**DoS Attack Detection and Prevention**

**A SDN Course Project Report**

***Submitted by***

**Konica Ranjan**

**201000019**

**CSE**

**Emany Sri Revant**

**201000042**

**CSE**

***Under the guidance of***

**Dr. Venkanna U.**

**(Assistant Professor, IIITNR)**



**Dr. Shyama Prasad Mukherjee International Institute of Information Technology, Naya Raipur**

**April 2023**

**Abstract (100–150 words)**

A denial-of-service (DoS) attack is a type of cyberattack in which an attacker attempts to disrupt the normal operation of a targeted computer system or network, making it unavailable to users. This is typically accomplished by overwhelming the targeted system with traffic or requests, causing it to crash or become unresponsive. DoS attacks can be carried out through a variety of methods, including flooding the target with traffic from multiple sources, exploiting vulnerabilities in the target's software or hardware, or using malware to take control of the target system and use its resources to carry out the attack. The consequences of a successful DoS attack can be severe, ranging from temporary inconvenience to serious financial or reputational damage. Organizations can protect themselves from DoS attacks by implementing security measures such as firewalls, intrusion detection systems, and content filtering. Additionally, having a disaster recovery plan in place can help organizations minimize the impact of a DoS attack and quickly restore normal operations. It is important to note that DoS attacks are illegal and can result in criminal charges for the attacker.

1. **Introduction**

Software-defined networking (SDN) is an architecture that separates the control plane from the data plane in network devices, allowing network administrators to manage network traffic flows centrally from a software-based controller. This approach offers greater flexibility and control over network traffic, allowing for more efficient management of network resources and better network performance.

Denial-of-service (DoS) attacks are a significant risk for both traditional and software-defined networking (SDN) architectures. However, the risks associated with these attacks can differ for each architecture. In traditional networks, DoS attacks typically target the network layer or the transport layer, with the goal of overwhelming the network's bandwidth or resources. When a DoS attack is in progress, the network bandwidth can become saturated, which means that the available bandwidth is fully utilized and no more traffic can be transmitted. This can lead to slow network performance, dropped packets, and increased latency. In some cases, the network may become completely unresponsive, making it impossible to access any services or resources. Moreover, a DoS attack can also cause a cascading effect, affecting other networks and servers that are connected to the target network. For example, if a web server is under DoS attack, it may become unresponsive, leading to slow or disrupted access to web pages. This can, in turn, affect other services or applications that rely on the web server, such as email or database services.

In SDN architecture, DoS attacks can target both the control plane and the data plane, with the goal of overwhelming the network's processing capabilities. This can cause disruption to the network's functionality and lead to financial loss as well as damage to the organization's reputation. In addition, SDN architecture is more vulnerable to attacks because it is centralized, making it a single point of failure.

One way to mitigate these risks is by limiting the number of packets that can be transmitted through a switch. By setting a maximum packet limit, the switch can prevent a DoS attack from overwhelming the network by limiting the amount of traffic that can be sent to a particular destination. This approach is known as "rate limiting,", and it can be implemented using various techniques, such as packet dropping, flow redirection, or bandwidth throttling. Implementing packet rate limiting requires a comprehensive understanding of network traffic patterns and behavior, as well as an understanding of the types of attacks that are most likely to occur. Additionally, SDN controllers must be configured to monitor network traffic and dynamically adjust packet limits in response to changing network conditions.

1. **Research problem (100 words)**

Distributed Denial of Service (DDoS) attacks have become a major threat to modern networks, and software-defined networking (SDN) architectures are not immune to this threat. In this research problem, we aim to investigate the effectiveness of rate-limiting packets in preventing DDoS attacks in SDN environments. The proposed solution involves controlling the flow of packets based on a predetermined threshold, which helps to mitigate the effects of the attack by limiting the amount of traffic that the target network has to process.

To implement this solution, we will use the P4 programming language, which allows us to create custom packet processing pipelines in SDN switches. We will implement a rate-limiting function in the switch that will monitor incoming packets and drop them if the incoming traffic rate exceeds the specified threshold. We will evaluate the effectiveness of the proposed solution by simulating various DDoS attack scenarios and measuring the impact on network performance.

The main goal of this research problem is to demonstrate that rate-limiting packets can be an effective approach to preventing DoS attacks in SDN environments. By limiting the amount of traffic that a target network has to process, we can reduce the impact of DoS attacks and maintain the network's availability and performance.

1. **Major contributions (two or three in bullet form)**

* The proposed solution uses a global timestamp to keep track of the number of packets received in a specified time window. If the number of packets exceeds a threshold, the program drops the packets.
* extraction of the IP address of the DoS attacker using the get\_digest program.

1. **Methodology (3-4 pages)**

**(Include block diagrams and algorithms.)**

1. **Creating topology**

The two hosts, Host 1 and Host 2, are connected to the switch. The switch is connected to the SDN controller, which manages the flow of traffic in the network.

In this architecture, the SDN controller is responsible for configuring the switch to forward traffic between the two hosts. The controller uses OpenFlow to communicate with the switch and to set up forwarding rules based on the network policies.

When Host 1 sends a packet to Host 2, the packet is first sent to the switch. The switch then forwards the packet to the SDN controller, which determines the best path for the packet to take and configures the switch to forward the packet to Host 2.

Overall, an SDN architecture with 2 hosts, 1 switch, and 1 controller provides a flexible and centralized approach to network management and can be easily customized to meet specific network requirements.

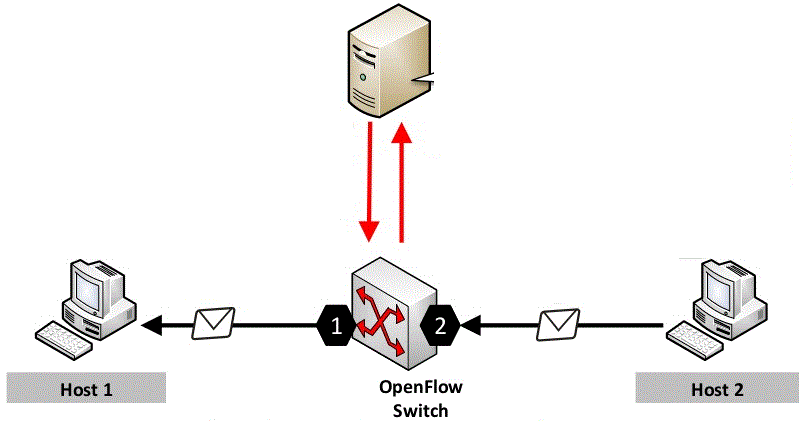


Fig. 1. Topology

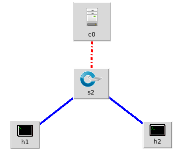


Figure 2. Topology made using miniedit

1. **Mimicked a DoS attack.**

Performing a DoS attack from host 1 to host 2 by sending 15000 packets per second is a simulated scenario where the attacker sends a high volume of packets to the victim host in a short time, which may cause the victim's network or application to become unavailable to legitimate users.

In this scenario, the attacker (host 1) sends 15000 packets per second to the victim host (host 2). The attacker's goal is to overwhelm the victim's network or application by consuming all of its resources and causing it to crash or become unavailable to legitimate users.

1. **Observation of network traffic, bandwidth, etc. in the network**

used the Iperf command to check the bandwidth between links h1 and h2. Iperf is a command-line tool that can measure network performance by generating traffic between two endpoints.

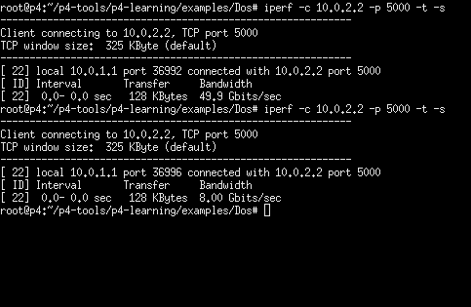


Fig. 3: The bandwidth between switch and host2 before and during the DoS attack

From Fig. 1, the bandwidth between switch and host 2 before the attack is 49.9 Gbits/sec, whereas during the attack it is 8.00 Gbits/sec.

It can be noted that there has been a significant drop in the bandwidth between the links when the attacker has performed a DoS attack on host 2.

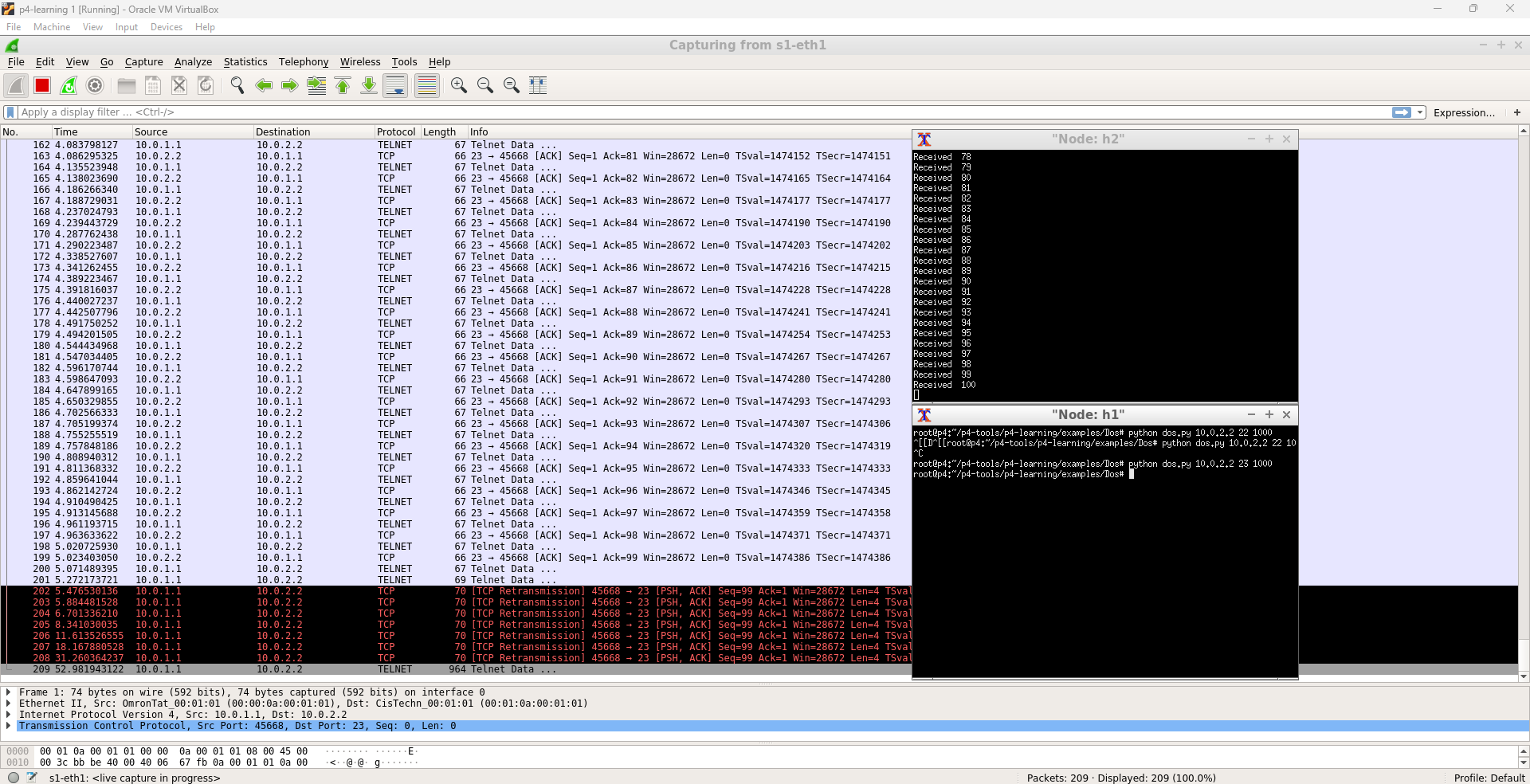


Fig. 4: Used Wireshark to analyze the traffic before and during the attack.

Wireshark is a network protocol analyzer tool that allows you to capture and analyze network traffic.

1. **Calculating the interval between the arrival of packets before and during the DoS attack using the p4 language**

A 48-bit register is created that stores the timestamp value of the first packet (tm1). A counter is also maintained that counts the number of packets. When the packet count reaches 101, the timestamp of that packet (tm101) is subtracted from (tm1), and the difference is compared with the time threshold (tthres), which is set to 6 milliseconds.

This threshold is set to 6 milliseconds by assuming that in any DoS attack, the rate is 15,000 packets per second, so if a DoS attack is happening, then 100 packets will reach within 6 milliseconds.

If (tm101 - tm1) < tthres, then we identify that the DoS attack is happening and we drop the remaining packets; otherwise, we forward all the remaining packets.

This process is executed by this if statement in the apply block of the p4 program:

if ( (meta.counter\_one > PACKET\_THRESHOLD && meta.counter\_two > PACKET\_THRESHOLD && standard\_metadata.ingress\_global\_timestamp-meta.tm1<6) ){

drop\_table.apply();

return; }

1. **Limiting the packets in case of an attack detected in the P4 language**

Heavy-hitter detection and mitigation are key strategies to prevent DoS attacks in SDN networks. In this project, we propose a P4-based solution that detects heavy hitters and limits the number of packets sent from these sources to prevent DoS attacks. The P4 program includes a hash-based data structure to keep track of packet counts for each source IP address. When a heavy hitter is detected, the program uses a rate limiter to reduce the number of packets sent from that source, preventing it from overwhelming the network.

The P4 program also includes a configurable threshold to adjust the sensitivity of the heavy-hitter detection algorithm. We evaluate our solution using the BMv2 software switch and a custom topology and show that it effectively limits the number of packets sent by heavy hitters, preventing DoS attacks while allowing legitimate traffic to flow through the network.

1. **Obtaining the IP address of the attacker**

The get\_digest program in P4 is used to calculate a hash function of packets flowing through the switch, allowing for efficient tracking of statistics about the network traffic. This can be useful for identifying heavy hitters or identifying the source IP address of a potential attacker during a DDoS attack.

1. **Simulation results and discussion**

Scenario 1:

(tm101 - tm1) < tthres:

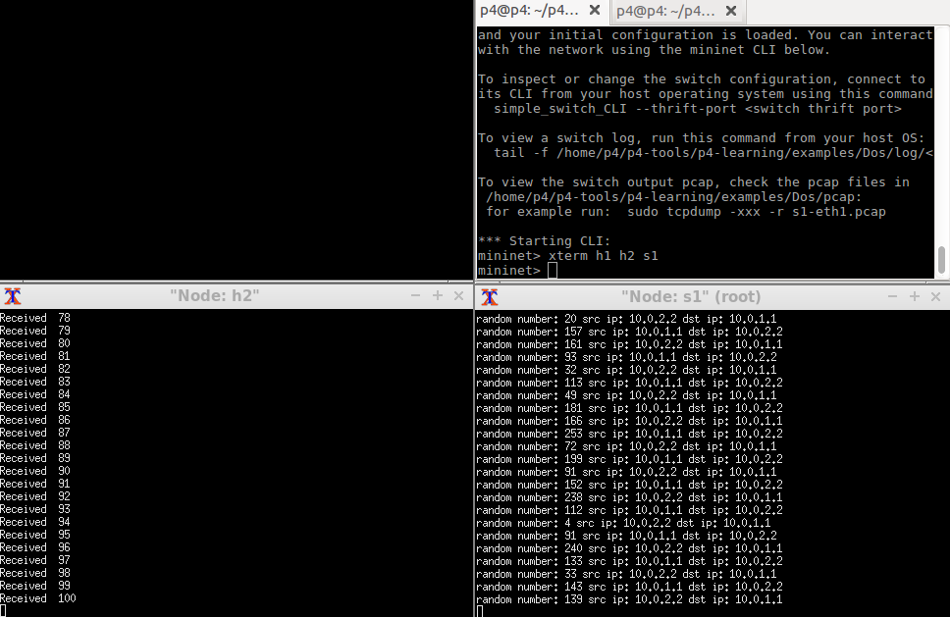
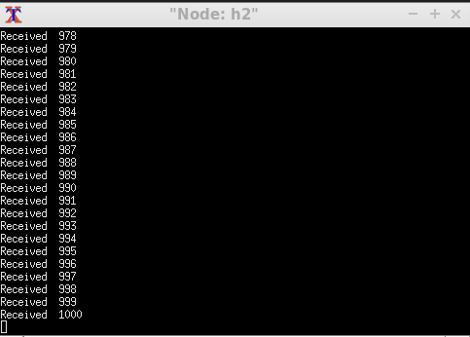


Fig 5

In case of global\_timestamp difference the 101th packet and 1st packet is less than the threshold value(15,000 packets per second) the p4 program detected DoS attack and dropped the remaining packets.

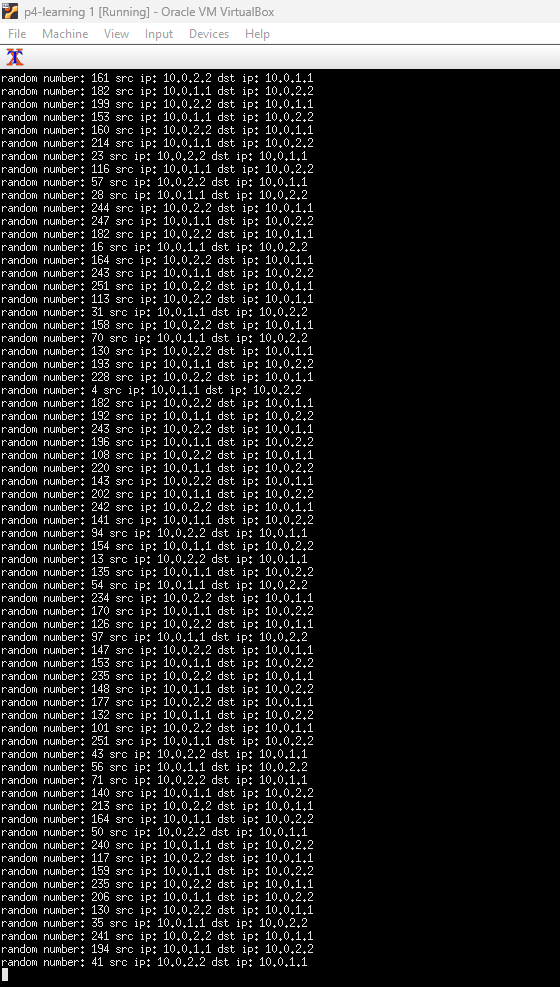
Scenario 2:

(tm101 - tm1) > tthres:

Fig 6

In scenario 2, the packets were sent at 1500 packets per second. The global\_timestamp difference between the 101th packet and 1st packet exceeded the 6 milliseconds, so the none of the packets were dropped.

Get\_digest program to track the ip address of the sender:



1. **Summary**

The use of Software-Defined Networking (SDN) has increased network flexibility and management, but it has also created new vulnerabilities, such as denial-of-service (DoS) attacks. In this context, the proposed work uses a global timestamp in a P4 program to keep track of the number of packets received in a specified time window. If the number of packets exceeds a threshold, the program drops the packets to prevent the DoS attack. The P4 program also uses a get\_digest function to obtain information about the IP address of the DoS attacker.

The experiment was conducted using Mininet, a network emulation tool, where a DoS attack was simulated by sending 1500 packets per second from host 1 to host 2. The results showed that the proposed method significantly reduced the number of received packets during the DoS attack, maintaining a stable bandwidth for legitimate traffic.

Overall, the use of a global timestamp and a threshold-based approach can be effective in preventing DoS attacks in SDN architectures without disrupting legitimate network traffic. The get\_digest function can also help identify the IP address of the DoS attacker, enabling further action against the attacker. This method can be beneficial in securing networks that heavily rely on SDN for their operation, ensuring network reliability and availability.

1. Code:

<https://drive.google.com/drive/folders/1pUN4SbpDecvctrJZevg0jlhfQBjvA7ZP?usp=sharing>

References

1. D. Kreutz, F. M. Ramos, P. Verissimo, C. E. Rothenberg, S. Azodolmolky, and S. Uhlig, *Software-defined networking: A comprehensive survey Computing Research Repository (CoRR)*, June 2014.
2. M. Antikainen, T. Aura, and M. Särelä, "Spook in Your Network: Attacking an SDN with a Compromised OpenFlow *The 19th Nordic Conference on Secure IT Systems (NordSec 2014)*, October 20.
3. D. Kreutz, F. M. Ramos, and P. Verissimo, "Towards Secure and Dependable Software-Defined Networks", *Proceedings of the Second ACM SIGCOMM Workshop on Hot Topics in Software-Defined Networking, Ser. HotSDN '13*,pp. 55–6000, 2013.
4. R. Klöti, V. Kotronis and P. Smith, "OpenFlow: A security analysis", *Network Protocols (ICNP) 2013 21st IEEE International Conference*, pp. 1-6, Oct 2013.
5. *Mininet*, [online] Available: <http://www.mininet.org/>.
6. *OpenFlow. Open Networking Foundation (ONF)*, [online] Available: <https://www.opennetworking.org/>.
7. "Open Networking Foundation (ONF)", *Technical Specification*, [online] Available: <https://www.opennetworking.org/images/stories/downloads/sdnresources/onf-specifications/openflow/openflow-spec-v1.3.0.pdf>.
8. K. Benton, L. J. Camp and C. Small, "OpenFlow Vulnerability Assessment", *Proceedings of the Second ACM SIGCOMM Workshop on Hot Topics in Software Defined Networking ser. HotSDN '13*, pp. 151-152, 2013.