How Flexible is Dynamic SDN Control Plane?

Mu He, Arsany Basta, Andreas Blenk, Wolfgang Kellerer
Chair of Communication Networks
Department of Electrical and Computer Engineering
Technical University of Munich, Germany
Email: mu.he, arsany.basta, andreas.blenk, wolfgang.kellerer@tum.de

***Abstract*—In Software Defined Networking (SDN), for performance reasons the control plane consists of multiple SDN controllers that provide a logically centralized network control. To react to traffic changes, the control plane may adapt to improve the performance of the whole network. This adaptation includes the migration of controllers, i.e. the change of their placement, as well as the change of switch to controller assignments. We call such adaptive network control a dynamic SDN control plane. Whereas most state of the art focuses on an optimal placement of controllers considering static traffic demands, we draw the attention to the need for a dynamic control plane and the performance of the adaption process itself. The involved controller migration and switch re-assignments significantly affect the control plane performance. Moreover, in this work we evaluate the flexibility of a dynamic SDN control plane with respect to the number of controllers. In particular, we refer to a flexibility metric that takes the time into account that is needed for successful migrations to handle traffic changes. Our evaluations are based on detailed models of controller placement and migration time. The results confirm that the migration time is a critical parameter and reveal that in case only a short migration time is allowed, the number of controllers does not bring more flexibility. For relaxed migration times, a larger number of controllers leads to more flexibility as expected. The results also show that a dynamic adaptation to traffic changes, can provide an improvement of up to 60% over a non-dynamic SDN control plane.**

I. Introduction

Software Defined Networking (SDN) decouples the control plane from the data plane of forwarding devices such as switches and routers. The control plane functionality is provided by a new network entity the SDN controller. The initial proposal of a single controller has shown several disadvantages. The most obvious is that it constitutes a single point of failure. As an SDN network size increases, scalability becomes an issue. First, the number flows that need to be processed increases drastically, whereas, e.g., a single default OpenDaylight controller implementation can only process Packet-In messages at the rate of around 15000 pps [1], which is not comparable with that of a backbone router. Second, the propagation delay between a switch to its master controller becomes a significant part of the control plane latency as the network scales up [2].

To cope with robustness and scalability issues, a logically centralized and physically distributed control plane has been introduced. Several controller instances form a cluster and each switch can be connected to one or more controllers. In general, the design of the SDN control plane can be modeled as a Controller Placement Problem [3].

For a distributed control plane, static controller locations or static switch-to-controller mapping may lead to unbalanced network resources and degraded flow processing performance over time. As traffic pattern inside a network can change dramatically due to unexpected events, adapting the number of controllers and their placement is necessary. We call such adaptive network control a Dynamic SDN Control Plane. If the overall network load increases, adding more controllers can balance controller load and guarantee an acceptable controller response time. With respect to control plane latency, an important network performance indicator is the average flow setup time, i.e., the time it takes from a packet-in event until the rules are established on all related switches. Such control plane latency aspect suffers from temporal local traffic peaks at some nodes. In this case, for performance improvement, it is not needed to increase the number of controllers, but to change the placement of the controllers for a better overall resource utilization.

In this paper, we focus on the control plane latency scenario and evaluate the migration of controllers to adapt to traffic changes for fixed sets of controllers. Here, for a dynamic SDN control plane, the current controller placement should be adapted to traffic conditions to reach a global optimal configuration in terms of control plane performance such as control plane latency. Yet, most state of the art address an optimal controller placement considering only static traffic demands. Moreover, they only deal with the placement optimization problem and do not consider the adaptation process. Actually, controller placement adaptation involves migrating controllers and re-assigning switches to controllers, which introduces a significant overhead.

In this work, we evaluate the flexibility of a dynamic SDN control plane. In particular, we refer to a flexibility metric that takes the time into account that is needed for successful migrations to handle traffic changes. This flexibility metric is based on our initial proposal for a flexibility measure of networks [4] where we did not consider time constraints yet. By comparing the quantitative values, we reach several conclusions about the flexibility of certain control plane settings, i.e., the number of controllers, as well as the gains of dynamic adaptations.

The main contributions of this paper are as follows.

- Introduction of a Dynamic SDN Control Plane taking migration time and switch re-assignment time into

account as critical parameters for performance analysis;

- Analysis of the flexibility of a Dynamic SDN Control Plane with respect to the number of controllers taking migration time constraints into account;
- System performance analysis to show the gains that can be achieved with a Dynamic Control Plane, which is in the order of up to 60%.

In the remainder of this paper, we present the necessary background and a related work analysis in Section II. Section III describes the models that we use in our study and the respective flexibility metric definition. The evaluation setup and our evaluation results are presented in Section IV.

II. Background & Related Work

In this section, we first introduce platform examples of a distributed SDN control plane in general. We discuss state-of-the-art models for dynamic controller placement that reacts to traffic dynamics in the network. (Supportive) examples from related work for controller and switch migration are presented.

Distributed SDN control plane. Onix [5], as one of the first distributed SDN control plane platforms, provides distribution primitives enabling efficient controller application implementation. As its successor, ONOS [6] is an open source project which focuses on improving performance in event processing and message synchronization. OpenDaylight [7] is another open source distributed SDN Platform that attracts the attention of the industry. These distributed SDN control platforms provide some adaption, e.g., as a reaction to controller failures, however, they do not provide mechanisms to adapt the location of the controllers in response to traffic change. We consider these platforms as a basis to our proposed models.

Dynamic controller placement. Bari et al. [8] initiated placing controllers and assigning switches to them dynamically, based on the real time traffic condition, i.e. number of flows and their origins. [9] considers only the switch assignment update, as the controllers do not change their locations, in the use case of data centers where aggregated traffic increases in the daytime and falls at night. However, these models give less attention to the migration process and its cost on the network performance, which we model in detail in our proposed models.

Controller migration. There are two approaches to migrate a controller. First, the controller application can be installed on a virtual machine (VM) that can be directly migrated between servers. The capability to migrate live VMs across a wide are network (WAN) has been discussed in [10]. [11] and [12] evaluated the service downtime during the VM migration on a server cluster, which ranged between tens and hundrends of milliseconds. The other approach, which can be used for stateless controller migration, involves booting up a new VM with controller application and transferring only the data store from the old controller application to the new one. In ONOS, each controller instance has a local copy of the data store saving switch membership and network topology. [13] introduces HopSwap as a system example for upgrading SDN controllers without service disruption and the "network state" is synchronized to the new controller through a consistent update. We use the insights from the above approaches to build our controller migration model.

TABLE I: List of input sets and parameters

Symbol	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 $u, v \in V$

SDN switch migration. ElastiCon [14] introduces a dynamic mapping between switches and controllers and thus guarantees dynamic load adaptation and elasticity. They propose a switch migration protocol as a part of their elastic distributed framework which enables load shifting between controllers. The protocol ensures that no messages will get lost or duplicated during the migration process. The work in [15] proposed another protocol for switch migration, however, for distributed SDN network hypervisors in the context of virtual SDN networks. The switch re-assignment model is based on the mechanisms proposed in the above protocols.

III. Dynamic Controller Placement Problem

Given varying traffic input, a dynamic SDN control plane adapts its controller placement to achieve better global resource utilization and network performance. However, the controller migration and switch re-assigments have an influence on the overall control plane performance. Therefore, we provide a model for this dynamic controller placement problem. The problem consists of two parts, i.e. controller placement model and migration model. As the flow profile changes, the optimal controller placement also updates accordingly, in case the control plane is flexible to support it. We provide a formulation for the migration model of controllers and switches re-assignment, which brings in the time constraint for supporting change requests.

We also define a quantitative measure for flexibility considering the migration time constraint comparing different number of controllers in a dynamic SDN control plane. A distributed control plane is expected to be more flexible in supporting dynamic traffic, however, a quantitative analysis of how flexible it can be is missing in the state-of-the-art so far.

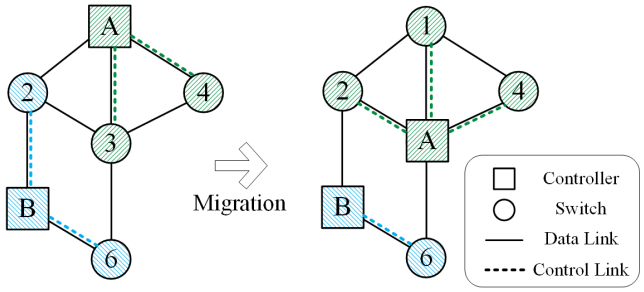
The input sets and parameters for our models are presented in Table I. The problem variables are listed in Table II.

A. Controller Placement Model

The linear optimization model, which is presented in our previous work [16], takes flow profiles as input and minimizes

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 $v \in V$
$dd_{u,v}$	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



Flow Profile 1 Flow Profile 2

Fig. 1: Example of controller migration and switch re-assignment upon a traffic profile change.

the average flow setup time. We define the objective function as Eq. (1), subject to constraints Eq. (2) - (7).

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

$$\sum_{c \in C} p_c = K \tag{2}$$

$$\sum_{c \in C} a_{v,c} = 1, \forall v \in V \tag{3}$$

$$\sum_{v \in V} a_{v,c} \leq |V| \cdot p_c, \forall c \in C \tag{4}$$

$$cl_v = \sum_{c \in C} a_{v,c} \cdot D_{v,c}, \forall v \in V \tag{5}$$

$$dd_{u,v} = 1 - \sum_{c \in C} a_{u,c} \cdot a_{v,c}, \forall u, v \in V \tag{6}$$

$$nr_{u,v} = cl_v \cdot dd_{u,v}, \forall (u,v) \in p_f \tag{7}$$

Eq. (2) ensures that K controllers need to be placed. Each switch being assigned by one controller that is placed in a location is ensured by Eq. (3) and (4). Eq. (5), (6), (7) help to build up variables cl_v, $dd_{u,v}$ and $nr_{u,v}$.

Algorithm 1 Total Migration Time Calculation

Input: last controller set C_{old}, current controller set C_{new}, switch set V, size of data store in controller $Size$, link bandwidth Bw, shortest path latency $D(u,v)$
Output: Total migration time T_{mig}

set of propagation latencies of the controllers to be migrated $\mathcal{T}_{\text{ctr\_prop}} = \emptyset$
$T_{\text{ctr\_prop}} = 0$, $T_{\text{ctr\_tran}} = 0$
for each c_{new} in C_{new} **do**
  if $c_{\text{new}} \notin C_{\text{old}}$ **then**
    find the $c_{\text{old}} \in C_{\text{old}}$ such that the latency $D(c_{\text{old}}, c_{\text{new}})$ is minimal.
    insert $D(c_{\text{old}}, c_{\text{new}})$ into $\mathcal{T}_{\text{ctr\_prop}}$
  end if
end for
$T_{\text{ctr\_prop}} = \max(\mathcal{T}_{\text{ctr\_prop}})$
$T_{\text{ctr\_tran}} = Size/Bw$
set of reassignment times of switches $\mathcal{T}_{\text{sw}} = \emptyset$
for each $sw \in V$ **do**
  if sw is assigned to a different controller instance **then**
    get the master controller c of sw
    insert $D(sw, c)$ into $\mathcal{T}_{\text{sw}}$
  end if
end for
$T_{\text{sw}} = \max(\mathcal{T}_{\text{sw}})$
$T_{\text{mig}} = T_{\text{ctr\_prop}} + T_{\text{ctr\_tran}} + T_{\text{sw}}$

B. Controller and Switch Migration Model

Fig. 1 illustrate a typical migration of controllers and switch re-assignment. As new flow profile is present, the placement model results in a different optimal controller placement and switch assignment, hence, migration is necessary. As shown in the figure, controller A moves from node 1 to node 3, while switch 2 is re-assigned to controller B. In this section, we illustrate the model of migration that defines associated migration time (T_{mig}). Generally, a migration consists of controller migration and switch re-assignment. Controller migration time contains a transmission part ($T_{\text{ctr\_tran}}$), as data needs to be transmitted from old location to new location, and a forwarding part ($T_{\text{ctr\_prop}}$), as in large area networks the forwarding latency is normally non-negligible. Switch re-assignment takes place as soon as controller migration finishes, and it involves several rounds of control information exchange, in which forwarding latency plays the major role. And the time for re-assignment is represented as T_{sw}. Alg. 1 describes the detailed process in which the total amount of time for migration is calculated. Note that $D(u,v)$ is a function returning the shortest path latency between any two nodes.

1) Controller Migration: We assume that the control plane is fully distributed and each controller instance has access to the same full network state, e.g., flows and network topology. For example, in ONOS, each controller has a local copy of the network state called data store. We consider stateless

Algorithm 2 Flexibility Measure Calculation

Input: migration time threshold T
Output: flexibility measure $\varphi_T^{\text{placement}}$

1: randomly create an order of all flow profiles
2: total number of migration requests $n_{\text{req}} = 0$
3: total number of migration success $n_{\text{suc}} = 0$
4: **for** each flow profile in the order **do**
5: $n_{\text{req}} = n_{\text{req}} + 1$
6: get optimal controller set C_{new} and switch assignment set A_{new}
7: **if** this is the first flow profile **then**
8: $C_{\text{old}} = C_{\text{new}}$, $A_{\text{old}} = A_{\text{new}}$
9: continue
10: **end if**
11: **if** $C_{\text{new}} = C_{\text{old}}$ and $A_{\text{new}} = A_{\text{old}}$ **then**
12: $n_{\text{suc}} = n_{\text{suc}} + 1$
13: **else**
14: call Alg. 1 to calculate T_{mig}
15: **if** $T_{\text{mig}} < T$ **then**
16: $n_{\text{suc}} = n_{\text{suc}} + 1$
17: $C_{\text{old}} = C_{\text{new}}$, $A_{\text{old}} = A_{\text{new}}$
18: **end if**
19: **end if**
20: **end for**
21: $\varphi_T^{\text{placement}} = n_{\text{suc}}/n_{\text{req}}$

controllers, which means that no state information is involved in migration. Therefore, when the control plane calls for a new controller location, only the data store needs to be copied from whichever old location, that is closest to the new location. We also assume that controllers can be migrated simultaneously.

When a new flow profile arises and the new optimal controller locations differ from the old one, the time for controller migration will be computed. First each new controller finds an old controller that is closest to it and calculate the corresponding propagation latency and put into a set $\mathcal{T}_{\text{ctr\_prop}}$. $T_{\text{ctr\_prop}}$ is then the maximum of $\mathcal{T}_{\text{ctr\_prop}}$. As for $T_{\text{ctr\_tran}}$, it is the division of data store size and migration link bandwidth. No matter how many controllers are deployed, the data store size inside one controller is always the same, because the data store is only related to the underlying topology. The above procedure is shown in Alg. 1 from line 1 to line 10.

2) Switch Re-Assignment: After the controller starts running in the new location, switch also stops the old control channel and establish a new one with the new controller. However, the Openflow specification does not provide a native mechanism in re-assigning SDN switch to different controller instances and leaves it as an open question to network providers. Nevertheless, during the re-assignment process of a switch, the liveness and safety properties should be guaranteed [14]. On one hand, a switch is always connected to at least one active controller. A controller that has issued Packet-Out and Flow-Mode messages as a response of Packet-In from a switch should not be turned off before the switch finishes the corresponding flow operation. On the other hand, a Packet-In should only be responded exactly once and duplicate Flow-Mod messages to a switch may cause the same entry being pushed into the flow table repeatedly.

TABLE III: Simulation parameter settings

Parameters	Values
Number of controllers	1, 2, 3, 4
Number of flow S-D pairs	6 (Abilene), 14 (Germany)
Flow number distribution	ln$\mathcal{N}$ with mean of 20
Total data store size of a controller	100 MB
Migration Link Bandwidth	10 Gbps
# RTTs of re-assigning a switch	6
Number of requests per run	100
Number of runs	50

A switch re-assignment protocol typically consists of several rounds of control message exchanges between switch and controller. The new controller first initiates a TCP connection with the switch, involving a three-way handshake. Then the Hello and Openflow Handshake messages are issued from the controller, in order to complete a standard configuration procedure of the switch. The switch migration protocol in ElastiCon [14] requires 6 round trip times from the switch to the controller.

For each switch that need to be re-assigned, the propagation latency between it and its new master controller is put into a set $\mathcal{T}_{\text{sw}}$, whose maximum is T_{sw}. Alg. 1 from line 11 to line 18 shows this process.

C. Flexibility Measure With Time Constraint

When facing dynamic input or requirements, a flexible system is one that can adapt itself with a goal of better performance. Our flexibility measure definition thus goes as follows. Given a system S, its flexibility can be measured with respect to a certain aspect, e.g. flow steering, function placement and function operation [4], with a consideration of time constraint. The number of change requests that can be supported within a time constraint T is divided by the number of change requests.

$$\varphi_T(S) = \frac{|\text{supported change requests within T}|}{|\text{change requests}|} \tag{8}$$

Obviously $\varphi(S) \in [0, 1]$. When $T \to \infty$, the denominator becomes the absolute value of all supported requests without any time constraint. Our previous work [4] performed a use case study with regard to this flexibility measure definition for the function placement in next generation mobile core network. In order to capture the dynamics of system and requests, we analyze the needed time T to support a system change. In our use case study, which focuses on dynamic controller placement, the denominator and nominator in Eq. (8) are supported placement change requests within T and total change requests respectively.

In one run, a consecutive order of flow profiles work as the dynamic changing requests. Considering each new request, we expect an optimal placement that leads to minimal average flow setup time. If the new optimal placement is different from the old one, the adaptation time, i.e. total migration time, will be calculated and verified with the threshold T, so as to decide whether it is a "supported" request. Flexibility is the fraction of successful migrations achieved within T and total number of migration requests. A detailed description of the process of one simulation run is shown in Alg. 2.

IV. Evaluation

In this section, we show the evaluation of our proposed dynamic controller placement problem considering different number of controllers and network typologies. We also evaluate the flexibility of the SDN control plane using our proposed flexibility measure with different migration time constraints.

A. Simulation Setup and Parameters

In order to produce dynamic requests, we generate 100 flow profiles and optimize the controller placement model for different number of controllers. The 100 flow profiles, i.e. requests, are shuffled to further represent dynamics and then fed into the migration model with migration time threshold T. Since we assume seamless migration, T represents the duration, in which response time of asynchronous messages like Packet-In will be degraded. In order to convey the relation between T and our flexibility measure, we choose T in a way that it covers the range from almost no requests to nearly all requests can be supported. Detailed parameter settings are listed in Table III.

B. Results and Observations

Evaluation results of different topologies are shown in Fig. 2. Each column of plots correspond to one topology, while each row of plots show different observations, i.e., flexibility measure, dynamic controller placement performance, and performance ratio of dynamic compared to static placement.

A general trend can be observed in Fig. 2a and Fig. 2b that the flexibility increases as the migration time constraint T increases. This is because a larger T guarantees more successful migrations. More controllers lead to less inter-controller distance and less control latency of switch, which provides higher flexibility. When T is small, the flexibility of 1-ctr is higher than that of all other cases. This is because the single controller tends to stay in one optimal location even for different flow profiles, which requires no migration time.

We use the average flow setup time as an indicator for the placement performance. Note that it is the average of all flows from all flow profiles. If the migration time threshold T is not fulfilled, the new optimal placement, if exists, will not be accepted and thus average flow setup time will be degraded. In general, the performance improves as the control plane becomes more flexible. Given a single controller, only large T generates a marginal performance improvement. In Fig. 2c and Fig. 2d, when T is small, the average flow setup time of different number of controllers in most runs is close. 1-ctr even outperforms 2-ctr in some cases, because under a stringent T two controllers may remain in a placement for the following flow profiles that degrades control latencies. However, as T increases, average flow setup time of 4-ctr outperforms the other three cases in every run. Comparing with 1-ctr, 4-ctr leads to more than 35% performance increase for both topologies, given a loose T.

Finally, we compare the average flow setup time of a dynamic system, which is able to adapt but with a time constraint, with that of a static system shown in Fig. 2e and Fig. 2f. For the static placement, we keep the optimal controller placement from the first flow profile for all new flow profiles. It is clear that the more controllers are deployed, the stronger need for system's adaptation with respect to changing input. The improvement in performance from being dynamic can be as high as 60% for Abilene. Note that the static placement can outperform the dynamic placement in case it could not adapt under strict migration time T. This happens if the initial static placement is by chance better than the placement solutions that the dynamic problem remain at without being able to adapt to change requests.

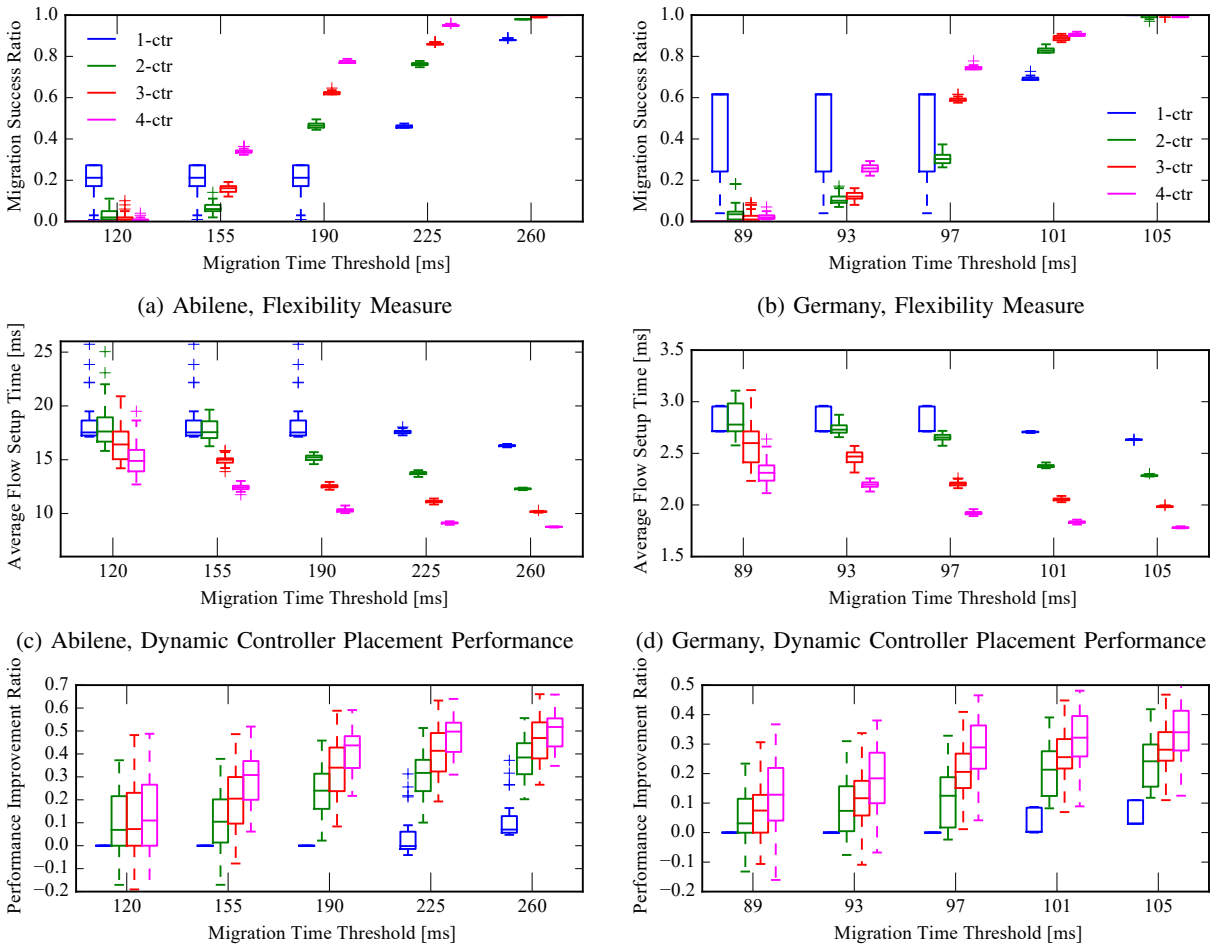
In general, the above observations show that 1-ctr provides more flexibility when the migration time T is small. It proposes not only a higher flexibility, but also a satisfying performance in most cases. However, more controllers outperform the single controller given a higher T, as they show a more flexible control plane as well as a improved flow setup performance.

V. Conclusion

A Dynamic SDN Control Plane can adapt its performance to changing traffic conditions via modifying the placement of its controllers. For analysis of different control plane settings, adaptive controller placement has to take controller migration as well as switch to controller re-assignment into account. In this paper, we model controller placement and migration time and evaluate the flexibility of a Dynamic SDN Control Plane with respect to the number of controllers. We refer to a flexibility metric that takes the time into account that is needed for successful migrations. Our evaluation results reveal that in case only a short time is allowed for controller migrations and switch re-assignments the number of controllers has almost no influence on the flexibility. For longer times, we confirm that, as expected, a higher number of controllers leads to more flexibility. Our results also underline the need for a Dynamic SDN Control Plane addressing changing traffic conditions and show that gains of up to 60% in performance can be achieved compared to the non-dynamic case. For future work, controller migration will be modeled in a more elaborate manner which takes controller's load into account. Other possible flexibility metrics for the SDN network design will be investigated.

Acknowledgment

This work is part of a project that has received funding from the European Research Council (ERC) under the European



(a) Abilene, Flexibility Measure

(b) Germany, Flexibility Measure

(c) Abilene, Dynamic Controller Placement Performance

(d) Germany, Dynamic Controller Placement Performance

(e) Abilene, Dynamic to Static Placement Performance Ratio

(f) Germany, Dynamic to Static Placement Performance Ratio

Fig. 2: Flexibility measure, performance and performance improvement ratio comparing different number of controllers and different T. Each column shows the three observations of one topology. X-axis represents the time constraint T. The legends are omitted in performance and performance difference ratio plots.

Union's Horizon 2020 research and innovation program (grant agreement No 647158 - FlexNets).

References

[1] C. Liang, R. Kawashima, and H. Matsuo, "Scalable and crash-tolerant load balancing based on switch migration for multiple open flow controllers," in *2014 Second International Symposium on Computing and Networking*. IEEE, 2014, pp. 171–177.

[2] S. Schmid and J. Suomela, "Exploiting locality in distributed sdn control," in *Proceedings of HOTSDN*. ACM, 2013, pp. 121–126.

[3] B. Heller, R. Sherwood, and N. McKeown, "The controller placement problem," in *Proceedings of HOTSDN*. ACM, 2012, pp. 7–12.

[4] W. Kellerer, A. Basta, and A. Blenk, "Using a flexibility measure for network design space analysis of sdn and nfv," in *Proceedings of INFOCOM workshop SWFAN*. IEEE, 2016.

[5] T. Koponen, M. Casado *et al.*, "Onix: A distributed control platform for large-scale production networks." in *OSDI*, vol. 10, 2010, pp. 1–6.

[6] P. Berde, M. Gerola *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.

[7] Opendaylight. [Online]. Available: http://www.opendaylight.org

[8] M. F. Bari, A. R. Roy *et al.*, "Dynamic controller provisioning in software defined networks," in *Proceedings of CNSM*. IEEE, 2013.

[9] 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.

[10] 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.

[11] C. Clark, K. Fraser, S. Hand, J. G. Hansen, E. Jul, C. Limpach, I. Pratt, and A. Warfield, "Live migration of virtual machines," in *Proceedings of the 2nd conference on Symposium on Networked Systems Design & Implementation-Volume 2*. USENIX Association, 2005, pp. 273–286.

[12] R. Bradford, E. Kotsovinos, A. Feldmann, and H. Schiöberg, "Live wide-area migration of virtual machines including local persistent state," in *Proceedings of the 3rd international conference on Virtual execution environments*. ACM, 2007, pp. 169–179.

[13] L. Vanbever, J. Reich, T. Benson, N. Foster, and J. Rexford, "Hotswap: Correct and efficient controller upgrades for software-defined networks," in *Proceedings of HOTSDN*. ACM, 2013, pp. 133–138.

[14] A. A. Dixit, F. Hao, S. Mukherjee, T. Lakshman, and R. Kompella, "Elasticon: an elastic distributed sdn controller," in *Proceedings of ACM/IEEE ANCS*. ACM, 2014, pp. 17–28.

[15] A. Basta, A. Blenk, H. B. Hassine, and W. Kellerer, "Towards a dynamic sdn virtualization layer: Control path migration protocol," in *Proceedings of CNSM*. IEEE, 2015, pp. 354–359.

[16] M. He, A. Basta, A. Blenk, and W. Kellerer, "Modeling flow setup time for controller placement in sdn: Evaluation for dynamic flows," in *Proceedings of IEEE ICC 2017*. IEEE, 2017.