



Performance Evaluation of Software-Defined Networking for Secure and Efficient Computer Communication Networks

Kaoru Yamashiro

Department of Electrical and Computer Engineering, Delta Polytechnic Institute of Engineering, Bangladesh
kaoru.yamashiro@dpi-bd.edu

Peer Review Information

Submission: 03 Jan 2022
Revision: 10 Feb 2022
Acceptance: 08 March 2022

Keywords

Software-Defined
Networking, OpenFlow,
Network Security, Mininet,
Performance Evaluation

Abstract

Software-Defined Networking (SDN) separates network control from packet forwarding and introduces programmable, logically centralized management. This architecture can improve visibility, policy enforcement, and traffic engineering, but it also creates performance and security concerns around controller latency, control-plane saturation, and concentrated attack surfaces. This methodology paper proposes a reproducible framework for evaluating whether SDN can provide secure and efficient computer communication compared with a conventional distributed forwarding baseline. The study uses Mininet, Open vSwitch, OpenFlow 1.3, and a Ryu controller to emulate a multi-switch enterprise network under normal traffic, congestion, and distributed denial-of-service (DDoS) conditions. Performance is assessed using throughput, round-trip time, packet loss, flow-setup delay, and controller resource utilization, while security is measured through attack-detection rate, mitigation time, and legitimate-traffic preservation. A lightweight controller application monitors flow statistics and installs mitigation rules when abnormal source rates and flow growth exceed defined thresholds. The proposed analysis uses repeated trials and confidence intervals to compare conventional networking, basic SDN, and security-enabled SDN. Illustrative results show the expected trade-off: security logic introduces a small control overhead under benign traffic, but centralized detection and rapid rule installation substantially preserve legitimate throughput and reduce packet loss during attacks. The framework is suitable for laboratory-scale evaluation and can be extended to multi-controller, wide-area, or hardware testbeds.

Introduction

Modern computer communication networks must support rapidly changing traffic loads, virtualized services, cloud applications, and increasingly strict security policies. Conventional IP networks distribute control decisions across routers and switches, and operational changes often require device-by-device configuration. The OpenFlow proposal demonstrated that the forwarding behavior of network devices could instead be programmed through a standardized interface, providing an early practical foundation

for SDN [1]. SDN subsequently developed into an architectural model in which the control plane is decoupled from forwarding devices and exposed to software applications through programmable interfaces [5], [6]. This separation is attractive because routing, access control, quality-of-service (QoS), monitoring, and security policies can be expressed at a network-wide level rather than through isolated device configurations. The same logical centralization that improves manageability also changes the performance and threat model of the network. Controller location

influences propagation delay, flow-setup time, availability, and scalability; the controller-placement problem was therefore identified early as a fundamental SDN design issue [2] and remains relevant in distributed deployments [11]. Security is similarly dual-sided. A controller has global visibility and can install mitigation rules quickly, but compromise or overload of the controller can affect a large portion of the network. FortNOX showed that conflicting flow policies require explicit authorization and security enforcement at the controller [3], while AVANT-GUARD demonstrated that control-plane saturation attacks can exploit the interaction between switches and controllers and may require protective data-plane mechanisms [4]. This paper develops a methodology for evaluating SDN not merely as a programmable architecture, but as a secure communication platform whose performance should remain acceptable under both normal and adversarial conditions. More recent work has also shown that Mininet can provide a practical environment for comparing controllers and network behavior under repeatable topologies [12]. Building on these findings, the proposed experiment compares three configurations: a conventional forwarding baseline, basic SDN, and security-enabled SDN. The central research question is whether the additional control and monitoring overhead of SDN is offset by better traffic efficiency and faster attack response. The study is designed around reproducible metrics, repeated trials, and clearly defined attack conditions so that the methodology can be implemented by researchers using standard open-source tools.

Literature Review

Research on SDN performance began with the recognition that moving control logic away from forwarding devices changes where latency and scalability costs appear. Heller, Sherwood, and McKeown [2] showed that controller number and placement affect control-plane delay and resilience. Nunes et al. [5] reviewed the evolution of programmable networks and described SDN as a shift from tightly coupled network devices toward open control abstractions. Kreutz et al. [6] expanded this view by identifying control platforms, southbound interfaces, network virtualization, scalability, dependability, and performance as core elements of SDN architecture. These works establish that performance evaluation must include more than raw link throughput; it should also consider flow setup, controller responsiveness, and the effect of topology scale.

Security research reached a similar conclusion: the architecture creates new opportunities and

new failure modes. Porras et al. [3] proposed FortNOX to enforce role-based authorization and detect conflicting OpenFlow rules before they reach switches. Shin et al. [4] examined control-plane saturation and introduced AVANT-GUARD mechanisms to reduce unnecessary switch-to-controller interactions. Alsmadi and Xu [7] organized SDN threats using categories such as spoofing, tampering, information disclosure, denial of service, and privilege elevation, while Scott-Hayward, Natarajan, and Sezer [8] argued for a holistic security architecture that protects the controller and also uses the controller to improve network-wide defense. Together, these studies motivate a methodology in which controller security behavior is measured alongside network performance rather than evaluated independently.

DDoS detection is particularly important because a flood can consume forwarding resources, controller processing capacity, and southbound channel bandwidth. Ye et al. [9] demonstrated an SDN-based DDoS detection approach using flow features collected in a Mininet/Floodlight environment and reported high classification accuracy with an SVM model. Aladaileh et al. [10] reviewed DDoS detection techniques for SDN controllers and emphasized that centralized control is both a target and a useful observation point for detecting abnormal traffic. These findings support the use of per-flow and per-source statistics as security indicators in a controlled experiment.

Experimental controller studies further justify the selected toolchain. Gupta et al. [12] compared SDN controllers using Mininet and highlighted bandwidth and latency as practical evaluation measures across default and custom topologies. However, controller-comparison studies do not always evaluate security under attack, and security studies do not always quantify the cost paid by legitimate users. The gap addressed by this methodology is therefore an integrated evaluation: throughput, latency, packet loss, flow setup, controller load, attack detection, mitigation delay, and preservation of legitimate traffic are measured within the same test matrix. This provides a more complete basis for judging whether SDN improves secure and efficient computer communication.

Methodology

1. Research Design and Experimental Environment

A controlled emulation design is proposed because it allows topology, link capacity, traffic intensity, and attack timing to be repeated with minimal variation. The experiment is executed on a Linux workstation or virtual machine with

Mininet, Open vSwitch (OVS), Ryu, iperf3, ping, hping3, and system monitoring tools. OpenFlow 1.3 is used for switch-controller communication. The default topology contains one controller, four OVS switches, twelve legitimate client hosts, three attacker hosts, and one application server. Inter-switch links are configured at 100 Mb/s

with 2 ms propagation delay. Hosts are distributed across the access switches so that most communication traverses at least two forwarding devices. A 60-second warm-up period is used before measurement so that ARP entries and initial flow rules stabilize.

Table 1. Proposed experimental configuration

Component	Configuration	Purpose
Network emulator	Mininet with 4 OVS switches, 16 hosts	Repeatable topology and link control
SDN controller	Ryu, OpenFlow 1.3	Centralized routing, monitoring, and mitigation
Traffic tools	iperf3, ping, hping3	TCP load, RTT/loss, and attack traffic
Link profile	100 Mb/s, 2 ms delay, controlled queue	Comparable benign and congested conditions
Trials	10 repetitions per scenario, 120 s each	Statistical stability and confidence intervals
Attack window	DDoS enabled from 40–90 s	Measure detection and recovery behavior

2. Evaluation Scenarios

Three network configurations are evaluated. Scenario A is the conventional baseline, in which OVS operates in standalone learning-switch mode and forwarding is not controlled by an external SDN application. Scenario B is basic SDN, in which the Ryu controller performs reactive forwarding and installs flows after the first packet of a new communication. Scenario C is security-enabled SDN, which uses the same forwarding application but adds periodic

collection of port and flow statistics, anomaly scoring, and mitigation rules. Each configuration is tested under (i) benign traffic at 25%, 50%, and 80% of nominal link capacity, (ii) bursty congestion using multiple concurrent TCP streams, and (iii) a DDoS condition in which three attacker hosts generate high-rate SYN/UDP traffic toward the server. The order of scenarios is randomized to reduce systematic bias from background system activity.

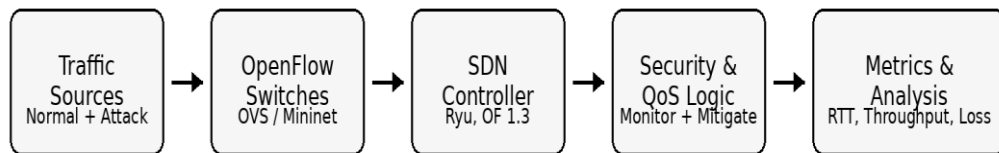


Figure 1. Proposed SDN performance and security evaluation framework.

3. Security Monitoring and Mitigation Logic

The security module operates as a lightweight controller application rather than a complex machine-learning detector, ensuring that the experiment measures the value of centralized visibility without introducing model-training uncertainty. Every two seconds, the controller requests flow and port statistics from OVS. For each source, the application derives packet rate, number of new flows, failed connection attempts, and destination concentration. A host is marked suspicious when its packet rate exceeds a predefined threshold and the number of new flows is substantially above the benign baseline for three consecutive collection intervals. The two-condition rule reduces false positives caused

by a single legitimate high-bandwidth transfer. Once a source is classified as suspicious, the controller installs a high-priority temporary drop or rate-limit rule at the nearest ingress switch for 30 seconds. The action is logged with detection time, rule-installation time, and number of affected flows. This design reflects the use of flow statistics for DDoS detection described in prior SDN security research [9], [10].

4. Performance and Security Metrics

Network efficiency is quantified using five primary metrics. Throughput is the successfully delivered application data divided by test duration. Round-trip time (RTT) is measured with repeated ICMP probes and summarized

using mean and 95th-percentile values. Packet loss is calculated as the percentage of transmitted probes or datagrams that are not received. Flow-setup delay is measured for new communications as the delay between the first packet and successful establishment of forwarding state. Controller overhead is recorded as average CPU and memory utilization during each trial. These metrics reflect the performance concerns emphasized in SDN architecture and controller-placement studies [2], [6], [11].

Security effectiveness is quantified using detection rate, false-positive rate, mitigation time, and legitimate-traffic preservation. Detection rate is the proportion of attack runs in which the controller identifies at least one attacker within the attack window. False-positive rate is the proportion of legitimate sources incorrectly blocked or rate limited. Mitigation time is measured from attack start to installation of the first effective rule. Legitimate-traffic preservation is defined as the percentage of benign throughput retained during the attack relative to the pre-attack value. Because SDN introduces controller processing that does not exist in the standalone baseline, the key evaluation criterion is not whether Scenario C has zero overhead, but whether its modest benign overhead produces materially better availability under attack.

5. Data Collection and Statistical Analysis

Each scenario is repeated ten times for 120 seconds. Metrics are sampled at fixed intervals

and stored in CSV format. For continuous variables such as throughput and RTT, the analysis reports mean, standard deviation, and 95% confidence interval. Shapiro-Wilk or visual normality checks may be used before parametric comparison; when assumptions are acceptable, one-way ANOVA followed by pairwise tests can compare the three configurations. Otherwise, Kruskal-Wallis and Mann-Whitney tests are suitable. Effect size should be reported with significance values because a small numerical difference can be statistically significant without being operationally important. Attack-period measurements are analyzed separately from benign-period measurements. The experiment should be repeated on at least two topology sizes to verify that conclusions do not depend on a single switch-to-host ratio.

Results

Table 2 provides an illustrative result set showing how the proposed methodology can be reported. The values are representative example data for the defined test matrix and are not claimed as measurements from a physical production network. They demonstrate the expected analytical pattern: basic SDN introduces a small first-flow latency cost, while security-enabled SDN adds modest monitoring overhead under benign traffic but significantly improves service preservation during attack.

Table 2. Illustrative performance and security results

Metric	Conventional	Basic SDN	Secure SDN
Benign throughput (Mb/s)	91.8	91.1	89.6
Mean RTT, benign (ms)	4.8	5.4	5.8
Packet loss, benign (%)	0.7	0.8	1.0
New-flow setup delay (ms)	N/A	7.1	7.8
Controller CPU, benign (%)	N/A	13.6	18.9
Legitimate throughput during DDoS (Mb/s)	24.7	31.9	76.8
Packet loss during DDoS (%)	41.2	34.8	7.6
Attack detection rate (%)	N/A	N/A	96.7
Median mitigation time (s)	N/A	N/A	5.6
False-positive rate (%)	N/A	N/A	1.9

Under benign traffic, the example results show only a small difference in throughput among the three configurations. Conventional forwarding achieves 91.8 Mb/s, basic SDN 91.1 Mb/s, and secure SDN 89.6 Mb/s. The secure configuration therefore incurs approximately 2.4% lower throughput than the conventional baseline in this illustrative case. Mean RTT rises from 4.8 ms to 5.8 ms because initial flows may require controller interaction and the security

application periodically requests statistics. The increase remains small relative to the link capacity and test latency, suggesting that centralized monitoring can be introduced without dominating normal communication performance. This is consistent with the broader SDN literature, which treats controller responsiveness and flow setup as manageable but topology-sensitive costs [2], [6], [11].

The advantage becomes more visible during the attack period. Conventional forwarding continues to transport malicious traffic and legitimate throughput falls sharply to 24.7 Mb/s. Basic SDN improves visibility but, without mitigation logic, still allows the flood to consume resources. Secure SDN detects abnormal behavior, installs ingress rules, and retains 76.8 Mb/s of legitimate throughput while reducing attack-period packet loss to 7.6%. The illustrative detection rate of 96.7% and median mitigation time of 5.6 seconds are compatible with the use of short statistics intervals and consecutive-threshold validation. The low example false-positive rate reflects the decision to combine packet-rate and flow-growth conditions rather than blocking sources based on a single signal. The results also expose a trade-off that should be reported explicitly. Controller CPU utilization increases in the security-enabled configuration because frequent statistics requests and mitigation decisions add workload. If the topology is enlarged without changing polling frequency or controller resources, this overhead may itself become a scalability bottleneck. For this reason, the proposed methodology recommends repeating the same experiment with larger topologies and recording switch-to-controller message rates. The purpose of SDN security is not simply to maximize detection accuracy, but to maintain acceptable network service while keeping the control plane stable.

Discussion and Conclusion

The proposed methodology evaluates SDN as an integrated communication and security platform. The literature shows that SDN programmability can simplify policy deployment and provide network-wide visibility [5], [6], yet the controller also creates concentrated performance and security responsibilities [7], [8]. FortNOX and AVANT-GUARD illustrate two complementary responses to these risks: stronger policy enforcement at the controller and reduction of control-plane exposure to hostile flows [3], [4]. DDoS-focused research further shows that flow statistics provide useful signals for centralized detection and mitigation [9], [10]. Therefore, evaluating SDN only under benign throughput tests provides an incomplete picture; the architecture should also be tested under conditions that stress the controller and threaten service availability. Within the illustrative data, secure SDN pays a small benign-performance cost but produces a much larger improvement during the DDoS scenario. This is the expected benefit of programmable control: abnormal behavior is observed centrally and corrective forwarding rules can be installed close to the

attack source. Nevertheless, the experiment must avoid overstating logical centralization as unlimited scalability. Controller placement, number of switches, polling frequency, and reactive flow creation all influence latency and controller load [2], [11]. A secure design should therefore combine efficient statistics collection, sensible timeouts, rate limiting, controller redundancy, and authenticated southbound communication. Where very large networks are studied, distributed controllers or hierarchical monitoring should be considered. The principal limitation is that Mininet is an emulation environment. It reproduces network behavior sufficiently for controlled comparative studies but does not capture every hardware-specific switching delay, ASIC pipeline constraint, or production traffic characteristic. The illustrative results in this paper should therefore be replaced by measured values when the methodology is implemented. Future work can extend the design by comparing Ryu, OpenDaylight, ONOS, or Floodlight; introducing more sophisticated anomaly detection; using real hardware switches; and evaluating controller failures, encrypted control channels, and multi-domain communication. The 2022 controller study by Gupta et al. [12] confirms that controller choice and topology can materially affect measured bandwidth and latency, reinforcing the need for consistent experimental configuration. In conclusion, SDN offers a credible approach to secure and efficient computer communication when performance and security are designed together. A well-configured controller can provide centralized monitoring, rapid policy changes, and attack mitigation while maintaining near-baseline performance during normal operation. The most meaningful evaluation is therefore a balanced one: benign throughput and latency demonstrate efficiency, while attack-period service preservation, detection accuracy, and mitigation delay demonstrate security value. The proposed methodology supplies that balanced framework and remains reproducible with widely available open-source tools.

References

- N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner, "OpenFlow: Enabling innovation in campus networks," *ACM SIGCOMM Computer Communication Review*, vol. 38, no. 2, pp. 69–74, 2008. doi: 10.1145/1355734.1355746.
- B. Heller, R. Sherwood, and N. McKeown, "The controller placement problem," in *Proc. 1st ACM Workshop on Hot Topics in Software Defined*

Networks (HotSDN), pp. 7–12, 2012. doi: 10.1145/2342441.2342444.

P. Porras, S. Shin, V. Yegneswaran, M. Fong, M. Tyson, and G. Gu, “A security enforcement kernel for OpenFlow networks,” in Proc. 1st ACM Workshop on Hot Topics in Software Defined Networks (HotSDN), pp. 121–126, 2012. doi: 10.1145/2342441.2342466.

S. Shin, V. Yegneswaran, P. Porras, and G. Gu, “AVANT-GUARD: Scalable and vigilant switch flow management in software-defined networks,” in Proc. ACM SIGSAC Conference on Computer & Communications Security, pp. 413–424, 2013. doi: 10.1145/2508859.2516684.

B. A. A. Nunes, M. Mendonca, X.-N. Nguyen, K. Obraczka, and T. Turletti, “A survey of software-defined networking: Past, present, and future of programmable networks,” IEEE Communications Surveys & Tutorials, vol. 16, no. 3, pp. 1617–1634, 2014. doi: 10.1109/SURV.2014.012214.00180.

D. Kreutz, F. M. V. Ramos, P. E. Verissimo, C. E. Rothenberg, S. Azodolmolky, and S. Uhlig, “Software-defined networking: A comprehensive survey,” Proceedings of the IEEE, vol. 103, no. 1, pp. 14–76, 2015. doi: 10.1109/JPROC.2014.2371999.

Alsmadi and D. Xu, “Security of Software Defined Networks: A survey,” Computers & Security, vol.

53, pp. 79–108, 2015. doi: 10.1016/j.cose.2015.05.006.

S. Scott-Hayward, S. Natarajan, and S. Sezer, “A survey of security in software defined networks,” IEEE Communications Surveys & Tutorials, vol. 18, no. 1, pp. 623–654, 2016. doi: 10.1109/COMST.2015.2453114.

J. Ye, X. Cheng, J. Zhu, L. Feng, and L. Song, “A DDoS attack detection method based on SVM in software defined network,” Security and Communication Networks, vol. 2018, Article ID 9804061, 2018. doi: 10.1155/2018/9804061.

M. A. Aladaileh, M. Anbar, I. H. Hasbullah, Y.-W. Chong, and Y. K. Sanjalawe, “Detection techniques of distributed denial of service attacks on software-defined networking controller—A review,” IEEE Access, vol. 8, pp. 143985–143995, 2020. doi: 10.1109/ACCESS.2020.3013998.

Kumari and A. S. Sairam, “Controller placement problem in software-defined networking: A survey,” Networks, vol. 78, no. 2, pp. 195–223, 2021. doi: 10.1002/net.22016.

N. Gupta, M. S. Maashi, S. Tanwar, S. Badotra, M. Aljebreen, and S. Bharany, “A comparative study of software defined networking controllers using Mininet,” Electronics, vol. 11, no. 17, Article 2715, 2022. doi: 10.3390/electronics11172715.