University of Cyprus

Department of Electrical and Computer Engineering

Detection of Collaborative Misbehaviour in Distributed Cyber-Attacks through Correlation and Time Dependency Analysis

Marios Thoma

A Dissertation

Submitted in Partial Fulfillment of the

Requirements for the Degree of

Doctor of Philosophy

at the University of Cyprus

October 2017

VALIDATION PAGE

Doctoral Candidate: Marios Thoma

Doctoral Thesis Title: Detection of Collaborative Misbehaviour in Distributed Cyber-Attacks through Correlation and Time Dependency Analysis

The present Doctorate Dissertation was submitted in partial fulfillment of the requirements for the Degree of Doctor of Philosophy at the ***Department of Electrical and Computer Engineering****, and was approved on the* 26*th* *of October,* 2017 *by the members of the* ***Examination Committee.***

Examination Committee

Committee Chair ________________________________
(Dr. Georgios Ellinas, Associate Professor)

Research Supervisor ________________________________
(Dr. Christoforos Hadjicostis, Professor)

Committee Member ________________________________
(Dr. George Hadjichristofi, Lecturer)

Committee Member ________________________________
(Dr. Ioannis Krikidis, Assistant Professor)

Committee Member ________________________________
(Dr. Vassos Vassiliou, Assistant Professor)

Declaration of Doctoral Candidate

The present doctoral dissertation was submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy at the University of Cyprus. It is a product of original work of my own, unless otherwise mentioned through references, notes, or any other statements.

Marios Thoma ______________________

Abstract

In this thesis, we consider modelling and detection of suspiciously high correlation between malicious Internet users that are collaborating in order to cause a Denial of Service (DoS) attack or a Distributed Denial of Service (DDoS) attack. The main goal is to recognise cyber incidents and obtain a method for judging early enough any collaborative misbehaviour (more specifically, collaboration/dependency between the requests that are issued by different users) in order to ultimately isolate their behaviour and overcome the consequences of the DoS attack. The proposed method relies on the analysis of data traffic across the concerned network (with both incoming and outgoing traffic) in an effort to identify correlations between different users, based on the frequency with which they simultaneously issue requests for service. The thesis models user behaviour via hidden Markov models and analyses the performance of the proposed method, using both mathematical reasoning and simulations. The approach represents a step towards achieving an effective and proactive defence against DoS/DDoS attacks. Furthermore, we examine the performance and detection time of the proposed method, and its relationship to other methods and work produced in this direction by other researchers. Our intention is the implementation of a warning method capable of identifying early enough any abnormal behaviour, before a minor cyber incident results in a catastrophic failure.

Περίληψη

Σε αυτή τη διατριβή, εξετάζουμε τη μοντελοποίηση και την ανίχνευση ύποπτα υψηλού βαθμού συσχέτισης μεταξύ κακόβουλων χρηστών του διαδικτύου που συνεργάζονται για να προκαλέσουν Επίθεση Άρνησης Εξυπηρέτησης [Denial of Service (DoS) Attack] ή Κατανεμημένη Επίθεση Άρνησης Εξυπηρέτησης [Distributed Denial of Service (DDoS) Attack]. Ο κύριος στόχος της διατριβής είναι η έγκαιρη αναγνώριση τυχόν κυβερνοπεριστατικών μέσα στον κυβερνοχώρο. Η προτεινόμενη μέθοδος είναι σε θέση να κρίνει αρκετά έγκαιρα οποιαδήποτε συσχετισμένη επιλήψιμη συμπεριφορά των διαφόρων χρηστών (πιο συγκεκριμένα, συνεργασία-εξάρτηση μεταξύ των αιτημάτων που εκδίδονται από διαφορετικούς χρήστες) προκειμένου να απομονωθούν τελικά οι εν λόγω χρήστες και να αποφευχθούν οι συνέπειες μιας επίθεσης DDoS . Η προτεινόμενη μέθοδος βασίζεται στην ανάλυση της κυκλοφορίας δεδομένων στο υπό εξέταση δίκτυο (εισερχόμενη/ εξερχόμενη κίνηση), προκειμένου να εντοπιστούν οι συσχετισμοί με βάση τη συχνότητα με την οποία εκδίδονται ταυτόχρονα αιτήματα. Τα μοντέλα της διατριβής εξετάζονται με βάση την λογική των Κρυφών Μαρκοβιανών Μοντέλλων (Μαρκοβιανά Μοντέλλα που περιέχουν κρυφές καταστάσεις) [Hidden Markov Models (HMM)] και η ανάλυση για την απόδειξη της προτεινόμενης μεθόδου χρησιμοποιεί τόσο μαθηματικούς συλλογισμούς όσο και προσομοιώσεις. Η προσέγγιση αυτή αποτελεί ένα βήμα προς την επίτευξη αποτελεσματικής, έγκυρης και προληπτικής άμυνας κατά των επιθέσεων DoS / DDoS . Επιπλέον, εξετάζεται η απόδοση της μεθόδου σε σχέση με άλλες μεθόδους αλλά και το έργο που έχει επιτελεστεί σε αυτό τον τομέα από άλλους ερευνητές. Με την παρούσα διατριβή, πρόθεση ήταν να αναπτυχθεί μια αποτελεσματική, έγκυρη μέθοδος, ικανή να εντοπίζει έγκαιρα οποιαδήποτε μη φυσιολογική συμπεριφορά από μέρους των χρηστών. Στόχος ήταν να αποφευχθεί η επέκταση κυβερνοπεριστατικού που βρίσκεται σε εξέλιξη στο αρχικό του στάδιο και πριν αυτό προλάβει να λάβει την τελική του μορφή, που ενδέχεται να είναι μια καταστροφική DoS/DDoS επίθεση, με ανυπολόγιστες συνέπειες.

Acknowledgments

First of all, I wish to express my sincere gratitude to my thesis supervisor Professor Christoforos Hadjicostis, who supported me during every stage of my PhD studies. His insightfull comments and novel ideas, as well as his open mindedness and structured thinking, were essential for overcoming any obstacle and for achieving all the goals during this PhD project. It was a great honour for me to work with Professor Christoforos Hadjicostis, and I thank him.

Special thanks to my thesis defence committee, Professor George Hadjichristofi, Professor Ioannis Krikidis and Professor Vassos Vassiliou, for their support and helpful suggestions, and especially to the Committee Chair Professor Georgios Ellinas for his fruitful collaboration and guidance.

Finally, I want to express my sincere gratitude to my parents who taught me to apply discipline and consistency. I also want to thank my wife Amalia, who stood by me through the whole process encouraging me in difficult times and motivating me to move on, with her love and confidence in me.

Dedication

To my wife Amalia and to my daughters Anastasia and Ioli-Maria

Publications

Submitted Journal Publications

1. M. Thoma and C.N. Hadjicostis, ”Detection of Collaborative Misbehavior in Distributed Cyber-Attacks,” Transactions on Information Forensics and Security (submitted).

Conference Proceedings

1. M. Thoma and C.N. Hadjicostis, ”Detection of Collaborative Cyber-Attacks through Correlation and Time Dependency Analysis,” Proceedings of 18th Mediterranean Electrotechnical Conference (MELECON 2016) (presented by M. Thoma).

Contents

List of Figures

Chapter 1

Introduction

This chapter provides terminology and background information related to cyber attacks, as well as the motivation, objectives and contributions of this dissertation and a general overview about its structure.

1.1 General

Classical computer security considers confidentiality, integrity and availability [1]. Integrity and confidentiality are typically achieved through the use of cryptographic methods or other related procedures. Availability is the property that captures the ability of a given communication and information system (CIS) to operate normally and serve its users. In general, CIS is any system, which handles information in electronic form (stand alone or plugged in a network).

Nowadays CIS are major assets in daily life. Almost everybody has a mobile phone, a tablet, a PC. Further to our digital devices, we also have our digital identity, which is determined through the internet applications we use such as, social media, email, financial services, and other. Taking into account the above, new security matters arise such as email harvesting, identity hijacking or extortion. In most of these cases, we are the victims, but there is one special case in which we are the victim but also the attacker at the same time. This case is when we become part of a botnet that executes a denial of service (DoS) attack. Once someone is hacked, its computer becomes a remote robot, connected in a vast network of compromised computers and all of these computers, are in the control of someone else. The botnet network has the ability to cause a Denial of Service attack. The main objective of this dissertation is to recognise cyber incidents and obtain a method for judging early enough any collaborative misbehaviour. The first part of our work was published in [2].

1.2 Denial-of-Service (DoS) Attacks

DoS attacks are typically collaborative and target specific victims aiming to exhaust all of their resources (routers, servers, CIS computational strength, etc.). For example, if one CIS is flooded with an overly large number of data requests or/and input through the network, the CIS will be unable to cope and will eventually stop functioning and become unavailable for the users. This kind of cyber attack is called DoS attack. The DoS can be split to those that exhaust the resources of honest participants (such as flooding attacks [3], [4] or spam attacks) and to those which are implemented through malicious activities (target the protocols or/and exploit their vulnerabilities) [5], [4]. In order to be protected from this kind of cyber attacks, we need to detect and identify early enough any abnormal traffic in the CIS. This can be accomplished by assessing indications that may be available, and correctly identifying them before any damage is caused.

When dealing with security, we need to identify possible security gaps and built appropriate security procedures. In the case of a DoS attack, in order to be effective, we need to specify (at the stage of preparation) respective patterns, related with the attacker's coordination strategies, the ways in which possible attacks can be recognized, the main possible steps of the attacker, the overall strategy of the attacker, appropriate preventive actions (to be taken before the intrusion in order to avoid it) and necessary improvements in our response during an event (in terms of operation procedures).

Due to the fact that we lack information about the attacker, our approach relies solely on an assumption about the user's behaviour in the network. Specifically, our method is based on effectively analyzing the user's behaviour and actions during a specific period of time, which we call the inspectable period of time. We find that this approach can help us better understand the strategy of these attacks, in order to catch them early enough (for example, by inspecting the requests directed to the possible victim).

To build our approach, we seek information from related work [6] and we take into account the following:

- How can the coordinated attackers shut down a system (service-request procedure)?
- How important is data analysis of the network and behaviour analysis of the users, in order to recognise possible threats (off-line situation)?
- How can continuous traffic monitoring of the network (on-line situations) be exploited to identify DoS attacks?

DoS attacks have increased dramatically in recent years. It is very hard to defend against them and, when successful, they manage to misuse mainly resources of the network and transport layers. In general, the goal of a DoS attack is to consume the resources of the host and the bandwith of the network. Recent history shows that it is very difficult to authenticate whether an event comprises normal traffic or a malicious attempt/attack [7]. In addition, DoS attacks have one special characterestic which distinguishes them from other network attacks [8]: in order to attack, there is no need for the intruder to penetrade and exploit the target network.

1.3 Distributed Denial of Service (DDoS) attack

Distributed denial of service (DDoS) attack is a special form of DoS attack where a large number of infected systems, with malicious software, aim at a specific target causing a DoS attack. The attacker uses a multitude of computers (often referred to as zombies and/or bots) to carry out the attack. A typical example of such an attack is presented in Fig. 4.1. During the attack, the hackers mimick the features of legitimate network events [9], such as flash crowds [7], [10], in order to act under the security threshold of the network [11].

The following steps are generally followed by the attackers during a DDoS attack [12]:

- Exploitation of the used technology of the target network (i.e., protocol vulnerabilities) and establishing bots among its computers.
- Sending commands to the botnet and launching the attack.
- Finally, causing the entire system to be brought down.

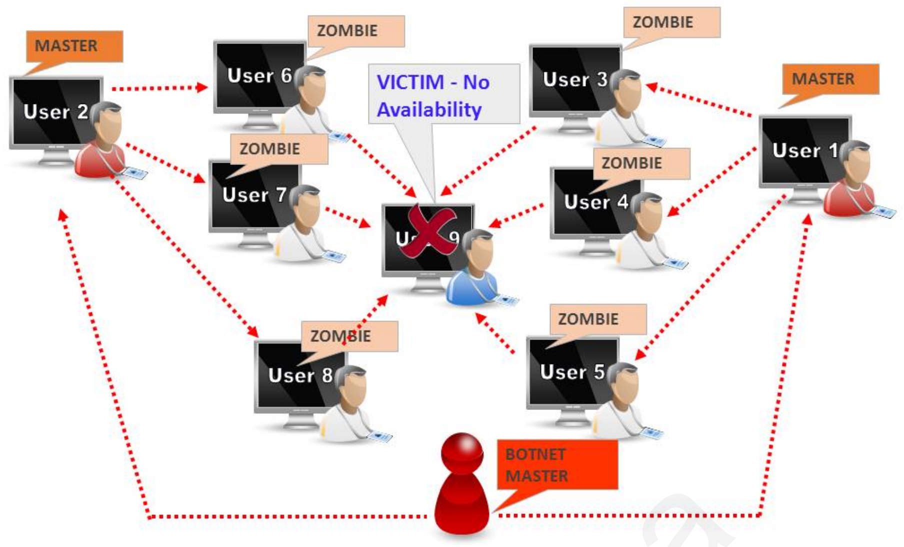


Figure 1.1: Example of a DDoS attack.

Detecting DDoS attacks poses great challenges because the Internet is an open architecture comprising of many different types of networks. As a consequence, the countermeasures that we are able to take are limited [10].

DDoS attacks raise challenges that need to be addressed proactively. We live in a cybernetic world, where behind almost any activity of our daily life there is a hidden Communication Information System (e.g., smartphones, computers, etc.). The objective of these Communication Information Systems (CIS) is to serve us. The availability of CIS is crucial and vital. This issue becomes a major challenge if we take into account that the majority of these CIS are responsible for functions related to security, navigation, transportation, communication, financial transactions and even health issues. For these applications, delay in the provision of services has great cost and in some of the cases it may even be life-threatening. The contingency plans that are in effect in the above areas of interests are not enough. In almost all cases, when something really costly happens during a DDoS attack, the primary objective of the contingency plans is to minimize any further damages.

1.4 Thesis Motivation

Taking into account all the above, our motivation for working on the topic was the rapidly increasing number of zero day attacks in recent past. If we take into account the realization that all of these CIS devices will become a part of the IoT in the next years, then we can begin to envision the structure of the possible future robot network(botnet). The effective defence to this scenario is the identification of the botnet network early enough, before it gains its full power.

1.5 Thesis Objectives

Since there is no way of knowing the strategy of the botnet or other related information and we can only observe its behaviour in the network, our objectives were to implement a reliable method in order to (1) avoid the consequences of a Distributed Denial of Service attack, (2) recognise early enough any cyber incidents, and (3) find any collaboration/dependency among different users and (4) to ultimately isolate any abnormal behaviour.

1.6 Thesis Contribution

The contribution of this dissertation is a simple and effective methodology for detecting DoS attacks. The proposed methodology can potentially be implemented in any kind of information system, including anti-DDoS devices or an Intrution Detection Systems (IDS), or can be used as a second line of defence in relation with other methods. In order for this method to be implemented, there is no need to have any related information about the botnet nor its strategy. The method is based on the theory of hidden Markov models. Furthermore, the method gives a novel approach in malware analysis, and provides novel ideas for the future research of detection of malicious internet users.

1.7 Thesis Organization

The remaining parts of the thesis are organized as follows. Chapter 2 provides background material and presents the state of the art-best practices regarding DDoS attacks. In Chapter 3, we provide the definition of a DoS attack, the description of hidden Markov models and related theory, and the theoretical analysis of the proposed approach. Chapter 4 describes our enhanced proposed approach, which includes an improved detection strategy; lessons learnt are also discussed in detail. Chapter 5 includes simulations, an evaluation of the performance of the proposed model with real data, and a summary of the main findings regarding our evaluation. In Chapter 6, we conclude the thesis with a summary of our findings and some directions for future research in this area.

Chapter 2

State of the Art

2.1 Background Information

DDoS attacks can be split into categories according to the following criteria:

- Intensity/impact of the attack (typical DDoS and Low-rate DDoS attack) [4];
- Type (direct/indirect) of DDoS attack, such as typical DDoS or Distributed Reflection DoS (DRDoS) attack [13].

The hosts of the above categories are compromised machines which have been infected by malicious code. In most cases, the attackers hide their IP trace using various techniques known as spoofing [10]. In general, the used countermeasures consist of three components: detection [14], [15], [16], [17], [18], defense (or mitigation) [7], [19], and IP trace back [20], [21].

A typical DDoS attack is achieved through the use of master zombies and slave zombies. The attacker coordinates and orders master zombies, which in turn coordinate and trigger slave zombies. Specifically, the attacker sends certain commands to master zombies in order to activate all the attack processes that are in hibernation, waiting for the appropriate command. After that, the master zombies send attack commands to slave zombies, which in turn attack the victim by sending a huge volume of packets, flooding its system with useless load and exhausting all of its resources

The difference between DRDoS and the typical DDoS is that the DRDoS attack is achieved through the use of master zombies, slave zombies, and reflectors. The attack idea is similar to the typical DDoS, with the main difference being that the slave zombies are instructed by master zombies to send a stream of packets with the victim's IP address as the source IP address to other uninfected machines, known as reflectors, exhorting them to connect with the victim. With this technique, the attacker leads the reflector to send an even greater volume of traffic to the victim, believing that the victim was the host who asked for the connection [22].

Low-rate DDoS is an intelligent attack [23], [24], [25], [26], [27] where the attacker sends attack packets to the possible victim, under the threshold traffic, and in that way deceives the defense mechanism of the network since the traffic appears to be normal, while the Low-rate DDoS attack manages to consume vital resources of the network.

Large scale DDoS attacks may use all mechanisms mentioned above (intensity, propagation) during different attack stages. Furthermore, DDoS attacks based on protocol vulnerabilities of the TCP/IP can be split into four categories (Application, Transport, Internet and Network Access [4]).

Unfortunately, until now DDoS attacks (volume attacks) remain a huge threat. The defense against DDoS attacks becomes a continuous struggle in order to be able to identify any abnormal events. There is no clear evidence of the time, place and identity of the intruder, it looks like a catch-me-if-you-can game [22]. Proposed defense mechanisms for this kind of attacks are split into four basic categories, namely attack prevention, attack detection, attack source identification, and attack reaction [4].

A good example of a DDoS attack and the consequences it can have is the 2007 cyber-attack in Estonia. The country experienced a huge DDoS cyber attack where thousands of PCs acted as zombies and targeted Estonian websites with political, commercial, govermental, economic and financial value. Until that time, it was the first experience of such a type of DDoS attack, not only for Estonia. The attack was conducted in a specific way and had unique characteristics related to the operational plan, the intensity of the attack, the resources used, and the chosen targets [28], [29], [30], [31].

2.2 DDoS Attacks in Different Networks

The scope of this subsection, is to indicate types of networks in which DDoS attacks have been implemented. As it will be shown next, DDoS are implemented in a variety of networks, such as Wireless Sensor Networks, Software Defined Networks, Cloud Computing, Cyber Physical Systems, Internet of Things Networks, and others. The ability of DDoS attacks to manifest themselves in diverse types of networks is an indicator of their capability. The above discussions lead to the conclusion that appropriate security measures need to be taken for DDoS attacks. This is vital for ensuring the availability of the above mentioned Communication Information Systems, and for enhancing the resiliency of their services.

2.2.1 DDoS Attacks in Wireless Sensor Networks

According to [32], Wireless Sensor Networks (WSNs) are at high risk of DDoS attack. The objective of this paper is to design a secure routing scheme, which can effectively protect the wireless sensor network against DDoS attacks. The methodology introduced in [32] refers to a secure routing protocol for WSNs, which is implemented under the following security modules: (1) Propagate DDoS attack on a normal network; (2) Enable the network to overcome the attack infection. The methodology is implemented with two algorithms: the first algorithm launches the DDoS attack in the normal scenario and the second algorithm safeguards the network from the effects of the DDoS attack. The proposed methodology provides a security module to prevent the network from the DDoS attack, by implementing their prevention algorithm with the compromised network. The authors set specific nodes as IPS nodes and then simulate, analyze and compare the performance under three scenarios namely, Normal Ad-hoc On-demand Multipath Distance Vector (AOMDV), DDoS-AOMDV and Secure-AOMDV (find the nodes which are involved in unwanted, huge and frequent message passing). The performance comparison between Normal-AOMDV, DDoS-AOMDV and Secure-AOMDV establishes that the proposed security mechanism works properly for AOMDV (it manages to prevent the network from the DDoS attack by blocking the intruder nodes). According to the proposed approach, the performance is decreasing the number of nodes increases. More information can be found in [32].

2.2.2 DDoS Attacks in Software Defined Networks

Software Defined Networks (SDN) are also at high risk of DDoS attack [33], [34]. In SDNs, DDoS attacks become more sophisticated. Specifically, the basic capabilities of SDNs, such as software-based traffic analysis, centralized control, global view of the network and dynamic updating of forwarding rules, make it easier to detect and react to DDoS attacks. On the other hand, the security of SDN itself remains to be addressed, and potential DDoS vulnerabilities exist across SDN platforms.

The authors of [34] propose a feasible system called FL-GUARD (Floodlight-based guard system) to tackle a DDoS attack, taking advantage of the SDN network architecture. First, the system realizes anti-spoofing of source IP address and analyzes a variety of amplification attacks to avoid their effect. Then, a support vector machine algorithm is used to detect attacks. Finally, a flow table can be issued to block attacks at the source port, taking advantage of the centralized control of SDNs. The authors also design a detection and defense system under an SDN architecture, which adds an anti-spoofing module of source IP and sFlow-RT Collector Components in the controller layer, and an attack detection module and an attack blocking module in the application layer. The system adds an anti-spoofing module of source IP in controllers on the basis of Floodlight, an enterprise-class controller. Based on the module, the dynamic IP bindings for user access to the network via DHCP service and the configuration of static IP can be implemented regarding attack detection. The method classifies whether the flow is abnormal or not via an SVM classification algorithm, where training samples adopt a normal flow training sample set and an abnormal one. Normal samples are the records for traffic and source IP entropy when no attack is present. According to the authors, the SVM classification algorithm achieves a fully automated dynamic binding of source IP address, which could prevent spoofing. Each protected server can set different alarm thresholds, thus the method increases flexibility. The blocking of attacks uses a multi-process design, which can connect multiple attack detection sides. The experimental results show the FLGUARD system has good detection and defense effect against DDoS attacks. Further details can be found in [34].

The authors of [33] discuss the new trends and characteristics of DDoS attacks in Software Defined Networking and DDoS attacks in cloud computing environments, and provide a comprehensive survey of defense mechanisms against DDoS attacks using SDN. In addition, the authors review studies about launching DDoS attacks on the control layer, infrastructure layer and application layer of SDNs, as well as the methods against DDoS attacks in SDNs. The motivation for presenting this paper is to indicate that even SDNs are possible targets for a DDoS attack. Further details can be found in [33].

2.2.3 DDoS Attacks in Cloud Computing

Cloud computing has become a convenient way of accessing services, resources and applications over the internet. The authors of [35] review 96 publications on DDoS attack and defense approaches in cloud computing networks, published between January 2009 and December 2015; they also discuss existing research trends. A taxonomy and a conceptual cloud DDoS mitigation framework based on change point detection are presented and future research directions are also outlined.

The authors of [36] present developments related to DDoS attack mitigation solutions in the cloud. In particular, they present a comprehensive survey with a detailed insight into the characterization, prevention, detection, and mitigation mechanisms of DDoS attacks. The paper aims to highlight through the above mentioned complete survey that cloud networks are one of the main targets of DDoS attacks.

In [35], it is mentioned that the National Institute of Standard and Technology (NIST) defines the essential characteristics of cloud computing as on-demand self-service, resource pooling, rapid elasticity and measured service. The service model can be broadly categorized into

- Software-as-a-service (SaaS). In SaaS, Software is presented to the end users as services on demand, usually in a browser.
- Platform-as-a-Service (PaaS). PaaS, often referred as cloudware, provides a development platform with a set of services to assist application design, development, testing, deployment, monitoring, hosting on the cloud.
- Infrastructure-as-a-Service (IaaS). Built on top of data centers layer, the IaaS layer virtualizes computing power, storage and network connectivity of the data centers, and offers it as provisioned services to consumers.

The service model can be deployed as either a private, public, community or hybrid cloud. Despite the fact that cloud computing provides various benefits to users, there are also underlying security and privacy risks that need to be addressed correctly.

According to [35], typical DDoS detection techniques classify packet traffic as either legitimate or malicious, and can be broadly categorized into signature-based, anomaly-based and hybrid.

Furthermore, several DDoS defense solutions have been proposed in the last two decades. DDoS defenses proposed for cloud services are categorized according to [35], using the DDoS defense taxonomy outlined below (more detailed descriptions can be found in [35]).

- Cloud DDoS defense deployment. DDoS defenses for cloud services can be deployed in four key locations as follows :
  - Source-end deployment. The advantages of source-end deployment include more effective protection of network resources and bandwidth.
  - Access point deployment. Access point deployment is usually deployed in the front-end, back-end or on each virtual machine(VM) in the cloud computing environment.
  - Intermediate-network deployment. These are defenses deployed on network nodes to limit the impact of DDoS attacks on the network before the attacks affect the intended target.
  - Distributed defense. Distributed defense is a hybrid deployment model comprising source-end, access point and/or intermediate network deployments.
- DDoS detection. Typical DDoS detection techniques classify packet traffic as either legitimate or malicious, and can be broadly categorized into signature based, anomaly based and hybrid.
  - Signature based detection. This technique uses a set of rules and known signature attack patterns stored in a knowledge database. Traffic patterns are monitored and compared against existing signatures to detect malicious traffic
  - Anomaly based detection. Anomaly based or behavioral classification approach involves the collection of a normal traffic behavioral profile pattern over a pre-determined period.
    * Anomaly detection techniques. In categorizing anomaly detection of cloud DDoS attacks, we group existing techniques into different ring classes based on the algorithm(s) used.

        · Statistical anomaly detection. The statistical features of a normal traffic are compiled to generate a normal traffic pattern, which will be compared with incoming traffic to detect anomaly packets.
        · Data mining. The significant increase in Internet traffic complicates efforts to detect DDoS anomaly patterns. To address this, one can use a map reduce model, which is a parallel processing model that has been used to expedite batch job operations.
        · Artificial Intelligence. It is a soft computing approach, based on techniques such as Genetic Algorithms, Artificial Neural Networks and Fuzzy Sets; it requires a continuous learning process to effectively detect new anomalies.
        · Classifier. These are techniques that learn from a set of labeled data instances in order to classify a test instance into one of the classes.
        · Machine learning. Deploying machine learning to detect cloud DDoS attacks encompasses techniques, such as statistics and data mining, but these techniques have a subtle difference from statistical techniques.
    – Hybrid detection. This approach involves the use of both signature-based and anomaly-based techniques.
    – Traceback and IP spoofing detection. This technique can help to locate the true source of DDoS attacks, as these attacks tend to spoof their addresses (e.g. launching a reflector attack).
    – Other forms of DDoS attack defenses. For example, in solving a DDoS attack issue in cloud computing, we can consider the scenario where an individual cloud user is being targeted. In this approach, an intrusion prevention system (IPS) can be deployed at different access points of the cloud environment to monitor incoming packets during DDoS attacks. This is a reactive method that dynamically allocates available resources during DDoS attacks to compensate for the attacks.

According to [36], the taxonomy of the DDoS solutions are the following (Further detailed descriptions can be found in [36]):

- Attack prevention. DDoS prevention in the cloud is a pro-active measure, where suspected attacker requests are filtered or dropped before these requests start affecting the server. Prevention methods do not have any presence of attack state as such, which is usually available to the attack detection and mitigation methods.
  - Challenge Response. Challenge-Response Protocols (CRP) are designed to identify the presence of real users. Many times, this concept has been applied in an opposite manner, where the protocol tries to determine if the user is a bot/attacker machine, especially in the case of crypto-puzzles or proof-of-work.
  - Hidden Servers/Ports. This is an important method to remove a direct communication link between the client and the server. The objective of hiding the servers is achieved by keeping an intermediate node/proxy to work as a forwarding authority. The important jobs of this forwarding authority may include balancing the load among the servers, monitoring the incoming traffic for any vulnerability, and fault-tolerance and recovery of the servers.
  - Restrictive Access. These techniques are basically admission control methods to take preventive action against the service capacity. Some of these strategies have implemented the prevention by delaying responses/access to the suspected attackers or even additional clients. In many of the contributions, this delay is introduced by prioritizing the legitimate clients or selecting clients with good past behaviors.
  - Resource Limit. The economic bills generated by a DDoS attack can be enormous. Resource limits can help in preventing these economic losses by correct auto-scaling decisions. However, deciding whether the resource surge has come due to the DDoS attack or due to the real genuine traffic, is a very difficult task.
- Attack detection. Is achieved in a situation where attack signs are present on the server in terms of its services and monitored performance metrics. These attack signs are initial signs, where the attack has just started to take the shape, or there may be a situation, where the attack has already deteriorated the

performance. These methods may seem to be similar to attack prevention at times, and many contributions have provided solutions in the same manner.

– Anomaly Detection. Anomalous patterns are usually identified from packet traces, established connections, web access logs or request headers. The specific pattern to identify in the log or the trace is decided by attack traces and other past historic behaviors.
– Source and Spoof Trace. Multiple trace back algorithms have been proposed in the literature, which identify and stop the spoof attack by tracing the source. Source traceback schemes are employed to stop/detect the identity spoofing techniques. These techniques are important as most of the detection/prevention methods model the user behavior or profile based on some identity which is mostly an IP address in case of web access. In the attack cases where IP spoofing is employed, the detection mechanisms can be defeated very easily
– Count Based Filtering. This specific classification on Count Based Filtering also fits in few attack prevention mechanisms as well, however, many times thresholds are used to detect the initialization of attack and later to identify the presence of the attack. The parameters on which these count thresholds are applied are basically network resources such as hop-count, number of connections or number of requests in a unit time from a single source.
– BotCloud Detection. Any cloud DDoS attacker may also use cloud infrastructure for its own nefarious purpose. Cloud infrastructure can be used for the purpose of installing botnets. These clouds are known as BotClouds. This subcategory describes the contributions which tries to find or detect the internal attack VMs in the cloud network. Most of these BotCloud related solutions are source based or Cloud Service Provider (CSP) based approaches.
– Resource Usage. Utilization of various resource of the cloud or a physical server by a VM can also provide important information about the presence of the DDoS attack or an anticipation of the upcoming DDoS attack. Cloud environments run Infrastructure as a Service cloud using virtualized servers where hypervisor can monitor the resource usage of each VM

on physical server. Once these VMs start reaching the decided resource utilization thresholds, the possibility of an attack can be suspected.

- Attack mitigation. Attack mitigation refers to all methods which would help a victim server continue serving requests in the presence of an attack.
  - Resource Scaling. Dynamic auto-scaling of resources is one of the most popular features of the clouds. It is also treated as one of best mitigation methods to counter DDoS attack allowing server availability or continuity with scaled resources.
  - Victim Migration(VM). VM has changed the way the entire running server is shifted to another physical server without noticeable downtime. Migration can be used to shift the victim server to a different physical server, which is isolated from the attack and once the DDoS is detected and mitigated, the server can again be shifted back to the actual place.
  - OS Resource Management (ORM). These OS level resource management methods argues that DDoS attacks being the resource intensive attacks may affect the overall mitigation methods running inside the victim VMs. By minimizing the contention at the level of the operating systems, mitigation and recovery can be expedited.
  - Software Defined Networking (SDN). SDN is an emerging reconfigurable network paradigm which may change the whole DDoS mitigation space. SDN in its core separates data and control planes of switching to support the network reconfigurability on the fly.
  - DDoS Mitigation as a Service (DMaaS). There are multiple cloud based service/third party services which are capable of providing DDoS protection. Mostly, DDoS protection is done on a server or an intermediate node forwarding packets to the server. There are solutions which are hosted in the cloud and provide DDoS mitigation as a service.

The motivation of presenting [36] is to indicate that cloud networks are possible targets for a DDoS attack.

2.2.4 DDoS Attacks in Cyber Physical Systems

According to [37], recent years have witnessed the surge of significant interest in security issues in cyber-physical systems (CPS). Consider, for example, malicious cyber attacks in a remote state estimation application where a smart sensor node transmits data to a remote estimator equipped with a false data detector. The authors of [37] consider deception attacks in a remote state estimation scenario. They propose optimal linear deception attacks on the sensor data without being detected by a false data detector at the remote state estimator. The need for analyzing the consequences of deception attacks on a dynamic system is important, because, in order to propose effective countermeasures, one needs to understand what the worst attack might be. The problems that are answered according to the authors are the following. (1) What are the possible attack strategies under which the attacker remains undetectable to the false data detector? (2) What is the corresponding estimation error at the remote estimator under such an attack? (3) Does there exist an optimal attack strategy that renders maximum estimation error?

Furthermore, [37] proposes a novel type of linear attack strategy (using Kalman filters) and presents the corresponding feasibility constraint (which guarantees that the attacker can successfully inject false data and remain undetected by the false data detector) and computes the evolution of the estimation error covariance at the remote estimator (also analyzing the degradation of system performance under various linear attack strategies).

2.2.5 DDoS Attacks in Internet of Things Environments

Internet of Things (IoT) is a term that refers to the vast network of devices connected to the Internet. According to [38], IoT is a network of heterogeneous devices. In that way, it opens extra channels for information transmission and remote control to our physical world and needs to be highly self-managed and self-secured [38]. For these reasons, security issues of IoT need to be properly addressed. The authors propose an IoT DDoS defense algorithm for an IoT end network, for preventive measuring and avoiding DDoS attack. The design of the defense algorithm is guided by several research motivations including ways to enable working nodes (which are mostly data collecting nodes in an IoT network), intelligently detect and avoid a DoS-like attack, and remain functioning. Also important are ways to make such intelligence lightweight and inexpensive and to make a local IoT end network sensitive to a certain attacker for a long time after the first detection of its malicious behavior. Following these questions, major types of network elements and their behavior are designed to meet the above demands in a modeled IoT end sub-network. According to the authors, in order for a working node to defend itself from DDoS attack, it should be able to distinguish malicious requests from legitimate ones. Since DDoS requests usually contain the same meaningless content, the proposed defending algorithm determines a sender is malicious according to the consistency of the content in the packets it sends. If a sender repeatedly sends request with same content, it will be flagged as an attacker. Upon the reception of a request from this exact address, the working node will refuse its request and retain bandwidth for service providing. In order to implement the above features, a list of records of served requests is maintained. Each record contains information such as sender address, the most recent request content, and a flag to mark whether a sender has been determined as an attacker. Upon the detection of repeated request content or a true flag for being malicious, service will not be provided. Furthermore, considering the limitation of the working node devices, the length of record list is maintained short.

To summarise, the authors of [38] propose a lightweight defensive algorithm for DDoS attack over IoT network environments. As explained above, the idea is to help working nodes in an IoT network distinguish malicious requests from legitimate ones, and process them differently.

2.3 Approaches in DDoS Detection

The scope of this section is to record the mathematical /logical approaches that are in use, in order to detect early enough the DDoS attack. These include neurocomputing, entropy theory, classification theory, graph theory, correlation analysis and traffic analysis. The conclusion of the above discussions is that, for the detection of the DDoS attacks, it is not sufficient to use one approach of detection. Real life scenarios indicate that the best practice, in order to detect early enough the DDoS attacks, is to implement multiple defenses.

2.3.1 Neurocomputing

Artificial neural networks could become a major asset in order to decisively address the threat of DDoS attacks [39], [40].

The purpose of [40] is to detect and mitigate known and unknown DDoS attacks in real time environments. The authors choose an Artificial Neural Network (ANN) algorithm to detect DDoS attacks based on specific characteristic features (patterns) that separate DDoS attack traffic from genuine traffic. In particular, they used a trained Artificial Neural Network algorithm to detect TCP, UDP and ICMP DDoS attacks based on their characteristic patterns (ANN learning process, Java Neural Network Simulator-JNNS). The objectives of their work is to (1) detect known and unknown DDoS attacks in real time as opposed to only detect known attacks (2) identify high volume of genuine traffic as genuine without being dropped (3) prevent DDoS attacking packets from reaching the target while allowing genuine packets to get through (4) train, deploy and test the solution in a physical environment as opposed to simulators (5) reduce the strength of the attack before it reaches the victim as opposed to near-by detection systems (6) evaluate their approach using both old and up-to-date datasets with related work, based on accuracy, sensitivity, specificity and precision. According to the authors, the detection mechanism is based on a supervised ANN (Feed-forward, Error Back-Propagation with a Sigmoidal activation function where accuracy primarily relies on how well the algorithm is trained with relevant data sets. The patterns used for training purposes are instances of packet headers, which include source addresses, ID and sequence numbers coupled with source destination port numbers. To summarise, the authors have used a trained Artificial Neural Network algorithm to detect TCP; UDP and ICMP DDoS attacks based on characteristic patterns that separate genuine traffic from DDoS attacks. The ANN learning process was started by reproducing a network environment that is a mirror image of a real life environment. Furthermore, the detection mechanism is integrated with Snort-AI[1], where it is tested against known and unknown DDoS attacks.

The authors of [39] use a lightweight method to detect DDoS attacks based on traffic flow features. According to the authors, the first challenge one should tackle

[1]Snort-AI is a family of Snort plug-ins based on Artificial Intelligence (AI) technologies (i.e. Artificial Neural Networks or Fuzzy Logic) to detect different kinds of hostile traffic.

to successfully detect a DDoS attack is the difficulty arising from packet header fields being modified to look like normal ones. As a result, distinguishing between legitimate packets of normal traffic and useless ones sent by compromised hosts to their victims is a very hard task. Another issue is that of the huge number of packets to be analyzed. These challenges together make the accuracy of detection difficult and its response time even worse. This method is implemented over a NOX-based network[2], where OpenFlow (OF) switches keep Flow Tables with statistics about all active flows. All feature information needed is accessed in an efficient way by means of a NOX controller and then processed by an intelligent mechanism of attack detection. The method consists of monitoring NOX registered switches of a network during predetermined time intervals. During such intervals, they extract existing features of interest from flow entries of all switches. Each sample is then passed to a classifier module that will indicate, using the spatial location of the winning neuron in the topological map, whether this information corresponds to normal traffic or an attack. Furthermore according to the authors, this method is in direct contrast to existing approaches, most of which require heavy processing in order to extract feature information needed for traffic analysis (the technique extracts features of interest with a low overhead). It is also able to monitor more than one observation point. The method is also very efficient at detecting DDoS attacks. It uses Self Organizing Maps, an unsupervised artificial neural network, trained with features of the traffic flow. The detection rate obtained is remarkably good as it is very close to other approaches. For this reason, the authors call their method as lightweight.

2.3.2 Entropy Theory

Entropy theory is a major asset against DDoS attacks. References [41], [42], [43], [44], [45] contribute to this effort with certain implementation methodologies.

The authors of [41] propose a novel mechanism for IP traceback (based on entropy variations between normal and DDoS attack traffic) using information theoretical parameters. According to the authors, they propose a novel traceback method for DDoS attacks that is based on entropy variations between normal and DDoS attack traffic, which is fundamentally different from commonly used packet marking techniques. They develop a categorization of packets into flows that are passing

[2]Software-defined networking (SDN) platform for building network control applications.

through routers. These flows are defined by the upstream router where a packet came from and the destination address of the packet. During non-attack periods, routers are required to observe and record entropy variations of local flows. According to the authors, the proposed traceback mechanism is effective and efficient compared with the existing methods. In particular, the proposed strategy is fundamentally different from the existing PPM or DPM traceback mechanisms, and it outperforms the available PPM and DPM methods. Because of this essential change, the proposed strategy overcomes the inherited drawbacks of packet marking methods, such as limited scalability, huge demands on storage space, and vulnerability to packet pollutions. The implementation of the proposed method brings no modifications on current routing software. Both PPM and DPM require update on the existing routing software, which is extremely hard to achieve on the Internet. On the other hand, the proposed method can work independently as an additional module on routers for monitoring and recording flow information, and communicating with its upstream and downstream routers when the pushback procedure is carried out. It is independent of traffic patterns and can archive real-time traceback to attackers. Once the short-term flow information is in place at routers, and the victim notices that it is under attack, it will start the traceback procedure. The workload of traceback is distributed, and the overall traceback time mainly depends on the network delays between the victim and the attackers. According to the authors, the strategy they proposed can trace back faster than previous works in larger scale attack networks.

The authors of [43] use a combination of unsupervised data mining techniques as intrusion detection systems. The non-existence of predefined rules to correctly identify the genuine network flow made the task of DDoS attack detection very difficult. In this paper, a combination of unsupervised data mining techniques as intrusion detection system are introduced. The entropy concept in terms of windowing the incoming packets is applied with a data mining technique using Clustering Using Representative (CURE) as cluster analysis to detect the DDoS attack in network flow. According to the authors, the proposed approach has been evaluated and compared against several existing approaches in terms of accuracy, false alarm rate, detection rate, F-measure and Phi-coefficient, with the results indicating the superiority of their proposed approach. An efficient Entropy Method with CURE (EM-CURE) is introduced. A proactive manner to detect DDoS attacks is implemented in many steps. In a preprocessing step, the entropy windows are conducted using consecutive packets in a different size. The entropy windows capture the network flow using the packet information headers and then applies the entropy for each distinct feature in window size. The entropy concept can be used to represent randomness in the network flow.

The authors of [44] empirically evaluate several major information metrics, namely, Hartley entropy, Shannon entropy, Rnyi entropy, generalized entropy, KullbackLeibler divergence and generalized information distance measure in their ability to detect both low-rate and high-rate DDoS attacks. Then, they use the above metrics to describe characteristics of network traffic data and find an appropriate metric to facilitate the building of an effective model to detect both low-rate and high-rate DDoS attacks. For their simulation, they used DDoS data sets from the MIT Lincoln Laboratory, CAIDA and TUIDS, to illustrate the efficiency and effectiveness of each metric for DDoS detection. In this paper the authors, evaluate information metric measures to detect both low-rate and high-rate DDoS attacks in real-life DDoS data sets. The following are some observations. Information entropy provides better results when one increases the order of generalized entropy in detecting both low-rate and high-rate DDoS attacks. The information distance measure also provides better result than KullbackLeibler when it increases the order of information divergence measure in detecting both low-rate and high-rate DDoS attacks. An information metric produces better result in terms of complexity because it uses a minimum number of parameters during detection. For both generalized entropy and information divergence, parameter values can be adjusted easily for better spacing between normal and attack traffic. According to the author's observation, the use of an appropriate information metric helps to magnify the spacing between legitimate and attack traffic for both low-rate and high-rate DDoS attack detection in real world network traffic. The low computing overhead is another significant advantage of such a metric in detecting DDoS attacks in near real-time. The outcome is that the use of an appropriate information metric helps magnify the spacing between legitimate and attack traffic for both low-rate and high-rate DDoS attack detection in real world network traffic. The motivation of presenting this survey paper, is to indicate the importance of using entropy theory in DDoS attack.

According to the authors of [45], for anomaly based DoS detection, the detector uses network traffic statistics, such as the entropy of incoming packet header fields (e.g. source IP addresses or protocol type). It calculates the observed statistical features and triggers an alarm if an extreme deviation occurs. Entropy features are common in recent DDoS detection publications. They are also one of the most effective features for detecting these attacks. However, intrusion detection systems (IDS) using entropy based detection approaches can be a victim of spoofing attacks. An attacker can sniff the network and calculate background traffic entropy before a DDoS attack starts and then spoof attack packets to keep the entropy value in the expected range during the attack. This paper explains the vulnerability of entropy based network monitoring systems. It also resents a proof of concept entropy spoofing attack and shows that, by exploiting this vulnerability, the attacker can avoid detection or degrade detection performance to an unacceptable level. By exploiting first the vulnerability, intrusion detection systems (IDS) using entropy based network monitoring can become useless. Specifically, the method to deceive entropy based DoS detection relies on generating spoofed packets to make the traffic entropy during the attack indistinguishable from the entropy before the attack. Entropy based detection is one of the most effective and popular approaches used in the past decade. According to the authors, the paper presents an important vulnerability of network monitoring systems using entropy and introduces a proof of concept spoofing attack showing it is possible. An attacker can deceive entropy based DDoS attack detection systems by either inserting new packets to the network to keep the observed entropy value in the expected range or by generating spoofed attack traffic that is invisible to entropy based detectors using background traffic entropy distribution. Also the attacker, can generate false positives to make the detection system unreliable. The motivation of presenting this paper is to highlight that for DDoS, the appropriate defence mechanism involves multiple solutions.

2.3.3 Classification Theory

The method proposed in [46] tries to detect the entire possible seven layer DDoS attack or application layer DDoS attacks on a web server, by using the parameters of the network packet (like http GET, POST request and delta time) in order to compute the accuracy in finding out the possible attack. The authors of [46] use different classifiers like Naive Bayes, Naive Bayes Multinomial, Multilayer Perception, RBF network, Random Forest, and others to classify the attack generated data set. Then, they compare the accuracy, true positive rate, and false positive rate of each algorithm by finding the confusion matrix. According to the authors, DDoS attack is broadly classified into two categories, network layer attack and application layer attack. The main aim of layer 3, DDoS attack, is to overwhelm the server and use up the bandwidth with floods. The motives of the application layer attack are to crash the server by low and slow connections.

Furthermore, according to the authors, the proposed method tries to detect the entire possible Layer Seven DDoS attack or application layer DDoS attacks on the web server, by extracting parameters like http count and delta time of the packet captured. A layer seven DDoS attack is low volume and acts as a legitimate transaction, thus we are not able to detect it via a firewall or an IDS system. The early stage of the proposed method captures all the packets from the attack source thereby enabling us to select parameters like the number of http GET or POST requests from a single IP address. The authors also select parameters like delta time, which can be defined as the time interval between any two consecutive http requests sent by a single IP address. Since Layer Seven or application layer DDoS attack uses the http protocol to use up the recourses in the web server, the authors consider the IP addresses having maximum number of http count towards a single IP destination address. They assume that a normal human user will not be able to send http requests one after another at high speed, and they consider the delta time between any two consecutive requests. The smaller the delta time value, the greater is possibility of carrying out the attack. According to the authors, Naive Bayes Multinomial achieves better accuracy and a smaller false positive rate. Specifically, it is not possible to achieve 100 % accuracy in detecting the DDoS attack in a network or to achieve a complete defense against these attacks at a single stage. They authors conclude that Naive Bayes Multinomial achieves 93.67 % accuracy in detecting the attacks and a small false positive rate of 3.10%.

2.3.4 Graph Theory: Clique Community

The Clique community problem is also related to DDoS attack detection [47] and is crucial to understanding the relation inside the social networks (who is related with whom).

The authors of [47] propose an efficient algorithm for k-clique community detection using Formal Concept Analysis (FCA). For its implementation, they use a typical computational intelligence technique, namely the FCA-based k-clique community detection algorithm. First, a formal context is constructed from a given social network using a modified adjacency matrix. Second, the authors define a type of special concept named k-equiconcept, which has the same k-size of extent and intent in a formal concept lattice. Then, the paper proves that the k-clique detection problem is equivalent to finding the k-equiconcepts. Finally, efficient algorithms for detecting the k-cliques and k-clique communities are devised by virtue of k-equiconcepts and k-intent concepts, respectively. According to the authors, the paper aims at exploiting the network type of community detection method with a focus on the k-clique community detection. With the help of FCAs powerful analysis ability on network topology, this paper studies the FCA-based k-cliques and k-clique community detection. According to the authors, this work is the first to study the k-cliques and k-clique community detection problems using FCA. First, the transformation from a social network to a formal context, which is an input of the FCA method, is studied; then, a formal concept lattice is obtained. Then, they prove that the problem of k-clique detection is equivalent to the problem of finding the k-equiconcepts. Finally, efficient algorithms to detect the k-cliques and k-clique communities are devised with the help of k-equiconcepts and k-intent concepts, respectively. According to the authors, the major contributions of this paper are (1) Formal Context Construction provides for a social network by using a modified adjacency matrix (2) the k-clique detection problem is shown to be equivalent to finding the k-equiconcepts in the concept lattice of a social network (an interesting conclusion is that extra k-cliques can be derived from the detected k-equiconcepts; then, an algorithm for detecting k-cliques with FCA is presented) (3) following k-clique detection, an FCA-based k-clique community detection approach is devised. They prove that the k-clique community detection problem is equivalent to finding the k-intent equiconcepts in the concept lattice of a social network. Experimental results demonstrate that the algorithm proposed in [47] has a higher F -measure value and significantly reduces the computational cost compared with previous works. In addition, a correlation between k and the number of k-clique communities is investigated. According to the authors the proposed algorithm has a higher F-measure value compared to other previous works. The proposed approach is described in detail in [47].

2.3.5 Correlation Analysis

The authors of [48] first analyzed the correlation information of flows in data center. They presented an effective detection approach based on CKNN (K-nearest neighbors traffic classification with correlation analysis) to detect DDoS attacks. The approach exploits correlation information of training data to improve the classification accuracy and reduce the overhead caused by the density of training data. Aiming to reduce the huge computational cost (minimize the complexity), the authors of [48] also present a grid-based method named r-polling method for reducing training data involved in the calculation. In correlation analysis, the computational complexity is a huge problem. According to the authors, the proposed approach is able to detect attacks by examining flow features only. With correlation analysis, the approach can improve the classification accuracy and is not affected by the density of training data. According to the authors the major contributions of this paper are (1) a design to detect DDoS attacks with high efficiency and low cost, which can quickly and efficiently identify the normal flows and abnormal ones in the data center (2) a novel approach that uses the correlation in formation and CKNN classification, which not only improves the classification accuracy, but also reduces the overhead significantly (3) experimental evaluations that show that correlation information helps reduce the size of training data, which in turn reduces overhead significantly and improves the accuracy of classification. Also, the classification of CKNN with grid mapping can provide fewer response time with low overhead. According to the authors, their method is based on flows and thus able to detect existing attacks by examining flow features only. With the correlation analysis, the authors can find the hidden relations of training data from data center, which can improve the classification accuracy and is not affected by the density of training data. To reduce the overhead o fKNN, the authors map the training data into a grid. The testing samples are only calculated with the training samples in neighboring cells instead of all the training data by using r-polling method,which can significant reduce the overhead of CKNN. Furthermore, the CKNN method is affected less by the density of training data which directly influences the efficiency and precision of the traditional KNN classifier. Otherwise, to keep the maximum correlation, they do not make any change to the original density of the training data.

2.3.6 Traffic Analysis: Flash Crowd

The authors of [49] develop a distributed change-point detection (DCD) architecture using change aggregation trees (CAT). The idea is to detect abrupt traffic changes across multiple network domains at the earliest time. A community network often operates within the same ISP (Internet Service Provider) domain or the network is administered by a virtual organization spanning across multiple network domains with an established trust relationship. Early detection of DDoS attacks minimizes the flooding damages to the victim systems serviced by the provider. The system is built over attack-transit routers, which work together cooperatively. Each ISP domain has a CAT server to aggregate the flooding alerts reported by the routers. CAT domain servers collaborate among themselves to take the final decision. The methodology is the following: when the flooding traffic starts propagating towards the victim, routers along the path capture the suspicious patterns. Then each router generates an alert packet and sends it to the CAT construction server, where an alert will be raised once a CAT tree is formed. The alert packets report where the suspicious pattern are captured, from which port(s) abnormal traffic is detected, and by which port the abnormal traffic is heading. The CAT-based detection scheme consists of two algorithms. One is the algorithm for attack pattern recognition at local routers and the others for network-wide attack information fusion at the CAT server. The CAT scheme is deployed in the core network routers where high data rate and limited resource routers can share information to perform complicated security functions. According to the authors, the complexity of DDoS attack patterns grows fast, as new network vulnerability is identified and more sophisticated attack tools are available. There is no magic that can handle all types of DDoS attacks. The shared sources in collaboration Grids and community networks are especially prone to such attacks. One solution works well in a given network environment but may fail in other networks. Furthermore, to resolve policy conflicts at different ISP domains, a new secure infrastructure protocol (SIP) is developed to establish mutual trust or consensus. The DCD system was simulated up to 16 network domains on the Cyber Defense Technology Experimental Research (DETER) testbed, a 220-node PC cluster for Internet emulation experiments at the University of Southern California (USC) Information Science Institute. Experimental results show that four network domains are sufficient to yield a 98 percent detection accuracy with only 1 percent false-positive alarms. Based on a 2006 Internet report on autonomous system (AS) domain distribution, the authors prove that this DDoS defense system can scale well to cover 84 AS domains. This security coverage is wide enough to safeguard most ISP core networks from real-life DDoS flooding attacks. According to the authors, their work focuses in detection of DDoS flooding attacks against Grid resource sites or hotspot servers in community networks. They point out that it is essential to detect DDoS attacks sufficiently early before harm is done to legitimate applications. Their contribution is in early detection of (1) DDoS Flooding Wave, (2) Deployment in ISP Core Networks and (3) Tradeoffs between Detection Rate and False Alarm Tolerance.

2.4 Synopsis

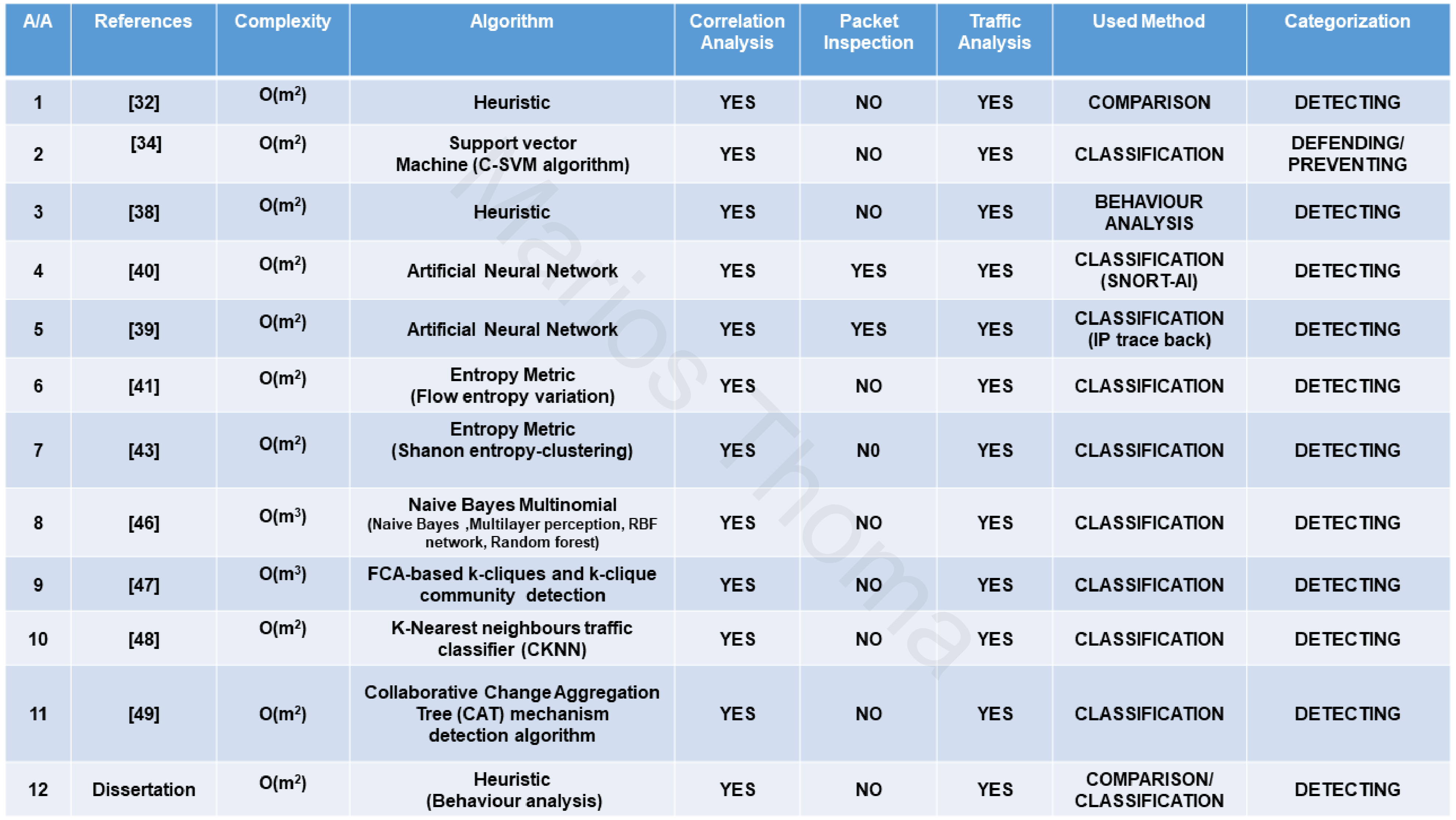
2.4.1 Categorization

This chapter presented an overview of the state of the art for DDoS attacks. The proposed approaches can be divided into two main categories. The first category reiterates the importance of DDoS attacks by describing the volume of the work that has been done to this direction ([35] and references within). The second category involves proposed approaches that discuss technical aspects of DDoS attacks, through various methodologies. The scope of these papers is to present a quantitative critical evaluation regarding the work that has been done by others researchers, related with the work discussed in this dissertation.

2.4.2 Main Remarks

In order to ultimately defend against DDoS attacks a lot of methodologies have been developed. Unfortunately, there is no method in place, until now, capable to face proactively the DDoS attacks. The growth of IoT in the coming years means that bots will be much bigger, which can be a lethal weapon in the hands of the attackers.

Regarding the work done by others in relation with the work proposed in this dissertation, the findings are presented in quantitative analysis in Figure 2.1. The main point is that our method does not need any further information related to the attacker. The other important point is that the implementation algorithm is simple and can be easily applied. Regarding all presented methods, this dissertation is close to the methods presented in [46], [47] and [49]. The difference is that our method is very simple in terms of its implementation, and its complexity with respect to the number of involved users is low, especially if we use distributive command-control and processing technics.



A/A	References	Complexity	Algorithm	Correlation Analysis	Packet Inspection	Traffic Analysis	Used Method	Categorization
1	[32]	O(m^2)	Heuristic	YES	NO	YES	COMPARISON	DETECTING
2	[34]	O(m^2)	Support vector Machine (C-SVM algorithm)	YES	NO	YES	CLASSIFICATION	DEFENDING/ PREVENTING
3	[38]	O(m^2)	Heuristic	YES	NO	YES	BEHAVIOUR ANALYSIS	DETECTING
4	[40]	O(m^2)	Artificial Neural Network	YES	YES	YES	CLASSIFICATION (SNORT-AI)	DETECTING
5	[39]	O(m^2)	Artificial Neural Network	YES	YES	YES	CLASSIFICATION (IP trace back)	DETECTING
6	[41]	O(m^2)	Entropy Metric (Flow entropy variation)	YES	NO	YES	CLASSIFICATION	DETECTING
7	[43]	O(m^2)	Entropy Metric (Shanon entropy-clustering)	YES	N0	YES	CLASSIFICATION	DETECTING
8	[46]	O(m^3)	Naive Bayes Multinomial (Naive Bayes ,Multilayer perception, RBF network, Random forest)	YES	NO	YES	CLASSIFICATION	DETECTING
9	[47]	O(m^3)	FCA-based k-cliques and k-clique community detection	YES	NO	YES	CLASSIFICATION	DETECTING
10	[48]	O(m^2)	K-Nearest neighbours traffic classifier (CKNN)	YES	NO	YES	CLASSIFICATION	DETECTING
11	[49]	O(m^2)	Collaborative Change Aggregation Tree (CAT) mechanism detection algorithm	YES	NO	YES	CLASSIFICATION	DETECTING
12	Dissertation	O(m^2)	Heuristic (Behaviour analysis)	YES	NO	YES	COMPARISON/ CLASSIFICATION	DETECTING

Figure 2.1: Critical evaluation of solutions.

Chapter 3

Baseline Approach

In this chapter, we describe notation and related theory regarding Hidden Markov Models (HMMs), and use it to develop a baseline approach for detecting collaborative (malicious) user activity.

3.1 Background on Hidden Markov Models

A hidden Markov model (HMM) is a statistical Markov model, whose states are hidden and cannot be observed directly. The hidden state feature of HMMs gives them more flexibility in modeling stochastic processes. Some uses of HMMs include applications to biological sequence analysis, pattern recognition (e.g., in speech), economic and financial modeling, signature verification and others.

3.2 Motivation for Hidden Markov Models

An HMM consists of a finite set of states, each of which is associated with a (generally multi-dimensional) probability distribution. Transitions among the states are ruled by a set of probabilities, called transition probabilities, that are generated according to a given probability distribution. Each transition is associated with an outcome (output) that is not necessarily unique (or even deterministic) for that transition. Only the outcome, but not the state, is visible to an external observer (states are hidden to the outside and this is how the name hidden Markov model arises).

An HMM can be viewed as a doubly embedded stochastic process with two hierarchical levels. The upper level is a Markov process whose states are unobservable.

Observation is typically a probabilistic function of the upper level Markov states. Different Markov states will have different observation probabilistic functions. The two level hierarchical structure is the main idea and advantage of an HMM. It can be used to model stochastic processes that are much more complicated than traditional Markov models [50].

Typically, HMMs are used as statistical models of sequential data processes. For instance, one goal is to use the HMM in order to identify the pattern of normal or abnormal behavior of a given process. The HMM model of the normal profile can be generated using historical data of the system operating under normal conditions (i.e., past observed activity of the system can be analysed to infer the parameters of the HMM model of interest [51], [52]).

In our case, the HMMs of interest are essentially partially observed Markov chains, which serve as stochastic models that represent the profile of computer events (transitions), under normal/usual operating conditions in a computer system or network.

3.3 Mathematical Description of Hidden Markov Models

3.3.1 General

A discrete time Markov chain can be viewed as a stochastic process X_n with finite state space $\theta = \{x_1, x_2, ..., x_N\}$ that satisfies the Markovian property, i.e., for all $n \geq 1$ and for all $x_{i_0}, \ldots, x_{i_n} \in \theta$, we have

$$Pr(X_n = x_{i_n} | X_{i_0} = x_{i_0}, \ldots, X_{n-1} = x_{i_{n-1}}) = Pr(X_n = x_{i_n} | X_{n-1} = x_{i_{n-1}}) \qquad (3.1)$$

where $X_k = x_{i_k}$ denotes that the process X takes value x_{i_k} at iteration k.

A time-invariant (or homogeneous) Markov model is described through a transition probability matrix that captures the transition probabilities $Pr(X_n = x_{i_n} | X_{n-1} = x_{i_{n-1}})$ (which are time-invariant and do not depend on n. If, for example, we have a three state model (i.e., $\theta = \{x_1, x_2, x_3\}$), then the transition probability matrix is a 3×3

matrix of the form

$$P = \begin{pmatrix} p_{11} & p_{12} & p_{13} \\ p_{21} & p_{22} & p_{23} \\ p_{31} & p_{32} & p_{33} \end{pmatrix},$$

where $p_{ij} = Pr(X_n = x_i | X_{n-1} = x_j)$ and remains invariant with respect to n. The matrix P is stochastic, i.e., its columns sum to 1.

3.3.2 Hidden Markov Models

Let the states of the Markov model be

$$\theta = \{x_1, x_2, ..., x_N\}$$

(where N is the number of states) and let $X_n = x_i$, denote that the Markov Chain is at state x_i at time step n. At initialization, we have some initial probability distribution that captures the probability of starting at each state:

$$\pi(0) = \begin{pmatrix} Pr(X_o = x_1) \\ Pr(X_o = x_2) \\ \dots \\ Pr(X_o = x_N) \end{pmatrix}.$$

For a Markov chain, i.e., when (3.1) is satisfied, the probabilities of being at state x_j at time step k, denoted by

$$\pi(k) = \begin{pmatrix} Pr(X_k = x_1) \\ Pr(X_k = x_2) \\ \dots \\ Pr(X_k = x_N) \end{pmatrix},$$

can be obtained iteratively via

$$\pi(k+1) = P(k)\pi(k) \ ,$$

where $\pi(0)$ are the initial probabilities and $P(k)$ is the transition probability matrix whose $(i, j)th$ entry is

$$Pr(X_{k+1} = x_i | X_k = x_j) \ . \tag{3.2}$$

If the above probabilities are not a function of k (i.e., we are dealing with a homogeneous Markov chain), then we have

$$\pi(k+1) = P\pi(k),$$

where

$$P = \begin{pmatrix} p_{11} & p_{12} & \cdots & p_{1N} \\ p_{21} & p_{22} & \cdots & p_{2N} \\ \ddots & \ddots & \ddots & \ddots \\ p_{N1} & p_{N1} & \cdots & p_{NN} \end{pmatrix}$$

with $p_{ij} = Pr(X_{k+1} = x_i | X_k = x_j)$.

In summary, a homogeneous Markov chain can be defined as a 3-tuple (θ, P, π_0), where

- $\theta = \{x_1, x_2, ..., x_N\}$ is a finite set of states;
- P is the $N \times N$ state transition probability matrix;
- π_0 is the N-dimensional initial state probability distribution vector.

A hidden Markov model (HMM) can be defined as a 5-tuple $(\theta, P, \pi_0, \Omega, \psi)$, where (θ, P, π_0) is a Markov chain, and

- Ω is the set of output symbols, i.e., the symbol values that the output Y_k of the process can take at any time step k (e.g., $\Omega = \{0, 1\}$ in most of the examples we consider later);
- ψ is the emission probability matrix of symbols in Ω and defines the probability $Pr(Y_k = \omega_l | X_k = x_j)$ of producing ("emitting") symbol $\omega_l \in \Omega$ at a given state $x_i \in \theta$ at time step k.

3.3.3 Example of a Hidden Markov Model

Modeling the Tossing of Two Different Coins

Consider the tossing of two different coins, Fig. 3.1, one of which is fair but the other is not (assume that the unfair coin comes out Heads with probability 0.75). Notice that the outcome (Heads or Tails) does not tell us which coin has been used. We assume that we start by randomly selecting one of the two coins with equal probability, and switch coins whenever Heads appears. To model this, we can use a two state hidden Markov model: state 1 represents the selection of the fair coin, whereas state 2 represents the selection of the unfair coin (of course, the state is unknown to somebody who only sees the outcome of the coin toss). The transition probabilities are $p_{11} = p_{21} = 0.5$ (because of the fair coin), whereas the transition probabilities for the unfair coin are $p_{12} = 0.75, p_{22} = 0.25$ whenever Heads appear, we switch coins. This Markov Model is hidden, because the sequence of observations (sequence of outcomes, Heads or Tails) is not uniquely associated with a sequence of states.

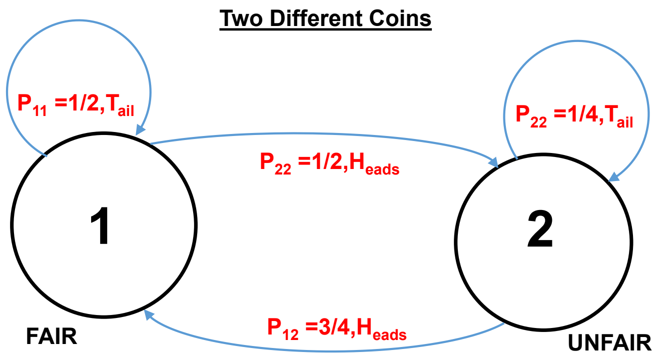


Figure 3.1: Two-state hidden Markov model - Tossing of two different coins.

3.3.4 Example of a Markov Model

Modeling the Tossing of One Fair Coin (FCM)

Consider the tossing of a fair coin, Fig. 3.2, which comes out Heads or Tails with equal probability p=0.5. We can model this as a two-state Markov chain. Each state is uniquely associated with either Heads or Tail. The transition probabilities of the fair coin model are $p_{11} = p_{12} = p_{21} = p_{22} = 0.5$). The state of the Markov chain is uniquely associated with the outcome of the toss (thus, knowing the outcome implies that we know the state of the Markov chain). This Markov model is not hidden, because the sequence of observations (sequence of outcomes, Heads or Tails) is uniquely associated with the sequence of states. Note that we could also use this model to capture the tossing of an unfair coin (e.g., $p_{22} = p_{21} = 0.75$ and $p_{11} = p_{12} = 0.25$ would correspond to a biased coin that comes out Tail 75% of the times).

3.4 Hidden Markov Models used in the Thesis

Our method relies on being able to reasonably capture the behavior of requests by a typical user to a particular point of interest by an HMM model (but not-necessarily knowing the model). As we argued in the previous subsection, an HMM is a stochastic process with an underlying state sequence that is not directly observable

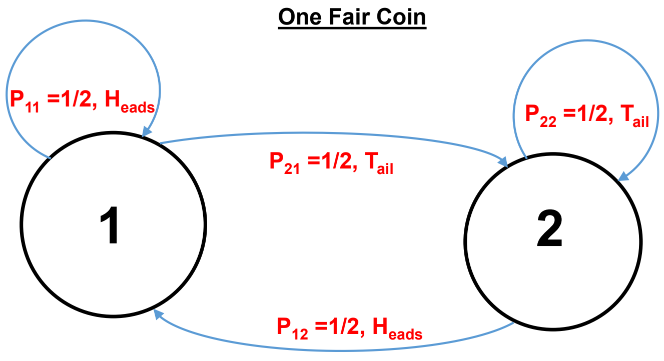


Figure 3.2: Two-state Markov model - Tossing of one fair coin.

through a sequence of observed symbols. To explain our approach, we describe two HMMs below (HMM1 and HMM2), which can be thought as representative of the typical behavior of two different users (refer to Fig. 3.3 and Fig. 3.5): HMM1 has three states and it is parameterized by r, $0 < r < 1$, whereas HMM2 has four states and it is parameterized by q and r, $0 < q, r < 1$. Both HMMs have outputs in the set $\{0, 1\}$.

HMM1 of Fig. 3.3 has transition probabilities

$$P_{HMM1} = \begin{pmatrix} 0 & 0 & r \\ 1 & 0 & 0 \\ 0 & 1 & (1-r) \end{pmatrix}.$$

HMM2 of Fig. 3.5 has transition probabilities

$$P_{HMM2} = \begin{pmatrix} r & (2*q/5) & 0 & (3*q/4) \\ (1-r) & (1-q) & (r/4) & 0 \\ 0 & (3*q/5) & (1-r) & (q/4) \\ 0 & 0 & (3*r/q) & (1-q) \end{pmatrix}$$

Figures 3.4 and 3.6 show sample runs of the HMMs in Figures 3.3 and 3.5 respectively.

3.5 Steady State Probability Distribution of a Hidden Markov Model

The state probabilities at time κ, can be obtained via the iteration $\pi(\kappa + 1) = P_{HMM1}\pi(\kappa)$, $\kappa = 0, 1, 2, \ldots$, where

$$\pi(0) = \pi_0.$$

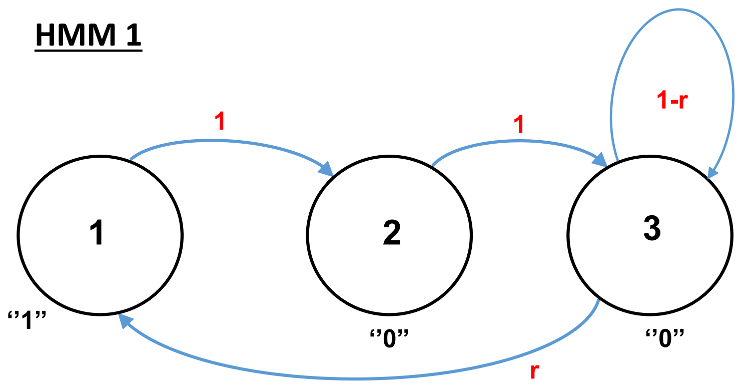


Figure 3.3: Three-state hidden Markov model.

TIME STEPS	HMM1 (r=0.6)	
	STATES HMM1	HMM1$_{sequence}$
1	state1	1
2	state2	0
3	state3	0
4	state1	1
5	state2	0
6	state3	0
7	state1	1
8	state2	0
9	state3	0
10	state1	1

Figure 3.4: One run for HMM1 and corresponding sequence of states and sequence of outputs (related to Fig. 3.3).

Steady state is reached if

$$\lim_{\kappa\to\infty} \pi_i(\kappa) = \pi_i$$

for

$$i = 1, 2, \ldots N.$$

In such case, the vector $\pi = [\pi_1, \pi_2, \ldots, \pi_N]^T$ is called the steady state probability vector and satisfies

$$\pi = P_{HMM1}\pi$$

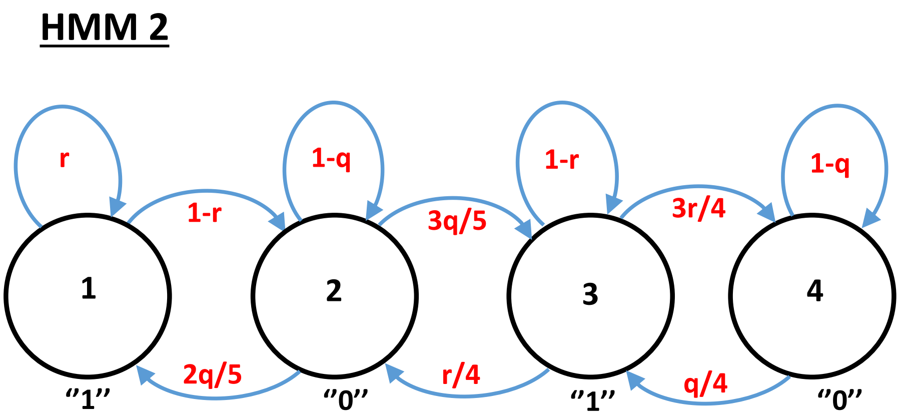


Figure 3.5: Four-state hidden Markov model.

TIME STEPS	HMM2(r=0.6 and q=0.8)	
	STATES HMM1	HMM2sequence
1	state1	1
2	state1	1
3	state1	1
4	state1	1
5	state2	0
6	state3	1
7	state3	1
8	state4	0
9	state1	1
10	state1	1

Figure 3.6: One run for HMM2 and corresponding sequence of states and sequence of outputs (related to Fig. 3.5).

and

$$1 = \pi_1 + \pi_2 + \pi_3 \dots \pi_N.$$

3.6 Proposed Approach

3.6.1 Theoretical Analysis

Let $y_1[k]$ and $y_2[k]$ be the output sequences of HMM1 and HMM2 respectively: $y_1[k] \in \{0, 1\}, \forall\, k \geq 0$, and $y_2[k] \in \{0, 1\}, \forall\, k \geq 0$. We assume that "1" denotes a request from the user, whereas "0" denotes no request from the user. Let n be the number of terms of the given output sequences $y_1[k]$ and $y_2[k]$, i.e., n is the total number of time steps (inspectable period of time). We define the indicator functions $\mathbf{I_1}$ and $\mathbf{I_2}$ as

$$\mathbf{I_1}(y_1[k]) = \begin{cases} 0, & y_1[k] = 0, \\ 1, & y_1[k] = 1. \end{cases}$$

and

$$\mathbf{I_2}(y_2[k]) = \begin{cases} 0, & y_2[k] = 0, \\ 1, & y_2[k] = 1. \end{cases}$$

The empirical frequencies $\hat{u}_1$ and $\hat{u}_2$ denote, respectively, the empirically seen percentage of time user 1 and user 2 make requests. They are defined as

$$\begin{aligned} \hat{u}_1 &= \frac{1}{n} \sum_{k=1}^{n} \mathbf{I_1}(y_1[k]) \, , \\ \hat{u}_2 &= \frac{1}{n} \sum_{k=1}^{n} \mathbf{I_2}(y_2[k]) \, . \end{aligned}$$

Similarly, we use

$$\hat{u} = \frac{1}{n} \sum_{k=1}^{n} \mathbf{I_1}(y_1[k]) \mathbf{I_2}(y_2[k])$$

to denote the *comparable frequency* that user 1 and user 2 make a simultaneous request.

If $n \to \infty$, then the empirical frequencies $\hat{u}_1 \to \mathsf{P}_1$ and $\hat{u}_2 \to \mathsf{P}_2$, where P_1 and P_2 are respectively the steady state probabilities for user 1 and user 2 to send in requests. These probabilities can be easily obtained by calculating the steady state probabilities of the underlying Markov chains (assuming that user request models are known and admit steady state probability vectors) and then taking into account the emission probabilities from each state (in our examples later on, we illustrate this process in more detail). Similarly, when $n \to \infty$, then the comparable frequency $\hat{u} \to \mathsf{P}$, where P is the empirical steady state probability that user 1 and user 2 make a simultaneous request. For large n, we expect $\hat{u}$ to approach

$$\hat{u} \to \mathsf{P} \, .$$

If the above two users are independent, the probability that they make a simultaneous request is easily seen to be $\mathsf{P} = \mathsf{P}_1\mathsf{P}_2$, which implies that

$$\hat{u} \rightarrow \mathsf{P}_1\mathsf{P}_2 \, ,$$

and (since $\hat{u}_1 \rightarrow \mathsf{P}_1$ and $\hat{u}_2 \rightarrow \mathsf{P}_2$) we have $\hat{u} \rightarrow \underbrace{\hat{u}_1\hat{u}_2}_{\hat{u}_{12}}$, where $\hat{u}_{12}$ is referred to as the *relative frequency*.

In fact, if the two users are independent, we expect to have $\hat{u}_{12} = \hat{u}$ as n grows to infinity. This relationship should hold under mild assumptions on the ergodicity of the underlying HMM models, which practically means that, during the inspectable period of time, the tactics and the strategy of the intruder do not change.

3.6.2 Methodology

Analysis of HMM1. Let $\hat{u}_1$ be the empirical frequency, which indicates the number of terms with the value "1" in the output sequence $y_1[k]$ of n bits, i.e.,

$$\hat{u}_1 = \frac{number \quad of \quad terms \quad where \quad y_1[k] = 1}{n} \, .$$

Analysis of HMM2. Let $\hat{u}_2$ be the empirical frequency, which indicates the number of terms with the value "1" in the output sequence $y_2[k]$ of n bits, i.e.,

$$\hat{u}_2 = \frac{number \quad of \quad terms \quad where \quad y_2[k] = 1}{n} \, .$$

Output Sequences. By comparing the output sequences $y_1[k]$ and $y_2[k]$ of HMM1 and HMM2 respectively, we find the comparable frequency $\hat{u}$. The idea is to identify the specific instants where both sequences (for HMM1 and HMM2) have respective terms that are equal to "1," i.e.,

$$\hat{u} = \frac{number \quad of \quad terms \quad where \quad y_1[k] = y_2[k] = 1}{n} \, .$$

If the two users are independent, then the relative frequency $\hat{u}_{12}$ and the comparable frequency $\hat{u}$, should be close to equal i.e, $\hat{u}_{12} \simeq \hat{u}$.

The conclusion of the above analysis, which is our baseline approach, is the following:

- For independent events (independent users), when $n \rightarrow \infty$, we have $\hat{u}_{12} \simeq \hat{u}$.
- For dependent events (correlated users), when $n \rightarrow \infty$, we do *not* have $\hat{u}_{12} \simeq \hat{u}$.

The above is the basic concept of our approach, in order to identify early enough any abnormal incoming or/and outgoing traffic in the CIS, before any damage is caused. By examining the correlation between terms with the value "1" among $y_1[k]$ and $y_2[k]$, we can characterize the degree of cooperation between the users in the network (for instance, during the implementation of the ARQ protocol) and subsequently identify users who deliberately, perhaps in a coordinated manner, flood the network with useless traffic.

We should point out that knowledge of HMMs that model the user behavior is *not* needed to implement the above approach. The key in the above analysis is the fact that, despite lack of all pertinent information about the users, we are able to use the only observable to us (i.e., the requests they make in the network) to identify any abnormal activities, by calculating the difference between the relative frequency and the comparable frequency; this can essentially be used as an indicator of the degree of their correlation (whether they are independent or dependent). The study of the exact degree of the correlation between users (in terms of specific values), is beyond the scope of this thesis. In practice, one should also try to identify correlations that extend over time windows (by trying the same approach on shifted versions of one of the two sequences).

3.7 Computational Considerations

3.7.1 General

As it was already mention in Section 3.6.1 (Theoretical Analysis), for large n, we expect $\hat{u}_{12}(n)$ (where $\hat{u}_{12}(n) = \hat{u}_1(n)\hat{u}_2(n)$) to approach P_{12} , i.e.

$$\lim_{n\to\infty} \hat{u}_1(n)\hat{u}_2(n) = \hat{u}(n).$$

We refer to $\hat{u}_{12}$ as the relative frequency and to $\hat{u}(n)$ as the *comparable frequency*. The above relationship should hold under mild assumptions on the ergodicity of the underlying HMM models, which practically means that the tactics and the strategy of the intruder do not change. By examining the correlation between terms with value "1" among $y_1[k]$ and $y_2[k]$, we can characterize the degree of cooperation between users in the network (e.g., as it was already told during the implementation of the ARQ protocol, etc.) and subsequently identify users who deliberately, perhaps in a coordinated manner, flood the network with unnecessary requests and traffic.

Note that the above analysis can only happen in an ideal world, so it can be taken only as a baseline for further actions. In order to identify any unusual behaviour and reach safe conclusions about the attackers, we have to deal with several parameters (e.g continous time scale, finite time windows, and others), which will be discussed later in this thesis.

3.7.2 Eigenvectors and Eigenvalues

In this section we use eigenvector decomposition to understand how the convergence to steady state occurs in a given (hidden) Markov model. We assume for simplicity that $\xi_1, ..., \xi_N$ is a linearly independent set of eigenvectors of $P_{HMM} \in \mathbf{R}^{NxN}$, i.e.,

$$P_{HMM}\xi_i = \lambda_i \xi_i, i = 1, ..., N$$

which can be expressed as

$$P_{HMM}[\xi_1, ..., \xi_N] = [\xi_1, ..., \xi_N] \begin{pmatrix} \lambda_1 & 0 & \ddots & 0 \\ 0 & \lambda_2 & \ddots & 0 \\ \ddots & \ddots & \ddots & \ddots \\ 0 & 0 & : & \lambda_N \end{pmatrix} = \Xi\Lambda \ ,$$

where

$$\Xi = [\xi_1, ..., \xi_N]$$

and

$$\Lambda = \begin{pmatrix} \lambda_1 & 0 & : & 0 \\ 0 & \lambda_2 & \ddots & 0 \\ \ddots & \ddots & \ddots & \ddots \\ 0 & 0 & \ddots & \lambda_N \end{pmatrix}.$$

where $1 \geq |\lambda|$.

We can perform eigenvalue/eigenvector decomposition analysis as described below. We write

$$P_{HMM} = \Xi\Lambda\Xi^{-1}$$

and replace the term P_{HMM} in the equation

$$\pi(k) = P_{HMM}^k \pi(0)$$

to obtain

$$\pi(k) = (\Xi\Lambda\Xi^{-1})^k\pi(0)$$

which simplifies to

$$\pi(k) = \Xi\Lambda^k\Xi^{-1}\pi(0).$$

If we put the term $c(k) = \Xi^{-1}\pi(0)$ then the above equation takes the form

$$\pi(k) = \Xi\Lambda^k c(k),$$

which, in turn, results in the equation

$$\pi(k) = \sum_{\mathbf{i=1}}^{\mathbf{n}} \Xi_i c_i \lambda_i^k.$$

Consider the Hidden Markov Models in Fig. 3.3 with

$$P_{HMM1} = \begin{pmatrix} 0 & 0 & r \\ 1 & 0 & 0 \\ 0 & 1 & (1-r) \end{pmatrix}$$

and in Fig. 3.5 with

$$P_{HMM2} = \begin{pmatrix} r & (2*q/5) & 0 & (3*q/4) \\ (1-r) & (1-q) & (r/4) & 0 \\ 0 & (3*q/5) & (1-r) & (q/4) \\ 0 & 0 & (3*r/q) & (1-q) \end{pmatrix}.$$

- For HMM1 (three state model), we have the respective diagonal matrix of eigenvalues and eigenvectors. If we use the paremeter $r = 0.7$, then

$$\Lambda_{HMM1} = \begin{pmatrix} 1.0000 + 0.0000i & 0 & 0 \\ 0 & -0.3500 - 0.7599i & 0 \\ 0 & 0 & -0.3500 - 0.7599i \end{pmatrix}$$

and

$$\Xi_{HMM1} = \begin{pmatrix} 0.4975 + 0.0000i & -0.2130 - 0.4625i & -0.2130 - 0.4625i \\ 0.4975 + 0.0000i & -0.3956 - 0.4625i & -0.3956 - 0.4625i \\ 0.7107 + 0.0000i & -0.3956 - 0.4625i & -0.3956 - 0.4625i \end{pmatrix}.$$

- For HMM2 (four state model), we have the respective diagonal matrix of eigenvalues. If we use the paremeters $r = 0.7$ and $q = 0.9$, then

$$\Lambda_{HMM2} = \begin{pmatrix} 1.0 + 0.0i & 0 & 0 & 0 \\ 0 & -0.4089 + 0.0i & 0 & 0 \\ 0 & 0 & 0.3045 - 0.2360i & 0 \\ 0 & 0 & 0 & 0.3045 - 0.2360i \end{pmatrix}$$

and

$$\Xi_{HMM2} = \begin{pmatrix} -0.8547 + 0.0i & -0.4767 + 0.0i & 0.4699 + 0.3757i & 0.4699 + 0.3757i \\ -0.3494 + 0.0i & 0.4603 + 0.0i & 0.3676 - 0.1048i & 0.3676 - 0.1048i \\ -0.3317 + 0.0i & -0.5213 + 0.0i & -0.2348 + 0.2709i & -0.2348 + 0.2709i \\ -0.1935 + 0.0i & 0.5377 + 0.0i & -0.6028 + 0.0i & -0.6028 + 0.0i \end{pmatrix}.$$

The rate of convergence of $\hat{u}_1 \rightarrow \mathsf{P}_1$ (or $\hat{u}_2 \rightarrow \mathsf{P}_2$) depends on the eigenvalues $\lambda_1, \lambda_2, ..., \lambda_N$ of the transition probability matrix P of the given HMM (here, P is an $N \times N$ matrix with N being the number of states of the HMM). Recall that an eigenvalue λ_i is a constant that satisfies $Pv_i = \lambda_i v_i$ for some N-dimensional vector v_i, called the eigenvector corresponding to λ_i. Since matrix P is column stochastic, it is well known that it has N eigenvalues, one of which is $|\lambda_1| = 1$ and the remaining satisfy $|\lambda_i| \leq 1$.

The rate of convergence of $\hat{u}_1 \rightarrow \mathsf{P}_1$ is governed by the value of $|\lambda_2|$ where λ_2 is the eigenvalue that has the second largest magnitude ($|\lambda_1| = 1$ has the largest magnitude). For example, if $\left|\lambda_{2,HMM1}\right|$ is smaller than $\left|\lambda_{2,HMM2}\right|$, then HMM1 will (generally) reach steady state faster than HMM2. As a consequence, for large $\left|\lambda_{2,HMM}\right|$, we need more inspectable period of time n, in order to safely reach conclusions about the correlation between different users.

In general, the larger the values of $|\lambda_i|$ (closer to unity) , the more difficult it becomes, to reach steady-state.This is the direct implication of the equation

$$\pi(k) = \sum_{\mathbf{i=1}}^{\mathbf{n}} \Xi_i c_i \lambda_i^k.$$

3.7.3 Complexity Analysis

The complexity of the proposed method depends on two basic factors: The number of users (m) and the number of steps (n). According to our method, there are m users that have to be formed in pairs, so the possible combinations are

$$\binom{m}{2} = \frac{m(m-1)}{2}.$$

This means that the complexity of our method is $O(m^2n)$, where n is the length of the inspectable window in the baseline approach. If we assume that we have time shifting constraints, then the complexity gets bigger and it is $O(m^2nx)$, where x is the maximum value of shifting.

3.7.4 Scalability Analysis

The scalability of the proposed method does not get affected by the increase in the number of the users under investigation. On the contrary, if we have big number of users (m) and big number of steps (n), the outcome of the method will be more reliable and stable. This happens because the large number of formed pairs among the users cover all possible failures (false negative results); furthermore, for a large number of the steps, more reliable results are obtained because more time is given for the method to reach steady state.

3.8 Summary

In this chapter we described the usage of HMMs. In our analysis we introduced the basic concept and theory of HMMs and showed how an HMM can be used to model stochastic processes that are much more complicated. For our case, HMMs of interest are essentially partially observed Markov chains, which serve as stochastic models that represent the profile of computer events (transitions), under normal/usual operating conditions in a computer system or network. To this direction, we quoted some examples, with respective discusions. We then described our baseline proposed approach and its implementation, taking into account the related theory of the HMMs. Finally, we also presented the complexity analysis of our method.

Chapter 4

Enhanced Approach

4.1 Real World Considerations

Cyber attacks in the real world have become very sophisticated, with the attackers trying to deceive the security measures that are in place. Let us consider, for instance the strategies illustrated in Fig. 4.1 and Fig. 4.2. The underlying HMMs are not necessarily governed by time invariant probabilities. As shown in Fig. 4.1, the attacker (red colored) orders the master zombies (leaders shown in, yellow, light blue and green) who in turn deliberately change the behaviour of the slave zombies (teams shown in, yellow, light blue and green) that are members of the attacker network. In Fig. 4.2, we see how this strategy is implemented at different time steps. Each user represents a slave zombie, which acts at a different time step.

If we take the above description as our baseline view of this problem, then it is clear that it is very difficult to identify the respective HMMs of each user (slave zombie). Furthermore, it is very difficult to understand the botnet's actual geo-location and its full structure. The only evidence that is viewable to the network, is the behaviour of the users (through the inspectable time steps), as manifested in terms of the traffic of the network (both incoming and outcoming traffic). These new strategies reshape and complicate the problem. The key question is how we can identify correlations between these users and categorize them in cliques, keeping in mind that their behaviour may be implemented via time-varying mechanisms. By adjusting the method of Chapter 3, in this chapter we partially address the attacker's time-varying strategies indirectly.

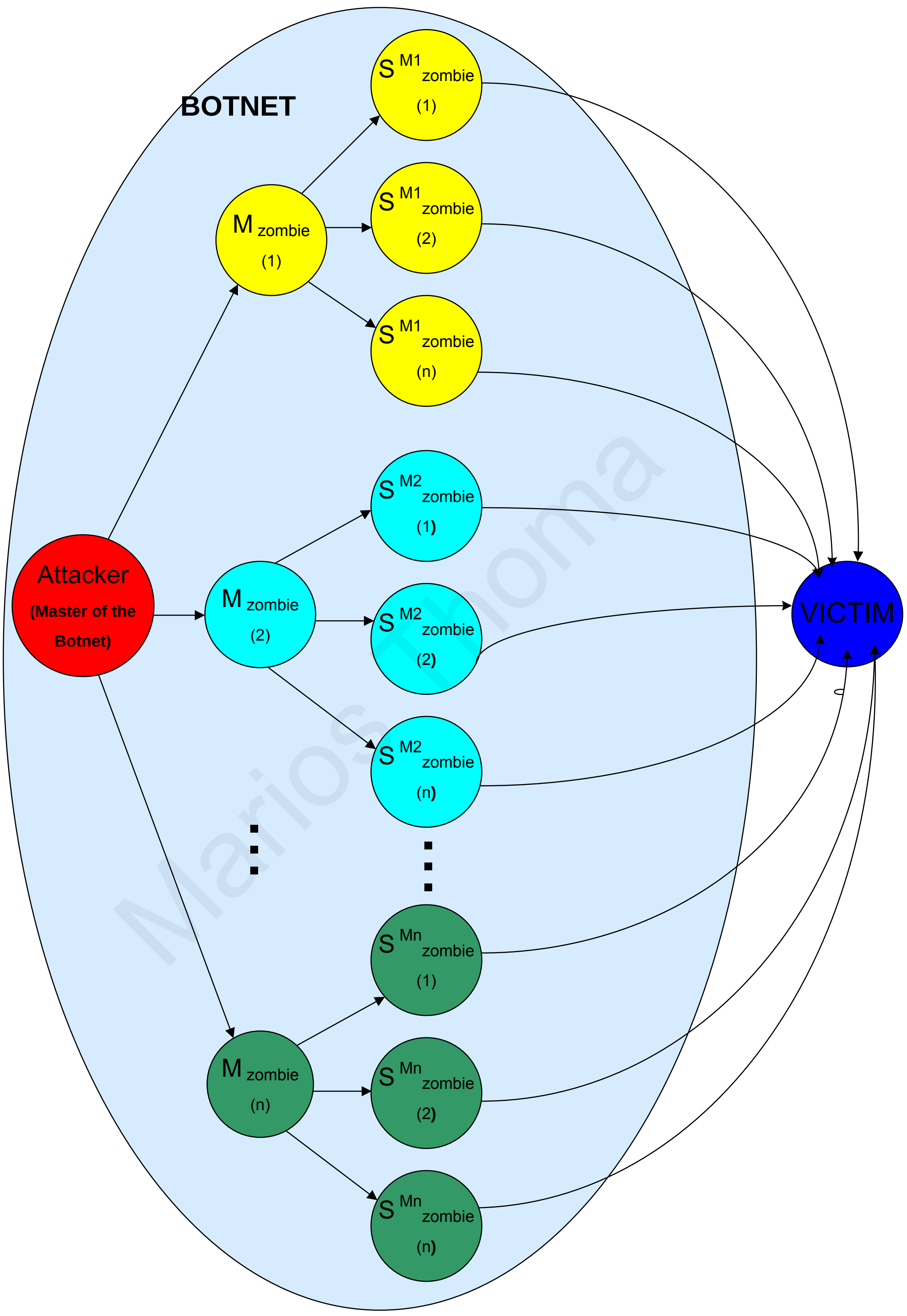


Figure 4.1: DDoS attack graph.

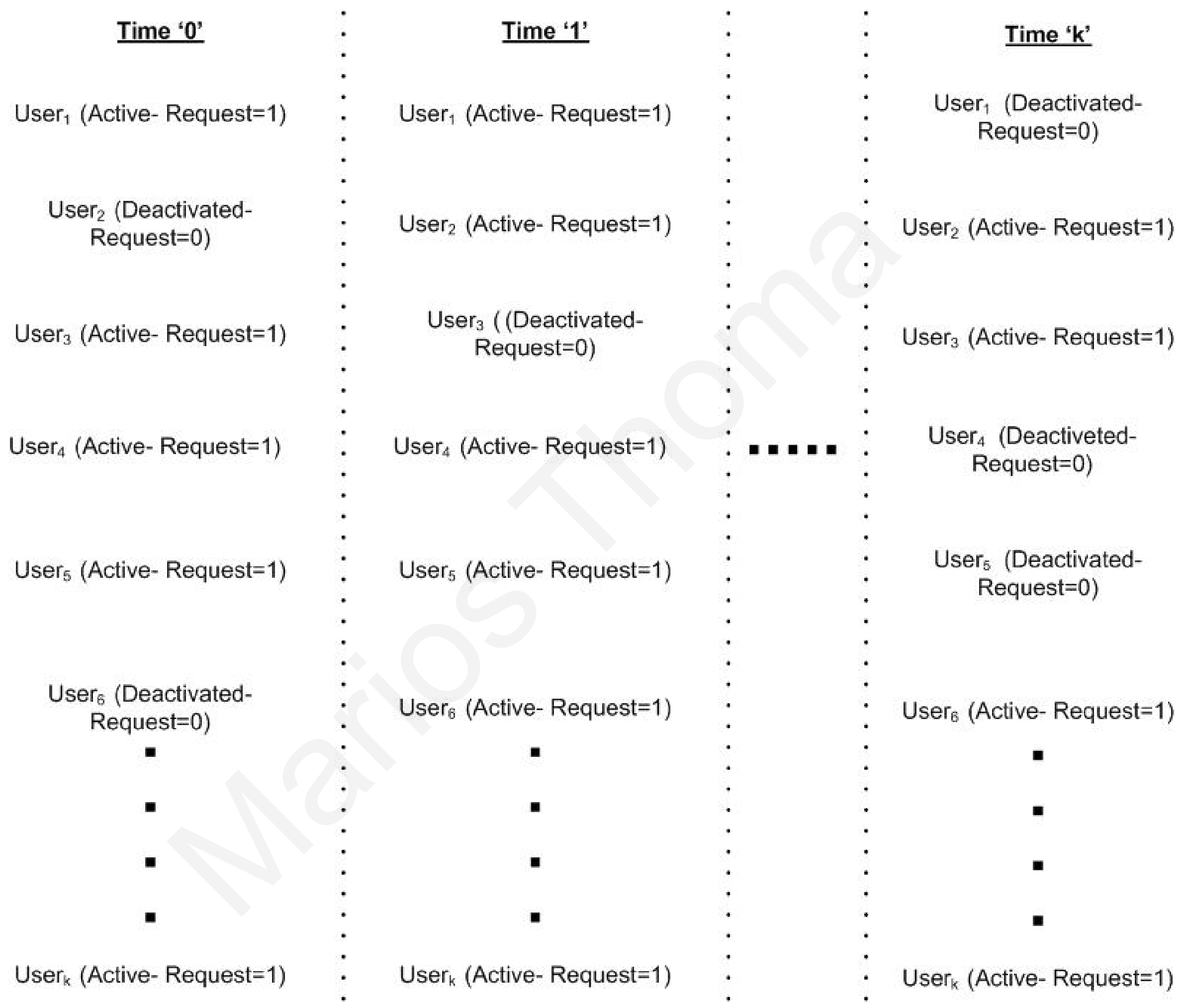

Time '0'
User$_1$ (Active- Request=1)
User$_2$ (Deactivated- Request=0)
User$_3$ (Active- Request=1)
User$_4$ (Active- Request=1)
User$_5$ (Active- Request=1)
User$_6$ (Deactivated- Request=0)
User$_k$ (Active- Request=1)
Time '1'
User$_1$ (Active- Request=1)
User$_2$ (Active- Request=1)
User$_3$ ((Deactivated- Request=0)
User$_4$ (Active- Request=1)
User$_5$ (Active- Request=1)
User$_6$ (Active- Request=1)
User$_k$ (Active- Request=1)
Time 'k'
User$_1$ (Deactivated- Request=0)
User$_2$ (Active- Request=1)
User$_3$ (Active- Request=1)
User$_4$ (Deactiveted- Request=0)
User$_5$ (Deactivated- Request=0)
User$_6$ (Active- Request=1)
User$_k$ (Active- Request=1)

Figure 4.2: User strategy for time-varying schema.

4.2 Improved Methodology

This chapter updates the prediction strategy that we introduced in the previous work chapter. The extented method relies on the analysis of data traffic across the concerned network through the use of fixed time frames, in an effort to identify correlations between different users (both incoming and outcoming traffic).

4.2.1 Lessons Learned

Real life lessons indicate that we typically have neither a clear view of when something goes wrong, nor the actual number and the identity of the users that are involved in the DDoS attack. A recent example of this was the DDoS attack powered by a new botnet dubbed Leet Botnet, which hit the network of the firm Imperva [53]. The DDoS attack took place in two waves with the first one lasting 20 minutes and peaking at 400Gbps, and the second one lasting 17 minutes and peaking at 650Gbps. The attack used spoofed IPs, making it impossible to trace the botnet's actual geo-location or learn any additional information regarding the devices which were used [53].

4.2.2 Basic Elements of Enhanced Strategy

Updating our approach in Chapter 3, we notice through simulations that when $F_{Relative} \geq F_{Comparable}$ (where $F_{Relative}$ is the relative frequency $\hat{u}_{12}$ and $F_{Comparable}$ is the comparable frequency $\hat{u}$ of Chapter 3, the involved users are probably independent. This new assumption is different from the approach expressed in Chapter 3, where independent users are those that satisfy $\hat{u}_{12} \simeq \hat{u}$.

In order to be more realistic, we separate users into several categories, in order to cover cases of interest. The main problem in real scenarios is that, we do not have concrete boundaries about the behavior of users. As a consequence, it is hard to easily index the users in good and bad guys. Taking the above into account, we separate users into the following main categories:

- Dependent users: These are users that have similar behavior, based on relative/comparable frequencies. Users with similar behavior form the dependent clique (they act as a collaborative clique).

- Independent users: These users, have no dependency between them (non-collaborative).
- Users with traffic above normal: This category of users is called flash crowd. Its activity is above the normal limit of traffic.
- Users with very high traffic: This kind of users are the abnormal users. Their activity is very high and above the normal limit of traffic, with no logical reasoning.
- Users with no activity (only when we use historical data): This kind of users are also categorized as independent (non-collaborative).

The specific criteria for the above categories of users, will be analyzed later in this chapter.

In order to implement our approach, all users in question are compared pairwise in specific time frames, which are called the inspectable periods of time, and range between 10^3 to 10^4 time steps. Chapter 3 indicated that HMMs reach the steady state condition approximately in the range of 10^3 to 10^4 time steps (depending on the complexity of the HMM that governs the user's behavior). Each time step is in the time unit of 100*ms* (their activity is represented with the values '1' or '0'). User activity/behaviour is examined through criteria related to user independency, dependency, abnormality, flash crowd behaviour, or absence of any activity.

Chapter 3 established that there is no need to have knowledge of the strategy of the attacker in order to implement the baseline approach (as long as some nominal ergodicity conditions are satisfied). As already discussed, the key in the above analysis is the fact that, despite the lack of all pertinent information about the users, we are able to use the only observable to us (i.e., the requests they make in the network for both incoming and outcoming traffic) to identify any abnormal activity (by calculating the difference between the relative frequency and the comparable frequency). This can essentially be used as an indicator of the degree of their correlation (whether they are independent or dependent).

4.2.3 Implemention

This subsection provides the necessary information related to the used methodology, in order to implement the updated approach. Further to Chapter 3, we enhance our method in two main directions.

The first direction is related to the representation and the processing of the users. Specifically, we represent the dependencies between users, using undirected graphs $G_{comparable}(V, E)$ and $G_{relative}(V, E)$, where the set of nodes V represents the set of users and the set of edges, capture the degree of correlation among them. This approach gives us the chance to create two undirected graphs with the respective square matrices $M_{comparable}$ and $M_{relative}$. Each element of $M_{comparable}$ and $M_{relative}$, is related with a pair of users, such as i and j (where $i, j \in V$). The number of rows and columns of the above square metrices is equal to the total number of users.

The second direction deals with the way we implement our approach in the real world. Regarding this, the pseudocode of Fig. 4.3 has been implemented, in order to identify correlations between pairs of users, by putting additional criteria to the approach described in Chapter 3. The goal is to identify possible relations between the users, and establish respective cliques. Specifically, the pseudocode of Fig. 4.3 has been implemented as follows:

- First Criterion: Find Dependent Users[1]

$$F_{comparable} > F_{relative}$$

- Second Criterion: Find Independent Users

$$F_{relative} \geq F_{comparable}$$

- Third Criterion: Find Users with traffic above normal

$$F_{Observable_{Rate}} \geq F_{MeanPacketRate}$$

- Fourth Criterion: Find Users with very high traffic

$$U_{Observable_{Rate}} \geq 0.9$$

- Fifth Criterion: Find Users with no activity

$$U_{Observable_{Rate}} = 0$$

The decision tree (Fig. 4.3), which implements the algorithm of the proposed approach and the respective flowchart of functions, are the following:

[1]What we know is that $F_{comparable} = F_{relative}$, if the users are independent. In practice, we need to pick a small threshold τ (based on empirical evidence) and assume that the users are independent if $F_{relative} - \tau \leq F_{comparable} \leq F_{relative} + \tau$.

- First Criterion/Find Dependent Users

  Further to our analysis in Chapter 3, after evaluating the proposed procedure, we find that when $F_{comparable} > F_{relative}$, the involved users $y_1[k]$ and $y_2[k]$, are more likely to be dependent. When the empirical frequencies of the users, for instance for $\hat{u}_i$ and $\hat{u}_j$, (where $i, j \in V$) are equal with their comparable frequency ($F_{comparable}$), then they are indicated as dependent with infinite relation.

- Second Criterion/Find Independent Users

  Independent users are those users that fulfill the equality $F_{relative} \geq F_{comparable}$. We find that when the above equality is in force, the involved users are more likely to be independent.

- Third Criterion/Find Users with traffic above the usual

  When the empirical frequency of a user is greater than $F_{MeanPacketRate}$, then that user is indicated as Flash Crowd. Users that satisfy this criterion are likely to be dependent and, as a consequence, they are assigned to the clique of collaborating nodes.

- Fourth Criterion/Find Users with very high traffic

  Using the criterion $U_{Observable_{Rate}} \geq 0.9$, we identify abnormal Users. It is a very strong criterion and the threshold can change based on empirical data.

- Fifth Criterion/Find Users with no activity

  When $U_{Observable_{Rate}} = 0$, we identify users with no activity in the inspectable time period. This criterion can be applied only if historical data is in use. The meaning of this criterion is that when we compare different windows of time, we can find relations between the users in question. Users that present no activity for the specific time window are essentially inactive for the specific inspectable period of time (fixed time frame).

The goal of the enhanced approach is to improve the methodology presented in Chapter 3 in order to deal effectively with scenarios of DDoS attacks in the real world. The next chapter will provide simulations and describe with the implementation and performance of the enhanced approach.

4.3 Summary

In this chapter we described an enhanced approach based on the proposed baseline approach. We presented the real world considerations regarding DDoS attacks and lessons learnt from real security events. Based on these observations, we updated the prediction strategy that we introduced in Chapter 3. The extended method relies on the analysis of the data across the concerned network through the use of fixed time frames, in an effort to identify correlations between different users (both incoming and outcoming traffic). Towards this direction, we presented the basic elements of an adjusted strategy. The idea is to split users, according to respective criteria, into Dependent, Independent, Abnormal, Flash Crowd and No activity users. Furthermore, we quoted the flowchart of the proposed approach and analysed the used pseudocode.

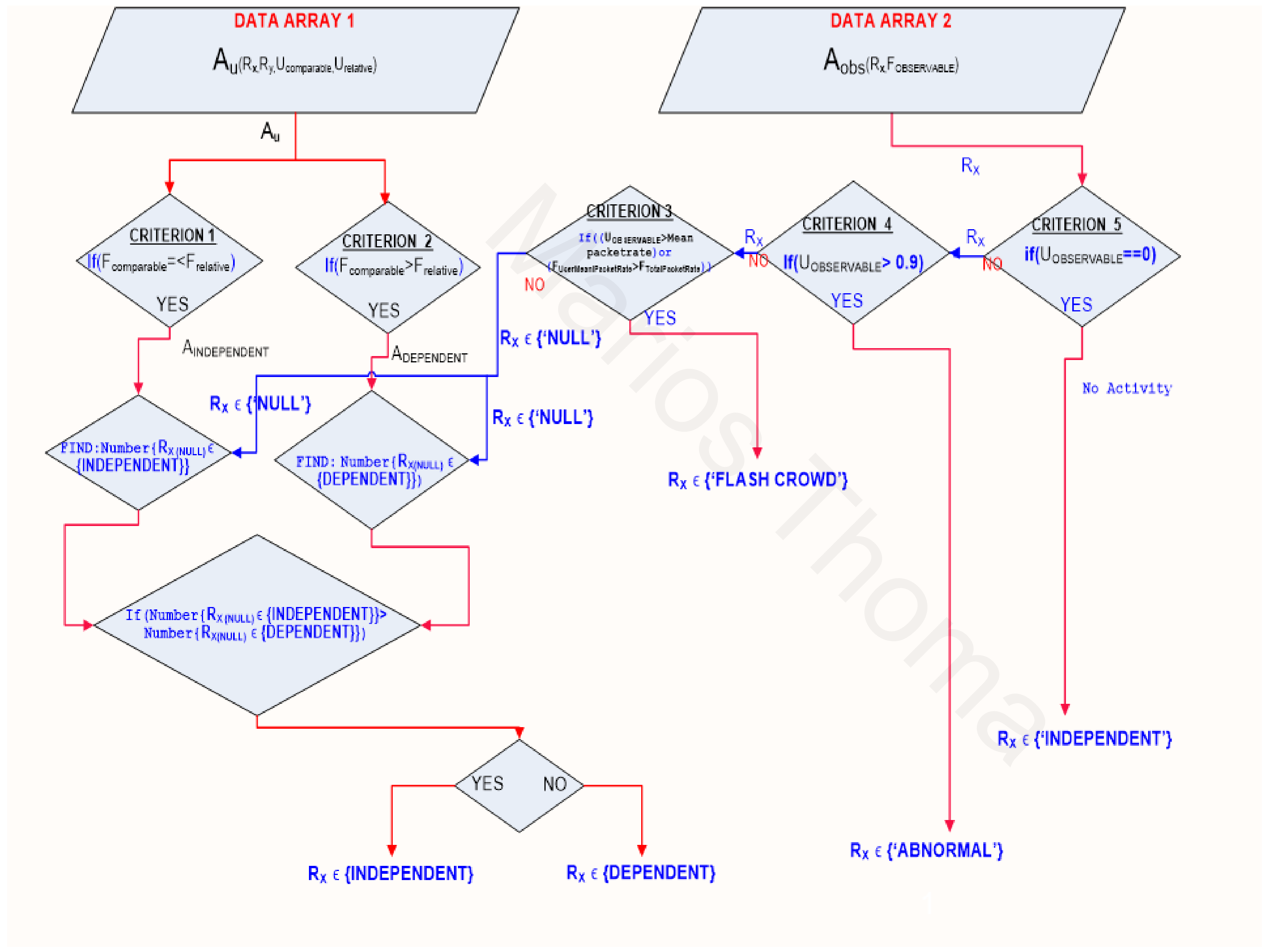

DATA ARRAY 1
$A_u(R_x, R_y, U_{comparable}, U_{relative})$
A_u
CRITERION 1
$If(F_{comparable} =< F_{relative})$
YES
$A_{INDEPENDENT}$
CRITERION 2
$If(F_{comparable} > F_{relative})$
YES
$A_{DEPENDENT}$
$R_x \in \{'NULL'\}$
FIND: Number{$R_{x(NULL)} \in$ {INDEPENDENT}}
FIND: Number{$R_{x(NULL)} \in$ {DEPENDENT}}
$R_x \in \{'NULL'\}$
If (Number{$R_{x(NULL)} \in$ {INDEPENDENT}}> Number{$R_{x(NULL)} \in$ {DEPENDENT}})
YES
NO
$R_x \in$ {INDEPENDENT}
$R_x \in$ {DEPENDENT}
DATA ARRAY 2
$A_{obs}(R_x F_{OBSERVABLE})$
R_x
CRITERION 3
If(($U_{OBSERVABLE}$>Mean packetrate)or ($F_{UserMeanPacketRate}$>$F_{TotalPacketRate}$))
NO
YES
$R_x \in \{'NULL'\}$
$R_x \in$ {'FLASH CROWD'}
CRITERION 4
If($U_{OBSERVABLE}$> 0.9)
NO
YES
$R_x \in$ {'ABNORMAL'}
CRITERION 5
if($U_{OBSERVABLE}$==0)
NO
YES
No Activity
$R_x \in$ {'INDEPENDENT'}

Figure 4.3: Pseudocode - Flowchart of the proposed approach.

Chapter 5

Evaluation of Proposed Method

In this chapter we implement the approach in Chapter 4, and evaluate its performance in small and large simulated examples, as well as real data.

5.1 Small Examples with Fixed Botnet Data

In this section we use a small example to illustrate the performance of the proposed approach. The used paremeters are those described in Chapter 3, namely n, $\hat{u}_1$, $\hat{u}_2$, $\hat{u}$, $\hat{u}_{12}$, $\lambda_{2,HMM1}$, $\lambda_{2,HMM2}$, where the models considered are the ones in Fig. 3.3 and Fig. 3.5 with parameters $r = 0.7$ and $q = 0.9$ (the selected values result in significant difference between $\lambda_{2,HMM1}$ and $\lambda_{2,HMM2}$).

- For HMM1 (three state model), we have the following transition matrix:

$$P_{HMM1} = \begin{pmatrix} 0 & 0 & 0.7 \\ 1 & 0 & 0 \\ 0 & 1 & 0.3 \end{pmatrix},$$

with the respective matrices of eigenvalues and eigenvectors given by

$$\Lambda_{HMM1} = \begin{pmatrix} 1.0000 + 0.0000i & 0 & 0 \\ 0 & -0.3500 - 0.7599i & 0 \\ 0 & 0 & -0.3500 - 0.7599i \end{pmatrix}$$

and

$$\Xi_{HMM1} = \begin{pmatrix} 0.4975 + 0.0000i & -0.2130 - 0.4625i & -0.2130 - 0.4625i \\ 0.4975 + 0.0000i & -0.3956 - 0.4625i & -0.3956 - 0.4625i \\ 0.7107 + 0.0000i & -0.3956 - 0.4625i & -0.3956 - 0.4625i \end{pmatrix},$$

i.e., $\{|\lambda_1| = 1, |\lambda_2| = 0.8366, |\lambda_3| = 0.8366\}$.

- For HMM2 (four state model), we have the following transition matrix:

$$P_{HMM2} = \begin{pmatrix} 0.7 & 0.36 & 0 & 0.675 \\ 0.3 & 0.1 & 0.175 & 0 \\ 0 & 0.54 & 0.3 & 0.225 \\ 0 & 0 & 0.525 & 0.1 \end{pmatrix}$$

  with the respective matrices of eigenvalues and eigenvectors given by,

$$\Lambda_{HMM2} = \begin{pmatrix} 1.0 + 0.0i & 0 & 0 & 0 \\ 0 & -0.4089 + 0.0i & 0 & 0 \\ 0 & 0 & 0.3045 - 0.2360i & 0 \\ 0 & 0 & 0 & 0.3045 - 0.2360i \end{pmatrix}$$

  and

$$\Xi_{HMM2} = \begin{pmatrix} -0.8547 + 0.0i & -0.4767 + 0.0i & 0.4699 + 0.3757i & 0.4699 + 0.3757i \\ -0.3494 + 0.0i & 0.4603 + 0.0i & 0.3676 - 0.1048i & 0.3676 - 0.1048i \\ -0.3317 + 0.0i & -0.5213 + 0.0i & -0.2348 + 0.2709i & -0.2348 + 0.2709i \\ -0.1935 + 0.0i & 0.5377 + 0.0i & -0.6028 + 0.0i & -0.6028 + 0.0i \end{pmatrix},$$

  i.e., $\{|\lambda_1| = 1, |\lambda_2| = 0.4089, |\lambda_3| = 0.3851, |\lambda_4| = 0.3851\}$.

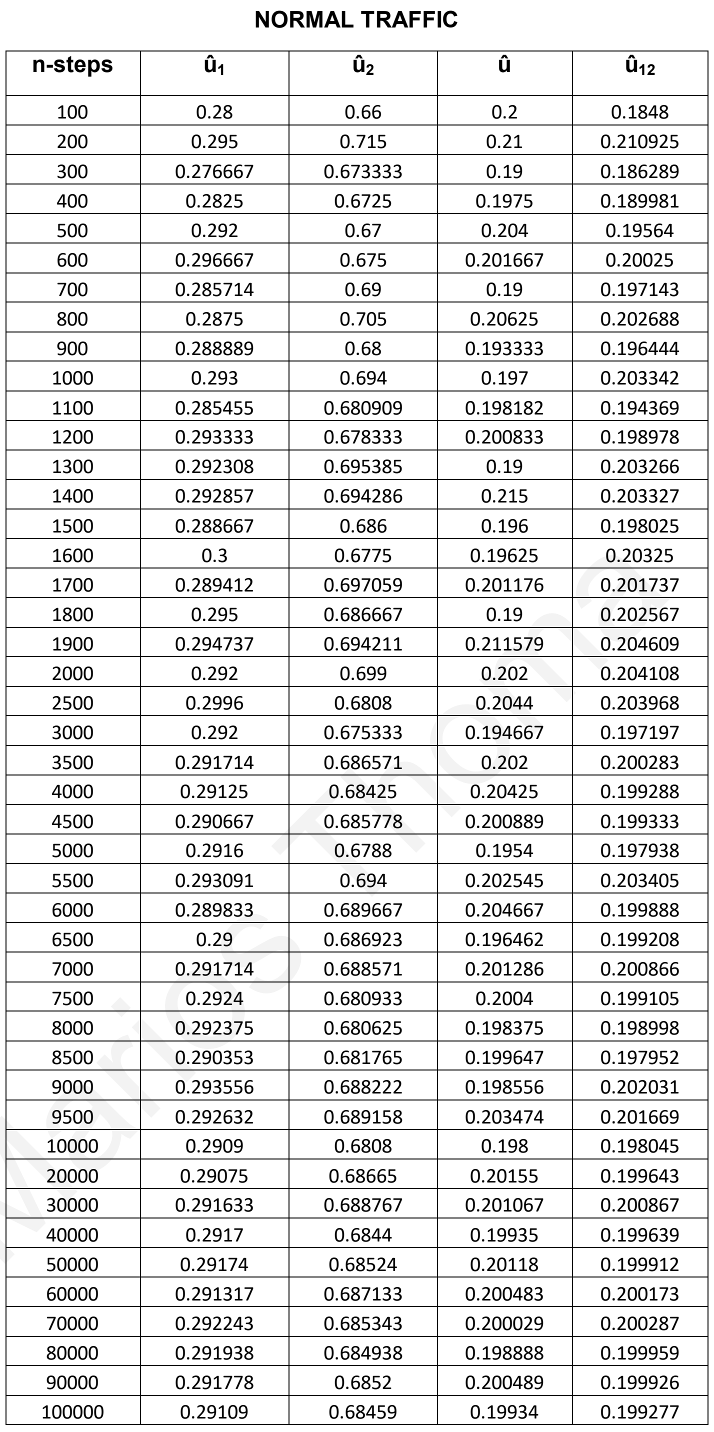
5.1.1 Normal Behaviour

In this subsection we deal with normal behaviour (refer to Fig. 5.1) among the two users, whose behaviour is captured by HMM1 and HMM2, based on the theory described in Chapter 3. The first observation is that after n steps, if we have independent users, then $\hat{u}_{12} = \hat{u}_1\hat{u}_2$ (Fig. 5.2).

The second observation is that the model with the smallest $|\lambda_2|$ reaches steady-state quicker. In our example, due to the fact that $|\lambda_{2,HMM2}| < ||\lambda_{2,HMM1}|$, HMM2 reaches steady state quicker (Fig. 5.3). Specifically, HMM2 reaches steady-state in the range of $\kappa = 12$ steps instead of HMM1 which needs to approach $\kappa = 50$ steps, where κ is the power of P^{κ}_{HMM1} and P^{κ}_{HMM2}.

5.1.2 Abnormal Behaviour

In this subsection we deal with abnormal behaviour (refer to Fig. 5.4). Following the theory described in Chapter 3, we represent the abnormal behaviour between



NORMAL TRAFFIC

n-steps	$\hat{u}_1$	$\hat{u}_2$	$\hat{u}$	$\hat{u}_{12}$
100	0.28	0.66	0.2	0.1848
200	0.295	0.715	0.21	0.210925
300	0.276667	0.673333	0.19	0.186289
400	0.2825	0.6725	0.1975	0.189981
500	0.292	0.67	0.204	0.19564
600	0.296667	0.675	0.201667	0.20025
700	0.285714	0.69	0.19	0.197143
800	0.2875	0.705	0.20625	0.202688
900	0.288889	0.68	0.193333	0.196444
1000	0.293	0.694	0.197	0.203342
1100	0.285455	0.680909	0.198182	0.194369
1200	0.293333	0.678333	0.200833	0.198978
1300	0.292308	0.695385	0.19	0.203266
1400	0.292857	0.694286	0.215	0.203327
1500	0.288667	0.686	0.196	0.198025
1600	0.3	0.6775	0.19625	0.20325
1700	0.289412	0.697059	0.201176	0.201737
1800	0.295	0.686667	0.19	0.202567
1900	0.294737	0.694211	0.211579	0.204609
2000	0.292	0.699	0.202	0.204108
2500	0.2996	0.6808	0.2044	0.203968
3000	0.292	0.675333	0.194667	0.197197
3500	0.291714	0.686571	0.202	0.200283
4000	0.29125	0.68425	0.20425	0.199288
4500	0.290667	0.685778	0.200889	0.199333
5000	0.2916	0.6788	0.1954	0.197938
5500	0.293091	0.694	0.202545	0.203405
6000	0.289833	0.689667	0.204667	0.199888
6500	0.29	0.686923	0.196462	0.199208
7000	0.291714	0.688571	0.201286	0.200866
7500	0.2924	0.680933	0.2004	0.199105
8000	0.292375	0.680625	0.198375	0.198998
8500	0.290353	0.681765	0.199647	0.197952
9000	0.293556	0.688222	0.198556	0.202031
9500	0.292632	0.689158	0.203474	0.201669
10000	0.2909	0.6808	0.198	0.198045
20000	0.29075	0.68665	0.20155	0.199643
30000	0.291633	0.688767	0.201067	0.200867
40000	0.2917	0.6844	0.19935	0.199639
50000	0.29174	0.68524	0.20118	0.199912
60000	0.291317	0.687133	0.200483	0.200173
70000	0.292243	0.685343	0.200029	0.200287
80000	0.291938	0.684938	0.198888	0.199959
90000	0.291778	0.6852	0.200489	0.199926
100000	0.29109	0.68459	0.19934	0.199277

Figure 5.1: Normal traffic - Values of various parameters of interest after the execution of a run of 100000 steps in the two HMMs.

the two users. In this example, the second user, behaves just like the first user who is governed by HMM1. In other words, the second user is mimicking the actions of

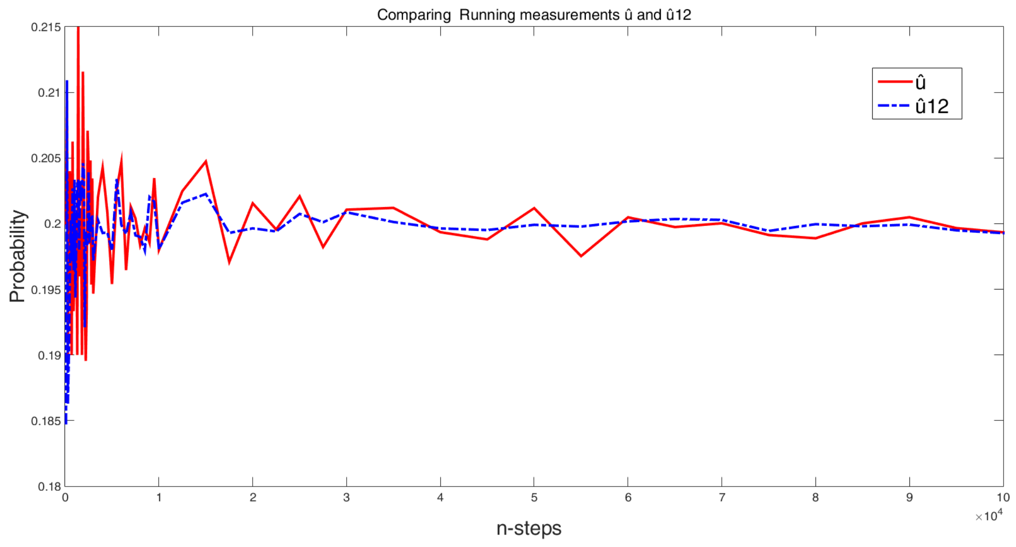


Figure 5.2: Normal traffic - Comparing parameter values during a sample run.

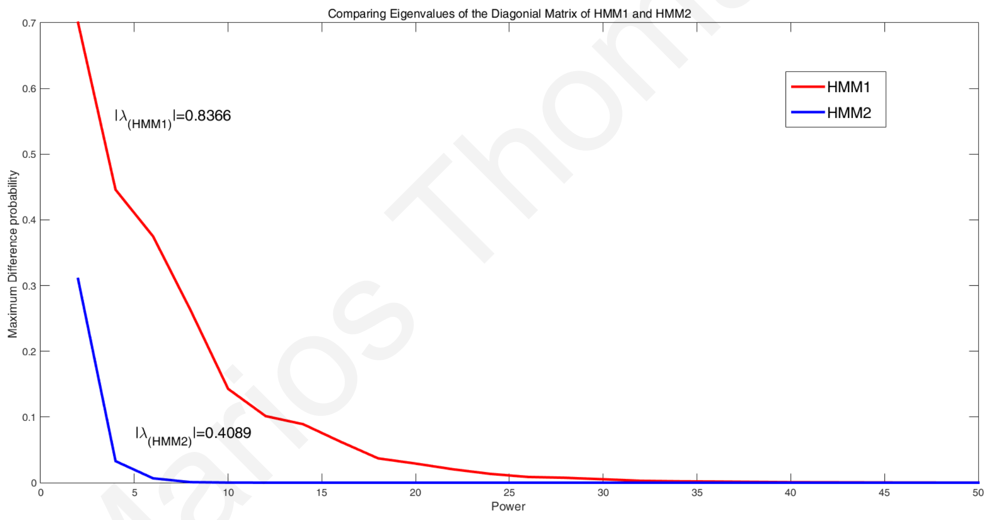


Figure 5.3: Normal traffic - Graph/eigenvalues $\lambda_{2,HMM1}$ and $\lambda_{2,HMM2}$.

HMM1 and acts as zombie in a DDoS attack (The gap between $\hat{u}_{12}$ and $\hat{u}$ is presented in Fig. 5.5. Due to the fact that $|\lambda_{2,HMM2}| < ||lambda_{2,HMM1}|$, HMM2 reaches steady state quicker as can be seen in Fig. 5.6 (it is not affected from the fact that we deal with abnormal behaviour).

As described above, the output sequence of HMM2 is the same with the output sequence of HMM1 (Fig. 5.4), i.e.,

$$y_{HMM1}[k] = y_{HMM2}[k].$$

As a result of the above, in abnormal traffic (refer to Fig. 5.5), we can identify

ABNORMAL TRAFFIC

n-steps	$\hat{u}_1$	$\hat{u}_2$	$\hat{u}$	$\hat{u}_{12}$
100	0.27	0.27	0.27	0.0729
200	0.29	0.29	0.29	0.0841
300	0.293333	0.293333	0.293333	0.086044
400	0.28	0.28	0.28	0.0784
500	0.298	0.298	0.298	0.088804
600	0.295	0.295	0.295	0.087025
700	0.291429	0.291429	0.291429	0.084931
800	0.2975	0.2975	0.2975	0.088506
900	0.29	0.29	0.29	0.0841
1000	0.289	0.289	0.289	0.083521
1100	0.285455	0.285455	0.285455	0.081484
1200	0.291667	0.291667	0.291667	0.085069
1300	0.286923	0.286923	0.286923	0.082325
1400	0.295714	0.295714	0.295714	0.087447
1500	0.294667	0.294667	0.294667	0.086828
1600	0.289375	0.289375	0.289375	0.083738
1700	0.282353	0.282353	0.282353	0.079723
1800	0.294444	0.294444	0.294444	0.086698
1900	0.286316	0.286316	0.286316	0.081977
2000	0.29	0.29	0.29	0.0841
2500	0.2936	0.2936	0.2936	0.086201
3000	0.289667	0.289667	0.289667	0.083907
3500	0.293143	0.293143	0.293143	0.085933
4000	0.2895	0.2895	0.2895	0.08381
4500	0.293333	0.293333	0.293333	0.086044
5000	0.2934	0.2934	0.2934	0.086084
5500	0.290727	0.290727	0.290727	0.084522
6000	0.294667	0.294667	0.294667	0.086828
6500	0.293231	0.293231	0.293231	0.085984
7000	0.293286	0.293286	0.293286	0.086017
7500	0.291733	0.291733	0.291733	0.085108
8000	0.291	0.291	0.291	0.084681
8500	0.293529	0.293529	0.293529	0.08616
9000	0.291222	0.291222	0.291222	0.08481
9500	0.290737	0.290737	0.290737	0.084528
10000	0.2904	0.2904	0.2904	0.084332
20000	0.29015	0.29015	0.29015	0.084187
30000	0.2919	0.2919	0.2919	0.085206
40000	0.2921	0.2921	0.2921	0.085322
50000	0.29202	0.29202	0.29202	0.085276
60000	0.291533	0.291533	0.291533	0.084992
70000	0.292	0.292	0.292	0.085264
80000	0.291375	0.291375	0.291375	0.084899
90000	0.2914	0.2914	0.2914	0.084914
100000	0.29134	0.29134	0.29134	0.084879

Figure 5.4: Abnormal traffic - Values of various parameters of interest after the execution of a run of 100000 steps in the two HMMs.

that refer to $\hat{u} > \hat{u}_{12}$. This is completely reasonable because $\hat{u}$ is the multiplication of the empirical values $\hat{u}_1$ and $\hat{u}_2$.

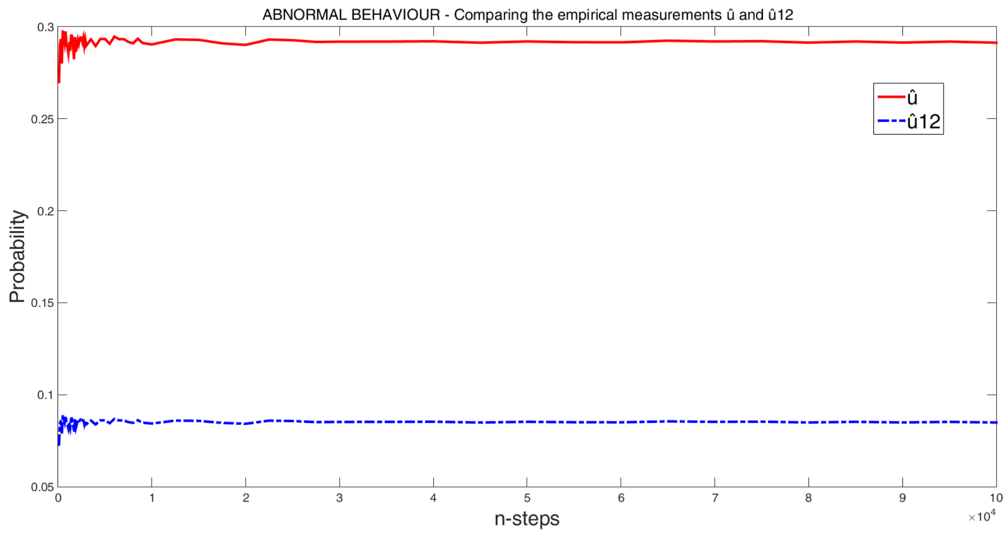


Figure 5.5: Abnormal traffic - Comparing parameter values during a sample run.

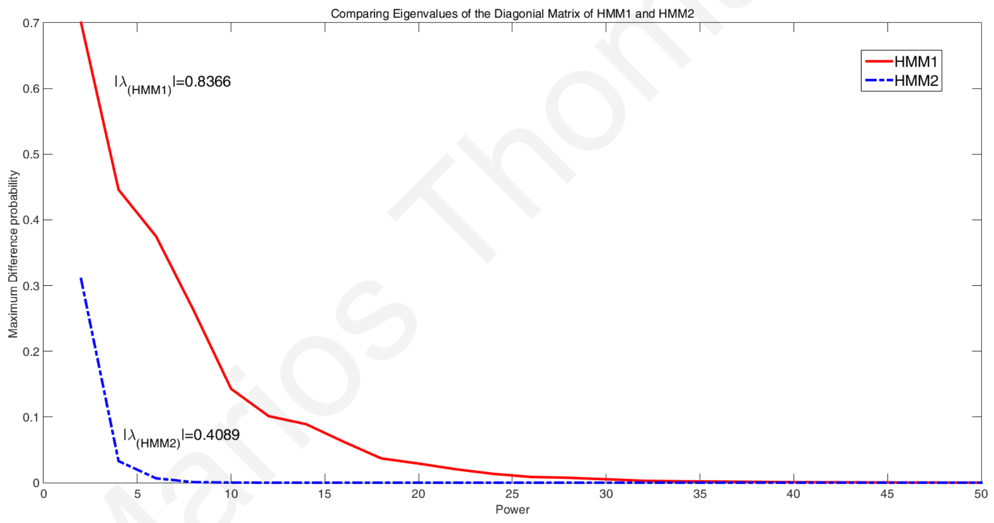


Figure 5.6: Abnormal Traffic - Graph/eigenvalues $\lambda_{2,HMM1}$ and $\lambda_{2,HMM2}$.

For $\kappa = 50$, HMM1 reaches the steady-state since

$$P^{50}_{HMM1} = \begin{pmatrix} 0.291 & 0.291 & 0.291 \\ 0.291 & 0.291 & 0.291 \\ 0.416 & 0.416 & 0.416 \end{pmatrix}.$$

5.1.3 Comparing Similar Models

In this subsection we compare HMM1 with HMM1plus and HMM2 with HMM2plus. In HMM1plus and HMM2plus the measurements are taken at different (shifted) times (HMM1plus is the same model as HMM1 and HMM2plus is the same model as HMM2). The result is shown in Fig. 5.7.

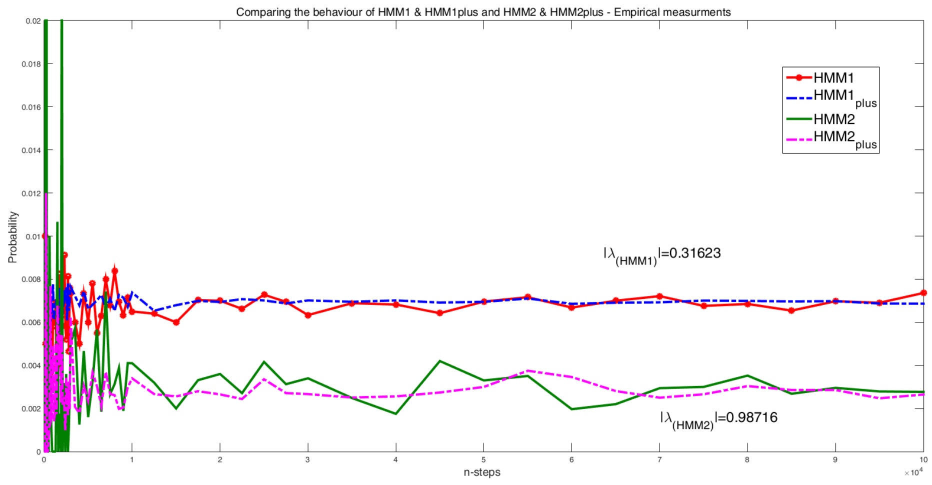


Figure 5.7: HMM1 vs HMM1plus and HMM2 vs HMM2plus

5.1.4 Different Model

In this subsection, we use additional models, HMM3 (Fig. 5.8) and HMM4 (Fig. 5.9), to represent normal behaviour, as shown below

$$P_{HMM1_1} = \begin{pmatrix} 0 & 0 & 0.1 \\ 1 & 0 & 0 \\ 0 & 1 & 0.9 \end{pmatrix},$$

$$P_{HMM2_1} = \begin{pmatrix} 0.1 & 0.048 & 0 & 0.09 \\ 0.9 & 0.88 & 0.025 & 0.202 \\ 0 & 0 & 0.072 & 0.9 \\ 0.03 & 0 & 0.075 & 0.88 \end{pmatrix},$$

$$P_{HMM3} = \begin{pmatrix} 0.3 & 0.7 \\ 0.7 & 0.3 \end{pmatrix},$$

$$P_{HMM4} = \begin{pmatrix} 0 & 0.3 & 0.7 \\ 1 & 0 & 0 \\ 0 & 0.7 & 0.3 \end{pmatrix}.$$

The goal of the above is to indicate that the outcome remains the same (Fig. 5.10).

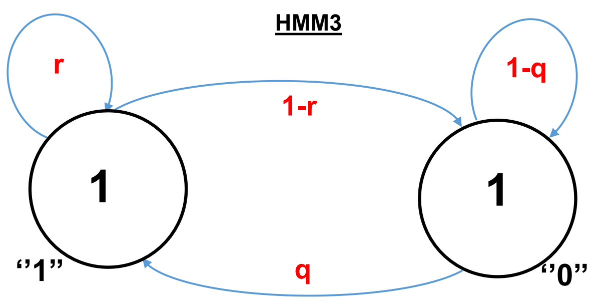


Figure 5.8: Two-state hidden Markov model.

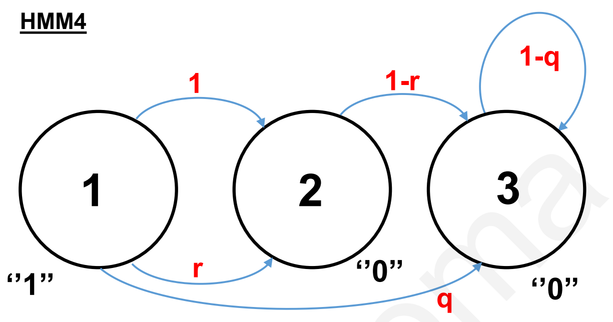


Figure 5.9: Three-state hidden Markov model.

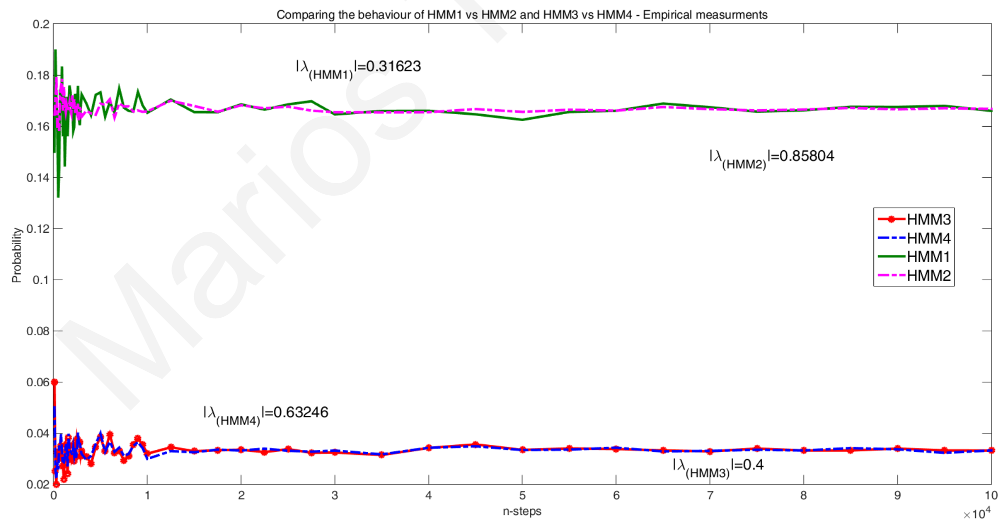


Figure 5.10: HMM1 vs HMM2 and HMM3 vs HMM4.

5.2 Enhanced Simulation Study

5.2.1 General Description of Marked Users

For the verification of the proposed methodology, we built networks of 8, 10, 12, 24, 36, and 48 of marked users[1].

[1]Marked users (known parameters): We namely indicate the users as dependent, flash crowd, no activity, independent and abnormal. This helps us to identify whether our method works properly.

Note that the users were constructed via simulation in order to fulfil the different conditions of Dependent Users, Independent Users, Flash Crowd users, Abnormal Users and No Activity Users. More specifically, their requests at different points in time (sequence of 0s and 1s) were chosen according to the type of user. A sample implementation of the above approach (built in Matlab using the random function where needed) is shown in Fig. 5.14 for the case of 12 users. Furthermore for all the networks of users, we used the basic parameters[2] identified for the network of 8 users (refer to Section 5.2.2).

For the networks of 8, 10 and 12 users, the data was generated through simulation software, using the indicated parameters. For the network of 12 users, the memo in Fig. 5.14 displays all the parameters. For the networks of 24, 36 and 48 users (additional to the above), due to the increased number of users, specifically for the independent users, we used built probability functions with known parameters.

The reasons for building the networks of 8, 10, 12, 24, 36 and 48 users with known parameters and simulating them, were (1) to identify that the method works properly (taking account that all the users are marked), (2) to assess how complexity increases when the number of users is increased (graphically).

Figure 5.14 is a small snapshot of total frame 4.10^3 time steps. We constract a network of 12 users. Initially, (1) the five of them were Independent ($R3, R4, R5, R6, R11$), (2) the two of them Dependent ($R1, R2$), (3) the other two of them Flash Crowd ($R7, R8$), (4) the other two of them Abnormal ($R9, R10$) and the rest one of them No Activity ($R12$). The outcome is described next.

5.2.2 Simulations with 8 users

For 8 users, we can see in Fig. 5.11, the outcome of the proposed approach for users 1 to 8, according to criteria 1 to 5 of Fig. 4.3. Each node represents a user (from 1 to 8), and gets a value which shows how many times it was member of the respective clique (independent, dependent, abnormal), based on the number of other users with whom it has high correlation. This additional criterion (not shown in the pseudocode of Fig. 4.3) gives us an extra way to identify suspicious users. Note that the threshold τ used to identify the closeness of $F_{comparable}$ and $F_{relative}$ is

[2]The basic parameters include the steps described in a unit time of 100ms, the mean packet rate is taken to be equal to 0.6 and $\tau = 10^{-4}$.

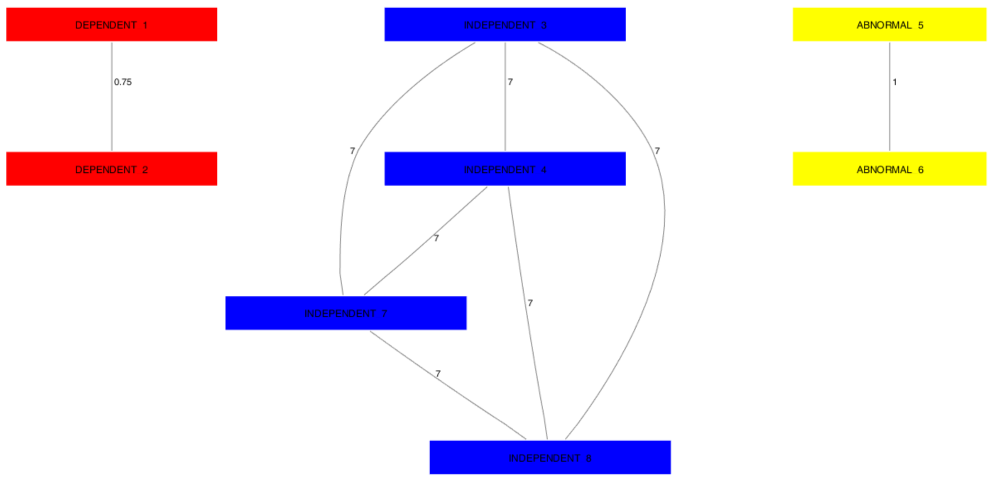


Figure 5.11: Classification of 8 users according to their behaviour.

chosen to be $\tau = 10^{-4}$. This value can be changed if needed (it was chosen based on empirical trials). With different coloring of the nodes, we can see in Fig. 5.11 the three different networks which represent the independent (blue), dependent (red nodes) and abnormal (yellow nodes) cliques. Specifically, users 3, 4, 7, 8 are independent, users 1, 2 are dependent, and users 5, 6 are abnormal. The edges of the graph of Fig. 5.11, are weighted and indicate the strength of the connection between the above users (independent, dependent and abnormal cliques). The steps described are in a unit time of 100ms and the mean packet rate is equal to 0.6.

5.2.3 Simulations with 10 users

For 10 users, we can see in Fig. 5.12, the outcome of the proposed approach for users 1 to 10.

Also in Figure 5.12 with different coloring of the nodes, we can see the three different networks which represent the independent (blue), dependent (red nodes) and abnormal cliques (yellow nodes). Specifically, user 3 is independent, users 1, 2, 4, 5, 6, 7, 8 are dependent, and users 9, 10 are abnormal. The edges of the graph of Fig. 5.12, are weighted and indicate the strength of the connection between the above users (independent, dependent and abnormal cliques).

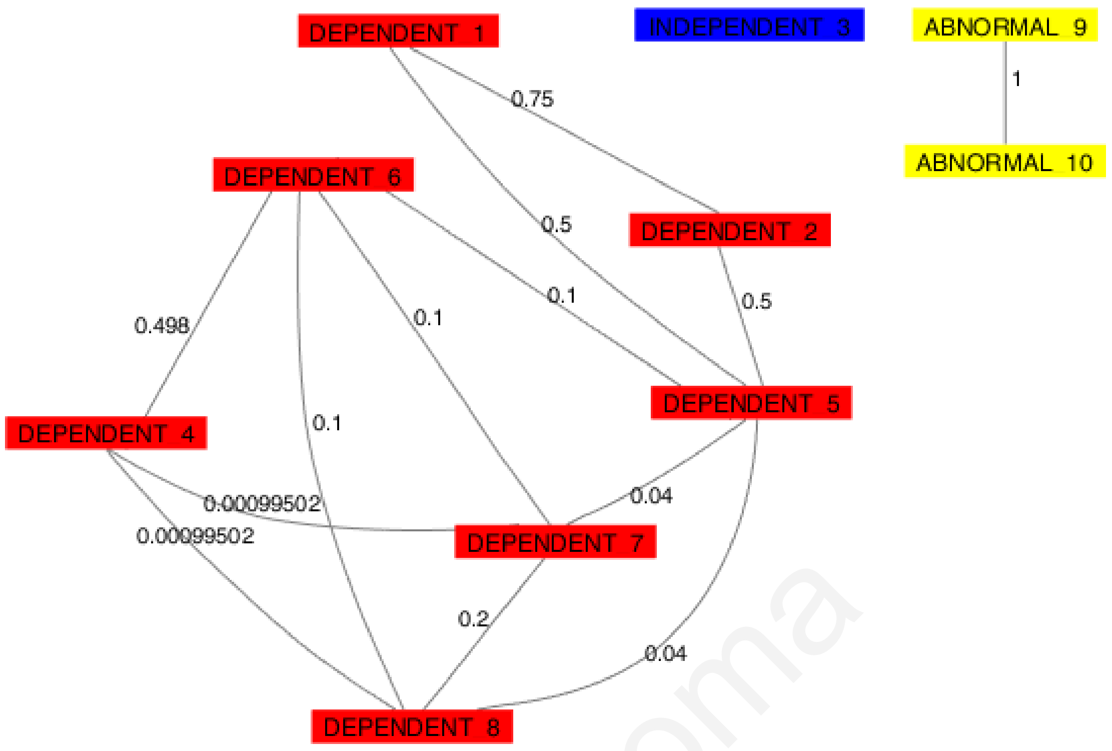


Figure 5.12: Classification of 10 users according to their behaviour.

5.2.4 Simulations with 12 users

For 12 users, we can see in Fig. 5.13, the outcome of the proposed approach for users 1 to 12.

Also in Fig. 5.13 with different coloring of the nodes, we can see the three different networks which represent the independent (blue, purple nodes), dependent (red nodes) and abnormal cliques (yellow nodes). Specifically, users 3, 12 are independent, users 1, 2, 4, 5, 6, 7, 8, 11 are dependent, and users 9, 10 are abnormal. The edges of the graph of Fig. 5.13, are weighted and indicate the strength of the connection between the above users (independent, dependent and abnormal cliques). Specifically, for dependent users, we can see the three sub-cliques that arise. The first sub-clique (red edges) involves users (1, 2, 5), the second sub-clique (blue edges) involves users (4, 6, 7, 8, 11) and the third sub-clique (gray edges) involves users (5, 6, 7, 8, 11). During the various phases of the implementation, the users are split into sub-cliques and then unified as one clique, for the dependent users. As it turns out, we do not have flash crowd users (specifically, users 7 and 8 are deemed dependent by the program). The steps described are in a unit time of 100ms and the mean

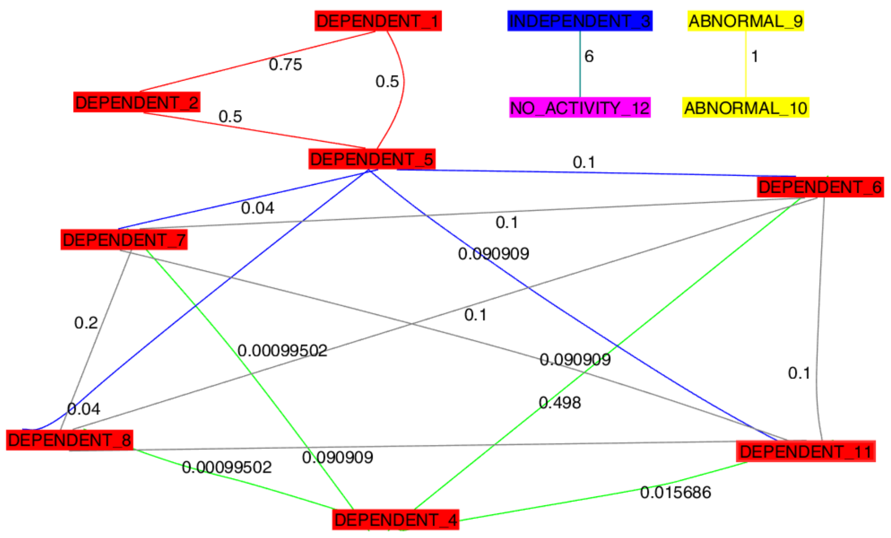


Figure 5.13: Classification of 12 users according to their behaviour (related to Fig. 5.14).

packet rate is equal to 0.6. The requests shown in Fig. 5.14, is a small snapshot of the total time frame of 4×10^3 time steps (unit time 100ms per step).

5.2.5 Simulations with 24 users

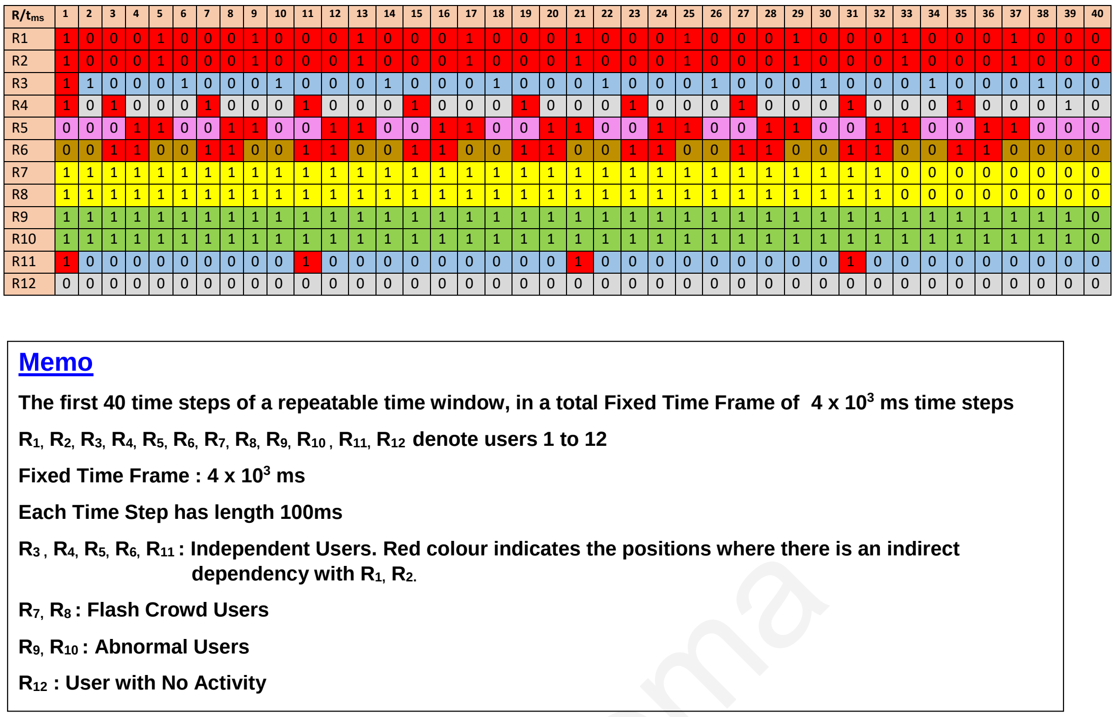
For 24 users, we can see in Fig. 5.15, the outcome of the proposed approach for users 1 to 24.

Also in Fig. 5.15 with different coloring of the nodes, we can see the three different networks which represent the independent (blue, purple nodes), dependent (red nodes), flash crowd (green nodes) and abnormal (yellow nodes) users. Specifically, users 1, 3, 4, 5 are independent and 12, 17, 20, 21, 22, 23, 24 no activity/independent. Users 6, 8, 9, 10, 11, 13, 14, 15, 16, 18, 19 are dependent/flash crowd. Users 2, 7 are abnormal. The edges of the graph in Fig. 5.15 are weighted and indicate the strength of the connection between the above users (independent, dependent and abnormal cliques).

5.2.6 Simulations with 36 users

For 36 users, we can see in Fig. 5.16, the outcome of the proposed approach for users 1 to 36.

Also in Fig. 5.16 with different coloring of the nodes, we can see the three different



R/t_{ms}	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40
R1	1	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0
R2	1	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0
R3	1	1	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0	1	0	0
R4	1	0	1	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0	1	0
R5	0	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0	0	0
R6	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0	0	0	0
R7	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0
R8	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0
R9	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0
R10	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0
R11	1	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
R12	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Memo

The first 40 time steps of a repeatable time window, in a total Fixed Time Frame of 4 x 10^3 ms time steps

R_1, R_2, R_3, R_4, R_5, R_6, R_7, R_8, R_9, R_{10}, R_{11}, R_{12} denote users 1 to 12

Fixed Time Frame : 4 x 10^3 ms

Each Time Step has length 100ms

R_3, R_4, R_5, R_6, R_{11} : Independent Users. Red colour indicates the positions where there is an indirect dependency with R_1, R_2.

R_7, R_8 : Flash Crowd Users

R_9, R_{10} : Abnormal Users

R_{12} : User with No Activity

Figure 5.14: Requests from users $R_1, R_2, \ldots, R_{12}$ (related to Fig. 5.13).

networks which represent the independent (blue, purple nodes), dependent (red nodes), flash crowd (green nodes) and abnormal (yellow nodes) users. Specifically, users 22, 23, 24, 28, 29, 30, 31, 32, 33, 34, 35, 36 are independent and 5, 11, 17 no activity/independent. Users 1, 2, 3, 4, 7, 8, 9, 10, 13, 14, 15, 16, 19, 27 are dependent and 20, 21, 25 dependent/flash crowd. Users 6, 12, 18, 26 are abnormal. The edges of the graph of Fig. 5.16, are weighted and indicate the strength of the connection between the above users (independent, dependent and abnormal cliques).

5.2.7 Simulations with 48 users

For 48 users, we can see in Fig. 5.17, the outcome of the proposed approach for users 1 to 48.

Also in Fig. 5.17 with different coloring of the nodes, we can see the three different networks which represent the independent (blue, purple nodes), dependent (red nodes), flash crowd (green nodes) and abnormal (yellow nodes) users. Specifically, users 5, 11, 17, 25, 29, 30,31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48 are independent and 5, 11, 17 no activity/independent. Users 1, 2, 3, 4, 7, 8, 9, 10, 13, 14, 15, 16, 19, 20, 21, 22, 23, 26, 27, 28 are dependent and 19, 20, 21, 23, 26, 27, 28 dependent/flash crowd. Users 6, 12, 18, 24 are abnormal. The edges of the graph of Fig. 5.17, are weighted and indicate the strength of the connection between the above users (independent, dependent and abnormal cliques).

5.3 Simulations with Real Botnet Data

For better evaluation of the method, further to the built data of the previous chapter, we use in this chapter real data made available as part of a research project at the CVUT University of Prague in the Czech Republic [54]. The file includes botnet traffic (the project only provided data related to botnet traffic). For our simulations, we used the file $botnet - capture - 20110810 - neris.pcap$. The data was transformed (encoded) and loaded in the simulation program. We ran our method in three scenarios. The first scenario, solely used the data of the above data file. For the second scenario, we used the above file with additional five, independent users that were computer generated for simulation purposes. For the third scenario we used the five additional independent users with verified random behaviour.

Note that in order to process the real data we amended the pseudocode of Chapter 4 (Fig. 4.3), to cover the possibility that some users may send multiple packets, within the same unit of time (time step, e.g. $100ms$, $1s$, etc.). If this quantity exceeds the mean packet rate of normal users, then we have a strong indication that the user in question may be flash crowd.

The total time window of the above pcap file, has length 4444.097 seconds (relative time between the first and the last packet for both incoming/outgoing traffic). The total number of users is 215. The 214 users interact with the user with IP address 147.32.84.165. Each IP address was encoded with a number in order of appearance. For example, the IP address 147.32.84.165, is represented by the number 4. The other users are given numbers 1 to 215.

5.3.1 Scenario 1: Verification of Completed Time Window

In Scenario 1, we have the traffic in Fig. 5.18, which consists of all the data of the file, $botnet - capture - 20110810 - neris.pcap$. Note that Fig. 5.18 represents both the incoming/outgoing traffic (we observe the incoming and outgoing traffic from a crossroad point when using unit time step 1s). Furthermore, in this figure, we can see graphically the *activity packet rate arrival* (the total number of packets which arrive per unit time - straight dark blue line in Fig. 5.18), the *mean packet rate arrival per user* (straight light blue line in Fig. 5.18) and the *mean packet rate arrival per time* (straight red line in Fig. 5.18). The activity packet rate arrival gives a graphical representation of the actual traffic (we can identify any unusual high traffic). The mean packet rate arrival per user gives the mean packet rate regarding arrival packets per user. The mean packet rate arrival per time indicates the mean packet arrival rate per unit of time. In summary, Fig. 5.18 is a statistical representation of the incoming/outgoing traffic.

Examined time step equal to 1s

In Fig. 5.19, we can see the graphical representation of the resulting classification of users, according to their behaviour, when using unit time step of 1s. From the total number of 215 users, we have found 82 independent users, 0 abnormal users, 1 flash crowd user, 132 dependent users, 0 no activity users (13 depedent/flash crowd users).

Examined time step equal to 100ms

In Fig. 5.20, we can see the graphical representation of resulting classification of users, according to their behaviour when using unit time step 100ms. From the total number of 215 users, we have found 180 independent users, 0 abnormal users, 3 flash crowd users, 32 dependent users, 2 no activity/independent users (9 depedent/flash crowd users).

Comparing Fig. 5.19 and Fig. 5.20

As we can see, independent users in Fig. 5.19 total 82 and in Fig. 5.20 total 180. This is reasonable due to the fact that we rely on simultaneous activity between different users within the same unit time step; thus, if the unit time step has smaller value (i.e. 100ms), then it is more difficult to identify simultaneous activity between different users. For this reason, we amend the pseudocode of Fig. 4.3, in order to cover the possibility, for some users, to send additional packets (above the usual), within the same unit of time.

5.3.2 Scenario 2: Verification of Completed Time Window with Additional Independent Users

For Scenario 2, we consider the traffic in Fig. 5.21 (the statistical representation is the same as in Fig. 5.18). The only difference between Scenario 2 and Scenario 1 is in the inclusion of five additional independent users. Our purpose is to extract further results regarding the implementation of our method, by involving users with known behaviour.

Examined time step equal to $1s$

In Fig. 5.22, we can see the graphical representation of the resulting classification of users, according to their behaviour when using unit time step of $1s$. From the total number of 220 users (additional 5 independent users with fixed behaviour), we have found as independent users 85, abnormal users 0, flash crowd users 3, dependent users 132, no activity users 0, and depedent/flash crowd users 13 (2 no activity/independent users, and 13 depedent/flash crowd users). Note that for the five additional independent users, the result after the execution of our method showed that three of them are independent and two of them are flash crowd (activity above the mean packet rate).

Examined time step equal to $100ms$

In Fig. 5.23, we can see the graphical representation of the resulting classification of users, according to their behaviour when using unit time step of $100ms$. From the total number of 220 users (additional 5 independent users with fixed behaviour), we have found as independent users 191, abnormal users 0, flash crowd users 1, dependent users 28, no activity users 0 (4 no activity/independent users, and 4 dependent/flash crowd users). Note that, for the five additional independent users, our method identified, three of them as independent and two of them as flash crowd (activity above the mean packet rate).

Compairing Fig. 5.22 and Fig. 5.23

As we can see independent users in Fig. 5.22 total 85 and in Fig. 5.23 they total 191. Again, this is reasonable as explained for Scenario 1.

5.3.3 Scenario 3: Verification of Completed Time Window with Additional Independent Users - Random Behaviour

For Scenario 3, we consider the traffic in Fig. 5.24 (the statistical representation is the same as in Fig. 5.18). The only difference between Scenario 3 and Scenario 2 is that the five additional independent users act at random and their exact behaviour is not assumed known by the classification methodology. Their maximum traffic is chosen to be equal to the maximum historical data of the independent users.

Examined time step equal to $1s$

In Fig. 5.25, we can see the graphical representation of the resulting classification of users, according to their behaviour when using unit time step of $1s$. From the total number of 220 users (additional 5 independent users with random behaviour), we have found 92 independent users, 1 abnormal users , 0 flash crowd users, 127 dependent users, 0 no activity users (2 no activity/independent users, and 8 dependent/flash crowd users). The five additional users are indicated as independent.

Examined time step equal to $100ms$

In Fig. 5.26, we can see the graphical representation of the resulting classification of users, according to their behaviour when using unit time step of $100ms$. From the total number of 220 users (additional 5 independent users with random behaviour), we have found 191 independent users, 1 abnormal users, 0 flash crowd users, 28 dependent users, 0 no activity users (4 no activity/independent users, and 5 dependent/flash crowd users). The five additional users are indicated as independent.

As we can see independent users in Fig. 5.25 total 92 and in Fig. 5.26 they total 191. Again, this is reasonable as explained for Scenario 1.

5.4 Summary - Main Remarks

In this chapter we implemented the enhanced approach described in Chapter 4. First we presented small examples with fixed botnet data, and later proceeded with a simulation study of our enhanced methodology, which includes more complicated examples of botnet data. Finally, we presented simulations with real botnet data. The full analysis is as follows.

5.4.1 Main Remarks of the 1st stage - Small Examples with Fixed Botnet Data

The main remarks of this stage are the following.

- After n steps (approximately $n = 1000$), the comparable frequency is equal to the relative frequency under normal behaviour.
- In abnormal behaviour, when the users are mimicking each other there is a gap between the comparable and the relative frequency.
- The rate of convergence is governed by the values of the respective eigenvalues. This is the reason that the minimum steps required in order to have clear result is approximately $n = 1000$ steps.
- The important value is the second largest magnitude of the eigenvalues of the respective HMM. The smaller it is, the sooner the HMM will reach the steady steady.

5.4.2 Main Remarks of the 2nd stage - Enhanced Simulation Study

The main remarks of this stage are the following:

- After a total of 4000 steps, we have clear results.
- During the implementation of the method, we found some internal cliques regarding the dependent and flash crowd users. The results are quoted in Figures 5.27, 5.28 and 5.29 (different coloring with corresponding explanation and comments are included within the figures).
- Despite the fact that some users seem to behave normally, it is shown from Fig. 5.27 that there is an indirect relation among them, which leads us to the conclusion that perhaps the method can be modified to handle time varying probabilities (shifted time).
- When the number of users or steps is increased, the elapsed run time is increased accordingly. The justification of complexity is shown in Fig. 5.30 (different coloring with corresponding explanation and comments are included within the figure).

5.4.3 Main Remarks of the 3rd stage - Simulations with Real Botnet Data

This stage represents the main part of our work and illustrates the novel approach for malware analysis. At this stage, we use our method with real botnet data. The processing of the data, and how they get transformed from the pcap file format, in order to be used is shown in Figs. 5.31 [3] and 5.32 [4] (different coloring with corresponding explanation and comments are included within the figures).

- When the value of the unit of time is decreased (e.g. from 1*s* to 100*ms*), the accuracy increases, and makes it more difficult to identify any simultaneous requests among the users in question (for both incoming/outgoing traffic). One can, of course, consider ways to count requests that occur approximately simultaneously, but this is something that we did not pursue explicitly.
- Users with very high incoming/outgoing traffic, which are identified as Flash Crowds only, may be potentially the victims or the attacker.
- When there is an increase in the users or/and the total number of time steps or/and the samples of unit of time (e.g., when going from 1*s* to 100*ms*) the computational complexity increases, due to the increased number of comparisons that are needed.
- Despite the fact that the file includes only botnet traffic, we identified users with normal behaviour. The reason for this is that due to the limited traffic of these users, they do not fulfill the condition $F_{comparable} > F_{relative}$, and their behaviour looks normal.
- As already mentioned, the real data consist only with botnet traffic (the project only provided data related to botnet traffic). In order to have a clear view (safe results) we added in the traffic five independent users (in each scenario). The results show that these users are identified correctly as independent.

[3]Fig. 5.31- Represents the real data loaded in WireShark.

[4]Fig. 5.32- Represents the way the real data transformed in order to be used according to the variable time step (1*s* or 100*ms*).

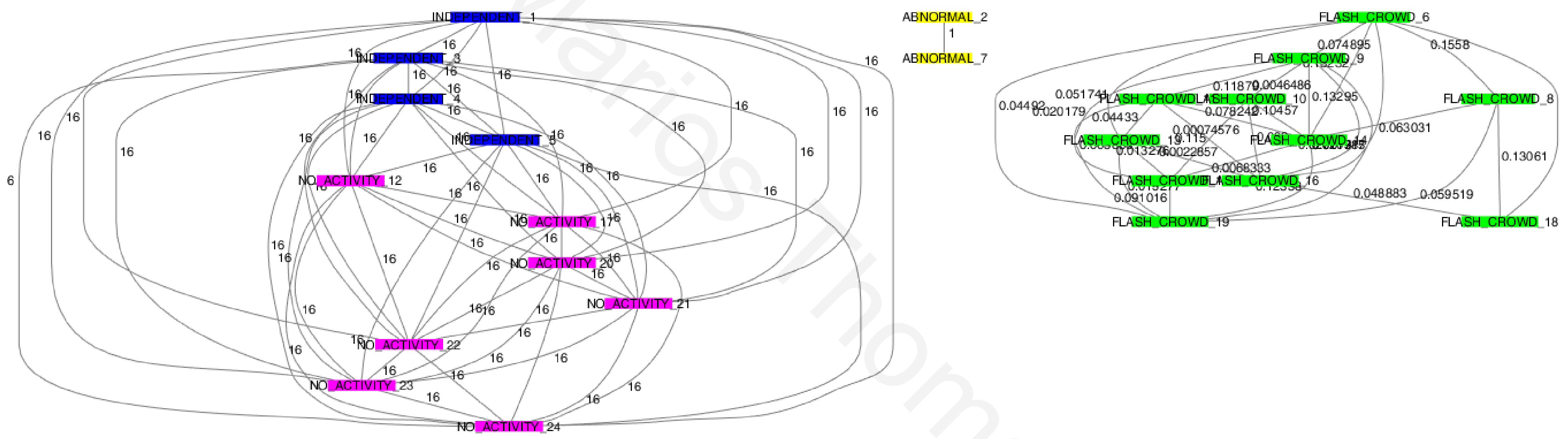


Figure 5.15: Classification of 24 users according to their behaviour.

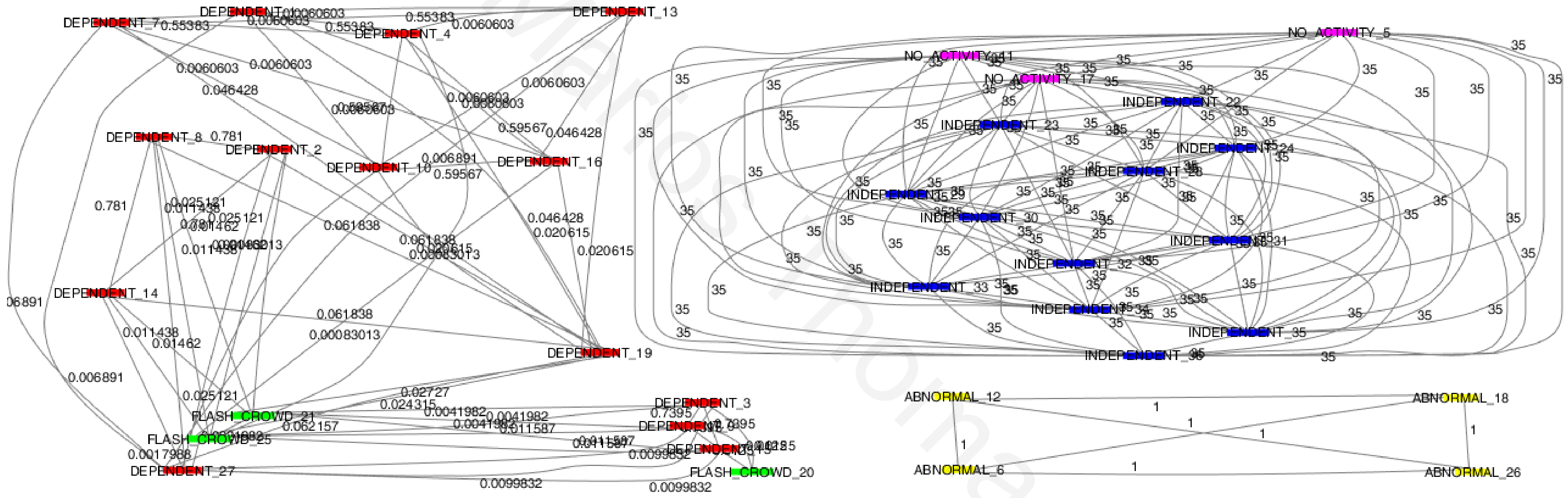


Figure 5.16: Classification of 36 users according to their behaviour.

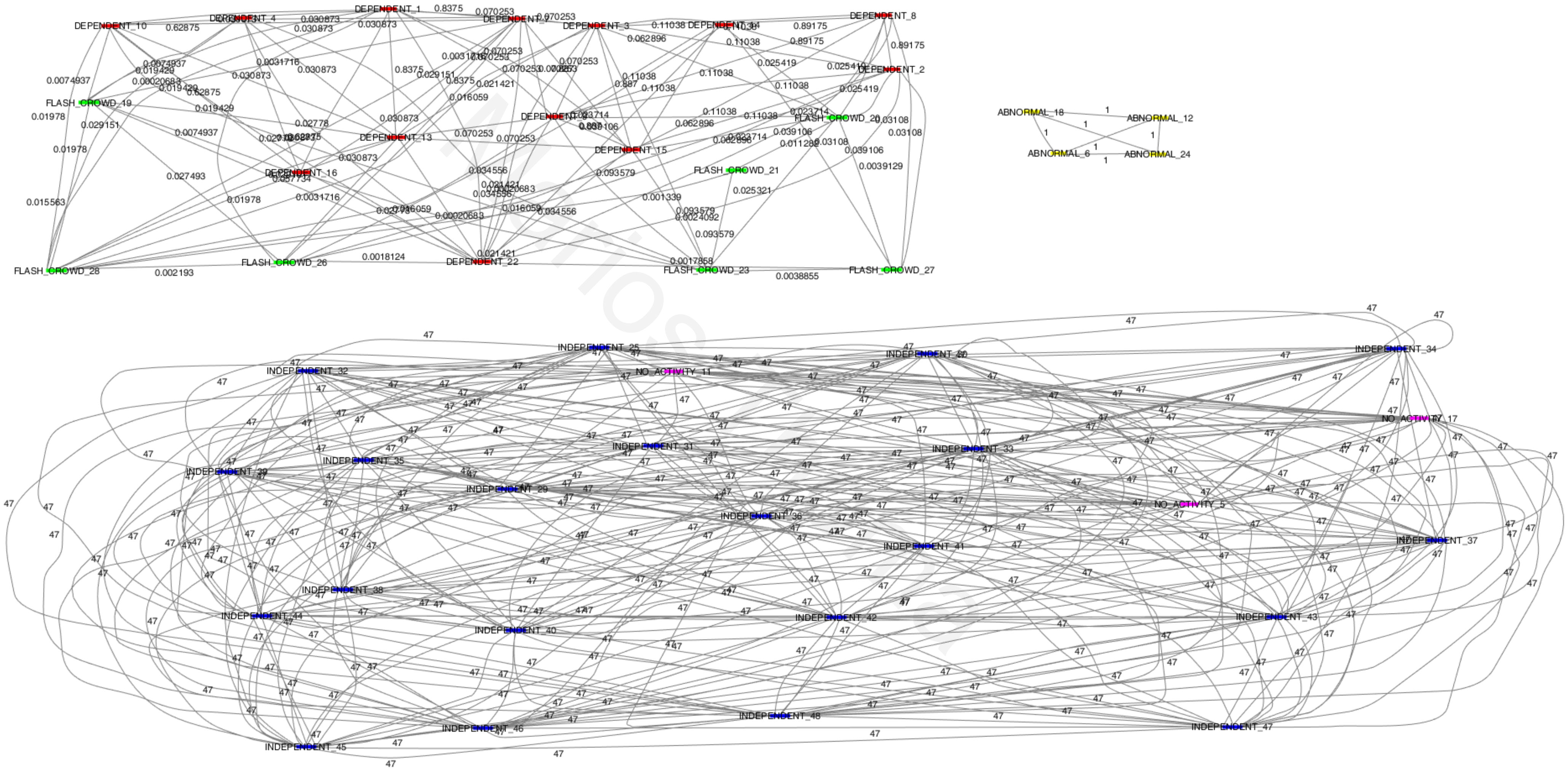


Figure 5.17: Classification of 48 users according to their behaviour.

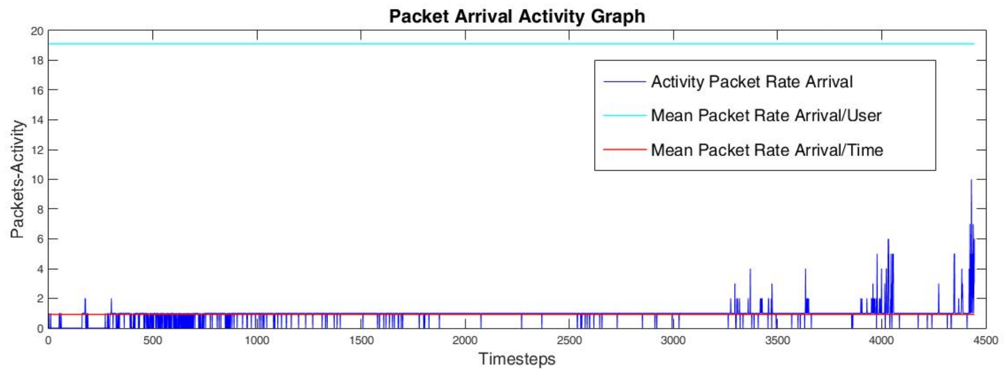


Figure 5.18: Scenario 1: Packet arrival activity graph - unit time step 1*s*.

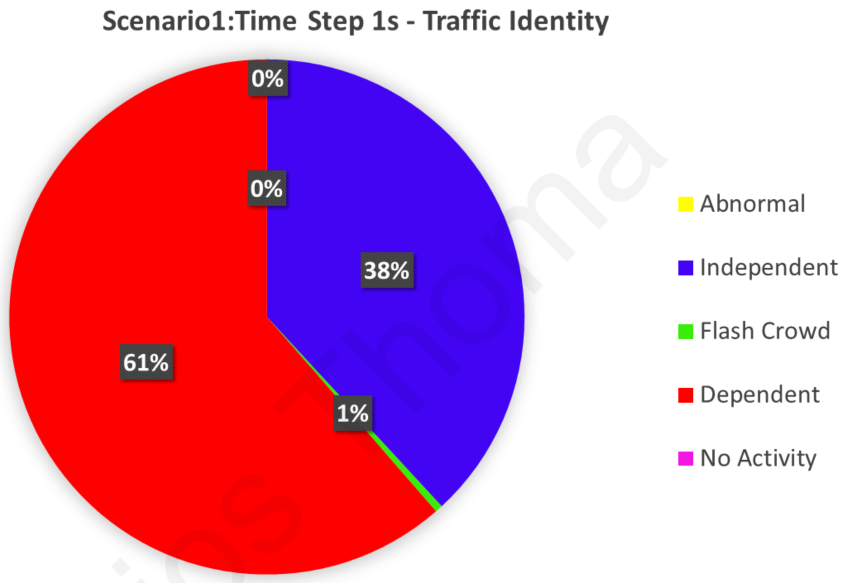


Figure 5.19: Scenario 1: Time Step 1s - Traffic identity.

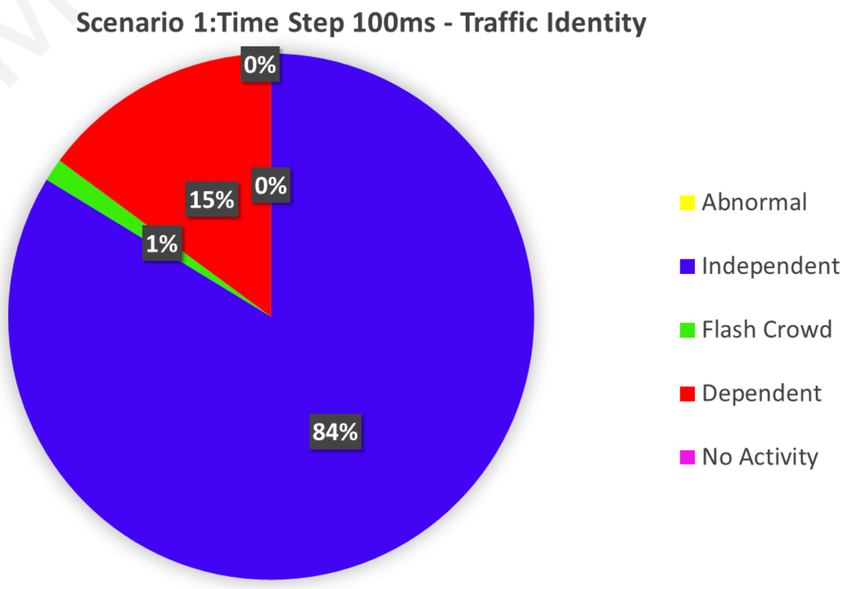


Figure 5.20: Scenario 1: Time Step 100ms - Traffic identity

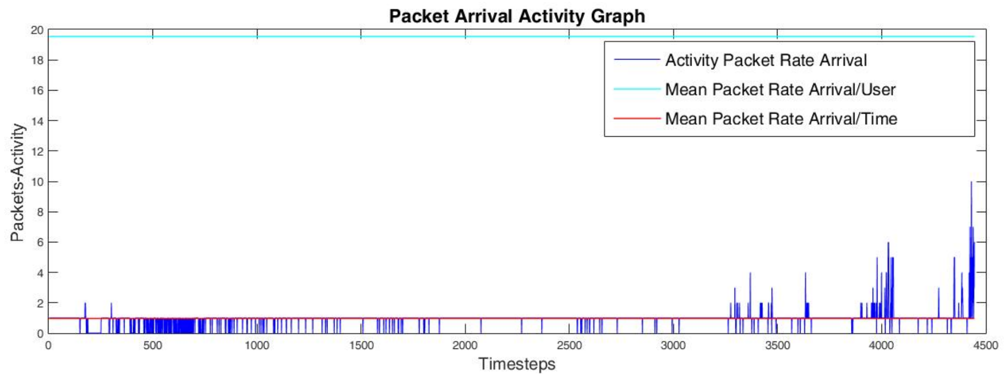


Figure 5.21: Scenario 2: Packet arrival activity graph - unit time step 1*s*.

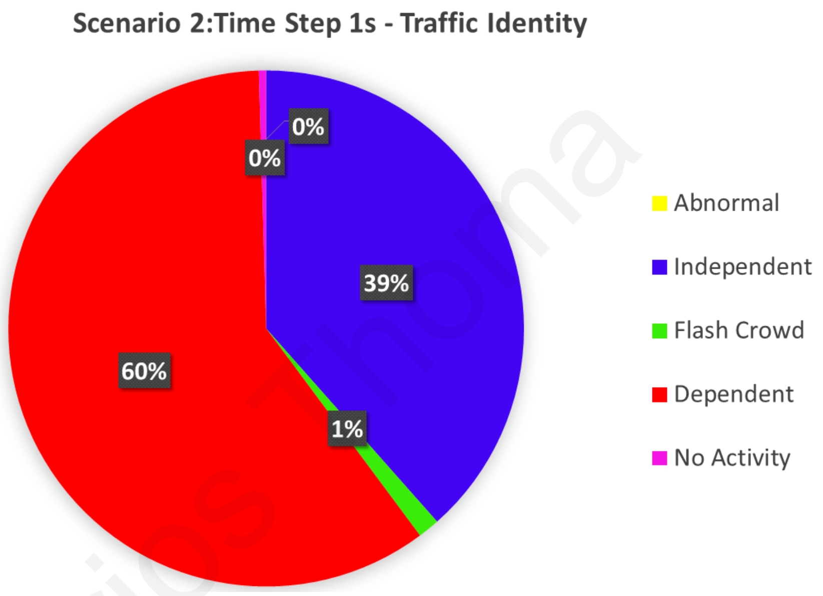


Figure 5.22: Scenario 2: Time Step 1s - Traffic identity.

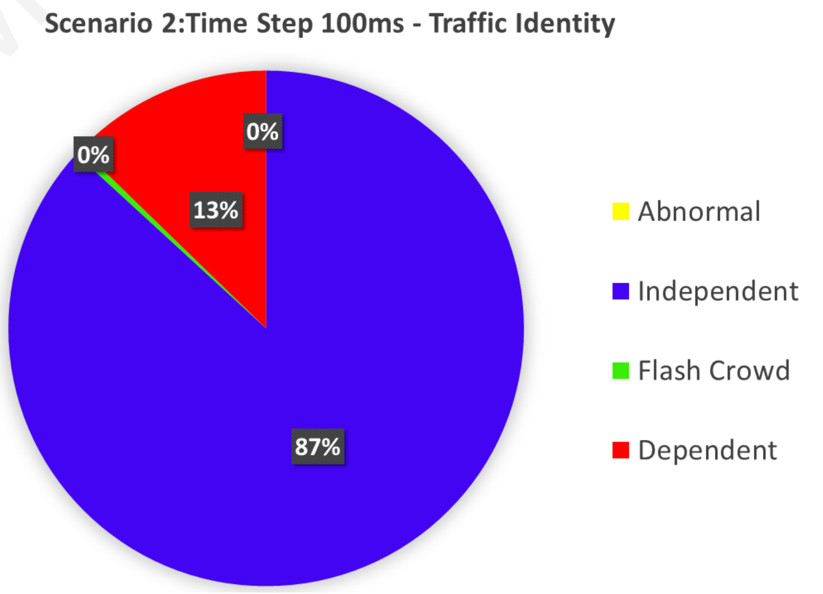


Figure 5.23: Scenario 2: Time Step 100ms - Traffic identity.

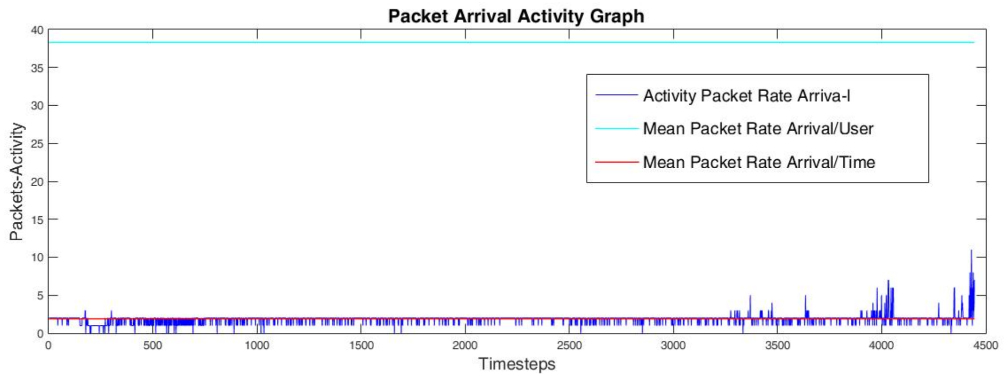


Figure 5.24: Scenario 3: Packet arrival activity graph - unit time step 1*s*.

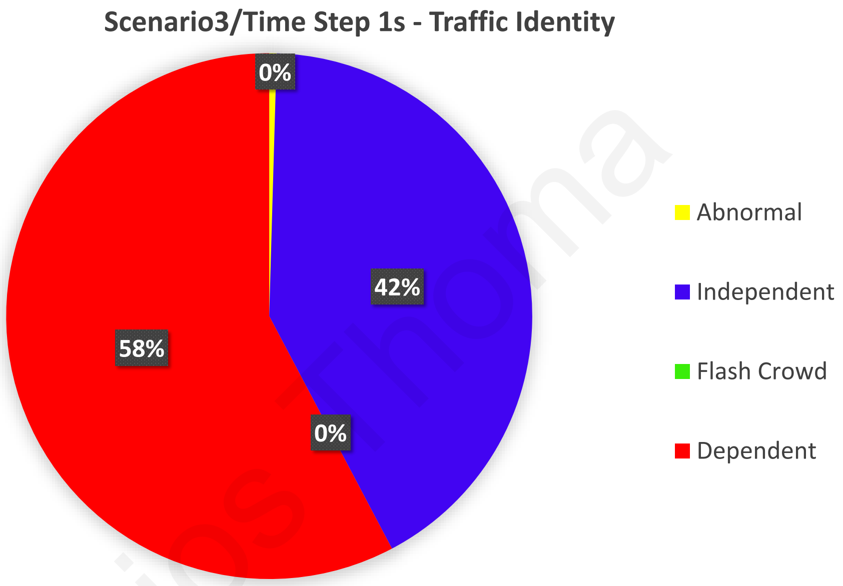


Figure 5.25: Scenario 3: Time Step 1s - Traffic identity.

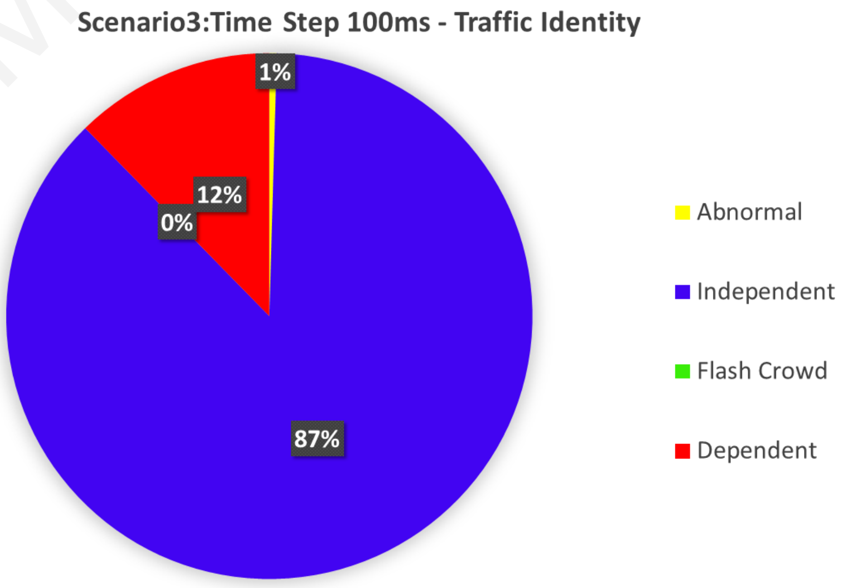
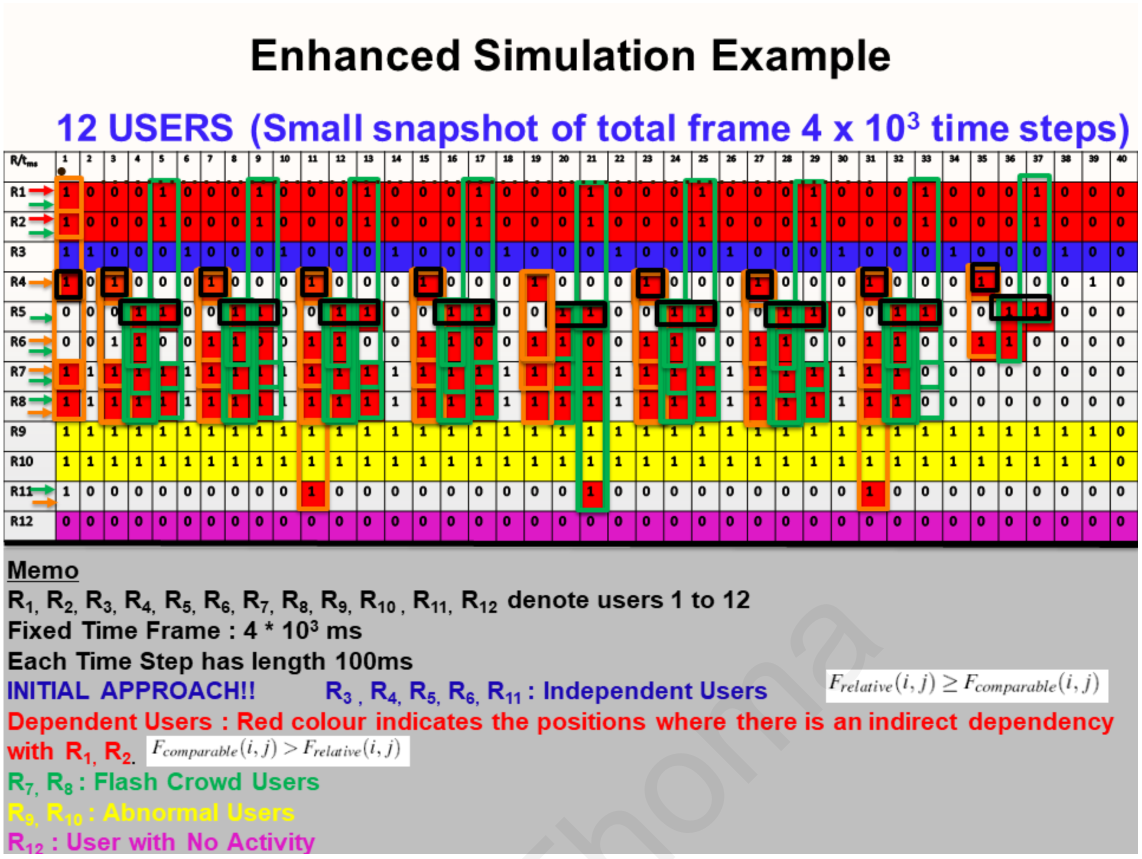


Figure 5.26: Scenario 3: Time Step 100ms - Traffic identity.



Enhanced Simulation Example

12 USERS (Small snapshot of total frame 4 x 10^3 time steps)

R/t_{ms}	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40
R1	1	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0
R2	1	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0
R3	1	1	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0	1	0	0
R4	1	0	1	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0	1	0
R5	0	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0	0	0
R6	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0	0	0	0
R7	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0
R8	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0
R9	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0
R10	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0
R11	1	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
R12	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Memo
$R_1, R_2, R_3, R_4, R_5, R_6, R_7, R_8, R_9, R_{10}, R_{11}, R_{12}$ denote users 1 to 12
Fixed Time Frame : 4 * 10^3 ms
Each Time Step has length 100ms
INITIAL APPROACH!! R_3 , R_4, R_5, R_6, R_{11} : Independent Users $F_{relative}(i,j) \geq F_{comparable}(i,j)$
Dependent Users : Red colour indicates the positions where there is an indirect dependency with R_1, R_2. $F_{comparable}(i,j) > F_{relative}(i,j)$
R_7, R_8 : Flash Crowd Users
R_9, R_{10} : Abnormal Users
R_{12} : User with No Activity

Figure 5.27: Example 1: Enhanced simulation example - Indirect Cliques.

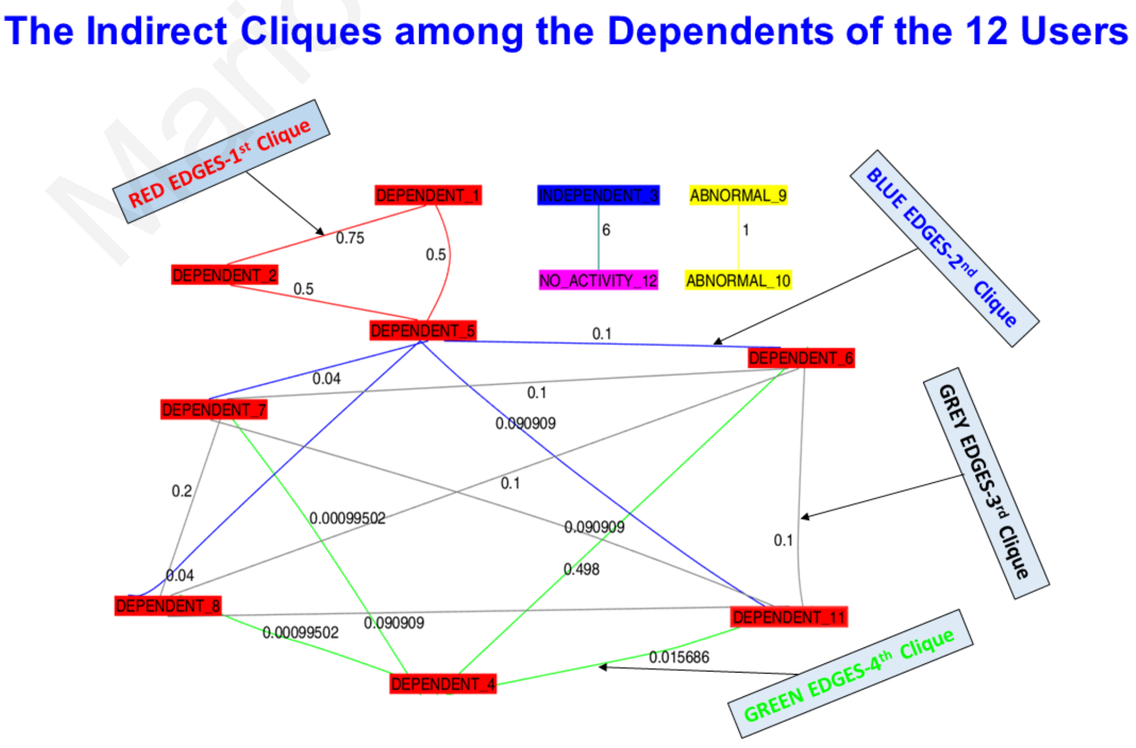


Figure 5.28: Example 1: Graph representation - Indirect Cliques.

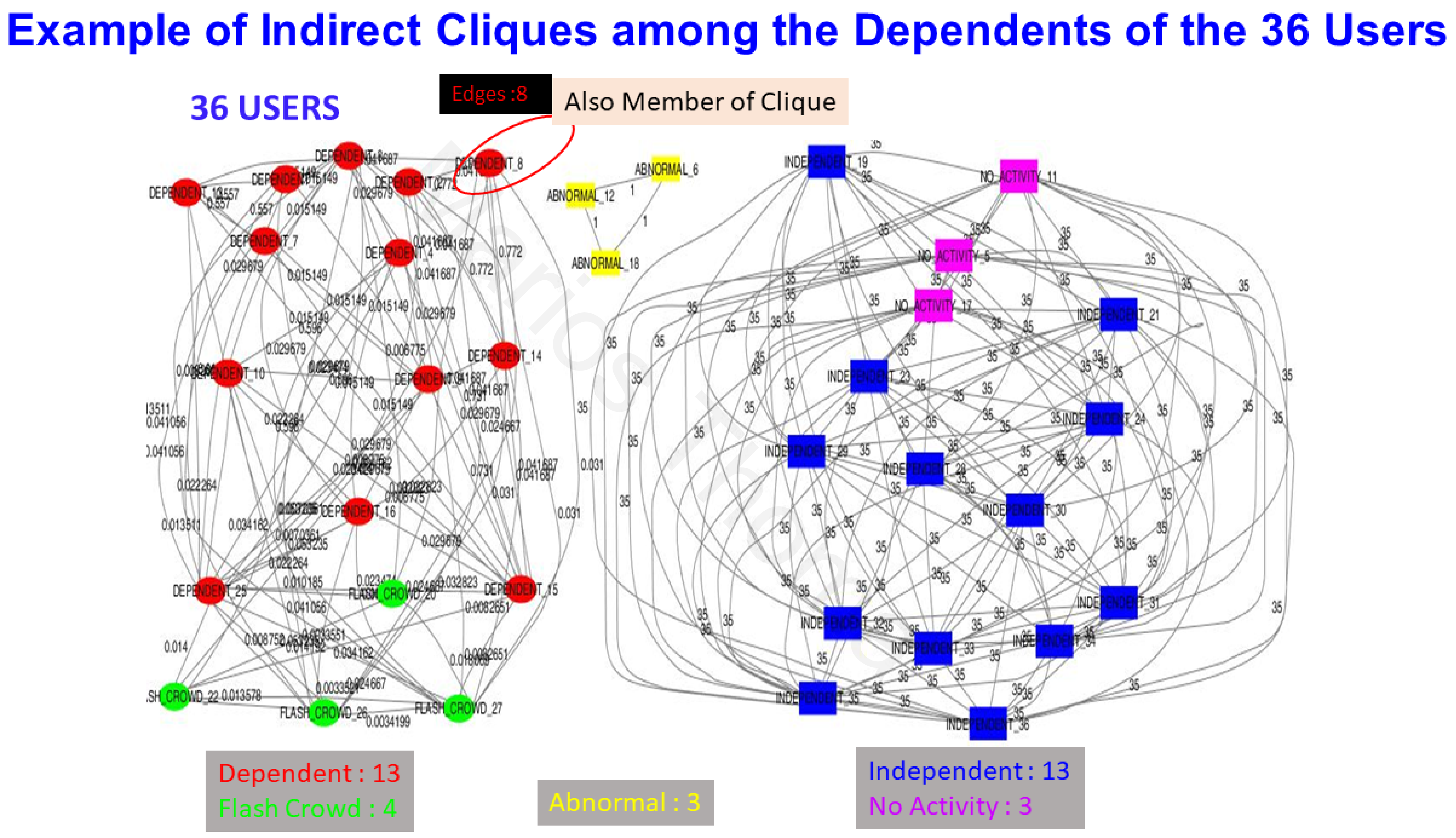
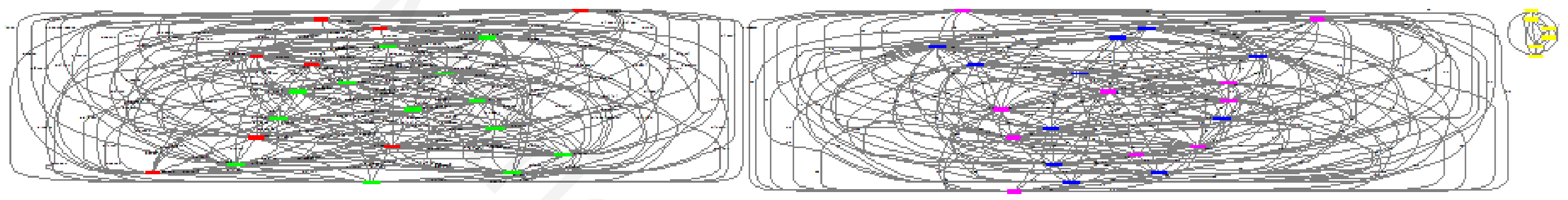


Figure 5.29: Example 2: Graph representation - Indirect Cliques.

Example of Complexity for 48 USERS in 4096 Steps



Dependent : 8
Flash Crowd : 13

Independent : 11
No Activity : 10

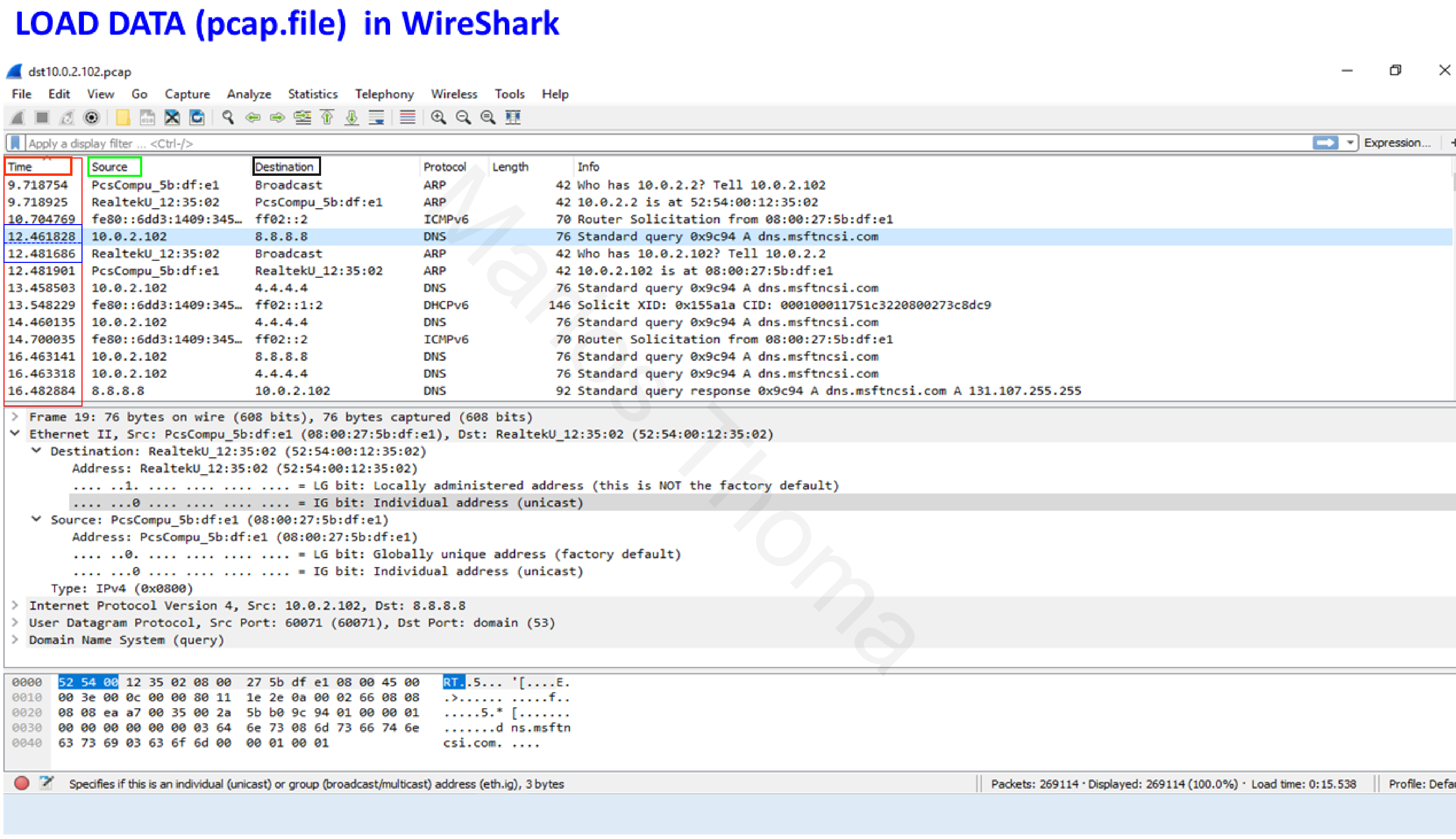
Abnormal : 6

Matrix of Number of Steps/Outcome

Number of Steps (100ms)	256	2048	4096	8192	16384	32768	65536
INDEPENDENT (#)	22	21	21	21	21	21	21
ABNORMAL (#)	5	5	6	6	5	6	6
FLASH CROWD (#)	13	14	13	13	14*(1)	13	13
DEPENDENT (#)	21	22	21	21	22	21	21
NO ACTIVITY (#)	10	10	10	10	10	10	10
Elapse run Time (s)	16.0167	16.1231	17.7814	19.7735	30.6639	61.3971	185.2791

1

Figure 5.30: Example 3: Justification of complexity for 48 Users.


LOAD DATA (pcap.file) in WireShark
dst10.0.2.102.pcap
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Apply a display filter ... <Ctrl-/>
Expression...

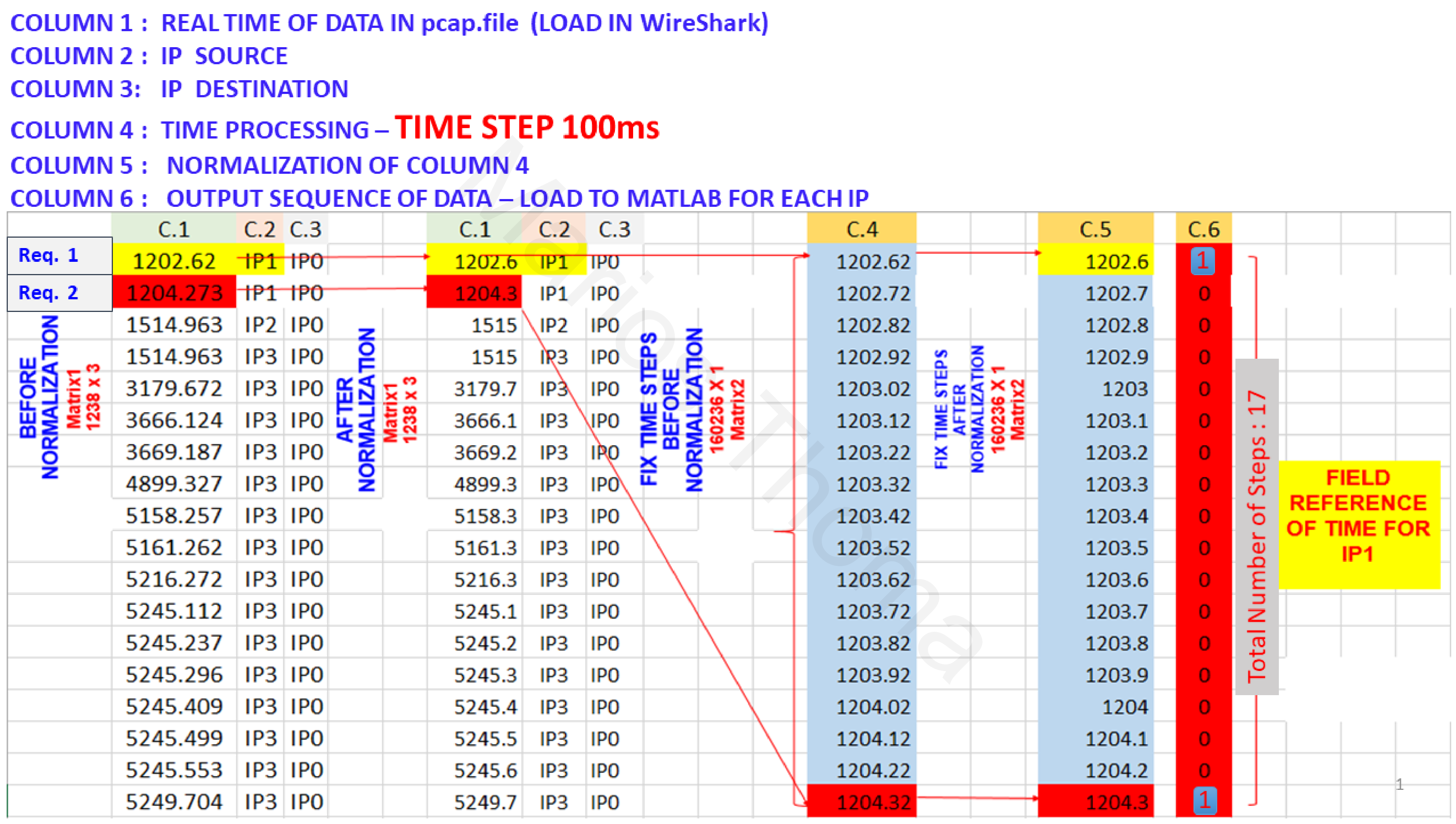
```
Time        Source                    Destination        Protocol  Length  Info
9.718754    PcsCompu_5b:df:e1         Broadcast          ARP        42 Who has 10.0.2.2? Tell 10.0.2.102
9.718925    RealtekU_12:35:02         PcsCompu_5b:df:e1  ARP        42 10.0.2.2 is at 52:54:00:12:35:02
10.704769   fe80::6dd3:1409:345…      ff02::2            ICMPv6     70 Router Solicitation from 08:00:27:5b:df:e1
12.461828   10.0.2.102                8.8.8.8            DNS        76 Standard query 0x9c94 A dns.msftncsi.com
12.481686   RealtekU_12:35:02         Broadcast          ARP        42 Who has 10.0.2.102? Tell 10.0.2.2
12.481901   PcsCompu_5b:df:e1         RealtekU_12:35:02  ARP        42 10.0.2.102 is at 08:00:27:5b:df:e1
13.458503   10.0.2.102                4.4.4.4            DNS        76 Standard query 0x9c94 A dns.msftncsi.com
13.548229   fe80::6dd3:1409:345…      ff02::1:2          DHCPv6    146 Solicit XID: 0x155a1a CID: 000100011751c3220800273c8dc9
14.460135   10.0.2.102                4.4.4.4            DNS        76 Standard query 0x9c94 A dns.msftncsi.com
14.700035   fe80::6dd3:1409:345…      ff02::2            ICMPv6     70 Router Solicitation from 08:00:27:5b:df:e1
16.463141   10.0.2.102                8.8.8.8            DNS        76 Standard query 0x9c94 A dns.msftncsi.com
16.463318   10.0.2.102                4.4.4.4            DNS        76 Standard query 0x9c94 A dns.msftncsi.com
16.482884   8.8.8.8                   10.0.2.102         DNS        92 Standard query response 0x9c94 A dns.msftncsi.com A 131.107.255.255
```

```
> Frame 19: 76 bytes on wire (608 bits), 76 bytes captured (608 bits)
v Ethernet II, Src: PcsCompu_5b:df:e1 (08:00:27:5b:df:e1), Dst: RealtekU_12:35:02 (52:54:00:12:35:02)
  v Destination: RealtekU_12:35:02 (52:54:00:12:35:02)
      Address: RealtekU_12:35:02 (52:54:00:12:35:02)
      .... ..1. .... .... .... .... = LG bit: Locally administered address (this is NOT the factory default)
      .... ...0 .... .... .... .... = IG bit: Individual address (unicast)
  v Source: PcsCompu_5b:df:e1 (08:00:27:5b:df:e1)
      Address: PcsCompu_5b:df:e1 (08:00:27:5b:df:e1)
      .... ..0. .... .... .... .... = LG bit: Globally unique address (factory default)
      .... ...0 .... .... .... .... = IG bit: Individual address (unicast)
    Type: IPv4 (0x0800)
> Internet Protocol Version 4, Src: 10.0.2.102, Dst: 8.8.8.8
> User Datagram Protocol, Src Port: 60071 (60071), Dst Port: domain (53)
> Domain Name System (query)
```

```
0000  52 54 00 12 35 02 08 00  27 5b df e1 08 00 45 00   RT..5... '[....E.
0010  00 3e 00 0c 00 00 80 11  1e 2e 0a 00 02 66 08 08   .>...... .....f..
0020  08 08 ea a7 00 35 00 2a  5b b0 9c 94 01 00 00 01   .....5.* [.......
0030  00 00 00 00 00 00 03 64  6e 73 08 6d 73 66 74 6e   .......d ns.msftn
0040  63 73 69 03 63 6f 6d 00  00 01 00 01               csi.com. ....
```

```
Specifies if this is an individual (unicast) or group (broadcast/multicast) address (eth.ig), 3 bytes    Packets: 269114 · Displayed: 269114 (100.0%) · Load time: 0:15.538    Profile: Defa
```

Figure 5.31: Loading of real date in Wireshark.



COLUMN 1 : REAL TIME OF DATA IN pcap.file (LOAD IN WireShark)
COLUMN 2 : IP SOURCE
COLUMN 3: IP DESTINATION
COLUMN 4 : TIME PROCESSING – TIME STEP 100ms
COLUMN 5 : NORMALIZATION OF COLUMN 4
COLUMN 6 : OUTPUT SEQUENCE OF DATA – LOAD TO MATLAB FOR EACH IP

	C.1	C.2	C.3	C.1	C.2	C.3	C.4	C.5	C.6
Req. 1	1202.62	IP1	IP0	1202.6	IP1	IP0	1202.62	1202.6	1
Req. 2	1204.273	IP1	IP0	1204.3	IP1	IP0	1202.72	1202.7	0
	1514.963	IP2	IP0	1515	IP2	IP0	1202.82	1202.8	0
	1514.963	IP3	IP0	1515	IP3	IP0	1202.92	1202.9	0
	3179.672	IP3	IP0	3179.7	IP3	IP0	1203.02	1203	0
	3666.124	IP3	IP0	3666.1	IP3	IP0	1203.12	1203.1	0
	3669.187	IP3	IP0	3669.2	IP3	IP0	1203.22	1203.2	0
	4899.327	IP3	IP0	4899.3	IP3	IP0	1203.32	1203.3	0
	5158.257	IP3	IP0	5158.3	IP3	IP0	1203.42	1203.4	0
	5161.262	IP3	IP0	5161.3	IP3	IP0	1203.52	1203.5	0
	5216.272	IP3	IP0	5216.3	IP3	IP0	1203.62	1203.6	0
	5245.112	IP3	IP0	5245.1	IP3	IP0	1203.72	1203.7	0
	5245.237	IP3	IP0	5245.2	IP3	IP0	1203.82	1203.8	0
	5245.296	IP3	IP0	5245.3	IP3	IP0	1203.92	1203.9	0
	5245.409	IP3	IP0	5245.4	IP3	IP0	1204.02	1204	0
	5245.499	IP3	IP0	5245.5	IP3	IP0	1204.12	1204.1	0
	5245.553	IP3	IP0	5245.6	IP3	IP0	1204.22	1204.2	0
	5249.704	IP3	IP0	5249.7	IP3	IP0	1204.32	1204.3	1

BEFORE NORMALIZATION
Matrix1
1238 x 3
AFTER NORMALIZATION
Matrix1
1238 x 3
FIX TIME STEPS BEFORE NORMALIZATION
160236 X 1
Matrix2
FIX TIME STEPS AFTER NORMALIZATION
160236 X 1
Matrix2
Total Number of Steps : 17
FIELD REFERENCE OF TIME FOR IP1
1

Figure 5.32: Processing real data in implementation form.

Chapter 6

Conclusions

Communication Information Systems have become a major asset in human society and economy. Their use in social media, email, financial services, and other applications, through the internet determines the digital identity of the humans which is in fact their physical identity. Effectively, humans have become part of the cyberspace. Taking account all the above, new security matters arise. All of our activities use and, in fact, rely on the internet.

The availability of these communication information systems, is extremely crucial to humans. At this point, a new challenge has been developed, threatening the availability of the Communication Information Systems, namely DDoS attack. Until now, dozens of different approaches have been proposed to deal proactively with DDoS attacks. Their main objective is to identify early enough any collaboration between users. Despite these efforts, DDoS attacks still happen, and are growing in terms of volume and catastrofic impact. Security experts are not optimistic about the way ahead; they believe that the worst case scenario is in front of us as the Internet of Things is on the way, and proactive in-depth solutions have not been found yet.

Taking account all the above, our motivation in order to work to this field we're the numerous zero day attacks that happened in recent past years and the security gap that seems to grow at a rapid pace. If we take account that all of the communication information systems, will become a part of the Internet of Things in the next years, then we can figure out easily the size and type of future robot network (botnet). The effective defense to this scenario is the identification of the botnet network early enough, before it gains its full power.

The main objective of this dissertation is to fill up the above mentioned security gap by introducing a novel approach regarding malware analysis by establishing a reliable and simple in the implementation method regarding collaboration in user behaviour in order (1) to avoid the consequences of a Distributed Denial of Service attack, (2) to recognise early enough any cyber incidents, (3) find any collaboration/dependency among different users, (4) ultimately isolate any abnormal behaviour before a minor cyber incident expands to a catastrophic DDoS attack with extremely high cost.

The contribution of this dissertation is the proposal of a simple method that can be implemented in any kind of information system, including anti-DDoS devices or Intrution Detection Systems (IDS), or can be used as a second line of defense in relation with other methods. In order to implemented the proposed method, there is no need to have any related information about the botnet nor its strategy. The method is based on the theory of hidden Markov models, without using them in the implementation (only for justification). Furthermore, the method gives a novel approach in malware analysis and provides a new idea for future research in the detection of malicious internet users. It is estimated that there are many prospects of developing the method due to the fact this dissertation only proposed the core theory without the combinations or variants that can be applied.

To achieve all the above, we organised the dissertation by providing background material and presented the state-of-the-art best practices in place regarding DDoS attacks. Next we provide, the definition of a DoS attack, the description of hidden Markov models and related theory, and the theoretical analysis of the proposed approach. We also describe our enhanced proposed approach, which includes an improved detection strategy; lessons learned are also discussed in detail. Finally, we include simulations, an evaluation of the performance of the proposed model with real data, and a summary of the main findings regarding our evaluation in real scenario events.

6.1 Remarks

Our method can be applied to any kind of Communication Information System. Specifically, it can be part of the main configuration of an Intrusion Detection System (IDS) and it can be implemented as a pre-processor module, in order to identify early enough any abnormal events, and establish the appropriate cliques of dependent users (which are in collaboration during a DDoS attack).

The complexity of the method should not be seen as an inhibiting factor. Taking into account that there is no need for additional information about the users (except for their behaviour during their activity in the network), this becomes very interesting and at the same time extremely valuable. The main asset of the method is that there is no need to have any additional information related with the attacker. If this method is implemented in distributive way, then the main complexity factors (which are the number of the users and the number of the steps), can be minimized. With this technique, if we assume that we use parallel processing in real time, the processing of the data will take place in distributive manner, so that the processed data may be transmitted to the main processing unit of the implemented device. In this way, the defender may have the whole picture in a sense of a network tomography with all related information and all updated critical information in real time.

If we take account the above, it is clear that with the distributive and in depth implementation of the method which has been described above, is not an inhibiting factor and not so complicated for practical applications.

Recent cyber-events related to DDoS attacks have taught us that a DDoS attack is executed in full strength during a time window of approximately 15-20 minutes [53]. Our proposed method can used as a second line of defense, in relation with other methods. In that way, we believe that appropriate safe and reliable results can be found in the first 10 minutes and potentially will allow us to to stop a DDoS attack before it reaches its full strength and consume all of our resources.

6.2 Future Work

As it was already stated above, the proposed approach only refers to the core related theory. Areas of further development have already been identified in terms of two pillars. The first pillar refers to proactive measures and the second pillar to reactive measures. Both measures are critical and contribute to the overall safety, in terms of overcoming DDoS attacks. The contribution of these measures is critical in terms of reliability, due to the fact that they can minimize effectively the risk of findings related to false positive alarms or false negative alarms. The proposed approach can be explored further in terms of proactive and reactive measures discussed below.

6.2.1 Proactive Measures

The areas that can be explored further, regarding proactive measures include the following.

- The mean packet rate can be more precisely identified using Poisson process analysis and information theory measures (entropy). With this technique, it is believed that the proposed approach can be upgraded and work independenly.
- Simulate behaviours among the users involving time shifting constraints. With this technique the proposed approach will be able to identify malicious users, in direct mode, so that botnets that are activated through time by the same attacker can be found early enough. This capability will give to defenders appropriate time to act proactively and even uncover the attacker himself, who is hiding behind the botnet.

6.2.2 Reactive Measures

- In relation with the techniques used in sandboxes, the revealed challenge is the decryption of the hidden information related with the bots and specifically how they implement their command/control channels. This approach can lead us closer to the identification of the hidden strategy of the attackers, which means that we can actually identify the geo-location of the botnet and all related information. This can be accomplished if we can collect early enough the possible information related to the attacker and specifically the cliques of the botnet that are on effect using time shifting constraints.
- Parallel HMM composition can be used to predict the needed time to reach steady-state. This will be an asset when trying to understand in depth time-varying strategies. With this methodology, the defenders can be able to identify new signatures of malwares and in practice to identify the different transformation coding of the attacker, which will give us the capability to understand how it works, in order to find them. If this is combined with somewhat of a data base, then the all system can be transformed in an autonomous machine learning system.

Bibliography

[1] P. Campbell, "The denial-of-service dance," *IEEE Security Privacy*, vol. 3, pp. 34–40, Nov. 2005.

[2] M. Thoma and C. N. Hadjicostis, "Detection of collaborative cyber-attacks through correlation and time dependency analysis," in *Proc. of the 18th Mediterranean Electrotechnical Conference (MELECON)*, pp. 1–6, April 2016.

[3] C. Schuba, I. Krsul, M. Kuhn, E. Spafford, A. Sundaram, and D. Zamboni, "Analysis of a denial of service attack on TCP," in *Proc. of the IEEE Symposium on Security and Privacy*, pp. 208–223, May 1997.

[4] K. Zeb, O. Baig, and M. Asif, "DDoS attacks and countermeasures in cyberspace," in *Proc. of the 2nd World Symposium on Web Applications and Networking (WSWAN)*, pp. 1–6, March 2015.

[5] J. Chang and R. Xue, "A framework based on time and space for analyzing denial of service attacks," in *Proc. of the International Symposium on Information Science and Engineering (ISISE)*, pp. 412–417, Dec. 2012.

[6] W.-Z. Lu and S. zheng Yu, "An http flooding detection method based on browser behavior," in *Proc. of the International Conference on Computational Intelligence and Security*, vol. 2, pp. 1151–1154, Nov 2006.

[7] T. Peng, C. Leckie, and K. Ramamohanarao, "Protection from distributed denial of service attacks using history-based IP filtering," in *Proc. of the IEEE International Conference on Communications, ICC*, vol. 1, pp. 482–486, May 2003.

[8] R. Poisel, M. Rybnicek, and S. Tjoa, "Game-based simulation of Distributed Denial of Service DDoS attack and defense mechanisms of critical infrastructures," in *Proc. of the IEEE 27th International Conference on Advanced Information Networking and Applications (AINA)*, pp. 114–120, March 2013.

[9] S. Yu, W. Zhou, and R. Doss, "Information theory based detection against network behavior mimicking DDoS attacks," *IEEE Communications Letters*, vol. 12, pp. 318–321, April 2008.

[10] Y. Tao and S. Yu, "DDoS attack detection at local area networks using information theoretical metrics," in *Proc. of the 12th IEEE International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom)*, pp. 233–240, July 2013.

[11] G. Carl, G. Kesidis, R. Brooks, and S. Rai, "Denial-of-service attack-detection techniques," *IEEE Internet Computing*, vol. 10, pp. 82–89, Jan. 2006.

[12] Y. Zhang, Q. Liu, and G. Zhao, "A real-time DDoS attack detection and prevention system based on per-IP traffic behavioral analysis," in *Proc. of the 3rd IEEE International Conference on Computer Science and Information Technology (ICCSIT)*, vol. 2, pp. 163–167, July 2010.

[13] K. Kumar, A. Sangal, and A. Bhandari, "Traceback techniques against DDoS attacks: A comprehensive review," in *Proc. of the 2nd International Conference on Computer and Communication Technology (ICCCT)*, pp. 491–498, Sept. 2011.

[14] D. Wang, L. He, Y. Xue, and Y. Dong, "Exploiting artificial immune systems to detect unknown DoS attacks in real-time," in *Proc. of the IEEE 2nd International Conference on Cloud Computing and Intelligent Systems (CCIS)*, vol. 02, pp. 646–650, Oct. 2012.

[15] R. R. Kompella, S. Singh, and G. Varghese, "On scalable attack detection in the network," *IEEE/ACM Transactions on Networking*, vol. 15, pp. 14–25, Feb 2007.

[16] W. Wang and S. Gombault, "Efficient detection of DDoS attacks with important attributes," in *Proc. of Third International Conference on Risks and Security of Internet and Systems, CRiSIS*, pp. 61–67, Oct. 2008.

[17] J.-H. Jun, H. Oh, and S.-H. Kim, "DDoS flooding attack detection through a step-by-step investigation," in *Proc. of the IEEE 2nd International Conference on Networked Embedded Systems for Enterprise Applications (NESEA)*, pp. 1–5, Dec. 2011.

[18] Z. Li, G. Hu, and X. Yao, "Spatial correlation detection of DDoS attack," in *Proc. of the International Conference on Communications, Circuits and Systems, ICCCAS*, pp. 304–308, July 2009.

[19] Z. Gao and N. Ansari, "Differentiating malicious DDoS attack traffic from normal TCP flows by proactive tests," *IEEE Communications Letters*, vol. 10, pp. 793–795, November 2006.

[20] M. Sung and J. Xu, "IP traceback-based intelligent packet filtering: A novel technique for defending against internet DDoS attacks," in *Proc. of the 10th IEEE International Conference onNetwork Protocols*, pp. 302–311, Nov. 2002.

[21] A. Habib and D. Roy, "Steps to defend against DoS attacks," in *Proc. of the 12th International Conference on Computers and Information Technology, ICCIT*, pp. 614–619, Dec. 2009.

[22] S. Yu, T. Thapngam, J. Liu, S. Wei, and W. Zhou, "Discriminating DDoS flows from flash crowds using information distance," in *Proc. of the Third International Conference on Network and System Security, NSS*, pp. 351–356, Oct. 2009.

[23] H. Chen and Y. Chen, "A novel embedded accelerator for online detection of shrew DDoS Attacks," in *Proc. of the International Conference on Networking, Architecture, and Storage, NAS*, pp. 365–372, June 2008.

[24] Y. Xiang, K. Li, and W. Zhou, "Low-rate DDoS attacks detection and traceback by using new information metrics," *IEEE Transactions on Information Forensics and Security*, vol. 6, pp. 426–437, June 2011.

[25] H. Hu, J. Zhang, B. Liu, L. Chen, and X. Chen, "Simulation and analysis of distributed low-rate denial-of-service attacks," in *Proc. of the 5th International Conference on Computer Sciences and Convergence Information Technology (ICCIT)*, pp. 620–626, Nov. 2010.

[26] K. Chen, H. Liu, and X. Chen, "Ebdt: A method for detecting LDoS attack," in *Proc. of the International Conference on Information and Automation (ICIA)*, pp. 911–916, June 2012.

[27] Z. Wu, C. Wang, and H. Zeng, "Research on the comparison of Flood DDoS and Low-rate DDoS," in *Proc. of the International Conference on Multimedia Technology (ICMT)*, pp. 5503–5506, July 2011.

[28] E. Iasiello, "Cyber attack: A dull tool to shape foreign policy," in *Proc. of the 5th International Conference on Cyber Conflict (CyCon)*, pp. 1–18, June 2013.

[29] J. Caso, "The rules of engagement for cyber-warfare and the Tallinn manual: A case study," in *Proc. of the IEEE 4th Annual International Conference on Cyber Technology in Automation, Control, and Intelligent Systems (CYBER)*, pp. 252–257, June 2014.

[30] M. Donner, "Cyberassault on Estonia," *IEEE Security and Privacy*, vol. 5, pp. 4–4, July 2007.

[31] M. Lesk, "The new front line: Estonia under cyberassault," *IEEE Security and Privacy*, vol. 5, pp. 76–79, July 2007.

[32] S. Nagar, S. S. Rajput, A. K. Gupta, and M. C. Trivedi, "Secure routing against DDoS attack in wireless sensor network," in *Proc. of the 3rd International Conference on Computational Intelligence Communication Technology (CICT)*, pp. 1–6, Feb 2017.

[33] Q. Yan, F. R. Yu, Q. Gong, and J. Li, "Software-defined networking (SDN) and distributed denial of service (DDoS) attacks in cloud computing environments: A survey, some research issues, and challenges," *IEEE Communications Surveys Tutorials*, vol. 18, pp. 602–622, First quarter 2016.

[34] J. Liu, Y. Lai, and S. Zhang, "Fl-guard: A detection and defense system for DDoS attack in SDN," in *Proc. of the 2017 International Conference on Cryptography, Security and Privacy*, pp. 107–111, ACM, 2017.

[35] O. Osanaiye, K.-K. R. Choo, and M. Dlodlo, "Distributed denial of service (DDoS) resilience in cloud: review and conceptual cloud DDoS mitigation framework," *Journal of Network and Computer Applications*, vol. 67, pp. 147–165, 2016.

[36] G. Somani, M. S. Gaur, D. Sanghi, M. Conti, and R. Buyya, "DDoS attacks in cloud computing: issues, taxonomy, and future directions," *Computer Communications*, 2017.

[37] Z. Guo, D. Shi, K. H. Johansson, and L. Shi, "Optimal linear cyber-attack on remote state estimation," *IEEE Transactions on Control of Network Systems*, vol. 4, no. 1, pp. 4–13, 2017.

[38] C. Zhang and R. Green, "Communication security in internet of thing: preventive measure and avoid DDoS attack over IoT network," in *Proc. of the 18th Symposium on Communications & Networking*, pp. 8–15, Society for Computer Simulation International, 2015.

[39] R. Braga, E. Mota, and A. Passito, "Lightweight DDoS flooding attack detection using nox/openflow," in *IEEE Local Computer Network Conference*, pp. 408–415, 2010.

[40] A. Saied, R. E. Overill, and T. Radzik, "Detection of known and unknown DDoS attacks using artificial neural networks," *Neurocomputing*, vol. 172, pp. 385–393, 2016.

[41] S. Yu, W. Zhou, R. Doss, and W. Jia, "Traceback of DDoS attacks using entropy variations," *IEEE Transactions on Parallel and Distributed Systems*, vol. 22, pp. 412–425, March 2011.

[42] O. Osanaiye, H. Cai, K.-K. R. Choo, A. Dehghantanha, Z. Xu, and M. Dlodlo, "Ensemble-based multi-filter feature selection method for DDoS detection in cloud computing," *EURASIP Journal on Wireless Communications and Networking*, vol. 2016, no. 1, p. 130, 2016.

[43] W. Bhaya and M. EbadyManaa, "DDoS attack detection approach using an efficient cluster analysis in large data scale," in *Proc. IEEE Annual Conference on, New Trends in Information & Communications Technology Applications (NTICT)*, pp. 168–173, 2017.

[44] M. H. Bhuyan, D. Bhattacharyya, and J. K. Kalita, "An empirical evaluation of information metrics for low-rate and high-rate DDoS attack detection," *Pattern Recognition Letters*, vol. 51, pp. 1–7, 2015.

[45] İ. Özçelik and R. R. Brooks, "Deceiving entropy based DoS detection," *Computers & Security*, vol. 48, pp. 234–245, 2015.

[46] K. J. Singh and T. De, *An Approach of DDoS Attack Detection Using Classifiers*. Springer, 2015.

[47] F. Hao, G. Min, Z. Pei, D.-S. Park, and L. T. Yang, "*K*-Clique community detection in social networks based on formal concept analysis," *IEEE Systems Journal*, vol. 11, no. 1, pp. 250–259, 2017.

[48] P. Xiao, W. Qu, H. Qi, and Z. Li, "Detecting DDoS attacks against data center with correlation analysis," *Computer Communications*, vol. 67, pp. 66–74, 2015.

[49] Y. Chen and K. Hwang, "Collaborative change detection of DDoS attacks on community and ISP networks," in *Proc. International Symposium on Collaborative Technologies and Systems (CTS'06)*, pp. 401–410, May 2006.

[50] B. Gao, H.-Y. Ma, and Y.-H. Yang, "HMMs (Hidden Markov models) based on anomaly intrusion detection method," in *Proc. of the International Conference on Machine Learning and Cybernetics*, vol. 1, pp. 381–385, 2002.

[51] L. Rabiner, "A tutorial on Hidden Markov models and selected applications in speech recognition," *Proc. of the IEEE*, vol. 77, pp. 257–286, Feb 1989.

[52] L. Rabiner and B. Juang, "An introduction to Hidden Markov models," *IEEE, ASSP Magazine*, vol. 3, pp. 4–16, Jan 1986.

[53] "http://www.securityweek.com,"

[54] "https://mcfp.felk.cvut.cz,"

Appendix A

Acronyms

AOMDV	Auto Request Protocol
AI	Artificial Inteligence
ANN	Artificial Neural Network
ARQ	Auto Request Protocol
BOTNET	Robot Network
CIS	Communication Information System
CPS	Cyber Physical System
DoS	Denial of Service Attack
DDoS	Distributed Denial of Service Attack
DRDoS	Distributed Reflection Denial of Service Attack
FCM	Fair Coin Markov Model
JNNS	Java Neural Network Simulator
NIST	National Institute of Standard and Technology
HMM	Hidden Markov Model
IDS	Intrusion Detection Systems
IoT	Internet of Things
ICMP	Internet Control Message Protocol
IPS	Intrusion Prevention System
IP	Internet Protocol
ISP	Internet Service Provider
PCs	Personal Computers
SDN	Software Defined Network
TCP	Transmission Control Protocol
WSN	Wireless Sensor Network
UDP	User Datagram Protocol

Appendix B

Notation

$\hat{u}_1$	The requests from User One
$\hat{u}_2$	The requests from User Two
$\hat{u}$	Comparable Frequency (Baseline Feature)
$\hat{u}_{12}$	Relative Frequency (Baseline Feature)
$F_{relative}$	Relative Frequency (Enhanced Feature)
$F_{comparable}$	Comparable Frequency (Enhanced Feature)
$F_{MeanPacketRate}$	The Frequency Mean Packet Rate (Enhanced Feature)
$U_{Observable} rate$	Users Activity Rate (Enhanced Feature)

Appendix C

Definitions

Attacker	The top botnet leader
Abnormal User	Users with very high traffic
Dependent User	Users that appear dependent with other users
Flash Crowd User	Users that do not behave normally or/and appear dependent with other users
Independent User	Users that they do not appear dependent with other users
Master Zombies	Users that are member of the botnet and lead groups of zombies
No Activity User	Users that appear inactive
Zombies	Simple individual users that are members of the botnet