Research Square

Preprints are preliminary reports that have not undergone peer review.
They should not be considered conclusive, used to inform clinical practice,
or referenced by the media as validated information.

# TCP-State-Driven Multilayer Heterogeneous Graph for P2P Malicious Traffic Detection

**Minyu Jin**
Huaxin Consulting Design Institute Co., Ltd
**Bin Li**
Huaxin Consulting Design Institute Co., Ltd
**Xun Gong**
Huaxin Consulting Design Institute Co., Ltd
**Xinlong Wu**
wxl.hx@chinaccs.cn

Huaxin Consulting Design Institute Co., Ltd
**Jiachen Guo**
Sichuan University
**Dexian Zhu**
Sichuan University

**Research Article**

**Keywords:** Network traffic, multilayer heterogeneous graph, intrusion detection, peer-to-peer networks, TCP state

**Posted Date:** November 13th, 2026

**DOI:** https://doi.org/10.21203/rs.3.rs-11386616/v1

**License:** This work is licensed under a Creative Commons Attribution 4.0 International License.
Read Full License

**Additional Declarations:** No competing interests reported.

# TCP-State-Driven Multilayer Heterogeneous Graph for P2P Malicious Traffic Detection

Minyu Jin[1], Bin Li[2], Xun Gong[2], Xinlong Wu[3*], Jiachen Guo[4], Dexian Zhu[4]

[1]Technical Development Department, Huaxin Consulting Design Institute Co., Ltd, Hangzhou, China.
[2]Data Intelligence and Security Research Institute, Huaxin Consulting Design Institute Co., Ltd, Hangzhou, China.
[3*]Institute of Cybersecurity and Information Technology, Huaxin Consulting Design Institute Co., Ltd, Hangzhou, China.
[4]School of Cyber Science and Engineering, Sichuan University, Chengdu, China.

*Corresponding author(s). E-mail(s): 2023226245042@stu.scu.edu.cn;

**Abstract**

In decentralized peer-to-peer (P2P) networks, distributed denial-of-service (DDoS) attacks can resemble normal multi-source concurrent communication, making detection challenging. Although DDoS attacks can exploit different transport protocols, TCP-based attacks remain an important threat because they can target connection establishment and state maintenance, exhausting connection-related resources. However, existing detection methods often aggregate packets across communication stages into session-level statistics, overlooking TCP state transitions and making it difficult to distinguish normal interactions from anomalous behavior. To address this issue, this paper proposes a TCP-state-driven multilayer heterogeneous graph method for malicious traffic detection in P2P environments. First, a multilayer heterogeneous graph comprising packet nodes and client and server state nodes is constructed based on the TCP finite state machine. Temporal, state transition, and mapping edges capture relationships between packets and communication states. Second, packets are divided into state-specific subsequences, from which statistical features are extracted to characterize fine-grained communication anomalies. Experiments on the CTU-13, CICIDS2018, and ISCXVPN2016 datasets achieve a maximum F1 score of 99.998%. Representation effectiveness experiments further demonstrate that state-level features distinguish malicious from benign traffic more effectively

than whole-session features. These results demonstrate the method's effectiveness in detecting multiple types of malicious traffic and its potential for fine-grained anomaly detection in distributed environments such as P2P networks.

**Keywords:** Network traffic, multilayer heterogeneous graph, intrusion detection, peer-to-peer networks, TCP state

# 1 Introduction

The rapid expansion of Internet services has made the availability of networked systems increasingly important, while also increasing their exposure to attacks. The 2024 report of the European Union Agency for Cybersecurity (ENISA)[1] shows that the peak traffic volume of a single distributed denial-of-service (DDoS) attack has exceeded 1 Tbps. A related Cloudflare report[2] also indicates that large-scale DDoS activities continue to evolve. Against this background, malicious traffic detection in P2P environments faces additional challenges. During communication, P2P nodes can dynamically switch between the roles of service requester and resource provider as needed, causing normal traffic itself to exhibit multi-source, highly concurrent, and dynamic interaction characteristics that share certain similarities with the distributed communication patterns of DDoS attacks. This similarity implies that, when deploying malicious traffic detection systems in P2P scenarios, relying solely on session-level global statistical features may lead to a higher likelihood of misclassification. Therefore, exploring finer-grained representations of intra-session behavior is of practical relevance for improving the reliability of detection systems in distributed environments such as P2P networks.

DDoS attacks can be launched using various protocols, among which UDP-based flooding and amplification attacks represent major threats. Nevertheless, TCP-based attacks also remain a significant concern, as they can exploit the connection-oriented nature of TCP. For example, SYN flooding attacks exhaust server resources by sending large numbers of connection requests without completing the three-way handshake. Consequently, the resulting traffic typically consists primarily of packets associated with the connection establishment phase [3]. This issue is particularly relevant to P2P environments, where nodes may establish multiple concurrent connections and alternate between client and server roles. Under such conditions, malicious communication may exhibit abnormal connection establishment patterns, state transitions, or connection termination behaviors.

However, existing studies typically characterize the overall communication states of TCP connections at the session level, without adequately capturing state evolution and transition relationships throughout TCP communication, resulting in an incomplete representation of communication behavior. Meanwhile, most methods extract features from packets across the entire session without segmenting and modeling traffic according to different communication states, leading to insufficient representation granularity. These limitations collectively restrict the ability of detection models to identify and interpret complex network threats. Therefore, a traffic representation

method that incorporates communication state information is needed to provide a more comprehensive and fine-grained characterization of malicious TCP traffic.

To address these limitations, we propose a P2P malicious TCP traffic detection method based on a TCP-state-driven multilayer heterogeneous graph. The method first represents the packet sequence of a session and its associated communication states in a graph, aiming to increase sensitivity to state anomalies such as those associated with SYN flooding. Next, the TCP finite-state machine is used to associate packets with state-specific subsequences, from which statistical features are extracted separately to capture behavioral differences within states. Finally, experiments using CICIDS2018 and CTU-13 evaluate binary classification, representation effectiveness, and multiclass detection.

The main contributions are summarized as follows:

(1) This paper proposes a TCP traffic representation method based on a TCP-state-driven multilayer heterogeneous graph structure. The method associates packet sequences with client-side and server-side state sequences and explicitly preserves packet order, state transitions, and packet–state mapping relationships, thereby supplementing the protocol-state semantics missing from conventional session representations and enhancing the detection model's sensitivity to attack traffic involving abnormal communication states.

(2) On this basis, a malicious TCP traffic detection method is designed. Packets are grouped according to communication states, and statistical features are extracted within each state-specific subsequence. This changes the representation from global session aggregation to fine-grained state-level features, enabling the detector to capture behavioral anomalies at individual communication stages.

(3) Experiments are conducted on CTU-13 and CICIDS2018. The detection experiments evaluate compatibility with different classifiers; the representation experiments examine the discriminative value of state-specific statistics; and the multiclass experiments assess recognition of different attack types. An additional cross-task experiment on ISCXVPN2016 evaluates the applicability of the proposed representation to traffic application classification.

## 2 Related Work

Existing research on attack detection in distributed environments has examined traffic representation, detection models, and evaluation datasets. This section reviews commonly used traffic features, detection methods built on these features, and public datasets employed in related studies.

Feature vectors are among the most established approaches to network traffic representation. By extracting attributes of network flows or sessions, they encode communication processes as fixed-dimensional numerical vectors. Depending on the information described, traffic features can be considered from temporal, spatial, and protocol-state perspectives.

Temporal features characterize changes in traffic over time, including packet interarrival times, flow duration, packet arrival rates, and active and idle periods. Their temporal variation can be further summarized using means, variances, maxima, and

minima. Velarde Alvarado et al. [4] extracted flow duration and packet inter-arrival times in a traffic-aggregation framework for generating NIDS datasets. Ullah et al. [5] also used flow duration and packet rates for intrusion detection. Sahin et al. [6] investigated packet inter-arrival time as a principal detection feature across several intrusion detection datasets, demonstrating its utility for low-overhead traffic screening. Idrissi et al. [7] examined how active and idle flow timeouts affect flow-level feature extraction and intrusion detection, showing that aggregation boundaries influence feature quality. Luay et al. [8] further incorporated fine-grained timestamps and inter-arrival-time statistics into NetFlow features and analyzed the contribution of temporal dependencies to intrusion detection. Their findings indicate that flow-level vectors without explicit temporal information may overlook important behavioral changes. These features do not require payload inspection and can reveal anomalous behavior through changes in communication timing.

Spatial features describe differences in traffic volume, direction, and distribution, including packet lengths, flow bytes, packet counts, and forward and backward traffic. Bour et al. [9] constructed traffic features from packet counts and traffic volume for intrusion detection, while CICFlowMeter [10] extracts packet lengths, byte counts, and forward and backward traffic statistics from bidirectional flows. Kiflay et al. [11] combined flow-level statistics with protocol payload information for intrusion detection, illustrating the discriminative value of numerical properties such as packet counts and traffic volume. Amara Korba et al. [12] compared traffic representations and feature sets at the raw-byte, packet, unidirectional-flow, and bidirectional-flow levels, highlighting the influence of feature extraction granularity. Mimi et al. [13] constructed unidirectional and bidirectional traffic features, including forward and backward packet counts and packet-length distributions, for real-time malicious traffic classification. Sudyana et al. [14] compared and integrated flow-level and packet-level detection, noting that flow aggregation is computationally efficient but may discard packet-level information useful for anomaly identification. Together with temporal features, spatial features provide a common means of representing traffic volume and transmission patterns.

Protocol-state features describe protocol semantics and state-related information expressed during communication. Existing studies select protocol-specific fields: Velarde Alvarado et al. [4], for example, counted TCP flags such as PSH and URG; Lee and Jin [15] used TLS versions, cipher suites, and extensions from handshake messages to detect encrypted malicious traffic; and Shah et al. [16] analyzed ICMP attacks using ICMP packet information. Xu et al. [17] combined network feature sets with traffic feature graphs to retain static attributes and internal traffic relationships. Ma et al. [18] considered protocol types, protocol fields, and payload distributions in both flow-level and packet-level feature extraction, using cross-granularity information to support intrusion detection. Such protocol features capture particular aspects of communication behavior and help distinguish different types of communication.

Given a traffic representation, existing attack detection approaches can broadly be divided into rule-based methods, conventional machine learning, and deep learning.

Rule-based approaches identify anomalous traffic by comparing observed features against predefined thresholds, signatures, or attack patterns. Aladaileh et al. [19] combined traffic entropy with dynamic thresholds to detect low-rate DDoS attacks in software-defined networking (SDN) environments. Praseed and Thilagam [20] generated dynamic attack signatures from HTTP request patterns for early detection of application-layer DDoS attacks. Such methods are relatively simple and interpretable, but their dependence on predefined rules or attack patterns limits their adaptability to previously unseen or evolving attacks. Conventional machine-learning approaches classify traffic by learning differences between benign and malicious feature distributions; common models include decision trees, random forests, support vector machines, and k-nearest neighbors. Zeinalpour and Ahmed [21] applied clustering and ensemble methods to DDoS detection, whereas Sahoo et al. [22] compared J48, random forests, support vector machines, and k-nearest neighbors for DDoS detection in SDN. Recent studies have addressed feature dimensionality and sample distributions through feature selection [23] and class-imbalance handling [24], illustrating the dependence of classifier performance on input data quality. Although these methods are relatively economical to compute, their effectiveness relies strongly on the discriminative power of the traffic representation.

Deep-learning methods employ neural networks to learn complex traffic patterns. Doriguzzi-Corin et al. [25] proposed LUCID, a lightweight CNN-based detector for distinguishing benign traffic from DDoS traffic. Ullah et al. [5] applied Transformer-based transfer learning to imbalanced traffic, while other studies combined BERT and MLP architectures [26], hybrid sampling and deep learning [27], or attention with deformable convolution [28] for intrusion detection. HiViT-IDS [29] and a CNN-BiLSTM-Transformer approach [30] further explored visual and sequential modeling, while ADFCNN-BiLSTM [28] focused on jointly extracting spatial and temporal information. For relational modeling, DIMK-GCN [31], EL-GNN [32], and a spatiotemporal graph model [33] explored graph-based representations or continual learning to capture complex network interactions. These methods improve the modeling of nonlinear patterns, but their effectiveness still depends on the communication semantics preserved in the input data.

Public datasets are commonly used to evaluate these approaches. CICIDS2018 contains benign traffic and several attack types, including DoS, DDoS, and brute-force attacks, and provides raw traffic and flow-level features [34]. CTU-13 consists of 13 real-network botnet scenarios covering benign, background, and malicious communications [35]. ISCXVPN2016 contains application traffic collected under VPN and non-VPN conditions, enabling evaluation of traffic representations across different communication scenarios [36]. All three datasets provide raw packet captures, allowing packet- and session-level features to be extracted anew for evaluating the effectiveness and applicability of different traffic representations.

Overall, previous studies have represented malicious traffic using diverse feature dimensions and combined these representations with various detection methods. However, many existing representations operate on entire flows or packet sequences, without explicitly characterizing TCP communication states and the behaviors within each state. To address this limitation, the present study incorporates TCP state

information, establishes associations between packets and communication states, and extracts statistics within individual states to improve the representation of anomalous communication states and behaviors. Moreover, anomaly detection in P2P networks faces additional challenges. Their decentralized architecture increases the cost of traffic monitoring and analysis [34], while the changing roles of P2P nodes across sessions require connection-specific identification of clients and servers [35]. Although graph-based methods have shown promise in detecting P2P botnets [36], maintaining communication graphs in large-scale dynamic networks remains challenging [37]. Our method instead constructs a multilayer heterogeneous graph for each TCP session, avoiding the need for a global communication graph and enabling fine-grained, state-level traffic representation.

# 3 P2P Malicious TCP Traffic Detection Based on a TCP-State-Driven Multilayer Heterogeneous Graph

The proposed detection framework consists of four stages: data preprocessing, multilayer heterogeneous graph construction, feature extraction, and traffic detection, as illustrated in Fig. 1.

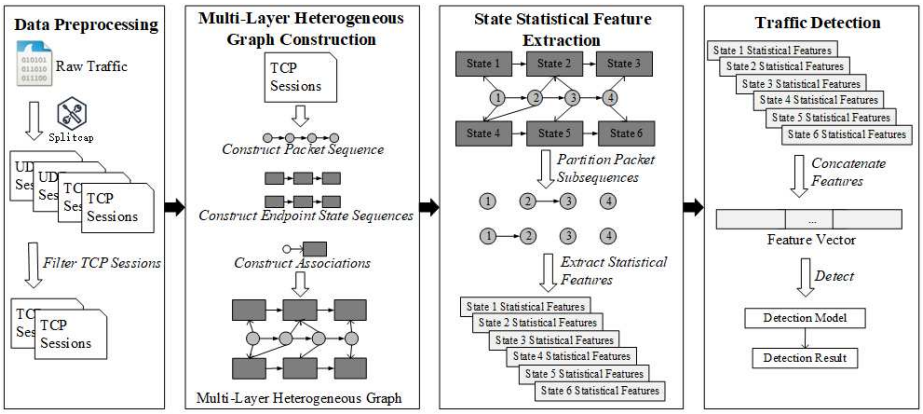



**Fig. 1** Framework for TCP traffic representation and detection using a multilayer heterogeneous graph

In the data preprocessing stage, session segmentation and TCP session filtering are performed on raw PCAP traffic. Tools are employed to split sessions according to the five-tuple and filter out non-TCP traffic and invalid records, thereby yielding clean and complete TCP session PCAP files. This process avoids modeling biases caused by protocol mixing and session truncation.

In the multi-layer heterogeneous graph construction stage, each TCP session is represented as a three-layer structure consisting of a packet layer, a client-side state layer, and a server-side state layer. Temporal edges, state transition edges, and mapping edges are utilized to establish associations between packets and communication states. The resulting multi-layer heterogeneous graph explicitly characterizes the state

evolution throughout the entire session lifecycle, compensating for the state semantic information absent in traditional methods.

In the state feature extraction stage, leveraging the mapping relationships within the multi-layer heterogeneous graph, the packet sequence is partitioned into distinct state subsequences according to TCP communication states. Statistical features across temporal, spatial, and protocol dimensions are independently extracted from each subsequence. This approach mitigates the feature dilution problem inherent in conventional session-level global statistics and preserves the distinctive characteristics of each communication phase.

In the traffic detection stage, the core step involves concatenating the feature vectors from each state to form a unified session-level feature representation, which is subsequently fed into the detection model. Specifically, for each TCP session, the statistical features extracted across the fourteen communication states are concatenated sequentially to construct a high-dimensional feature vector. This vector comprehensively captures the behavioral patterns of the session across different communication phases. Subsequently, the concatenated feature vector is input into a pre-trained detection model, which determines whether the TCP session constitutes malicious traffic and outputs the final detection result.

## 3.1 TCP-state-driven Multilayer Heterogeneous Graph Construction

Based on the fact that node communication roles in P2P networks change dynamically with sessions, this paper builds a multi-layer heterogeneous graph that includes packet nodes as well as client and server state nodes. By explicitly linking packets with the TCP states on both ends, it captures the state changes under different communication roles to better represent abnormal communication attacks. As shown in Fig. 2, construction comprises three steps: generating the packet sequence, generating the endpoint state sequences, and establishing their associations.

**Packet Sequence Construction.** In constructing the packet sequence, the TCP session packets are first sorted according to their arrival times, and then the relevant attributes of each packet are extracted through packet parsing. Assuming a TCP session consists of $N$ packets, the packet sequence can be formally defined as $P = p_1, p_2, \ldots, p_N$, where $p_i$ denotes the $i$-th packet node in the TCP session. Each packet node $p_i$ is characterized by a 10-dimensional feature vector, whose dimensions include TCP flags, packet transmission direction, capture timestamp, transport layer payload size, header length, total packet length, sequence number, acknowledgment number, retransmission flag, and the index of the original packet corresponding to the retransmission.

In peer-to-peer (P2P) networks, each node typically functions simultaneously as a resource requester and a resource provider. These client and server roles are not statically assigned; rather, they are dynamically assumed and relinquished according to the communicating peer and the specific communication session. The same node might act as a client in one TCP session and a server in another. Consequently, role classification based solely on node identity or address fails to accurately characterize node behavior. To address this, we look at each TCP session individually, determine

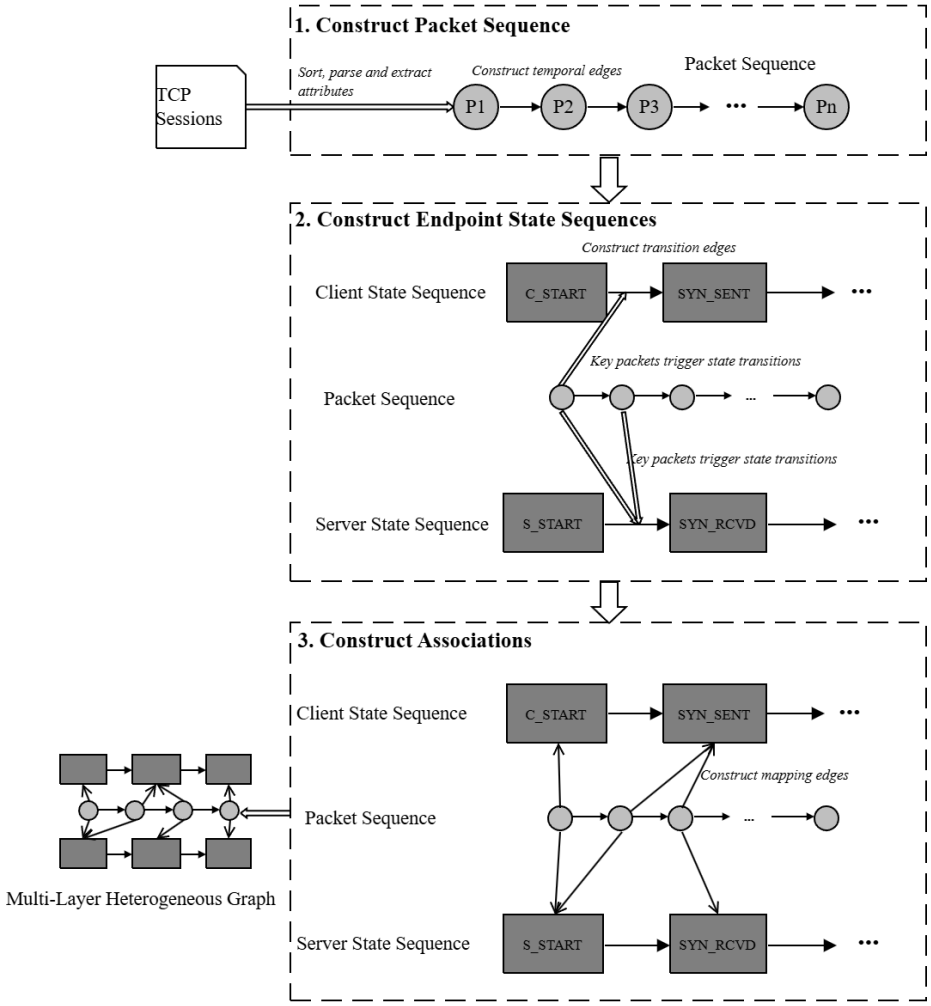



**Fig. 2** Construction of the TCP-state-driven multilayer heterogeneous graph

the roles on both ends during connection setup, and create separate client and server state sequences. On top of that, by mapping packets to the state nodes on both ends, we can clearly track which role and TCP state each packet corresponds to. This way, the multi-layer heterogeneous graph can reflect the dynamic roles and communication states in P2P scenarios, providing a basis for spotting abnormal communication behaviors in different roles and states.

**Endpoint State Sequence Construction.** To construct the state sequences of the client and server endpoints, this paper defines a state set $S$ containing 14 states based on the TCP finite state machine. These states describe the evolution of both communicating endpoints during connection establishment, data transmission, and connection termination. The client and server use C_START and S_START as their initial states, respectively, while C_CLOSED and S_CLOSED indicate connection

termination. State transitions are determined by the TCP flags, packet transmission direction, total packet length, sequence number, and acknowledgment number.

Let the client state sequence be $C = \{c_0, c_1, \ldots, c_M\}$ and the server state sequence be $S_{srv} = \{s_0, s_1, \ldots, s_K\}$. They are formally defined as follows:

$$C = \{c_j \mid c_j \in S,\ j = 0, 1, \ldots, M\} \tag{1}$$

$$S_{srv} = \{s_k \mid s_k \in S,\ k = 0, 1, \ldots, K\} \tag{2}$$

where $M$ and $K$ denote the numbers of state transitions for the client and server, respectively. The initial states $c_0$ and $s_0$ are C_START and S_START, respectively, while $c_j$ and $s_k$ represent the corresponding endpoint states after the $j$-th and $k$-th transitions. State transitions are triggered by packets satisfying specific conditions. The transition conditions and relationships among states are illustrated in Figure 3.

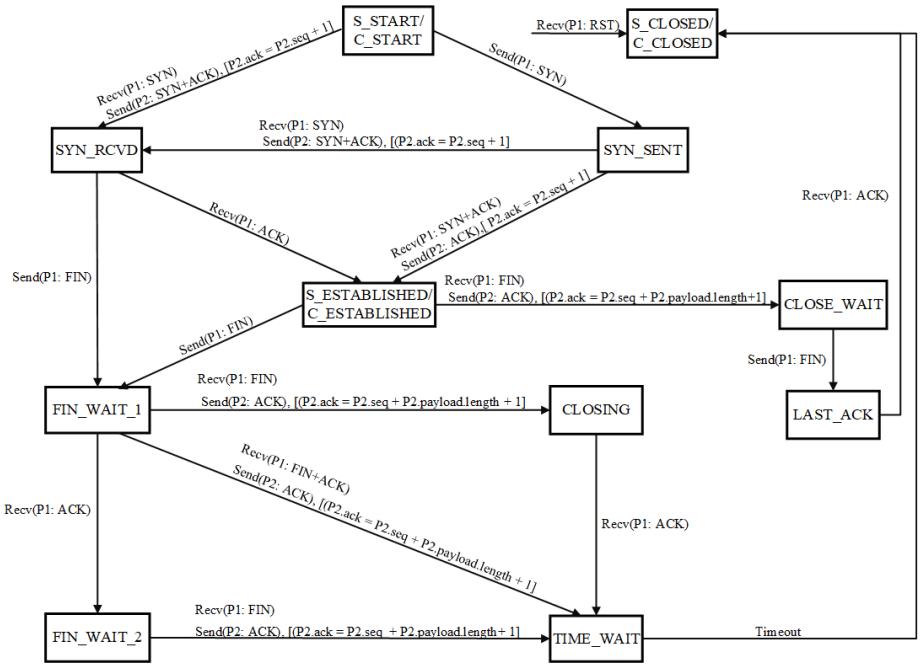



**Fig. 3** TCP state transition conditions

As shown in Figure 3, TCP communication consists of three main phases: connection establishment, data transmission, and connection termination. During connection establishment, the two endpoints begin in the C_START and S_START states. The active opener enters SYN_SENT after sending a SYN packet and waits for a response, while the passive opener enters SYN_RCVD after receiving the SYN and replying with SYN+ACK, awaiting the final acknowledgment. Once the three-way handshake is completed, the client and server enter C_ESTABLISHED and S_ESTABLISHED,

respectively, indicating that the connection has been established and normal data transmission can proceed.

During connection termination, the active closer enters FIN_WAIT_1 after sending a FIN packet and waits for an acknowledgment or a closure request. Upon receiving an ACK, it enters FIN_WAIT_2 and waits for the peer's FIN packet. If both endpoints initiate closure almost simultaneously, the connection may enter CLOSING while awaiting the remaining acknowledgment. After completing the FIN exchange, the active closer enters TIME_WAIT and waits for twice the Maximum Segment Lifetime (2MSL) before reaching the closed state. In contrast, the passive closer enters CLOSE_WAIT after receiving a FIN and sending an ACK, waiting for the application to terminate the connection. It then sends a FIN and enters LAST_ACK, eventually reaching the closed state after receiving the final acknowledgment. C_CLOSED and S_CLOSED represent the terminal states of the client and server, respectively.

Through these state transition rules, the proposed method tracks the state evolution of both endpoints throughout a TCP session and associates packets from different communication phases with their corresponding states, providing a foundation for subsequent fine-grained feature extraction.

**Association Construction**. During association construction, temporal edges between packet nodes, transition edges between state nodes, and mapping edges between packet nodes and state nodes are established based on the packet sequence $P$, client state sequence $C$, and server state sequence $S_{srv}$. These relationships collectively form the multilayer heterogeneous graph $G$. The detailed construction procedure is presented in Algorithm 1.

Algorithm 1 describes the complete association construction process within the multi-layer heterogeneous graph. The algorithm takes the packet sequence $P$, client state sequence $C$, and server state sequence $S_{srv}$ as inputs and outputs the multi-layer heterogeneous graph $G$. The algorithm first iterates through the packet sequence to establish temporal edges $E_t$ between adjacent packet nodes, capturing the chronological order of packet arrivals. It then traverses the client state sequence to construct transition edges $E_c$ between consecutive state nodes, characterizing the client's state evolution, and similarly constructs transition edges $E_s$ for the server state sequence. Finally, based on the state transition condition triggered by each packet, mapping edges are established between the packet node $p_i$ and the corresponding client state node $c_j$ and server state node $s_k$, thereby associating the packet layer with the state layers. Consequently, a multi-layer heterogeneous graph $G$ comprising a packet layer, a client state layer, and a server state layer is constructed. The resulting multi-layer heterogeneous graph generated through these three steps contains two node types—packet nodes and state nodes—and three edge types—temporal edges, transition edges, and mapping edges. Figure 4 provides a visualized illustration of the final multi-layer heterogeneous graph generated from a complete TCP session example.

As depicted in Figure 4, the middle layer of the graph represents the packet layer, where packet nodes are sequentially connected by directed edges according to their arrival times. The upper and lower layers correspond to the client state layer and the server state layer, respectively, with transition edges recording the state evolution process between state nodes. The packet layer and the two endpoint state layers are

**Algorithm 1:** Association Construction Algorithm

**Input:** Packet sequence $P = \{p_1, \ldots, p_N\}$, Client state sequence $C = \{c_0, \ldots, c_M\}$, Server state sequence $S_{srv} = \{s_0, \ldots, s_K\}$

**Output:** Multi-layer heterogeneous graph $G = (V, E)$

Initialize graph $G$ with $V \leftarrow \emptyset$, $E \leftarrow \emptyset$;
Add all packet nodes $p_i \in P$ to $V$;
Add all client state nodes $c_j \in C$ to $V$;
Add all server state nodes $s_k \in S_{srv}$ to $V$;
**for** $i \leftarrow 1$ **to** $N - 1$ **do**
    Add temporal edge $e_{i,i+1} = (p_i, p_{i+1})$ to $E$;
**for** $j \leftarrow 0$ **to** $M - 1$ **do**
    Add transition edge $e_{c_j,c_{j+1}} = (c_j, c_{j+1})$ to $E$;
**for** $k \leftarrow 0$ **to** $K - 1$ **do**
    Add transition edge $e_{s_k,s_{k+1}} = (s_k, s_{k+1})$ to $E$;
**foreach** *packet* $p_i \in P$ **do**
    **for** $j \leftarrow 0$ **to** $M$ **do**
        **if** `IsTrigger`$(p_i, C, j)$ **then**
            Add mapping edge $e_{p_i,c_j} = (p_i, c_j)$ to $E$;
            **break**;
    **for** $k \leftarrow 0$ **to** $K$ **do**
        **if** `IsTrigger`$(p_i, S_{srv}, k)$ **then**
            Add mapping edge $e_{p_i,s_k} = (p_i, s_k)$ to $E$;
            **break**;
**return** $G$;

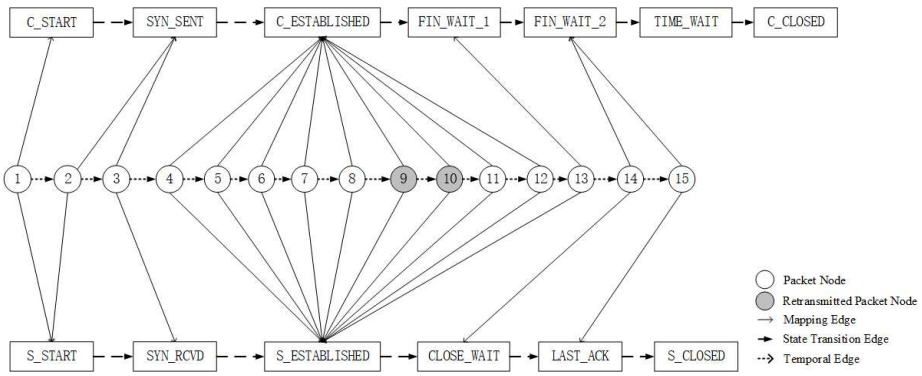



**Fig. 4** Visualization of a TCP-state-driven multilayer heterogeneous graph

interconnected by mapping edges, reflecting the communication states of both endpoints at the moment each packet arrives. Collectively, this multi-layer graph structure

constitutes a representational framework that integrates fundamental packet information, temporal information, and state information, thereby providing a structured vehicle for subsequent feature extraction at the state granularity.

## 3.2 State-Level Feature Extraction

To characterize the behavioral information of a TCP session across different states, this paper designs a state-level feature extraction method. This method partitions the packet sequence into multiple subsequences corresponding to distinct states by leveraging the state nodes and mapping edges within the multi-layer heterogeneous graph. Statistical features are then computed based on the node attributes within each subsequence, yielding finer-grained state-level features. The workflow of state feature extraction is illustrated in Figure 5.

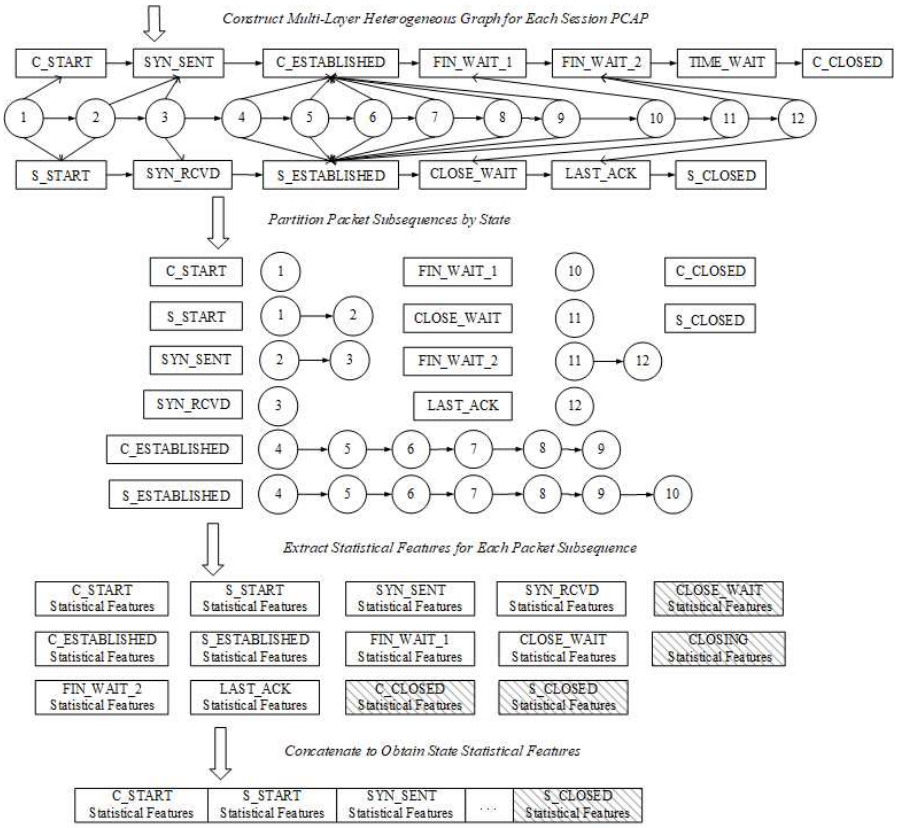



**Fig. 5** State-level feature extraction procedure

Specifically, the partitioning of packet subsequences relies on the mapping edges in the multi-layer heterogeneous graph. Within the multi-layer heterogeneous graph, each packet node $p_i$ is already associated, via mapping edges, with the client state node

$c_j$ and server state node $s_k$ corresponding to its arrival instant. Therefore, during the feature extraction stage, one only needs to iterate through all state nodes and collect all packet nodes connected to a given state node via mapping edges to obtain the packet subsequence corresponding to that state. Let the packet subsequence associated with state $u \in S$ be denoted as $P_u$. All subsequent statistical features are computed over $P_u$. If a state has no associated packets, i.e., $P_u = \emptyset$, all feature values for that state are set to zero, thereby ensuring uniformity in the feature vector dimensionality.

For each state subsequence $P_u$, this paper extracts statistical features across three dimensions: temporal, spatial, and protocol. Temporal dimension features characterize the timing properties of the state phase, including total duration, forward duration, backward duration, and various statistics of time intervals. Spatial dimension features describe the volumetric distribution of packets and payloads within the state phase, encompassing packet count, total payload length, and statistics of packet length and payload size. Protocol dimension features capture the protocol behavioral patterns of the session based on TCP flag statistics, counting the occurrences of flags such as SYN, ACK, FIN, and RST in the total, forward, and backward directions, respectively. These features do not rely on packet payload content or application-layer parsing, but only use TCP side-channel characteristics, which can remain effective in encrypted traffic scenarios and capture transmission behavior in a lightweight manner[37].

The statistical features included for each state are summarized in Table 1.

**Table 1** State-level feature definitions

| Feature | Description | Count | Details |
|---|---|---|---|
| Duration | Duration | 3 | Total/forward/backward |
| Pkt_Count | Packet count | 3 | Total/forward/backward |
| Payload_Sum | Total payload length | 3 | Total/forward/backward |
| Payload | Payload size | 15 | (Total/forward/backward) × (Max/Min/Mean/Var/Std) |
| PktLen | Packet length | 15 | (Total/forward/backward) × (Max/Min/Mean/Var/Std) |
| IAT | Inter-arrival time | 15 | (Total/forward/backward) × (Max/Min/Mean/Var/Std) |
| FLAG | TCP flag count | 18 | (Total/forward/backward) × (SYN/ACK/FIN/RST/PSH/URG) |
| Pkt_Rate | Packet rate | 3 | Total/forward/backward |
| Payload_Rate | Payload rate | 3 | Total/forward/backward |

Compared with traditional session-level feature extraction methods, the proposed approach mitigates the feature dilution problem inherent in session-granularity feature extraction. Conventional methods treat all packets of an entire TCP session as a single entity for uniform aggregation, performing global averaging or aggregation on features that exhibit markedly distinct distributions across different communication state phases. As a result, the discriminative features unique to each phase are diluted, key behavioral patterns are obscured, and the model struggles to capture the true distribution of attacks within specific stages. In contrast, the proposed method decomposes the session into multiple fine-grained sub-phases based on states and independently extracts statistical features per state. This avoids the diluting effect of global

statistics on critical local features, fully preserves the original feature distributions and behavioral disparities of each communication phase, and enhances both feature discriminability and the model's capacity to detect complex and stealthy attacks.

# 4 Experimental Results and Analysis

## 4.1 Datasets and Evaluation Metrics

The experimental data were obtained by CTU-13 and filtering CICIDS2018. Traffic was separated by five-tuple and attack type, and TCP sessions were retained. The resulting sample distributions are reported in Table 2.

**Table 2** Experimental sample counts

| Dataset | Traffic type | Count |
|---|---|---|
| CTU-13 | benign | 3826 |
| | malicious | 368 |
| CICIDS2018 | benign | 2384763 |
| | bot | 91685 |
| | DDoS | 312177 |
| | DoS GoldenEye | 14116 |
| | DoS Hulk | 14116 |
| | DoS Slowhttptest | 14116 |
| | DoS Slowloris | 6979 |
| | FTP-Patator | 14116 |
| | SSH-Patator | 14146 |
| | Web Attack - Brute Force | 260 |
| | Web Attack - SQL Injection | 50 |
| | Web Attack - XSS | 116 |

Four metrics were used: accuracy (ACC), precision (PRE), recall (REC), and F1-score. These are calculated from true positives (TP), false positives (FP), true negatives (TN), and false negatives (FN), as defined in Table 3.

**Table 3** Evaluation metrics

| Metric | Description | Formula |
|---|---|---|
| Accuracy (ACC) | Fraction of correctly predicted samples | $\frac{TP+TN}{TP+TN+FP+FN}$ |
| Precision (PRE) | Fraction of positive predictions that are correct | $\frac{TP}{TP+FP}$ |
| Recall (REC) | Fraction of actual positives predicted as positive | $\frac{TP}{TP+FN}$ |
| F1-score (F1) | Harmonic mean of precision and recall | $\frac{2\,PRE \cdot REC}{PRE+REC}$ |

## 4.2 Effectiveness of the Detection Method

This experiment aims to evaluate the proposed method's ability to distinguish between benign and malicious traffic in distributed environments, thereby assessing its accuracy and reliability in practical detection tasks.

CICIDS2018 and CTU-13 were used to train random forest (RF), k-nearest neighbors (KNN), and deep neural network (DNN) classifiers. Samples in Table 2 were merged into binary classification datasets, with benign traffic labeled negative and attack traffic labeled positive. Ten-fold cross-validation was used to obtain the results in Table 4.

**Table 4** Binary detection performance (%)

| Dataset | Algorithm | ACC | PRE | REC | F1 |
|---|---|---|---|---|---|
| CTU-13 | RF | 99.59 | 97.86 | 97.56 | 97.70 |
| | KNN | 98.31 | 93.02 | 87.54 | 90.00 |
| | DNN | 97.09 | 81.07 | 75.56 | 78.03 |
| CICIDS2018 | RF | 99.996 | 99.998 | 99.997 | 99.998 |
| | KNN | 99.992 | 99.995 | 99.996 | 99.996 |
| | DNN | 99.958 | 99.971 | 99.985 | 99.978 |

As shown in Table 4, the proposed representation achieves high detection scores across the tested classifiers. Differences between classifiers are relatively small on CICIDS2018, whereas CTU-13 exhibits greater variation. These results suggest that state-level features can support multiple classification algorithms rather than requiring a single detector architecture.

## 4.3 Effectiveness of the Traffic Representation

To assess whether state-level features are more effective than whole-session statistics, we compared three representations: whole-session statistical features, non-semantic equal-packet segmentation, and the proposed TCP-state segmentation. This comparison isolates the influence of feature granularity and protocol semantics on benign-versus-malicious classification.

The training data from the preceding effectiveness experiment were used. First, TCP sessions were reconstructed as packet sequences and represented by whole-session statistics. Second, each session was partitioned into subsequences corresponding to 14 TCP states; the same statistical features were extracted from each subsequence and concatenated before RF training. As a non-semantic control, each session was divided into 14 segments containing approximately equal numbers of packets, and features were extracted from these segments. All three representations were evaluated using ACC, PRE, REC, and F1, as shown in Figure 6.

As shown in Figure 6, the proposed TCP state-based representation achieves strong detection performance on both datasets. On CTU-13, which includes P2P-related botnet communication scenarios, complex node interactions increase the difficulty

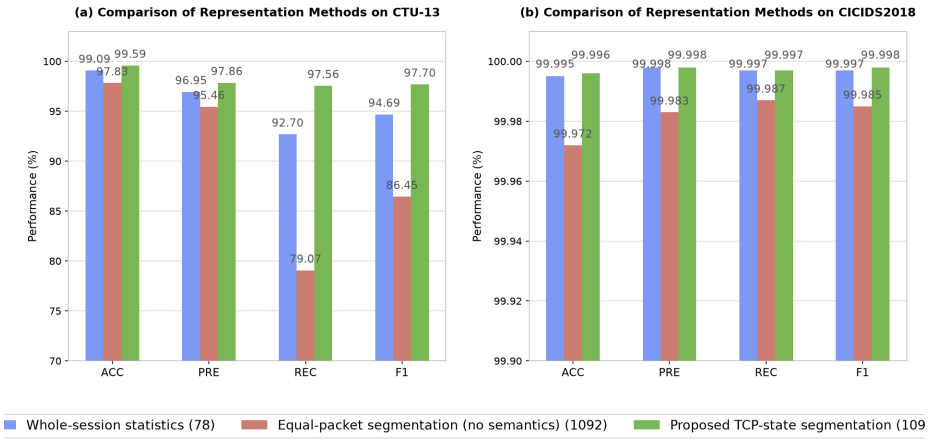



**Fig. 6** Detection performance comparison on CTU-13 and CICIDS2018

of distinguishing malicious from benign traffic. Compared with whole-session representation, our method improves recall and F1 by 4.86 and 3.01 percentage points, respectively, suggesting that TCP state-level features can better preserve discriminative information across communication stages. On CICIDS2018, the differences among representations are relatively small, as whole-session statistical features already provide high detection accuracy. Nevertheless, our method improves the F1 score from 99.997% to 99.998%.

Furthermore, the semantics-agnostic equal-packet segmentation method produces the same 1,092-dimensional feature vectors as the proposed method but achieves a substantially lower F1 score on CTU-13 (86.45% versus 97.70%). This indicates that finer segmentation and increased feature dimensionality alone do not guarantee better detection performance. Instead, aligning statistical features with TCP communication states enables the proposed method to capture more discriminative behavioral patterns, demonstrating the importance of state-aware representation for malicious traffic detection.

## 4.4 Applicability to Multiple Attack Types

To evaluate the applicability of the proposed method to multiple attack types, an RF classifier was employed for multiclass classification on CICIDS2018. TCP sessions were represented using the proposed state-level features, and ten-fold cross-validation was adopted for performance evaluation. The results are presented in Table 5.

**Table 5** Multiclass attack detection performance (%)

| Attack category | ACC | PRE | REC | F1 |
|---|---|---|---|---|
| Bot | 99.9996 | 99.9989 | 99.9989 | 99.9989 |
| DDoS | 100.0000 | 100.0000 | 100.0000 | 100.0000 |
| DoS GoldenEye | 99.9998 | 100.0000 | 99.9929 | 99.9965 |
| DoS Hulk | 100.0000 | 100.0000 | 100.0000 | 100.0000 |
| DoS Slowhttptest | 100.0000 | 100.0000 | 100.0000 | 100.0000 |
| DoS Slowloris | 99.9992 | 99.9427 | 100.0000 | 99.9714 |
| FTP-Patator | 100.0000 | 100.0000 | 100.0000 | 100.0000 |
| SSH-Patator | 100.0000 | 100.0000 | 100.0000 | 100.0000 |
| Web Attack - Brute Force | 99.9983 | 98.4685 | 98.4615 | 98.4577 |
| Web Attack - Sql Injection | 99.9981 | 89.5455 | 94.0000 | 91.5015 |
| Web Attack - XSS | 99.9992 | 99.1304 | 97.3913 | 98.2213 |

As shown in Table 5, the method achieves high detection scores for several attack categories in CICIDS2018. DDoS, DoS Hulk, DoS Slowhttptest, FTP-Patator, and SSH-Patator approach or reach 100% on the reported metrics. High scores are also reported for DoS GoldenEye, DoS Slowloris, and several web attacks. SQL Injection exhibits relatively poor detection performance because its malicious characteristics are mainly reflected in application-layer content, with limited differences in TCP communication states. In contrast, although XSS is also a content-based attack, it may involve additional variations in communication behavior and state characteristics, allowing the proposed method to capture more discriminative features and achieve better detection performance. It should also be noted that the SQL Injection category contains only 50 samples, and its relatively small sample size may lead to greater variability in classification performance. These results indicate that the representation can distinguish multiple attack patterns.

## 4.5 Comparison with Other Detection Methods

To evaluate the performance of the proposed method in malicious traffic detection, Optimized DNN [38], MLP-PSO IDS [39], LightGBM [40], RF-RFE + Hybrid Ensemble [41], and KNN [42] are selected as comparison methods. Existing intrusion detection studies mainly improve detection performance through parameter optimization, feature selection, or classifier ensembles. Specifically, Optimized DNN and MLP-PSO IDS represent deep learning methods based on swarm intelligence optimization, while RF-RFE + Hybrid Ensemble combines feature selection with ensemble learning. LightGBM and KNN serve as representative tree-based and distance-based classifiers, respectively. Unlike these methods, which primarily focus on model optimization or feature selection, our method constructs fine-grained traffic representations based on TCP communication states. These representative methods are selected to provide a reference comparison with existing detection strategies. The performance results of the baseline methods are taken directly from their respective publications, while the proposed method is evaluated using ten-fold cross-validation. The comparison results are presented in Figs. 7 and 8.

As shown in Figs 7 and 8, the proposed method achieves strong detection performance on both CTU-13 and CICIDS2018. On CTU-13, ACC, PRE, and F1 reach

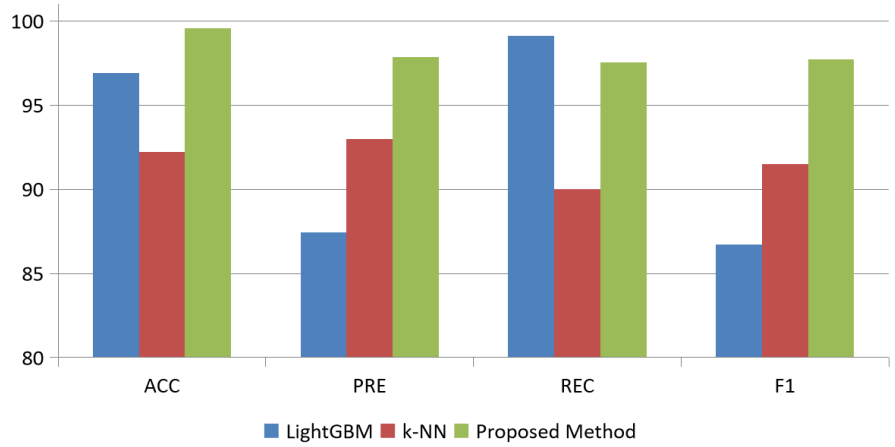



**Fig. 7** Detection performance comparison on CTU-13

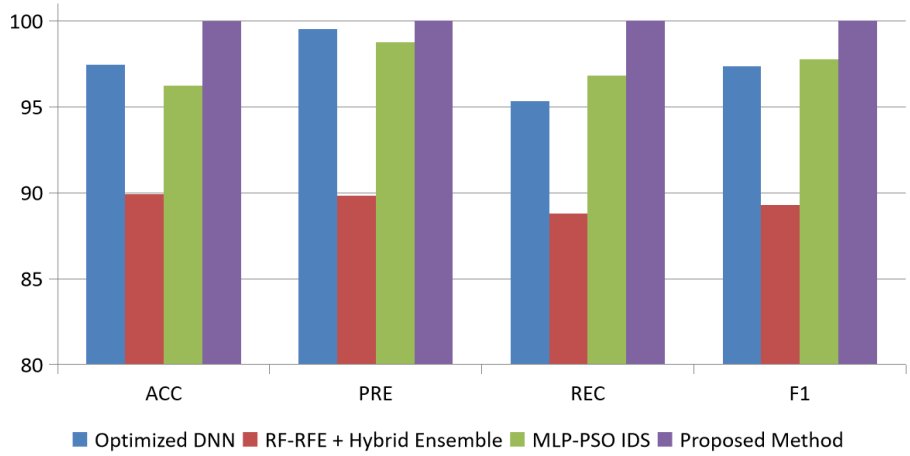



**Fig. 8** Detection performance comparison on CICIDS2018

99.59%, 97.86%, and 97.70%, respectively, with F1 outperforming k-NN by 6.19 percentage points. Although LightGBM achieves a higher recall of 99.14%, its lower precision and F1 indicate more false positives. In contrast, our method achieves a better balance between precision and recall. On CICIDS2018, ACC, PRE, REC, and F1 reach 99.996%, 99.998%, 99.997%, and 99.998%, respectively, with F1 exceeding that of the best-performing baseline, MLP-PSO IDS, by 2.238 percentage points. These results demonstrate that, compared with methods primarily relying on model optimization, feature selection, or ensemble learning, the proposed fine-grained TCP state-based representation more effectively captures communication behavior across different stages, achieving strong malicious traffic detection performance on both datasets.

## 4.6 Cross-Task Evaluation

To further evaluate applicability to different tasks, the proposed method was tested on an application-traffic classification dataset.

In the experiment, we applied the method described in this paper to different types of datasets and completed the traffic application identification task to evaluate how this method performs in network traffic analysis tasks beyond just malicious traffic detection. The experiment used the ISCXVPN2016 dataset for the traffic application classification task, with the categories and amounts shown in Table 6.

**Table 6** Sample categories and counts in the ISCXVPN2016 dataset

| Dataset | Traffic type | Count |
|---|---|---|
| ISCXVPN2016 | Chat | 2058 |
| | Email | 5703 |
| | File Transfer | 1191 |
| | Streaming | 1278 |
| | P2P | 667 |
| | VoIP | 3975 |

Sessions were partitioned into TCP-state subsequences and represented by state-level features. The same training procedure and parameter settings as in the preceding experiments were used, with ten-fold cross-validation. The results are provided in Table 7.

**Table 7** ISCXVPN2016 application classification

| Classifier | ACC | PRE | REC | F1 |
|---|---|---|---|---|
| RF | 91.6 | 91.7 | 91.6 | 91.5 |
| KNN | 87.2 | 87.9 | 87.2 | 87.2 |
| SVM | 88.0 | 88.6 | 88.0 | 88.1 |
| DNN | 79.2 | 89.0 | 79.2 | 70.9 |

The experimental results demonstrate that the proposed method also achieves satisfactory performance in traffic application classification. The key evaluation metrics remain consistently high across different classification algorithms. In particular, Random Forest achieves an accuracy of 91.6%, with strong performance in terms of accuracy, precision, recall, and F1-score. These findings indicate that the proposed method is effective not only for malicious traffic detection but also for traffic application classification, demonstrating its adaptability and generalization capability across different tasks. Furthermore, the results suggest that state-level feature modeling can capture network traffic behavior in greater detail, thereby providing more effective feature representations for various traffic analysis tasks. Therefore, the proposed method

shows promising applicability and potential for broader adoption in diverse network traffic analysis scenarios.

# 5 Conclusion

This paper presented a P2P malicious TCP traffic detection method based on a TCP-state-driven multilayer heterogeneous graph. To address the limited representation of protocol states and the coarse granularity of conventional features, the method combines packet attributes with client- and server-side communication states and extracts statistics from state-specific packet subsequences. Experiments were conducted to evaluate the resulting representation and its detection performance.

However, the approach has certain limitations. Although evaluated on general intrusion detection datasets, the proposed method has not been validated in real-world P2P environments. The effects of dynamic node roles and the computational overhead of distributed graph construction require further investigation. Nevertheless, its session-based processing and linear-time feature extraction offer potential for lightweight deployment on P2P nodes. Future work will focus on real-world P2P validation and distributed detection architectures.

# Statements and Declarations

**Funding:** No specific funding was received for this work.
**Competing interests:** The authors declare no competing interests.
**Data availability:** The datasets analyzed in this study are publicly available. Relevant dataset sources are provided in the manuscript.
**Code availability:** The code is available from the corresponding author upon reasonable request.
**Author contributions:** Conceptualization, Methodology, Software, Formal analysis, Investigation, Validation, Writing – original draft, Visualization: Minyu Jin. Supervision, Project administration, Methodology, Research design, Validation, Writing – review & editing: Xinlong Wu. Data curation, Experiment execution, Result analysis: Bin Li, Xun Gong. Data preprocessing, Experiment execution, Partial visualization: Jiachen Guo, Dexian Zhu. All authors read and approved the final manuscript.
**Ethics approval:** Not applicable.
**Consent to participate:** Not applicable.
**Consent for publication:** Not applicable.

# References

[1] European Union Agency for Cybersecurity (ENISA): 2024 Report on the State of the Cybersecurity in the Union. ENISA (2024).https://www.enisa.europa.eu/publications/2024-report-on-the-state-of-the-cybersecurity-in-the-union. Accessed 28 January 2026

[2] Cloudflare: Cloudflare's 2025 Q3 DDoS threat report: including Aisuru, the apex of botnets. Cloudflare (2025). https://blog.cloudflare.com/ddos-threat-report-2025-q3/. Accessed 28 January 2026

[3] Eddy W (2007) TCP SYN Flooding Attacks and Common Mitigations. RFC 4987. https://doi.org/10.17487/RFC4987

[4] Velarde Alvarado, P., Gonzalez, H., Martínez-Peláez, R., et al.: A novel framework for generating personalized network datasets for NIDS based on traffic aggregation. Sensors 22(5), 1847 (2022).

[5] Ullah, F., et al.: IDS-INT: intrusion detection system using transformer-based transfer learning for imbalanced network traffic. Digital Communications and Networks 10(1), 190–204 (2024).

[6] Sahin, O., Uludag, S.: Leveraging inter-arrival time for efficient threat filtering: a parsimonious approach. Computers & Security 154, 104471 (2025). https://doi.org/10.1016/j.cose.2025.104471

[7] Idrissi, M.J., Alami, H., El Mahdaouy, A., et al.: Flow timeout matters: investigating the impact of active and idle timeouts on the performance of machine learning models in detecting security threats. Future Generation Computer Systems 166, 107641 (2025). https://doi.org/10.1016/j.future.2024.107641

[8] Luay, M., Layeghy, S., Noorbin, N., Sarhan, M., Kulatilleke, G., Moustafa, N., Portmann, M.: Time matters: temporal NetFlow features for ML-based network intrusion detection. IEEE Access 14, 66899–66913 (2026). https://doi.org/10.1109/ACCESS.2026.3688204

[9] Bour, H., Abolhasan, M., Jafarizadeh, S., et al.: A multi-layered intrusion detection system for software defined networking. Computers and Electrical Engineering 101, 108042 (2022).

[10] ahlashkari: CICFlowMeter. GitHub. https://github.com/ahlashkari/CICFlowMeter.

[11] Kiflay, A.Z., et al.: Network intrusion detection leveraging multimodal features. Array 22, 100349 (2024). https://doi.org/10.1016/j.array.2024.100349

[12] Amara Korba, A., Diaf, A., Bouchiha, M.A., et al.: Mitigating IoT botnet attacks: an early-stage explainable network-based anomaly detection approach. Computer Communications 241, 108270 (2025). https://doi.org/10.1016/j.comcom.2025.108270

[13] Mimi, M.A., Ng, H., Yap, T.T.V.: Directional features and rule-based labeling for real-time network traffic-based Android spyware classification. The Journal of Supercomputing 81, 932 (2025). https://doi.org/10.1007/s11227-025-07391-8

[14] Sudyana, D., Yudha, F., Lin, Y.-D., Lai, C.-H., Lin, P.-C., Hwang, R.-H.: From flow to packet: a unified machine learning approach for advanced intrusion detection. Security and Communication Networks 2025, 5729035 (2025). https://doi.org/10.1155/sec/5729035

[15] Lee, W., Jin, S.: Encrypted malware traffic detection using TLS features and random forest. In: Computational and Experimental Simulations in Engineering, pp. 85–100 (2021).

[16] Ali Shah, S.Q., Zeeshan Khan, F., Ahmad, M.: The impact and mitigation of ICMP based economic denial of sustainability attack in cloud computing environment using software defined network. Computer Networks 187, 107825 (2021).

[17] Xu, C., Zhan, Y., Wang, Z., et al.: Multimodal fusion based few-shot network intrusion detection system. Scientific Reports 15, 21986 (2025). https://doi.org/10.1038/s41598-025-05217-4

[18] Ma, Z., Liu, Y., Chen, Y., Liu, Z., Li, Y.: XMF-GNN: a cross-modality dynamic fusion heterogeneous graph neural network for network intrusion detection. Neurocomputing 655, 131285 (2025). https://doi.org/10.1016/j.neucom.2025.131285

[19] Aladaileh, M.A., Anbar, M., Hasbullah, I.H., et al.: Dynamic threshold-based approach to detect low-rate DDoS attacks on software-defined networking controller. Computers, Materials & Continua 73(1), 1403–1416 (2022). https://doi.org/10.32604/cmc.2022.029369

[20] Praseed, A., Thilagam, P.S.: HTTP request pattern based signatures for early application layer DDoS detection: a firewall agnostic approach. Journal of Information Security and Applications 65, 103090 (2022). https://doi.org/10.1016/j.jisa.2021.103090

[21] Zeinalpour, A., Ahmed, H.A.: Addressing the effectiveness of DDoS-attack detection methods based on the clustering method using an ensemble method. Electronics 11, 2736 (2022).

[22] Sahoo, K.S., Puthal, D., Tiwary, M., et al.: DDoS attacks detection and mitigation in SDN using machine learning. In: 2019 IEEE World Congress on Services (SERVICES). IEEE (2019). https://doi.org/10.1109/SERVICES.2019.00051

[23] Almohaimeed, M., Albalwy, F.: Enhancing IoT network security using feature selection for intrusion detection systems. Applied Sciences 14, 11966 (2024). https://doi.org/10.3390/app142411966

[24] Shanmugam, V., Razavi-Far, R., Hallaji, E.: Addressing class imbalance in intrusion detection: a comprehensive evaluation of machine learning approaches. Electronics 14, 69 (2025). https://doi.org/10.3390/electronics14010069

[25] Doriguzzi-Corin, R., Millar, S., Scott-Hayward, S., Martinez-del Rincon, J., Siracusa, D.: LUCID: a practical, lightweight deep learning solution for DDoS attack detection. IEEE Transactions on Network and Service Management 17(2), 876–889 (2020).

[26] Ali, Z., Tiberti, W., Marotta, A., Cassioli, D.: Empowering network security: BERT transformer learning approach and MLP for intrusion detection in imbalanced network traffic. IEEE Access **12**, 137618–137633 (2024). https://doi.org/10.1109/ACCESS.2024.3465045

[27] Guo, D., Xie, Y.: Research on network intrusion detection model based on hybrid sampling and deep learning. Sensors 25, 1578 (2025). https://doi.org/10.3390/s25051578

[28] Li, B., Li, J., Jia, M.: ADFCNN-BiLSTM: a deep neural network based on attention and deformable convolution for network intrusion detection. Sensors 25, 1382 (2025). https://doi.org/10.3390/s25051382

[29] Zhou, H., Zou, H., Li, W., Li, D., Kuang, Y.: HiViT-IDS: an efficient network intrusion detection method based on vision transformer. Sensors **25**, 1752 (2025). https://doi.org/10.3390/s25061752

[30] Zhang, C., Li, J., Wang, N., Zhang, D.: Research on intrusion detection method based on transformer and CNN-BiLSTM in Internet of Things. Sensors **25**, 2725 (2025). https://doi.org/10.3390/s25092725

[31] Han, Z., et al.: DIMK-GCN: a dynamic interactive multi-channel graph convolutional network model for intrusion detection. Electronics 14, 1391 (2025). https://doi.org/10.3390/electronics14071391

[32] Nguyen, T.T., Park, M.: EL-GNN: a continual-learning-based graph neural network for task-incremental intrusion detection systems. Electronics 14, 2756 (2025). https://doi.org/10.3390/electronics14142756

[33] Liu, J., Wang, J., Yan, T., Qi, F., Chen, G.: DLG-IDS: dynamic graph and LLM-semantic enhanced spatiotemporal GNN for lightweight intrusion detection in industrial control systems. Electronics **14**, 3952 (2025). https://doi.org/10.3390/electronics14193952

[34] Canadian Institute for Cybersecurity: CSE-CIC-IDS2018 on AWS. (2018). https://www.unb.ca/cic/datasets/ids-2018.html (accessed 12 October 2026).

[35] Garcia, S., Grill, M., Stiborek, J., et al.: An empirical comparison of botnet detection methods. Computers & Security 45, 100–123 (2014). https://doi.org/10.1016/j.cose.2014.05.011

[36] Draper-Gil, G., Lashkari, A.H., Mamun, M.S.I., et al.: Characterization of encrypted and VPN traffic using time-related features. In: Proceedings of the 2nd International Conference on Information Systems Security and Privacy, pp. 407–414. SciTePress (2016). https://doi.org/10.5220/0005740704070414

[37] Stergiopoulos, G., Chronopoulou, G., Bitsikas, E., Tsalis, N., Gritzalis, D.: Using side channel TCP features for real-time detection of malware connections. J. Comput. Secur. 27(5), 507–520 (2019). https://doi.org/10.3233/JCS-191286

[38] Elmasry, W., Akbulut, A., Zaim, A.H.: Evolving deep learning architectures for network intrusion detection using a double PSO metaheuristic. Comput. Netw. 168, 107042 (2020). https://doi.org/10.1016/j.comnet.2019.107042

[39] Alzughaibi, S., El Khediri, S.: A cloud intrusion detection systems based on DNN using backpropagation and PSO on the CSE-CIC-IDS2018 dataset. Appl. Sci. 13(4), 2276 (2023). https://doi.org/10.3390/app13042276

[40] Ke, G., Meng, Q., Finley, T., Wang, T., Chen, W., Ma, W., Ye, Q., Liu, T.Y.: LightGBM: A highly efficient gradient boosting decision tree. In: Advances in Neural Information Processing Systems, vol. 30, pp. 3146–3154 (2017).

[41] Abbas, Q., Hina, S., Sajjad, H., Zaidi, K.S., Akbar, R.: Optimization of predictive performance of intrusion detection system using hybrid ensemble model for secure systems. PeerJ Comput. Sci. 9, e1552 (2023). https://doi.org/10.7717/peerj-cs.1552

[42] Cover, T.M., Hart, P.E.: Nearest neighbor pattern classification. IEEE Trans. Inf. Theory 13(1), 21–27 (1967). https://doi.org/10.1109/TIT.1967.1053964