DEEP PACKET INSPECTION METHODS FOR NETWORK INTRUSION DETECTION AND APPLICATION CLASSIFICATION

by

Çağatay Ateş

B.S., Electrical & Electronics Engineering, Boğaziçi University, 2019

Submitted to the Institute for Graduate Studies in
Science and Engineering in partial fulfillment of
the requirements for the degree of
Master of Science

Graduate Program in Graduate Program in Electrical & Electronics Engineering

Boğaziçi University

2022

ACKNOWLEDGEMENTS

I would like to thank my thesis advisors Prof. Emin Anarım and Prof. Mutlu Koca, for their patience, contributions, and valuable guidance during this thesis. I am very grateful for their endless support and always being ready to help with every obstacle I encountered during this journey.

In addition, I would like to thank Prof. Hakan Delic, Prof. Fatih Alagöz, and Assoc. Prof. Şerif Bahtiyar for being on my thesis committee and their time.

My special thanks should be given to my friends and colleagues, Pelin Damla Ateş, Süleyman Özdel and Metehan Yıldırım, for their support all through this thesis. I appreciate their friendship and assistance.

Finally, and very importantly, I wish to thank my mother, Demet Ateş, my father, Mustafa Ateş, and my sister Elif Ateş, for their support throughout my life.

This work is supported by the Boğaziçi University Scientific Research Projects under the Spectral and Entropy Approaches in Packet Classification, 18281.

ABSTRACT

DEEP PACKET INSPECTION METHODS FOR NETWORK INTRUSION DETECTION AND APPLICATION CLASSIFICATION

Deep packet inspection methods have become more sophisticated with the rapidly developing technology. To understand the condition of the network, many different packet inspection techniques have been evolved. Newly developing machine learning methods have been used recently on these systems. The aim is to know which type of traffic is running through the network. In this thesis, different deep packet inspection methods are proposed to detect malicious traffic and find the applications running on the network. Time–series and flow–based methods are proposed to accomplish these tasks. Novel feature sets are constructed to execute these methods. Greedy algorithm which finds an upper bound for the distance between the probability distributions with different sizes is utilized in feature extraction process. The extracted features can be divided into two categories which are statistical features and payload–based features. Packet header values such as IP addresses are used to derive statistical features. Also, payload portion of packets are used to extract novel payload–based features. The feature sets are used with decision tree models in supervised learning to execute detection procedures. Proposed approaches are used in network intrusion detection and network application classification tasks. For network intrusion detection, performance evaluation is given by using different publicly available well–known intrusion detection data sets consisting of different types of attacks. For network application classification, a data set consisting of real–world network traces from popular applications is used. Simulation results show that the proposed flow–based approaches have good performance in fulfilling these tasks.

ÖZET

AĞ SALDIRI TESPİTİ VE UYGULAMA SINIFLANDIRMASI İÇİN DERİN PAKET İNCELEME YÖNTEMLERİ

Derin paket inceleme yöntemleri, hızla gelişen teknoloji ile daha sofistike hale geldi. Ağın durumunu anlamak için birçok farklı paket inceleme tekniği geliştirilmektedir. Bu sistemlerde son zamanlarda yeni gelişen makine öğrenmesi yöntemleri kullanılmaya başlanmıştır. Amaç, ağ üzerinde hangi tür trafiğin olduğunu bilmektir. Bu tezde, kötü niyetli trafiği tespit etmek ve ağ üzerinde çalışan uygulamaları bulmak için farklı derin paket inceleme yöntemleri önerilmiştir. Bu görevleri yerine getirmek için zaman serileri ve akış tabanlı yöntemler önerilmiştir. Bu yöntemleri uygulamak için yeni öznitelik kümeleri oluşturulur. Öznitelik çıkarma işleminde, farklı büyüklükteki olasılık dağılımları arasındaki uzaklık için bir üst sınır bulan açgözlü algoritma kullanılmaktadır. Çıkarılan öznitelikler, istatistiksel öznitelikler ve yüke dayalı öznitelikler olmak üzere iki kategoriye ayrılabilir. IP adresleri gibi paket başlık değerleri istatistiksel özellikleri elde etmek için kullanılır. Ayrıca, yeni yük tabanlı özellikleri çıkarmak için paketlerin yük kısmı kullanılır. Öznitelik kümeleri, algılama prosedürlerini yürütmek için denetimli öğrenmede karar ağacı modelleriyle birlikte kullanılır. Önerilen yaklaşımlar, ağ saldırı tespiti ve ağ uygulaması sınıflandırma görevlerinde kullanılır. Ağ saldırı tespiti için, farklı saldırı türlerinden oluşan, halka açık farklı ve iyi bilinen saldırı tespit veri setleri kullanılarak performans değerlendirmesi yapılır. Ağ uygulamaları sınıflandırması için popüler uygulamalardan gerçek ağ izlerinden oluşan bir veri seti kullanılmıştır. Simülasyon sonuçları, önerilen akışa dayalı yaklaşımların bu görevleri yerine getirmede iyi bir performansa sahip olduğunu göstermektedir.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ACRONYMS/ABBREVIATIONS

ACK	Acknowledgement
ANN	Artificial Neural Network
CLDAP	Connection–less Lightweight Directory Access Protocol
CNN	Convolutional Neural Network
DDoS	Distributed Denial of Service
DNS	Domain Name System
DoS	Denial of Service
DPI	Deep Packet Inspection
FTP	File Transfer Protocol
GRU	Gated Recurrent Unit
HTTP	Hyper–text Transfer Protocol
IANA	Internet Assigned Number Authority
ICMP	Internet Control Message Protocol
IDS	Intrusion Detection System
IoT	Internet of Things
IPS	Intrusion Prevention System
ISP	Internet Service Provider
KL	Kullback–Leibler
kNN	k Nearest Neighbor
LAN	Local Area Network
LLC	Logical Link Control
LSTM	Long Short–Time Memory
MAC	Media Access Control
MLP	Multilayer Perceptron
MPI	Medium Packet Inspection
NLP	Natural Language Processing
NTP	Network Time Protocol
OSI	Open Systems Interconnection

P2P	Peer–to–peer
PIN	Personal Identification Number
POP3	Post Office Protocol 3
RNN	Recurrent Neural Network
SDN	Software Defined Network
SMTP	Simple Mail Transfer Protocol
SOM	Self Organizing Map
SPI	Shallow Packet Inspection
SSH	Secure Shell
SVM	Support Vector Machine
SYN	Synchronize
TCP	Transmission Control Protocol
UDP	User Datagram Protocol
VPN	Virtual Private Network
XMAS	Christmas Tree Type Scan

1. INTRODUCTION

Deep Packet Inspection (DPI) is the process of analyzing the network traffic by using different techniques [1]. The aim is to stabilize the health of the network by protecting it against the anomalous activities. It can be considered as a filtering of the network packets with a sophisticated manner. The network packets are checked to be redirected to the desired destination. Any packets not complying with the set of rules defined before are filtered out from the network to keep its robustness stable. This process is executed by examining the contents of network packets. The details of DPI systems with the other packet inspection systems are given in the next section.

1.1. Deep Packet Inspection (DPI) Systems

DPI can be regarded as the third method of network packet inspection techniques which are Shallow Packet Inspection (SPI), Medium Packet Inspection (MPI) and DPI [2]. SPI is the most basic packet inspection technique where only header information of network packets are controlled. Network packets mainly have two parts which are header information and payload portion. Header information constitutes source and destination IP addresses, source and destination port numbers, protocol information etc. Payload is the data that is sent from the source to destination. In most applications, payload is encrypted to ensure the safety during the communication. In SPI techniques, the header part of network packets are examined. It can be regarded as a static analysis of network packets. The aim is to direct the network packets to the desired location by checking their header information. The content of the packets, payload, is not checked. Therefore, this method is regarded as the weakest packet inspection method. In MPI, besides checking the header information of network packets, some portions of payload are analyzed. It can be regarded as a better version of SPI. However, MPI is also a static method in which header information and payload content are checked within some predefined rules. Therefore, the most sophisticated packet inspection method is DPI.

Table 1.1. 7 Layers of the OSI Model.

Layer Name	**Properties**
Application	Allow access to network resources
Presentation	Translates, encrypts and compresses data
Session	Establishes, manages and terminates session, API, sockets, etc.
Transport	Provides reliable processes for message and error delivery
Network	Moves packets from source to destination to provide internetworking.
Data Link	Organizes bits into frames and provides hop-to-hop delivery.
Physical	Transmits bits over a medium, provides mechanical and electrical specifications like coax, wireless, hubs etc.

Apart from statistical analysis, machine learning and deep learning algorithms implemented in DPI methods makes DPI more complex and effective. Before analyzing the DPI methods, the Open Systems Interconnection (OSI) model is described to show which packet inspection method is used in which layer of the communication.

The OSI model consists of seven layers describing any communication occurring through the network [3]. The aim is to define how the communication between two sides will be. The OSI model does not differ according to any type of hardware or computer network. The usage of OSI model is to visualize the communication network and isolate the problems occurring during communication. Seven layers of the OSI model are shown in Table 1.1. These layers can be briefly explained as follows:

- *Application Layer:* Application layer is the closest layer to the user. It is responsible for file transfers, mail and other network software services. Some protocols such as File Transfer Protocol (FTP), Domain Name System (DNS) and Hyper–

Text Transfer Protocol (HTTP) are run on this layer. It provides that programs on various computer systems and networks can efficiently communicate with one another.

- *Presentation Layer:* The presentation layer prepares data into understandable format. Encryption and decryption processes on data are executed on this layer. It sets the data to be displayed on the computer screen.
- *Session Layer:* The main duty of the session layer is to construct communication channels called sessions between two sides. These sessions should be open during data transmission and should be closed after the communication ends. This layer is responsible for this process. It also creates checkpoints to resume the transmission if any error occurs.
- *Transport Layer:* Transport layer takes data from Session layer and creates segments using it. This layer is responsible for controlling flows, regulating transmission speed, and controlling errors. Transmission Control Protocol (TCP) and User Datagram Protocol (UDP) are run on this layer.
- *Network Layer:* In this layer, data is transmitted as network packets. Switching and routing algorithms are executed in this layer. This layer uses IP addresses to direct the packets to the destination.
- *Data Link Layer:* The data link layer starts and terminates the connection between two sides of communication. In this layer, network packets are fragmented into the frames. Data is taken from network layer and sent to physical layer. There are two main parts of this layer which are Logical Link Control (LLC) and Media Access Control (MAC). LLC is responsible for frame robustness and error checking. MAC addresses are used to connect the communication sides.
- *Physical Layer:* Physical layer is the hardware part of the communication. It defines the type of connection which may be wireless or cable. The cables, hubs, and repeaters work on this layer.

To relate the packet inspection methods with OSI model layers, the weakest method, SPI, is carried out in data link and physical layers. MPI is more advanced than SPI and is carried out in transport, network, data link and physical layers. DPI

is better than the previous packet inspection techniques in terms of complexity and performance accuracy. It uses all layers in the OSI model especially the application layer. It provides exhaustive analysis on network packets using their payload portions.

DPI is used for the management of network and protect network from malicious activities such as viruses, worms, data leaks and other kind of anomalous traffic [4]. Packet inspection is executed using the header information and payload portion of network packets. Also, within the development of new technology, machine learning and deep learning methods are implemented in DPI systems. DPI techniques can be broadly classified into three groups which are pattern or signature matching, protocol anomaly and Intrusion Detection Systems (IDS) or Intrusion Prevention Systems (IPS) solutions. Each group can be explained as follows:

- *Pattern or signature matching:* In this method, the network traffic is controlled against a known network attacks. Anomaly detection is executed using the pre–issued signatures. The drawback of this approach is that zero–day attacks cannot be found using the old signatures.
- *Protocol anomaly:* In this method, the network traffic is controlled using the protocol definitions. These definitions decide which type of network traffic should be allowed. Compared to signature matching method, it can offer detection against unknown attacks.
- *IDS/IPS solutions:* The main difference between IDS and IPS is that, IDS is about to generate alerts during malicious traffic while IPS take action to prevent this kind of traffic. In these solutions, packet examination is carried out using header information and payload content of network traffic. Complex algorithms such as machine learning methods can be implemented using these systems. They have a widespread usage compared to signature matching and protocol anomaly methods.

There are also some challenges associated with DPI systems. By adding complex algorithms such as machine learning methods, the DPI system can slow down the

network. Increasing the complexity of inspection algorithms may increase the reaction time of the systems. Also, these systems may cause other security systems such as firewalls to become harder to manage. Apart from these issues, DPI systems are well used and studied in the network security field. In this work, a DPI system aiming at the detection of malicious traffic and classification of the network traffic is proposed. The contributions of this work is given in the next section.

1.2. Thesis Contribution

In this work, a DPI system is proposed against network attacks. This system is also able to classify the network traffic into the applications. The main contributions of the thesis can be stated as follows:

- To implement the DPI techniques, both packet–based and flow–based approaches are used.
- Information theory–based features are proposed in the classification of network traffic. Among them, the usage of Greedy distance given in [5] on network systems is a novel perspective of the proposed system. This distance is used in both packet–based and flow–based feature extraction procedures.
- A novel clustering algorithm given in [6] is implemented in the detection of different type of network attacks. This algorithm requires no parameters to execute compared to other clustering algorithms. This is the main advantage of it.
- Novel payload–based features using the Greedy distance are extracted. It is shown that these features are helpful in the classification of network traffic.
- The proposed system is evaluated with the popular publicly available datasets such as ISCXIDS2012 given in [7] and CIC-IDS2017 given in [8].

1.3. Thesis Organization

The rest of the thesis is organized as follows. In Chapter 2, information about different network attacks types is given. Also, literature review about network intrusion detection systems and network application classification systems is given. In Chapter 3, the theoretical background about the information theory metrics used in the proposed systems is given. In Chapter 4, extracted features are explained in two categories which are packet–based features and flow–based features. In Chapter 5, the general architectures of proposed approaches are given. Also, simulation results are shown with the explanation of data set used. Finally, Chapter 6 concludes the thesis by summarizing the results and giving future work plans.

2. NETWORK INTRUSION DETECTION AND APPLICATION CLASSIFICATION SYSTEMS

In this work, new DPI techniques are implemented for network anomaly detection and application classification. For network anomaly detection, detection of different kind of attacks is implemented with the proposed system. The most common attacks are Denial of Service (DoS) attacks, Distributed Denial of Service (DDoS) attacks, port scan attacks and brute force attacks. Each type of attack has also different types of execution styles. As an example, flood attacks and protocol exploit attacks are two types of DDoS attacks. For the network application classification, flows are labelled in which application they belong to using the proposed system. In this chapter, common attacks evaluated in this work are briefly explained. Then, related work about the detection of these attacks is given. In the second part of this chapter, information about the network application classification with the related work is given.

2.1. Network Intrusion Detection

Network intrusion detection is the process of monitoring the network traffic to detect the malicious activities [9]. Network security systems can be divided into two categories which are IDS and IPS. In this work, proposed system is regarded as an IDS. The aim of IDS systems is to protect the network by detecting suspicious activities and give alerts when necessary. If an action is taken, this system can be regarded as intrusion prevention system. Intrusion detection systems are responsible for generating alerts and reports such as log files to the control center of the network. These systems can be classified into two categories which are explained below:

- *Network Intrusion Detection System*: These systems search the whole incoming network traffic from all devices within the network. They are placed on strategic points of the network to analyze the network more effectively.
- *Host–based Intrusion Detection System*: Instead of looking for the whole incoming

network traffic, these systems run on hosts or devices in the network. They are only responsible for the device or host they run on.

Apart from categorizing the intrusion detection systems for where they are placed, they can also be classified into two categories for their detection methods which are signature–based and anomaly–based explained below [10]:

- *Signature–based IDS*: These systems use known attack patterns to detect malicious activities. Examples of these patterns are byte sequences of network traffic or known attack sequences used by malware. Existing attacks can be easily detected by these systems but detecting zero–day attacks is tough process with them.
- *Anomaly–based IDS*: In these systems, the aim is to detect zero–day or unknown attacks. This is the method that has been studied extensively nowadays with the developing machine learning methods. The challenge is to keep false positive rate low while increasing the detection accuracy.

Intrusion detection systems are used to find the malicious traffic to keep the network stable. Different type of attacks can be executed for different purposes. For example, to search for the possible open ports on the network, port scan attack can be executed. The most common attacks encountered in network systems are DoS/DDoS attacks, brute force attacks and port scan attacks. These attacks are briefly explained in the upcoming section.

2.1.1. Common Network Attacks

Research community of network security have been looking for better solutions to combat with different kind of attacks. The most popular attacks evaluated by them are DoS/DDoS attacks, brute force attacks and port scan attacks.

2.1.1.1. DoS/DDoS Attacks. DoS attack is the malicious attempt to disrupt the service of any source by flooding it with different methods [11]. DoS is generally executed by overloading the target machine or source with unnecessary requests. This causes the blockage of some or all legitimate requests due to occupancy. DoS attack can be exemplified as a group of people blocking the doors of a shop or workplace and disrupting normal operations by not allowing legitimate parties to enter the store or business. DDoS is similar to DoS attack except that multiple sources are used in it. Attacker gains access to multiple computers or sources and use them to execute the attack. These compromised computers can be called botnets in network terminology. Attacker uses different methods such as viruses, trojans, malware etc. to gain access to these devices.

The frequency of executing DDoS attacks is increasing day by day. Nowadays, even small organizations can face with disruptive DDoS attacks. The increasing number of available attack tools makes it easier to execute [12]. A successful DDoS attack can cause serious damage to the victims. Even large companies around the world having strong security mechanisms can face with DDoS attacks. As an example, in October 16, 2020, Google faces with massive DDoS attack reported in [13]. UDP amplification attack is executed with a pace of 167 Mpps (millions of packets per second). Another example is that in February, 2020, Amazon Web Services experiences an extreme DDoS attack targeting an unidentified customer. The technique used in this attack is called Connection–less Lightweight Directory Access Protocol (CLDAP) reflection. The duration of the attack is measured as three days. It achieves the pace of 2.3 terabytes per second.

There are many techniques used to implement DDoS attacks. These attacks may last from a few minutes to a few months. The damage of the attack depend on the pace and duration of it. To categorize the DDoS attacks, the network layers in OSI model where the attack is targeted are considered. Within this property, DDoS attacks can be classified into three categories which are volume–based attacks, protocol attacks and application layer attacks. These attacks can be explained briefly as follows:

- *Volume–based Attacks:* The aim of this type of attack is to deplete the bandwidth of the victim's side. Usually, these attacks are executed with a high pace. In general, small size packets are sent with high transmission rate to overwhelm the victim's side. Some examples of this attack type are UDP flood, Internet Control Message Protocol (ICMP) flood and other type of spoofed–packet floods.
- *Protocol Attacks:* These attacks aim at the capacity of resources of victim side. In OSI Model, weaknesses of network and transport layers are exploited. One of the example of this attack type is synchronize (SYN) flood in which the attacker sends multiple connection request packets,SYN packets, to open the ports of victim side. It is based on using the TCP handshake principle where victim side tries to respond all the connection requests but the final connection cannot occur. The other examples of this attack type are ping of death, fragmented packet attacks and Smurf DDoS.
- *Application Layer Attacks:* This type of DDoS attack is relatively new compared to the aforementioned attack types. Compared to them, this attack type is has slower pace. It focuses on the vulnerabilities in the Application layer of the OSI model. The most popular examples are HTTP floods, DNS amplification, Network Time Protocol (NTP) amplification and memcached reflection. For example, DNS amplification is a reflection–based attack where the attacker uses DNS resolvers to execute the attack with a huge amount of traffic.

Among all category of DDoS attacks, some of them are mostly used in the attack scenarios. These are UDP flood, ICMP flood, SYN flood, ping of death, slowloris, and HTTP flood. In UDP and ICMP flood, victim side is overwhelmed with a huge number of UDP and ICMP packets. SYN flood as explained above, depends on the handshake principle of TCP packets. In ping of death, packets are fragmented and sent to the victim side as in the other flood types. Slowloris is a slow pace DDoS attack type where it uses minimal bandwidth. HTTP flood uses HTTP GET or POST requests to attack the victim side. It does not use spoofing IP addresses and reflection techniques. Compared to other attack types, it requires less bandwidth like slowloris.

2.1.1.2. Port Scan Attacks. In order to discover an active port and take advantage of victim side's known vulnerability, a port scan attack involves sending client requests to a variety of server port addresses on a host of victim [14]. The aim is to find weakness parts of victim side to execute more damaged attacks. Attackers can use a port scan attack to identify open ports and determine whether they can send data or not. Additionally, this attack can also show whether the victim side uses firewalls or other active security measures. Apart from the malicious intentions, using the port scanning technique, businesses can also send packets to particular ports and examine the responses for any potential vulnerabilities. The information taken with the port scanning operation can be stated as,

- The active services,
- Users having the services,
- If anonymous logins are permitted,
- What network services need to be authenticated.

A port is a location on a computer where data interchange occurs between various programs, the internet, and hardware or other computers. Ports are given as port numbers in order to maintain consistency and make programming procedures simpler. Port numbers are sorted in the order of popularity ranging from 0 to 65536. Well known ports, which are normally set aside for internet usage but may also have some specific applications, are those with a port number between 0 and 1023. UDP, which is primarily used for establishing low–latency and loss–tolerating connections between applications, and TCP, which specifies how to establish and maintain a network conversation between applications, are typically responsible for managing ports. Most frequently used port number can be exemplified as port 20 (UDP) for FTP, port 80 (TCP) for HTTP and port 22 (TCP) for Secure Shell (SSH) protocol. There are different methods for implementing port scanning techniques [15]. They can be stated as follows:

- *Ping Scan:* This is the easiest port scanning method. The procedure is to send ICMP requests to various sources to get a response.
- *Vanilla Scan:* In this method, all 65536 port numbers are tried to be connected at the same time. This connection is enabled using responses from SYN and Acknowledgement (ACK) packets. However, connection attempt to the all ports gives alert in a short time.
- *SYN Scan:* In this method, attacker sends SYN request, and after the response the attacker does not continue communication. In this way, the attacker learns whether the port is open even though the interaction is not recorded. Attackers utilize this rapid method to identify vulnerabilities.
- *Christmas Tree Type Scan (XMAS):* The set of flags that are enabled within a packet and which, when seen in a protocol analyzer like Wireshark, appear to be blinking like a Christmas tree give XMAS scans their name. A sequence of flags are sent during this kind of scan, and depending on how they are handled, they may reveal information about the firewall and the status of the ports. The system's response to them can give the attacker information about the system's degree of activity.
- *FTP Bounce Scan:* By using an FTP server to bounce a packet, the sender can conceal the locations using this method.
- *Sweep Scan:* This basic port scanning approach sends communication to a port across a network of computers to determine which ones are active.

2.1.1.3. Brute Force Attacks. A brute force attack is a hacking technique that makes use of trial and error to break encryption keys, password and login information [16]. It is a straightforward but effective strategy for getting unauthorized access to user accounts, networks and systems of companies. The term, brute force, refers to attacks that utilize excessive force in an effort to obtain critical information. Although it seems like an old method, brute force attacks continue to be a favorite among hackers. An application of brute–force search, which is a broad method of listing all candidates and checking each one, is considered to be a brute–force attack.

In order to gain illegal access and steal user data, attackers can utilize a variety of brute force attack techniques. Different strategies can be used by each brute force attack to find sensitive data. These techniques can be stated as follows:

- *Simple Brute Force Attacks:* When the attacker uses no software at all and tries to guess a user's login information manually, this is known as simple brute force attack. Usually, this is done using Personal identification number (PIN) codes or common password combinations.
- *Dictionary Attacks:* A dictionary attack is a fundamental type of brute force hacking in which the attacker chooses a target and runs potential passwords to find the real one. Dictionary attacks are so named because they involve hackers going through dictionaries and replacing words with symbols and numbers.
- *Hybrid Brute Force Attacks:* A dictionary attack method combined with a simple brute force attack is known as hybrid brute force attack. The attacker first needs to have access to a username before using dictionary and brute force attack techniques to find an account login information.
- *Reverse Brute Force Attacks:* By beginning with a known password, a reverse brute force attack turns the attack tactic on its head. Once they locate a match, attackers look through millions of usernames.
- *Credential Stuffing:* Attackers will test a username and password combination on numerous websites if they have one that works for one of them. Users with reusing same login information across numerous websites are the targets of this type of attack.

In this section, common network attacks are explained with the examples. The most used ones are explained in more detail. In the next section, related work in the literature about network intrusion detection systems is given.

2.1.2. Related Work About Network Intrusion Detection Systems

With the development of internet technologies, network attacks have evolved. These attacks have become serious threat to any organization from small ones to the large ones around the world. Therefore, with this rapid increase in the network attacks, attack detection methods are also evolving day by day. This section summarizes the existing detection mechanisms against common network attacks defined above.

The prevailing attacks against network systems are DoS/DDoS attacks. There have been numerous solutions against these attacks ranging from signature–based techniques to newly developed machine learning–based techniques. The overview of the existing solutions against DoS/DDoS attacks is given in [17]. In this survey, first, the DDoS attack execution strategies are explained in detail. Authors categorize the DDoS attacks into four categories which are bandwidth depletion attacks, resource depletion attacks, infrastructure attacks and zero–day attacks. These different types of DDoS attacks are explained with visual demonstrations. Then, existing solutions against DDoS attacks are given as two groups which are signature–based detection and anomaly–based detection. Also, some DDoS mitigation tools are explanied such as Bro given in [18] and Snort given in [19]. Another survey evaluating detection and mitigation of DDoS attacks in Software Defined Network (SDN) systems is given in [20]. In this survey, authors explain how SDN structure is utilized in the detection of DDoS attacks. They review about 70 popular DDoS attack detection and mitigation methods prevalent in the literature. These methods are classified into four groups which are machine learning–based methods, Artificial Neural Network (ANN) based methods, information theory–based methods and other statistical methods. They also discuss the unresolved research questions, limitations and difficulties in applying DDoS attack detection techniques in SDN. Another survey regarding DDoS attack detection schemes against Internet of Things (IoT) networks is given in [21]. In this survey, security issues in IoT networks are evaluated. Then, the taxonomy of DDoS attacks in IoT networks is given. Then, various defence mechanisms against DDoS attacks are explained and compared to each other in term of performance.

Among the solutions against DDoS attacks, information theory–based ones have an important place. Entropy is a measure of randomness of a system. It is used in many different forms to detect different kind of attacks. There have been numerous solutions using entropy to detect DDoS attacks. The entropy of each packet header values can be used in the detection of different type of attacks. Destination IP entropy based DDoS attack detection methods are given in [22–26]. These solutions are proposed for the security of SDN. In [22], authors focus on the security of data plane of SDN. They use Floodlight controller. The data set they choose are CAIDA 2007 given in [27] and synthetic data set created with Scapy tool. For this paper, publicly available data set used is an old data set. Another solution given in [23] focuses on controller plane of SDN by using POX controller. To calculate the destination IP entropy value, network packet window with a size of 50 is used. They use synthetic data set generated with Scapy tool. One of the drawbacks that can be stated about this work is that they do not use publicly available data set. They state that by changing the parameters, their solution is applicable to different controller requirements. Another proposed method given in [24] is similar to the previous methods except that in this method mitigation policy is also applied. The paper given in [25] combines two modules into one system which are statistical collection module and anomaly detection module. Information distance was used by authors to measure the deviation in network flows. They also analyze the difference between high–rate DDoS attacks and flash events in [26]. Information theory metrics are used to discriminate these network events.

Apart from the entropy of destination IP addresses, entropy of other packet headers such as source IP addresses, source port numbers, destination port number etc. can be used for the detection of DDoS attacks. The methods using the entropy of different packet headers are given in [28–32]. The proposed system given in [28] is designed for all layers in SDN which are application, control and data. Shannon entropy of IP addresses and port numbers are used for the detection phase. Both mitigation and detection of DDoS attacks are implemented in this system. Besides DDoS attacks, proposed system is also able to detect port scan attacks. Joint entropy of source and destination IP addresses are used in [29]. In this work, both mitigation and detection

are performed. Simulation environment is created in Mininet using real world traffic traces taken from MAWI Working Group Traffic Archive [33]. In this system, joint entropy of pair profiles is calculated and compared with threshold value to detect DDoS attacks. Also, in mitigation stage, according to the joint entropy results, different rules are applied to mitigate the attacks. High performance is obtained using the proposed approach. Another approach using the entropy of packet header values is given in [31]. Statistical approach is implemented where a score is calculated using the entropy values. Then, this score is compared with a threshold value to determine whether there is anomalous traffic or not. Synthetic data set is created using Scapy and Hping tools. Besides Shannon entropy, different type of entropy values are used to detect DDoS attacks. Conditional entropy is used for the detection of DDoS attacks given in [34]. In this approach, conditional entropy between source and destination IP addresses is used for the detection. Dynamic threshold is used to decide whether incoming traffic is anomalous or not. Authors use LLS 2.0 data set from MIT Lincoln Lab given in [35] to validate their results. They reach 99.372% average detection rate. Another different usage of entropy called as ϕ–entropy is given in [36]. Authors define a new information theory metric using Shannon entropy and sinh function. They consider destination IP to calculate ϕ–entropy values. Network packet window size is taken as 50 and attack detection is performed if the ϕ–entropy value is higher than the static threshold value for 5 consecutive windows. Synthetic data set is used to validate the results with different attack traffic rates.

With the development of deep learning methods, network intrusion detection systems have focused on applying these techniques. One of the solutions using Convolutional Neural Network (CNN) in the detection of DDoS attacks is given in [37]. Authors call their system as LUCID which is a practical, lightweight deep learning solution. In this work, authors build matrices to represent the network traffic. They treat these matrices as input images and feed them into the constructed CNN architecture. For each packet, they extract 11 features to represent it. Some of these features are packet length, TCP flags, TCP window size, UDP length etc. Time series analysis is performed using network packet windows. Proposed approach is validated using dif-

ferent publicly available data sets which are ISCX2012 [7], CIC2017 and CSECIC2018 given in [8]. They reach approximately 99% accuracy in all data sets. The approach given in [38] combines entropy based features with deep learning. Authors focus on removing the static threshold situation using this approach. Simulation results are taken using ISCX2012 data set. It is observed that there is an increase in the performance compared to traditional threshold–based approaches. They reach 94.74% accuracy which is higher than the paper they compare with the accuracy value of 90.04%. CNN is also implemented in some approaches using flow–based statistical features such as flow duration, total number of bytes etc. given in [39–42]. Statistical features are combined with payload–based features to detect network attacks in [39]. To extract payload–based features, word embedding and text–CNN methods are applied. Then, whole feature set is fed into Random Forest classifier. Experiment results are taken with ISCX2012 data set. Authors give performance results with different number of features and different length of payload. In overall, they reach 99.13% accuracy. In [40], CNNs are compared with Recurrent Neural Networks (RNNs). Authors compare the performance of four deep learning architectures which are basic CNN, inception architecture CNN, Long Short–Time Memory (LSTM) and Gated Recurrent Unit (GRU). They use ISCX2012 to compare the selected deep learning architectures. Flow–based statistical features are used to implement these architectures. Simulations are done using five–class model including normal, infiltrating, HTTPDoS, DDoS and Brute Force SSH traffic. Different architectures perform better for different type of traffic. For example, while GRU gives the best performance in normal traffic, basic CNN gives the best performance in HTTPDoS. The approach given in [41] uses deep learning models for intrusion detection in IoT networks. Authors compare the performance of four different models which are Multilayer Perceptron (MLP), CNN, LSTM and combination of CNN and LSTM. They use DDoS samples from CICIDS2017 data set to validate their results. By using the models mentioned, they reach highest accuracy as 97.16%.

Among different techniques implemented in intrusion detection systems, the approach given in [43] uses Graph theory to find attacks in SDN systems. They use flow–based approach to construct graph models. To measure the similarity between the

constructed graphs, Pearson correlation is used. The parameters considered during the construction of graphs are IP addresses, port numbers, number of packets and protocol type. In the classification phase, k Nearest Neighbor (k–NN) classifier is selected. Simulation results are taken with CAIDA data set and a synthetic data set created using Hping3 attack creation module. Authors show that their approach outperforms other selected approaches using Self Organizing Map (SOM) and Support Vector Machine (SVM). Another approach using Graph theory is given in [44]. This system uses graph inference model which is a relational graph between the known traffic patterns and their labels (anomalous or normal). In the feature selection process, Chow-Liu algorithm is used to select only observable features. These features are constructed using flow–based statistics. Simulations are done using ISCX2012 data set. They reach 89.30% detection accuracy. Besides using Graph theory, Bloom filters are extensively used in intrusion detection systems. An approach given in [45] uses Bloom filters for the detection of DDoS attacks. Flow–based features are selected which are flow duration, number of packets and total byte. Synthetic data set is created using Iperf tool. Performance of the system is measured using different attack rates. Another different approach used in intrusion detection systems is the application of cumulative sum given in [46]. Authors consider number of packets to use cumulative sum. Also, adaptive threshold is obtained using the cumulative sum. The proposed system is tested with CAIDA [27] and DARPA [35] data sets. False alarm rate is calculated as under 11.64% and detection duration for DDoS attacks is measured as 4.15 seconds. Queuing theory is another method implemented in intrusion detection systems. An approach given in [47] uses queuing theory to detect DDoS attacks. Flow table space is constructed to use queuing theory. The approach examines the flow table state of all other switches to find a suitable switch when the target switch is under attack. Calculating the unused slots of other switches involves using a mathematical model based on queuing theory. Experiments are done using synthetic data set created on Mininet. Performance of the system is evaluated with different attack rates.

2.2. Network Application Classification

Traffic classification is the process of categorizing traffic flows according to the service class, which identifies the application category to which the flow belongs. The management of network performance is crucial for Internet Service Providers (ISPs). The ability to classify network traffic has the potential to address a variety of network issues for businesses, individuals, ISPs, and the government, including capacity planning, anomaly detection, application performance and trend analysis. Utilizing this method, network administrators can control resources and prevent certain flows among other things. The growth of the network applications can also be observed. Researchers are working to develop lightweight algorithms for classification that have the least amount of computing needs in order to deal with the issues caused by the growth in traffic types and transmission rates.

To define Virtual Private Networks (VPNs), it can be said that users are safely connected to an enterprise network using VPNs. VPN uses packet–level encryption to ensure the security of the data sent over the internet. Due to the network traffic encryption, traffic classification for VPN connections and encrypted traffic, which requires associating traffic flows towards a type of application, is difficult to complete as stated in [48]. Therefore, newly developed solutions must consider the classification of network traffic with encrypted packets. Only analyzing the IP addresses and port numbers will not be adequate to determine which class the network traffic belongs.

The process of identifying the network applications or protocols that are present in a network is known as network application classification. In the last two decades, network application classification has become increasingly important. Numerous approaches to classify network applications have been put forth by researchers. The techniques used in network application classification can be broadly divided into three groups which are port–based technique, payload–based technique and machine learning–based technique. These techniques are explained as follows:

- *Port–based Technique:* In this method, the well–known port numbers are used to classify network traffic into the applications. Traditional network applications are listed in the Internet Assigned Number Authority (IANA) under their ports. Applications that make use of particular protocols such as HTTP, FTP, Post Office Protocol 3 (POP3) and ICMP, can benefit from these techniques. For instance, email applications utilize the port number 25 Simple Mail Transfer Protocol (SMTP) to send emails, and the port number 110 (POP3) to receive emails. Web applications use port number 80 in this fashion. However, the success rate of port–based techniques has significantly dropped over time as more applications began to employ dynamic port allocation. Additionally, some applications have not submitted their port registrations to IANA, which poses a significant obstacle for port–based techniques.
- *Payload–based Technique:* In this method, the packet contents are inspected in search of the characteristics and signatures of network applications in the network traffic. It is similar to the signature–based approach explained as one of the approaches of DPI techniques. One of the drawbacks of this approach is that in order to look for patterns in payload, relatively expensive hardware is required. Also, in encrypted network application traffic, this method fails. In addition to encryption, this approach is unable to find the application if the payload's signature is not recognized.
- *Machine learning–based Technique:* In this method, unknown classes are identified using the trained sample prediction and a machine learning classifier that has been taught as input. Flow–based statistical features can be utilized to classify applications. The machine learning methods offer significantly more versatility because these features are independent of port and payload. On the other hand, payload–based features can be added into these features to be used in application classification as done in this thesis.

In this section, brief introduction of network application classification is given. The approaches used in application classification are explained. In the next section, literature review about these methods is given.

2.2.1. Related Work About Network Application Classification Systems

To look for the solutions provided for network traffic classification in application identification, first attempts were made using port–based techniques. Example studies of this approach are given in [49, 50]. In [49], authors investigate the network traffic of peer–to–peer (P2P) applications. They mention that these applications use arbitrary port numbers. To overcome this situation, they suggest some heuristics about the behavior of these applications. For example, they claim that port 80 is generally selected as destination port for the communication of these applications. Another study given in [50] mentions about the insufficiency of well–known port–based approach. They compare the port–based approach with content–based approach and show the superiority of content–based approach. One of the studies using payload–based approach in network traffic classification is given in [51]. In this work, authors compare the performances of lightweight traffic classification approach with a completely stateful approach. They show that even though the first method is less exact, it is nevertheless suitable for a wide range of applications. On the other hand, the approach given in [52] aims at increasing the processing speed of the network traffic classification architecture. Authors propose many design options to execute the traffic classification process in real time. They analyze the flow–based parameters such as number of packets within a flow and packet size to find the optimum combination of them. Lastly, the approach given in [53] is also deals with the problem of computation duration in payload–based classification systems. Authors analyze top ten applications of the categories such as gaming, multimedia, web browser, commercial, file sharing etc. Distributions of IP addresses and port numbers are examined using cumulative distribution function. They suggest Same Server IP Port Cache–based classification to improve the performance of payload signature–based method. They reach 10–fold increase in the speed of process and more than 10% increase in the completeness of the classification.

With the newly developing technology, network traffic classification methods focus on machine learning techniques. Also, encrypted messages make it tough for older algorithms to classify the network traffic into applications. There are several

approaches using machine learning techniques for the network traffic classification given in [54–59]. Comparison of ten different classification algorithms with features selection techniques is given in [54]. Some of these algorithms are C4.5 Decision Tree, Naive Bayes, Bayesian Networks, MLP Network etc. Authors use publicly available data set National Laboratory for Applied Network Research network traces given in [60]. By using 22 flow–based features, they reach more than 97% accuracy. Review of the studies using machine learning techniques in network traffic classification from 2004 to early 2007 is given in [55]. The approach given in [56] tries to classify the network traffic into three classes which are normal web traffic, web traffic and audio traffic. They combine keyword matching and statistical profiles for the classification. The features they use are average received packet size, flow duration and ratio of packet count. The recall rate they achieve is 84% using the synthetic data set they generated. In [57], authors implement machine learning methods on to the payload–based features. They use data set collected from the campus network of SunYat–Sen University. Within the seven categories, they reach more than 90% accuracy. Another study given in [59] focuses on network traffic classification in cloud–based systems. This research suggests an architecture for a cloud–based traffic classification service for parallel classification and sharing models. They use statistical information of flows as features. This data is stored in a database on the cloud, and a machine learning–based training system is also created.

One of the studies using different machine learning algorithms in network traffic classification is given in [61]. In this paper, authors implement four different classification algorithms which are J48, Random Forest, k–NN and Bayesian Network. They use a subset of flow–based features decided using chi–square statistic value. For the simulations, they use two data sets which are publicly available UNB ISCX Network Traffic data set [48] and the synthetic data set they generated. The selected features differ for these data sets due to the flow characteristics. Among 111 features 12 features are selected for each data set. Among different classifier selections, for ISCX data set, they reach overall 93.94% accuracy with k-NN classifier and for the synthetic data set they reach 90.87% accuracy with Random Forest classifier.

Another study given in [62] proposes a multimodal multitask deep learning system for encrypted network traffic classification. They use payload of packets to build the features using their proposed deep learning architecture. For the simulations, ISCX data set is used. Simulations are done using three objectives. First objective is called encapsulation in which network traffic is classified as VPN or nonVPN. Second objective is called traffic type in which classification is based on the application type categories which are VoIP, File Transfer, P2P, Streaming, Chat and Email. The last objective is called application in which directly the application of network traffic is found.
15 different applications are used such that Skype, Facebook and YouTube are the most popular ones. Authors compare their results with 8 different approaches and they claim that their system outperforms them. Accuracy values obtained for their system are 93.75% for encapsulation objective, 80.78% for traffic type objective and 77.63% for application type objective.

3. THEORETICAL BACKGROUND IN INFORMATION THEORY

In network anomaly detection and application classification systems, some metrics based on information theory are extensively used. The most common used metrics are entropy and divergence. In network anomaly detection systems, these metrics reflect the behavior of network traffic effectively during both attack and attack–free cases. On the other hand, to characterize the network traffic to find which applications are running through network, these metrics are also helpful because different values are observed with different applications. In the beginning, entropy concept is introduced. After defining what the entropy is, divergence concept is explained.

3.1. Entropy

Entropy can be defined as a measure of uncertainty or randomness of a system. It is introduced by Claude Shannon given in [63] called as Shannon entropy. Given a random variable X, Shannon entropy of X is given as

$$H(X) = -\sum_{i=1}^{n} p_i \log_2(p_i), \tag{3.1}$$

where p_i is the probability of i^{th} outcome of the event X among n different outcomes with $p_i \geq 0$. After the concept of Shannon entropy is introduced, Alfred Renyi generalized the definition of entropy by introducing α-Entropy [64], given as

$$H_\alpha(X) = \frac{1}{1-\alpha} \log_2 \Big(\sum_{i=1}^{n} p_i^\alpha \Big), \tag{3.2}$$

where α is called order. The addition of the order α generalizes the concept of entropy. In this way, by choosing different α values, this metric can highlight the different portions of probability distributions.

For example, when $\alpha \geq 0$, this metric becomes more sensitive to the events that occur frequently. On the other hand, by defining $\alpha < 0$, less frequent events are more focused. Different kind of entropy metrics can be obtained by changing the value of order α. Maximum value of the entropy can be found by taking $\alpha = 0$, which is known as Hartley entropy given as

$$H_0(X) = \log_2(n). \tag{3.3}$$

From α-Entropy, Shannon entropy can be found by taking $\alpha \to 1$. To show this, when $\alpha \to 1$, $\frac{0}{0}$ uncertainty appears. To solve this uncertainty, l'Hopital's Theorem is used.

Proof. The theorem states that

$$\lim_{\alpha \to m} \frac{f(\alpha)}{g(\alpha)} = \lim_{\alpha \to m} \frac{f'(\alpha)}{g'(\alpha)}$$

where $m = 1$ in this case. Putting

$$f(\alpha) = \log_2 \Big(\sum_{i=1}^{n} p_i^{\alpha} \Big)$$
$$g(\alpha) = 1 - \alpha,$$

gives

$$\frac{d}{d\alpha} g(\alpha) = -1$$
$$\frac{d}{d\alpha} f(\alpha) = \frac{1}{\sum_{i=1}^{n} p_i^{\alpha}} \sum_{i=1}^{n} \frac{d}{d\alpha} p_i^{\alpha},$$

where

$$\frac{d}{d\alpha} p_i^{\alpha} = p_i^{\alpha} \log_2(p_i).$$

Then, letting $\alpha \to 1$ gives

$$\frac{d}{d\alpha} f(\alpha) = \frac{1}{\sum_{i=1}^{n} p_i} \sum_{i=1}^{n} p_i \log_2(p_i)$$

where denominator is the summation of all probabilities which equals to 1. To conclude,

$$\lim_{\alpha \to 1} \frac{1}{1-\alpha} \Big(\sum_{i=1}^{n} p_i^{\alpha} \Big) = \lim_{\alpha \to 1} = \frac{f'(\alpha)}{g'(\alpha)} = -\sum_{i=1}^{n} p_i \log_2(p_i)$$

which is Shannon entropy $H(X)$. □

By taking $\alpha = 2$, different type of entropy called Collision entropy is defined given as

$$H_2(X) = -\log_2 \Big(\sum_{i=1}^{n} p_i^2 \Big). \tag{3.4}$$

This entropy metric is also known as Renyi entropy. It has many application areas such as signal processing, economics, physics etc. When $\alpha \to \infty$, minimum entropy is defined given as

$$H_\infty(X) \doteq \min_i(-\log_2(p_i)) = -(\max_i \log_2(p_i)) = -\log_2 \max_i p_i. \tag{3.5}$$

Proof.

$$\lim_{\alpha \to \infty} H_\alpha(X) = \lim_{\alpha \to \infty} \frac{1}{1-\alpha} \log_2 \Big(\sum_{i=1}^{n} p_i^{\alpha} \Big)$$

For $\alpha \to \infty$,

$$\frac{1}{1-\alpha} \approx -\frac{1}{\alpha}$$

Since $0 \leq p_i \leq 1$, and if $p_i \leq p_j$ then $p_i^\alpha \leq p_j^\alpha$, when $\alpha \to \infty$, the largest term, $(\max_i p_i)$, dominates the summation, that is

$$\sum_{i=1}^{n} p_i^\alpha \approx \max_i p_i^\alpha$$

and putting together with the first appromation,

$$\lim_{\alpha\to\infty} H_\alpha(X) = H_\infty(X) \approx -\frac{1}{\alpha}\log_2 \max_i p_i^\alpha = -\frac{\alpha}{\alpha}\log_2 \max_i p_i^\alpha = -\log_2 \max_i p_i.$$

□

The minimum entropy has some important application areas such as randomness extractors in theoretical computer science. It also has a vital space in quantum cryptography in the field of privacy amplification.

The different kind of entropies, by changing the order α, have the relationship given as

$$H_0(X) \geq H_1(X) \geq H_2(X) \geq ... \geq H_\infty(X), \tag{3.6}$$

where $H_0(X)$ is the maximum entropy and $H_\infty(X)$ is the minimum entropy. For the same probability distribution, by increasing the order α, the entropy value decreases.

Apart from the order α, base of the logarithm may be changed. Different choices of base of logarithm function can be used for different applications. Base 2 corresponds to the unit of bits. On the other hand, base e corresponds to natural units and base 10 corresponds to dits. Common selections of the base are 2 and e. Entropy is a popular information theory metric used in deep packet inspection systems such as anomaly detection or network application classification. It highlights the behavior of packet header values such as source IP addresses. In these systems, Shannon entropy

is the most used one. By using α-Entropy with different α values, sensitivity can be put on which portion of distributions whether frequent ones or less frequent ones will be analyzed. Therefore, by using this property, entropy is used in many deep packet inspection studies.

3.2. Divergence

Based on the entropy concept, divergence is the distance between two probability distributions [65]. Divergence measures how much these probability distributions close to each other. Smaller divergence value show that probability distributions have similar quantities of information. In the context of deep packet inspection, divergence measures how much the behavior of packet header values are different from each other. For example, source IP addresses and destination IP addresses may behave differently during a network attack. In this case, divergence may highlight this behavior difference. Compared to attack–free traffic, divergence value may be higher or lower. This situation can be utilized in attack detection schemes.

Like in the entropy case, divergence is defined with the order α. For two discrete probability distributions $P = [p_1, p_2, ...p_n]$ and $Q = [q_1, q_2, ...q_n]$, the divergence can be calculated as

$$D_\alpha(P||Q) = \frac{1}{1-\alpha} \log_2 \Big(\sum_{i=1}^{n} p_i^\alpha q_i^{1-\alpha} \Big), \tag{3.7}$$

where $\alpha \geq 0$. When α is taken as 1, Kullback–Leibler (KL) divergence [66], which is also called relative entropy is derived given as

$$D_1(P||Q) = D_{KL}(P||Q) = \sum_{i=1}^{n} p_i \log_2 \Big(\frac{p_i}{q_i} \Big). \tag{3.8}$$

KL divergence measures the closeness of two probability distributions P and Q. In other words, it is the measured distance from the approximated distribution Q when

the true distribution is P. KL divergence is a statistical distance but it is not a metric. Properties of KL divergence can be written as follows:

- $D_{KL}(P||Q) \geq 0$ and $D_{KL}(P||Q) = 0 \Leftrightarrow P = Q$
  KL divergence takes values on the interval $[0, \infty]$. When the probability distributions are same, log term gives zero which in turn makes divergence zero. This property is also known as Gibbs' inequality.
- $D_{KL}(P||Q) \neq D_{KL}(Q||P)$ if $P \neq Q$
  The distance from P to Q is generally not the same as the distance from Q to P. Therefore, KL divergence is not commutative.
- KL divergence generally does not satisfy the triangle inequality.

  Proof. The proof is given with a contrary example. The triangle inequality states that

  $$d(x, z) \leq d(x, y) + d(y, z),$$

  where d is the distance metric, x, y and z are the variables. To apply triangle inequality in KL divergence case,

  $$D_{KL}(P||R) \leq D_{KL}(P||Q) + D_{KL}(Q||R)$$

  must hold for probability distributions P, Q and R. As an example, take the event space as $X = \{0, 1\}$. Let $P = [0.5, 0.5]$, $Q = [0.25, 0.75]$ and $R = [0.1, 0.9]$. Calculations give, $D_{KL}(P||R) = 0.51$, $D_{KL}(P||Q) = 0.14$ and $D_{KL}(Q||R) = 0.09$. Therefore,

  $$0.51 \not\leq 0.14 + 0.09$$
  $$D_{KL}(P||R) \not\leq D_{KL}(P||Q) + D_{KL}(Q||R)$$

  in general. □

KL divergence has many application areas ranging from Natural Language Processing (NLP), computer vision to compression. In the context of deep packet inspection, the behavior of packet header values are analyzed using KL divergence. To satisfy the symmetry using KL divergence, modified version of it given as

$$D(P||Q) = \frac{D_{KL}(P||Q) + D_{KL}(Q||P)}{2} \tag{3.9}$$

is used. $D(P||Q)$ clearly satisfies commutativity as $D(P||Q) = D(Q||P)$. However, it is again generally does not satisfy the triangle inequality. Therefore, it is not a metric.

KL divergence requires both probability distributions to have same size. In other words, as it can be seen from (3.8), both distributions, P and Q, must have n number of elements. If one of them has different number of elements, KL divergence cannot be applied. To solve this problem, one method is to append very small values ($\epsilon \approx 0$ and $\epsilon > 0$) to the probability distribution with less number of elements, and calculate KL divergence with the new constructed probability distribution. However, it is not seen as an effective method when the number of elements of the probability distributions are very different from each other. To overcome this situation, Greedy algorithm given in [5], is applied to probability distributions. Greedy algorithm calculates an upper bound for the distance between two probability distributions. It is based on maximizing the mutual information between two different probability distributions. Details of the Gredy algorithm is given in the next section.

3.3. Greedy Distance

Greedy algorithm is used to find a distance between two probability distributions with different cardinalities [5]. The aim is to maximize the mutual information between these two variables having different probability distributions. For two random variables

X and Y, mutual information can be given as

$$I(X;Y) = \sum_{x \in X} \sum_{y \in Y} P(x,y) \log_2 \frac{P(x,y)}{P(x)P(y)}, \tag{3.10}$$

where $P(X)$ is the marginal distribution of the random variable X, $P(Y)$ is the marginal distribution of the random variable Y and $P(X,Y)$ is the joint distribution of the random variables X and Y. It can also be written in terms of entropy such that

$$I(X;Y) = H(X) - H(X|Y) = H(Y) - H(Y|X), \tag{3.11}$$

where $H(X)$ and $H(Y)$ are the entropies defined in (3.1), $H(X|Y)$ and $H(Y|X)$ are the conditional entropies given as

$$H(X|Y) = -\sum_{x \in X, y \in Y} p(x,y) \log_2 \frac{p(x,y)}{p(y)} \tag{3.12}$$

and $H(Y|X)$ is the same as (3.12) except that the denominator is $p(x)$ in the $\log_2$ term. Mutual information given in (3.11) can also be written in terms of joint entropy of X and Y such that

$$I(X;Y) = H(X) + H(Y) - H(X,Y) = H(X,Y) - H(X|Y) - H(Y|X), \tag{3.13}$$

where joint entropy of X and Y can be given as

$$H(X,Y) = -\sum_{x \in X} \sum_{y \in Y} p(x,y) \log_2 p(x,y). \tag{3.14}$$

The Greedy distance can be calculated as follows. Let $\phi \in S_n$, $\psi \in S_m$ be the two probability distributions with different cardinalities. Suppose $n \geq m$. There are two NP–hard problems, which are deciding whether ψ is a fusion of ϕ or not, and obtaining the ideal bin allocations in the bin packing problem. To overcome these situations, the

Greedy algorithm is one of the proposed solutions. The algorithm is explained below.

(i) Let $s = 1$, where s is the number of iterations. For each iteration, $n_s = n$, $m_s = m$, $\phi_s = \phi$ and $\psi_s = \psi$ are defined.

(ii) Elements of ψ are sorted from largest to smallest. The first element of ϕ is allocated to the largest element of ψ. Then, the capacity (value) of this element of ψ is decreased by the value of the assigned element of ϕ. Then, sorting is applied to the ψ because one of its elements' value is changed. With this recursive process, bin packing process is applied. If an element $(\phi_s)_i$ cannot be assigned to any bin, its index i is added to the outlier set defined as K_s

(iii) Let $I_1^{(s)}, ..., I_{m_s}^{(s)}$ denote the indices of allocated elements of ϕ with corresponding indices of ψ. Also, K_s corresponds to the outlier set of indices defined in Step (ii). If $|K_s| > 1$, move to Step (iv); otherwise move to Step (v).

(iv) Unused capacities of each bin of ψ_s are defined as $\beta_1^{(s)}, ..., \beta_{m_s}^{(s)}$, and construct $\boldsymbol{\beta^{(s)}} = [\beta_1^{(s)}, ..., \beta_{m_s}^{(s)}]$. Then, the total unused capacity $c_s := \boldsymbol{\beta^{(s)}} e_{m_s}$ is given as

$$c_s = \sum_{j=1}^{m_s} \beta_j^{(s)} = \sum_{i \in K_s} (\phi_s)_i. \tag{3.15}$$

Since each $(\psi_s)_i$, $i \in K_s$ cannot be assigned to any bin, it gives $(\psi_s)_i > \beta_j^{(s)}$, $\forall i, j$. It means that $|K_s| < m_s$. Next, set $n_{s+1} = m_s$, $m_{s+1} = |K_s|$, and define

$$\phi_{s+1} = \frac{1}{c_s} \boldsymbol{\beta^s} \in S_{n_s+1}, \tag{3.16}$$

$$\psi_{s+1} = \frac{1}{c_s} [(\phi_s)_i)] \in S_{m_s+1}. \tag{3.17}$$

Increase the iteration number by one and move to Step (ii).

(v) In this step, $|K_s|$ must be equal to 0 or 1. $|K_s| = 0$ implies that ψ_s is an exact fusion of ϕ_s. $|K_s| = 1$ implies that only one element of ϕ_s, call it $(\phi_s)_k$ cannot be

allocated into any bin, therefore c_s equals to the value of this element. Then, let

$$\boldsymbol{v_s} = \frac{1}{c_s}\boldsymbol{\beta^{(s)}} \in S_{m_s}, \tag{3.18}$$

$$V_s = c_s H(\boldsymbol{v_s}), \tag{3.19}$$

$$U_s = V_s + H(\phi_s) - H(\psi_s). \tag{3.20}$$

Define $P_s \in S_{n_s \times m_s}$ by $\boldsymbol{p_i} = \boldsymbol{b_j}$ if $i \in I_j^{(s)}$ and $\boldsymbol{p_k} = \boldsymbol{v_s}$ where $\boldsymbol{b_j}$ is the j^{th} unit vector with m_s components. Defining

$$J_\phi(P) = \sum_{i=1}^{n} \phi_i H(p_i), \tag{3.21}$$

V_s is the minimum value of $J_{phi_s}(.)$ given in (3.21), and P_s reaches that minimum. Also defining $Q_s \in S_{m_s \times n_S}$ by

$$Q_s = [diag(\psi_s)]^{-1} P_s^T diag(\phi_s), \tag{3.22}$$

$J_{\psi_S}(.)$ is minimized by Q_s, and this minimum value is equal to U_s.

(vi) All previous steps are inverted by starting with Q_{s+1}. It gives U_{s+1} since it is the minimum value of the cost function defined in (3.21). Then, the calculation of V_s is given as

$$V_s = c_s U_{s+1}, \tag{3.23}$$

and U_s is calculated using (3.20). This backward process is applied by decreasing the iteration number by one. In each step, $m_s = n_{s+1}$ holds and c_s values are recalled.

After completing these steps and reaching from the last iteration to the initial iteration, a possible maximum value of the distance between two probability distribu-

tions is found by

$$d_{greedy}(\phi, \psi) \leq V_1 + U_1. \tag{3.24}$$

$d_{greedy}(.,.)$ is used as a Greedy algorithm function which takes two probability distributions as inputs and gives the upper bound value of the distance between these probability distributions. Details of the algorithm can be found in [5]. The illustration of the Greedy algorithm is given with an example below.

Example: In this example, the Greedy algorithm is illustrated with two randomly selected probability distributions. The probability distributions are $\phi \in S_{40}$ and $\psi \in S_{10}$ such that

$$\phi = \begin{bmatrix} 0.0304 & 0.0333 & 0.0153 & 0.0335 & 0.0253 & 0.0148 & 0.0178 & 0.0232 \\ 0.0157 & 0.0355 & 0.0350 & 0.0219 & 0.0299 & 0.0155 & 0.0205 & 0.0336 \\ 0.0259 & 0.0139 & 0.0314 & 0.0342 & 0.0265 & 0.0287 & 0.0283 & 0.0199 \\ 0.0273 & 0.0139 & 0.0177 & 0.0141 & 0.0148 & 0.0307 & 0.0270 & 0.0185 \\ 0.0350 & 0.0353 & 0.0297 & 0.0351 & 0.0259 & 0.0160 & 0.0348 & 0.0139 \end{bmatrix},$$

$$\psi = \begin{bmatrix} 0.1241 & 0.1205 & 0.1192 & 0.1139 & 0.1069 \\ 0.0914 & 0.0875 & 0.0869 & 0.0821 & 0.0675 \end{bmatrix},$$

where ϕ and ψ are row vectors but for demonstration purposes, they are shown in matrix format. The application of best–fit algorithm in the first round gives,

$$\begin{aligned} &I_1^{(1)} = \{1, 7, 16, 20, 32\},\ I_2^{(1)} = \{2, 11, 17, 25, 33\},\ I_3^{(1)} = \{3, 6, 9, 22, 28\} \\ &I_4^{(1)} = \{4, 15, 26, 34\},\ I_5^{(1)} = \{5, 14, 21, 30\},\ I_6^{(1)} = \{8, 18, 31\} \\ &I_7^{(1)} = \{10, 23, 35, 38\},\ I_8^{(1)} = \{12, 24\},\ I_9^{(1)} = \{13, 27, 40\} \\ &I_{10}^{(1)} = \{19, 29\},\ K_1 = \{36, 37, 39\}, \end{aligned}$$

where elements inside the sets given above represent the indices of values of the probability distribution ϕ. After the first round, the total unallocated capacity c_1 is 0.0924.

In the second round, new probability distributions are constructed such that

$$\phi_2 = \begin{bmatrix} 0.1237 & 0.0721 & 0.0183 & 0.0825 & 0.1934 \\ 0.0793 & 0.0639 & 0.1856 & 0.0521 & 0.1291 \end{bmatrix},$$
$$\psi_2 = \frac{1}{c_1}\begin{bmatrix} \phi(36) & \phi(37) & \phi(39) \end{bmatrix} = \begin{bmatrix} 0.3317 & 0.2917 & 0.3766 \end{bmatrix},$$

where again ϕ_2 is a row vector but shown in matrix format. In this round, the results of the application of best–fit algorithm are shown as

$$I_1^{(2)} = \{2, 5\},\ I_2^{(2)} = \{3, 4, 7\},\ I_3^{(2)} = \{1, 6, 9\},\ K_2 = \{8, 10\},$$

where indices in the sets belong to the values of ϕ_2. The unallocated capacity in this round is $c_2 = 0.3146$. New constructed probability distributions are given as

$$\phi_3 = \begin{bmatrix} 0.2103 & 0.4037 & 0.3860 \end{bmatrix},$$
$$\psi_3 = \frac{1}{c_2}\begin{bmatrix} \phi_2(8) & \phi_2(10) \end{bmatrix} = \begin{bmatrix} 0.5898 & 0.4102 \end{bmatrix}.$$

Now, by using (3.21), P_3 can be computed as

$$P_3 = \begin{bmatrix} 0.9691 & 0.0309 \\ 0 & 1 \\ 1 & 0 \end{bmatrix}.$$

After that, Q_3 can be computed using (3.22) given as

$$Q_3 = \begin{bmatrix} 0.3456 & 0 & 0.6544 \\ 0.0158 & 0.9842 & 0 \end{bmatrix}.$$

After the calculation of these matrices, backward computation can be applied. V_3 and

U_3 can be computed as

$$V_3 = J_{\phi_3}(P_3) = (\phi_3)_1 H((P_3)_1) = 0.0290,$$
$$U_3 = V_3 + H(\phi_3) - H(\psi_3) = 0.4136.$$

Now, going one step further, V_2 and U_2 can be computed. To compute V_2, P_2 is necessary. P_2 is 10×3 matrix having 8^{th} and 10^{th} rows from Q_3 and the rest of its rows are elementary row vectors. The i^{th} row of P_2 is equal to the j^{th} elementary row vector if the index i belongs to $I_j^{(2)}$. Then,

$$V_2 = J_{\phi_2}(P_2) = c_2 U_3 = 0.1301,$$
$$U_2 = V_2 + H(\phi_2) - H(\psi_2) = 1.1894.$$

Now, in the final stage, Q_2 is calculated from P_2 using (3.22) given as

$$\begin{bmatrix} 0 & 0.2174 & 0 & 0 & 0.5831 & 0 & 0 & 0.1933 & 0 & 0.0062 \\ 0 & 0 & 0.0627 & 0.2827 & 0 & 0 & 0.2191 & 0 & 0 & 0.4354 \\ 0.3286 & 0 & 0 & 0 & 0 & 0.2105 & 0 & 0.3225 & 0.1384 & 0 \end{bmatrix}.$$

By using Q_2, P_1 is calculated where the rows given above are 36^{th}, 37^{th} and 39^{th} rows of P_1. Other 37 rows of P_1 are elementary vectors. Lastly, V_1 and U_1 are calculated as

$$V_1 = J_{\phi}(P) = c_1 U_2 = 0.1099,$$
$$U_1 = V_1 + H(\phi_1) - H(\psi_1) = 1.4655.$$

Now, to conclude, the Greedy distance which is an upper bound for the distance between the probability distributions $\phi \in S_{40}$ and $\psi \in S_{10}$ is found as

$$d_{greedy}(\phi, \psi) \leq V_1 + U_1 = 1.5744.$$

4. FEATURE EXTRACTION METHODS FOR DEEP PACKET INSPECTION SYSTEMS

In this chapter, feature extraction methods for deep packet inspection systems are explained. Features in network systems are used to model the network behavior effectively to make analysis and inspection better. Different methods for feature extraction are proposed to put emphasis on different behavior of the network traffic. In this work, these extracted features are used in anomaly detection and network traffic classification processes. While some features can be used in both processes, some of them are useful for only one process.

Feature extraction methods can be broadly categorized into two groups which are packet–based features, flow–based features. Network packets contain packet–header values such as source IP address, destination IP address, source port number, destination port number, protocol etc. and data part which is the actual message called payload. The payload part is often encrypted. In packet–based and flow–based feature extraction methods, packet–header values are used. The behavior of these values during different network traffic is highlighted with these methods. Then, these features are used in packet inspection algorithms. On the other hand, as the name shows, payload–based features use the payload part of the packet. These two feature extraction method categories are not sharply separated from each other. For example, one payload–based feature can be used within network flows. These categories are build for explaining the methods effectively. There are some fundamental differences when processing network traffic between different feature extraction methods. For example, in flow–based feature extraction methods, network packets are gathered into flows according to their packet–header information such that packets with same source IP address, destination IP address, source port number, destination port number and protocol are gathered into flow. These differences are explained in the upcoming sections.

4.1. Packet–Based Feature Extraction Methods

Packet–based feature extraction methods can be used in two different ways which are analysis of each packet individually or analysis of a packet group together. In the second method, a window is required to collect some number of packets to perform feature extraction. Network traffic window can be selected according to some duration or some predefined number of packets. Each choice method has some advantages and drawbacks. When network traffic window is selected as some duration such as one second, time–series analysis can be performed. However, if the window size is not selected according to the network traffic pace, with the complexity of feature extraction methods, system may not work perfectly. This may result in missing some packets and performance slowdown. On the other hand, network traffic window size can be selected with predefined number of packets. In the number of packets are low, network traffic may not be modelled effectively. Also, if the number of packets are high, feature extraction method complexity may increase. This also affect the performance of the system. Therefore, selection of network traffic window size is an important phase for feature extraction methods. Adaptive window size selection has to be studied and implemented for product level works.

Packet–based features extraction methods implemented in this work uses network traffic windows. Network packets are gathered according to some rules defined before and feature extraction is implemented. There are different proposed algorithms used for modelling network traffic in different ways. These algorithms use packet–header values to analyze the behavior of these values during different network traffic. As an example, the behavior between source IP addresses and destination IP addresses can be analyzed by defining features focusing on this behavior. First algorithm implemented in this work is called Modularity method consisting of a clustering algorithm followed by an indicator function called modularity.

4.1.1. Modularity

The main idea in this feature extraction method is to examine the relationship between different packet header values such as source IP addresses and destination IP addresses. After selecting one packet header from the source side and another from the destination side, clustering is applied to the source side based on the connections with the destination side. Compared to the classical clustering algorithms, the clustering algorithm used in the system has a major advantage. It requires no parameters to inialize the clustering unlike the k–parameter in k–means clustering or ϵ value in DBSCAN clustering. This in turn provides modeling the network more effectively. Also, another strong side of this clustering algorithm is that it does not use the packet payloads which are often unavailable from flow records. The process of the clustering algorithm is explained below.

(i) Selected packet headers is divided into two side. X represents the source side, and Y represents the destination side.

(ii) Bipartite graph, $G = (X, Y, E)$, is constructed. In this graph G, X corresponds to the all unique packet header values belonging to the source side and Y corresponds to the all unique packet header values belonging to the destination side. Also, E corresponds to the connections between the source side and destination side. An example of a bipartite graph is shown in Figure 4.1. Let n be the number of unique packet header for the source side ($n = |X|$) and m be the number of unique packet header for the destination side ($m = |Y|$). This constructed bipartite graph can be represented by an adjacency matrix which is

$$(A_{i,j}) = \begin{cases} 0, & \text{if there is no connection between i and j,} \\ 1, & \text{otherwise,} \end{cases} \tag{4.1}$$

where $i \in \{1, 2, ..., n\}$ and $j \in \{1, 2, ..., m\}$. The size of the adjacency matrix A is $n \times m$.

For a node v, $deg(v)$ represents the number of connections it has with the other side. Sum of the degrees in one side of a graph is equal to the total number of edges shown as

$$\sum_{x \in X} deg(x) = \sum_{y \in Y} deg(y) = |E|. \tag{4.2}$$

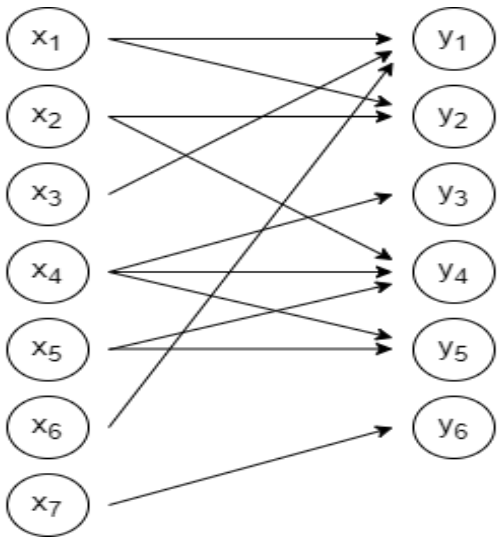

Figure 4.1. Example of a bipartite graph.

(iii) One mode projection is applied on the constructed bipartite graph. In the one mode projection, new edges are constructed between the nodes on the same side of the previously constructed bipartite graph. If two nodes on the same side have connection with a common node on the other side, one edge is created between these nodes. For every common destination node, an edge is created between these source nodes. It means that source nodes are projected over the destination nodes. This in turn gives a new graph, $G' = (X, E')$, where E' corresponds to the connections between the elements of the source side. One mode projection applied version of the previous example is shown in Figure 4.2. In this example, since x_7 has no common connections to the destination side with other elements of the source side, it can be regarded as an outlier.

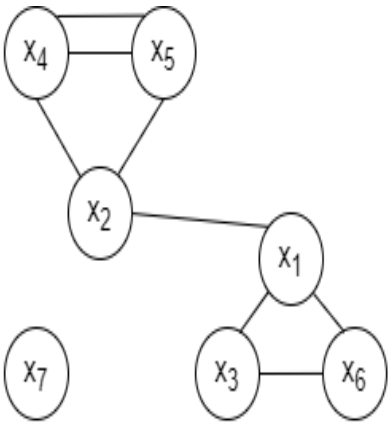

Figure 4.2. One mode projection applied graph.

After applying the one mode projection, the connected nodes could be considered as similar in terms of the social relationship in the network. The adjacency matrix of this one mode projection is called as Similarity matrix $S_{n\times n} = (s_{ij})$, where s_{ij} corresponds to the number of common destination nodes which are connected source node i and j. It is symmetric matrix and all diagonal entities are equal to the zero, $s_{ii} = 0$. The representation of this matrix is given as

$$S_{n\times n} = \begin{bmatrix} s_{11} & s_{12} & \dots & s_{1n} \\ s_{21} & s_{22} & \dots & s_{2n} \\ \vdots & \ddots & \ddots & \vdots \\ s_{n1} & s_{n2} & \dots & s_{nn} \end{bmatrix}. \tag{4.3}$$

(iv) In this step, construction of clusters begins. Similarity matrix constructed in the previous step is used to form the inital clusters. Each row is assigned to a cluster. In each row, if an entry has a value greater than zero in the similarity matrix, then it is added to the cluster formed by this row. After forming initial clusters by using the rows and nonzero values at the columns from the similarity matrix, if a cluster has only one element, then it is removed from the remaining process.

(v) After obtaining the initial clusters, clusters that are subsets of other clusters are removed. The aim of this operation is to decrease the initial number of clusters and reduce the computation time.

(vi) In this step, in order to assign each element to only one cluster, if an element x_i exits in clusters C_k and C_l, then Affiliation factors of this element to these clusters are calculated and compared. Affiliation factor indicates the loyalty degree of an element to a cluster given as

$$AF(x_i, C_k) = \sum_{j \in C_k} s_{ij} \tag{4.4}$$

where x_i is a node belonging to cluster C_k. After calculating Affiliation factors $AF(x_i, C_k)$ and $AF(x_i, C_l)$, the element is removed from the cluster which gives lower affiliation factor. If there is an equality between affiliation factors, the element is removed from the cluster having less number of elements. After each process, clusters having only one element are removed. This element is added to the compared cluster having the higher affiliation factor.

Constructed clusters of the previous example are given in Figure 4.3. Outlier node is removed from the remaining process. After completing the steps of the clustering algorithm, clusters have high weighted interconnections and low weighted connections with the other nodes in the network. As a result, highly connected clusters are obtained and they correspond to the similar nodes.

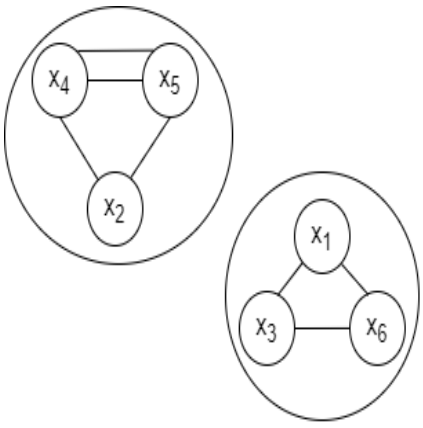

Figure 4.3. Constructed clusters after the proposed method.

After the clustering process, a function is necessary to signify the strength of the clusters. Modularity metric was proposed in [6] to measure the performance of clustering and community detection method algorithms in different network models. The modularity of a clustering process evaluates the strength of the clustering and for a graph $G = (V, E)$ it is calculated as

$$Q = \sum_{i=1}^{k} \left(\frac{l_i}{|E|} - \left(\frac{d_i}{2|E|} \right)^2 \right), \tag{4.5}$$

where k is the number of clusters, l_i is the number of edges between the nodes in the i^{th} cluster, $|E|$ is the total number of edges and d_i is the sum of the degrees of the nodes in the i^{th} cluster. The modularity can be regarded as the difference between the fraction of all edges that fall into the constructed clusters and the fraction that would do so if the graph nodes were randomly connected defined in [67]. That means higher modularity value indicates more optimal clustering of graphs. Networks with high modularity have strong connections between the nodes in the same cluster but weak connections between the nodes from different clusters.

By taking packet header values pairwise, after applying the clustering algorithm mentioned above, modularity can be calculated to show the strength of the clusters. In anomaly detection systems, it is expected that modularity value of attack traffic is different than the modularity value of attack–free traffic. This situation is utilized in the proposed deep packet inspection system. In [68], DDoS attack detection module based on the mentioned clustering algorithm and modularity value is proposed. Simulation results show that proposed algorithm can detect DDoS attacks effectively.

4.1.2. Graph–Based Features

In graph–based feature extraction methods, network packet headers are modelled as a graph. Then, connections between these packet headers are analyzed. Packet header values are selected pairwise to obtain graph representation. These constructed graphs may be directed or undirected. In directed graphs, connection direction is taken into account whereas in undirected graphs, it is not. By building both type of graphs, number of features extracted can be increased.

To apply graph–based feature extraction methods, a pair of network packet header values such as source IP addresses and destination IP addresses are taken. Then, connections between them are modelled as a graph. By using this graph, different features can be extracted. These features reflect the relationship between selected packet header values. Different features can be calculated using both types of constructed graphs which are directed and undirected one. An example of a directed graph with four nodes is shown in Figure 4.4. For this graph, as an example, nodes may represent IP addresses and arrows between nodes represent the connections between IP addresses. It can be said that IP address c sends network packets to IP address a.

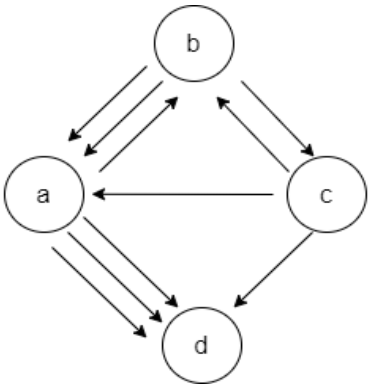

Figure 4.4. A directed graph with four nodes.

In network terminology, arrows in graphs represent a connection between selected packet header values. If a network packet is sent from one side to another, connection arises between them. If only the unique connections between selected network packet headers are considered, the constructed graph is called unweighted graph. On the other

hand, if the network packets transmitted between sides are considered, the constructed graph is called weighted graph. According to the selection of the type of the graph, different features can be extracted.

4.1.2.1. In degree. In degree values are calculated using the directed graph constructed by selecting two packet header values. In the graph, for each node, in degree refers to the number of nodes that start connection with this node. In other words, in degree is the number of nodes sending packets to the selected node. For example, in Figure 4.4, node a has an in degree value of two because nodes b and c send packets to it. Also, node d takes packets from nodes a and c which makes its in degree value two. If IP addresses are taken to construct graphs, in degree refers to the number of IP addresses that send packets to the selected IP address. This value is calculated for all nodes in the graph.

4.1.2.2. In degree weight. In degree weight is the weighted type of in degree value. In the calculation of in degree feature, unique connections are considered. However, for the in degree weight, how many times these connections appear are taken into consideration. In network graphs, it refers to the number of packets that selected node takes from other nodes. For the example given in Figure 4.4, in degree weight of node a is three. Also, in degree weight of node d is four. Compared to in degree value, how many times the connections appear are considered.

4.1.2.3. Out degree. The out degree is the inverse version of in degree. In the constructed graph, for each node, out degree represents the number of nodes that the selected node sends packets. By taking IP addresses as an example, out degree refers to the number of IP addresses that a specific IP address send packets to. In the graph given in Figure 4.4, out degree for node a is two because it sends packets to nodes b and d. Out degree is the reverse version of in degree where the direction is reversed.

4.1.2.4. Out degree weight. Out degree weight is the weighted version of out degree. Instead of unique connections, number of packets are considered. As an example given in Figure 4.4, out degree weight of node a is four.

4.1.2.5. Node betweenness centrality. Node betweenness centrality is considered as a measure of node centrality in a graph. To calculate node betweenness centrality, all of the shortest paths between each node pair in the graph are evaluated. The number of shortest paths passes through the specific node corresponds to the betweenness centrality of this node. This feature can be considered as a measure of loyalty of nodes to the network. A node having high betweenness value shows that this node has remarkable control power over the information transmitted between other nodes. Node betweenness centrality is given as

$$N_B(v) = \sum_{s \neq t \neq v \in V} \frac{\sigma_{st}(v)}{\sigma_{st}}, \tag{4.6}$$

where V is the set of all nodes in the graph, σ_{st} denotes the total number of shortest paths from node s to node t and $\sigma_{st}(v)$ denotes the total number of shortest paths passing through node v. Directed graph is considered when calculating node betweenness centrality values. As an example given in Figure 4.5, node betweenness centrality for node a can be calculated as $N_b(a) = \frac{3.5}{6} \approx 0.583$.

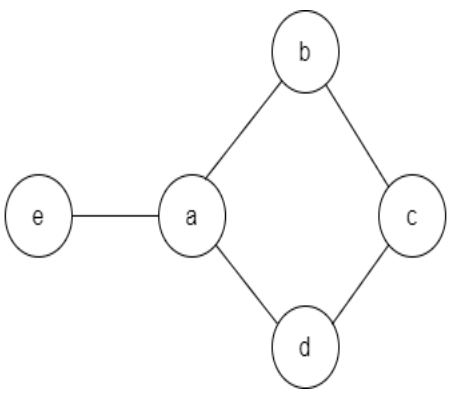

Figure 4.5. Demonstration of node betweenness centrality.

4.1.2.6. Eigenvector centrality:. In graph theory, one of the measures which shows the influence of a node in a network is called eigenvector centrality. It can be considered as the relative weight of a node in the graph. Eigenvector centrality value of a node can be called score of a node. The principle is that connections to high–scoring nodes contribute more to the score of the node than low–scoring nodes. This scoring system implies that a node having high score is connected to lots of nodes having high score as well. To calculate eigenvector centrality, adjacency matrix, $A = (a_{s,t})$, can be constructed as

$$a_{s,t} = \begin{cases} 1, & \text{if node } s \text{ is linked to node } t, \\ 0, & \text{if node } s \text{ is not linked to node } t. \end{cases} \tag{4.7}$$

Eigenvector centrality scores are calculated for each node by using eigenvectors of the matrix defined in (4.7). The relative score of a node s is the s^{th} value of the related eigenvector of the adjacency matrix defined in [69]. Then, the centrality score for node s can be calculated as

$$x_s = \frac{1}{\lambda} \sum_{t \in M(s)} a_{s,t} x_t, \tag{4.8}$$

where $M(s)$ is the set of neighbors of node s, x_t is the related eigenvalue of node t and λ is a constant. It is seen that (4.8) can also be written as

$$A\boldsymbol{x} = \lambda \boldsymbol{x}, \tag{4.9}$$

where $\boldsymbol{x}$ is an eigenvector of the adjacency matrix A. In general, there will be many different eigenvalues λ for which a nonzero eigenvector solution exists. Perron–Frobenius theorem given in [70] guarantees that if all the components of the eigenvector are demanded positive, then there is only one eigenvalue that satisfies this requirement. Thus, a centrality score can be assigned to each node.

Graph–based features are calculated for each node in the constructed network graph. By using these features, different algorithm can be done such as clustering of IP addresses. Then, in packet–based analysis, features of network windows can be calculated. As an example given in [71], graph–based features are used to cluster the IP addresses. Then, to measure the strength of the clustering, entropy of cluster sizes is considered. This work is implemented in DDoS attack detection schemes. It is shown that attack and attack–traffic are differentiated from each other with the help of graph–based features. Like this work, graph–based features may be helpful in different deep packet inspection systems.

4.1.3. Information Theory Based Features

Information theory based features can be divided into two categories which are entropy and Greedy distance. After collecting packets into windows by predefined window length or time series analysis, feature extraction is performed. The importance of using information theory based features in network traffic characterization is that these features reflect the behavior of packet headers effectively. The change in the value of these features corresponds to different activities such as anomalous traffic. Therefore, these features are used widely in network traffic characterization modules.

4.1.3.1. Entropy. Entropy value of packet headers represents how much randomness a packet header value has. Details of the entropy concept are given in Chapter 3.1. To calculate the entropy of a packet header, first all packets within a window are gathered. Then, unique elements of a packet header are found. There are two methods for constructing the probability distributions of packet headers. Firstly, the occurrence of each unique element within a window is calculated. Then, by dividing each element's occurrence value to the total number of packets, the probability distribution of a packet header is constructed. In the second method, instead of counting the number of packets that a unique packet header element is used, only unique connections it has can be considered. These methods can be summarized as the method with the consideration on number of packets and the method with the consideration of unique connections. Both

methods have some advantages and disadvantages. Considering unique connections may be helpful in the detection of some anomalous traffic such as DDoS attacks. Then, by using these probability distributions, entropy values are calculated. The most used entropy values for network traffic characterization are entropy of source IP address, destination IP address, source port number, destination port number and protocol name.

4.1.3.2. Greedy Distance. Greedy algorithm is used to find an upper value for the distance between two probability distributions with different sizes. Details of Greedy algorithm is given in Chapter 3.3. To use Greedy algorithm, two of the packet header values are selected. Then, by constructing their probability distributions, distance between them is calculated. This feature shows the relationship between the selected packet header values. For example, during DDoS attack, there is a dispersion on the source IP addresses and concentration on the destination IP addresses. This different behavior of source and destination IP addresses is highlighted with the usage of Greedy distance.

Different selection of packet header values gives different Greedy distances. These different selections may be helpful for detecting various attack types. For example, while the Greedy distance between source and destination IP addresses is used in the detection of DDoS attack, the Greedy distance between source IP address and destination port number may be helpful in the detection of Port Scan attacks if the attacker uses limited sources. Depending on the type of attack traffic and attacking style, these Greedy distance values are useful for detecting that kind of anomalous traffic. Usually, source IP addresses, destination IP addresses, source port numbers, destination port numbers and packet length are considered when calculating Greedy distances. Packet length is used with the quantization of possible values. If p number of packet header values are considered, it will give $\binom{p}{2}$ Greedy distance values. Depending on the traffic needs to be detected, useful ones can be considered.

Table 4.1. Packet–based Features.

Feature	**Description**
Modularity	Clustering integrity measure
In degree	Number of unique connections packet taken
In degree weight	Number of connections packet taken
Out degree	Number of unique connections packet sent
Out degree weight	Number of connections packet sent
Node betweenness centrality	Measure of connection similarity
Eigenvector centrality	Influence of node in a network
Entropy	Measure of randomness of packet headers
Greedy distance	Distance between packet header pairs

To summarize, list of packet–based features is given in Table 4.1. These features are calculated by gathering the packets using windows. Time–series analysis can be implemented by using these windows. The aim of the extraction of these features is to explore the behavior of packet header values during different network traffic cases.

4.2. Flow–Based Feature Extraction Methods

In network terminology, network flow is defined as the network packets having same source IP address, destination IP address, source port number, destination port number and protocol information. It can be considered as the packets flowing through two communication points. Depending on the protocol used, these packets can be two–way where source and send packets to destination and destination can send packets to the source. In this way, packets within a flow can be considered as forward and backward packets. By constructing the network flows, depending on the network traffic running, connection–based features can be extracted. These features can be used in the characterization of network traffic.

In packet–based analysis, packets running through the network are gathered to extract features reflecting the behavior of network. The difference in the flow–based analysis is that features are extracted for each flow. Each flow is represented by the feature set explained below. Compared to packet–based analysis, this approach may have some advantages such as instead of constructing network windows to analyze the packets, connection specific features are extracted. This situation is helpful when deciding which flows belong to which type of application or which flows containing anomalous network traffic. In packet–based analysis, after deciding a network window containing anomalous traffic, another investigation should be performed to find which connections including anomalous traffic. However, in flow–based analysis, anomalous connections are directly found. The flow–based features can be divided into two categories which are statistical features and payload–based features. Both these categories are explained in the upcoming sections.

4.2.1. Statistical Flow–Based Features

Statistical features are used to highlight the statistical properties of network flows. In these features some attributes are considered such as packet arrival time, packet size, number of packets within a flow etc. By using these attributes, different features are

extracted for each flow. These features are useful in understanding the behavior of network packets within the flow.

4.2.1.1. Inter–arrival Time. First attribute considered in the extraction of flow–based features is the inter–arrival time. Inter–arrival time can be defined as the arrival time difference of consecutive network packets within a flow. A flow consisting of N number of packets can be shown as $F_i = [P_1, P_2, ..., P_N]$, where where F_i represents the i^{th} flow and $P_j,\ j \in \{1, 2, ...N\}$ represents the network packets within the flow. For this flow, the inter–arrival time vector can be shown as

$$\Delta_i = [\delta_{2,1}, \delta_{3,2}, ..., \delta_{N,N-1}], \tag{4.10}$$

where Δ_i denotes the inter–arrival time vector for i^{th} flow. Also, $\delta_{j,j-1}$ represents the arrival time difference between the j^{th} and $j - 1^{th}$ packets. Since inter–arrival time is defined between two consecutive network packets within a flow, for a flow having N number of network packets, there will be $N - 1$ time difference values for this flow. Therefore, a flow is represented by this $N - 1$ arrival time difference valued vector.

To use deep packet inspection methods combined with machine learning or deep learning methods, the network flows must have same number of features. This situation can be achieved in two ways. First way is to limit the number of packets within a flow to the same value for all flows. In this way, if the limit packet number is N, then $N - 1$ valued inter–arrival time vector represents the flow. However, this method causes information lost because packets are running through the network and additional information can be gained about a flow when a new packet comes into it. Therefore, second method can be implemented which is taking minimum, maximum, mean and standard deviation of inter–arrival time vector. In this way, number of packets within a flow do not affect the size of extracted features. For each inter–arrival time vector, same number of features are extracted for each flow. Calling these features iat_{min}, iat_{max}, iat_{mean} and iat_{std}, inter–arrival time vector is used in feature extraction process.

At the beginning of this section, it is mentioned that depending on the type of protocol used, packets within a flow may be in two–ways which are forward and backward. If the network flow starts with connection side A sending packets to connection side B, packets floating from A to B are called forward packets. In a similar manner, packets floating from B to A within a flow are called backward packets. In the feature extraction process, like using all packets within a flow once, forward and backward packets can be analyzed separately. These features give information about the behavior of the network flow deeply. For the forward packets in the flow, by using same statistical attributes, these features are called $fiat_{min}$, $fiat_{max}$, $fiat_{mean}$ and $fiat_{std}$. In a similar manner, for the backward packets, these features are called $biat_{min}$, $biat_{max}$, $biat_{mean}$ and $biat_{std}$.

4.2.1.2. Packet Size. The amount of data transferred between two connection points is important when characterizing the network traffic. According to the usage of network packets, for example in streaming traffic or in network attack traffic, size of packets may differ. To highlight this property, packet sizes within flows are considered. For i^{th} flow, packet size vector can be shown as

$$L_i = [l_1, l_2, ...l_N], \tag{4.11}$$

where l_j, $j \in \{1, 2, ...N\}$ denotes the size of j^{th} packet in the flow, and L_i denotes the packet size vector of i^{th} flow. As in the previous feature, to use this feature independent of the number of packets within a flow, minimum, maximum, mean and standard deviation of this packet size vector are used. These features are called $byte_{min}$,$byte_{max}$,$byte_{mean}$ and $byte_{std}$. Since the packets within a flow can be labelled as forward and backward, packet size feature can be derived also for forward and backward packets. In this way, feature set is increased. These derived features are called $fbyte_{min}$, $fbyte_{max}$, $fbyte_{mean}$, $fbyte_{std}$ for forward packets and $bbyte_{min}$, $bbyte_{max}$, $bbyte_{mean}$, $bbyte_{std}$ for backward packets.

Apart from the packet sizes, the pace of data traffic can be considered. For example, flows regarded to gaming have larger packets sent in short time intervals. To use the data transmission rate property for flows, summation of packet sizes is divided by the duration of the packets. It can be calculated as

$$byte_{sec} = \frac{\sum_{j=1}^{N} L_j}{\sum_{j=2}^{n} \delta_{j,j-1}}, \tag{4.12}$$

where v_i denotes the data transmission rate for i^{th} flow, numerator part correspond to the summation of packets sizes and denominator part corresponds to the total duration of N packets. Total duration is the summation of inter–arrival times of the packets within a flow. It can also be calculated for forward and backward packets, denoted as $fbyte_{sec}$ and $bbyte_{sec}$ respectively.

4.2.1.3. Packet Arrival Pace. For the last feature, packet arrival pace is considered. It demonstrates how fast one packet is transferred by considering all packets within a flow. For the i^{th} flow, packet arrival pace can be calculated as

$$pac_{sec} = \frac{N}{\sum_{j=2}^{n} \delta_{j,j-1}}, \tag{4.13}$$

where numerator N is the number of packets within a flow and denominator is the total duration of N packets. This feature shows how fast one packet is arrived into the flow. As in the previous features, for the forward and backward packets, this feature can be called as $fpac_{sec}$ and $bpac_{sec}$ respectively.

To summarize, total 30 flow–based statistical features are extracted for classification process. These features are calculated by using all packets within a flow, forward packets within a flow and backward packets within a flow. Therefore, same feature is calculated three times for three cases. The aim is to analyze packets within a flow more deeply. Summary of flow–based statistical features calculated using all packets within a flow are shown in Table 4.2.

Table 4.2. Flow–based Statistical Features.

Feature	**Description**
iat_{min}	Minimum value of inter-arrival time vector
iat_{max}	Maximum value of inter-arrival time vector
iat_{mean}	Mean value of inter-arrival time vector
iat_{std}	Standard deviation value of inter-arrival time vector
$byte_{min}$	Minimum value of packet size vector
$byte_{max}$	Maximum value of packet size vector
$byte_{mean}$	Mean value of packet size vector
$byte_{std}$	Standard deviation value of packet size vector
$byte_{sec}$	Data transmission pace
pac_{sec}	Packet arrival pace

4.2.2. Payload–Based Features

In network terminology, the payload represents the actual data carried on the packet. In other words, it can be defined as the actual part of the data that is intended to be transferred. On the packets transferred with the connection created between the source and destination, there is information that does not belong to the transferred data such as IP addresses of the source and destination, port numbers, protocol names but provide communication and recognition. The data to be transferred other than this information is called payload.

New features can be derived by using the payload portions of the packets within flows. These features are expected to behave similar for the network traffic created with the same purpose. In this way, they can be used as distinguishing features in the classification of different kind of network traffic. Payload–based network traffic characterization methods classify the network traffic using pure application layer payload information, excluding the packet header information of the network traffic. These packet header information are source and destination IP addresses, source and destination port numbers and protocol name.

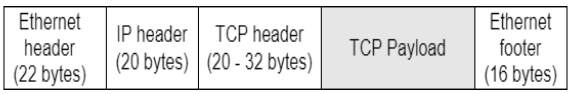

Figure 4.6. Representation of a network packet.

Nowadays, it is no longer convenient to classify the network traffic by using packet header information containing well–known port number or IP addresses, as many devices use private or dynamic IP addresses and changeable port numbers. Payload–based network traffic characterization methods overcome the problem of IP address and port number dependency, as they are not affected event if the packet header information changes. A simple representation of TCP packets in the network traffic is given in Figure 4.6. On this Figure, the payload part is called data. In the payload–based network traffic characterization methods, the parts containing the header information in a packet are not used.

A flow containing N number of packets can be demonstrated as

$$F_i = [P_1, P_2, ..., P_N], \tag{4.14}$$

where F_i represents the i^{th} flow, and P_j, $j \in \{1, 2, ...N\}$ represents the j^{th} packet in this flow. In a similar way, for each packet P_j, the payload portion of it can be demonstrated as

$$P_j = [b_1^j, b_2^j, ..., b_m^j], \tag{4.15}$$

where $b_k \in [0, 255]$ represents the k^{th} byte in the payload of the j^{th} packet. Also, the value of m represents the total byte that is taken from the packet P_j's payload.

During the extraction of payload–based features, byte values is the payload can be taken single, pair, triple etc. The process of taking how many bytes together is called N–gram analysis of payload. This method is usually used in the analysis of language characteristics in the field on NLP. N represents how many elements are taken together into the analysis. For example, if N equals one, it is called unigram and each element

only contains one byte value. Also, if N equals two, it is called bigram analysis. In this method, two byte values are taken to construct elements. Depending on the parameter N, the size of elements constructed using the payload changes. Higher value of N means that there are more number of elements to be analyzed. However, if this number increases, complexity of feature extraction methods will increase. Therefore, careful selection of N value should be determined previously before implementing the proposed system.

Payload–based features are extracted for each packet within a flow. As shown in (4.15), a network packet in a flow has m number of bytes. The values of these bytes are in the interval $[0, 255]$. To extract the payload–based features, histogram of payload is constructed. Then, this histogram is turned into a probability distribution. By using methods based on information theory given in Chapter 3 on the constructed probability distributions, different payload–based features are extracted. In the process of constructing probability distribution based on histogram of payload, the important parameter is the value of N denoting how many bytes are taken together to be analyzed. If unigram ($N = 1$) analysis is performed, each byte is treated as individually. Therefore, since values of bytes are in the interval $[0, 255]$ the histogram of payload has length 256. On the other hand, if bigram ($N = 2$) analysis is performed, bytes are taken pairwise with the overlapping window with step size one running through the payload. In this case, histogram of payload has length 256^2 because there are 256^2 combinations of all possible byte values pairwise. Compared to unigram case, size of histogram increases. To illustrate the process with another example, consider the trigram analysis where $N = 3$. In this case, each element has three byte values. Therefore, the number of possible byte values are 256^3. Also, for the cases where $N \geq 3$, the step size of overlapping window is another parameter to be considered during the construction of histogram. For the case of $N = 3$, the step size of window can be one or two. Both selections will be different probability distributions.

To generalize the procedure mentioned above, let D be a dictionary of length $l = 256^N$ where N is the parameter of N–gram analysis of payload. Dictionary D can be expressed as

$$D = \{d_1, d_2, ..., d_i, ..., d_l\}, \tag{4.16}$$

where $d_i,\ i \in \{1, 2, ...l\}$ represents a unique element of all possible elements constructed using N–gram analysis. Each d_i can be represented as

$$d_i = [b_1, b_2, ..., b_N], \tag{4.17}$$

where the parameter N determines the length of the element d_i and $b_j \in [0, 255]$ with $j \in \{1, 2, ..., N\}$. If the occurrence number or frequency of an element d_i is represented as O_i, then the comparable or normalized frequency of the element d_i can be expressed as

$$f_i = \frac{O_i}{\sum_{k=1}^{l} O_k}, \tag{4.18}$$

where f_i is the normalized frequency of element d_i. Normalized frequency values correspond to probability values of elements constructed using N–gram analysis of payload. In this way, the probability distribution of a payload is constructed. The probability distribution of a k^{th} packet within a flow can be expressed as

$$PD_k = \{f_1, f_2, ...f_i, ...f_l\}. \tag{4.19}$$

Different payload–based features are extracted using the probability distribution of payload given in (4.19) or directly using the dictionary D defined in (4.16). The probability distributions are used in the extraction of features based on information theory. Also, different type of features such as printable character ratio can be extracted using the constructed dictionary D directly.

4.2.2.1. Entropy–Based Features. Entropy is used as a measure of randomness of probability distributions. It shows how much concentrated or dispersed a probability distribution is. In payload–based feature extraction method, entropy measure the randomness of byte values of payload. If similar bytes are used within the payload repeatedly, its entropy value will be low. On the other hand, if many different byte values are used, entropy value of the payload will be high. Detailed analysis of entropy is given in Chapter 3.1. To extract the entropy of payload, Shannon entropy given in (3.1) is used as

$$H(PD_k) = -\sum_{i=1}^{l} f_i \log(f_i), \tag{4.20}$$

where $H(PD_k)$ is Shannon entropy of the payload of k^{th} packet within the flow. To calculate the entropy value of the probability distribution of a payload, any entry of the histogram having zero value is discarded from the probability distribution. For each flow, if a packet within the flow contains payload, its entropy value can be calculated. Therefore, as mentioned in Chapter 4.2.1, there will be as many entropy values as the number of packets containing the payload. These entropy values constitute the entropy vector of the flow. To derive same number of features for each flow, minimum, maximum, mean and standard deviation statistics are used again. By taking the minimum, maximum, mean and standard deviation values of the entropy vector, features are extracted. These features can be called $entropy_{min}$, $entropy_{max}$, $entropy_{mean}$ and $entropy_{std}$. Also, as given in Chapter 4.2.1, these features can be extracted for forward and backward packets within a flow. These derived features can be called $fentropy_{min}$, $fentropy_{max}$, $fentropy_{mean}$ and $fentropy_{std}$ for forward packets and $bentropy_{min}$, $bentropy_{max}$, $bentropy_{mean}$ and $bentropy_{std}$ for backward packets.

4.2.2.2. Greedy Distance–Based Features. Greedy algorithm explained in Chapter 3.3, tries to find an upper bound value for the probability distributions with different sizes. Constructed probability distributions of payloads of packets may contain different number of elements since number of unique bytes used in payloads may differ from each

other. Therefore, KL–divergence defined in Chapter 3.2 cannot be applied here. Instead of it, Greedy algorithm can be used to find a distance between probability distributions with different sizes. The aim of the application of Greedy algorithm is to measure the closeness of payloads of packets within a flow. If the packets within a flow have similar payloads, their distance value will be low. However, dissimilar payloads give higher Greedy distance value. Since the distance value is calculated for two probability distributions, for a flow having p number of packets containing payload, there will be $\binom{p}{2}$ Greedy distance values. By taking two probability distributions of payloads of packets within a flow, say k^{th} and l^{th}, Greedy distance between them is given as

$$G_{k,l} = d_{greedy}(PD_k, PD_l), \tag{4.21}$$

where $G_{k,l}$ denotes the Greedy distance between them. By using this approach, for the i^{th} flow, the Greedy distance vector can be shown as

$$\mathbf{G}_i = [G_{1,2}, G_{1,3}, ..., G_{1,p}, G_{2,3}, ..., G_{p-1,p}]. \tag{4.22}$$

This Greedy distance vector is used to extract the payload–based features. As in the previous sections, minimum, maximum, mean and standard deviation values of this vector is taken. Constructed features can be called $greedy_{min}$, $greedy_{max}$, $greedy_{mean}$ and $greedy_{std}$. These features are extracted using all packets having payload within the flow. Also, by using forward packets $fgreedy_{min}$, $fgreedy_{max}$, $fgreedy_{mean}$ and $fgreedy_{std}$ and by using backward packets $bgreedy_{min}$, $bgreedy_{max}$, $bgreedy_{mean}$ and $bgreedy_{std}$ can be extracted.

4.2.2.3. Ratio of Printable Characters. The printable characters are frequently analyzed in encrypted packet classification algorithms. The aim is to classify packets using their payload by deeply investigating the characters. Also, in the network traffic anomaly detection scenarios, randomly generated payloads may include low number of printable characters. This property can be utilized in the detection of network attacks. In most communication scenarios with packet transferring, payload of the packet is

encrypted using the ASCII characters usually between 0 and 255. Only bytes between 32 and 127 correspond to printable characters. They are used to compose text messages. For randomly generated payloads, the printable character ratio corresponds to approximately 37.5% of the entire payload.

To construct the ratio of printable characters features, each element d_i defined in (4.17) is analyzed. Each byte in the element d_i is checked whether it is printable or not according to the rule given as

$$\rho_i = \begin{cases} 1, & \text{if} \quad 32 < b_k < 127 \quad \text{for all} \quad b_k \in d_i, \\ 0, & \text{otherwise}, \end{cases} \tag{4.23}$$

where ρ_i is equal to one if all bytes in the element d_i is printable. After constructing the dictionary of a packet payload defined in (4.16), each element in this dictionary is analyzed. If an element is not printable, its printable character occurrence rate is set to zero. Printable character occurrence rate for an element d_i can be defined as

$$O_{pc_i} = \begin{cases} O_i, & \text{if} \quad \rho_i = 1 \\ , 0, & \text{otherwise}. \end{cases} \tag{4.24}$$

Then, for the payload of the k^{th} packet within the flow, ratio of printable characters can be calculated as

$$R_{pc_k} = \frac{\sum_{i=1}^{l} O_{pc_i}}{\sum_{i=1}^{l} O_i}, \tag{4.25}$$

where numerator is the summation of occurrences of printable elements and denominator is the summation of occurrences of all elements. In this way, for each packet containing payload within the flow, the printable character ratio is defined. The printable character ratio vector is constructed using all printable character ratio values for the packets in the flow. Then, as in the previous features, minimum, maximum, mean and standard deviation statistics are used to extract the features for each flow.

Table 4.3. Payload–based Features.

Feature	**Description**
$entropy_{min}$	Minimum value of payload entropy vector
$entropy_{max}$	Maximum value of payload entropy vector
$entropy_{mean}$	Mean value of payload entropy vector
$entropy_{std}$	Standard deviation value of payload entropy vector
$greedy_{min}$	Minimum value of Greedy distances between payloads
$greedy_{max}$	Maximum value of Greedy distances between payloads
$greedy_{mean}$	Mean value of Greedy distances between payloads
$greedy_{std}$	Standard deviation value of Greedy distances between payloads
$print_{min}$	Minimum value of printable character ratio vector
$print_{max}$	Maximum value of printable character ratio vector
$print_{mean}$	Mean value of printable character ratio vector
$print_{std}$	Standard deviation value of printable character ratio vector

These features can be called $print_{min}$, $print_{max}$, $print_{mean}$ and $print_{std}$. Printable character ratio feature can also be derived for forward and backward packets. They can be called $fprint_{min}$, $fprint_{max}$, $fprint_{mean}$ and $fprint_{std}$ for forward packets, $bprint_{min}$, $bprint_{max}$, $bprint_{mean}$ and $bprint_{std}$ for backward packets respectively.

To summarize, payload–based features extracted using all packets having payload within a flow are given in Table 4.3. These features are grouped into three main categories which are entropy, Greedy distance and printable character ratio. This feature set can be enriched using different N values for the N–gram payload analysis. Each different N value gives whole new feature set. However, increasing the value of N results in the increase of the complexity of feature extraction process. Also, for the values of $N \geq 3$, the step size of window should be defined. Different selections give different features. Also, these features can be defined for forward and backward packets within a flow. Thus, among different combination of features, most useful ones have to be selected.

5. PERFORMANCE EVALUATION OF PROPOSED DPI SOLUTIONS

In this chapter, simulation results of proposed DPI solutions for network anomaly detection and network traffic classification cases are given. The proposed systems are divided into two categories as in Chapter 4 for feature extraction methods such as packet–based solutions and flow–based solutions. In both packet–based and flow–based solutions, network anomaly detection solutions are available. On the other hand, network traffic classification solutions are available for only flow–based case due to the characteristics of network traffic generated by different applications. Before explaining the proposed approaches with the results taken from different data sets, the data sets used in the performance evaluation of these systems will be explained.

5.1. Data Sets

In this section, data sets used to test the proposed approaches are explained. For network anomaly detection, three data sets are used which are Boğaziçi University DDoS Attack data set (BOUN data set), IDS 2012 and IDS 2017 data sets. All data sets include DDoS attack. Also, IDS 2012 and IDS 2017 data sets include different type of attacks such as brute force, port scan, DoS etc. On the other hand, for the network traffic classification case, the application classification data set generated by Boğaziçi University members is used. It includes network traces from multiple popular applications such as WhatsApp, Youtube, Steam etc. All the details about the data sets are given below.

5.1.1. Boğaziçi University DDoS Attack Data Set

In this data set, one victim server on the campus serves as the attack's main hub. The data set was captured in one of the campus's main switch [72]. Therefore, in addition to the DDoS attack traffic, it also includes varied kinds of network traffic. By

using a mirroring technique, network traffic is taken from the port on campus routers. The network traffic recording server receives perfect copies of all incoming and outgoing packets flowing through the mirrored interface thanks to the mirroring operation on router interfaces. Average network traffic pace in terms of packets per second is 1800. There are two different attack scenarios implemented in this data set. One of them use UDP Flood and the other one uses TCP Flood. In all cases, 8–minute network traffic is collected. In each of them, DDoS attack is executed four times with different packet rates. The attack rates for both UDP and TCP flood are 1000, 1500, 2000 and 2500 packets per second. The collection of data is done by using repeatedly 80 s waiting period and 20 s attack period.

This DDoS attack data set includes properties like network–based probing of two–way legal user traffic that has been mixed in with DDoS attack packets. Additionally, it contains attacks of various speeds to aid researchers in developing and testing their intrusion detection methods for a range of attack densities. This data set offers a broad overview of DDoS attacks of the resource depletion category that were gathered from campus network's backbone routers.

5.1.2. IDS 2012 Network Intrusion Data Set

The UNB ISCX IDS 2012 data set given in [7] includes seven days of network traffic consisting of normal and malicious traces. While tree day of the data don't include attack traces, the other four days include different attack traces such as brute force, DoS etc. An important point about the data set is that it includes full packet payloads. The characteristics of the data set can be stated as follows:

- *Realistic network and traffic:* The generated data set do not include any artificial trace insertion that may result in the performance downgrade. The aim is to paint a clearer picture of the actual consequences of network attacks and the related actions taken by workstations.

Table 5.1. Number of Flows of IDS 2012 Data Set.

	Flow Label	**Number of Flows**
Sunday	Normal	106705
	Infiltrating network from inside	9966
Monday	Normal	111160
	HTTP DoS	3130
Tuesday	Normal	232888
	DDoS via IRC	10995
Thursday	Normal	152233
	Brute Force - SSH	4753

- *Labelled data set:* All flows given in the data set is labelled as normal or anomalous. Also, anomalous flows are labelled with the type of the attack such as DoS attack.
- *Total interaction capture:* All network interactions, whether internal or between Local Area Networks (LANs), are taken into account.
- *Complete capture:* Due to the privacy concern, most data sets are not suitable for the usage because of the anonymization or complete removal of packet payloads. This data set completely eliminates the need for any sanitization, maintaining the data set's naturalness in the process by being collected in a controlled testbed environment.
- *Diverse attack scenarios:* Many network attacks are considered with diverse schemes.

To use this data set, days containing both normal and malicious traffic are used. On these days, besides normal traffic, the attacks considered are HTTP DoS attack on Monday, DDoS attack on Tuesday, Brute Force SSH attack on Thursday and infiltrating the network from inside attack on Sunday. Number of flows generated for each day with the type of traffic are given in Table 5.1

5.1.3. IDS 2017 Network Intrusion Data Set

Another data set generated for anomaly detection purposes including different network attacks is given in [8]. This data set, which closely reflects actual real–world data, contains the most recent and benign common attacks. It contains five day of network traffic with four of them including both normal and attack traces. One day consists of fully normal network traces. The properties of the generated data set can be stated as follows:

- *Complete network configuration:* Modem, firewall, switches, routers as well as the existence of different operating systems make up a full network topology.
- *Labelled data set:* All flows whether anomalous or not labelled with normal or the type of attack it includes.
- *Complete interaction:* By using two distinct networks, internet connectivity as well as the communication between internal LANs are covered.
- *Available protocols:* Different type of protocols are available on the network traffic such as HTTP, HTTPs, FTP, SSH etc.
- *Attack diversity:* Most common attacks are executed such as brute force, DoS, DDoS, port scan etc.

To use this data set, days with normal and anomalous traffic are used. These are brute force attacks on Tuesday, DoS attacks on Wednesday and port scan, DDoS and bot attacks on Friday. The number of flows for each day of data is given in Table 5.2.

Table 5.2. Number of Flows of IDS 2017 Data Set.

	Flow Label	**Number of Flows**
Tuesday	Normal	142674
	Brute Force - Patator - FTP	3941
	Brute Force - Patator - SSH	2945
Wednesday	Normal	144935
	DoS GoldenEye	5529
	DoS Hulk	150924
	DoS Slowhttptest	804
	DoS Slowloris	1988
Friday	Normal	76723
	Botnet	393
	DDoS - LOIC	47158
	Port Scan	329

5.1.4. Boğaziçi University Network Application Classification Data Set

The application traffic classification data set generated by Boğaziçi University members is given in [73]. The data set contains network traces from well–known applications such as Facebook, Discord, Telegram, WhatsApp, Zoom etc. It has real–world network traces prepared for the performance testing of proposed network traffic classification algorithms. This data set contains network traffic from 22 different applications. Number of flows of each application are given in Table 5.3.

From Table 5.3 it is seen that while some applications have large number of flows, some of them have small number of flows. To prepare a balanced data set to train the model, some flows of applications having large number of flows are randomly selected. For each simulation, random selection of flows is performed to obtain a more realistic result.

Table 5.3. Number of Flows of Applications.

Amazon Prime	2001	**CyberGhost**	474
Dropbox	510	**Deezer**	760
Discord	646	**Epic Games**	1650
Facebook	477	**Hotspot**	395
ITunes	787	**Microsoft Teams**	559
Proton VPN	557	**Skype**	996
Slack	1364	**Soulseekqt**	2249
Spotify	2129	**Steam**	547
Telegram	374	**Tunnelbear**	15076
Tunneln	3110	**Ultrasurf**	12279
Whatsapp	442	**Zoom**	703

5.2. Performance Metrics

To measure the performance of the proposed systems, well–known performance metrics such as precision, recall, accuracy etc. are used. To illustrate the derivation of these metrics, consider a confusion matrix for 2×2 binary classification case given in Table 5.4.

Table 5.4. Confusion matrix for 2×2 binary classification case.

	Actual: Negative	**Actual: Positive**
Prediction: Negative	True Negative	False Negative
Prediction: Positive	False Positive	True Positive

True Positive (TP) is the number of correctly predicted samples in Positive class. True Negative (TN) is the number of correctly predicted samples in Negative class. False Positive (FP) is the number of wrong predictions of Negative class. In other works, it is the number of samples labelled as Positive while being Negative. In a similar manner, False Negative (FN) it the number of samples labelled as Negative while being Positive. By using these values, performance metrics are calculated which can be shown as follows:

- False Positive Rate (FPR) = $\frac{FP}{\text{All Negative}} = \frac{FP}{TN+FP}$,
- False Negative Rate (FNR) = $\frac{FN}{\text{All Positive}} = \frac{FN}{TP+FN}$,
- True Positive Rate (TPR) = $\frac{TP}{\text{All Positive}} = \frac{TP}{TP+FN}$,
- True Negative Rate (TNR) = $\frac{TN}{\text{All Negative}} = \frac{TN}{TN+FP}$.

The aim of the proposed approaches is to keep TPR and TNR values high while keeping FPR and FNR values low. Besides these metrics, there are some other metrics representing the performance of the proposed solution effectively which are precision, recall, accuracy and F_1–score. These metrics can be calculated as follows:

- Accuracy = $\frac{TP+TN}{\text{All Samples}} = \frac{TP+TN}{TP+TN+FP+FN}$,
- Precision = $\frac{TP}{TP+FP}$,
- Recall = $\frac{TP}{TP+FN}$ = TPR,
- Specificity = $\frac{TN}{TN+FP}$ = TNR,
- F_1–score = $\frac{2}{\frac{1}{\text{Precision}}+\frac{1}{\text{Recall}}} = \frac{2\times\text{Precision}\times\text{Recall}}{\text{Precision}+\text{Recall}}$.

Among these accuracy gives prediction performance of all samples. Precision measures among all positive predictions, how many of them are truly positive. F_1–score is the harmonic mean of precision and recall. Both false positives and false negatives are considered in the calculation of it. As a result, it works well on an imbalanced data set.

5.3. Packet–Based DPI Solutions

In this section, packet–based DPI solutions for network anomaly detection case are given. In packet–based solutions, network packets are gathered in a time–series manner to be analyzed. This gathering process is done using non–overlapping windows. The length of these windows may be time dependent or number of packets dependent. In time dependent case, network window duration is predetermined by using historical knowledge of the analyzed network or by considering the packet arrival rate. The important condition is that duration of the network window should be larger than the execution time of proposed packet inspection method. If this condition is not met, proposed packet inspection system lags behind the ongoing network traffic. On the other hand, length of the network window may be selected using predetermined number of packets. In this case, same problem mentioned before may occur because if the pace of network traffic increases more than expected, proposed packet inspection system may lags behind this traffic. Therefore, careful selection of the length of the network window is an important research point. Also, there are some studies focusing on determining the length of the network window dynamically considering the pace of the network traffic.

5.3.1. Network Anomaly Detection Method Based on Header Information Using Greedy Algorithm

The proposed system uses Greedy algorithm given in Chapter 3.3 as basis to inspect DDoS attacks. In this system, packet–based inspection is performed in which non–overlapping windows are run through the network traffic. In a specified duration, by using the window, network packets are gathered. Then, feature extraction is implemented using these collected packets. After that, by using the pre–trained model using the features, new incoming network window is labelled as anomalous or not. In this system, network window is labelled. It means that network packets constituting the window have anomalous packets or attack–free packets.

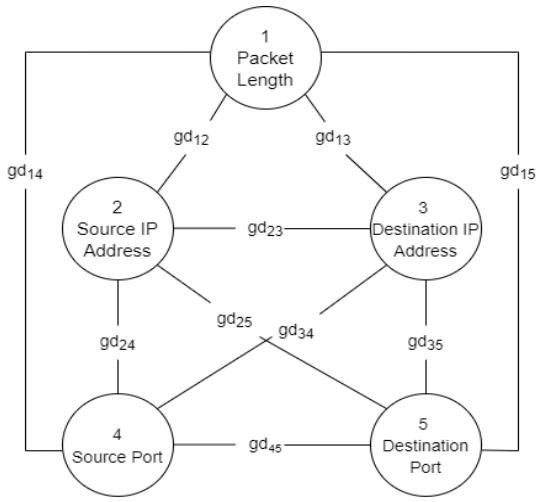

Figure 5.1. Demonstration of the distances between selected packet header values.

To use the Greedy algorithm, packet header values are used. In this system, the distances between packet header values are calculated using the Greedy algorithm. The packet header values used in this system are source IP address, destination IP address, source port number, destination port number and packet length. By taking these packet header values pairwise, distance between each pair of them is calculated. Let $V = \{v_1, v_2, v_3, v_4, v_5\}$ represents the selected packet header values. For example, v_1 may correspond to source IP addresses. By taking a pair of packet header values, probability distributions of selected packet header values are constructed. To construct these probability distributions, unique connections between the selected packet header values are considered. Therefore, probability distribution of i^{th} packet header value depending on j^{th} packet header value can be shown as Pv_{ij}. By using these constructed probability distributions, the Greedy distance between them is calculated as

$$gd_{ij} = d_{greedy}(Pv_{ij}, Pv_{ji}), \tag{5.1}$$

where $i \in \{1, 2, 3, 4, 5\}$, $j \in \{1, 2, 3, 4, 5\}$ and $i \neq j$. Since five different packet header values are selected for the analysis, it gives $\binom{5}{2} = 10$ different distance values. The illustration of the distances between packet header values is shown in Figure 5.1.

For each network window, ten different features are calculated. These features

are calculated to reflect the relationship between packet header values. The aim is to highlight the behavior change in the packet header values when anomalous event occurs. For example, during DDoS attack, if spoofed source IP addresses are used, there will be a dispersion in the source IP addresses side. On the other hand, since anomalous packets tend to go to the same destination, there will be a concentration in the destination IP address side. Therefore, compared to the attack–free case, it is expected that the Greedy distance between source and destination IP addresses increases. Another example is that during port scan attacks, numerous destination ports are checked for availability. Therefore, there will be a dispersion in the destination port number side. If a same source is used for the execution of the attack, the Greedy distance between source IP address and destination port number increases. As in these examples, by using different combination of packet header values, characteristics of different network attacks can be revealed.

The DDoS attack detection method is tested with a data set taken from Boğaziçi University campus network given in [72]. A DDoS attack's primary characteristics include sudden changes in traffic and flow dis–symmetries such as dispersed source IP addresses and concentrated destination IP addresses. Additionally, it has been noted that this attack has an impact on the distribution of port numbers and packet lengths. The difference in probability distribution concentration leads to higher distance values. In contrast, if the probability distributions are similar, the distance measure becomes small. Therefore, distance values are more stable with one another when there is no attack.

From Boğaziçi University data set, simulation results on the UDP Flood data are shown here. Seven of the ten distance results are useful for the detection of DDoS attacks. In different kind of attacks, this number may change. The best indicator of the DDoS attack is the distance between the source and destination IP addresses given in Figure 5.2. Spoofed source IP addresses and most of the packets going through victim IP address is the reason of these results. A similar relationship is observed between source IP addresses and destination port numbers given in Figure 5.3.

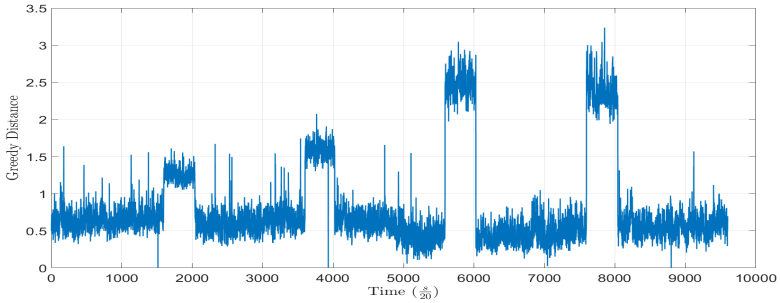

Figure 5.2. The Greedy distance between source and destination IP addresses.

IP addresses and port numbers at the same side tend to move together during the DDoS attack but the limited number of source ports compared to source IP addresses affects results. From Figures 5.4 and 5.5, it is observed that the separation of distance values is not clear like in Figures 5.2 and 5.3.

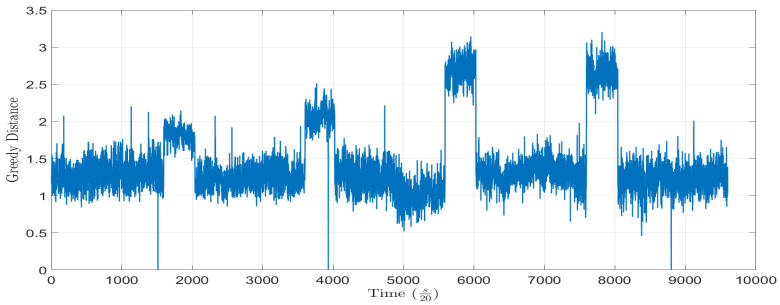

Figure 5.3. The Greedy distance between source IP addresses and destination port numbers

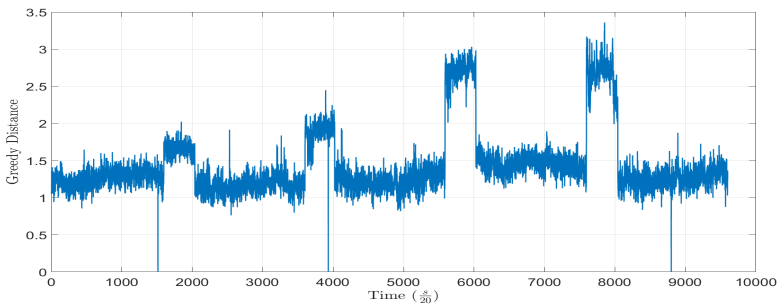

Figure 5.4. The Greedy distance between source port numbers and destination IP addresses

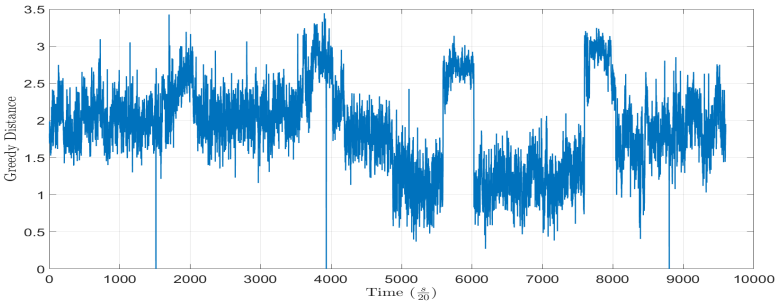

Figure 5.5. The Greedy distance between source and destination port numbers

Table 5.5. Simulation results on BOUN data set with Greedy–Based Approach

Day	**Type of Traffic**	**Accuracy**	**Recall**	**Precision**	**F1 Score**
TCP Flood	Normal	0.9990	0.9992	0.9996	0.9994
	DDoS	0.9990	0.9981	0.9962	0.9971
	Overall	**0.9990**	**0.9990**	**0.9990**	**0.9990**
UDP Flood	Normal	0.9993	0.9991	1	0.9996
	DDoS	0.9993	1	0.9962	0.9981
	Overall	**0.9993**	**0.9993**	**0.9993**	**0.9993**

For the simulation results on Boğaziçi University data set, supervised learning is applied using quadratic SVM kernel. Window size is taken as 50 ms. %70 percent of the data is taken as training, and %30 percent of the data is taken as test data. 10–fold cross–validation is applied in the training phase. Simulation results for the two different scenarios implemented in Boğaziçi University data set is given in Table 5.5. It is observed that proposed system detects generated DDoS attacks in this data set effectively. Only a few samples are miss–labelled. The overall precision rate obtained is 99.98% in UDP Flood and 99.94% in TCP Flood. The strength of the proposed algorithm is measured with its ability to detect even slow rate attacks. The first attack in UDP Flood is the slowest one with the pace of 1000 packets per second. The Greedy distance values are lower compared to other attack cases. Since the average network traffic rate is 1800 packers per second, even in the half rate of it, DDoS attacks are detected. However, if the pace of the attack is lowered much more, the proposed system will have difficulty in the attack detection.

Besides Boğaziçi University data set, the proposed approach is tested with IDS 2012 and IDS 2017 data sets. To execute the simulations on these data sets, window duration is defined with a number of packets. The reason for this situation is that both data sets are prepared for flow–based analysis. Therefore, there are some huge gaps between packets in terms of duration. Therefore, by considering the detection algorithm complexity, 100 packets are taken as a window for the analysis. As in Boğaziçi University data set, quadratic SVM is used as the classification kernel.

IDS 2012 data set contains four days of data combining normal and malicious activities. In this data set, for each day, the number of benign samples are much larger than the number of attack samples. Each day contains malicious traffic from one attack type. Four days of attacks are simulated. The simulation results are given in Table 5.6. Monday data contains HTTP DoS attack. It is almost perfectly detected with the proposed approach. The calculated precision value for this attack type is 99.98%. Tuesday data contains benign traffic with DDoS attack traffic. The accuracy calculated for this case is 93.02% which is lower than expected. Precision rate for the DDoS attack is 90.77%. The performance metrics are above 90% but these are not satisfactory results. Thursday data contains Brute Force SSH attack with benign traffic. The obtained precision value for this attack type is 90.75%. In overall, 97.57% accuracy is obtained. The worst performance is obtained with Infiltration attack on Sunday data. The detection of this type of attack is tough with the proposed packet–based approach. For this attack, 82.67% detection rate is obtained.

IDS 2017 data set contains four days of data combining normal and malicious activities. On Friday, there are DDoS, Port Scan and Botnet attacks. Since in time–series analysis, there are small number of Botnet attack samples compared to other attacks, this class is excluded from the remaining process. Also, on Wednesday, there are different types of DoS attacks. When labelling all DoS attacks into a same class, the proposed approach detects these attacks effectively. On Tuesday data, two different Brute Force attack which are FTP Patator and SSH Patator are combined into one Brute Force class.

Table 5.6. Simulation Results on IDS 2012 data set with Greedy–Based Approach

Day	**Type of Traffic**	**Accuracy**	**Recall**	**Precision**	**F1 Score**
Monday	Normal	0.9998	0.9998	1	0.9999
	HTTP DoS	0.9998	1	0.9846	0.9992
	Overall	**0.9998**	**0.9998**	**0.9998**	**0.9998**
Tuesday	Normal	0.9302	0.9384	0.9454	0.9419
	DDoS	0.9302	0.9291	0.9077	0.9128
	Overall	**0.9302**	**0.9301**	**0.9302**	**0.9302**
Thursday	Normal	0.9757	0.9843	0.9873	0.9858
	Brute Force SSH	0.9757	0.9239	0.9075	0.9156
	Overall	**0.9757**	**0.9756**	**0.9757**	**0.9756**
Sunday	Normal	0.9124	0.8642	0.9893	0.9225
	Infiltration	0.9124	0.9858	0.8267	0.8993
	Overall	**0.9124**	**0.9217**	**0.9124**	**0.9115**

The simulation results for the IDS 2017 data set is given in Table 5.7. On Friday, overall 98.06% precision rate is achieved. For DDoS attacks, the precision rate is 98.44% and for Port Scan attacks the precision rate is 98.68%. On Wednesday, different type of DoS attacks are detected. Overall precision rate is 98.34%. DoS attacks are detected with a 98.2% precision rate. Relatively low performance is obtained on Tuesday data with Brute Force attacks. The obtained precision value for Brute Force attacks is 83.06%.

Table 5.7. Simulation Results on IDS 2017 data set with Greedy–Based Approach.

Day	**Type of Traffic**	**Accuracy**	**Recall**	**Precision**	**F1 Score**
Friday	Normal	0.9817	0.9886	0.9768	0.9827
	DDoS	0.9809	0.9653	0.9844	0.9747
	Port Scan	0.9986	0.9981	0.9868	0.9924
	Overall	**0.9830**	**0.9808**	**0.9806**	**0.9806**
Wednesday	Normal	0.9834	0.9793	0.9851	0.9822
	DoS	0.9834	0.9871	0.9820	0.9846
	Overall	**0.9834**	**0.9835**	**0.9834**	**0.9835**
Tuesday	Normal	0.9048	0.8647	0.9713	0.9149
	Brute Force	0.9048	0.9629	0.8306	0.8919
	Overall	**0.9048**	**0.9112**	**0.9048**	**0.9040**

5.3.2. Clustering–based DDoS Attack Detection Using The Relationship Between Packet Headers

In this system, a clustering algorithm explained in Chapter 4.1.1 is implemented for the detection of DDoS attacks. This clustering algorithm is proposed for analyzing the pair–wise behavior of packet header values. For example, according to the connections between source and destination IP addresses, the IP addresses having similar behavior are gathered into the same group. This process can also be applied to the other packet header values such as port numbers. The aim is to find suspicious packet header values such as attacking IP addresses to identify the anomalous traffic effectively.

The proposed system is a packet–based system where non–overlapping network windows are used for the time–series analysis. As in the previous system, network window is labelled as anomalous or not. The advantage of this proposed system is that by clustering the packet header values according to the connections between each other, when the attack traffic is detected, suspicious packet header values are found. For example, the attacking IP addresses tend to be clustered into one group while

the attack–free IP addresses are dispersed into the other groups. Another advantage of the proposed clustering algorithm is that it does not require any parameters. For example, in k–Means clustering, the k parameter is required to form the clusters. Also, in dBScan method, ϵ parameter is required to define the minimum distance between cluster centers and points. Modularity value measures how well constructed clusters are resemble to each other. If the clusters have similar connections and similar number of elements, the modularity value of the constructed network will be high. However, if there is an outlier cluster like a cluster having much more elements than the other clusters, the modularity value will be low. This property is examined in the proposed attack detection module. During an attack, since attacking packet header values tend to form a cluster seperate from the other clusters, modularity value is expected to decrease.

Non–overlapping network window is run through the network traffic to extract the features and implement the detection algorithm. Window size is taken as 50 ms. Packet header values are selected pair–wise to apply clustering algorithm. Simulations are carried out using BOUN data set including three different DDoS attack scenarios. From the results it is seen that the modularity value of the clusters of source IP addresses constructed using the connections of them with the destination IP addresses gives discriminative results when comparing the attack and attack–free cases. During the attack traffic, attacking IP addresses form a cluster that is different and larger than the other clusters since these IP addresses have the similar behavior that is to go to the same destination IP address. Therefore, the size of this cluster is higher compared to other clusters. This situation lowers the modularity value. Also, in the same way, modularity value decreases during the attack traffic when the clusters of source port numbers are constructed using the connections of them with destination IP addresses. These results can be seen from Figures 5.6 and 5.7.

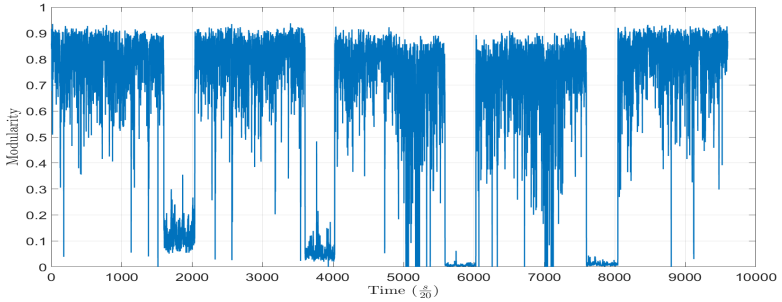

Figure 5.6. The modularity value obtained using source and destination IP addresses.

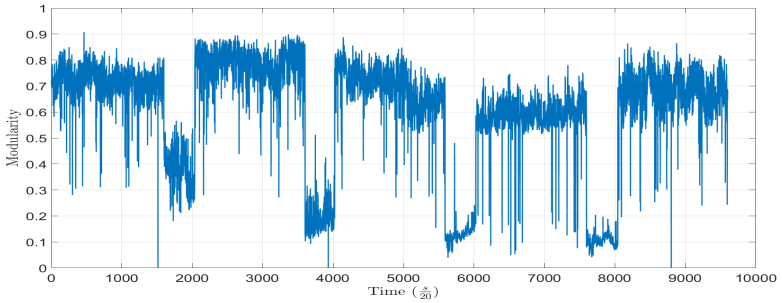

Figure 5.7. The modularity value obtained using source port numbers and destination IP addresses.

Simulation results are shown using true positive rate, false positive rate, accuracy and precision values. Two attack scenarios, UDP Flood and TCP Flood, are evaluated using the proposed algorithm. For each case, 70% percent of data is taken as training and 30% of data is taken as test. 10–fold cross–validation method is applied in training phase. For the kernel of the training model, quadratic SVM is selected. Performance results are shown in Table 5.8. Best results are obtained with the UDP Flood attack case. Compared to the Greedy algorithm–based method explained in Chapter 5.3.1, slightly worse performance is obtained. However, the proposed system has many advantages such as no parameter requirement in the clustering algorithm and direct detection of anomalous packet header values. The obtained precision value for TCP Flood is 99.86% and for UDP Flood is 99.93%.

Table 5.8. Simulation results on BOUN data set with Clustering–Based Approach.

Day	**Type of Traffic**	**Accuracy**	**Recall**	**Precision**	**F1 Score**
TCP Flood	Normal	0.9986	0.9987	0.9996	0.9992
	DDoS	0.9986	0.9981	0.9943	0.9962
	Overall	**0.9986**	**0.9986**	**0.9986**	**0.9986**
UDP Flood	Normal	0.9993	0.9996	0.9996	0.9996
	DDoS	0.9993	0.9981	0.9981	0.9981
	Overall	**0.9993**	**0.9993**	**0.9993**	**0.9993**

Besides BOUN data set, four days of attack traffic with normal traffic of IDS 2012 data set are simulated. Simulatin results are given in Table 5.9. It is observed that the proposed approach has a high detection rate of HTTP DoS attacks. The precision rate for this attack is 99.34%. Compared to Greedy–based approach given in Chapter 5.3.1, DDoS attack detection rate increases to 92.29%. Also, another improvement is observed on Brute Force SSH attacks with a detection rate of 94.42%. However, infiltration attacks are hard to detect with this proposed approach. The obtained precision value for this attack type is 84.90%.

Three days of IDS 2017 data set are also tested with this proposed approach. Simulation results are given in Table 5.10. In this approach, the detection rate of Port Scan attacks decreases. It is observed that the proposed system is more suitable for the detection of DoS and DDoS attacks. DoS attacks ara detected with 100% precision rate. However, compared to the Greedy–based approach given in Chapter 5.3.1, the detection rate of DDoS attacks decreases to 97.54%. For Port Scan attacks 83.40% precision are is observed. Also, for Brute Force attacks 82.03% precision rate is observed. These results show the deficiencies of the proposed approach. The reason for this low detection rate is that the network window cannot highlight the behavior of the attack traffic perfectly. Since both Port Scan and Brute Force attacks have low pace compared to DoS and DDoS attacks, the windows do not have enough attacks packets to represent their behavior.

Table 5.9. Simulation Results on IDS 2012 data set with Clustering–Based Approach.

Day	**Type of Traffic**	**Accuracy**	**Recall**	**Precision**	**F1 Score**
Monday	Normal	0.9963	0.9941	0.9989	0.9965
	HTTP DoS	0.9963	0.9988	0.9934	0.9961
	Overall	**0.9963**	**0.9963**	**0.9963**	**0.9963**
Tuesday	Normal	0.9455	0.9332	0.9658	0.9492
	DDoS	0.9455	0.9603	0.9229	0.9412
	Overall	**0.9455**	**0.9460**	**0.9455**	**0.9454**
Thursday	Normal	0.9709	0.9521	0.9948	0.9730
	Brute Force SSH	0.9709	0.9940	0.9442	0.9684
	Overall	**0.9709**	**0.9719**	**0.9709**	**0.9709**
Sunday	Normal	0.9127	0.8725	0.9772	0.9219
	Infiltration	0.9127	0.9706	0.8409	0.9011
	Overall	**0.9127**	**0.9189**	**0.9127**	**0.9121**

5.3.3. Graph–based Fuzzy Approach Against DDoS Attacks

In this system, graph–based features defined in Chapter 4.1.2 are used to implement DDoS attack detection module. As in the previous solutions, packet header values are analyzed. By selecting a pair of packet header values, the relationship between them is examined. This relationship is modelled as a graph represented as $G(V, E)$ where V corresponds to the nodes or entities and E corresponds to the connections between the nodes. After the construction of graph, several graph–based features mentioned in Chapter 4.1.2 are calculated for each node of the graph. These features are in degree, in degree weight, out degree, out degree weight, node betweenness centrality and eigenvector centrality. They are chosen in order to distinguish attack traffic from benign traffic more efficiently.

Table 5.10. Simulation Results on IDS 2017 data set with Clustering–Based Approach.

Day	**Type of Traffic**	**Accuracy**	**Recall**	**Precision**	**F1 Score**
Friday	Normal	0.9760	0.9816	0.9732	0.9774
	DDoS	0.9621	0.9268	0.9754	0.9505
	Port Scan	0.9839	0.9933	0.8340	0.9067
	Overall	**0.9715**	**0.9623**	**0.9610**	**0.9607**
Wednesday	Normal	0.9946	1	0.9884	0.9942
	DoS	0.9946	0.9901	1	0.9950
	Overall	**0.9946**	**0.9947**	**0.9946**	**0.9946**
Tuesday	Normal	0.9019	0.8582	0.9750	0.9128
	Brute Force	0.9019	0.9671	0.8203	0.8877
	Overall	**0.9019**	**0.9097**	**0.9019**	**0.9010**

The extracted graph–based features are used to cluster the nodes of the graph. The aim is to observe the behavior of packet header values during normal and malicious traffic by examining the relationship between them. Instead of strictly dividing nodes into groups, Fuzzy C–means clustering developed by Dunn [74] and Jim Bezdek [75] is implemented. In this clustering technique, for each node, there is a probability distribution signifying the closeness of the node to the clusters. Each entry of this probability distribution is the probability of the closeness of this node to the cluster with the selected index. Fuzzy C–means clustering technique is based on the minimization of the cost function given as

$$J_m = \sum_{i=1}^{N}\sum_{j=1}^{C} u_{ij}^{m}||x_i - c_j||^2 \qquad , \qquad 1 \leq m < \infty \tag{5.2}$$

where fuzzification coefficient is represented by m and $(m > 1)$, the membership degree of data instance x_i in the j^{th} cluster is represented by u_{ij}, x_i is the i^{th} sample of d–dimensional measured data, c_j corresponds to the d–dimensional center of the cluster j and $||*||$ is the norm operator which corresponds to the Euclidean distance between

data instance and the cluster center. The data sample number is represented with N and C refers to a predetermined number of clusters. With iterative updating the cluster centers and the membership values for each data instance, the cost function given in (5.2) is minimized. Membership values u_{ij} are updated according to the equation given as

$$u_{ij} = \frac{1}{\sum_{k=1}^{C} \left(\frac{||x_i - c_j||}{||x_i - c_k||} \right)^{\frac{2}{m-1}}} \tag{5.3}$$

and cluster centers c_j are updated according to the equation given as

$$c_j = \frac{\sum_{i=1}^{N} u_{ij}^m x_i}{\sum_{i=1}^{N} u_{ij}^m}. \tag{5.4}$$

This iteration process is terminated when the following condition is satisfied. Termination condition can be defined as

$$|J_{m+1} - J_m| < \epsilon \tag{5.5}$$

where ϵ is a termination criterion between 0 and 1 and m is the number of iteration steps. With this method, the algorithm converges to a local minimum or a saddle point of J_m. As a result of the process, the membership matrix U with size $N \times C$ given as

$$U = \begin{bmatrix} u_{11} & u_{12} & \cdots & u_{1C} \\ u_{21} & u_{22} & \cdots & u_{2C} \\ \vdots & u_{ij} & \ddots & \vdots \\ u_{N1} & u_{N2} & \cdots & u_{NC} \end{bmatrix} \tag{5.6}$$

is obtained. Each row corresponds to the data instances and each column represents the clusters. The probability of the i^{th} data sample being in the j^{th} cluster is given in the element u_{ij}. Most clustering techniques assign each data instance directly to one cluster. However, in Fuzzy C–means clustering, the membership values specified

in (5.6) are used to determine the likelihood that a data instance would be in a cluster. This value identifies how close a data instance is to a cluster. Due to the fact that the Fuzzy C–means clustering algorithm provides information regarding whether the data instances are easily separable or not, it may be more effective to employ this algorithm than more conventional clustering algorithms that just provide binary decisions. After the membership matrix U is created, for each data instance, entropy of membership probability distribution is calculated by using the belonging probabilities of the data instance to a cluster. Entropy of membership probability distribution can be calculated as

$$H_i = \sum_{j=1}^{C} u_{ij} \log \frac{1}{u_{ij}} \qquad , \qquad \forall i = 1, 2, ..., N. \tag{5.7}$$

Entropy of membership probability distribution is calculated for each node in the graph $G(V, E)$. High value of entropy value shows the tendency of the nodes to go all clusters. On the other hand, low value of entropy shows the tendency of nodes to go similar clusters. For example, in DDoS attack case, most source IP addresses behave similarly by going through a same destination IP address. Therefore, when all source IP addresses in the network within a window are considered during attack traffic, most of them will go to the same cluster. This lowers the entropy value of a membership probability distribution for each node. For a network window having N number of nodes, there will be N number of membership probability distribution entropy values. Thus, the entropy vector of the window can be shown as $\mathbf{H} = [H_1, H_2, ..., H_N]$. The distribution of this entropy vector is analyzed by using its histogram.

To highlight the differences of the entropy vectors between attack and attack–free traffic cases, skewness value is used. Skewness can be defined as distortion or asymmetry from a normal (Gaussian) distribution. The normal distribution has zero skewness value. A curve is said to be skewed if its density is accumulated on the left or right side. In a negative skew case, the mass of the distribution is accumulated on the right side. It means that the left tail is longer. Conversely, the mass of the distribution

is accumulated on the left side in a positive skew case. For a random variable X, skewness can be calculated as

$$\gamma = \mathbf{E}[(\frac{X-\mu}{\sigma})^3] = \frac{\mathbf{E}[(X-\mu)^3]}{(\mathbf{E}[(X-\mu)^2])^{\frac{3}{2}}} \tag{5.8}$$

where μ is mean and $\mathbf{E}$ is expectation operator. For a distribution of n values, skewness can be calculated as

$$\gamma = \frac{\frac{1}{n}\sum_{i=1}^{n}(x_i-\overline{x})^3}{[\frac{1}{n-1}\sum_{i=1}^{n}(x_i-\overline{x})^2]^{\frac{3}{2}}} \tag{5.9}$$

where $\overline{x}$ is the mean value of the distribution.

During attack–free traffic, the entropy values of selected packet header values are expected to be similar to each other. Therefore, this makes the distribution of entropy values becoming closer to uniform distribution. It can be said that skewness values will be closer to zero in attack–free traffic because the distribution of entropy values is closer to normal distribution. On the other hand, during attack traffic, attacking IP addresses or port numbers tend to behave together. Therefore, considering the entropy vector of selected packet header values, lower values dominate this entropy vector. Therefore, it can be said that the distribution of entropy vector in attack traffic resembles the Gamma distribution. Thus, skewness value of it will differ from zero. This property is utilized in the detection of attack traffic.

The proposed approach is tested on BOUN data set. Two different DDoS attack scenarios are tested which are UDP Flood and TCP Flood. The simulation results are given in Table 5.11. It is observed that good detection rates are achieved for both scenarios. For TCP Flood, 99.71% precision rate is achieved. For UDP Flood, 99.86% precision rate is achieved.

Table 5.11. Simulation results on BOUN data set with Graph–Based Approach.

Day	**Type of Traffic**	**Accuracy**	**Recall**	**Precision**	**F1 Score**
TCP Flood	Normal	0.9971	0.9971	0.9974	0.9972
	DDoS	0.9971	0.9971	0.9967	0.9969
	Overall	**0.9971**	**0.9971**	**0.9971**	**0.9971**
UDP Flood	Normal	0.9986	0.9987	0.9987	0.9987
	DDoS	0.9986	0.9986	0.9986	0.9986
	Overall	**0.9986**	**0.9986**	**0.9986**	**0.9986**

Another data set used for the simulations is IDS 2012 data set. The simulation results of this data set are given in Table 5.12. As in the previous packet–based methods, DoS attacks are detected with a high accuracy. The obtained precision value for DoS attacks is 99.57%. To compare the packet–based methods which are Greedy–based one given in Chapter 5.3.1 and Clustering–based one given in Chapter 5.3.2, best detection rate for DDoS attacks is achieved with this approach. The obtained precision rate for DDoS attacks is 92.49%. Also, for Brute Force SSH attacks, among the packet–based methods, this approach gives best precision value which is 96.60%. For infiltration attack, low performance is observed like the other packet–based methods with 83.84% precision rate.

IDS 2017 is the another data set used in the simulations of the proposed approach. The simulation results are given in Table 5.13. From Wednesday data, it is observed day DoS attacks are detected with a precision value of 99.90%. For all packet–based methods, it is seen that DoS attack detection rate is high compared to other attacks. Also, by using this approach, Brute Force attacks are detected with a 82.24% precision rate which is higher than the Clustering–based approach but lower than the Greedy–based approach. For Port Scan attacks, 85.04% precision rate is achieved which is the best among the proposed packet–based algorithms. For DDoS attacks, 98.45% precision rate is obtained. It is also seen that for each day, benign samples are detected with a high precision rate which are 99.47% for Friday, 99.96% for Wednesday and 97.87% for Tuesday.

Table 5.12. Simulation Results on IDS 2012 data set with Graph–Based Approach.

Day	**Type of Traffic**	**Accuracy**	**Recall**	**Precision**	**F1 Score**
Monday	Normal	0.9959	0.9961	0.9961	0.9961
	HTTP DoS	0.9959	0.9957	0.9957	0.9957
	Overall	**0.9959**	**0.9959**	**0.9959**	**0.9959**
Tuesday	Normal	0.9416	0.9342	0.9566	0.9453
	DDoS	0.9416	0.9503	0.9249	0.9374
	Overall	**0.9416**	**0.9418**	**0.9416**	**0.9416**
Thursday	Normal	0.9790	0.9701	0.9906	0.9802
	Brute Force SSH	0.9790	0.9893	0.9660	0.9775
	Overall	**0.9790**	**0.9792**	**0.9790**	**0.9789**
Sunday	Normal	0.9104	0.8706	0.9750	0.9198
	Infiltration	0.9104	0.9678	0.8384	0.8985
	Overall	**0.9104**	**0.9165**	**0.9104**	**0.9097**

5.4. Flow–Based DPI Solutions

In flow–based DPI solutions, network packets are gathered into network flows by using 5–tuple packet header information which are source IP address, destination IP address, source port number, destination port number and protocol. Flow–based approach has many advantages such that each flow is evaluated using its own characteristics. Depending on the type of network traffic, characteristics of flows differ from each other. In packet–based analysis, whole network is analyzed with time windows. These windows reflect the behavior of the ongoing network traffic instantaneously. On the other hand, in flow–based analysis, each flow is evaluated independently from other flows. Extracted features differ for different type of flows. In the proposed solutions, statistical features like data transmission rate, size of packets etc. are combined with the features extracted from payload portion of packets. The contribution of these solutions is that novel payload–based features such as the application of Greedy algorithm on payload, improve the performance of flow–based statistical features.

Table 5.13. Simulation Results on IDS 2017 data set with Graph–Based Approach.

Day	**Type of Traffic**	**Accuracy**	**Recall**	**Precision**	**F1 Score**
Friday	Normal	0.9720	0.9524	0.9947	0.9731
	DDoS	0.9818	0.9456	0.9845	0.9646
	Port Scan	0.9620	0.9878	0.8504	0.9140
	Overall	**0.9721**	**0.9591**	**0.9579**	**0.9569**
Wednesday	Normal	0.9993	0.9991	0.9996	0.9994
	DoS	0.9993	0.9996	0.9990	0.9993
	Overall	**0.9993**	**0.9993**	**0.9993**	**0.9993**
Tuesday	Normal	0.9048	0.8600	0.9787	0.9155
	Brute Force	0.9048	0.9719	0.8224	0.8909
	Overall	**0.9048**	**0.9129**	**0.9048**	**0.9039**

There are two important parameters must be considered in the proposed flow–based system. First parameter is that how many packets should be considered within flows. In other words, this parameter defines when the decision about the classification is done about the flow. For example, if five packets are considered, when a flow reaches five packets, it goes to the classification stage and then labelled accordingly. This parameter should not be constant because for some applications, predefined number of packets may not be enough to represent the flow. In these situations, the probabilities of the flow closeness to each class taken from the prediction of training model can be considered. If one of the probabilities reaches that threshold, classification can be performed. However, this approach may also suffer from the situation that desired probability threshold cannot be reachable even after many packets transmitted for the flow. Therefore, the selection of number of packets parameter or the probability threshold parameter should be done carefully to maximize the detection accuracy. Another parameter is to be considered is the number of bytes of the payload. Using small portion of payload may not represent the flows effectively. On the other hand, using whole payload may increase the complexity of the detection algorithms. While some packets have large payloads, some of them have small ones. Selecting the same number of bytes for all flows requires zero–padding for the packets with small payloads.

On the other hand, if a small number of bytes is selected, cropping should be applied to the packets with large payloads. These situations affect the performances of the proposed solutions. Therefore, this parameter is also should be selected carefully or whole payload should be used.

Two flow–based solutions are proposed which are for network anomaly detection and network application classification. The performance of proposed solutions are shown using publicly available data sets. Both systems have similar architecture. However, they are some different requirements between them. For example, in network anomaly detection system, the important condition is to detect the attack as soon as possible. Therefore, it should be done using small number of packets within flows. On the other hand, in network application classification system, classification accuracy is the most important consideration. Therefore, larger number of packets can be used compared to anomaly detection systems. To summarize, both systems require high detection accuracy. Besides that, network anomaly detection system requires fast detection to combat with the attack immediately.

5.4.1. Flow–Based Network Intrusion Detection System

In the flow–based approach, network packets are gathered into flows by using their packet header information as explained before. Characteristics of the flows are extracted using the features mentioned in Chapter 4.2. In both approaches for network intrusion detection and for network traffic classification, statistical and payload–based features are used. The simulations are done using the combination of these features and usage of each type of features alone. The block diagram of the flow–based system is given in Figure 5.8. The proposed system consists of three modules which are Pre–processing Module, Feature Extraction Module and Classification Module. These modules are explained below.

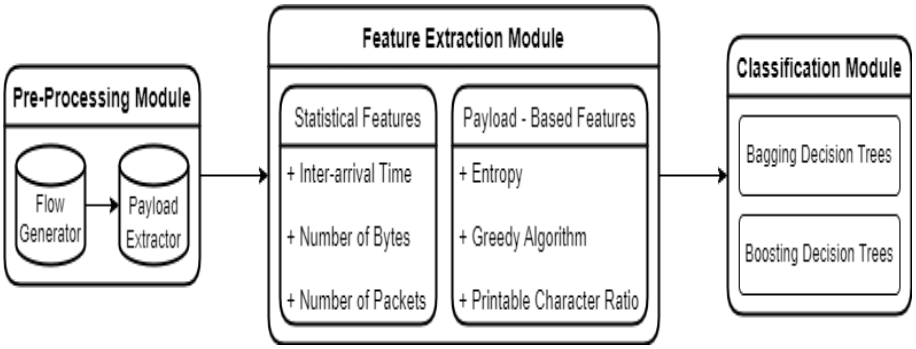

Figure 5.8. The block diagram of the proposed flow–based approach.

In the Pre–processing Module, by using the network traffic, network packets are gathered into flows. The header information and payload parts of packets are extracted. The data is prepared for the execution of Feature Extraction Module. In this approach, historical network traffic is used for the preparation of the training model. The training of the model is performed offline. By using the feature extraction module, flows having more packets than the predefined threshold, are analyzed. These flows are used in the training of the model. In the testing phase, while the network traffic is running, packets are gathered into the flows. When a new packet comes, if it belongs to a existing flow, then it is added to this flow's packet list. However, if this new packet does not belong to any flows, a new flow is created with this packet's header information. If a flow reaches predefined number of packets, feature extraction is performed. Then, this flow is sent to the Classification Module to be predicted its behavior. When the flow reaches this module, there are two options. In the first option, if the closeness probability of flow to a class is high enough, then this flow is labelled with this class. After this labelling, new packets related to this flow are directly sent to the desired location. However, in the second options, if the closeness probability is not enough, then new packets for the flow are waited. After the arrival of new packets, classification is performed again. In this way, reliability of the classification can be increased. However, newly arriving packets may also affect the classification worse. Therefore, all probability values with each incoming packet are saved and then decision is made accordingly.

To verify the effectiveness of this approach, simulations are performed using IDS 2012 and IDS 2017 data sets. The details about the data sets are given in Chapter 5.1.2

and 5.1.3 respectively. These data set are popularly used in network intrusion detection systems. To show the results, each day of each data set is evaluated individually. For example, Wednesday data in IDS 2017 data set consists of benign traffic with multiple types of DoS traffic such as DoS Hulk, DoS Golden Eye etc. The detection of malicious traffic even with the execution type of the attack is performed on these data sets. To take the simulation results, packets having payloads are selected. Statistical and payload–based features are extracted using the packets having payloads. Also, the header information such as IP addresses or port numbers are removed from the payloads to make the analysis more reliable. Instead of selecting a portion of payloads, total payloads are used for the feature extraction. The reason for this implementation is that in this approach, payloads are used to build probability distributions by taking histograms of them. For some approaches, deep learning techniques are implemented on payloads to construct new features. To make this, input size has to be constant for each flow. Therefore, a byte number is decided and taken from each payload. If a payload does not have that size, zero–padding is applied. In this proposed approach, this implementation is not necessary.

Another parameter is to be considered is the minimum number of packets within a flow. The feature extraction is performed after a flow reaches that predefined number of packets. This number is set to be five in these simulations. Flows having less number of packets than this threshold is discarded from the classification process. If this threshold value is set to high, many flows have to be discarded from the remaining process. On the other hand, setting this value too low results in the addition of insignificant flows into the classification. Therefore, an optimal value of five is selected for the rest of the simulations.

Simulation results are given by using IDS 2012 and IDS 2017 data sets respectively. The performance of the proposed approach is demonstrated using the performance metrics defined in Chapter 5.2. These values are calculated for the overall system as well as for each class considered in a simulation. Results are shown using statistical features alone, payload–based features alone, and the combination of both

features. The contribution of the proposed payload–based features is demonstrated with the results. For the classification phase, Ensemble Decision Tree methods are used which are bagging and boosting decision trees. These methods can be briefly explained as follows:

- *Bagging Decision Trees:* The idea behind bagging is that more accurate results would be obtained if several trees are trained and the average (or, in the case of classification, the majority vote) of their output is used to predict the label of a new observation. For example, if there are four decision trees, three of them giving same label to a new observation and one of them gives different label, by majority counting the label given by three of them is given to the observation. This is the basic idea behind bagging decision trees. The goal is to divide the training samples, which were selected at random with replacement, into several subsets of data. Each subset of data is now used to train the decision trees of each group. In the end, there will be an ensemble of various models. It is more reliable than using just one decision tree to utilize the average of all the predictions from many trees.
- *Boosting Decision Trees:* Another ensemble method for building a set of predictors is boosting. Building a sequence of trees, each of which is an improved version of previous one, is the concept of boosting. Every tree is created by learning from its previous errors. This method involves teaching learners sequentially, starting with early learners fitting basic models to the data and moving on to later learners checking the data for errors. In other words, successive trees (random sample) are fit with the aim of resolving the net error from the previous tree at each stage. An input's weight is increased when a hypothesis incorrectly classifies it, increasing the likelihood that the subsequent hypothesis will classify it properly. Weak learners can become better performers by merging the entire set at the end.

Table 5.14. Number of flows of Monday data on IDS 2012.

Flow Type	**Number of Flows**
Normal	3240
HTTP DoS	1620

5.4.1.1. Simulation Results on IDS 2012 Data Set. IDS 2012 data set is a well–known data set used in many intrusion detection systems. It includes collected network traffic within duration of a week. Among the days of data set, since three days do not include malicious traffic, they are excluded from the simulations. The other four days are used in simulations. These days include benign traffic with HTTP DoS, Brute Force SSH, DDoS and infiltrating the network from inside attacks traffic. All network traces include full payloads. Simulation results are given day by day. Then, all network traces are gathered to build multi–attack detection system. In the end, this system's performance metrics are given. In all simulations, 70% of data is taken as training and 30% of data is taken as test. 10–fold cross–validation method is applied in training phase. AdaBOOST Ensemble Decision Tree is used as a classification method. Simulation results are given with different feature sets which are statistical features, payload–based features and combination of statistical and payload–based features.

Monday: First day of the analyzed data of IDS 2012 is Monday which includes HTTP DoS attacks with benign traffic. Binary classification is implemented in this case. Since benign traces have huge amount of flow samples, appropriate portion of them is selected for simulations. The distributions of the number of flows after implementing 5–packet threshold for each flow is given in Table 5.14.

Table 5.15. Simulation Results of IDS 2012 - Monday.

Feature Set	**Flow Type**	**Accuracy**	**Recall**	**Precision**	**F1 Score**
Statistical	Normal	0.9952	0.9949	0.9979	0.9964
	HTTP DoS	0.9952	0.9959	0.9897	0.9928
Features	**Overall**	**0.9952**	**0.9952**	**0.9952**	**0.9952**
Payload	Normal	0.9979	0.9969	1	0.9985
Based	HTTP DoS	0.9979	1	0.9938	0.9969
Features	**Overall**	**0.9979**	**0.9979**	**0.9979**	**0.9979**
All	Normal	0.9973	0.9959	1	0.9979
	HTTP DoS	0.9973	1	0.9918	0.9959
Features	**Overall**	**0.9973**	**0.9973**	**0.9973**	**0.9973**

Simulation results with different features sets on Monday of IDS 2012 data set is given in Table 5.15. It is observed that both feature sets are able to discriminate the attack traffic from normal traffic. From the performance metrics for each class as well as the overall system, it is observed that the proposed approach has a good performance on the classification of HTTP DoS attacks. The calculated overall accuracy by using payload–based features is 99.79%. Also, precision value of the attack class is found to be 99.38%. Moreover, since the data set is not balanced, flow numbers are very different for normal and attack cases, F_1–score can be considered. It is found to be 99.79% which shows the good performance of the proposed approach.

Confusion matrix for the Monday data of IDS 2012 data set taken by using all features is given in Figure 5.9. In this matrix, diagonal entries represent true detections. For example, 4 HTTP DoS samples are labelled as benign class. For the benign class, all test samples are correctly labelled. In total, 1454 out of 1458 samples are correctly labelled which corresponds to 99.73% accuracy.

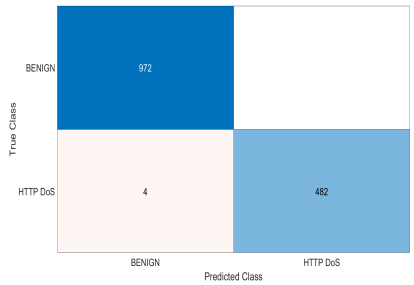

Figure 5.9. Confusion matrix of Monday of IDS 2012.

Table 5.16. Number of flows of Tuesday data on IDS 2012.

Flow Type	**Number of Flows**
Normal	21962
DDoS	10981

Tuesday: This day of the IDS 2012 data set contains network traces from benign and DDoS attack traffic. As in the other days of this data set, binary classification model is trained. Since benign samples have relatively large number of flow samples, appropriate amount of them is selected randomly. The distribution of the number of flows after the random selection of benign samples and application of 5–packet threshold is given in Table 5.16.

Simulation results with selecting different feature sets are given in Table 5.17. Since benign traffic has many flow samples, its size is reduced to get away from the over–fitting problem. The random selection of benign samples process is repeated multiple times to get a more reliable result. The result given in Table 5.17 is the mean of all repetitions. When all features are used, it is observed that the precision value of 99.76% is reached for benign class and 98.91% is reached for DDoS class. In overall, 99.47% accuracy is obtained. Also, it is observed that statistical and payload–based features have similar results. The best performance is obtained when combining both feature sets.

Table 5.17. Simulation Results of IDS 2012 - Tuesday.

Feature Set	**Flow Type**	**Accuracy**	**Recall**	**Precision**	**F1 Score**
Statistical	Normal	0.9928	0.9976	0.9917	0.9946
	DDoS	0.9928	0.9835	0.9951	0.9893
Features	**Overall**	**0.9928**	**0.9929**	**0.9928**	**0.9928**
Payload	Normal	0.9930	0.9966	0.9929	0.9948
Based	DDoS	0.9930	0.9858	0.9933	0.9896
Features	**Overall**	**0.9930**	**0.9930**	**0.9930**	**0.9930**
All	Normal	0.9947	0.9976	0.9945	0.9960
	DDoS	0.9947	0.9891	0.9951	0.9921
Features	**Overall**	**0.9947**	**0.9948**	**0.9947**	**0.9947**

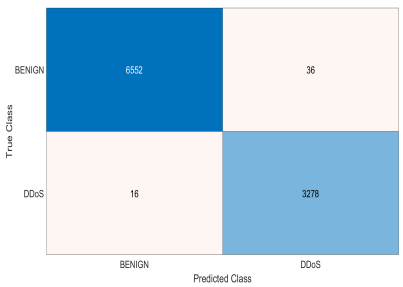

Figure 5.10. Confusion matrix of Tuesday of IDS 2012.

Confusion matrix of Tuesday data of IDS 2012 data set taken with using all features is given in Figure 5.10. It is observed that 16 out of 3294 DDoS samples are labelled as benign and 36 out of 6588 benign samples are labelled as DDoS. In total, 9830 out of 9882 samples are detected correctly which corresponds to 99.47% accuracy.

Thursday: Thursday of IDS 2012 data set includes benign traffic with brute force attack traces implemented on SSH protocol. In this data set, one–day network traffic is classified as binary classification. Number of samples are tried to be balanced using random selection of the samples from the benign class. Also, flows having less than 5 packets are discarded from the remaining process. The distribution of number of flows for each class is given in Table 5.18.

Table 5.18. Number of flows of Thursday data on IDS 2012.

Flow Type	**Number of Flows**
Normal	9420
Brute Force – SSH	4710

Table 5.19. Simulation Results of IDS 2012 - Thursday.

Feature Set	**Flow Type**	**Accuracy**	**Recall**	**Precision**	**F1 Score**
Statistical	Normal	1	1	1	1
	Brute Force - SSH	1	1	1	1
Features	**Overall**	**1**	**1**	**1**	**1**
Payload	Normal	0.9995	0.9996	0.9996	0.9996
Based	Brute Force - SSH	0.9995	0.9993	0.9993	0.9993
Features	**Overall**	**0.9995**	**0.9995**	**0.9995**	**0.9995**
All	Normal	0.9998	0.9996	1	0.9998
	Brute Force - SSH	0.9998	1	0.9993	0.9996
Features	**Overall**	**0.9998**	**0.9998**	**0.9998**	**0.9998**

Simulation results taken with different feature sets are given in Table 5.19. Since random selection is applied for the samples of benign class, the classification process is repeated multiple times and mean of these results are given. Statistical features gives the best result. The precision value of 100% is obtained for benign class and 100% is obtained for Brute Force SSH class. There is a slight performance difference between statistical and payload–based features. With payload–based features, 99.95% accuracy is obtained.

Confusion matrix taken with using all features in Thursday data is given in Figure 5.11. 1412 out of 1413 Brute Force SSH samples are truly detected. Also, all benign samples are correctly labelled. In total, among 4239 samples, 4238 of them are truly detected.

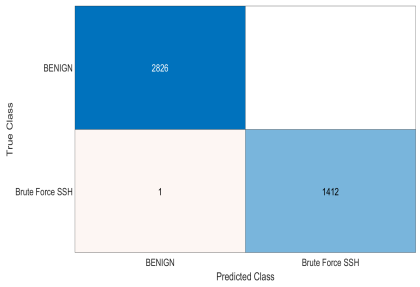

Figure 5.11. Confusion matrix of Thursday of IDS 2012.

Table 5.20. Number of flows of Sunday data on IDS 2012.

Flow Type	**Number of Flows**
Normal	2630
Infiltration	1315

Sunday: Sunday data consists of traces called infiltrating the network from inside and normal activities. As in the previous days, binary classification is performed. To obtain a more balanced data set, appropriate number of flows are selected from benign samples. The distribution of flow numbers of both classes is given in Table 5.20.

Simulations results taken with both feature sets alone and the combination of feature sets are given in Table 5.21 in terms of performance metrics. Similar results are obtained by using these feature sets. When all features are used, for normal classes 96.57% precision is obtained while for infiltration class 100% precision is obtained. Overall accuracy is found to be 97.63%. Statistical and payload–based feature sets give similar results. When they are combined, better performance results are obtained.

Confusion matrix taken with all features on Sunday of IDS 2012 data set is given in Figure 5.12. 28 out of 394 infiltration attack samples are labelled as normal while all normal samples are labelled as normal. In total, among 1183 samples, 1155 of them are truly detected.

Table 5.21. Simulation Results of IDS 2012 - Sunday.

Feature Set	Flow Type	Accuracy	Recall	Precision	F1 Score
Statistical	Normal	0.9704	0.9724	0.9835	0.9779
	Infiltration	0.9704	0.9662	0.9442	0.9551
Features	**Overall**	**0.9704**	**0.9704**	**0.9704**	**0.9703**
Payload	Normal	0.9713	0.9655	0.9924	0.9788
Based	Infiltration	0.9713	0.9839	0.9289	0.9556
Features	**Overall**	**0.9713**	**0.9716**	**0.9713**	**0.9710**
All	Normal	0.9763	0.9657	1	0.9826
	Infiltration	0.9763	1	0.9289	0.9632
Features	**Overall**	**0.9763**	**0.9771**	**0.9763**	**0.9761**

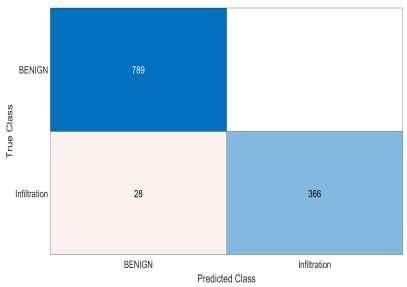

Figure 5.12. Confusion matrix of Sunday of IDS 2012.

Total: In this part, attack samples from all days are gathered to build a multi–class network intrusion detection model. Attack samples from each day are taken with an equal amount of benign samples to build this model. Therefore, benign samples have most flows. The distribution of the number of flows for each class is given in Table 5.22. As in the previous days, same methods are applied such as flows having more than 5 packets are considered.

Table 5.22. Number of flows of IDS 2012 data set with all classes.

Flow Type	**Number of Flows**
Normal	5587
Infiltration	394
HTTP Dos	486
DDoS	3294
Brute Force – SSH	1413

Simulation results of the constructed multi–class classification model is given in Table 5.23. All feature sets give similar performances. To compare them in terms of overall accuracy, statistical features give the best performance with a value of 99.08%. The most miss-labelled class is the infiltration class. Its precision values are low compared to other classes. Although best performance is obtained with statistical features in terms of accuracy, in some cases different feature sets are better. For example, in DDoS class, payload–based features give the best accuracy. Also, in Brute Force – SSH class, all features give the best accuracy. If the detection simplicity of attacks are compared, it is observed that the easiest one is Brute Force – SSH. On the other hand, the toughest one is infiltration attack. DDoS attacks are detected in a better accuracy than HTTP DoS attacks.

Confusion matrix of the multi–class model constructed using all features is given in Figure 5.13. It is observed that for DDoS attacks, 45 samples are labelled as normal. The other samples are correctly labelled. Also, 15 samples of Brute Force – SSH attack are labelled as normal. The most wrong labelling is seen on the Infiltration attack. 97 samples of it are labelled as normal. In general, missed samples of attack traces are labelled as normal. However, it is observed that 16 samples of normal class are labelled as DDoS attack. In total, among 11174 samples 10982 of them are truly labelled which corresponds to 98.88% accuracy.

Table 5.23. Simulation Results of IDS 2012.

Feature Set	Flow Type	Accuracy	Recall	Precision	F1 Score
Statistical Features	Normal	0.9858	0.9795	0.9923	0.9859
	Infiltration	0.9911	0.9773	0.7665	0.8592
	HTTP DoS	0.9988	0.9958	0.9774	0.9865
	DDoS	0.9952	0.9909	0.9927	0.9918
	Brute Force - SSH	0.9978	0.9860	0.9965	0.9912
	Overall	**0.9908**	**0.9843**	**0.9843**	**0.9839**
Payload Based Features	Normal	0.9806	0.9701	0.9918	0.9808
	Infiltration	0.9889	0.9787	0.7005	0.8166
	HTTP DoS	0.9979	0.9833	0.9691	0.9762
	DDoS	0.9956	0.9954	0.9897	0.9925
	Brute Force - SSH	0.9969	0.9832	0.9922	0.9877
	Overall	**0.9881**	**0.9801**	**0.9800**	**0.9791**
All Features	Normal	0.9830	0.9732	0.9934	0.9832
	Infiltration	0.9913	0.9714	0.7766	0.8632
	HTTP DoS	0.9991	1	0.9794	0.9896
	DDoS	0.9926	0.9902	0.9845	0.9874
	Brute Force - SSH	0.9989	0.9986	0.9929	0.9957
	Overall	**0.9888**	**0.9825**	**0.9825**	**0.9820**

5.4.1.2. Simulation Results on IDS 2017 Data Set. IDS 2017 data set is a popular data set extensively used in network intrusion detection research area. It includes five–day network traffic. Days with malicious traffic are used in these simulations. Therefore, network traces from Tuesday, Wednesday and Friday are used in the simulations. Day by day simulation results are given in this section. Also, by combining these days, overall result for the IDS 2017 data set is given. In all simulations, 70% of the data is taken as training and 30% of the data is taken as test. 10–fold cross–validation is applied in the training phase. 3 different simulation results are given which are taken with only statistical features, only payload–based features and combination of statistical and payload–based features.

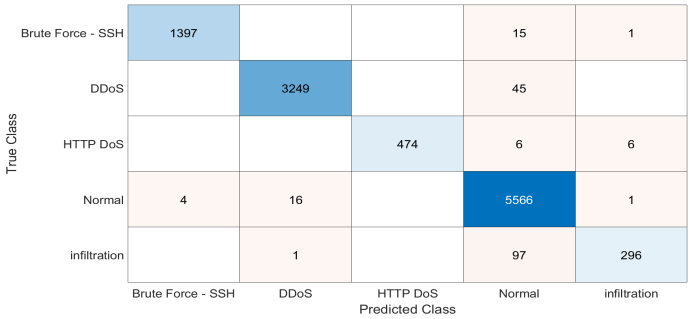

Figure 5.13. Confusion matrix of IDS 2012 data set with all classes.

Table 5.24. Number of flows of Wednesday data on IDS 2017.

Flow Type	**Number of Flows**
Normal	8742
DoS Hulk	7968
DoS Golden Eye	5529
DoS slowloris	1988
DoS Slowhttptest	804

Wednesday: Wednesday data consists of benign traces with malicious traces from different DoS attacks. The types of DoS attacks implemented in this data are DoS Hulk, DoS Golden Eye, DoS slowloris and DoS Slowhttptest. 5–class classification model is trained. Most of the flows are belong to DoS Hulk and benign traffic. To prepare a more balanced data set, appropriate number of samples are taken from Normal and DoS Hulk classes. Then, simulation is performed multiple times to obtain more reliable results. The distribution of the number of flows after implementing 5–packet threshold for each flow in given in Table 5.24.

Simulation results with different feature sets on Wednesday of IDS 2017 data set are given in Table 5.25. It is observed that payload–based features outperforms statistical features with a slight improvement in performance metrics. When these feature sets are combined, there is a slight performance improvement observed. The results are given with different performance metrics defined for each class since the number of flows of classes differ from each other.

Table 5.25. Simulation Results of IDS 2017 - Wednesday.

Feature Set	**Flow Type**	**Accuracy**	**Recall**	**Precision**	**F1 Score**
Statistical Features	Normal	0.9925	0.9996	0.9790	0.9892
	DoS Hulk	0.9798	0.9556	0.9820	0.9686
	DoS Golden Eye	0.9855	0.9714	0.9626	0.9670
	DoS Slowloris	0.9996	0.9983	0.9966	0.9975
	DoS Slowhttptest	0.9995	0.9837	1	0.9918
	Overall	**0.9877**	**0.9788**	**0.9784**	**0.9785**
Payload Based Features	Normal	0.9923	0.9996	0.9783	0.9888
	DoS Hulk	0.9904	0.9719	0.9987	0.9851
	DoS Golden Eye	0.9963	0.9957	0.9873	0.9915
	DoS Slowloris	0.9992	0.9933	0.9966	0.9950
	DoS Slowhttptest	0.9997	0.9918	1	0.9959
	Overall	**0.9934**	**0.9892**	**0.9889**	**0.9890**
All Features	Normal	0.9933	0.9992	0.9817	0.9904
	DoS Hulk	0.9927	0.9791	0.9983	0.9886
	DoS Golden Eye	0.9977	0.9946	0.9952	0.9949
	DoS Slowloris	0.9997	1	0.9966	0.9983
	DoS Slowhttptest	1	1	1	1
	Overall	**0.9948**	**0.9919**	**0.9917**	**0.9917**

From the results, it is seen that the best performance is obtained by using all features together. The overall accuracy for this case is 99.48%. Since the data is unbalanced where most of the flows belong to Normal and DoS Hulk classes, F_1–score is a better evaluation metric. The overall F_1–score is calculated as 99.17% which shows the good performance of the proposed approach. DoS Slowhttptest class is perfectly detected in all cases.

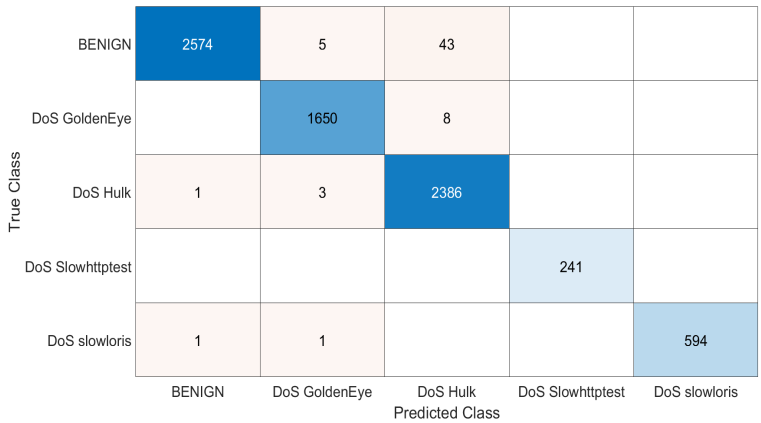

Figure 5.14. Confusion matrix of Wednesday of IDS 2017.

Table 5.26. Number of flows of Tuesday data on IDS 2017.

Flow Type	**Number of Flows**
Normal	13772
FTP Parator	3941
SSH Parator	2945

For the demonstration of simulation results, the confusion matrix taken with payload–based features is given in Figure 5.14. It is observed that for DoS Slowloris class, one sample is labelled as benign and one sample is labelled as DoS Golden Eye. Other samples are correctly labelled. Also, it it seen that 4871 out of 4885 attack samples are correctly labelled. It means 99.71% attack detection rate. However, 48 samples of benign traces are labelled as different DoS attacks. In total, among 7445 flow samples, 7507 of them are correctly detected.

Tuesday: Tuesday data consists of normal network traffic with malicious network traffic from FTP Patator and SSH Patator attacks. These are brute force attacks implemented using FTP and SSH protocols. In this day, 3–class classification model is trained. The distribution of the number of flows after implementing 5–packet threshold for each flow in given in Table 5.26.

Table 5.27. Simulation Results of IDS 2017 - Tuesday.

Feature Set	**Flow Type**	**Accuracy**	**Recall**	**Precision**	**F1 Score**
Statistical Features	Normal	0.9987	0.9985	0.9995	0.9990
	FTP Patator	0.9997	1	0.9983	0.9992
	SSH Patator	0.9990	0.9997	0.9954	0.9996
	Overall	**0.9989**	**0.9987**	**0.9987**	**0.9987**
Payload Based Features	Normal	0.9994	0.9998	0.9993	0.9995
	FTP Patator	0.9997	0.9983	1	0.9992
	SSH Patator	0.9997	0.9989	0.9989	0.9989
	Overall	**0.9995**	**0.9994**	**0.9994**	**0.9994**
All Features	Normal	0.9992	0.9995	0.9993	0.9994
	FTP Patator	0.9995	0.9983	0.9992	0.9987
	SSH Patator	0.9997	0.9989	0.9989	0.9989
	Overall	**0.9993**	**0.9992**	**0.9992**	**0.9992**

Simulation results with different feature sets on Tuesday of IDS 2017 data set are given in Table 5.27. Both feature sets perform good results. Payload–based features have slightly better results than statistical features. Overall accuracy obtained with using all features is 99.93%. Also, for the attack traces, 99.83% precision is achieved for FTP Patator attack and 99.89% precision is achieved for SSH Patator attack. Compared to the previous day of data which is Wednesday including different type of DoS attacks, it is observed that the proposed system also detects different type of attacks which is brute force attack on this day. This shows that the proposed approach has a good performance of this data. While attack traces have similar number of flows, benign traces have larger number of flows. Therefore, F_1–score should also be considered which is calculated as 99.92% by using all features.

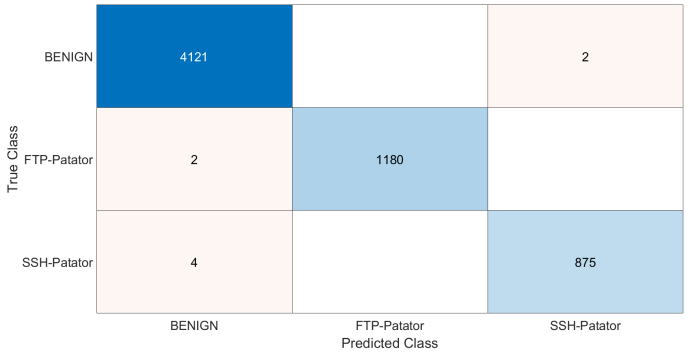

Figure 5.15. Confusion matrix of Tuesday of IDS 2017.

Table 5.28. Number of flows of Friday data on IDS 2017.

Flow Type	**Number of Flows**
Normal	22045
DDoS	15788
Botnet	393
Port Scan	329

The confusion matrix generated using all features on Tuesday of IDS 2017 data set is given in Figure 5.15. It is observed that 2 samples of FTP Patator and 4 samples of SSH Patator are labelled as benign. It is important to note that the attack traces are not mixed to each other. In other words, there are not any FTP Patator samples labelled as SSH Patator and vice versa. Some benign samples are labelled as attack but the number of these samples have a small portion among all benign samples. Among 6184 flow samples, 6176 of them are correctly labelled.

Friday: The other day used in the simulations is Monday which contains attack traces from DDoS, Botnet and Port Scan attacks. Compared to other days, this day contains different type of attacks. Most samples belong to benign and DDoS traces. Botnet and Port Scan attack traces have low number of samples compared to them. 4–class classification model is trained. The distribution of the number of flows after implementing 5–packet threshold for each flow is given in Table 5.28.

Table 5.29. Simulation Results of IDS 2017 - Friday.

Feature Set	Flow Type	Accuracy	Recall	Precision	F1 Score
Statistical Features	Normal	0.9578	0.9837	0.9418	0.9623
	DDoS	0.9580	0.9235	0.9785	0.9502
	Botnet	0.9999	0.9915	1	0.9957
	Port Scan	0.9999	1	0.9898	0.9949
	Overall	**0.9587**	**0.9593**	**0.9578**	**0.9579**
Payload Based Features	Normal	0.9610	0.9867	0.9445	0.9652
	DDoS	0.9610	0.9269	0.9823	0.9538
	Botnet	1	1	1	1
	Port Scan	1	1	1	1
	Overall	**0.9617**	**0.9625**	**0.9610**	**0.9611**
All Features	Normal	0.9625	0.9815	0.9524	0.9667
	DDoS	0.9626	0.9336	0.9751	0.9552
	Botnet	1	1	1	1
	Port Scan	0.9999	1	0.9898	0.9949
	Overall	**0.9632**	**0.9632**	**0.9625**	**0.9626**

Simulation results with different feature sets on Friday of IDS 2017 data set are given in Table 5.29. Payload–based features outperform statistical features with a slight difference in terms of performance metrics. Compared to other days, this day includes different type of attacks. The simulations results show that proposed approach clearly discriminate these attacks. Overall accuracy obtained with using all features is 96.32%. Since benign and DDoS traces have large amount of samples, other performance metrics should be considered.

As in the previous days, since the data has unequal distributions, F_1–score should be considered. In overall, it is calculated as 96.26% which shows the detection capability of the proposed approach. Although this result is low compared to the other F_1–scores in other days, the main reason for this situation is the unbalanced flow samples for the classes in this model.

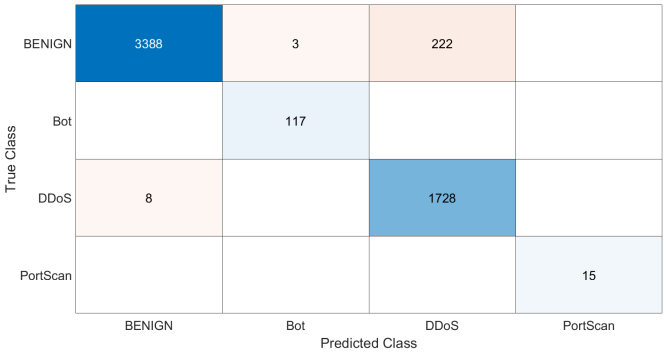

Figure 5.16. Confusion matrix of Friday of IDS 2017.

The confusion matrix taken with payload–based features is given in Figure 5.16. It is observed that as in the previous day Tuesday, attack traces are not mixed to each other. In other words, miss–labelling of attack traces results in benign labels. While only 118 samples of DDoS attack traces are labelled as benign, 315 benign traces are labelled as DDoS. This corresponds to the 4.76% of all benign samples. All flow samples from botnet class are correctly labelled. Also, only 1 sample of Port Scan class is labelled as benign. In overall, 11130 samples out of 5481 samples are correctly detected which corresponds to 96.25% detection rate.

Table 5.30. Number of flows of whole data on IDS 2017.

Flow Type	**Number of Flows**
Normal	31193
DoS Hulk	5578
DoS Golden Eye	3871
DoS Slowloris	1392
DoS Slowhttptest	5788
FTP Parator	2759
SSH Parator	2062
DDoS	11052
Botnet	276
Port Scan	563

Table 5.31. Simulation Results using Statistical Features on IDS 2017.

Application Type	**Accuracy**	**Recall**	**Precision**	**F1 Score**
Normal	0.9785	0.9905	0.9686	0.9794
Bot	1	1	1	1
DDoS	0.9803	0.9236	0.9759	0.9490
Port Scan	0.9999	1	0.9897	0.9948
FTP Patator	0.9999	0.9991	1	0.9995
SSH Patator	0.9998	0.9966	0.9988	0.9977
DoS Hulk	0.9979	0.9826	0.9962	0.9894
DoS Slowloris	0.9999	1	0.9983	0.9991
DoS Golden Eye	0.9990	0.9951	0.9909	0.9930
DoS Slowhttptest	0.9998	0.9917	0.9958	0.9937
Overall	**0.9847**	**0.9785**	**0.9778**	**0.9780**

Total: By using all attack traces from all days of IDS 2017 data set, with a selection of suitable amount of benign traffic samples, performance on the whole IDS 2017 data set is given below. By combining all days of IDS 2017, 10–class model is constructed containing classes which are benign, DoS Hulk, DoS Golden Eye, DoS slowloris, DoS Slowhttptest, FTP Parator, SSH Parator, DDoS, Botnet and Port Scan. The number of flows for each class is given in Table 5.30.

The simulation results taken by using only statistical features are given in Table 5.31. Performance metrics for all classes with the overall results are given. The overall accuracy obtained in this case is 98.47%. Since the data set in unbalanced where most of the samples belong to Normal, DDoS and Dos Hulk classes, it is important to consider the F_1–score which gives better interpretation when the data is unbalanced. The overall F_1–score is calculated as 97.80% where it shows the good performance of the proposed approach. Only one class has F_1–score below 95% which is DDoS attack. The reason for this situation is that since DDoS class has more samples compared to other attacks, some attacks are labelled as DDoS. Brute force attacks which are FTP Patator and SSH Patator are detected almost perfectly.

Table 5.32. Simulation Results using Payload–Based Features on IDS 2017.

Application Type	**Accuracy**	**Recall**	**Precision**	**F1 Score**
Normal	0.9764	0.9911	0.9640	0.9773
Bot	1	1	1	1
DDoS	0.9786	0.9141	0.9778	0.9449
Port Scan	0.9999	1	0.9897	0.9948
FTP Patator	0.9998	0.9991	0.9983	0.9987
SSH Patator	0.9997	0.9965	0.9954	0.9960
DoS Hulk	0.9979	0.9826	0.9962	0.9894
DoS Slowloris	0.9998	0.9966	0.9983	0.9974
DoS Golden Eye	0.9992	0.9927	0.9963	0.9945
DoS Slowhttptest	0.9999	1	0.9958	0.9979
Overall	**0.9833**	**0.9769**	**0.9759**	**0.9761**

The simulation results taken with using only payload–based features are given in Table 5.32. The overall F_1–score obtained in this case is 97.61% which is lower than the one obtained with using statistical–fetaures. The worst performance is obtained with DDoS attack class with an F_1–score of 94.49%. Wrongly detected DDoS samples are labelled as Normal. Compared to the results taken with statistical features, only for two classes which are DoS Golden Eye and Dos Slowhttptest results improved. However, although for the other classes, performance decreases, it is still in a good level with having 97.59% precision rate.

The simulation results obtained by using all features are given in Table 5.33. The overall F_1–score in this case is 97.78%. The improved results in terms of F_1–score are for the classes Normal, DDoS, DoS Hulk, Golden Eye Slowhttptest. There is low performance decrease resulting from a few miss–labelling of samples seen on the other classes. However, in overall detection rate is improved with a value of 97.85%.

Table 5.33. Simulation Results using All Features on IDS 2017.

Application Type	**Accuracy**	**Recall**	**Precision**	**F1 Score**
Normal	0.9789	0.9945	0.9655	0.9798
Bot	1	1	1	1
DDoS	0.9807	0.9173	0.9860	0.9504
Port Scan	0.9999	1	0.9795	0.9896
FTP Patator	0.9999	0.9991	1	0.9995
SSH Patator	0.9998	0.9966	0.9988	0.9977
DoS Hulk	0.9983	0.9835	0.9991	0.9912
DoS Slowloris	0.9999	1	0.9983	0.9991
DoS Golden Eye	0.9994	0.9963	0.9945	0.9954
DoS Slowhttptest	0.9999	1	0.9958	0.9979
Overall	**0.9851**	**0.9797**	**0.9785**	**0.9787**

The confusion matrix obtained by using all features is given in Figure 5.17. It is observed that some benign samples are miss–labelled with different type of attacks such as DDoS, Dos Golden Eye, DoS Hulk, FTP Patator and SSH Patator. For example, 421 benign samples are labelled as DDoS. Also, for the DDoS class, 66 samples of it are labelled as benign. The other types of attacks are slightly mixed to each other such as 9 DoS Golden Eye samples are labelled as DoS Hulk. Also, 2 Dos Hulk samples are labelled as DoS Golden Eye. In total, among 25267 samples 24725 of them are truly detected which corresponds to 97.85% detection rate.

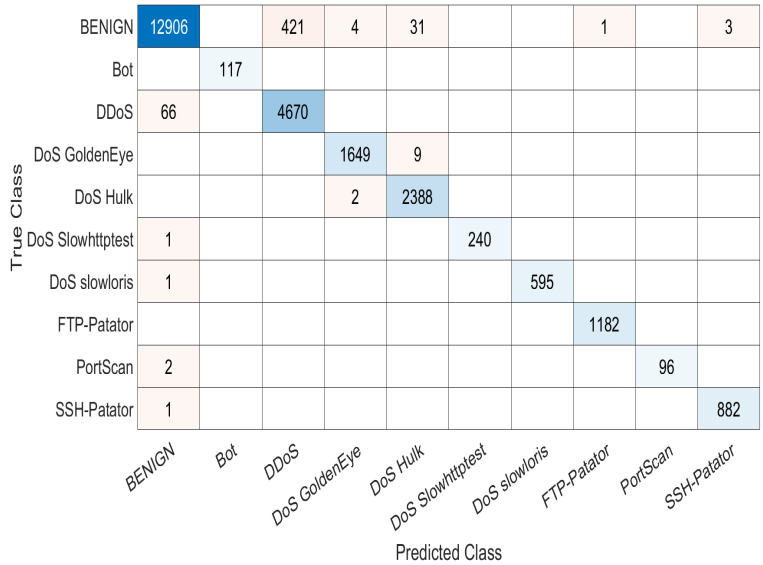

Figure 5.17. Confusion matrix of IDS 2017 data set with all classes.

5.4.2. Flow-Based Network Traffic Classification System

In the proposed approach is tested on the application classification data set given in [73]. The details of the data set is given in Chapter 5.1.4. This data set includes real–world up–to-date network traces from multiple applications. The proposed system is used for detecting which type of applications flows belong to. As in the intrusion detection system given in Chapter 5.4.1, statistical and payload–based features are used alone and in combination to take the performance results. 70% of the data is taken as training and 30% of the data is taken as test. 10–fold cross–validation is applied in the training stage. All flows in the data set are given into the simulations without applying any random selections.

The simulation results are given in three parts as in Chapter 5.4.1 where statistical and payload–based features are used alone and both features sets are combined. The results when statistical features are used are given in Table 5.34. For each application, accuracy, precision, recall and F_1–score values are given. Also, the overall performance metric values are given at the end of the table. To compare the results, since the data set is unbalanced where Tunnelbear and Ultrasurf applications have relatively high number of flow, F_1–score is a better evaluation metric. Calculated F_1–score for

statistical features is 98.06%. While most of the applications are detected with a high precision, Discord, Dropbox, Microsoft Teams and Steam applications give precision values below 90%. The reason for this situation is that statistical properties of the flows of these applications may resemble with the other flows in the model. On the other hand, two different chat applications which are Whatsapp and Telegram are detected perfectly and not mixed to each other. Also, VPN applications such as Proton VPN, Tunnelbear and Ultrasurf give good F_1–score results which are above 98.21%, 99.51% and 99.24% respectively.

The simulation results when payload–based features are used alone are given in Table 5.35. Compared to statistical features, worse performance is obtained by using payload–features alone. Calculated overall F_1–score is 94.53% which is smaller than 98.06% calculated by using only statistical features. Similar payloads of packets make it tougher to detect applications using only payload–based applications. Among applications, in terms of F_1–score, three applications give better results which are ITunes, Soulseekqt and Tunnelbear. It is observed that payload–based features may be helpful for the detection of some applications but for the whole data set by using these features alone, the performance decreases.

The simulation results when both features sets are used are given in Table 5.36. The best performance is obtained when the feature sets are used together. Overall F_1–score calculated for this case is 98.55% which is best among all cases. Two applications give precision values below 90% which are Discord with 83.58% and Microsoft Teams with 75.83%. Compared to statistical features, the detection rates of 17 out of 22 applications increases. Also, both chat applications Telegram and Whatsapp are perfectly detected in this case. Most improvement in terms of F_1–score is observed on Steam application.

The confusion matrix when all features are used is given in Figure 5.18. By using this matrix, miss–labelled samples can be found. It is observed that most miss–labelling is given into Spotify and Ultrasurf applications. For example, 4 Amazon

Prime Video application flows are labelled as Ultrasurf. Also, 12 Microsoft Teams application samples are labelled as Spotify. 8 Microsoft Teams application samples are labelled as Skype. The other miss–labelling scenarios are small compared to these cases. Altough Tunnelbear application has the most samples, only 2 of its samples are labelled as Steam and only 3 Ultrasurf samples are labelled as Tunnelbear. In total, among 11520 samples 11355 of them are correctly detected which corresponds to 98.57% overall precision value.

Table 5.34. Simulation Results using Statistical Features.

Application Type	**Accuracy**	**Recall**	**Precision**	**F1 Score**
Amazon Prime	0.9988	0.9862	0.9913	0.9888
CyberGhost	0.9996	0.9805	0.9805	0.9805
Deezer	0.998	0.9436	0.9571	0.9503
Discord	0.9968	0.9224	0.7985	0.8560
Dropbox	0.9990	0.9726	0.8875	0.9281
Epic Games	0.9976	0.9700	0.9627	0.9663
Facebook	0.9996	0.9840	0.9840	0.9840
Hotspot	0.9997	1	0.9375	0.9677
ITunes	0.9983	0.9447	0.9669	0.9557
Microsoft Teams	0.9958	0.9173	0.7449	0.8222
Proton VPN	0.9994	0.9705	0.9939	0.9821
Skype	0.9960	0.9157	0.9191	0.9174
Slack	0.9996	0.9938	0.9938	0.9938
Soulseekqt	0.9990	0.9523	0.9433	0.9478
Spotify	0.9960	0.9455	0.9642	0.9548
Steam	0.9968	0.9076	0.8309	0.8676
Telegram	1	1	1	1
Tunnelbear	0.9967	0.9915	0.9987	0.9951
Tunneln	0.9994	0.9943	0.9943	0.9943
Ultrasurf	0.9959	0.9891	0.9958	0.9924
Whatsapp	0.9999	0.9891	1	0.9945
Zoom	0.9989	0.9825	0.9656	0.9740
Overall	**0.9971**	**0.9807**	**0.9810**	**0.9806**

Table 5.35. Simulation Results using Payload–Based Features.

Application Type	**Accuracy**	**Recall**	**Precision**	**F1 Score**
Amazon Prime	0.9930	0.9355	0.9258	0.9306
CyberGhost	0.9973	0.8461	0.8543	0.8502
Deezer	0.9931	0.8133	0.8095	0.8114
Discord	0.9957	0.8455	0.7761	0.8093
Dropbox	0.9987	0.9459	0.8750	0.9090
Epic Games	0.9950	0.9346	0.9230	0.9288
Facebook	0.9995	0.9761	0.9840	0.9800
Hotspot	0.9986	0.8809	0.7708	0.8222
ITunes	0.9990	0.9589	0.9905	0.9744
Microsoft Teams	0.9906	0.7157	0.4563	0.5573
Proton VPN	0.9971	0.8633	0.9518	0.9054
Skype	0.9951	0.8970	0.8970	0.8970
Slack	0.9915	0.8817	0.8080	0.8432
Soulseekqt	0.9993	0.9900	0.9339	0.9611
Spotify	0.9832	0.8387	0.7638	0.7995
Steam	0.9951	0.8412	0.7464	0.7910
Telegram	0.9990	0.9545	0.9459	0.9502
Tunnelbear	0.9991	0.9981	0.9992	0.9987
Tunneln	0.9986	0.9850	0.9850	0.9850
Ultrasurf	0.9781	0.9383	0.9839	0.9606
Whatsapp	0.9986	0.9411	0.8791	0.9090
Zoom	0.9979	0.9336	0.9656	0.9493
Overall	**0.9916**	**0.9452**	**0.9470**	**0.9453**

Table 5.36. Simulation Results using All Features.

Application Type	**Accuracy**	**Recall**	**Precision**	**F1 Score**
Amazon Prime	0.9986	0.9845	0.9879	0.9862
CyberGhost	0.9997	0.9807	0.9902	0.9855
Deezer	0.9988	0.9758	0.9619	0.9688
Discord	0.9977	0.96551	0.8358	0.8960
Dropbox	0.9994	1	0.9250	0.9610
Epic Games	0.9981	0.9727	0.9751	0.9739
Facebook	0.9998	1	0.9840	0.9919
Hotspot	0.9997	0.9787	0.9583	0.9684
ITunes	0.9994	0.9813	0.9905	0.9859
Microsoft Teams	0.9960	0.9186	0.7583	0.8308
Proton VPN	0.9996	0.9939	0.9819	0.9878
Skype	0.9971	0.9192	0.9632	0.9407
Slack	0.9995	0.9907	0.9938	0.9922
Soulseekqt	0.9997	1	0.9716	0.9856
Spotify	0.9953	0.9360	0.9583	0.9470
Steam	0.9982	0.9236	0.9366	0.9300
Telegram	1	1	1	1
Tunnelbear	0.9995	0.9992	0.9994	0.9993
Tunneln	0.9995	0.9962	0.9943	0.9953
Ultrasurf	0.9960	0.9882	0.9971	0.9926
Whatsapp	1	1	1	1
Zoom	0.9988	0.9741	0.9699	0.9720
Overall	**0.9982**	**0.9856**	**0.9857**	**0.9855**

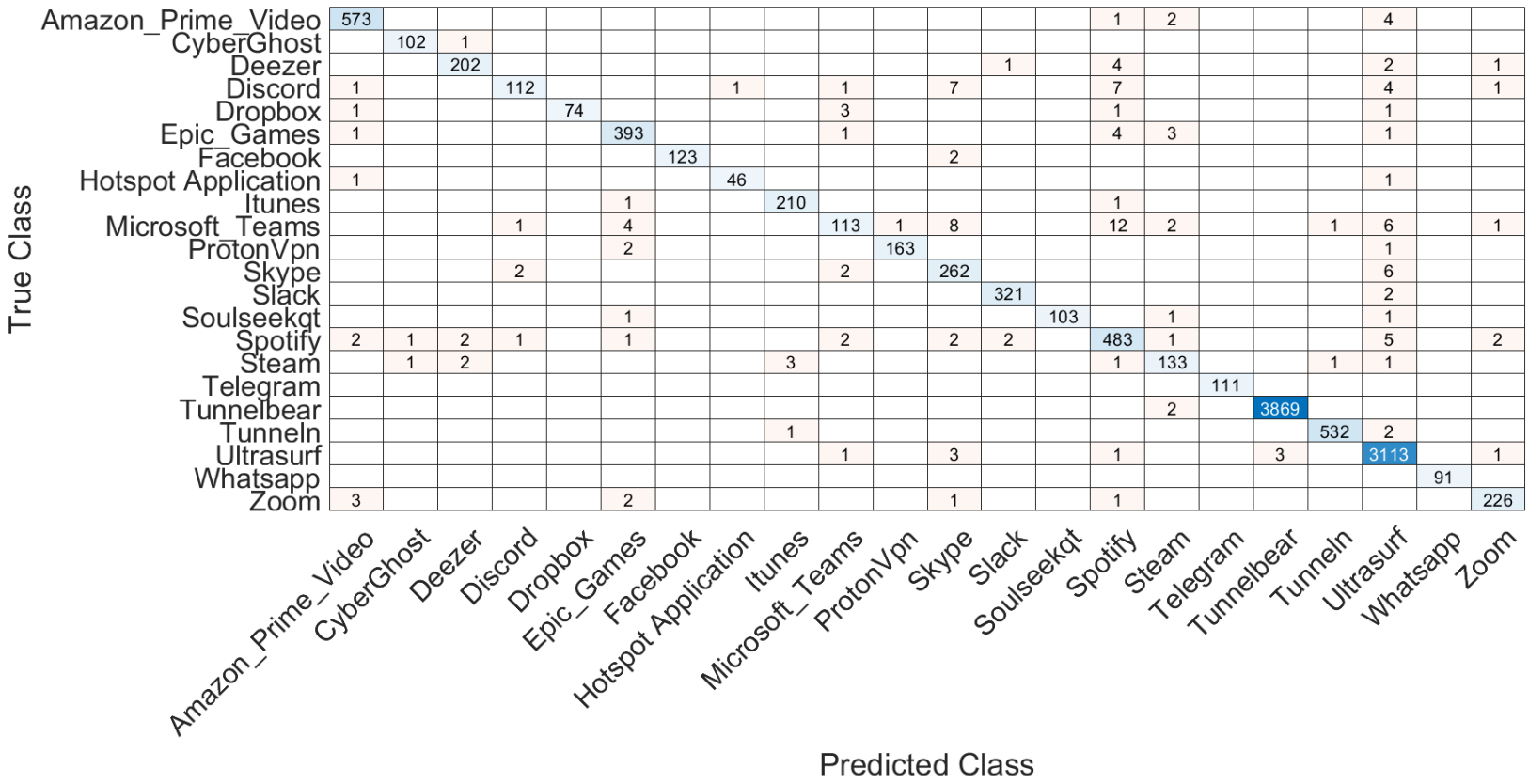

Figure 5.18. Confusion matrix when all features are used.

6. CONCLUSION & FUTURE WORK

In this thesis, different deep packet inspection systems are proposed for network intrusion detection and network application classification purposes. These systems are grouped into two categories which are packet–based systems and flow–based systems. To use these systems, novel features are extracted. For packet–based systems, Greedy algorithm is implemented for finding an upper bound for the distance between probability distributions with different sizes. Entropy which is a measure of the randomness of a system is also utilized in the extraction of features. Besides these information theory–based metrics, a clustering algorithm is proposed to analyze the relationship between a pair of packet header values. This algorithm is used for cluster one packet header values such as source IP addresses depending on the connections between another packet header value such as destination IP addresses. To highlight the strength of the clustering algorithm, modularity value is calculated. Strong modularity value shows the constructed clusters are more similar to each other. Also, in another approach, the network is modelled by using the graph theory features. As in the other approaches, relationships between the packet header values are analyzed. For flow–based systems, two different types of features are used which are statistical features and payload–based features. Statistical features are derived using the header information of the packets within the flows. For example, data transmission rate of the flow is one of the statistical features. Also, payload portions of packets are analyzed to derive the novel features. Entropy and Greedy algorithm concepts are used to derive these features. By constructing the histogram of payloads, they are treated as probability distributions. These probability distributions are used with information theory metrics to derive the features. Moreover, printable character ratio within a payload is also considered as a feature.

Packet–based approaches use windows to execute time–series analysis. The size of these windows may be decided using a pre–defined duration or pre–defined number of packets. Simulation results show that these approaches are successful in the detection

of DoS and DDoS attacks. However, it is observed that for Brute Force and Port Scan attacks, these approaches cannot reach the satisfactory detection rate. The reason for this situation is that in these types of attacks, the window may not have enough malicious packets to highlight the attack behavior. Therefore, the discrimination of the attack traffic from normal traffic cannot be observed clearly.

In flow–based approaches, network packets are gathered into flows by using their source IP addresses, destination IP addresses, source port numbers, destination port numbers and protocol information. This approach has advantages over packet–based approaches such that the behavior of network packets is extracted more effectively. Therefore, it is observed that the attacks that cannot be detected with sufficiently high accuracy in packet–based approaches are detected with a better accuracy. Also, flow–based approaches are used in the network application classification task. A data set consisting of real–world traces of 22 different popular applications is used for the performance evaluation. It is observed that the proposed approach has a high detection rate.

For a future work, the analysis of different types of attacks will be implemented. Also, network application classification system will be improved by adding new features. More applications will be tested using the proposed approach. Also, a comprehensive system is studied about combining packet–based and flow–based approaches for network intrusion detection task. For example, the first module of the system uses packet–based approaches to give early warnings to the system. Then, flow–based approaches make more detailed analysis to find the attack traces. Moreover, clustering algorithm and graph–based approach given in packet–based systems will be developed to be placed in flow–based systems.

REFERENCES

1. El-Maghraby, R. T., N. M. Abd Elazim and A. M. Bahaa-Eldin, "A Survey on Deep Packet Inspection", *International Conference on Computer Engineering and Systems (ICCES)*, pp. 188–197, Cairo, Egypt, 2017.

2. Bujlow, T., V. Carela-Español and P. Barlet-Ros, "Extended Independent Comparison of Popular Deep Packet Inspection (DPI) Tools for Traffic Classification", `https://vbn.aau.dk/en/publications/extended-independent-comparison`, accessed on July 15, 2022.

3. Froehlich, A., L. Rosencrance and K. Gattine, "OSI model (Open Systems Interconnection)", `https://www.techtarget.com/searchnetworking/definition/OSI`, accessed on July 15, 2022.

4. AbuHmed, T., A. Mohaisen and D. Nyang, "A Survey on Deep Packet Inspection for Intrusion Detection Systems", *arXiv preprint arXiv:0803.0037*, 2008.

5. Vidyasagar, M., "A Metric Between Probability Distributions on Finite Sets of Different Cardinalities and Applications to Order Reduction", *IEEE Transactions on Automatic Control*, Vol. 57, No. 10, pp. 2464–2477, 2012.

6. Newman, M. E., "Modularity and Community Structure in Networks", *Proceedings of the National Academy of Sciences*, Vol. 103, No. 23, pp. 8577–8582, 2006.

7. Shiravi, A., H. Shiravi, M. Tavallaee and A. A. Ghorbani, "Toward Developing a Systematic Approach to Generate Benchmark Datasets for Intrusion Detection", *Computers & Security*, Vol. 31, No. 3, pp. 357–374, 2012.

8. Sharafaldin, I., A. H. Lashkari and A. A. Ghorbani, "Toward Generating a New Intrusion Detection Dataset and Intrusion Traffic Characterization", *International*

Conference on Information Systems Security and Privacy (ICISSP), Vol. 1, pp. 108–116, 2018.

9. Mukherjee, B., L. T. Heberlein and K. N. Levitt, "Network Intrusion Detection", *Network Security Journals*, Vol. 8, No. 3, pp. 26–41, 1994.

10. Northcutt, S. and J. Novak, *Network Intrusion Detection*, Sams Publishing, 2002.

11. Warburton, D., "Distributed Denial of Service Attack Trends", `https://www.f5.com/labs/articles/threat-intelligence`, accessed on July 15, 2022.

12. Kaspersky, "Distributed Denial of Service: Anatomy and Impact of DDoS Attacks", `https://usa.kaspersky.com/resource-center/preemptive-safety`, accessed on July 15, 2022.

13. Nicholson, P., "Five Most Famous DDoS Attacks and Then Some", `https://www.a10networks.com/blog/5-most-famous-ddos-attacks`, accessed on July 15, 2022.

14. De Vivo, M., E. Carrasco, G. Isern and G. O. De Vivo, "A Review of Port Scanning Techniques", *SIGCOMM Computer Communication Review*, Vol. 29, No. 2, pp. 41–48, 1999.

15. Bhuyan, M. H., D. K. Bhattacharyya and J. K. Kalita, "Surveying Port Scans and Their Detection Methodologies", *The Computer Journal*, Vol. 54, No. 10, pp. 1565–1581, 2011.

16. Knudsen, L. R. and M. J. Robshaw, "Brute Force Attacks", *The Block Cipher Companion*, pp. 95–108, 2011.

17. Mahjabin, T., Y. Xiao, G. Sun and W. Jiang, "A Survey of Distributed Denial-of-service Attack, Prevention, and Mitigation Techniques", *International Journal of*

Distributed Sensor Networks, Vol. 13, No. 12, pp. 1–33, 2017.

18. Paxson, V., "Bro: A System for Detecting Network Intruders in Real-time", *Computer Networks*, Vol. 31, No. 24, pp. 2435–2463, 1999.

19. Roesch, M., "Snort: Lightweight Intrusion Detection for Networks", *Lisa*, Vol. 99, No. 1, pp. 229–238, 1999.

20. Singh, J. and S. Behal, "Detection and Mitigation of DDoS Attacks in SDN: A Comprehensive Review, Research Challenges and Future Directions", *Computer Science Review*, Vol. 37, pp. 1–25, 2020.

21. Vishwakarma, R. and A. K. Jain, "A Survey of DDoS Attacking Techniques and Defence Mechanisms in the IoT Network", *Telecommunication Systems*, Vol. 73, No. 1, pp. 3–25, 2020.

22. Wang, R., Z. Jia and L. Ju, "An Entropy-based Distributed DDoS Detection Mechanism in Software-defined Networking", *IEEE Trustcom/BigDataSE/ISPA*, Vol. 1, pp. 310–317, Helsinki, Finland, 2015.

23. Mousavi, S. M. and M. St-Hilaire, "Early Detection of DDoS Attacks Against SDN Controllers", *International Conference on Computing, Networking and Communications (ICNC)*, pp. 77–81, Garden Grove, CA, USA, 2015.

24. Tsai, S.-C., I. Liu, C.-T. Lu, C.-H. Chang, J.-S. Li *et al.*, "Defending Cloud Computing Environment Against the Challenge of DDoS Attacks Based on Software Defined Network", *Advances in Intelligent Information Hiding and Multimedia Signal Processing*, pp. 285–292, Springer, 2017.

25. Sahoo, K. S., D. Puthal, M. Tiwary, J. J. Rodrigues, B. Sahoo and R. Dash, "An Early Detection of Low Rate DDoS Attack to SDN Based Data Center Networks Using Information Distance Metrics", *Future Generation Computer Systems*, Vol. 89, pp. 685–697, 2018.

26. Sahoo, K. S., M. Tiwary and B. Sahoo, "Detection of High Rate DDoS Attack from Flash Events using Information Metrics in Software Defined Networks", *International Conference on Communication Systems & Networks (COMSNETS)*, pp. 421–424, Bengaluru, India, 2018.

27. CAIDA, "The CAIDA UCSD DDoS Attack 2007 Dataset", `https://www.caida.org/catalog/datasets/ddos-2007080-dataset`, accessed on July 15, 2022.

28. Boite, J., P.-A. Nardin, F. Rebecchi, M. Bouet and V. Conan, "Statesec: Stateful Monitoring for DDoS Protection in Software Defined Networks", *IEEE Conference on Network Softwarization (NetSoft)*, pp. 1–9, Bologna, Italy, 2017.

29. Kalkan, K., L. Altay, G. Gür and F. Alagöz, "JESS: Joint Entropy-based DDoS Defense Scheme in SDN", *IEEE Journal on Selected Areas in Communications*, Vol. 36, No. 10, pp. 2358–2372, 2018.

30. Hong, G.-C., C.-N. Lee and M.-F. Lee, "Dynamic Threshold for DDoS Mitigation in SDN Environment", *Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA ASC)*, pp. 1–7, Lanzhou, China, 2019.

31. Ahalawat, A., S. S. Dash, A. Panda and K. S. Babu, "Entropy Based DDoS Detection and Mitigation in OpenFlow Enabled SDN", *International Conference on Vision Towards Emerging Trends in Communication and Networking (ViTECoN)*, pp. 1–5, Vellore, India, 2019.

32. Cui, J., M. Wang, Y. Luo and H. Zhong, "DDoS Detection and Defense Mechanism based on Cognitive-inspired Computing in SDN", *Future Generation Computer Systems*, Vol. 97, pp. 275–283, 2019.

33. MAWILAB, "MAWI Working Group Traffic Archive", `http://mawi.wide.ad.jp/mawi/`, accessed on July 15, 2022.

34. Xuanyuan, M., V. Ramsurrun and A. Seeam, "Detection and Mitigation of DDoS Attacks using Conditional Entropy in Software-defined Networking", *nternational Conference on Advanced Computing (ICoAC)*, pp. 66–71, Chennai, India, 2019.

35. MIT, "MIT Lincoln Laboratory 2000 DARPA Intrusion Detection Data Set", `https://www.ll.mit.edu/r-d/datasets`, accessed on July 15, 2022.

36. Li, R. and B. Wu, "Early Detection of DDoS based on ϕ–entropy in SDN Networks", *IEEE Information Technology, Networking, Electronic and Automation Control Conference (ITNEC)*, pp. 731–735, Chongqing, China, 2020.

37. Doriguzzi-Corin, R., S. Millar, S. Scott-Hayward, J. Martinez-del Rincon and D. Siracusa, "Lucid: A Practical, Lightweight Deep Learning Solution for DDoS Attack Detection", *IEEE Transactions on Network and Service Management*, Vol. 17, No. 2, pp. 876–889, 2020.

38. Koay, A., A. Chen, I. Welch and W. K. Seah, "A New Multi Classifier System using Entropy-based Features in DDoS Attack Detection", *International Conference on Information Networking (ICOIN)*, pp. 162–167, Chiang Mai, Thailand, 2018.

39. Min, E., J. Long, Q. Liu, J. Cui and W. Chen, "TR-IDS: Anomaly-based Intrusion Detection through Text-convolutional Neural Network and Random Forest", *Security and Communication Networks*, Vol. 1, 2018.

40. Cui, J., J. Long, E. Min, Q. Liu and Q. Li, "Comparative Study of CNN and RNN for Deep Learning based Intrusion Detection System", *Lecture Notes in Computer Science*, Vol. 11067, pp. 159–170, 2018.

41. Roopak, M., G. Y. Tian and J. Chambers, "Deep Learning Models for Cyber Security in IoT Networks", *IEEE Annual Computing and Communication Workshop and Conference (CCWC)*, pp. 452–457, Las Vegas, NV, USA, 2019.

42. Homayoun, S., M. Ahmadzadeh, S. Hashemi, A. Dehghantanha and R. Khayami,

"BoTShark: A Deep Learning Approach for Botnet Traffic Detection", *Cyber Threat Intelligence*, pp. 137–153, 2018.

43. AlEroud, A. and I. Alsmadi, "Identifying Cyber-attacks on Software Defined Networks: An Inference-based Intrusion Detection Approach", *Journal of Network and Computer Applications*, Vol. 80, pp. 152–164, 2017.

44. Wang, B., Y. Zheng, W. Lou and Y. T. Hou, "DDoS Attack Protection in the Era of Cloud Computing and Software-defined Networking", *Computer Networks*, Vol. 81, pp. 308–319, 2015.

45. Xiao, P., Z. Li, H. Qi, W. Qu and H. Yu, "An Efficient DDoS Eetection with Bloom Filter in SDN", *IEEE Trustcom/BigDataSE/ISPA*, pp. 1–6, Tianjin, China, 2016.

46. Conti, M., A. Gangwal and M. S. Gaur, "A Comprehensive and Effective Mechanism for DDoS Detection in SDN", *IEEE International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob)*, pp. 1–8, Rome, Italy, 2017.

47. Bhushan, K. and B. B. Gupta, "Distributed Denial of Service (DDoS) Attack Mitigation in Software Defined Network (SDN)-based Cloud Computing Environment", *Journal of Ambient Intelligence and Humanized Computing*, Vol. 10, No. 5, pp. 1985–1997, 2019.

48. Draper-Gil, G., A. H. Lashkari, M. S. I. Mamun and A. A. Ghorbani, "Characterization of Encrypted and Vpn Traffic using Time-related Features", *International Conference on Information Systems Security and Srivacy (ICISSP)*, pp. 407–414, New Brunswick, Canada, 2016.

49. Karagiannis, T., A. Broido, N. Brownlee, K. Claffy and M. Faloutsos, "File-sharing in the Internet: A Characterization of P2P Traffic in the Backbone", *University of California, Riverside, USA, Tech. Rep*, Vol. 1, No. 1, pp. 1–13, 2003.

50. Moore, A. W. and K. Papagiannaki, "Toward the Accurate Identification of Network Applications", *International Workshop on Passive and Active Network Measurement*, pp. 41–54, Berlin, Germany, 2005.

51. Risso, F., M. Baldi, O. Morandi, A. Baldini and P. Monclus, "Lightweight, Payload-based Traffic Classification: An Experimental Evaluation", *IEEE International Conference on Communications*, pp. 5869–5875, Beijing, China, 2008.

52. Park, J.-S., S.-H. Yoon and M.-S. Kim, "Software Architecture for a Lightweight Payload Signature-based Traffic Classification System", *International Workshop on Traffic Monitoring and Analysis*, pp. 136–149, Berlin, Germany, 2011.

53. Park, J.-S., S.-H. Yoon and M.-S. Kim, "Performance Improvement of Payload Signature-based Traffic Classification System using Application Traffic Temporal Locality", *Asia-Pacific Network Operations and Management Symposium (APNOMS)*, pp. 1–6, Hiroshima, Japan, 2013.

54. Williams, N. and S. Zander, "Evaluating Machine Learning Algorithms for Automated Network Application Identification", *Swinburne University of Technology. Centre for Advanced Internet Architectures*, Vol. 1, No. 1, pp. 1–14, 2006.

55. Nguyen, T. T. and G. Armitage, "A Survey of Techniques for Internet Traffic Classification using Machine Learning", *IEEE Communications Surveys & Tutorials*, Vol. 10, No. 4, pp. 56–76, 2008.

56. Kaoprakhon, S. and V. Visoottiviseth, "Classification of Audio and Video Traffic over HTTP Protocol", *International Symposium on Communications and Information Technology*, pp. 1534–1539, Icheon, Korea (South), 2009.

57. Wang, J.-M., C.-L. Qian, C.-H. Che and H.-T. He, "Study on Process of Network Traffic Classification using Machine Learning", *Annual ChinaGrid Conference*, pp. 262–266, Guangzhou, China, 2010.

58. Dong, S., D. Zhou and W. Ding, "The Study of Network Traffic Identification based on Machine Learning Algorithm", *International Conference on Computational Intelligence and Communication Networks*, pp. 205–208, Mathura, India, 2012.

59. Huang, N.-F., G.-Y. Jai, C.-H. Chen and H.-C. Chao, "On the Cloud-based Network Traffic Classification and Applications Identification Service", *International Conference on Selected Topics in Mobile and Wireless Networking*, pp. 36–41, Avignon, France, 2012.

60. NLANR, "National Laboratory for Applied Network Research (NLANR) Network Traces", `http://pma.nlanr.net/Special/`, accessed on July 15, 2022.

61. Yamansavascilar, B., M. A. Guvensan, A. G. Yavuz and M. E. Karsligil, "Application Identification via Network Traffic Classification", *International Conference on Computing, Networking and Communications (ICNC)*, pp. 843–848, Silicon Valley, CA, USA, 2017.

62. Aceto, G., D. Ciuonzo, A. Montieri and A. Pescapé, "Distiller: Encrypted Traffic Classification via Multimodal Multitask Deep Dearning", *Journal of Network and Computer Applications*, Vol. 183, No. 1, pp. 1–17, 2021.

63. Shannon, C. E., "A Mathematical Theory of Communication", *The Bell System Technical Journal*, Vol. 27, No. 3, pp. 379–423, 1948.

64. Rényi, A., "On the Foundations of Information Theory", *Revue de l'Institut International de Statistique*, pp. 1–14, 1965.

65. Ash, R. B., *Information Theory*, Courier Corporation, 2012.

66. Van Erven, T. and P. Harremos, "Rényi Divergence and Kullback-Leibler Divergence", *IEEE Transactions on Information Theory*, Vol. 60, No. 7, pp. 3797–3820, 2014.

67. Han, J., J. Pei and M. Kamber, *Data Mining: Concepts and Techniques*, Elsevier, 2011.

68. Ateş, Ç., S. Özdel and E. Anarım, "Clustering Based DDoS Attack Detection Using the Relationship Between Packet Headers", *Innovations in Intelligent Systems and Applications Conference (ASYU)*, Izmir, Turkey, 2019.

69. Chowdhury, S., M. Khanzadeh, R. Akula, F. Zhang, S. Zhang, H. Medal, M. Marufuzzaman and L. Bian, "Botnet Detection Using Graph-based Feature Clustering", *Journal of Big Data*, Vol. 4, No. 1, 2017.

70. Kifer, Y., "Perron-Frobenius Theorem, Large Deviations, and Random Perturbations in Random Environments", *Mathematische Zeitschrift*, Vol. 222, No. 4, pp. 677–698, 1996.

71. Ates, C., S. Özdel and E. Anarim, "Graph-based Fuzzy Approach Against DDoS attacks", *Journal of Intelligent & Fuzzy Systems*, Vol. 39, No. 5, pp. 6315–6324, 2020.

72. Erhan, D. and E. Anarım, "Boğaziçi University Distributed Denial of Service Dataset", *Data in Brief*, Vol. 32, No. 1, pp. 1–6, 2020.

73. Karayaka, M., A. Bayer, S. Balkı, M. Koca and E. Anarım, "Application Based Network Traffic Dataset and SPID Analysis", *Signal Processing and Communications Applications Conference (SIU)*, Safranbolu, Turkey, 2022.

74. Dunn, J. C., *A Fuzzy Relative of the Isodata Process and its use in Detecting Compact Well-separated Clusters*, 1973.

75. Bezdek, J. C., *Pattern Recognition with Fuzzy Objective Function Algorithms*, Springer Science & Business Media, 2013.