Collaborative Misbehaviour Detection in Distributed Networks

Filippos Chrysostomou

UNIVERSITY OF CYPRUS

COMPUTER SCIENCE DEPARTMENT

May 2026

UNIVERSITY OF CYPRUS

COMPUTER SCIENCE DEPARTMENT

Collaborative Misbehaviour Detection in Distributed Networks

Filippos Chrysostomou

Supervisor

Dr. Vasos Vassiliou

Thesis submitted in partial fulfilment of the requirements for the award of degree of Bachelor in Computer Science at University of Cyprus

I declare that this dissertation and any accompanying software are my own original work, conducted in full compliance with the University of Cyprus Code of Conduct for Research (https://www.ucy.ac.cy/legislation/kodikas-deontologias-kai-politikes/), and with full accountability on my part for the accuracy, integrity, and validity of all content.

May 2026

Acknowledgments

I would like to express my sincere gratitude to my supervisor, Dr. Vasos Vasiliou, for providing me with this interesting thesis topic and for his continuous guidance throughout the challenges I encountered during my research. I am especially grateful for our discussions during the thesis meetings, where he always encouraged my ideas and supported my way of thinking.

I would also like to thank Dr. Marios Thoma for kindly providing his implementation and for his valuable insights, which greatly helped me better understand the underlying concepts of his PhD work.

In addition, I would like to acknowledge Nikolaos Venizelou, whose thesis work served as a useful reference and helped me better structure and organize my own approach throughout this project.

Furthermore, I would like to thank the Department of Computer Science and all the teaching staff for their support and for sharing their knowledge and passion for the field throughout my studies.

Finally, I would like to express my deepest gratitude to my family for their constant support and encouragement, as well as to my friends, with whom I have shared unforgettable experiences during my university years.

Abstract

Distributed Denial of Service (DDoS) attacks remain a major threat to the availability of networked systems, as attackers continuously evolve their strategies to evade detection. A key challenge in this domain is distinguishing between malicious coordinated attacks and legitimate high-volume traffic, such as flash crowd events.

This thesis builds upon the observation that coordinated malicious users exhibit higher temporal synchronization compared to legitimate users. The main objective is to implement, evaluate, and extend this concept through a practical detection framework based on temporal activity patterns and graph-based similarity analysis.

A complete detection pipeline is developed, transforming raw network traffic into time-based activity representations and constructing similarity graphs to identify coordinated behavior. To enable controlled experimentation, synthetic traffic patterns are injected on top of real network traces, allowing systematic evaluation under varying conditions, including different levels of synchronization, noise, and attack intensity.

In addition, a probabilistic temporal modeling approach based on Hidden Markov Models (HMM) is introduced to capture dependencies between successive observations. Experimental results demonstrate that incorporating temporal information significantly improves detection performance.

In particular, the critical synchronization threshold required for reliable DDoS detection is significantly reduced when using temporal modeling, although slightly higher synchronization levels are required under unseen conditions. These findings highlight the importance of temporal correlation in distinguishing coordinated attacks from legitimate traffic and demonstrate the effectiveness of combining structural and temporal analysis.

Contents

FIGURES

TABLES

1 Introduction

1.1 Problem: DDoS vs Flash Crowds

Distributed Denial of Service (DDoS) attacks remain one of the main threats to the availability of modern network services. In these attacks, multiple distributed entities generate a large amount of traffic in order to overwhelm a target system and prevent legitimate users from accessing it.

One of the main difficulties in DDoS detection is distinguishing malicious attacks from legitimate high-traffic situations, known as flash crowds. Flash crowds may appear during online events, breaking news, or sudden increases in user interest, where many legitimate users access the same service simultaneously. As a result, both DDoS attacks and flash crowds can produce similar traffic characteristics, including increased request rates and high traffic volume.

Many traditional detection approaches mainly focus on traffic intensity or request frequency. However, these features alone are often insufficient for separating malicious behavior from legitimate activity. In practice, an important difference between the two scenarios is the degree of coordination among participating users. Attackers involved in DDoS campaigns usually behave in a highly synchronized manner, while users participating in flash crowds tend to generate activity more independently over time.

Based on this observation, this work focuses on distinguishing coordinated malicious behavior from legitimate high-volume traffic through the analysis of temporal synchronization patterns among users.

1.2 Motivation

The increasing frequency and sophistication of DDoS attacks make their accurate and timely detection a critical requirement for maintaining the availability and reliability of networked systems. As attackers continue to evolve their techniques, traditional detection methods based solely on traffic volume or simple thresholds become less effective, especially in environments where legitimate high-traffic events are common.

One of the main challenges in this domain is the lack of clear separation between malicious and legitimate behavior. Flash crowd events, which are entirely benign, can generate traffic patterns that closely resemble those of DDoS attacks. This overlap makes it difficult for conventional detection systems to accurately classify traffic without producing false positives or false negatives.

Furthermore, the majority of publicly available network datasets do not provide sufficient diversity or controlled scenarios to thoroughly evaluate detection mechanisms under varying conditions. This limitation motivates the need for a more flexible and controlled experimental framework that allows the systematic study of different attack characteristics, such as synchronization, noise, and intensity.

In this context, analyzing temporal synchronization patterns between users provides a promising direction for distinguishing coordinated malicious behavior from legitimate traffic. This work is motivated by the need to explore and validate such approaches through practical implementation and extensive experimentation.

1.3 Contribution

This thesis presents a modular framework for the detection and analysis of coordinated network behavior, with a particular focus on distinguishing Distributed

Denial of Service (DDoS) attacks from legitimate high-traffic events such as flash crowds. The proposed approach introduces a unified data processing pipeline that converts raw network traffic into a discretized temporal activity representation, enabling the analysis of IP behavior across fixed time windows. Based on this representation, a graph-based feature extraction methodology is developed, where pairwise similarity between IPs is computed using Jaccard similarity and structural properties such as edge density and the size of the largest connected component are used to capture coordinated behavior.

Building on these features, three detection approaches are implemented and evaluated: a rule-based baseline detector, an enhanced version with temporal smoothing, and a probabilistic Hidden Markov Model (HMM) that captures temporal dependencies across consecutive time windows. In order to systematically evaluate the proposed methods, a controlled experimental framework is designed, combining real network traffic with synthetically injected flash crowd and DDoS scenarios. This hybrid setup allows the analysis of system performance under varying levels of synchronization, noise, and traffic intensity.

Furthermore, the proposed HMM-based approach enables improved detection accuracy and reduced detection delay by leveraging sequential patterns and state transitions, overcoming the limitations of per-window decision mechanisms. In addition to detection, the framework incorporates an attacker identification mechanism based on graph connectivity, where coordinated IPs are identified through the largest connected component during detected attack periods. Finally, a comprehensive evaluation methodology is presented, including detection accuracy, false negative rates, confusion analysis, detection delay, and attacker-level metrics such as precision, recall, F1-score, and coverage. The results demonstrate that synchronization is a key factor for both reliable detection and accurate attacker identification, highlighting the importance of combining structural and temporal analysis in the study of coordinated network attacks.

Internet traffic profiling methods can reveal coordinated temporal communication patterns and abnormal traffic behavior [10].

2 Background

2.1 DDoS Attacks

2.1.1 Definition

Distributed Denial of Service (DDoS) attacks are a class of cyberattacks that aim to disrupt the availability of a network service by overwhelming it with a large volume

of traffic. Unlike traditional Denial of Service (DoS) attacks, which originate from a single source, DDoS attacks are launched from multiple distributed systems, often forming a coordinated network of compromised devices known as a botnet.

In a typical DDoS scenario, attackers control a large number of infected machines and instruct them to send a high volume of requests to a target server simultaneously. This excessive traffic exhausts the server's computational or network resources, preventing legitimate users from accessing the service.

DDoS attacks can target different layers of the network stack, including application-layer attacks (e.g., HTTP floods), transport-layer attacks (e.g., SYN floods), and volumetric attacks that aim to saturate network bandwidth. Regardless of the specific technique used, the primary objective remains the same: to degrade or completely deny service availability.

2.1.2 Challenges in Detection

Detecting Distributed Denial of Service (DDoS) attacks remains a challenging task due to several inherent characteristics of modern network traffic.

One of the primary challenges is the similarity between malicious attack traffic and legitimate high-volume events, such as flash crowds. Both scenarios can generate a large number of requests within a short time period, making it difficult to distinguish between them using traditional volume-based metrics.

Another challenge lies in the distributed nature of DDoS attacks. Since traffic originates from multiple sources, often geographically dispersed, it becomes harder to identify a single point of origin or apply simple filtering techniques.

Furthermore, attackers continuously adapt their strategies to evade detection. Techniques such as low-rate attacks, randomization of request patterns, and partial synchronization can significantly reduce the effectiveness of traditional detection mechanisms.

In addition, real-world network traffic is inherently noisy and dynamic, which can lead to false positives and false negatives in detection systems. Variability in user behavior, network conditions, and background traffic further complicates the identification of coordinated malicious activity.

These challenges highlight the need for more advanced detection approaches that go beyond simple traffic statistics and incorporate behavioral and temporal analysis.

Machine learning approaches for intrusion detection often face generalization challenges when applied to real-world network traffic conditions [8].

2.1.3 Flash Crowds

Flash crowd events refer to situations where a large number of legitimate users simultaneously access a network service, typically due to popular events, breaking news, or viral content. These events generate a sudden surge in traffic, which can place significant strain on system resources, similar to DDoS attacks.

From a detection perspective, flash crowds present a major challenge, as they exhibit traffic characteristics that closely resemble those of malicious attacks. High request rates, increased number of active users, and sudden spikes in traffic volume are common in both scenarios. As a result, traditional detection mechanisms that rely solely on traffic intensity or volume may fail to distinguish between legitimate and malicious behavior.

Moreover, incorrectly classifying flash crowd traffic as a DDoS attack can lead to unnecessary mitigation actions, potentially blocking legitimate users and degrading the quality of service. Therefore, accurate differentiation between flash crowds and DDoS attacks is essential for maintaining both security and service availability.

2.1.4 Similarities with DDoS Attacks

Flash crowds and DDoS attacks share several observable characteristics at the network level, which makes their differentiation particularly challenging. In both cases, the system experiences a large number of incoming requests within a short period of time, leading to increased load and potential service degradation.

One of the key similarities lies in the aggregate behavior of users. Both scenarios involve multiple clients interacting with the same target, often resulting in similar traffic patterns when viewed from a high-level perspective. This similarity can make traditional detection approaches based on traffic volume, connection counts, or request frequency unreliable.

However, despite these similarities, an important distinction exists in the underlying behavior of participants. In flash crowds, users act independently, with no coordination in their actions, leading to more random and less synchronized traffic patterns. In contrast, DDoS attacks typically involve coordinated entities, such as botnets, where multiple nodes act in a synchronized manner, producing highly correlated activity over time.

This difference in synchronization forms the foundation for more advanced detection approaches, including the one explored in this work.

Traffic flow anomalies and abnormal synchronization patterns have been extensively studied in network traffic analysis research [1].

Distributed Denial-of-Service attacks can appear in multiple forms depending on attacker coordination strategies and traffic generation behavior [5].

2.2 Synchronization-based Detection

Traditional approaches to DDoS detection primarily rely on traffic volume, request rates, or statistical anomalies. However, as previously discussed, such methods often fail to distinguish between malicious attacks and legitimate high-traffic events, such as flash crowds.

An alternative approach, introduced in previous work, focuses on the concept of temporal synchronization among users. The key idea is that coordinated malicious entities, such as bots in a DDoS attack, tend to exhibit similar patterns of activity over time, as they are often controlled by a central command or follow predefined attack strategies. This results in a higher level of synchronization in their request behavior.

In contrast, legitimate users participating in a flash crowd event act independently, without coordination. As a result, their activity patterns are more random and less correlated over time. This fundamental difference provides a basis for distinguishing between malicious and benign traffic, beyond simple traffic volume analysis.

The synchronization-based detection model proposes analyzing user activity within discrete time intervals and comparing behavioral patterns across multiple users. By measuring the similarity of activity patterns, it is possible to identify groups of users that exhibit coordinated behavior, which may indicate the presence of a DDoS attack.

While this approach provides a strong theoretical foundation, previous implementations were limited in terms of practical evaluation and did not fully explore the impact of different levels of synchronization, noise, and attack variability. In particular, factors such as partial synchronization, temporal jitter, and packet loss can significantly affect the detectability of coordinated behavior.

In this work, we build upon this concept by implementing a graph-based detection framework that captures synchronization through similarity measures between user activity patterns. Furthermore, we extend the original approach by introducing controlled experimental scenarios and incorporating temporal probabilistic modeling, allowing for a more comprehensive analysis of synchronization-based detection under realistic conditions.

2.3 Hidden Markov Models

Hidden Markov Models (HMMs) are probabilistic models that are widely used to represent systems that evolve over time, where the underlying state of the system is not directly observable. Instead, the model relies on observable data that are assumed to be generated by these hidden states.

An HMM consists of a set of hidden states, a set of observable variables, and a set of probabilities that define transitions between states and the likelihood of observing certain outputs given a state. At each time step, the system is assumed to be in one of the hidden states, and it produces an observation according to a probability distribution associated with that state.

One of the key properties of HMMs is the Markov assumption, which states that the current state depends only on the previous state and not on the full history of past states. This allows the model to capture temporal dependencies in a simplified and computationally efficient manner.

In the context of network traffic analysis, HMMs can be used to model the evolution of system behavior over time. For example, different hidden states may correspond to normal operation, flash crowd conditions, or DDoS attack scenarios, while the observations may consist of measurable features such as activity levels, similarity metrics, or graph-based properties.

A common task when using HMMs is to determine the most likely sequence of hidden states given a sequence of observations. This is typically achieved using the Viterbi algorithm, which efficiently computes the most probable state sequence.

In this work, Hidden Markov Models are used to incorporate temporal information into the detection process, allowing the model to account for dependencies between successive time windows and improve the robustness of synchronization-based detection.

Network anomaly detection is a well-established research area with applications in intrusion detection, attack discovery, and abnormal traffic analysis [2].

2.4 Graph-based Botnet Detection

Graph-based approaches are widely used for detecting coordinated behavior in network traffic, particularly in the context of botnet and distributed attack detection.

The core idea is to represent entities (e.g., IP addresses) as nodes in a graph, and define edges between nodes based on a similarity or correlation measure. In the context of network traffic, similarity may capture temporal synchronization, shared activity patterns, or common communication targets.

In coordinated attacks such as DDoS, bots tend to exhibit highly synchronized behavior. This leads to the formation of dense subgraphs or strongly connected components, where many nodes are connected due to high similarity. In contrast, benign traffic or flash crowds may involve many active users, but without strong synchronization, resulting in sparse graph structures.

Several graph-based metrics have been proposed to characterize such behavior, including edge density and the size of the largest connected component. These metrics provide a compact way to capture collective behavior rather than analyzing individual flows independently.

Graph-based detection methods are particularly effective in distinguishing between coordinated attacks and uncoordinated high-volume traffic. However, they are typically applied in a static manner, analyzing each time interval independently. This limitation motivates the need for temporal models, such as Hidden Markov Models, which can capture the evolution of graph structures over time.

3 Methodology

3.1 Traffic Representation

The first step of the proposed methodology is the transformation of raw network traffic into a structured activity representation suitable for behavior analysis.

The input traffic is obtained from a PCAP file and processed packet by packet. For each packet, the source IP address and its timestamp are extracted. The timestamp of the first observed packet is used as a reference time, and all subsequent packet timestamps are discretized into fixed-length time slots of one second.

Formally, if t0 denotes the timestamp of the first packet and Δt the slot duration, each packet is mapped to a discrete slot index according to:

slot = floor((t - t0) / Δt)

where Δt = 1 second in our implementation.

Using this discretization, the traffic is represented as an activity map of the form:

IP → set(time slots)

This means that for each source IP address, we store the set of time slots during which the IP was active. Therefore, the representation does not store individual packets directly, but rather the temporal activity pattern of each IP.

To enable higher-level analysis, consecutive time slots are aggregated into fixed-length time epochs. In this work, each epoch consists of 60 one-second slots.

Thus, each epoch represents one minute of network activity. These epochs are later used as the basic temporal units for feature extraction and classification.

This representation offers two advantages. First, it provides a compact summary of user behavior over time. Second, it allows the detector to compare activity patterns between different IPs and identify signs of synchronization, which is a key indicator of coordinated attacks.

Table 1 Example mapping of packet timestamps to discrete time slots.

Packet Timestamp	Source IP	Time Slot
1313668392	147.32.84.192	0
1313668395	147.32.84.192	3
1313668400	147.32.84.192	8

After aggregation: 147.32.84.192 → {0, 3, 8}

The CTU-13 dataset is widely used for evaluating botnet detection and network intrusion analysis methods [3].

3.2 Similarity Graph Construction

In order to capture coordinated behavior among IP addresses, the activity representation is transformed into a similarity graph for each time epoch.

Each node in the graph corresponds to an active IP address within the given epoch. To define relationships between nodes, a similarity measure is computed between every pair of IPs based on their activity patterns within the epoch.

Specifically, for each IP address, we consider the set of time slots during which it was active within the epoch. The similarity between two IPs is then quantified using the Jaccard similarity coefficient, defined as:

$$J(A, B) = |A \cap B| / |A \cup B|$$

where A and B are the sets of active time slots for the two IPs. This metric captures the degree of temporal synchronization between their activities.

After computing pairwise similarities, an undirected edge is established between two nodes if their similarity exceeds a predefined threshold. This results in a similarity graph where edges represent strong temporal correlation between IPs.

The use of a threshold allows the model to focus on meaningful coordination while filtering out weak or incidental correlations. As a result, highly synchronized groups of IPs form dense subgraphs, which are indicative of coordinated behavior such as DDoS attacks.

This graph-based representation enables the detector to move beyond individual activity patterns and instead analyze collective behavior at the group level.

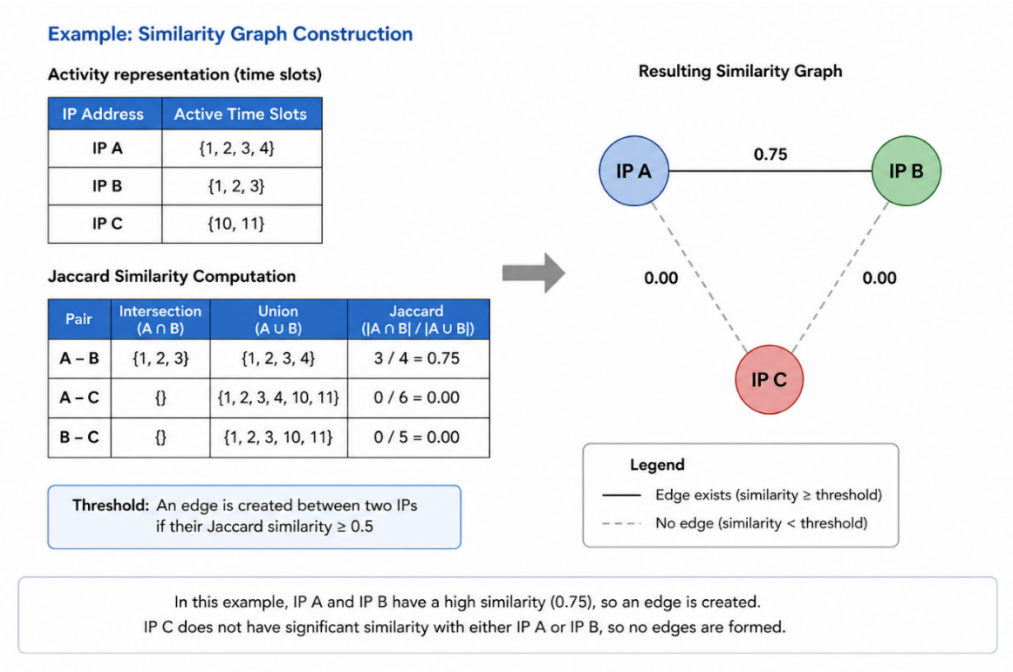

Figure 1 Example similarity graph construction using Jaccard similarity.

3.3 Graph Metrics

To characterize the structure of the similarity graph within each epoch, a set of graph-based metrics is computed. These metrics capture different aspects of coordination among IP addresses and serve as features for detection.

The first metric is the number of active IPs, which simply corresponds to the number of nodes in the graph. This provides a basic indication of the level of activity within the epoch.

The second metric is the mean similarity, defined as the average Jaccard similarity across all pairs of IPs. This reflects the overall degree of synchronization in the network.

The third metric is the edge density of the graph. Given N active IPs, the maximum possible number of edges is $N(N-1)/2$. The density is defined as the ratio of the number of actual edges to this maximum value. A high density indicates that many IPs exhibit strong similarity, suggesting coordinated behavior.

The fourth metric is the size of the largest connected component (LCC), normalized by the number of active IPs. The LCC represents the largest group of mutually connected nodes, and its relative size indicates the extent to which coordinated activity dominates the network.

These metrics provide complementary information. While mean similarity and density capture the overall level of synchronization, the LCC ratio highlights the presence of large coordinated groups. Together, they enable the detection of structured patterns that distinguish DDoS attacks from flash crowds and normal traffic.

In the implemented system, these metrics are extracted in `detector_core.py`, primarily through the `compute_window_metrics(...)` function, and are later assembled into feature vectors by `extract_window_sequence(...)`.

3.4 Baseline Detection Model

Based on the graph-derived features, a rule-based baseline detector is defined to classify each epoch into one of three states: Normal, Flash, or DDoS.

The decision is made using simple threshold-based rules that capture the expected behavior of each traffic type.

In the case of DDoS attacks, a high level of synchronization is expected. Therefore, an epoch is classified as DDoS when both the graph density and the size of the largest connected component exceed predefined thresholds. This reflects the presence of a large, highly coordinated group of IPs.

In contrast, flash crowds are characterized by a large number of active users with relatively low synchronization. For this reason, an epoch is classified as Flash when the number of active IPs exceeds a threshold, while the graph density remains low.

All remaining cases are classified as Normal traffic, corresponding to typical background activity without significant coordination or unusually high participation.

This rule-based approach provides a simple and interpretable baseline for detection. However, it operates independently on each epoch and does not

consider temporal dependencies between consecutive epochs, which can lead to inconsistent predictions in noisy conditions.

In the implementation, this logic is defined in the detector core module (`detector_core.py`), specifically in the function `predict_window_label(...)`, and applied across all epochs using `predict_baseline_sequence(...)`.

3.5 Synthetic Traffic Injection

To enable controlled evaluation of the detection system, synthetic traffic patterns are injected on top of the baseline network activity.

The baseline traffic, extracted from real PCAP data, represents normal background behavior. On top of this, synthetic traffic is introduced to simulate both flash crowds and coordinated DDoS attacks. This hybrid approach allows the creation of realistic yet fully controllable scenarios, enabling systematic experimentation under varying conditions.

To avoid potential leakage between traffic types, separate sets of synthetic IPs are used for flash crowd and DDoS scenarios. This ensures that the model does not rely on repeated IP identities across different traffic conditions, and instead focuses on structural and temporal patterns of behavior.

3.5.1 Bot Population

The synthetic traffic generation process begins with the creation of a set of artificial IP addresses, representing users or bots depending on the scenario.

These synthetic IPs are generated independently of the baseline dataset and are used to simulate additional activity in the network. The number of such IPs is controlled by the parameter "number of bots", which defines the scale of the simulated event.

A larger bot population results in more intense traffic and increases the difficulty of distinguishing between flash crowds and coordinated attacks. This parameter is varied across experiments to evaluate the robustness of the detection system under different load conditions.

3.5.2 Synchronization (Overlap)

The degree of coordination among synthetic bots is controlled through the overlap parameter, which plays a central role in the modeling of DDoS attacks.

Specifically, overlap defines the proportion of time slots that are shared among all bots within the attack interval. These shared time slots form the core synchronized activity of the attack.

High overlap values correspond to strong synchronization, where most bots are active at the same time, resulting in highly structured and easily detectable patterns. In contrast, lower overlap values produce more dispersed activity, making the attack resemble benign traffic and increasing the difficulty of detection.

By varying the overlap parameter, the experiments simulate different levels of coordination, allowing the analysis of the detector's performance under both strong and weak attack conditions.

3.5.3 Jitter

To simulate realistic deviations from perfect synchronization, additional variability is introduced through the jitter parameter.

Jitter represents the probability that a bot will exhibit extra activity outside the shared synchronized slots. This results in the addition of random time slots to each bot's activity pattern.

The presence of jitter reduces the apparent synchronization among bots, thereby making coordinated behavior less pronounced. This reflects real-world conditions where attackers may not be perfectly synchronized due to network delays or independent behavior.

As the jitter increases, the similarity between bots decreases, posing a greater challenge for graph-based detection methods.

3.5.4 Packet Drop

Another source of variability is introduced through the packet drop parameter, which simulates the loss or absence of activity within the attack pattern.

Packet drop represents the probability that a given time slot, originally part of the synchronized activity, is removed for a specific bot. This results in incomplete participation in the coordinated attack.

This mechanism reflects real-world conditions such as packet loss, unstable connections, or imperfect bot execution. As a result, the effective synchronization among bots is reduced.

Higher packet drop values lead to weaker coordination signals, making the detection of DDoS attacks more challenging.

3.5.5 Flash Crowd Modeling

In contrast to DDoS attacks, flash crowds are modeled as a large number of independent users generating activity without coordination.

Each synthetic user becomes active in each time slot with a fixed probability, resulting in a stochastic activity pattern. While this produces high overall traffic volume, the lack of synchronization leads to low similarity between users.

This distinction is critical for the detection task. Although both flash crowds and DDoS attacks involve many active IPs, only DDoS attacks exhibit strong temporal coordination. The flash crowd model is therefore essential for evaluating the detector's ability to avoid false positives.

In the implemented system, synthetic traffic generation is handled in `data_utils.py`, primarily through the functions `inject_flash_crowd(...)`, `inject_ddos(...)`, and `inject_hybrid(...)`. This framework enables reproducible experiments and systematic analysis across a wide range of traffic conditions.

3.6 Attacker Identification

Beyond detecting whether an epoch corresponds to a DDoS attack, it is equally important to identify which IP addresses are responsible for the coordinated behavior.

To address this, an attacker identification mechanism is incorporated into the system. This component operates on top of the graph-based representation and aims to detect the subset of IPs that participate in coordinated activity.

The identification process is applied only when an epoch is classified as DDoS, ensuring that the analysis focuses on relevant time periods where coordinated attacks are expected.

3.6.1 Problem Definition

The attacker identification problem can be defined as follows.

Given a time epoch and its corresponding similarity graph, the goal is to identify the subset of IP addresses that participate in a coordinated attack.

Formally, let G = (V, E) be the similarity graph for a given epoch, where V represents the set of active IPs and E the set of edges indicating high similarity. The task is to estimate a subset of nodes $A \subseteq V$ that correspond to the true attackers.

In the context of the synthetic scenarios, the ground truth attackers are known and correspond to the set of injected bot IPs during DDoS intervals. This allows for quantitative evaluation of the identification performance.

This problem extends the detection task by moving from a binary decision (attack or not) to a finer-grained analysis of which entities are responsible for the observed behavior.

3.6.2 Graph-based Identification (LCC)

The proposed approach for attacker identification is based on the structure of the similarity graph.

Specifically, when an epoch is classified as DDoS, the largest connected component (LCC) of the graph is extracted and considered as the set of detected attackers.

The rationale behind this approach is that coordinated attackers tend to form a densely connected group due to their high temporal synchronization. As a result, they are likely to appear within the same connected component of the graph, and in most cases, within the largest one.

This method provides a simple yet effective way to isolate coordinated groups without requiring additional clustering algorithms. It leverages the same graph

structure used for detection, ensuring consistency between detection and identification.

In the implementation, the LCC is computed using graph traversal techniques, and the corresponding nodes are returned as the detected attackers.

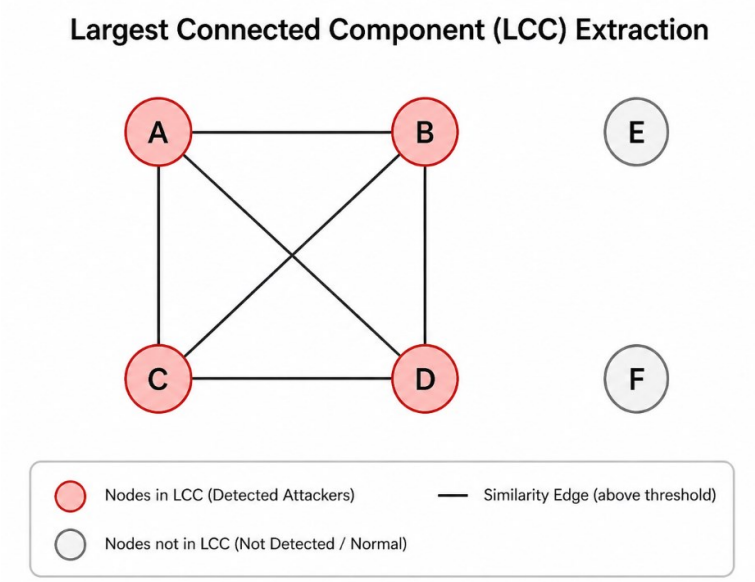

Figure 2 Example of Largest Connected Component (LCC) extraction used for coordinated attacker identification.

graph-based analysis [7]

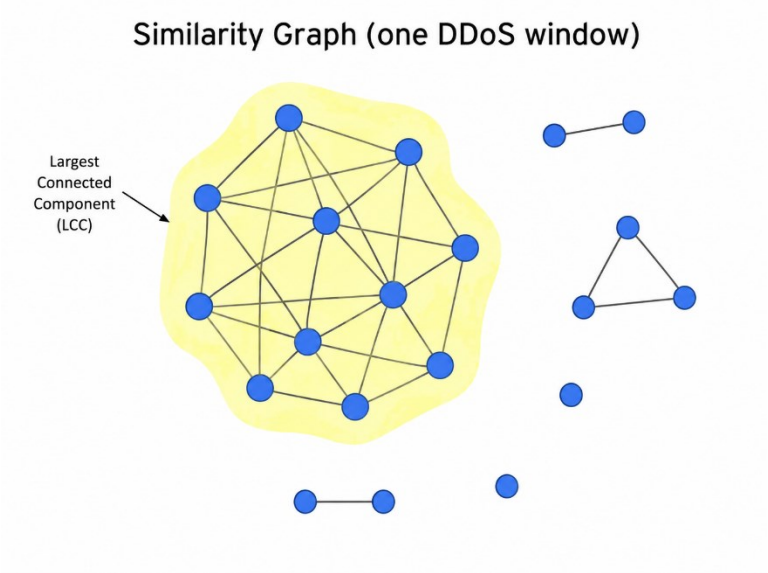

Figure 3 Example 2 of Largest Connected Component (LCC) extraction used for coordinated attacker identification.

3.6.3 Evaluation Metrics (Precision, Recall, F1)

To evaluate the performance of attacker identification, standard classification metrics are used, namely precision, recall, and F1-score.

Let T be the set of true attackers and D the set of detected attackers for a given epoch. The following quantities are defined:

- True Positives (TP): IPs correctly identified as attackers ($D \cap T$)

- False Positives (FP): IPs incorrectly identified as attackers ($D \setminus T$)

- False Negatives (FN): attackers that were not detected ($T \setminus D$)

Using these definitions, the evaluation metrics are computed as follows:

$$\text{Precision} = TP / (TP + FP)$$

$$\text{Recall} = TP / (TP + FN)$$

$$\text{F1-score} = 2 \times (\text{Precision} \times \text{Recall}) / (\text{Precision} + \text{Recall})$$

Precision measures the correctness of the detected attackers, while recall measures the completeness of the detection. The F1-score provides a balanced measure that combines both aspects.

In addition to these metrics, a coverage metric is also considered, representing the proportion of DDoS epochs in which at least one attacker is successfully identified.

These metrics are computed only for epochs that are relevant to attacker identification, namely those that correspond to true or predicted DDoS activity.

4 Temporal Modeling

While the baseline detector operates on each epoch independently, real-world network behavior evolves over time and exhibits temporal dependencies.

In particular, attack patterns such as DDoS do not appear randomly in isolated epochs, but rather persist over consecutive time intervals. This temporal continuity can be exploited to improve detection performance.

To capture such dependencies, a probabilistic temporal model is introduced, based on Hidden Markov Models (HMMs). This approach enables the system to model the evolution of network states over time and produce more consistent and robust predictions.

4.1 Limitations of Static Detection

The baseline detection approach treats each epoch independently, making decisions based solely on the features extracted within that epoch.

While this approach is simple and interpretable, it has several limitations. In particular, it is sensitive to noise and fluctuations in the data, which may lead to inconsistent predictions across consecutive epochs.

For example, a short deviation in the graph metrics may cause a temporary misclassification, even if the overall behavior remains unchanged. This is especially problematic in borderline cases, where the distinction between flash crowds and DDoS attacks is less clear.

Furthermore, static detection does not exploit the fact that network states typically evolve gradually over time. Once a DDoS attack begins, it is likely to persist for multiple consecutive epochs, and similarly, normal traffic rarely transitions abruptly to highly coordinated behavior.

These limitations motivate the use of temporal models that can incorporate dependencies between consecutive epochs and produce smoother and more reliable predictions.

4.2 Hidden Markov Model Formulation

To address the limitations of static detection, a Hidden Markov Model (HMM) is employed to model the temporal evolution of network behavior.

An HMM is defined by a set of hidden states, a transition probability matrix, and an observation model. In this context, the hidden states correspond to the underlying traffic conditions, while the observations correspond to the extracted graph-based features.

The model assumes that the system evolves over time according to a Markov process, where the current state depends only on the previous state. At each time step, an observation is generated based on the current state.

By modeling both the temporal transitions between states and the relationship between states and observations, the HMM provides a principled framework for sequence-based detection.

4.2.1 States

The hidden states of the model correspond to the three traffic conditions considered in this work:

- Normal
- Flash
- DDoS

These states represent the underlying behavior of the network at each epoch. While the true state is not directly observable, it is inferred through the observed graph-based features.

The transitions between states capture how network behavior evolves over time. For example, the model can learn that DDoS states tend to persist across consecutive epochs, while transitions from Normal to DDoS are less frequent.

4.2.2 Observations

The observations of the HMM correspond to the feature vectors extracted from each epoch.

Specifically, each epoch is represented by a vector of graph-based metrics, including the number of active IPs, graph density, mean similarity, and the ratio of the largest connected component.

These features are modeled using Gaussian distributions, assuming independence between feature dimensions. For each state, the model learns the mean and variance of each feature, forming a state-dependent emission model.

This allows the HMM to evaluate the likelihood of observing a given feature vector under each possible state.

4.3 Training Procedure

The HMM is trained in a supervised manner using labeled sequences of epochs derived from the synthetic traffic generation process. Instead of employing a traditional random split of repeated runs, the dataset is partitioned based on parameter configurations. Specifically, the training set consists of sequences generated from a subset of parameter values (such as bot population size, synchronization level, jitter, and packet drop), while the test set includes configurations that are not observed during training.

This design allows the model to learn general temporal and structural patterns of network behavior, rather than memorizing specific scenarios. As a result, the evaluation better reflects the model's ability to generalize to previously unseen conditions, which is more representative of real-world deployment settings.

During training, the model estimates the initial state distribution, the transition probabilities between hidden states, and the parameters of the Gaussian emission model for each state. The training data consists of sequences of feature vectors paired with their corresponding ground truth labels, enabling the model to associate observed graph-based features with the underlying traffic states.

This supervised training approach allows the HMM to capture both the temporal dynamics of state transitions and the statistical properties of the observations

associated with each state, while maintaining robustness under varying and previously unseen traffic conditions.

4.4 Viterbi Decoding

Once the HMM is trained, inference is performed using the Viterbi algorithm.

Given a sequence of observed feature vectors, the goal is to determine the most likely sequence of hidden states that could have generated these observations.

The Viterbi algorithm uses dynamic programming to efficiently compute this optimal state sequence, taking into account both the emission probabilities and the transition probabilities between states.

Unlike the baseline detector, which classifies each epoch independently, the HMM produces a globally optimal sequence of predictions. This results in smoother and more consistent classifications, as the model favors state sequences that are both likely in terms of observations and plausible in terms of temporal transitions.

This temporal smoothing effect significantly improves robustness to noise and reduces isolated misclassifications.

Graph-based community analysis techniques are commonly used for identifying coordinated structures and behavioral clusters within complex networks [4].

The graph representation used in this work follows principles commonly applied in network science and graph analysis [6].

5 Experimental Setup

This section describes the experimental setup used to evaluate the proposed detection and attacker identification framework.

The evaluation is based on a combination of real network traffic and synthetic injection, allowing the creation of controlled scenarios with known ground truth. A systematic parameter sweep is performed to assess the robustness of the system under varying conditions.

5.1 Dataset Description (CAIDA + Synthetic Injection)

The experiments are based on real network traffic obtained from a publicly available dataset (CAIDA), which provides packet-level traces of Internet traffic.

From this dataset, source IP addresses and timestamps are extracted and transformed into an activity representation using 1-second time slots. Only relevant IP traffic is retained, and the resulting dataset captures the natural background behavior of the network.

On top of this baseline traffic, synthetic events are injected to simulate both flash crowds and DDoS attacks. This hybrid approach combines the realism of actual traffic with the controllability of synthetic scenarios.

The synthetic injection process ensures that the ground truth is known for each epoch, enabling precise evaluation of both detection and attacker identification performance.

5.2 Parameter Sweep Design

To systematically evaluate the performance of the detection system, a parameter sweep is performed over multiple dimensions.

The experiments vary the following parameters:

- Number of bots (e.g., 10, 20, 30, 50)
- DDoS overlap (level of synchronization)
- Jitter (random activity)
- Packet drop (activity loss)
- Flash crowd probability

Each combination of parameters defines a distinct scenario. Multiple runs are executed for each configuration using different random seeds to ensure statistical robustness.

This exhaustive exploration of the parameter space allows the analysis of system performance under both ideal and challenging conditions.

5.3 Evaluation Metrics

The performance of the system is evaluated using a combination of detection and identification metrics.

For detection, the following metrics are used:

- Accuracy
- Confusion matrix (Normal, Flash, DDoS)
- DDoS False Negative Rate (FNR)
- Flash Crowd False Negative Rate (FNR)
- Misclassification rates (Flash → DDoS, DDoS → Flash)
- Detection delay

Detection delay measures how quickly the system identifies the onset of a DDoS attack after it begins.

For attacker identification, the following metrics are used:

- Precision
- Recall
- F1-score
- Detection coverage

These metrics are computed at the epoch level and averaged across runs. Importantly, attacker identification metrics are evaluated only on epochs where DDoS activity is present or predicted, ensuring meaningful evaluation.

Hidden Markov Models (HMMs) are widely used for probabilistic temporal sequence modeling and hidden state estimation [7].

5.4 Train/Test Split Strategy (Unseen Configurations)

To evaluate the generalization ability of the HMM-based detector, a configuration-based train/test split strategy is employed. Instead of partitioning the data using repeated runs of the same parameter settings, the dataset is divided based on distinct parameter configurations.

Specifically, the training set consists of sequences generated from a subset of parameter values, including selected combinations of bot population size, synchronization level (overlap), jitter, and packet drop. The test set, in contrast, includes configurations that are not observed during training, ensuring that the model is evaluated on previously unseen conditions.

This approach provides a more realistic and challenging evaluation setup, as it prevents the model from relying on familiarity with specific scenarios and instead requires it to generalize learned temporal and structural patterns to new situations. Such a setup better reflects real-world deployment, where attack characteristics may vary and differ from those encountered during training.

The training phase is used to estimate the HMM parameters, including the initial state distribution, transition probabilities, and emission distributions. The testing phase evaluates the model's detection accuracy, robustness, and attacker identification performance under unseen traffic conditions.

5.5 Synthetic Scenario Design

The synthetic scenarios are designed to simulate a wide range of network conditions, from normal traffic to highly coordinated attacks.

DDoS scenarios are controlled through the overlap parameter, which determines the level of synchronization among bots. Lower overlap values produce weakly coordinated activity, while higher values result in strongly synchronized attacks.

Additional variability is introduced through jitter and packet drop, which simulate noise and imperfections in real-world traffic.

Flash crowd scenarios are generated independently, with users exhibiting random activity patterns without coordination. This allows the system to distinguish between high-volume benign traffic and malicious coordinated behavior.

By combining these factors, the experimental setup captures both realistic and adversarial conditions, enabling comprehensive evaluation of the proposed approach.

6 Results and Analysis

6.1 Baseline Detector Performance

The baseline detector demonstrates moderate performance across the evaluated scenarios, achieving a mean accuracy of approximately 0.686. While it successfully detects a large proportion of DDoS events (detection rate ≈ 0.93), its overall effectiveness is limited by its inability to distinguish between flash crowds and coordinated attacks.

In particular, the model exhibits a very high Flash Crowd False Negative Rate (FNR) (≈ 0.62), indicating that the majority of flash crowd events are misclassified as either normal traffic or DDoS. This behavior highlights a key limitation of threshold-based detection, which relies solely on instantaneous graph features and does not fully capture the structural differences between benign and malicious high-volume traffic.

As shown in Figure 3, the baseline detector improves gradually as the overlap increases, indicating that it depends on stronger synchronization patterns to achieve better performance. However, even at higher overlap values, its accuracy remains significantly lower than that of the HMM-based approach.

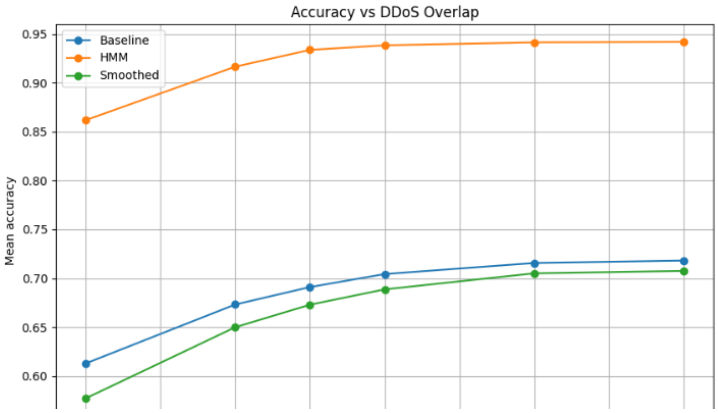

Figure 4 Accuracy vs DDoS overlap.

6.2 Effect of Temporal Smoothing

The introduction of temporal majority smoothing does not lead to a meaningful improvement in detection performance. In fact, the smoothed detector exhibits slightly lower overall accuracy compared to the baseline model, indicating that simple temporal aggregation does not enhance classification quality.

As shown in Figure 4, the Flash Crowd False Negative Rate (FNR) remains consistently high for both the baseline and smoothed models (≈ 0.62–0.63) across all overlap values. This indicates that smoothing does not improve the model's ability to correctly identify flash crowd events.

This behavior suggests that the core limitation of the baseline approach is not due to temporal noise, but rather due to the lack of a principled temporal modeling mechanism. While smoothing enforces short-term consistency between neighboring epochs, it does not capture the underlying state transitions or temporal dependencies of coordinated attacks.

As a result, smoothing fails to improve the separation between flash crowds and DDoS activity, reinforcing the need for a more advanced temporal model such as the Hidden Markov Model.

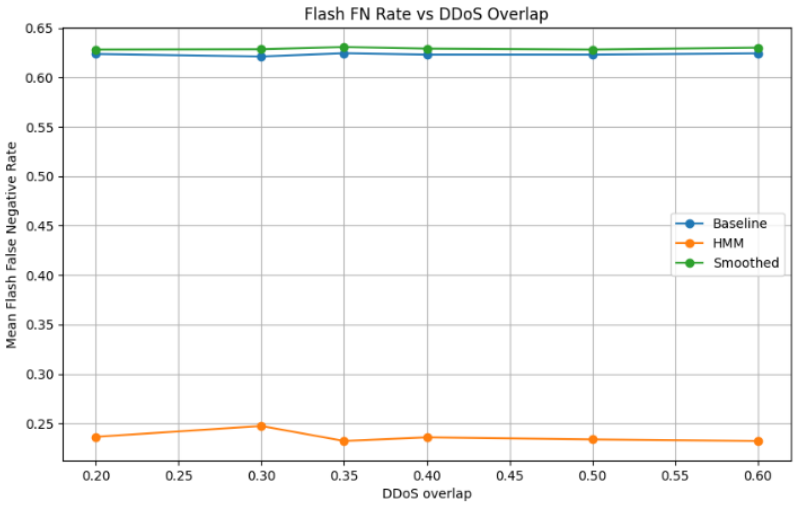

Figure 5 Flash Crowd FNR vs overlap.

6.3 HMM-based Detection Results

6.3.1 Accuracy and Error Analysis

The HMM-based detector significantly outperforms both the baseline and smoothed models across all overlap values. As shown in Figure 5, the HMM consistently achieves higher accuracy, maintaining high performance even under low synchronization conditions (e.g., overlap = 0.2).

A key improvement is observed in the reduction of DDoS false negative rates. As illustrated in Figure 6, the HMM achieves substantially lower error rates compared to both baseline and smoothed approaches. At low overlap values, the HMM

reduces the false negative rate from approximately 0.42 (baseline) and 0.51 (smoothed) to around 0.31. This improvement becomes more pronounced as overlap increases, with the HMM effectively eliminating false negatives at moderate to high overlap values.

This behavior indicates that the HMM is able to detect coordinated attacks even when synchronization is weak, while baseline approaches rely more heavily on strong overlap patterns. The probabilistic temporal modeling of the HMM allows it to capture underlying state transitions, leading to more robust and accurate classification.

Compared to the baseline approach, the HMM improves accuracy by more than 20 percentage points while significantly reducing error rates. This improvement stems from the ability of the HMM to model temporal dependencies between consecutive observations, allowing it to infer the underlying attack state even when instantaneous evidence is weak.

Table 2 summarizes the overall performance comparison between the evaluated models, clearly highlighting the superiority of the HMM-based approach across all metrics.

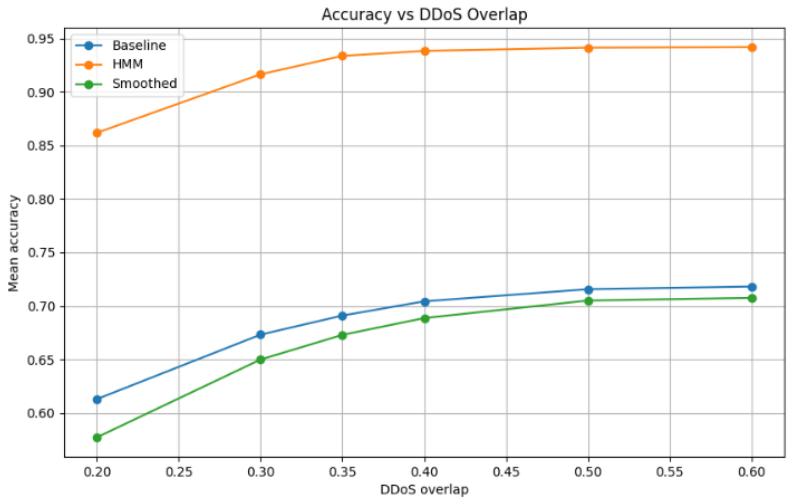

Figure 6 HMM detector accuracy vs DDoS overlap.

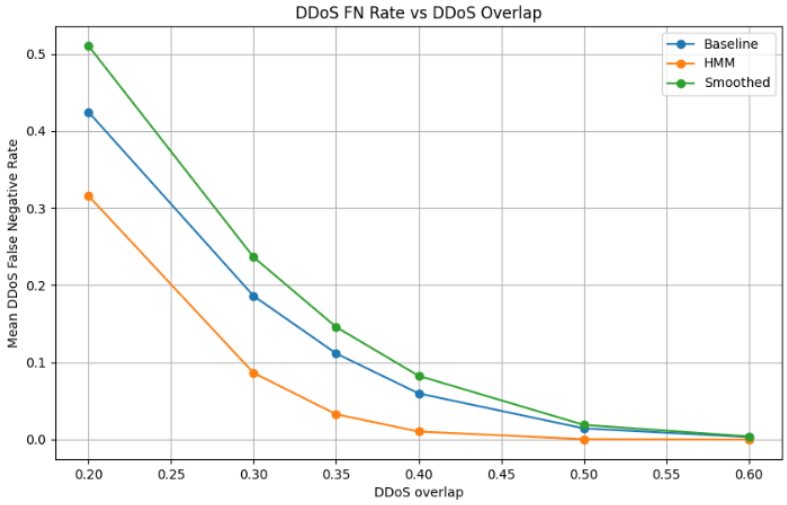

Figure 7 DDoS FNR vs overlap.

Table 2 Summary of Overall Performance

Model	Accuracy	DDoS FNR	Detection Delay
Baseline	~0.71	~0.42	~0.18
Smoothed	~0.70	~0.51	~0.24
HMM	~0.94	~0.00-0.31	~0.04

6.3.2 Detection Delay Analysis

The HMM-based detector also demonstrates superior performance in terms of detection delay. As shown in Figure 7, the HMM achieves consistently lower detection delay across all overlap values, significantly outperforming both baseline and smoothed models.

At low overlap levels, the baseline detector exhibits delays of approximately 0.18 windows, while the smoothed version performs even worse (≈0.24). In contrast, the HMM maintains very low delay values (≈0.04 at overlap = 0.2), which quickly approach zero as overlap increases.

This result highlights the ability of the HMM to rapidly detect transitions into the DDoS state. By modeling temporal dependencies between consecutive epochs, the HMM can identify coordinated behavior earlier than static methods, without requiring strong evidence from a single observation window.

This demonstrates that the HMM not only improves detection accuracy, but also enables faster response to emerging attacks. Such behavior is particularly important for real-time systems, where early detection is critical for effective mitigation.

Overall, the HMM provides both high accuracy and fast detection, making it well-suited for real-time deployment scenarios.

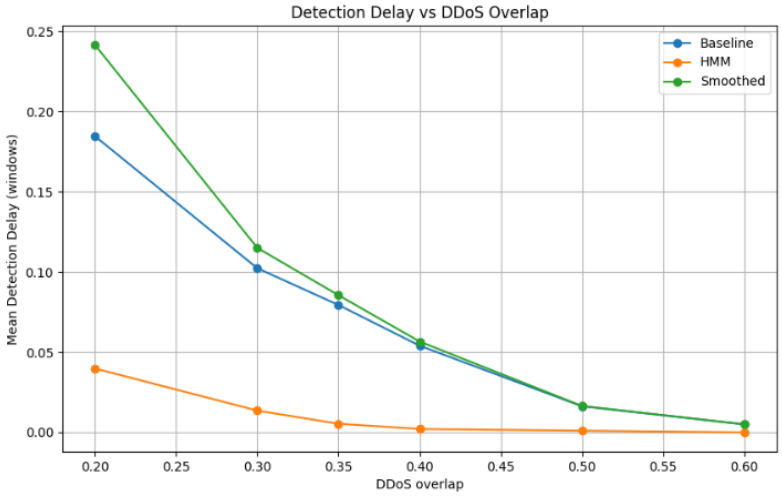

Figure 8 Average Detection delay vs overlap.

6.4 Critical Synchronization Threshold Analysis

The critical synchronization threshold represents the minimum level of overlap required for reliable detection of DDoS attacks.

As shown in Table 3, the baseline and smoothed models require relatively high overlap values (typically around 0.5) in order to achieve stable detection performance. This indicates that these models rely heavily on strong synchronization signals and are unable to detect loosely coordinated attacks.

In contrast, the HMM-based detector significantly lowers the required synchronization threshold. The critical overlap for the HMM ranges from approximately 0.30 to 0.40, depending on the number of bots. This demonstrates that the HMM can reliably detect coordinated behavior even under weaker synchronization conditions.

Furthermore, the improvement becomes more pronounced as the number of bots increases. For example, when 50 bots are present, the HMM achieves reliable detection at an overlap of approximately 0.30, whereas the baseline models still require overlap values close to 0.5.

These results confirm that temporal modeling enables the detection of more subtle coordination patterns, making the HMM-based approach significantly more robust in realistic attack scenarios.

Table 3 Critical synchronization threshold for each model and bot configuration. The HMM achieves reliable detection at significantly lower overlap values compared to baseline approaches.

Model	Bots	Critical Overlap
Baseline	10	0.50
Baseline	20	0.50
Baseline	30	0.40
Baseline	50	0.50
HMM	10	0.40
HMM	20	0.35
HMM	30	0.35
HMM	50	0.30
Smoothed	10	0.50
Smoothed	20	0.50
Smoothed	30	0.50

Smoothed	50	0.50

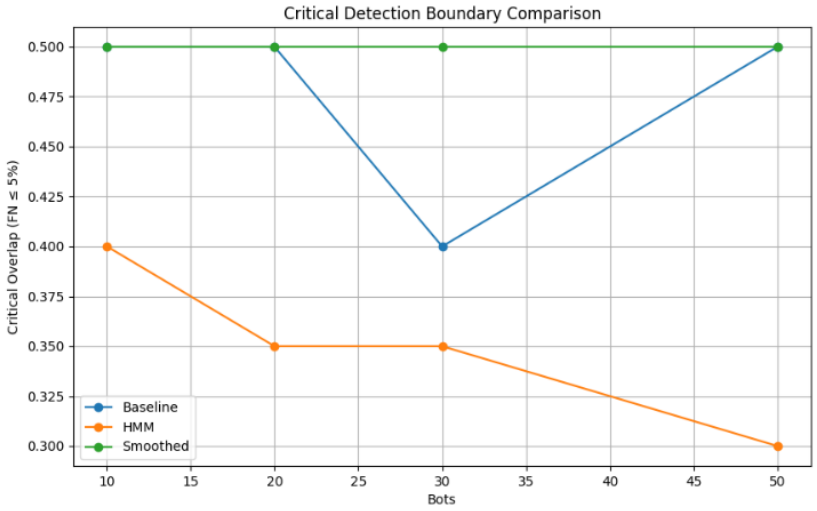

Figure 9 Critical overlap vs number of bots.

As shown in Figure 8, the HMM consistently operates at lower critical overlap values across all bot configurations, while baseline and smoothed models remain nearly constant.

6.5 Attacker Identification Results

6.5.1 Precision / Recall / F1 vs Overlap

The performance of the attacker identification module is strongly influenced by the level of synchronization among attacking nodes, as controlled by the overlap parameter.

At low overlap values (e.g., overlap = 0.2), the identification task is more challenging. In this case, the system achieves moderate performance, with precision and recall values around 0.6–0.7 and an F1-score close to 0.67. This behavior indicates that weakly coordinated attacks produce less distinguishable patterns in the similarity graph, making it harder to accurately isolate attackers.

However, performance improves significantly as overlap increases. At overlap = 0.3, both precision and recall rise above 0.85, with the F1-score exceeding 0.90. This demonstrates that even a moderate increase in synchronization leads to much clearer structural patterns in the graph.

At overlap = 0.35, the system achieves very high performance, with precision and recall both above 0.95 and F1-scores approaching 0.97. This indicates that the majority of attackers are correctly identified with very few false positives.

For overlap values of 0.4 and above, attacker identification becomes nearly perfect. Precision approaches 1.0, recall approaches 1.0, and the F1-score is effectively saturated across all configurations. This demonstrates that the graph-based identification method is highly effective when moderate to strong coordination is present.

Overall, these results highlight that attacker identification is highly sensitive to synchronization, and becomes reliable once a sufficient level of coordinated behavior emerges among attacking nodes.

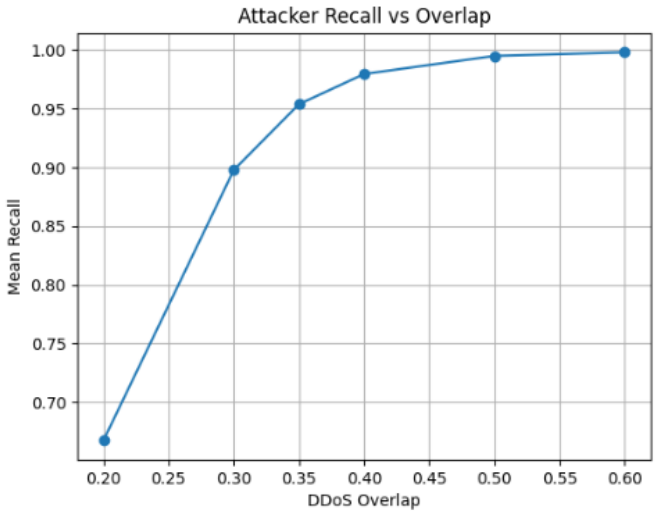

Figure 10 Attacker recall vs overlap.

Table 4 Attacker identification performance metrics across different overlap levels.

Overlap	Precision	Recall	F1-score
0.2	~0.65	~0.67	~0.67
0.3	~0.85	~0.85	~0.90
0.35	~0.95	~0.95	~0.97
>=0.4	~1.00	~1.00	~1.00

6.5.2 Effect of Number of Bots

The number of participating bots has a noticeable impact on attacker identification performance, particularly at low levels of synchronization.

At low overlap values (e.g., overlap = 0.2), increasing the number of bots significantly improves detection performance. For instance, when 10 bots are present, the system achieves a recall of approximately 0.56 and an F1-score of approximately 0.57. In contrast, when the number of bots increases to 50, recall rises to approximately 0.75 and the F1-score to approximately 0.75. This improvement occurs because a larger number of attackers creates stronger structural patterns in the similarity graph, making coordinated behavior easier to detect.

As the overlap increases, the influence of the number of bots gradually diminishes. At moderate overlap values (e.g., overlap = 0.3–0.35), all configurations achieve high performance, with recall and F1-scores exceeding 0.90, regardless of the number of bots.

At high overlap values (overlap ≥ 0.4), the system achieves near-perfect identification across all bot configurations. Precision, recall, and F1-score

approach 1.0, and the difference between scenarios with 10 bots and 50 bots becomes negligible.

Overall, these results indicate that while a larger number of bots improves detection under weak synchronization, the level of coordination remains the dominant factor in attacker identification performance.

6.5.3 Coverage Analysis

In addition to precision and recall, we evaluate attacker identification performance using the coverage metric, defined as the fraction of true DDoS windows in which at least one attacker is successfully detected.

As shown in Figure 10, coverage increases rapidly with the level of synchronization. At low overlap values (overlap = 0.2), coverage is approximately 0.68, indicating that in a significant fraction of attack windows the system fails to identify any attackers. This reflects the difficulty of detecting weakly coordinated behavior.

However, coverage improves sharply as overlap increases. At overlap = 0.3, coverage exceeds 0.9, meaning that the system is able to identify at least one attacker in the vast majority of DDoS windows.

For overlap values of 0.35 and above, coverage approaches 1.0, indicating that attacker identification becomes highly reliable. In these cases, the system consistently detects attackers in nearly all attack windows.

Overall, the coverage analysis confirms that synchronization not only improves the accuracy of identifying attackers, but also ensures that attack presence is

consistently captured across time, making the system robust in practical scenarios.

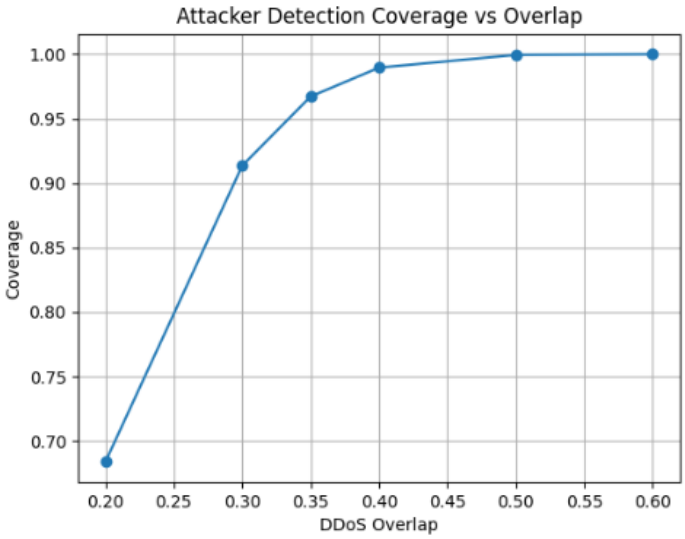

Figure 11 Attacker coverage vs overlap.

For example, at overlap = 0.2, a coverage of 0.68 indicates that in approximately 32% of attack windows, no attacker is detected.

6.5.4 Confusion Matrix

The matrix below shows that Normal traffic is classified perfectly, with all instances correctly identified as Normal. This indicates that the model is highly effective at distinguishing benign background traffic from anomalous behavior.

For Flash traffic, the model achieves a correct classification rate of approximately 74%. The majority of errors correspond to misclassification as Normal traffic (25%), while only a very small fraction (1%) is incorrectly classified as DDoS. This suggests that the detector is conservative in labelling traffic as malicious, preferring to avoid false alarms even at the cost of occasionally missing Flash events.

In the case of DDoS traffic, the model demonstrates strong performance, correctly identifying approximately 94% of the attack windows. A small portion (5%) is misclassified as Flash, while misclassification as Normal is negligible. This

behavior is consistent with previous results, where confusion between Flash and DDoS traffic is expected due to their similar high-volume characteristics.

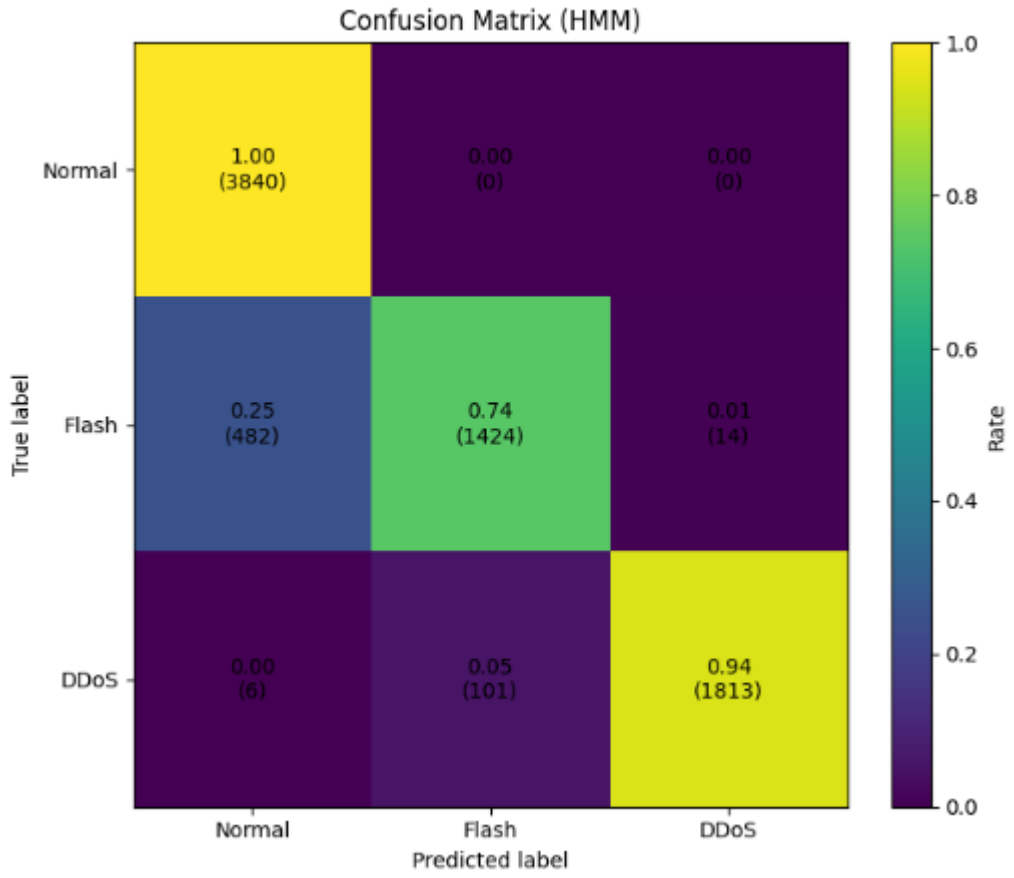

Figure 12 Confusion Matrix

Overall, the confusion matrix (Figure 11) confirms that the primary source of classification error lies in the distinction between Flash Crowds and DDoS attacks, rather than between Normal and anomalous traffic. This finding aligns with the overlap-based analysis presented earlier, confirming that the main classification challenge lies in separating legitimate high-volume flash crowd activity from coordinated DDoS behavior.

6.6 Validation on CTU10 DDoS Traffic

To further evaluate the attacker identification mechanism, an additional validation experiment was conducted using a CTU10 DDoS traffic sample. Unlike the controlled synthetic scenarios used in the main evaluation, this dataset contains

real DDoS communication records with timestamps, source IP addresses, and a common destination IP representing the victim.

The CSV traffic records are transformed into the same activity representation used by the proposed framework, where each source IP is mapped to the set of time slots in which it is active. The timestamp of the first packet is used as a reference, and subsequent packet timestamps are converted into discrete one-second slots. This enables the direct application of the graph-based detection and identification mechanism on real traffic data.

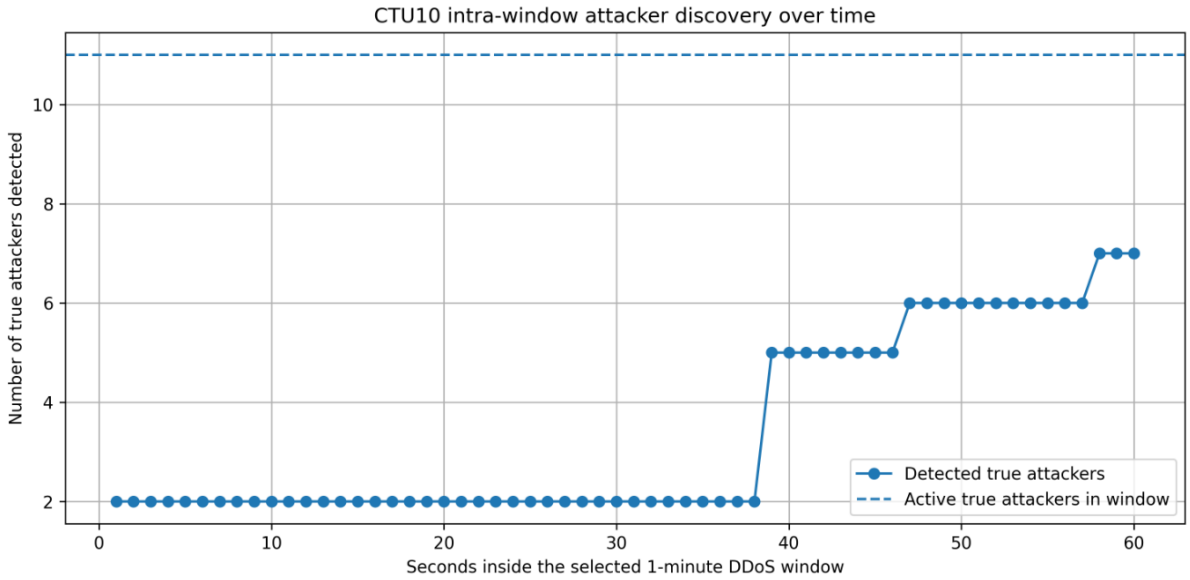

Figure 13 CTU10 intra-window attacker discovery over time.

Figure 12 illustrates the attacker discovery behavior within a selected one-minute DDoS window. The x-axis represents the elapsed time inside the window, while the y-axis shows the cumulative number of correctly identified attackers. The dashed horizontal line indicates the total number of active attackers within that specific window.

The results show that attacker identification progresses gradually as more temporal observations become available. At the beginning of the window, only a

subset of attackers is detected, while additional attackers are progressively identified later in the interval. This behavior reflects the fact that real DDoS traffic does not exhibit perfect synchronization across all attacking nodes.

Importantly, not all attackers are always detected within a single isolated window. Some attackers may generate sparse or weakly correlated activity during a specific time interval, preventing them from forming sufficiently strong similarity links with the main coordinated group.

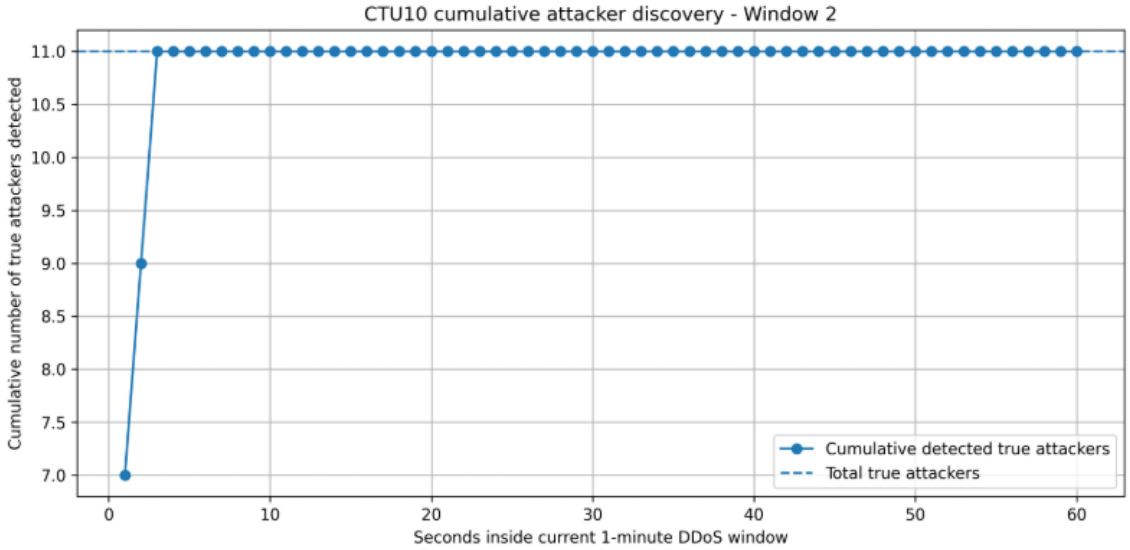

Figure 14 Cumulative attacker recovery across consecutive DDoS windows.

To further analyze this behavior, a consecutive DDoS window was also examined. Figure 13 shows that attackers not detected during the initial window are progressively recovered in subsequent windows as additional temporal evidence becomes available.

In the presented example, the detector initially identifies only part of the coordinated attack group, but achieves complete attacker recovery after aggregating information across neighboring windows. This demonstrates the advantage of temporal modeling and highlights the ability of the proposed

framework to refine attacker identification over time instead of relying exclusively on isolated observations.

The detected attacker set for the analyzed windows is presented below. The detector successfully identifies the majority of participating botnet nodes, although a small number of false positives may also appear. This behavior is expected in graph-based identification approaches, where nodes exhibiting highly similar temporal activity patterns may occasionally become connected to the coordinated component despite not belonging to the true attacker set.

Example of detected attackers

147.32.84.164

147.32.84.165

147.32.84.191

147.32.84.192

147.32.84.193

147.32.84.204

147.32.84.205

147.32.84.206

147.32.84.207

147.32.84.208

147.32.84.209

CTU dataset [3]

Overall, the CTU10 validation experiments demonstrate that the proposed graph-based framework is capable of identifying coordinated attackers in real-world DDoS traffic. Although complete attacker reconstruction may require aggregating information across multiple temporal windows, the method consistently captures the core coordinated malicious group with high temporal stability.

7 Discussion

7.1 Limitations

This chapter discusses the key findings of the proposed approach, analyses its limitations, and evaluates its applicability to real-world scenarios. Despite the strong performance demonstrated by the proposed approach, several limitations should be acknowledged.

First, the evaluation relies heavily on synthetic traffic injection. Although the synthetic scenarios are designed to realistically model DDoS and flash crowd behavior, they may not fully capture the complexity and variability of real-world network traffic.

Second, the detection performance depends on the level of synchronization among attackers. As shown in the experimental results, low-overlap attacks are significantly harder to detect, both at the detection and attacker identification levels. This indicates that the method is less effective against loosely coordinated or stealthy attacks.

Third, the current approach assumes that internal IPs can be reliably identified and isolated from the baseline dataset. In real-world deployments, such assumptions may not always hold, especially in environments with dynamic addressing or NAT.

Finally, the graph-based attacker identification method relies on the existence of strong connectivity patterns. In scenarios where attackers deliberately avoid synchronization or mimic legitimate traffic behavior, the structure of the similarity graph may become less informative, reducing identification accuracy.

These limitations highlight important directions for improving the robustness and applicability of the proposed system.

7.2 Impact of Synthetic Data

The experimental evaluation of the proposed system relies on synthetic traffic injection applied on top of a real baseline dataset. This design choice provides full control over the characteristics of the generated scenarios, including the number of attackers, synchronization level, and noise parameters.

The use of synthetic data enables systematic exploration of the parameter space, allowing the evaluation of the detection system under a wide range of controlled conditions. In particular, it allows for precise analysis of the effect of synchronization (overlap), jitter, and packet loss on both detection and attacker identification performance.

However, the reliance on synthetic data introduces certain limitations. Although the injected patterns are designed to resemble realistic DDoS and flash crowd behaviors, they may not fully capture the complexity of real-world attacks. In practice, attackers may exhibit more heterogeneous behavior, adaptive strategies, or attempt to evade detection by mimicking legitimate traffic patterns.

Despite these limitations, the combination of real baseline traffic and controlled synthetic injection provides a strong experimental framework. It ensures that the background traffic remains realistic while enabling reproducible and interpretable evaluation of the detection methods.

Overall, the results obtained using this hybrid approach provide valuable insights into the behavior of synchronization-based detection techniques, while highlighting the need for future validation on fully real-world datasets.

7.3 Assumptions and Practical Considerations

The proposed system is based on several assumptions that simplify the modeling and evaluation process, but may affect its applicability in real-world environments.

First, the method assumes that traffic can be partitioned into fixed-size time windows (epochs), within which activity patterns are extracted. While this simplifies temporal analysis, real network traffic may exhibit more irregular behavior that does not align perfectly with fixed window boundaries.

Second, the approach assumes that internal source IPs can be identified and isolated from the baseline dataset. In practice, this may not always be straightforward, particularly in networks with dynamic addressing, NAT, or encrypted traffic, where attribution of source activity becomes more challenging.

Third, the synthetic attacker model assumes a homogeneous population of bots with similar behavior patterns, controlled by parameters such as overlap, jitter, and packet drop. In real-world scenarios, attackers may exhibit more diverse and adaptive strategies, reducing the uniformity of the observed patterns.

Additionally, the graph-based detection mechanism relies on the assumption that coordinated behavior results in strong structural connectivity. While this holds for synchronized attacks, adversaries may attempt to evade detection by deliberately reducing synchronization or injecting noise into their activity patterns.

From a practical perspective, the computational cost of constructing similarity graphs and extracting graph-based metrics should also be considered, especially in large-scale or real-time deployment scenarios.

Overall, while the proposed assumptions enable a clear and controlled experimental framework, they highlight important considerations that must be addressed when transitioning the system to real-world applications.

7.4 Attacker Identification Reliability

The experimental results demonstrate that the proposed system achieves highly reliable attacker identification, particularly under moderate to high levels of synchronization.

As shown in the evaluation, precision, recall, and F1-score all improve significantly as the overlap parameter increases. For low synchronization levels (overlap = 0.2), attacker identification is more challenging, with moderate recall and F1-score values. This reflects the difficulty of detecting loosely coordinated attackers whose activity patterns are less distinguishable from normal traffic.

However, even a moderate increase in synchronization leads to substantial improvements. For overlap values around 0.3–0.35, the system achieves high precision and recall, indicating that most attackers are correctly identified with minimal false positives.

At higher synchronization levels (overlap ≥ 0.4), attacker identification becomes nearly perfect across all experimental configurations. Precision approaches 1.0, recall approaches 1.0, and F1-score is effectively saturated. This demonstrates that the graph-based identification method is highly effective in detecting coordinated behavior.

In addition to accuracy-based metrics, the coverage analysis shows that the system consistently detects attackers across nearly all DDoS windows when

sufficient synchronization is present. This indicates that the method is not only accurate but also robust in identifying attack presence over time.

Overall, these results confirm that attacker identification reliability is strongly dependent on coordination, and that the proposed approach is particularly well-suited for detecting synchronized DDoS attacks.

7.5 Generalization to Real Datasets

While the proposed system demonstrates strong performance under controlled experimental conditions, an important question concerns its ability to generalize to real-world datasets.

The current evaluation is based on a hybrid approach, combining real baseline traffic with synthetic attack injection. This setup ensures realistic background traffic while enabling controlled experimentation. However, real-world attack scenarios may exhibit additional complexities that are not fully captured by the synthetic model.

In practice, attackers may employ more sophisticated strategies, including varying levels of synchronization, adaptive behavior, and attempts to mimic legitimate traffic patterns. These factors could reduce the clarity of the structural patterns exploited by the graph-based detection method.

Despite these challenges, the results suggest that the core principle of synchronization-based detection is robust. The strong performance observed across a wide range of parameters indicates that the method captures fundamental characteristics of coordinated attacks, which are likely to persist in real-world scenarios.

Nevertheless, further validation on publicly available network traffic datasets or real operational environments is necessary to fully assess the generalization capabilities of the system. Such evaluation would provide stronger evidence of its practical applicability and robustness under realistic conditions.

Overall, the proposed approach shows promising potential for real-world deployment, while highlighting the importance of future work focused on external validation and adaptation to more complex traffic environments.

8 Conclusion

8.1 Summary of Findings

This thesis addresses the challenging problem of distinguishing between distributed denial-of-service (DDoS) attacks and flash crowd events, which often exhibit similar traffic characteristics but differ fundamentally in coordination behavior. The proposed system combines graph-based modeling of network activity with temporal analysis through a Hidden Markov Model (HMM), enabling both accurate detection and attacker identification.

The experimental results demonstrate that the baseline detector is able to identify highly synchronized attacks but struggles in scenarios with lower levels of coordination and exhibits high false negative rates for flash crowd events. Temporal smoothing provides only limited improvement and does not significantly enhance overall detection performance.

In contrast, the HMM-based approach achieves substantially higher accuracy across all evaluated scenarios. It significantly reduces both DDoS and Flash Crowd False Negative Rate (FNR), while maintaining high detection reliability and very low detection delay. The incorporation of temporal modeling allows the system to capture sequential patterns that are not visible to static methods, enabling robust detection even under weak synchronization conditions.

Furthermore, the attacker identification module demonstrates strong performance, particularly under moderate to high synchronization levels. Precision, recall, and F1-score approach near-perfect values, while the coverage analysis confirms that attackers are consistently detected across DDoS windows. These results indicate that the proposed method not only detects attacks effectively, but also reliably identifies the participating malicious nodes.

The results demonstrate that incorporating temporal modeling through Hidden Markov Models significantly enhances both detection accuracy and robustness, particularly under low synchronization conditions where traditional methods fail. Overall, the findings highlight the importance of synchronization as a key feature for distinguishing coordinated attacks and demonstrate the effectiveness of combining graph-based representations with temporal probabilistic models.

8.2 Key Contributions

This thesis makes the following key contributions:

• A synchronization-based detection framework that models network activity using graph representations, enabling effective differentiation between DDoS attacks and flash crowd events.

• The integration of temporal modeling through a Hidden Markov Model (HMM), significantly improving detection accuracy, reducing false negatives, and enabling robust performance under weak synchronization conditions.

• A synthetic traffic injection framework that allows controlled and reproducible evaluation across a wide range of attack parameters, including the number of attackers, synchronization level (overlap), jitter, and packet loss.

• A graph-based attacker identification method leveraging the largest connected component (LCC), achieving high precision and recall by capturing coordinated behavior among malicious nodes.

• A comprehensive experimental evaluation framework, including accuracy analysis, error rates, detection delay, critical synchronization thresholds, and attacker-level performance metrics such as precision, recall, F1-score, and coverage.

These contributions collectively demonstrate that combining structural graph-based analysis with temporal modeling provides a robust and effective approach for detecting and analyzing coordinated network attacks.

In conclusion, this work demonstrates that integrating structural and temporal analysis enables a powerful framework for detecting and understanding coordinated network attacks. By capturing both the relational structure of network activity and its temporal evolution, the proposed approach provides a robust solution to a complex problem in network security. The results indicate strong potential for real-world application in network monitoring and intrusion detection systems, where early and reliable identification of coordinated malicious behavior is critical.

9 Future Work

9.1 Model Improvements

While the proposed system achieves strong performance, several directions for further improvement can be explored.

One important extension is the incorporation of more advanced temporal models. Although the supervised Hidden Markov Model (HMM) provides significant improvements over static detection approaches, more expressive models such as deep learning-based sequence models (e.g., recurrent or transformer-based architectures) could capture more complex temporal dependencies and improve performance under highly irregular attack patterns.

Another potential improvement lies in enhancing the feature extraction process. Currently, the system relies primarily on graph-based metrics derived from activity patterns. Incorporating additional features, such as traffic volume statistics, flow-level characteristics, or protocol-level information, could further improve detection accuracy and robustness.

The synthetic attacker model could also be extended to include more heterogeneous and adaptive behaviors. For example, attackers could vary their activity dynamically, introduce temporal noise, or mimic legitimate user behavior. Such extensions would enable more realistic evaluation scenarios and help assess the resilience of the proposed method against sophisticated adversaries.

Finally, alternative graph-based techniques could be explored for attacker identification. While the current approach relies on the largest connected component (LCC), more advanced methods such as community detection

algorithms or graph clustering techniques may provide finer-grained identification and improved resistance to noise.

9.2 Real-time Detection

Another important direction for future work is the adaptation of the proposed system to real-time detection scenarios.

In its current form, the system operates in an offline setting, where the full dataset is available prior to analysis. For practical deployment, it is necessary to process incoming traffic streams continuously and make detection decisions in real time.

This transition would require efficient incremental computation of activity matrices, dynamic graph construction, and fast feature extraction mechanisms, as well as low-latency inference for the HMM model. Optimizing these components is essential to ensure scalability and responsiveness in real-world environments.

Additionally, real-time deployment introduces challenges such as incomplete observations, delayed data, and rapidly changing network conditions. The system must be able to handle these uncertainties while maintaining detection accuracy and stability.

Despite these challenges, the results of this thesis indicate that the proposed approach has strong potential for real-time applications, particularly in environments where coordinated attack behavior evolves over time and must be monitored continuously.

Feature-based pattern extraction and behavioral analysis techniques are commonly applied in network traffic mining tasks [9].

Bibliography

[1] Barford, P., & Plonka, D. (2001). Characteristics of network traffic flow anomalies. Proceedings of the ACM SIGCOMM Internet Measurement Workshop.

[2] Chandola, V., Banerjee, A., & Kumar, V. (2009). Anomaly detection: A survey. ACM Computing Surveys, 41(3), 1–58.

[3] Garcia, S., Grill, M., Stiborek, J., & Zunino, A. (2014). An empirical comparison of botnet detection methods. Computers & Security, 45, 100–123.

[4] Girvan, M., & Newman, M. E. J. (2002). Community structure in social and biological networks. Proceedings of the National Academy of Sciences, 99(12), 7821–7826.

[5] Mirkovic, J., & Reiher, P. (2004). A taxonomy of DDoS attack and DDoS defense mechanisms. ACM SIGCOMM Computer Communication Review, 34(2), 39–53.

[6] Newman, M. E. J. (2010). Networks: An Introduction. Oxford University Press.

[7] Rabiner, L. R. (1989). A tutorial on Hidden Markov Models and selected applications in speech recognition. Proceedings of the IEEE, 77(2), 257–286.

[8] Sommer, R., & Paxson, V. (2010). Outside the closed world: On using machine learning for network intrusion detection. IEEE Symposium on Security and Privacy, 305–316.

[9] Tan, P.-N., Steinbach, M., & Kumar, V. (2019). Introduction to Data Mining (2nd ed.). Pearson.

[10] Xu, K., Zhang, Z. L., & Bhattacharyya, S. (2005). Profiling internet backbone traffic: Behavior models and applications. Proceedings of the ACM SIGCOMM Conference.