Modeling Flow Setup Time for Controller Placement in SDN: Evaluation for Dynamic Flows

Mu He, Arsany Basta, Andreas Blenk, Wolfgang Kellerer
Chair of Communication Networks
Department of Electrical and Computer Engineering
Technische Universität München
Email: {mu.he, arsany.basta, andreas.blenk, wolfgang.kellerer}@tum.de

***Abstract*—Software-Defined Networking (SDN) controllers are network entities that act as strategic control points in an SDN network. Controller placement studies mostly aim at optimizing network performance in terms of control latency, reliability and resilience, given network characteristics that are static. Yet dynamic traffic conditions, if not adapted by the controller placement properly, may cause high end-to-end flow setup time. For reactive controllers, the end-to-end flow setup time of a flow implies the difference between sending time at the source and receiving time at the sink of the first packet in that flow. Therefore, end-to-end flow setup time indicates the amount of time needed to set up forwarding rules in all involved switches and acts as a primary concern in terms of service establishment of network operators. In this paper, we analyze the controller placement for dynamic traffic flows based on a combined controller placement model: controller locations and switch-to-controller assignments are simultaneously optimized for minimum average flow setup time with respect to different traffic conditions inside the network. Linearization method is applied to transform the problem into a Mixed Integer Programming (MIP) problem which can be solved optimally. Two derivatives are also presented for comparison, one optimizing only controller locations and the other optimizing only switch-to-controller assignments. Our simulations cover two real network topologies and we explain the effects of the models have on the flow setup time with respect to dynamic flows. For low flow densities, the controller placement that adapts to flows could reduce the average flow setup time by about 50% compared to the static placement. However, when densities are high, the need of changing controller placement to guarantee flow setup performance is marginal.**

I. Introduction

Software-Defined Networking (SDN) is a novel network architecture which targets programmable, flexible and dynamic management of network resources with minimal interruption of network services. Concerning scalability and single point of failure issues [1], the architecture of physically distributed controllers sharing global network states becomes a promising solution. HyperFlow [2] is proposed as the first prototype distributing control plane logic among several controllers. ONOS [3] and OpenDaylight [4], as distributed SDN platforms, address the performance and reliability concerns in large operator networks. Controller placement in SDN includes several system components to change, namely the controllers' placement and also the switch-to-controller assignments. Distributed controllers running on VMs in data centers as a cloud solution can be migrated over different locations [5]. Control plane migration protocol [6], on the other hand, guarantees a seamless change of switch assignment. However, the evaluation of which components should change given certain flow characteristics has not been considered in a comprehensive way so far.

Given topology's graph characteristics, we would expect one optimal placement suffices most network management concerns. However, it does not always satisfy controller response time requirement when traffic flows vary dynamically [7]. Suppose a large backbone network spanning the US. In the morning of eastern coast side, the probability of high volume traffic is much higher than that of western coast side because of time difference. Moreover, traffic variability and sudden bursts occur even in small time scales. Working reactively, a controller needs to install flow rules in all involved switches inside its control domain when it receives the first packet of a flow. We refer the total time involved to forward the first packet of a flow from source to destination as end-to-end flow setup time $\boldsymbol{T_f}$, as all switches will be ready to directly forward the following packets after the first one. For the flows that only contain a few packets, e.g. DNS requests, T_f is a non-negligible part in each flow's lifetime and the responsiveness of those network services thus greatly impacts user experience.

Consequently, it is critical that controller placement can react to different flow distributions, for lower T_f of all flows and better allocation of controller resources. A flow distribution describes the sources and destinations of current flows for clarification and different flow distributions reflect dynamic traffic flows varying over time. Up to now, the effect of flow dynamics on the flow setup time and, in particular, on the change of different components involved in controller placement has not received enough attention.

We build our work on top of [8], in which T_f is mentioned but not clearly defined. Because of SDN's architecture, T_f is a combination of control latencies of some (not all) switches in the path and the total forwarding latency. After proposing a mathematical formulation of T_f, we formulate the controller placement problem with respect to flow distribution as an optimization problem, which minimizes average end-to-end flow setup time. From a flow's perspective, it prefers the controller placement in which its own T_f is minimum. The global optimum is therefore the placement that average T_f of all flows in a flow distribution is minimized. As a different flow distribution presents, the controller placement will also

change accordingly.

This paper focuses on controller placement for average flow setup time improvement in SDN and makes the following contributions: 1) formulating end-to-end flow setup time in SDN; 2) building up a combined controller placement model which minimizes average flow setup, as well as two derivatives that represent the change of different components in controller placement; 3) evaluating optimal placement results of two real network topologies for different flow distributions and analyzing the influence of different models including different system components to change.

The remainder of this paper is structured as follows. In Section II, we give a summary of related work in controller placement. Formulation of flow setup time and a controller placement problem which minimizes this metric are proposed in Section III. Section IV presents the simulation results of our optimization model, followed by conclusions drawn and future work in Section V.

II. Related Work

Controller Placement in SDN. Heller et al. in [9] initiated the controller placement problem and formulated it as a general facility location problem. They observed the effects of controller placement on both average control latency and worst-case control latency, between which a trade-off exists. [10] brought in a combination of real costs of controller deployment and device interconnection and then solved the problem optimally via brute force search. The work in [11], while considered load alleviation on controllers in addition to control latency, proposed a heuristic approach. [12], [13], [14] and [15] paid attention on the reliability and resilience aspects of control paths. They compared algorithms such as random, greedy and simulated annealing and tried to find the most efficient one.

Controller Placement for Dynamic Traffic. The above controller placements would not adapt to dynamic traffic inside the network, which would lead to possible performance degradation. Yao et al. in [16] considered the node weight when placing the controllers, and balanced the load of multiple controllers by a dynamic switch migration algorithm. They assumed yet only one switch is allowed to change its assignment from one controller to another. The work in [7] did switch-to-controller assignment update with respect to dynamic aggregate traffic in data center networks and proposed a two-phase algorithm to solve it efficiently; nevertheless, controller placement update was not considered. Bari et al. in [8] proposed a framework which adapts the number of controllers and their locations with changing network conditions for lower average T_f, based on a compound cost function. However, T_f is not appropriately modeled and the optimization model could not be solved optimally. Also, the effect of flow dynamics on the flow setup time is not shown.

III. Model and Formulation

In this section, we start by presenting the system model and the assumptions. With the help of an example, we then give a mathematical formulation of end-to-end flow setup time. Afterwards, the controller placement problem is formulated as a non-linear programming problem with the objective of minimizing average flow setup time given flow profile as input, followed by necessary linearization methods. In the end, more constraints are taken into consideration to address the cases that only controller placement or only switch-to-controller assignment is able to change.

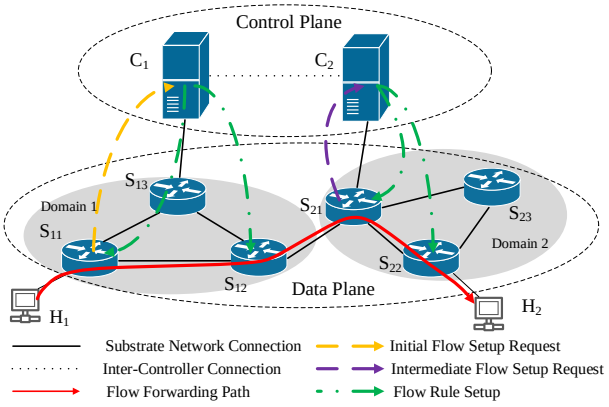
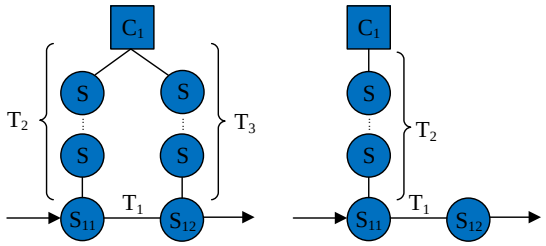


Fig. 1. Illustration of inter-domain flow and switch-controller interactions.

A. System Model: SDN with Distributed Controllers

We consider an SDN network with the topology following an undirected and connected graph $G = (V, E)$. Each node V in the graph hosts an SDN switch which is assigned to exactly one controller as its master. All switches that are controlled by the same controller form a control domain of that controller. K distributed controllers, that compose the control plane, need to be placed and the potential nodes that could host a controller are represented by $C \subseteq V$. Each controller only installs flow rules to its control domain switches reactively after it receives a flow setup request and the control-plane network is not separated from the data-plane network (in-band control). Shortest-path algorithm is applied for control-/data-plane routing and the forwarding latency between two nodes u and v is denoted as $D_{u,v}$.

We assume the scenario in which no controller is overloaded. [1] confirmed that the processing time for each flow setup request varies between 100 and 150 microseconds when controller is not overloaded. Therefore, compared with the forwarding latency on each link, which is in a magnitude of millisecond in large national topologies, the processing time can be neglected. Flow profile F represents the set of current flows in the network, indicating one flow distribution. Each element $f \in F$ is defined as a source-destination node pair and its forwarding path is represented as an ordered set of node pair p_f from source to destination. Note that as the control plane adapts to dynamic flow profiles, system reconfigurations, i.e. migrating controllers and reassigning switches, are inevitable. [17] and [6] mentioned that the reassignment of a switch takes



(a) $T_1 + T_2 > T_3$ (b) $T_1 + T_2 = T_3$

Fig. 2. (a) and (b) cover all possible topologies for depiction of flow setup time. A flow enters the control domain of C_1 at S_{11} and leaves from S_{12}. T_1 is the latency between S_{11} and S_{12}. T_2 and T_3 represent control latencies of S_{11} and S_{12}. There are two disjoint control paths in (a) just for illustration and switches in two control paths could actually overlap. The number of switches residing in each control path is greater than or equal to zero.

several RRTs between the switch and the controller, during which the switch experiences a performance degradation. However, in this paper we assume that the reconfiguration does not have any effect on T_f. An overview of the input sets and parameters is presented in Table I.

B. End-to-End Flow Setup Time

Following [8], [18], [19] and [20], we define end-to-end flow setup time T_f as follows. Consider an inter-domain flow in Figure 1, which originates at H_1 and ends at H_2. When the first data packet of this flow reaches switch S_{11}, the switch has no stored forwarding rule with respect to this new flow and it thus sends an initial flow setup request to its master controller C_1. After flow rules are properly decided, the controller sends them to every involved switch inside its own control domain, in this case S_{11} and S_{12}. The data packet is then forwarded inside the first domain until it enters the second domain. Similarly, the switch S_{21} initiates an intermediate flow setup request to its master controller C_2, waits for the flow rule setup and forwards it further. T_f is then the difference between the time S_{11} receives the first data packet and S_{22} successfully forwards it to the destination host H_2. Next we will provide a formulation of T_f.

For simplicity, we consider T'_f which is the time for the packet to go through one control domain. Figure. 2 illustrates this scenario in all possible topologies and S_{11} acts as the ingress switch, while S_{12} is the egress switch. The forwarding latency T_1 between S_{11} and S_{12} and the control latencies T_2 and T_3 of S_{11} and S_{12} are already known. If we compare the magnitude of $T_1 + T_2$ and T_3, we would encounter three situations. When $T_1 + T_2 > T_3$ as in Figure 2a, $T'_f = T_2 + max(T_2 + T_1, T_3) = T_1 + 2T_2$. The first data packet will be directly forwarded when it arrives at S_{12}, since the flow rule setup packet reaches S_{12} earlier than the data packet. Figure 2b shows another case that $T_1+T_2 = T_3$ and that the control path of S_{12} has to go through S_{11}, resulting in both flow rule setup packet and data packet simultaneously reach S_{12} and therefore $T'_f = T_2+T_2+T_1 = T_1+2T_2$. The case that $T_1+T_2 < T_3$ does not exist because the control path of S_{12} is not the shortest to the controller C_1, which violates the assumption of our system model. From both conditions, we could conclude that T'_f is a summation of twice the control latency of the ingress switch S_{11} and the forwarding latency of the flow inside this domain. As we consider all control domains that a flow goes through, T_f thus consists of every involved flow setup request and reply, as well as a total forwarding delay from source to destination.

TABLE I
LIST OF INPUT SETS AND PARAMETERS

Notations	Implication
V	Set of nodes
E	Set of links with $E \subseteq V \times V$
$G(V, E)$	Substrate network with node set V and link set E
C	Set of possible controller locations with $C \subseteq V$
F	Set of flows in the network (i.e. flow profile) with $f[s]$ and $f[d]$ as source node and destination node for $f \in F$
p_f	Ordered set of node pairs from source to destination on flow path of $f \in F$ with $p_f \subseteq E$
P	Set of flow paths p_f
K	Number of controllers to be placed
$D_{u,v}$	Forwarding latency from node u to node v with $u, v \in V$
CL_i	Set of nodes in ith cluster with $CL_i \in V$ and $1 \leq i \leq K$
PL	Set of controller locations with $PL \in V$ and $\|PL\| = K$

TABLE II
LIST OF VARIABLES

Variables	Implication
p_c	Binary variable representing whether a controller is placed on $c \in C$
$a_{v,c}$	Binary variable representing whether a switch $v \in V$ is assigned to a controller $c \in C$
cl_v	Non-negative variable representing the control latency of a switch $v \in V$
$sd_{c,u,v}$	Binary variable representing if both switches $u \in V$ and $v \in V$ are in the control domain of controller $c \in C$
$dd_{u,v}$	Binary variable representing whether two switches $u \in V$ and $v \in V$ are in different control domains
$nr_{u,v}$	Non-negative variable representing the necessary control latency if the flow goes from u to v

TABLE III
SIMULATION PARAMETER SETTINGS

Parameters	**Values**
Flow density D	0.05, 0.3, 0.6, 0.9
Traffic intensity	Following log-normal distribution with mean=1, var=0.8, in GBps
Data rate per flow	50 Mbps
Topologies	Abilene (11 nodes), AttMpls (24 nodes)
CPP models	CTR-SW, CTR, SW, STATIC

C. Combined Model for Controller Placement

1) Variables and Constraints: Table II lists all the variables. Constraint (1) ensures that K controllers are to be placed.

$$\sum_{c \in C} p_c = K \tag{1}$$

Constraint (2a) forces each switch to be controlled by only one controller and Constraint (2b) ensures that a switch needs be assigned to a location in which a controller is placed.

$$\sum_{c \in C} a_{v,c} = 1, \forall v \in V \tag{2a}$$

$$\sum_{v \in V} a_{v,c} \leq |V| \cdot p_c, \forall c \in C \tag{2b}$$

Constraint (3) assures the latency between a switch and its master controller to be either zero, when their places coincide, or the forwarding latency between them in the other case.

$$cl_v = \sum_{c \in C} a_{v,c} \cdot D_{v,c}, \forall v \in V \tag{3}$$

For every two switches, they need to be assigned to the same controller if they are in the same control domain and this is guaranteed by Constraint (4). Also, after checking the two switches' assignments to all potential controller locations with Constraint (5), it is clear whether or not they stand in different control domains. Both constraints can actually be combined; nevertheless, we keep them separately for latter linearization purposes.

$$sd_{c,u,v} = a_{u,c} \cdot a_{v,c}, \forall u, v \in V, \forall c \in C \tag{4}$$

$$dd_{u,v} = 1 - \sum_{c \in C} sd_{c,u,v}, \forall u, v \in V \tag{5}$$

For each consecutive node pair along the flow path, if they are in different domains, the second switch needs to issue a flow setup request and its control latency will get involved; otherwise it simply forwards the packet to the subsequent switch.

$$nr_{uv} = cl_v \cdot dd_{u,v}, \forall (u,v) \in p_f \tag{6}$$

2) Metric and Objective: After introducing variables and constraints, formulating the metric is quite straightforward. Following the modeling of end-to-end flow setup time in III-B, the coefficients of the first two terms in the objective function (7) represent the sum of latencies of flow setup request and flow rule installation. Our Metric M_{afst} is an average of T_f of all flows in a flow profile. Our objective is to minimize it.

$$M_{afst} = \frac{1}{|F|} \sum_{f \in F} (2 \cdot cl_{f[s]} + 2 \cdot \sum_{(u,v) \in p_f} nr_{u,v} + D_{f[s],f[d]}) \tag{7}$$

D. Linearization

As $nr_{u,v}$ in M_{afst} is a high-order entry, linearization by importing auxiliary constraints is necessary to make the problem solvable in generic optimizers, e.g. Gurobi and CPLEX. For Constraint (4), which has a product of two binary variables on its right side of the equal sign, we replace it by three equations.

$$sd_{c,u,v} \leq a_{u,c}, \forall u, v \in V, \forall c \in C \tag{8a}$$

$$sd_{c,u,v} \leq a_{v,c}, \forall u, v \in V, \forall c \in C \tag{8b}$$

$$sd_{c,u,v} \geq a_{u,c} + a_{v,c} - 1, \forall u, v \in V, \forall c \in C \tag{8c}$$

For Constraint (6), which consists binary and non-binary variables, four equations need to be introduced.

$$nr_{u,v} \leq dd_{u,v} \cdot \overline{cl}, \forall u, v \in V \tag{9a}$$

$$nr_{u,v} \leq cl_v, \forall u, v \in V \tag{9b}$$

$$nr_{u,v} \geq cl_v - (1 - dd_{u,v}) \cdot \overline{cl}, \forall u, v \in V \tag{9c}$$

$$nr_{u,v} \geq 0, \forall u, v \in V \tag{9d}$$

In order to generate a feasible solution space, constant $\overline{cl}$ should be an upper bound of $cl_v, \forall v \in V$. We assign $\overline{cl}$ with the worst case end-to-end forwarding latency in the topology.

E. Two Derivative Models

The previous optimization model, which we refer to as CTR-SW, allows the placement variable p and assignment variable a to change synchronously, when a new flow profile presents. In order to check the performance in case of only controller is allowed to move, or only switch can change its assignment, two additional models with the same objective are provided. For both models, Constraint (1) is replaced by other constraints.

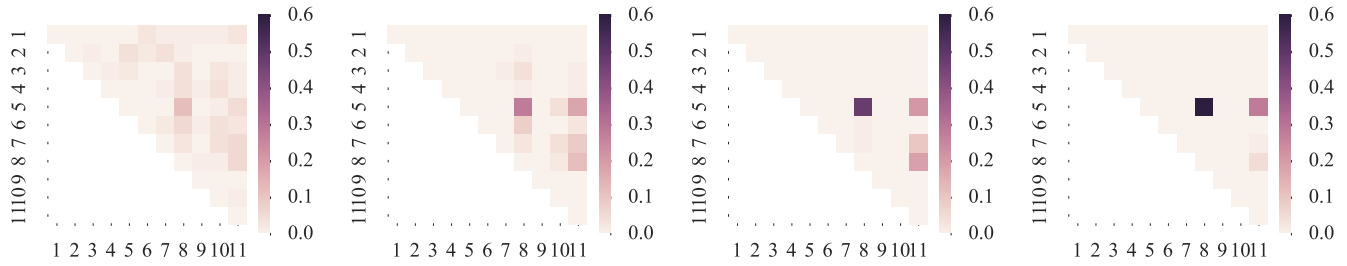
1) CTR model: This model stands for the scenario that switch-to-controller assignment is fixed. In other words, the control domains remain unchanged, while controller move inside their control domains. Control domain clustering information is assumed to be given. Constraint (10) and (11) assure that there is one and only one controller for each control domain and the domain-switches are assigned to that controller.

$$\sum_{c \in CL_i} p_c = 1, \text{for } i = 1, 2, ...k \tag{10}$$

$$\sum_{v \in CL_i, c \in CL_i} a_{v,c} = 1, \text{for } i = 1, 2, ..., k, \tag{11}$$

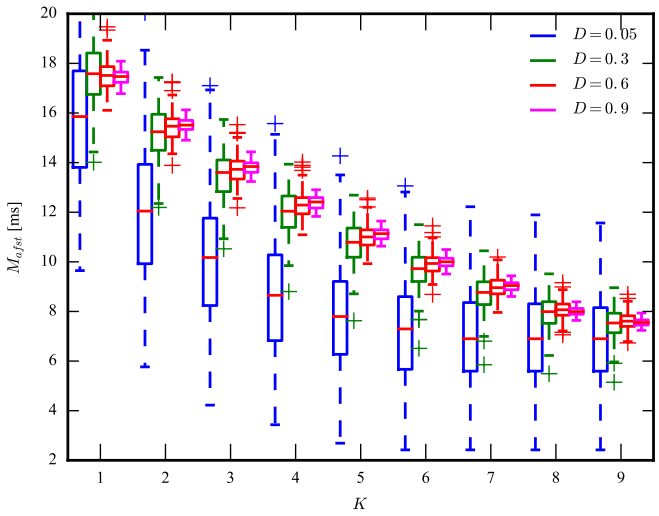
2) SW model: This model targets the scenario that controllers stay in their places and switches change their assignments, like the case in [7] and [16]. Controller location information is assumed to be given. Constraint (12) fixes the location of each controller.

$$p_c = 1, \forall c \in PL \tag{12}$$

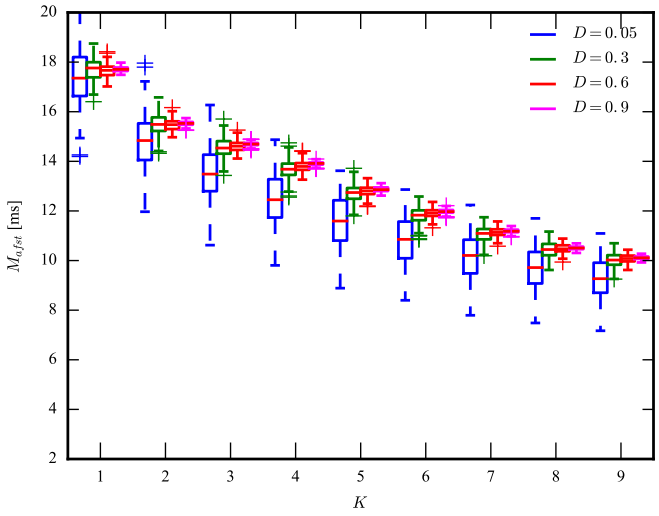


(a) $D = 0.05$ (b) $D = 0.3$ (c) $D = 0.6$ (d) $D = 0.9$

Fig. 3. Controller placement for Abilene considering different flow densities, each square shows one possible placement (two axes are two controller locations) and the darkness shows the possibility of being optimal ($K = 2$). Small squares in the upper triangle represent all combinations of two controller locations and the darker a square is, the higher possibility the corresponding combination has.



(a) Abilene, $K \in [1, 9]$



(b) AttMpls, $K \in [1, 9]$

Fig. 4. Optimal average flow setup time (in milliseconds [ms]) of CTR-SW for different flow densities as a function of number of controllers K.

IV. Simulation and Evaluation

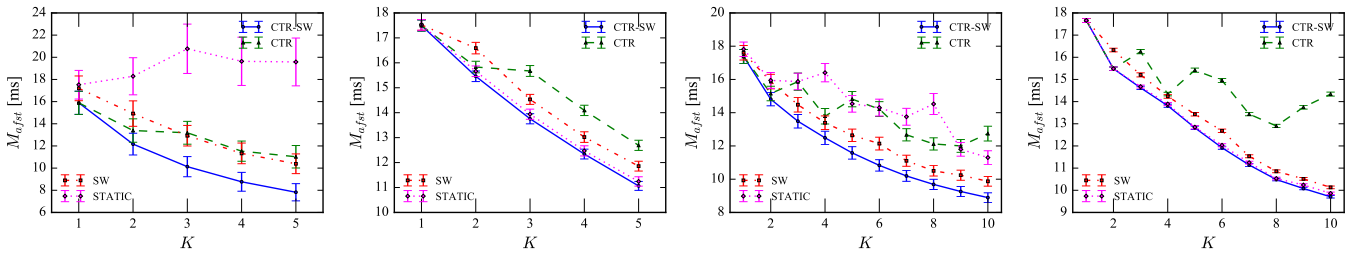
In this section, we investigate the behavior of the combined model as well as the derivatives and analyze their trade-offs. The chosen network topologies are Abilene (11 nodes) and AttMpls (24 nodes) [21].

A python-based framework based on the work in [22] is implemented and Gurobi 6.5 is chosen as our optimizer. The parameter settings are given in Table III. K is chosen from 1 to the total number of nodes, to exploit the effect of the number of controllers K on the metric. Flow density D is introduced as a parameter representing the level of sparseness of flow distribution inside the topology. Each flow distribution is represented by a flow profile and we produce flow profiles in the following way. There are in total $N = |V| \times |V|$ source-destination pairs in $G(V, E)$ and $N \cdot D$ pairs are randomly chosen. Afterwards, the number of flows on each pair will be generated. It is the quotient of total traffic volume and average data rate per flow. Traffic volume follows log-normal distribution [23]. For each D, flow profile generation is repeated 100 times for varying flow characteristics in order to achieve different flow distributions.

As is explained in Section III, we need clustering information as additional input for CTR and we refer to the spectral clustering method provided in [24]. Controller placement model minimizing average control latency [9] is applied to get controller location information for additional input for SW. As a baseline, STATIC applys the same placement, which is obtained by CTR-SW for the first flow profile, throughout all flow profiles.

A. Controller Placement With Respect To Flow Densities

We first stick with the CTR-SW model and vary flow densities, as well as the number of controllers. Figure 3 show the placements of CTR-SW ($K = 2$) in Abilene with D rising from 0.05 to 0.9. In Figure 3a, more than half of controller location combinations are optimal at least once. As flow distribution becomes denser in Figure 3d, controller locations converge to the network node 4 and 7. One resides in the center of the network (Kansas City) and the other is in the center of the west coast side topology (Sunnyvale). As the average flow setup time is partly impacted by control path of each switch, Kansas City is a compromise of east and west parts. Also, the distances in between the nodes of west part are longer than those in east part and Sunnyvale acts as a compensation.



(a) Abilene, $D = 0.05$, $K \in [1, 5]$ (b) Abilene, $D = 0.6$, $K \in [1, 5]$ (c) AttMpls, $D = 0.05$, $K \in [1, 10]$ (d) AttMpls, $D = 0.6$, $K \in [1, 10]$

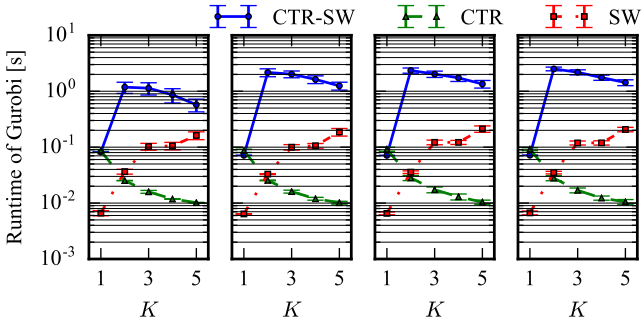
Fig. 5. Optimal average flow setup time (in milliseconds [ms]) for different models as a function of number of controllers K.

Flow density plays a role in controller placement. When flows are fully distributed inside the network, less candidate placements might contribute to potential controller locations in optimization. In other words, a set of potential controller placements can be considered to shrink the set size of C, which will lead to a smaller optimization runtime.

Figure 4 depicts the relation between the number of controllers K and average flow setup time. We get the optimal placements of CTR-SW with K changing from 1 to 10 for both topologies. Then M_{afst} are plotted with respect to D and K. T_f is the combination of a fixed part, which is the path forwarding latency, and a changing part, which comes from the summation of flow setup request latencies along the forwarding path. When more controllers are deployed, each flow has a higher probability of going through more control domains, but contrarily every involved flow setup request latency tends to be smaller, because of decreased control domain size. We can observe that M_{afst} monotonically decreases as K increases, yet adding controllers yields less than linear reduction in M_{afst}. A saturation point can also be detected in Figure 4a for $D = 0.05$ and $K \geq 9$. It means that a large number of controllers could assure the changing part in M_{afst} to be zero if the flow distribution is sparse. As D increases, two phenomena can be observed. The distribution of M_{afst} becomes narrower, meaning that the optimal placement converges. Also, the expectation of M_{afst} increases, since controller placement could not ensure the minimum T_f of each individual flow and when flow density is higher, more flows with longer T_f contribute to a larger M_{afst}.

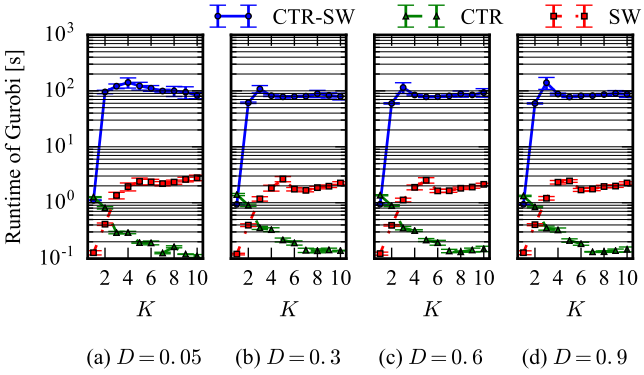
B. Effectiveness Of Adapting To Dynamic Flows

Now we focus on the comparison among all models. Figure 5 depicts average flow setup time M_{afst} of CTR-SW, CTR, SW and STATIC with 99.9% confidence in Abilene and AttMpls. We make the following observations: (1) For a sparse flow distribution ($D = 0.05$), CTR-SW, CTR and SW, which adapt to dynamic traffic flows, mostly outperform STATIC, which sticks to the first placement throughout all flow profiles. When $K > 2$ and in Abilene, M_{afst} of STATIC is more than twice that of CTR-SW. It demonstrates that static controller placement could result in severely slow flow setup, if the static placement we stick to is not properly chosen. (2) As flow distribution becomes much denser ($D = 0.6$), the advantage of CTR-SW over STATIC is not significant any more. Since



(a) $D = 0.05$ (b) $D = 0.3$ (c) $D = 0.6$ (d) $D = 0.9$

Fig. 6. Solution runtime (in seconds [s]) of different models and flow densities (Abilene, $K \in [1, 5]$, y-axis in log-scale).



(a) $D = 0.05$ (b) $D = 0.3$ (c) $D = 0.6$ (d) $D = 0.9$

Fig. 7. Solution runtime (in seconds [s]) of different models and flow densities (AttMpls, $K \in [1, 10]$, y-axis in log-scale).

CTR-SW ends up with nearly the same optimal solution for different flow profiles, abiding any flow profile only loses less than 1% optimality. Moreover, STATIC clearly beats CTR and SW. Since CTR and SW models attempt changing either the controller locations or the switch-to-controller assignments, they end up in a less optimal solution. (3) CTR-SW always guarantees the best M_{afst}. As the number of controllers K becomes large, the gap in between CTR-SW and SW stays unchanged. The performance difference between CTR and SW, however, highly depends on K and topology. For AttMpls, M_{afst} of CTR may even become worse, as more controllers are deployed. The goodness of clustering algorithm greatly affects the performance of CTR. Spectral clustering,

which only targets at nearly equal-sized clusters, does not always benefit T_f. Moreover, M_{afst} of SW decreases as K increases for both topologies, which demonstrates the positive effect of average control latency on M_{afst}.

C. Runtime Evaluation

Figures 6 and 7 show the runtime for solving the models of CTR-SW, CTR and SW. For CTR-SW, the runtime of $K > 1$ takes about 10 times and 100 times that of $K = 1$ for Abilene and AttMpls respectively. The difference between each K for $K > 1$ indicates that more controllers does not necessarily bring in a harder problem. Larger D, however, for both topologies and all K, results in an unapparent increase in runtime. Unlike other two models, CTR becomes simpler to solve as K increases, because having more clusters is also cutting off more search space. SW, on the contrary, takes more time for larger K. In case SW or CTR outputs near optimal solutions as CTR-SW, it can be used to obtain savings in runtime.

V. Conclusion

In this paper, we look into flow setup time in SDN which reflects the goodness of service that network provider can support. We design a controller placement model with the metric of average flow setup time, reduce the model's dimensionality and solve it optimally. Two derivatives are introduced to represent the change of different components involved in controller placement. Simulation results have shown the performance of CTR-SW for different flow dynamics and different number of controllers. Moreover, we demonstrate the effects of the different models with respect to dynamic flows. Whereas controller placement models, that adapt to different flows, can reduce the average flow setup time by up to 50% (CTR-SW), advantages of a static placement (STATIC) are revealed when flow densities are high. Optimization runtime measurements reveal the complexity of different models.

For future work, we will consider the cost of controller migration, depending on its load, and the cost of switch reassignment for generating new optimal placements. Besides, it is important to consider the impact on flow setup performance during controller replacement. The frequency of controller replacement will also be studied based on the dynamic traffic input.

Acknowledgment

This work is part of a project that has received funding from the European Research Council (ERC) under the European Unions Horizon 2020 research and innovation program (grant agreement No 647158 - FlexNets)

References

[1] A. Tootoonchian, S. Gorbunov, Y. Ganjali, M. Casado, and R. Sherwood, "On controller performance in software-defined networks," in *Presented as part of the 2nd USENIX Workshop on Hot Topics in Management of Internet, Cloud, and Enterprise Networks and Services*, 2012.

[2] A. Tootoonchian and Y. Ganjali, "Hyperflow: A distributed control plane for openflow," in *Proceedings of the 2010 internet network management conference on Research on enterprise networking*, 2010, pp. 3–3.

[3] P. Berde, M. Gerola, J. Hart, Y. Higuchi, M. Kobayashi, T. Koide, B. Lantz, B. O'Connor, P. Radoslavov, W. Snow *et al.*, "Onos: towards an open, distributed sdn os," in *Proceedings of the third workshop on Hot topics in software defined networking*. ACM, 2014, pp. 1–6.

[4] J. Medved, R. Varga, A. Tkacik, and K. Gray, "Opendaylight: Towards a model-driven sdn controller architecture," in *The Proceedings of 15th IEEE WoWMoM*. IEEE, 2014.

[5] F. Travostino, P. Daspit *et al.*, "Seamless live migration of virtual machines over the man/wan," *Future Generation Computer Systems*, vol. 22, no. 8, pp. 901–907, 2006.

[6] A. Basta, A. Blenk, H. B. Hassine, and W. Kellerer, "Towards a dynamic sdn virtualization layer: Control path migration protocol," in *Network and Service Management (CNSM), 2015 11th International Conference on*. IEEE, 2015, pp. 354–359.

[7] T. Wang, F. Liu, J. Guo, and H. Xu, "Dynamic sdn controller assignment in data center networks: Stable matching with transfers," in *Proc. of INFOCOM*, 2016.

[8] M. F. Bari, A. R. Roy, S. R. Chowdhury, Q. Zhang, M. F. Zhani, R. Ahmed, and R. Boutaba, "Dynamic controller provisioning in software defined networks," in *Proceedings of CNSM 2013*. IEEE, 2013.

[9] B. Heller, R. Sherwood, and N. McKeown, "The controller placement problem," in *Proceedings of the first workshop on Hot topics in software defined networks*. ACM, 2012, pp. 7–12.

[10] A. Sallahi and M. St-Hilaire, "Optimal model for the controller placement problem in software defined networks," *IEEE Communications Letters*, vol. 19, no. 1, pp. 30–33, 2015.

[11] G. Yao, J. Bi, Y. Li, and L. Guo, "On the capacitated controller placement problem in software defined networks," *IEEE Communications Letters*, vol. 18, no. 8, pp. 1339–1342, 2014.

[12] Y. Hu, W. Wang, X. Gong, X. Que, and S. Cheng, "On reliability-optimized controller placement for software-defined networks," *China Communications*, vol. 11, no. 2, pp. 38–54, 2014.

[13] Y. Hu, W. Wendong, X. Gong, X. Que, and C. Shiduan, "Reliability-aware controller placement for software-defined networks," in *2013 IFIP/IEEE International Symposium on Integrated Network Management (IM 2013)*. IEEE, 2013, pp. 672–675.

[14] M. Guo and P. Bhattacharya, "Controller placement for improving resilience of software-defined networks," in *2013 Fourth International Conference on Networking and Distributed Computing*. IEEE, 2013.

[15] D. Hock, M. Hartmann, S. Gebert, T. Zinner, and P. Tran-Gia, "Poco-plc: Enabling dynamic pareto-optimal resilient controller placement in sdn networks," in *Computer Communications Workshops, 2014 IEEE Conference on*. IEEE, 2014, pp. 115–116.

[16] L. Yao, P. Hong, W. Zhang, J. Li, and D. Ni, "Controller placement and flow based dynamic management problem towards sdn," in *2015 IEEE International Conference on Communication Workshop*. IEEE, 2015.

[17] A. A. Dixit, F. Hao, S. Mukherjee, T. Lakshman, and R. Kompella, "Elasticon: an elastic distributed sdn controller," in *Proceedings of the tenth ACM/IEEE symposium on Architectures for networking and communications systems*. ACM, 2014, pp. 17–28.

[18] K. Bu, "Gotta tell you switches only once: toward bandwidth-efficient flow setup for sdn," in *2015 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*. IEEE, 2015, pp. 492–497.

[19] Z. Su, T. Wang, Y. Xia, and M. Hamdi, "Cheetahflow: Towards low latency software-defined network," in *2014 IEEE International Conference on Communications (ICC)*. IEEE, 2014, pp. 3076–3081.

[20] D. Zeng, C. Teng, L. Gu, H. Yao, and Q. Liang, "Flow setup time aware minimum cost switch-controller association in software-defined networks," in *Heterogeneous Networking for Quality, Reliability, Security and Robustness (QSHINE), 2015 11th International Conference on*. IEEE, 2015, pp. 259–264.

[21] S. Knight, H. X. Nguyen, N. Falkner, R. Bowden, and M. Roughan, "The internet topology zoo," *IEEE Journal on Selected Areas in Communications*, vol. 29, no. 9, pp. 1765–1775, 2011.

[22] A. Blenk, P. Kalmbach, P. van der Smagt, and W. Kellerer, "Boost online virtual network embedding: Using neural networks for admission control," in *Proceedings of 12th CNSM*. IEEE, 2016.

[23] A. Nucci, A. Sridharan, and N. Taft, "The problem of synthetically generating ip traffic matrices: initial recommendations," *ACM SIGCOMM Computer Communication Review*, vol. 35, no. 3, pp. 19–32, 2005.

[24] P. Xiao, W. Qu, H. Qi, Z. Li, and Y. Xu, "The sdn controller placement problem for wan," in *2014 IEEE/CIC International Conference on Communications in China (ICCC)*. IEEE, 2014, pp. 220–224.