Distributed SDN Deployment in Backbone Networks for Low-Delay and High-Reliability Applications

Mohammed J.F. Alenazi
College of Computer and Information Sciences
Department of Computer Engineering
King Saud University, Riyadh, Saudi Arabia

***Abstract*—Internet applications, such as video streaming, critical-mission, and health applications, require real-time or near real-time data delivery. In this context, Software Defined Networking (SDN) has been introduced to simplify the network management providing a more dynamic and flexible configuration based on centralizing the network intelligence. One of the main challenges in SDN applications consists of selecting the number of deployed SDN controllers, and their locations, towards improving the network performance in terms of low delay and high reliability. Traditional k-center and k-median methods have been fairly successful in reducing propagation latency, but ignore other important network aspects such as reliability. This paper proposes a new approach for controller placement that addresses both network reliability and reducing network delay. The proposed heuristic algorithm focuses on four different robustness functions, viz, algebraic connectivity (AC), network criticality (NC), load centrality (LC), and communicability, and has been applied in four different real-world physical networks, for performance evaluation based on degree-, closeness-, and betweenness-based centrality-based attacks. Experimental results show that the proposed controller selection algorithms based on AC, NC, LC, and communicability, achieve a high network resilience and low C2C delays, outperforming the latest, widely-used baseline methods, such as k-median and k-center ones, especially when using the NC method.**

Keywords*—Software Defined Networking (SDN); Controller Placement Problem (CPP); physical network; graph robustness metrics; reliability; resilience

I. Introduction and Motivation

In recent years, the rapid growth in cloud computing coupled with the high demand for massive-scale data centers, have made it crucial to provide efficient network management and resource utilization towards ensuring an optimal system performance [1]. In this dynamic context, network design requirements are subject to change, making it necessary to periodically re-configure network devices, such as switches and routers. Traditionally, this re-configuration used to be performed manually. Nevertheless, as the number of deployed devices increases, re-configuration becomes harder; this situation is particularly noticeable in the case of distributed backbone networks. In recent years, software defined networking (SDN) technologies have been introduced to address such network management and scalability issues [2]. SDN aims at simplifying network management enabling a more dynamic and flexible configuration by centralizing network intelligence. Specifically, SDN decouples the data plane, *i.e.*, the forwarding process of network packets, from the control plane, *i.e.*, the routing process, thereby improving the efficiency and programmability of the former, while centralizing the latter in a single device called an SDN controller that is responsible for defining flow routes at each SDN switch [3]. Fig. 1 shows a typical SDN architecture.

The introduction of SDN technologies signifies a paradigm shift in the field of network infrastructure [4]. In particular, by allowing logical centralization of feedback control, decisions are based on a global view of the network, easing network maintainability and enabling consistency in network policies. In this way, SDN networks provide a favorable environment for the development of innovative applications, and is an important research area for both academia and industry. SDN has been used to improve network performance [5], network resilience [6], [7], [8], and energy consumption [9], [10].

A typical SDN network can operate with one controller [11], which remotely configures SDN switches and routers. However, to avoid single point failure, more than one controller is needed to ensure a higher degree of resilience in the event of network failure that disrupts controller connectivity [12]. In addition, adding more controllers lowers the control message latency between SDN switches, especially for backbone networks where propagation delay can significantly increase the time needed to configure SDN switches remotely. However, adding more controllers can negatively impact the delay in some cases, as the distributed control plane has to consistently exchange to synchronize the global topology among all deployed controllers. Thus, adding more controllers implies additional waiting time for synchronization. The optimal number of deployed controllers can vary from one application to another based on user requirements. Real-time applications, such as video streaming need low-delay to provide the best user experience for content viewers. On the other hand, high-availability applications, such as hosting servers, requires reliable and available connection with a higher emphasis on delivery than instantaneous response. For a successful deployment of SDN network these requirements have to be satisfied while minimizing the deployment cost.

In this paper, a new framework is introduced to help network designers to determine the number of SDN controllers and their locations to satisfy application requirements. This framework includes a greedy algorithm and set of graph metrics to study a given network to improve its overall performance in terms of minimizing latency, improving network resilience while minimizing overall deployment cost. The performance metrics that capture SDN network delays include: switch-to-controller (S2C) and the controller-to-controller (C2C) delays. Our introduced algorithm determines k SDN controller based on four graph-robustness metrics:

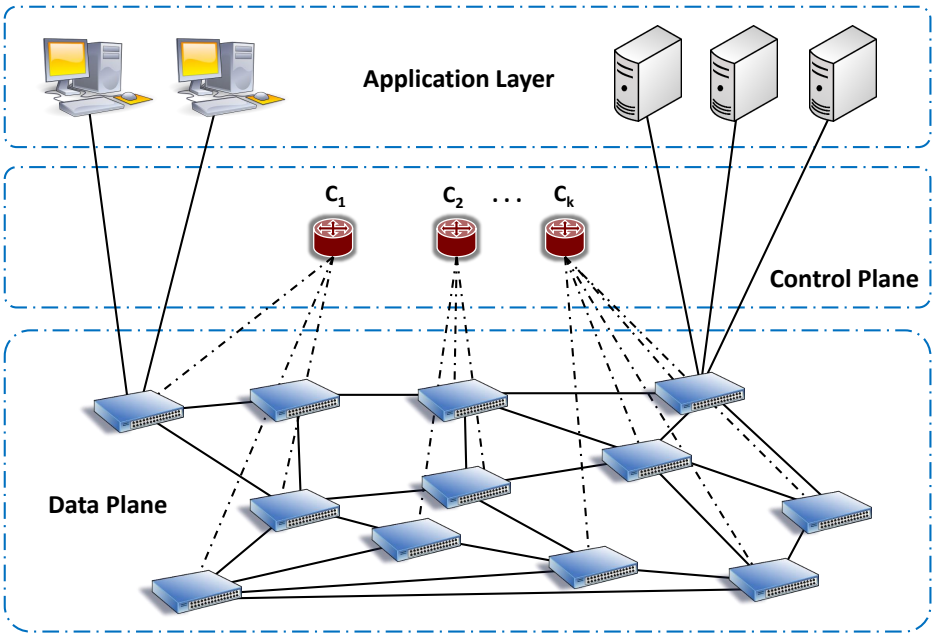


Fig. 1. SDN architecture.

algebraic connectivity (AC) [13], [14], [15], [16], network criticality (NC) [17], load centrality (LC) [18], and communicability [19]. To evaluate the proposed approach within the context of real-world physical networks, the proposed algorithm is applied in four different US-based backbone networks. In addition, the robustness of the proposed approach is evaluated in terms of three different centrality-based attacks, i.e., degree-, closeness-, and betweenness-based ones. Finally, in order to evaluate the network performance of the proposed approach with other state-of-the-art baseline approaches in use at present, the obtained results in terms of S2C and C2C delays and resilience, are compared with the ones obtained with k-median and k-center methods.

The contribution of this paper is threefold. Firstly, it proposes a new framework to aid network designers in determining the number of SDN controller to satisfy network requirements. The greedy-selection algorithm first introduced in a prior work by the author of this paper [20] is generalized towards determining locations of k SDN controllers to meet network requirements in terms of lowering delay, and improving network resilience while reducing deployment cost. In particular, the location algorithm is based on four different objective functions: AC, NC, LC, and communicability. Secondly, the proposed approach is evaluated within the context of four different real-world physical US-based backbone networks. Third, a novel resilience metric, called Attack Resilience, is defined and computed in the presence of three different centrality-based attacks, namely, degree-, closeness-, and betweenness-based ones. Thirdly, a novel resilience metric, called *attack resilience*, is defined and computed in the presence of three different centrality-based attacks, namely, degree-, closeness-, and betweenness-based ones.

The reset of this research article is organized as follows. A brief theoretical background of SDN controller placement and robustness metrics is presented in Section II. In Section III, the relevant related work is discussed. The k-controller selection algorithm proposed in this paper is introduced in Section IV. In Section V, the used evaluation protocol is described. In particular, in Subsection V-A, the used dataset is introduced, while in Subsection V-B, the different graph attack models used to test the proposed controller selection algorithm, are presented. Finally, in Section VI, the obtained results are presented and discussed. In particular, Section VI-A is devoted to the network resilience analysis while Section VI-B is devoted to the network delay analysis. Finally, a summary of our work with some concluding remarks and future directions for research are provided in Section VII.

II. Theoretical Background

In this section, a theoretical background of the controller placement problem (CPP) and graph robustness metrics, is provided in Subsection II-A and Subsection II-B, respectively. In addition, some relevant solutions proposed in the literature to address such issues are also discussed.

A. Controller Placement Problem

In a small-size network one SDN controller can be sufficient to remotely configure a set of SDN switches [11], [21]. However, as the size of the SDN network increases, more than one network controller is needed to maintain scalability requirements. To deploy a set of controllers for a large-size network such as a backbone network, three questions need to be answered during the network design phase [11]:

1) What is the minimum number of controllers to satisfy user-application requirements?
2) Where are the locations of the selected controllers?
3) How many switches should be attached to each selected controller?

In this context, identifying the optimal number of SDN controllers to be deployed and their locations, referred to as the CPP, becomes vital [11]. Several approaches can be found in the literature that addressing the CPP problem from different perspectives, such as reducing network delays [11] or increasing its reliability [22], [23], among others [24]. In [11], controllers are selected based on the k-median method. In this approach, controllers are selected to maintain minimum average delay delay among the SDN switches and controllers. Experimental results in [11], obtained by applying the proposed method on several mid-sized WANs, indicate that the propagation delay decreases as the number of controllers k increases. In [22], a location selection algorithm based on a reliability model computed using link operational probabilities, is proposed. The obtained results on several publicly available network topologies, show that the cost of deployment can be minimized while achieving reliability by carefully determining the placement of controllers. In [23], a control network reliability metric, based on the probability of path failures, is introduced to re-route traffic in case of control path loss. In addition, different placement algorithms are tested such as simulated annealing (SA) [25], which yield improved reliability against network failures. In addition, results in [23] also show that it is possible to find an optimal number of controllers to be deployed in terms of reliability, in the sense that locating more controllers would not further improve the results.

Here it is important to highlight that, although [11], [22] and [23] propose different CPP solutions, not all their results cannot be used for comparison purposes in this paper. In general, in order to be comparable, results should be obtained from a well-designed, clearly explained, and feasible to reproduce evaluation protocol. In particular, many of the parameters used in [23] are randomly selected, making it impossible to reproduce their results. The results in [22], are obtained on a publicly available dataset, making them useful benchmark results. Nevertheless, in [22] the controller locations are determined based on link operational probabilities, which are assumed to be identical, thus providing equally weightage to all the selected nodes. In this context, random results are generated, which cannot be reproduced. Finally, as discussed in Section I, the k-median method used in [11], is used in this paper for comparison. Nevertheless, the results reported in [11] cannot be directly compared with our results as they are not obtained on the same network topologies.

B. Robustness Graph Metrics

Graph metrics are used to evaluate the topological properties of a given network. They are used as indicators to differentiate between network topological designs in terms of performance and resilience. In particular, the topology of a given network can determine, among other essential aspects, its robustness against node removals. For instance, removing the central node in a star network can lead to full dysconnectivity. On the other hand, a full-mesh network can provide optimal resilience against such a challenge. However, a full-mesh network is infeasible in large-area topologies such as backbone, due to its excessive costs [26]. In our earlier studies, we have introduced a graph metric, namely 'nodal path disjoint', which captures the node centrality in terms of the number of disjoint paths to other nodes. This metric has been useful in determining the placement of SDN controllers to achieve network resilience [27].

In this paper, we selected four graph robustness metrics based on a comprehensive evaluation of most graph metrics to capture network resilience against network failures [28], specifically, AC and NC are used. In particular, the AC metric is focused on measuring the graph robustness against node removals [29], [26], while the NC metric measures the network robustness in case of topological failures [17]. In addition, to further evaluate the robustness of the proposed approach, two betweenness-based measures, such as the LC and communicability metrics, are also used. The four selected robustness metrics are described as follows:

- *Algebraic Connectivity:* AC, usually denoted as $a(G) = \lambda_2$, is the second smallest eigenvalue of its Laplacian matrix. AC has been extensively researched, showing several advantages when compared to other well-known robustness metrics, such as, for instance, the average node degree, for evaluating network resilience [30], [31], [32]. In addition, in [14], [15], [16], the AC capability of predicting graphs flow robustness is also highlighted.
- *Network Criticality:* NC, denoted as $\hat{\tau}$, is a spectral graph metric calculated as follows:

$$\hat{\tau} = \frac{2}{n-1}\text{Trace}(\mathcal{L}^{+}) \tag{1}$$

  where n represents the number of nodes while $\text{Trace}(\mathcal{L}^{+})$ is the trace of the Moore–Penrose inverse of the Laplacian matrix of the graph [17]. A lower value of NC, calculated as in Eq. 1, indicates a higher network robustness. A comprehensive study of this metric within the context of different network topologies as well as a comparison with other widely used robustness measures, can be found in [33].
- *Load Centrality*: LC, first introduced in [18], measures the load incurred by other nodes, in terms of node betweenness, by computing the shortest paths passing the given node. A larger value of LC indicates a higher network robustness.
- *Communicability:* The communicability measure, first introduced in [19], evaluates the number of walks that connects every pair of nodes. A larger value of communicability indicates a higher degree of network robustness.

III. Related Work

In recent years, several SDN protocols have been proposed in the literature that implement SDN networks in different scenarios, such as healthcare [34] and traffic [35] applications. One of the most widely used SDN protocols is OpenFlow. Since it was first introduced in 2011, this SDN protocol, originally developed at Stanford University [36], has gained high popularity, calling the attention of both academia and industry. In this section, some of the most relevant works addressing SDN technologies are discussed.

In [34], an SDN protocol is proposed to separate the application from the underlying physical infrastructure towards

reducing the total capital and maintenance costs. The proposed protocol allows intelligent health monitoring as well as customized data collecting, transmitting, and processing. The promising results reported in [34] provide a solid for further innovation in the field of healthcare applications. In [4], authors highlight that SDN can favor big data acquisition, transmission, storage, and processing. Further this line, authors in [4] studied the feasibility of applying SDN protocols in the context of big data networking. In particular, the available technologies allowing a joint design of big data and SDN were analyzed towards achieving a synergistic environment capable of exploiting SDN and big data advantages making both of them both benefit from each other. The reported results in [4] are promising highlighting the potential benefits of using SDN technologies in the context of big data applications. In [37], a software-defined IoT infrastructure, consisting of physical, control and application layers, is applied in the context of a new industry concept called Industry 4.0, with the purpose of providing flexible network resource allocation management and improving data exchange. Simulation results in [37], obtained from different Industry 4.0 scenarios, show that SDN technology is essential to the successful development of such new industry concept. In [35], a time-sensitive SDN (TSSDN) is introduced towards providing real-time guarantees in time-sensitive and non-time-sensitive traffic systems. The proposed TSSDN is based on bounding the non-deterministic queuing delays for time-sensitive traffic by exploiting the logical centralization paradigm of SDN to compute a transmission schedule for time-sensitive traffic initiated by the end systems based on a global view. Results in [35], show that the proposed TSSDN achieves deterministic end-to-end delays with low and bounded jitter.

The above discussion shows that SDN technologies can be implemented in diverse scenarios, such as healthcare, big data, industry, and traffic applications, to achieve different objectives, including customizing, reducing costs, minimizing delays, and improving performance. Regardless of the scenario, the successful implementation of SDN networks is highly dependent on network reliability. Although as previously discussed in Subsection II-A, while some network requirements can be fulfilled using only one controller, deploying more controllers, *i.e.*, making the network more redundant, can increase its reliability [24]. Several researchers have focused their works in developing multi-controller approaches towards improving the network reliability and resilience. In [24], a comprehensive survey of multi-controller based on SDN can be found. In Subsection II-A, the proposed approaches in [11], [22], and [23], have been discussed. In addition, some other approaches can be found in [38] and [39]. In [38], the possibility of improving the resilience of smart grids through SDN applications is evaluated using three illustrative use cases, showing that SDN technologies allow the strengthening of smart grid resilience, even under catastrophic circumstances. In [39], a CPP solution based on mobility-aware adaptative flow-rule placement is proposed for a software-defined access network (SDAN) to support IoT applications; good simulation results were obtained in terms of network delay, optimal number of activated access points (APs), control overhead, energy consumption, and costs.

Finally, despite the huge efforts made in the literature towards addressing the CPP problem in SDN applications, researchers agree that there still exist many research gaps and open challenges in the field [24]. In particular, according to the extensive analysis conducted in [24], further research needs to be conducted towards addressing scalability, consistency, reliability, resilience, and load balancing issues. In addition, most of the reported results in the literature have been obtained based on simulation experiments rather than experiments conducted in the real-world scenario. Then, in order to give some insight into the identified research gaps, a novel SDN controller selection approach aimed at improving the network resilience against targeted attacks is proposed in this paper and tested within the context of four different real-world US-based backbone physical networks.

IV. SDN Controller Selection Algorithm

In this section, the proposed approach aimed at selecting the locations of k controllers subject to the four objective functions, namely, AC, NC, LC, and communicability—towards optimizing network performance not only in terms of the S2C and C2C delays, but also of the resilience against centrality-based attacks, is introduced. The proposed greedy algorithm is implemented as follows, being the corresponding pseudo-code shown in Algorithm 1. First, the nodes are divided into k groups using a k-clustering algorithm. Then, for each cluster, the node satisfying the objective function is selected. The three functions defined in Algorithm 1: graphMetric(G), k-means(G), and nodeSelection($List$), are applied as follows. First, the graphMetric(G) function returns the graph metric value of a given graph G. This function is especially defined to support any graph metric, allowing the use of any of the four metrics proposed in this paper (AC, NC, LC, and

```
Functions:
graphMetric(G) := a generic graph metric of G
k-means(k,G) := k partitions of a graph G
nodeSelection(List) :=select a node based selected
graph metric function
Input:
G_i := input graph
k := number of returned SDN controllers
Output:
kControllers := a list of the selected controllers
begin
    selectedControllers = []
    partitions = k-partitions(k,G_i)
    for partition in partitions do
        graphMetricsVales = []
        for node in partition.nodes() do
            G_i.remove(node)
            nodeImpact = graphMetric(G_i)
            graphMetricsVales.append(node,nodeImpact)
            G_i.add(node)
        end
        bestCandidateNode =
          nodeSelection(graphMetricsVales)
        kControllers.add(bestCandidateNode)
    end
    return kControllers
end
```

Algorithm 1: A greedy algorithm for selecting k-controller.

TABLE I. US-BASED BACKBONE NETWORK TOPOLOGIES

Graph	Nodes	Links	Radius	Diameter
Internet2	57	65	8	14
Level 3	99	132	10	19
Sprint	264	313	19	37
AT&T	383	488	20	39

communicability). Then, the k-means(G) function divides N nodes into k partitions based on their Euclidean locations. Finally, the nodeSelection(N) function returns the node that maximizes the network robustness.

V. EVALUATION FRAMEWORK

In this section, the evaluation Framework used in this paper is described. In particular, subsection V-A describes the used dataset is introduced, while subsection V-B presents the three different centrality-based attacks used to test the robustness of the proposed approach.

A. Dataset

In this paper, four different backbone fiber-level network topologies available in the KU-TopView Network Topology Tool [40], particularly, Internet2[1], Sprint, Level 3 [41], and AT&T, are used to evaluate the performance of the proposed approach. In order to illustrate their graph properties, some commonly used graph metrics are shown in Table I.

B. Centrality-based Attacks

In this paper, graph-theoretic models are used to attack the network and evaluate its robustness against node removals. In particular, three different centrality-based attacks are considered as follows the degree-, closeness-, and betweenness-based ones [42]. The degree-based attack targets nodes with the highest number of connected links while closeness-based attack targets the nodes closest to central nodes with respect to hop-count. The betweenness-based attack targets the node through which the highest number of shortest paths pass. For each one of them, the list of removed nodes is determined in an adaptive way, allowing a better selection of the highest centrality than in the case of using a single evaluation for selecting the highest number of targeted nodes [43].

VI. RESULTS AND DISCUSSION

In this section, the results obtained from the implementation of the proposed controller selection approach based on the four robustness functions, namely, AC, NC, LC, and communicability, in the four physical-level networks shown in Table I, are presented and discussed. In addition, the k-center and k-median methods are also applied within the context of these four networks for the sake of comparison.

A. Controller Selection Evaluation

In order to select the controller locations with each of the six different tested methods, that is, for the AC, NC, LC, communicability, k-center and k-median methods, the number of controllers is varied from $k = 1$ to $k = 20$. The obtained results are geographically illustrated in Fig. 2, where the optimal controller locations computed by the six methods for the AT&T network are shown. In particular, for the sake of a better visualization, only the locations for $k = 4$, $k = 8$, $k = 12$, and $k = 16$ are shown. The results obtained for the other three physical networks yield similar conclusions and are not included here due to space constraints.

B. Delay Analysis

In this subsection, the obtained end-to-end delay results are analyzed. Typically, while a packet is sent through the Internet it experiences four types of delay: nodal processing, queuing, transmission, and propagation delays. In this paper, the propagation delay is particularly addressed, assuming that routes are selected by means of the shortest lengths or hop-count between the SDN switches and the nearest controllers. In this sense, different delay related metrics, namely, the C2C and the S2C ones, are considered. The former is analyzed in subsection VI-C, while the latter is analyzed in subsection VI-D.

C. Controller-to-Controller Delay

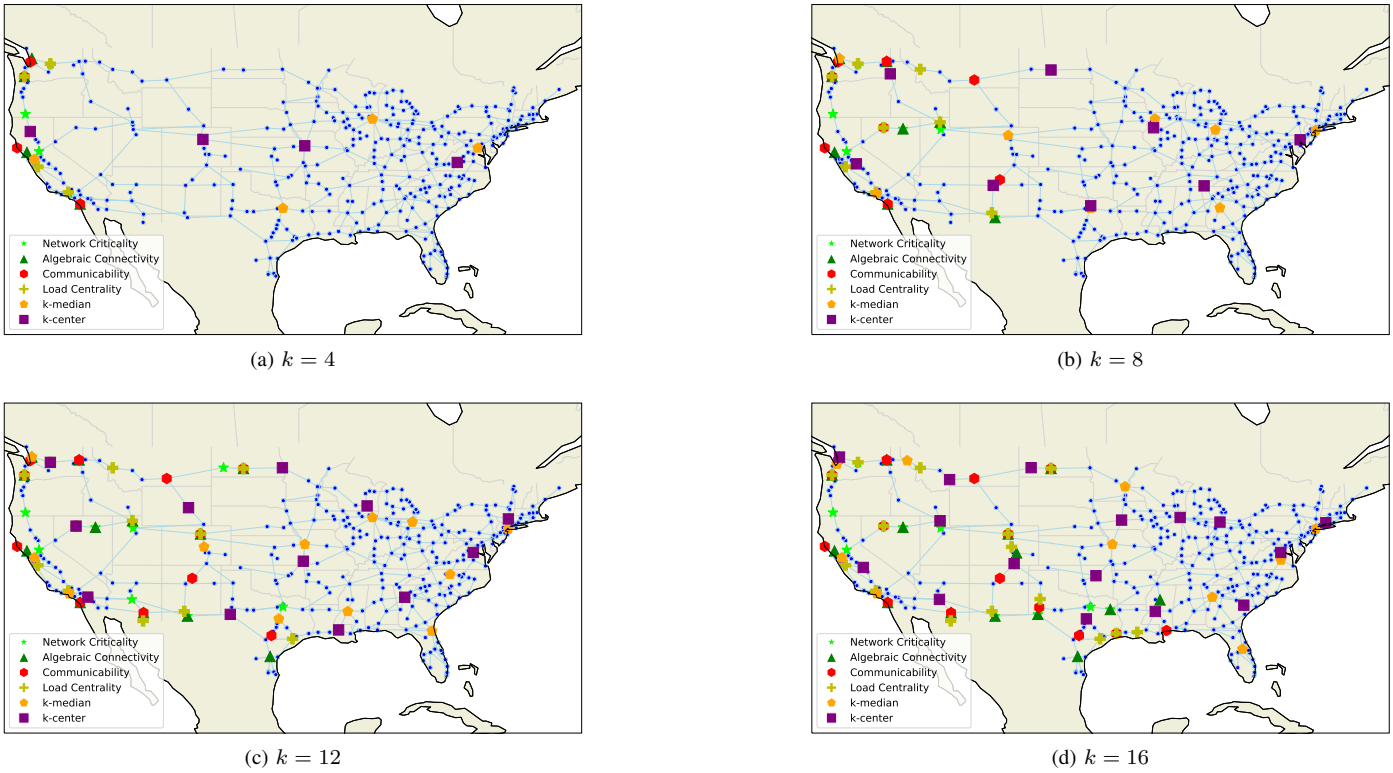
In a distributed SDN network, multiple controllers are deployed to ensure load-balancing and network resilience against attacks. In this context, controllers should be able to deal with real-time data exchange towards sharing a global network view and providing proper flow rules to the switches. In the case of backbone networks, these controllers can be in different cities, increasing the signal propagation time and the number of required hops, causing additional delay. In this paper, the inter-controllers delay is evaluated based on these two factors through the C2C delay and the C2C hop-count, respectively. The former, defined as the average of the propagation delay between the deployed controllers, is computed as the shortest path length divided by the propagation speed[2]. The latter, is defined as the average number of hops corresponding to the shortest paths between the deployed controllers.

In this subsection, the C2C delay is analyzed as a function of the number of deployed controllers k. In particular, the impact of increasing k on the end-to-end delay (C2C delay) and on the number of hops (C2C hop-count) for the four physical networks is shown in Fig. 3 and 4, respectively.

Results in Fig. 3 show that k-median and k-center methods yield high C2C delays, for all the network topologies. For instance, for the Internet2 topology, the maximum value reaches 15 ms. This is due to the fact that these methods select controllers to minimize the S2C delay, rather than the C2C delay. Then, controllers are distributed uniformly, increasing the shortest path distance between them. In addition, Fig. 3 also shows that, for the k-median and k-center methods, the C2C delay decreases as the number of deployed controllers k increases, reaching a maximum for $k = 2$, for all topologies. Note the reader that the C2C delay is not defined for $k = 1$

[1] http://www.internet2.edu

[2] The propagation speed is assumed to be $2 \times 10^8 m/s$.



(a) $k = 4$ (b) $k = 8$

(c) $k = 12$ (d) $k = 16$

Fig. 2. Controller Placement for AT&T.

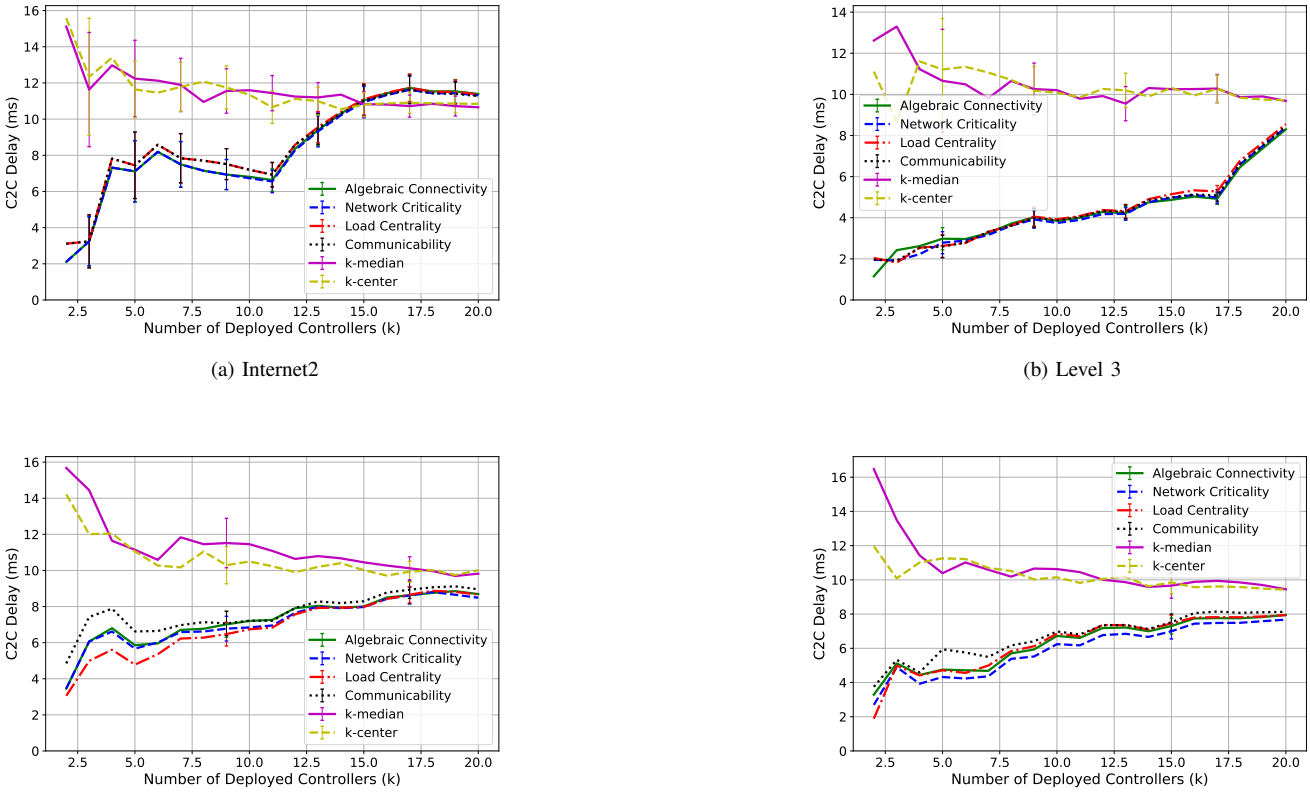
because, in such a case, only one controller is deployed. For the AC, NC, LC and communicability methods, the C2C delay is very low for all the topologies, as they tend to select controller locations that are close to each other, as shown in Fig. 2. In these cases, in contrast to the the k-median and k-center methods, the C2C delay tends to increase as k increases for all the topologies excepting the Internet2 one, where the C2C delay increases for k values between 2 and 4, and for k values greater than 11, but showing no delay penalty as k reaches 11. Nevertheless, even for the largest number of controllers considered here ($k = 20$), the C2C delay obtained with the AC, NC, LC, and communicability methods, are still lower than the ones obtained with the k-median and k-center ones. Finally, similar results can be observed in Fig. 4, making it possible to conclude that the AC, NC, LC, and communicability methods outperform k-median and k-center methods in terms of the C2C delay and hop-count.

D. Switch-to-Controller Delay

In a distributed SDN network, SDN switches are commonly connected to the nearest controller, making it crucial to minimize the delay between them. In this paper, this delay is captured taking into account two delay related metrics, specifically, the S2C delay, defined as the average of the propagation delay between the deployed switches and the nearest controller, and the S2C hop-count, defined as the average number of hops of the shortest paths between the deployed switches and the nearest controller. In this subsection, the S2C delay is analyzed as a function of k. In particular, the impact of increasing k on the end-to-end delay (S2C delay) and on the number of hops (S2C hop-count) for the four physical networks is shown in Fig. 5 and 6, respectively.

The results in Fig. 5 show that the average S2C delay obtained with the k-median and k-center methods are significantly lower than the ones obtained in the case of the C2C delay for all the topologies. This was expected since, as mentioned in Subsection VI-C, k-median and k-center methods select controllers by minimizing the S2C delay. These results are also better than the ones corresponding to the AC, NC, LC and communicability methods for all the topologies. In addition, Fig. 5 also shows that the S2C delay decreases as k increases for all the methods and topologies. This tendency is particularly noticeable in the case of the AC, NC, LC, and communicability methods, because of which the difference between these methods and the k-median and k-center ones to decrease as k increases. Moreover, as k increases, the S2C delay tends to a low value (almost the same one) for all the methods and topologies. Take, for instance, the case of the Internet2 topology, where the S2C delay of k-median and k-center methods for $k = 11$ is around 2.0 ms, while for the other four methods the delay is around 3.6 ms. This difference is not particularly significant. Moreover, depending on the type of application and its requirements, this difference could be taken into consideration or be even neglected.

Finally, the results in Fig. 6, show a similar behavior in terms of the S2C hop-count for all the methods and topologies. As such, the same comments stand.



(a) Internet2

(b) Level 3

(c) Sprint

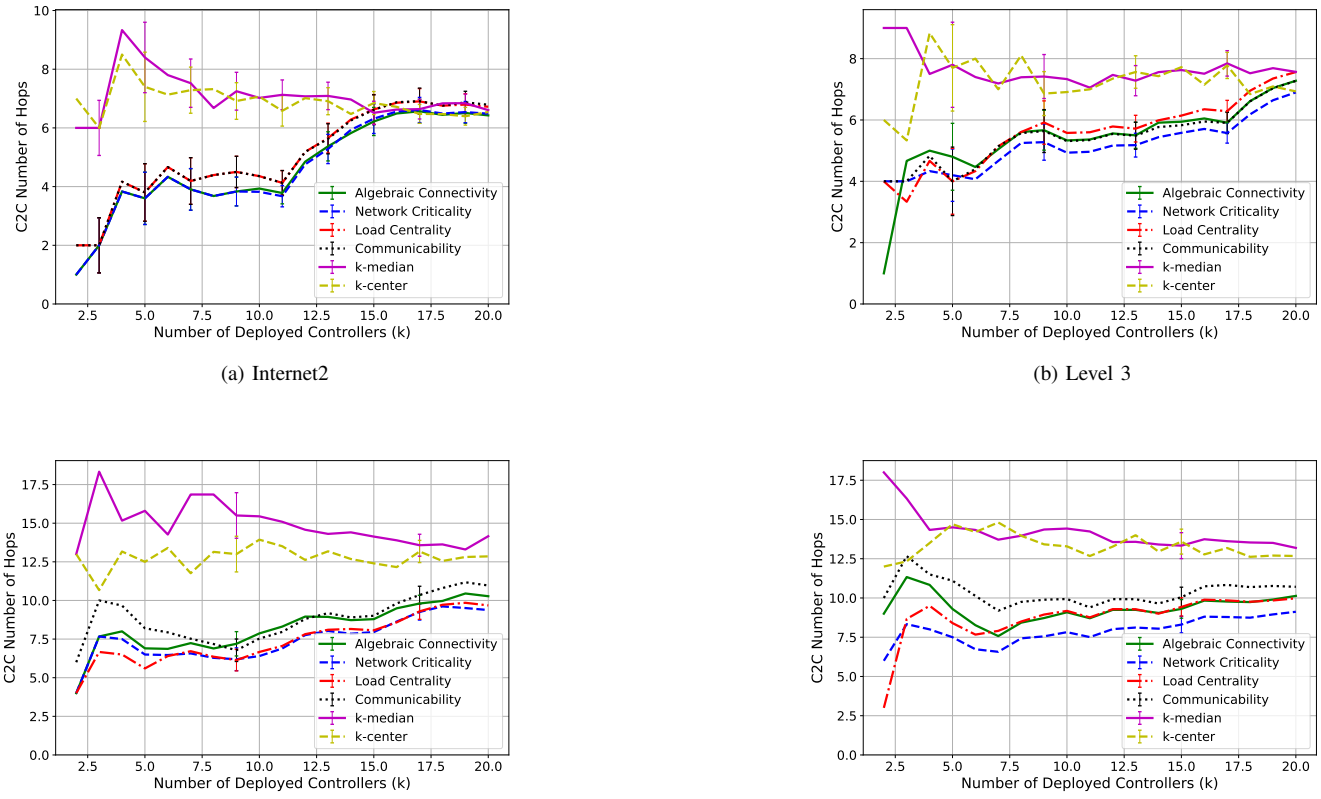
(d) AT&T

Fig. 3. Inter-controller delay.

E. Robustness to Targeted Attacks Analysis

In this section, the network resilience of the proposed approach against targeted attacks is evaluated as a function of the number of deployed controllers k. ITherefore, a new metric, termed as attack resilience (AR), that captures the connectivity to SDN controllers during a given attack, is defined. The AR is computed as the sum of controller reachability, first introduced in [20], during a given attack. The Attack Resilience results of the four physical networks are shown in Fig. 7, 8, 9, and 10, for the six controller selection methods (AC, NC, LC, communicability, k-median and k-center), and the three centrality-based attacks. In particular, Fig. 7a, 7b and 7c show resilience results for the Internet2 topology, Fig. 8a, 8b, and 8c, show the corresponding results for the Level-3 topology, Fig. 9a, 9b, and 9c show the results for the Sprint topology, and Fig. 10a, 10b, and 10c show results for the AT&T topology, for the degree-, closeness- and betweenness-based attacks, respectively. In all cases, the AR is minimum at $k = 1$, confirming that network resilience against targeted attacks is poor when the SDN network has only one controller, as previously suggested in [24]. Similarly, AR increases as k increases for all methods [24]. In all cases, the behavior of all methods remains similar until a particular value of k is reached, becomes different when k further increases, indicating that different methods have different impacts on the network resilience. In this regard, the behavior is similar for all topologies, differing only in the value of k for which the difference among the methods becomes appreciable. These different behaviors are described as follows:

- Internet2:For the Internet2 topology, the resilience values are similar for the six methods for values of k between 1 and 7, after which the NC method outperforms the other ones. For instance, from Fig. 7a it can be seen that for k = 11 the attack resilience against the degree-based attack is 75 for the k-median method and 145 for the NC method, showing approximately 95% improvement in resilience. Here, it is important to highlight that, although increasing k does always increase AR, it is not always a good strategy. For instance, the AR against the degree-based attack improves by 3% as k increases from 11 to 17. This will be cost-inefficient since, while there exists a significant extra cost for deploying 6 additional controllers, the corresponding resilience improvement is not significant.
- Level-3: For the Level-3 topology, the AR increases for all methods (and attacks) correspondingly between $k = 1$ and $k = 5$. When k further increases, the AC and NC methods are the ones that most distinguish themselves from the rest. For k values between 5 and 9, the AC method provides the best results, while when k is greater than 9, the NC method outperforms all the



(a) Internet2

(b) Level 3

(c) Sprint

(d) AT&T

Fig. 4. Inter-controller hop-count.

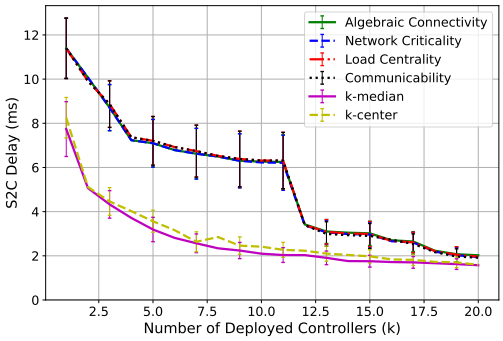
other methods. For instance, when $k = 9$, the AR is 100 for all methods but for the AC one that provides approximately 30% of improvement in the presence of the degree-based attack.

- Spring: For the Spring topology, the AR increases for all the methods in a similar way for k between 1 and 11, for all the attacks, being particularly noticeable in the case of the degree-based one. For k values greater than 11, the NC method outperforms the other methods. For instance, for $k = 14$, the closeness-based Attack Resilience value achieved by the NC method is 350, which is around 75% higher than communicability, which provides the worst results.
- AT&T: For the AT&T topology the AR increases correspondingly for k values between 1 and 3, for all methods and attacks. For this topology, the k-median method provides the best resilience results when k values between 3 and 7 for all the attacks. For greater values of k, the NC method outperforms the other ones for the closeness- and betweenness-based attacks, while for the degree-based attack, the k-median remains as the best one.

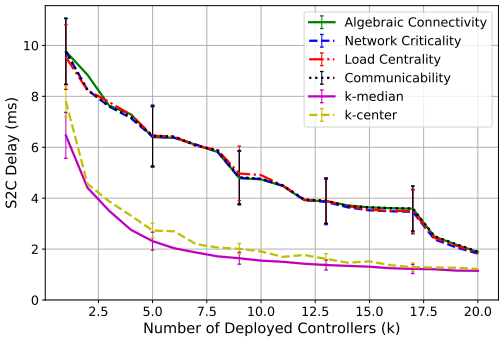
Based on the above discussion, it can be concluded that the NC method provides the best results in most of the cases, while AC and k-median methods show better results than the LC, communicability, and k-center. In addition, once the optimal method to place the controllers has been selected, which in this paper is the NC method, the minimum number of deployed controllers should be determined in terms of minimizing both network resilience and deployment costs. The elbow method can be used for finding the minimum k with the maximum resilience gain, in the sense that further increasing k would not result in significant resilience improvement. Based on the elbow method, the optimal k when using the NC method has been computed for each of the four network topologies, being $k = 11$ for the Internet2 topology, $k = 17$ for the Level-3 and AT&T topologies, and $k = 16$ for the Spring topology, showing that this optimal number highly depends on the structure of the physical network.

VII. Conclusions and Future Work

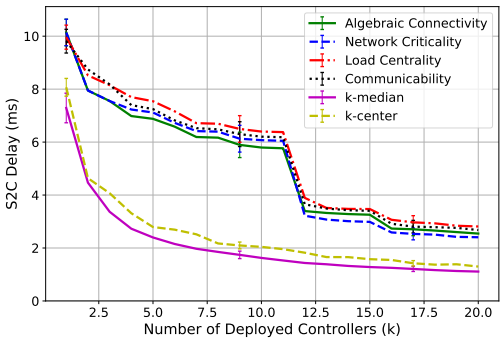
Nowadays, real-time or near real-time data delivery is a crucial aspect in many Internet applications. In this context, SDN technologies can provide efficient solutions, being widely used in diverse scenarios, such as healthcare and traffic applications, for customizing, reducing costs, and minimizing delays. As they are publicly available, the vulnerability of SDN networks is high, making it crucial to evaluate its robustness against different attacks. In this line, increasing the number of deployed controllers can be used to improve reliability through resilience. Nevertheless, deploying multiple controllers is a



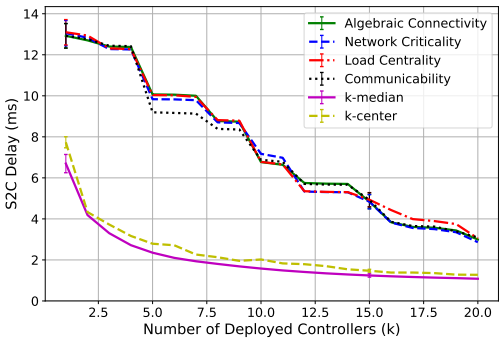
(a) Internet2



(b) Level3



(c) Sprint



(d) AT&T

Fig. 5. Switch delay.

challenging task, especially in terms of determining their number and location. In this paper, a new controller placement approach based on four robustness metrics, namely, AC, NC, LC, and communicability, has been proposed to minimize both S2C and C2C delays, as well as to maximize the network resilience. To bridge the research gap regarding benchmark results obtained in real-world networks, the proposed approach is tested within the context of four different physical graphs, and compared with the present state-of-the-art baseline methods.

The performance of the proposed approach has been evaluated in terms of the C2C and S2C delays as well as the resilience against three different centrality-based attacks, i.e., degree-, closeness-, and betweenness-based methods. In particular, for evaluating the network resilience, a novel robustness metric, the attack resilience, has been defined. The obtained results showed that the NC method serves as a good metric to select the optimal number of deployed controllers for an SDN environment in terms of achieving low C2C delay and high resilience against centrality-based attacks in backbone networks. Moreover, for the S2C delay, although being outperformed by the baseline methods (k-median and k-center), the deployment of the NC method does not have a significant influence.

Finally, for future work, it is the authors' intention to extend the present analysis by applying the proposed controller selection approach to other widely used topologies, such as data centers and smart cities.

ACKNOWLEDGMENTS

The authors extend their appreciation to the Deanship of Scientific Research at King Saud University for funding this work through the Research Project No. R5-16-03-03.

REFERENCES

[1] I. F. Akyildiz, A. Lee, P. Wang, M. Luo, and W. Chou, "Research challenges for traffic engineering in software defined networks," *IEEE Network*, vol. 30, pp. 52–58, May 2016.

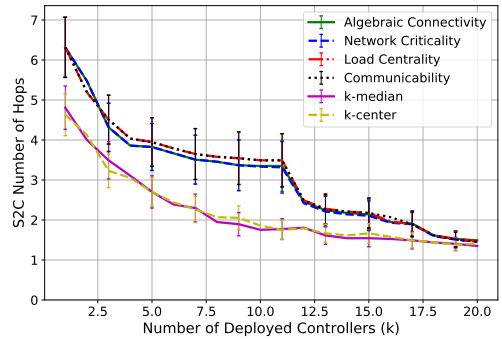
[2] H. Kim and N. Feamster, "Improving network management with software defined networking," *IEEE Communications Magazine*, vol. 51, no. 2, pp. 114–119, 2013.

[3] O. N. Fundation, "Software-defined networking: The new norm for networks," *ONF White Paper*, vol. 2, pp. 2–6, 2012.

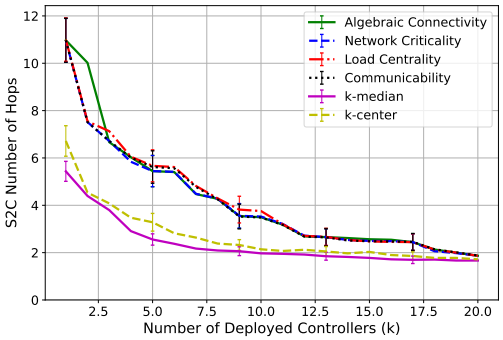
[4] L. Cui, F. R. Yu, and Q. Yan, "When big data meets software-defined networking: Sdn for big data and big data for sdn," *IEEE Network*, vol. 30, pp. 58–65, January 2016.

[5] M. Wang, N. Karakoc, L. Ferrari, P. Shantharama, A. S. Thyagaturu, M. Reisslein, and A. Scaglione, "A multi-layer multi-timescale network utility maximization framework for the sdn-based layback architecture enabling wireless backhaul resource sharing," *Electronics*, vol. 8, no. 9, 2019.

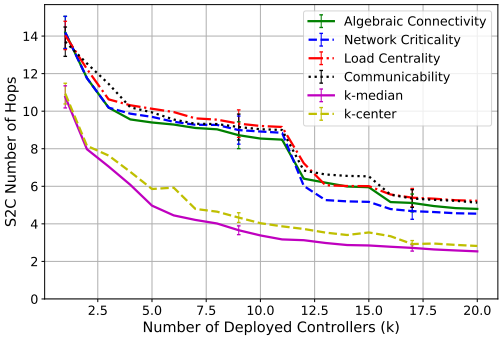
[6] M. Hussain, N. Shah, and A. Tahir , "Graph-based policy change detection and implementation in sdn," *Electronics*, vol. 8, no. 10, 2019.



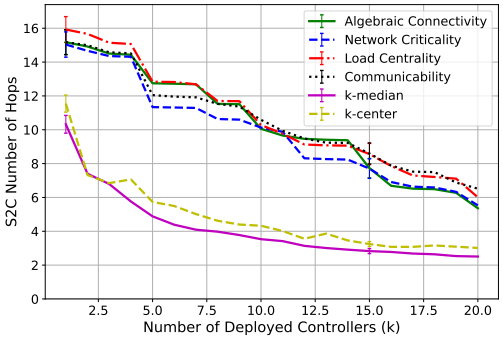
(a) Internet2



(b) Level3

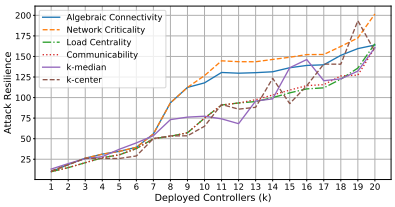


(c) Sprint

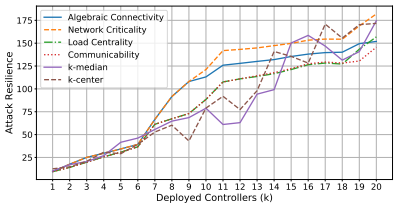


(d) AT&T

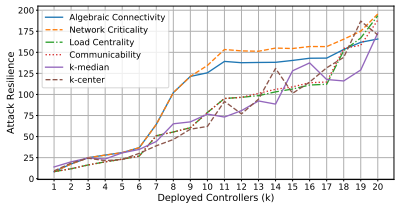
Fig. 6. Switch hop-count.



(a) Internet2 degree-based attack

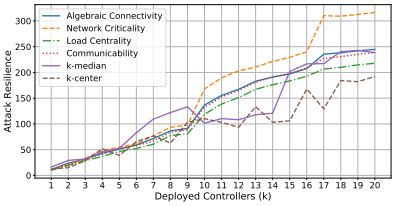


(b) Internet2 closeness-based attack

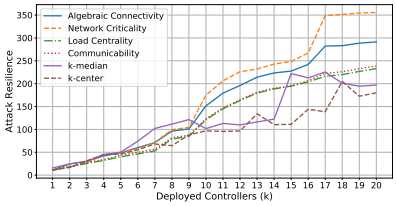


(c) Internet2 betweenness-based attack

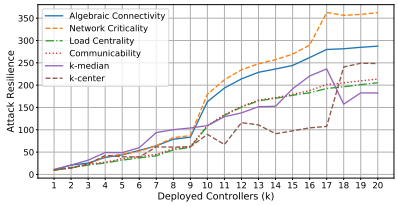
Fig. 7. Centrality-based attacks evaluation of internet2 backbone network



(a) Level 3 degree-based attack

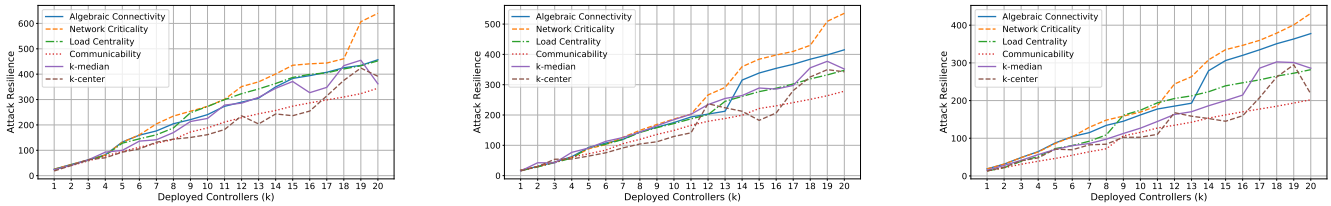


(b) Level 3 closeness-based attack



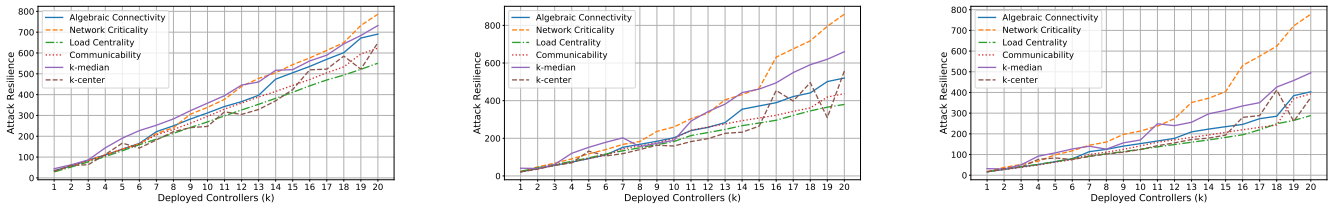
(c) Level 3 betweenness-based attack

Fig. 8. Centrality-based attacks evaluation of Level 3 backbone network



(a) Sprint degree-based attack (b) Sprint closeness-based attack (c) Sprint betweenness-based attack

Fig. 9. Centrality-based attacks evaluation of Sprint backbone network



(a) AT&T degree-based attack (b) AT&T closeness-based attack (c) AT&T betweenness-based attack

Fig. 10. Centrality-based attacks evaluation of AT&T backbone network

[7] Z. Shah and S. Cosgrove, "Mitigating arp cache poisoning attack in software-defined networking (sdn): A survey," *Electronics*, vol. 8, no. 10, 2019.

[8] M. J.F. Alenazi, "Evaluating multipath tcp resilience against link failures," *The ISC International Journal of Information Security*, vol. 11, no. 3, pp. 113–122, 2019.

[9] P. Charalampou and E. D. Sykas, "An sdn focused approach for energy aware traffic engineering in data centers," *Sensors*, vol. 19, no. 18, 2019.

[10] M. U. Younus, S. u. Islam, and S. W. Kim, "Proposition and real-time implementation of an energy-aware routing protocol for a software defined wireless sensor network," *Sensors*, vol. 19, no. 12, 2019.

[11] B. Heller, R. Sherwood, and N. McKeown, "The Controller Placement Problem," in *Proceedings of the First ACM SIGCOMM Workshop on Hot Topics in Software Defined Networks (HotSDN)*, (Helsinki), pp. 7–12, Proceedings of the 1st ACM SIGCOMM Workshop on Hot Topics in Software Defined Networks (HotSDN), August 2012.

[12] J. P. Sterbenz, D. Hutchison, E. K. Cetinkaya, A. Jabbar, J. P. Rohrer, M. Scholler, and P. Smith, "Redundancy, Diversity, and Connectivity to Achieve Multilevel Network Resilience, Survivability, and Disruption Tolerance," *Telecommunication Systems*, vol. 56, no. 1, 2011.

[13] M. Fiedler, "Algebraic connectivity of graphs," *Czechoslovak Mathematical Journal*, vol. 23, no. 2, pp. 298–305, 1973.

[14] E. K. Çetinkaya, M. J. F. Alenazi, J. P. Rohrer, and J. P. G. Sterbenz, "Topology Connectivity Analysis of Internet Infrastructure Using Graph Spectra," in *Proceedings of the 4th IEEE/IFIP International Workshop on Reliable Networks Design and Modeling (RNDM)*, (St. Petersburg), pp. 752–758, October 2012.

[15] E. K. Çetinkaya, M. J. F. Alenazi, A. M. Peck, J. P. Rohrer, and J. P. G. Sterbenz, "Multilevel Resilience Analysis of Transportation and Communication Networks," *Springer Telecommunication Systems Journal*, 2013. (accepted in July 2013).

[16] M. J. F. Alenazi, E. K. Çetinkaya, and J. P. G. Sterbenz, "Network Design and Optimisation Based on Cost and Algebraic Connectivity," in *Proceedings of the 5th IEEE/IFIP International Workshop on Reliable Networks Design and Modeling (RNDM)*, (Almaty), pp. 193–200, September 2013.

[17] A. Tizghadam and A. Leon-Garcia, "Autonomic traffic engineering for network robustness," *Selected Areas in Communications, IEEE Journal on*, vol. 28, no. 1, pp. 39–50, 2010.

[18] K.-I. Goh, B. Kahng, and D. Kim, "Universal behavior of load distribution in scale-free networks," *Physical Review Letters*, vol. 87, no. 27, p. 278701, 2001.

[19] E. Estrada, D. J. Higham, and N. Hatano, "Communicability betweenness in complex networks," *Physica A Statistical Mechanics and its Applications*, vol. 388, pp. 764–774, March 2009.

[20] M. J. F. Alenazi, "On sdn controller placement to achieve robustness against targeted attacks," in *Complex Networks & Their Applications VI* (C. Cherifi, H. Cherifi, M. Karsai, and M. Musolesi, eds.), (Cham), pp. 633–645, Springer International Publishing, 2018.

[21] 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. e2018 nem.2018.

[22] F. J. Ros and P. M. Ruiz, "On reliable controller placements in Software-Defined Networks," *Computer Communications*, vol. 77, pp. 41–51, March 2016.

[23] Y. Hu, W. Wendong, and X. Gong, "Reliability-aware controller placement for Software-Defined Networks," *Integrated Network . . .*, pp. 672–675, 2013.

[24] T. Hu, Z. Guo, P. Yi, T. Baker, and J. Lan, "Multi-controller based software-defined networking: A survey," *IEEE Access*, vol. 6, pp. 15980–15996, 2018.

[25] S. Kirkpatrick, C. D. Gelatt, M. P. Vecchi, *et al.*, "Optimization by simulated annealing," *science*, vol. 220, no. 4598, pp. 671–680, 1983.

[26] M. J. Alenazi, "Graph resilience improvement of backbone networks via node additions," in *2016 8th International Workshop on Resilient Networks Design and Modeling (RNDM)*, pp. 231–237, Sept 2016.

[27] M. J. Alenazi and E. K. Çetinkaya, "Resilient placement of sdn controllers exploiting disjoint paths," *Transactions on Emerging Telecommunications Technologies*, vol. 0, no. 0, p. e3725. e3725 ett.3725.

[28] M. J. Alenazi and J. P. Sterbenz, "Comprehensive comparison and accuracy of graph metrics in predicting network resilience," in *Design of Reliable Communication Networks (DRCN), 2015 11th International Conference on the*, pp. 157–164, IEEE, 2015.

[29] A. Jamakovic and S. Uhlig, "On the relationship between the algebraic connectivity and graph's robustness to node and link failures," in *Proceedings of 3rd EuroNGI Conference on the Next Generation Internet Networks*, pp. 22:1–22:8, 2007.

[30] H. Wang and P. Van Mieghem, "Algebraic connectivity optimization via link addition," in *Proceedings of the 3rd ICST International Conference*

on Bio-Inspired Models of Network, Information and Computing Sytems (BIONETICS), (Hyogo, Japan), pp. 22:1–22:8, November 2008.

[31] W. Liu, H. Sirisena, K. Pawlikowski, and A. McInnes, "Utility of algebraic connectivity metric in topology design of survivable networks," in *Proceedings of the 7th IEEE International Workshop on Design of Reliable Communication Networks (DRCN)*, (Washington, DC), pp. 131–138, October 2009.

[32] A. Sydney, C. Scoglio, and D. Gruenbacher, "Optimizing algebraic connectivity by edge rewiring," *Applied Mathematics and Computation*, vol. 219, no. 10, pp. 5465–5479, 2013.

[33] A. Bigdeli, A. Tizghadam, and A. Leon-Garcia, "Comparison of network criticality, algebraic connectivity, and other graph metrics," in *Proceedings of the 1st Annual Workshop on Simplifying Complex Network for Practitioners*, p. 4, ACM, 2009.

[34] L. Hu, M. Qiu, J. Song, M. S. Hossain, and A. Ghoneim, "Software defined healthcare networks," *IEEE Wireless Communications*, vol. 22, pp. 67–75, December 2015.

[35] N. G. Nayak, F. Dürr, and K. Rothermel, "Time-sensitive software-defined network (tssdn) for real-time applications," in *Proceedings of the 24th International Conference on Real-Time Networks and Systems*, RTNS '16, (New York, NY, USA), pp. 193–202, ACM, 2016.

[36] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner, "OpenFlow: Enabling Innovation in Campus Networks," *SIGCOMM Comput. Commun. Rev.*, vol. 38, pp. 69–74, March 2008.

[37] J. Wan, S. Tang, Z. Shu, D. Li, S. Wang, M. Imran, and A. V. Vasilakos, "Software-defined industrial internet of things in the context of industry 4.0," *IEEE Sensors Journal*, vol. 16, pp. 7373–7380, Oct 2016.

[38] X. Dong, H. Lin, R. Tan, R. K. Iyer, and Z. Kalbarczyk, "Software-defined networking for smart grid resilience: Opportunities and challenges," in *Proceedings of the 1st ACM Workshop on Cyber-Physical System Security*, CPSS '15, (New York, NY, USA), pp. 61–68, ACM, 2015.

[39] S. Bera, S. Misra, and M. S. Obaidat, "Mobi-flow: Mobility-aware adaptive flow-rule placement in software-defined access network," *IEEE Transactions on Mobile Computing*, vol. 18, pp. 1831–1842, Aug 2019.

[40] J. P. Sterbenz, J. P. Rohrer, E. K. Çetinkaya, M. J. F. Alenazi, A. Cosner, and J. Rolfe, "Ku-topview network topology tool." http://www.ittc.ku.edu/resilinets/maps, 2010.

[41] KMI Corporation, "North American Fiberoptic Long-haul Routes Planned and in Place," 1999.

[42] L. C. Freeman, "A Set of Measures of Centrality Based on Betweenness," *Sociometry*, vol. 40, no. 1, pp. 35–41, 1977.

[43] P. Holme, B. J. Kim, C. N. Yoon, and S. K. Han, "Attack Vulnerability of Complex Networks," *Phys. Rev. E*, vol. 65, p. 056109, May 2002.