



Topology Aware Evaluation of SDN Controller Performance and Placement Strategies in Star, Linear, and Ring Networks

B. V. Prasanthi¹, P. Chenna Reddy²

¹Vishnu Institute of Technology & Research Scholar, JNTUA College of Engineering, Anantapuramu, Andhra Pradesh, India

²Department of CSE, JNTUA College of Engineering, Anantapuramu, Andhra Pradesh, India
E-mails: prasanthibeera@gmail.com chennareddy.cse@jntua.ac.in

Abstract: *Software-defined networking allows centralized and flexible control of a network, but its performance largely depends on the controller and its location in the network. The current work presents an experimental study of the Software-Defined Networking (SDN) controller performance considering various placement methods using star, linear, and ring topologies. Placement methods, including centrality-based, greedy, clustering, metaheuristic, capacitated, and baseline approaches, are analyzed with four popular controllers: Beacon, RYU, OpenDaylight, and Open Network Operating System (ONOS). Experiments are conducted using Mininet-based emulation, and findings indicate that controller performance varies based on location within the network. Among the evaluated techniques, the partition-based approach is identified as balanced controller placement in linear and ring topologies, given stable latency and throughput values comparable to other optimized strategies. Overall, the lowest latency is produced by RYU, the maximum throughput by OpenDaylight, and the intermediate performance by ONOS and Beacon.*

Keywords: *Software-defined networking, Controller placement, Beacon, RYU, OpenDaylight, ONOS, Linear topology, Ring topology, Centrality, Clustering.*

1. Introduction

Software-Defined Networking (SDN) separation of control and data plane allows centralized management of network state and provides a global view of the network, which improves flexibility, scalability, and efficient decision-making in modern network environments [1, 2]. Traditional networking architectures bind the control plane and data plane within network devices closely, which makes network management complex and inflexible. However, centralized control in SDN introduces challenges such as scalability of controllers, inter-switch latency, and fault tolerance. In practice, controllers must be deployed at specific locations in the network. Improper controller placement may increase the path length between

switches and controllers, leading to control-plane congestion, higher communication delays, and degraded network performance, thereby giving rise to the Controller Placement Problem (CPP). Most prior studies on CPP focus either on algorithmic aspects or on single-controller platforms, without offering a comparative view across multiple controllers and topologies [3, 4]. Therefore, identifying optimal controller placement strategies is an important research problem in SDN environments. The main contributions of this paper include a comprehensive comparative evaluation of four widely used SDN controllers, namely Beacon, RYU, Open Network Operating System (ONOS), and OpenDaylight. The study conducts an extensive experimental analysis of eleven controller placement strategies across three representative network topologies: star, linear, and ring. Furthermore, it systematically investigates the impact of topology structure on key performance metrics, including controller-to-switch latency and data-plane throughput. Based on the obtained results, the paper identifies effective controller placement strategies and demonstrates that clustering-based approaches are particularly suitable for enhancing performance in linear and ring topologies. The rest of the paper is organized as follows. Section 2 reviews related work on SDN controllers and controller placement. Section 3 describes the system model, including the controllers, topologies, and placement techniques. Section 4 explains the experimental methodology. Section 5 analyses the results for star, linear, and ring topologies. Section 6 concludes the paper and outlines future research directions.

2. Related work

Early SDN deployments relied on single centralized controllers, and placement was often treated as a practical configuration issue rather than an optimization problem [5]. As networks grew, researchers began to study CPP formally, modelling it as facilities location, k-median, or k-center problems, often under latency and resilience constraints. Survey papers on CPP categorize approaches into centrality-based heuristics, greedy methods, clustering algorithms, and metaheuristic optimization, with additional attention to capacity-constrained and fault-tolerant placement [6]. In parallel, a variety of SDN controllers have been developed. Beacon, written in Java, is designed for high-performance event processing. RYU implemented in Python emphasizes programmability and ease of experimentation. OpenDaylight is a modular, OSGi-based platform with support for multiple southbound protocols, and ONOS offers a distributed, logically centralized control plane intended for carrier-grade networks [7]. Comparative evaluations of controllers typically focus on throughput, flow setup time, or scalability under synthetic traffic, but rarely consider how controller placement interacts with controller architecture [8]. Existing CPP studies typically evaluate a single controller on synthetic topologies, which limits their applicability to real deployments in which controller choice and topology structure are intertwined [9]. Several studies have identified the controller placement problem in SDN networks. The controller placement problem of Heller, Sherwood and McKeown

[10] was designed as a k -median optimization problem, which aimed at reducing the latency among switches and controllers. Sahakyan and Astsatryan [11] came up with latency and reliability-conscious placement algorithms in distributed SDN control planes. Badotra and Singh [12] examined fault-tolerance control decision-making in order to enhance network resilience. Recent studies evaluate controller performance using Mininet-based simulation environments and compare controllers like ONOS, OpenDaylight, and RYU to various workloads. The current study is unique and valuable as it analyses several controllers and placement in common strategies on three topologies.

3. System model and placement strategies

In this work, four widely used SDN controllers are evaluated to gain insight into how controllers perform under various architectures and to find the performance effect under different placement strategies. Beacon is designed for high performance, modularity, and developer productivity. RYU is a lightweight Python-based controller designed for ease of development and experimentation. OpenDayLight (ODL) provides a modular service-oriented framework protocol support. ONOS depends on distributed platforms that provide high consistency and fault-tolerance [13]. To provide uniformity in our assessment, all controllers were deployed as logically centralized instances in Mininet; depending on the placement scenario, either a single or dual-controller configuration is used. The analysis takes into account three typical network topologies: a star, a linear topology, and a ring topology, a canonical design in metro-Ethernet and transport systems that rely on cyclic connectivity for resilience [14]. Each topology is instantiated with 50 switches, and controllers are positioned on selected nodes according to the placement strategy under study. 50 switches were selected to represent a moderately sized network that is large enough to demonstrate controller placement effects while remaining computationally manageable in the Mininet emulation environment. Networks of this size are commonly used in SDN research for evaluating scalability and latency characteristics of controller placement strategies. A single controller is used in the star-like configuration, whereas the linear and ring settings employ two controllers to better capture load distribution and path diversity effects [15]. We evaluated eleven controller placement strategies, including centrality-based methods (center, degree, betweenness); each method is tested for $k=1$ and $k=2$, where k denotes the number of controllers, greedy methods (k -center-greedy, k -median-greedy, and k -cover-radius), clustering methods (partition_kmeans), metaheuristic methods (Genetic Algorithm), constrained methods (capacitated placement), and baseline methods (random and uniform_k). Centrality-based methods select nodes with high structural importance in the connectivity graph [16]. Greedy approaches iteratively improve a performance objective such as maximum or average distance [17, 18]. Partition_kmeans clusters nodes and places controllers at cluster centroids. The genetic algorithm explores a broader search space by mutating and recombining candidate placements [19]. The capacitated method enforces per-controller capacity limits, while random and uniform_k serve as

baseline approaches [20]. Centrality-based SDN controller placement in various topologies is shown in Fig. 1.

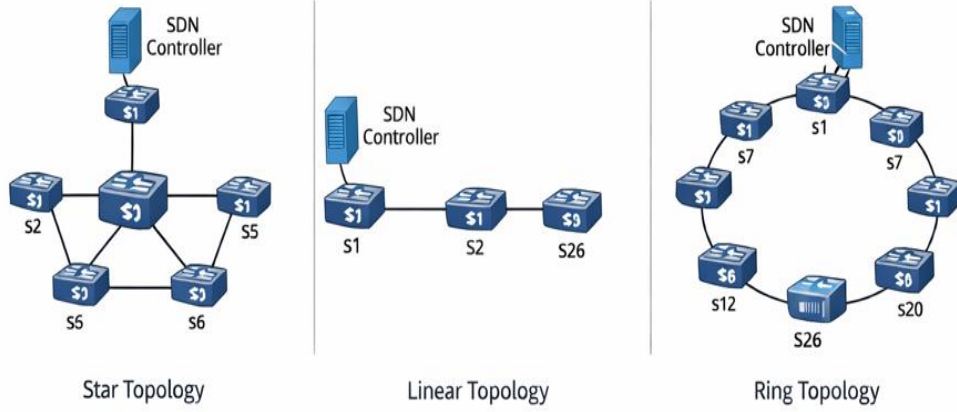


Fig. 1. Centrality-based SDN controller placement in different network topologies using Mininet simulation

4. Experimental methodology

All experiments were conducted using the Mininet network emulator running on Ubuntu 22.04. The host machine was equipped with an Intel Core i7 processor, 16 GB RAM, and Python 3.10. The Mininet emulator was used to simulate the data-plane switches and links. Each controller runs as a separate process and communicates with the emulated switches using the OpenFlow protocol. Each link in the topology was configured with a bandwidth of 100 Mbps and a propagation delay of 5 ms. The following metrics are considered: average controller-to-switch latency (avg_latency_ms), 95th-percentile latency (p95_latency_ms), average round-trip time for control messages (avg_RTT_ms), and average data-plane throughput (avg_throughput_MBps). In every experiment, three network topologies, which are star, linear, and ring, were instantiated in Mininet, which comprised 50 OpenFlow switches. Four SDN controllers (Beacon, ONOS, Open Daylight, and RYU) were installed as logically centralized devices, and there was one controller for a star topology ($k = 1$) that was used, and for linear and ring topologies ($k = 2$). The placement of attachment points in controllers was determined by the heuristic placement of controllers as center, degree, betweenness, partition_kmeans, uniform_k, and capacitated approaches, which were used only to choose the switching of the candidates and not to maximize performance metrics directly. Traffic was created with iPerf between data planes, host pairs were used, and the controller-to-switch behavior was measured, as well as control-plane behavior in terms of delays and round-trip times. All experiments were repeated in order to achieve stability, and the reported values are the averaged steady-state values. In this study, each topology contains 50 OpenFlow switches, and the architecture is represented in Fig. 2.

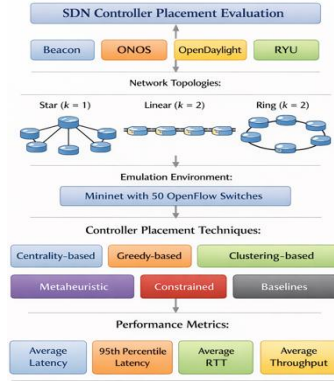


Fig. 2. Proposed architecture

4.1. Performance metrics

The SDN-enabled network is represented as an undirected graph $G = (V, E)$, where $V = \{v_1, v_2, \dots, v_n\}$ denotes the set of switches and $E \subseteq V \times V$ represents the physical links between them. Controller placement is denoted by selecting a subset of nodes. $C \subseteq V$, with $|C| = k$, where each controller is logically centralized and manages a disjoint subset of switches [21, 22].

Each switch $v \in V$ is associated with exactly one controller $c(v) \in C$, determined by the selected placement technique. The controller-switch communication delay is modeled as the shortest path distance in the topology graph [23]. Let $d(v, c(v))$ denote the one-way propagation delay between the switch v and its assigned controller. The average controller-to-switch latency is computed as

$$(1) \quad \text{AvgLatency} = \frac{1}{|V|} \sum_{v \in V} d(v, c(v)).$$

To capture tail latency effects and account for worst-case control-plane behavior, the 95th percentile latency is calculated as shown in the equation

$$(2) \quad P95\text{Latency} = \text{percentile}_{95}(\{d(v, c(v)) \mid v \in V\}).$$

Here, the Round-Trip Time (RTT) of control messages is estimated to be twice the one-way latency between controller and switch. The average RTT is defined as

$$(3) \quad \text{AvgRTT} = \frac{2}{|V|} \sum_{v \in V} d(v, c(v)).$$

Data-plane performance is evaluated in terms of throughput, measured as the average rate of successfully delivered data over active flows. If T_i denotes the throughput observed for the flow i , and F is the total number of flows, the average throughput is expressed as

$$(4) \quad \text{AvgThroughput} = \frac{1}{F} \sum_{i=1}^F T_i.$$

To enable fair comparison across controllers and topologies, the effective bandwidth utilization is computed as the ratio between the observed throughput and the nominal link capacity shown in the equation

$$(5) \quad \text{EffectiveBW} = \frac{\text{AvgThroughput}}{\text{LinkCapacity}}.$$

Controller load is characterized by the number of switches assigned to a controller. Let $S_c \subseteq V$ denote the set of switches managed by the controller c . The load on the controller c is given by the equation

$$(6) \quad L(c) = |S_c|.$$

Given a predefined controller capacity threshold Γ_c overload is identified when $L(c) > \Gamma_c$. The placement techniques evaluated in this work implicitly approximate the objective of minimizing average controller-to-switch latency under a fixed number of controllers, denoted in the equation

$$(7) \quad \min_{C \subseteq V} \text{AvgLatency} \text{ subject to } |C| = k.$$

4.2. Experimental design

For each topology, all eleven placement strategies are evaluated for each of the four controllers. The most representative placements and top-performing strategies are presented, and we also analyzed grouped tables that aggregate performance across all techniques. The label s1 represents the first switch node in the Mininet topology. In the star topology, s1 corresponds to the central switch connected to all other peripheral switches.

Table 1. Stars grouped best by technique

Controller	Technique	k	Avg latency, ms	Avg throughput, MBps	Placement
Beacon	betweenness	1	8.78	13.84055	s1
Beacon	betweenness	2	8.78	13.84055	s1
Beacon	capacitated	1	8.78	13.84055	s1
Beacon	capacitated	2	8.78	13.84055	s1
Beacon	center	1	8.78	13.84055	s1
Beacon	center	2	8.78	13.84055	s1
Beacon	degree	1	8.78	13.84055	s1
Beacon	degree	2	8.78	13.84055	s1
Beacon	genetic	1	8.78	13.84055	s1
Beacon	genetic	2	8.78	13.84055	s1
Beacon	kcenter greedy	1	8.78	13.84055	s1
Beacon	kcenter greedy	2	8.78	13.84055	s1
Beacon	kcover radius	1	8.78	13.84055	s1
Beacon	kcover radius	2	8.78	13.84055	s1, s1
Beacon	kmedian greedy	1	8.78	13.84055	s1
Beacon	kmedian greedy	2	8.78	13.84055	s1
Beacon	partition kmeans	1	8.78	13.84055	s1
Beacon	partition kmeans	2	8.78	13.84055	s1
Beacon	random	1	8.78	13.84055	s1
Beacon	random	2	8.78	13.84055	s1
Beacon	uniform k	1	8.78	13.84055	s1
Beacon	uniform k	2	8.78	13.84055	s1
ONOS	betweenness	1	4.294	27.934	s1
ONOS	betweenness	2	4.294	27.934	s1
ONOS	capacitated	1	4.294	27.934	s1
ONOS	capacitated	2	4.294	27.934	s1
ONOS	center	1	4.294	27.934	s1
ONOS	center	2	4.294	27.934	s1
ONOS	degree	1	4.294	27.934	s1
ONOS	degree	2	4.294	27.934	s1
ONOS	genetic	1	4.294	27.934	s1
ONOS	genetic	2	4.294	27.934	s1
ONOS	kcenter greedy	1	4.294	27.934	s1
ONOS	kcenter greedy	2	4.294	27.934	s1
ONOS	kcover radius	1	4.294	27.934	s1
ONOS	kcover radius	2	4.294	27.934	s1, s1
ONOS	kmedian greedy	1	4.294	27.934	s1
ONOS	kmedian greedy	2	4.294	27.934	s1

Table 1 (continued)

Controller	Technique	k	Avg latency, ms	Avg throughput, MBps	Placement
ONOS	partition kmeans	1	4.294	27.934	s1
ONOS	partition kmeans	2	4.294	27.934	s1
ONOS	random	1	4.294	27.934	s1
ONOS	random	2	4.294	27.934	s1
ONOS	uniform k	1	4.294	27.934	s1
ONOS	uniform k	2	4.294	27.934	s1
OpenDaylight	betweenness	1	8.88	55.36937	s1
OpenDaylight	betweenness	2	8.88	55.36937	s1
OpenDaylight	capacitated	1	8.88	55.36937	s1
OpenDaylight	capacitated	2	8.88	55.36937	s1
OpenDaylight	center	1	8.88	55.36937	s1
OpenDaylight	center	2	8.88	55.36937	s1
OpenDaylight	degree	1	8.88	55.36937	s1
OpenDaylight	degree	2	8.88	55.36937	s1
OpenDaylight	genetic	1	8.88	55.36937	s1
OpenDaylight	genetic	2	8.88	55.36937	s1
OpenDaylight	kcenter greedy	1	8.88	55.36937	s1
OpenDaylight	kcenter greedy	2	8.88	55.36937	s1
OpenDaylight	kcover radius	1	8.88	55.36937	s1
OpenDaylight	kcover radius	2	8.88	55.36937	s1, s1
OpenDaylight	kmedian greedy	1	8.88	55.36937	s1
OpenDaylight	kmedian greedy	2	8.88	55.36937	s1
OpenDaylight	partition kmeans	1	8.88	55.36937	s1
OpenDaylight	partition kmeans	2	8.88	55.36937	s1
OpenDaylight	random	1	8.88	55.36937	s1
OpenDaylight	random	2	8.88	55.36937	s1
OpenDaylight	uniform k	1	8.88	55.36937	s1
OpenDaylight	uniform k	2	8.88	55.36937	s1
RYU	betweenness	1	0.389	50.81645	s1
RYU	betweenness	2	0.389	50.81645	s1
RYU	capacitated	1	0.389	50.81645	s1
RYU	capacitated	2	0.389	50.81645	s1
RYU	center	1	0.389	50.81645	s1
RYU	center	2	0.389	50.81645	s1
RYU	degree	1	0.389	50.81645	s1
RYU	degree	2	0.389	50.81645	s1
RYU	genetic	1	0.389	50.81645	s1
RYU	genetic	2	0.389	50.81645	s1
RYU	kcenter greedy	1	0.389	50.81645	s1
RYU	kcenter greedy	2	0.389	50.81645	s1
RYU	kcover radius	1	0.389	50.81645	s1
RYU	kcover radius	2	0.389	50.81645	s1, s1
RYU	kmedian greedy	1	0.389	50.81645	s1
RYU	kmedian greedy	2	0.389	50.81645	s1
RYU	partition kmeans	1	0.389	50.81645	s1
RYU	partition kmeans	2	0.389	50.81645	s1
RYU	random	1	0.389	50.81645	s1
RYU	random	2	0.389	50.81645	s1
RYU	uniform k	1	0.389	50.81645	s1
RYU	uniform k	2	0.389	50.81645	s1

Table 1 presents the performance of different controller placement techniques under a star-controller deployment across four SDN controllers. An important outcome of this evaluation is that all placement strategies, ranging from centrality-based and greedy methods to clustering, constrained, and baseline approaches, consistently select the same switch (s1) as the optimal controller location for every

controller and for both values of k . This convergence indicates that the star topology is structurally dominated by a central node, making the controller placement decision insensitive to the choice of heuristic. Consequently, latency and throughput values remain identical across all placement strategies. Among the evaluated RYU has the lowest average latency (0.389 ms), which is also indicative of its light-weighted event-driven design. The maximum average throughput (55.37 MBps) is provided by OpenDaylight, taking advantage of its data-plane optimized and modular design. ONOS exhibits moderate latency (4.294 ms) and competitive throughput (27.93 MBps), which is a balancing factor in star logical instance mode.

Table 2. Linear grouped best by technique

Controller	Technique	k	Placement	Avg latency, ms	Avg throughput, MBps
Beacon	Betweenness	1	s25	21.28	6.554298
Beacon	Betweenness	2	s25, s26	20.78	6.759161
Beacon	Capacitated	1	s26	21.28	6.554298
Beacon	Capacitated	2	s26, s9	16.94	8.186741
Beacon	Center	1	s25	21.28	6.554298
Beacon	Center	2	s25, s26	20.78	6.759161
Beacon	Degree	1	s2	32.32	4.947722
Beacon	Degree	2	s2, s3	31.36	5.181729
Beacon	Genetic	1	s25	21.28	6.554298
Beacon	Genetic	2	s37, s15	15.12	8.590916
Beacon	kcenter greedy	1	s3	31.40	5.130373
Beacon	kcenter greedy	2	s26, s1	17.90	7.688499
Beacon	kcover radius	1	s16	23.08	6.354628
Beacon	kcover radius	2	s16, s32	15.88	8.385196
Beacon	kmedian greedy	1	s26	21.28	6.554298
Beacon	kmedian greedy	2	s26, s9	16.94	8.186741
Beacon	partition kmeans	1	s25	21.28	6.554298
Beacon	partition kmeans	2	s13, s38	15.02	8.617837
Beacon	Random	1	s26	21.28	6.554298
Beacon	Random	2	s16, s40	15.16	8.581531
Beacon	uniform k	1	s26	21.28	6.554298
Beacon	uniform k	2	s1, s26	17.90	7.688499
ONOS	Betweenness	1	s25	16.794	9.2339
ONOS	Betweenness	2	s25, s26	16.294	9.7107
ONOS	Capacitated	1	s26	16.794	9.2339
ONOS	Capacitated	2	s26, s9	12.454	12.4675
ONOS	Center	1	s25	16.794	9.2339
ONOS	Center	2	s25, s26	16.294	9.7107
ONOS	Degree	1	s2	27.834	6.7627
ONOS	Degree	2	s2, s3	26.874	7.2756
ONOS	Genetic	1	s25	16.794	9.2339
ONOS	genetic	2	s39, s14	10.554	13.1229
ONOS	kcenter greedy	1	s42	22.234	8.3195
ONOS	kcenter greedy	2	s45, s1	14.274	10.9847
ONOS	kcover radius	1	s16	18.594	8.9681
ONOS	kcover radius	2	s16, s32	11.394	12.7584
ONOS	kmedian greedy	1	s26	16.794	9.2339
ONOS	kmedian greedy	2	s26, s9	12.454	12.4675
ONOS	partition kmeans	1	s25	16.794	9.2339
ONOS	partition kmeans	2	s13, s38	10.534	13.1314
ONOS	random	1	s26	16.794	9.2339
ONOS	random	2	s12, s40	10.634	13.0887
ONOS	uniform k	1	s26	16.794	9.2339
ONOS	uniform k	2	s1, s26	13.414	11.4211
OpenDaylight	betweenness	1	s25	21.38	26.35417

Table 2 (continued)

Controller	Technique	k	Placement	Avg latency, ms	Avg throughput, MBps
OpenDaylight	betweenness	2	s25, s26	20.88	27.17131
OpenDaylight	capacitated	1	s26	21.38	26.35417
OpenDaylight	capacitated	2	s26, s9	17.04	32.87750
OpenDaylight	center	1	s25	21.38	26.35417
OpenDaylight	center	2	s25, s26	20.88	27.17131
OpenDaylight	degree	1	s2	32.42	19.90796
OpenDaylight	degree	2	s2, s3	31.46	20.84246
OpenDaylight	genetic	1	s26	21.38	26.35417
OpenDaylight	genetic	2	s36, s9	15.34	34.34810
OpenDaylight	kcenter greedy	1	s50	33.38	19.08255
OpenDaylight	kcenter greedy	2	s13, s50	17.28	31.37264
OpenDaylight	kcover radius	1	s16	23.18	25.55125
OpenDaylight	kcover radius	2	s16, s32	15.98	33.67410
OpenDaylight	kmedian greedy	1	s26	21.38	26.35417
OpenDaylight	kmedian greedy	2	s26, s9	17.04	32.87750
OpenDaylight	partition kmeans	1	s25	21.38	26.35417
OpenDaylight	partition kmeans	2	s13, s38	15.12	34.60644
OpenDaylight	random	1	s25	21.38	26.35417
OpenDaylight	random	2	s37, s12	15.14	34.58492
OpenDaylight	uniform k	1	s26	21.38	26.35417
OpenDaylight	uniform k	2	s1, s26	18.00	30.88888
RYU	betweenness	1	s25	12.889	3.631116
RYU	betweenness	2	s25, s26	12.389	4.631874
RYU	capacitated	1	s26	12.889	3.631116
RYU	capacitated	2	s26, s9	8.549	6.010457
RYU	center	1	s25	12.889	3.631116
RYU	center	2	s25, s26	12.389	4.631874
RYU	degree	1	s2	23.929	2.867460
RYU	degree	2	s2, s3	22.969	3.875619
RYU	genetic	1	s25	12.889	3.631116
RYU	genetic	2	s13, s38	6.629	6.190731
RYU	kcenter greedy	1	s43	19.009	3.376668
RYU	kcenter greedy	2	s50, s1	12.389	4.631874
RYU	kcover radius	1	s16	14.689	3.571665
RYU	kcover radius	2	s16, s32	7.489	6.083775
RYU	kmedian greedy	1	s26	12.889	3.631116
RYU	kmedian greedy	2	s26, s9	8.549	6.010457
RYU	partition kmeans	1	s25	12.889	3.631116
RYU	partition kmeans	2	s13, s38	6.629	6.190731
RYU	random	1	s25	12.889	3.631116
RYU	random	2	s11, s37	6.689	6.183162
RYU	uniform k	1	s26	12.889	3.631116
RYU	uniform k	2	s1, s26	9.509	5.411302

Table 2 shows that the impact of placement of controllers is strongly influential on performance in the linear topology, with long routes in the traffic and the proximity of controllers. Across all two-controller ($k = 2$) configurations have a steady and consistent advantage over star controller deployments in both low latency and higher throughput. Optimization-based methods are used to reduce latency with partition_kmeans, genetic, and kcover_radius (down to 15.02 ms) as an enhancement of throughput to over 8.5 MBps. This is also the same with the ONOS controller, though degree-based placement is still underperforming, while partition_kmeans, genetic, and capacitated placements get the lowest value of latency. The lowest throughput (below 13 MBps) and the maximum throughput (10.5-12.4 ms).

Table 3. Ring grouped best by technique

Controller	Technique	k	Avg latency, ms	Avg throughput, MBps	Placement
Beacon	betweenness	1	21.28	6.554298	s1
Beacon	betweenness	2	20.78	6.759161	s1, s2
Beacon	capacitated	1	21.28	6.554298	s16
Beacon	capacitated	2	15.02	8.617837	s16, s41
Beacon	center	1	21.28	6.554298	s1
Beacon	center	2	20.78	6.759161	s1, s2
Beacon	degree	1	21.28	6.554298	s1
Beacon	degree	2	20.78	6.759161	s1, s2
Beacon	genetic	1	21.28	6.554298	s3
Beacon	genetic	2	15.02	8.617837	s41, s16
Beacon	kcenter greedy	1	21.28	6.554298	s25
Beacon	kcenter greedy	2	15.02	8.617837	s38, s13
Beacon	kcover radius	1	21.28	6.554298	s16
Beacon	kcover radius	2	15.84	8.392521	s16, s32
Beacon	kmedian greedy	1	21.28	6.554298	s16
Beacon	kmedian greedy	2	15.02	8.617837	s16, s41
Beacon	partition kmeans	1	21.28	6.554298	s25
Beacon	partition kmeans	2	15.02	8.617837	s13, s38
Beacon	random	1	21.28	6.554298	s5
Beacon	random	2	15.02	8.617837	s28, s3
Beacon	uniform k	1	21.28	6.554298	s26
Beacon	uniform k	2	15.02	8.617837	s1, s26
ONOS	betweenness	1	16.794	9.2339	s1
ONOS	betweenness	2	16.294	9.7107	s1, s2
ONOS	capacitated	1	16.794	9.2339	s16
ONOS	capacitated	2	10.534	13.1314	s16, s41
ONOS	center	1	16.794	9.2339	s1
ONOS	center	2	16.294	9.7107	s1, s2
ONOS	degree	1	16.794	9.2339	s1
ONOS	degree	2	16.294	9.7107	s1, s2
ONOS	genetic	1	16.794	9.2339	s45
ONOS	genetic	2	10.534	13.1314	s39, s14
ONOS	kcenter greedy	1	16.794	9.2339	s11
ONOS	kcenter greedy	2	10.534	13.1314	s44, s19
ONOS	kcover radius	1	16.794	9.2339	s16
ONOS	kcover radius	2	11.354	12.769	s16, s32
ONOS	kmedian greedy	1	16.794	9.2339	s16
ONOS	kmedian greedy	2	10.534	13.1314	s16, s41
ONOS	partition kmeans	1	16.794	9.2339	s25
ONOS	partition kmeans	2	10.534	13.1314	s13, s38
ONOS	random	1	16.794	9.2339	s11
ONOS	random	2	10.534	13.1314	s45, s20
ONOS	uniform k	1	16.794	9.2339	s26
ONOS	uniform k	2	10.534	13.1314	s1, s26
OpenDaylight	betweenness	1	21.38	26.35417	s1
OpenDaylight	betweenness	2	20.88	27.17131	s1, s2
OpenDaylight	capacitated	1	21.38	26.35417	s16
OpenDaylight	capacitated	2	15.12	34.60644	s16, s41
OpenDaylight	center	1	21.38	26.35417	s1
OpenDaylight	center	2	20.88	27.17131	s1, s2
OpenDaylight	degree	1	21.38	26.35417	s1
OpenDaylight	degree	2	20.88	27.17131	s1, s2
OpenDaylight	genetic	1	21.38	26.35417	s19
OpenDaylight	genetic	2	15.12	34.60644	s6, s31
OpenDaylight	kcenter greedy	1	21.38	26.35417	s29
OpenDaylight	kcenter greedy	2	15.12	34.60644	s26, s1
OpenDaylight	kcover radius	1	21.38	26.35417	s16

Table 3 (continued)

Controller	Technique	k	Avg latency, ms	Avg throughput, MBps	Placement
OpenDaylight	kcover radius	2	15.94	33.70350	s16, s32
OpenDaylight	kmedian_greedy	1	21.38	26.35417	s16
OpenDaylight	kmedian_greedy	2	15.12	34.60644	s16, s41
OpenDaylight	partition_kmeans	1	21.38	26.35417	s25
OpenDaylight	partition_kmeans	2	15.12	34.60644	s13, s38
OpenDaylight	random	1	21.38	26.35417	s16
OpenDaylight	random	2	15.12	34.60644	s42, s17
OpenDaylight	uniform_k	1	21.38	26.35417	s26
OpenDaylight	uniform_k	2	15.12	34.60644	s1, s26
RYU	betweenness	1	12.889	3.631116	s1
RYU	betweenness	2	12.389	4.631874	s1, s2
RYU	capacitated	1	12.889	3.631116	s16
RYU	capacitated	2	6.629	6.190731	s16, s41
RYU	center	1	12.889	3.631116	s1
RYU	center	2	12.389	4.631874	s1, s2
RYU	degree	1	12.889	3.631116	s1
RYU	degree	2	12.389	4.631874	s1, s2
RYU	genetic	1	12.889	3.631116	s46
RYU	genetic	2	6.629	6.190731	s38, s13
RYU	kcenter_greedy	1	12.889	3.631116	s29
RYU	kcenter_greedy	2	6.629	6.190731	s4, s29
RYU	kcover radius	1	12.889	3.631116	s16
RYU	kcover radius	2	7.449	6.086399	s16, s32
RYU	kmedian_greedy	1	12.889	3.631116	s16
RYU	kmedian_greedy	2	6.629	6.190731	s16, s41
RYU	partition_kmeans	1	12.889	3.631116	s25
RYU	partition_kmeans	2	6.629	6.190731	s13, s38
RYU	random	1	12.889	3.631116	s24
RYU	random	2	6.629	6.190731	s41, s16
RYU	uniform_k	1	12.889	3.631116	s26
RYU	uniform_k	2	6.629	6.190731	s1, s26

Table 3 shows the performance of controllers in various placement strategies of a ring topology. According to the results, both controller architectures have an impact on placement technique on latency and throughput when acting with one ($k = 1$) and two ($k = 2$) controllers. Centrality-based betweenness, center, and degree approaches tend to choose the neighboring or symmetric nodes. e.g. s1, s2, moderate performance gains between $k = 1$ and $k = 2$. Greater gains are shown by topology-aware strategies. In the case of Beacon, the methods like capacitated, genetic, kcenter_greedy, kmedian_greedy, and partition_kmeans, latency is reduced from 21.28 ms to approximately 15.02 ms, and the throughput is about 8.62 MBps. The same trend applied to ONOS, which shows a decrease in the latency to advanced placement methods, almost 10.53 ms, and raises the throughput to over 13 MBps. OpenDaylight has the best throughput of all the techniques, with a high value of over 34 MBps place under capacitated, genetic, and partition_kmeans. While its latency remains more than that of RYU, the throughput gains indicate the high scalability of OpenDaylight, making it suitable for data-intensive ring network environments. RYU, on the contrary, has the shortest latency, 6.63 ms, with genetic and partition_kmeans placement having less throughput, which is indicative of its lightweight architecture and the lack of parallel processing capability.

Table 4. Performance of the optimal controller-placement strategy for each SDN controller

Topo- logy	Controller	k	Technique	Placement	Avg latency, ms	P95 latency, ms	Avg RTT, ms	Avg throughput, MBps	Effective BW, Gbps	CU
star	Beacon	1	center	s1	8.78	8.78	17.56	13.84055	0.11072	{'s1': 50.0}
star	ONOS	1	center	s1	4.294	4.294	8.588	27.93400	0.00223	{'s1': 50.0}
star	OpenDaylight	1	center	s1	8.88	8.88	17.76	55.36937	0.44295	{'s1': 50.0}
star	RYU	1	center	s1	0.389	0.389	0.778	50.81645	0.40653	{'s1': 50.0}
linear	Beacon	2	partition_kmeans	s13, s38	15.02	20.78	30.04	8.61783	0.06894	{'s13': 25.0, 's38': 25.0}
linear	ONOS	2	partition_kmeans	s13, s38	10.534	16.294	21.06	13.1314	0.00105	{'s13': 25.0, 's38': 25.0}
linear	OpenDaylight	2	partition_kmeans	s13, s38	15.12	20.88	30.24	34.60644	0.27685	{'s13': 25.0, 's38': 25.0}
linear	RYU	2	partition_kmeans	s13, s38	6.629	12.389	13.25	6.190731	0.04952	{'s13': 25.0, 's38': 25.0}
ring	Beacon	2	uniform_k	s1, s26	15.02	20.78	30.04	8.617837	0.06894	{'s1': 25.0, 's26': 25.0}
ring	ONOS	2	uniform_k	s1, s26	10.534	16.294	21.06	13.1314	0.00105	{'s1': 25.0, 's26': 25.0}
ring	OpenDaylight	2	uniform_k	s1, s26	15.12	20.88	30.24	34.60644	0.27685	{'s1': 25.0, 's26': 25.0}
ring	RYU	2	uniform_k	s1, s26	6.629	12.389	13.25	6.190731	0.04952	{'s1': 25.0, 's26': 25.0}

Table 4 summarizes the best-performing controller placement configurations identified for each topology by selecting the most effective placement technique and controller combination. The results indicate strong differences in the control plane behavior of the topologies and controllers, and at the same time illustrate the impact of topology-aware placement strategies. In the star topology, all the controllers are deployed by using the central placement strategy ($k=1$), as naturally, the topology itself offers a unique structurally dominant node. Under this configuration, RYU achieves the lowest average latency (0.389 ms) as well as the smallest round-trip time, reflecting its light and event-driven design. OpenDaylight offers the best throughput (55.37 MBps), followed closely by RYU, indicative of good handling of data planes despite centralized control. ONOS has moderate latency (4.29 ms). Beacon has balanced but comparatively higher latency. For the linear topology, the placement of two controllers ($k = 2$) by the partition_kmeans placement technique is a factor that consistently improves performance among all controllers. RYU again records the lowest latency (6.63 ms), whereas OpenDaylight has the highest throughput (34.61 MBps), having parallel control processing over partitioned clusters.

5. Results and discussion

This section tests the performance of SDN controllers in star, linear, and ring topologies in various placement techniques of the controller.

5.1. Star topology

In the star-topology scenario, all three centrality placement strategies – center, degree, and betweenness – always choose the same switch (s1) as controller location for all 4 controllers, as in Table 5a, 5b, 5c, and 5d. As a result, the observed performance is the same in all placement techniques, indicating that the topology is dominating the placement decision.

Table 5a. Beacon – star topology

Technique	k	Placement	Avg latency, ms	Avg throughput, MBps
Center	1	s1	8.78	13.84055
Degree	1	s1	8.78	13.84055
Betweenness	1	s1	8.78	13.84055

Table 5b. ONOS – star topology

Technique	k	Placement	Avg latency, ms	Avg throughput, MBps
Center	1	s1	4.294	27.934
Degree	1	s1	4.294	27.934
Betweenness	1	s1	4.294	27.934

Table 5c. OpenDaylight – star topology

Technique	k	Placement	Avg latency, ms	Avg throughput, MBps
Center	1	s1	8.88	55.36937
Degree	1	s1	8.88	55.36937
Betweenness	1	s1	8.88	55.36937

Table 5d. RYU – star topology

Technique	k	Placement	Avg latency, ms	Avg throughput, MBps
Center	1	s1	0.389	50.81645
Degree	1	s1	0.389	50.81645
Betweenness	1	s1	0.389	50.81645

For Beacon, the average latency stabilizes at 8.78 ms with an average throughput of approximately 13.84 MBps, a balanced but moderate performance in a centralized configuration. ONOS exhibits a lower average latency of 4.29 ms and much better throughput of 27.93 MBps, it can handle centralized control traffic efficiently despite the fact that it has a distributed internal architecture. OpenDaylight shows latency with Beacon (8.88 ms), but it gets the highest throughput among all controllers (55.37 MBps), taking advantage of its optimized data-plane interaction after flow installation. RYU provides the minimum latency (0.389 ms), while preserving the high throughput rate (50.82 MBps), which is consistent with its lightweight, event-driven design.

The latency and throughput results for the star topology from Fig. 3 and Fig. 4 highlight clear performance differences among the evaluated controllers fixed at the central node. RYU has the lowest average latency, which is indicative of fast control plane response. ONOS shows relatively lower latency than Beacon and OpenDaylight, suggesting efficient control message handling even in a centralized

deployment. From a throughput perspective, OpenDaylight achieves the highest data plane throughput that is close to RYU, that both controllers to efficiently install and manage flow rules after the control paths are established. ONOS has moderate throughput, and Beacon records the lowest throughput. In a star-centered topology, controller implementation has a stronger impact on performance as compared to placement strategy. RYU demonstrates balanced low latency and high throughput; OpenDaylight is better at maximizing throughput despite increased control plane latency.

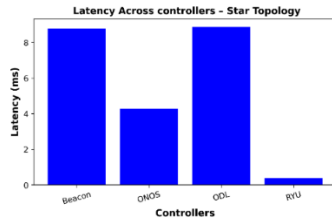


Fig. 3. Latency across various controllers in a star topology

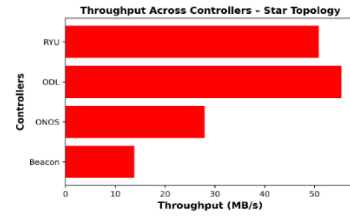


Fig. 4. Throughput across various controllers in a star topology

5.2. Linear topology

In the linear topology, all three centrality-based placement strategies – center, degree, and betweenness – have always selected the same pair of switches (s1 and s26) as controller locations for all of the evaluated controllers. In a linear network, the structurally central nodes near the ends of the chain offer balanced coverage for minimizing controller-to-switch distances in the case of two controllers being deployed ($k = 2$).

Table 6a. Beacon – linear topology

Technique	k	Placement	Avg latency, ms	Avg throughput, MBps
Center	2	s1, s26	15.02	8.62
Degree	2	s1, s26	15.02	8.62
Betweenness	2	s1, s26	15.02	8.62

Table 6b. ONOS – linear topology

Technique	k	Placement	Avg latency, ms	Avg throughput, MBps
Center	2	s1, s26	10.53	13
Degree	2	s1, s26	10.53	13
Betweenness	2	s1, s26	10.53	13

Table 6c. OpenDaylight – linear topology

Technique	k	Placement	Avg latency, ms	Avg throughput, MBps
Center	2	s1, s26	15.12	34.61
Degree	2	s1, s26	15.12	34.61
Betweenness	2	s1, s26	15.12	34.61

Table 6d. RYU – linear topology

Technique	k	Placement	Avg latency, ms	Avg throughput, MBps
Center	2	s1, s26	6.63	6.19
Degree	2	s1, s26	6.63	6.19
Betweenness	2	s1, s26	6.63	6.19

From Table 6a, 6b, 6c, and 6d, Beacon average latency is 15.02 ms with an associated throughput of 8.62 MBps, reflecting moderate under distributed control. ONOS has lower latency at 10.53 ms while maintaining a better throughput of 13 MBps, implying a better control-plane efficiency when multiple controllers are symmetrically placed along the chain. OpenDaylight records a similar latency as Beacon (15.12 ms) but much higher throughput (34.61 MBps), highlighting its efficient data-plane processing. RYU continues to provide the lowest latency among the controllers (6.63 ms), though its throughput remains comparatively lower at 6.19 MBps in line with its lightweight and less aggressive flow management approach.

From Fig. 5 and Fig. 6, linear topology shows clear performance differences among the evaluated controllers. RYU consistently gets the lowest average latency, which means faster control-plane responsiveness. ONOS demonstrates moderate latency with better throughput than Beacon. OpenDaylight with high-throughput controllers with higher latency, suggesting a trade-off between control overhead and data-plane efficiency. Beacon has comparatively higher latency and moderate throughput. Overall, the results highlight that in linear topologies, the choice of controller has a tremendous effect on performance, with RYU leaner for low-latency applications while OpenDaylight excels for throughput-intensive applications.

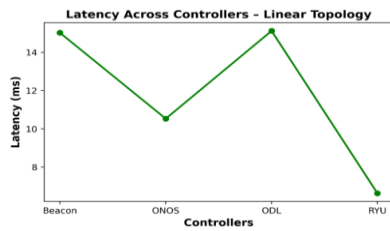


Fig. 5. Latency across various controllers in linear topology

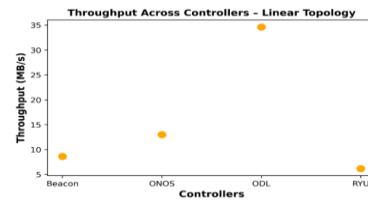


Fig. 6. Throughput across various controllers in linear topology

5.3. Ring topology

The ring topology provides additional balancing of path lengths by having two routes between any two nodes. For $k=2$, different algorithms have different placements. Uniform_k spreads controllers at positions such as {s1, s26} equally spaced, while k -median-greedy and capacitated tend to {s16, s41}, positioning controllers on opposite sides of the ring. For RYU, k -center-greedy chooses nodes that are intermediate, such as {s4, s29}.

Table 7a. Beacon – ring topology

Technique	k	Placement	Avg latency, ms	Avg throughput, MBps
kmedian greedy	2	s16, s41	15.02	8.617837
uniform k	2	s1, s26	15.02	8.617837
Capacitated	2	s16, s41	15.02	8.617837

Table 7b. ONOS – ring topology

Technique	k	Placement	Avg latency, ms	Avg throughput, MBps
kmedian greedy	2	s16, s41	10.534	13.1314
uniform k	2	s1, s26	10.534	13.1314
Capacitated	2	s16, s41	10.534	13.1314

Table 7c. OpenDaylight – ring topology

Technique	k	Placement	Avg latency, ms	Avg throughput, MBps
kmedian greedy	2	s16, s41	15.12	34.60644
uniform k	2	s1, s26	15.12	34.60644
Capacitated	2	s16, s41	15.12	34.60644

Table 7d. RYU – ring topology

Technique	k	Placement	Avg latency, ms	Avg throughput, MBps
uniform k	2	s1, s26	6.629	6.190731
kcenter greedy	2	s4, s29	6.629	6.190731
Capacitated	2	s16, s41	6.629	6.190731

The results for the ring topology, summarized in the above Tables 7a, 7b, 7c, 7d, indicate that multiple controller placement strategies across two controllers ($k = 2$) are being deployed. For Beacon, the k -median greedy, uniform_k, and capacitated approaches give an average latency of 15.02 ms and an average throughput of 8.62 MBps. These techniques are equally effective, suggesting that the cyclic symmetry of the ring topology has the effect of limiting the particular placement heuristic once the controllers are evenly distributed. A similar trend is seen in the case of ONOS, with all three techniques having an average latency of 10.53 ms and a throughput of 13.13 MBps. When compared to Beacon, ONOS always provides reduced latency. For OpenDaylight, the same filtering set of placement techniques has an average latency of 15.12 ms and a significantly higher throughput of 34.61 MBps. In the case of RYU, the uniform_k, k -center greedy, and capacitated strategies all achieve the lowest latency between the controllers at 6.63 ms with a throughput of 6.19 MBps. While RYU's throughput is lower than OpenDaylight and ONOS, because of the benefits of its lightweight control architecture, as shown in Figs 7 and 8.

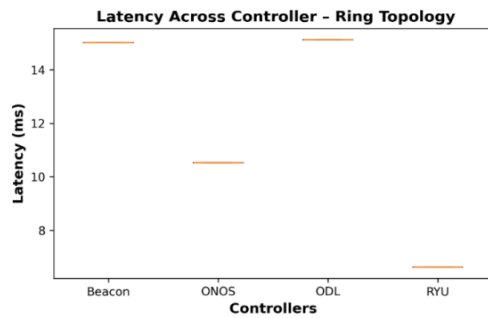


Fig. 7. Latency across various controllers in ring topology

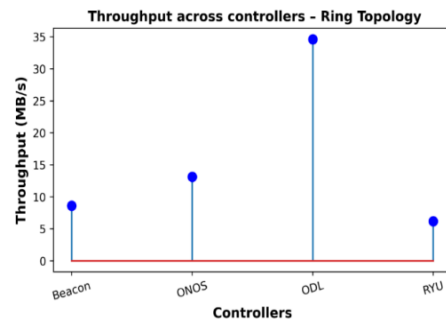


Fig. 8. Throughput across various controllers in ring topology

5.4. Cross-topology and cross-controller comparison

The placement of the controller affects network performance, as shown in Fig. 9.

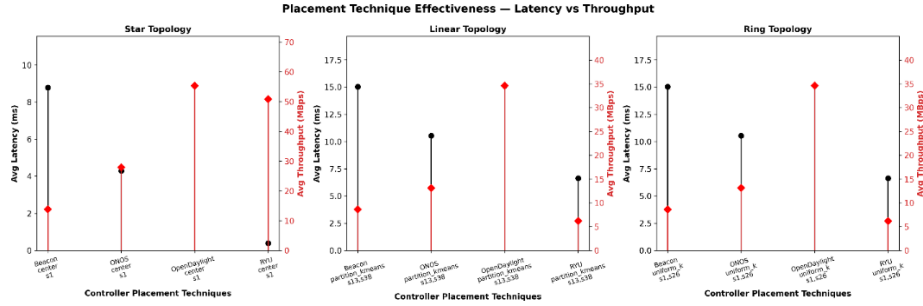


Fig. 9. Placement techniques effectiveness – latency vs throughput

In a star topology, all the controllers experience low latency, but RYU is notable for providing very high throughput and keeping the delay minimal. This indicates that having only a single controller is beneficial both in terms of responsiveness and data handling capability. In the linear topology, `partition_kmeans` always results in balanced performance, but the controllers have a strong distinction in the trade-off between latency and throughput. OpenDaylight has the highest throughput at the expense of latency, and RYU has the lowest delay and a moderate throughput, and a similar pattern is exhibited by the ring topology. The placement of `uniform_k` spreads the load evenly. OpenDaylight has the best throughput, and RYU has the lowest latency.

6. Conclusion

This paper presents in-depth empirical research of SDN controller placement and performance in the star, linear, and ring network topologies using four commonly used controllers and a wide diversity of placement strategies. The results show that in highly symmetric topologies, all the placement techniques lead to the same controller locations, resulting in various performances. Among the evaluated methods, the `partition_kmeans` technique provides stable performance. Even in linear and ring topologies with multiple placements, the performance differences between advanced placement algorithms and baseline methods remain relatively small. RYU consistently has the lowest control plane latency because of its lightweight design, while OpenDaylight has the highest data plane throughput, which takes advantage of efficient flow installation and parallel processing abilities. ONOS and Beacon provide stable and predictable performance. These findings suggest that, for small and medium-sized networks with regular structures, selecting an appropriate controller platform is more important than deploying complex placement optimization algorithms. Results are limited to medium-scale static topologies and Mininet-based emulation. Future work will extend this evaluation to larger and more irregular topologies, such as mesh and Fat-Tree networks, in which controller placement is expected to play a more significant role. Additional parameters such as scalability under heavy load, fault tolerance, dynamic controller

migration, and adaptive placement strategies will also be explored to further inform SDN implementation in the real world.

References

1. Kreutz, D., F. M. Ramos, P. E. Verissimo, C. E. Rothenberg, S. Azodolmolkly, S. Uhlig. Software-Defined Networking: A Comprehensive Survey. – Proceedings of the IEEE, Vol. **103**, 2015, No 1, pp. 14-76.
2. Prasanthi, B. V., P. C. Reddy. Software-Defined Networking (SDN) Technologies and Architectures for Efficient, Adaptive Networks. – In: Proc. of EAI International Conference on Advanced Technologies in Electronics, Communications and Signal Processing, Springer Nature Switzerland, Cham, 2026, pp. 27-38.
3. Singh, A. K., S. Srivastava. A Survey and Classification of Controller Placement Problem in SDN. – International Journal of Network Management, Vol. **28**, 2018, No 3, e2018.
4. Khan, S., A. Gani, A. W. A. Wahab, M. Guizani, M. K. Khan. Topology Discovery in Software-Defined Networks: Threats, Taxonomy, and State-of-the-Art. – IEEE Communications Surveys and Tutorials, Vol. **19**, 2017, No 1, pp. 303-324.
5. Amin, R., E. Rojas, A. Aqduş, S. Ramzan, D. Casillas-Perez, J. M. Arco. A Survey on Machine Learning Techniques for Routing Optimization in SDN. – IEEE Access, Vol. **9**, 2021, pp. 104582-104611.
6. Bajaber, F. Exploring the Performance and Characteristics of Single Linkage and Complete Linkage Hierarchical Clustering Methods for IoT Sensor Networks. – Cybernetics and Information Technologies, Vol. **23**, 2023, No 4, pp. 147-164.
7. Das, T., V. Sridharan, M. Gurusamy. A Survey on Controller Placement in SDN. – IEEE Communications Surveys and Tutorials, Vol. **22**, 2020, No 1, pp. 472-503.
8. Mayoral, A., R. Vilalta, R. Muñoz, R. Casellas, R. Martínez. SDN Orchestration Architecture and Its Integration with Cloud Computing Applications. – Optical Switching and Networking, Vol. **26**, 2017, pp. 2-13.
9. Ram, A., S. K. Chakraborty, U. Chatterjee. RL-HCP: A Reinforcement Learning-Based Heuristic Controller Placement Algorithm for Dynamic SDN Environments. – Applied Soft Computing, 2025, 113997.
10. Heller, B., R. Sherwood, N. McKeown. The Controller Placement Problem. – ACM SIGCOMM Computer Communication Review, Vol. **42**, 2012, No 4, pp. 473-478.
11. Sahakyan, H., H. Astsatryan. Dynamic Load Balancing for Direct Server Returns Networks Using eBPF for in-Band Metric Feedback. – Cybernetics and Information Technologies, Vol. **25**, 2025, No 3, pp. 142-155.
12. Badotra, S., J. Singh. OpenDaylight is a Controller for Software-Defined Networking. – International Journal of Advanced Research in Computer Science, Vol. **8**, 2017, No 5.
13. Singh, A., N. Kaur, H. Kaur. Extensive Performance Analysis of OpenDayLight (ODL) and Open Network Operating System (ONOS) SDN Controllers. – Microprocessors and Microsystems, Vol. **95**, 2022, 104715.
14. Feng, S., Y. Wang, X. Zhong, J. Zong, X. Qiu, S. Guo. A Ring-Based Star-Link Failure Recovery Approach in the SDN Data Plane. – In: Proc. of IEEE/IFIP Network Operations and Management Symposium (NOMS'18), IEEE, 2018, pp. 1-7.
15. Belgaum, M. R., S. Musa, M. M. Alam, M. M. Su'ud. A Systematic Review of Load Balancing Techniques in Software-Defined Networking. – IEEE Access, Vol. **8**, 2020, pp. 98612-98636.
16. Amaral, P., P. F. Pinto, L. Bernardo, F. Silva. SDN-Based Traffic Engineering without Optimization: A Centrality-Based Approach. – In: Proc. of IEEE International Conference on Communications (ICC'17), IEEE, 2017, pp. 1-7.
17. Obadia, M., M. Bouet, J. L. Rougier, L. Iannone. A Greedy Approach for Minimizing SDN Control Overhead. – In: Proc. of IEEE Conference on Network Softwarization (NetSoft'15), IEEE, 2015, pp. 1-5.

18. Salam, R., A. Bhattacharya. Efficient Greedy Heuristic Approach for Fault-Tolerant Distributed Controller Placement in Scalable SDN Architecture. – Cluster Computing, Vol. **25**, 2022, No 6, pp. 4543-4572.
19. Isong, B., R. R. S. Molose, A. M. Abu-Mahfouz, N. Dladlu. Comprehensive Review of SDN Controller Placement Strategies. – IEEE Access, Vol. **8**, 2020, pp. 170070-170092.
20. Killi, B. P. R., S. V. Rao. Capacitated Next Controller Placement in Software-Defined Networks. – IEEE Transactions on Network and Service Management, Vol. **14**, 2017, No 3, pp. 514-527.
21. Yusuf, M. N., et al. Distributed Controller Placement in Software-Defined Networks with Consistency and Interoperability Problems. – Journal of Electrical and Computer Engineering, Vol. **2023**, 2023, 6466996.
22. Padmavathy, N., K. Srinivas, P. Srinivas. Performance Evaluation of Multicast Routing Protocols in Mobile ad hoc Networks. – In: Proc. of International Conference on Cognitive Computing and Cyber Physical Systems, Cham, Springer Nature Switzerland, 2023.
23. Chahal, M., S. Harit, K. K. Mishra, A. K. Sangaiah, Z. Zheng. A Survey on Software-Defined Networking in Vehicular ad hoc Networks: Challenges, Applications, and Use Cases. – Sustainable Cities and Society, Vol. **35**, 2017, pp. 830-840.

*Received: 02.02.2026, First revision: 23.03.2026, Second revision: 31.03.2026,
Accepted: 05.04.2026*