Network Topology

The network topology for this project is designed and implemented using Mininet, a network emulation tool that enables the creation and testing of virtual network environments. The topology consists of three key components: controllers, switches, and hosts. These components are interconnected to simulate a structured and efficient network.

Controllers

The topology employs three remote controllers implemented using the Ryu SDN framework. These controllers are responsible for managing the network's operation and ensuring smooth communication between devices. Each controller is linked to two switches, dividing the network into three manageable segments. This segmentation allows for better load balancing and fault isolation.

Switches

Six switches are utilized in this topology, all of which support OpenFlow version 1.3. OpenFlow serves as the communication protocol between the switches and the controllers. The switches are distributed among the controllers as follows:

- **Controller 1:** Manages Switch 1 and Switch 2.
- **Controller 2:** Manages Switch 3 and Switch 4.
- **Controller 3:** Manages Switch 5 and Switch 6.

This hierarchical arrangement ensures that each controller oversees a specific part of the network, reducing complexity and improving overall performance.

Hosts

The topology includes ten hosts, which are end devices that connect to the switches. These hosts are distributed across the switches as follows:

- **Switch 1:** 2 hosts
- **Switch 2:** 2 hosts
- **Switch 3:** 1 host
- **Switch 4:** 1 host
- **Switch 5:** 2 hosts
- **Switch 6:** 2 hosts

The allocation of hosts ensures balanced network utilization and prevents congestion. Hosts connected to the same switch can communicate directly, while inter-switch communication is managed by the controllers.

Design Considerations

This network topology is designed to provide scalability, fault tolerance, and efficient load balancing. By dividing the network into segments managed by separate controllers, the topology minimizes the risk of a single point of failure. Additionally, the use of OpenFlow v1.3 enables flexible and programmable control of the switches, allowing for dynamic adjustments to network policies and configurations.

The topology provides a robust platform for simulating various networking scenarios, including traffic engineering, quality of service (QoS) management, and failure recovery, making it an ideal setup for testing and experimentation.

Balanced Multicontroller SDN Network

In this project, a balanced multicontroller SDN network is implemented to efficiently monitor and manage the traffic inflows across the network. The system works as follows:

Traffic Monitoring

Each switch in the network is configured to collect and monitor the packet inflows to measure the total traffic handled. A specific threshold value for inflow traffic is determined for each switch. When traffic inflow approaches or exceeds this threshold, it indicates potential congestion or overload at that switch.

Data Collection and Reporting

The controllers collect traffic information from all their associated switches. This data is then transmitted by each controller to a centralized Kafka messaging system. Kafka acts as a message broker, aggregating traffic information from all controllers into a unified data stream. This setup ensures real-time and scalable monitoring of the network state.

Load Balancing and Migration

Using the aggregated data, Kafka has the logic to identify overloaded controllers and determines which switch is contributing to the overload. Once identified, the system initiates a migration process where the overloaded switch is reassigned to a less burdened controller. Kafka communicates this migration decision to all controllers, ensuring that the updated controller-switch assignments are synchronized across the network.

This approach ensures optimal utilization of controllers and minimizes the risk of congestion, leading to a more efficient and reliable network performance.

Another approach can be to use another master controller instead of Kafka which is in review.

DDoS Detection Using Machine Learning Algorithms

The project incorporates a robust system for detecting Distributed Denial of Service (DDoS) attacks by leveraging advanced machine learning algorithms. The methodology involves creating a custom dataset within the simulated environment to capture the

dynamic nature of network traffic under both normal and attack scenarios. This dataset forms the foundation for training and evaluating multiple machine learning models.

Dataset Creation

The dataset is generated by monitoring the packet flows across the network topology, including features such as source and destination IP addresses, packet sizes, protocols, flow durations, and timestamps. Both normal traffic patterns and simulated DDoS attack patterns are recorded to ensure a balanced dataset. This enables the machine learning models to effectively distinguish between benign and malicious traffic.

Model Training and Evaluation

The XGBoost (Extreme Gradient Boosting) algorithm is one of the primary models trained on the dataset due to its efficiency in handling large-scale and high-dimensional data. The training process involves splitting the dataset into training and testing sets to evaluate the model's performance using metrics like accuracy, precision, recall, and F1-score.

In addition to XGBoost, other models such as Random Forest, Support Vector Machines (SVM), and Neural Networks are being explored to identify the most effective algorithm for DDoS detection. Each model is optimized through hyperparameter tuning to maximize its predictive accuracy.

Detection

Once a suitable model is identified, it is deployed to perform inference on live network traffic. The system collects packet-level attributes from switches and forwards them to the deployed model for classification. The model analyzes these attributes to identify anomalies that indicate potential DDoS attacks. Detected anomalies are flagged and logged, and appropriate mitigation measures are initiated.

Mitigation

After the traffic has beem deemed malicious, the mitigation begins by analyzing the flow control and extract the information to determine the source of the attack. After the source and corrupted switch has been identified ,the mitigation module will request the controller to block the traffic flows from the corrupted switch and the port. Suppose the switch with switch-id-1 and port 1 is currently receiving this hazardous traffic flow. The Mitigation Module will send a request to the Controller to instruct switch-id-1 to remove the traffic flows coming from port 1. Simultaneously, the Mitigation Module sends continuous alerts to the administrator for system monitoring. After the attack flow concludes, after some time, the system will automatically re-establish the connection to port 1 to maintain system operation.

Future work includes expanding the dataset with more diverse attack scenarios, testing additional machine learning models, and incorporating ensemble methods to further enhance detection capabilities. By integrating these advancements, the system aims to provide a robust and adaptive defense mechanism against DDoS attacks.