doi: 10.1109/ICCW.2019.8756718

D2M: Mobility-Aware Dynamic Data Multicasting in Software-Defined Data Center Networks

Sudip Misra and Ayan Mondal
Department of Computer Science and Engineering
Indian Institute of Technology Kharagpur, India
Email: {sudipm, ayanmondal}@iitkgp.ac.in

Pankaj Kumar
Department of Computer Science and Engineering
National Institute of Technology Patna, India
Email: pankajkumar1406094@nitp.ac.in

***Abstract*—In this paper, we study the problem of mobility-aware dynamic data multicasting in software-defined Data Center Networks (DCNs) in the presence of Internet of Things (IoT) devices. In the existing literature, researchers studied that due to unbalanced traffic distribution in DCN, the network throughput and delay degrade significantly. Moreover, with the integration of heterogeneous IoT devices, the difficulty in achieving balanced traffic distribution increases. Hence, there is a need to design an efficient data multicasting scheme in DCN, while integrating the software-defined network architecture. In this work, we propose a Dynamic Data Multicasting scheme, named D2M, using single leader multiple follower Stackelberg game for ensuring high utilization of network capacity and efficient load balancing. In D2M, the controller acts as the leader, and the switches act as the followers. The leader decides the distribution of flow-rules among the switches for ensuring efficient load-balancing. We represent bandwidth distribution as a pseudo-Cournot competition, where each follower decides the optimal bandwidth to be allocated for each flow. The existence of at least one Nash equilibrium using D2M is ensured. Through simulation, we observed that using D2M, the network throughput increased by 6.13-95.32% than using existing schemes, while ensuring 21.32-99.29% reduction in delay.**

***Index Terms*—Data Multicasting, Mobile IoT Devices, Software-Defined Data Center Networks, Load Balancing, Network Capacity, Stackelberg Game**

I. Introduction

In the last decades, with the increase in the number of Internet of Things (IoT) devices, the amount of generated data, named 'big-data', increased significantly [1]. Traditionally, big-data is processed in Data Center Networks (DCNs) formed by interconnecting multiple data centers. Hence, there is a need to integrate IoT devices into the DCN architecture. Moreover, due to the increase in the number of IoT devices, the number of flows in the network increases. Hence, in this work, we consider software-defined DCN for ensuring flow-based data traffic management in DCN. In the existing literature, researchers focused on designing schemes for traffic distribution in traditional DCNs. Additionally, few works, viz. [2], considered data broadcasting among IoT devices. However, there is a need for designing data multicasting schemes for DCNs in the presence of mobile IoT devices. In this work, we consider the fat-tree based software-defined DCN architecture in order to ensure a reduction in blocking probability [3].

A. Motivation

In the existing literature, most researchers consider the fat-tree architecture for DCNs, in order to ensure multiple paths having equal-cost between any pair of hosts [4] and high bandwidth inter-connectivity. However, unbalanced traffic distribution is one of the important problems in fat-tree DCNs. In the existing literature, researchers proposed different scheduling techniques for data unicasting [5] and multicasting [3] in fat-tree DCNs. However, there exists no scheme for data multicasting scheme in software-defined fat-tree DCNs. Additionally, in the presence of IoT devices [6], multicasting data in real-time is a challenge [2], which needs to be addressed in fat-tree-based software-defined DCNs. In this work, we consider that the controller decides the flow-rule association matrix for efficient load balancing. On the other hand, the switches decide the optimum amount of bandwidth to be allocated to each flow for high utilization of network capacity.

B. Contribution

In this paper, we use a single leader multiple follower Stackelberg game for designing dynamic data multicasting scheme, named D2M, in software-defined DCN. In D2M, the controller takes strategic decision for rule placement among the switches while ensuring efficient load balancing. On the other hand, the switches are responsible for processing the data packets. Additionally, we consider that the switches are capable of bisecting the capacity into multiple resource blocks, where a subset of resource blocks needs to be allocated for each flow. Thereby, in D2M, we consider that the controller acts as the leader and the switches act as the followers. In summary, the specific contributions of this paper are as follows:

a) We present the D2M scheme for dynamic data multicasting in real-time in fat-tree based software-defined DCNs with mobile IoT devices.

b) We use single leader multiple follower Stackelberg game in order to ensure high utilization of network capacity and efficient load balancing, where efficient bandwidth distribution problem is visualized as a *pseudo-Cournot competition*.

c) We present two different algorithms. The first algorithm is used for rule placement among the switches while ensuring efficient load balancing. Thereafter, using the second algorithm, each switch decides the optimal amount of bandwidth to be allocated for each flow.

II. Related Work

In recent years, a number of research works studied big-data processing and data delivery in DCNs. Some of the existing literature are discussed in this section. Chen *et al.* [7] surveyed the challenges in generation, acquisition, storage, and processing of data. They also mentioned the applications involving big-data such as — enterprise management, IoT, and social networks, while including different medical applications and smart grid. Lakhlef *et al.* [2] proposed agent-based broadcast protocols for mobile IoT devices, while considering parallel data broadcasting with limited channels. In another review article, Jagadish *et al.* [8] cataloged different challenges for understanding big-data while citing a case study about cleaning, analyzing, and interpretation of data or information. Paul *et al.* [9] studied the optimal server provisioning problem in DCN and proposed two different schemes — for minimizing operational cost and for minimizing capital and operational cost, jointly, based on a discrete-time model. Wu *et al.* [10] proposed a big-data broadcasting scheme for distributed system. The authors considered that the source device has the maximum bandwidth or capacity, and modeled the network as a Lock-Step Broadcast Tree (LBST).

On the other hand, a few research works studied in data unicasting and multicasting in fat-tree based DCNs. Raiciu *et al.* [11] proposed a Multipath Transmission Control Protocol (MPTCP) in DCN for data unicasting. Chiu and Lau [12] proposed a scheme for efficient multicast broadcast services using transmitter-side channel state information. Al-Fares *et al.* [13] proposed a Dynamic Flow Scheduling (HEDERA) scheme for data multicasting, while aggregating network resources. Guo and Yang [3] studied multicasting in DCNs with fat-tree topology. The authors claimed that their work is one of the pioneering work while exploring multicasting in fat-tree based DCNs. However, these works do not consider multicasting in software-defined fat-tree DCNs, where the controller considers each flow separately and installs the flow-rules in the SDN switches while ensuring reduced network delay.

***Synthesis**:* Thus, we infer that there exist a few works on big-data processing and data broadcast in DCN. Additionally, there are few works on data unicasting and multicasting in traditional DCN. Additionally, using the multicasting approaches proposed for the traditional DCN, optimal throughput with an optimal delay in the network cannot be ensured due to the presence of heterogeneous IoT devices. Hence, there is a need for designing a multicasting scheme for proper utilization of available bandwidth in software-defined DCN, while maximizing network throughput and minimizing the network delay.

III. System Model

In this paper, we consider a software-defined DCN, where DCN follows the fat-tree architecture. A fat-tree is a network architecture having three tiers — *core*, *aggregation*, and *edge*. The routers at the core-tier are responsible for connecting the intra-networks. We consider that the routers have enough bandwidth to support the connected switches at the aggregation-tier, as shown in Figure 1. On the other hand,

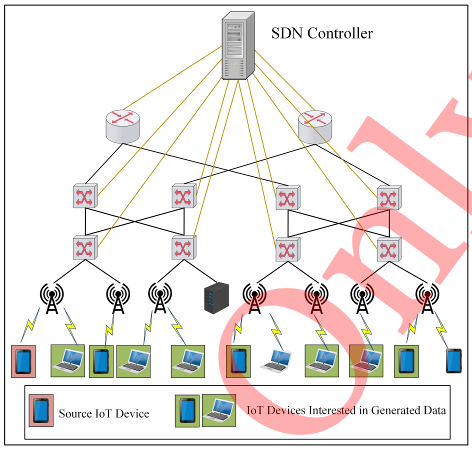


Fig. 1: Schematic Diagram of Fat Tree-based Software-define DCN with IoT Devices

the switches are responsible for providing communication services to the end-devices at the edge-tier. Additionally, we consider that the switches and the routers in the fat-tree DCN have limited ternary content-addressable memory (TCAM). In other words, the flow-tables have limited number of flow-entries. We consider that each switch $s \in \mathcal{S}$, where $\mathcal{S}$ is the set of switches, provides services to $\mathcal{F}_s(t)$ set of heterogeneous flows at time instant t. Considering that each IoT device generates at most one flow at time instant t, we have — $|\mathcal{N}_s(t)| = |\mathcal{F}_s(t)|$, where $\mathcal{N}_s(t) \subseteq \mathcal{N}$, and $\mathcal{N}_s(t)$ and $\mathcal{N}$ denote the set of active devices in data generation and the available heterogeneous IoT devices, respectively. These heterogeneous IoT devices are mobile in nature and connected wirelessly with the switches at the aggregation-tier through access points (APs). In addition to the IoT devices, the edge-tier includes data-servers. These data-servers are connected with the switches at the aggregation-tier using d wired connection, as shown in Figure 1.

Additionally, we consider that each SDN switch $s \in \mathcal{S}$ has a capacity of C_s (in *kbps*). We consider that each flow $f_n \in \cup\mathcal{F}_s(t)$, where $n \in \mathcal{N}$, has a minimum data-rate requirement which is denoted as r_n^{min} (in *kbps*). The maximum data-rate of the flows generated by the heterogeneous IoT devices are constrained by hardware limitations. We denote the maximum data-rate of flow f_n as r_n^{max} (in *kbps*). Hence, the actual data-rate $r_n(t)$ (in *kbps*) of each flow f_n has to satisfy the following constraint:

$$r_n^{min} \leq r_n(t) \leq r_n^{max} \tag{1}$$

Therefore, at time instant t, the number of IoT devices which are getting served by each switch s, needs to satisfy the following constraint:

$$C_s^{rem} \geq \sum_{f_n \in \mathcal{F}_s(t)} r_n(t), \quad \forall s \in \mathcal{S} \tag{2}$$

where C_s^{rem} is the remaining capacity of switch s after allocating bandwidth to the data center servers. Considering that each flow f_d associated with data-server $d \in \mathcal{D}$, where $\mathcal{D}$ is the set of data-servers in the software-defined DCN, handles data-rate $r_d(t)$, we define C_s^{rem} as follows:

$$C_s^{rem} = C_s - \sum_{f_d \in \mathcal{F}_s(t)} r_d(t) \tag{3}$$

We consider that a set IoT devices $\overline{\mathcal{N}}(t) \subseteq \mathcal{N}$ generates a set of flows $\overline{\mathcal{F}}(t)$ at time instant t.Therefore, we have — $\overline{\mathcal{F}}(t) = \bigcup_s \mathcal{F}_s(t)$. Additionally, we define an *associative* variable x_{ns}, which is defined as follows:

$$x_{ns} = \begin{cases} 1, & \text{if } f_n \in \mathcal{F}_s(t) \\ 0, & \text{otherwise} \end{cases} \tag{4}$$

Proposition 1. *For each flow $f_n \in \overline{\mathcal{F}}(t)$, the following condition is true:*

$$1 \leq \sum_{s \in \mathcal{S}} x_{ns} \leq 4 \tag{5}$$

Proof. In the software-defined DCN, we argue that flow-rule associated with each flow f_n needs to be installed at least *one* SDN switch in case of intra-pod communication. On the other hand, the flow f_n needs to be matched at most *four* intermediate SDN switches, as we have considered two-tier fat-tree architecture. Therefore, we argue that the condition mentioned in Equation (5) is true. □

A. Assumptions

While designing the proposed scheme, D2M, we considered the following assumptions:

(i) We consider that each IoT device is always connected with one the of access points available in the software-defined DCN.

(ii) We consider that the centralized controller controls the flow-rules to be installed at the switches.

(iii) The controller can change its strategy of choosing an optimal source node at any given instant if it finds a source node with high payoff.

(iv) Each IoT device, which is downloading data, is interested in a single flow.

IV. Dynamic Data Multicasting (D2M) Scheme

A. Game Formulation

For modeling the interaction between the SDN switches and the controller, we use single leader multiple follower Stackelberg game. In the proposed scheme, D2M, we consider that the controller acts as the leader and installs the flow-rules in the SDN switches. Additionally, the controller decides the source node of the flow for each destination. On the other hand, the SDN switches, which act as the followers, decide their respective strategy, non-cooperatively. The followers help the controller to manage the network properly while deciding the amount of bandwidth to be allocated for each flow and optimize the usage of overall capacity. The proposed scheme, D2M, is formulated as a *pseudo-Cournot competition*. The components in the D2M are as follows:

(i) The controller acts as the leader. It decides the optimal route of for each flow and flow-rules to be installed to which switches.

(ii) Each SDN switch act as a follower. The switches make a trade-off between the associated flow-rules and the bandwidth allocated for each flow.

(iii) Each switch tries to maximize its satisfaction factor, which is defined in Definition 1 while maximizing the network throughput.

(iv) We consider that each flow f_n, which is generated by IoT device n, comprises of M_n amount of data (in Kb).

Definition 1. *The satisfaction factor $\rho_s(t)$ of each switch s is defined as the ratio of amount of bandwidth allocated to $\mathcal{F}_s(t)$ flows and the capacity C_s of the switch. Mathematically,*

$$\rho_s(t) = \frac{\sum_{f_n \in \mathcal{F}_s(t)} r_n(t) + \sum_{f_d \in \mathcal{F}_s(t)} r_d(t)}{C_s} \tag{6}$$

1) Justification for Single Leader Multiple Follower Stackelberg Game: In SDN, the tasks – network control and packet processing – are divided into two planes – control and data planes, respectively. The switches contain the data plane and inform the controller if there is a table-miss. On the other hand, the controller, which is associated with the control plane, takes care of the flow routing and updates the flow-tables at the switches. Hence, we consider that software-defined DCN follows a leader-follower architecture. Thereby, in order to model the interactions among the controller and the switches, we consider single leader multiple follower Stackelberg game theoretic approach. In the proposed scheme, D2M, the controller decides the optimal routing path among the source-destination pair, for ensuring delay-optimal data multicasting. Thereafter, the switches decide the amount of bandwidth to be allocated for each flow while ensuring the throughput-optimal data multicasting. Hence, data multicasting in software-defined DCN resembles an oligopolistic market scenario. Hence, we argue that a single leader multiple follower Stackelberg game is the most suitable approach for data multicasting in software-defined DCN.

2) Utility Function of Controller: The utility function $\mathcal{U}_c(\cdot)$ of the controller signifies the overall network delay for multicasting while maximizing the network lifetime. The controller decides the source node for each flow in multicasting and the routing path while maximizing the payoff value of utility function $\mathcal{U}_c(\cdot)$. The utility function $\mathcal{U}_c(\cdot)$ needs to satisfy the following properties:

(i) With the increase in hop-count, the payoff value decreases.

(ii) With the increase in residual energy of the source IoT device, payoff value increases.

(iii) If the source node selected for a flow associated with a destination is serving other flow(s) associated with other destination(s), the payoff value increases.

Therefore, we consider that the utility function $\mathcal{U}_c(\cdot)$ of the controller is a linear function and is defined as follows:

$$\mathcal{U}_c(\cdot) = \sum_{s \in \mathcal{S}} \sum_{f_n \in \mathcal{F}_s(t)} \left(\alpha_n \frac{E_n^{res}(t)}{E_{max}} + \beta_n \frac{h_{f_n}(t)}{4} + \gamma_n a_n(t) \right) \tag{7}$$

where E_{max} and $E_n^{res}(t)$ denote the maximum and residual energy of IoT device n at time instant t; $h_{f_n}(t)$ denotes the hop-count associated with flow f_n; and $a_n(t)$ is a binary variable and denotes the state of IoT device s. We defined $a_n(t)$ in Definition 2. In Equation (7), α_n, β_n, and γ_n are constants specified for flow f_n. These constants ensure that the associated coefficients have similar numeric impact on the payoff value of utility function $\mathcal{U}_c(\cdot)$. The controller tries to maximize $\mathcal{U}_c(\cdot)$, while ensuring the constants mentioned in Equations (2) and (5).

Definition 2. *We consider that each IoT device n have two states — idle and active. Additionally, we consider that the IoT devices at idle state has zero energy consumption, whereas the energy consumption of the IoT devices in active state is finite and positive. Therefore, the binary variable $a_n(t)$ of IoT device n is defined as follows:*

$$a_n(t) = \begin{cases} 1, & \textit{if IoT device } n \textit{ is in active state} \\ 0, & \textit{otherwise} \end{cases} \tag{8}$$

3) Utility Function of Each Switch: The utility function $\mathcal{U}_s(\cdot)$ of each switch s signifies the utilization of capacity of the switch, and the flow-associated delay. Each switch decides the amount of bandwidth to be allocated to each flow $f_n \in \mathcal{F}_s(t)$, while maximizing the payoff value of the utility function $\mathcal{U}_s(\cdot)$. Hence, $\mathcal{U}_s(\cdot)$ must satisfy the following properties:

(i) With the increase in the satisfaction factor $\rho_s(t)$, the payoff value increases.

(ii) With the increase in each flow-associated delay, the payoff value decreases.

(iii) With the increase in overall delay associated with $\mathcal{F}_s(t)$ flows, the payoff value decreases.

Therefore, we define the utility function $\mathcal{U}_s(\cdot)$ as follows:

$$\mathcal{U}_s(\cdot) = \zeta_s \sum_{f_n \in \mathcal{F}_s(t)} \frac{r_n(t)}{r_n^{max}} + \phi_s \sum_{f_n \in \mathcal{F}_s(t)} \frac{M_n}{r_n(t)} + \varphi_s \rho_s(t) \tag{9}$$

where ζ_s, ϕ_s, and φ_s are constants for switch s. These constants ensure that the associated coefficients have similar numeric impact on the payoff value of utility function $\mathcal{U}_s(\cdot)$. The first and second terms in Equation (9) signify the utilization of capacity per flow and the flow-associated delay, respectively. Hence, each switch aims to decide an optimal data-rate for each flow $f_n \in \mathcal{F}_S(t)$, while maximizing the payoff value of $\mathcal{U}_s(\cdot)$ and satisfying the constraints mentioned in Equations (1) and (2).

B. Existence of Nash Equilibrium

We consider the Nash equilibrium of the proposed scheme, D2M, as defined in Definition 3. We argue that at the Nash equilibrium point, the players cannot increase their payoff value by deviating from the equilibrium point.

Definition 3. *We define the Nash equilibrium point as a tuple of $r_n^*(t)$ and n^*, where n^* and $r_n^*(t)$ represent the optimum source node for flow f_n and the optimum bandwidth allocated to IoT device n^*, respectively. The Nash equilibrium point needs to satisfy the following constraints:*

$$\mathcal{U}_c(E_{n^*}^{res}(t), h_{f_{n^*}}(t), a_{n^*}(t), E_{-\boldsymbol{n}^*}^{res}(t), h_{f_{-\boldsymbol{n}^*}}(t), a_{-\boldsymbol{n}^*}(t)) \geq \mathcal{U}_c(E_n^{res}(t), h_{f_n}(t), a_n(t), E_{-\boldsymbol{n}^*}^{res}(t), h_{f_{-\boldsymbol{n}^*}}(t), a_{-\boldsymbol{n}^*}(t)) \tag{10}$$

$$\mathcal{U}_s(r_n^*(t), r_{-\boldsymbol{n}}^*(t)) \geq \mathcal{U}_c(r_n(t), r_{-\boldsymbol{n}}^*(t)) \tag{11}$$

where $a_{-\boldsymbol{n}^*}(t) = \{a_{k^*}(t) | \forall f_k \in \overline{\mathcal{F}}(t)$ and $k \neq n\}$. Similarly, we can define $E_{-\boldsymbol{n}^*}^{res}$ and $h_{f^*_{-\boldsymbol{n}}}(t)$. On the other hand, $r_{-\boldsymbol{n}}^*(t) = \{r_p^*(t) | \forall p \in \mathcal{F}_s(t)$ and $p \neq n\}$

We argue that the proposed scheme, D2M, follows a finite perfect information game theoretic approach. Additionally, the players in D2M follow pure strategies. Therefore, using the backward-induction method, we can ensure that there exists at least one Nash equilibrium point in the proposed scheme, D2M [14].

C. Solution of D2M Scheme

The controller selects its optimal strategy based on preference relation of the available strategies. For example, we consider there are two flows f_1 and f_2, and IoT devices n' and n'' have the data for multicasting, i.e., these devices are probable source nodes. We consider that the controller has a preference relation —

$$(f_1 \to n', f_2 \to n') \succeq (f_1 \to n'', f_2 \to n'') \succeq (f_1 \to n', f_2 \to n'') \succeq (f_1 \to n'', f_1 \to n') \tag{12}$$

based on the following information:

$$\mathcal{U}_c(\cdot)|_{(f_1 \to n', f_2 \to n')} \succeq \mathcal{U}_c(\cdot)|_{(f_1 \to n'', f_2 \to n'')} \succeq \mathcal{U}_c(\cdot)|_{(f_1 \to n', f_2 \to n'')} \succeq \mathcal{U}_c(\cdot)|_{(f_1 \to n'', f_1 \to n')} \tag{13}$$

On the other hand, each switch s decides the amount of bandwidth to be allocated for each flow $f_n \in \mathcal{F}_s(t)$. Using Karush-Kuhn-Tucker (KKT) condition, we get Equation (14), where $\lambda_{1,n}$, $\lambda_{2,n}$, where $f_n \in \mathcal{F}_s(t)$, and λ_3 are Lagrangian multipliers. Additionally, we have:

$$\lambda_{1,n}, \lambda_{2,n}, \lambda_3 \geq 0, \quad \forall f_n \in \mathcal{F}_s(t) \tag{15}$$

$$\left.\begin{array}{l} \lambda_{1,n}(r_n(t) - r_n^{min}) = 0 \\ \lambda_{2,n}(r_n(t) - r_n^{max}) = 0 \end{array}\right\}, \quad \forall f_n \in \mathcal{F}_s(t) \tag{16}$$

$$\lambda_3 \left(C_s - \sum_{f_n \in \mathcal{F}_s(t)} r_n(t) - \sum_{f_d \in \mathcal{F}_s(t)} r_d(t) \right) = 0 \tag{17}$$

Hence, performing $\nabla_{r_n(t)} \mathcal{L} = 0$, we get:

$$r_n^*(t) = [\frac{M_n}{\phi_s}]^{\frac{1}{2}} \left(\frac{\zeta_s}{r_n^{max}} + \frac{\varphi_s}{C_s} \right)^{-\frac{1}{2}} \tag{18}$$

$$\mathcal{L} = \mathcal{U}_s(t) - \sum_{f_n \in \mathcal{F}_s(t)} \lambda_{1,n}(r_n(t) - r_n^{min}) + \sum_{f_n \in \mathcal{F}_s(t)} \lambda_{2,n}(r_n(t) - r_n^{max}) - \lambda_3[C_s - \sum_{f_n \in \mathcal{F}_s(t)} r_n(t) - \sum_{f_d \in \mathcal{F}_s(t)} r_d(t)] \tag{14}$$

Additionally, we get that $\nabla^2_{r_n(t)}\mathcal{L} \leq 0$. Hence, we argue that at $r_n(t) = r^*_n(t)$, $\mathcal{U}_s(\cdot)$ is maximized.

Algorithm 1 Optimal Flow Association Vector

```
INPUTS:
 1: $\mathcal{N}$, $\overline{\mathcal{N}}(t)$, $\mathcal{F}(t)$, $\overline{\mathcal{F}}(t)$, $\mathcal{S}$
 2: $E^{res}_n(t)$, $E_{max}$, $h_{f_n}(t)$, $a_n(t)$, $\forall f_n \in \mathcal{F}$
 3: $\alpha_n$, $\beta_n$, $\gamma_n$
OUTPUT:
 1: $\{x_{ns}\}$, $\forall f_n \in \mathcal{F}$
PROCEDURE:
 1: Find Cartesian product of $\mathcal{F}$ and $\mathcal{S}$
 2: do
 3:     Chose a vector of $\{f_n, s|\forall f_n \in \mathcal{F}\}$
 4:     Calculate $\mathcal{U}_n(\cdot)$ using Equation (7);
 5: while each element in Cartesian set is not visited for $|\mathcal{S}|$ times;
 6: Calculate a preference relation among the calculated $\mathcal{U}_n(\cdot)$ values
 7: Select the vector with highest preference value
 8: return Corresponding $\{x_{ns}\}$, $\forall f_n \in \mathcal{F}$
```

Algorithm 2 Optimal Data-rate for Each Flow

```
INPUTS:
 1: $\mathcal{F}_s(t)$, $C_s$; $r_d(t)$, $\forall f_d \in \mathcal{F}_s(t)$; $r^{min}_n$, $r^{max}_n$, $M_n$, $\forall f_n \in \mathcal{F}_s(t)$
 2: $\zeta_s$, $\phi_s$, $\varphi_s$
 3: $\delta$                                    ▷ Increment factor
OUTPUT:
 1: $\{r^*_n(t)\}$, $\forall f_n \in \mathcal{F}_s(t)$
PROCEDURE:
 1: for each $f_n \in \mathcal{F}_s(t)$ do
 2:     $r^*_n(t) \leftarrow r^{min}_n$
 3: end for
 4: Calculate $\mathcal{U}_s(\cdot)$ using Equation (9)
 5: do
 6:     for each $f_n \in \mathcal{F}_s(t)$ do
 7:         $r_n(t) \leftarrow r^*_n + \delta$
 8:         if then$r_n(t) < r^{max}_n$
 9:             $\mathcal{U}^{prev}_s(\cdot) \leftarrow \mathcal{U}_s(\cdot)$
10:             Calculate $\mathcal{U}_s(\cdot)$ using Equation (9)
11:             if then$\mathcal{U}_s(\cdot) \geq \mathcal{U}^{prev}_s(\cdot)$
12:                 $r^*_n(t) \leftarrow r_n(t)$
13:             end if
14:         end if
15:     end for
16: while There is any change in $\{r^*_n(t)\}$, $\forall f_n \in \mathcal{F}_s(t)$
17: return $\{r^*_n(t)\}$, $\forall f_n \in \mathcal{F}_s(t)$
```

D. Proposed Algorithms

In the proposed scheme, D2M, initially, the controller decides the $x_{ns} \in \{0, 1\}$, $\forall f_n \in \mathcal{F}(t)$, and tries to minimize overall network delay by using Algorithm 1. Here, we assume that the controller has the knowledge that the IoT devices are connected to which access points (APs) and the corresponding SDN switches. Moreover, we consider that the controller knows the set of IoT devices which are interested in downloading the data. On the other hand, using Algorithm 2, each switch s decides the amount of bandwidth to be allocated to the associated flows $\mathcal{F}_s(t)$, while satisfying the constraints mentioned in Equations (2) and (3). Algorithm 2 needs to be executed by each switch, individually and non-cooperatively. The time complexity for Algorithm 1 is $O(\max(|\mathcal{F}|^{|\mathcal{S}|}, |\mathcal{S}|^{|\mathcal{F}|}))$. On the other hand, the time complexity for Algorithm 2 is $O(K)$, where $K \in \mathbb{R}^+$. Therefore, the over complexity of D2M is $O(\max(|\mathcal{F}|^{|\mathcal{S}|}, |\mathcal{S}|^{|\mathcal{F}|}) + K) \approx O(\max(|\mathcal{F}|^{|\mathcal{S}|}, |\mathcal{S}|^{|\mathcal{F}|}))$.

V. Performance Evaluation

A. Simulation Parameters

For the performance evaluation, we simulate using JAVA-based platform and deployed the IoT devices randomly over a terrain of $1000 \times 1000\ m^2$. However, the switches and the routers are deployed in a grid fashion, while ensuring full coverage. We consider that the source IoT devices generate 1000 number of data chunks, and the size of each data chunk is 800 Mb, as mentioned in Table I. Motivated by the device distribution of the Internet [10], we consider that the distribution of IoT device capacities follows the distribution mentioned in Table II.

TABLE I: Simulation Parameters

Parameter	Value
Simulation Area	$1000\ m \times 1000\ m$
Number of Nodes	1000-50000
Number of Switches	4
Number of Servers	3
Velocity of Source Nodes	$5\ m/s$
Capacity of Switches	10 $Gbps$
Data chunks generated	1000
Size of each data chunk	800 Mb
Mobility model (MM)	Random Gauss-Markov

TABLE II: Node Capacity Distribution [15]

Capacity $(Kbps)$	**Nodes** (%)	**Capacity** $(Kbps)$	**Nodes** (%)
128	20	384	40
1000	25	5000	15

B. Benchmarks

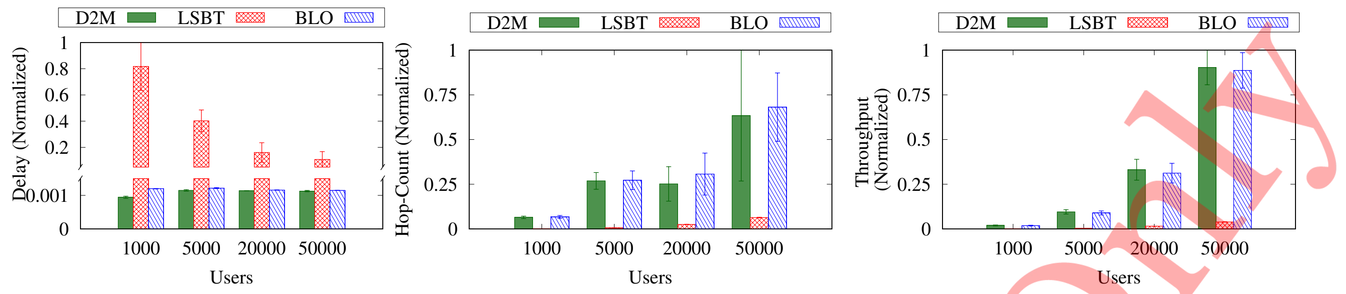
The performance of the proposed scheme, D2M, is evaluated while comparing with two existing schemes for DCNs — the Lock-Step Broadcast Tree based big-data broadcasting (LSBT) [10] and the Multicast Fat-Tree Data Center Networks (BLO) [3] schemes. In LSBT, Wu *et al.* [10] proposed a big-data broadcasting scheme, while forming a *Lock Step Broadcast Tree*. The authors also considered that the source device, which has the maximum capacity in the network, is at the root of the tree. On the other hand, in BLO [3], Guo and Yang proposed a fat-tree based DCN. In BLO, the authors tried to minimize the number of core switches needed to overcome the problem of over subscriptions. Additionally, the authors overlooked the problem of balanced bandwidth distribution. Moreover, these works do not consider the software-defined DCN in the presence of mobile IoT devices. In D2M, we improve the network performance for data multicasting, while ensuring optimal throughput and delay of the network.

C. Performance Metrics

We evaluate the performance of the proposed scheme, D2M, using the following metrics:

Network Delay: We define network delay as the total time required to complete the data multicasting in the software-defined DCN.

Hop-Count for Network Flows: We calculate the overall hop-count for the data multicasting. With the increase in hop-count, the real-timeliness of the multicasted data degrades.



(a) Network Delay (b) Hop-Count (c) Network Throughput

Fig. 2: Performance Analysis of D2M

Network Throughput: The network throughput signifies that the amount of data successfully transmitted from the source node to the destination. With efficient load balancing, network throughput increases significantly.

D. *Results and Discussions*

Figure 2(a) depicts that using D2M, the network delay decreases by 21.32% and 99.29% than using BLO and LSBT, respectively. In LSBT, delay increases significantly due to the fact that LSBT multicasts data in intra-networks. Hence, only after it reaches to a server at the edge-tier, the data gets multicasted in the inter-network. On the other hand, the network delay using BLO is higher than D2M due to inefficient network load balancing. Similarly, from Figure 2(b), we observed that the hop-count using LSBT is significantly low as the source IoT device multicasts data in the intra-network. Therefore, in fat-tree architecture, LSBT cannot ensure data transfer to each IoT device. However, we yield that using D2M, the hop-count reduces by 3.69-7.44% than using BLO. We argue that D2M ensures efficient load balancing in software-defined DCN.

On the other hand, from Figure 2(c), we observed that using D2M, the network throughput increases by 6.13% and 95.32% than using BLO and LSBT, respectively. As mentioned earlier, using LSBT, the source node can only multicast data in intra-network. Therefore, network throughput reduces significantly. The network throughput using D2M is comparable with the throughput achieved using BLO, as both schemes are designed for DCN with fat-tree architecture. However, as D2M is designed for software-defined DCN, the controller has an overview of the network, which BLO lacks in. Thereby, we argue that D2M ensures high utilization of network capacity while distributing the network load efficiently compared to the existing schemes — LSBT and BLO.

VI. Conclusion

In this paper, we formulated a single leader multiple follower Stackelberg game based D2M scheme to ensure high utilization of network capacity and efficient load balancing in software-defined DCN. We observed that D2M ensures the reduction in network delay in the presence of mobile IoT devices. Additionally, using D2M, network throughput increases. From the simulation, we observed that D2M outperforms the other existing schemes — LSBT and BLO.

Future extension of this work includes an understanding of network bandwidth distribution in the presence of IoT devices with heterogeneous data at the edge-tier of DCN. Additionally, this work can be extended to understand how network bandwidth is to be distributed while reducing the energy consumption of the network.

References

[1] O. B. Data. What is big data? [Accessed on: May 7, 2018]. [Online]. Available: https://www.oracle.com/in/big-data/guide/what-is-big-data.html
[2] H. Lakhlef, A. Bouabdallah, M. Raynal, and J. Bourgeois, “Agent-Based Broadcast Protocols for Wireless Heterogeneous Node Networks,” *Comp. Comm.*, vol. 115, pp. 51 – 63, 2018.
[3] Z. Guo and Y. Yang, “Multicast Fat-Tree Data Center Networks with Bounded Link Oversubscription,” in *Proc. of IEEE INFOCOM*, Apr. 2013, pp. 350–354.
[4] A. Greenberg, J. R. Hamilton, N. Jain, S. Kandula, C. Kim, P. Lahiri, D. A. Maltz, P. Patel, and S. Sengupta, “VL2: A Scalable and Flexible Data Center Network,” in *Proc. of ACM SIGCOMM*, Aug. 2009, pp. 51–62.
[5] F. R. Dogar, T. Karagiannis, H. Ballani, and A. Rowstron, “Decentralized Task-aware Scheduling for Data Center Networks,” in *Proc. of ACM SIGCOMM*, Aug. 2014, pp. 431–442.
[6] B. Ciubotaru, C. H. Muntean, and G. M. Muntean, “Mobile Multi-Source High Quality Multimedia Delivery Scheme,” *IEEE Trans. on Broadcasting*, vol. 63, no. 2, pp. 391–403, Jun. 2017.
[7] M. Chen, S. Mao, and Y. Liu, “Big Data: A Survey,” *Mobile Net. and App.*, vol. 19, no. 2, pp. 171–209, 2014.
[8] H. V. Jagadish, J. Gehrke, A. Labrinidis, Y. Papakonstantinou, J. M. Patel, R. Ramakrishnan, and C. Shahabi, “Big Data and Its Technical Challenges,” *Comm. of the ACM*, vol. 57, no. 7, pp. 86–94, Jul. 2014.
[9] D. Paul, W. D. Zhong, and S. K. Bose, “Demand Response in Data Centers Through Energy-Efficient Scheduling and Simple Incentivization,” *IEEE Syst. J.*, vol. 11, no. 2, pp. 613–624, Jun. 2017.
[10] C. J. Wu, C. F. Ku, J. M. Ho, and M. S. Chen, “A Novel Pipeline Approach for Efficient Big Data Broadcasting,” *IEEE Trans. on Knowledge and Data Engg.*, vol. 28, no. 1, pp. 17–28, Jan. 2016.
[11] C. Raiciu, S. Barre, C. Pluntke, A. Greenhalgh, D. Wischik, and M. Handley, “Improving Datacenter Performance and Robustness with Multipath TCP,” in *Proc. of ACM SIGCOMM*, Aug. 2011, pp. 266–277.
[12] E. Chiu and V. K. N. Lau, “Precoding Design for Multi-Antenna Multicast Broadcast Services With Limited Feedback,” *IEEE Syst. J.*, vol. 4, no. 4, pp. 550–560, Dec. 2010.
[13] M. Al-Fares, S. Radhakrishnan, B. Raghavan, N. Huang, and A. Vahdat, “Hedera: Dynamic Flow Scheduling for Data Center Networks,” in *Proc. of USENIX Conf. on Net. Sys. Des. and Impl.*, Mar. 2010, pp. 1–15.
[14] R. W. Rosenthal, “Games of perfect information, predatory pricing and the chain-store paradox,” *Journal of Economic Theory*, vol. 25, no. 1, pp. 92–100, Aug. 1981.
[15] S. Saroiu, P. K. Gummadi, and S. D. Gribble, “Measurement Study of Peer-to-Peer File Sharing Systems,” in *Proc. of Int. Soc. for Optics and Photonics - Mult. Com. and Net.*, vol. 4673, 2002, pp. 156–170.