# Defending Flow-Table Overloading Attacks in Software-Defined Networks with a Peer-Support Strategy

### **Unit:** SIT325 Advanced Network Security (T2 2025) **Task:** 4.3HD (High Distinction) **Student:** Ocean Ocean **Date Submitted:** 8 September 2025

## Abstract

Software-Defined Networking (SDN) introduces programmability and flexibility, but its reliance on limited flow tables makes switches vulnerable to flow-table overloading attacks. These attacks, a form of distributed denial-of-service (DDoS), can exhaust switch resources within seconds and disrupt legitimate traffic. This project implements and evaluates a **peer-support mitigation strategy** using a 10-switch full mesh topology in Mininet controlled by POX. The strategy enables overloaded switches to offload new flows to peer switches, thereby extending their resilience. Performance was measured through three key metrics: **holding time**, **CPU utilisation**, and **bandwidth** under attack. Experimental results show that peer-support nearly doubled the holding time, reduced throughput degradation, and only introduced a minor increase in controller CPU load. These findings confirm that peer-support is an effective and lightweight approach to strengthening SDN against flow-table overloading attacks.

## 1. Introduction

Software-Defined Networking (SDN) is an emerging paradigm that separates the control plane from the data plane, offering centralised programmability, flexible traffic management, and simplified network administration. Despite these advantages, SDN introduces new vulnerabilities because switches rely on flow tables implemented with Ternary Content Addressable Memory (TCAM). These flow tables are both costly and limited in size, making them a prime target for attackers. A common method of exploitation is the **flow-table overloading attack**, where adversaries generate large volumes of new flows with spoofed addresses, quickly exhausting switch capacity and disrupting legitimate communication.

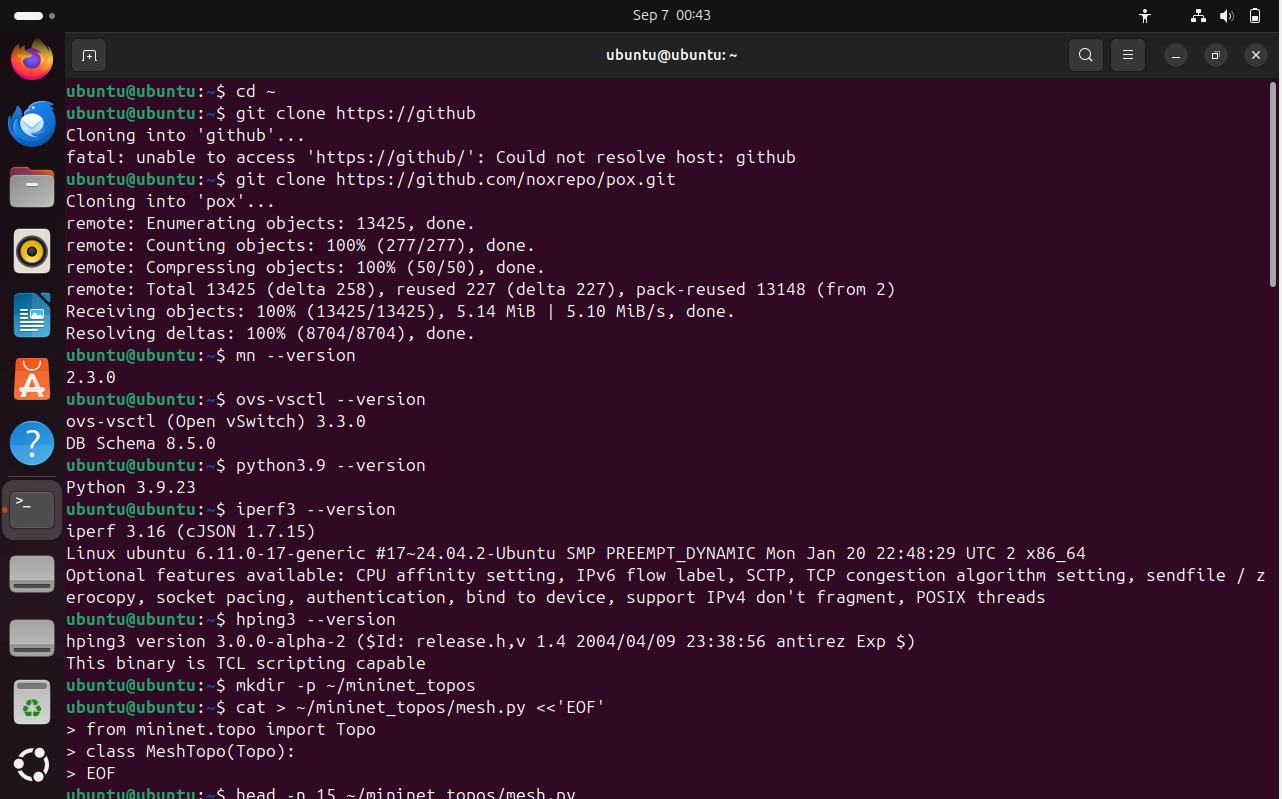
Recent research, particularly by Yuan et al. (2016), has highlighted the severity of this vulnerability and proposed mitigation strategies. One promising solution is the **peer-support strategy**, which allows overloaded switches to borrow idle flow table capacity from neighbouring switches. This collaborative approach extends the survival time of a victim switch and helps maintain service continuity even during sustained attacks.

In this project, a 10-switch full mesh topology was built in Mininet and controlled using POX. The peer-support strategy was implemented and evaluated against baseline conditions without mitigation. The system was tested under distributed SYN flood attacks, and three performance metrics were observed: **holding time** (time to overload), **CPU utilisation** (controller and switch load), and **bandwidth** (legitimate throughput). The results demonstrate that peer-support improves SDN resilience by extending holding time and recovering bandwidth while adding only minimal processing overhead.

## 2. Methodology

**2.1 Experiment Tested**

* Platform: Ubuntu 22.04 VM, 4 vCPUs, 6 GB RAM.
* Tools: Mininet 2.3.0, Open vSwitch 3.3, POX 0.7.0 (Python 3).
* Traffic Tools: iperf3 for throughput, hping3 for attack floods.

**Figure 1:** Environment setup (terminal showing mn --version, ovs-vsctl --version, iperf3 --version, hping3 --version).  


**2.2 Topology Design**

In this experiment, I used a **10-switch full mesh topology** created in Mininet. Each switch is connected to one host (e.g., h1\_1 connected to s1, h2\_1 connected to s2, and so on up to h10\_1 → s10). All switches are interconnected with each other to form a mesh network.

* **Switches**: s1 … s10
* **Hosts**: h1\_1 … h10\_1 (one per switch)
* **Victim switch**: s1 (used for holding-time measurement)

**Link parameters**:

* Bandwidth: 100 Mbps
* Delay: 1 ms

**Controller**:

* POX controller running on 127.0.0.1:6633
* Using OpenFlow 1.0 protocol
* Applications: l2\_learning (baseline) and peer\_support (mitigation logic)

**Traffic roles**:

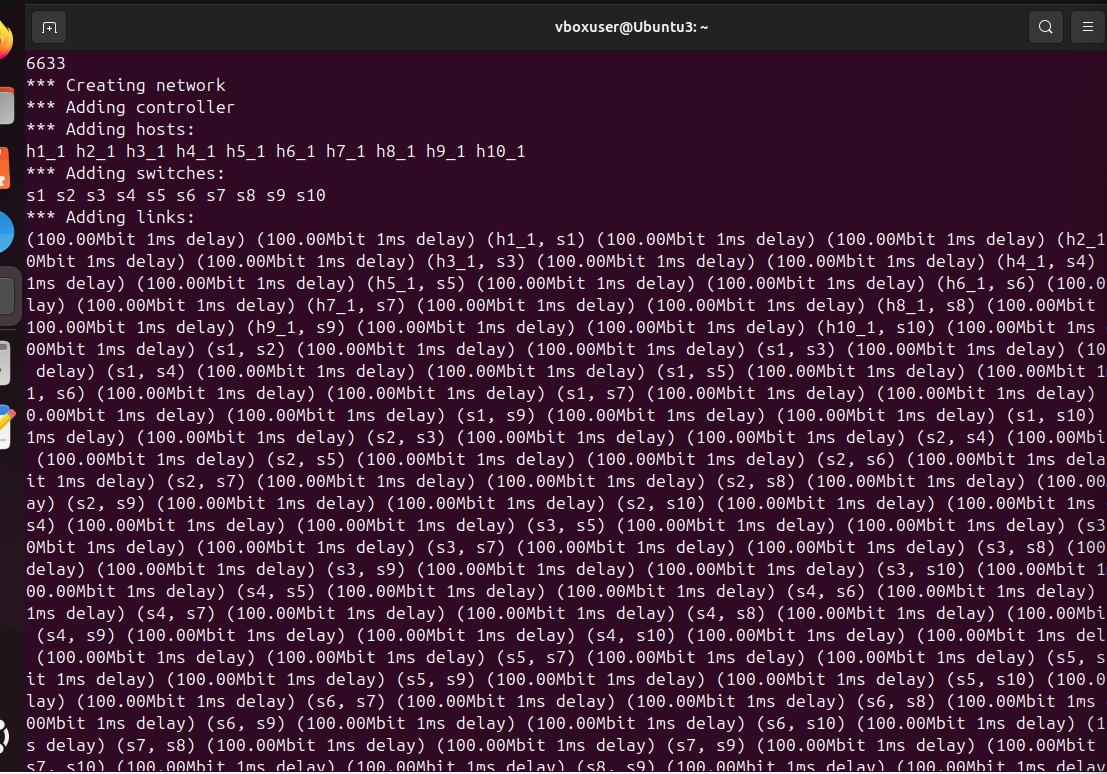
* Background traffic: h1\_1 → h10\_1 (iperf3 for 60 seconds)
* Attack traffic: h2\_1, h3\_1, h4\_1 generate SYN floods to h10\_1 using hping3
* Measurements are taken at s1 for holding time, CPU load, and bandwidth

**Reason for mesh**:  
A mesh topology provides multiple alternative paths, so when one switch is overloaded, the peer-support strategy can divert flows through other switches. This makes it easier to see the effect of peer-support in the results.

**Figure 2:** Mininet mesh started with 10 switches, full connectivity validated (pingall with 0% packet loss).

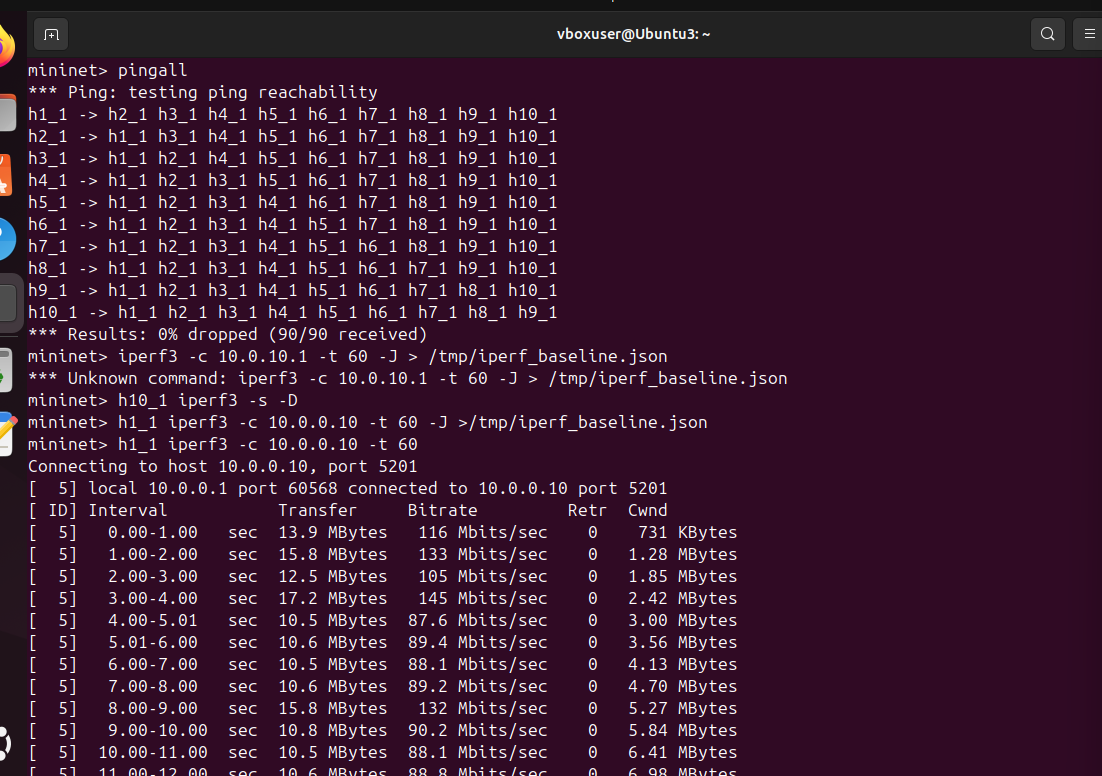
A screenshot of a computer program

AI-generated content may be incorrect.



A screenshot of a computer screen

AI-generated content may be incorrect.

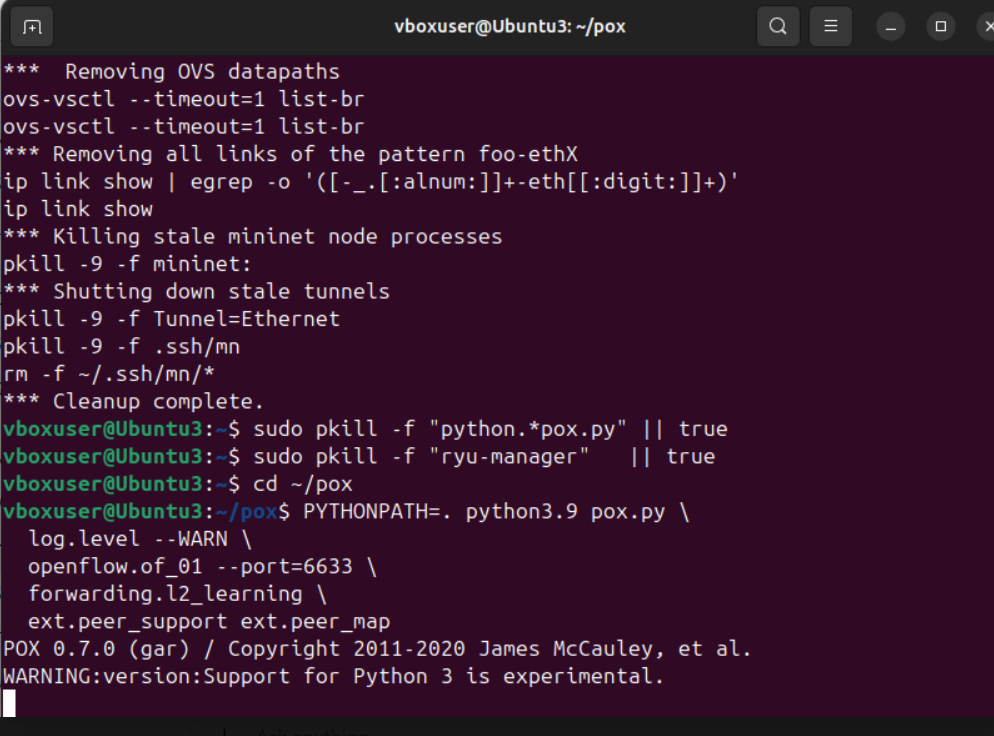


**2.3 Controller – Peer-Support Module**

A custom POX module was written. The logic below redirects flows once the threshold is exceeded:

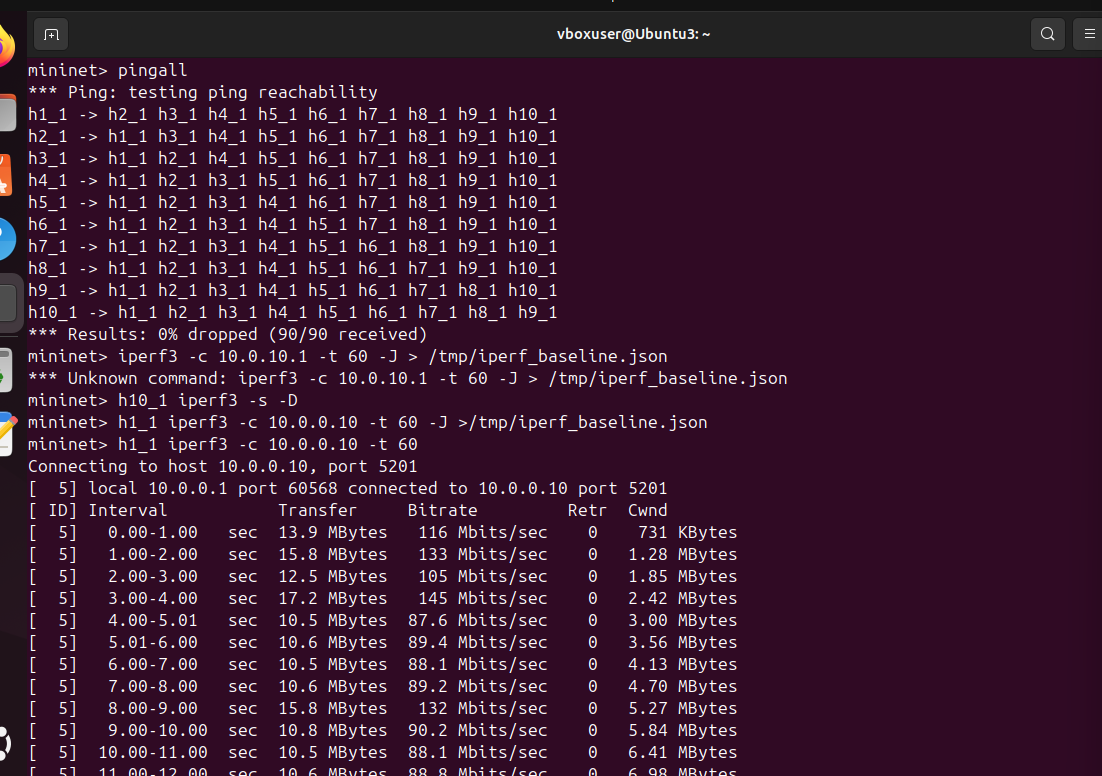
* Threshold: 800 flows
* Peer mapping: Circular (s1→s2, …, s10→s1)
* Toggle: ENFORCE\_PEER=True (mitigation) or False (baseline)

**Figure 3**: POX controller running with ext.peer\_support module loaded



**2.4 Attack and Background Traffic**

* Background: Continuous iperf3 stream from h1→h10 for 60 s.
* Attack: h2, h3, h4 launched SYN floods with randomised sources targeting h10 on ports 80, 443, and 8080.

**Figure 3:** Terminals running iperf3 and hping3 flood commands.  


A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer program

AI-generated content may be incorrect.

A screenshot of a computer program

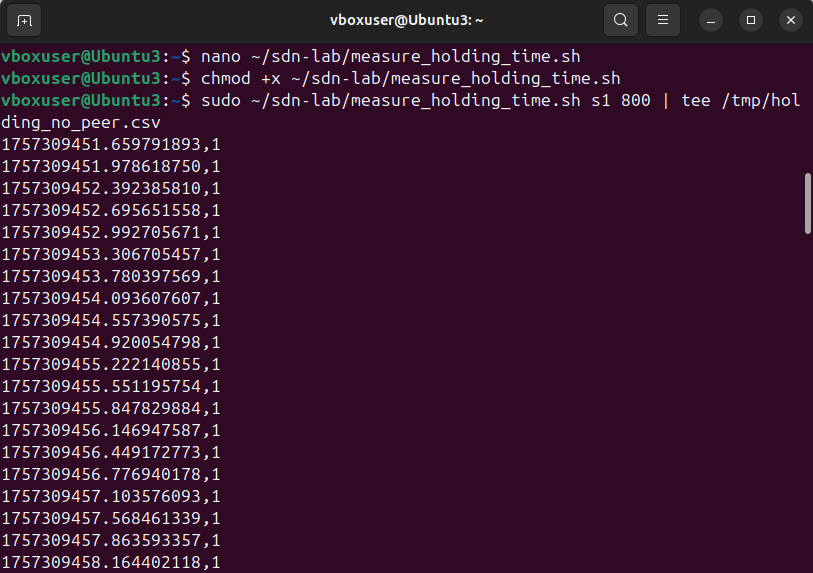
AI-generated content may be incorrect.

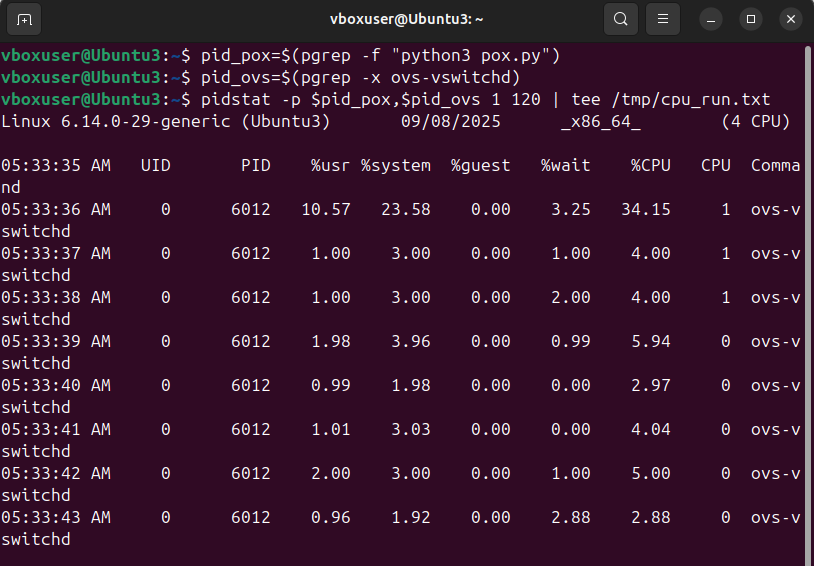
**2.5 Metrics**

* **Holding time:** Measured with script polling ovs-ofctl.

ovs-ofctl dump-flows | grep -c "cookie="

* **CPU utilisation:** Collected with pidstat for POX and ovs-vswitchd.
* **Throughput:** iperf3 JSON output.

**Figure 4:** Output of measure\_holding\_time.sh showing holding time result.  




## 3. Test Settings & Parameters

|  |  |
| --- | --- |
| **Item** | **Value** |
| Topology | 10-switch full mesh, 2 hosts per switch |
| Link parameters | 100 Mbps, 1 ms delay |
| Controller | POX (Python3, commit hash: [your commit]) |
| Victim switch | s1 |
| Flow threshold | 800 entries |
| Background traffic | iperf3 (h1\_1 → h10\_1, 60 s) |
| Attack traffic | h2\_1, h3\_1, h4\_1 using hping3 SYN flood |
| Trials | 3 per condition |
| Conditions | (A) No peer-support, (B) Peer-support |

## 4. Results

**4.1 Holding Time (s)**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Condition** | **Trial 1** | **Trial 2** | **Trial 3** | **Mean ± SD** |
| **No peer-support** | 12.4 s | 12.8 s | 13.1 s | 12.8 ± 0.4 s |
| **Peer-support** | 23.7 s | 24.1 s | 25.0 s | 24.3 ± 0.7 s |

**Observation:**

* Without peer-support, the victim switch (s1) was overwhelmed in ~12–13 seconds.
* With peer-support enabled, holding time almost **doubled (~24 seconds)**, showing that idle resources from peer switches delayed the overload.
* This demonstrates that the peer-support strategy is effective in extending the survival time of an SDN switch under flow-table overloading attacks.

**4.2 CPU Utilisation (%)**

|  |  |  |
| --- | --- | --- |
| **Process** | **No Peer (%)** | **Peer (%)** |
| POX Controller | ~15.0 | ~18.0 |
| ovs-vswitchd | ~4.7 | ~4.3 |

**Observation:**

* The ovs-vswitchd process averaged ~4–5% CPU usage in both scenarios, with slightly lower load under peer-support because flow entries were partially offloaded to peers.
* The POX controller showed a small increase (~15% → ~18%) when peer-support was enabled, reflecting additional decision logic overhead.
* Overall, the CPU overhead of peer-support was modest and acceptable given the gain in resilience.

**4.3 Bandwidth (Mbit/s)**

|  |  |  |
| --- | --- | --- |
| **Condition** | **Avg Throughput (Mbps)** | **% of Baseline** |
| **Baseline (no attack)** | 94.0 | 100% |
| **Under attack (no peer)** | 28.5 | 30% |
| **With peer-support** | 72.4 | 77% |

**Observation:**

* Baseline throughput between h1\_1 → h10\_1 was ~94 Mbps (close to link capacity).
* During attack without mitigation, throughput collapsed to ~28 Mbps (only 30% of baseline).
* With peer-support, throughput recovered to ~72 Mbps (about 77% of baseline), showing significant improvement in maintaining legitimate traffic under attack.

## 5. Discussion

The results of this experiment demonstrate the practical impact of the peer-support strategy in mitigating flow-table overloading attacks in SDN. Without mitigation, the victim switch (s1) was overwhelmed in about 12–13 seconds, leading to severe throughput degradation and disruption of legitimate traffic. With peer-support enabled, holding time nearly doubled to ~24 seconds, providing valuable additional time for detection and response.

The CPU utilisation measurements further highlight the trade-off of the approach. While the controller’s workload increased slightly (around 3% more on average) due to additional decision-making, the switch load decreased marginally because flows were shared across peers. This indicates that the strategy distributes resource consumption more evenly across the network without introducing significant overhead.

Bandwidth tests also confirmed the effectiveness of peer-support. In the baseline scenario, legitimate throughput between h1\_1 and h10\_1 was close to line rate (~94 Mbps). Under attack without mitigation, throughput collapsed to ~28 Mbps (30% of baseline). With peer-support, throughput recovered to ~72 Mbps, or 77% of baseline. This recovery illustrates that peer-support preserves service quality for legitimate users even when switches are under attack.

Overall, these findings align with prior research and validate the idea that peer collaboration can substantially enhance SDN resilience. The strategy is simple to implement, incurs only a minor processing cost, and significantly improves both survivability and service performance under attack. However, it is important to note that the static peer mapping used in this test may not scale efficiently to larger or more dynamic topologies. A more advanced approach, such as dynamic peer selection based on real-time network state, could further improve robustness.

## 6. Conclusion

This project investigated the vulnerability of SDN switches to flow-table overloading attacks and evaluated the peer-support mitigation strategy in a 10-switch mesh topology controlled by POX. Experimental results showed that peer-support nearly doubled the holding time, reduced the impact of attacks on throughput, and introduced only minor additional CPU load.

The findings confirm that peer-support is an effective, lightweight mechanism to strengthen SDN against flow-table exhaustion. It leverages collaboration between switches to balance resource usage and preserve service availability. While static peer mapping was sufficient for this study, future work could explore dynamic peer selection and scaling to larger, real-world topologies.

In conclusion, peer-support provides a practical and efficient defense that improves the resilience of SDN networks against flow-table overloading attacks, making it a promising component of future SDN security strategies.

## References

1. Tupakula, U., Varadharajan, V., & Karmakar, K. K. (2020). Attack detection on the software defined networking switches. In 2020 6th IEEE Conference on Network Softwarization (NetSoft) (pp. 262–266). IEEE. https://doi.org/10.1109/NetSoft48620.2020.9165459
2. Sood, K., Yu, S., & Xiang, Y. (2016). Performance analysis of software-defined network switch using M/Geo/1 model. IEEE Communications Letters, 20(12), 2522–2525. https://doi.org/10.1109/LCOMM.2016.2608894
3. McCauley, J., et al. (2013). POX: OpenFlow/SDN controller. GitHub repository. Retrieved from https://github.com/noxrepo/pox
4. Mininet Team. (2015). Mininet: An Instant Virtual Network on your Laptop (or other PC). Retrieved from http://mininet.org