Routing Performance in Distributed SDN under Synchronization Constraint

Ziyao Zhang
Imperial College London
London, United Kingdom
ziyao.zhang15@imperial.ac.uk

Qiaofeng Qin
Yale University
New Haven, CT, United States
qiaofeng.qin@yale.edu

Konstantinos Poularakis
Yale University
New Haven, CT, United States
konstantinos.poularakis@yale.edu

Liang Ma
IBM T. J. Watson Research Center
Yorktown Heights, NY, United States
maliang@us.ibm.com

Franck Le
IBM T. J. Watson Research Center
Yorktown Heights, NY, United States
fle@us.ibm.com

Leandros Tassiulas
Yale University
New Haven, CT, United States
leandros.tassiulas@yale.edu

Sastry Kompella
U.S. Naval Research Lab
Washington, DC, United States
sastry.kompella@nrl.navy.mil

Konstantinos Poularakis
Yale University
New Haven, CT, United States

ABSTRACT

Distributed software-defined networking (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 this work, we study such network architecture from both experimental and analytical perspectives. Specifically, we implement a wireless SDN multi-controller system, where the controllers communicate to each other and to data plane-devices through WiFi multi-hop links. We measure the corresponding traffic delay and overheads. The results reveal the sensitivity of delay to the location of controllers and the magnitude of inter-controller and controller-node overheads. Such experiments highlight the feasibility of distributed SDN and the overheads of their operations including the overheads due to synchronizations. Therefore, it is challenging to achieve full status synchronizations among controllers in a real system. In this regard, we analyze and quantify the performance enhancement of distributed SDN architectures, which is influenced by intra-/inter-domain synchronization levels and network structural properties. Based on a generic network model, we establish analytical methods for performance estimation under four canonical inter-domain synchronization scenarios. Specifically, we first derive an asymptotic expression to quantify how dominating structural and synchronization-related parameters affect the performance metric. We then provide performance analytics for an important family of networks, where all links are of equal preference for path constructions. Finally, we establish fine-grained performance metrics expressions for networks with dynamically adjusted link preferences. Our theoretical results reveal how network performance is related to synchronization levels and intra-/inter-domain connections, the accuracy of which is confirmed by simulations based on both real and synthetic networks. To the best of our knowledge, this is the first work quantifying the performance of distributed SDN analytically, which provides fundamental guidance for future SDN protocol designs and performance estimation.

1 INTRODUCTION

Software-Defined Networking (SDN) [1] [2], an emerging networking architecture, significantly improves the network performance 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, the number of flow requests and operational constraints are likely to increase drastically. Such 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 [3].

In this regard, distributed SDN is proposed [4–9] to balance the centralized and distributed control. Specifically, a distributed SDN network is composed 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; such inter-connected domains then form the distributed SDN architecture. In the distributed SDN architecture, controllers...
are expected to exchange information via proactive probing or passive listening. Such additional status information at each controller, called the \textit{synchronized information}, can assist in enhancing decision making for inter-domain tasks. In distributed SDN, network performance relies heavily on the inter-controller synchronization level. Since complete synchronization among controllers, i.e., each controller knows the network status in all other domains, will incur high synchronization costs especially in large networks, practical distributed SDN networks can only afford partial inter-domain synchronization.

For partial synchronization, most existing works focus on promoting the inter-domain synchronization so that the final decision making approaches optimality. For instance, information sharing algorithms are proposed in \cite{6,7} for negotiating common traffic policies among various domains. Similarly, frameworks are designed in \cite{8,9}, aiming to facilitate inter-domain routing selection via fine-grained network status exchanges. However, one fundamental question regarding the distributed SDN architecture has generally been ignored: \textit{How does the network performance in distributed SDN relate to network synchronization levels and structural properties?} It is possible that under certain network conditions, e.g., the number of gateways and their connections to external domains, the benefit of increasing the synchronization level is only marginal. Without such a fundamental understanding, it is impossible to justify why a complicated mechanism for information sharing or flow construction is necessary in distributed SDN. We, therefore, investigate this unsolved yet critical problem in the distributed SDN paradigm, aiming at quantifying its performance under any given network conditions.

To highlight the applicability of distributed SDN solutions in different types of networks, we implement a wireless system co-managed by multiple SDN controllers communicating to each other and to data plane devices through WiFi links. This testbed is built from off-the-shelf edge network devices (Android smartphones) that are programmed to run Open vSwitch data plane \cite{10} and ONOS controller \cite{11} implementations. We perform experiments to measure the delay of managing a data plane device by a controller device and show its sensitivity on the location of controllers. In order to analyze large-scale systems consisting of hundreds of devices (which goes beyond our testbed), we provide an emulation study based on the same controller implementation. We find that inter-controller and controller-node traffic overheads can be at the same order of magnitude (up to a few Mbps each).

To this end, we first propose a network topological model to capture the intra-/inter-domain connections in distributed SDN. Based on this network topological model, we further associate a preference level (see Section 3.2) to each link for path constructions, which, in practice, is adjusted by SDN controllers based on the collected network information (e.g., traffic and congestion status). Such network model is generic in that it only requires node degree/link preference distributions and the number of gateways in each domain as the input parameters, i.e., they are independent of any specific graph models. Based on this network model, we then derive analytical expressions of the network performance focusing on the average cost of the constructed paths with respect to (w.r.t.) random flow requests. Such performance metric is investigated under four canonical synchronization levels, ranging from the minimum to the maximum level of synchronization. If a given synchronization scenario cannot be described by any of these four cases, then its performance can be bounded by our analytical results corresponding to the two extreme cases (i.e., maximum/minimum synchronization). Specifically, we first establish an asymptotic expression to highlight the relationship between the performance metric and dominant parameters. Then, we conduct detailed analysis on two families of networks - network with uniform and network with non-uniform link preference. The main difference between them is the simplicity of link preference, where in the former case controllers do not specify any preference for links due to the lack of network status information, thus all links have equal link preference; in the later case, however, controllers assign preference to links to achieve control objectives based on up-to-date network status information collected. Analytical results reveal that the performance metric is a logarithmic function of the network structural parameters even under the minimum synchronization level. Moreover, the performance gain declines with the increasing synchronization level and the number of gateways. To validate the accuracy of the derived analytical expressions, they are compared against evaluation results using both real and synthetic networks.

1.1 Related Work

1.1.1 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 notes, both from a theoretical and experimental approaches. For example, \cite{12} shows that hierarchical routing where the topologies of the clusters are hidden can lead to suboptimal routing, and forwarding loops. \cite{13} proposes to aggregate topologies with theoretical bounds. \cite{14} analyzes the effectiveness of hierarchical routing (e.g., ATM PNNI [15], Nimrod[16]), and [17] studies the performance of several different aggregation schemes in terms of network throughput, and network control load. However, most of these early works are either driven by simulations or looked at different aspects of the problem analytically. Thus, they have not tried to study the impact of synchronization and other network structural properties on the network performance from an analytical approach yet. In contrast, we conduct a rigorous mathematical analysis in order to understand these aspects in distributed SDN.

1.1.2 Distributed SDN. The distributed SDN architecture, which integrates the advantages in early hierarchical networks, has stimulated many research efforts in this area. Specifically, \cite{18} investigates the problem of SDN upgrade in ISP (Internet Service Provider) networks under the constraint of migration costs. In addition, protocols and systems, such as HyperFlow \cite{19} and ONOS \cite{20}, are proposed to realize logically centralized but physically distributed SDN architecture. Devollow \cite{21} and Kandoo \cite{22} are designed to reduce the overheads introduced by the interaction between the control and data planes. However, again, most of these works are experiment-based. By contrast, our goal is to investigate distributed SDN from the perspective of fundamental analytics.

1.1.3 Performance Analytics. Since all theoretical results in this paper are obtained based on a graph model, our work is also related to the area of graphical analysis of complex networks. Most works in these areas are dedicated to the study of specific graph properties,
e.g., small-world effect [23], network motif [24], scale-free [25], etc. Thus, they are substantially different from our work. Our work is most related to [26] as they also consider a layered-network model for the study of communication networks. However, the authors are mainly concerned with modeling the co-existing connectivity.

1.2 Summary of Contributions
Our main contributions are five-fold.

1) We implement a proof-of-concept prototype of a wireless edge system with multiple SDN controllers and measure the corresponding traffic delay and overheads;
2) We propose a generic two-layer network model to capture intra-/inter-domain connections and their properties;
3) On top of the network model in 1), we use the average path cost (APC) of the constructed paths as the performance metric to develop the asymptotic expression of the APC under any given synchronization levels;
4) For networks with uniform link preference, we develop the analytical expression of the APC lower bound under each synchronization scenario;
5) For networks with non-uniform link preference, we integrate dynamic link preference levels and develop the fine-grained analytical expression of the APC under each synchronization scenario;
6) We evaluate our analytical results by extensive simulations using both real and synthetic networks, which confirm their high accuracy and ability in revealing insights into the actual performance under various network conditions.

In this paper, we do not intend to design improved inter-domain routing mechanisms, and thus only the basic and representative routing strategy (see Section 4 for details) is employed for theoretical analysis. To the best of our knowledge, this is the first work that studies distributed SDN from the analytical perspective. The significance of these results is that they lay a strong theoretical foundation for the research community in distributed SDN.

The rest of the paper is organized as follows. Section 2 presents the testbed implementation and experimentation results. Section 3 formulates the problem. Section 4 describes the path construction mechanism used for our analysis. Section 5 establishes asymptotic APC expression which shows the interplay of dominant parameters. Then under different synchronization scenarios, Sections 6–8 discuss the APC and its performance bound in networks with uniform link preference, based on which Section 9 provides a universal expression of the APC lower bound, which is applicable to any cases. Further, sections 10–13 derive the fine-grained expressions of APC in networks with non-uniform link preference under four different synchronization scenarios. Evaluations of the derived analytical expressions are conducted in Section 14. Finally, Section 15 concludes the paper.

2 EXPERIMENTATION & EMULATION ANALYSIS
In this section, we present experimentation and emulation results using commercial SDN controller and data plane implementations. The results provide insights about the delay and overheads of multi-controller edge systems.

2.1 Experimentation Results for Management Delay
Testbed Set-up. In this subsection, we set-up a testbed of a multi-controller edge system using off-the-shelf network devices. Specifically, we deploy four Nexus 4 Android smartphones to form a wireless network as it is depicted in Figure 1(a). The first smartphone works as an access point (hotspot) to provide the remaining three smartphones with WiFi connections. This represents a common network setting, where a node can either establish multihop connections to backbone networks, or exchange data with its peer in a D2D fashion. Besides, smartphones are representative devices widely used in edge networks. Due to their constraints of calculating and storage, our testbed shows a challenging scenario that is worthy to investigate.

We take several steps to make the network SDN-enabled. In each smartphone, we create a chroot environment to install the Ubuntu system running with its original Android system at the same time. By this, we are able to install popular SDN-related software in the smartphone. First, we make all devices working as data plane nodes by installing Open vSwitch [10]. This creates a virtual switch that supports SDN in each smartphone. Second, we deploy SDN controllers in Node 2 and Node 4. Though constrained in resources, the smartphone is still capable enough to run a controller instance, such as ONOS. ONOS is designed particularly for scalability and permits multiple controllers working together in the form of a controller cluster. Then, we establish a connection between the two
controllers and assign each smartphone to its nearest controller with respect to the hop count length distance metric.

**Measurement Methodology.** We mainly take measurements on the network delay between the controller and data plane nodes. The devices are placed within an empty lab room, and a distance over each wireless link is 2 meters. One frequent and important interaction between a controller and a data plane node is the request and reply of flow statistics. Therefore, we measure the delay at controller nodes, by analyzing the interval between sending such an OpenFlow request message and receiving its corresponding reply. We keep capturing messages since the cluster reaches the steady state and collect 250 measurements. Figure 1(b) shows the cumulative distribution function of the delays we measured. The average value is tens of milliseconds. From the CDF plot, we notice that the variance is large, corresponding to the relatively unstable wireless links. It is common for the delay to go even beyond 100 milliseconds.

We also notice that the location of controllers is important, because the delay relies highly on the distance between the data plane node and its controller. For example, Node 2 and Node 4 contain controllers locally, while Node 1 and Node 3 have one-hop and two-hop connections to the controller, respectively. As a result, drawn separately in Figure 1(b), local connection shows almost zero delays while the one-hop and two-hop connections show notable delays. To further demonstrate this, we move ONOS controllers from nodes 2 and 4 to nodes 1 and 3. If we still assign each data plane node to its nearest controller, we can find that the average delay is 25% lower, as it is depicted in Figure 1(c).

**Main Takeaways.** Modern edge network devices (smartphones) can act as controllers to manage other devices. The management delay can significantly change for different controller locations (up to 25% difference in our testbed).

### 2.2 Emulation Results for Control Overheads

In the previous subsection, we focused on the delay required to manage the edge nodes. In this subsection, we will analyze another important factor; the overheads of SDN control.

**Types of Control Overheads.** When considering the SDN control overheads, people usually refer to the traffic between the controllers and data plane nodes. Namely, controllers and nodes exchange various messages through a specific protocol, which is OpenFlow in most cases, including periodic heartbeat messages and statistic requests/replies. It is intuitive that such overheads grow when the network scales up. Moreover, there are also overheads related to the routing of packet flows. When a new flow is generated and a node receives packets that cannot be matched in its forwarding table, it will report to the controller through PacketIn messages. On the contrary, the controller may install new forwarding rules to nodes using FlowMod messages. Therefore, the overhead is also influenced by the number of flows. The more frequently new flows emerge, the larger overhead is needed to install forwarding rules.

If a cluster of controllers, rather than only one controller, is deployed, a second kind of overheads will be generated, which is the state synchronization between different controllers. Among controllers, one or more consensus algorithms must be run to share the network topology and flow tables in each controller’s domain, in order to reach a consistency. It can be expected that the more information they should share, and the more members they should share with, the larger such overheads are.

**Emulation Set-up.** Since both the types of control overheads depend heavily on the scale of the network, we need to deploy many more nodes than what we have in our testbed if we wish to analyze them. A more accessible way to take large-scale measurements is by running emulations on a virtual network generated by Mininet. This method allows us to test networks with hundreds of nodes and several controllers using a common CPU machine. To be consistent with the previous subsection, we pick again ONOS as our controller instance and run it along with Mininet. Specifically, we create a virtual network with ring topology and evenly assign nodes to the placed controllers, as shown in Figure 2(a). All controllers run a simple application; reactive forwarding.

**Measurement Methodology.** In order to show the impact of the scale of the network, we take measurements on both types of control traffic with different number of nodes in the virtual network. Figure 2(b) verifies the intuition about both types of traffic’s growth when the network scales up. What is more, we also consider other factors that have impact. For controller–node traffic, we create large amount of one-hop iPerf flows randomly, with a fixed rate (0.1 flows per second for each node). For inter-controller traffic, we deploy clusters of different sizes, i.e. number of controllers in the cluster. According to Figure 2(b), in all of these situations, the two types of overheads are at the same order of magnitude (up to a
Figure 3: Two-layer network model: Top-layer abstracts the domain-wise topology; bottom-layer determines all physical connections in the network.

e few Mbps each). We also note that the inter-controller traffic is not affected by the number of flows and this is because of the reactive forwarding application we chose.

Next, we elaborate on the inter-controller traffic overhead. In ONOS, each pair of controllers exchange periodically heartbeat messages of fixed number and sizes. Besides, there are two more consensus algorithms running. The first one is anti-entropy gossip protocol. With an interval, a controller sends the information (network topology, flow tables, etc.) within its domain (nodes assigned to it) to a random peer controller. Therefore, this overhead is proportional to the domain size (also called load) of this controller. The second algorithm is RAFT. ONOS uses it to synchronize controller-node assignments within the cluster, generating an overhead also related to the controller load. Specifically, RAFT has a leader-follower mechanism. The controllers periodically hold elections, which can be seen as an additional constant overhead. We make separated measurements on these different components. Results in Figure 2(c) verify and quantify above analysis that with the increasing number of nodes, the heartbeat overhead is constant. The flow table synchronization overhead (majority of anti-entropy protocol) is linear. The RAFT overhead is a combination of these two types.

Main Takeaways. The two types of overheads (inter-controller and controller-node traffic) are at the same order of magnitude in representative scenarios (up to few Mbps each). The traffic exchanged between a pair of controllers can be modeled by two terms; (i) a constant term (heartbeat and part of RAFT messages) and (ii) a term that increases linearly to the load of controllers (flow table synchronization and part of RAFT messages).

3 PROBLEM FORMULATION

3.1 Network Model

We formulate the distributed SDN network as an undirected graph according to a two-layer network model (Fig. 3), where the top-layer abstracts the inter-domain connections, and under such constraints, the bottom-layer characterizes physical connections among all network elements. Specifically, the top-layer is a graph consisting of $m$ vertices, where each vertex represents a domain in the distributed SDN. These $m$ vertices are connected via undirected links according to a given inter-domain degree distribution, which refers to the distribution of the number of neighbouring domains of an arbitrary domain. The top-layer graph, denoted by $G_d = (V_d, E_d)$ ($|V_d|/|E_d|$; set of vertices/edges in $G_d$; $|V_d| = m$), is called domain-wise topology in the sequel. The existence of an edge in $E_d$ connecting two vertices $v_1, v_2 \in V_d$ in the domain-wise topology implies that the two network domains corresponding to $v_1$ and $v_2$ are connected. Based on this domain-wise topology, we next construct the physical network in the bottom-layer. In particular, each of the $m$ domains in $G_d$ 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 neighboring nodes of an arbitrary node within the same domain. 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 $A_i$ and $A_j$, we (i) randomly select two nodes $v_1, v_2$ from $A_i$ and $A_j$ and connect these two nodes if link $v_1v_2$ does not exist, and (ii) repeat such link construction process between $A_i$ and $A_j$ $\beta$ times. By this link construction process, the bottom-layer network topology $G = (V, E)$ is therefore formed ($V/E$: set of nodes/links in $G$; $|V| = mn$); see Fig. 3 for illustrations. In each domain, nodes having connections to other domains are called gateways. 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 $\beta$ represents the maximum number of links between any two domains. Hence, if two domains, each with $n$ nodes, are connected in the domain-wise topology, then the expected number of links connecting these two domains is $n^2(1 - (1 - \frac{1}{n})^\beta)$. Without loss of generality, we assume that all inter/intra-domain topologies are connected graphs.

Discussion: Our two-layer network model is generic in that the input node degrees and link preference levels can be of any distributions; such distributions can be empirical or extracted from real networks of interest. Moreover, 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.

3.2 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 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 the link is 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 status. Therefore, the goal for constructing an optimized end-to-end inter-domain path under a given network status 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, in this paper, we conduct our analysis in two types of networks which we call network with uniform link preference (Type-1 Network) and network with non-uniform link preference (Type-2 Network). For Type-1 Networks, all link preference are static and equal; therefore, without loss of generality, all link weights in Type-1 Networks are set to 1.
By contrast, in Type-2 Networks, random variables (r.v.) are used to capture the dynamicity of link preference. Specifically, for Type-2 Networks, we assume that intra-domain link preference 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. Without loss of generality, we assume that the link preference levels for all inter-domain links are 1; all our theoretical results can be easily extended to other inter-domain link preference values.

3.3 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.

3.3.1 Data Plane. We exploit graph $G$ generated by the two-layer network model in Section 3.1 to represent the data plane of the distributed SDN. Specifically, a node/link exists in $G$ if and only if it can be used for data transmission in the network.

3.3.2 Control Plane. Under the two-layer network model, each domain contains one logical SDN controller that carries out control operations and facilitates information sharing. SDN controllers together with all inter/intra-domain controlling channels form the control plane.

3.4 Synchronization Among SDN Controllers

Since link preference (weight) captures the controller’s view of current domain, i.e., network status information; the process of controller synchronization involves the exchange of such information, which we formally define below.

Definition 1. Domain $A_i$ is synchronized with domain $A_j$ if and only if the SDN controller in $A_i$ knows the minimum path cost between any two nodes in $A_j$.

By Definition 1, clearly there exist a significant number of synchronization cases. Moreover, in real networks, it is usually the case that synchronization difficulty is high when two SDN controllers are far apart. In this paper, we therefore categorize the inter-domain synchronization into the following synchronization cases, sorted by their corresponding synchronization difficulties.

a) Minimum Synchronization (MS): Under MS, no domains synchronize with any other domains. As a result, each controller only knows its own intra-domain topology and the domain-wise topology, but the controller does not assign link preference levels (all links have an equal link preference of 1) due to the lack of network status information. This scenario captures IGP routing protocols that do not take into account any link weights but select routes purely based on the hop count (e.g., Routing Information Protocol (RIPv2)). Note that MS corresponds to the minimum network knowledge that is always available, including scenarios in b)-d);

b) Self-domain Synchronization (SS): In addition to the information under MS, each controller under SS knows something more except for the intra-domain link preference levels in its own domain. With this additional information, one controller can find the optimal intra-domain path for any intra-domain flow requests;

c) Partial Synchronization (PS): PS refers to any synchronization levels between SS and the following complete synchronization (CS);

d) Complete Synchronization (CS): Under CS, every pair of domains $A_i$ and $A_j$ synchronize with each other. As such, there is effectively one logically centralized controller, which can make globally optimal decisions. Among all these synchronization scenarios, CS experiences the highest synchronization difficulty.

3.5 Problem Statement and Objective

Given the distributed SDN network model in Section 3.1, our goal is to study the performance of the paths constructed by a basic and representative path construction mechanism (see Section 4 for details) under various synchronization scenarios. In real networks, the performance of routing can be measured by many metrics, such as delay and congestion level, depending on the goal of network management. In order to make our analytical work sufficiently generalized to capture the performance metric that is important to most network management tasks, 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. Formally, our research objective is:

Objective: Suppose (i) each network realization following the two-layer network model exists with the same probability, and (ii) the source-destination node pair belonging to two different domains in a given network realization also exists with the same probability. Our goal is to derive the mathematical expression of APC under each of the four synchronization scenarios, i.e., MS, SS, PS, and CS, in both Type-1 and Type-2 Networks (networks with uniform/non-uniform link preference).

In this paper, we are only interested in studying the cross-domain routing, since controllers can easily find the optimal intra-domain paths without relying on inter-controller synchronizations.

Remark: Note that our two-layer network model is a random graph model, i.e., there exist multiple network realizations satisfying the same set of input parameters. Therefore, APC is an expected value over not only random source/destination node pairs but also random network realizations. All our theoretical results on APC are based on the given network parameters (e.g., degree and weight distributions) rather than a specific network realization.

4 PATH CONSTRUCTION MECHANISM

We describe a path construction mechanism for 4 synchronization scenarios introduced in Section 3.4. The intuition behind the path construction mechanism is that given a particular synchronization level, the synchronized controllers attempt to use the synchronized information and make joint decision to minimize the overall accumulated cost of the constructed path in their domains. Then the selected path segments in all participating domains between the source/destination nodes concatenate into a cross-domain, end-to-end path. Before presenting the path construction mechanism, we first introduce several definitions as follows.

Definition 2. a) In the domain-wise topology $G_d$, the vertex corresponding to domain $A$ in $G$ is denoted by $\delta(A)$. Given a pair
routing cluster 1 (RC1) routing cluster 2 (RC2) routing cluster 3 (RC3)

Figure 4: Path construction w.r.t. $v_1$ and $v_2$, whose shortest domain-wise path traverses $A_1, A_2, A_3, A_4, A_5$, and $A_6$, of source and destination nodes $v_1$ and $v_2$ with $\ell = 5$, $\tau = 2$, constructed path.

Where $\phi \in G$ and $\tau \in G_d$ and $\mu = \lfloor q/\tau \rfloor$, the domain-wise path w.r.t. $v_1$ and $v_2$ is a path in $G_d$ starting at vertex $\delta(A_1)$ and terminating at vertex $\delta(A_2)$.

b) The domain-wise distance w.r.t. domains $A_1$ and $A_2$ is the length of the shortest path from the vertex corresponding to $A_1$ to the vertex corresponding to $A_2$ in the domain-wise topology $G_d$.

Based on Definition 2, we then define synchronization radius to capture different levels of synchronizations as follows.

Definition 3. The synchronization radius $\tau$ ($\tau \geq 1$) is an integer such that (i) any two domains with their domain-wise distance less than or equal to $\tau$ are synchronized, and (ii) no two domains with their domain-wise distance greater than $\tau - 1$ are synchronized.

According to the definition of synchronization radius, $\tau = 1$ for MS or SS, depending on link preference status; $\tau = \phi$ for CS, where $\phi$ is the maximum domain-wise distance between any two domains in the network. Any value of $\tau$ between 1 and $\phi$ falls in the category of PS. As such, we use a given $\tau$ to capture the PS scenario. Under a specified synchronization level, the synchronized controllers leverage the shared information to jointly make routing decisions on any domain-wise paths between source/destination nodes. Formally, we have the following definition.

Definition 4. The group of domain(s) on the domain-wise path where routing decisions are made jointly by their synchronized controller(s) is referred to as a routing cluster (RC). Specifically, given a domain-wise path between the source and destination nodes, for all domains on this domain-wise path:

a) Under MS or SS ($\tau = 1$), each domain constitutes an RC;

b) Under PS ($1 < \tau < \phi$), starting from the source domain, every $\tau$ domains form an RC such that each domain belongs to one and only one RC, and only the RC including the destination domain may have less than $\tau$ domains;

c) Under CS ($\tau = \phi$), all domains on the domain-wise path form an RC, where $\phi$ is the maximum domain-wise distance between any two domains in the network.

Based on Definition 4, let $q$ and $\mu$ denote the number of domains and the number of RCs on the domain-wise path, respectively. For PS with synchronization radius $\tau$, the RC that includes the destination domain has $q - \tau(\mu - 1)$ domains, whereas all other RCs have $\tau$ domains. Now, we are ready to introduce the path construction mechanism between two arbitrary nodes $v_1$ and $v_2$ in the following steps:

Step 1) Select the shortest domain-wise path w.r.t. $v_1$ and $v_2$, which consists of $q$ domains, with ties (if any) broken arbitrarily. That is, no domain-wise path w.r.t. $v_1$ and $v_2$ traverses less than $q$ domains.

Step 2) Based on the given synchronization status of all involved domains on the above domain-wise path, partition these domains into $\mu$ RCs ($\mu = q$ for MS and SS, $\mu = 1$ for CS, and $\mu = \lfloor q/\tau \rfloor$ for PS).

\[ \text{Step 3) For each RC}_i (\text{RCs are sequentially labeled from the source to the destination, } i = 1, 2, \ldots, \mu), \text{ a path segment starting from the entering node (which is } v_1 \text{ if } i = 1, \text{ or is specified by } \text{RC}_{i-1}) \text{ and terminating at one of the exiting nodes (which are gateways connecting to } \text{RC}_{i+1}, \text{ or node } v_2 \text{ if } i = \mu) \text{ with the minimum cost is constructed. Such path segment is denoted by } P_i \text{ in } \text{RC}_i. \text{ Also let } e_{i,i+1} \text{ be the edge leading from } P_i \text{ in } \text{RC}_i \text{ to connect to the entering node in } \text{RC}_{i+1} \text{ if } i \leq \mu - 1; \]

Step 4) The final $v_1$-to-$v_2$ path $P$ is

\[ P = P_1 + e_{1,2} + P_2 + e_{2,3} + \ldots + P_{\mu-1} + e_{\mu-1,\mu} + P_{\mu}. \]

Discussion: Step 1) is similar to the BGP protocol used for inter-domain routing in the Internet. We further justify the selection of the shortest domain-wise path in Theorem 12 and Corollary 13. The path construction mechanism described above rely on routing clusters as the basic routing unit, it is therefore referred to as routing cluster-based path construction (RCPC) in the sequel. Fig. 4 shows a PS example with $q = 5$ and $\tau = 2$ under RCPC. After the selection of domain-wise path which consists of domains $A_1 - A_5$, the domains are partitioned into 3 RCs according to Step 2), as shown in the figure. Then, by Step 3), routing decision is made jointly by controllers in each RC to minimize the corresponding path cost. For example, assume that all link preference are 1 in Fig. 4, the controllers of $A_1$ and $A_2$ jointly choose node $a$ as the exit point and thus construct a path segment between $v_1$ and $a$ in $\text{RC}_1$.

Note that the intention in this paper is not to design a new routing mechanism; on the contrary, the goal is to use a basic routing mechanism, RCPC, to understand the network performance in distributed SDC. For improved routing mechanisms, our RCPC-based analytical results serve as performance bound.

5 ASYMPTOTIC APC UNDER DIFFERENT SYNCHRONIZATION LEVELS

Before the discussion of fine-grained analytical results on APC, we first present the asymptotic analysis of APC (called asymptotic APC) under various synchronization scenarios in this section. The basic idea here is that we highlight, in the form of directly observable expressions, the interactions among different parameters in determining the overall APC, before their relationships become less visible in more accurate but more complicated fine-grained APC expressions.

Let $m$ and $n$ be the number of domains and the number of nodes in each domain in the network, and $y$ the average number of gateways connecting two neighbouring domains ($y = n(1 - (1 - 1/n)^\ell)$). Next, within a domain $A$, let $z_i$ denote the average number of vertices that are $i$-hop away from a random vertex within $A$. Similarly, in the top-layer $G_d$ of our two-layer model, let $z'_i$ denote the average number of vertices (here each vertex represents a domain) that are $i$-hop away from a random vertex in $G_d$. In addition, let $z''_i$ be the average number of vertices which are $i$-hop away from a random vertex in a special graph, called Randomized Degree-Preserving Network (RDPN); see Definition 9 for details. Main notations and abbreviations used in this paper are summarized in Table 1. Under all above definitions and path construction mechanisms, we present the asymptotic APC in the following theorem. All theorems, corollaries, and lemmas in the rest of this paper are proved in the Appendix.

Theorem 5. Given the synchronization radius $\tau$, the asymptotic APC (denoted by $L$) in the two-layer network model is
In this section, we study the APC under MS in Type-1 Networks (all links are of equal preference, i.e., link preference levels are 1 for all links) based on the path constructions mechanism RCPC introduced in Section 4. The basic idea is that we first compute the average domain-wise distance w.r.t. two arbitrary source/destination nodes. Since the path construction in each domain under MS is independent, we then consider a domain-wise path with the above average length, calculate the average number of hops in each traversed domain, and add them together to get the final estimation of APC for MS. To this end, we first present the results in the existing work [27] to assist our mathematical analysis.

**Proposition 6.** [27] In an undirected connected graph $\mathcal{H}$ with $n_0$ vertices and the vertex degree satisfying a given distribution, let $x_i$ be the average number of vertices that are $i$-hop away from a random vertex in $\mathcal{H}$. Suppose all edge weights are 1, and $x_2 \gg x_1$. Then

\begin{align}
& \text{a)} \quad x_i = (x_2/x_1)^{i-1}x_1; \\
& \text{b)} \quad \text{APC in } \mathcal{H} \text{ is} \\
& \quad \log(n_0/x_1) + 1. \tag{4}
\end{align}

In our two-layer model, the top-layer graph $\mathcal{G}_d$ (domain-wise topology with $m$ vertices) itself is a random graph following a given domain-wise degree distribution. Therefore, similar to [27], let $z'_i$ denote the average number of vertices that are $i$-hop away from a random vertex in $\mathcal{G}_d$. For two arbitrary nodes $v_1$ and $v_2$ with $v_1 \in A_1$, $v_2 \in A_q$, and $A_1 \neq A_q$, let $\Delta$ denote the average distance of the shortest domain-wise path from domain $A_1$ to domain $A_q$. Assuming $z'_2 \gg z'_1$, then according to (4), we have

\begin{align}
\Delta & = \log(z'_2/z'_1) + 1. \tag{5}
\end{align}

With (5), we know that the average number of domains for MS under RCPC is $\Delta + 1$. If we further know the average cost of $\mathcal{P}_i$ associated with the traversed domain $A_i$, then we can estimate the average cost of $\mathcal{P}$. To this end, let $|\mathcal{P}|$ denote the number of hops on path $\mathcal{P}$. Then, $|\mathcal{P}| = |\mathcal{P}_1| + |\mathcal{P}_2| + \ldots + |\mathcal{P}_{\Delta+1}| + \Delta$ according to (1), where $|\mathcal{P}_i|$ is a r.v. The expectation of $|\mathcal{P}|$ is:

$$
\mathbb{E}[|\mathcal{P}|] = \mathbb{E}[|\mathcal{P}_1| + |\mathcal{P}_2| + \ldots + |\mathcal{P}_{\Delta+1}| + \Delta] = \mathbb{E}[|\mathcal{P}_1|] + \mathbb{E}[|\mathcal{P}_2|] + \ldots + \mathbb{E}[|\mathcal{P}_{\Delta+1}|] + \Delta. \tag{6}
$$

According to the path construction procedure for MS, $\mathbb{E}[|\mathcal{P}_1|] = \mathbb{E}[|\mathcal{P}_2|] = \ldots = \mathbb{E}[|\mathcal{P}_q|]$ for two reasons. First, all domains have the same statistical properties. Second, in each domain $A_i (i \leq \Delta)$, the routing mechanism selects a gateway (from a set of gateway options) that is closest to the ingress node. By contrast, in domain $A_{\Delta+1}$, the routing mechanism only selects the minimum-cost path from the ingress node to the destination node $v_2$. Thus, (6) is simplified as

$$
\mathbb{E}[|\mathcal{P}|] = \Delta \cdot \mathbb{E}[|\mathcal{P}_1|] + \mathbb{E}[|\mathcal{P}_{\Delta+1}|] + \Delta. \tag{7}
$$
In a domain $\mathcal{A}$ with $n$ nodes, let $z_i$ denote the average number of intra-domain nodes that are $i$-hop away from an arbitrary node $v$ ($v \in \mathcal{A}$). Then again by (4), we have

$$\mathbb{E}[|\mathcal{P}_{A+1}|] = \frac{\log(n/z_i)}{\log(z_2/z_1)} + 1, \quad (8)$$

assuming $z_2 \gg z_1$. Hence, to compute $\mathbb{E}[|\mathcal{P}_1|]$ in (7), it suffices to consider only $\mathbb{E}[|\mathcal{P}_1|]$ associated with domain $\mathcal{A}_1$.

In $\mathcal{A}_1$, on average, there are $n = n(1 - (1 - 1/n)^\beta)$ gateways connecting to $\mathcal{A}_2$. Suppose $\mathcal{A}_1$ contains exactly $\gamma$ gateways, denoted by set $S$. Then regarding path $\mathcal{P}_1$ from the starting point $v_1 \in \mathcal{A}_1$ to set $S$, there are two cases. First, $v_1 \in S$, then $\mathcal{P}_1$ is a degenerate path containing only one node $v_1$, i.e., $|\mathcal{P}_1| = 0$. Second, $v_1 \not\in S$, which complicates the computation of $|\mathcal{P}_1|$. For the second case, let $l := \mathbb{E}[|\mathcal{P}_1|] / |v_1 \not\in S|$, i.e., the expectation of $|\mathcal{P}_1|$ conditioned on $v_1 \not\in S$. Regarding the gateway set $S$, there are up to $y z_1$ non-gateways that are $i$-hop away from the closest gateways. Let $l_{\text{max}} := \max \{y z_1; y + \sum_j z_j \leq n\}$. According to (3), $z_j$ increases exponentially with $i$. In other words, the majority of non-gateways are $l_{\text{max}}$-hop away from the closest gateways; therefore, we use $l_{\text{max}}$ to approximate $l$. Thus, $z_j \approx z_{l_{\text{max}}} = n - y \approx n + 1 - y$ when $n$ is large. By solving $z_j = n + 1 - y$, we obtain

$$l = \frac{\log(n + 1 - y)}{\log(z_j/z_1)} + 1, \quad (9)$$

where $y = n(1 - (1 - 1/n)^\beta)$. By close examination of (9), we notice that it is also needed to guarantee $l \geq 1$. Hence, (9) can be calibrated as follows.

$$l = \begin{cases} \frac{\log(n + 1 - y)}{\log(z_j/z_1)} + 1 & \text{if } y \leq \frac{n+1}{n}, \\ 1 & \text{otherwise}. \end{cases} \quad (10)$$

We can verify that when $y = 1$, (10) reduces to (8) as expected. A key threshold $y_0 = (n + 1)/(z_1 + 1)$ is revealed in (10). When $y \leq y_0$, the distance from an arbitrary non-gateway to the closest gateway is relatively large; when $y > y_0$, there are sufficiently many gateways randomly distributed in one domain, causing each non-gateway to have a gateway neighbour with high probability. Hence,

$$\mathbb{E}[|\mathcal{P}_1|] = \mathbb{E}[|\mathcal{P}_1|] / |v_1 \not\in S|Pr(v_1 \not\in S) + \mathbb{E}[|\mathcal{P}_1|] / |v_1 \in S|Pr(v_1 \in S) + 1 = \left(1 - \frac{y}{n}\right) n + 1, \quad (11)$$

where $n - y$ is the percentage of non-gateway nodes in a domain. Putting (5), (8), and (11) into (7), the final expression of APC under MS is summarized in the following theorem.

**Theorem 7.** The APC in Type-1 Networks under MS (denoted by $L_{\text{Type-1}}^{\text{MS}}$) is

$$L_{\text{Type-1}}^{\text{MS}} = \begin{cases} \frac{n - y}{n} \left(1 - \frac{\log(n + 1 - y)}{\log(z_j/z_1)} + 1\right) + \frac{\log(n/z_1)}{\log(z_2/z_1)} + 1 & \text{if } y \leq \frac{n+1}{n}, \\ \frac{n - y}{n} + 2 + \frac{\log(n/z_1)}{\log(z_2/z_1)} + 1 & \text{otherwise}, \end{cases} \quad (12)$$

see Table 1 for notations.

It can be observed that the domain-wise distance ($\Delta$) and the number of gateways ($\gamma$) in domains are the most influential factors in shaping the APC for MS. Specifically, $L_{\text{Type-1}}^{\text{MS}}$ is logarithmic in domain structural parameters $n$, $y$, $z_1$, and $z_2$, and it is near linear in $\Delta$. Since SS coincides with MS in Type-1 Networks, we therefore discuss synchronization scenario PS in the next section.

7 **APC UNDER PS IN TYPE-1 NETWORKS**

In this section, we consider the partial synchronization (PS) model as defined in Definition 3. RCS are created as basic routing units according to Definition 4 under PS. Note that the network graph of an RC is no longer a random graph, because multiple domains are connected via inter-domain connections in a specific way as dictated by the network model. As such, we cannot directly apply the results obtained in Section 6 for the APC expression under MS in Type-1 Networks. Regarding such difficulties, in this section, we instead derive the APC lower bound for PS with the assistance of an auxiliary network called the Randomized Degree-Preserving Network (RDPN) (see Definition 8). Here is the sketch of our methodology.

**Sketch of Analytical Methodology:**

a) Given a domain-wise path, we identify all RCS along the path according to Definition 4;

b) We construct the RDPN associated with each RC;

c) We compute path cost incurred in RDPNs, and prove it is a lower bound of the actual path cost incurred in its original RC;

d) Adding up RDPN path costs and the number of inter-RC connections, we get the lower bound of APC under PS.

Based on this methodology, we next describe the details on how the APC lower bound under PS is derived.

7.1 **The Line Network and its Randomized Degree-Preserving Network (RDPN)**

We first formally define the following terms: (i) the line network that generalizes RCSs; and (ii) the Randomized Degree-Preserving Network (RDPN) of a line network. These concepts are also used in the analysis of Type-2 Networks.

**Definition 8.** A line network with $k$ domains is a special graph generated via the two-layer network model, consists of $k$ domains, where its domain-wise topology is a connected linear graph (i.e., a connected tree where no vertex has degree $3$ or more). The domains with inter-domain degree being $1$ and $2$ in a line network are called end-domains and transit-domains, respectively.

**Definition 9.** For a line network (denoted by $F$) with $k$ domains and $n$ nodes in each domain, the corresponding Randomized Degree-Preserving Network (RDPN) of $F$, denoted by $F_R$, is a randomly generated network with $kn$ nodes such that $F_R$ and $F$ have the same degree distribution.

**Discussion:** Although $F$ and $F_R$ have the same degree distribution and the number of nodes, they differ significantly from the perspective of randomness. In particular, $F$, as a line network, is constrained to certain structural properties, i.e., the domain-wise topology must be a linear graph with $k$ vertices. The RDPN $F_R$, however, is purely random without such constraints. Thus, let $S_F$ and $S_{F_R}$ be the sets of all graph instances of $F$ and $F_R$, respectively. There is $S_F \subseteq S_{F_R}$.

7.2 **Path Cost in RDPN**

With the concept of RDPN, we now show the relationships between path costs in the line network and its corresponding RDPN. Specifically, we discuss the minimum path cost between a randomly
chosen vertex and a vertex set in a line network and its corresponding RDPN. To this end, we first derive the following theorem.

Theorem 10. For a line network (denoted by \( F \)) consisting of \( k \) domains sequentially labeled as \( A_1, A_2, \ldots, A_k \), let \( F_R \) denote the RDPN of \( F \). Let \( \rho \) be the average path cost of the minimum-cost path between a random node \( \mu \in A_i \) and a random node set \( M \) \( (\mu \notin M, M \subseteq A_k) \), and \( \rho_F \) the average path cost of the minimum-cost path between a random node \( \mu \in F_R \) and a random node set \( M_R \) \( (\mu \notin M_R, M_R \subseteq F_R) \). Then, \( \rho_R \leq \rho \) holds.

With Theorem 10, the APC lower bound under PS can be obtained by combining the path costs of RDPNs of all associated RCs. Therefore, we only need to focus on the computation of path cost in each RDPN. Viewing each RDPN of RCs as a random graph following a certain degree distribution, we reaply the results in Section 6. Specifically, the first step of path cost calculation in a random network is to determine the number of 1-hop and 2-hop vertices from a randomly selected vertex. As such, we present the following lemma.

Lemma 11. In the RDPN of a line network \( F \) with \( k \) domains and the inter-domain connection parameter \( \beta \), let \( \xi_1 \) and \( \xi_2 \) denote the number of vertices that are 1-hop and 2-hop away from a random vertex, respectively. Then, the following holds: \( \xi_1 = z_1 + \frac{2\beta(k-1)}{nk} \), \( \xi_2 = z_2 + \frac{\beta(k-1)}{nk} \), if \( \beta < n \), where \( z_1 \) and \( z_2 \) are the average number of 1-hop and 2-hop nodes from a randomly chosen node within a domain in \( F \), respectively.

By applying (8), which gives an estimation of the path cost between two random nodes within a domain, and substituting relevant parameters of the RDPN, we can express the path cost between two random nodes in an RDPN as \( g(k) \), a function of the number of domains (\( k \)) in the RDPN:

\[
g(k) = \frac{\log(nk/\xi_1^\gamma)}{\log(\xi_1^\gamma)} + 1, \quad (13)
\]

where \( \xi_1 \) and \( \xi_2 \) are defined in Lemma 11. In (13), it estimates the path cost between two random nodes. However, as discussed in Section 6, path construction needs to consider the gateway selection in domains that are not the destination domain. Similarly, in an RC that does not contain the destination node, the constructed path in its RDPN is the minimum-cost path from a random vertex to a random vertex set with the cardinality \( \gamma \). Therefore, by applying (10) and considering the probability of a random vertex not belonging to the random vertex set, the path cost in an RDPN that does not include the destination node is

\[
h(k) = \begin{cases} 
\frac{nk-\gamma}{nk} \left( \frac{\log(n^k/\xi_1^\gamma)}{\log(\xi_1^\gamma)} \right) + 1, & \text{if } \gamma \leq \frac{nk+1}{1+\gamma}, \\
\frac{nk}{nk-\gamma}, & \text{otherwise}.
\end{cases} \quad (14)
\]

7.3 APC lower bound for PS

In a line network, the path cost in all RCs, excluding the one with the destination node, is estimated by (14), whereas the path cost in the RC that includes the destination node is estimated by (13). Thus, the APC lower bound under PS is

\[
\rho_{PS}^{lower} = \begin{cases} 
\eta_1(h(\tau^\gamma) + 1) + g(\tau^\gamma) & \text{if } \eta^\gamma = 0, \\
(\eta + 1)(h(\tau^\gamma) + 1) + g(\eta^\gamma) & \text{if } \eta^\gamma > 0.
\end{cases} \quad (15)
\]

Hence, when \( \eta^\gamma = 0 \),

\[
\rho_{PS}^{lower} \eta^\gamma = \begin{cases} 
\eta_1(1 + \frac{\log(n^\gamma/\xi_1^\gamma)}{\log(\xi_1^\gamma)}) + \xi + 1 + \frac{\log(n^\gamma/\xi_1^\gamma)}{\log(\xi_1^\gamma)} + 1 & \text{if } \gamma \leq \frac{n^\gamma+1}{\xi_1^\gamma+1}, \\
\eta_1(\xi + 1) + \log(n^\gamma/\xi_1^\gamma) + 1 & \text{otherwise},
\end{cases} \quad (16)
\]

where \( \xi = 1-\frac{\beta}{n^\gamma} \). When \( \eta^\gamma > 0 \), we have \( \eta_2 = (\Lambda+1) \mod \tau^\gamma = \eta^\gamma \); therefore,

\[
\rho_{PS}^{lower} \eta^\gamma > 0 = \begin{cases} 
(\eta_1 + 1)(1 + \frac{\log(n^\gamma/\xi_1^\gamma)}{\log(\xi_1^\gamma)}) + \frac{\log(n^\gamma/\xi_1^\gamma)}{\log(\xi_1^\gamma)} + 1 & \text{if } \gamma \leq \frac{n^\gamma+1}{\xi_1^\gamma+1}, \\
(\eta_1 + 1)(\xi + 1) + \log(n^\gamma/\xi_1^\gamma) + 1 & \text{otherwise}.
\end{cases} \quad (17)
\]

Clearly, \( \rho_{PS}^{lower} \) is linear in the number of RCs, and is logarithmic in network structural parameters such as \( n \) and \( \gamma \). This suggests that enlarging the synchronization radius to reduce the number of RCs on the domain-wise path results in near linear reduction in APC.

8 APC UNDER CS IN TYPE-1 NETWORKS

For complete synchronization (CS), all SDN domains are synchronized, which is equivalent to the case where there exists a logical centralized controller in the network. Therefore, under CS, all controllers can make the global optimal decisions that generate the end-to-end path with minimum path cost. In this regard, we first study whether RCPC can construct such global optimal path, and then establish the APC expression under CS in Type-1 Networks.

Given two arbitrary nodes \( v_1 \) and \( v_2 \), suppose the shortest domain-wise path \( P^* \) w.r.t. \( v_1 \) and \( v_2 \) contains \( k \) vertices in the domain-wise topology. If \( P^* \) (selected by RCPC) corresponds to the minimum-cost path between \( v_1 \) and \( v_2 \), then the APC lower bound under CS can be easily obtained by calculating the APC between two random nodes in the end-domains of a line network consisting of \( k \) domains. However, for the global minimum-cost path, it may visit more than \( k \) domains to yield the minimum end-to-end path cost. We, therefore, examine how the domain-wise shortest path \( P^* \) is related to the global minimum-cost path between \( v_1 \) and \( v_2 \) in the following.

Theorem 12. Let \( L_k(\beta) \) be the APC between two random nodes in the two end-domains of a line network, which consists of \( k \) domains and all inter-domain connections are governed by parameter \( \beta \). Then, \( L_k(\beta) = L_{k+1}(\beta) \) when \( k \geq 3 \).

Theorem 12 reveals a property of \( L_k(\beta) \) that is vital to our analysis, i.e., a longer domain-wise path incurs higher end-to-end path cost if the shortest domain-wise path between two nodes contains at least three vertices. Note that in Theorem 12, there are two uncovered cases. First, \( k = 1 \). Since we are not interested in determining APC for two random nodes within the same domain, this case does not exist. Second, \( k = 2 \). From numerical results, we observe that \( L_2(\beta) \) may be slightly greater than \( L_3(\beta) \) when \( \beta \) satisfies certain
conditions. Nevertheless, the case that two random nodes residing in two neighbouring domains only happen with probability $z_i^2/m$, which can be ignored as $z_i^2 \gg z_i^2$ and $m$ is large.

An implicit assumption for Theorem 12 is that the domain-wise path associated with the constructed path is a simple path, i.e., a path without repeated vertices. To show that visiting more domains cannot construct a shorter end-to-end path, we still need to prove that visiting one domain more than once is also disadvantageous. To this end, we define $L_k^\prime(\beta)$ which is the same as $L_k(\beta)$ except that the corresponding domain-wise path contains repeated vertices.

**Corollary 13.** For the two-layer network model, $L_k(\beta) < L_k^\prime(\beta)$ for $3 \leq k \leq k'$.

Theorem 12 together with Corollary 13 suggest the following corollary.

**Corollary 14.** For any source-destination node pair residing in different domains, on average, the optimal path between them traverses the minimum number of domains.

Recall that the average number of domains on the shortest domain-wise path between two random domains is $\Delta + 1 = \log(m/z_i^2) + 2$. Therefore, we compute the APC under CS based on a domain-wise path traversing $\Delta + 1$ domains. Under CS, the path construction in each domain is independent of other domains’ structures, thus complicating the mathematical analysis. We, therefore, leverage RDPN of a line network with $\Delta + 1$ domains to estimate the APC for CS, which is a lower bound according to Theorem 10. Thus, reapplying (13) with $\Delta + 1$ as the input, we obtain the APC lower bound for CS, denoted by $L_{CS}^{\text{lower}}$:

$$
L_{CS}^{\text{lower}} = g(\Delta + 1) = \frac{\log(n \log(m/z_i^2) + \Delta n)}{\log(z_i^2/z_1^2)} + 1. \quad (18)
$$

The expression of $L_{CS}^{\text{lower}}$ shows a function that bounds the APC under the best-case scenario, i.e., CS, which therefore is also a lower bound under other synchronization scenarios. Since (18) is a logarithmic function of a logarithmic function, it suggests that the routing efficiency can be significant if CS is achieved in the network. Moreover, under CS, (18) is of the form of $\log(n \log(m))$, showing that the number of nodes $n$ has a stronger impact than the number of domains $m$ on the value of $L_{CS}^{\text{lower}}$, i.e., intra-domain routing is more critical.

### 9 UNIVERSAL APC LOWER BOUND

In this section, we present the Universal APC lower bound, which provides an estimation of APC under any synchronization levels for both Type-1 and Type-2 Networks. The phrase lower bound carries two separate meanings. First, it summarizes the APC obtained for MS and the APC lower bounds obtained for PS and CS in Type-1 Networks. Second, since link preference is at least 1 for Type-2 Networks, this universal lower bound derived for Type-1 Networks also applies to Type-2 Networks.

**Theorem 15.** Universal APC lower bound: Given the synchronization radius $\tau$, the lower bound of APC (denoted by $L^{\text{lower}}$) in the two-layer network model is

$$
L^{\text{lower}} = \begin{cases} 
\eta_1 \frac{\log(n \log(m/z_i^2) + \Delta n)}{\log(z_i^2/z_1^2)} + \log(m/z_i^2) + \eta_1 (\xi + 1) + 1 & \text{if } \eta \leq \frac{\tau^2 + 1}{\xi + 1} \\
\log(n \log(m/z_i^2) + \Delta n) + \log(m/z_i^2) + \eta_1 (\xi + 1) + 1 & \text{otherwise, }
\end{cases} \quad (19)
$$

where $\tau = \min(\tau, \Delta + 1)$, $\eta_1 = [(\Delta + 1)/\tau'] - 1$, $\eta_2 = (\Delta \mod \tau') + 1$, and $\xi = 1 - \frac{\tau^2}{\eta_1}$.

In (19), $L^{\text{lower}}$, requiring the network topologies and synchronization levels as inputs, is a logarithmic function non-increasing with $\tau$. Moreover, when the number of gateways in each domain is sufficiently large (i.e., large $\gamma$), the expression of $L^{\text{lower}}$ is significantly simplified due to easier inter-domain routing. In addition, the synchronization radius $\tau$, representing different levels of inter-domain synchronizations, is instrumental in determining the APC lower bound $L^{\text{lower}}$. For example, (19) reduces to the (12) for MS in Type-1 Networks when $\tau = 1$; (19) reduces to (16) for PS when $\eta' = 0$.

**Discussion:** Sections 6, 7, 8 jointly prove the correctness of the universal APC lower bound in Theorem 15 for Type-1 Networks. In addition, since all link preference levels in Type-2 Networks are at least 1, this universal APC lower bound still holds in Type-2 Networks, thus providing insights into the routing performance under any synchronization and network scenarios. In Sections 10–13, we derive fine-grained APC expressions under different synchronization scenarios in Type-2 Networks. More importantly, these fine-grained APC expressions can also be applied to Type-1 Networks by setting all edge weights to 1.

### 10 APC UNDER MS IN TYPE-2 NETWORKS

In this section, we present the APC expression under MS in Type-2 Networks, denoted by $L^{\text{MS}}$. Though edges in Type-2 Networks exhibit various edge weights, such weight information is not available to any controllers under MS. Thus, the path construction from the source to the destination is independent of the edge weight distributions. Recall that in our two-layer network model, all intra-domain link weights are modeled as a given i.i.d. r.v., denoted by $W$, and all inter-domain edges are of weight 1. Hence,

$$
L^{\text{MS}} = \Delta \cdot (\mathbb{E}[|P_1|] \cdot \mathbb{E}[W] + 1) + \mathbb{E}[|P_{\Delta + 1}|] \cdot \mathbb{E}[W] + \Delta 
$$

where $P_1$ and $l$ are defined in (1) and (10), respectively. Substituting the expressions of $l$ and $\Delta$ into (20), we obtain the full expression of $L^{\text{MS}}$ under MS in Type-2 Networks:

$$
L^{\text{MS}} = \begin{cases} 
\left(n - \eta \left(\frac{n \log(m/z_i^2) + \Delta n}{\log(z_i^2/z_1^2) + 1}\right) + 1\right) \mathbb{E}[W] + \log(m/z_i^2) + 1 & \text{if } \eta \leq \frac{\tau^2 + 1}{\xi + 1}, \\
\left(n - \eta \left(\frac{n \log(m/z_i^2) + \Delta n}{\log(z_i^2/z_1^2) + 1}\right) + 1\right) \mathbb{E}[W] + \log(m/z_i^2) + 1 & \text{otherwise. }
\end{cases} \quad (21)
$$

It is verifiable that (21) is same as (12) when $\mathbb{E}[W] = 1$, i.e., the Type-2 Network is reduced to the Type-1 Network.

### 11 APC UNDER SS IN TYPE-2 NETWORKS

SS is a special synchronization scenario that only exists in Type-2 Networks. Similar to MS, under SS, no two domains synchronize. To analyze APC under SS, we first introduce a new concept, named path cost distribution, as the basis for further analysis. Here is the sketch of our analytical methodology.

**Sketch of Analytical Methodology:**
a) We compute the distribution of the path cost (in terms of accumulated link preference levels) between two random nodes within the same domain, called intra-domain path cost distribution; 
b) By (1), we need to determine the average cost of $\mathcal{P}_i$ for $i = 1, 2, \ldots, \mu$. Since the total number of RCs is the same as that under MS, we have that the expected value of $\mu$ in (1) is $\Delta + 1$; 
c) As all controllers involved in the path construction process follow the same procedure, similar to (7), it suffices to only quantify the average cost of $\mathcal{P}_1$ and $\mathcal{P}_{\Delta+1}$ using the intra-domain distance distribution derived in a).

Based on this methodology, we next discuss the details.

11.1 Intra-Domain Path Cost Distribution

In one domain, consider a path with $\lambda$ links. Let $W_1, W_2, \ldots, W_\lambda$ be i.i.d. r.v. of link weights on this path with the probability density functions (pdf) being $f_W(x) = f_W(x) = \ldots = f_W(x)$. Let the path pdf of $W_j$ is the convolution of the pdfs of $W_1, W_2, \ldots, W_j$, i.e., $f_{W_j}(x) = f_{W_1}(x) \ast f_{W_2}(x) \ast \ldots \ast f_{W_j}(x)$. By the principle in mixture distribution [28], we still need to know the pdfs $f_{W_j}$ that the minimum-cost path between two random nodes containing $\lambda$ links. Note that when $\lambda = 0$, $z_0 = 1$ and the cumulative distribution function (cdf) of $W_0$ is a step function [29]. Let r.v. $D$ be the minimum path cost (in terms of accumulated link weights) between two random nodes in one domain, with the pdf being $f_D(x)$, i.e., intra-domain distance distribution. Then by mixture distribution, $f_D(x)$ can be estimated as follows

$$f_D(x) = \sum_{i=0}^{h_{\max}} p_{W_i} f_{W_i}(x) = \sum_{i=0}^{h_{\max}} z_i/n \cdot f_{W_i}(x),$$

(22)

where $h_{\max} := \arg \max_i z_i$ s.t. $\sum_{i=0}^{h_{\max}} z_i \leq n$. Hence, the APC between two nodes in one domain $\mathbb{E}[D]$ can be computed using (22).

11.2 Domain-wise Path

Though SS and MS represent different synchronization levels, the corresponding domain-wise paths are exactly the same w.r.t. a pair of source and destination nodes in a given network. Thus, by (5), again, we have that $\mu$ in (1) equals $\Delta + 1$. Let $L(P)$ be the end-to-end accumulated link preference levels (i.e., cost) of path $P$, which is a random variable. Then the expectation of $L(P)$, i.e., the APC for SS in Type-2 Networks, denoted by $L_{\text{SS}^{\text{type}}}$, is

$$L_{\text{SS}^{\text{type}}} = \mathbb{E}[L(P)] = \mathbb{E}[L(P_1) + L(P_2) + \ldots + L(P_{\Delta+1})] + \Delta = \Delta \cdot \mathbb{E}[L(P_1)] + \mathbb{E}[L(P_{\Delta+1})] + \Delta$$

(23)

The reason for the last row in (23) is that $\mathbb{E}[L(P_{\Delta+1})]$ essentially is the path cost between two nodes in one domain. Thus, it suffices to determine $\mathbb{E}[L(P_1)]$ next, i.e., the minimum path cost between a random node and the closest gateway in one domain connecting to the next domain on the domain-wise path.

11.3 Minimum Path Cost Between An Arbitrary Node and Gateways

Section 11.1 provides the estimation of path cost between two arbitrary nodes in Type-2 Networks. Based on (22), we quantify the path cost between an arbitrary node and the gateway from the candidate gateway set that incurs the minimum path cost, which is formally presented in the following theorem:

**Theorem 16.** Let r.v. $M^{(\beta)}$ denote the path cost between an arbitrary node and gateway in the candidate gateway set (established with parameter $\beta$) that incurs the minimum path cost. Then, the pdf of $M^{(\beta)}$ is:

$$f_{M^{(\beta)}}(x) = \begin{cases} (1 - F_D(x - 1))^\beta & \text{for } x \geq 1, \\ (1 - (1 - F_D(0))^\beta & \text{for } x = 0. \end{cases}$$

(24)

With Theorem 16, we can derive $\mathbb{E}[L(P_1)] = \mathbb{E}[M^{(\beta)}]$ using the pdf expression in Theorem 16. Then, substituting (5), $\mathbb{E}[L(P_1)]$, and $\mathbb{E}[D]$ into (23), we get the expression of the APC under SS in Type-2 Networks, denoted by $L_{\text{SS}^{\text{type}}}$:

$$L_{\text{SS}^{\text{type}}} = \Delta \cdot \int_{x=0}^{+\infty} x f_M^{(\beta)}(x) + 1 + \int_{x=0}^{+\infty} x f_D(x) + \Delta.$$

(25)

Comparing to $L_{\text{SS}^{\text{type}}}$, the expression of $L_{\text{MS}^{\text{type}}}$ is more complicated as we do not impose any constraint on the distributions of link preference levels. Nevertheless, it is verifiable that $L_{\text{SS}^{\text{type}}}$ is smaller than $L_{\text{MS}^{\text{type}}}$, thus bounded by $L_{\text{MS}^{\text{type}}}$.

12 APC UNDER PS IN TYPE-2 NETWORKS

To compute the corresponding APC under PS in Type-2 Networks, denoted by $L_{\text{PS}^{\text{type}}}$, we first present the APC w.r.t. two nodes with their shortest domain-wise path containing exactly $q$ vertices, denoted by $L_q$, in the following theorem.

**Theorem 17.** Let $L_q$ denote the APC between two arbitrary nodes under PS with synchronization radius $\tau$ in Type-2 Networks where there are $q$ domains on the domain-wise path. Then, we have

$$L_q = \begin{cases} ([q/\tau] - 1) \cdot (\mathbb{E}[M^{(\beta)}] + 1) + \mathbb{E}[D^{(\beta)}] & \text{if } \theta = 0; \\ ([q/\tau] - 1) \cdot (\mathbb{E}[M^{(\beta)}] + 1) + \mathbb{E}[D^{(\beta)}] & \text{if } \theta > 0. \end{cases}$$

(26)

where $\theta = q \mod \tau$, $M^{(\beta)}$ is the r.v. of the minimum path cost incurred in an RC with $i$ non-destination domains, and $D^{(\beta)}$ is the r.v. of the minimum path cost incurred in the RC with $i - 1$ non-destination domains and the destination domain; see Section 11.3 for the computation of $\mathbb{E}[M^{(\beta)}]$ and $\mathbb{E}[D^{(\beta)}]$.

Recall that the probability that two arbitrary nodes with their domain-wise path containing $q$ domains is $z_q' (m-1) = z_q'/m$. Therefore, the APC under PS in Type-2 Networks, $L_{\text{PS}^{\text{type}}}$, is

$$L_{\text{PS}^{\text{type}}} = \sum_{q=2}^{h_{\max} + 1} L_q z_q' - 1/m.$$  

(27)

where $h_{\max}' := \arg \max_i z_i'$. Comparing to $L_{\text{SS}^{\text{type}}}$ and $L_{\text{MS}^{\text{type}}}$, $L_{\text{PS}^{\text{type}}}$ is more complicated due to path construction correlations in RCs; its accuracy is evaluated in Section 14.

13 APC UNDER CS IN TYPE-2 NETWORKS

For complete synchronization (CS), all SDN domains are synchronized with the complete view of the Type-2 Network. As the case of Type-1 Network, globally optimal routing decisions are made for
CS in Type-2 Networks as well. Here, let $L_k(\beta) := \mathbb{E}[D_k^{(\beta)}]$, where $D_k^{(\beta)}$ is the r.v. of the minimum path cost incurred in the RC with $k \rightarrow 1$ non-destination domains and the destination domain. By close examination of $L_k(\beta)$, some additional conclusions are made in the following corollaries.

**Corollary 18.** For the two-layer network model, $L_{k+1}(1) - L_k(1) = \mathbb{E}[|D_k|] + 1$.

**Corollary 19.** For the two-layer network model, $\lim_{\beta \rightarrow \infty} (L_{k+1}(\beta) - L_k(\beta)) = 1$.

Note that Theorem 12 and Corollary 13 remain valid for the Type-2 Network scenario, which suggest that for any source-destination node pair residing in different domains, on average, the optimal path between them traverses the minimum number of domains. Therefore, when the shortest domain-wise path between two nodes contain $k$ vertices, we can use $L_k(\beta)$ to approximate the corresponding optimal APC. Thus, let $L_{CS}^{\text{Type-2}}$ denote the APC under CS in Type-2 Networks. We have

$$L_{CS}^{\text{Type-2}} \approx \sum_{k=2}^{n} \mathbb{E}[|D_k|]z_{k-1}/m = \frac{h_{\text{max}}^{(\beta)}}{m},$$

(28)

where $h_{\text{max}}^{(\beta)}$ is defined in (27). Though experiencing high complexity due to global cross-domain routing optimality, $L_{CS}^{\text{Type-2}}$ is shown in Section 14 to have high accuracy in estimating APC under CS. An efficient computation of $\mathbb{E}[|D_k^{(\beta)}|]$ is also documented in Appendix A.10.

14 EVALUATIONS

To evaluate our analytical results of distributed SDN for various synchronization scenarios, we conduct two sets of experiments (called Evaluation 1 and Evaluation 2), with different focuses, on network topologies generated from both real and synthetic datasets. In Evaluation 1, we test the accuracy of the asymptotic analysis presented in Theorem 5, which, in its concise form, demonstrates the interplay of different parameters in determining the overall APC. Second, we validate the accuracy of the derived fine-grained expressions for $L_{\text{Type-2}}$, $L_{\text{Type-1}}$, $L_{\text{Type-2}}$, and $L_{\text{Type-2}}$ in Type-2 Networks in Evaluation 2. We compare these theoretical results with the actual APCs collected from the above networks. Based on these evaluation results, we can validate the accuracy of our theoretical results and observe to what extent synchronization levels and network structural properties affect APCs.

14.1 Network Realizations

14.1.1 Network Topologies Based on Real Datasets. To generate network topologies based on real datasets, we need the degree distributions as the input. Specifically, we use the real datasets collected by the University of Oregon Route Views Project (Routeview project) [30], the Rocketfuel project[31], and the CAIDA project [32] for input degree distributions.

Given a specific degree distribution, one graph realization is generated in the following way: We assign each vertex (the total number of vertices is given) 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.

14.2 Evaluation Settings

14.2.1 Evaluation 1. In Evaluation 1, we conduct three experiments in networks with varying gateway connection parameter $\beta$, varying synchronization radius $\tau$, and varying number of nodes in each domain $n$, respectively. The APCs collected from these simulated networks are compared with the predictions made by the asymptotic expressions. Intra-domain degree distributions for three experiments are all derived from Rocketfuel “AS 1239”, in which $z_1 = 6.165$, and $z_2 = 41.835$. We configure the domain-wise topologies to have an average domain-wise distance of 10, using statistics collected from CAIDA “AS 27524”. Unless otherwise specified, the default parameter settings for three experiments are: $\beta = 5, \tau = 2,$
and \( n = 200 \). For each value of the varying parameter, the result plotted is the average of 30 topology realizations with 50 random selections of source-destination node pairs for each topology realization.

### 14.2.2 Evaluation 2

For Evaluation 2, three evaluation cases are studied to validate fine-grained APC expressions derived: (i) Case 1, where we use intra- and inter-domain degree distributions extracted from CAIDA and Routeview datasets; (ii) Case 2, where the intra-domain and inter-domain degree distribution are derived from randomly selected Rocketfuel topologies (AS 1239) and the CAIDA Internet topology dataset (AS 27524), respectively; (iii) Case 3, where all intra-domain topologies are BA graphs and the inter-domain topology is an ER graph (we pick \( p = 0.015 \) for ER graphs, and \( q = 1 \) for BA graphs). In all three cases, the distribution of link preference levels (weight) is derived from Rocketfuel topologies, i.e., the intra-domain link preference ranges from 1 to 16 with the expectation and variance being 3.2505 and 4.5779, respectively. For each case, the two-layer network consists of 100 domains, each containing 200 nodes, i.e., \( m = 100 \) and \( n = 200 \). For a given \( \beta \), 30 two-layer networks are realized. In each network realization, 50 source-destination pairs (in different domains) are randomly selected to construct paths between them under each of MS, SS, PS, and CS. In addition, for PS, two special cases, i.e., \( \tau = 2 \) and \( \tau = 3 \), are studied to compare against other synchronization scenarios. It should be noted that these settings are determined arbitrarily, as our analytical model does not require the input degree distributions to have any patterns/properties.

### 14.3 Evaluation Results

The simulated APCs and the APCs estimated by the asymptotic analysis are presented in Fig. 6(a)-(c) for Evaluation 1. For Evaluation 2, the simulated APC averaged over all network realizations and source-destination node pairs are reported in Fig. 7(a)-(c), for the three simulation cases, respectively. In these figures, each curve is also accompanied by our developed fine-grained APC estimations.

#### 14.3.1 Accuracy of the Theoretical Results

The asymptotic analysis is conducted to enable direct and clear observations of the relationships between APC and parameters related to synchronization levels and network structural properties. The asymptotic analysis’s ability to reveal these relationships is confirmed in Evaluation 1. From three figures in Fig. 6, we can see that the trends in APC changes with varying parameters are closely captured by the curves obtained using expressions of the asymptotic analysis, as the simulation and analysis curves have common shapes. The presence of the gap between two curves is due to the fact that the asymptotic analysis is only intended to highlight the relative relationship among different parameters in simple expressions, and thus it is not meant to be employed as an exact estimation. In comparison, the evaluations of various real/synthetic networks in Evaluation 2 demonstrated in Fig. 7 confirm the high accuracy of our fine-grained theoretical results in predicting the performance metric APC in distributed SDN networks. Specifically, the simulation curves can be closely approximated by the theoretical results for all values of \( \beta \) and synchronization scenarios. Moreover, the theoretical results for PS and CS are obtained by the efficient computation method in Appendix A.10. Fig. 7 shows that even such simplified method for approximating \( L_k(\beta) \) exhibits high accuracy. Intuitively, this is because the process of establishing inter-domain connections in our network model is purely random, which enables us to use the approximation method in Appendix A.10 to estimate the number of path construction options between two random nodes in the end-domains of a line network of length \( k \).

#### 14.3.2 APC Variations for Different Synchronization Levels and Structural Parameters

Both Fig. 6 and Fig. 7 confirm that the APC in distributed SDN is related to the amount of information available to the controllers, i.e., synchronization levels. As expected, higher synchronization levels are superior in reducing APCs. This can be observed in Fig. 6(b), where APC decreases when the synchronization radius \( \tau \) gets larger. For Evaluation 2, Fig. 7 shows that APC for CS corresponds to the minimum APC that is achievable in all cases, i.e., a lower bound. By contrast, the results for MS act as an upper bound due to the minimum intra-/inter-domain information availability. Since the APC for MS is expressed as a logarithmic function (21), Fig. 7 shows that even with the minimum synchronization level, APC is still significantly smaller than the network size (20,000 nodes in total) when link preference levels are at least 1. Fig. 7 shows that comparing to MS, the APC reduction for CS can be up to 70%. Moreover, comparing to MS, only intra-domain link preference information is available to SS. Nevertheless, such additional information is able to reduce APC by up to 30%. However, when more synchronized information is available, the reduction in APC starts to degrade (i.e., diminishing return). In particular, for PS, comparing against the case of \( \tau = 2 \), the APC reduction for \( \tau = 3 \) is rather small, especially when \( \beta \) is small. This observation is also confirmed by Fig. 6(b) where the most significant decrease in APC takes place when \( \tau \) change from 1 to 2. Consequently, it is expected that with the increase of \( \tau \), the benefit to cost ratio declines sharply.
In addition, we observe that the network performance improves when $\beta$ increases. This is intuitive as a larger $\beta$ directly renders the probability of finding a shorter path to be notably high, as there exist more inter-domain connections. In fact, Fig. 6(a) and Fig. 6(b) show that on average, increasing $\beta$ is more effective in reducing APC than increasing the synchronization radius. Furthermore, Fig. 6(a) and Fig. 7 also demonstrate that APC converges to a certain value when $\beta$ is large, which can be explained by Corollaries 18–19.

Finally, Fig. 6(c) reveals that the size of the network does not have a significant impact on APC. Specifically, when the number of nodes triples from 100 to 300 in each domain, APC only marginally increases by 6. Moreover, given that in Evaluation 1 there are on average 10 domains on the domain-wise path, this gives an average increase of APC by 0.3 in each domain.

In summary, these evaluation results reveal that in distributed SDN, the performance improvement space is only marginal when domains synchronize with other domains in an increasingly larger radius, or when each domain adds more gateways while the number of existing gateways is already large. Such constraints need to be addressed in practical network design and optimizations.

15 CONCLUSIONS

We have studied the performance of distributed SDN networks for different inter-domain synchronization levels and network structural properties from the analytical perspective. For this goal, a generic network model is first proposed to capture key attributes of existing gateways is already large. Such constraints need to be addressed in practical network design and optimizations.

REFERENCES


A APPENDIX

A.1 Proof of Theorem 5

Theorem 5 is derived by further analysis of the APC expressions obtained in Type-1 Network, which is summarized in Theorem 15. Specifically, in (19),
\[
\log \left( \frac{n^r + 1 - y}{n^r} \right) = \log \left( \frac{n^r}{n^r} \right) + \frac{y - 1}{\frac{n^r}{n^r}} \leq \log \left( \frac{n^r}{n^r} \right),
\]
\[
\text{as } y \geq 1. \quad \text{Next, for } y_1 \text{ in (19), we know }
\]
\[
y_1 = \log \left( \frac{m(z_1)}{r(z_1)^{z_1}} \right) \leq \frac{2}{r(z_1)^{z_1}} - 1 \leq \log \left( \frac{m(z_1)}{r(z_1)^{z_1}} \right) + \frac{2}{r(z_1)^{z_1}} - 1. 
\]
There are four cases for (30). First, when $r = 1$, then $y_1 \leq \log \left( \frac{m(z_1)}{r(z_1)^{z_1}} \right)$. Third, when $2 < r \leq \frac{m(z_1)}{r(z_1)^{z_1}}$.
in $T_i$, in $T_j$, $v$ either remains unchanged or moves further away from the root. Hence, let $\rho'$ be the expected cost of the shortest path (in terms of hop counts) between any two random nodes in $\mathcal{F}$, we have $\rho_R \leq \rho'$ when $|M| = |M_R| = 1$. On the other hand, when $|M| = |M_R| > 1$, we can view root $r$ above as a set of nodes in $\mathcal{F}$ (or $\mathcal{F}'$) with the cardinality the same as that of $M_R$ and $M$ (i.e., a root set); then the depth of a node $v$ in the tree represents the shortest path (in terms of hop counts) from $v$ to the closest node in this "root set". In this way, the above argument still applies when $|M| = |M_R| > 1$, and again we get $\rho_R \leq \rho'$.

So far, we only discuss pure random node set selection of $M$ (which forms the root set) in $T_j$, which includes situations where some nodes in $M$ are in transit-domains (defined in Definition 8), i.e., $A_2, \ldots, A_{k-1}$. However, we are only concerned with the distance between node $v$ ($v \in A_1$) and node set $M (M \subseteq A_k)$ in $\mathcal{F}$. It is obvious that the cost of such shortest path ($\rho$) is even larger than $\rho'$, since $\rho'$ is the average value over all scenarios, each of which is no worse (i.e., shorter or equal path cost) than the case we are interested in. Thus, we have $\rho_R \leq \rho' \leq \rho$. □

A.3 Proof of Lemma 11

In order to derive the number of 1-hop and 2-hop nodes from a random node in the RDPN of a line network with $k$ domains, the first step is to obtain the degree distribution of the RDPN which incorporates the added degree from the inter-domain connections on top of the intra-domain connections.

Let $\chi$ be the random variable of the degree distribution of a node in a domain before inter-domain connections are established (i.e., intra-domain connections). Denote the random variables of the added degrees of nodes in end-domains and transit-domains (defined in Definition 8) in a line network by $\psi$ and $\gamma$, respectively, after the inter-domain connections are established. Then, when $\beta \ll n, \psi$ and $\gamma$ follow binomial distribution with the following parameters: $\psi ~ B \left( \beta, \frac{k}{k-1} \right)$ and $\gamma ~ B \left( \gamma, \frac{1}{2} \right)$. Then the overall degree distributions for a random node in a line network of length $k (k \geq 2)$ is represented by $(\chi + \frac{k}{k-1}\psi + \frac{1}{2}\gamma)$.

The average number of 1-hop nodes from a random node within a domain, denoted by $z_1$, is $z_1 = \mathbb{E} [\chi] = \sum_{k=0}^{\infty} kp_k$, where $p_k$ is the percentage of nodes with degree $k$. For the RDPN, whose degree distribution is captured by random variable $\chi \sim B \left( \chi, \frac{k}{k-1} \psi + \frac{1}{2}\gamma \right)$, we have $\zeta_1 = \mathbb{E} [\chi] + \frac{1}{2} \mathbb{E} [\psi] + \frac{1}{4} \mathbb{E} [\gamma]$. The mean of $\chi$ is $z_1$. As for $\psi$ and $\gamma$, since they follow binomial distribution, their means are $\frac{\beta}{k}$ and $\frac{1}{2}$, respectively. Thus, we have $\zeta_1 = z_1 + \frac{2\beta(k-1)}{nk}$. For the calculation of the average number of 2-hop nodes from a random node, denoted by $z_2$, [27] gives an expression: $z_2 = \sum_{k=0}^{\infty} k(k-1)p_k$. However, it is difficult to directly apply this expression to derive $\zeta_2$. Instead, we use $z_2 = \sum_{k=0}^{\infty} k(k-1)p_k = \sum_{k=0}^{\infty} k^2 p_k$ as an approximation. The calculation now becomes tractable, because for a random variable $x$, the following result holds: $\mathbb{E} [x^2] = \sigma_x^2 + \mathbb{E} [x]$, where $\sigma_x$ is the variance of $x$. Therefore, we can calculate $\zeta_2$ in the following way: $\zeta_2 = \mathbb{E} \left[ \chi + \frac{1}{2} \psi + \frac{1}{2} \gamma \right] = \sigma_x + \frac{1}{2} \psi + \frac{1}{2} \gamma + z_1$. Since $\chi$, $\psi$ and $\gamma$ are independent random variables, $\sigma_x^2 + \frac{1}{4} \psi^2 + \frac{1}{4} \gamma^2 = \sigma_\chi^2 + \frac{1}{2} \psi^2 + \frac{1}{2} \gamma^2 = z_1$. Given that the variance of a random variable $x \sim B(n, p)$
is $np(1-p)$, we calculate $\sigma_1^2, \sigma_2^2,$ and $\sigma_3^2$ accordingly. Since we assume $\beta \ll n$, we have $\beta/n^2 \approx 0$ and $\beta^2/n^2 \approx 0$; therefore, we conclude that $\xi_2 = z_2 + z_1 - \frac{4\beta(k-1)}{nk}$.

\[ \square \]

A.4 Proof of Theorem 12

Consider two line networks of length $k$ and $k+1$, whose domains are labeled as $\mathcal{A}_1, \mathcal{A}_2, \ldots, \mathcal{A}_k$ and $\mathcal{B}_1, \mathcal{B}_2, \ldots, \mathcal{B}_{k+1}$, respectively. Let $V_m(C_i)$ ($V_{out}(C_i)$) be the set of gateways in domain $C_i$ connecting to domain $C_{i-1}$ ($C_{i+1}$). Without loss of generality, we assume that $\mathcal{A}_1 = \mathcal{B}_1$ and $\mathcal{A}_k = \mathcal{B}_{k+1}$. This implies that $V_{out}(\mathcal{A}_1) = V_{out}(\mathcal{B}_1)$ and $V_{in}(\mathcal{A}_k) = V_{in}(\mathcal{B}_{k+1})$. Therefore, $L_k(\beta)$ and $L_{k+1}(\beta)$ are determined by the pair-wise distance between $V_{out}(\mathcal{A}_1)$ and $V_{in}(\mathcal{A}_k)$, and $V_{out}(\mathcal{B}_1)$ and $V_{in}(\mathcal{B}_{k+1})$, respectively. When $k \geq 3$, there exist at least one transit domain, apart from the end-domains in a line network (see Definition 8 for the concept of transit/end-domains). Since all transit domains have the same statistical parameters, each transit domain offers the same probability of finding a path with certain path cost in that domain. Furthermore, more transit domains introduce more inter-domain edges. Thus, there is no higher possibility of finding a path with lower path cost due to the presence of more transit domains. Therefore, more transit domains result in higher expectation of pair-wise distance between $V_{out}(\mathcal{B}_1)$ and $V_{in}(\mathcal{B}_{k+1})$. Moreover, these arguments apply to both Type-1 and Type-2 networks. This concludes the proof.

\[ \square \]

A.5 Proof of Corollary 13

We start the proof by comparing $L^*_4(\beta)$ and $L_k(\beta)$. We consider the simplest form of domain repetition where only one domain is traversed twice. We use Fig. 8 to facilitate the proof, where $k' = 4$. Suppose that a random node in domain $\mathcal{A}_1$ needs to communicate with a random node in domain $\mathcal{A}_3$ and the selected domain-wise path is $\mathcal{A}_1 - \mathcal{A}_2 - \mathcal{A}_3 - \mathcal{A}_4$. Apparently, this is not a simple path because domain $\mathcal{A}_1$ is traversed twice; the corresponding APC is denoted by $L^*_4(\beta)$. We also consider a similar scenario with a simple domain-wise path of the same path cost, $\mathcal{A}_a - \mathcal{A}_b - \mathcal{A}_c - \mathcal{A}_d$, whose APC is denoted by $L_4(\beta)$. We observe that the computation of $L^*_4(\beta)$ is almost the same as that in $L_4(\beta)$ except that there are effectively less inter-domain path options. Hence, $L^*_4(\beta) > L_4(\beta)$. Such analysis remains valid in cases where there are more repeated domains in the domain-wise path. We also know from Theorem 12 that $L_k(\beta) > L_{k-1}(\beta)$. Finally, since $k' \geq k \geq 3$, we have $L_k(\beta) < L^*_k(\beta)$, thus completing the proof.

\[ \square \]

A.6 Proof of Theorem 16

Recall that in our two-layer model, gateways are randomly selected. Therefore, let $D_1, D_2, \ldots, D_\beta$ be i.i.d. random variables, denoting
A.10 Efficient Computation of $\mathbb{E}[D^{(\beta)}_k]$

Since $D^{(\beta)}_k$ is defined in a recursive way, it is expensive to compute the exact value of $\mathbb{E}[D^{(\beta)}_k]$. As such, we establish an efficient strategy to estimate $\mathbb{E}[D^{(\beta)}_k]$. Specifically, let $D_1, D_2, \ldots, D_k$ denote i.i.d. random variables following the same distribution as $D$. Then we define random variable $Z^{(k)} := \sum_{i=1}^{k} D_i + k - 1$. For the two-layer network model, when the length of the line network is increased by 1, the number of path options w.r.t. two random nodes at the end-domains grows $\beta$-fold. Therefore, let $Z_1^{(k)}, Z_2^{(k)}, \ldots, Z_{\beta^{k-1}}^{(k)}$ be i.i.d. random variables following the same distribution as $Z^{(k)}$. Define $\overline{D}^{(\beta)}_k := \min(Z_1^{(k)}, Z_2^{(k)}, \ldots, Z_{\beta^{k-1}}^{(k)})$. We then use $\mathbb{E}[\overline{D}^{(\beta)}_k]$ to approximate $\mathbb{E}[D^{(\beta)}_k]$. Since $\overline{D}^{(\beta)}_k$ does not rely on $\overline{D}^{(\beta)}_{k-1}$, $\mathbb{E}[\overline{D}^{(\beta)}_k]$ is easily computable using the method in Appendices A.6–A.7. Such efficient approximation method is highly accurate, as validated in Section 14.