DDOS DETECTION USING MULTI AGENT/REINFORECEMENT LEARNING

Limitations of Traditional Machine Learning for DDoS Detection

1. **Static Models**: ML models are trained on historical data and lack the ability to adapt to new, evolving attack patterns, making them ineffective against the dynamic nature of DDoS attacks.
2. **Extensive Feature Engineering**: ML requires extensive feature engineering to capture relevant patterns, which is time-consuming and may not generalize well across different network environments or novel attacks.
3. **Batch Learning**: ML models generally operate in a batch learning mode, training on a fixed dataset and lacking the ability to adapt in real-time to ongoing changes in network traffic.
4. **Signature-Based Detection**: Many ML models rely on signature-based detection, which identifies known patterns associated with attacks. This method struggles to detect new, previously unseen attacks and is vulnerable to evasion techniques by attackers.
5. **Supervised Learning Limitation**: Most ML techniques depend on labelled data, which can be difficult to obtain in sufficient quantities for all possible DDoS attack types.

Advantages of Reinforcement Learning Over Traditional ML

1. **Adaptive Learning**: RL agents continuously learn and adapt their behaviour based on real-time feedback from the environment, making them effective against dynamic and non-stationary network conditions.
2. **Real-Time Decision Making**: RL is designed for real-time decision-making, critical for the timely detection and mitigation of DDoS attacks.
3. **Exploration and Exploitation**: RL balances exploration (trying new actions) and exploitation (using known actions) to discover optimal defence strategies without requiring extensive prior knowledge.
4. **Minimal Feature Engineering**: RL can learn directly from raw network traffic data, reducing the need for extensive feature engineering and enabling the agent to identify subtle patterns indicative of attacks.
5. **Per-Flow Adaptation**: RL agents can adapt their policies for each individual traffic flow, providing precise control over traffic and improving the detection of malicious activities while minimizing the impact on legitimate traffic.
6. **Temporal Difference Learning**: RL uses temporal difference (TD) learning methods, which update the value of states based on observed transitions, enabling efficient learning of optimal policies. TD combines the strengths of Monte Carlo methods and dynamic programming, facilitating online learning without a complete model of the environment.

Key Factors Making RL Better for DDoS Detection

1. **Non-Stationary Environments:** RL adapts to changing traffic patterns and evolving attack strategies, maintaining effectiveness over time.
2. **Temporal Dependencies:** RL algorithms capture temporal dependencies in network traffic, enabling better understanding and reaction to ongoing flow behaviours.
3. **Scalability:** Multi-agent RL systems can scale across large networks, with agents operating locally and sharing information to manage extensive and complex network environments.
4. **Safe Exploration**: RL allows safe exploration strategies to minimize the risk of disrupting legitimate traffic while learning effective defence policies.
5. **Goodput Optimization:** RL agents focus on optimizing goodput, ensuring that legitimate traffic is maintained while minimizing the impact of DDoS attacks.

Training RL for DDoS Detection

State Representation: Efficient state representation is crucial. RL agents should use features like traffic volume, packet sizes, flow durations, temporal patterns, and goodput metrics to represent the network state accurately.

Feature Extraction:

Tile Coding: This technique discretizes the state space into tiles, allowing the RL agent to handle large and continuous state spaces efficiently. By mapping continuous variables to discrete tiles, the agent can generalize across similar states, improving learning efficiency**.**

Linear Function Approximation: This method approximates the value function, enabling the RL agent to handle high-dimensional state spaces. It simplifies the learning process by representing the value function as a linear combination of features, making it computationally efficient and scalable.

Handling Different Attack Types:

TCP Attacks: RL should focus on detecting anomalies in connection patterns, handshake processes, and sudden spikes in connection attempts. TCP attacks, such as SYN floods, can be mitigated by monitoring and adapting to these patterns.

UDP Attacks: These attacks typically involve high-volume traffic to random ports. RL agents should monitor for unusual spikes in UDP traffic and adapt quickly to changes in traffic volume and packet size distributions. Due to the stateless nature of UDP, RL agents need to prioritize monitoring traffic patterns and packet rates over connection-oriented features**.**

Hyperparameter Tuning:

Number of Agents: Optimizing the number of agents balances the load across the network and ensures comprehensive coverage.

Message Dropout Fraction: Adjusting the dropout rate helps control communication overhead and avoid overfitting in distributed RL systems.

Exploration Rate: Properly tuning the exploration rate (ε) ensures agents explore enough to discover effective policies without destabilizing the network.

Single-Destination vs. Multiple-Destination Networks

1. **Single-Destination Networks**: In a tree-structured network with a single server, RL agents manage traffic through hierarchical switches. The challenge is optimizing traffic flow towards a single point of aggregation, ensuring minimal disruption from DDoS attacks while maintaining service quality for legitimate users.
2. **Multiple-Destination Networks:** In a fat-tree topology with multiple servers, RL agents face the complexity of distributing efforts across various destinations. This requires balancing traffic loads and defending multiple critical points simultaneously. Effectiveness hinges on agents' ability to coordinate and share information efficiently, adapting policies based on traffic directed towards each server.

Reward System:

Reward Design: Rewards should reflect the goal of minimizing DDoS impact while maintaining legitimate traffic flow, penalizing dropped legitimate packets, and rewarding successful mitigation actions.

Sparse Rewards: Given the infrequent nature of attacks, the reward system should handle sparse rewards effectively, ensuring the agent learns from infrequent but significant events**.**