electronics

MDPI

Article

A Time-Based Hierarchical Detection Method for TCP SYN Flood Attacks Using Self-Information and Three-Way Handshake Analysis

Nan-I Wu [1], Jia-Rong Sun [2], Cheng-Ying Yang [3] and Min-Shiang Hwang [4,5,*]

[1] Department of Information Management, Lunghwa University of Science and Technology, Taoyuan 33306, Taiwan; naiwu@gm.lhu.edu.tw
[2] BCCS Co., Ltd., Kaohsiung 80284, Taiwan; riley@bccs.com.tw
[3] Department of Computer Science, University of Taipei, Taipei 100234, Taiwan; cyang@utaipei.edu.tw
[4] AI Research Center, Department of Computer Science & Information Engineering, Asia University, Taichung 41354, Taiwan
[5] Department of Medical Research, China Medical University Hospital, China Medical University, Taichung 40402, Taiwan
* Correspondence: mshwang@nchu.edu.tw

Abstract

Controlled numerical analyses examine the mathematical relationships among connection-state probability, self-information, and entropy. The results demonstrate the theoretical feasibility and interpretability of the proposed framework. In particular, self-information provides a one-to-one mapping between a nonzero connection-state probability and its information value, whereas different probability distributions may produce identical entropy values. However, these proof-of-concept results do not constitute an operational IDS benchmark or establish detection performance in terms of Accuracy, FPR, FNR, or detection delay. Therefore, validation using labeled public datasets and real network environments is required before the framework can be considered for operational deployment.

Keywords: denial of service (DoS); distributed DoS (DDoS); syn flooding; self-information; hierarchical detection; three-way handshake

Academic Editor: Porfirio Tramontana

Received: 4 August 2026
Revised: 3 September 2026
Accepted: 14 September 2026
Published: 19 September 2026

1. Introduction

TCP SYN (synchronization) flood attack [1,2] is one of the most common denial-of-service (DoS) attacks. This attack exhausts the server's resources by creating numerous half-open TCP connections during the three-way handshake procedure. When launched as a Distributed Denial of Service (DDoS) attack [3–5], where multiple compromised hosts simultaneously generate malicious traffic, TCP SYN flooding can rapidly consume the victim server's connection backlog and computational resources, leading to service interruption. Although various countermeasures have been incorporated into modern operating systems and network devices, TCP SYN flooding remains one of the major Layer-4 attack vectors due to its simplicity, scalability, and ability to evade conventional filtering mechanisms. Recent surveys have also indicated that TCP SYN flood attacks continue to be an important research topic in cloud computing, Software-Defined Networking (SDN), fog computing, and Internet of Things (IoT) environments because of their real-time characteristics and low attack cost [6,7].

Consequently, numerous defense mechanisms [8] and detection methods [9–12] have been proposed during the past decade. Recent studies have mainly focused on three direc-

tions: machine learning and deep learning approaches [13–17], entropy-based statistical analysis [18], and hybrid detection frameworks integrating multiple traffic features [19–22]. Machine learning methods generally achieve high detection accuracy but require large labeled datasets and considerable computational resources, making them unsuitable for lightweight or real-time deployment. Hybrid detection frameworks further improve robustness by combining entropy analysis with SDN controllers or multiple statistical metrics, but their computational complexity and implementation cost are also increased [6,23,24]. Therefore, lightweight statistical detection methods remain attractive for practical TCP SYN flood detection, especially for high-speed servers and resource-constrained environments.

In this study, lightweightrefers to a detection mechanism that requires no offline model training, labeled dataset, packet-payload inspection, or high-dimensional feature extraction. The proposed method maintains only the counts of completed, unfinished, and timeout TCP connections within a fixed number of observation windows. Each connection event requires only counter updates, probability calculation, and self-information evaluation. Consequently, its processing cost increases linearly with the number of observed connections, while its storage requirement is determined by the fixed number of connection states and observation windows rather than by the size of a training dataset.

Among these proposed methods, Chen et al. [25] proposed an entropy-based detection method for identifying SYN flood attacks on email servers. Their approach analyzes the TCP three-way handshake process and email message flow within each observation time window. Since each detection is independently performed on a time window, the entropy of the TCP connection states is directly employed instead of long-term accumulated statistics. Consequently, the method does not require historical traffic records and can describe the server status using linguistic values for every observation period.

However, it should be noted that practical entropy-based detection systems do not rely solely on Shannon entropy. Instead, entropy is often combined with additional traffic features, adaptive thresholds, sequential hypothesis testing, or SDN-assisted traffic monitoring to improve detection robustness. Nevertheless, Shannon entropy remains the primary statistical indicator in these approaches. Since entropy measures the expected information of an entire probability distribution rather than the information associated with an individual event, different traffic distributions may produce similar entropy values even though their underlying packet behaviors are significantly different. This characteristic may reduce the discriminative capability of entropy itself and motivates the investigation of alternative statistical measures, such as self-information.

In contrast, self-information measures the information content associated with an individual event rather than the expectation over an entire probability distribution. Therefore, it provides a more direct representation of instantaneous network events during each sampling interval. In this paper, we investigate whether self-information can serve as an effective statistical indicator for TCP SYN flood detection. Based on this observation, we calculate the self-information of TCP connection states and incorporate it into a time-based hierarchical detection framework.

The objective of this study is to investigate whether self-information can provide a more explicit statistical representation of TCP connection states than Shannon entropy. Based on this property, we construct a lightweight hierarchical framework for interpreting potentially abnormal connection behavior. The present study focuses on the theoretical formulation and proof-of-concept analysis rather than claiming complete detection performance under real network traffic.

The advantages of the proposed method over other lightweight approaches are threefold.

1. We theoretically analyze the relationship between TCP connection-state probabilities, self-information, and Shannon entropy.

2. We propose a training-free hierarchical statistical framework that integrates connection-state information with multiple observation windows.
3. We conduct controlled numerical analyses to illustrate the discriminative properties of self-information. Comprehensive dataset-based detection evaluation is outside the scope of the present proof-of-concept study.

The remainder of this paper is organized as follows. Section 2 reviews TCP SYN flooding attacks, self-information theory, and representative lightweight detection methods, including SYN/ACK-based, threshold-based, and entropy-based approaches. Section 3 presents the proposed detection model and hierarchical algorithm. Section 4 analyzes the statistical behavior of the proposed method and compares it with representative lightweight TCP SYN flooding detection approaches. Finally, Section 5 concludes the paper.

2. Related Work

This section briefly reviews TCP SYN flooding attacks, self-information, and representative lightweight and learning-based detection methods, focusing on the characteristics and limitations relevant to the proposed framework.

TCP SYN flooding [1,2] is a Layer-4 DDoS attack that exploits the TCP three-way handshake and continues to threaten cloud, SDN, edge, and IoT environments [15,16,24]. Existing countermeasures include lightweight statistical, entropy-based, machine learning, and hybrid approaches [6,7,13,16,21,22]. This study employs self-information to reduce the statistical ambiguity associated with entropy-based detection [25].

2.1. TCP SYN Flooding Attack

A TCP connection is established through a three-way handshake: the client sends a SYN packet, the server replies with SYN-ACK, and the client completes the connection with ACK [1,2]. If the final ACK is not received, the server retains the half-open connection until timeout.

A TCP SYN flooding attack sends numerous SYN requests without completing the handshake, thereby exhausting the server's connection backlog and preventing legitimate connections. Botnets, spoofed addresses, and SDN-aware strategies may further increase the attack intensity and complicate detection [6,15,16]. Therefore, real-time monitoring of abnormal half-open connections is essential for detecting TCP SYN flooding attacks.

2.2. Self-Information

Self-information measures the information conveyed by an individual event, whereas entropy represents the expected information of an entire probability distribution [26]. For an event X_n with probability $P(X_n)$, its self-information is defined as

$$I(X_n) = \log_b \frac{1}{P(X_n)} = -\log_b P(X_n), \tag{1}$$

where b denotes the logarithmic base. If $P(X_n) = 0$, the event is not observed in the current interval and its self-information is not evaluated.

According to (1), rare events have greater self-information than frequent events. Moreover, $I(X_n)$ is strictly decreasing for $0 < P(X_n) \leq 1$; therefore, each nonzero probability corresponds to a unique self- information value. In contrast, different probability distributions may produce the same or similar entropy values. This property motivates the use of self-information to characterize TCP connection states.

2.3. Lightweight TCP SYN Flooding Detection

Lightweight TCP SYN flooding detection methods use limited packet- or connection-level statistics without requiring large labeled datasets, offline model training, or high-dimensional feature extraction. Representative methods include rate and ratio monitoring, connection-state counting, list-assisted filtering, entropy analysis, sequential testing, and SDN-assisted detection.

Rate- and threshold-based methods monitor indicators such as the SYN arrival rate, SYN-to-ACK ratio, or number of half-open connections. Although computationally efficient, they are sensitive to threshold selection and may confuse legitimate traffic bursts with attacks. Yang et al. [1] combined SYN/ACK analysis with black and white lists, but this approach requires source- address maintenance and may be affected by spoofed or frequently changing addresses.

Chen et al. [25] calculated entropy from completed and unfinished events within each observation interval. Ali et al. [18] combined entropy with the Sequential Probability Ratio Test to accumulate evidence across successive intervals, but this requires statistical hypotheses and decision boundaries. SynFloWatch [7] integrates entropy analysis with SDN traffic monitoring, although it requires SDN switches and controller-side processing.

Thus, existing lightweight methods involve different trade-offs: rate-based methods are threshold-sensitive, list- assisted methods require address maintenance, sequential methods require parameter calibration, and SDN-assisted methods depend on additional infrastructure. Moreover, entropy may assign the same or similar values to different traffic distributions. These limitations motivate a training-free framework that provides more direct and interpretable representations of TCP connection states.

2.4. Machine Learning-Based TCP SYN Flooding Detection

Machine learning and deep learning methods formulate SYN flooding detection as a traffic-classification or anomaly-detection problem [13,15,16,24]. Although these methods can learn complex traffic patterns, they generally require representative datasets, feature extraction, model training, and periodic updating. Their computational and deployment requirements differ from the objective of this study, which is to develop a training-free and interpretable statistical framework. Therefore, detailed reviews of individual learning algorithms are beyond the scope of this paper.

2.5. Research Gap and Advantages of the Proposed Method

The preceding review reveals that existing TCP SYN flooding detection methods exhibit several unresolved trade-offs. Machine learning and deep learning approaches can learn complex attack patterns, but usually depend on representative training data, feature engineering or feature learning, model optimization, and periodic updating. Conventional lightweight approaches avoid these requirements, but many rely on a single traffic rate, a fixed threshold, a source-address list, or an aggregate statistical value. Consequently, they may provide limited information about the progression of individual TCP connection states and the persistence of resource- consuming half-open connections.

A further limitation concerns the use of Shannon entropy as the primary statistical indicator. Entropy represents the expected information of a probability distribution rather than the information associated with an individual connection event. Therefore, different distributions of completed and unfinished connections may yield the same or similar entropy values. Adding adaptive thresholds, sequential hypothesis testing, or SDN-based traffic collection may improve the overall detection framework, but these mechanisms do not eliminate the many-to- one relationship inherent in the entropy measure itself. This

motivates the use of self-information, which directly associates an observed event with its occurrence probability.

The proposed method addresses these gaps by integrating self-information, multiple observation windows, and hierarchical connection-state analysis into a unified lightweight framework. In this study, lightweight means that the detection mechanism requires no labeled dataset, offline model training, high-dimensional feature extraction, packet-payload inspection, source-address list, or additional SDN infrastructure. The method maintains only a fixed set of statistics for completed, unfinished, and timeout TCP connections over predefined observation windows. For a fixed number of connection states and observation windows, processing n observed connection events requires $O(n)$ time, while the maintained state does not grow with the size of a training dataset.

Compared with other lightweight approaches, the proposed method has the following advantages:

- Event-oriented statistical representation: Self-information directly measures the information associated with each observed TCP connection state. Its one-to-one relationship with a nonzero event probability avoids the statistical ambiguity that may arise when different traffic distributions produce the same entropy value.
- Joint connection-state analysis: Rather than relying only on the SYN arrival rate, SYN-to-ACK ratio, or a source-address list, the method jointly analyzes completed, unfinished, and timeout connections. These states provide a more direct representation of the TCP three-way handshake and the accumulation of half-open connections.
- Multiple observation windows: The method evaluates connection behavior over different time intervals. Short windows support timely identification of abnormal behavior, whereas longer windows reveal whether unfinished connections continue to accumulate and potentially exhaust server resources.
- Hierarchical and interpretable decisions: The first detection layer evaluates the security state of the server, whereas the second layer estimates its resource-utilization condition. The numerical detection results are translated into linguistic system states, allowing administrators to interpret the detected condition without analyzing opaque model outputs.
- Low deployment requirements: The proposed method does not require model training, periodic model updating, black- or white-list maintenance, or an SDN controller. It can, therefore, be implemented as an online monitoring component on a conventional TCP server.

Recent studies have demonstrated that jointly analyzing complementary dimensions can provide more comprehensive representations than relying on a single indicator. Liang et al. [27] jointly optimized spatial position, beamforming, and power allocation for secure IoT communications, whereas Zhang et al. [28] integrated scale, spectrum, and state information for real-time detection. Although their application domains differ from TCP SYN flooding detection, their multidimensional joint and collaborative designs support the general principle of combining complementary information. Following this principle, the proposed framework jointly considers connection-state information, multiple observation windows, and resource occupancy rather than relying on a single traffic count or entropy value.

Accordingly, the main contribution of the proposed method is not merely a reduction in computational overhead relative to machine learning and deep learning models. Its distinctive advantage lies in jointly providing event-level statistical interpretation, temporal connection analysis, security-state detection, and resource-utilization assessment within a training-free framework. The proposed method is therefore intended as an interpretable

lightweight alternative for environments in which computational resources, labeled data, or additional network infrastructure are limited.

3. The Proposed Statistical Framework

The proposed method is designed to detect TCP SYN flooding attacks by employing self-information to overcome the limitations of the entropy-based detection method proposed by Chen et al. [25]. The fundamental idea is to monitor the number of completed and unfinished TCP connections within each sampling interval and calculate their corresponding self-information values. These measurements are then used to characterize the current workload and security status of the server. Compared with entropy-based approaches, the proposed method directly measures the information associated with individual TCP connection events, thereby reducing the ambiguity caused by different traffic distributions producing similar entropy values.

Before calculating the self-information, the proposed method checks whether the corresponding connection state has occurred during the current observation interval. If the number of a specific connection type is zero, its occurrence probability is treated as zero, and the corresponding self-information is not calculated because the event does not appear in the current sampling interval.

To further improve the detection capability, a time-based hierarchical detection algorithm is introduced based on the TCP connection process. The proposed algorithm consists of two hierarchical detection layers. The first layer employs self-information to evaluate the security state of the server by identifying abnormal TCP connection behavior. The second layer does not sum self-information values. Instead, it estimates connection-resource utilization from the cumulative number of unfinished connections observed over the resource release interval. This separation is necessary because self-information measures the statistical unexpectedness of a connection state, whereas the number and persistence of unfinished connections are directly related to the occupation of the TCP connection backlog and associated server resources.

Each layer employs an appropriate observation interval according to its detection objective. Consequently, the proposed method enables continuous monitoring while allowing the detection frequency to be adjusted according to different server workloads and system requirements, thereby providing comprehensive and flexible real-time detection.

Since the proposed method is specifically designed for TCP SYN flooding attacks, its architecture follows the standard TCP connection procedure. Figure 1 illustrates the TCP connection process using a finite-state machine consisting of four phases: Connection Start, TCP Connection, Transport Connection, and Close Connection. This state-machine model serves as the theoretical foundation for the proposed detection approach.

In the proposed state machine, each phase is associated with one or more observation time windows. The TCP Connection phase and the Transport Connection phase each consist of two observation time windows, whereas the Connection Start phase and the Close Connection phase each contain a single observation time window. Consequently, the minimum execution time of a complete TCP connection process is six time units. The TCP Connection phase corresponds to the TCP three-way handshake, while the Transport Connection phase represents the data transmission process between the client and the server. If either the connection establishment or the data transmission is not successfully completed, the process returns to the previous state until the waiting time expires, after which the occupied system resources are released. Based on the characteristics of the TCP connection process, the proposed method consists of the following two components.

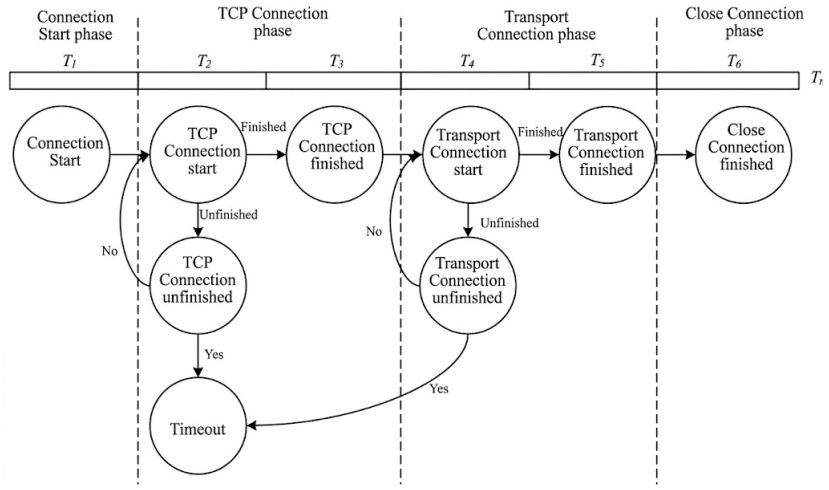

Figure 1. State machine of TCP connection process.

1. Time-Based Detection Approach Using Self-Information

   The proposed detection approach aims to identify TCP SYN flooding attacks by monitoring the number of TCP connection requests and completed TCP connections during each sampling interval. The underlying assumption is that a TCP SYN flooding attack introduces several observable changes in the server behavior. First, the number of incoming TCP connection requests increases significantly, resulting in a corresponding increase in server workload. Second, although the number of connection requests continues to increase, the number of successfully completed TCP connections decreases because many connection requests remain unfinished. Consequently, the number of half-open (static) TCP connections gradually accumulates. Finally, when these half-open connections exceed the predefined waiting time, timeout events occur, and the corresponding system resources are released. Based on these observations, the proposed detection architecture is illustrated in Figure 2.

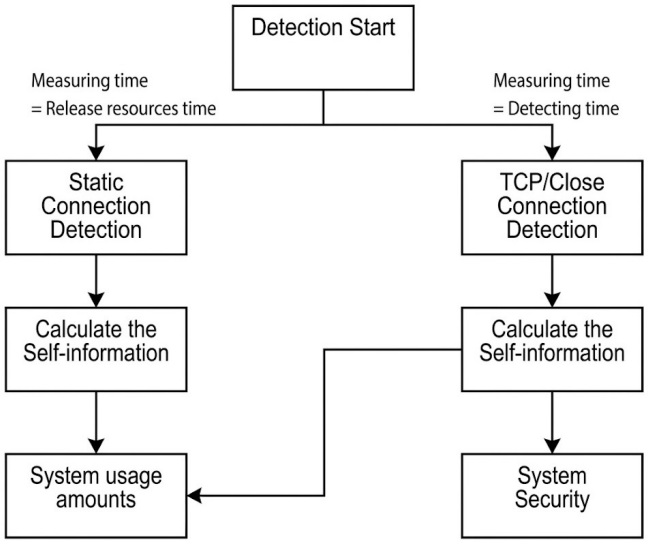

Figure 2. Detection approach architecture.

As shown in Figure 2, the proposed architecture consists of two detection modules: Static Connection Detectionand TCP/Close Connection Detection. The former is executed whenever the measuring time reaches the resource release interval, whereas the latter is executed whenever the measuring time reaches the predefined detection interval. Since server resource utilization changes more slowly than TCP connection

states, the resource release interval is configured to be longer than the detection interval. Both intervals can be adjusted by the system administrator according to the server workload and operational requirements.

During the TCP/Close Connection Detection phase, the numbers of completed and unfinished TCP connections are collected within each sampling interval, and their corresponding self-information values are calculated to determine whether abnormal TCP connection behaviors exist. Subsequently, the numbers of completed and timeout close connections are also evaluated using self-information to determine whether timeout events increase abnormally. According to these statistical results, the current security state of the server is determined.

During the Static Connection Detection phase, the number of unfinished TCP connections is sampled at each detection interval. These observations are accumulated over the resource release interval to estimate the persistence of half-open connections. The resulting cumulative unfinished-connection occupancy represents a discrete approximation of the area under the half-open connection queue-length curve. Therefore, it reflects both the number of unfinished connections and the duration for which they occupy server connection resources. This count-based quantity, rather than the sum of their self-information values, is used to determine the relative resource-utilization state.

2. Hierarchical Detection Algorithm

The proposed algorithm evaluates the security status of TCP connections and provides a statistical indication of connection resource utilization based on the observed TCP connection states. For security assessment, the self-information values of completed close connections, timeout connections, completed TCP connections, and unfinished TCP connections are calculated to determine the current security state of the server. Accordingly, three linguistic security states are defined: System Safe, System Normal, and System Abnormal.

For security assessment, the first detection layer calculates the self-information values of completed, unfinished, active-close, and timeout connection states. These values characterize whether the observed connection behavior is statistically normal or abnormal.

For resource assessment, the second detection layer directly accumulates the observed number of unfinished connections over the resource release interval. The cumulative value measures unfinished-connection occupancy and is partitioned into four linguistic resource states: Safe, Normal, Abnormal, and Overload. Accordingly, self-information is used to characterize statistical abnormality, whereas cumulative connection occupancy is used to characterize resource consumption. Before presenting the proposed algorithm, the notations used throughout the algorithm are summarized in Table 1.

Algorithm 1 summarizes the workflow of the proposed hierarchical TCP SYN flooding detection framework. For each observed TCP connection event, the algorithm updates the completed, unfinished, and timeout connection counters. At every detection interval T_d, it calculates the corresponding self- information values and determines the server security state. At every resource release interval T_r, it evaluates the normalized cumulative occupancy of unfinished connections and classifies the resource state as Safe, Normal, Abnormal, or Overload.

Table 1. Notation used in the proposed hierarchical detection algorithm.

Notation	Definition
P	The monitored TCP transport protocol.
T_m	The current measurement time.
T_d	The security-detection interval.
T_r	The resource release interval, where $T_r > T_d$.
K	The number of security-detection intervals contained in one resource release interval, where $K = T_r/T_d$.
Δt	The duration of one security-detection interval.
n_{T_m}	The number of normally completed close connections observed during T_m.
a_{T_m}	The number of connections that terminate because of timeout during T_m.
u_{T_m}	The number of unfinished TCP connections remaining active during T_m.
c_{T_m}	The number of TCP connections that successfully complete the three-way handshake during T_m.
d_{T_m}	The number of TCP connections that fail to complete the three-way handshake during T_m.
C_{P,T_m}	The total number of observed TCP connection-state events during T_m.
$C(P, T_m, s)$	The number of observations belonging to connection state s during T_m, where $s \in \{n, a, u, c, d\}$.
p_{s,T_m}	The observed probability of connection state s during T_m, defined as $p_{s,T_m} = C(P, T_m, s)/C_{P,T_m}$.
I_{s,T_m}	The self-information of connection state s during T_m, defined as $I_{s,T_m} = -\log_2 p_{s,T_m}$ for $p_{s,T_m} > 0$.
x_{P,T_m}	The self-information of normally completed close connections during T_m, i.e., $x_{P,T_m} = I_{n,T_m}$.
y_{P,T_m}	The self-information of timeout close connections during T_m, i.e., $y_{P,T_m} = I_{a,T_m}$.
z_{P,T_m}	The self-information of unfinished TCP connections during T_m, i.e., $z_{P,T_m} = I_{u,T_m}$. This value is used for security-state analysis and is not accumulated to estimate resource utilization.
c_{P,T_m}	The self-information of TCP connections that successfully complete the three-way handshake during T_m, i.e., $c_{P,T_m} = I_{c,T_m}$.
d_{P,T_m}	The self-information of TCP connections that fail to complete the three-way handshake during T_m, i.e., $d_{P,T_m} = I_{d,T_m}$.
R_a	The decision threshold used to identify abnormal connection behavior in the first-layer security assessment.
R_n	The decision threshold used to distinguish normal from safe connection behavior in the first-layer security assessment.
B	The configured capacity of the server's TCP half-open connection backlog.
U_{T_r}	The cumulative unfinished-connection occupancy during T_r, defined as $U_{T_r} = \sum_{k=1}^{K} u_k$.
A_{T_r}	The unfinished-connection-time exposure during T_r, defined as $A_{T_r} = \Delta t \sum_{k=1}^{K} u_k$.
ρ_{T_r}	The normalized unfinished-connection occupancy during T_r, defined as $\rho_{T_r} = (KB)^{-1} \sum_{k=1}^{K} u_k$. It represents the average fraction of the TCP half-open connection backlog occupied during T_r.
θ_s	The threshold between the Safe and Normal resource-utilization states.
θ_n	The threshold between the Normal and Abnormal resource-utilization states.
θ_a	The threshold between the Abnormal and Overload resource-utilization states, where $0 \le \theta_s < \theta_n < \theta_a \le 1$.

Algorithm 1 Proposed Hierarchical TCP SYN Flooding Detection

Require: TCP connection events, T_d, T_r, and decision thresholds
Ensure: Security state and resource-utilization state
  Initialize all connection-state counters
  for each TCP connection event **do**
    Update the completed, unfinished, and timeout counters
    if T_m mod $T_d = 0$ **then**
      Calculate the connection-state probabilities
      Calculate the corresponding self-information values
      Determine the security state as Safe, Normal, or Abnormal
    end if
    if T_m mod $T_r = 0$ **then**
      Calculate the normalized unfinished-connection occupancy
      Determine the resource state as Safe, Normal, Abnormal, or Overload
      Reset the interval counters
    end if
  end for

3. Parameter Selection

Let u_k denote the number of unfinished TCP connections observed at the kth detection interval within a resource release interval T_r, and let $K = T_r/T_d$. The cumulative unfinished-connection occupancy is defined as

$$U_{T_r} = \sum_{k=1}^{K} u_k.$$

If the sampling interval has duration Δt, the corresponding connection-time exposure is

$$A_{T_r} = \Delta t \sum_{k=1}^{K} u_k.$$

Thus, A_{T_r} is a discrete approximation of the area under the half-open connection queue-length curve. Its unit is connection-time, and a larger value indicates that more unfinished connections remain active for longer periods.

To support comparisons among servers with different TCP backlog capacities, the normalized unfinished- connection occupancy is defined as

$$\rho_{T_r} = \frac{1}{KB} \sum_{k=1}^{K} u_k,$$

where B is the configured TCP half-open connection backlog capacity. The value ρ_{T_r} represents the average fraction of the backlog occupied by unfinished connections during T_r. When the observed number of unfinished connections does not exceed B, $0 \leq \rho_{T_r} \leq 1$.

The proposed method employs three decision thresholds, R_n and R_a, together with two time windows, T_d and T_r. These parameters are implementation-dependent rather than fixed theoretical constants.

In practice, R_n and R_a can be determined from the statistical characteristics of normal network traffic collected during an initial training period. Specifically, the thresholds are selected such that

$$R_n < R_a,$$

where larger values correspond to increasingly abnormal connection behaviors.

The time windows T_d and T_r should be selected according to the expected traffic volume and the desired trade-off between detection responsiveness and statistical stability. A smaller time window provides faster attack detection but may increase traffic fluctuations, whereas a larger time window provides smoother statistical estimation at the expense of increased detection delay.

The parameters T_d, T_r, R_n, and R_a affect the trade-off between detection responsiveness and stability. A smaller T_d provides faster detection but may be more sensitive to short-term traffic fluctuations, whereas a larger T_d produces more stable estimates at the cost of increased detection delay. Similarly, a smaller T_r enables faster resource assessment, while a larger T_r better captures the persistent accumulation of unfinished connections.

In practice, the decision thresholds should be calibrated using benign traffic collected from the protected server. During an initial benign-traffic training period, administrators may calculate the empirical distributions of the self-information differences and select R_a as a lower-tail quantile corresponding to the acceptable false- alarm rate, with $R_n > R_a$ selected to distinguish normal from clearly safe behavior. Similarly, θ_s, θ_n, and θ_a may be initialized using ordered upper quantiles of the benign backlog-occupancy distribution and adjusted according to the TCP backlog capacity and the server's operational tolerance. These thresholds should be periodically recalibrated when the workload or traffic profile changes substantially.

Lower thresholds increase detection sensitivity but may produce more false alarms, whereas higher thresholds reduce false alarms but may delay or miss low-rate attacks. Therefore, the time windows and thresholds should be selected according to the server workload, TCP backlog capacity, and required response time. A quantitative sensitivity analysis under dynamic traffic will be conducted in future work.

For resource-utilization assessment, the normalized unfinished-connection occupancy ρ_{T_r} is compared with three ordered thresholds:

$$0 \leq \theta_s < \theta_n < \theta_a \leq 1,$$

where θ_s, θ_n, and θ_a delimit the Safe, Normal, Abnormal, and Overload states, respectively. These thresholds should be selected according to the configured TCP backlog capacity, the normal occupancy distribution, and the operational tolerance of the protected server.

In the proposed detection algorithm, the information unit is represented by the binary logarithm ($b = 2$), since each TCP connection can be classified into two complementary states, such as completed or unfinished, normal or abnormal, and online or timeout.

When $T_m \bmod T_d = 0$, the algorithm performs the security assessment by evaluating the self-information differences $(x_{P,T_m} - y_{P,T_m})$ and $(c_{P,T_m} - d_{P,T_m})$. If $(x_{P,T_m} - y_{P,T_m}) \leq R_a$, it indicates that a large number of timeout connections have occurred during the current sampling interval. Similarly, if $(c_{P,T_m} - d_{P,T_m}) \leq R_a$, it indicates that many TCP connections have failed to complete the three-way handshake. Either condition implies that the server resources are being rapidly exhausted, and the server is, therefore, classified as System Abnormal.

If

$$R_a < (x_{P,T_m} - y_{P,T_m}) \leq R_n$$

and

$$R_a < (c_{P,T_m} - d_{P,T_m}) \leq R_n,$$

the numbers of completed and unfinished TCP connections remain approximately balanced, and the numbers of timeout and active connections are also comparable. This observation

indicates that the current TCP connection behavior does not exhibit either a clearly normal or a clearly abnormal pattern. Accordingly, the server is classified as System Normal.

When

$$(x_{P,T_m} - y_{P,T_m}) > R_n$$

and

$$(c_{P,T_m} - d_{P,T_m}) > R_n,$$

most TCP connections are successfully completed, indicating that the observed TCP connection behavior is consistent with normal network operation. Accordingly, the server is classified as System Safe. Since the variations of TCP connection states and closed connection states are generally proportional within the same sampling interval, their differences are expected to remain consistent. Therefore, all remaining cases are also regarded as System Normal.

When $T_m \bmod T_r = 0$, the second detection layer evaluates the connection-resource utilization by calculating the normalized unfinished-connection occupancy ρ_{T_r}. Unlike the first detection layer, this assessment does not sum self-information values. The quantity ρ_{T_r} measures the average fraction of the TCP backlog occupied by unfinished connections during the resource release interval.

The server resource state is determined as follows:

$$\text{State}_r = \begin{cases} \text{Safe}, & 0 \le \rho_{T_r} < \theta_s, \\ \text{Normal}, & \theta_s \le \rho_{T_r} < \theta_n, \\ \text{Abnormal}, & \theta_n \le \rho_{T_r} < \theta_a, \\ \text{Overload}, & \theta_a \le \rho_{T_r}. \end{cases} \tag{2}$$

A large value of ρ_{T_r} indicates that a substantial proportion of the TCP half-open connection backlog remains occupied over successive detection intervals. Therefore, the resource-utilization assessment captures both the number and persistence of unfinished connections.

Within each detection interval, every TCP connection is assigned to exactly one connection state; therefore, no connection is counted simultaneously as completed, unfinished, and timeout within the same interval. However, an unfinished connection that persists across multiple intervals contributes once to the unfinished-connection count in each corresponding interval. This repeated observation is intentional rather than duplicate counting because it represents the duration for which the connection continues to occupy backlog and connection- management resources. Once the connection is completed or times out, it no longer contributes to the unfinished-connection occupancy in subsequent intervals.

4. Controlled Proof-of-Concept Analysis

The numerical experiments in this section are controlled proof-of-concept analyses. They are designed to examine the mathematical relationship between connection-state probability, self-information, and entropy under identical conditions. They are not intended to estimate Accuracy, Precision, Recall, F1-score, FPR, FNR, or AUC, because the generated values do not constitute labeled real network traces. Accordingly, the results should not be interpreted as a complete performance evaluation of an operational intrusion detection system.

This section first analyzes the differences between the proposed approach and representative entropy-based detection methods, including the integrated entropy measurement proposed by Chen et al. [25], the entropy-based sequential hypothesis testing approach proposed by Ali et al. [18], and the Hybrid SDN-based entropy detection framework, SynFloWatch, proposed by Sinha et al. [7]. Since Chen et al.'s method [25] is the most closely

related work and serves as the foundation of our proposed approach, the simulation study primarily compares the proposed self-information-based method with the entropy-based method of [25] through curve distribution analysis. Furthermore, different numbers of completed TCP connections are simulated to evaluate the detection behavior under TCP SYN flooding attacks.

In addition to the simulation results, a qualitative comparison is conducted with the representative entropy-based methods [7,18,25]. The comparison focuses on several important characteristics, including the misjudgment rate, detection mechanism, real-time monitoring capability, abnormal connection detection, half-open connection detection, applicability, and computational complexity. These comparisons provide qualitative observations regarding the characteristics of the proposed approach and existing entropy-based methods.

4.1. Simulation and Analysis

The experiments in this subsection are controlled proof-of-concept analyses designed to examine the mathematical behavior of self-information and entropy under identical connection-state conditions. They illustrate how the proposed framework represents completed and unfinished TCP connections but do not constitute a comprehensive evaluation of operational intrusion detection performance.

Chen et al. [25] considered completed and unfinished request–response pairs in an email-server environment, whereas the present study analyzes completed and unfinished TCP handshakes. Therefore, the two methods are not compared at the application-protocol or operational-performance level. Instead, the comparison adopts their common binary statistical abstraction: each event within an observation interval is classified as completed or unfinished, and both statistical measures are calculated from the same controlled connection counts. Accordingly, Reference [25] is used only as an analytical baseline for comparing the mathematical behavior of Shannon entropy and self-information under identical probability inputs.

The controlled traffic patterns are intentionally simplified to clarify the statistical differences between self- information and entropy rather than to reproduce real-world network traffic. The thresholds used in the simulations are selected only to illustrate the operation of the proposed hierarchical framework.

In each sampling interval, the numbers of completed and abnormal connections vary complementarily from $(1, 300)$ to $(300, 1)$, resulting in a total of 301 connection-state observations. Figure 3 presents the corresponding self-information values, while Figure 4 presents the entropy values. The differences between the completed and abnormal states are shown in Figures 5 and 6, respectively.

To demonstrate how the hierarchical algorithm processes the ambiguity illustrated in Figure 6, two representative traffic conditions are examined. Case A contains 15 completed and 286 abnormal connections, whereas Case B contains 121 completed and 180 abnormal connections. Both cases contain 301 connection-state observations.

Let

$$p_n = \frac{n}{n+a}, \qquad p_a = \frac{a}{n+a},$$

where n and a denote the numbers of completed and abnormal connections, respectively. The entropy-component difference is defined as

$$E = |-p_n \log_2 p_n - (-p_a \log_2 p_a)|.$$

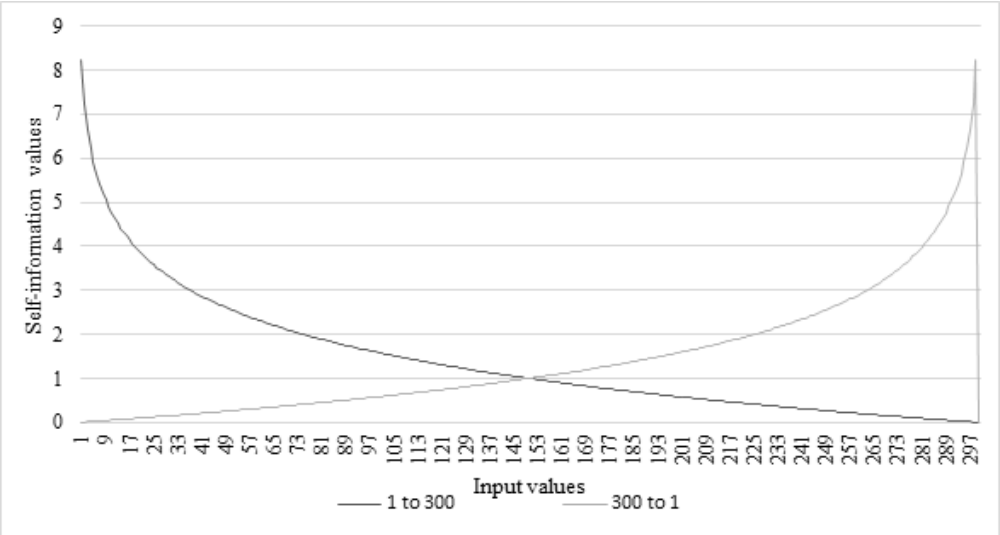

Figure 3. Self-information values of the connection states.

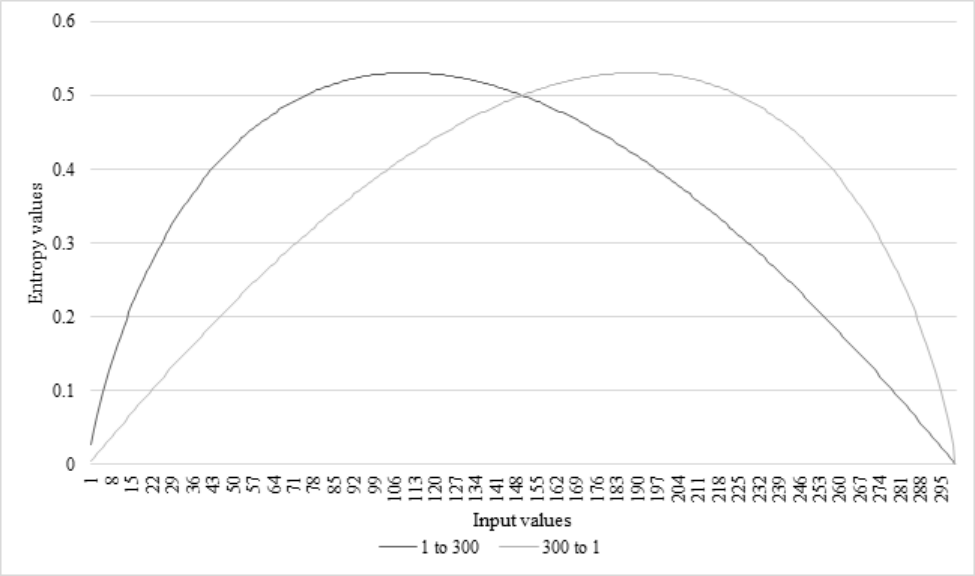

Figure 4. Entropy values of the connection states.

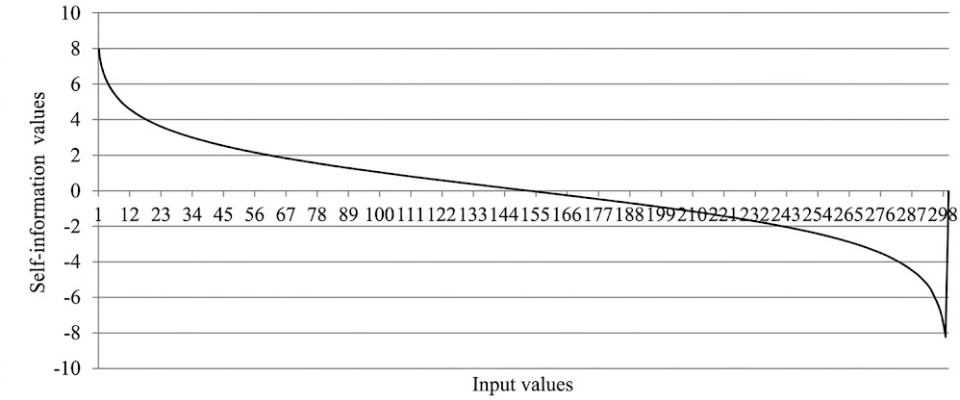

Figure 5. Differences between the self-information values of completed and abnormal connections.

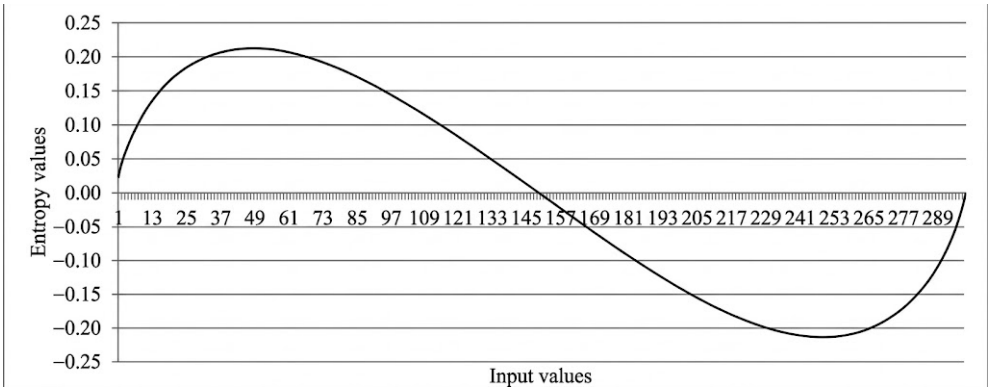

Figure 6. Differences between the entropy components of completed and abnormal connections.

For the self-information-based method, the connection-state difference used in the hierarchical decision is

$$\begin{aligned} D &= I_n - I_a \\ &= -\log_2 p_n + \log_2 p_a \\ &= \log_2\left(\frac{p_a}{p_n}\right) = \log_2\left(\frac{a}{n}\right). \end{aligned}$$

Thus, D represents the logarithm of the ratio of abnormal to completed connections. A larger positive value indicates a greater dominance of abnormal connections within the observation interval.

As shown in Table 2, the entropy-component differences are 0.1455 and 0.0849, both of which appear as approximately 0.1 in Figure 6 when displayed to one decimal place. In contrast, the corresponding self-information differences are clearly separated at 4.2530 and 0.5730.

Table 2. Hierarchical classification of two traffic conditions with similar entropy-component differences.

Case	n	a	a/n	E	$D = \log_2(a/n)$	Hierarchical State
A	15	286	19.067	0.1455	4.2530	Abnormal
B	121	180	1.488	0.0849	0.5730	Normal

For this illustrative decision, $D \geq 1$ is classified as Abnormal, which is equivalent to $a/n \geq 2$; that is, the number of abnormal connections is at least twice the number of completed connections. A value of $0 \leq D < 1$ is classified as Normal. Accordingly, Case A is classified as Abnormal, whereas Case B is classified as Normal. This example demonstrates that the self-information difference preserves the underlying connection-state ratio when the entropy-component differences appear similar at the displayed precision.

Figures 4 and 6 also illustrate that the entropy component $-p\log_2 p$ is non-monotonic. For example, an entropy value of approximately 0.3 in Figure 4 corresponds to traffic conditions containing approximately 25 and 230 completed connections. Similarly, the two conditions containing approximately 15 and 121 completed connections produce entropy-component differences close to 0.1 in Figure 6. Thus, different connection-state probabilities may produce the same or similar entropy measurements.

In contrast, $I(p) = -\log_2 p$ is strictly decreasing for $0 < p \leq 1$, so each nonzero event probability corresponds to a unique self-information value. This property provides a more explicit representation of the connection-state probability. The comparison demonstrates a mathematical distinction between the two statistical measures but does not, by itself, establish superior detection accuracy over other entropy-based frameworks.

The following simulation randomly generates 100 TCP sessions within one sampling interval. Although SMTP traffic is used to generate the sessions, only the SYN, SYN-ACK, and ACK packets involved in TCP connection establishment are analyzed; subsequent application-layer message exchanges are excluded. Therefore, the proposed framework does not depend on the application-layer protocol.

Figure 7 presents the generated TCP connection states. A completed handshake is represented by zero, whereas an unfinished handshake is represented by the number of missing steps required to complete the TCP three-way handshake.

Figure 8 presents the corresponding self-information values. The unfinished connections are represented by distinct self-information values and subsequently processed by the hierarchical decision rules. Because the entropy behavior has already been illustrated in Figures 4 and 6, the additional entropy plot based on the formulation of Chen et al. [25] has been removed to avoid redundancy.

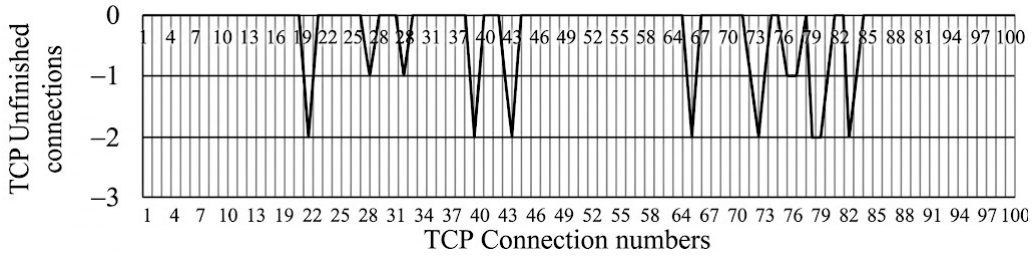

Figure 7. Generated TCP connection states.

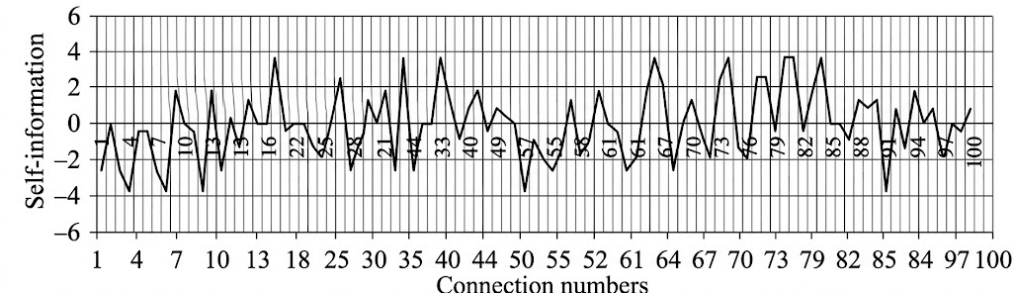

Figure 8. Self-information values of the generated TCP connection states.

The proposed framework does not rely solely on a difference between two connection-state counts. Instead, its hierarchical decision process jointly evaluates the self-information values of multiple TCP connection states over predefined observation intervals. The results in Figures 3–8 are intended to illustrate these statistical properties rather than to quantify detection accuracy, false-positive rate, or false-negative rate.

The controlled simulations do not reproduce the full variability of real network traffic, including flash crowds, retransmissions, packet loss, asymmetric routing, mixed attacks, and changing background traffic. Therefore, the present results establish the feasibility and interpretability of the proposed statistical framework but do not provide Accuracy, F1-score, FPR, or detection-delay measurements for real network environments.

4.2. Qualitative Comparison with Existing Approaches

This subsection qualitatively compares the proposed framework with representative lightweight statistical approaches. The comparison focuses on their statistical indicators, decision structures, connection-state analysis, infrastructure requirements, and theoretical computational complexity. Because these methods have not been implemented and evaluated under identical traffic conditions, the comparison does not establish quantitative

superiority in terms of Accuracy, Precision, Recall, F1-score, FPR, FNR, AUC, or detection delay.

Chen et al. [25] are retained as the foundational analytical baseline because the controlled analysis applies their entropy formulation to the same connection-state probabilities used by the proposed framework. To reflect recent developments, the comparison also includes SynFloWatch [7], SYNTROPY [29], and SF-DaM [30]. These methods represent Tsallis-entropy-based, Rényi-entropy-based, and multi-phase threshold- and behavior-based TCP SYN flooding detection, respectively.

The proposed framework differs from the considered approaches in its statistical representation and decision structure. Chen et al. [25] use Shannon entropy in a single-stage statistical decision, whereas the proposed framework incorporates self-information into a hierarchical structure comprising security-state and resource-utilization assessments. Ali et al. [18] combine entropy with SPRT to accumulate evidence across successive observations, but their method requires predefined statistical hypotheses and decision boundaries. SynFloWatch [7] integrates entropy analysis with traffic monitoring in a hybrid SDN environment, whereas the proposed framework is designed to operate on a conventional TCP server without requiring an SDN controller.

Compared with the SYN/ACK- and list-assisted approach of Yang et al. [1], the proposed framework does not require black- or white-list maintenance. It jointly analyzes completed, unfinished, and timeout connections and evaluates the persistence of unfinished connections over multiple observation windows. It, therefore, provides both a security-state assessment and a connection-resource utilization indicator.

For a fixed number of connection states and observation windows, the proposed framework performs a constant number of counter updates, probability and self-information calculations, and threshold comparisons for each connection event. Therefore, processing n connection events has a theoretical upper-bound time complexity of $O(n)$. This estimate is derived analytically from the per-event online operations and should not be interpreted as an experimentally measured execution-time advantage. Physical execution-time profiling under high-throughput traffic will be conducted in future deployment studies. The structural characteristics of the considered approaches are summarized in Table 3.

Table 3. Qualitative comparison of the structural characteristics of representative lightweight detection approaches.

Characteristic	Proposed	Chen et al. [25]	SynFloWatch [7]	SYNTROPY [29]	SF-DaM [30]
Year	2026	2014	2024	2024	2026
Primary indicator	Self-information	Shannon entropy	Tsallis entropy	Rényi entropy	Counters and behavior
Decision structure	Hierarchical	Single-stage	Entropy-based	Multi-stage	Multi-phase
Connection-state analysis	Yes	Limited	Yes	Yes	Yes
Source-spoofing analysis	No	No	No	No	Yes
Additional network infrastructure	No	No	Hybrid SDN	SDN	SDN
Offline model training	No	No	No	No	No
Resource-state assessment	Yes	No	No	No	No
Theoretical time complexity	$O(n)$	Not reported	Not reported	Not reported	Not reported

The entries in Table 3 are based on the structural and operational characteristics reported in the corresponding studies. They should not be interpreted as a quantitative performance ranking because the methods were evaluated using different datasets, traffic conditions, parameter settings, and deployment environments.

The comparison indicates that the proposed framework integrates self-information, multiple observation windows, and hierarchical connection-state analysis without requiring offline model training or additional SDN infrastructure. These are structural distinctions rather than experimentally verified performance advantages. Direct benchmark experiments under identical traffic conditions are required to establish quantitative superiority.

The $O(n)$ complexity reported for the proposed framework is a theoretical estimate derived from its principal online operations. It should not be interpreted as an experimentally measured execution-time advantage. For the other approaches, "Not reported" indicates that their theoretical time complexities were not explicitly provided in the corresponding studies.

4.3. Limitations

The present study has several limitations. First, the controlled numerical scenarios are intentionally simplified and do not reproduce the background-traffic variability, packet loss, retransmissions, asymmetric routing, flash crowds, or mixed attacks observed in operational networks. Second, because the experiments do not contain independently labeled benign and attack windows, standard classification metrics, including Accuracy, Precision, Recall, F1-score, FPR, FNR, and ROC/AUC, cannot be reliably calculated. Third, the selected thresholds have not been calibrated or evaluated across different network environments. Therefore, the current results demonstrate only the mathematical feasibility and interpretability of the proposed statistical framework.

Future work will validate the framework using raw packet traces containing both benign and TCP SYN flooding traffic, such as CICDDoS2019, together with deployment-based experiments. This validation is required before conclusions can be drawn regarding practical detection effectiveness. Accordingly, the proposed framework should not yet be regarded as a replacement for a fully evaluated operational intrusion detection system.

5. Conclusions

This paper has presented a lightweight hierarchical framework for analyzing TCP SYN flooding behavior by integrating self-information, multiple observation windows, and connection-state analysis. Unlike conventional entropy-based approaches, the proposed framework uses self-information to characterize individual TCP connection states and monitors their temporal evolution without requiring offline model training or additional SDN infrastructure.

The proposed framework evaluates two complementary aspects of the server. The first layer uses the self- information of completed, unfinished, normally closed, and timeout connections to determine the server security state. The second layer estimates connection-resource utilization from the normalized cumulative occupancy of unfinished connections. This count-based indicator approximates the average fraction of the TCP half-open connection backlog occupied during the resource release interval. Thus, self-information characterizes statistical abnormality, whereas cumulative occupancy represents persistent resource occupation. The controlled analyses demonstrate the theoretical feasibility of the proposed framework. In particular, self- information provides a unique value for each nonzero event probability, whereas different connection-state probabilities may produce the same or similar entropy measurements. Under the controlled conditions considered in this study, self-information, therefore, provides a more explicit and interpretable representation of TCP connection states. However, these results do not establish quantitative superiority over the approaches of Chen et al. [25], Ali et al. [18], or Sinha et al. [7].

The present study is limited to the theoretical formulation and controlled proof-of-concept analysis of the proposed framework. Accordingly, the reported results demonstrate only its mathematical feasibility and interpretability and should not be interpreted as operational IDS benchmarks. They do not establish detection performance in terms of Accuracy, Precision, Recall, F1-score, FPR, FNR, ROC/AUC, or detection delay under real network traffic. Future work will validate the framework using packet-level traces from labeled public datasets, such as CICDDoS2019 and CAIDA, and deployment-based

experiments in actual server environments. These evaluations will also examine the effects of legitimate traffic bursts, packet loss, retransmissions, parameter settings, and different server workloads.

Author Contributions: N.-I.W. proposed the idea and reviewed the methodology and manuscript. J.-R.S. wrote this paper and discussed the methodology. C.-Y.Y. discussed and reviewed the manuscript. M.-S.H. supervised, discussed, and reviewed the manuscript. All authors read and approved the final manuscript.

Funding: This research was funded by National Science and Technology Council, Taiwan, grant number NSTC 115-2221-E-468-001.

Data Availability Statement: The original contributions presented in this study are included in the article. Further inquiries can be directed to the corresponding author.

Conflicts of Interest: Author Jia-Rong Sun was employed by the company BCCS Co., Ltd. The remaining authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

References

1. Yang, C.-H.; Wu, J.-P.; Lee, F.-Y.; Lin, T.-Y.; Tsai, M.-H. Detection and Mitigation of SYN Flooding Attacks through SYN/ACK Packets and Black/White Lists. *Sensors* **2023**, *23*, 3817. [CrossRef] [PubMed]
2. AlArnaout, Z.; Mostafa, N.; Alabed, S.; Aly, W.H.F.; Shdefat, A. RAPT: A Robust Attack Path Tracing Algorithm to Mitigate SYN-Flood DDoS Cyberattacks. *Sensors* **2023**, *23*, 102. [CrossRef] [PubMed]
3. Liu, J. One-R Random Forest and Fusion Feature Selection for Japanese Data DDoS Attack Detection. *Int. J. Netw. Secur.* **2026**, *28*, 487–494.
4. Balaji, K.; Balachandra, M. Multi-Layer Traffic Analysis Framework for DDoS Attacks in Software-Defined IoT Networks. *Future Internet* **2026**, *18*, 164. [CrossRef]
5. Miri Kelaniki, S.; Komninos, N. Efficient Detection of XSS and DDoS Attacks with Bent Functions. *Information* **2026**, *17*, 80. [CrossRef]
6. Su, Y.; Xiong, D.; Qian, K.; Wang, Y. A Comprehensive Survey of Distributed Denial-of-Service Detection and Mitigation Technologies in Software-Defined Networks. *Electronics* **2024**, *13*, 807. [CrossRef]
7. Sinha, M. SynFloWatch: A Detection System against TCP-SYN Based DDoS Attacks Using Entropy in Hybrid SDN. In Proceedings of the 25th International Conference on Distributed Computing and Networking (ICDCN 2024), Chennai, India, 4–7 January 2024; pp. 359–364. [CrossRef]
8. Cai, Z.; Li, P.; Zhang, J.; Zhu, L.; Ding, L.; Feng, Y. Research on DDoS Attack Detection and Defense Based on SDN. *Int. J. Netw. Secur.* **2026**, *28*, 710–718. [CrossRef] [PubMed]
9. Qureshi, S.S.; He, J.; Zhu, N.; Jia, M.; Qureshi, S.; Ullah, F.; Nazir, A.; Wajahat, A. A New Deep Learning Paradigm for IoT Security: Expanding Beyond Traditional DDoS Detection. *Int. J. Netw. Secur.* **2024**, *26*, 349–360.
10. Angulo, E.; Lizcano, L.; Marquez, J. A Stacking-Based Ensemble Model for Multiclass DDoS Detection Using Shallow and Deep Machine Learning Algorithms. *Appl. Sci.* **2026**, *16*, 578. [CrossRef]
11. Gelgi, M.; Guan, Y.; Arunachala, S.; Samba Siva Rao, M.; Dragoni, N. Systematic Literature Review of IoT Botnet DDoS Attacks and Evaluation of Detection Techniques. *Sensors* **2024**, *24*, 3571. [CrossRef] [PubMed]
12. Qian, J.; Zhang, J.; Li, A.; Xu, L. Research on the Detection and Tracing of Large-Scale Botnets. *Int. J. Netw. Secur.* **2026**, *28*, 719–728.
13. Ganeshan, S.; Ramasamy, R.K. A Systematic Review of Machine-Learning-Based Detection of DDoS Attacks in Software-Defined Networks. *Future Internet* **2026**, *18*, 109. [CrossRef]
14. Song, Y.; Zhang, Z.; Li, B. Federated Reinforcement Learning-Based Intrusion Detection Method in Vehicular Networks. *Int. J. Netw. Secur.* **2026**, *28*, 109–117.
15. Ibrahim, A.J.; Répás, S.R.; Bektaş, N. Feature-Optimized Machine Learning Approaches for Enhanced DDoS Attack Detection and Mitigation. *Computers* **2025**, *14*, 472. [CrossRef]
16. Batool, S.; Aslam, M.; Akpokodje, E.; Jilani, S.F. A Comprehensive Review of DDoS Detection and Mitigation in SDN Environments: Machine Learning, Deep Learning, and Federated Learning Perspectives. *Electronics* **2025**, *14*, 4222. [CrossRef]
17. Kong, X.; Kong, X. Network Intrusion Detection Method Integrating Multi-Scale One-Dimensional Convolutional Neural Network and BiLSTM. *Int. J. Netw. Secur.* **2026**, *28*, 549–556.

18. Ali, B.H.; Sulaiman, N.; Al-Haddad, S.A.R.; Atan, R.; Hassan, S.L.M.; Alghrairi, M. Identification of Distributed Denial-of-Service Anomalies by Using a Combination of Entropy and Sequential Probability Ratio Test Methods. *Sensors* **2021**, *21*, 6453. [CrossRef] [PubMed]
19. Ogunseyi, T.B.; Avoussoukpo, C.B.; Wang, L.; Zhou, X. An Effective Network Intrusion Detection System on Diverse IoT Traffic Datasets: A Hybrid Feature Selection and Extraction Method. *Int. J. Netw. Secur.* **2026**, *28*, 215–226.
20. Xie, X.; Zheng, H.; Yu, S.; Zheng, X. A Network Traffic Classification Method Based on Shapelet Features and Ensemble Learning. *Int. J. Netw. Secur.* **2026**, *28*, 595–604.
21. Estupiñán Cuesta, E.P.; Martínez Quintero, J.C.; Avilés Palma, J.D. DDoS Attacks Detection in SDN Through Network Traffic Feature Selection and Machine Learning Models. *Telecom* **2025**, *6*, 69. [CrossRef]
22. Han, D.; Li, H.; Fu, X.; Zhou, S. Traffic Feature Selection and Distributed Denial-of-Service Attack Detection in Software-Defined Networks Based on Machine Learning. *Sensors* **2024**, *24*, 4344. [CrossRef] [PubMed]
23. Bhattacharya, S.; Nayek, A.K.; Sen, M. Employing Multi-Format Log Analysis to Detect Distributed Denial-of-Service (DDoS) Attacks in Software-Defined Networks. *Discov. Netw.* **2026**, *2*, 6. [CrossRef]
24. Hamad, S.A.; Hasan, S.M. Enhancing Network Security: A Review of Machine Learning Techniques for Detecting TCP SYN Flood Attacks. *AARO Sci. J. Koya Univ.* **2026**, *14*, 86–99. [CrossRef]
25. Chen, H.-C.; Tseng, S.-S.; Mao, C.-H.; Lee, C.-C.; Churniawan, R. An Approach for Detecting Flooding Attack Based on Integrated Entropy Measurement in Email Server. In *Advanced Technologies, Embedded and Multimedia for Human-Centric Computing*; Lecture Notes in Electrical Engineering; Springer: Dordrecht, The Netherlands, 2014; Volume 260, pp. 941–952.
26. Shannon, C.E. A Mathematical Theory of Communication. *Bell Syst. Tech. J.* **1948**, *27*, 379–423. [CrossRef]
27. Liang, L.; Xiang, P.; Huang, H.; Zhang, N.; Qi, P.; Yin, Z.; Zhai, W.; Zhang, D. Sec-GNN-Driven Joint Multidimensional Anti-Eavesdropping Optimization for Secure UAV-Satellite Communications. *IEEE Internet Things J.* **2026**, *13*, 28694–28706. [CrossRef]
28. Zhang, D.; Fan, C.; Li, Z.; Qin, C.; Dan, J. Scale-Spectrum-State Collaborative Fusion Network for Real-Time Extreme Small Object Detection in Intelligent Sports Analytics. *Expert Syst. Appl.* **2026**, *333*, 133831. [CrossRef]
29. Shirsath, V.A.; Chandane, M.M.; Lal, C.; Conti, M. SYNTROPY: TCP SYN DDoS attack detection for software-defined networks based on Rényi entropy. *Comput. Netw.* **2024**, *244*, 110327. [CrossRef]
30. Rabed, A.; Abdulrazzak, F.; Al-Mqdashi, A.; Al-Sanabani, M. SF-DaM: An efficient lightweight approach for detecting and mitigating TCP SYN flooding attacks in SDN. *Cybersecurity* **2026**, *9*, 73. [CrossRef]