN-based ISP networks having different types of network links and switches. In DRAGON, the flow optimization problem is broken down into sub-tasks having different objectives and the sub-tasks are executed in parallel to reduce complexity. In another work, Allybokus *et al.* [15] proposed fair resource allocation scheme in the presence of multiple network paths in a distributed SDN scenario using the alternating direction method of multipliers. Sanvito *et al.* [16] proposed a scheme for deciding the time frame to reconfigure flow-tables, while considering overlap data flow paths. Mondal *et al.* [3] proposed a scheme to ensure high throughput in SDN while assuming that the volume of data to be generated is known *a priori*. The authors optimally distributed the traffic load among the switches and ensured high network throughput. Tahaei *et al.* [17] presented a SDN-based flow management scheme for data center networks with multiple controllers and selected an optimal number of switches. In another work, Görkemli *et al.* [18] proposed a novel distributed dynamic control plane architecture in which the switches communicate with their controllers through a virtual overlay network. The authors also proposed the introduction of a "control flow table" to manage the dynamic control plane traffic. Another dynamic traffic engineering scheme was proposed by Bera *et al.* [6] in which the authors attempted to reduce the control overhead by reducing the number of messages to the controller. The authors proposed a greedy heuristics-based approach to determine the optimal number of candidate switches (with higher TCAM) necessary to reduce the involvement of the controller.

Some of the schemes in the existing literature, focusing on the flow-rule processing of SDN are discussed here. Meiners *et al.* [19] proposed a flow-rule compression scheme to accommodate high number of switches in SDN. Mao *et al.* [20] proposed a Convolutional Neural Networks (CNNs)-based intelligent traffic management scheme. The authors considered that the controller has high computational resources. Mogul *et al.* [21] proposed a hashing-based scheme and reduced flow-table lookups. Rottenstreich *et al.* [22] studied the shared multi-core processing scheme and evaluated a trade-off between the number of allocated cores and the associated delay

in the context of network virtualization. Reitblatt *et al.* [23] studied the problem of consistent flow-rule update network-wide, while processing the incoming data packet. In another work, Katta *et al.* [5] evaluated a trade-off between the TCAM space and the duration for network update. On the other hand, Hayes *et al.* [24] classified the traffic in SDN. In another work, Rottenstreich and Tapolcai [25] proposed a limited-size classifier set to accommodate the flow-rules in limited TCAM memory.

Synthesis: From the survey of the existing literature, we argue that a few research works on data flow management in SDN consider the efficient utilization of TCAM of each switch. However, the researchers have not considered the presence of heterogeneous flows, i.e., elephant and mice flows, in the existing literature which leads to unbalanced data traffic. Additionally, dynamic flow management in SDN while maximizing throughput and minimizing delay is a significant requirement for ensuring high QoS.

III. System Model

We consider an SDN with multiple switches, a centralized controller, and multiple IoT devices and the SDN switches connected using access points (APs). We consider that these IoT devices generate *heterogeneous* data-traffic such as Internet browsing, Email, VoIP calls, and video streaming. Additionally, we consider that the switches are *heterogeneous* in nature and have different TCAM capacity. We present the SDN network as a flow network $G(V, E)$, where V and E denote the set of vertices and the set of edges in the flow network, respectively. Here, we consider that $V = \bigcup_L V_I^L \cup \bigcup_L V_S^L$, where V_I^L and V_S^L represent the set of IoT devices and the set of switches in the layer L of the flow network, respectively, $L \geq 0$, and $V_S^0 = \{\emptyset\}$. We consider that each edge $e_{ij} \in E$ has a limited bandwidth B_{ij} and $B_{ij} > 0$, given that $e_{ij} \neq 0$. We consider two QoS parameters — throughput and delay — for evaluating the performance of the network. The throughput and delay per flow depend on the allocated capacity of the corresponding flow. In this work, we consider that the network flows are comprised of multiple *elephant* (high-volume) and *mice* (low-volume) flows [1]. Hence, we argue that there is a need to focus on ensuring both the optimal throughput and delay of the network. We consider that each IoT device $n \in V_I^L$ generates $F_n^E(t)$ and $F_n^M(t)$ set of elephant and mice flows, respectively, at time instant t, where $|F_n^E(t)|, |F_n^M(t)| \geq 0$. Therefore, the number of flows $F_L(t)$ in layer L of the network is as follows:

$$F_L(t) = \sum_{n \in V_I^L} [|F_n^E(t)| + |F_n^M(t)|] \tag{1}$$

We assume that there is sufficient space in the switches to place $|F_L(t)|$ rules, i.e., $|F_L(t)| \leq \sum_{k \in V_S^{(L+1)}} R_k^{max}$, where R_k^{max} is the maximum flow-rules which can be accommodated at SDN switch k. On the other hand, each elephant flow $f_n^E(t) \in F_n^E(t)$ and each mice flow $f_n^M(t) \in F_n^M(t)$ generate data with rate $r_n^E(t)$ and $r_n^M(t)$, respectively. Therefore, the data-generation rate is given as $\Psi(t) = \sum_{f_n^E(t) \in F_n^E(t)} r_n^E(t) + \sum_{f_n^M(t) \in F_n^M(t)} r_n^M(t)$. Moreover, we consider that each SDN switch $k \in V_S^{(L+1)}$ handles a set of installed rules $R_k(t)$, where $R_k(t) \leq R_k^{max}$. Therefore, we have:

$$R_k(t) = |F_k^e(t)| + |F_k^m(t)| \tag{2}$$

where $F_k^e(t)$ and $F_k^m(t)$ represent the set of elephant and mice flow-rules installed in SDN switch k. Therefore, the throughput $T_k(t)$ of SDN switch k, where $\Psi(t) = \sum_{k \in V_S^{(L+1)}} T_k(t)$, is defined as follows:

$$T_k(t) = \sum_{f_n^E(t) \in F_k^e(t)} r_n^E(t) + \sum_{f_n^M(t) \in F_k^m(t)} r_n^M(t) \tag{3}$$

Additionally, we consider that each SDN switch k has processing capacity of $\mu_k^S(t)$. Considering that the flow processing at SDN follows a packet-centric approach and following the work of Park and Schaar [26], the processing delay $D_k(t)$ [10] is defined as follows:

$$D_k(t) = D_k^0 + \frac{\mu_k^S(t)}{T_k(t) - T_k^0(t)} \tag{4}$$

where T_k^0 denotes the minimum resource share of each switch k; D_k^0 is the minimum delay for processing a packet, while considering the switch is idle, i.e., there is no packet in the switch. In this work, our objective is to obtain an optimal association among the data traffic flows and the switches for maximizing the network throughput and minimizing the delay.

IV. QoS-Aware Data Flow Management (FlowMan) Scheme

To study the interaction among the switches and the IoT devices in the considered problem scenario, we use *generalized Nash bargaining* game theory and propose a QoS-aware dynamic data traffic management scheme, named FlowMan, for SDNs with IoT devices. Here, we consider that the switches act cooperatively in order to ensure high network throughput and low processing delay. In the proposed scheme, FlowMan, the strategy of each switch is to decide the optimal subset of flows to be handled by it. Based on the rule-space capacity of the switches, we introduce the bargaining power $\boldsymbol{\alpha} = \{\alpha_1, \cdots, \alpha_k, \cdots, \alpha_{|V_S^L|}\}$, where α_k denotes the bargaining power of switch k. Hence, we summarize the components of the generalized Nash bargaining game in the proposed scheme, FlowMan, as follows:

1) SDN switches act as the players and bargain among themselves to distribute the set of flow rules to be installed.

2) Data flows available in the network form the resource pool. Here, we only consider the flows generated/handled by one-hop neighbors, i.e., switches and IoT devices.

3) Switches are heterogeneous in terms of TCAM capacity.

4) Bargaining power of each switch depends on its TCAM capacity.

A. Justification for Using Generalized Nash Bargaining Game

As mentioned earlier, the problem of optimal distribution of heterogeneous data-flow among the available switches in SDN in order to simultaneously ensure high throughput and

low delay is an NP-hard problem, as it can be mapped to the zero-one knapsack problem [27]. In order to obtain a solution to this problem in polynomial time, we adopt a game-theoretic approach. In SDN, the aforementioned problem scenario can be visualized in the form of a "bargaining" situation in which the switches bargain among themselves to obtain their fair share of the data-flows. Here, the switches act cooperatively and try to agree upon a distribution of flow-rules which ensures mutual benefit. Moreover, in this work, the switches are also considered to be heterogeneous with respect to TCAM capacity. Hence, in order to model the considered problem scenario and the aforementioned properties of SDN, we use the generalized Nash bargaining game. Here, the generalized Nash bargaining game not only takes into consideration the co-operative nature of the switches but also perfectly captures the heterogeneity of switches by introducing the Nash bargaining power of each individual switch. The Nash bargaining solution provides a Pareto optimal solution (which is shown in the subsequent sections) for deciding the maximum data-rate to be handled by each switch for ensuring high network throughput with less delay. Hence, we argue that the generalized Nash bargaining game is the most suitable technique for managing dynamic flow-traffic in the presence of multiple elephant and mice flows in SDN.

B. Game Formulation

In the proposed game, we consider that each switch $k \in V_S^{(L+1)}$ decides its strategy, i.e., *bargain*, for distributing the one-hop network flows (including elephant and mice flows) while ensuring high network throughput and low delay in data traffic. The utility function $\mathcal{U}_k(\cdot)$ of each switch k signifies the utilization of its capacity and TCAM memory. Motivated by the work of Park and Schaar [26], we consider that $\mathcal{U}_k(\cdot)$ needs to ensure the following properties:

1) Each switch tries to maximize the utilization factor of its capacity. Therefore, each switch k tries to increase $T_k(t)$, while ensuring the constraint — $R_k(t) \leq R_k^{max}$.

2) Each switch k tries to reduce the overall network delay by reducing the packet queuing delay $D_k(t)$.

Therefore, we define $\mathcal{U}_k(\cdot)$ for each switch as follows:

$$\mathcal{U}_k(T_k(t)) = \frac{\lambda}{D_k^0 + \frac{\mu_k^S(t)}{T_k(t) - T_k^0}} = \frac{\lambda(T_k(t) - T_k^0)}{D_k^0(T_k(t) - T_k^0) + \mu_k^S(t)} \tag{5}$$

where λ is a constant and $\lambda > 0$. Moreover, we consider that in the proposed scheme, FlowMan, each switch k ensures a minimum payoff d_k which is termed as its *disagreement point*. Disagreement point is calculated as the equilibrium point while the players act non-cooperatively. Here, we have — $d_k = \mathcal{U}_k(T_k^0)$. Hence, in FlowMan, we ensure that the payoff of each switch will be higher than $\mathcal{U}_k(T_k^0)$. Therefore, from Equation (5), we get — $d_k = 0$. Considering that each switch k handles $T_k(t)$ amount of resource, we define a *feasible utility set* $\boldsymbol{S}$ as — $\boldsymbol{S} = \{\cdots, \mathcal{U}_k(T_k(t)), \cdots\}$. We argue that $\boldsymbol{S}$ is nonempty, convex, closed, and bounded. Additionally, we represent the disagreement point vector $\boldsymbol{d}$ as — $\boldsymbol{d} = \{\cdots, \mathcal{U}_k(T_k^0), \cdots\}$. Here, $\boldsymbol{d}$ is a set with elements having zero value. Thereby, the bargaining problem is defined as a tuple $< \boldsymbol{S}, \boldsymbol{d} >$. We argue that the Pareto optimal solution exists in the proposed scheme as defined in Definition 1.

Definition 1. *We define that $\boldsymbol{\mathcal{U}}^*(t)$ is Pareto optimal, where $\boldsymbol{\mathcal{U}}^*(t) = (\cdots, \mathcal{U}_k(T_k^*(t)), \cdots)$, iff we have —*

$$(\cdots, \mathcal{U}_k(T_k^*(t)), \cdots) \geq (\cdots, \mathcal{U}_k(T_k(t)), \cdots) \tag{6}$$

As each switch has different bargaining power, we introduce the function $F : \boldsymbol{S} \rightarrow \mathbb{R}^{|V_S^{(L+1)}|}$, where we have —

$$\begin{aligned} F(\boldsymbol{S}, \boldsymbol{d}) &= \prod \left(\mathcal{U}_k(T_k(t)) - \mathcal{U}_k(T_k^0)\right)^{\alpha_k} = \prod \mathcal{U}_k(T_k(t))^{\alpha_k} \\ &= \{\boldsymbol{\mathcal{U}} \in \boldsymbol{B} | \boldsymbol{\mathcal{U}} = \alpha \cdot \boldsymbol{T}(t), \textstyle\sum \alpha_k = 1, \alpha_k \geq 0, \forall k\} \end{aligned} \tag{7}$$

where $\boldsymbol{T}(t) = \{T_1(t), \cdots, T_k(t), \cdots, T_{|V_S^{(L+1)}|}(t)\}$ and $\boldsymbol{B}$ is the bargaining set defined in Definition 2.

Definition 2. *The bargaining set $\boldsymbol{B}$ represents a set containing the Pareto optimal payoff pairs $F(\boldsymbol{S}, \boldsymbol{d})$, and $\boldsymbol{B} \subseteq \boldsymbol{S}$.*

Hence, based on the feasible utility set $\boldsymbol{S}$ and disagreement point vector $\boldsymbol{d}$, the optimization problem is defined as follows:

$$\max_{\boldsymbol{\mathcal{U}} \in \boldsymbol{S}} F(\boldsymbol{S}, \boldsymbol{d}) \tag{8}$$

On the other hand, the solution $\boldsymbol{\mathcal{U}}^*$ needs to satisfy the following constraints —

$$\mathcal{U}_k(T_k(t)) > 0 \text{ and } \sum \alpha_k = 1, \text{ where } \alpha_k \geq 0, \forall k \tag{9}$$

C. Axioms for Generalized Nash Bargaining Solution

In this section, we examine the existence of the generalized Nash bargaining solution (please refer to Theorems 1-4) in the context of the proposed scheme, FlowMan, while ensuring that the following axioms are satisfied [26] — (1) Individual Rationality; (2) Feasibility; (3) Pareto Optimality; (4) Independence of Irrelevant Alternatives; and (5) Independence of Linear Transformations.

Theorem 1. *Feasible utility set $\boldsymbol{S}$ is convex.*

Proof. Please refer to the supplementary file. □

Theorem 2. *In FlowMan, the function $F(\boldsymbol{S}, \boldsymbol{d})$ satisfies Pareto optimality.*

Proof. Please refer to the supplementary file. □

Theorem 3. *The function $F(\boldsymbol{S}, \boldsymbol{d})$ is independent of linear transformation.*

Proof. Please refer to the supplementary file. □

Theorem 4. *The function $F(\boldsymbol{S}, \boldsymbol{d})$ is independent of irrelevant alternatives.*

Proof. Please refer to the supplementary file. □

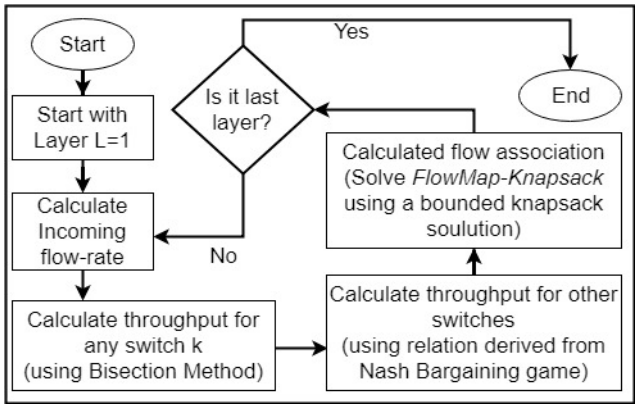

Fig. 3: Workflow Diagram of FlowMan

D. Existence of Generalized Nash Equilibrium

We prove the existence of generalized Nash equilibrium by using variational inequality (VI), as shown in Theorem 5.

Theorem 5. *Given a set of flows and the corresponding data-rates, there exists a generalized Nash equilibrium for each switch in the network.*

Proof. In FlowMan, we aim to maximize function $F(\boldsymbol{S}, \boldsymbol{d})$, which ensures cooperative benefit for the switches. Additionally, in order to prove that there exists a generalized Nash equilibrium, we need to show that the Hessian matrix of $F(\boldsymbol{S}, \boldsymbol{d})$ is negative. We derive the Karush-Kuhn-Tucker (KKT) conditions — Stationary condition, Primal feasibility constraints, Dual feasibility condition, and Complementary slackness condition. For detailed analysis, please refer to the supplementary file. From *primal feasibility* and *complementary slackness* conditions, we get — $\theta_k = 0$,where θ_k is the Lagrangian multiplier. Hence, from *stationary* condition, we get that:

$$\nabla \mathcal{L} = \left[\cdots, \alpha_k \frac{\prod_{l \neq k} \mathcal{U}_l(T_l(t))^{\alpha_l}}{\mathcal{U}_k(T_k(t))} - \frac{\nu \mu_k^S(t) \lambda}{(\lambda - D_k^0 \mathcal{U}_k(T_k(t)))^2}, \cdots \right]^T \tag{10}$$

Additionally, the Hessian matrix of $F(\boldsymbol{S}, \boldsymbol{d})$ is a negative matrix [26], which concludes the proof. □

E. Analysis of Generalized Nash Bargaining Solution

Based on the KKT conditions[3] as mentioned in the Section IV-D, we have $\nu \neq 0$, where ν is a Lagrangian multiplier. Therefore, we get that —

$$\nu = -\frac{\alpha_k \mathcal{U}_k(T_k(t))^{(\alpha_k - 1)} \prod_{l \neq k} \mathcal{U}_l(T_l(t))^{\alpha_l} (\lambda - D_k^0 \mathcal{U}_k(T_k(t)))^2}{\nu \mu_k^S(t) \lambda} \tag{11}$$

Here, ν and λ are constants. Hence, we rewrite Equation (11) as follows:

[3]For detailed calculation, please refer to the supplementary file.

$$\frac{\alpha_k (\lambda - D_k^0 \mathcal{U}_k(T_k^*(t)))^2}{\mu_k^S(t) \mathcal{U}_k(T_k^*(t))} = \frac{\alpha_l (\lambda - D_l^0 \mathcal{U}_l(T_l^*(t)))^2}{\mu_l^S(t) \mathcal{U}_l(T_l^*(t))} \tag{12}$$

where $k \neq l$, and $k, l \in V_S^{(L+1)}$. By replacing $\mathcal{U}_k(T_k(t))$ with $\frac{\lambda(T_k(t) - T_k^0)}{D_k^0(T_k(t) - T_k^0) + \mu_k^S(t)}$ (please refer to Equation (5)), we get:

$$\frac{\alpha_k \mu_k^S(t)}{(D_k^0 T_k(t) + \mu_k^S(t)) T_k(t)} = \frac{\alpha_l \mu_l^S(t)}{(D_l^0 T_l(t) + \mu_l^S(t)) T_l(t)} \tag{13}$$

We know that $\sum T_k(t) = \Psi(t)$. Hence, from Equation (13), we get that —

$$T_l(t) = \frac{1}{2\alpha_k \mu_k^S(t) D_l^0} \left(-\alpha_k \mu_k^S(t) \mu_l^S(t) + \sqrt{\Phi_{kl}} \right) \tag{14}$$

where $T_l(t) > 0$, $D_l^0 > 0$, $\zeta_k = (D_k^0 T_k(t) + \mu_k^S(t))$, and $\Phi_{kl} = \left[(\alpha_k \mu_k^S(t) \mu_l^S(t))^2 - 4\alpha_k \alpha_l \mu_k^S(t) \mu_l^S(t) \zeta_k T_k(t) D_l^0 \right]$.

Algorithm 1 Data Flow Management in FlowMan

INPUTS:
$\Psi(t)$, ϵ, $\cup_{n \in V_I^L} F_n^E(t)$, $\cup_{n \in V_I^L} F_n^M(t)$, R_k^{max}, $\boldsymbol{\alpha}$, $V_S^{(L+1)}$, $\boldsymbol{\mu}^S(t)$, $\boldsymbol{D}^0(t)$
OUTPUT:
$F_k^e(t)$, $F_k^m(t)$, $\forall k \in V_S^{(L+1)}$
PROCEDURE:
$l \leftarrow 0$ and $h \leftarrow \Psi(t)$
do
  $T_k(t) \leftarrow \frac{l+h}{2}$
  for each $l \in V_S^{(L+1)}$ and $l \neq k$ **do**
    Calculate $T_l(t)$ using Equation (14) ▷ Obtained from the proposed Nash bargaining game-theoretic model
  end for
  if $\sum T_k(t) \leq \Psi(t)$ **then**
    $l \leftarrow T_k(t)$
  else
    $h \leftarrow T_k(t)$
  end if
while $(h - l) < \epsilon$
$\mathcal{F}(t) \leftarrow (\cup_{n \in V_I^L} F_n^E(t)) \cup (\cup_{n \in V_I^L} F_n^M(t))$
for each $k \in V_S^{(L+1)}$ **do**
  $F_k^e(t), F_k^m(t) \leftarrow$ `FlowMan-Knapsack` $(T_k(t), \mathcal{F}(t))$
  $\mathcal{F}(t) \leftarrow \mathcal{F}(t) / (F_k^e(t) \cup F_k^m(t))$
end for
return $F_k^e(t)$, $F_k^m(t)$, $\forall k \in V_S^{(L+1)}$

F. Proposed Algorithm

From Section IV-E, we observe that, in SDN, the controller can ensure efficient load distribution using the proposed scheme, FlowMan, while ensuring that the data flows are fairly distributed among the switches. The controller needs to calculate the optimal data rate $T_k^*(t)$ for any switch k using a heuristic approach. In FlowMan, we use bisection method [28] having a fixed tolerance $\epsilon > 0$, as mentioned in Algorithm 1. Bisection method is used to evaluate the optimal value for $T_k^*(t)$ as the upper and lower bounds of the utility functions are known for the time instant t. Thereafter, using Equation (14), which is derived based on the utility function $\mathcal{U}_k(T_k(t))$, the controller calculates the optimal data rate $T_l^*(t)$ for each

switch l, where $k \neq l$. Finally, the controller uses a bounded Knapsack solution method [29], named *FlowMan-Knapsack*, to decide the optimal distribution of data flows for each switch. The workflow diagram of FlowMan is presented in Figure 3.

G. Complexity Analysis

In FlowMan, there are two methods which need to be executed sequentially. In the first method, i.e., the bisection method, the controller tries to find the optimal value for $T_k(t)$. Based on that,in the second method, the optimal data traffic associated with other switches are calculated using the bounded knapsack method. The complexity of the bisection method is $O(|V_S^{(L+1)}|)$, whereas, the complexity for bounded Knapsack method, i.e., FlowMan-Knapsack, is $O(H|V_S^{(L+1)}|)$, where $H = \max\{R_k(t)|\forall k \in V_S^{(L+1)}\}$. Hence, the overall complexity of the proposed scheme, FlowMan, is $O(H|V_S^{(L+1)}|)$, as $H \geq 1$. Thus, FlowMan has a complexity of $O(HC)$ for the overall network, where $C = \max\{|V_S^L|, \forall L\}$. Hence, we argue that with the increases in the number of switches in each layer, the complexity of FlowMan increases. However, the complexity of FlowMan does not get affected by the increase in the number of layers in the network.

V. Performance Evaluation

In this section, we analyze the performance of the proposed scheme, FlowMan, through simulation by varying the number of heterogeneous flows, i.e., elephant and mice flows, and the number of available switches.

A. Simulation Parameters

We simulated the proposed scheme, FlowMan, on Python3-based simulation platform. We considered that the switches and IoT devices are deployed randomly over the terrain of $10 \times 10\ km^2$, and each IoT device generates a single flow. We considered that each flow generates data traffic at a random rate. We considered that the threshold datarate is 0.1 million packets per second ($mpps$) to separate the flows into two categories – elephant and mice flows, as mentioned in Table I. Additionally, we varied the number elephant flows, i.e., 5–10% of the total flows [1]. We also considered that each switch has an infinite buffer size. We performed the simulation for 50 independent iterations. Each iteration is executed for 100 simulation seconds.

B. Benchmark

The performance of the proposed scheme, FlowMan, is evaluated by comparing with existing schemes — FlowStat [31] and CURE [32]. In FlowStat, Bera *et al.* [31] proposed a flow-rule placement scheme based on the per-flow statistics. In this work, the authors tried to accommodate the maximum number of flows while finding an end-to-end path at once. We consider FlowStat as it deals with similar problems such as forwarding switch selection for heterogeneous flows and flow-rules placement to maximize the throughput of the network. On the other hand, in CURE, Maity *et al.* [32] proposed a scheme for flow management. The authors considered that the flow-rules are updated according to the priority of the switches. We consider CURE as it deals with the generic problems of rule placement in SDN such as rule update while ensuring the high throughput of the network. In contrast to these existing schemes, FlowMan considers heterogeneous flows and aims to ensure high QoS, i.e., high network throughput and low delay, in SDN. Additionally, FlowMan reduces an NP-hard problem to an NP-complete problem with the help of the generalized Nash bargaining game within a finite time duration and ensures a Pareto optimal flow distribution among the switches in SDN.

TABLE I: Simulation Parameters

Parameter	Value
Number of SDN switches	$5, 10, 15$
Number of flows	$5000, 10000, 20000$
Number of elephant flows	5–10%
Data generation rate per mice flow	(0–0.1) $mpps$
Data generation rate per elephant flow	[0.1–0.5] $mpps$
Maximum flow rules /switch	4000–8000 [30]
Simulation duration	100 sec

C. Performance Metrics

We evaluated the performance of the proposed scheme, FlowMan, using the following metrics:

1) Per-Flow Throughput: Per-flow throughput is calculated as the average amount of data processed for each flow over a certain duration, individually. In the presence of heterogeneous flows, per-flow throughput not only depends on the number of flow-rules installed at the switch but also on the data-rate associated with each flow.

2) Network Throughput: Network throughput is calculated as the amount of data processed cumulatively by the switches available in the network. It depends on the flow-association of each switch and the data-rate of each associated flow.

3) Per-Flow Delay: Per-flow delay is calculated as the queuing delay and the processing delay incurred by each flow at the associated switch. Due to the presence of heterogeneous flows, unbalanced data traffic results in high delay per-flow for the mice flows, when the elephant flows and mice flows are associated with the same switch.

4) Network Delay: Network delay is calculated as the end-to-end delay incurred by the flows available in the network. We argue that, with balanced load distribution, the network delay can be minimized significantly.

D. Results and Discussions

From Figure 4, we observe that the per-flow throughput decreases with the increase in the number of flows, due to the increase in the number of elephant flows. However, using FlowMan, the per-flow throughput remains higher than using FlowStat and CURE, as FlowMan follows a Pareto optimal data-rate distribution for data flow management. We argue that FlowMan distributes the incoming flows with heterogeneous data-rate among the available switches, optimally. Hence, we yield that, using FlowMan, per-flow throughput increases by

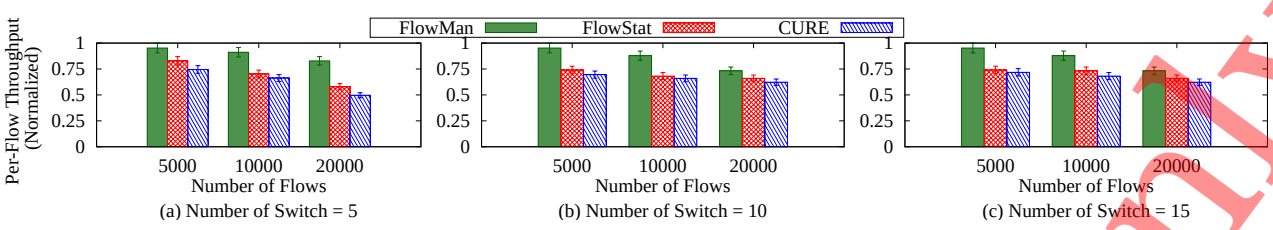

Fig. 4: Per-Flow Throughput Analysis

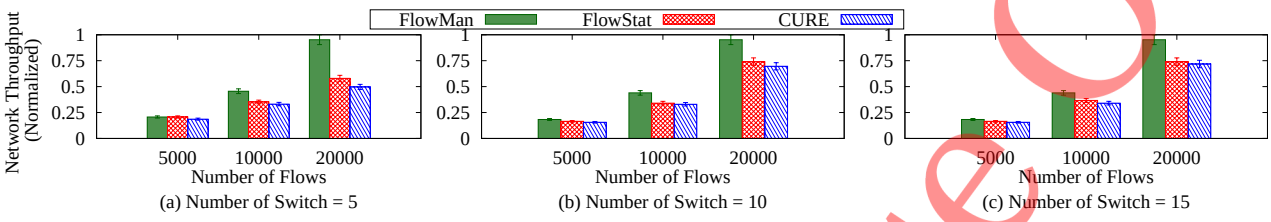

Fig. 5: Network Throughput Analysis

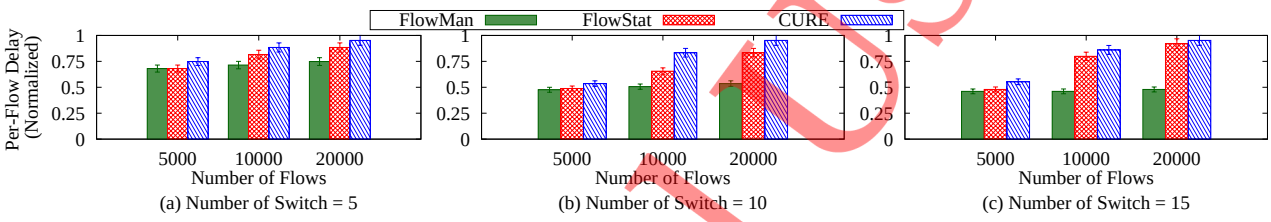

Fig. 6: Per-Flow Delay Analysis

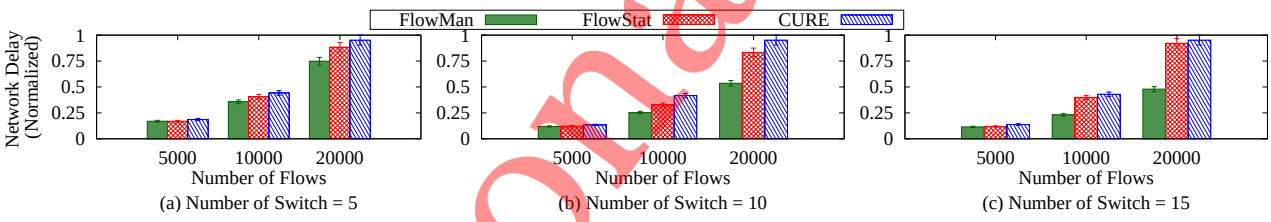

Fig. 7: Network Delay Analysis

24.6–47.8%. On the other hand, Figure 5 depicts that the network throughput increases significantly using FlowMan than using FlowStat and CURE. The network throughput depends on the elephant flows, as well as on the mice flows. We observe that, unlike FlowMan, the elephant flows get associated with a subset of switches using FlowStat and CURE, though these flows contribute almost 80% of the overall flow [1].

On the other hand, Figure 6 depicts that per-flow delay decreases by 27.7% using FlowMan than using FlowStat and CURE. In FlowMan, the traffic associated with each switch is optimal, hence there is no significant change in the delay of the associated flows. However, using FlowStat and CURE, the per-flow delay increases significantly due to the fact that these schemes do not take into consideration the presence of heterogeneous flows. We yield that, FlowMan reduces the queuing delay by 77.8–98.7%. Similarly, Figure 7 depicts that the network delay increases exponentially with the increase in the number of flows. This is due to the fact that queuing delay has a significant impact on the overall network delay. Hence, we argue that FlowMan ensures Pareto optimal flow distribution among the switches. Thereby, the queuing delay per-flow reduces significantly, which, in turn reduces the overall network delay. Additionally, through theoretical analysis, we observe that using FlowMan, the flow-setup delay remains constant for a fixed set of switches. However, using the existing schemes – FlowStat and CURE, the flow-setup delay increases linearly with the increase in the number of flows in the network. Therefore, we argue that FlowMan enhances the performance of network universally, i.e., enhances the network QoS, while ensuring the per-flow QoS is maintained

VI. Conclusion

In this paper, we argued that in the presence of heterogeneous flows, QoS-aware data flow management in SDN is NP-hard. Therefore, we proposed a novel scheme, named FlowMan, to address the aforementioned problem. In FlowMan, we used the generalized Nash bargaining game to obtain a Pareto optimal data-rate distribution for the switches. Thereby, we achieved a sub-optimal problem which can be mapped to the bounded Knapsack problem, i.e., an NP-complete problem.

Thereafter, we used a heuristic approach to solve the reduced sub-optimal problem. Additionally, we analyzed that the proposed scheme, FlowMan, follows the axioms of the generalized Nash bargaining game. Through simulations, we observed that FlowMan outperforms the existing benchmark schemes — CURE and FlowStat, while ensuring high throughput and low delay. In particular, FlowMan reduces network delay by 77.8–98.7% and increases network throughput by 24.6–47.8%, than using FlowStat and CURE.

Future extension of this work includes designing a data flow management scheme for broadcasting while minimizing the length of the routing path of the flows. We argue that the aforementioned problem can be mapped to the *traveling salesman problem*, which is an NP-hard problem. Hence, obtaining a Pareto optimal solution is challenging. Additionally, this work can be extended while incorporating the possibility of oversubscription in the presence of switches with limited flow-table capacity.

REFERENCES

[1] R. Trestian, K. Katrinis, and G. Muntean, "OFLoad: An OpenFlow-Based Dynamic Load Balancing Strategy for Datacenter Networks," *IEEE Trans. on Net. Serv. Man.*, vol. 14, no. 4, pp. 792–803, Dec 2017.

[2] D. Kreutz, F. M. V. Ramos, P. E. Veríssimo, C. E. Rothenberg, S. Azodolmolky, and S. Uhlig, "Software-Defined Networking: A Comprehensive Survey," in *Proc. of the IEEE*, vol. 103, no. 1, January 2015, pp. 14–76.

[3] A. Mondal, S. Misra, and A. Chakraborty, "TROD: Throughput-Optimal Dynamic Data Traffic Management in Software-Defined Networks," in *Proc. of IEEE Globecom Workshops*, Dec 2018, pp. 1–6.

[4] P. T. Congdon, P. Mohapatra, M. Farrens, and V. Akella, "Simultaneously Reducing Latency and Power Consumption in OpenFlow Switches," *IEEE/ACM Trans. on Net.*, vol. 22, no. 3, pp. 1007–1020, June 2014.

[5] N. P. Katta, J. Rexford, and D. Walker, "Incremental Consistent Updates," in *Proc. of ACM SIGCOMM Wrkshp*. New York, NY, USA: ACM, 2013, pp. 49–54.

[6] S. Bera, S. Misra, and N. Saha, "DynamiTE: Dynamic Traffic Engineering in Software-Defined Cyber Physical Systems," in *Proc. of IEEE ICC Workshops*, May 2018, pp. 1–6.

[7] Federal Communications Commission (FCC). (2019, Aug.) Broadband Speed Guide. [Online]. Available: https://www.fcc.gov/reports-research/guides/broadband-speed-guide

[8] H. Huang, S. Guo, P. Li, B. Ye, and I. Stojmenovic, "Joint Optimization of Rule Placement and Traffic Engineering for QoS Provisioning in Software Defined Network," *IEEE Trans. on Comp.*, vol. 64, no. 12, pp. 3488–3499, Dec 2015.

[9] O. Rottenstreich, Y. Kanizo, H. Kaplan, and J. Rexford, "Accurate Traffic Splitting on SDN Switches," *IEEE J. on Sel. Areas in Comm.*, vol. 36, no. 10, pp. 2190–2201, Oct 2018.

[10] A. Mondal, S. Misra, and I. Maity, "Buffer Size Evaluation of OpenFlow Systems in Software-Defined Networks," *IEEE Syst. J.*, pp. 1–8, 2018.

[11] M. Moradi, Y. Zhang, Z. Morley Mao, and R. Manghirmalani, "Dragon: Scalable, Flexible, and Efficient Traffic Engineering in Software Defined ISP Networks," *IEEE J.on Sel. Areas in Comm.*, vol. 36, no. 12, pp. 2744–2756, Dec 2018.

[12] Y. Sadeh, O. Rottenstreich, A. Barkan, Y. Kanizo, and H. Kaplan, "Optimal Representations of a Traffic Distribution in Switch Memories," in *Proc. of IEEE INFOCOM*, Apr 2019, pp. 2035–2043.

[13] S. Agarwal, M. Kodialam, and T. V. Lakshman, "Traffic Engineering in Software Defined Networks," in *Proc. of IEEE INFOCOM*, Apr 2013, pp. 1–9.

[14] S.-H. Tseng, A. Tang, G. L. Choudhury, and S. Tse, "Routing Stability in Hybrid Software-Defined Networks," *IEEE/ACM Trans. on Net.*, vol. 27, no. 2, pp. 790–804, Apr 2019.

[15] Z. Allybokus, K. Avrachenkov, J. Leguay, and L. Maggi, "Multi-Path Alpha-Fair Resource Allocation at Scale in Distributed Software-Defined Networks," *IEEE J. on Sel. Areas in Comm.*, vol. 36, no. 12, pp. 2655–2666, Dec 2018.

[16] D. Sanvito, I. Filippini, A. Capone, S. Paris, and J. Leguay, "Adaptive Robust Traffic Engineering in Software Defined Networks," in *Proc. of IFIP Networking and Workshops*, May 2018, pp. 1–9.

[17] H. Tahaei, R. B. Salleh, M. F. A. Razak, K. Ko, and N. B. Anuar, "Cost Effective Network Flow Measurement for Software Defined Networks: A Distributed Controller Scenario," *IEEE Access*, vol. 6, pp. 5182–5198, 2018.

[18] B. Grkemli, S. Tatlicioglu, A. M. Tekalp, S. Civanlar, and E. Lokman, "Dynamic Control Plane for SDN at Scale," *IEEE J. on Sel. Areas in Comm.*, vol. 36, no. 12, pp. 2688–2701, Dec 2018.

[19] C. R. Meiners, A. X. Liu, and E. Torng, "Bit Weaving: A Non-Prefix Approach to Compressing Packet Classifiers in TCAMs," *IEEE/ACM Trans. on Net.*, vol. 20, no. 2, pp. 488–500, April 2012.

[20] B. Mao, F. Tang, Z. M. Fadlullah, and N. Kato, "An Intelligent Route Computation Approach Based on Real-Time Deep Learning Strategy for Software Defined Communication Systems," *IEEE Trans. on Emerg. Topics in Comp.*, pp. 1–12, 2019.

[21] J. C. Mogul, J. Tourrilhes, P. Yalagandula, P. Sharma, A. R. Curtis, and S. Banerjee, "DevoFlow: Cost-effective Flow Management for High Performance Enterprise Networks," in *Proc. of ACM SIGCOMM Wrksp*, New York, NY, USA, 2010, pp. 1–6.

[22] O. Rottenstreich, I. Keslassy, Y. Revah, and A. Kadosh, "Minimizing Delay in Network Function Virtualization with Shared Pipelines," *IEEE Trans. on Par. and Dist. Syst.*, vol. 28, no. 1, pp. 156–169, Jan 2017.

[23] 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.

[24] M. Hayes, B. Ng, A. Pekar, and W. K. G. Seah, "Scalable Architecture for SDN Traffic Classification," *IEEE Syst. J.*, pp. 1–12, 2017, DOI: 10.1109/JSYST.2017.2690259.

[25] O. Rottenstreich and J. Tapolcai, "Lossy Compression of Packet Classifiers," in *Proc. of ACM/IEEE ANCS*, May 2015, pp. 39–50.

[26] H. Park and M. van der Schaar, "Bargaining Strategies for Networked Multimedia Resource Management," *IEEE Trans. on Sig. Proc.*, vol. 55, no. 7, pp. 3496–3511, July 2007.

[27] A. Drexl, "A Simulated Annealing Approach to the Multiconstraint Zero-One Knapsack Problem," *Computing*, vol. 40, no. 1, pp. 1–8, Mar 1988.

[28] R. V. Driessche and D. Roose, "An Improved Spectral Bisection Algorithm and Its Application to Dynamic Load Balancing," *Parallel Computing*, vol. 21, no. 1, pp. 29 – 48, 1995.

[29] V. Poirriez, N. Yanev, and R. Andonov, "A Hybrid Algorithm for the Unbounded Knapsack Problem," *Discrete Optimization*, vol. 6, no. 1, pp. 110 – 124, 2009.

[30] OpenFlow. (2014, December) OpenFlow Switch Specification Version 1.5.0. Open Networking Foundation. [Online]. Available: https://www.opennetworking.org/images/stories/downloads/sdn-resources/onf-specifications/openflow/openflow-switch-v1.5.0.noipr.pdf

[31] S. Bera, S. Misra, and A. Jamalipour, "FlowStat: Adaptive Flow-Rule Placement for Per-Flow Statistics in SDN," *IEEE J. on Sel. Areas in Comm.*, vol. 37, no. 3, pp. 530–539, March 2019.

[32] I. Maity, A. Mondal, S. Misra, and C. Mandal, "CURE: Consistent Update with Redundancy Reduction in SDN," *IEEE Trans. on Comm.*, vol. PP, no. 99, pp. 1–8, 2018.