Journal of Engineering and Science Research 8 (1): 01-07, 2024
e-ISSN: 2289-7127

DOI: 10.26666/rmp.jesr.2024.1.1

Efficient and Lightweight Link Discovery Protocol (ELDP) for Software Defined Networks

Aladesote, Olomi Isaiah[1.5], Azizol, Abdullah[2,*], Normalia, Samian[3], Zurina, Hanapi Mohd[4]

[1,2,3,4]Department of Communication Technology and Network, Faculty of Computer Science and Information Technology, Universiti Putra Malaysia, Serdang 43400, Malaysia.
[5]Department of Computer Science, Federal Polytechnic, Ile Oluji, Ondo State, Nigeria.
*Corresponding Author: gs57427@student.upm.edu.my

Received: 15 December 2023; Revised: 28 February 2024; Accepted: 01 March 2024; Published: 10 June 2024

Abstract: The paradigm change in the network industry has a strong correlation with the associated limitations of traditional networks. To mention, some of the reported limitations include inconsistent network regulations, vendor dependencies, complexity, and scalability issues, which have been addressed by the advent of software-defined networks (SDN). SDN introduces a novel network architecture that aims to overcome challenges such as inconsistent network regulations, vendor dependencies, complexity, and scalability issues. To achieve this, SDN relies on a controller that possesses a comprehensive view of the entire network, enabling effective resource allocation. However, the existing OpenFlow Discovery Protocol (OFDP), which utilizes LLDP packets for link topology discovery, suffers from inefficiency and the need for multiple messages (packet-out and packet-in) to detect links in OpenFlow devices. This research study proposes an Efficient and Lightweight Topology Discovery (ELDP) to address these shortcomings. In the proposed ELDP, controllers employ a more efficient approach by transmitting a single packet-in to a switch, which then distributes it to other switches. The switch maps adjacent switches connected to its ports and Short Switch Chase ID (SSCID) while communicating with the controller via a packet_in the message. This approach optimizes the link discovery process. The experimental results demonstrated some significant improvements, with an average reduction of 50% to 93.45% in the number of messages required for discovery across various network topologies. These findings validate the effectiveness of the ELDP approach in streamlining the link discovery process within SDN environments.

Keywords: *Link Topology, OpenFlow Discovery Protocol, Mapping, Software-defined network, Topology_Map_Database*

1. Introduction

Computer networks have advanced dramatically over the years (during the last thirty years), moving from conventional networks that are static to dynamically created architecture. Developing an open and programmable network is the overall goal of software-defined networking (SDN) [1]. SDN, the primary architecture of the current network paradigm, employs logically centralized control to enhance its appeal and acceptance in large networks such as corporations and data centers, even with the increase in internet traffic [2]. In comparison to traditional network approaches, SDN offers superior performance in centrally administering network traffic through the utilization of the control plane and data plane [3], [4], [5]. Conversely, the traditional network design with a single control plane for decision-making and traffic forwarding, along with nonprogrammable interfaces, proves to be ineffective [6], [7].

Corresponding Author: Aladesote Olomi Isaiah, Department of Communication Technology and Network, Universiti Putra Malaysia, +2348030657156 /+601126836003

The controller's primary function is to discover topology [8], [9]. The topology discovery involves the switch, host (carried out sometimes), and link discoveries. The discoveries result in topology management, which leads to a global view of SDN. The switch discovery offers the SDN controller the switch identities and their port status while the link discovery is necessary to carry out several topology-related tasks. Traditionally, the OpenFlow Discovery Protocol (OFDP) using LLDP packets was employed for link discovery in networks [10] and SDN continues to utilize a similar strategy. However, this approach suffers from inefficiency and security risks, ultimately affecting network performance. In this strategy, the controller sends a packet-out message to every port in a switch, resulting in an equal number of packet-out messages as there are network ports.

To provide a safe and lightweight link topology, Nehra et al. 2019 [11] propose SLDP to prevent, recognize, and lessen the security concerns associated with LLDP. The research employed a port eligibility list, which makes the ports that transmit vulnerable packets to be ineligible for subsequent packet(s) transmission. However, while addressing certain security concerns associated with LLDP, suffers from inefficiency due to the high volume of packets handled by the controller in a single cycle. This inefficiency impacts network performance and poses security risks. The research problem is centered around optimizing the link discovery process within SDN networks. Specifically, it aims to reduce the number of packets transmitted per cycle by proposing and implementing Efficient and Lightweight Topology Discovery (ELDP). The need for this research stems from the limitations of existing link discovery protocols in SDN networks, as highlighted by Nehra et al. in their 2019 study. SLDP, an improvement over LLDP, still presents challenges related to network efficiency and security. By proposing ELDP and its innovative features, such as SSCID and the mapping strategy, this research endeavours to provide a more effective solution to these challenges. The implementation and comparative analysis of ELDP against SLDP using metrics like efficiency, topology discovery time, and packet construction overhead will offer empirical evidence of its improvements over the existing protocol, thereby contributing to the advancement of SDN technology.

The proposed approach achieves this reduction by employing a mapping strategy and introducing the concept of Randomized Sequence Number (Check TLV) to complement the MAC address. Switches use MAC addresses when sending a packet-in message to the controller, and each switch port maps its neighboring switches. The contributions of the proposed scheme can be summarized as follows:

(i) Propose an ELDP, a pivotal advancement in Software-Defined Networking (SDN), offering a refined approach to topology discovery and network management. Its core contributions lie in streamlining the discovery process through the introduction of Short Switch Chase ID (SSCID) and mapping strategies. By optimizing switch-to-switch communication, ELDP drastically reduces the overhead associated with topology discovery. Its innovative mechanisms not only improve efficiency but also diminish unnecessary communication between switches and the controller.

(ii) ELDP provides practical implementations using the RYU controller and Mininet emulator, validating its effectiveness in real-world SDN settings. This holistic approach, encompasses efficiency, reduced overhead, scalability, security measures, and practical implementations.

The remainder of the paper is structured as follows: Section 2 presents the related work, Section 3 outlines the proposed approach, Section 4 discusses the simulation results, and Section 5 highlights future work and concludes the paper.

2. Related Works

This section gives general information on link discovery with a particular emphasis on the attempts made by different researchers to find linkages between switches with fewer messages. Collectively, the researchers mainly focused on lessening the difficulty in the controller application as it involves a continual operation which places a heavy burden on it. The authors [12] modified the OpenFlow Discovery Protocol (OFDP) and proposed OFDPv2, where the controller sends a packet_out message to each switch, instructing it to broadcast the message to its ports. In their subsequent work [13], they focused on reducing the Ternary Content Addressable Memory (TCAM) of OpenFlow. They introduced a ForCES-based approach that enhanced the switch-level computing capacity of LLDP, allowing the controller to periodically gather topology information from the switch [14]. Furthermore, they re-evaluated and re-implemented the OFDPv2 versions, only sending the OFTP_PACKET_OUT message to each switch and examining potential re-transmission activities if packets were lost between the controller and OpenFlow Switch [15].

Another proposed improvement is OFDPx, an enhanced version of OFDP, which creates multiple pathways through the network view and utilizes a packet probe to discover links along the same path [16]. The lm-OFDP approach employed rules for the discovery process. High-priority rules assigned the output port and configured it on the supporting switch. Medium-priority rules indicated that specific fields in LLDP packets reaching switches would be

modified, including the field description and the MAC address of the recipient port, before being returned to the same port. Low-priority rules stipulated that all LLDP packets received by the switch should be sent to the controller [17]. The authors [18] presented an approach that delegated functions for sending and receiving LLDP frames between nearby switches and reporting links to the controller's topology manager. This approach would reduce the resources needed for topology discovery and rerouting while preventing resource conflicts. proposed ICLF, a bidirectional link discovery protocol involving OpenFlow and legacy switches. Regarding link discovery involving OpenFlow switches and legacy switches, [19] proposed ICLF, a bidirectional link discovery approach. In this approach, the controller sends a packet_out message to the legacy switch, which then transmits it across the network. To minimize the packets utilized in the topology discovery process, SLDP [11] proposed a lightweight link discovery solution using an eligible port list to avoid broadcasting a Packet-out message to the ports connected to the host. In a related study, ESLD [20] employed a port classification strategy to prevent unnecessary packets from entering the network.

3. Enhanced Secure and Lightweight Discovery Protocol

The controller's responsibilities extend beyond link discovery and encompass diverse network administration tasks, managing hosts, links, and flows. However, the excessive use of packets for topology discovery negatively impacts the SDN performance. At the heart of the ELDP protocol lie two fundamental components: the Short Switch Chase ID (SSCID) and an innovative mapping strategy. SSCID introduces a unique identifier for switches, departing from the conventional reliance on MAC addresses. This innovation significantly enhances switch communication efficiency and strengthens security measures by reducing vulnerability to MAC spoofing attacks. The mapping strategy, intricately woven into ELDP, enables switches to establish intelligent connections with adjacent switches, fostering a cohesive and adaptive network framework. Its primary objectives encompass the optimization of network performance, the fortification of security measures, and the facilitation of a more adaptive and scalable network infrastructure. By reducing packet overhead, enhancing security through vulnerability testing, and fostering dynamic switch interactions, ELDP aspires to alleviate the inefficiencies and vulnerabilities prevalent in existing SDN link discovery protocols. The subsequent subsections provide a detailed description of the algorithm employed in ELDP and information about the emulation environment.

3.1. ELDP Packet Frame

The design of a novel probe frame structure for ELDP aimed at enhancing security, as illustrated in Figure 1. ELDP consists of a Packet header, which includes the Destination MAC address, Source MAC address, and Ethertype, and a Payload containing Check TLV, Switch ID, and Port ID. The construction of the frame is driven by the Check TLV, which incorporates a randomized sequence number, complementing the randomized source MAC address to significantly enhance security measures.

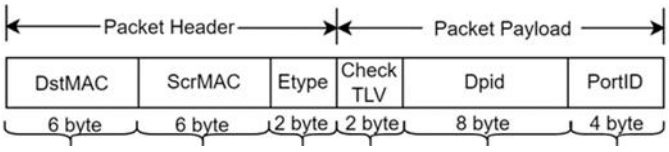

Figure 1: ELTD Packet Structure

3.2. Proposed ELDP Protocol

The ELDP protocol as depicted in Algorithm 1, begins by initializing port status variables for switches, allowing for periodic checks and updates (sections 1&2). Notably, ELDP incorporates the utilization of Short Switch Chase ID (SSCID) and a unique mapping strategy to revolutionize the topology discovery process. At the outset, the controller generates randomized MAC addresses (RMAC) and SSCIDs, utilizing RMAC for packet generation and flow entry, while SSCID serves as a distinct identifier for switches (section 3). The controller constructs a packet frame (section 4) and installs flow entries and SSCIDs in every OpenFlow Switch (OFS) within the network (section 5). Upon sending a packet-out message, switches broadcast the frame to their active ports and update their port status accordingly (section 6). Each port undergoes vulnerability testing upon receiving a packet, and if deemed vulnerable, the transmitting port is removed to mitigate security risks (section 7). Importantly, switches establish mappings with adjacent switches, facilitating the exchange of SSCIDs and forwarding packet-in messages to the controller using SSCIDs and port numbers instead of MAC addresses (section 8). ELDP maintains a Topology Map Database, updating it upon receiving packet-in messages from OFS. The controller validates frame data within the database, allowing bidirectional link establishment between switches if not previously recorded (sections 9 & 10). The ELDP architecture is presented in Figure 2.

3.2.1. ELDP Protocol Illustration

Figure 3 illustrates the operation of a network topology comprising six OFS (S1, S2, S3, S4, S5, & S6) within ELDP. The controller initiates the process by transmitting a Packet to S1. Upon S1's reception of the Packet-out, it appends interface details and dispatches the frame through an outgoing interface while updating its port status from 0 to 1.

Subsequently, as S2 receives the packet via P3, it establishes connections between switches based on switch ports, as depicted in Fig. 1: (P2 of S1) links to (P4 of S4); (P3 of S1) connects to (P4 of S3); (P1 of S1) correlates with (P3 of S2). Post connection establishment, S2 encapsulates the ELDP frame and forwards it in a Packet format to the controller using SSCIDs with port numbers. Upon reception of the Packet, the controller scrutinizes the Topology_Map_Database (refer to Table 1) and identifies S2's links with other switches (S1, S3, S4, S5, and S6). This update occurs bidirectionally, meaning a single packet-in message from S2 enables the discovery of bi-directional links between S1 & S2, S1 & S3, and S1 & S4.

3.2.2. Topology Discovery Approach

The Topology discovery algorithm is presented in Algorithm 1 below:

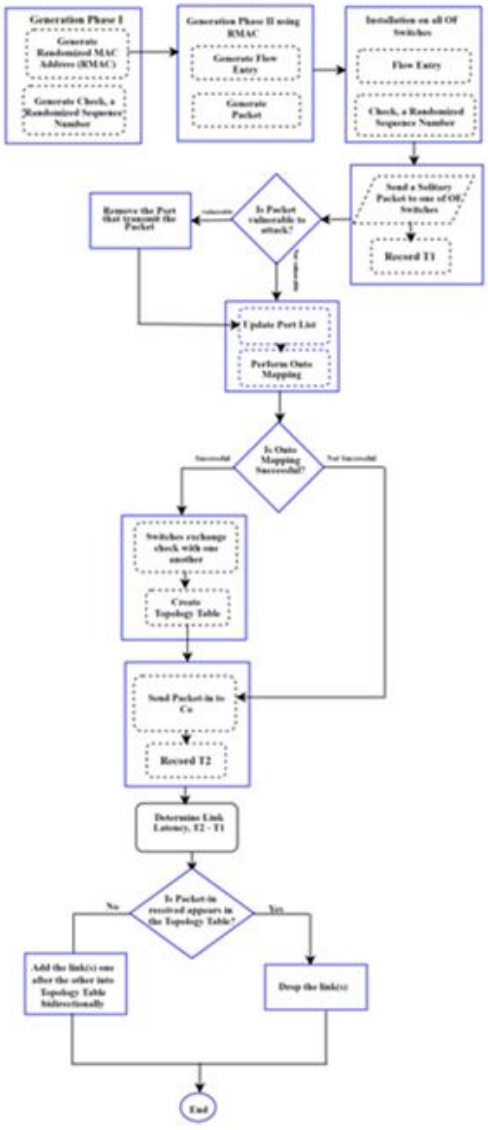

Figure 2: ELDP Architecture

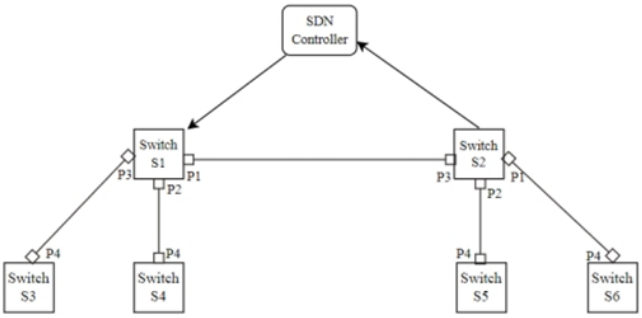

Figure 3: A network with 3 OFSs

Algorithm I: Topology Discovery Algorithm

Input:
co: controller
OFS: OpenFlow switch
sS: Set of Switches
PS: Switch's Ports
Pkt: packet frame
P_out: Packet Out
P_in: Packet In
RMAC: Randomized MAC Address
SSCID: Switch Short Chase Identity
F_E: Flow Entry
T1: Packet sending time
T2: Packet received time

Output: Topology Discovery
1. Switches maintain Port Status variables for each port at regular intervals 't'.
2. Initialize all switch ports at each 't' + Δt time interval.
3. The controller generates RMAC and SSCID.
4. Construct Pkt using RMAC.
5. Install both F_E and SSCID in every OpenFlow switch
6. For each switch (SS) in the network:
   (i) The controller sends a P_out to any OpenFlow switch.
   (ii) Record T1.
   (iii) Switch broadcasts Pkt to its ports.
   (iv) For each port PSi of switch SSk:
   Set PSi status = 1.
   End loop.
7. Upon receipt of a switch on port PSk:
   (i) Test the Pkt.
   (ii) If Pkt is identified as attack(s):

Drop (Pkt).
Remove (PSk): Ports transmitting Pkt.
Else:
Update (port list).
Conduct onto-mapping with switch neighbors.
End loop.
8. If onto-mapping is successful:
(i) Exchange SSCID with one another and forward P_in to controller using SSCIDs and Port numbers.
(ii) Record T2.
End loop.
9. For each received P_in from the OpenFlow switch:
(i) If the link exists in the Topology_Map_Database:
Drop (link).
Else:
Add (links) bidirectionally into Topology_Map_Database one after the other.
End loop.
10. Return.

Table 1: Topology Map Table

	S1	S2	S3	S4	S5	S6
S1	-	P1, P3	P3, P4	P2, P4	-	-
S2	P3, P1	-	-	-	P2, P4	P1, P4
S3	P4, P3	-	-	-	-	-
S4	P4, P2	-	-	-	-	-
S5	-	P4, P2	-	-	-	-
S6	-	P4, P1	-	-	-	-

Note: - indicates no connection between the switches

3.2.3 Switch and Map Algorithm

The primary objective of the algorithm is to systematically traverse through all switches, establish their port relationships, and construct a comprehensive Topology Map Database that encapsulates the interconnectedness and link information between switches in the network. The generated database can then be utilized for various network management and optimization purposes, providing crucial insights into the network's topology and connectivity. Algorithm II presents the switch and Port Mapping.

Algorithm II: Switch and Port Mapping
Input:
Ss: Set of all switches
PS: Ports of a switch
Ssi: Each switch in Ss
Pij: Port of a switch in Ss
PSi: Each port in PS

Output: Topology Map Database
1. For each SSi in Ss:
(i) Establish connections between SSi and Pij.
(ii) Add the link of Ssi & Sj.
2. End loop.
3. Return.

This algorithm operates by iterating through each switch in the set of all switches (Ss). For every switch encountered, it establishes connections between that switch and its ports (Pij) while concurrently adding the link information between switches (Ssi & Sj) to generate a comprehensive Topology Map Database. Algorithm II facilitates the mapping of switch connections through the utilization of switch ports. For instance, in Figure 3, the connection between port 2 of switch 1 (P2 of S1) and port 3 of switch 2 (P3 of S2) is established, while also linking port 3 of switch 1 (P3 of S1) to port 2 of switch 3 (P2 of S3). Additionally, the connection from port 2 of switch 1 (P1 of S1) to port 4 of switch 4 (P4 of S4) is evident. This implies that S1 establishes connections with S2, S3, and S4, as outlined in Table 1.

3.3. Emulation Environment

ELDP is assessed through an evaluation conducted on the Mininet emulator, a platform facilitating the creation of OpenFlow switches (OFSs), link establishment, and virtual host deployment. The evaluation took place on an Ubuntu-based operating system. To set up the experiment, we employed the Ryu controller and OpenvSwitch, both implemented in Python as the controller and virtual software switches, respectively. Tables II, III, and IV provide comprehensive details regarding the software, hardware specifications, and the topology adopted for the proposed scheme.

Table II: Software Used

Software	Version
OpenvSwitch	2.9.2
Mininet	2.17.1
Python	3.10.4
Ryu	4.34
Ubuntu	22.04(64-bit)

Table III: Hardware Used

Hardware Component	Specifications
Hard Disk	237 GB
Processor	11th Gen Intel(R) Core (TM) i7-1165G7 @ 2.80GHz 1.69 GHz
RAM	16.0GB

Table IV: Topology Used for Network Emulation

Topology	Number of Switch	Number of Link	Number of Port	Number of Host
Tree_4_4 (depth 4 & fanout 4)	85	340	424	256

Tree_7_2 (depth 7 & fanout 2)	127	254	380	128
Tree fat	80	384	705	64

4. Results and Discussion

This section evaluates ELDP performance across three topologies number of packets (Packet in and Packet out), the discovery time, and the overhead related to constructing link discovery packets. The outcomes derived from the emulation are depicted in Figures IV to VI:

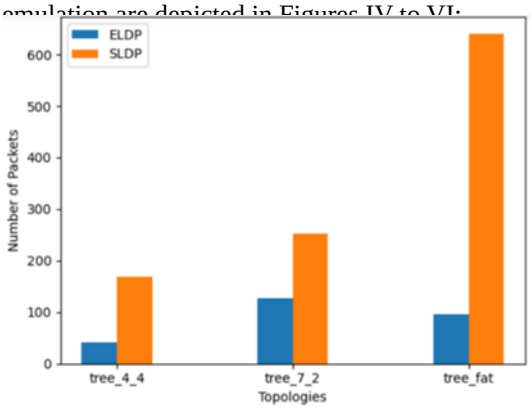

Figure IV: Packet utilized for link discovery.

Figure IV illustrates the utilization of packets in conducting link discovery across three topologies. Specifically, the Tree_4_4 topology exhibits a reduction rate of 75%, while the Tree_7_2 topology showcases a reduction rate of 50%. Remarkably, within the Tree_fat topology, the employed packets demonstrate an 87.5% reduction compared to SLDP. Equations i and ii detail the mathematical notation related to these packets.

$$\text{SLDP} = p - h \quad \text{(i)}$$

$$\text{ELDP} = (\text{p} - \text{h}))/(2 * \text{n}) \quad \text{(ii)}$$

Where p signifies the number of ports, h represents the number of hosts, and n represents the outcome derived from the mapping approach. It is essential to note that n must be at least 2.

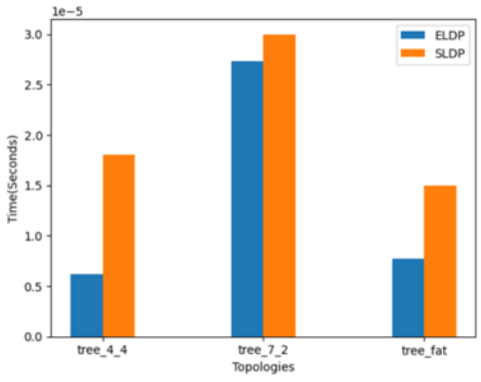

Figure V: Topology Discovery Time

Figure V demonstrates that within ELDP, the controller can expedite the construction of the entire topology within a shorter duration across the three topologies. This enhancement represents a significant improvement stemming from the modifications implemented in SLDP.

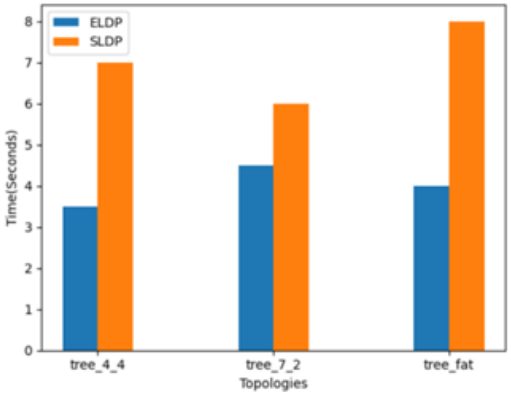

Figure VI: Link Discovery Packet Construction Overhead

Figure VI illustrates that ELDP conducts faster validation and parsing of incoming links compared to SLDP.

5. Conclusion

This study introduces ELDP as an advanced iteration surpassing SLDP, significantly improving the efficiency of link discovery by employing packets resilient against attacks. Employing a mapping strategy, the switch identifies neighboring switches connected to its ports and relays packet-in messages to the controller. Experimental outcomes showcase ELDP's remarkable capability, achieving a notable reduction of 50 to 87.5% in the requisite number of messages for link discovery across various network topologies. Significantly, ELDP surpasses SLDP in both efficiency and discovery time. However, it underscores the critical need for a robust security infrastructure to swiftly detect and mitigate diverse network attacks, particularly emphasizing defenses against the Man in the Middle attack.

6. Acknowledgements

We appreciate Universiti Putra Malaysia (UPM) for providing an enabling environment for research work. In addition, one of the authors of this paper would like to thank the Management of Federal Polytechnic, Ile Oluji,

Ondo State, Nigeria for their support to pursue his postgraduate studies.

7. References

[1] T. H. Obaida and H. A. Salman, "A novel method to find the best path in SDN using firefly algorithm," *J. Intell. Syst.*, vol. 31, no. 1, pp. 902–914, 2022, doi: 10.1515/jisys-2022-0063.

[2] S. Iqbal, K. N. Qureshi, F. Shoaib, A. Ahmad, and G. Jeon, "Minimize the delays in software defined network switch controller communication," *Concurr. Comput. Pract. Exp.*, vol. 34, no. 13, pp. 1–14, 2022, doi: 10.1002/cpe.5940.

[3] S. Saraswat, V. Agarwal, H. P. Gupta, R. Mishra, A. Gupta, and T. Dutta, "Challenges and solutions in Software Defined Networking: A survey," *J. Netw. Comput. Appl.*, vol. 141, no. April, pp. 23–58, 2019, doi: 10.1016/j.jnca.2019.04.020.

[4] C. S. Khin, A. T. Kyaw, M. M. Maw, and M. Zin Oo, "Reducing Packet-In Messages in OpenFlow Networks," *ECTI Trans. Electr. Eng. Electron. Commun.*, vol. 20, no. 1, pp. 1–9, 2022, doi: 10.37936/ecti-eec.2022201.244944.

[5] A. L. Stancu, S. Halunga, A. Vulpe, G. Suciu, O. Fratu, and E. C. Popovici, "A comparison between several Software Defined Networking controllers," *2015 12th Int. Conf. Telecommun. Mod. Satell. Cable Broadcast. Serv. TELSIKS 2015*, pp. 223–226, 2015, doi: 10.1109/TELSKS.2015.7357774.

[6] D. Kreutz, F. M. V. Ramos, P. E. Verissimo, C. E. Rothenberg, S. Azodolmolky, and S. Uhlig, "Software-defined networking: A comprehensive survey," *Proc. IEEE*, vol. 103, no. 1, pp. 14–76, 2015, doi: 10.1109/JPROC.2014.2371999.

[7] Y. Su, T. Peng, X. Zhong, and L. Zhang, "Matching model of flow table for networked big data," pp. 1–14, 2017, doi: 10.1007/978-981-13-6834-9_5.

[8] S. Khan, A. Gani, A. W. Abdul Wahab, M. Guizani, and M. K. Khan, "Topology Discovery in Software Defined Networks: Threats, Taxonomy, and State-of-the-Art," *IEEE Commun. Surv. Tutorials*, vol. 19, no. 1, pp. 303–324, 2017, doi: 10.1109/COMST.2016.2597193.

[9] R. Wazirali, R. Ahmad, and S. Alhiyari, "Sdn-openflow topology discovery: An overview of performance issues," *Appl. Sci.*, vol. 11, no. 15, 2021, doi: 10.3390/app11156999.

[10] L. Der Chou *et al.*, "Behavior Anomaly Detection in SDN Control Plane: A Case Study of Topology Discovery Attacks," *ICTC 2019 - 10th Int. Conf. ICT Converg. ICT Converg. Lead. Auton. Futur.*, pp. 357–362, 2019, doi: 10.1109/ICTC46691.2019.8939903.

[11] A. Nehra, M. Tripathi, M. S. Gaur, R. B. Battula, and C. Lal, "SLDP: A secure and lightweight link discovery protocol for software-defined networking," *Comput. Networks*, vol. 150, pp. 102–116, 2019, doi: 10.1016/j.comnet.2018.12.014.

[12] F. Pakzad, M. Portmann, W. L. Tan, and J. Indulska, "Efficient topology discovery in software-defined networks," *2014, 8th Int. Conf. Signal Process. Commun. Syst. ICSPCS 2014 - Proc.*, 2014, doi: 10.1109/ICSPCS.2014.7021050.

[13] F. Pakzad, M. Portmann, W. L. Tan, and J. Indulska, "Efficient topology discovery in OpenFlow-based Software Defined Networks," *Comput. Commun.*, vol. 77, pp. 52–61, 2016, doi: 10.1016/j.comcom.2015.09.013.

[14] G. Tarnaras, E. Haleplidis, and S. Denazis, "SDN and ForCES based optimal network topology discovery," *1st IEEE Conf. Netw. Softwarization Software-Defined Infrastructures Networks, Clouds, IoT Serv. NETSOFT 2015*, 2015, doi: 10.1109/NETSOFT.2015.7116181.

[15] D. Hasan and M. Othman, "Efficient Topology Discovery in Software Defined Networks: Revisited," *Procedia Comput. Sci.*, vol. 116, pp. 539–547, 2017, doi: 10.1016/j.procs.2017.10.051.

[16] H. Tong, X. Li, Z. Shi, and Y. Tian, "A Novel and Efficient Link Discovery Mechanism in SDN," *2020 IEEE 3rd Int. Conf. Electron. Commun. Eng. ICECE 2020*, pp. 97–101, 2020, doi: 10.1109/ICECE51594.2020.9353035.

[17] Y. Gu, D. Li, and J. Yu, "Im-OFDP : An Improved OpenFlow-based Topology Discovery Protocol for Software Defined Network," pp. 628–630, 2020.

[18] Y. C. Chang, H. T. Lin, H. M. Chu, and P. C. Wang, "Efficient Topology Discovery for Software-Defined Networks," *IEEE Trans. Netw. Serv. Manag.*, vol. 18, no. 2, pp. 1375–1388, 2021, doi: 10.1109/TNSM.2020.3047623.

[19] M. W. Hussain, K. H. K. Reddy, J. J. P. C. Rodrigues, and D. S. Roy, "An Indirect Controller-Legacy Switch Forwarding Scheme for Link Discovery in Hybrid SDN," *IEEE Syst. J.*, vol. 15, no. 2, pp. 3142–3149, 2021, doi: 10.1109/JSYST.2020.3011902.

[20] X. Zhao, L. Yao, and G. Wu, "ESLD: An efficient and secure link discovery scheme for software-defined networking," *Int. J. Commun. Syst.*, vol. 31, no. 10, pp. 1–18, 2018, doi: 10.1002/dac.3552.