Optimizing SDN Controller to Switch Latency for Controller Placement Problem

Firas Zobary
School of Computer Science and Artificial Intelligence, Wuhan University of Technology, Wuhan, China.
Email: firas_zobary@hotmail.com

Keywords: SDN, clustering, controller placement, latency

Received: March 6, 2024

Software-Defined Networking (SDN) updates network flexibility by decoupling the data plane from control planes, employing a logically centralized yet physically distributed multi-controller architecture. The optimal placement of controllers and their quantity presents a significant challenge known as the Controller Placement Problem (CPP). This study addresses the optimization of average propagation delay between controllers and switches, introducing an enhancement version of well-known K-Means algorithm for network partitioning and controller placement, called an Advanced K-Means algorithm. The proposed algorithm strategically minimizes the average propagation delay by situating controllers in optimal nodes within each sub-network. Evaluation through simulations on the Internet OS3E topology demonstrates the algorithm's efficacy, showcasing a 22%, 11%, 7%, and 3% reduction in average propagation delay compared to DBCP, POCO, CNPA, and HDIDS, respectively. These results establish the proposed algorithm as a competitive solution, emphasizing its capacity to achieve comparable or superior performance in mitigating latency between controllers and switches when compared to existing algorithms.

Povzetek: Ta študija izboljšuje optimizacijo latence med krmilniki SDN in stikali z uvedbo naprednega algoritma K-Means za učinkovito reševanje problema postavitve krmilnikov.

1 Introduction

Nowadays, a massive amount of data leads to cause network traffic and inflexible mobility in future mobile networks [1]. SDN has emerged as a transformative paradigm in the field of networking, updating the way networks are designed, managed, and operated. At the heart of SDN lies the centralized control plane, governed by SDN controllers that apply a new authority over network devices. The strategic placement of these controllers is a critical aspect of SDN architecture, influencing the efficiency, responsiveness, and overall performance of the network. The controller has an important role in managing tasks such as routing through the maintenance of switch forwarding tables, while switches in data plane primarily handle packet forwarding functions. When a new routing path is needed at the data plane layer, switches need to coordinate with their designated controller for guidance on routing decisions. In the expansion of SDN networks, employing multiple controllers becomes necessary to overcome bottlenecks encountered with a single physical controller, as illustrated in Figure 1. A solitary centralized controller struggles to meet high demands of flow processing, particularly when incoming packets lack matches in the existing flow entries at the switch [2].

Additionally, a singular controller faces limitations in scalability, resilience, security, and other aspects [3]. Hence, a multi-controller environment becomes urgent for effectively managing large-scale SDN networks. The placement of SDN controllers plays a crucial role in shaping the dynamics of communication within the network. Controllers acts as the orchestrators, overseeing and directing the flow of data packets, and their strategic positioning has a profound impact on factors such as latency, load balancing, and resource utilization. The complexity of modern networks, characterized by diverse topologies and dynamic traffic patterns, necessitates careful consideration in determining optimal controller locations.

One of the primary challenges in SDN deployment is the Controller Placement Problem (CPP), a complex issue that demands thoughtful analysis and innovative solutions. The objective of CPP is to determine the optimal number and locations of controllers in a network, striking a delicate balance between minimizing latency, ensuring load distribution, and maximizing the utilization of network resources. Addressing the CPP is essential for realizing the full potential of SDN, as improper controller placement can lead to issues such as long response times, and inefficient resource utilization.

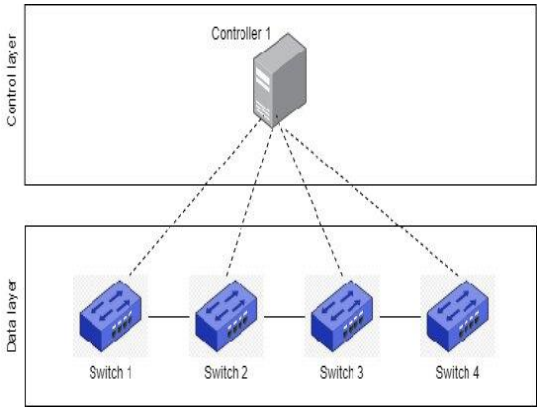

Figure 1: Single controller structure.

To optimize the controller placement problem in terms of controller to switch average propagation latency, a development of K-means algorithm is introduced, called Advanced K-means algorithm, to enhance the network partitioning process and distribute the controllers in optimal places to avoid the high latency between the controllers and the switches they are managing. The proposed algorithm aims strategically position the controllers, minimizing the average propagation delays and enhancing the responsiveness of the network. In terms of performance evaluation, we compare our proposed algorithm to some of existing literatures, benchmarking our work against state-of-the-art algorithms. Through this rigorous evaluation, we seek to contribute valuable insights into the realm of SDN controller placement, offering a novel approach that addresses the complex details of latency optimization in dynamic network environments.

2 Related works

Determining the optimal placement of SDN controller is a significant challenge for network administrator and designer because the location of the controller affects the network's ability to manage the traffic efficiently, especially when the network is large and complex. The first work addressed the controller placement problem was done by Heller et al.[2] in which the authors adopted a mathematical model that is used for facility location problem and used it for CPP. As the controller manages several switches and may has a global view of the whole topology [4]. Therefore, to locate appropriate positions for the controllers, it is necessary to partition the network into several clusters that are well-suited [5]. However, clustering the network will bring some tradeoff between several network metrics such as network delay and load balancing [5], [6]. Mamushiane et al [7] asked a question about how many controllers are needed and where should they go given an SDN topology. They proposed three algorithms to address the controller placement problem, namely, Silhouette Analysis, Gap Statistic, and Partition Around Medoids (PAM), but first they mentioned only the metric of latency. However, their algorithms are accurate but are exhaustive and don't work well in the presence of time constraints. In [8] and [9], the same authors proposed two different algorithms, High Degree with Independent Dominating Set (HDIDS) and Connected Dominating Set (CDS). However, in both works, the controllers are positioned in nodes with maximum connection degree, and they didn't take into consideration the controllers' loads and its effect on latency. A mathematical model in [10] was mapped into CPP and considered only the controller latency, but the authors didn't mention the effect of this model on the controller loads. Several research efforts [11], [12] used clustering algorithms to discuss and solve the controller placement problem. In clustering approach, the whole network is divided into several subnetworks, each subnetwork is controlled by one single controller. Lange et al. [13] proposed a Pareto-based Optimal COntroller placement algorithm (POCO) for CPP based one optimizing multi-objectives simultaneously, but it consumes excessive time to select the optimal controller locations. Wang et al. [14] designed a Clustering-based Network Partition Algorithm (CNPA) with the aim of minimizing delay between controllers and switches. However, it has been noted that CNPA often converges to local optimal solutions, potentially limiting its effectiveness. Many criteria should be addressed during the controller placement process. Network delay and load balancing are the most important parameters that should be taken into consideration during the network partitioning and finding the controller location. However, the propagation delay done in some research is not clearly defined. The authors in [15] defined the reliability of the network by estimating the control path loss percentage using a defined metric. However, they assumed that the switches could connect to the nearest controller but they didn't consider the controller's load as a factor. The well-known method called K-center is used in [2], [16] for controller placement, but the Euclidean distance is used to define the delay which is not suitable and accurate to be used in a real network topology. Moreover, the approaches begin by randomly initializing the centers. In each iteration, they assign switches to new centers until there are no further changes in the clusters. However, this method does not ensure the minimum propagation delay. For instance, in [14], it was noted that the delay may increase with the introduction of a new center, compared to the previous cluster. Density-based clustering techniques are widely used in data mining on various fields. DBSCAN is one of the most popular density-based clustering algorithms, characterized by its ability to discover clusters with different shapes and sizes, and to separate noise and outliers [17]. In [18], the authors proposed a Density Based Controller Placement DBCP for SDN. A virtual network embedding problem for SDN has been presented in [19] and its assignment to real physical resources depends on minimizing the delay between the controller and the switch in the virtual network. In [20], the placement of controllers in IoT based on SDN was examined using a sub-modularity approach that relied on a heuristic method. However, the authors did not

investigate E2E latency and did not demonstrate the results on actual internet topologies. An iterative algorithm was introduced in [21] as a solution to CPP. However, its impracticality for large-scale network topologies arises from its heavy reliance on manually assigned weights. Table 1. collates key findings from referenced research, focusing on their methodologies, strengths, and limitations.

Table 1: Literature works and limitations.

Related Works	Methodology	Strengths	Limitations
[2]	Standard K-means	Minimize latency and consideration of diverse network topologies.	Using Euclidean distance is not accurate for network topology partitioning.
[7]	Silhouette Analysis, Gap Statistic, and Partition Around Medoids (PAM)	These techniques are well-suited for analyzing complex network topologies.	- They didn't mention metrics such as load balancing. - They are exhaustive and don't work well in the presence of time constraints.
[8], [9]	High Degree with Independent Dominating Set (HDIDS) and Connected Dominating Set (CDS)	Select controllers based on their connection degrees and minimizing the average response time between each controller and its forwarding nodes.	Focusing only on latency and node degrees without taking controller load into consideration.
[10]	Hierarchical clustering technique	Decreasing the number of controllers using merging function.	Other metrics such as load balancing are not considered.
[12]	Pareto Integrated Tabu Search (PITS)	Addressing both the controllers' number and locations.	The identification of articulation points to calculate the controllers' number may not adequately account for dynamic network behaviors.
[15]	Greedy-SA algorithm	Defining the reliability of the network by estimating the control path loss percentage.	They assumed that the switches could connect to the nearest controller, but they didn't consider the controller's load as a factor.
[16]	Capacitated K-center	Reduce the number of required controllers, reduce the load of the maximum-load controller.	They focus only on controllers' load without mentioning latency.
[13]	Pareto-based Optimal COntroller placement algorithm (POCO)	Optimizing multi-objectives simultaneously.	it consumes excessive time to select the optimal controller locations.
[14]	Clustering-based Network Partition Algorithm CNPA	Decrease the maximum end-to-end latency between controllers and	It tends to fall into local optimal solutions, potentially

		their associated switches.	limiting its effectiveness.
[18]	Density Based Controller Placement DBCP	It provides the fast response with minimum iterations.	This work is only scalability-aware without focusing on controller load.
[20]	Submodularity optimization approach	Address different aspects of the controller placement problem in a distributed network.	The authors did not investigate E2E latency and did not demonstrate the results on actual Internet topologies.
[21]	A minimum eccentricity-based controller deployment algorithm	Achieve the tradeoff between network response time and the cost of controllers.	Its impracticality for large-scale network topologies arises from its heavy reliance on manually assigned weights.

3 Problem formulation and system model

3.1 Problem formulation

In SDN networks, communication latency includes queuing, transmission, propagation, and processing delays. When considering large network topologies such as WAN, our focus lies primarily on propagation latency due to several reasons. First, in unobstructed networks, queuing latency becomes negligible. Second, with advancements and development in SDN switches, they can achieve a throughput with 100 Gbps [22], [23], [24], so the transmission latency across long distances in SDN-based backbone networks is minimal. Moreover, processing latency, influenced by controller performance, is typically not a concern in large networks such as WAN scenarios, because most controllers operate below their maximum capacity [25]. Therefore, propagation latency emerges as the dominant factor in WAN latency.

3.2 System model

The network topology is illustrated as a graph $G = (V, E)$, where G is an undirected graph, V is the set of nodes composing the graph $V = \{v_1, v_2, \dots, v_N\}$ as N is the number of nodes in the topology and v_i indicates the i^{th} node (network device) so $1 \le i \le N$. E is the set of edges or links between the nodes and E_{ij} is the link connecting v_i with v_j. We define a binary variable α_{ij} indicates if there is a direct connection between v_i and v_j so the two nodes are adjacent nodes. If they are adjacent to each other's (i.e., there is a direct connection between them) then $\alpha_{ij} = 1$, otherwise, $\alpha_{ij} = 0$. Thus, $E = \{E_{ij} \mid \alpha_{ij} = 1,\ 1 \le i, j \le N\}$ will be the set of physical links connecting two switches v_i and v_j.

$$\alpha_{ij} = \begin{cases} 1, & v_i \text{ and } v_j \text{ are adjacent nodes} \\ 0, & \text{otherwise} \end{cases} \quad (1)$$

We denote K as the number of clusters or subnetworks in the topology. That means there is K controllers placed in the network, and we define a set C contains the controllers, so $C = \{C_1, C_2, \dots, C_i \mid i = 1,2,..,K\}$. In the whole network, each subnetwork can be denoted as SDN_i which represents the i^{th} subnetwork of the whole topology and it is composed of a set of switches and a controller C_i that controls these switches. The multiple subnetworks in the whole network should follow the following requirements:

$$SDN_i \cap SDN_j = \phi, \quad \forall\, i, j \in C, i \ne j \quad (2)$$

$$\bigcup_{i=1}^{K} SDN_i = V \quad (3)$$

Equation (2) indicates that each switch in the network can be controlled by one and only one controller. Equation (3) indicates that the sum of all subnetworks should form the whole network so the multiple clusters will cover all the switches in the network.

We define two binary variables x_{ij} and y_{ij} as constrains for the network. The former variable indicates if the switch v_i is controller by the controller C_j or not, while the latter variable indicates that the controller C_j must be placed in a node v_i.

$$x_{ij} = \begin{cases} 1, & \text{if the switch } v_i \text{ is controlld by the controller } C_j \\ 0, & \text{Otherwise} \end{cases} \quad (4)$$

$$y_{ij} = \begin{cases} 1, & \text{if the controller } C_j \text{ is placed in the node } v_i \\ 0, & \text{Otherwise} \end{cases} \quad (5)$$

The shortest path distance between each switch v_i and its controller C_j is defined as d_{ij}, and as the main goal in CPP is to decrease the average propagation delay between controllers and switches, this decreasing can be formulated as:

$$min \sum_{i \in V} \sum_{j \in C} d_{ij}\, x_{ij} \quad (6)$$

All the previous variables and indicators are subject to the following constraints:

$$x_{ij} \in 0,1,\ \forall i \in V,\ \forall j \in C \quad (7)$$
$$y_{ij} \in 0,1,\ \forall i \in V,\ \forall j \in C \quad (8)$$
$$\sum_{j \in C} x_{ij} = 1,\ \forall i \in V \quad (9)$$
$$\sum_{j \in C} y_{ij} \leq 1,\ \forall i \in V \quad (10)$$
$$\sum_{i \in V} \sum_{j \in C} y_{ij} = K,\ \ \forall i \in V,\ \forall j \in C \quad (11)$$
$$x_{ij} \leq y_{ij},\ \forall i \in V,\ \forall j \in C \quad (12)$$

Constraint (7) and constraint (8) indicate the variables x_{ij} and y_{ij} as binary variables. Constraint (9) ensures that each switch is controlled by one and only one controller. Constraint (10) makes sure that each controller can only be placed at a switch location and no more than one controller can be placed at the same node at the same time. Constraint (11) ensure that there are K controllers deployed in the network, and constraint (12) makes sure that the switch is controlled by a deployed controller (i.e., the switch can't be controlled by a controller if the controller is not placed in a node).

The controller should be an adjacent to more than one switch, and not separated from switches. In other words, the controller should be placed in a location with more than one connection. If the controller is located in a node with only one connection, it can be a single point of failure. For this reason, we will use the node degree parameter. The node degree D_i by definition is the number of connections that it has to other nodes in the network. So, placing the controller at a node with a degree of 1 is not a practical solution and it may affect the behavior of the network in the future and can cause problems in term of reliability. For that reason, we firstly define a node average degree ρ s follows:

$$\rho = round\left(\frac{\sum_{i=1}^{N} D_i}{N}\right) \quad (13)$$

then, during the controller's selection process, it should be assured to place the controller in a node that follow the condition:

$$D_i \geq \rho,\ \forall i \in C \quad (14)$$

The average propagation delay between controllers and switches is defined as follows:

$$L(C_j) = \sum_{i \in V} d_{ij}\, x_{ij} \quad (15)$$

$$L_{avg}(K) = \frac{\sum_{j \in C} L(C_j)}{N-K} \quad (16)$$

Equation (15) calculates the propagation latency for each cluster, and then using equation (16) the average propagation latency for the whole network is calculated.

On that basis, the description of symbols is listed and summarized in Table 2.

Table 2: List of Notations

Notation	Description
α_{ij}	The binary variable that indicates if there is a direct connection between v_i and v_j
SDN_i	The subnetwork which contains switches controlled by C_i
d_{ij}	The shortest path distance between each switch v_i and its controller C_j
x_{ij}	The binary variable indicates if the switch v_i is controller by the controller C_j or not
y_{ij}	The binary variable indicates that the controller C_j must be placed in a node v_i
D_i	The node degree that reflects the number of connections that is has to other nodes in the network
ρ	The node average degree
$L(C_j)$	The average propagation latency in the cluster j
L_{avg}	The average propagation latency in the whole network
$B(C_j)$	The controller load that reflects the number of switches the controller manages
β	The maximum difference between the loads of each two controllers
BI	The balance index for the network

4 Methodology

In partitioning tasks, K-means is frequently used and known to be effective and fast [26], [27]. In our research, we developed an enhanced version of K-means called Advanced K-means, specifically for network partitioning. Our focus was on reducing the average delay between controllers and switches. We refer to the initial controller placements as 'centers', and after running the algorithm, they become 'centroids'.

First, let's delve into the standard K-means algorithm used for clustering networks. This approach contains the following key steps:

1. K points from the dataset are selected randomly as the centers for K clusters, ensuring each cluster has one and only one center.
2. Each data point is assigned to its nearest cluster based on Euclidean distance to the cluster center.
3. The centers of each cluster are updated.

4. Repeating step 2 and 3 until there are no changes in cluster centers.

However, when applied to network topology partitioning, standard K-means faces several limitations. Firstly, choosing random centers does not guarantee minimum propagation latency between nodes and their centroids. Secondly, the algorithm doesn't ensure that updated centroids will be chosen from the actual nodes in the topology, which is crucial for establishing a real connection between the centroid and nodes in the cluster, as K-means selects the mean between two nodes. Thirdly, utilizing Euclidean distance does not guarantee the physical existence of the link between the centroid and the node.

To address challenges, we introduced an advanced version of K-means, detailed in Algorithm 1, aimed to overcome the limitations of the standard K-means approach. The key steps in our algorithm involve initializing the centers, distributing nodes to clusters, and updating the centroids. Unlike standard K-means, our algorithm avoids randomly initializing of the centers, opting for a more effective method proposed in previous study [28]. This method not only reduce computational complexity but also ensure better convergence to a local minimum, making the process more efficient.

Algorithm 1 Selecting the First Initial Center

Input: $G = (V, E), N$
1: **for** *each* $i \in V$ **do**
2:$D_i = Calculate\ Switch_i\ Degree$;
3: **end for**;
4:$\rho = Calculate\ the\ average\ node\ degree\ (N, D_i)$;
5: *first cluster = Calculate the first initial cluster center* (D_i, ρ, G);
Output *first cluster centroid*

Algorithm 2 Network Partitioning

Input: $G = (V, E), N, K, \rho, \beta, first\ cluster\ center$
1: $C = \phi$;
2: *select first initial center in Algorithm 1 as* C_1;
3: *compute distance* d_{ij} *matrix for the nodes in the graph;*
4: $j = 2$;
5: **while** $j \leq K$ **do**
6: **for** each $i \in V$ **do**
$C_j =$ *select the next initial cluster center* (i, C_{j-1}, G);
7: **end for;**
8: *distribute switches between clusters;*
9: *calculate the sum of the shortest path distances to every node in* $cluster_i$ *and update the cluster center to be the node that has the minimum sum;*
10: *repeat 8 and 9 until the centers are not updated anymore;*
11: $C += C_j$;
12: $j += 1$;
13: **end while**
14: *Calculate* $SDN_j, \quad \forall j \in C$

The Advanced K-means algorithm involves several key steps. Firstly, the degree of each switch in the network is calculated, followed by determining the average node degree using Equation (13). Subsequently, the switch with the highest degree is selected, and the condition outlined in (14) is applied, as detailed in Algorithm 1. In cases where multiple nodes share the same degree, the node with the minimum sum of shortest path distances to all other nodes is chosen. Then, the remaining clusters are formed, and nodes are allocated based on their proximity to the cluster centers, as specified in Algorithm 2.

More specifically, nodes with a high degree centrality are pivotal in network communication and connectivity. By selecting initial centers based on the maximum node degree, we prioritize nodes that are highly connected within the network. These nodes are likely to have significant influence and control over neighboring nodes, making them strategic choices for controller placement. Placing controllers at high-degree nodes can help optimize network performance by ensuring efficient communication and management of network traffic. In addition, the sum of shortest path distances from a node to all other nodes in the network provides insights into its centrality and importance in network communication. Nodes with lower sums of shortest path distances are more centrally located within the network and have shorter paths to reach other nodes. By selecting initial centers based on nodes with the minimum sum of shortest path distances, we prioritize locations that are centrally located and well-connected to other nodes.

Incorporating these criteria into the selection of initial centers ensures that controllers are strategically positioned to optimize network performance and minimize propagation delay. By leveraging nodes with high degree centrality and minimal distances to other nodes, our algorithm can effectively distribute controllers across the network, enhancing the overall resilience, efficiency, and responsiveness of SDN deployments.

Algorithm 2's clustering process optimizes average propagation latency. Initially, we select the switch farthest from the previously chosen center as the next initial center. Nodes are then assigned to centers based on their minimum distance to the centroids. Within each cluster, we compute the sum of distances for every node, including the center, and choose the node with the smallest sum as the new centroid. Nodes are then redistributed according to the updated list of centroids, repeating this process until the list stabilizes under the condition outlined in (14).

We choose a real network topology from Internet Topology Zoo [29] which is a store of network data created from the information that are published publicly by network operators and it is the most accurate large-scale collection of network topologies available. The topology we choose is Internet 2 OS3E [30]. We chose to utilize the Internet2 OS3E topology for several reasons. Firstly, the Internet2 OS3E topology mirrors real-world network infrastructures utilized by research and education institutions. Its utilization ensures that our study reflects practical network scenarios. Secondly, the complexity of the Internet2 OS3E topology enables us to assess our

algorithm's scalability and effectiveness in managing large-scale networks with varied communication requirements. Additionally, the availability of publicly accessible data associated with the Internet2 OS3E topology promotes in our research endeavors. Moreover, the Internet2 OS3E topology enjoys widespread recognition and adoption as a benchmark in the networking research community. Leveraging this standard topology facilitates direct comparisons with other studies and algorithms, streamlining benchmarking efforts and enhancing the evaluation of our proposed approach. In summary, the selection of the Internet2 OS3E topology underscores our commitment to conducting rigorous, credible, and relevant research in the field of network optimization and management. OS3E topology has 34 nodes and 41 edges. We will set the optimal controller number as 5 for this topology which will be explained in Subsection 4.1.

The nodes average degree is calculated according to Equation (13) and the nodes' degrees are listed in Table 3. below. As a result, there are 4 nodes has the same maximum degree equals to 4 which are V_{11}, V_{22}, V_{28}, and V_{33}. The switch V_{11} will be selected as the first initial center because it has the minimum sum of shortest path distance to all other nodes in the network 262.88 compared to 345, 401.2, and 525.5 for the other 3 nodes respectively.

Table 3: Nodes degrees calculation for OS3E

Nodes	Switch name	D_i	Nodes	Switch name	D_i
1	Boston	2	18	Kansas	3
2	New York	2	19	Memphis	2
3	Philadelphia	2	20	Jackson	2
4	Washington	3	21	Baton-Rouge	2
5	Ashburn	2	22	Houston	4
6	Pittsburg	2	23	Dallas	2
7	Clevland	3	24	Denver	3
8	Buffalo	2	25	Albuquerque	2
9	Raleigh	2	26	El paso	3
10	Indianapolis	2	27	Tucson	2
11	Chicago	4	28	Salt Lake	4
12	Louisville	2	29	Los Angeles	3
13	Nashville	3	30	Sunnyvale	3
14	Atalanta	3	31	Missoula	2
15	Jacksonville	3	32	Portland	2
16	Miami	1	33	Seattle	4
17	Minneapolis	2	34	Vancouver	1

4.1 Evaluation of K

Choosing the optimal number of controllers for the network is not well solved problem, and the question "How many controllers the network needs?" is still under discussion since the controller placement problem was first introduced in [2]. Some studies suggest to use the least number of controllers depending on their limitations, such as controller load capacity, network latency and standards of reliability. In general, it's hard to define the optimal number of controllers as the controller placement problem is defined as a multi-objective combinatorial optimization problem (MOCO) [13]. Speaking generally, placing more controllers in the network will definitely improve the performance and data processing, but at the same time it will cost more and may affect another network criteria. However, placing a minimum number of controllers as an efficient number and improve the performance at the same time is the best solution for the controller placement problem [2]. The best answer for the question "How many controllers should we deploy?" depends on the network administrator goals. The average propagation latency is $L_{avg}(1) = 7.7$, and to reduce this value to the half, 4-5 controllers are needed as $L_{avg}(4) = 4.1$ and $L_{avg}(5) = 3$. In Figure 2., concerning OS3E topology, the correlation between the number of controllers deployed in the topology and the cost-benefit ratio, characterized as $(L_{avg}(1)/L_{avg}(K))/K$, is illustrated. A cost-benefit ratio approaching 1.0 indicates a better performance. It's evident that there is an inflection point observed when the number of controllers reaches 5. More precisely, as the number of controllers transitions from 4 to 5, there is a notable 11% increase in the associated cost-benefit ratio. However, with a further increase in the number of controllers, the cost-benefit ratio subsequently decreases. In summary, the overall trend of the cost-benefit ratio shows a decreasing pattern with minor fluctuations, which are influenced by the complexities of the network topology. This implies that deploying 5 controllers achieves a higher cost/benefit ratio for reducing average propagation latency compared to deploying 4 controllers. However, the general line of the cost-benefit ratio for average propagation latency gradually diminishes as the number of placed controllers increases. Notably, when $K > 1$, each indicator remains below 1, and an increase in the number of controllers reveals a diminishing effect on the cost-benefit ratio shown especially when the controller number increases from 4 to 6.

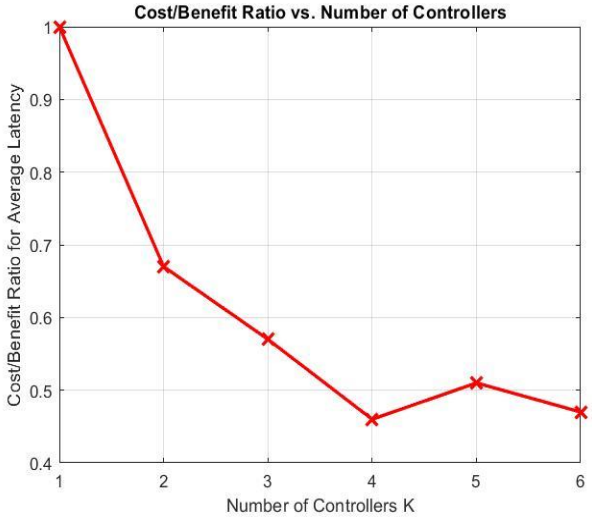

Figure 2: Relationship between cost/benefit ratio and the number of applied controllers in OS3E.

In [2], the experimental findings affirm that with the rise in the number of controllers, there is a discernible diminishing trend in the cost-benefit. Within the scope of this work, we adhere to the selection of the smallest number of clusters based on distinct network structures. This choice aligns not only with the intricacies of the network topology but also guarantees that the chosen

number of controllers attains a superior cost-benefit. As a conclusion, even with evaluation of K, choosing an appropriate controller number is a multi-object problem. While some network administrators prefer to deploy more controllers to achieve less latency, others can accept a high latency to avoid the increasing cost of deploying large number of controllers.

4.2 Latency-aware

The strategic placement of controllers in a software-defined network (SDN) plays a critical role in determining network performance, specifically regarding latency. Latency, the delay in data transmission between source and destination, is a critical metric influencing to overall efficiency and responsiveness of network operations. The location and distribution of controllers within an SDN infrastructure significantly impact how efficiently routing decisions are made and how swiftly these decisions are communicated to network devices.

Essentially, the closeness of controllers to network components like switches and routers significantly impacts the latency encountered by data packets during their journey across the network. A well-designed strategy for placing controllers aims to minimize latency by strategically locating them to efficiently manage routing requests. Conversely, poor controller placement can cause unnecessary delays, resulting in higher latency and reduced network performance. Understanding the complex interplay between controller placement and latency is crucial for producing SDN architectures that meet the needs of modern, dynamic networks. Efficient controller placement not only decreases latency but also improves the network's responsiveness, scalability, and ability to adapt to shifting traffic patterns.

In our investigation, we systematically evaluate the impact of controller placement strategies on network latency by comparing the average latency under two different scenarios: the first involves randomly selecting controller centers, and the second entails the application of Advanced K-means algorithm. The comparison is conducted in the context of a ratio, specifically the average latency divided by the optimal latency, as the number of controllers (K) varies from 1 to 6. The utilization of random centers serves as a baseline, allowing us to detect the efficacy of the Advanced K-means algorithm in optimizing controller placement. In the methodology of this experiment, and for accuracy, we employ a randomized approach for selecting controller centers, repeating this process 10 times for each value of K. subsequently, we calculate the average latency across these 10 randomized selections. This randomized average latency is then utilized as a comparative metric against the optimal average latency that is determined by considering the Advanced K-means algorithm. The resulting ratio offers a comprehensive measure of how well the randomized controller placement performs relative to an optimal configuration. Figure 3. shows that if we simply choose random centers to place the controllers, the average latency is between 1.3x and 1.5x larger than the optimal solution when $K \leq 3$, while it rises to be between 1.5x and 1.68x larger than the optimal solution when $K > 3$. The increase in latency underscores the need for more advanced placement algorithms beyond random selection. This calls for the exploration of advanced techniques to improve the efficiency of deploying controllers in SDN.

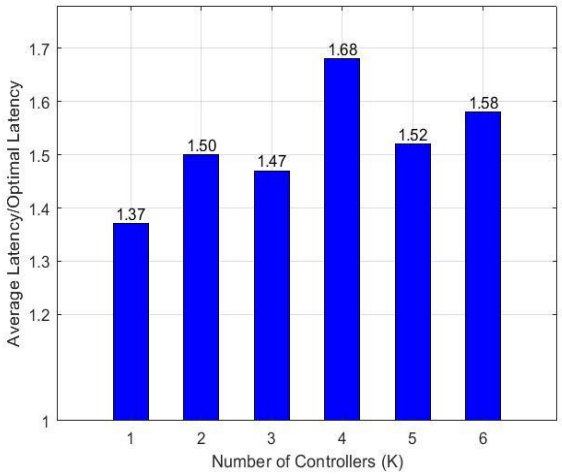

Figure 3: Ratio of random centers selection vs. optimal.

5 Performance evaluation

In order to substantiate the efficacy of our proposed algorithm, we conduct a performance evaluation of the Advanced K-means. First, we compare our proposed approach with four solutions: POCO, CNPA, HDIDS, and DBCP using Internet OS3E topology in terms average propagation delay between controllers and their switches. Moreover, the maximum number of controller load (i.e., the maximum number of switches the controller manages) is evaluated compared to DPCB, POCO and CNPA. The simulation is conducted on Windows 11 PC with Intel Core i5-12500H 2.50 GHz processor and 16.0 GB RAM. The simulation platform for this work is MATLAB R2018a.

5.1 Performance in average propagation latency

To assess the efficacy of the Advanced K-means algorithm in reducing average propagation latency, we implement our proposed algorithm on the Internet OS3E topology. Utilizing the optimal number of controllers, set at 5, we conduct a comparative analysis of average propagation latency between controllers and their associated switches. This comparison involves Advanced K-means, as well as benchmarking against POCO, CNPA, HDIDS and DBCP algorithms. Figure 4. shows the comparison results of the average propagation latency of each algorithm. In evaluating the performance of our proposed algorithm in the context of SDN controller placement, we observed an average propagation latency of 3.00 ms, surpassing other prominent algorithms in the field. POCO exhibited a latency of 3.37 ms, with our algorithm achieving a 11% reduction. Compared to CNPA, HDIDS and DBCP, our algorithm achieved reduces the average propagation latency by 7, 3 and 22%

respectively. This is because Advanced K-means uses the minimum shortest path distance during the updating controllers' list in each sub-network. While DBCP and CNPA deploy a controller in each cluster in the network, they neglect the consideration of the node degree during network partitioning and focus more on centers' density. Additionally, POCO exhibits suboptimal performance, primarily attributed to an imbalance condition where the latency between controllers is excessively emphasized, and unlike POCO, Advanced K-means doesn't distribute the controllers near to each other's to decrease the latency among controllers which may increase the latency between the switches and their controllers. The closest algorithm to ours in terms of average latency is HDIDS, because they both use the degree of the node as a parameter for being selected as a center but HDIDS doesn't take into consideration the distance between that node and the previous center. As a result, Advanced K-means outperform POCO, HDIDS, CNPA and DBCP on reducing the average propagation delay between the nodes and centers.

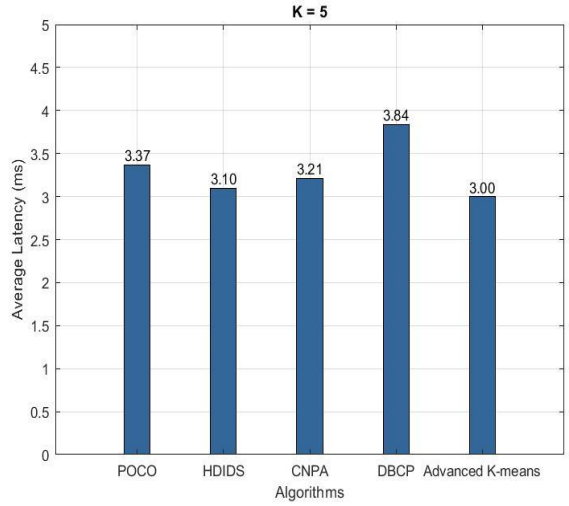

Figure 4: Comparison of average latency on OS3E topology with other algorithms.

5.2 Controller load evaluation

The placement of controllers in inappropriate locations is recognized to result in the occurrence of controller overload, characterized by extended response times and insufficient processing capacity. Additionally, such improper placement leads to the under-utilization of network resources. To evaluate the controller load among different algorithms we compare the maximum number of nodes connected to a single controller in each strategy that are DBCP, POCO, CNPA and Advances K-means with an increasing number of controllers.

The results in Figure 5. illustrates the maximum controller load for different algorithms as the number of controllers increases from 2 to 6. It is evident that, at $K = 2$, all algorithms exhibit relatively high maximum controller loads, with DBCP, CNPA and POCO reaching a load of 22, while Advances K-means has a slightly higher load of 25. As the number of controllers increases, our algorithm exhibits enhanced performance, manifesting in a diminishing gap compared to other algorithms. Notably, it surpasses POCO and DBCP, achieving parity with CNPA when the controller number reaches 5. Furthermore, our algorithm demonstrates a reduced load in comparison to DBCP, while maintaining load equivalence with POCO and CNPA. These findings underscore the achieving competitive performance with other algorithms even if it focuses on reducing the average latency between controller and switches as it achieves comparable or superior results compared to existing algorithms. It can be explained as our algorithm selects the centroids far from each other, so the load can be distributed among the controllers as balanced as possible.

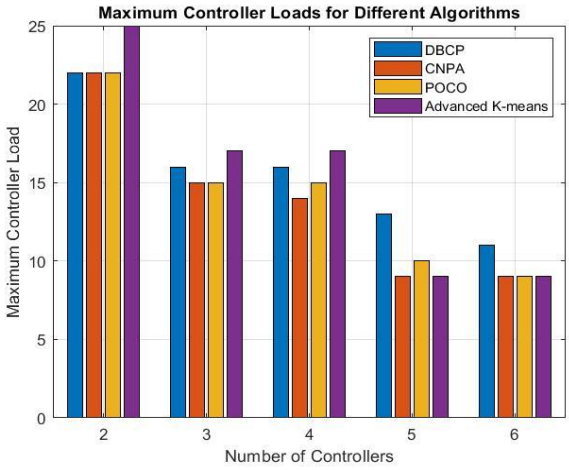

Figure 5: Maximum controller load comparison with different approaches in OS3E.

6 Discussion

In this section, we have conducted an in-depth analysis to clarify the key differences and contributions of our proposed algorithm, and we also discuss some potential limitations that may affect our algorithms' applicability.

6.1 Comparative analysis

In this subsection, we analyze why some disparities exist between our results and those of state-of-the-art (SOTA) solutions. One of the key factors contributing to the disparities between our results and those of SOTA solutions is the inherent differences in the methodologies and algorithms employed. While our study focuses on the Advanced K-Means algorithm for controller placement optimization in SDN networks, SOTA solutions may utilize alternative algorithms, heuristics, or optimization techniques. These differences in approach can lead to variations in performance metrics, such as average propagation latency, controller load distribution, and scalability.

Additionally, disparities may arise due to variations in experimental setups, including network topologies, traffic patterns, and simulation parameters. Our study utilizes the Internet2 OS3E topology for evaluation, which may differ from the topologies used in SOTA solutions. Variations in network characteristics and configurations can influence

the performance of algorithms and impact the comparability of results.

Furthermore, disparities may stem from differences in the evaluation criteria and metrics employed across studies. While our study focuses on optimizing average propagation latency and controller load distribution, SOTA solutions may prioritize other performance metrics or objectives, such as fault tolerance, reliability, or energy efficiency. Variations in evaluation criteria can lead to divergent results and interpretations, highlighting the importance of considering a comprehensive set of metrics when assessing algorithm performance.

Unlike previous approaches that primarily focus on clustering based on factors such as density or random initialization, the key aspect of our algorithm is its utilization of node degree and shortest path distance metrics for selecting optimal controller locations and partitioning the network into sub-networks. By prioritizing nodes with high degrees and minimizing the sum of shortest path distances to all other nodes, our algorithm effectively minimizes propagation latency and ensures efficient resource utilization. In addition, this controller placement strategy, which relies on node degree and shortest path metrics, facilitates the reasonable distribution of workload among controllers. By leveraging these metrics and ensuring that initial centroids are positioned sufficiently apart from each other, our algorithm aims to identify central and well-connected nodes in the network as well as enhances the effectiveness of traffic load balancing, thereby mitigating congestion and bottlenecks more effectively. Placing controllers at these strategic locations can rise the overall efficiency and effectiveness of network management, as controllers positioned closer to high-degree nodes can do a greater control over network traffic and communication. Thus, we acknowledge that certain disparities may arise between our results and those of existing solutions due to various factors, including differences in algorithmic approaches. Our algorithm incorporates several enhancements, such as considering node degree and shortest path distance during network partitioning, which may contribute to its superior performance compared to previous approaches. This unique approach to controller placement may have influenced the observed results compared to related works that employ different placement strategies or algorithms.

6.2 Potential Limitations

While the Advanced K-Means algorithm demonstrates promising performance in optimizing controller placement in SDN networks, we acknowledge that scalability could be a potential limitation, particularly in large-scale network deployments. As with any clustering algorithm, the computational complexity of Advanced K-Means may increase with the size and complexity of the network topology. In scenarios where the number of switches and controllers is large, the algorithm may encounter challenges in terms of computational resources and execution time. To mitigate scalability issues, future research could explore optimization techniques or parallelization strategies to enhance the efficiency of the algorithm and enable its applicability to larger network deployments. Another consideration is the adaptability of the Advanced K-Means algorithm to diverse network topologies. While our evaluation on real network topologies from the Internet Topology Zoo demonstrates promising results, it is essential to recognize that the algorithm's performance may vary depending on the specific characteristics of the network. Certain network topologies, such as highly dense or sparse networks, may pose challenges for the algorithm in terms of achieving optimal controller placement and minimizing propagation latency. It is important to consider any underlying assumptions that may limit the applicability of the Advanced K-Means algorithm. For instance, the algorithm assumes that network topology information, including node degree and shortest path distances, is accurately available for analysis and computation. Additionally, the algorithm may implicitly assume homogeneous network conditions and uniform traffic patterns, which may not always hold true in real-world deployments. In conclusion, while the Advanced K-Means algorithm presents a promising approach to controller placement optimization in SDN networks, we acknowledge the need for further exploration of its limitations and considerations for scalability, adaptability, and underlying assumptions. By addressing these concerns, we aim to enhance the robustness and applicability of the algorithm in different real-world network deployments.

7 Conclusion and future works

In this work, we discuss the SDN controller placement problem by optimizing the average propagation latency and the number of controllers required for the network. An advanced K-means algorithm is proposed and applied to reduce the average delay between the controllers and switches. Firstly, the number of controllers required for the network is discussed and calculated in terms of the cost-benefit ratio. Secondly, the first initial controller is selected based on the node degree to ensure not to choose a controller with a degree of one. After selecting the first initial center, the next center is chosen based on the largest distance to the previous selected center to ensure the well-distribution of the switches to the selected controllers. Then, the switches are distributed among the controllers based on the minimum shortest path distance between the switch and each controller. The sum of the shortest path distances to every node in the cluster is calculated and the node with a minimum sum will be chosen as a new center for the cluster and the cluster center is updated. We repeat this process until the required number of controllers is placed. For performance evaluation, a simulation is conducted using a topology of Internet OS3E from Topology Zoo as it is a well-known and the most used topology for controller placement problem. The simulation results verify that our algorithm reduce the average propagation delay between the controllers and their assigned switches by 22, 11, 7 and 3% compared to DBCP, POCO, CNPA and HDIDS respectively. For the load balancing, we compare the maximum controller load for different algorithms as the number of controllers

increases from 2 to 6. For small number of controllers, our algorithm has a slightly higher load of 25 switches, and as the number of controllers increases, it exhibits enhanced performance, manifesting in a diminishing gap compared to other algorithms. Notably, it surpasses POCO and DBCP, achieving parity with CNPA when the controller number reaches 5. These findings underscore the achieving competitive performance with other algorithms even if it focuses on reducing the average latency between controller and switches as it achieves comparable or superior results compared to existing algorithms.

In future works, other metrics such as resilience, reliability and energy saving can be addressed so the controller placement decision can be more accurate. Firstly, regarding resilience, future research could focus on developing algorithms and strategies to improve the robustness of SDN networks against failures and attacks. This may involve designing fault-tolerant controller placement algorithms that can dynamically adapt to network changes and disruptions while ensuring continuous service availability. Secondly, in terms of reliability, there is a need to investigate methods for enhancing the reliability of controller communication and coordination in distributed SDN environments. This could involve exploring redundant communication paths, load balancing techniques, and fault detection mechanisms to minimize the impact of controller failures on network operations. Lastly, with respect to energy saving, future research efforts could explore energy-efficient controller placement strategies and network management techniques. This may include optimizing the allocation of resources, reducing idle energy consumption, and leveraging energy-aware scheduling algorithms to minimize energy usage while maintaining network performance. By addressing these future directions, we aim to contribute to the development of more resilient, reliable, and energy efficient SDN networks. These efforts not only advance the state-of-the-art in network management but also have the potential to yield significant benefits in terms of cost savings, environmental sustainability, and overall network performance.

References

[1] A. Sureshkumar and D. Surendran, "Novel group mobility model for software defined future mobile networks," *Informatica*, vol. 46, no. 4, 2022.

[2] B. Heller, R. Sherwood, and N. McKeown, "The controller placement problem," *ACM SIGCOMM Computer Communication Review*, vol. 42, no. 4, pp. 473–478, 2012.

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

[4] M. Tanha, D. Sajjadi, R. Ruby, and J. Pan, "Capacity-aware and delay-guaranteed resilient controller placement for software-defined WANs," *IEEE Transactions on Network and Service Management*, vol. 15, no. 3, pp. 991–1005, 2018.

[5] A. K. Singh and S. Srivastava, "A survey and classification of controller placement problem in SDN," *International Journal of Network Management*, vol. 28, no. 3, p. e2018, 2018.

[6] Y.-W. Ma, J.-L. Chen, Y.-H. Tsai, K.-H. Cheng, and W.-C. Hung, "Load-balancing multiple controllers mechanism for software-defined networking," *Wirel Pers Commun*, vol. 94, pp. 3549–3574, 2017.

[7] L. Mamushiane, J. Mwangama, and A. A. Lysko, "Given a SDN Topology, How Many Controllers are Needed and Where Should They Go?," in *2018 IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN)*, IEEE, 2018, pp. 1–6.

[8] A. Alowa and T. Fevens, "Towards minimum inter-controller delay time in software defined networking," *Procedia Comput Sci*, vol. 175, pp. 395–402, 2020.

[9] A. Alowa and T. Fevens, "Combined degree-based with independent dominating set approach for controller placement problem in software defined networks," in *2019 22nd Conference on Innovation in Clouds, Internet and Networks and Workshops (ICIN)*, IEEE, 2019, pp. 269–276.

[10] J.-M. Sanner, Y. Hadjadj-Aoufi, M. Ouzzif, and G. Rubino, "Hierarchical clustering for an efficient controllers' placement in software defined networks," in *2016 Global Information Infrastructure and Networking Symposium (GIIS)*, IEEE, 2016, pp. 1–7.

[11] A. Shirmarz and A. Ghaffari, "Taxonomy of controller placement problem (CPP) optimization in Software Defined Network (SDN): a survey," *J Ambient Intell Humaniz Comput*, vol. 12, no. 12, pp. 10473–10498, 2021.

[12] G. Ramya and R. Manoharan, "Enhanced optimal placements of multi-controllers in SDN," *J Ambient Intell Humaniz Comput*, vol. 12, pp. 8187–8204, 2021.

[13] S. Lange *et al.*, "Heuristic approaches to the controller placement problem in large scale SDN networks," *IEEE Transactions on Network and Service Management*, vol. 12, no. 1, pp. 4–17, 2015.

[14] G. Wang, Y. Zhao, J. Huang, and Y. Wu, "An effective approach to controller placement in software defined wide area networks," *IEEE Transactions on Network and Service Management*, vol. 15, no. 1, pp. 344–355, 2017.

[15] K. S. Sahoo, B. Sahoo, R. Dash, and N. Jena, "Optimal controller selection in software defined network using a greedy-SA algorithm," in *2016 3rd International Conference on Computing for Sustainable Global Development (INDIACom)*, IEEE, 2016, pp. 2342–2346.

[16] G. Yao, J. Bi, Y. Li, and L. Guo, "On the capacitated controller placement problem in software defined networks," *IEEE*

communications letters, vol. 18, no. 8, pp. 1339–1342, 2014.

[17] A. Bouchemal and M. T. Kimour, "Multi-Density Datasets Clustering Using K-Nearest Neighbors and Chebyshev's Inequality," *Informatica*, vol. 47, no. 8, 2023. https://doi.org/10.31449/inf.v47i8.4719

[18] J. Liao, H. Sun, J. Wang, Q. Qi, K. Li, and T. Li, "Density cluster based approach for controller placement problem in large-scale software defined networkings," *Computer Networks*, vol. 112, pp. 24–35, 2017.

[19] A. A. Nasiri and F. Derakhshan, "Assignment of virtual networks to substrate network for software defined networks," *International Journal of Cloud Applications and Computing (IJCAC)*, vol. 8, no. 4, pp. 29–48, 2018.

[20] A. K. Tran, M. J. Piran, and C. Pham, "SDN controller placement in IoT networks: An optimized submodularity-based approach," *Sensors*, vol. 19, no. 24, p. 5474, 2019.

[21] R. Chai, X. Yang, C. Du, and Q. Chen, "Network cost optimization-based capacitated controller deployment for SDN," *Computer Networks*, vol. 197, p. 108326, 2021.

[22] "NoviSwitch - SDN Programmable Network Switch - OpenFlow Switch | NoviFlow." Accessed: Nov. 08, 2023. [Online]. Available: https://noviflow.com/noviswitch/

[23] "Home - Corsa Security." Accessed: Nov. 08, 2023. [Online]. Available: https://www.corsa.com/

[24] "Resource & Documentation Center." Accessed: Nov. 08, 2023. [Online]. Available: https://www.intel.com/content/www/us/en/resources-documentation/developer.html

[25] A. Jalili, M. Keshtgari, and R. Akbari, "Optimal controller placement in large scale software defined networks based on modified NSGA-II," *Applied Intelligence*, vol. 48, no. 9, pp. 2809–2823, Sep. 2018, doi: 10.1007/S10489-017-1119-5/TABLES/2.

[26] C. Cristian López *et al.*, "Parallelization of the Algorithm K-means Applied in Image Segmentation," *Article in International Journal of Computer Applications*, vol. 88, no. 17, 2014, doi: 10.5120/15441-4051.

[27] H. Al-Mohair, J. Saleh, S. S.-A. S. Computing, and undefined 2015, "Hybrid human skin detection using neural network and k-means clustering technique," *Elsevier*, Accessed: Nov. 09, 2023. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1568494615002732

[28] I. Katsavounidis, C. C. J. Kuo, and Z. Zhang, "A New Initialization Technique for Generalized Lloyd Iteration," *IEEE Signal Process Lett*, vol. 1, no. 10, pp. 144–146, 1994, doi: 10.1109/97.329844.

[29] "Internet Topology Zoo." Accessed: Nov. 10, 2023. [Online]. Available: https://www.topology-zoo.org/

[30] "Internet 2 OS3E." Accessed: Aug. 12, 2023. [Online]. Available: https://internet2.edu/