More Is Not Always Better: An Analytical Study of Controller Synchronizations in Distributed SDN

Ziyao Zhang, *Student Member, IEEE,* Liang Ma, *Member, IEEE,*
Kin K. Leung, *Fellow, IEEE,* and Franck Le

***Abstract*—Distributed software-defined networks (SDN), consisting of multiple inter-connected network domains, each managed by one SDN controller, is an emerging networking architecture that offers balanced centralized control and distributed operations. In such networking paradigm, most existing works focus on designing sophisticated controller-synchronization strategies to improve joint controller-decision-making for inter-domain routing. However, there is still a lack of fundamental understanding of how the performance of distributed SDN is related to network attributes, thus impossible to justify the necessity of complicated strategies. In this regard, we analyse and quantify how the performance enhancement of distributed SDN architectures is influenced by inter-domain synchronization levels, in terms of the resulting number of abstracted routing clusters, and network structural properties. Based on a generic network model incorporating link preference for path constructions, we establish analytical lower bounds for quantifying the routing performance under any arbitrarily given network synchronization status. The significance of these performance bounds is that they can be used to quantify the contribution of controller synchronization levels in improving the network performance under different network parameters, which therefore serves as a fundamental guidance for future SDN performance analysis and protocol designs.**

***Index Terms*—Distributed SDN, SDN controller synchronization, performance evaluation, network analysis.**

I. Introduction

Software-Defined Networking (SDN) [1]–[3], an newly-deployed [4], [5] networking architecture, offers potentially significant network performance improvements due to its programmable network management, easy reconfiguration, and on-demand resource allocation, which has therefore attracted considerable research interests. One key attribute that differentiates SDN from classical networks is the separation of the SDN's data and control plane. Specifically, in SDN, all control functionalities are implemented and abstracted on the control plane for operational decision making, e.g., flow construction and resource allocation, while the data plane only passively executes the instructions received from the control plane. For a typical SDN architecture, all network decisions are made in the control plane by a control entity, called *SDN controller*, in a centralized manner. Since the centralized SDN controller has the full knowledge of network status, it is able to make global optimal decisions. Yet, such centralized control suffers from major scalability issues. In particular, as a network grows, high computation and communication requirements may impose substantial burden on the SDN controller, potentially resulting in significant performance degradation (e.g., delays) or even network failures [6].

In this regard, distributed SDN is proposed [7]–[11] to balance the centralized and distributed control. Specifically, a distributed SDN network consists of a set of subnetworks, referred to as *domains*, each managed by an independent SDN controller. Moreover, each domain contains several gateways connecting to some other domains to form the distributed SDN architecture. With such architectures, controllers exchange network information via proactive probing or passive listening to assist decision making for inter-domain tasks. In distributed SDN, network performance relies heavily on the inter-controller synchronizations. Since complete synchronization among controllers, i.e., each controller knows the network status in all other domains, incurs high synchronization costs especially in large networks, practical distributed SDN can only afford partial inter-domain synchronization.

For partial synchronization, most existing works focus on promoting the inter-domain synchronization level so that the final decision making approaches optimality. However, one fundamental question regarding the distributed SDN architecture has generally been ignored: *How does the network performance in distributed SDN relate*

to different network synchronization levels and structural properties? Without such fundamental understanding, it is impossible to justify why a complicated mechanism for information sharing or flow construction is necessary in distributed SDN. Furthermore, understanding the relative importance of different factors lays foundations for efficient and targeted protocol design. We, therefore, exploit analytical methods to investigate this unsolved yet critical problem in the distributed SDN paradigm, aiming at quantifying the performance metric and its relationship with various network parameters and controller synchronization levels.

To this end, we first propose a network model to capture intra-/inter-domain connections and network controller-specific link preference for path constructions in distributed SDN. Such network model is generic in that it is independent of any specific graph or synchronization models. Based on this network model, we then derive analytical expressions of the network performance, focusing on the average cost (see further discussions in Section III-C) of the constructed paths with respect to (w.r.t.) random flow requests, under any arbitrarily given network synchronization level. Analytical results reveal the relative contributions of different parameters to the lower bound of the performance metric. For example, the number of domains on constructed paths contributes linearly to the overall performance metric lower bound; while the number of gateway nodes in domains contribute logarithmically (see Theorem 8). Moreover, this lower bound shows that the performance metric is a linear function of the number of routing unit (called *routing cluster*, which abstracts the level of controller synchronization; see Definition 3) that are used for path constructions, but is independent of the detailed routing unit structures and distributions. Next, we quantify the tightness of such lower bound and reveal the interplay among different parameters for various network scenarios. By all these theoretical results, we prove that the contribution of network synchronization levels depends on specific network parameter settings, which therefore provides insights into real network protocol design. Finally, to validate the accuracy of the derived analytical expressions, they are compared against evaluation results obtained using simulation networks constructed using both real and synthetic network datasets.

A. Quick Background

Here, we brieftly introduce some conecpts and definitions related to distributed SDN and controller synchronization.

1) Controller State Information refers to controller's view of various status information of the network under its control. It also includes information about other networks gained through controller syncrhonzaiton.

2) Controller Synchronization refers to the process of distributed controllers exchanging their state inforamtion. Synchronization level is a general term used to describe to what extent controllers synchronize with each other. Currently, there are no unified standard on the paradigms through which controllers synchronize with each ohter. Hence, several synchronization mechanisms coexist with their respective strengths and weaknesses. See Section II-B for some examples. In this paper, we do not impose any assumptions on specific controller synchronizaitoon schemes being employed; instead, we assume that how controllers are syncrhonized is known to us for analysis.

3) Performance Metric: Since SDN has the capability in supporting many fine-grined network applications, the perofrmance metric depends on the scenario under consideration. We use Average Path Cost (APC), which generalizes over additive performance metrics, as the performance metric for our analysis. Therefore, our results are applicable to scenarios where the performance metrics are additive (see Section III-C).

B. Summary of Contributions

1) On top of the two-layer network model, we use the APC, measured by the average cost of the constructed path (see Section III-F), of the constructed paths as the performance metric to develop the analytical lower bounds of the APC, which are linear or even logarithmic functions of network structural or synchronization-related parameters;

2) We quantify the tightness of the APC lower bound expressions in *2)*, which shows that the APC lower bound can approach the actual APC value under some conditions;

3) Based on the analytical results, we draw revelations that relate the quality of the constructed paths to various network structural parameters and

the controller synchronization level. Insights for protocol design are also provided;

4) We evaluate our analytical results by extensive simulations using real and synthetic networks, both of which confirm their correctness and ability in quantifying the effectiveness of controller synchronizations under different network settings.

In this paper, our goal is to derive mathematical expressions to quantify the performance metric under any given synchronization levels. Therefore, we do not have any assumptions on how controller synchronizations take place; neither are we interested in proposing any new controller synchronization schemes. Moreover, we do not intend to design improved inter-domain routing mechanisms, and thus only basic and representative routing mechanism is employed for theoretical analysis. To the best of our knowledge, this is the *first work* that analyses the routing performance of distributed SDN in any given synchronization scenarios. The significance of these results is that they lay a strong theoretical foundation for optimizing distributed SDN and for synchronization protocol design.

The rest of the paper is organized as follows. Section II summarises and discusses existing works. Section III formulates the problem. Sections IV describes the inter-domain routing mechanism used for our analysis. Section V establishes the APC lower bounds and analyses their tightness under different network settings. Evaluations of the derived analytical expressions are conducted in Sections VI. Finally, Section VII concludes the paper.

II. Related Work

A. Information Sharing for Routing Quality Improvement

Researchers have looked into better understanding the performance of hierarchical routing where the internal structure of each domain is not revealed to outside nodes. For example, [12] shows that hierarchical routing where the topologies of the clusters are hidden can lead to suboptimal routing, and forwarding loops; [13] proposes solutions to aggregate topologies with theoretical bounds. Moreover, [14] analyses the effectiveness of hierarchical routing (e.g., ATM PNNI [15], Nimrod [16]). However, most of these early works are either driven by simulations or with different focus. In contrast, we conduct a rigorous mathematical analysis to understand the benefits of controller synchronization in distributed SDN. It should be noted that although some of the theoretical results presented in this paper could be applied to the analysis of legacy networks under certain conditions, our work is SDN-focused because many of the assumptions we have for modelling can only be realized through fine-grained control under SDN.

B. Distributed SDN

The distributed SDN architecture, which integrates the advantages in early hierarchical networks, have stimulated many research efforts in this area. Specifically, the feasibility of deploying SDN-based mechanisms incrementally to the current BGP-glued Internet is considered in [7], where routing control planes of multiple domains are outsourced to form centralized control planes for optimizing routing decisions. In addition, protocols and systems, such as HyperFlow [17], DISCO [18], and ONOS [19], are proposed to realize logically centralized but physically distributed SDN architecture. Devoflow [20] and Kandoo [21] are designed to reduce the overheads introduced by the interaction between the control and data planes; while DIFANE [22] and Fibbing [23] are conceived for limiting the level of centralization and addressing robustness issues, respectively. However, most of these works are experiment-based. By contrast, our goal is to investigate distributed SDN from the perspective of fundamental analytics.

C. Performance Analytics

Since all theoretical results in this paper are obtained based on a graph model, our work is related to the area of graphical analysis of complex networks. Most works in this area are dedicated to the study of specific graph properties, e.g., small-world effect [24], network motif [25], scale-free [26], etc. On the other hand, models in [24], [27]–[29] are purely randomized, which cannot differentiate intra-/inter-domain links in the context of distributed SDN. Thus, they are substantially different from our work. Our approach is also comparable to [30] as they also consider a layered-network model for the study of communication networks. However, the authors are mainly concerned with modelling the co-existing connectivity. Our previous work [31] also aims at quantifying the benefits of

controller synchronizations from the fundamental analytical perspective; however, it differs from this work significantly on the methodologies employed and the synchronization scenarios considered.

III. Problem Formulation

A. Network Model

We formulate the distributed SDN network as an undirected graph according to a two-layer network model (Fig. 1), where (i) the top-layer abstracts the inter-domain connections, and (ii) the bottom-layer characterizes physical connections among all network elements under the inter-domain connection constraint in the top-layer. Specifically, the top-layer is a graph $\mathcal{G}_d = (V_d, E_d)$ (V_d/E_d: set of vertices/edges) mapped directly from the real inter-domain connections in a given distributed SDN network: (i) each vertex in V_d represents a domain, (ii) two vertices are connected by an edge if and only if their corresponding domains are directly connected by communication links. We refer to $\mathcal{G}_d$ as the *domain-wise topology* in the sequel.

Based on the above domain-wise topology, we next model the physical network in the bottom-layer. In particular, domain i in $\mathcal{G}_d$ corresponds to an undirected graph with n nodes in the bottom-layer; these n nodes are connected following a given *intra-domain degree distribution*, which is the distribution of the number of neighbouring nodes of an arbitrary node within the same domain. [1] We also assume that such intra-domain degrees across all domains are independently and identically distributed (i.i.d.). The graph of each domain is referred to as *intra-domain topology*. Then for each $e \in E_d$ with end-points corresponding to domains $\mathcal{A}_i$ and $\mathcal{A}_j$, we (i) randomly select two nodes w_1 from $\mathcal{A}_i$ and w_2 from $\mathcal{A}_j$ and connect these two nodes if link w_1w_2 does not exist, and (ii) repeat such link construction process between $\mathcal{A}_i$ and $\mathcal{A}_j$ β times. By this link construction process, the bottom-layer network topology $\mathcal{G} = (V, E)$ is therefore formed (V/E: set of nodes/links in $\mathcal{G}$, $|V| = n|V_d|$) (Fig. 1). Without loss of generality, we assume that all inter/intra-domain topologies are *connected graphs*.

Note that the above process indicates that the i-th selected link may overlap with existing links (i.e., the same end-points); therefore, parameter β represents the maximum number of links between any two domains. In each domain, nodes having connections to other domains are called *gateways*. Hence, if domains $\mathcal{A}_i$ and $\mathcal{A}_j$ are connected in the domain-wise topology, then the expected number of gateways in $\mathcal{A}_i$ connecting to $\mathcal{A}_j$, denoted by γ, is $\gamma = n(1-(1-1/n)^{\beta})$.

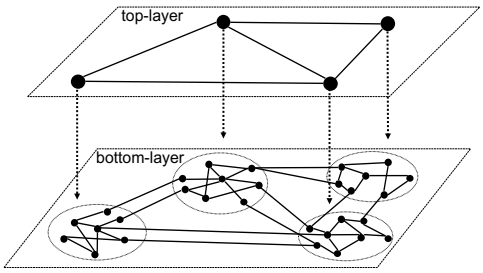


Fig. 1: Two-layer network model: Top-/bottom-layer abstracts domain-wise topology/all physical connections, respectively.

Discussion: Our two-layer network model is generic in that the input can be any domain-wise topology and node degree distribution, empirical or extracted from real networks. Note that in practice, gateways are a fixed set of nodes in a domain; our random gateway selection only indicates that the gateway locations can be anywhere in the context of network graphs.

B. SDN Data and Control Plane

Thus far, we have only discussed the graphical properties of the distributed SDN networks. One critical aspect of SDN that differentiates it from other networks is the separation of the data and control planes, which are formulated as follows.

1) Data Plane: We exploit graph $\mathcal{G}$ generated by the two-layer network model in Section III-A to represent the data plane of the distributed SDN. Specifically, a node/link exists in $\mathcal{G}$ if and only if it can be used for data transmission in the network.

2) Control Plane: We assume that in this two-layer network model, each domain contains one logical SDN controller that carries out control operations and facilitates information sharing. Note that for our analysis, we do have any assumptions on how SDN cotrollers are implemented in real systems, i.e., the domain controllers can be a single unit or a collection of physically distributed units. It should be noted that the number of physical controllers in a domain can have impacts on synchro-

[1] In one domain, some nodes may have connections to other domains; such external connections are not considered in the concept of intra-domain degree.

nization overheads and other factors. However, this work focuses on inter-domain controller synchronization. As such, we assume that each domain has one logical controller. SDN controllers together with all inter/intra-domain controlling channels form a control plane. Under the SDN architecture, to construct a path between a pair of source and destination nodes, the corresponding routing path is logically determined by the controllers in the source, destination, and all intermediate domains collectively, using the synchronized information among them. The performance of the constructed paths may vary, depending on the network status information at each involved controller (see Section IV).

C. Link Preference and Path Cost

In the distributed SDN architecture, a routing path construction between a pair of nodes is determined by all involved controllers. To reach an optimized routing decision, controllers need to take into account the traffic status, load balancing, and other policy-related factors. To this end, controllers can proactively assign a weight to each link to indicate the link preference based on the collected network information, i.e., the smaller the link weight, the better for path construction, so that the end-to-end accumulated weight of any path matches its corresponding path construction preference. Moreover, such link weight assignment is generally adjusted dynamically according to the current network condition. Therefore, the goal for constructing an optimized end-to-end path under a given network condition is reduced to finding the end-to-end path with the minimum accumulated weight under the given link weight assignment. We refer to such accumulated path weight as the *path cost*.

Since the link preference (weight) can be dynamic, we use random variables to capture its dynamicity. Specifically, we assume that intra-domain link weights across all domains are at least 1 and i.i.d. Furthermore, in real distributed SDN environment, unlike the intra-domain links which are potentially wireless, inter-domain gateway-to-gateway links are likely to be wired with high bandwidth, thus more stable. In this regard, we characterize all inter-domain link weights by a non-negative constant C. Furthermore, without loss of generality, we assume $C = 1$; all our theoretical results can be trivially extended to other values of C, if the weights of inter-domain links can be captured by random variables with certain distributions.

It should be noted that the concept of APC generalizes over many performance metrics. Our analytical results in this paper can be directly applied to any additive performance metric. For example, when routing performance is delay (additive metric), the domain controller simply assigns as link weights the delays of intra-domain links under the given traffic levels. Then, APC in this case corresponds to the average end-to-end delay of constructed paths.

D. Synchronization Among SDN Controllers

In distributed SDN, the controller synchronized information directly affects the quality of the constructed paths. Thus, we formally define *synchronization* among SDN controllers.

Definition 1. *Domain $\mathcal{A}_i$ is* synchronized *with domain $\mathcal{A}_j$ if and only if the SDN controller in $\mathcal{A}_i$ knows the minimum path cost between any two nodes in $\mathcal{A}_j$.*[2] *$\mathcal{A}_i$ and $\mathcal{A}_j$ are mutually synchronized if $\mathcal{A}_i$ is synchronized with $\mathcal{A}_i$ and $\mathcal{A}_j$ is synchronized with $\mathcal{A}_i$.*

In this paper, we do not have any requirements on how controllers are synchronized with each other. Instead, we assume that the synchronization status, i.e., which domains are synchronized with which other domains, is known to us for analysis. Moreover, we assume that each controller always has the up-to-date complete view (i.e., any path cost) of its own domain by proactive polling or passive information collection. Moreover, we assume that the domain-wise topology is always known to every controller irrespective of the inter-domain synchronization status. In real networks, such domain-wise topology can be identified by techniques such as the BGP protocol.

E. Routing Mechanisms

With the distributed SDN architecture and the given inter-controller synchronization status, we then need a routing mechanism to respond to a path construction request between two arbitrary nodes (potentially in two different domains). The aim of the routing mechanism is to minimize the associated

[2]Note that the internal topologies (including network node status) need not to be shared for privacy protection.

path cost under the given network synchronization status. The general rule governing the routing mechanism is to first determine which domains are involved by finding a *domain-wise path*, defined as:

Definition 2.

a) *In the domain-wise topology $\mathcal{G}_d$, the top-layer vertex corresponding to domain $\mathcal{A}$ in $\mathcal{G}$ is denoted by $\vartheta(\mathcal{A})$;*

b) *Given a pair of source and destination nodes v_1 and v_2 with* [3] *$v_1 \in \mathcal{A}_1$, $v_2 \in \mathcal{A}_2$, and $\mathcal{A}_1 \neq \mathcal{A}_2$, the* domain-wise path *w.r.t. v_1 and v_2 is a path (not necessarily the shortest) in $\mathcal{G}_d$ starting at vertex $\vartheta(\mathcal{A}_1)$ and terminating at vertex $\vartheta(\mathcal{A}_2)$ without containing any repeated vertices.*

Note that although the information of network domain-wise topology is always known, domain-wise paths are selected according to the given network policies (e.g., load balancing, security issues, etc.). The routing mechanism constructs a path segment in each involved domain, and then concatenate all these segments into one end-to-end path; see Section IV for the details of the routing mechanism.

F. Problem Statement and Objective

Regarding a path construction request for two nodes, after the (policy-based) selection of a domain-wise path, the path construction quality depends on the synchronization status of the involved domains. For instance, if every two domains on this domain-wise path are synchronized, then the minimum end-to-end path can be constructed; however, if no domains are synchronized, then the path construction falls back to follow BGP-like protocols. To quantify the performance of the constructed routing path in a selected domain-wise path under any given inter-domain synchronization status, we employ the *Average Path Cost (APC)*, measured by the average cost of the constructed path, as the performance metric. Here APC is a natural generalized performance metric, as link weights are dynamically adjusted by controllers based on the current network status to reflect time-varying link preference, and APC is equivalent to the number of hops in unweighted networks. Formally, our research objective is:

[3]In this paper, for graph $\mathcal{G} = (V, E)$, by abusing graph theory notations, we use vertex $v \in \mathcal{G}$ to denote $v \in V$ and edge $e \in \mathcal{G}$ to denote $e \in E$.

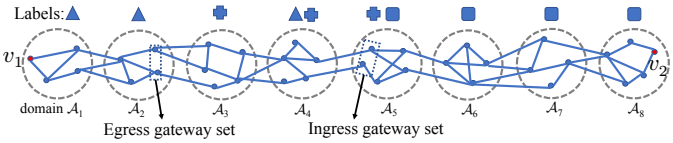


Fig. 2: Sample domain-wise path between source/destination pair v_1/v_2; domains with the same labels are synchronized.

Objective: Given the distributed SDN network model, suppose each network instance following the two-layer network model exists with the same probability. For an arbitrary source-destination node pair with their selected domain-wise path containing m domains ($m \geq 2$), our goal is to derive the mathematical expression of APC for these node pairs under any arbitrarily given network synchronization scenario.

Note that our two-layer network model is a random graph model, i.e., there exists multiple network realizations satisfying the same set of input parameters. Therefore, APC is an expected value over not only random source/destination node pairs (in a domain-wise path containing a particular number of domains) but also random network realizations. All our theoretical results on APC are based on the given network parameters rather than a specific network realization.

Example: Fig. 2 shows a sample domain-wise path consisting of 8 domains ($m = 8$) w.r.t. a path construction request between two random nodes $v_1 \in \mathcal{A}_1$ and $v_2 \in \mathcal{A}_8$, where all intra-/inter-domain connections are captured by the two-layer model. Moreover, Fig. 2 also illustrates one sample synchronization status among these 8 domains. In particular, there is only partial syncrhonizaiton among these domains where domains sharing the same labels are mutually synchronized. Since the number of synchronization scenarios among these 8 domains can be up to $2^{\binom{8}{2}}$, it potentially affects the performance of the constructed path. We therefore aim to quantify APC between v_1 and v_2 for any given m and synchronization scenario.

IV. Routing Cluster-based Path Construction

The prerequisite for the analysis of APC between two random nodes is a routing mechanism that leverages the given inter-domain synchronized information. To this end, we describe a path construction mechanism called *Routing Cluster-based*

Path Construction (RCPC). It should be noted that our intention here is *not* to design an optimal routing mechanism. Instead, our goal is to quantify APC under a basic and representative routing mechanism. In cases where improved routing mechanisms are applied to distributed SDN, our theoretical results can serve as a performance bound.

A. Routing Clusters

As the name suggests, RCPC is based on the concept of *routing clusters*, which is defined below.

Definition 3. *For a domain-wise path with all involved domains sequentially labeled as* $\mathcal{A}_1, \mathcal{A}_2, \ldots, \mathcal{A}_m$*, domains* $\mathcal{A}_{i+1}, \mathcal{A}_{i+2}, \ldots, \mathcal{A}_{i+c}$ *(*$0 \leq i \leq m-c$*) that are mutually synchronized form a* Routing Cluster (RC) *containing* c *domains.*

The concept of RC provides a method to leverage the given inter-domain synchronized information. Specifically, domains belonging to the same RC are all synchronized. Therefore, every controller in this RC is able to determine the optimal intra-RC path w.r.t. any two nodes within this RC, i.e., controllers within the same RC can be regarded as one logical controller. On the other hand, to ensure the optimality of the constructed intra-RC path, we also require the indices of the involved domains be continuous. This can be explained by the example in Fig. 2, where although $\mathcal{A}_1$, $\mathcal{A}_2$, and $\mathcal{A}_4$ are all synchronized, it is impossible for controllers in these three domains to compute the minimum cost path for any node pairs within them, because the path cost information in $\mathcal{A}_3$ is unknown.

Next, to efficiently utilize the synchronized information on the domain-wise path, we also need to partition all involved domains into different RCs. Note that two RCs may have overlapped domains. If this happens, the controller in the overlapped domain may have conflicts in determining path constructions under different RC memberships. Hence, we only consider non-overlapping RCs as follows.

Definition 4. RC partition *of domains on a given domain-wise path is a set* $\mathcal{R}$ *of RCs, such that (i) each domain on this domain-wise path belongs to one and only one RC, and (ii)* $|\mathcal{R}|$ *is minimized.*

In Definition 4, the first condition guarantees that RCs do not overlap, and the second condition ensures there are as many domains as possible in each RC, so that the optimal path traversing multiple domains (within the same RC) can be found. For instance, in Fig. 2, one way for RC partition is $\text{RC}_1 = \{\mathcal{A}_1, \mathcal{A}_2\}$, $\text{RC}_2 = \{\mathcal{A}_3, \mathcal{A}_4, \mathcal{A}_5\}$, and $\text{RC}_3 = \{\mathcal{A}_6, \mathcal{A}_7, \mathcal{A}_8\}$. Note that the RC partition is not unique. For example, another way for RC partition in Fig. 2 is $\text{RC}_1 = \{\mathcal{A}_1, \mathcal{A}_2\}$, $\text{RC}'_2 = \{\mathcal{A}_3, \mathcal{A}_4\}$, and $\text{RC}'_3 = \{\mathcal{A}_5, \mathcal{A}_6, \mathcal{A}_7, \mathcal{A}_8\}$. In Section V, we will show how different RC partitions (e.g., the number of RCs and the size of each RC) affect the APC. Based on the RC partition, we are now ready to present the routing mechanism - RCPC.

Discussion: Before the introduction of RCPC, here we briefly disucss the relationship between synchronizaiton level and RC partitions. By Definition 3 and 4, we can see that the number of RCs is smaller when synchronizaiton level is high. For example, in Fig. 2, the highest possible synchronizaiton level is when $\mathcal{A}_1$ is synchronized with all other domains on the path. In this case, there is only one RC. In the other extreme scenario where no controller is synchronizaed , there will be 8 RCs. Therefore, the level of controller synchronization is reflected by the number of domains in routing clusters and the number of routing clusters on the constructed paths.

B. Routing Cluster-based Path Construction (RCPC)

RCPC is described in the following steps:

Step 1) W.r.t. two nodes v_1 and v_2 on a selected domain-wise path containing m domains, identify the domains on this domain-wise path;

Step 2) Based on the given synchronization status of all involved domains, partition these domains into the minimum number of μ ($1 \leq \mu \leq m$) RCs according to Definition 4;

Step 3) For each RC_i, a path segment starting from the entering node (which is v_1 if $i = 1$, or is specified by RC_{i-1}) and terminating at one of the exiting nodes (which are gateways connecting to RC_{i+1}, or node v_2 if $i = \mu$) with the minimum cost is constructed. Such path segment is denoted by $\mathcal{P}_i$ in RC_i. Also let $e_{i,i+1}$ be the edge leading from $\mathcal{P}_i$ in RC_i to connect to the entering node in RC_{i+1} if $i \leq \mu - 1$;

Step 4) The final v_1-to-v_2 path $\mathcal{P}$ is $\mathcal{P} = \mathcal{P}_1 + e_{1,2} + \mathcal{P}_2 + e_{2,3} + \ldots + \mathcal{P}_{\mu-1} + e_{\mu-1,\mu} + \mathcal{P}_\mu$.

In essence, RCPC intends to minimize the path cost at each RC. As such, ECMP or similar schemes

could be applied within RCs for traffic balancing or other control objectives; since equal intra-RC costs would incur as the result, our analytical results still hold. The fact that, instead of each domain making routing decisions based on its view of the network, path construction inside one RC is agreed using commonly synchronized information and carried out consistently in all domains within the RC is to avoid routing loops and other anomalies which are likely to arise because of the mixed centralized-distributed routing. This is analogous to the routing mechanism in ONOS [19] and OpenDaylight [32] in the sense that, for these two controller architectures, the domains on a domain-wise path essentially form one RC and they jointly install the same set of forwarding rules.

V. APC UNDER ANY GIVEN SYNCHRONIZATION STATUS

With RCPC, we are ready to analyse APC of the path constructed between source and destination nodes, and investigate its relationship with various network structural and synchronization-related parameters. In particular, we first derive the expressions of the lower bound of APC. These results explicitly show how different parameters interact with each other in determining the overall APC. In addition, to guarantee the tightness of our developed APC lower bounds, we also provide the maximum gap between the actual APC and the derived APC lower bounds. The significance of such APC lower bounds is that they provide insights into how the constructed path quality relates to any given network parameters without assuming any inter-domain synchronization models.

The basic idea of our derivation is to explore how path construction behaviours differ in different types of domains in each RC, and understand how such behaviours are influenced by intrinsic topological properties and the synchronized information. Specifically, we give a sketch of our methodology.

Sketch of Analytical Methodology:

1) Given the domain-wise path with m domains, and the corresponding RC partitions, identify different types of domains (see Definition 5) based on their relative positions in RCs;

2) Derive the APC lower bound for different types of domains and add these lower bounds together to obtain the overall APC lower bound;

3) Quantify the maximum gap between the actual APC and APC lower bound by comparing the APC lower bound with the APC incurred without synchronized information.

Before discussing technical details, for ease of presentation, we first formally introduce the following definitions.

Definition 5.

a) Source/destination domain: *the domain where the source/destination node is located;*

b) Type-1 domain: *the domain in an RC where the constructed path segment in this RC starts; the source domain is also a Type-1 domain;*

c) Type-2 domain: *any domain that is not a Type-1 domain or the destination domain;*

d) Ingress and egress gateway sets: *the sets of gateways in a domain through which data packets can enter or leave this domain w.r.t. the domain-wise path selected between the source and the destination node.*

Examples: In Fig. 2, based on the RC partition $\text{RC}_1 = \{\mathcal{A}_1, \mathcal{A}_2\}$, $\text{RC}_2 = \{\mathcal{A}_3, \mathcal{A}_4, \mathcal{A}_5\}$, and $\text{RC}_3 = \{\mathcal{A}_6, \mathcal{A}_7, \mathcal{A}_8\}$, $\mathcal{A}_1$, $\mathcal{A}_3$ and $\mathcal{A}_6$ are Type-1 domains; $\mathcal{A}_2$, $\mathcal{A}_4$, $\mathcal{A}_5$ and $\mathcal{A}_7$ are Type-2 domains. The ingress and egress gateway sets are illustrated in $\mathcal{A}_5$ and $\mathcal{A}_2$ in Fig. 2, respectively.

Discussion: The difference between Type-1 and Type-2 domains is that the starting nodes in Type-1 domains are not decided by controllers in their associated RCs, since the starting node is either the source node or an ingress gateway chosen by the previous RC. In contrast, both ingress and egress gateways in Type-2 domains are jointly decided by controllers in their associated RCs, using the synchronized information they all share. As for the destination domain, if it constitutes an RC itself, then the ingress gateway is decided by the previous RC; otherwise, it is decided jointly by all controllers within its associated RC. Whether or not the starting node of a domain can be manipulated determines the minimum APC achievable in this domain. Specifically, we derive the minimum APC traversed in different types of domains in Theorem 7, which then serves as the lower bound. *Note that the proofs of all theorems in this paper and the lemma, proposition used therein are documented in the appendix in the supplementary material attached to this paper.* Before presenting Theorem 7, we first define the path cost between two sets of nodes as follows.

TABLE I: Main Notations

Notation	Meaning
m	the number of domains on the domain-wise path
n	the number of nodes in each domain
γ	the average number of ingress or egress gateways in a domain w.r.t. a domain-wise path
z_1, z_2	average number of nodes within the same domain that are 1-hop and 2-hop away from a random chosen node
μ	the number of Routing Clusters (see Definition 3) on the domain-wise path
ϕ	$\phi = n/z_1$
ξ	the probability that the ingress node is not in the egress gateway set in a domain
ζ	the probability that the ingress and egress gateway sets in a domain do not have common elements
ω	maximum link preference (weight) in the network

Definition 6. *The* path cost between node sets W and S *is the minimum path cost among all paths in set* $\{\mathcal{P}_{ws}$*: minimum cost path between node* w *and node* s, $w \in W, s \in S\}$.

Theorem 7. *Let* l_1 *denote the APC between the starting node and the egress gateway set for Type-1 domains, and let* l_2 *denote the APC between the ingress and egress gateway sets for Type-2 domains. Then, when* $\omega = 1$,

$$l_1 \geq \begin{cases} \zeta\big(\frac{\log(\phi/\gamma)}{\log(z_2/z_1)} + 1\big) & \text{if } \gamma \leq \phi, \\ \zeta & \text{otherwise}, \end{cases}$$

and

$$l_2 = \begin{cases} \zeta\big(\frac{\log(\phi/\gamma^2)}{\log(z_2/z_1)} + 1\big) & \text{if } \gamma \leq \sqrt{\phi}, \\ \zeta & \text{otherwise}, \end{cases}$$

where ζ *is the probability that the ingress and egress gateway sets in a domain do not have common elements, and other notations are defined in Table I.*

In Theorem 7, according to the definition of ζ, its value is $\zeta = \Pi_{i=1}^{\gamma-1}(1 - \frac{\gamma}{n-i})$ if $\gamma \leq n/2$, or $\zeta = 0$ if $\gamma > n/2$. Note that Theorem 7 is derived by setting all link weights to 1 (i.e., equal link preference). In this way, the mathematical expression of APC is simplified. More importantly, since link weights are at least 1 in the distributed SDN model, all derived APC lower bounds are still valid in graphs with dynamic weights. Theorem 7 tells us that the minimum APC exhibits different behaviours, depending on the range of the number of ingress/egress gateways. As such, we discuss the APC lower bound in the following three regimes of γ: *sparse inter-domain connections* ($\gamma \leq \sqrt{\phi}$), *medium inter-domain connections* ($\sqrt{\phi} < \gamma \leq \phi$), and *dense inter-domain connections* ($\gamma > \phi$), where $\phi = \frac{n}{z_1}$.

A. Sparse Inter-domain Connections ($\gamma \leq \sqrt{\phi}$)

We present the APC lower bound when the number of gateways is relatively small, i.e., $\gamma \leq \sqrt{\phi}$.

Theorem 8. *When* $\gamma \leq \sqrt{\phi}$*, the APC lower bound in a domain-wise path with* m *domains partitioned into* μ *RCs, denoted by* L^{LB}*, is*

$$L^{LB} = \frac{\mu\zeta\log(\gamma) + m\zeta\log(\frac{\phi z_2/z_1}{\gamma^2})}{\log(z_2/z_1)} - 1.$$

Before we draw observations from the expression of L^{LB}, we first show the tightness of such lower bound to justify its close representation of the APC. The basic idea behind the derivation of the gap between APC and APC lower bound is to find a tractable APC upper bound, and use the difference between this APC upper bound and L^{LB} to capture the maximum gap. Such APC upper bound is estimated by assuming that each domain is an RC, i.e., no inter-domain synchronized information is available. Although this APC upper bound fails to capture controller synchronizations, it helps us quantify the tightness of derived APC lower-bound. The tightness guarantee is provided in the following theorem.

Theorem 9. *Let* L *denote the APC in a domain-wise path with* m *domains partitioned into* μ *RCs, and* ω *the maximum link weight in the network. Then, when* $\gamma \leq \sqrt{\phi}$*, the gap between* L *and* L^{LB} *is bounded by*

$$L - L^{LB} \leq \frac{\rho\log(\gamma) + \nu\log(\phi)}{\log(z_2/z_1)},$$

where $\rho = \omega\xi(1-m) + \zeta(2m-\mu)$, $\nu = \omega(\xi(m-1)+\zeta) - \zeta m$.

From the expressions in Theorems 8-9, they show that neither APC lower bound nor the APC tightness expression are affected by the specific number of domains in each RC. That is to say, the quality of two paths constructed by RCPC are similar, as long as they have the same number of domains and RCs. Furthermore, the APC lower bound grows linearly in both μ and m. Nevertheless, as γ increases, the influence of μ becomes more dominant according to Theorem 8. In contrast, m is more important when γ is small. Intuitively, this is because when the number of gateways is small, there are not

many gateway options for inter-domain routing, and thus controller synchronizations do not improve the quality of routing that much. In other words, even a random selection of gateways has a relatively high probability of hitting the best choice. This is also echoed by Theorem 9 where the upper bound of $L-L^{\text{LB}}$, which represents the gap between the actual APC and the derived APC lower bound, increases when γ gets larger, as the informed routing makes a significant difference when there are more gateways to choose from. Moreover, by Theorem 9, in the extreme case where there is no inter-domain synchronizations (i.e., $m = \mu$) and all link weights are 1 (i.e., $\omega = 1$), with small number of gateways, we have $L - L^{\text{LB}} \leq \log(\gamma)/\log(z_2/z_1) \approx 0$. Therefore, L^{LB} becomes the accurate expression of APC.

Insights for protocol design: When inter-domain connections are sparse, having more inter-domain connections enhances the benefit of improved inter-domain synchronizations. On the other hand, when the number of inter-domain connections is really small, efforts should be diverted to increase domain-wise topological connectivity to reduce m (average number of domains on domain-wise paths), which will contribute more in reducing APC, due to the effect of the reduction of m is magnified by a relatively large coefficient $\log(\frac{\phi z_2/z_1}{\gamma^2})$ of m.

Although most exisiting controller synchronization mechanisms do not consider domain-wise connectivity levels in their designs; our analysis reveals that when there are not sufficient inter-domain connections, the differences in performance resulted from various controller syncrhonization paradigms are small. Therefore, under such network conditions, the focus of system implementation should be more on improving inter-domain connectivity than designing complicated synchronization schemes.

B. Medium Inter-domain Connections ($\sqrt{\phi} < \gamma \leq \phi$)

Similar to the previous case, we also provide the APC lower bound and its tightness guarantee in the case of medium inter-domain connections.

Theorem 10. *When $\sqrt{\phi} < \gamma \leq \phi$, the APC lower-bound in a domain-wise path with m domains partitioned into μ RCs, denoted by L^{LB}, is*

$$L^{LB} = \frac{\mu\zeta \log(\frac{\phi}{\gamma z_2/z_1})}{\log(z_2/z_1)} + m(\zeta+1) - 1.$$

Theorem 11. *Let L denote the APC in a domain-wise path with m domains partitioned into μ RCs. Then, when $\sqrt{\phi} < \gamma \leq \phi$, the gap between L and L^{LB} is bounded by*

$$L - L^{LB} < \frac{\tau \log(\phi/\gamma) + \omega \log(\phi)}{\log(z_2/z_1)} - \zeta(m-\mu),$$

where $\tau = \xi\omega(m-1) - \zeta\mu$.

In the medium inter-domain connection regime ($\sqrt{\phi} < \gamma \leq \phi$), although L^{LB} still grows linearly in μ and m, one noticeable difference comparing to sparse inter-domain connections is that the influence of μ diminishes when γ is large, which is the opposite of the case in Theorem 8. This suggests a weakening role of synchronization (i.e., μ) in reducing APC when γ is large. By closer examinations, this can be explained by the fact that the relative abundance in egress gateways significantly reduces the APC between non-gateway nodes and gateways. One consequence of this is that even most domains operate only on the knowledge of their own domains without any inter-domain synchronized information, i.e., select the gateway that incurs the minimum path cost in each domain, the overall APC of paths constructed in such way is not much worsened. This revelation is also confirmed by the fact that the coefficient of m are only related to the intrinsic property of the network. Another difference comparing to the sparse inter-domain connection case is that the parameter $\phi = \frac{n}{z_1}$, which is an indicator of the intra-domain graph connectivity level, is now part of the coefficient of μ. This suggests that high network connectivity also reduces the impact of a small μ (i.e., richer synchronized information among controllers).

Insights for protocol design: In medium inter-domain connections, improving controller synchronizations is still helpful in reducing APC. However, the effectiveness of it is dwindling. Therefore, the synchronization policy must consider the connectivity level insde domains. Specifically, when the intra-domain connectivity level is relatively low, increasing the inter-controller synchronization level is more rewarding.

Comparing to the analysis in the previous section where we emphasize the important role of domain-wise connectivity, here we highlight the impact of intra-domain connectivity for medium inter-domain connection regimes. In particular, it is suggested that SDN domains with relative low intra-domain

connection levels benefit more from controller synchronizations. Therefore, these domain controllers should be given priorities for synchronization.

C. Dense Inter-domain Connections ($\gamma > \phi$)

When the number of inter-domain connections is significantly large, the contribution of the increased inter-controller synchronization level continues diminishing, as proved below.

Theorem 12. *When $\gamma > \phi$, the APC lower-bound in a domain-wise path with m domains partitioned into μ RCs, denoted by L^{LB}, is*

$$L^{LB} = (1+\zeta)m - 1.$$

Theorem 13. *Let L denote the APC in a domain-wise path with m domains partitioned into μ RCs. Then, when $\gamma > \phi$, the gap between L and L^{LB} is bounded by*

$$L - L^{LB} < \frac{(\omega+\chi)\log(\phi) - \chi\log(\gamma)}{\log(z_2/z_1)} + \zeta\big((m-1)\omega - m\big),$$

where $\chi = \omega(\xi - \zeta)(m-1)$.

When $\gamma > \phi$, μ (which represents the level of controller synchronizations) disappears from the expressions in Theorems 12 and 13. This suggests that in the dense inter-domain connection regime ($\gamma > \phi$), the role of controller synchronization matters little. This is the continuation of the trend observed in the medium inter-domain connection regime ($\sqrt{\phi} < \gamma \leq \phi$) where the influence of controller synchronizations is discounted as γ increases. Intuitively, this is because when γ is considerably large, every node inside a domain has a high probability to connect directly to nodes in its neighbouring domains, thus yielding the high probability that the ingress and egress gateways are the same in each domain.

Insights for protocol design: When the inter-domain connections are dense, the synchronization of controllers achieves little in reducing the APC. Without additional synchronization cost, a distributed routing algorithm, such as BGP, becomes a fair alternative in this case. Moreover, from Theorem 13, it is evident that the performance gap between APC and the APC lower bound is approximately the APC between two arbitrary nodes in the destination domain multiplied by the worst case link weight when γ is large (i.e., $\xi \approx 0$ and $\zeta \approx 0$; see the proof of Theorem 13 in the appendix). This can be explained as follows: For a domain-wise path, the destination domain is the only domain that has a specified ingress gateway and a destination node, i.e., the destination domain does not have gateway options which may incur lower cost. As such, the destination domain becomes the bottleneck when γ is large. Therefore, the destination domain should be the focus of the synchronization design when inter-domain connections are dense.

Overall, we can see that the influence of controller synchronization first increases and then decreases as the inter-domain connection levels increase. We also observe how intra-domain connectivity levels can impact the benefits brought by controller synchronizaitons. These revelations serve as guidelines for controller syncrhonization policy designs in real distributed SDN systems.

VI. Evaluations

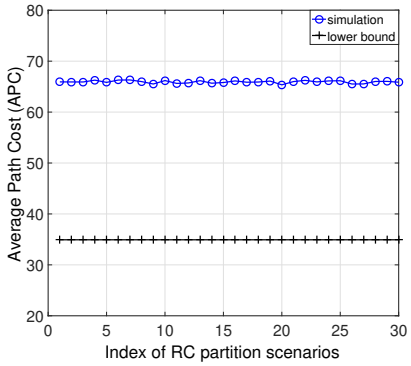
A series of experiments based on network topologies generated from both real and synthetic datasets are conducted in this section.

A. Network Realizations

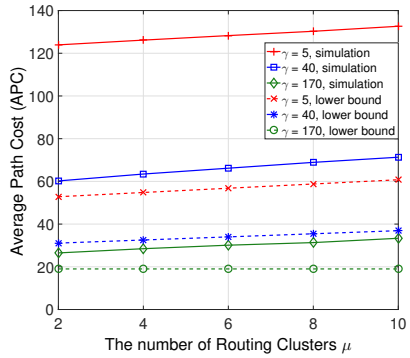
1) Network Topologies Based on Real Datasets: To generate network topologies based on real datasets, we refer to the Rocketfuel project [33] for the input node degree distribution to generate the network topology for each domain.

Given a specific node degree distribution, one graph realization is generated in the following way: We assign each vertex a target degree according to the degree distribution. We then select two vertices randomly and add an edge between them; the number of edges added w.r.t. each vertex is then recorded. If the degree target w.r.t. a vertex is met, this vertex will not be selected again to connect with other vertices. Such process repeats until all vertices reach their degree targets.

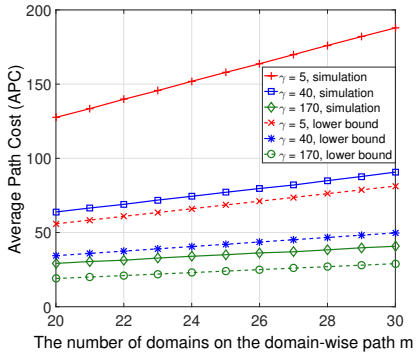
2) Network Topologies Based on Synthetic Models: In our evaluations, we test the trend of APC changes with varying network parameters and compare them with our derived APC lower bounds. Since we are not always able to obtain real network datasets with desired varying network parameters that meet our requirement, we also use network topologies generated by synthetic datasets according to certain network models. Specifically, we select Erdös-Rényi [27] model to generate network topologies for some evaluations.



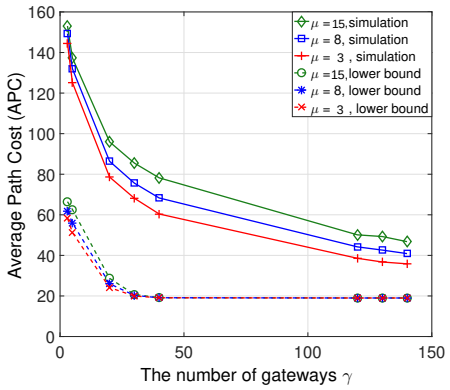
(a) Scenario 1: APCs of paths constructed with different RC partitions ($m = 20$, $\mu = 5$).



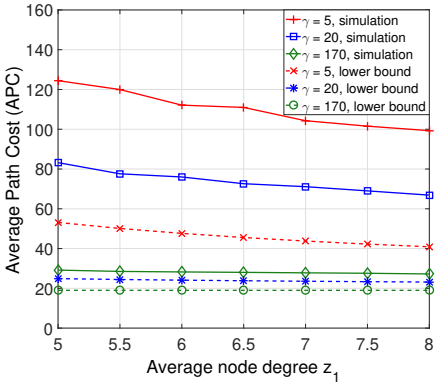
(b) Scenario 2: APCs of paths constructed with increasing number of RCs ($m = 20$).



(c) Scenario 3: APCs of paths constructed with increasing number of domains ($\mu = 5$).



(d) Scenario 4: APCs of paths with increasing number of gateways ($m = 20$).



(e) Scenario 5: APCs of paths with increasing number of average node degree ($m = 20$, $\mu = 5$).

Fig. 3: APC for different evaluation scenarios with varying network structural or synchronization-related parameters.

Erdös-Rényi (ER) model: An ER graph is generated by adding an edge between two nodes with a fixed probability p. The result is a purely random topology where all graphs with an equal number of links are equally likely to be selected. Vertex degree under ER model follows binomial distribution.

B. Evaluation Scenarios

We conduct our evaluations for the following scenarios, with different objectives: (1) Scenario 1, where we test the APC against different RC partitions that have the same total number of RCs on the domain-wise path; (2) Scenario 2, where we test the APC for different numbers of RCs on the domain-wise path; (3) Scenario 3, where we compare the APCs with different numbers of domains but the same number of RCs; (4) Scenario 4, where we test the influence of the number of gateways on the APC; (5) Scenario 5, where we evaluate the APC for networks with different connectivity levels.

C. Evaluation Settings

All evaluations are conducted on graphs with 300 nodes in each domain, where the link preference (weight) of intra-domain links are i.i.d. random variable uniformly distributed between 1 and 5. Except for Scenario 3 where the number of domains on the domain-wise path is varied from 20 to 30, there are 20 domains on the domain-wise path from the source to the destination domain in all other scenarios. Network topologies of all domains for evaluation Scenarios 1 - 4 are generated using statistics collected from the RocketFuel project, where we use AS "orange1010331" that has $10,670$ nodes whose node degrees are distributed between 2 and 49. The average number of 1-hop and 2-hop nodes from a random node in this AS are 3.16 and 13.31, respectively. All datasets used in our evaluations can be downloaded from the Stanford SNAP project website [34].The network topology for evaluation Scenario 5 is generated using the ER network model, because we need a continuous variation of the average node degree for

the purpose of this evaluation. For Scenarios 1, 3, and 5, the domain-wise path is partitioned into 5 RCs. Moreover, some evaluation scenarios require that the constructed path be partitioned into certain number of RCs. Since there are potentially many partitions that result in the same number of RCs, we randomly select 30 such partitions under the constraint of the required number of RCs (for Scenarios 1 - 5) and use results averaged over these partitions as evaluation results (for Scenarios 2 - 5). It should be noted that the plotted simulated APCs are the average results of multiple network graph realizations sharing the same given network settings, as the simulation results of a single network instance cannot indicate the characteristics of the network with the given set of settings. Specifically, for each evaluation scenario, 50 network graphs are realized, and with each realization, 50 source-destination pairs are randomly picked from the source and destination domains for path constructions.

D. Evaluation Results

1) Accuracy of the APC prediction: In all evaluation scenarios, the derived APC lower bounds capture the trend of APC changes against variations in different network structural or synchronization-related parameters, as detailed in the following.

- Fig. 3a confirms that the APC lower bound is not affected by how domain-wise paths are organized into RCs, as long as these paths share the same number of domains and the same number of RCs (Theorems 8, 10, and 12).
- Fig. 3b shows that the APC increases in μ for sparse ($\gamma = 5$) and medium ($\gamma = 40$) inter-domain connections, but stays relatively unchanged under dense inter-domain connections ($\gamma = 170$). This confirms our argument that controller synchronizations only exhibit limited contributions when the number of gateways is large (insights in Section V-C).
- Fig. 3c demonstrates that the APC increases linearly in m under all inter-domain connection ranges (Theorems 8, 10, and 12).
- Fig. 3d and Fig. 3e display the non-linear relationships between the APC and the number of gateways and the average node degree. In particular, Fig. 3d shows that the number of gateway nodes in domains contribute logarithmically (Theorems 8 and 10).

2) Contributions of different controller synchronization levels and network structural parameters on the APC: The central question we aim to answer in this paper is whether higher controller synchronization levels always deliver performance improvements. In addition, we also want to understand the influence of other parameters on APC. On top of our analytical results, these evaluation results further offer us answers to these questions.

- The slope of lines in Fig. 3b represents the rate of the APC increase versus RCs increase, which signifies a worsening controller synchronization level. We observe that the slopes decrease when the number of gateways increases. This suggests that the worsening controller synchronization levels affect networks with less number of gateways more, thus showing the importance of having more gateways in situations where controller synchronizations are limited (insights in Section V-A).
- Despite its role in reducing the APC, Fig. 3d shows the limit of having more gateways. This is demonstrated by the diminishing decrease in APC when the number of gateways continues to grow from an already significant amount (insights in Section V-C).
- Fig. 3e reveals that the improved network connectivity level, reflected by the average node degree of intra-domain topologies, also leads to the reduction of the APC. Intuitively, this is because higher network connectivity levels essentially offer more path construction options, from which more cost-saving paths are likely to emerge. However, again such benefit becomes constrained when more gateways are added (insights in Section V-B).
- Fig. 3c demonstrates that the influence on the APC by the number of domains on the domain-wise path is not strongly coupled with the number of gateways present in domains. This observation suggests that the efforts invested in improving the connectivity of the domain-wise topology, which brings down the average number of domains on a domain-wise path, is always beneficial for reducing the APC (Theorems 8, 10, and 12).

VII. Conclusions

We have studied the role of SDN network structural parameters and controller synchronization lev-

els in determining the quality of the constructed inter-domain paths from an analytical perspective. For this goal, a generic network model is proposed to capture key attributes in distributed SDN. Based on this model, we employed the average path cost as the performance metric, and derived its lower bounds with proven tightness guarantees for quantifying the routing performance for any given network synchronization status. Our results reveal the contributions of controller synchronizations and other parameters in improving the quality of the constructed paths for different network settings. Extensive simulations on both real and synthetic networks confirm the validity of our developed analytical results and their abilities in providing helpful guidance on synchronization protocol design.

REFERENCES

[1] N. McKeown. Software-defined networks and the maturing of the internet. [Online]. Available: https://tv.theiet.org/?videoid=5447

[2] D. Kreutz, F. M. Ramos, P. E. Verissimo, C. E. Rothenberg, S. Azodolmolky, and S. Uhlig, "Software-defined networking: A comprehensive survey," *Proceedings of the IEEE*, vol. 103, no. 1, pp. 14–76, 2015.

[3] B. A. A. Nunes, M. Mendonca, X.-N. Nguyen, K. Obraczka, and T. Turletti, "A survey of software-defined networking: Past, present, and future of programmable networks," *IEEE Communications Surveys & Tutorials*, vol. 16, no. 3, pp. 1617–1634, 2014.

[4] M. Casado, M. J. Freedman, J. Pettit, J. Luo, N. McKeown, and S. Shenker, "Ethane: Taking control of the enterprise," in *ACM SIGCOMM*, 2007.

[5] S. Jain, A. Kumar, S. Mandal, J. Ong, L. Poutievski, A. Singh, S. Venkata, J. Wanderer, J. Zhou, M. Zhu *et al.*, "B4: Experience with a globally-deployed software defined WAN," *SIGCOMM Computer Communication Review*, vol. 43, no. 4, pp. 3–14, 2013.

[6] S. H. Yeganeh, A. Tootoonchian, and Y. Ganjali, "On scalability of software-defined networking," *IEEE Communications Magazine*, vol. 51, no. 2, pp. 136–141, 2013.

[7] V. Kotronis, X. Dimitropoulos, and B. Ager, "Outsourcing the routing control logic: Better Internet routing based on SDN principles," in *ACM HotNets-XI*, 2012.

[8] V. Kotronis, R. Klöti, M. Rost, P. Georgopoulos, B. Ager, S. Schmid, and X. Dimitropoulos, "Stitching inter-domain paths over IXPs," in *ACM SOSR*, 2016.

[9] G. Petropoulos, F. Sardis, S. Spirou, and T. Mahmoodi, "Software-defined inter-networking: Enabling coordinated QoS control across the Internet," in *IEEE ICT*, 2016.

[10] Z. Chen, J. Bi, Y. Fu, Y. Wang, and A. Xu, "MLV: A multi-dimension routing information exchange mechanism for inter-domain SDN," in *IEEE ICNP*, 2015.

[11] P. Thai and J. C. de Oliveira, "Decoupling policy from routing with software defined interdomain management: Interdomain routing for SDN-based networks," in *IEEE ICCCN*, 2013.

[12] Y. Lai and W. S. Lai, "A graph-theoretic model of routing hierarchies," in *IEEE WAINA'*, 2009.

[13] B. Awerbuch and Y. Shavitt, "Topology aggregation for directed graphs," *IEEE/ACM Transactions on Networking*, vol. 9, no. 1, pp. 82–90, 2001.

[14] B. Awerbuch, Y. Du, and Y. Shavitt, "The effect of network hierarchy structure on performance of atm pnni hierarchical routing," *Computer Communications*, vol. 23, no. 10, pp. 980–986, 2000.

[15] R. Cherukuri, D. Dykeman, and M. Goguen, "PNNI draft specification," in *ATM Forum*, 1995, pp. 94–0471.

[16] I. Castineyra, N. Chiappa, and M. Steenstrup, "The Nimrod routing architecture," Tech. Rep., 1996.

[17] A. Tootoonchian and Y. Ganjali, "Hyperflow: A distributed control plane for OpenFlow," in *ACM INM/WREN*, 2010.

[18] K. Phemius, M. Bouet, and J. Leguay, "Disco: Distributed multi-domain SDN controllers," in *IEEE NOMS*, 2014.

[19] 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 *ACM HotSDN*, 2014.

[20] A. R. Curtis, J. C. Mogul, J. Tourrilhes, P. Yalagandula, P. Sharma, and S. Banerjee, "Devoflow: Scaling flow management for high-performance networks," in *ACM SIGCOMM*, 2011.

[21] S. Hassas Yeganeh and Y. Ganjali, "Kandoo: A framework for efficient and scalable offloading of control applications," in *ACM HotSDN*, 2012.

[22] M. Yu, J. Rexford, M. J. Freedman, and J. Wang, "Scalable flow-based networking with DIFANE," *ACM SIGCOMM Computer Communication Review*, vol. 40, no. 4, pp. 351–362, 2010.

[23] S. Vissicchio, O. Tilmans, L. Vanbever, and J. Rexford, "Central control over distributed routing," *ACM SIGCOMM Computer Communication Review*, vol. 45, no. 4, pp. 43–56, 2015.

[24] D. J. Watts and S. H. Strogatz, "Collective dynamics of 'small-world' networks," *Nature*, vol. 393, no. 6684, pp. 440–442, 1998.

[25] R. Milo, S. Shen-Orr, S. Itzkovitz, N. Kashtan, D. Chklovskii, and U. Alon, "Network motifs: Simple building blocks of complex networks," *Science*, vol. 298, no. 5594, pp. 824–827, 2002.

[26] A. L. Barabási, "Scale-free networks: A decade and beyond," *Science*, vol. 325, no. 5939, pp. 412–413, 2009.

[27] P. Erdös and A. Rényi, "On the evolution of random graphs," *Publ. Math. Inst. Hung. Acad. Sci*, vol. 5, no. 1, pp. 17–60, 1960.

[28] A.-L. Barabási and R. Albert, "Emergence of scaling in random networks," *Science*, vol. 286, no. 5439, pp. 509–512, 1999.

[29] M. E. Newman, S. H. Strogatz, and D. J. Watts, "Random graphs with arbitrary degree distributions and their applications," *Physical Review E*, vol. 64, no. 2, p. 026118, 2001.

[30] B. Jiang, P. Nain, D. Towsley, and S. Guha, "On a class of stochastic multilayer networks," in *ACM SIGMETRICS*, 2018.

[31] Z. Zhang, L. Ma, K. K. Leung, F. Le, S. Kompella, and L. Tassiulas, "How advantageous is it? An analytical study of controller-assisted path construction in distributed SDN?" *submitted to IEEE/ACM Transactions on Networking (under review)*. [Online]. Available: https://goo.gl/WEvPm6

[32] The Opendaylight Controller. [Online]. Available: https://www.opendaylight.org

[33] "Rocketfuel: An ISP topology mapping engine," University of Washington, 2002. [Online]. Available: http://www.cs.washington.edu/research/networking/rocketfuel/interactive/

[34] "Stanford network analysis platform (SNAP)," Standford University, 2017. [Online]. Available: https://snap.stanford.edu/snap/