Towards Coordinated, Network-Wide Traffic Monitoring for Early Detection of DDoS Flooding Attacks

by

Saman Taghavi Zargar

B.Sc., Computer Engineering (Software Engineering), Azad University of Mashhad, 2004

M.Sc., Computer Engineering (Software Engineering), Ferdowsi University of Mashhad, 2007

Submitted to the Graduate Faculty of
the School of Information Sciences in partial fulfillment
of the requirements for the degree of

Doctor of Philosophy

University of Pittsburgh

2014

UNIVERSITY OF PITTSBURGH

SCHOOL OF INFORMATION SCIENCES

This dissertation was presented

by

Saman Taghavi Zargar

It was defended on

June 3, 2014

and approved by

James Joshi, Ph.D., Associate Professor, School of Information Sciences

David Tipper, Ph.D., Associate Professor, School of Information Sciences

Prashant Krishnamurthy, Ph.D., Associate Professor, School of Information Sciences

Konstantinos Pelechrinis, Ph.D., Assistant Professor, School of Information Sciences

Yi Qian, Ph.D., Associate Professor, College of Engineering,

Dissertation Advisors: James Joshi, Ph.D., Associate Professor, School of Information Sciences,

David Tipper, Ph.D., Associate Professor, School of Information Sciences

Towards Coordinated, Network-Wide Traffic Monitoring for Early Detection of DDoS Flooding Attacks

Saman Taghavi Zargar, PhD

University of Pittsburgh, 2014

DDoS flooding attacks are one of the biggest concerns for security professionals and they are typically explicit attempts to disrupt legitimate users' access to services. Developing a comprehensive defense mechanism against such attacks requires a comprehensive understanding of the problem and the techniques that have been used thus far in preventing, detecting, and responding to various such attacks.

In this thesis, we dig into the problem of DDoS flooding attacks from four directions: (1) We study the origin of these attacks, their variations, and various existing defense mechanisms against them. Our literature review gives insight into a list of key required features for the next generation of DDoS flooding defense mechanisms. The most important requirement on this list is to see more distributed DDoS flooding defense mechanisms in near future, (2) In such systems, the success in detecting DDoS flooding attacks earlier and in a distributed fashion is highly dependent on the quality and quantity of the traffic flows that are covered by the employed traffic monitoring mechanisms. This motivates us to study and understand the challenges of existing traffic monitoring mechanisms, (3) We propose a novel distributed, coordinated, network-wide traffic monitoring (DiCoTraM) approach that addresses the key challenges of current traffic monitoring mechanisms. DiCoTraM enhances flow coverage to enable effective, early detection of DDoS flooding attacks. We compare and evaluate the performance of DiCoTraM with various other traffic monitoring mechanisms in terms of their total flow coverage and DDoS flooding attack flow coverage, and (4) We evaluate the effectiveness of DiCoTraM with cSamp, an existing traffic monitoring mechanism that

outperforms most of other traffic monitoring mechanisms, with regards to supporting early detection of DDoS flooding attacks (i.e., at the intermediate network) by employing two existing DDoS flooding detection mechanisms over them. We then compare the effectiveness of DiCoTraM with that of cSamp by comparing the detection rates and false positive rates achieved when the selected detection mechanisms are employed over DiCoTraM and cSamp. The results show that DiCoTraM outperforms other traffic monitoring mechanisms in terms of total flow coverage and DDoS flooding attack flow coverage.

PREFACE

First, I am very grateful to my advisor, Professor James Joshi, for his great guidance, continuous encouragement, invaluable suggestions, and support throughout my PhD. I would like to thank him for giving me the great opportunity to work under his supervision as a member of LERSAIS lab.

I am also very grateful to my co-advisor, Professor David Tipper, for his help and guidance that have shaped much of the theoretical and technical basis in this thesis. I have been and am going to be amazed by his incredible attention to technical detail. I hope I have inherited at least some of these characteristics from him during this time.

I would also like to thank my dissertation committee members Professor Prashant Krishnamurthy, Professor Yi Qian, and Professor Konstantinos Pelechrinis for their valuable feedback on my work. I would like to acknowledge the confidence and support that Professor Prashant Krishnamurthy have shown for my work. His support when things were getting tough was a great morale booster and he has always surprised me on many occasions by his great insights to get to the depth of a research question even without detailed discussions.

I wish to thank my fellow colleagues at LERSAIS lab, Hassan Takabi, Amirreza Masoumzadeh Tork, Nathalie Angel Baracaldo, Jesus Gonzales, Lei Jin, and Xulian Long for the fruitful discussions and the knowledge they shared with me. I also thank the faculty and staff of the School of Information Sciences, especially people listed below for being incredibly resourceful and awesome. I have never come across something that they did not know the answer to.

- Dean Ron Larsen, Dr. Martin Weiss, Kelly Shaffer, Wesley Lipschultz, Brandi Belleau, Sharon Bindas, Sandra Brandon, Marcy Walls, and Corey James.

I also acknowledge that the research presented in this dissertation has been supported by

the financial assistance provided through the School of Information Sciences at the University of Pittsburgh, US National Science Foundation Computing and Communication Foundations (CCF) award CCF-0720737, and Cisco systems Inc. award.

Finally, I would like to thank my beloved parents, sister, and brother for their endless support in every step I took so far towards my career goals. I have been so lucky to have such luxury. This thesis would not have been possible without the encouragement of my family.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

1.0 Introduction and Motivation

According to CERT [1], with the rapid growth of Internet services, we have also seen increased number of possible attacks against these services. DDoS flooding attacks have been reported as one of the attacks with the highest occurrence rate over the past decade and many Internet service providers (ISPs) and users have seriously suffered from these attacks. Despite all of the efforts towards decreasing the number of DDoS flooding attack incidents, they have expanded rapidly in the frequency and the size of the targeted networks and computers over the past decade [2].

Various defense mechanisms have been proposed in the literature to address DDoS flooding attacks [2]. These mechanisms could be categorized into four categories based on the location where they are implemented [2–5]: *Source-based*, *destination-based*, *network-based*, and *hybrid (a.k.a. distributed)*. The detection accuracy is high at the victim side (i.e., destination) but it is not robust; victims cannot tolerate high volume of DDoS traffic. Stopping the attacks at the source could be the best solution but it is very difficult since the volume of the traffic at the sources is not significant to differentiate between legitimate and malicious traffic. Furthermore, the collateral damage is high at the intermediate networks because there is not enough resources to profile the traffic. Therefore, centralized mechanisms in which all the defense components (i.e., prevention, detection, and response) are deployed at the same place, are not practical against DDoS flooding attacks (See Chapter 2 for detailed comparison).

Distributed collaboration among heterogeneous components within and across independent domains has been indicated in the recent literature as a key challenge and future direction for research in cybersecurity [6]. Since the attackers cooperate to perform successful attacks, defenders must also form alliances and collaborate with each other to defeat

the attacks. Our review of the literature on DDoS flooding defense mechanisms show a list of key features the next generation of DDoS flooding defense mechanisms should have [2] (explained fully in Section 2.4). The most important requirement on this list that motivates us to carry out the research in this thesis is to see more distributed DDoS flooding defense mechanisms (i.e., distributed monitoring, detection, and response) in near future; since, as explained above, they are more practical and effective than centralized mechanisms [2].

We argue that in an ideal distributed DDoS flooding defense mechanism, within each Autonomous System (AS), traffic flows/packets should be monitored in a distributed fashion, the monitored flows/packets should be analyzed by the detection mechanisms distributively, and finally, upon detection of DDoS flooding attacks, the distributed response mechanisms should respond appropriately to the attack flows/packets. Hence, the success in detecting the DDoS flooding attacks is highly dependent on the quality and quantity of the monitored flows/packets by the distributed traffic monitoring mechanisms that will be employed by the distributed DDoS flooding defense mechanisms. Implementing such distributed traffic monitoring mechanism have various challenges (see Chapter 3 for detail). In this thesis, we propose a novel distributed traffic monitoring mechanism that plays an important role, as the first step in defending against DDoS flooding attacks, in maximizing the coverage of flows that are monitored to support the detection of DDoS flooding attacks. Moreover, it will address the main limitations of existing traffic monitoring mechanisms. Such an approach will be beneficial to future distributed DDoS flooding defense mechanisms.

1.1 Thesis Statement and Challenges

In this thesis, we carry out research to validate the following hypothesis: *a network-wide traffic monitoring mechanism that coordinates monitoring responsibilities among various monitoring devices (centrally managed) can eliminate the redundant flow monitoring issue, while satisfying the resource constraints of the monitoring devices. Such traffic monitoring mechanism can also help prioritize the coverage of the DDoS flooding attack flows which can enable early detection of DDoS flooding attacks through distributed DDoS flooding detection*

mechanisms. Towards this, we propose a novel distributed, coordinated, network-wide DDoS flooding attack tailored flow monitoring mechanism (DiCoTraM) that addresses the key challenges presented below. In section 1.2, we overview the specific research components and contributions.

1.1.1 Challenge 1: Network-wide approach for traffic monitoring policy enforcement

As computer networks get more complex and dynamic, and as traffic patterns evolve, network management tasks such as: traffic monitoring, performance optimization, etc. have become more and more challenging. This is because such changes in computer networks and traffic patterns may require new functionalities and high-level (a.k.a. network-wide) policies that network operators are responsible for and struggle with enforcing them. For instance, dynamic flow coverage in different monitoring devices within the network for specific flows is an example of such a high-level policy. Enforcing such high-level policies is challenging since employing traditional low-level vendor-specific configuration and analysis approaches or deploying third party middleboxes for this purpose are very difficult and impractical.

The main reason for the insufficiency of available policy enforcement approaches (see Chapter 3) to the network operators is the device-centric view of these approaches. Moreover, all of the network elements are constrained by their technological constraints such as their processing and memory capabilities. For instance, monitoring devices duplicate monitoring tasks and some of them become overloaded. Moreover, translating a network-wide traffic monitoring objective (e.g., guaranteed minimum flow coverage for specific flows) into routers' configuration and in a router-by-router fashion (device-centric) is tedious and there is no guarantee if routers could correctly satisfy their translated objective.

Some of the recent proposals argue that network management tasks such as traffic monitoring can be viewed as resource management problems (e.g., [136]); hence, network-wide traffic monitoring objectives can be achieved using a network-wide resource management approach. Such approaches perform within the technological constraints of the existing vendor-specific monitoring devices that are already deployed in the network. This minimizes

the costs of alternative solutions such as: adding special purpose middleboxes or monitoring devices with special capabilities.

1.1.2 Challenge 2: Resource-aware DDoS flooding attack tailored monitoring policy

Another challenge in traffic monitoring mechanisms originates from the fact that the monitoring devices (e.g., routers) have limited resources (e.g., memory and CPU). Hence, when the number of traffic flows passing through them is very large they only cover a subset of flows/packets. Moreover, this flow/packet selection is mostly done in a random fashion and without prioritizing the coverage of possible important flows/packets (e.g., DDoS flooding flows).

We argue that it is important for a DDoS defense mechanism to ensure that the majority of the flows destined for the same destination (possible DDoS flooding flows) are monitored in each and every AS [1]. Hence, guaranteeing the monitoring coverage for the majority of those flows should be an important policy to enforce for the monitoring mechanisms employed by the DDoS defense mechanisms while satisfying the monitoring devices' memory constraints. However, this sometimes will require network operators to augment some of the monitoring devices with more resources for their short-term/long-term network planning decisions. Obviously, there is a trade-off between the amount of resources in monitoring devices and the monitoring coverage for all the flows, an issue that we explore and analyze later in this thesis.

1.1.3 Challenge 3: Distributed monitoring policy to enable distributed detection of DDoS flooding attack flows

Currently, most of the DDoS flooding defense mechanisms analyze the flows/packets that are monitored by distributed traffic monitoring devices centrally within each AS. This approach requires routers (monitoring devices) to report their monitored flows/packets to some cen-

[1] The bandwidth starvation DDoS flooding attacks are the focus of this thesis. The port starvation DDoS flooding attacks are not the focus of this thesis in its current version

tral points for analysis; this is challenging since it causes a huge communication overhead. We believe that with practical distributed monitoring mechanisms that can cover as many flows/packets as possible (specially DDoS flooding attack flows) in place, it is more feasible, effective, and reasonable to have light-weight distributed detection mechanisms in place than performing centralized analysis (detection).

1.2 Proposed Research and Contributions

DiCoTraM eliminates the redundant flow monitoring while satisfying the resource constraints of the monitoring devices. DiCoTraM prioritizes the coverage of the DDoS flooding attack flows. DiCoTraM's capability in covering most of the DDoS flooding flows enables the early detection of DDoS flooding attacks in the intermediate network and before these attack flows get closer to their victims. In particular, the research presented in this dissertation makes the following contributions:

- We study the origin of DDoS flooding attacks, their variations, and various existing defense mechanisms against them. We summarize the list of required features for the next generation of DDoS flooding defense mechanisms. The most important requirement on this list is to see more distributed DDoS flooding defense mechanisms in near future which motivated us to conduct the research presented in this thesis. As our second step, we study the existing traffic monitoring mechanisms and identify their various challenges in terms of the problem of DDoS flooding attack. The success in detecting DDoS flooding attacks in a distributed fashion is highly dependent on the quantity of (i.e., flow coverage) the distributed traffic monitoring mechanisms that are being employed.
- We present the design and proof of concept implementation of DiCoTraM as a novel traffic monitoring mechanism to address the limitations of current traffic monitoring mechanisms with regards to the problem of DDoS flooding attack. DiCoTraM is a resource constraint aware, DDoS flooding attack tailored, coordinated, and network-wide traffic monitoring mechanism (see Chapter 3 [7]). We compare and evaluate the performance of DiCoTraM with various other traffic monitoring mechanisms in terms of

total flow coverage and DDoS flooding attack flow coverage in order to show DiCoTraM's effectiveness.

- We analyze the scalability of DiCoTraM and improve it by proposing a scalable heuristic. Moreover, in order to reduce the computation time of the monitoring assignment process in our proposed heuristic, we enhance the heuristic by pre-processing the input to the monitoring assignment process; consequently, our proposed enhanced heuristic only runs the monitoring assignment process for the set of new or modified flows in compared to the previous assignments. In doing so, the enhanced heuristic keeps the same assignment for the long-term flows as far as the total DDoS flooding flow coverage is maintained or does not significantly fallen below some predefined threshold.
- We extend DiCoTraM to help the network operators in determining a list of candidate monitoring devices to upgrade (i.e., memory upgrade) for achieving better flow coverage. Extended version of DiCoTraM computes the amount of additional memory needed to fully cover the traffic flows. We evaluate the memory consumption of the routers within a well-known AS topology for both with and without DDoS flooding attacks by means of simulations. Moreover, we perform a set of experiments in the same AS topology to evaluate the effectiveness of the network planning capability that our extension provides in terms of flow coverage in both with and without DDoS flooding attacks.

- Finally, we evaluate the effectiveness of DiCoTraM with an existing traffic monitoring mechanism, that outperforms most of other traffic monitoring mechanisms, with regards to supporting effective, early detection of DDoS flooding attacks by employing two existing DDoS flooding detection mechanisms over them. These two DDoS flooding detection mechanisms are selected since they have been shown to have good performance in detecting DDoS flooding attacks in the literature. We then compare the effectiveness of DiCoTraM with that of other traffic monitoring mechanism by comparing the detection rates and false positive rates achieved when the selected detection mechanisms are employed over DiCoTraM and the other traffic monitoring mechanism.

1.3 Scope of the Thesis

The proposed work in this thesis is motivated by the defense against DDoS flooding attacks. However, currently, we focus only on monitoring the bandwidth starvation DDoS flooding attacks and other types of DDoS flooding attacks such as port starvation DDoS flooding attacks have not considered at present. Moreover, our proposed traffic monitoring mechanism can only identify and monitor the potential DDoS flooding attack flows and not the actual attackers who generated those flows. Hence, the actual sources of in particular DDoS flooding attacks such as reflected/spoofed DDoS flooding attacks (e.g., DNS amplification attacks) cannot be correctly identified through the detection mechanisms employed in our experiments; hence these attack types are not within the scope of this thesis at present.

1.4 Organization

The remainder of this dissertation is organized as follow: Chapter 2 presents a comprehensive survey of DDoS flooding attacks, their variations, existing solutions to the problem, and a summary of features that an ideal comprehensive DDoS defense mechanism must have to combat DDoS flooding attacks [2]. Chapter 3 studies the existing traffic monitoring mechanisms and their pros and cons in detail. Chapter 4 presents the design and implementation of DiCoTraM towards addressing the challenges of existing traffic monitoring mechanisms in achieving the requirements of the next generation of DDoS flooding defense mechanisms; traffic monitoring mechanisms that provide larger DDoS flow coverage [7]. Chapter 5, evaluates the effectiveness of DiCoTraM with an existing traffic monitoring mechanism, that outperforms most of the other traffic monitoring mechanisms, with regards to supporting effective, early detection of DDoS flooding attacks by employing two existing well-performing detection mechanisms over them. Finally, Chapter 6 concludes the dissertation, summarizes some possible limitations, and discusses the future work.

2.0 DDoS Flooding Attacks & Existing Defense Mechanisms

This chapter briefly (see [2] for more detailed version) overviews the DDoS phenomena. We first explain DDoS problem and its background in Section 2.1. Then we categorize the existing DDoS flooding attacks in Section 2.2. We categorize DDoS flooding attacks into two types based on the protocol level that is targeted: network/transport-level attacks and application-level attacks. Then, we enumerate some of the major attacks in each category. In Section 2.3, we describe our classification of the defense mechanisms for DDoS flooding attacks and enumerate various defense mechanisms against DDoS flooding attacks. We classify the defense mechanisms against the two types of DDoS flooding attacks that we present in section 2.2 using two criteria. First we classify both the defense mechanisms against network/transport-level DDoS flooding attacks and the defense mechanisms against application-level DDoS flooding attacks based on the location where prevention, detection, and response to the DDoS flooding attacks occur. Then we classify both types of defense mechanisms based on the point in time when they prevent, detect, and respond to DDoS flooding attacks. Finally, we highlight the need for a comprehensive distributed and collaborative defense solution against DDoS flooding attacks by enumerating some of the important advantages of distributed DDoS defense mechanisms over centralized ones. Section 2.4 concludes this chapter and provides some insights for implementing a comprehensive distributed collaborative defense mechanism against DDoS flooding attacks. A more comprehensive and detailed version of this literature review is published in [2]. Our main intention for this survey was to help in understanding the scope of the DDoS attack problem which in turn could spur our goal in this thesis which is to monitor the traffic at the intermediate network level in such a way that, with the monitored traffic data, it is possible to detect and stop DDoS flooding attacks at the intermediate networks.

2.1 DDoS problem definition & past incidents

Denial of Service (DoS) attacks, which are intended attempts to stop legitimate users from accessing a specific network resource, have been known to the research community since the early 1980s. In the summer of 1999, the Computer Incident Advisory Capability (CIAC) reported the first Distributed DoS (DDoS) attack incident [9] and most of the DoS attacks since then have been distributed in nature. Currently, there are two main methods to launch DDoS attacks in the Internet. The first method is for the attacker to send some malformed packets to the victim to confuse a protocol or an application on a compromised machine (i.e., vulnerability attack [3]). The other method, which is the most common method, to perform a DDoS attack is for the attacker to either:

(i) disrupt legitimate user's connectivity by exhausting bandwidth, router processing capacity or network resources (i.e., network/transport-level flooding attacks [3]) or

(ii) disrupt legitimate user's services by exhausting the server resources (e.g., Sockets, CPU, memory, disk/database bandwidth, and I/O bandwidth) (i.e., application-level flooding attacks [10]).

Today, DDoS attacks are often launched by a network of remotely controlled, well organized, and widely scattered Zombies[1] or Botnet computers that simultaneously and continuously send a large amount of traffic and/or requests to the target system. The target system either responds so slowly as to be unusable or crashes completely [3, 11]. Zombies or botnet computers are usually recruited through the use of worms, Trojan horses or backdoors [1, 12, 13]. Employing the resources of recruited computers to perform DDoS attacks allows attackers to launch a much larger and more disruptive attack. Furthermore, it becomes more complicated for the defense mechanisms to recognize the original attacker due to the use of counterfeit (i.e., spoofed) IP addresses by zombies under the control of the attacker [14].

[1]Those devices (e.g., computers, routers, etc.) controlled by attackers are called zombies or bots which derives from the word "robot." The term bots is commonly referred to software applications running as an automated task over the Internet (Wikipedia, "Internet bot")

Many DDoS flooding attacks had been launched against different organizations since the summer of 1999 [9]. Most of the DDoS flooding attacks launched to date made the victims' services unavailable, leading to revenue losses and increased costs of mitigating the attacks and restoring the services. For instance, in February 2000, Yahoo! experienced one of the first major DDoS flooding attacks that kept the company's services off the Internet for about 2 hours and resulted in a significant loss in advertising revenue [15]. In October 2002, 9 of the 13 root servers[2] that provide the Domain Name System (DNS) service to Internet users around the world shut down for an hour due to a DDoS flooding attack [16]. Another major DDoS flooding attack occurred in February 2004 that made the SCO Group website inaccessible to legitimate users [17]. This attack was launched by using systems that had previously been infected by the Mydoom [17] virus. The virus contained code that instructed thousands of infected computers to access SCO's website at the same time. The Mydoom virus code was re-used to launch DDoS flooding attacks against major government news media and financial websites in South Korea and the United States in July 2009 [18, 19]. One of the recent DDoS flooding incidents occurred on December 2010; a group calling themselves "Anonymous" launched orchestrated DDoS flooding attacks on organizations such as Mastercard.com, PayPal, Visa.com and PostFinance [20]. The attack brought down the Mastercard, PostFinance, and Visa websites.

Recent advances in DDoS defense mechanisms put an end to the era during which script-kiddies could download a tool and launch an attack against almost any website. In today's environments, attackers use more complicated methods to launch a DDoS attack. Despite all of the efforts towards decreasing the number of DDoS attack incidents, they have expanded rapidly in the frequency and the size of the targeted networks and computers. In a recent survey which was commissioned by VeriSign, it was found that 75% of respondents had experienced one or more attacks during July 2008-July 2009 [22]. Furthermore, Arbor Networks[3] recently published a paper that also reports similar results. In their results, they showed that 69% of the respondents had experienced at least one DDoS attack from October 2009 through September 2010, and 25% had been hit by ten such attacks per month [23]. Accord-

[2]DNS root servers translate logical addresses such as www.google.com into a corresponding physical IP address, so that users can connect to websites through more easily remembered names rather than numbers.

[3]Arbor networks include 111 IP network operators worldwide.

ing to Prolexic Technologies, which offers services to protect against DDoS attacks, there are 7000 DDoS attacks observed daily and they believe this number is growing rapidly [24]. DDoS attacks are also increasing in size, making them harder to defend against. Arbor Networks found that there has been around 100% increase in the attack size over 2010, with attacks breaking the 100Gbps barrier for the first time [23]. Therefore, protecting resources from these frequent and large DDoS attacks necessitates the research community to focus on developing a comprehensive DDoS defense mechanism that can appropriately respond to DDoS attacks before, during and after an actual attack.

2.2 DDoS attacks: scope and classification

The distributed nature of DDoS attacks makes them extremely difficult to combat or trace-back. Attackers normally use spoofed (fake) IP addresses in order to hide their true identity, which makes the traceback of DDoS attacks even more difficult. Furthermore, there are security vulnerabilities in many Internet hosts that intruders can exploit. Moreover, these days, attackers are targeting the application layer by employing techniques that are very difficult to detect and mitigate. One of the necessary steps towards deploying a comprehensive DDoS defense mechanism is to understand all the aspects of DDoS attacks. Various classifications of DDoS attacks have been proposed in the literature over the past decade [3, 4, 9, 25–27, 29]. In this survey, we are interested in providing a classification of DDoS flooding attacks based on the attacked protocol level. We review various DDoS flooding incidents of each category, some of which have been well reviewed/analyzed in [3, 4, 9, 25–27, 29] and the rest are recent trends of DDoS flooding attacks. In this thesis, we mainly focus on DDoS flooding attacks as it is one of the most common forms of DDoS attacks. Vulnerability attacks, in which attackers exploit some vulnerabilities or implementation bugs in the software implementation of a service to bring that down, are not the focus of this thesis.

As we mentioned earlier, DDoS flooding attacks can be classified into two categories as follows based on the attacked protocol:

1. Network/transport-level DDoS flooding attacks
2. Application-level DDoS flooding attacks

2.2.1 Network/transport-level DDoS flooding attacks

These attacks have been mostly launched using TCP, UDP, ICMP and DNS protocol packets. There are four types of attacks in this category [3, 29]:

2.2.1.1 Flood attacks Attackers focus on disrupting legitimate user's connectivity by exhausting victim network's bandwidth (e.g., Spoofed/non-spoofed UDP flood, ICMP flood, DNS flood, VoIP Flood and etc. [25, 28]).

2.2.1.2 Protocol exploitation flood attacks Attackers exploit specific features or implementation bugs of some of the victim's protocols in order to consume excess amounts of the victim's resources (e.g., TCP SYN flood, TCP SYN-ACK flood, ACK & PUSH ACK flood, RST/FIN flood and etc. [25, 28]).

2.2.1.3 Reflection-based flood attacks Attackers usually send forged requests (e.g., ICMP echo request) instead of direct requests to the reflectors; hence, those reflectors send their replies to the victim and exhaust victim's resources (e.g., Smurf and Fraggle attacks) [25, 29].

2.2.1.4 Amplification-based flood attacks Attackers exploit services to generate large or multiple messages for each message they receive to amplify the traffic towards the victim. Botnets have been constantly used for both reflection and amplification purposes. Reflection and amplification techniques are usually employed in tandem as in the case of Smurf attack where the attackers send requests with spoofed source IP addresses (Reflection)

to a large number of reflectors by exploiting IP broadcast feature of the packets (Amplification) [25, 29].

All of the above attack types with the detailed explanation of their incidents are presented in detail in [3,25,28,29]; hence, we skip further explanation of these attacks; instead we focus on the application-level DDoS flooding attacks as they are expected to be the future trend of DDoS flooding attacks since they are stealthier than network/transport-level flooding attacks and they masquerade as flash crowds.

2.2.2 Application-level DDoS flooding attacks

These attacks focus on disrupting legitimate user's services by exhausting the server resources (e.g., Sockets, CPU, memory, disk/database bandwidth, and I/O bandwidth) [10]. Application-level DDoS attacks generally consume less bandwidth and are stealthier in nature when compared to volumetric attacks since they are very similar to benign traffic. However, application-level DDoS flooding attacks usually have the same impact to the services since they target specific characteristics of applications such as HTTP, DNS, or Session Initiation Protocol (SIP). Here we briefly describe the DNS amplification flooding attack and the SIP flooding attack as they are two of the famous application-level reflection/amplification flooding attacks. Then we classify various flavors of application-level flooding attacks that employ the HTTP protocol and we closely look at each of them; since these attacks are consistently reported as the major types of recent DDoS flooding attacks [31].

2.2.2.1 Reflection/amplification based flood attacks [3, 29] These attacks use the same techniques as their network/transport-level peers (i.e., sending forged application-level protocol requests to the large number of reflectors). For instance, the DNS amplification attack employs both reflection and amplification techniques. The attackers (zombies) generate small DNS queries with forged source IP addresses which can generate a large volume of network traffic since DNS response messages may be substantially larger than DNS query messages. Then this large volume of network traffic is directed to a victim system to paralyse it. Another application-level attack example that employs reflection technique is *VoIP*

flood [28]. This attack is a variation of an application specific UDP flood. Attackers usually send spoofed VoIP packets through SIP at a very high packet rate and with a very large source IP range. The victim VoIP server has to distinguish the proper VoIP connections from the forged ones which consumes the resources drastically. VoIP floods can overwhelm a network with packets with randomized or fixed source IP addresses. If the source IP address has not been changed the VoIP flood attack mimics traffic from large VoIP servers and can be very difficult to identify since it resembles good traffic.

2.2.2.2 HTTP flooding attacks [10, 28, 30, 32] There are four types of attacks in this category:

A. Session flooding attacks: In this type of attack, session connection request rates from the attackers are higher than the requests from the legitimate users; hence, this exhausts the server resources and leads to DDoS flooding attack on the server process. One of the famous attacks in this category is the HTTP get/post flood attack (a.k.a., Excessive VERB) [28] in which attackers generate a large number of valid HTTP requests (get/post) to a victim web server. Attackers usually employ botnets to launch these attacks. Since each of the bots can generate a large number of valid requests (usually more than 10 requests a second) there is no need for a large number of bots to launch a successful attack. HTTP get/post flood attacks are non-spoofed attacks.

B. Request flooding attacks: In this type of attack, attackers send to sessions more number of requests than usual and leads to a DDoS flooding attack on the server process. One of the well-known attacks in this category is the single-session HTTP get/post flood (a.k.a., Excessive VERB single session) [28]. This attack is a variation of HTTP get/post flood attack which employs the feature of HTTP 1.1 to allow multiple requests within a single HTTP session. Hence, the attacker can limit the session rate of an HTTP attack and bypass session rate limitation defenses of many security systems.

C. Asymmetric attacks: In this type of attack, attackers send to sessions high-workload requests. Here we enumerate some of the famous attacks in this category.

C.1. Multiple HTTP get/post flood (a.k.a., Multiple VERB Single Request) [28]: This attack is also a variation of HTTP get/post flood attack. A attacker creates multiple HTTP requests by forming a single packet embedded with multiple requests and without issuing them one after another during a single HTTP session [28]. This way attacker can still maintain high loads on the victim server with a low attack packet rate which makes the attacker nearly invisible to netflow anomaly detection techniques. Also, attackers can easily bypass deep packet inspection techniques if they carefully select the HTTP VERB.

C.2. Faulty Application [28]: In this attack, attackers take advantage of websites with poor designs or improper integration with databases. For instance, they can employ SQL-like injections to generate requests to lock up database queries. These attacks are highly specific and effective because they consume server resources (memory, CPU, etc.).

D. Slow request/response attacks: In this type of attack, attackers send to sessions number of high-workload requests. There are number of famous attacks in this category that we describe in the following.

D.1. Slowloris attack (a.k.a, Slow headers attack) [33]: Slowloris is a HTTP get-based attack that can bring down a Web server using a limited number of machines or even a single machine. The attacker sends partial HTTP requests (not a complete set of request headers [34]) that proliferate endlessly, update slowly, and never close. The attack continues until all available sockets are taken up by these requests and the Web server becomes inaccessible. In this attack, usually attackers use their genuine IP addresses and not the spoofed ones.

D.2. HTTP fragmentation attack [28]: Similar to Slowloris, the goal of this attack is to bring down a Web server by holding up the HTTP connections for a long time without raising any alarms. Attackers (bots) (non-spoofed) establish a valid HTTP connection with a web server. Then they fragment legitimate HTTP packets into tiny fragments and send each fragment as slow as the server time out allows. Using this approach, by opening multiple sessions on each bot, the attacker can silently bring down a Web server with just a handful of bots.

D.3. Slowpost attack (a.k.a, Slow request bodies or R-U-Dead-Yet (RUDY) attack) [35]: Wong and Brennan presented a very similar attack to Slowloris that send HTTP post commands slowly to bring down Web servers. The attacker sends a complete HTTP header

that defines the "content-length" field of the post message body as it sends this request for benign traffic. Then it sends the data to fill the message body at a rate of one byte every two minutes. This way the server waits for each message body to be completed and meanwhile Slowpost attack proliferates and causes the DDoS on the Web server.

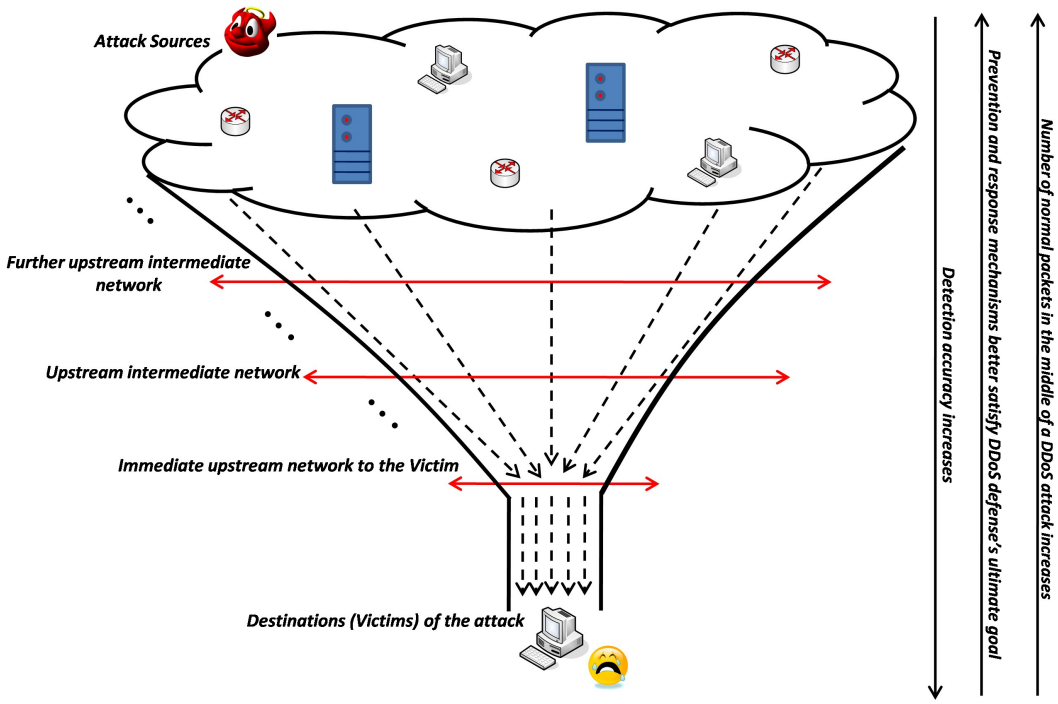


Figure 2.1: Different locations for performing DDoS detection and response.

D.4. Slowreading attack (a.k.a, Slow response attack) [36]: Shekyan presented another type of attack in this category which works by slowly reading the response instead of slowly sending the requests. This attack achieves its purpose by setting a smaller receive window-size than the target server's send buffer. The TCP protocol maintains open connections even if there is no data communication; hence, the attacker can force the server to keep a large number of connections open and eventually DDoS the server.

The message here is that DDoS, like most malicious security threats, is multidimensional. One must be prepared to detect and counter both the more well-known attacks that aggressively assault systems and the ingenious creations that will slip in and undermine systems before you know what hit them.

2.3 DDoS defense: scope and classification

Usually by the time a DDoS flooding attack is detected, there is nothing that could be done except to disconnect the victim from the network and manually fix the problem. DDoS flooding attacks waste a lot of resources (e.g., processing time, space, etc.) on the paths that lead to the targeted machine; hence, it is the ultimate goal for every DDoS defense mechanism to detect them as soon as possible and stop such attacks as near as possible to their sources. Figure 2.1 shows that detection and response can be performed in different places on the paths between the victim and the sources of the attack. As depicted in the diagram, a DDoS flooding attack resembles a funnel in which attack flows are generated in a dispersed area (i.e., sources), forming the top of the funnel. The victim, at the narrow end of a funnel, receives all the attack flows generated. Thus, it is not difficult to see that detecting a DDoS flooding attack is relatively easier at the destination (victim), since all the flows can be observed at the destination. On the contrary, it is difficult for an individual source network of the attack to detect the attack unless a large number of attack flows are initiated from that source. Obviously, it is desirable to respond to the attack flows closer to the sources of the attacks, but there is always a trade-off between accuracy of the detection and how close to the source of attack the prevention and response mechanism can stop or respond to the attack.

Moreover, the number of normal packets that reach the victims even when the victims are under a DDoS attack increases when response mechanisms drop the attack packets closer to the sources of the attack. Otherwise, as attack flows reach closer to the victims, packet filtering mechanisms drop more legitimate packets that are destined to the victims[5].

[5]During large scale DDoS attacks, victims or their immediate upstream networks drop all the packets

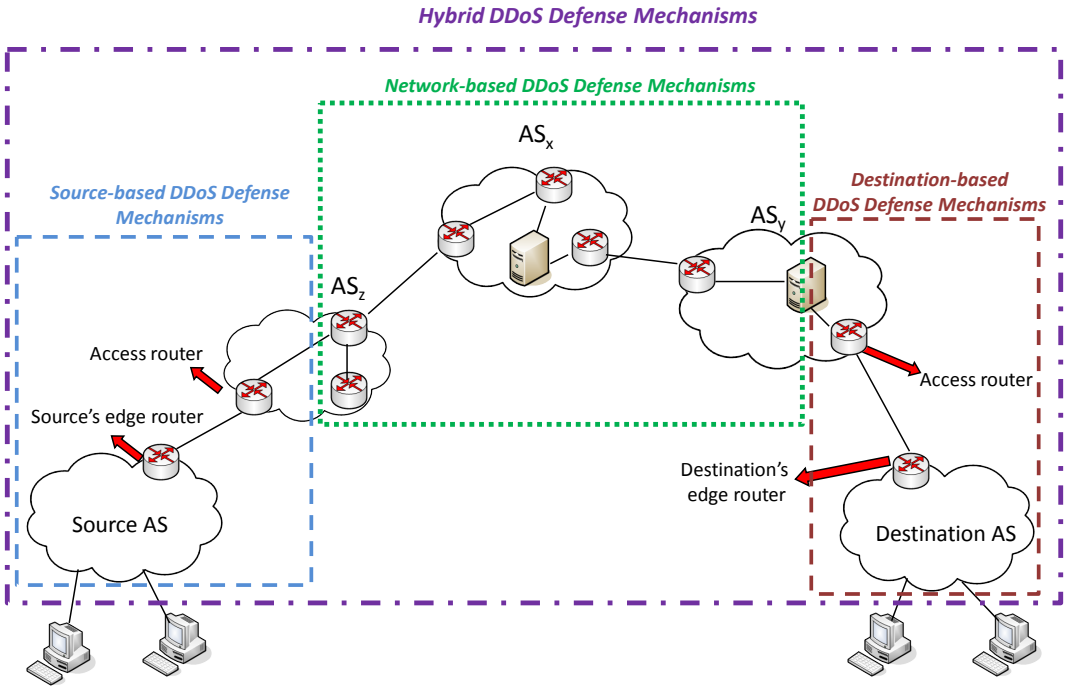


Figure 2.2: A classification of the defense mechanisms against network/transport-level DDoS flooding attacks based on their deployment location in a simple network of ASs.

Several mechanisms to combat DDoS flooding attacks have been proposed to date in the literature [3, 4, 25–27, 29]. In this section we classify the defense mechanisms against two types of DDoS flooding attacks that we presented in section 2.2 using two criteria. We believe that these classification criteria are important in devising robust defense solutions. The first criterion for classification is the location where the defense mechanism is implemented (i.e., Deployment location). We classify the defense mechanisms against network/transport-level DDoS flooding attacks into four categories: *source-based*, *destination-based*, *network-based*, and *hybrid (a.k.a. distributed)* and the defense mechanisms against application-level DDoS flooding attacks into two categories: *destination-based*, and *hybrid (a.k.a. distributed)* based on their deployment location. Figure 2.2 shows the classification of the defense mechanisms against network/transport-level DDoS flooding attacks based on their deployment location in a simple network of AS. There is no network-based defense mechanism against application-level DDoS flooding attacks since the application-level DDoS flooding attack traffic is not

destined to the victims.

accessible at the layer 2 (switches) and layer 3 (routers) devices. Classification of DDoS defense mechanisms based on their deployment location was first presented in [9] and it is used by some other surveys as one of their classification criteria [3,11,14,29]. In this survey, we extend this classification criterion by adding a *hybrid* category and analyzing several recent DDoS defense mechanisms in each category.

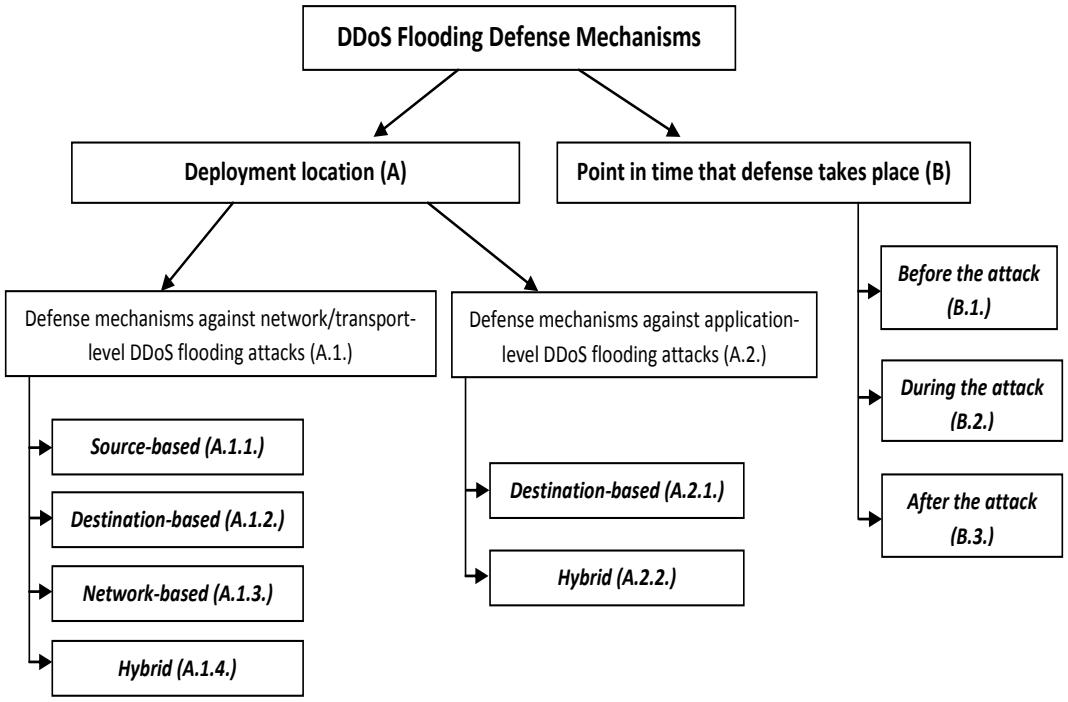


Figure 2.3: A taxonomy of defense mechanisms against DDoS flooding attacks

The second criterion for classification is the point of time when the DDoS defense mechanisms should act in response to a possible DDoS flooding attack. Based on this criterion we classify both defense mechanisms against application-level and network/transport-level DDoS flooding attacks into three categories (i.e., three points of defense against the flooding attack): *before the attack* (attack prevention), *during the attack* (attack detection), and *after the attack* (attack source identification and response) [3]. However, a comprehensive DDoS defense mechanism should include all three defenses since there is no one-size-fits-all solution to the DDoS problem. Our contribution to the last classification criterion is to classify and

enumerate most of the recent defense mechanisms against DDoS flooding attacks into the aforementioned categories. Figure 2.3 shows the above mentioned taxonomy of the defense mechanisms against DDoS flooding attacks.

2.3.1 Classification based on the deployment location

2.3.1.1 Defense mechanisms against network/transport-level DDoS In the following, we extensively discuss the defense mechanisms against network/transport-level DDoS flooding attacks in each of the categories of the first classification criterion.

A. Source-based mechanisms: Source-based mechanisms are deployed near the sources of the attack to prevent network customers from generating DDoS flooding attacks. These mechanisms can take place either at the edge routers of the source's local network or at the access routers of an AS that connects to the sources' edge routers [9]. Various source-based mechanisms have been designed to defend against DDoS flooding attacks at the source; some of the major ones are as follows: Ingress/Egress[6] filtering at the sources' edge routers [48], D-WARD [51, 52], MUlti-Level Tree for Online Packet Statistics (MULTOPS) [53] & Tabulated Online Packet Statistics (TOPS) [54], and MANAnet's Reverse Firewall [55].

Source-based defense mechanisms aim to detect and filter the attack traffic at the sources of the attack; however, they are not entirely effective against DDoS flooding attacks. There are three main reasons which make these mechanisms a poor choice against DDoS flooding attacks. First, the sources of the attacks can be distributed in different domains making it difficult for each of the sources to detect and filter attack flows accurately. Second, it is difficult to differentiate between legitimate and attack traffic near the sources, since the volume of the traffic may not be big enough as the traffic typically aggregates at points closer to the destinations. Finally, the motivation for deployment of the source-based mechanisms is low since it is unclear who (i.e., customers or service providers) would pay the expenses associated with these services. Hence, pure source-based mechanisms are not efficient and effective against DDoS flooding attacks.

[6]It is called either Ingress or Egress depending on where you stand in the network and apply the filters.

B. Destination-based mechanisms: These mechanisms are deployed at the destination of the attack (i.e., victim). In the destination-based defense mechanisms, detection and response is mostly done at the destination. There exist various destination-based mechanisms that can take place either at the edge routers or the access routers of the destinations' AS. The destination-based DDoS defense mechanisms can closely observe the victim, model its behavior and detect any anomalies. Some of the major destination-based DDoS defense mechanisms are as follows: IP Traceback mechanisms [56–61], Management Information Base (MIB) [63], Packet marking and filtering mechanisms [66, 67, 69], and Packet dropping based on the level of congestion(e.g., Packetscore [70]).

Most of the destination-based mechanisms cannot accurately detect and respond to the attack before it reaches the victims and wastes resources on the paths to the victims; hence, they are not capable of detecting and responding to the DDoS attack traffic properly. Therefore, network-based DDoS defense mechanisms have been proposed to address this problem and to help both source and destination based mechanisms to carry out their duties more accurately.

C. Network-based mechanisms: These mechanisms are deployed inside networks and mainly on the routers of the ASs [71]. Detecting attack traffic and creating a proper response to stop it at intermediate networks is an ideal goal of this category of defense mechanisms. Some of the main network-based DDoS defense mechanisms are as follows: Route-based packet filtering [73, 148], and Detecting & filtering malicious routers [74] (e.g., Watchers [75]).

Network-based mechanisms usually lead to high storage and processing overhead at the routers. These overheads get even worse if each router does redundant detection and response through the path to the destination [79], which can present a significant burden. Various researchers have proposed different approaches to reduce the amount of storage and consumption of CPU cycles for detection and response at the routers such as Bloom filters [71] [80], Packet sampling [81], etc. But these approaches are not sufficient when routers still do redundant jobs. Moreover, reducing the amount of redundant detection and response between the routers requires coordination among them [79]. Different communication protocols have been proposed to coordinate attack detection and response among the routers [9].

However, network-based defense mechanisms that have been proposed thus far are not effective and efficient due to their large overhead of network communication. For instance, the lack of bandwidth during DDoS attacks may limit the protocol for communication and cause network-based mechanisms to fail.

D. Hybrid (Distributed) mechanisms: In most of the previously discussed categories of DDoS flooding defense mechanisms (source-based, destination-based, and network-based), there is no strong cooperation among the deployment points. Furthermore, detection and response is mostly done centrally either by each of the deployment points (e.g., source-based mechanisms) or by some responsible points in the group of deployment points (e.g., network-based mechanisms). Hence, we call these categories of DDoS defense mechanisms *centralized*. As opposed to centralized defense mechanisms, hybrid defense mechanisms are deployed at (or their components are distributed over) multiple locations such as source, destination or intermediate networks and there is usually cooperation among the deployment points. For instance, detection can be done at the victim side and the response can be initiated and distributed to other nodes by the victim. Some of the hybrid DDoS defense mechanisms are as follows: Hybrid packet marking and throttling/filtering mechanisms (e.g., Aggregate-based Congestion Control (ACC) [82], Pushback [82], Attack Diagnosis (AD) and parallel-AD [84], and TRACK [85]), DEFCOM [86], COSSACK [87], Capability-based mechanisms [88–95], Active Internet Traffic Filtering (AITF) as a filter-based (datagram) mechanism [96, 97], and StopIt [94].

In [95, 98], TVA as a capability-based mechanism is compared to StopIt as a filter-based mechanism under similar assumptions and practical constraints. They compare six different DDoS flooding mitigation systems, including TVA and StopIt. They use simulations on realistic topologies and cover different attack strategies in their research. The outcome of their research is summarized in Figure 2.4 [95, 98]. In Figure 2.4, *attacks' power* is a generalized term which is defined in [98] based on: number of attackers (i.e., number of bots), and size of attack (i.e., packet size); As the number of attackers and their packet sizes increase the attack power increases. Effectiveness is also measured [98] based on the legitimate hosts' TCP transfer performance (i.e., percentage of completed TCP transfers). As the number of completed transfers increases the mechanisms that have been employed

are more effective. As *Yang et al.* showed in this figure, when the attackers' power is low, both filters and capabilities work well, although filters work slightly better. As the attackers' power increases, filters become ineffective when they cannot be properly installed, and then capabilities become ineffective when attackers can get capabilities from colluders. When the attackers' power is extremely high, both filters and capabilities become ineffective, and there should be some fail-safe mechanisms (e.g., fair queuing) in place to resolve the problem.

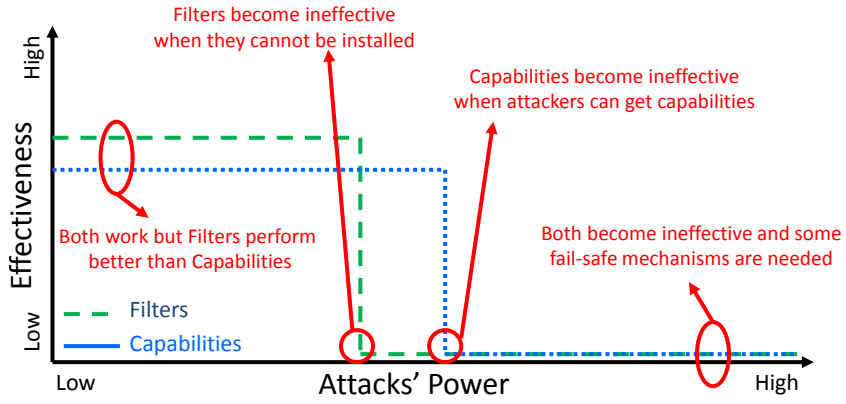


Figure 2.4: Capability-based (capabilities) vs. datagram-based (Filters) mechanisms [95].

Discussion: Since attackers cooperate to perform successful attacks, defenders must also form alliances and collaborate with each other to defeat the DDoS attacks. The DDoS defense community is currently more involved in proposing novel hybrid DDoS defense mechanisms and most of the recently proposed mechanisms belong to the hybrid category. No single deployment point (centralized) can successfully defend against DDoS because of the fundamental challenges we enumerated for each of the deployment points. A hybrid (*Distributed*) defense deployment is the best way to combat DDoS Attacks. We have enumerated various hybrid defense mechanisms in this section; they are comprised of multiple defense nodes deployed at various locations that cooperate with each other towards attack prevention, detection, and response. Detecting DDoS attack as soon as possible and before it reaches the victims, identifying the attack sources, and finally stopping the attack as close as possible to the attack sources is the ultimate goal of DDoS defense mechanisms; we strongly believe that this can be best achieved through hybrid (*Distributed*) DDoS defense mechanisms.

Table 2.1: Summary of features, advantages, and disadvantages of defense mechanisms against network/transport-level DDoS flooding attacks based on their deployment location

		Features	Disadvantages	Advantages
Centralized	***Source-based***	Detection and response are deployed at the source hosts	Sources are distributed among different domains; hence, it is difficult for each of the sources to detect and filter attack flows accurately Difficult to differentiate legitimate and DDoS attack traffic at the sources, since the volume of the traffic is not big enough Low motivation for deployment; since, it is unclear who would pay the expenses associated with these services	Aims to detect and respond (i.e., filter) to the attack traffic at the source and before it wastes lots of resources
	Destination-based	Detection and response are deployed at the destination hosts (i.e., victims)	They cannot accurately detect and respond to the attack before it reaches the victims and wastes resources on the paths to the victim	Easier and cheaper than other mechanisms in detecting DDoS attacks due to access to the aggregate traffic near the destination hosts
	Network-based	Detection and response are deployed at the intermediate networks (i.e., routers)	High storage and processing overhead at the routers Attack detection is difficult because of lack of availability of sufficient aggregated traffic destined for the victims	Aims to detect and respond to (i.e., filter) the attack traffic at the intermediate networks and as close to source as possible
Distributed	***Hybrid (Distributed)***	Detection and response are deployed at various locations: detection usually occurs at destinations & intermediate networks, and response usually occurs at the sources & upstream routers near the sources There is a cooperation among various defense components	Complexity and overhead due to cooperation and communication among distributed components scattered all over the Internet Lack of incentives for the service providers to cooperate/collaborate Trusted communication among various distributed components in order to cooperate/collaborate	Less vulnerable and hence robust against DDoS attacks More resources are available to tackle DDoS attacks

Combining source address authentication (to prevent IP spoofing), capabilities, and filtering would be the most effective and efficient solution due to the robustness of capabilities and the relative simplicity of a capability-based design. However, there will be a trade-off between performance and accuracy in any DDoS defense solution and the goal is to minimize the gap between performance and accuracy. Table 4.1 summarizes the features of the four categories of defense mechanisms against network/transport-level DDoS flooding attacks that we classified in this section and enumerates the advantages and disadvantages of each category.

2.3.1.2 Defense mechanisms against application-level DDoS In the following, we extensively discuss the defense mechanisms against application-level DDoS flooding attacks in each of the categories of the first classification criterion.

A. Destination-based (server-side) mechanisms: Most of the application layer protocols are organized in terms of *client-server* model. A *server* is a process that implements a specific service (e.g., DNS server, Web server). A *client* is a process that requests a service from a server. As we mentioned earlier, destination-based defense mechanisms are deployed at the destination of the attack (i.e., victim), which is the *server* of the application layer protocols' client-server model or the reverse proxy[7] when we consider a web cluster hosting different web applications. Most of these mechanisms closely observe the server and model its clients' behavior so that they can detect any anomalies and drop/rate limit the malicious requests. Some of these major mechanisms against application-level DDoS flooding attacks are as follows: Defense against Reflection/Amplification attacks (e.g., DAAD [99], VFDS [100]), DDoS-Shield [10, 101], Anomaly detector based on hidden semi-Markov model [102], and DAT [103].

B. Hybrid (Distributed) mechanisms: Hybrid defense mechanisms are those mechanisms that there is a collaboration/cooperation between both clients and servers in the application layer protocols' *client-server* model to detect and respond to the attacks. For instance, detection is done at the victim (web server/reverse proxy) and the response is

[7]The reverse proxy is a type of proxy server that retrieves resources on behalf of a client from one or more servers

initiated and distributed to the client-sides by the victim. Some of the hybrid mechanisms against application-level DDoS flooding attacks are as follows: Speak-up [104], DOW (Defense and Offense Wall) [105], Differentiate DDoS flooding bots from human (e.g. [106], CAPTCHA [107], and [108]), Admission control and congestion control [109], TMH (Trust Management Helmet) [110], and Hybrid detection based on trust and information theory based metrics [111].

Discussion: Detecting and responding (rate limit/blocking) the application-level DDoS flooding attacks at the servers/reverse proxies is not effective enough since attack traffic could already affect the victims. As we pointed out in section *A.1.5*, hybrid defense mechanisms are the best way to combat DDoS flooding attacks since all of the defense nodes collaborate with each other to defeat coordinated DDoS flooding attacks. We have enumerated some of the recent state of the art hybrid defense mechanisms against application-level DDoS flooding attacks in this section and since recent attack incidents proved that current mechanisms have not been fully successful, advanced defense mechanisms with novel features are yet to be deployed. Here, we briefly discuss some of those required features:

(i) Defense mechanisms must be capable of detecting the attacks independent of the attack's exact nature of operation since predicting and detecting all possible attacks by the attackers is hard.

(ii) Enhanced detection mechanisms should be in place to better distinguish between the legitimate and malicious requests. Using metrics such as the request rate, the packet headers, or the contents of the request may not be sufficient enough.

(iii) Response mechanisms should be more adaptive in a sense that legitimate users can claim their fair share of resources. In other words, more request throttling mechanism which assign more server resources to the legitimate clients should be in place than the request blocking mechanisms.

2.3.2 Classification by the point in time (i.e., between the start and end of a DDoS attack) that defense takes place

2.3.2.1 Before the attack (attack prevention) The best point in time to stop a DDoS attack is at its launching stage. In other words, attack prevention is the best DDoS defense solution. The prevention mechanisms can be deployed at the attack sources, intermediate networks, destinations or combination of them. Most of the prevention mechanisms aim to fix security vulnerabilities (e.g., insecure protocols, weak authentication schemes, and vulnerable computer systems) that can be exploited to launch DDoS attacks. Several prevention mechanisms have been proposed in the literature [112]. There are some general prevention mechanisms that should be employed almost everywhere (e.g., servers, hosts, and intermediate networks) and in as many places as possible by both end hosts and service providers. Some of these general prevention mechanisms are as follows:

A. System & Protocol security mechanisms to increase the overall security of the systems: For instance, by preventing illegitimate accesses to the machines, removing bugs, updating installed protocols, installing software patches, removing unused software, etc [3].

B. Fail-safe protections: Anticipation in case something goes wrong (e.g., replication of services and applications in diverse locations in case DDoS attack occurs successfully, business continuity and disaster management plans, etc.).

C. Resource allocation & accounting [3]: Providing resources to counter DDoS attacks and control users' access based on their privileges and behaviors [113–115].

D. Reconfiguration mechanisms: These mechanisms alter the topology of either the victim network to add more resources to tolerate the DDoS attack (e.g., resource replication services [116]) or the intermediate network to isolate the attack sources (e.g., attack isolation strategies) [3].

E. Installing firewalls and improved Intrusion Detection & Prevention Systems (IDPSs): All of the end hosts are encouraged to install IDPSs to prevent them from being compromised by the adversaries.

F. Employing *local filters* (e.g., Ingress/Egress [48], History-based IP filtering [66], hop-count filtering [67], Pi [69], route-based packet filtering [73,148], etc.) and *globally coordinated filters* (e.g., ACC [82], Pushback [82], [83], AD and parallel-AD [84], TRACK [85], etc.) to block attack flows before their bombardment is another important category of the prevention mechanisms against DDoS attacks.

G. Load balancing [112] *and Flow control* are two other mechanisms to prevent DDoS attacks. The former improves both the performance and mitigation against DDoS attack, and the latter prevents servers from going down.

H. Server-side specific security considerations: One of the main problems regarding application-level flooding attacks is that there is a lack of security mechanisms or security policies in place to address the servers vulnerabilities against application-level DDoS flooding attacks. Such security mechanisms or policies can protect servers from various attacks. For instance, *Shekyan* [36] suggested the following policies as the best protections for the servers on handling the write readiness for active sockets:

- Do not accept connections with abnormally small advertised window sizes.
- Do not enable persistent connections and HTTP pipe-lining unless performance really benefits from it
- Limit the absolute connection lifetime to some reasonable value

As another example, disabling open recursion[8] on name servers from external sources and only accepting recursive DNS originating from trusted sources has been proposed as an effective prevention mechanism to diminish amplification vector of DNS amplification attacks [117]. Similar security mechanisms or security policies for different servers such as: Web servers, application servers, database servers, etc. should be defined and should be in place by considering current vulnerabilities of the servers against various application-level DDoS flooding attacks.

I. Finally, service providers can have *strategies* in place to better identify their legitimate users. For instance, they can put dynamic pricing to network resource usages and charge their customers differently for the use of different resources [27]. Another effective service

[8]Name servers on the Internet that have recursion enabled and provide recursive DNS responses to anyone (a.k.a. open resolvers).

provider strategy was recently employed by Cisco in their IPS 7.0 code upgrade [47] . IPS 7.0 upgrade has *global correlation* feature that can be configured on every service provider IPS sensors so that they are aware of the network devices with a reputation for malicious activity, and can take action against them. This feature is useful when service providers' network is under attack from a botnet DDoS attack since sensors can drop all the traffic coming from bad reputation sources. Furthermore, this whole process is very inexpensive since it occurs before the signatures are used.

Prevention mechanisms aim to provide systems with increased security. However, these mechanisms can never completely remove the threat of DDoS attacks since they are always vulnerable to novel attacks for which signatures and patches are not available.

2.3.2.2 During the attack (attack detection) The next step in defending against DDoS attacks is *attack detection*, which happens during the attack. The detection mechanisms can be also deployed at sources, intermediate networks, destinations or combinations of them.

There are various mechanisms to detect DDoS attacks. Some of the detection mechanisms detect attack flows when the network links are congested to a certain level [70] [118]. Other mechanisms detect DDoS flooding attack traffic (*not vulnerability attacks*[9]) when anomalous patterns are discovered in both the network/transport-level traffic or application-level traffic (e.g., MIB *information analysis* [63], D-WARD [51, 52], MULTOPS [53], TOPS [54], [121–123], DDoS-Shield [10, 101], DAT [103]. There are many IDPSs that are based on these detection mechanisms. They employ data mining and artificial intelligence mechanisms for more accurate detection. These mechanisms monitor some features/headers of the traffic flows at various locations and points in time. Basically, they learn the normal behavior of either the network/transport-level or application-level traffic. Then, based on the information they monitored and collected they could detect any changes on the traffic patterns and usage patterns of the resources. Based on the analysis in [119], anomaly detection algorithms to detect a DDoS flooding attacks could be classified depending on either monitored parameters [120] or statistical technique used (e.g., change point detection [121], wavelet

[9]Vulnerability attacks are mostly detected by employing databases of known signatures.

analysis [122]) or granularity level of the analysis [123].

As we discussed earlier, the most practical place to detect DDoS flooding attack is at the victims' side since abnormal deviations cannot be easily found until the attack turns to its final stage. Even after the attack is detected, it is difficult for the victims to launch an efficient response mechanism due to numerous malicious packets that have been aggregated at the victims' side. Therefore, defending DDoS flooding attacks should be initiated at earlier points in time and as near as possible to the sources of the attacks. Detecting (defending) at either intermediate networks or sources of the attacks have two main advantages: (1) the detection is more concealed since it is deployed in a separate location from attack target and (2) the detection mechanism is less vulnerable to DDoS attacks. However, accurate detection is not easy or it is even impossible to achieve since there is not enough evidence to detect attacks at these stages (e.g., source and upstream routers). Two fundamental challenges to detect DDoS flooding attacks in time and as near as possible to the attack sources are: (1) the lack of a wide deployment of DDoS defense mechanisms at different points of the Internet, and (2) the lack of collaboration and cooperation among distributed deployed defense mechanisms in order to increase the detection accuracy, decrease unnecessary redundant tasks (due to lack of coordination), and, finally, to increase the performance efficiency of DDoS defense mechanisms. In case of application-level DDoS flooding attacks, all of the current detection mechanisms are deployed at the destination (servers) since it is not possible to perform detection at the layer 2/layer 3 intermediate networks. However, it will be possible to stop application-level DDoS flooding attacks at the intermediate networks if some layer 2/layer 3 extractable features of these attacks are found through studying these attacks and their in-depth architecture.

2.3.2.3 After the attack (attack source identification and response After a DDoS attack is detected, the defense system should identify the source of the attack and block the attack traffic. Today, most of the DDoS response mechanisms cannot completely prevent or stop DDoS attacks. Therefore, minimizing the attack impact and maximizing the availability of services is the main focus of all after the attack mechanisms. Moreover, law enforcement agencies must collaborate and cooperate with each other in order to gather and submit

evidences that could be used to prosecute attackers. It is necessary for all the Internet providers to understand that even if a particular provider could be able to secure its own assets, it does not secure itself against DDoS attacks as other compromised hosts of other providers could still be used to launch attacks on it. Therefore, without collaborating with others to make sure their assets are also secured, defending against DDoS attack is almost ineffective.

There are two main categories for most of the after the attack mechanisms:

A. Attack source identification: The first category of after the attack mechanisms is responsible to identify the source of the attack. For instance, an attacker uses host X to launch an attack by representing the spoofed source address of host Y, IP traceback mechanism must find out the real source address of the attacker which is host X. This can be accomplished if there is a way of traversing all the routers from X to the victim in the reverse order or marking the legitimate paths or packets so that spoofed or illegitimate ones are identifiable. Towards this, traceback mechanisms [56–61] have been proposed in the literature.

B. Initiating a proper response: The second category of after the attack mechanisms is responsible to initiate a proper response to the attack. Most of the DDoS defense mechanisms apply throttling (rate limit) or packet filtering on upstream routers and hosts for the traffic coming from those *identified attack flows* (e.g., spoofed IP addresses) after identifying the source of the attack. For instance, history-based IP filtering [66], hop-count filtering [67], Pi [69], AD [84], TRACK [85], and StopIt [94, 98] employ packet filtering upon detecting DDoS attacks and ACC [82], Pushback [82], [83], PAD [84], AITF [96], and DEFCOM [86] employ throttling upon detecting DDoS attacks. Other mechanisms specially in the case of application-level DDoS flooding attacks employ some encouragement models in which servers are asking the legitimate clients to increase their session rates to crowd out the malicious clients (e.g., Speak-up [104], and DOW [105]).

2.4 Summary

In this chapter, we presented a comprehensive classification of various DDoS flooding attacks and defense mechanisms along with the advantages and disadvantages of these defense mechanisms based on where and when they detect and respond to DDoS flooding attacks. An ideal comprehensive DDoS defense mechanism must have specific features to combat DDoS flooding attacks both in real-time and as close as possible to the attack sources. These features are as follows:

1. *More nodes in the Internet should be involved in preventing, detecting, and responding to DDoS flooding attacks (i.e., Hybrid (Distributed) defense).* As we discussed earlier, the detection accuracy is high at the victim end but it is not robust; victims cannot tolerate the high volume of DDoS traffic. Stopping the attacks at the source could be the best response option but it is very difficult as the volume of the traffic at the sources is not significant to differentiate between legitimate and malicious traffic. Furthermore, the collateral damage is high at intermediate networks because there is not enough memory and CPU cycles to profile the traffic. Therefore, central mechanisms in which all the defense components (i.e., prevention, detection, and response) are deployed at the same place, are not practical against DDoS flooding attacks.

2. *There should be collaboration and cooperation among the key defensive points within and between service providers in the Internet.* The main challenge towards achieving this goal is the need for some economic incentives among different service providers in order to achieve highly cooperative defense mechanisms.

3. *More reliable mechanisms are required to authenticate the source of the Internet traffic so that malicious users could be identified and held accountable for their activities (i.e., Anti-spoofing mechanisms).*

4. *Trusted communication mechanisms for cooperation and collaboration among various distributed components are needed.* For instance, in the pushback mechanism, rate limit requests to the upstream routers could be sent by a malicious point in the network.

We strongly believe that combining source address authentication, capability mechanisms, and filtering mechanisms could be the most effective and efficient way to address the DDoS flooding attacks in a distributed cooperative/collaborative DDoS defense mechanism. More development and deployment of distributed defense mechanisms from researchers and Internet service providers respectively is what we expect to see in the near future (short to medium term). In a longer term we expect to see:

- The cooperation and collaboration among Internet service providers to monitor the traffic, detect, and stop the DDoS flooding attacks close to their sources, especially with the rapid growth of collaborative environments such as Cloud Computing [124] and the Internet of Things (IoT) [125–127].
- Cross layer traffic analysis and defense (i.e., looking at the information at multiple protocol layers simultaneously to detect and respond to the attacks)

3.0 Traffic Monitoring Mechanisms: Current Practice & Challenges

In order to better understand the contributions of this thesis, in this chapter, we first review and compare two different methods of traffic monitoring in Section 3.1. Then, in Section 3.2, we review some possible mechanisms currently available to network operators to enforce future traffic monitoring policy requirements for different network management applications and discuss their challenges and recent trends of proposed solutions to address those challenges. Next, in Section 3.3, we review one of the recently proposed traffic monitoring mechanisms that is closely related to our proposed traffic monitoring mechanism in terms of policy enforcement approach. Finally, in Section 3.4, we summarize the required features of the traffic monitoring mechnaisms suitable for the next generation of distributed DDoS flooding defence mechanisms. Hereby, we motivate and briefly introduce our proposed traffic monitoring mechanism.

3.1 Packet sampling vs. Flow sampling (a.k.a. Flow monitoring)

Today, most of the router vendors use packet sampling with different sampling rates (e.g., Deterministic sampling, random sampling, time-based sampling [129] [128]). This way, routers select a subset of packets and aggregate the sampled packets into flow reports at the end of each monitoring window. The packet sampling mechanisms have their inherent limitations. For instance, there are known biases toward sampling larger flows (e.g., [130–132]) and several studies have questioned its faithfulness for many management applications [130–132]. Moreover, some new flows will not be covered when the memory sizes (caches) of the monitoring devices are full. Also, generated flow reports at the end of the monitoring time window

do not necessarily correspond to the order in which the flow traffic arrived at the monitoring devices [133].

More recently, there have been tremendous efforts towards addressing the limitations of the packet sampling approaches (i.e., their biases toward sampling larger flows and less coverage for small flows) by proposing yet another trend of traffic monitoring approaches namely flow sampling [134–136]. In flow sampling approaches, flows are picked randomly instead of packets. In other words, for each packet, router checks if it is responsible for tracking this packet's *flowkey*, defined over one or more fields of the 5-tuples of the IP header. If yes, the router updates the appropriate counter (statistics) for that flow. If not, the flowkey for this packet is selected with probability p, and the router keeps an exact count for this selected *flowkey* subsequently. This needs per-packet counter updates and because of that the counters are kept in SRAM. The router responsibilities on what flows to monitor is done randomly or deterministically through a flow selection process. The probability p for each *flowkey* in router X can be calculated as follows [136]:

$$p = \frac{NumFlows_{X}}{NumPKTs_{X}} \tag{3.1}$$

in which, $NumFlows_{X}$ is the number of flows that are possible to monitor on router X considering the router's memory constraint. $NumPKTs_{X}$ is the total number of anticipated packets on router X for a specific time interval.

Figure 3.1 depicts how flow sampling and packet sampling methods monitor the traffic for nine flows with different flow rates (in pkts/sec) in a fixed monitoring time-window (resulted in total of 30 packets). In case of random packet sampling, the packet sampling rate is equal to 1 in 5 packets, and in case of random flow sampling, the flow sampling rate of all the flows is equal to 1 in 5 flows. Figure 3.1 shows how flow sampling methods eliminate the biases toward sampling larger flows that all the packet sampling methods suffer from. Moreover, Figure 3.1 shows that with flow sampling it is possible to increase the coverage of total number of flows that their packets are getting sampled which is an important feature for some of the traffic monitoring applications such as: monitoring the patterns of traffic anomalies or DDoS flooding attack flow coverages [137]. However, monitoring devices are constrained by their resources (e.g., memory and CPU) and in large traffic volume cases

some of the traffic flows may not be monitored by the monitoring mechanisms and only a selection of flows will be monitored during each monitoring time window. This selection of flows is most of the time done in a completely random fashion. One of the main challenges of some of the traffic monitoring mechanisms is if it is possible to decide and assure the monitoring of specific flows. In other words, the prioritization of the monitoring coverage for specific flows is very critical/desirable.

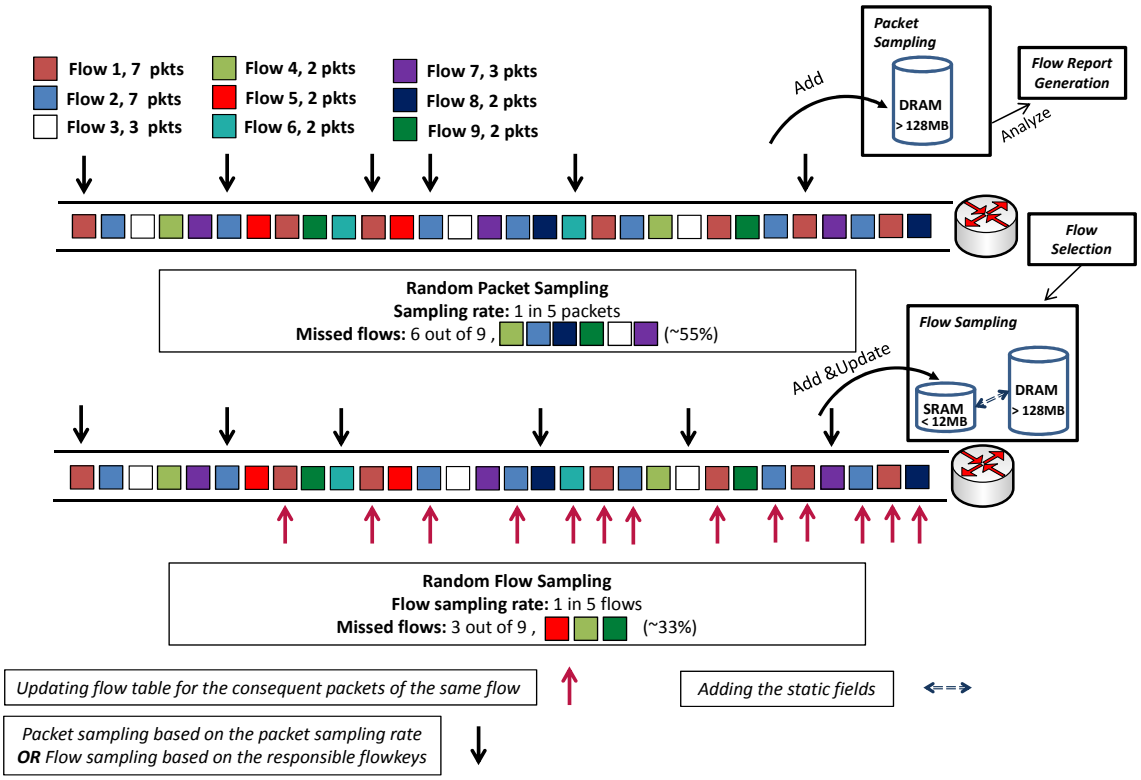


Figure 3.1: Flow sampling vs. Packet sampling

3.1.1 Discussion: Prioritized flow monitoring

Recently, researchers have attempted to prioritize the flow coverage for some of the traffic monitoring applications [136, 138]. For instance, cSamp [136] assigns monitoring responsibilities in such a way that the assignment ensures the minimum coverage for all the flows as a

high-level monitoring objective/policy. We argue that for the distributed detection components of any future distributed DDoS defense mechanism to distributively detect attack flows at the intermediate network level, it is important to ensure that all the flows destined for a given destination (i.e., possible DDoS flooding attacks) are monitored all together in one place so that the distributed detection mechanism in place can analyse these flows. Hence, guaranteeing the monitoring coverage for those flows (as a priority) should be part of a high-level traffic monitoring objective/policy for the traffic monitoring mechanisms employed by the next generation of DDoS flooding defense mechanisms.

3.2 Traffic monitoring as a network management task

Nowadays, addressing the challenging network management tasks such as: traffic monitoring, secure communication, and performance optimization and their high-level policy goals is very crucial for data-centers, enterprise networks, and even individual ISPs. For instance, network operators may want either very good monitoring coverage to capture end-to-end traffic patterns' anomaly, or effective configurations to provide the best end-to-end performance for their customers.

As computer networks get more and more complex and dynamic, and as traffic patterns evolve, network management tasks such as traffic monitoring become increasingly challenging. Because, such changes in computer networks and traffic patterns may require new functionalities and high-level policies that network operators are responsible for and struggle with enforcing them. Enforcing network operators' high-level traffic monitoring policies/objectives (e.g., maximizing total flow coverage, minimum coverage for specific flows, load balancing, *etc.*) is possible through:

1. *Device-centric approaches*: Each monitoring device (e.g., routers) is configured to sample flows/packets (employing packet/flow monitoring mechanisms [128]) individually and independently of other monitoring devices to achieve the network operators' requested monitoring policies/objectives. There are different device-centric approaches for the network operators to configure monitoring devices in order to enforce their high-level policies/objectives that we will enumerate next. However, since in all of the device-centric approaches there is no coordination among the monitoring devices as what their monitoring responsibilities are (e.g., what flows to monitor), some of the monitoring devices may perform the same monitoring task (e.g., the flows/packets are monitored redundantly) which causes a huge overhead and leads to a tremendous decrease in the monitoring coverage. Moreover, there is a tremendous effort required by the network operators to configure each of the monitoring devices to enforce their monitoring policies/objectives.

2. *Network-wide approaches*: Recent trends in network management have inspired network-wide monitoring [136, 139–142] as opposed to device-centric monitoring in which, all the monitoring devices are managed and coordinated to achieve various high-level network-wide traffic monitoring policies/objectives. Moreover, other recent network management proposals have shown that the centrally managed network-wide traffic monitoring mechanisms could reduce both the operational costs and the management complexity of such network management tasks [136, 143–145]. Network-wide approaches address the aforementioned shortcomings of the device-centric approaches (e.g., redundant monitoring tasks, and per-device configuration to achieve high-level network-wide objectives). For instance, coordination among the monitoring devices is achieved by means of communication among the monitoring devices [146] (i.e., monitoring devices send an image of what they are monitoring to each other); however, this would lead to large communication overhead. As another example, Bloom filters have been employed to reduce the redundant measurements or there is a trend of related work focused on the placement of monitoring devices at appropriate locations to cover the paths while using minimum monitoring devices [147–149].

Next, we briefly review various device-centric and network-wide approaches in the literature.

3.2.1 Device-centric approaches

Device-centric approaches can be categorized into four categories (The functionality and cost of deployment increases from the first category to the last):

3.2.1.1 Existing router primitives: In this category, network operators use techniques that work with existing router primitives (i.e., built-in support) provided by different router vendors (e.g., Cisco, Juniper) to monitor the traffic. Moreover, ISPs and enterprise networks also develop various configuration and analysis tools to help in achieving the required functionalities out of the existing router primitives. For instance, there are different proposed techniques to infer interesting traffic activity patterns from existing measurement feeds [140, 150]. Techniques in this category are the easiest and cheapest to develop, and they are the most commonly used since no additional support is required from the network elements. However, the quality and quantity of the monitored data is bounded by the available router primitives.

3.2.1.2 Additional middleboxes: When there are new capabilities required by various new monitoring applications that are not provided by the existing network elements, network operators can deploy middleboxes that are mostly developed by third party vendors. However, these temporary solutions themselves can soon become inapplicable due to possible new traffic monitoring applications with new additional capabilities. Moreover, since these middleboxes are proprietary solutions, they can easily become black-boxes to the network operators [136].

3.2.1.3 Enhancing current router primitives: Another alternative is for the monitoring device vendors to upgrade and enhance their device-centric primitives to reflect the required functionalities of the network operators. For instance, there have been several proposals to enhance the monitoring mechanisms of the routers (e.g., [151–153]). However, this requires network operators and router vendors to commit to a fixed set of required features which is risky since these features can soon become inapplicable.

3.2.2 Network-wide approaches

There have been various proposals for network-wide approaches as opposed to device centric approaches for traffic monitoring in the literature [147–149]. More recently, the emergence of SDN [154] as one of the major efforts towards creating the future Internet provides network operators and developers with newer capabilities and opportunities to create novel networking applications and to explore newer opportunities that were not possible before. SDN architecture decouples the software that controls the network devices (i.e., brain) from the hardware that forwards the packets (i.e., muscle). The key enablers for this are open and clearly focused interfaces that allow developers to program how their applications control and interact with devices and the network data plane. Network providers who leverage these enablers can achieve greater network intelligence, resulting in networks that can adapt better to network infrastructural changes and different networking contexts, have better capabilities to provide higher service levels, are easier to manage, and that allow for the deployment of newer flexible and highly scalable network architectures [155]. Hence, nowadays, programmable network devices or SDN enabled monitoring devices could be the basis for most of the network-wide approaches to address the aforementioned challenges of the device-centric approaches (e.g., [156–158]).

For instance, *Kekely et al.* in their recent publication in *Infocom 2014* [159], propose Software Defined Monitoring (SDM), a novel hardware accelerated monitoring mechanism for flexible flow-based application level monitoring as one of the above-mentioned high-level policies that is required for the next generation of DDoS flooding defense mechanisms. Their proposed approach relies on high-level monitoring policies implemented in the software in conjunction with a configurable hardware accelerator. In other words, the hardware accelerator is an application-specific processor tailored for stateful flow processing with the capability to keep the states in the software layer if needed. The traffic monitoring tasks or objectives/policies rest in the software layer and they can easily control how much detail should be kept in the hardware layer for each flow. Hence, the measurement of non-important traffic is offloaded to the hardware layer, while the high-level traffic monitoring tasks over the important traffic is done in the software layer. Authors believe that their proposed approach

allows for creating flexible monitoring systems capable of deep packet inspection at high throughput. Moreover, their pilot implementation in FPGA is shown capable of performing 100Gb/s flow traffic measurement augmented by a selected application-level protocol parsing.

Sekar et al. propose cSamp as a network-wide traffic monitoring enforcement policy and by casting the network-wide traffic monitoring enforcement problem as a network-wide resource management problem. Next, we review cSamp and perform a case study to better understand: how its flow sampling structure outperforms traditional packet sampling mechanisms, and how its network-wide approach eliminates the majority of the problems with the device-centric approaches. We study cSamp traffic monitoring mechanism that is closely related to our proposed traffic monitoring mechanism. Both cSamp and our proposed mechanism deploy similar policy enforcement (network-wide) approach to enforce different monitoring objectives/policies. If there can be different traffic monitoring policies to achieve different objectives, cSamp and our proposed mechanism are two of those traffic monitoring policies.

3.3 cSamp: A centrally managed system-wide flow monitoring mechanism

Here, we briefly review cSamp, one of the recently proposed traffic monitoring mechanisms that deploys a centrally managed network-wide traffic monitoring policy/objective enforcement approach.

Sekar et al. propose cSamp [136] that treats a network of routers within an AS as a system of routers that are managed and coordinated centrally in order to accomplish a high-level traffic monitoring objective/policy of maximizing the total flow-coverage across all the Origin-Destination (OD)-pairs [1] ($\sum_i T_i \times C_i$) subject to ensuring the optimal minimum fractional coverage per OD-pair ($min_i\{C_i\}$). Figure 3.2 shows an assignment for an OD_i with 20 flows with maximum of 80% flow coverage (total of 16 out of 20 flows) that is calculated through the optimization formulation of cSamp. As shown, not all the flows of OD pair OD_i

[1]Number of flows that are expected to enter an AS through an ingress router and leave an AS through an egress router and these flows are specified by their ingress and egress routers

can be covered considering the router constraints in this specific scenario.

Motivated by the recent proposals on centrally managed network-wide monitoring systems, cSamp's centralized system and its use of hash-based sampling lead to coordination among all the routers within the AS in terms of their monitoring responsibilities and without exploiting any specific communication protocol. Hence, cSamp significantly reduces the communication overhead which is otherwise required to coordinate all the routers in terms of their monitoring responsibilities within each AS. Moreover, cSamp considers router constraints in each monitoring time window to maximize the minimum flow coverage for all the flows.

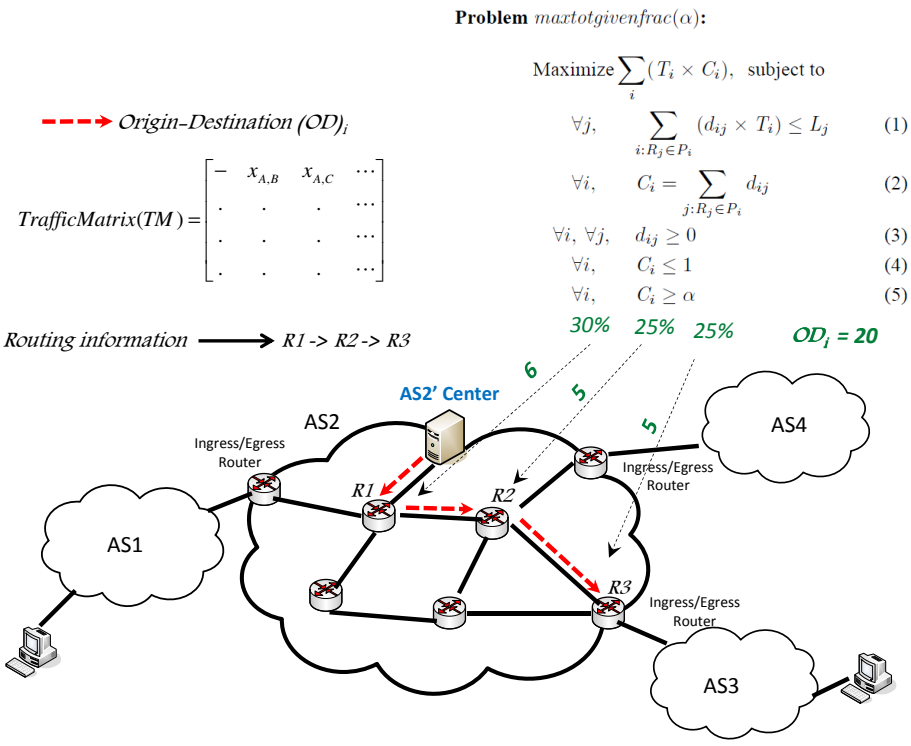


Figure 3.2: cSamp: a working example

As also shown in Figure 3.2, the optimization engine of cSamp uses the traffic matrix and the routing information as inputs in each monitoring time window to compute the optimized distribution of monitoring responsibilities among all the routers within the AS. The output of the optimization engine is then translated into sampling manifests or a list of hash ranges

for each OD pair and then manifests are sent to all of the routers within the AS. Hence, at the beginning of each monitoring time window, routers already know how many flows of which OD pair to monitor.

3.3.1 Case Study: cSamp's vs. other packet/flow sampling mechanisms

In this section, we perform a case study to better understand the benefits of cSamp (as a network-wide flow sampling) over traditional packet/flow sampling mechanisms. In doing so, we implement a prototype of cSamp and reproduce the results of the experiments that cSamp performed in [136] for 5 Real-world topologies to also validate the correctness of our cSamp's prototype implementation. Table 3.1 shows the parameters of these five topologies that we employ for our experiments. These parameters are the estimated *maximum* number of routers (R), flows (F), and unique destination IP addresses (D) for 5 Real-world topologies. In other words, these parameters represent the worst case scenario in terms of the size of the monitoring problem.

Table 3.1: The parameters of the experiment (Estimated maximum number of routers (R), flows (F), and unique destination IP addresses (D) for 5 Real-world topologies)

AS number	**Name**	**(POPs)**	**Number of Routers**	**Number of Flows**	**Number of Destination IPs**
1221	Telstra (Australia)	61	305	44.36×10^6	44.36×10^6
1239	Sprintlink (US)	43	215	31.27×10^6	31.27×10^6
3356	Level3 (US)	52	260	37.81×10^6	37.81×10^6
3257	Tiscali (Europe)	50	250	36.36×10^6	36.36×10^6
4755	Internet2	11	55	8×10^6	8×10^6

The estimated number of destination IP addresses is equal to maximum number of flows with different destination IP addresses in each monitoring time window. Hence, as done

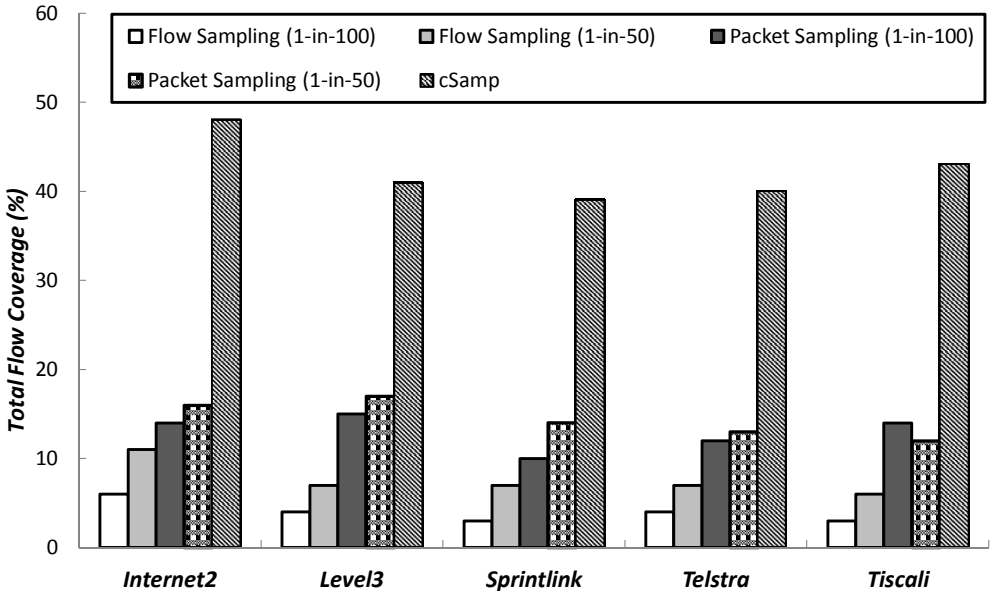
in [136] for cSamp, we also use a baseline traffic volume of 8 million IP flows for Internet2 with 11 PoPs (there is roughly 5 routers per PoP [136])(per five-minute monitoring time window) to scale the total number of flows by the number of routers in each of the topologies. For instance, Telstra with roughly 305 (61*5) routers has roughly $\frac{305}{55} \times 8 = 44.36$ million flows. Moreover, we built cSamp's OD-pairs by considering all possible pairs of PoPs and employ the shortest-path routing to compute the PoP-level path per OD-pair. We employ the static intermediate system-intermediate system (IS-IS) weights and link weights for shortest-path routing for these topologies.

We compare cSamp with uniform packet sampling with the sampling rate of 1-in-100 packets, uniform packet sampling with the sampling rate of 1-in-50 packets, constant-rate flow sampling with the sampling rate of 1-in-100 flows, and constant-rate flow sampling with the sampling rate of 1-in-50 flows. cSamp's LP works with the Origin-Destination (OD) concept. Each OD can be defined as the number of flows that enter an AS from the same gateway router (Origin) and exit the AS from the same gateway (Destination). So, origin and destination in each OD do not imply the origin and destination of the flows within that OD and only imply the entrance and exit gateways within an AS. In order to provide the cSamp's LP with the OD information for its monitoring assignment decision, we added a function on the edge routers to pre-process each incoming packet and implant its OD-flow identification to its header so that later on. We, step by step, followed cSamp's on how to implant OD-flow identification to the packets. The results of our experiments were almost the same as what is reported in [136].

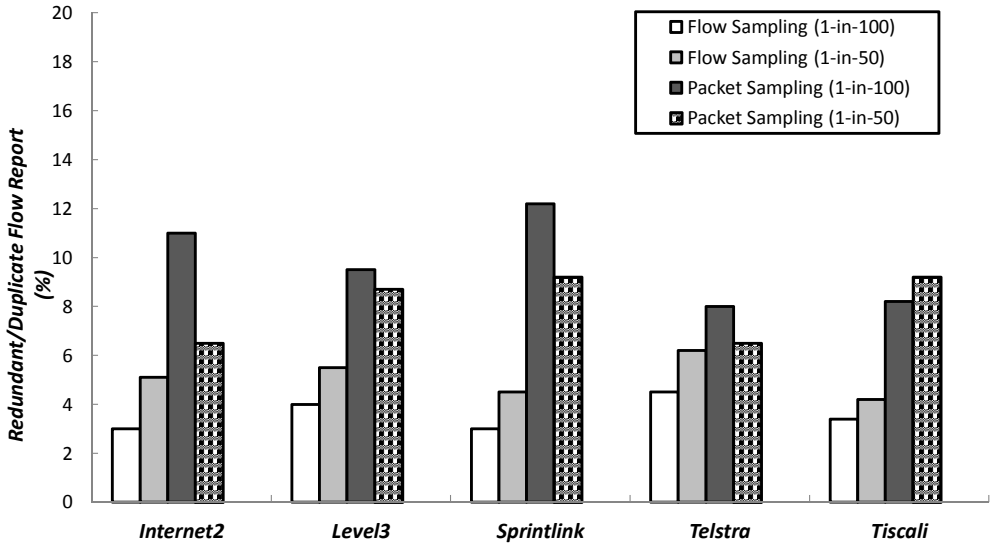
In our simulation, for all the flow sampling and packet sampling mechanisms, for each topology, we use the same: traffic matrix, routing information, number of flows, and flow size distribution [2]. For each topology, we compute the total flow coverage obtained with different packet sampling or flow sampling mechanisms. Figure 3.3a shows the results. The total flow coverage of cSamp is much higher than other flow/packet sampling mechanisms for all of the topologies. cSamp's flow coverage is larger than other flow sampling mechanisms since it's architecture allows for fractional flow coverage of the flows on the same path on separate routers which is not the case for other flow sampling mechanisms. Moreover,

[2]We, assume that the flow size measured in number of packets is Pareto-distributed.

other packet sampling and flow sampling mechanisms suffer from the wasted amount of redundant flows/packets that are monitored (i.e., redundant flow reports) which is shown in Figure 3.3b. cSamp's central coordination structure eliminates the redundant flow/packet monitoring overhead.



(a) Total Flow Coverage



(b) Redundant/Duplicate Flow Reports

Figure 3.3: Comparing cSamp with other packet sampling and flow sampling mechanisms

3.3.2 Discussion:

As we mentioned earlier, traffic monitoring mechanisms can be tailored to achieve various high-level or low-level objectives/policies. However, one of the ideal objectives to achieve in most of the traffic monitoring mechanisms should be to cover as many attack flows as possible in order to provide a clearer picture of network status for different traffic analysis applications. We believe it is fair to compare traffic monitoring mechanisms with each other in terms of their success in providing the best flow coverage when it comes to DDoS flooding attack flow coverage. cSamp's goal is to cover as many flows as possible while guaranteeing the maximum minimum flow coverage for all the flows within each monitoring time window. Another important benefit of cSamp is to eliminate redundant flow monitoring by means of its network-wide coordination within each AS. However, cSamp is not specifically designed for DDoS flooding flow coverage. Therefore, it has some deficiencies when it comes to monitoring DDoS flooding attacks at the intermediate network that motivated us towards our proposed traffic monitoring mechanism. We enumerate these deficiencies here and they will also be highlighted when we discuss the experimental results later.

First, cSamp assumes that its monitored flows/packets are reported to a centralized server within an AS after each monitoring time window for analysis (i.e., centralized analysis). Therefore, it has a major limitation in terms of its role in enabling the faster reaction to the attack flows such as DDoS flooding attack flows. Moreover, collecting all the monitored flows/packets centrally increases the communication overhead drastically. Second, cSamp only guarantees the minimum optimal coverage for all the flows within each Origin-Destination (OD) pair while satisfying the routers' resource constraint. However, in case of DDoS flooding attacks, we believe that, the traffic monitoring mechanism could be tailored in such a way that it could prioritize the monitoring of possible DDoS flooding attack flows over other flows (i.e., Maximize DDoS flooding attack flow coverage as an objective).

3.4 Summary

In this chapter, we focused on traffic monitoring, which is one of the key network management tasks. We studied two existing types of traffic monitoring methods namely packet sampling and flow sampling that are employed in different traffic monitoring mechanisms. We pointed out the limitations of existing packet sampling methods as opposed to flow sampling methods for network security and anomaly detection applications by means of several measurements and by employing one of the recently proposed well-performed flow sampling traffic monitoring mechanisms (cSamp) in a case study. Moreover, our literature review showed that network-wide policy enforcement mechanisms, in which several network management tasks could be cast as network-wide resource management problems, is a better alternative than device-centric policy enforcement mechanisms to be employed by the network operators.

In addition, we emphasized that it is really important for the next generation DDoS defense mechanisms to ensure that the majority of the attack flows are monitored in each and every AS and its the key motivation for the work in this dissertation. In doing so, guaranteeing the flow coverage for the majority of such flows while satisfying the monitoring devices' resource constraints (i.e., memory constraints in this thesis) is a valid and vital traffic monitoring objective/policy to be enforced by monitoring mechanisms that will be employed by the next generation of DDoS defense mechanisms.

Finally, we argue that a centrally managed network-wide traffic monitoring mechanism that coordinates monitoring responsibilities among various monitoring devices and eliminates the redundant flow monitoring overhead, while satisfying the resource constraints of the monitoring devices and prioritizing the coverage of the DDoS flooding attack flows, can enable early detection of DDoS flooding attacks through distributed DDoS flooding detection mechanisms that could be deployed in each of the monitoring devices; hence, this reduces the communication overhead caused by employing the centralized DDoS flooding detection mechanisms. Next, we introduce our proposed traffic monitoring mechanism that attempts to satisfy the aforementioned goals.

4.0 DiCoTraM: A Distributed and Coordinated DDoS flooding attack tailored flow Traffic Monitoring

4.1 Introduction

In order to address the key requirements for the next generation of DDoS flooding defense mechanisms that we summarized in Chapter 3, in this Chapter, we propose DiCoTraM, a DDoS flooding attack tailored, coordinated, and network-wide traffic monitoring mechanism within each AS. DiCoTraM has a centralized traffic monitoring assignment component (Task Assignment Server(TAS)) that assigns monitoring responsibilities within each AS in such a way that those flows intended for the same destination are analyzed together in one place. DiCoTraM enables distributed DDoS flooding detection mechanisms in detecting DDoS flooding attacks at upstream ASs further from the victims' AS.

The rest of this chapter is organized as follows. In Section 4.2, we present DiCoTraM in detail. Section 4.3 compares the performance of the proposed MIP formulation with the proposed scalable heuristic. In Section 4.4, a modified version of the scalable heuristic that is capable of dynamic flow monitoring assignment by pre-processing the input to the assignment process before running the assignment is presented and its performance is evaluated through a set of experiments. Section 4.5 compares DiCoTraM with other packet sampling and flow monitoring mechanisms in terms of total flow coverage and DDoS flooding attack flow coverage for various real-world topologies. In Section 4.6, we present an extended version of DiCoTraM which provides network administrators with the required information to determine the list of candidate monitoring devices to upgrade for achieving better flow coverage as part of their short-term/long-term network planning by tracking the additional memory requirements of the monitoring devices to fully covering the traffic flows. We also

performed some experiments to demonstrate the effectiveness of the proposed modification. Finally, in Section 4.7, we summarize and conclude this chapter.

4.2 DiCoTraM: an overview

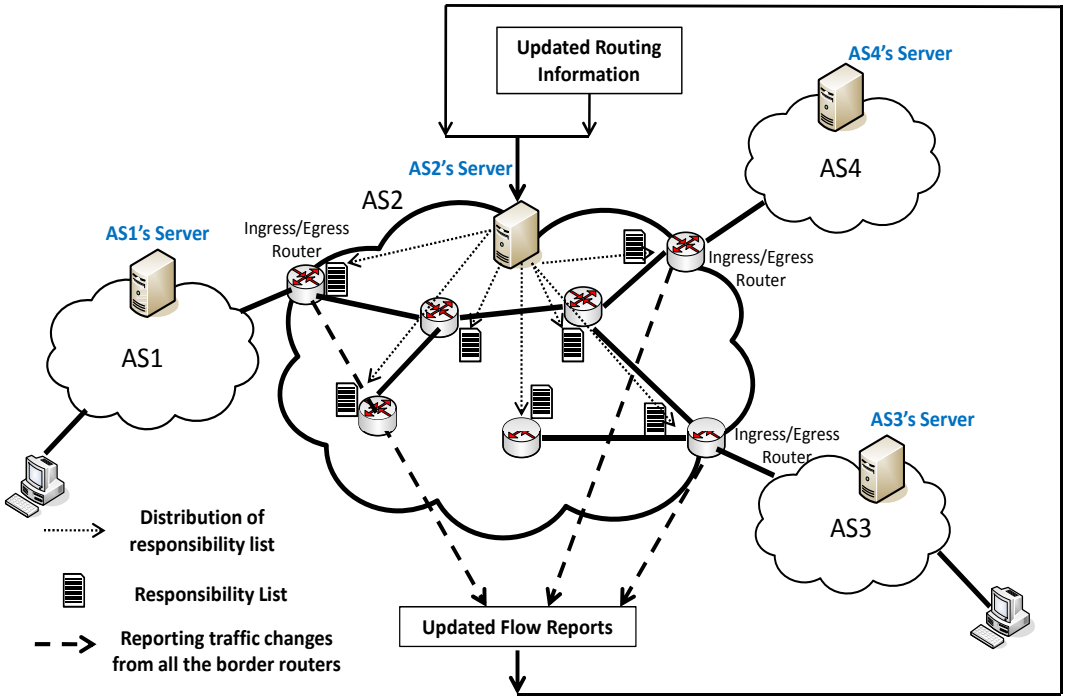


Figure 4.1: DiCoTraM: An architectural overview

As depicted in Figure 2.1, a DDoS flooding attack resembles a funnel in which attack flows are generated in a dispersed area (i.e., sources), forming the top of the funnel. The victim, at the narrow end of the funnel, receives all the attack flows generated. Thus, it is not difficult to see that detecting a DDoS flooding attack is relatively easier at the destination (victim), since all the flows and the correlation among them can be observed at the destination.

Therefore, in order to increase the chances for distributed (hybrid) DDoS flooding defense mechanisms to detect such attacks further upstream, DiCoTraM mimics the role of the victim for the flows heading to the same destination at the upstream ASs. In doing so, DiCoTraM assigns the flow monitoring responsibilities to each of the routers within an AS in such a way

that all the flows intended for the same destination are monitored at the same router. As we mentioned earlier, DiCoTraM enables routers with distributed DDoS flooding detection capabilities to detect DDoS flooding attacks at their early stages at the intermediate network level.

Figure 4.1 illustrates the overall architecture of DiCoTraM. Each AS has a number of ingress/egress routers as well as several access routers. We designate a server as a task assignment server (TAS) within each AS. The TAS is responsible for assigning monitoring responsibilities to all the routers within its AS at the beginning of each monitoring time window [1]. TAS uses the current traffic matrix and routing information as inputs of its assignment algorithm for the next monitoring time window. The outcome of DiCoTraM's assignment decision for the next monitoring time window is a *responsibility list* for each router, which TAS distributes to the routers. A router's responsibility list is *the list of flows that are assigned to that router for the next monitoring time window to monitor*. Each router maintains a flow table [2] on the combination of SRAM and DRAM as we explain in Subsection 4.2.1.

As we mentioned earlier, DiCoTraM would be suitable to be employed as part of a hybrid DDoS defense mechanism (e.g., DCD architecture [160]). Hence, TASs could also be responsible for pulling or receiving alerts that are generated by the distributed DDoS flooding detection mechanism employed by the routers within an AS (as CAT servers in [160]). However, different challenges would be necessary to address if TASs were to communicate their reported alerts or their efficient response policies (for various situations) with each other, as part of a hybrid DDoS defense mechanism. For instance, TASs communication requires to be synchronized. In the current version, we assume there is no communication between the TASs and TAS's main responsibility is within the scope of an individual AS. Moreover, we assume that TAS is implemented on a single machine/router; however, an alternative approach would be to run TAS on multiple machines for load balancing and fault tolerance.

[1] We set monitoring time window to 5 minutes in our current implementation. However, the monitoring time window can be reduced based on the size of the network.

[2] A set of all the flows that it is currently monitoring.

In DiCoTraM, each router maintains specific statistics for its flows in SRAM as we explain in Subsection 4.2.1. In doing so, for any monitored flow, the router either (a) creates a new entry if there is no flow entry already in the flow table or (b) updates the counters of the corresponding entry in the flow table (if the aggregation of the flow's destination IP with the existing destination IP prefix in the flow table is possible[3]). Moreover, aggregating the flows that are intended to the same destination IP as one aggregated flow with the same destination IP prefix and maintaining the statistics for that aggregated flow in SRAM, reduces the number of entries in SRAM. The destination IP field for the aggregated flow is an IP prefix of all the aggregated flows. All the ingress/egress routers within each AS are responsible for reporting traffic dynamics to the TAS so that it can update the responsibility lists promptly for subsequent monitoring time windows. Routing information will also be updated through flow-based monitors such as OSPF monitors [161].

We previously proposed a Linear Programming (LP) formulation to assign monitoring responsibilities to the monitoring devices in such a way that such monitoring mechanism could enable distributed DDoS flooding detection mechanisms in detecting DDoS flooding attacks through the routers within an AS assuming that the memory size of the routers was infinite [79]. Also, we had the load balancing of monitoring responsibilities as an objective/policy to be enforced in our previous work. Here, we reformulate and improve our previous monitoring mechanism by incorporating the memory limitations of the routers and by proposing an advanced high-level DDoS flooding attack tailored objective/policy. Our current objective is to maximize the total flow monitoring coverage for all the flows while prioritizing the coverage of possible DDoS flooding flows.

We assume that each router has ≈128MB of DRAM as the minimum default DRAM capacity of most of the current routers. As discussed in [162], we also assume that only 8MB out of total of 12MB SRAM size is available for retaining flow states on each router and each router's memory size (i.e., SRAM) is limited to 8MB (i.e., m_r). Since DRAM capacity is not currently as scarce as SRAM on the routers and the required DRAM capacity for our proposed mechanism is much lower than DRAM's default capacity (≈128MB), we only look at SRAM capacity as a constraint in our proposed monitoring mechanism; however, consid-

[3]In this case, the destination IP prefix is updated with the newly aggregated destination IP prefix.

ering DRAM capacity as a constraint in our proposed work (if required in future) is easily implementable. We discuss routers' memory constraints in detail in Section Subsection 4.2.1.

We propose a Mixed Integer Programming (MIP) formulation that improves our previously proposed LP formulation in terms of performance. The new MIP formulation aims at finding a feasible solution. A feasible solution for our assignment algorithm determines the monitoring responsibilities of all the routers for the next monitoring time window by maximizing the flow monitoring coverage of all the flows while prioritizing the coverage of possible DDoS flooding flows at the same router and when the memory size of the routers is limited.

Next, we first discuss the routers' memory constraints within the ASs in Subsection 4.2.1. Then, we describe the assumptions we make in Section 4.2.2. We present the notation adopted in Subsection 4.2.3. Our set-up process to provide data required as input to the assignment MIPs for determining the monitoring responsibilities is presented in section 4.2.4. Finally, our network-wide assignment of the monitoring responsibilities is presented in sections 4.2.5.

4.2.1 Discussion: Router memory constraints

The accuracy and scalability of traffic measurement techniques employed for various purposes (e.g., QoS provisioning, accounting, *etc.*) are usually bounded by the memory sizes of the monitoring devices (e.g., a router's memory).With migration of networks from 1Gbps to 10Gbps and most recently to 40Gbps in all the networked environments (e.g., ISPs, Internet core, and data centers), significant increase in traffic flows is expected. Monitoring these flows effectively and in a scalable manner on slow (around 50 ns), large (up to 1GB), and cheap DRAM memory modules or fast (around 5 ns), relatively small (up to 12MB out of which 8MB is available to use [162]), and expensive SRAM memory modules of the measurement devices has been challenging [163].

Cisco's NetFlow [129] used to keep 64-byte per flow state in DRAM which had required updating DRAM for each packet per flow and was not found to be scalable (i.e., high loss rates) as the number of traffic flows increased. Hence, packet sampling approaches that

increment per flow counters only for sampled packets has been implemented as an enhancement to NetFlow, but these sampling approaches have led to inaccurate estimations. *Estan et. al*, in [163], propose two scalable algorithms that use SRAM to store per flow entries to address both the inaccuracy of sampled NetFlow and the slow DRAM accessibility of NetFlow. As suggested in [164], the amount of SRAM needed to hold on to the flow states can be further reduced by using a combination of SRAM and DRAM on each router [165]. Similarly, there are other proposals that try to offload most of the static flow fields to DRAM and hold dynamic fields relevant to counting per flow states in SRAM [136].

In our experiments, we also only maintain important fields of each flow that are relevant to counting per flow states in SRAM. We keep two lists of incoming and outgoing flows in SRAM. In the incoming flows list, we only keep a 4-byte packet counter, a 4-byte packet size, a 4-byte destination IP (or aggregated destination IP prefix), and a 1-byte incoming router port for each flow. Since most of the DDoS flooding detection mechanisms require the packet counts and packet sizes of the outgoing packets of suspicious flows in the outgoing flows list, we keep a 4-byte packet counter, a 4-byte packet size, a 4-byte destination IP (or aggregated destination IP prefix), and a 1-byte outgoing router port for each flow. Since we are aggregating the flows that are headed to the same destination IP as one aggregated flow with an IP prefix of all the aggregated flows, we can save more on SRAM space and our adopted detection algorithms also perform well with the aggregated flow states. In our monitoring mechanism, we only need $ReqMem_{f,r}$[4] bytes of SRAM for flow f on router r. $ReqMem_{f,r}$ is defined as follows:

$$ReqMem_{f,r} = [(np_{in})_{f,r} + (np_{out})_{f,r}] \cdot 13\ bytes \tag{4.1}$$

Where, $(np_{in})_{f,r}$ and $(np_{out})_{f,r}$ are number of incoming and outgoing ports of the routers that each flow enters or leaves through on its path. The total number of ports on each router is variable (e.g., typically 4, 8, or 16 ports). When flows are aggregated, $(np_{in})_{f,r}$ and $(np_{out})_{f,r}$ are the maximum number of incoming and outgoing ports for the aggregated flows based on

[4]If we assume that each PoP in the network is expected to hold up to 500,000 flow records and If there is roughly 5 routers per PoP [136], 10 interfaces per router, and 13 bytes per flow record, this requires $\frac{500,000 * 13}{5 * 10}$ = 130 KB SRAM per linecard, which is well within the 8 MB technology limit suggested by [162].

the paths of all the aggregated flows. In our experiments, for each flow or aggregated flows, the number of incoming and outgoing ports are known for the simulated topology. The rest of the flow fields such as: a 4-byte source IP, a 4-byte aggregated destination IP prefix (if any), and a 4-byte actual (i.e., not aggregated) destination IP are maintained in DRAM. A 4-byte aggregated destination IP prefix that we add to all the aggregated flows on DRAM enables the retrieval of various source IP addresses of the aggregated flows later.

4.2.2 Assumptions

For DiCoTraM, we assume the following:

- Each flow has a fixed route and there is no bifurcation of traffic flows.
- ASs are only multi-homed to multiple ASs (not multi-homed to a single AS) [5].

4.2.3 Notation

In this section, we introduce the notation adopted in order to formulate each of the MIPs in our proposed mechanism. Figure 4.2 shows a sample topology of an AS with six routers, five flows, three source IP addresses, and six destination IP addresses to depict the notation. Here, we introduce the notation:

Sets and parameters:

- $\mathcal{F} =$ Set of all the flows within a given AS.
- $\mathcal{R} =$ Set of all the routers within a given AS.
- $\mathcal{D} =$ Set of all the destination IP-addresses of the flows within a given AS.
- $f \in \mathcal{F}$ is represented as a tuple $(s_f, d_f, p_f, ReqMem_f)$,
  where,
  $s_f =$ Source IP address.
  $d_f =$ Destination IP address.
  $p_f =$ Set of routers within a given AS that are in the path of flow f, $p_f \subseteq \mathcal{R}$.
  $ReqMem_{f,r} = [(np_{in})_{f,r} + (np_{out})_{f,r}] \cdot 13\, bytes \qquad \forall r \subseteq p_f$

[5] Multi-homed ASs are ASs that maintain connections either to a single or multiple ASs in order to keep their Internet connectivity in case of a complete connection failure of one of their AS connections [166]

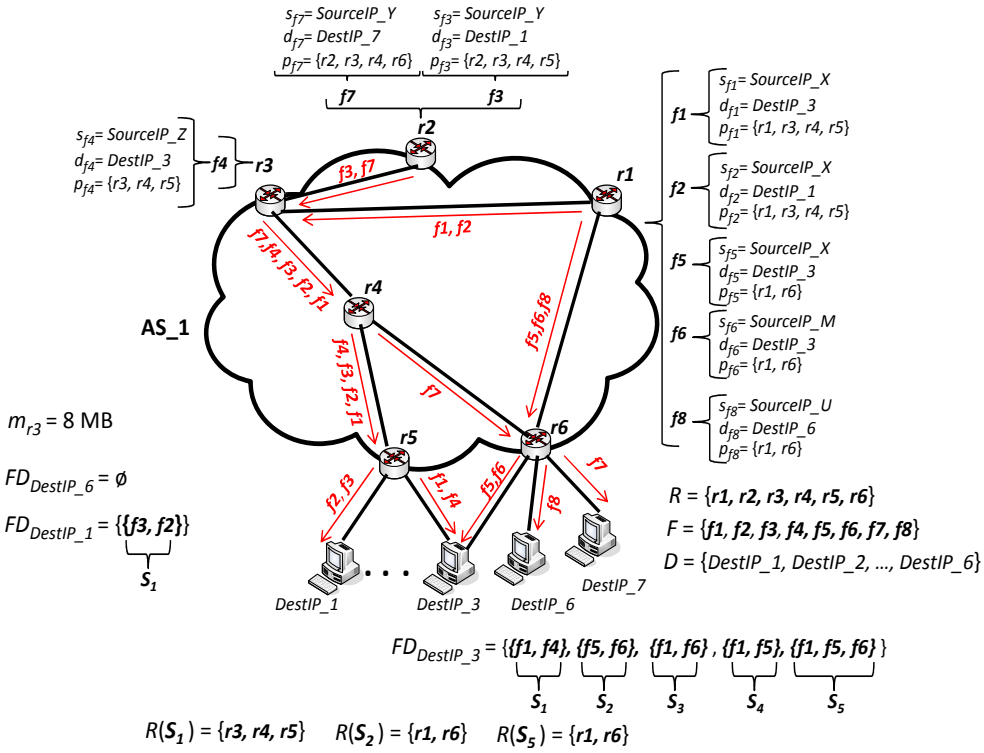


Figure 4.2: Topology of an AS illustrating the used notation.

- $\mathcal{FD}_d = \{S_1, S_2, ...\} \subseteq 2^{\mathcal{F}}$ such that $\forall S_i \in \mathcal{FD}_d$, the following holds:
  - $|S_i| \geq 2$, and
  - For each $f \in S_i$, $d_f = d$, and
  - Let $R(S_i) = \bigcap_{f \in S_i} p_f$; then $|R(S_i)| \geq 1$
  - $ReqMem_{S_i,r} = \left[\max_{f \in S_i,\ r \in p_f} (np_{in})_{f,r} + \max_{f \in S_i,\ r \in p_f} (np_{out})_{f,r} \right] \cdot 13\ bytes$

  i.e., each S_i is a subset of the power set of flows in $\mathcal{F}$ with at least two flows that have the same destination and share at least one router on their paths to that destination. $\mathcal{FD}_d$ is a set of all such subsets (i.e., S_i). Moreover, $R(S_i)$ is a subset of the routers in $\mathcal{R}$ that flows in $S_i \in \mathcal{FD}_d$ share on their path to destination d. For instance, in Figure 4.2, $\mathcal{FD}_{DestIP_3} = \{S_1, S_2, S_3, S_4, S_5\}$, where $S_1 = \{f_1, f_4\}$, $S_2 = \{f_5, f_6\}$, *etc.* Moreover, $R(S_1) = \{r3, r4, r5\}$, $R(S_2) = \{r1, r6\}$, and *etc.* As another example, $\mathcal{FD}_{DestIP_6} = \emptyset$ since there is only one flow to $DestIP_6$.
- $m_r \geq 0$ is the current capacity/memory space of router r. In Figure 4.2, $m_{r3} = 8$ MB.

Variables:

- $x_{S_i,r,d} = \begin{cases} 1, & \text{if all the flows in } S_i \in \mathcal{FD}_d \text{ are assigned to a router } r \in R(S_i); \\ 0, & \text{otherwise.} \end{cases}$

- $y_{f,r} = \begin{cases} 1, & \text{if flow } f \in \mathcal{F} \text{ is assigned to router } r \in p_f; \\ 0, & \text{otherwise.} \end{cases}$

Parameters:

- $\alpha_{f,S_i} = \begin{cases} 1, & \text{if flow } f \in \mathcal{F} \text{ is in } S_i \in \mathcal{FD}_d \\ 0, & \text{otherwise.} \end{cases}$

4.2.4 The Set-up Process

There are various inputs that should be initialized for our MIP formulations in the TASs. These inputs include the following: set of all the routers ($\mathcal{R}$), set of all the flows ($\mathcal{F}$), and the set of all the destination IP addresses ($\mathcal{D}$). These inputs are directly available through the flow reports and the routing information. Each flow has a source IP, destination IP, and a dedicated path. Furthermore, other inputs are: $\mathcal{FD}_d$, and $R(S_i)$ for each $S_i \in \mathcal{FD}_d$.

In order to better understand the scale of the MIP assignment problem, Table 4.1, an extended version of Table 3.1, shows the estimated number of flows ($\mathcal{F}$) and destination IP addresses ($\mathcal{D}$) based on the number of Points of Presence (POPs) for the ten real-world topologies (ISPs) measured in a study in 2002 by Rocketfuel [167]. We have estimated the number of routers for each topology ($\mathcal{R}$) (assuming 5 routers per POP [136]).

The estimated number of destination IP addresses is equal to maximum number of flows with different destination IP addresses in each monitoring time window. Hence, as done in [136] for cSamp approach, we also use a baseline traffic volume of 8 million IP flows for Internet2 with 11 POPs (per five-minute monitoring time window) to scale the total number of flows by the number of routers in each of the topologies. For instance, Telstra with roughly 305 routers has roughly $\frac{305}{55} \times 8 = 44.36$ million flows; hence, the estimated number of different destination IP addresses for Telstra is 44.36 million IP addresses.

Table 4.1: The estimated maximum number of routers ($\mathcal{R}$), flows ($\mathcal{F}$), and unique destination IP addresses ($\mathcal{D}$) for 10 Real-world topologies

AS number	**Name**	**(POPs)**	**Number of Routers ($\mathcal{R}$)**	**Number of ($\mathcal{F}$)**	**Number of ($\mathcal{D}$)**
1221	Telstra (Australia)	61	305	44.36×10^6	44.36×10^6
1239	Sprintlink (US)	43	215	31.27×10^6	31.27×10^6
1755	Ebone (Europe)	25	125	18.18×10^6	18.18×10^6
7018	AT&T (US)	108	540	78.54×10^6	78.54×10^6
3356	Level3 (US)	52	260	37.81×10^6	37.81×10^6
2914	Verio (US)	121	605	88×10^6	88×10^6
3257	Tiscali (Europe)	50	250	36.36×10^6	36.36×10^6
3967	Exodus (US)	23	115	16.72×10^6	16.72×10^6
4755	Internet2	11	55	8×10^6	8×10^6
6461	Abovenet (US)	21	105	15.27×10^6	15.27×10^6

4.2.5 Proposed MIP formulation

Our general goal here is to maximize the flow monitoring coverage within each AS and for each monitoring time window while satisfying the memory constraints of the routers and prioritizing the flow coverage of possible DDoS flooding attack flows. In doing so, the proposed MIP first assigns as many flows, that are intended for the same destination (i.e., Probable DDoS flooding attack flows), as possible to be monitored at one of the routers on the path of those flows to their destination. Then, for the rest of the flows, the MIP assigns as many flows as possible to the routers on flows' paths to their destinations.

Given the earlier notation and definitions, maximizing the flow monitoring coverage problem can be formulated as follows:

$$\max_{x_{S_i,r,d},\, y_{f,r}, S_i} \sum_{d\in\mathcal{D}} \sum_{S_i\in\mathcal{FD}_d} \sum_{r\in R(S_i)} |S_i| \cdot x_{S_i,r,d} + \sum_{r\in\mathcal{R}} \sum_{f\in\mathcal{F}} y_{f,r} \tag{4.2}$$

subject to,

$$\sum_{d\in\mathcal{D}} \sum_{S_i\in\mathcal{FD}_d} ReqMem_{S_i,r} \cdot x_{S_i,r,d} + \sum_{f\in\mathcal{F}} ReqMem_{f,r} \cdot y_{f,r} - m_r \leq 0 \qquad \forall r \in \mathcal{R} \tag{4.3}$$

$$\sum_{S_i\in\mathcal{FD}_d} \sum_{r\in R(S_i)} x_{S_i,r,d} \geq 1 \qquad \forall d \in \mathcal{D} \tag{4.4}$$

$$\sum_{r\in R(S_i)} x_{S_i,r,d} \leq 1 \qquad \forall d \in \mathcal{D},\ \forall S_i \in \mathcal{FD}_d \tag{4.5}$$

$$\sum_{r\in\mathcal{R}} y_{f,r} \leq 1 \qquad \forall f \in \mathcal{F} \tag{4.6}$$

$$\sum_{S_i\in\mathcal{FD}_d} \sum_{r\in R(S_i)} \alpha_{f,S_i} \cdot x_{S_i,r,d} = 1 \qquad \forall d \in \mathcal{D},\ \forall f \in \mathcal{F} \tag{4.7}$$

$$1 - \sum_{r\in\mathcal{R}} y_{f,r} \geq \sum_{d\in\mathcal{D}} \sum_{S_i\in\mathcal{FD}_d} \sum_{r\in R(S_i)} \alpha_{f,S_i} \cdot x_{S_i,r,d} \qquad \forall f \in \mathcal{F} \tag{4.8}$$

$$x_{S_i,r,d} \in \{0,1\} \qquad \forall S_i \in \mathcal{FD}_d, \forall r \in R(S_i), \forall d \in \mathcal{D} \tag{4.9}$$

$$y_{f,r} \in \{0,1\} \qquad \forall r \in \mathcal{R},\ \forall f \in \mathcal{F} \tag{4.10}$$

The objective function in (4.2) is to find the maximum number of flows that will be monitored during the next monitoring time window. The set of constraints in (4.3) ensures that both (i) the flows intended to a given destination are assigned to one of their common routers on their paths, and (ii) the rest of the flows are assigned to the rest of the routers

considering the current capacity of the routers [6]. The set of constraints in (4.4) ensures that at least one S_i is selected among all the S_is that are in $\mathcal{FD}_d$ for all the destinations. The set of constraints (4.5) and (4.6) ensures that S_is that are in $\mathcal{FD}_d$ for all the destinations and the rest of the independent flows are assigned to at most one router, respectively. The set of constraints in (4.7) forbids the selection of two distinct S_is that constitutes common flows for all the destinations. The set of constraints in (4.8) implies that one flow is going to be an independent flow unless it is covered as part of one of the selected S_is, for all the flows. Finally, the set of constraints in (4.9) and (4.10) define $x_{S_i,r,d}$ and $y_{f,r}$ as binary variables.

Without loss of generality, we assume that for each destination (d) there exists a collection of flows (S_i) and a common router (r) on the paths of flows in S_i such that the memory capacity of router r is enough to fulfil the memory requirements of all the flows in S_i.

We could reduce the known NP-complete problem of PARTITION [168] to the above MIP formulation in (4.2)-(4.10). Hence, the proposed MIP is a NP-hard problem.

4.2.6 Proposed heuristic

Computer networks are changing rapidly and our proposed DiCoTraM mechanism should accommodate the dynamic nature of the network. However, computing and distributing the responsibility lists by employing our proposed MIP formulation requires few minutes and sometimes more on large topologies (as we show later in Table 4.2 and Table 4.3); hence, our MIP formulation do not scale well. In order to provide an alternative solution that reduces the computation time (scales better), in this section, we propose a heuristic to produce the responsibility lists. In our experiments, as we explain later in section 4.3, we compare our proposed MIP formulation with our proposed heuristic, for various scenarios with different topology sizes, in a limited time window. We show that the proposed heuristic is not that far off from the optimum solutions for most of the scenarios in which the MIP could find the optimum solution on-time.

[6] Our objective function (4.2) makes sure that the flow coverage is maximized

In order to explain our proposed heuristic, we use the notation in section 4.2.3 and also define set B as follows:

$$B = \bigcup_{d \in \mathcal{D}} \mathcal{F}\mathcal{D}_d \tag{4.11}$$

i.e., B is the set of all $S_i \in \mathcal{F}\mathcal{D}_d$ for all $d \in \mathcal{D}$.

Algorithm 1 The proposed heuristic algorithm

Inputs: B , $m_r, \mathcal{F}$
Output: $Assigned, Covered$, also sending out the routers' responsibility lists

Sort B by the size of its elements (x_i) in a descending order
$Assigned = 0$
$Covered = \emptyset$
for all $x_i \in B$ **and** $x_i \cap Covered = \emptyset$ **do**
  $Assigned_R = 1$
  for all $r \in R(x_i)$ **and** $Assigned_R = 1$ **do**
    if $m_r \geq ReqMem_{x_i,r}$ **then**
      Assign flows in x_i to router r
      Update router r's capacity ⇒ using formulation 4.1
      Remove all $f \in x_i$ from $\mathcal{F}$
      $Assigned = Assigned + |x_i|$
      $Covered = Covered \cup x_i$
      $Assigned_R = 0$
for all $f \in \mathcal{F}$ **do**
  if $|p_f| = 1$ **and** $m_{r \in p_f} \geq ReqMem_{f,r}$ **then**
    Assign flow f to router r
    Update router r's capacity ⇒ using formulation 4.1
    Remove f from $\mathcal{F}$
    $Assigned++$
    $Covered = Covered \cup f$
for all $f \in \mathcal{F}$ **do**
  $Assigned_R = 1$
  for all $r \in p_f$ **and** $Assigned_R = 1$ **do**
    if $m_r \geq ReqMem_{f,r}$ **then**
      Assign flow f to router r
      Update router r's capacity ⇒ using formulation 4.1
      $Assigned++$
      $Covered = Covered \cup f$
      $Assigned_R = 0$

In our scalable heuristic, first, we create set B. Next, we sort B in a descending order based on the size of its elements (i.e., x_i) for various destinations (i.e., the largest $|S_i|$ will be on top of the sorted list).

For each element $x_i \in B$ and for each of the routers on $R(x_i)$, we calculate $ReqMem_{x_i,r}$ through the formulation we presented in section 4.2.1. Then, we assign all the flows that belong to x_i to one of the routers that belongs to $R(x_i)$. Next, for the rest of the flows, we also calculate $ReqMem(f, r)$ for each of the routers on their p_f. Then, we first assign flows that only have one router on their p_f to ensure their coverage. Finally, we assign the

rest of the flows (as many as possible) to one of the routers on their p_f. Algorithm 1 is the pseudo-code of the proposed heuristic algorithm that runs on each TAS.

4.3 Scalability analysis: MIP vs. the proposed heuristic

As we explained earlier, the proposed MIP formulation is a NP-hard problem. Achieving an efficient and scalable solution for a NP-hard problem using available solvers is challenging and it really depends on the size of the problem, time limitations, and its parameters. In our experiments, we have used Gurobi optimizer [169] to implement the MIP formulation. The MIP formulation is implemented in C++ code using the Gurobi callable library. MIP problems are generally solved using a branch-and-bound algorithm. Gurobi optimizer also implements an improved branch-and-bound algorithm to solve the MIP problems. Since the monitoring time window is fixed to 5 minutes in our DiCoTraM implementation, it is ideal to have the responsibility lists available to all the routers within the monitoring time window ($<$ 5 minutes). Hence, we assume the time limitation for both our proposed MIP formulation and our proposed heuristic is bounded to 4-minute (240 seconds). As it is shown in Table 4.2 and Table 4.3, in our experiments, there were scenarios that the proposed MIP formulation could find optimal solution within the time limit (i.e., where it solved in $<$ 240 seconds) or in other words it was scalable and there were many scenarios that it could not find the optimal solution within the time limit (i.e. where it took $>$ 240 seconds) or in other words, it was not scalable.

In order to show that our proposed alternative heuristic performs reasonably well in terms of the problem computation time and solution closeness to the optimal in comparison with the MIP formulation in both the scenarios where MIP provides the optimal solution and for the scenarios where MIP can not find the optimal solution, we run Gurobi to solve the MIP formulation with 4 minutes as the time limit and run the heuristic with the same 4 minutes time limit for the experimental set-up we explain next.

4.3.1 Experimental set-up:

We choose three real-world topologies namely Internet2, Ebone (Europe), and AT&T (US) from [167]. For each topology, we randomly generate different problem sizes with: fixed number of flows ($\mathcal{F}$), fixed number of routers ($\mathcal{R}$), fixed number of destination IP ($\mathcal{D}$), and fixed memory size for the routers (m_r). Then, for each problem size with the fixed number of flows, we generate 5 problem instances by randomly generating tuples (s_f, d_f, p_f) for each of the flows in each problem instance. Then, we implement straight forward algorithms in C++ to generate the rest of the required inputs that are introduced in section 4.2.4 for each run of our MIP formulation. We restate here that all of the experiments are bounded to 4-minute time limit as the monitoring time window is assumed to be 5 minutes and responsibility lists should be made available to all the routers within the monitoring time window (< 5 minutes).

Moreover, we generated flow's tuples in such a way that for the half of all the destination IP addresses of the flows on each monitoring time window, there is at least two flows going to each of those destination IP addresses for which there is at least one router in common on their paths. NS-2 [170], a widely used network simulator, has been chosen to generate these topologies with the desired traffic inputs on them.

4.3.2 Experimental results:

Table 4.2 and Table 4.3 show the performance comparison between our heuristic and the Gurobi 5.5 solver's result in solving our MIP for 3 real-world topologies with various problem sizes and problem instances for each topology on a desktop PC system with *Intel Core 2 Duo E6700 2.66 GHz* CPU.

Table 4.2: Performance evaluation: Proposed heuristic vs. Gurobi 5.5

Topology	$\mathcal{F}$	$\mathcal{D}$	Ins. No.	Scalable Heuristic (sec.)	Deviation from Optimal (%)	Closeness (%)	Gurobi (sec.)	Optimality Gap (%)
Internet2 ($\mathcal{R} = 55$)	1×10^6	0.6×10^6	1	1.14	3.4	-	15.37	0.00
			2	1.85	2.2	-	20.32	0.00
			3	1.7	0	-	24.4	0.00
			4	1.65	0.3	-	20.57	0.00
			5	8.37	2.2	-	24.28	0.00
	2×10^6	1.2×10^6	1	1.27	0	-	31.22	0.00
			2	1.36	2.7	-	33.43	0.00
			3	1.60	3.4	-	32.17	0.00
			4	1.15	0.2	-	31.23	0.00
			5	1.37	1.5	-	34.31	0.00
	3×10^6	2.1×10^6	1	12.03	2	-	51.96	0.00
			2	2.13	0	-	56.11	0.00
			3	2.65	0	-	87.16	0.00
			4	2.43	1.6	-	91.14	0.00
			5	22.30	2.0	-	94.1	0.00
	5×10^6	3.4×10^6	1	3.23	2.3	-	154.27	0.00
			2	3.13	1.6	-	150.35	0.00
			3	3.85	0.5	-	157.21	0.00
			4	13.53	0.8	-	152.16	0.00
			5	4.11	-	0.7	> 240	21.47
Ebone (Europe) ($\mathcal{R} = 125$)	3×10^6	1.9×10^6	1	3.12	1.5	-	63.71	0.00
			2	23.33	2.6	-	69.0	0.00
			3	3.50	1.8	-	72.36	0.00
			4	3.21	0	-	63.66	0.00
			5	3.37	0.8	-	79.25	0.00
	6×10^6	4.8×10^6	1	14.51	2.7	-	126.25	0.00
			2	4.24	2.1	-	121.51	0.00
			3	4.58	3.4	-	136.21	0.00
			4	24.14	3.8	-	130.61	0.00
			5	4.49	2.1	-	114.31	0.00
	9×10^6	7.8×10^6	1	5.43	1.2	-	169.9	0.00
			2	15.91	1.5	-	195	0.00
			3	11.05	-	2.9	> 240	30.69
			4	37.60	2.6	-	187.77	0.00
			5	25.95	-	1.6	> 240	28.72
	11×10^6	8.5×10^6	1	6.87	4.1	-	197.85	0.00
			2	7.03	-	5.8	> 240	35.80
			3	10.60	-	4.2	> 240	37.24
			4	9.43	-	3.3	> 240	36.10
			5	17.30	-	0.9	> 240	37.43

Table 4.3: Performance evaluation: Proposed heuristic vs. Gurobi 5.5 (continued...)

Topology	$\mathcal{F}$	$\mathcal{D}$	Ins. No.	Scalable Heuristic (sec.)	Deviation from Optimal (%)	Closeness (%)	Gurobi (sec.)	Optimality Gap (%)
AT&T (US) ($\mathcal{R} = 540$)	11×10^6	9.0×10^6	1	36.96	3.8	-	201.13	0.00
			2	17.14	-	1.7	> 240	40.95
			3	7.60	-	2.8	> 240	24.24
			4	31.34	-	1.2	> 240	42.56
			5	27.67	-	3.6	> 240	35.87
	18×10^6	12.1×10^6	1	49.14	-	2.2	> 240	48.32
			2	19.23	-	3.7	> 240	27.10
			3	28.89	-	1.9	> 240	38.24
			4	39.35	-	4.2	> 240	49.51
			5	45.21	-	3.2	> 240	39.16
	25×10^6	17.6×10^6	1	107.23	1.6	-	232.96	0.00
			2	141.68	-	1.0	> 240	32.30
			3	62.05	-	3.1	> 240	43.24
			4	52.41	-	2.5	> 240	23.98
			5	53.30	-	1.4	> 240	45.59
	37×10^6	21.9×10^6	1	116.43	-	2.2	> 240	39.00
			2	117.24	-	4.2	> 240	50.24
			3	107.80	-	2.1	> 240	41.07
			4	97.03	-	2.5	> 240	49.30
			5	118.88	-	4.2	> 240	33.21

Gurobi reports optimality gap (between 0-100%) for the problems that it can not provide the optimal solution within the time limit. The closer is optimality gap to 0 the closer is the solution to optimal solution. In order to compare the quality of the results of the proposed scalable heuristic, we calculate *Deviation from optimal* [7] as the deviation of the heuristic solution value from the optimal MIP value for the scalable scenarios of the MIP problem.

For the scenarios that Gurobi could not solve the MIP within the time limit, we calculate *Closeness* as the deviation of the heuristic solution value from the MIP value that is available at the end of the time limit (although it is not optimal). The results show that Gurobi cannot solve the majority of our problem instances within the time limit. However, the proposed scalable heuristic solves all of the problem instances in no more than 141 seconds and it is faster than the proposed MIP in most of the scenarios. Moreover, the quality of the results of our scalable heuristic shows that it is not far off from the MIP solution.

4.4 Modified heuristic with input pre-processing capability

Although the current version of our proposed heuristic outperforms our MIP formulation, the time it takes for the heuristic to solve the problem instances is still high and need to be reduced. Presently, TAS runs our heuristic every 5 minutes for the whole set of inputs calculated from the flow reports and the routing information. However, some of the flows may be long-term and they will be active for quite sometime without any significant changes on their traffic volume and routing; hence, it sounds reasonable to keep the same routers to monitor those flows. In other words, there is no need to change the monitoring assignment of such flows in the next runs of the assignment algorithm as long as there is no significant changes on the flows' routing information or traffic volume and more importantly as long as the coverage required for DDoS flooding attack flows is met. In this section, we present a modified version of our previously presented heuristic that is capable of dynamic flow monitoring assignment by pre-processing the input to the assignment process before

[7] $\frac{(Optimal\ \ solution) - (Heuristic\ \ solution)}{(Optimal\ \ solution)}$

running the assignment. Basically, our goal is to run the assignment only for the set of new or modified flows and skip the unnecessary flow monitoring re-assignment process for the set of flows for which nothing has been changed from the last flow monitoring assignment period. However, since the flow monitoring reassignment at the beginning of each monitoring time-window (current heuristic) ensures the maximal distributed DDoS flooding flow coverage, our modified heuristic should still meet such requirement by sacrificing/tolerating some degradation ($\epsilon \geq 0$) in DDoS flooding flow coverage, which we call the *DDoS monitoring coverage tolerance*, as it is defined as follow:

$$DDoSCov_d^{t_{n-2}} - DDoSCov_d^{t_{n-1}} \leq \epsilon \qquad \forall d \in \mathcal{D} \tag{4.12}$$

Assuming $t_1, t_2, t_3, \cdots, t_n$ be the discrete monitoring time windows, $DDoSCov_d^{t_{n-1}}$ is the total number of covered flows heading to destination d to the total number of flows that are heading to destination d in the previous monitoring time window (i.e., t_{n-1}) and $DDoSCov_d^{t_{n-2}}$ is the total number of covered flows heading to destination d to the total number of flows that are heading to destination d in the previous previous monitoring time window (i.e., t_{n-2}). In equation 4.12, we call ϵ the *uncovered DDoS flooding flow tolerance*. The uncovered DDoS flooding flow tolerance is: the tolerance to the number of DDoS flooding flows that are not covered by the traffic monitoring mechanism or the tolerance to the reduction on the DDoS flooding flow coverage of the traffic monitoring mechanisms. The uncovered DDoS flooding flow tolerance (ϵ) can be defined experimentally either as a general tolerance threshold defined with the same value for all the flows or it can be defined with various values for each group of flows that are heading to each destination IP address (i.e., $d \in \mathcal{D}$) depending on the Service Level Agreements (SLAs) that different service providers have with their customers.

In our modified heuristic, we pre-process the input to eliminate the reassignment of the long-term flows that are still active without any changes on their tuples. In doing so, first, we look at the long-term flows that are heading to the same destination and are still active for the current monitoring time-window ($InterSect$). The comparison of the DDoS flooding attack coverage for the past two monitoring time-windows for such flows based on the predefined threshold (ϵ), as explained in 4.12, determines if such flows should be reassigned or their

current assignment suffices. Next, we perform the same monitoring assignment for the rest of the flows like our previously presented heuristic. We create set B and sort it in descending order based on the size of its elements (i.e., x_i) for various destinations (i.e., the largest $|S_i|$ will be on top of the sorted list). For each element $x_i \in B$ and for each of the routers on $R(x_i)$, we calculate $ReqMem_{x_i,r}$ through the formulation we presented in section 4.2.1. Then, we assign all the flows that belong to x_i to one of the routers that belongs to $R(x_i)$. Next, for the rest of the flows, we calculate $ReqMem(f,r)$ for each of the routers on their p_f. Then, we first assign flows that only have one router on their p_f to ensure their coverage. Finally, we assign the rest of the flows (as many as possible) to one of the routers on their p_f. Algorithm 2 is the pseudo-code of the modified heuristic algorithm that runs on each TAS.

Algorithm 2 Modified Heuristic with Input Pre-processing Capability

Inputs: m_r, $\mathcal{F}$, $\mathcal{F}^{t_{n-1}}$, $\mathcal{FD}_d$, $\mathcal{F}_r^{t_{n-1}}$, $DDoSCov_d^{t_{n-2}}$, $DDoSCov_d^{t_{n-1}}$, $\epsilon \geq 0$

Output: $Assigned, Covered, \mathcal{F}^{t_{n-1}}, \mathcal{F}_r^{t_{n-1}}, DDoSCov_d^{t_{n-2}}$, $DDoSCov_d^{t_{n-1}}$, sending out the routers' responsibility lists

$Assigned = 0$
$Covered = \emptyset$, $InterSect = \emptyset, \mathcal{F}_r = \emptyset$
——— **Preprocessing The Input** ———
if $\mathcal{F}^{t_{n-1}} \neq \emptyset$ **then**
  Initialize $InterSect$ as $\mathcal{F}^{t_{n-1}} \cap \mathcal{F}$
  for all $f \in InterSect$ **do**
    if $DDoSCov_{d_f}^{t_{n-2}} - DDoSCov_{d_f}^{t_{n-1}} \leq \epsilon$ **then**
      for all $S_i \in \mathcal{FD}_{d_f}$ **and** $f \in S_i$ **do**
        $Assigned_R = 1$
        for all $r \in R(S_i)$ **and** $f \in \mathcal{F}_r^{t_{n-1}}$ **and** $Assigned_R = 1$ **do**
          if $S_i \cap Covered = \emptyset$ **then**
            Update r's capacity ⇒ using 4.1
            $Assigned = Assigned + |S_i|$
            $Covered = Covered \cup S_i$
            $DDoSCov_{d_f}^{t_{n-1}} = DDoSCov_{d_f}^{t_{n-1}} \cup S_i$
            $\mathcal{F}_r = \mathcal{F}_r \cup S_i$
            $Assigned_R = 0$
          Remove S_i from $\mathcal{FD}_{d_f}$
          Remove all $f \in S_i$ from $\mathcal{F}$ and $InterSect$
  for all $f \in InterSect$ **do**
    $Assigned_R = 1$
    for all $r \in p_f$ **and** $f \in \mathcal{F}_r^{t_{n-1}}$ **and** $Assigned_R = 1$ **do**
      Update r's capacity ⇒ using 4.1
      $Assigned++$
      $Covered = Covered \cup f$
      $\mathcal{F}_r = \mathcal{F}_r \cup f$
      $Assigned_R = 0$
    Remove f from $\mathcal{F}$ and $InterSect$
Generate $B = \bigcup_{d \in \mathcal{D}} \mathcal{FD}_d$
if $\mathcal{F} \neq \emptyset$ **then**
——— **Monitoring Assignment** ———
Sort B by the size of its x_is in a descending order
$DDoSCov_d^{t_{n-2}} \longleftarrow DDoSCov_d^{t_{n-1}} \quad \forall d \in \mathcal{D}$
for all $x_i \in B$ **and** $x_i \cap Covered = \emptyset$ **do**
  $Assigned_R = 1$
  for all $r \in R(x_i)$ **and** $Assigned_R = 1$ **do**
    if $m_r \geq ReqMem_{x_i,r}$ **then**
      Assign flows in x_i to router r
      Update r's capacity ⇒ using 4.1
      $Assigned = Assigned + |x_i|$
      $Covered = Covered \cup x_i$
      $\mathcal{F}_r = \mathcal{F}_r \cup f$
      $Assigned_R = 0$
      Remove all $f \in x_i$ from $\mathcal{F}$
for all $f \in \mathcal{F}$ **do**
  if $|p_f| = 1$ **and** $m_{r \in p_f} \geq ReqMem_{f,r}$ **then**
    Assign flow f to router r
    Update r's capacity ⇒ using 4.1
    $Assigned++$
    $Covered = Covered \cup f$
    $\mathcal{F}_r = \mathcal{F}_r \cup f$
    Remove f from $\mathcal{F}$
for all $f \in \mathcal{F}$ **do**
  $Assigned_R = 1$
  for all $r \in p_f$ **and** $Assigned_R = 1$ **do**
    if $m_r \geq ReqMem_{f,r}$ **then**
      Assign flow f to router r
      Update r's capacity ⇒ using 4.1
      $Assigned++$
      $Covered = Covered \cup f$
      $\mathcal{F}_r = \mathcal{F}_r \cup f$
      $Assigned_R = 0$
$\mathcal{F}^{t_{n-1}} = Covered$
$\mathcal{F}_r^{t_{n-1}} = \mathcal{F}_r \quad \forall r \in \mathcal{R}$
$Calculate \quad DDoSCov_d^{t_{n-1}} = \frac{\#f \in Covered | d_f = d}{\#f \in TempF | d_f = d} \quad \forall d \in \mathcal{D}$

4.4.1 Experimental set-up:

In order to compare the performance of our proposed modified heuristic with our original heuristic, which is equivalent to our modified heuristic with the uncovered DDoS flooding flow tolerance equal to zero ($\epsilon = 0$), we perform an experiment that we explain in detail here in this section.

In this experiment, we choose AT&T (US) topology with $\mathcal{R} = 540$ with initial $\mathcal{F} = 11 \times 10^6$ with $\mathcal{D} = 9 \times 10^6$ which is one of the challenging problem sizes in our previous experimental results in Table 4.2. We perform an hour long simulation and gradually increase the size of the problem (number of flows and destination IP addresses) to $\mathcal{F} = 15 \times 10^6$ and $\mathcal{D} = 11 \times 10^6$. In this set-up we assume destination X is the DDoS flooding attack target for which there is initially 10^6 flows with that destination IP at the start time. This amount doubles up to 2×10^6 at minute 20 and increases again to 3×10^6 at minute 40 in our simulation set-up. For the rest of the flows, we randomly generate tuples (s_f, d_f, p_f) and calculate $ReqMem_{f,r}$ for all the router in p_f for each f. Moreover, we generate the flow's tuples in such a way that for the half of all the destination IP addresses of the flows on each monitoring time window, there is at least two flows going to each of those destination IP addresses for which there is at least one router in common on their paths. Then, we generate the rest of the required inputs that are introduced in section 4.2.4. We use NS-2 [170] to generate these topologies with the desired traffic inputs on them. We restate here that all of the experiments are bounded to 4-minute time limit as the monitoring time window is assumed to be 5 minutes and responsibility lists should be made available to all the routers within the monitoring time window (< 5 minutes). This set-up is used with our MIP formulation as the monitoring assignment and it finds an optimal solution within the time limit or the best solution it can obtain within the time-limit. Both optimal and non-optimal solutions are recorded to compare with our proposed heuristics. We also employ both our heuristic and its modified version as the monitoring assignment mechanisms and record their flow coverage with their computation time.

We compare the MIP formulation, the heuristic algorithm, and the modified heuristic algorithm [8] in terms of their total flow coverage. Moreover, we compare the heuristic and the modified heuristic in terms of their computation time and skip the MIP formulation's computation time since in all of the scenarios for all the heuristics perform much better than the MIP in terms of their computation time. Finally, we compare the heuristic and the modified heuristic in terms of their DDoS flooding flow coverage and exclude the MIP formulation's results since it covers all of the DDoS flooding flows that original heuristic covers. Our comparisons are performed in two different scenarios. In the first scenario, we assume that the initially generated 11×10^6 flows and their destinations (including initial 1×10^6 flows heading to destination X) are not changed for the duration of the experiment as the representation of the scenario in which most of the flows are long-term. This scenario is where the elimination of the monitoring responsibility reassignment reduces the whole assignment computation time the most. In the second scenario, we assume that half of the initially generated flows and their paths (including initial 1×10^6 flows heading to destination X) are regenerated every 10 minutes representing a more dynamic scenario in which most of the flows are short-term.

4.4.2 Experimental results:

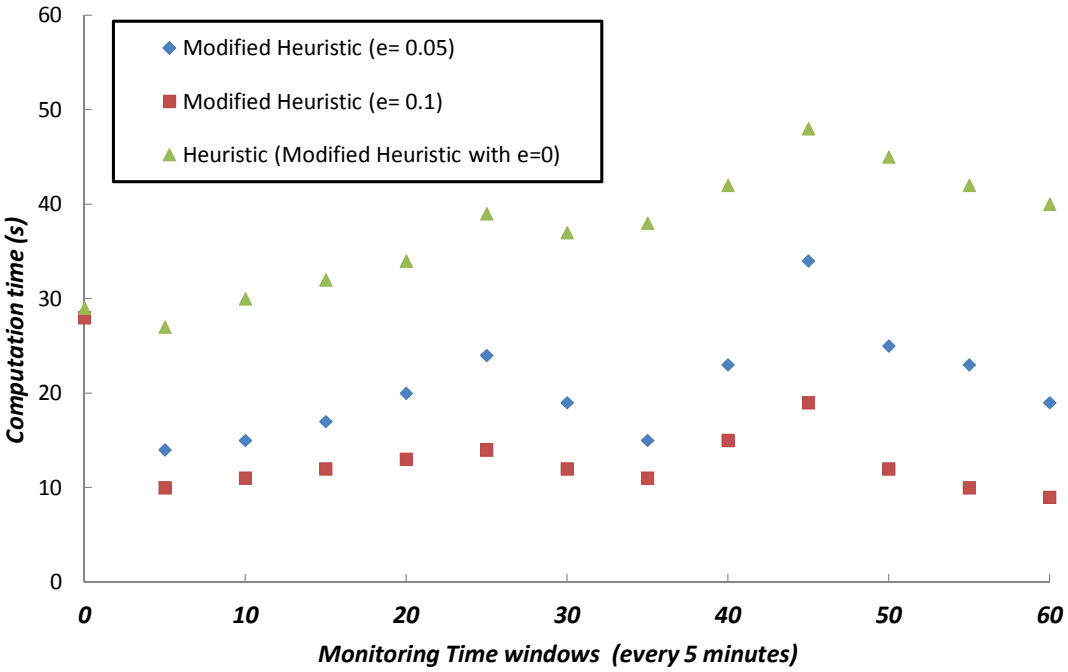
As we mentioned earlier, we expect our experimental results to be convincing in highlighting the benefits that our modified heuristic brings to the table in reducing the time it takes for the assignment computation by tolerating a reduction on the DDoS flooding flow coverages. The performance of all our experiments is bounded by the resources of a desktop PC system with *Intel Core 2 Duo E6700 2.66 GHz* CPU that we used in our experiments.

Figure 4.3 shows the monitoring assignment computation time for both the short-term and long-term scenarios. In Figure 4.3a, where the majority of the flows are long-term, computation time significantly decreases in cases where the modified heuristic is employed. However, in Figure 4.3b, where the majority of the flows are short-term, computation time is higher than the long-term flow scenario since there are more new flows as inputs for

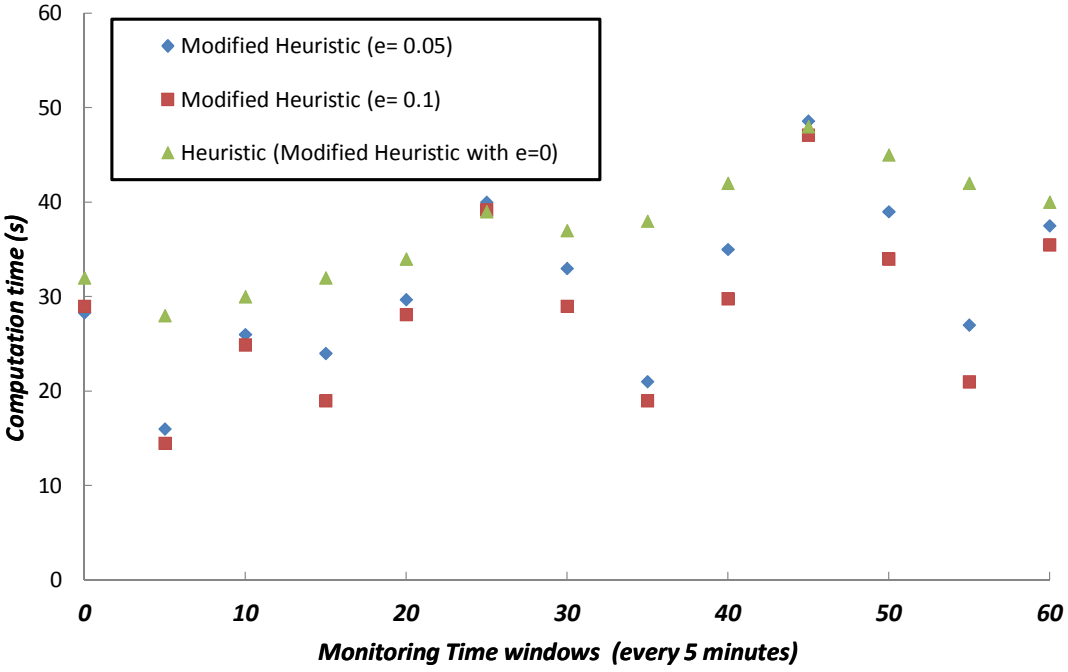
[8] with $\epsilon = 0.05$ and $\epsilon = 0.1$

the monitoring assignment process which increases the size of the problem. Moreover, in both of the scenarios, the higher the uncovered DDoS flooding flow tolerance is the lower the computation time of the monitoring assignment will be. This is shown in both figures through the comparison between two modified heuristics with different uncovered DDoS flooding flow tolerance ($\epsilon = 0.05$ and $\epsilon = 0.1$). The one with the greater uncovered DDoS flooding flow tolerance has the lower computation time than the one with lower uncovered DDoS flooding flow tolerance.

Figure 4.4 shows the flow coverage for DDoS flooding flows heading to destination X for both the short-term and long-term scenarios. In both of the figures, our original heuristic covers 100% of the DDoS flooding flows to destination X as it performs the monitoring assignment for all the flows and for every monitoring time windows. Hence, we compare the performance of our modified heuristic with the original heuristic in terms of the DDoS flooding flow coverage in order to show the sacrifices in terms of DDoS flooding flow coverage that service providers should be willing to tolerate to reduce the monitoring assignment computation time. In Figure 4.4a, where the majority of the flows are long-term, DDoS flooding flow coverage decreases at minute 20 where the number of DDoS flooding flows to destination X doubles up. In minute 25, none of the modified heuristics with different uncovered DDoS flooding flow tolerances passes its uncovered DDoS flooding flow tolerance threshold comparing their DDoS flooding flow coverages for the two previous monitoring time windows. This reduction in DDoS flooding flow coverage continues until minute 40 that the number of DDoS flooding flows heading to destination X increases again. At minute 40, there is another reduction in DDoS flooding flow coverage which causes the modified heuristic with lower uncovered DDoS flooding flow tolerance to perform the reassignment for all the flows heading to destination x at minute 45. This causes a significant increase in DDoS flooding flow coverage for that modified heuristic as it is shown. However, the modified heuristic with the higher uncovered DDoS flooding flow tolerance still has low DDoS flooding flow coverage.

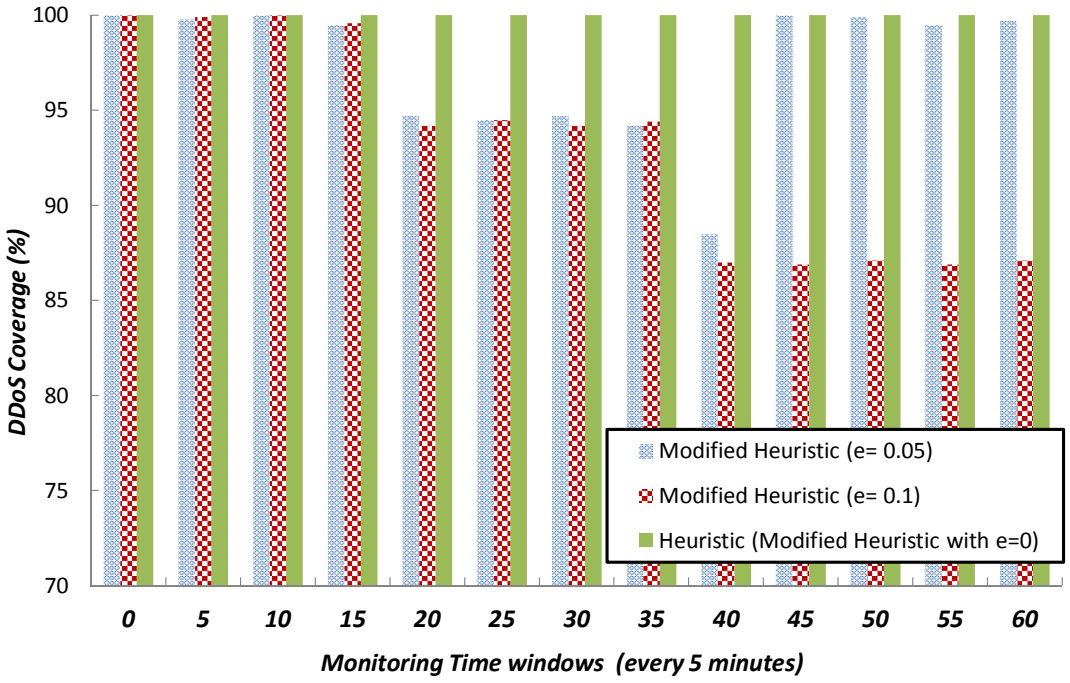


(a) Long-term static flows

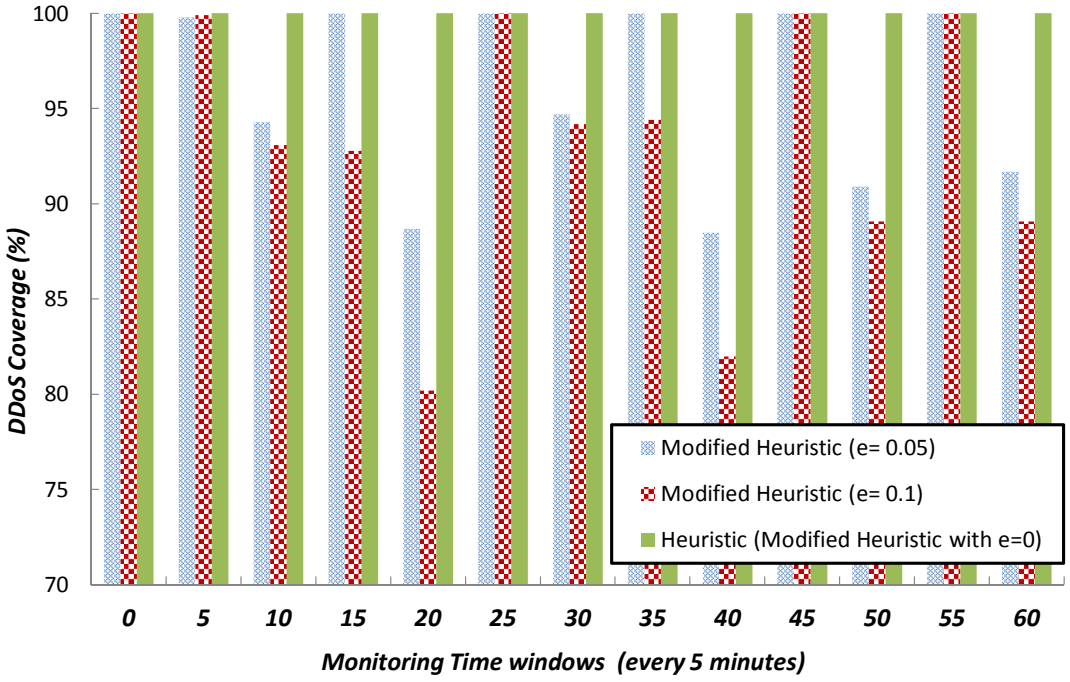


(b) Short-term dynamic flows

Figure 4.3: Monitoring assignment computation time.

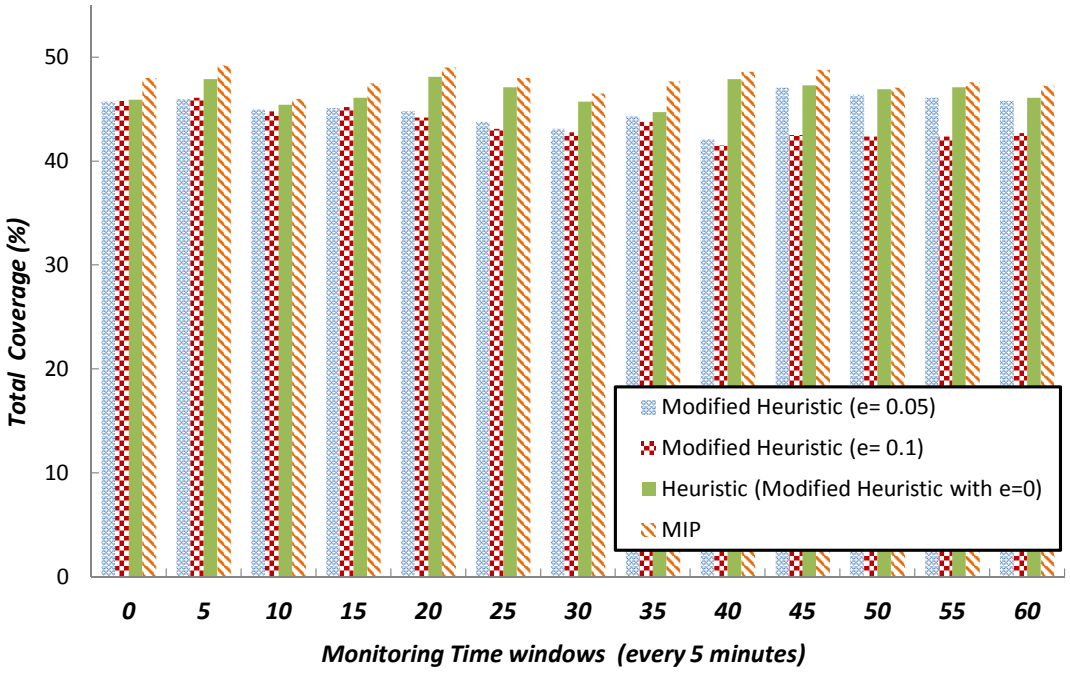


(a) Long-term static flows

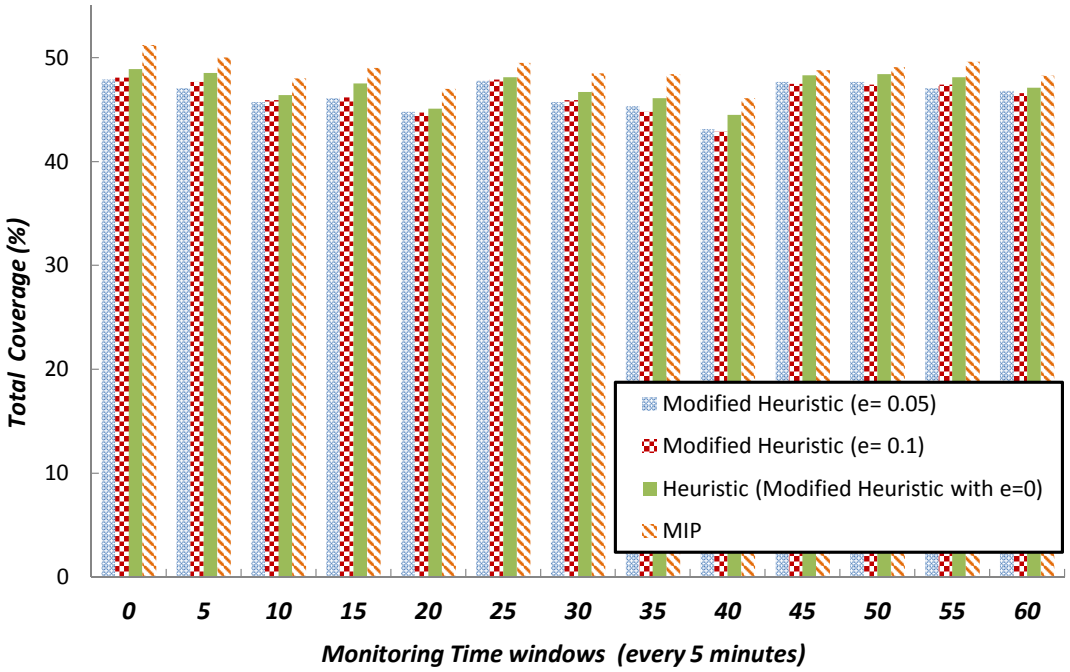


(b) Short-term dynamic flows

Figure 4.4: DDoS flooding flow coverage.



(a) Long-term static flows



(b) Short-term dynamic flows

Figure 4.5: Total flow coverage.

However, in Figure 4.4b, where the majority of the flows are short-term and more dynamic in nature, there will be more situations where the monitoring assignment is performed for most of the flows heading to destination X, which both increases the monitoring assignment computation time (as it was shown in Figure 4.3b) and the DDoS flooding flow coverage. Moreover, the modified heuristic with $\epsilon = 0.1$ covers less DDoS flooding flows as it tolerates the absence of more DDoS flooding flows.

Figure 4.5 shows the total flow coverage for both the short-term and long-term scenarios. Comparing Figure 4.5a and Figure 4.5b, the total flow coverage for the modified heuristic with different uncovered DDoS flooding flow tolerances is really close to the original heuristic (closer to optimal) in the short-term flow scenario compared to the long-term flow scenario. This is mainly because of more monitoring reassignments due to more new flows (i.e., less pre-processing).

4.5 DiCoTraM vs. other monitoring mechanisms

The main network-wide objective of DiCoTraM is to maximize the flow coverage while considering both the routers' resource limitations and satisfying the maximized DDoS flooding attack flow coverage. Hence, in this section, we compare DiCoTraM with other traditional packet/flow sampling mechanisms in terms of the total flow coverage and DDoS flooding attack flow coverage to highlight the benefits of DiCoTraM over other previously proposed traffic monitoring mechanisms.

4.5.1 Total flow coverage

In doing so, we conduct a series of simulation experiments on 5 well-known topologies (the same experiments we performed in our previously explained case study for cSamp). The parameters of these five topologies that are employed in our experiments are previously explained in subsection 3.3.1 and they are shown in Table 3.1.

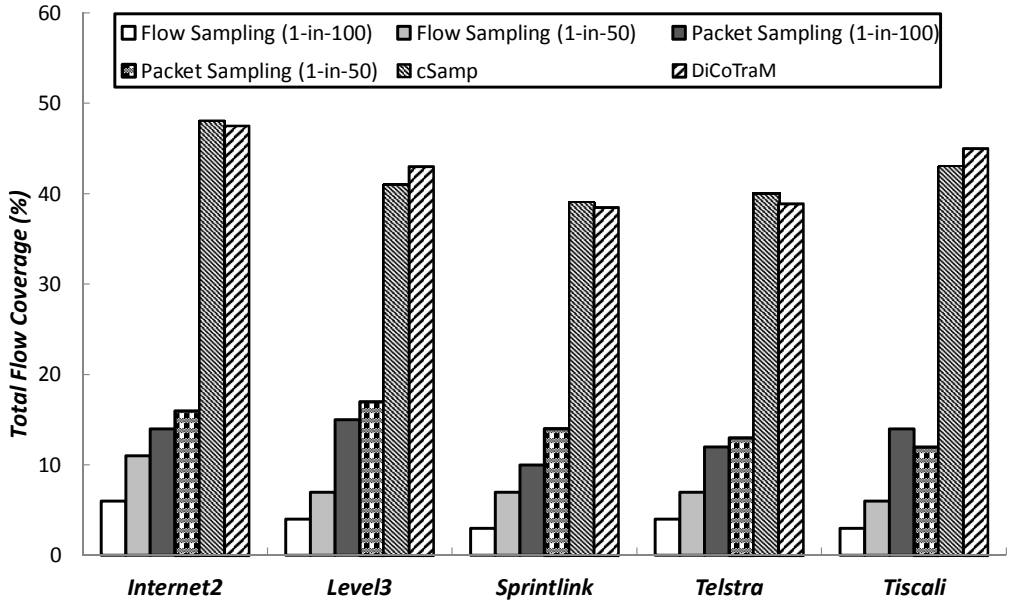


Figure 4.6: Total Flow Coverage: DiCoTraM vs. other packet/flow sampling mechanisms

As we mentioned earlier in subsection 3.3.1, we have implemented cSamp (validated our implementation) to compare with DiCoTraM [9]. We have compared DiCoTraM with: cSamp, uniform packet sampling with the sampling rate of 1-in-100 packets, uniform packet sampling with the sampling rate of 1-in-50 packets, constant-rate flow sampling with the sampling rate of 1-in-100 flows, and constant-rate flow sampling with the sampling rate of 1-in-50 flows.

In our simulation, for all the flow/packet sampling mechanisms, for each topology, we use the same: traffic matrix, routing information, number of flows, and flow size distribution [10]. For each topology, we compute the total flow coverage obtained with different packet sampling or flow sampling mechanisms and the results are shown in Figure 4.6. The total flow coverage of DiCoTraM is very close to cSamp for *Level3* and *Tiscali*, and it is greater than cSamp for the rest of the topologies. cSamp and DiCoTraM both have larger total flow coverages than either of other packet/flow sampling mechanisms. As it was the case for cSamp and it is true in case of DiCoTraM, other packet/flow sampling mechanisms suffer from the wasted amount of redundant flows/packets that are monitored (i.e., redundant flow reports and wasted bandwidth to report them) which was shown in Figure 3.3b. The central

[9] We used the modified heuristic with $\epsilon = 0.05$

[10] We, also like [136], assume that the flow size measured in number of packets is Pareto-distributed.

coordination structure in both DiCoTraM and cSamp eliminates the redundant flow/packet monitoring overhead.

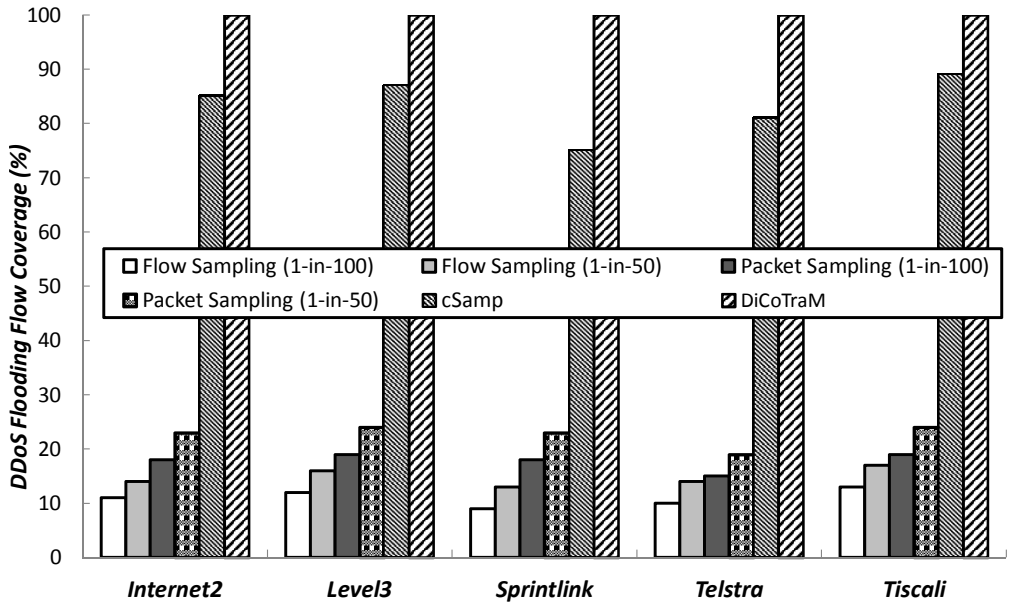


Figure 4.7: DDoS flooding flow coverage: DiCoTraM vs. other packet/flow sampling mechanisms

4.5.2 DDoS flooding attack flow coverage

In order to show the major benefit of DiCoTraM in providing the higher DDoS flooding attack flow coverage, in the same simulation set-up and for each topology, we generate 1×10^6 flows destined for the same destination (possible DDoS flooding attack flows) and distribute them along various paths towards the destination. The number of flows heading to the same destination is chosen carefully so that it is possible for the monitoring mechanisms to cover all of the flows considering the routers' capacities and other constraints. Then, we compare DiCoTraM with other traffic monitoring mechanisms in terms of their coverages of 1×10^6 flows destined to the same destination. The higher the flow coverage for the flows destined for the same destination, the better suited is the monitoring mechanism for DDoS flooding

defense mechanisms. Figure 4.7 shows the results. As we expected, DiCoTraM covers all of the DDoS flooding flows and outperforms all other traffic monitoring mechanisms.

4.5.3 Discussion:

As we mentioned earlier, one of the ideal traffic monitoring objectives to achieve in most of the traffic monitoring mechanisms is to cover as many attack flows as possible in order to provide a clearer picture of the network status for different traffic analysis/attack detection applications. In case of DDoS flooding attack flow coverage, we argue that it is fair to compare different traffic monitoring mechanisms in terms of their success in providing the best DDoS flooding flow coverage. Since DiCoTraM is tailored in such a way that the DDoS flooding flow coverage is prioritized and maximized, it outperforms all other traffic monitoring mechanisms in terms of DDoS flooding flow coverage without sacrificing too much in its performance in terms of total flow coverage compared with other traffic monitoring mechanisms.

4.6 Network planning & memory requirements

As we pointed out in subsection 4.2.1, the total flow coverage in any autonomous system or enterprise network is bounded by the memory capacities (constraints) of the monitoring devices. Hence, all of the network traffic monitoring mechanisms (e.g., cSamp, DiCoTraM) consider the memory constraints of the monitoring devices. Moreover, we have made it clear that increasing the flow monitoring coverage leads to providing a clearer picture of what is happening inside the network to the detection mechanisms and consequently leads to a better protection for enterprise networks against dangerous attacks such as DDoS flooding attacks.

Nowadays, memory modules are much cheaper than what they were a decade ago; hence, adding reasonable amount of additional memory to the monitoring devices to increase the flow coverage of an enterprise network is feasible. Hence, upgrading some of the heavily

loaded (bottleneck) routers that lead to low flow coverage could be considered as a reasonable network planning strategy. However, the amount of required additional memory for each router is not fixed and it should be determined dynamically according to the structure of the network (e.g., topology, traffic history, and etc.). Moreover, the bottleneck routers that caused the low flow coverage also needs to be determined dynamically according to the structure of the network (e.g., topology, traffic history, and etc.).

Therefore, in order to provide the network administrators with such valuable information for their short-term or long-term network planning, we propose an off-line MIP formulation that relaxes the memory constraints of the routers within an AS and calculates the maximum amount of additional memory required for each router in order to cover all its assigned flows. The bottleneck routers, as we defined earlier, are among the routers with the maximum required additional memories. The goal of the proposed MIP is not to suggest an immediate or dynamic change of the network structure to cover all the flows. However, as we mentioned earlier, we strongly believe that the information that our off-line MIP could provide to the network administrators can help them in their short-term/long-term network planning and in determining the bottleneck routers that cause poor traffic monitoring coverage. For instance, in our experiments in this chapter, we show that increasing the memory capacity of (updating) the short percentage of heavily loaded routers, that are determined as the bottleneck routers, with their maximum required additional memory during days, weeks, or months can lead to an increased total traffic monitoring coverage which is an acceptable strategy within an enterprise network.

The amount of additional required memory for each router can be queried by ASs' administrators at any given time or it can be reported to them upon passing some predefined thresholds (e.g., Total flow monitoring coverage, DDoS flooding attack monitoring coverage). We assume that administrators query the system for the list of required additional memories. As we mentioned earlier, we perform some experiments to show that the amount of additional memory is not significantly large and upgrading a few heavily loaded routers maybe sufficient and feasible for service providers as a short-term adjustment.

At the beginning of each monitoring time window, TASs run our previously proposed MIP formulation for monitoring assignment presented in section 4.2.5 to assign monitoring

responsibilities to the routers within an AS. If the current memory capacity of some of the routers are not sufficient enough to assign flow monitoring responsibilities to them, TASs run our second off-line MIP which relaxes the memory constraints of the routers and keep the maximum required memory for each of the routers in a list and update them after each run of the MIP formulation.

TASs keep a list of maximum additional required memory for all the routers. Each time TASs run the off-line MIP, they update the list for each router if the amount of memory requirement for that router should be increased more than its current amount based on the outcome of the current MIP run (i.e, the maximum amount of additional memory for each router is maintained on this list).

ASs' administrators will be provided with the updated list of the maximum additional memories of all the routers upon request or whenever the amount of additional required memory (μ_r) passes some predefined threshold for at least one heavily loaded router. Next we present our proposed off-line MIP formulation.

4.6.1 Off-line MIP formulation

Given the notation and definitions (summarized in 4.2.3), the problem of finding the minimum additional memories for each router can be formulated as follows:

$$\min_{\mu_r} \sum_{r \in \mathcal{R}} \mu_r \tag{4.13}$$

subject to,

$$\sum_{d \in \mathcal{D}} \sum_{S_i \in \mathcal{FD}_d} ReqMem_{S_i,r} \cdot x_{S_i,r,d} + \sum_{f \in \mathcal{F}} ReqMem_{f,r} \cdot y_{f,r} - (m_r + \mu_r) \leq 0 \qquad \forall r \in \mathcal{R} \tag{4.14}$$

$$\sum_{S_i \in \mathcal{FD}_d} \sum_{r \in R(S_i)} x_{S_i,r,d} \geq 1 \qquad \forall d \in \mathcal{D} \tag{4.15}$$

$$\sum_{r \in R(S_i)} x_{S_i,r,d} = 1 \qquad \forall d \in \mathcal{D},\ \forall S_i \in \mathcal{FD}_d \qquad (4.16)$$

$$\sum_{r \in \mathcal{R}} y_{f,r} = 1 \qquad \forall f \in \mathcal{F} \qquad (4.17)$$

$$\sum_{S_i \in \mathcal{FD}_d} \sum_{r \in R(S_i)} \alpha_{f,S_i} \cdot x_{S_i,r,d} = 1 \qquad \forall d \in \mathcal{D},\ \forall f \in \mathcal{F} \qquad (4.18)$$

$$1 - \sum_{r \in \mathcal{R}} y_{f,r} \geq \sum_{d \in \mathcal{D}} \sum_{S_i \in \mathcal{FD}_d} \sum_{r \in R(S_i)} \alpha_{f,S_i} \cdot x_{S_i,r,d} \qquad \forall f \in \mathcal{F} \qquad (4.19)$$

$$x_{S_i,r,d} \in \{0,1\} \qquad \forall S_i \in \mathcal{FD}_d, \forall r \in R(S_i), \forall d \in \mathcal{D} \qquad (4.20)$$

$$y_{f,r} \in \{0,1\} \qquad \forall r \in \mathcal{R},\ \forall f \in \mathcal{F} \qquad (4.21)$$

The objective function in (4.13) is to find the minimum sum of additional required memories for all the routers. The set of constraints in (4.14) ensures that both (i) the flows intended to the same destination are assigned to one of their common routers on their paths, and (ii) the rest of the flows are assigned to the rest of the routers relaxing the capacity constraints of the routers. In (4.14), we assume that there is one aggregated flow entry (S_i) for all the flows intended to the same destination IP. The set of constraints in (4.15) ensures that at least one S_i is selected among all the S_is that are in $\mathcal{FD}_d$ for all the destinations. The set of constraints (4.16) and (4.17) ensures that S_is that are in $\mathcal{FD}_d$ for all the destinations and the rest of the flows are assigned to at least one router, respectively. The set of constraints in (4.18) forbids the selection of two distinct S_is that consists of common flows for all the destinations. The set of constraints in (4.19) implies that one flow is going to be an independent flow unless it is covered in one of the selected S_is for all the flows. Finally, the set of constraints in (4.20) and (4.21) define $x_{S_i,r,d}$ and $y_{f,r}$ as binary variables.

Without loss of generality, we assume that for each destination ($d \in \mathcal{D}$) there exist a collection of flows (S_i) and a common router (r) on the paths of flows in S_i such that the memory capacity of router r is enough to fulfil the memory requirements of all the flows in S_i.

We could reduce the known NP-complete problem of PARTITION [168] to the above MIP formulation in (4.13)-(4.21). Hence, the proposed MIP is a NP-hard problem.

4.6.2 The modified heuristic with network planing capability

Since our proposed MIP formulation for network planning is also solving a NP-hard problem, it does not scale well. Hence, we further change our modified heuristic (Algorithm 2 in section 4.4) to calculate the amount of additional required memory needed for network planning purposes, presented in Algorithm 3.

Our modified heuristic with network planing capability keeps track of the maximum additional memory requirement of the monitoring devices to cover all the flows and also provides the list of routers with the maximum additional memory requirements as the bottleneck routers during a desired time period (e.g., day, month, etc.). Next, we perform a set of experiments to evaluate the network planning capability of such information in scenarios with or without DDoS flooding attack to show how network administrators could benefit from such information in their short-term/long-term network planning.

Algorithm 3 Modified heuristic algorithm with the network planning capability

Inputs: m_r, $\mathcal{F}$, $\mathcal{F}^{t_{n-1}}$, $\mathcal{FD}_d$, $\mathcal{F}_r^{t_{n-1}}, DDoSCov_d^{t_{n-2}}$, $DDoSCov_d^{t_{n-1}}, \epsilon \geq 0, \mu_r^{Max}$

Output: $Assigned, Covered, \mathcal{F}^{t_{n-1}}, \mathcal{F}_r^{t_{n-1}}, DDoSCov_d^{t_{n-2}}$, $DDoSCov_d^{t_{n-1}}, \mu_r^{Max}$, sending out the routers' responsibility lists

$Assigned = 0$, $Additional = 0$
$Covered = \emptyset$, $InterSect = \emptyset, \mathcal{F}_r = \emptyset$
——— **Preprocessing The Input** ———
if $\mathcal{F}^{t_{n-1}} \neq \emptyset$ **then**
  Initialize $InterSect$ as $\mathcal{F}^{t_{n-1}} \cap \mathcal{F}$
  for all $f \in InterSect$ **do**
    if $DDoSCov_{d_f}^{t_{n-2}} - DDoSCov_{d_f}^{t_{n-1}} \leq \epsilon$ **then**
      for all $S_i \in \mathcal{FD}_{d_f}$ **and** $f \in S_i$ **do**
        $Assigned_R = 1$
        for all $r \in R(S_i)$ **and** $f \in \mathcal{F}_r^{t_{n-1}}$ **and** $Assigned_R = 1$ **do**
          if $S_i \cap Covered = \emptyset$ **then**
            Update r's capacity $\Rightarrow$ using 4.1
            $Assigned = Assigned + |S_i|$
            $Covered = Covered \cup S_i$
            $DDoSCov_{d_f}^{t_{n-1}} = DDoSCov_{d_f}^{t_{n-1}} \cup S_i$
            $\mathcal{F}_r = \mathcal{F}_r \cup S_i$
            $Assigned_R = 0$
          Remove S_i from $\mathcal{FD}_{d_f}$
          Remove all $f \in S_i$ from $\mathcal{F}$ and $InterSect$
  for all $f \in InterSect$ **do**
    $Assigned_R = 1$
    for all $r \in p_f$ **and** $f \in \mathcal{F}_r^{t_{n-1}}$ **and** $Assigned_R = 1$ **do**
      Update r's capacity $\Rightarrow$ using 4.1
      $Assigned++$
      $Covered = Covered \cup f$
      $\mathcal{F}_r = \mathcal{F}_r \cup f$
      $Assigned_R = 0$
    Remove f from $\mathcal{F}$ and $InterSect$
Generate $B = \bigcup_{d \in \mathcal{D}} \mathcal{FD}_d$
if $\mathcal{F} \neq \emptyset$ **then**
——— **Monitoring Assignment** ———
  Sort B by the size of its x_is in a descending order
  $DDoSCov_d^2 \longleftarrow DDoSCov_d^1 \quad \forall d \in \mathcal{D}$
  for all $x_i \in B$ **and** $x_i \cap Covered = \emptyset$ **do**
    $NotAssigned = 0$
    $Assigned_R = 1$
    for all $r \in R(x_i)$ **and** $Assigned_R = 1$ **do**
      if $m_r \geq ReqMem_{x_i,r}$ **then**
        Assign flows in x_i to router r
        Update r's capacity $\Rightarrow$ using 4.1
        $Assigned = Assigned + |x_i|$
        $Covered = Covered \cup x_i$
        $\mathcal{F}_r = \mathcal{F}_r \cup f$
        $Assigned_R = 0$
        Remove all $f \in x_i$ from $\mathcal{F}$
        $NotAssigned = 1$
    if $NotAssigned = 0$ **then**
      $Additional = Additional + |x_i|$
      for all $r \in R(x_i)$ **do**
        $\mu_r = \mu_r + ReqMem_{x_i,r}$
  for all $f \in \mathcal{F}$ **do**
    $NotAssigned = 0$
    if $|p_f| = 1$ **and** $m_{r \in p_f} \geq ReqMem_{f,r}$ **then**
      Assign flow f to router r
      Update r's capacity $\Rightarrow$ using 4.1
      $Assigned++$
      $Covered = Covered \cup f$
      $\mathcal{F}_r = \mathcal{F}_r \cup f$
      Remove f from $\mathcal{F}$
      $NotAssigned = 1$
    if $NotAssigned = 0$ **then**
      $\mu_{r \in p_f} = \mu_{r \in p_f} + ReqMem_{f,r}$
      $Additional++$
  for all $f \in \mathcal{F}$ **do**
    $Assigned_R = 1$
    $NotAssigned = 0$
    for all $r \in p_f$ **and** $Assigned_R = 1$ **do**
      if $m_r \geq ReqMem_{f,r}$ **then**
        Assign flow f to router r
        Update r's capacity $\Rightarrow$ using 4.1
        $Assigned++$
        $Covered = Covered \cup f$
        $\mathcal{F}_r = \mathcal{F}_r \cup f$
        $Assigned_R = 0$
        $NotAssigned = 1$
    if $NotAssigned = 0$ **then**
      $Additional++$
      for all $r \in p_f$ **do**
        $\mu_r = \mu_r + ReqMem_{f,r}$
  $\mathcal{F}^{t_{n-1}} = Covered$
  $\underset{\forall r \in \mathcal{R}}{\mathcal{F}_r^{t_{n-1}}} = \mathcal{F}_r$
  $\underset{\forall d \in \mathcal{D}}{Calculate} \quad DDoSCov_d^{t_{n-1}} = \dfrac{\#f \in Covered | d_f = d}{\#f \in TempF | d_f = d}$
  for all $r \in \mathcal{R}$ **do**
    if $\mu_r > \mu_r^{Max}$ **then**
      $\mu_r^{Max} = \mu_r$

4.6.3 Experimental set-up & results:

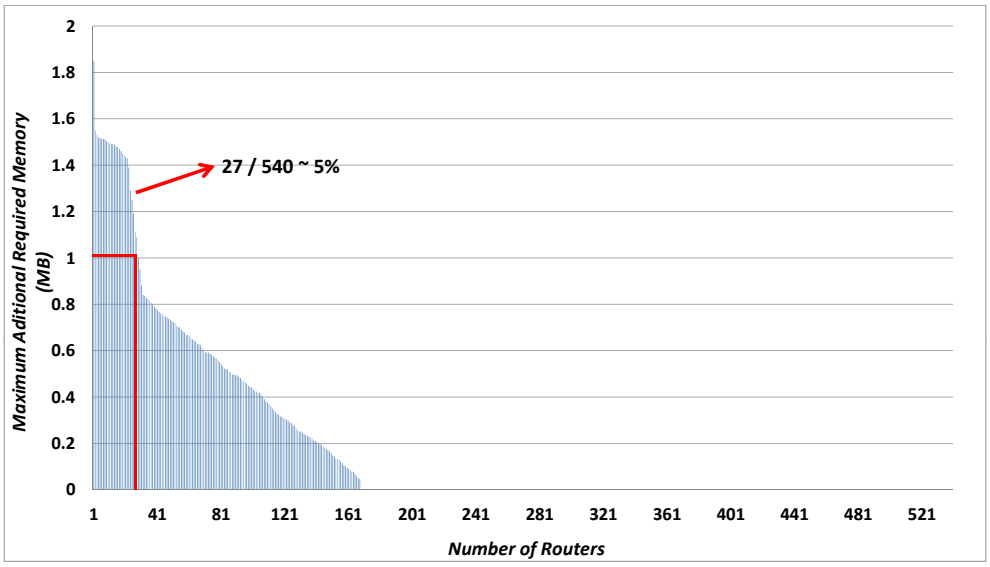
In this section, we perform two sets of experiments employing the AT&T (US) topology with $\mathcal{R} = 540$. We generate two simulation set-ups. First, we initially add $\mathcal{F} = 11 \times 10^6$ with $\mathcal{D} = 9 \times 10^6$ and increase the number of flows gradually within 24 hours to $\mathcal{F} = 55 \times 10^6$ and $\mathcal{D} = 45 \times 10^6$. The number of flows and destination IP addresses are increased to values close to their maximum estimated values as we explained in Table 4.1 and as a worse-case scenario in terms of the size of the problem or the number of distinct traffic flows in the AT&T (US) topology. We run this simulation set-up with the modified heuristic with the network planning capability (with $\epsilon = 0.05$), explained above, for 24 hours. We query the list of maximum additional memories for the routers within this topology every hour. Hence, for each router, we have 24 values as its maximum additional required memory for each hour. We also keep the total flow coverage for the duration of the simulation. Moreover, we keep the generated inputs (every 5 minutes) of the monitoring assignment heuristic for this 24 hour simulation in order to have the same simulation set-up to run our other experiments with the same set-up in this section.

In our second simulation set-up, we have the same simulation set-up with an additional assumption that the destination X is under DDoS flooding attack at hour 19 and for an hour. The DDoS flooding attack starts at hour 19:00 with 10^6 flows heading to destination IP X, the number of flows doubles up to 2×10^6 at time 19:20, and the number of flows increases again to 4×10^6 at time 19:40. We run the second simulation set-up also with the same modified heuristic as our previous set-up and for 24 hour. Like our first simulation, we query the list of maximum additional memories for the routers every hour, keep the total flow coverage for 24 hours, and keep the generated inputs to the heuristic.

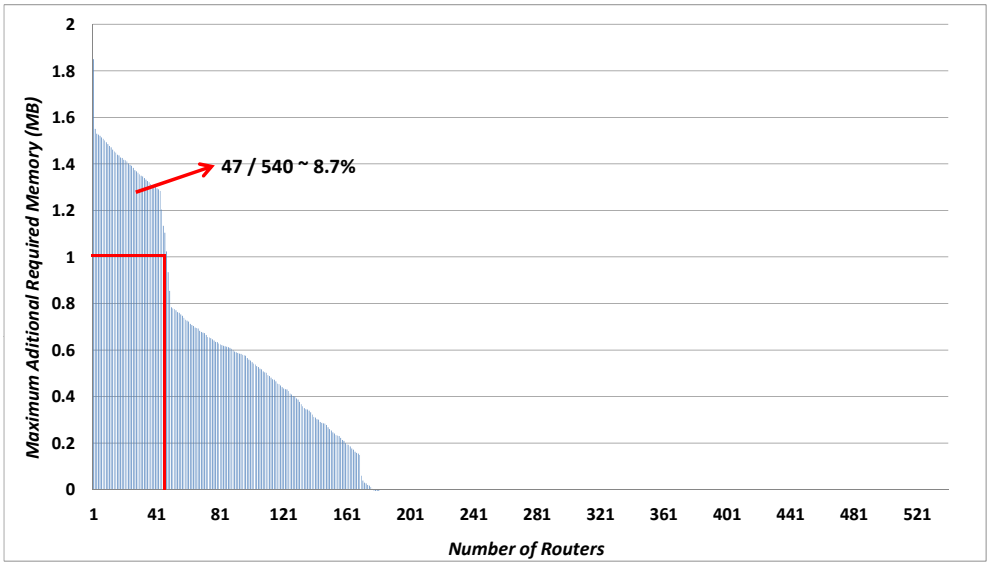
Next, for each router, we calculate the maximum of 24 queried maximum additional required memory values and sort the routers in a descending order based on their maximum additional required memory values. Figure 4.8 shows the distribution of the maximum additional required memory of the routers during the 24 hour for both of the simulation set-ups for the AT&T (US) topology and as the number of flows increases during the simulation. In Figure 4.8a, $\approx 31\%$ of the routers require additional memories and only 5% of the routers

require more than 1MB of memory. However, in Figure 4.8b, the number of routers that require additional memory increases to $\approx 33\%$ and the percentage of the routers that require more than 1MB of memory increases to 8.7% since the DDoS flooding attack on destination X causes some of the routers on the attack path to pass their capacity constraints to cover all the flows by satisfying the DiCoTraM goals.

As we mentioned earlier, tracking the additional memory requirements of the routers to monitor all the flows passing through them in different time periods (e.g., days, weeks, etc.) is useful for network planning purposes. Here, we perform another set of experiments to show that by upgrading the memory capacities of only small percentage of the routers in the At&T topology for the same experiment we could significantly increase the total flow coverage. Our experiment is based on the assumption that the routers with the maximum additional required memory are the ones that could be possibly the bottleneck routers. Hence, we rank the routers that are listed as the ones that require additional memory in a descending order and based on the amount of memory they require. Then, we repeat both of the experiments (i.e., with and without DDoS flooding attack) that we explained above by upgrading the memory capacities of top 1% and 2% of the routers with their reported largest additional required memory. We compare the total flow coverage of both of these cases (1% and 2% of routers memory capacities are upgraded) with the previous case where all the routers were bounded by their memory constraints.

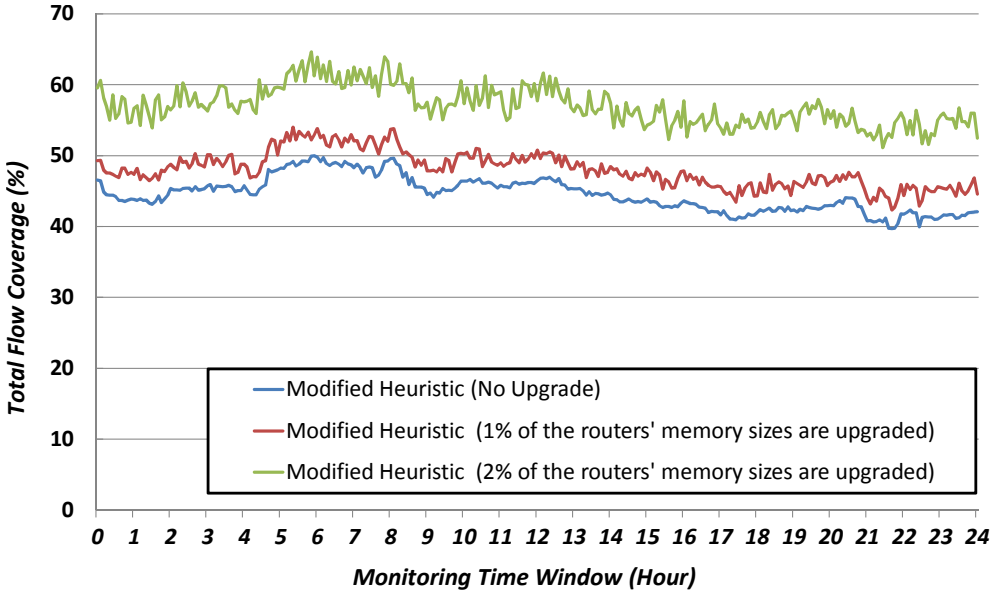


(a) Without DDoS flooding attack

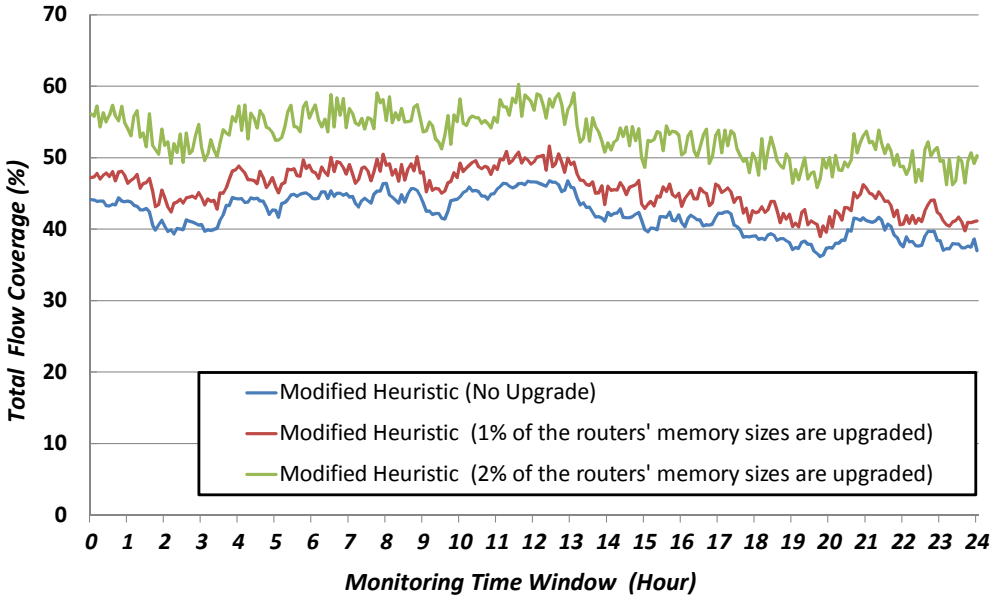


(b) Under DDoS flooding attack

Figure 4.8: Distribution of the maximum additional required memory per router for 24 hour for the AT&T topology.



(a) Without DDoS flooding attack



(b) Under DDoS flooding attack

Figure 4.9: Total flow coverage during the 24-hour for the AT&T topology.

Figure 4.9 shows the total flow coverage during the 24 hour for both of the simulation set-ups for the AT&T (US) topology and the number of flows increases during the simulation. As shown in Figure 4.9a and 4.9b, in both with and without DDoS flooding attack scenarios, upgrading a small number of monitoring devices (i.e., upgrading the memory sizes of 1% and 2% of the monitoring devices to their maximum requested memory sizes) increases the total flow coverage significantly. During our simulation, the total flow coverage increases up to 6% when only 1% of the routers' memory sizes are upgraded to their maximum requested memory sizes and it increases up to 11% when only 2% of the routers' memory sizes are upgraded to their maximum requested memory sizes. The maximum additional memory required for the routers reported in our experiment was less than 1.9 MB which sounds pretty reasonable and feasible in terms of upgradability for network operators with the current state of the art memory modules.

Hence, we show that short-term and long-term network planning considerations in order to facilitate autonomous systems in defending against dangerous attacks such as DDoS flooding attacks is possible and useful. The network planning could be managed in such a way that only small percentage of the monitoring devices are upgraded to achieve larger total flow coverages; since full flow coverage, as our current objective in calculating the list of maximum additional memories for the routers, is the worse case scenario and not all the autonomous systems necessarily require that much flow coverage.

4.7 SUMMARY

In this chapter, we presented the design and proof of concept implementation of DiCoTraM as a novel traffic monitoring mechanism to address the challenges of current traffic monitoring mechanisms with regards to the problem of DDoS flooding attack. DiCoTraM is a resource constraint aware, DDoS flooding attack tailored, network-wide traffic monitoring mechanism that centrally coordinates the monitoring responsibilities among the monitoring devices and distributes their responsibilities periodically among all the monitoring devices within each AS. DiCoTraM's coordination structure eliminates the redundant flow monitoring overheads

due to lack of coordination. Moreover, DiCoTraM enables distributed detection of DDoS flooding attacks at the router-level.

We compared and evaluated the performance of DiCoTraM with various other traffic monitoring mechanisms in terms of metrics such as: total flow coverage, redundant flow reporting, and DDoS flooding attack flow coverage in order to show its effectiveness through a set of experiments. The experimental results showed that DiCoTraM covered more DDoS flooding attack flows, and it had reasonably large overall flow coverage compared to others. Moreover, we analyzed and improved DiCoTraM's scalability through a step-by-step evolved heuristic algorithm; we performed various experiments that could confirm this claim.

Finally, in order to provide the network administrators with the required information to determine the list of candidate monitoring devices to upgrade for achieving better flow coverage as part of their short-term/long-term network planning, we extended DiCoTraM to also track the additional memory requirements of the monitoring devices to fully covering the traffic flows.

Next, we evaluate the effectiveness of DiCoTraM with cSamp, that outperforms most of the other traffic monitoring mechanisms, with regards to supporting effective, early detection of DDoS flooding attacks (i.e., at the intermediate network) by employing two existing detection mechanisms over them. We compare the effectiveness of DiCoTraM with that of cSamp by comparing the detection rates and false positive rates achieved when the selected detection mechanisms are employed over DiCoTraM and cSamp.

5.0 DiCoTraM's Impact on DDoS Flooding Attack Detection Mechanisms

In this section, we compare the effectiveness of DiCoTraM with cSamp with regards to supporting effective, early detection of DDoS flooding attacks. In doing so, we employ two existing DDoS flooding detection mechanisms over DiCoTraM and cSamp. Then, we compare the effectiveness of DiCoTraM with that of cSamp by comparing the detection rates and false positive rates achieved when the selected detection approaches are employed over DiCoTraM and cSamp.

5.1 Case study 1: Distributed Change-point Detection (DCD) architecture

We have adopted the DCD architecture presented in [160] in our first case study. The router-level traffic surge detection algorithm of the DCD architecture requires super-flow level information as its input. Each super-flow comprises of all the flows with the same flow key (e.g., same destination). Since only DiCoTraM and cSamp could provide such information, we only compare these two flow sampling mechanisms as employed monitoring mechanisms in DCD architecture. We made minor modifications to the DCD's local sub-tree construction algorithm as we explain next. We adopted DCD architecture since it was proven to be successful in detecting suspicious DDoS flooding attacks [160, 171] and because the statistics DiCoTraM collects for each flow, could be easily fed to the DCD's traffic surge detection algorithm which works at the router level. We have explained the statistics that routers collect in SRAM for each flow in section 4.2.1. We have also collected the same statistics explained in section 4.2.1 for cSamp.

DCD architecture employs Change Aggregation Trees (CAT) to detect the abrupt traffic changes across multiple ASs. DCD is built over Attack Transit Routers (ATRs) (routers in Figure 4.1 [1]). Each AS has a CAT server (TAS servers in Figure 4.1) to aggregate the flooding alerts reported by the ATRs. CAT servers construct local CAT sub-trees and collaboratively detect DDoS flooding attacks by reporting their sub-trees to their neighbours along the super-flow [2] paths in order to construct the global CAT tree at the victim end. Victim constructs the global CAT tree and makes the final decision based on the threshold (θ) defined by the number of routers that raise alarms on the global CAT tree. CAT servers communicate with each other through a Secure Infrastructure Protocol (SIP) (Secure communication channels in Figure 4.1).

Since DiCoTraM monitors all the flows destined for the same destination (super-flows) at one of the routers on their paths (super-flows paths) and it eliminates redundant traffic monitoring of the flows among the routers, we need to provide the DCD's local CAT sub-tree construction algorithm with all other routers on the super-flows path that would also raise the alarm so that it can generate the local CAT sub-trees. Hence, TASs, upon receiving the alerts from the routers, update the input to CAT sub-tree construction and add router-IDs of the routers on the super-flow path of each of the super-flow alerts.

Next, we briefly describe the DCD's adopted router-level traffic surge detection algorithm [160]. We omit the description of both local and global sub-tree construction algorithms and refer readers to DCD architecture paper [160].

5.1.1 Adopted router-level traffic surge detection algorithm

Here we explain the traffic surge detection algorithm by explaining both the minor modifications we made and details on how we employed the algorithm in our implementation. Hence, first, we define the statistical parameters for the incoming traffic of each aggregated flow at each of the router ports that are employed in the algorithm below.

The historical average of the incoming packets is defined as:

$$\overline{IN}(t_m, i) = (1 - \alpha) \cdot \overline{IN}(t_{m-1}, i) + \alpha \cdot in(t_m, i) \tag{5.1}$$

[1] All the routers are equipped with the traffic surge detection algorithm.
[2] All the packets/flows destined for the same network domain from all possible source IP addresses [160].

Assuming $t_1, t_2, t_3, \cdots, t_m$ be the discrete monitoring time instances, $in(t_m, i)$ is the number of packets which router received on its port i during the monitoring time slot m. The detection mechanism queries SRAM for the updated flow table at each monitoring time instance. For instance, if the monitoring time slot is 200ms, the detection mechanism queries SRAM every 200ms. In our experiments, we change the monitoring time slot from 100ms to 1s to comprehensively study the results, as it was studied in [160], to find the optimum monitoring time slot. Increasing the monitoring time slot from 100ms to 1s only increases the false positive rate and it does not change the number of alerts that are received. Hence, the optimum monitoring time slot based on our experimental results is, as in [160], chosen to be 100ms. α, where $0 < \alpha < 1$, is the sensitivity factor of the long-term average behaviour to the current fluctuation of the traffic (i.e., $in(t_m, i)$).

The deviation of input traffic from the average at monitoring time slot m is defined as follows:

$$Z_{in}(t_m, i) = max\{0, Z_{in}(t_{m-1}, i) + in(t_m, i) - \overline{IN}(t_m, i)\} \tag{5.2}$$

As explained in [160], the abnormal deviation from historical average of the incoming packets, as an indicator of DDoS flooding attacks, is defined as follows:

$$DeviationFromAverage_{in} = \frac{Z_{in}(t_m, i)}{\overline{IN}(t_m, i)} \tag{5.3}$$

β is the router detection threshold which is based on router's past experience. We also use the range of $2 < \beta < 5$ in our experiments as suggested in [160].

All the above mentioned statistics could be similarly defined for the outgoing traffic at each router. Specifically, $out(t_m, i)$, $\overline{OUT}(t_m, i)$, and $Z_{out}(t_m, i)$ are measured similarly.

$DR_{I/O}(i, j)$ is defined to measure the ratio of the incoming packets from port i that are propagated to output port j for the incoming suspicious flows that have already passed the threshold β. In other words, $DR_{I/O}(i, j)$ is the ratio of traffic deviations among various router ports. It has been shown in [160] that if $DeviationFromAverage_{in}$ for specific flow exceeds β and $DR_{I/O} \approx 1$, the traffic flow should be considered as suspicious DDoS flooding attack traffic. We measure $DR_{I/O}$ for those flows by accessing the packet counters (which

resides on DRAM) of the outgoing traffic of those flows in all the router ports. $DR_{I/O}(i,j)$ is defined as follows:

$$DR_{I/O}(i,j) = \frac{Z_{out}(t_m,j)}{Z_{in}(t_m,i)} \tag{5.4}$$

Each router runs the detection algorithm for each incoming aggregated flow (for each entry on the flow table) on the flow's incoming port on the router. Algorithm 4 is the pseudo-code of the detection algorithm on each router.

Algorithm 4 Traffic surge detection at the routers [160]

Input: $in(t_m,i)$, $out(t_m,i)$, $\overline{IN}(t_{m-1},i)$, $\overline{OUT}(t_{m-1},i)$, *and* β

Output: Sending proper alerts to TASs

1: Calculate and update $\overline{IN}(t_m,i)$ and $Z_{in}(t_m,i)$
2: Calculate $DeviationFromAverage_{in}$
3: **if** $DeviationFromAverage_{in} \geq \beta$ **then**
4: Calculate $DR_{I/O}$
5: **if** $DR_{I/O} \approx 1$ **then**
6: Send suspicious flow alert to TAS

DCD's Limitation: One of the main limitations of the DCD architecture is its incapability in discriminating between the DDoS flooding attacks and the flash crowds (fluctuations of the legitimate traffic) [160]; since the CAT server creates the same tree when there is a flash crowd and could raise a false alarm (false positive). Hence, the false positive rate would increase significantly if we would add some sources of flash crowds in our experiments in this section. *Chen et al.* proposed to check the newly appeared source IP addresses and their distribution to find a new metric in order to differentiate between the flash crowds and the DDoS flooding attacks [160]. Next, we present a recently proposed DDoS flooding detection mechanism that incorporated *Chen et al.*'s suggestion and could outperform DCD architecture in terms of both detection rate and false positive, specially when there is flash crowds.

5.2 Case study 2: Distributed DDoS Flooding Detection based on Total Variation Distance (TVD)

In our second case study, as we mentioned before, we have adopted one of the recently proposed DDoS flooding detection mechanisms proposed in [176]. *Rahmani et al.* could show some promising results in terms of detection performance compared to other detection mechanisms. Moreover, *Rahmani et al.*'s mechanism, we call TVD from now on, is a distributed DDoS flooding detection mechanism that flow monitoring mechanisms such as DiCoTraM and cSamp could provide with more covered flows as opposed to packet sampling mechanisms. TVD is a two-stage mechanism that detects DDoS flooding attacks and differentiates non-legitimate flows from legitimate flows (i.e., flash crowds) based on the detection of breaks in the distribution of connection sizes. Connection is defined in [176] as the aggregate traffic between two IP addresses (i.e., a source IP and a destination IP address) and the connection size is measured by the number of packets traversing the connection. The concept of connection in TVD mechanism is the same as super-flow concept defined by *Chen et al.* in [160].

We have implemented a prototype of the TVD mechanism and conducted a series of experiments by employing DiCoTraM and cSamp as two flow monitoring mechanisms to compare their impact in terms of TVD's DDoS flooding attack detection. Similar to our previous case study, DiCoTraM and cSamp were the best options as TVD's traffic monitoring mechanism since TVD mechanism requires to analyze the super-flow level statistics (connection-level) as its input and both DiCoTraM and cSamp could provide such statistics without further processing of the monitored data to extract the super-flows or connections.

For each connection (supper-flow), TVD not only requires to monitor and count the number of packets heading to the destination IP address of the connection but also requires to monitor and count the number of packets coming from the source IP address of the connection. Hence, both DiCoTraM and cSamp prototype implementations are modified to collect the number of packets coming from the source IP addresses of each of the connections in addition to the number of packets heading to the destination IP addresses of each of the connections.

5.2.1 TVD's detection algorithm

As we mentioned earlier, TVD has two stages. In the first stage, horizontal TVD, TVD detects the abnormal disruption in the inflow (i.e., same source IP address) connections size distribution by observing the connection size distribution of the same connections in two consecutive monitoring time windows. The outcome of the first stage determines the list of suspicious attack flows (i.e. surge flows) and *Rahmani et al.* also shown that these results are better than entropy-based detection mechanisms. The metric that TVD uses to detect abnormal disruption in the inflow connections size distribution (supper-flow size) is the horizontal TVD between two consecutive monitoring time windows. Assuming $t_1, t_2, t_3, \cdots, t_m$ be the discrete monitoring time instances, $Source(t_m, i)$ is an array of normalized frequencies of source address bin size distribution for the ith connection during the monitoring time slot m. Considering only K common connections (repetitive connections) in two consecutive time windows[3], the horizontal TVD between two consecutive monitoring time windows is defined as follow:

$$Horizontal\ _{TVD} = \frac{1}{2}\sum_{i=1}^{K} \mid Source(t_m, i) - Source(t_{m-1}, i) \mid \tag{5.5}$$

The detection mechanism queries SRAM for the updated flow table at each monitoring time instance to extract the required statistics for inflow traffic. In our experiments, we chose the optimum monitoring time window, based on the results of our experiments, to be equal to 100ms.

The second stage of TVD detection algorithm differentiates the DDoS flooding attack flows from the legitimate traffic changes that causes a surge (a.k.a. flash crowds). DDoS flooding attack flows lead to divergence between the number of packets sent to and received from a specific destination. In order to measure this divergence, *Rahmani et al.* proposed total variation distance between inflow and outflow connection size distributions for two consecutive monitoring time windows. Usually, false positives occur when the size of one or several legitimate connections are larger than the average size of all active connections. Consequently, distinguishing between the flash crowds and DDoS flooding attacks when the

[3]Super-flows that last for two consecutive monitoring time windows

variation in the size of the aggregate incoming flows are not accompanied by a proportional change in the number of active connections is very difficult. When there exist a legitimate aggregate traffic flows, the variation of the aggregate traffic size from source to destination is accompanied by a relative change in the opposite direction. However, when there is an aggregate DDoS flooding attack flows, the aforementioned dependency vanishes and there will be a disparity in the size of flows that are exchanged between the attacker and the victim [51–53, 176] and this is a sign of the beginning of the congestion caused by DDoS flooding attacks. *Rahmani et al.* proposed a differentiation distance metric to detect the divergence between the number of packets sent to and received from a specific destination as vertical TVD during the current monitoring time window. Assuming $t_1, t_2, t_3, \cdots, t_m$ be the discrete monitoring time instances, $Source(t_m, i)$ is an array of normalized frequencies of source address bin size distribution for the ith connection during the monitoring time slot m for all the N connections during the current monitoring time window. $Destination(t_m, i)$ is an array of normalized frequencies of destination address bin size distribution for the ith connection during the monitoring time slot m for all the N connections during the current monitoring time window. We assume that the ith connection corresponds to the same pair of IP addresses (source and destination IP addresses) and that both $Source(t_m, i)$ and $Destination(t_m, i)$ may be equal to zero. The vertical TVD during the current monitoring time window is defined as follow:

$$Vertical\ _{TVD} = \frac{1}{2} \sum_{i=1}^{N} \mid Source(t_m, i) - Destination(t_m, i) \mid \tag{5.6}$$

Similar to horizontal TVD measurement, vertical TVD measurement occurs by querying SRAM for the updated flow table at each monitoring time window (i.e., 100ms) to extract the required statistics. The thresholds for vertical and horizontal distances as suggested by *Rahmani et al.* should be determined based on the normal traffic.

TVD, like any other detection scheme, could not completely detect all the attacks but it could significantly reduce the false positive rate and increase the detection rate.

5.3 Evaluations & experiments

The main objective of any traffic monitoring mechanism must be to cover as many attack flows as possible; hence, we believe it is fair to employ both cSamp and DiCoTraM as traffic monitoring mechanisms that feed the monitored flow information to the DCD architecture and TVD detection mechanism and compare these two traffic monitoring mechanisms in terms of how successful they are in covering as many DDoS flooding attack flows as possible, which leads to higher detection rate [4] and lower false positive rate. Hence, we perform series of experiments that we explain next.

5.3.1 Experimental set-up

We used DETER testbed [172] and DETER's SEER tool [173] for our experiments. We have created the topology presented in Figure 5.1 on DETER. We used Harpoon [174] traffic generator tool and Packet flooder attack tool which are part of SEER on DETER testbed to generate legitimate and malicious traffic from various legitimate and malicious sources in our experiments.

Harpoon is designed to create background network traffic. Harpoon can characterize a traffic trace and generate a sequence of packets with the same timing characteristics but with random data as packets' payload. Harpoon also supports predefined statistical distributions such as: minmax, gamma, pareto, exponential for generating traffic traces. In our experiments, we employed exponential and pareto distributions randomly to create traffic from various sources to target destination in order to create legitimate traffic load heading to the target destination. Moreover, we used Harpoon to generate malicious traffic loads in our experiments.

[4] Detection rate: Ratio of the number of malicious packets/flows that are detected (TP) to the total number of malicious packets/flows (M) $\longrightarrow \frac{TP}{M}$

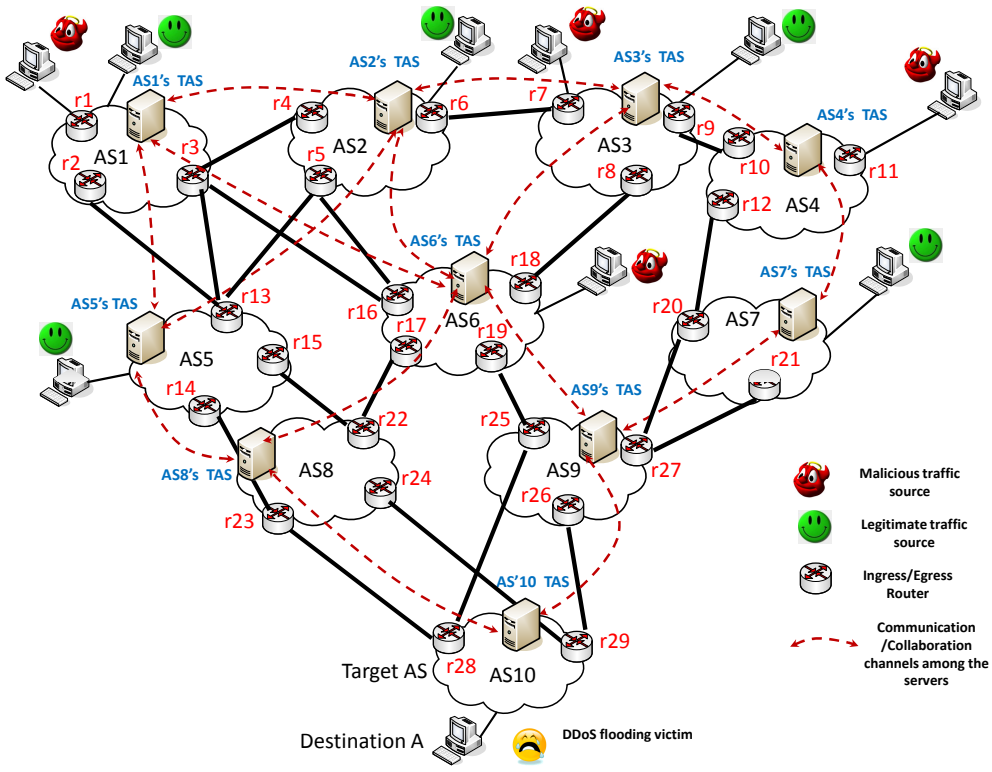


Figure 5.1: DETER simulation set-up

Packet flooder attack tool of SEER is designed to generate various rates of TCP/UDP (e.g., SYN, FIN, RST, ACK, PSH, URG) or ICMP packet floods from group of sources to a DDoS flooding victim. We have generated three types of DDoS flooding attacks using packet flooder, as it was suggested in [160], from the designated attack sources in our experiments to flood our target destination that are explained as follow:

1. **TCP SYN**: TCP SYN flooding attacks are generated using fixed packet size of 64 bytes with a fixed packet rate (packet rate is adjusted for each topology).
2. **UDP/ICMP floods with 512B and 1024B packets sizes**: Two types of UDP/ICMP flooding attacks are generated using fixed packet sizes of 512 and 1024 bytes with adjusted packet rates for each packet size setting.

We have implemented the simulation set-up shown in Figure 5.1 on DETER in which, destination (A) in AS10 is under DDoS flooding attack. Since most of the packets in the Internet can reach their destinations by traversing through 3 AS-hops (38.55%) or 4 AS-hops

(38.12%) [175], we chose the topology in Figure 5.1 in which destination (A) is 3 AS-hops away from all the sources. In all of our experiments, we assume that there is a legitimate traffic load to destination (A) that consumes ≈ 50% of the destination's bandwidth. Figure 5.1 simply depicts our DETER simulation set-up to launch DDoS flooding attacks against destination (A) through 10 different ASs. The legitimate load originates from AS1, AS2, AS3, AS5, and AS7 (12-13% of the total traffic each). We added malicious traffic loads originating from AS1, AS3, AS4, and AS6 in our experiments.

Each node (i.e., router in our design) uses a classifier to sample the packets in NS-2. We have modified the classifier to sample the packets at our defined sampling rates and to save the fields and statistics, which we explained in Subsection 4.2.1, of the monitored packets on SRAM and DRAM, respectively.

5.3.2 Performance evaluations

We run the simulation set-up in Figure 5.1 for one hour and separately employing DCD architecture and TVD detection mechanism [5]. For each of the detection mechanisms we employed cSamp and DiCoTraM separately. We run the simulation set-up for each detection mechanism and for each of the traffic monitoring mechanisms for 10 times. There is a background traffic load to destination (A) that consumes ≈ 50% of the A's bandwidth. We have generated five different mix of malicious/legitimate traffic that consume 60%, 70%, 80%, 90%, and 100% of the destination A's bandwidth. In other words, we are overloading (flooding) destination (A) by consuming 110%, 120%, 130%, 140%, and 150% of its bandwidth. The mix of malicious/legitimate traffic are generated in such a way that half of the traffic were malicious and the other half were legitimate. These scenario generate the case of flash crowds to put TVD in test since it is capable of differentiating DDoS flooding attacks from flash crowds. As we mentioned earlier, we run our simulation 10 times for each the traffic mix scenarios (i.e., 60% , 70% , etc.). The bandwidth of the links for the destination A were set at 100 MB/s.

[5]The correctness of our implementations of the DCD architecture and the TVD detection mechanism are separately validated by replicating the results of some of the experiments that have been performed in [160] and [176], respectively.

The malicious traffic and the legitimate traffic are known by their traffic sources so that we can measure the detections rates and false positive rates of each set-up later (i.e., ground truth). We added malicious loads (i.e., UDP/ICMP floods with 512b and 1024b packet sizes, and TCP SYN flood) from AS1, AS3, AS4, and AS6, using Packet flooder attack tool of SEER. We have also added legitimate traffic loads from AS1, AS2, AS3, AS5, and AS7, using Harpoon traffic generator tool of SEER. The size of the buffer for all of the routers during the simulation was kept constant and equal to 8MB as in our previous experiments.

The packet rates (number of packets/second) for the packet flooder attack tool were adjustable based on the size of the packets. In our experiments, based on the size of the packets, packet rates of different flooding types were chosen as it is shown in Table 5.1. Figure 5.2 shows the trade-off between the detection rate and false-positive rate for DCD architecture and for three different types of DDoS flooding attacks employing DiCoTraM and cSamp.

Table 5.1: Packet Flooder (Packet Size vs. Packet Rate)

Flood Type	Packet Size	Packet Rate
UDP	512 bytes	20-22 KPkt/s
	1024 bytes	11-12 KPkt/s
ICMP	512 bytes	21-22 KPkt/s
	1024 bytes	12-13 KPkt/s
TCP	64 bytes (fixed)	60-62 KPkt/s

Figure 5.2a, Figure 5.2b, and Figure 5.2c show the ROC curves including 95% confidence interval for both detection rate (TPr) and false positive rate (FPr) [6] [7] of the DCD architecture employing DiCoTraM and cSamp under TCP SYN with 64-bytes, and UDP/ICMP attacks with 512-bytes and 1024-bytes. The reported results are based on $\alpha = 0.1$, $\beta = 3.2$,

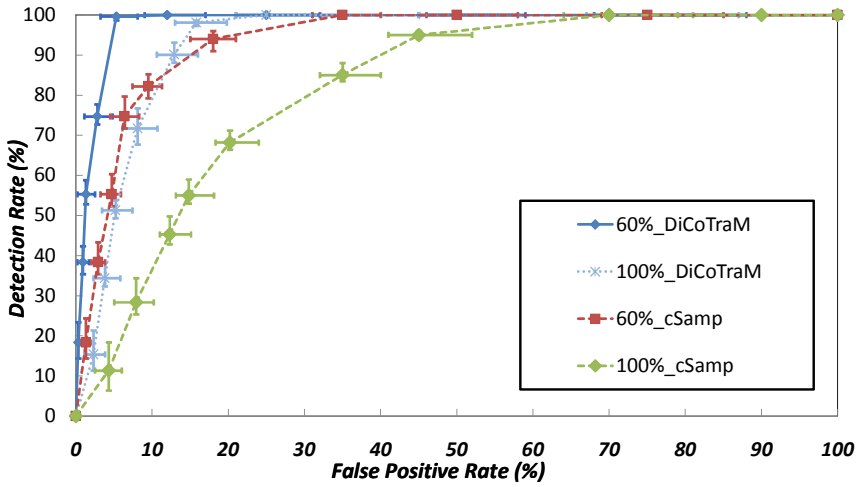
[6]The 95% confidence interval is the interval in which the true area under the ROC curve lies with 95% confidence

[7]95% confidence interval (TPr/FPr) = average (TPr/FPr) of runs $\pm$ 1.96 $\times$ Standard deviation (TPr/FPr) of runs

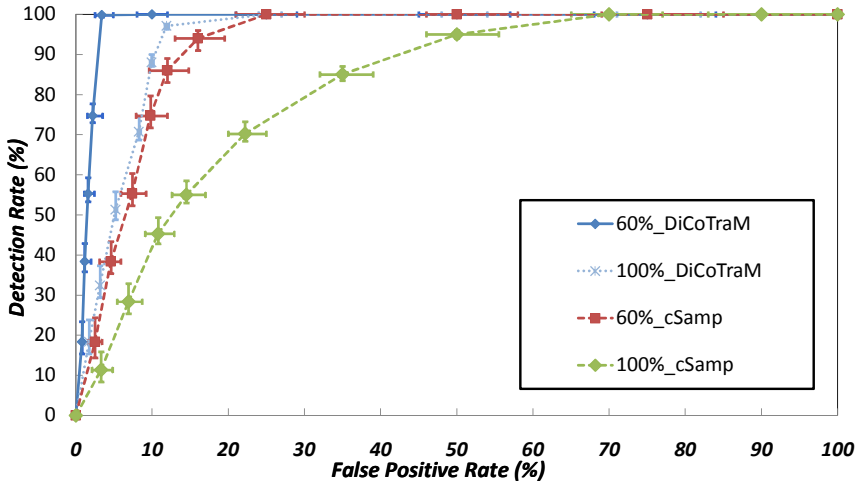
and monitoring time slot = 100 ms in DCD architecture [160] by varying θ from 0 to 9. The aforementioned parameters are chosen because of their best results in our various experiments. As seen in Figure 5.2, DiCoTraM performs significantly better than cSamp, specially in terms of detection rate, as the DCD architecture's θ is changing to its optimum value $\theta = 5$ where the detection rate is reaching its maximum value. As we mentioned earlier, DiCoTraM outperforms cSamp because of its higher DDoS flooding attack flow coverage.

In Figure 5.2a, the DCD architecture employing DiCoTraM, achieves a detection rate as high as 99% and as low as 98% for high-rate (TCP SYN 64-bytes) DDoS flooding attack. The amount of false positive rate increases from less than 5% to less than 15% as the amount of traffic load to destination A increases to consumes 150% of the destination A's bandwidth. In Figure 5.2b, the DCD architecture employing DiCoTraM, achieves a detection rate as high as 99% and as low as 97% for high-rate (UDP 512-bytes) DDoS flooding attack. Moreover, the amount of false positive rate increases from less than 3% to less than 12% as the amount of traffic load to destination A increases to consumes 150% of the destination A's bandwidth.

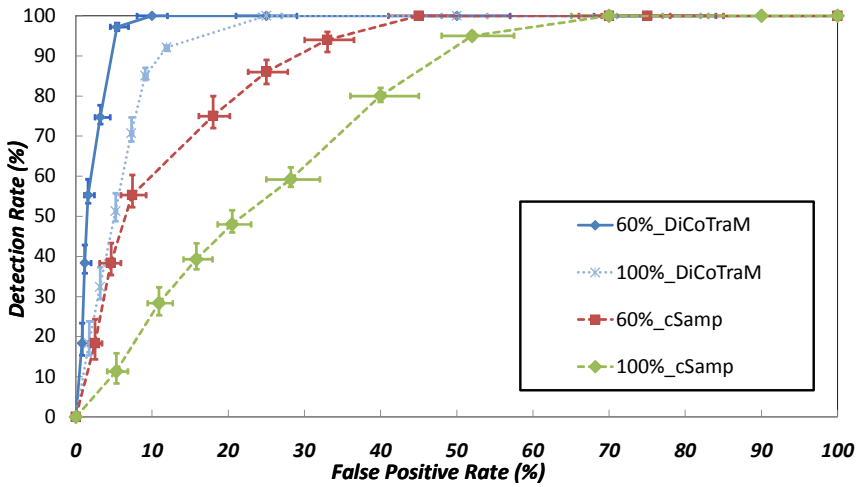
For the low-rate UDP attacks (UDP 1024-bytes) for the DCD architecture that employs DiCoTraM, as it is shown in Figure 5.2c, the choice of a low CAT threshold ($\theta = 5$) led to the detection rate as high as 97% (when there is less malicious load) and as low as 92% (with the maximum malicious load equal to 100%). The amount of false-positive rate also increases from less than 5% to less than 13%. As we mentioned earlier, in the case of DCD architecture that employs cSamp, the detection rates are decreased and false positive rates are increased as opposed to the DCD architecture that employs DiCoTraM. The main reason behind this is that cSamp covers the flows intended for the same destination fractionally (not necessarily fully) over the routers on their paths. Hence, the traffic surge detection algorithm of those routers may not detect a surge because specific fraction of flows may not pass the router's alarm threshold. Consequently, in all the figures, cSamp's low flow coverage significantly affects both the detection rate and the false positive rate of the DCD architecture and DiCoTraM outperforms cSamp in terms of its resulting detection and false positive rates on DCD architecture.



(a) TCP SYN 64-bytes



(b) UDP 512-bytes



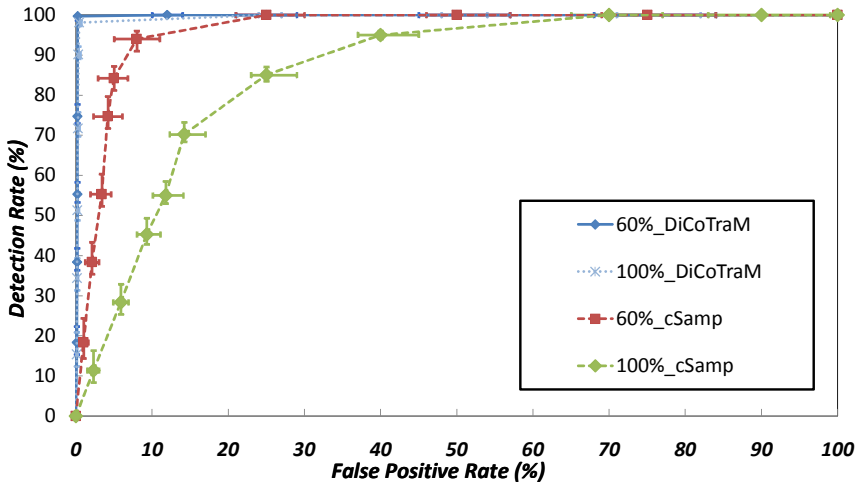
(c) UDP 1024-bytes

Figure 5.2: ROC curves including 95% confidence interval employing DCD architecture

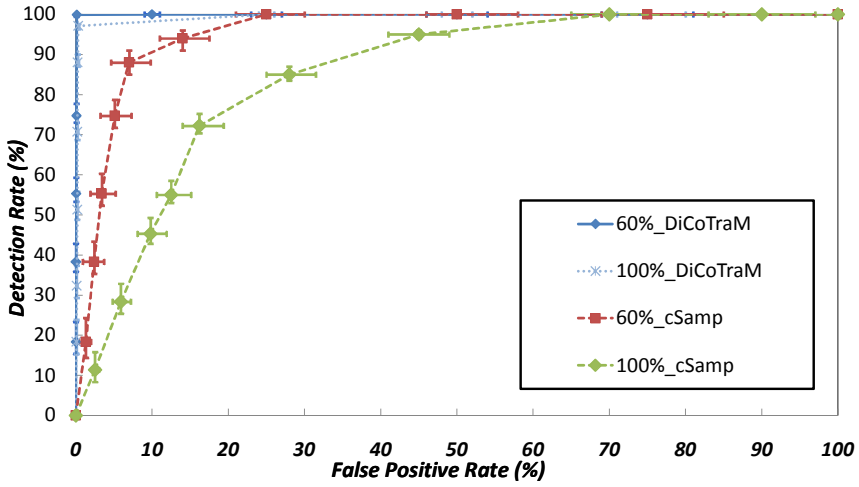
Figure 5.3a, Figure 5.3b, and Figure 5.3c show the ROC curves including 95% confidence interval for both detection rate (TPr) and false positive rate (FPr) of the TVD detection mechanism employing DiCoTraM and cSamp under TCP SYN with 64-bytes, and UDP/ICMP attacks with 512-bytes and 1024-bytes. The reported results are based on the monitoring time slot = 100 ms and by varying both $Horizontal_{TVD}$ and $Vertical_{TVD}$ from 0 to 1 (in increments of 0.1). As it is shown in Figure 5.3, DiCoTraM performs significantly better than cSamp in terms of both detection rate and false positive rate, in all the figures, when the amount of the thresholds are reaching their optimum values. The $Horizontal_{TVD} = 0.5$ and $Vertical_{TVD} = 0.65$ are the optimum values to reach the best performance results in our experiments.

Again, as is shown in Figure 5.3, we can see that DiCoTraM has much better performance than cSamp because of its higher DDoS flooding attack flow coverage. In Figure 5.3a and Figure 5.3b, the TVD detection mechanism that employs DiCoTraM, achieves a detection rate as high as 99.8% with less than 1% false positive rate for high-rate DDoS flooding attacks (TCP SYN 64-bytes and UDP 512-bytes). Also, for the low-rate UDP flooding attacks (UDP 1024-bytes), employing TVD significantly improved both detection rate and false positive rate. Employing the $Horizontal_{TVD}$ led to the detection rate as high as 98% and employing $Vertical_{TVD}$ led to the false-positive rate of less than 2%. An increased malicious load from 60% to 100% did not affect the detection rates significantly in all the figures for the case of DiCoTraM as it aggregates all the flows heading to the same destination. As it is shown in all of these figures, employing TVD significantly decreased the false positive rate since TVD is capable of differentiating between DDoS flooding attacks and the flash crowds.

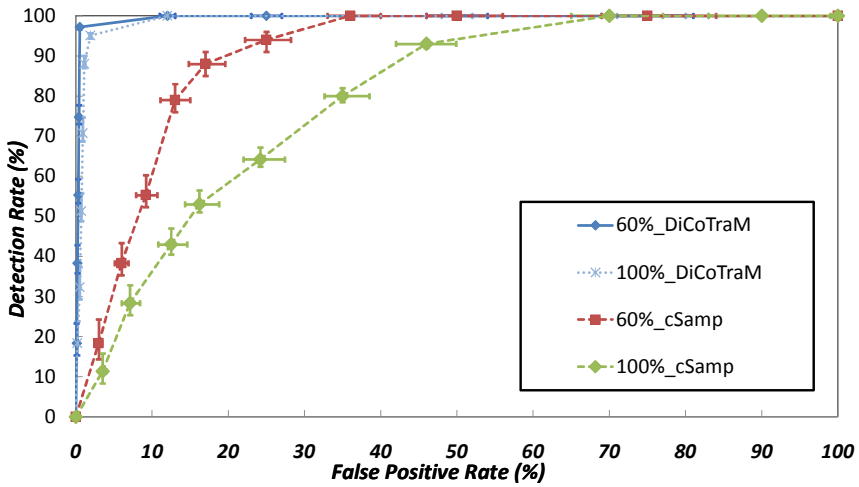
Employing TVD even decreased the false positive rates in all of the experiments (figures) for cSamp as well. However, DiCoTraM still outperforms cSamp in terms of its resulted detection and false positive rates using TVD detection mechanism since cSamp covers less DDoS flooding attack flows than DiCoTraM because the flows intended for the same destination are covered fractionally (not necessarily fully) over the routers on their paths.



(a) TCP SYN 64-bytes



(b) UDP 512-bytes



(c) UDP 1024-bytes

Figure 5.3: ROC curves including 95% confidence interval employing TVD

5.4 Support for earlier detection of DDoS flooding attacks

One of our key motivations for proposing DiCoTraM is to increase the DDoS flooding attack flow coverage at the intermediate networks and closer to the sources of the attack; so that the DDoS flooding defense mechanisms in place can analyze these covered attack flows, detect the attacks, and eventually respond to these attacks earlier at the upstream ASs closer to the sources of these attacks. In this section, we perform series of experiments in order to evaluate DiCoTraM's increased DDoS flooding attack flow coverage capability at their earlier stages and closer to their sources.

For this purpose, we redo the same experimental set-up that we explained in section 5.3.1 and the same experiment we explained and performed in section 5.3.2 for the DCD architecture. In other words, for different attack traffic types (i.e., TCP SYN 64B, UDP flooding 512B, and UDP flooding 1024B) and for each of the two traffic monitoring mechanisms (DiCoTraM and cSamp), we run the simulation set-up 10 times. In each run, there are number of routers that raised the alarm after running their local surge detection algorithm. The raised alarms were either false positives or true positives. During each run of the simulation set-up and for each router in our topology, we counted the number of times it raised an alarm and the set of flows it raised an alarm for. At the end of each run of the simulation set-up and by considering the results of the majority voting algorithm among the collaborating ASs, we identified the routers that their raised alarms (for the suspicious flows) could successfully lead to a collaborative detection.

As we explained earlier, DCD architecture collaboratively detects the DDoS flooding attack flows by performing a majority voting like algorithm among the collaborative ASs that reported those flows as suspicious flows. The majority voting algorithm decides on whether the suspicious reported flows are the attack flows or not based on a predefined threshold (θ). This threshold is ($\theta = 5$) in our experiments.

Figure 5.4, on the DETER simulation-setup (Figure 5.1), shows the routers that were effectively involved in detecting the DDoS flooding attack flows for TCP SYN 64B DDoS flooding attack type employing the cSamp monitoring mechanism after all runs of the simulation-setup. As shown in Figure 5.4, the majority of the effective routers in detecting TCP SYN 64B attack are located on AS8, AS9, or AS10 and at most one AS-hops away from the victim's AS.

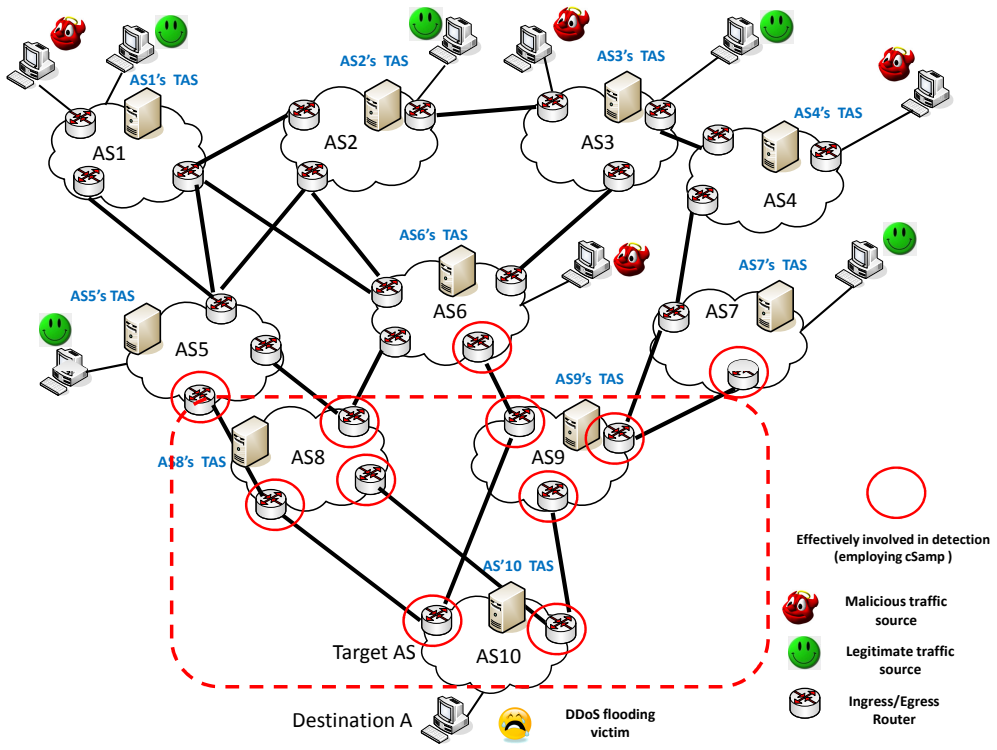


Figure 5.4: Routers effectively involved in detecting TCP SYN 64B flooding attack (employing cSamp)

Figure 5.5 shows the routers that were effectively involved in detecting the DDoS flooding attack flows for TCP SYN 64B DDoS flooding attack type employing the DiCoTraM monitoring mechanism after all runs of the simulation-setup. As shown in Figure 5.5, the majority of the effective routers in detecting TCP SYN 64B attack are located on either AS5, AS6, AS7, AS8, or AS9 and at most two AS-hops away from the victim's AS which is much closer to sources of the attacks than the results in Figure 5.4.

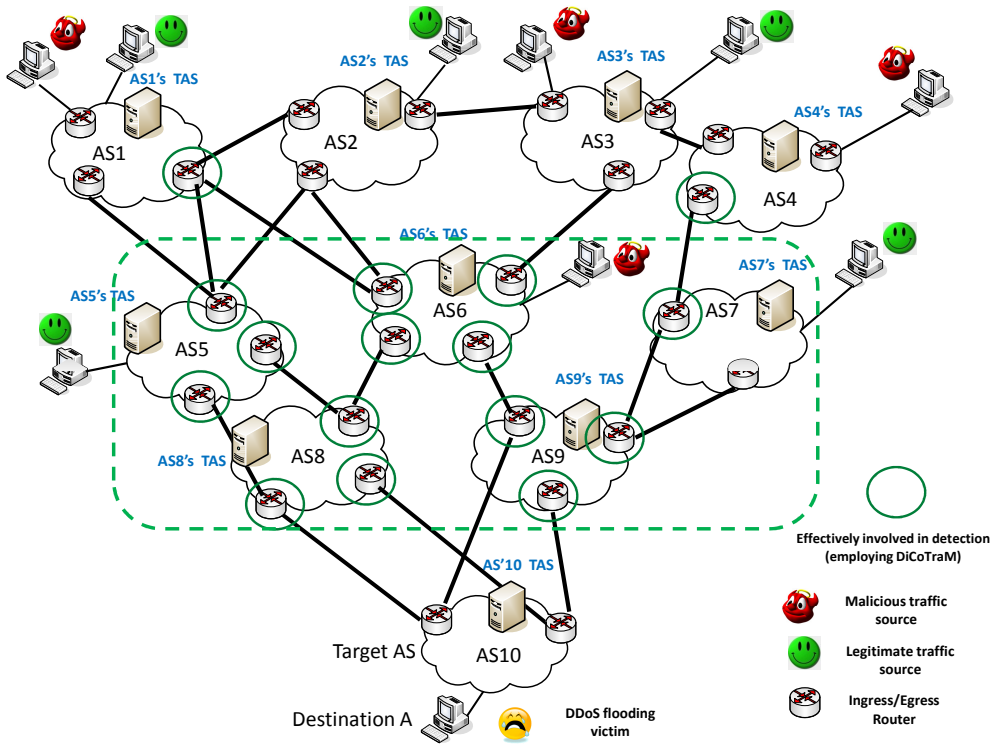


Figure 5.5: Routers effectively involved in detecting TCP SYN 64B flooding attack (employing DiCoTraM)

Note that most of the well-known ASs (e.g., AT&T, Sprintlink, etc.) are comprised of large number of routers or monitoring devices and detecting DDoS flooding attacks even one AS-hop (AS-hop level) earlier is a huge success; hence, these results are promising positive results. DiCoTraM's large DDoS flooding flow coverage at the intermediate network and as close as possible to the sources of the attacks, in this case, leads to the DDoS flooding

attack detection at least one AS-hop earlier than the case of cSamp by employing the DCD architecture.

We believe more work needs to be done in order to be able to generalize these results. Various detection mechanisms need to be employed in different scenarios to ensure and validate the results. In future, we are planning to evaluate DiCoTraM's capability in supporting early detection by employing more detection mechanisms and various scenarios.

5.5 Summary

In this chapter, we evaluated the effectiveness of DiCoTraM with cSamp (since cSamp's performance outperforms other traffic monitoring mechanisms) with regards to supporting effective, early detection of DDoS flooding attacks (i.e., at the intermediate network). In doing so, we implemented two of the existing DDoS flooding detection mechanisms, namely, DCD architecture and TVD detection mechanism, where DiCoTraM and cSamp served as the traffic monitoring mechanisms of both of these DDoS flooding detection mechnaisms. Then, we compared both the detection rates and false positive rates achieved when DCD architecture and TVD mechanism are employed over DiCoTraM and cSamp. As we mentioned earlier, these two DDoS flooding detection mechanisms were selected since they have been shown to have good performance in detecting DDoS flooding attacks in the litrature. Our experimental results show that DiCoTraM outperforms cSamp in all of our experiments when they were employed as traffic monitoring mechanisms for both DCD architecture and TVD detection mechanism. Moreover, in the same set-up and by employing DCD architecture, we evaluated and validated the DiCoTraM's capability in early detection of DDoS flooding attacks closer to their sources of the attacks (i.e., at upstream AS-hop levels closer to the source ASs).

6.0 Conclusions and Future work

The work in this dissertation was motivated by two key observations we made in our literature review: (i) the incapability of the centralized DDoS flooding defense mechanisms in detecting and stopping such attacks effectively and the expectation in seeing more distributed DDoS flooding defense mechanisms in short term, and (ii) the intrinsic challenges of existing traffic monitoring mechanisms employed by the next generation of DDoS flooding defense mechanisms, in enabling successful, early detection of DDoS flooding attacks at the intermediate network:

- Coordination challenges of the device-centric approaches
- Low and costly flow coverage of the device-centric approaches
- Random flow coverage as opposed to prioritized flow coverage

We proposed a novel coordinated, network-wide, and DDoS flooding attack-tailored flow monitoring mechanism (DiCoTraM) that could address the above key challenges of current device-centric traffic monitoring mechanisms and helps effective, early detection of possible DDoS flooding attack flows.

Next, we summarize our contributions in this dissertation and enumerate some potential directions for future work.

6.1 Contributions

First, we studied the origin of DDoS flooding attacks, their variations, and present a survey of existing solutions to the problem of DDoS flooding attacks. We summarized the list of required features for the next generation of DDoS flooding defense mechanisms by categorizing them into: short-to-medium term and long term. The most important requirement on this list is to see more distributed DDoS flooding defense mechanisms in near future which motivated us to conduct the research presented in this thesis. As our second step, we study the existing traffic monitoring mechanisms and identify their various challenges within the concept of DDoS flooding attack. The success in detecting DDoS flooding attacks in a distributed fashion is highly dependent on the quantity and quality of the distributed traffic monitoring mechanisms that are being employed. Our review of the literature revealed various deficiencies in currently employed traffic monitoring mechanisms (e.g. uncoordinated monitoring in device-centric approaches). Our key take out from this study was the need for the next generation of DDoS defense mechanisms that ensure that the majority of the attack flows are monitored/covered in each and every AS while satisfying the resource constraints of the monitoring devices. Moreover, we highlighted the benefits of network-wide traffic monitoring policy enforcement approaches over device-centric approaches.

We proposed DiCoTraM, a novel, centrally managed, network-wide traffic monitoring mechanism to address the limitations of current traffic monitoring mechanisms with regards to the problem of DDoS flooding attack. DiCoTraM coordinates monitoring responsibilities among the monitoring devices while satisfying the resource constraints of the monitoring devices. Moreover, DiCoTraM prioritizes the coverage of the DDoS flooding attack flows. DiCoTraM's coordination structure eliminates the redundant flow monitoring overheads due to lack of coordination. Furthermore, DiCoTraM covers more DDoS flooding attack flows which enables distributed detection of DDoS flooding attacks at the router-level and leads to reduced communication overhead compared to centralized detection mechanisms. Centralized detection mechanisms centrally collect the monitored flows/packets for further analysis at the end of each monitoring window which leads to huge communication overhead.

We presented a proof of concept implementation of DiCoTraM to show its applicability. We reported the results of the experiments we performed to evaluate the performance of DiCoTraM. We analyzed the scalability of DiCoTraM and improved it by proposing a heuristic. Moreover, in order to reduce the computation time of the monitoring assignment process in our proposed heuristic, we modified the heuristic by pre-processing the input to the monitoring assignment process; consequently, the modified heuristic only run the monitoring assignment process for the set of new or changed flows compared to the previous assignment. In doing so, the modified heuristic kept the same assignment for the long-term flows as long as the total DDoS flooding flow coverage was maintained or was not significantly fallen below some predefined threshold.

We extended DiCoTraM to help the network operators in determining a list of candidate monitoring devices to upgrade (i.e., memory upgrade) for achieving better flow coverage. Extended version of DiCoTraM computed the amount of additional memory needed to fully cover the traffic flows. We evaluated the memory consumption of the routers within a well-known AS topology for both with and without DDoS flooding attacks by means of simulations. Moreover, we performed a set of experiments in the same AS topology to evaluate the effectiveness of the network planning capability that our extension provided in terms of flow coverage in both with and without DDoS flooding attacks.

Finally, we evaluated the effectiveness of DiCoTraM with cSamp, that outperforms most of other traffic monitoring mechanisms, with regards to supporting effective, early detection of DDoS flooding attacks by employing two existing DDoS flooding detection mechanisms over them. These two DDoS flooding detection mechanisms were selected since they had been shown to perform well in detecting DDoS flooding attacks in the literature. We then compared the effectiveness of DiCoTraM with that of cSamp by comparing the detection rates and false positive rates achieved when the selected detection mechanisms (DCD architecture and TVD detection mechanism) were employed over DiCoTraM and cSamp. Our results showed that DiCoTraM outperformed cSamp in all of our experiments when they were employed as traffic monitoring mechanisms of both DCD architecture and TVD detection mechanism.

6.2 Limitations of Proposed Work

There are potential limitations of DiCoTraM; we overview them in this section.

Input dependency: DiCoTraM's optimization formulation, like any other centralized network management application that employs an optimization formulation to achieve its network-wide objective (e.g., cSamp), is highly dependent on the accessibility and the accuracy of its input parameters, namely: traffic matrices, routing information, and the technological capabilities of the monitoring devices (e.g., CPU, memory, etc.). Fortunately, various traffic matrix estimation techniques [139,177,178] and routing tracking management tools [179] have been proposed in the literature that are currently deployed in many networks for these purposes. Moreover, the capabilities of the monitoring devices are available through their vendors or through straight-forward measurement techniques [180] that could be performed on these devices to obtain these required capabilities. However, the accuracy of the available inputs and the effects of inaccurate input parameters on the optimization engines of any network management application (e.g., DiCoTraM) is still in question. For instance, errors in estimated traffic matrices or inaccurate routing information due to recent link failures/new nodes could lead to non-optimized solutions for the network management applications that include those optimization models; hence, their high-level objectives will not be achieved.

However, the effects of inaccurate inputs or approximate measurement may still lead to reasonable results for various network management applications. For instance, as we have shown in our scalability-improved proposed heuristic, we could still achieve reasonable results by tolerating a certain performance degradation (i.e., predefined DDoS flooding tolerance threshold) when we decided to pre-process the input, which would lead to a case of inaccurate inputs, to reduce the computation time of the optimization formulation. The reported results were not far off from the optimized results. It is possible to propose some specially tailored heuristics for various network management applications that could workaround errors in input estimates [136]. Moreover, an alternative approach is for the network administrators to run the optimized solution to obtain the most accurate results when there is a change in the network status or even periodically.

Scalability: One of the key challenges that we encountered in the DiCoTraM's centralized optimization engine design which is also common in most of the centralized optimization techniques (e.g., cSamp) is the scalability of the optimization engine when the problem sizes are significantly large (e.g., large network topologies). For instance, as we showed in chapter 4, sometimes the problem size is too large that the optimization engine can not solve the problem in a desirable time limit. However, most of the times, such scalability issues can be handled through existing algorithmic techniques (heuristics) such as: Max-Flow or binary search. For instance, for DiCoTraM, we could address its scalability challenge through our proposed heuristic algorithm. Moreover, it is possible to reduce the input size by preprocessing the input parameters, like what we proposed in our modified heuristic for DiCoTraM. Furthermore, *Sekar et. al* proposed alternative approaches in which either the solutions for expected configurations could be pre-computed (e.g., to adapt to predictable traffic dynamics) or distributed agents could run local algorithms with smaller problem sizes (smaller input sizes) to achieve a global optimum objective [136].

Vulnerability: In random sampling traffic monitoring mechanisms, it is not easy for the adversaries to determine which packets or flows are going to be monitored. However, this is not the case for the centrally coordinated traffic monitoring mechanisms (e.g., cSamp, DiCoTraM) and adversaries can evade the detection mechanisms easier [136]. In other words, knowing the traffic monitoring configurations, especially with traffic monitoring mechanisms like cSamp in which some of the hash ranges are fixed for large number of flows, attackers can generate or redirect their traffic in such away that they can evade monitoring mechanisms; consequently, they can evade detection mechanisms. However, the probability of a flow that will not be covered, employing the coordinated sampling mechanisms, is really low as opposed to random sampling mechanisms because of the significantly large flow coverage that coordinated sampling mechanisms can provide.

Moreover, in case of DiCoTraM, assignments are mostly changing and they are randomized in nature (less deterministic) which would require significant efforts from the adversaries in order to evade detection mechanisms. Moreover, the intrinsic funnel-like feature of the DDoS flooding flows requires them to get aggregated at some points on their paths from the attack sources to the victim as we mentioned in chapter 2; hence, the malicious traf-

fic will be eventually detected in the AS further down on the path from attack sources to the victim since traffic monitoring mechanisms of downstream AS will be able to monitor them. Furthermore, DiCoTraM's monitoring configuration is dynamic and random since it performs dynamic readjustments [1] which makes it even harder for adversaries to determine which packets or flows are going to be monitored.

Adoptability: Our proposed traffic monitoring mechanism can be employed by network operators with the current device-centric set-ups only as a third-party middle-box [2]. As we highlighted in Chapter 3, the device-centric approaches are not capable of addressing the future requirements of the network management applications in achieving various high-level network management objectives and in enforcing these high-level management policies.

Nowadays, with the capabilities that recent trends in network management like SDN provides (separation of control plane from data plane led to various customizable control plane applications) [155,157], network operators in network enterprises and ISPs are expected to be capable of centrally specifying and achieving their network management objectives/policies through transparent network management tools [158,159]. The development of various unified control interfaces that provide the network operators their required network-wide visibility to control and configure their networks centrally with various network management tools/applications to achieve various high-level network management objectives (e.g., DiCoTraM), will increase and ease the adoption of the network management applications such as DiCoTraM in near future [158,159].

[1] Monitoring responsibility reassignments between the monitoring time windows by receiving feedback from the system and when some predefined monitoring thresholds (i.e., reduced DDoS flooding flow coverage) are passed.

[2] One of the main limitations of middle-boxes is that their functionality is not visible and transparent to network operators.

6.3 Future Work

There are several potential future directions related to the research presented in this dissertation that we briefly present them in this section.

Extending DiCoTraM to monitor statistics other than flow statistic: Some of the Quality of Service (QoS) metrics (e.g., throughput, delay) cannot be monitored through flow-level statistics that DiCoTraM is currently monitoring. Moreover, DiCoTraM, in its current version, cannot provide all the statistics required for some pattern analysis applications. Hence, one future direction could be to extend DiCoTraM to monitor in a more fine-grained manner and not only the flows' 5-tuples [156].

Extending DiCoTraM to achieve other network management objectives: It is possible to achieve/enforce various network management objectives/policies (e.g., traffic engineering and network planning) through a network management task, like DiCoTraM, that could integrate those objectives/policies. For instance, as we showed in chapter4, DiCoTraM could coordinates traffic monitoring responsibilities among the routers by maximizing the total flow coverage and also tracks the amount of minimum additional memory required for full flow coverage as two integrated objectives. Hence, another possible future direction could be to extend DiCoTraM in such away that it could achieve some other useful network management objectives (e.g., maximizing the minimum flow coverage).

Providing a framework which includes multi-purpose network management applications: With the new trends in SDN, various network management tasks (e.g., traffic monitoring tasks) can be managed and optimally employed by the network operators within their network enterprises. DiCoTraM could be one example of such tasks. Each of these various network management tasks will be the network management applications of such framework. Such framework could also reduce the complexity of managing the networks through an automated translation of network management policies (applications) into network devices primitives. For instance, *Sharma et. al*, towards providing such framework, proposed an an integrated network management and control system (i-NMCS) framework that combines legacy network management functions such as discovery, fault detection with the end-to-end flow provisioning and control enabled by SDN [181].

More productive resource provisioning: DiCoTraM is provisioning the memory resources of the monitoring devices for incremental upgrades and several factors affect such provisioning such as: routing dynamics, traffic changes, and etc. However, current resource provisioning of DiCoTraM is only for one network management task namely traffic monitoring and it only considers the memory resources of the monitoring devices. A promising future goal would be to integrate other important network management tasks such as: traffic engineering, performance optimization, and etc. with their specific resource requirements (e.g., CPU, memory, etc.) as part of a multi-purpose network management framework so that the outcome of any resource provisioning component will be robust and could be immediately applied by network operators.

BIBLIOGRAPHY

[1] CERT, *Denial of Service Attacks*, June 4, 2001, [online] http://www.cert.org/tech_tips/denial_of_service.html

[2] S. T. Zargar, J. B. D. Joshi, D. Tipper, *A Survey of Defense Mechanisms Against Distributed Denial of Service (DDoS) Flooding Attacks*, IEEE Communications Surveys & Tutorials, vol. 15, no. 4, pp. 2046-2069, November 2013.

[3] J. Mirkovic and P. Reiher, *A taxonomy of DDoS attack and DDoS defense mechanisms*, ACM SIGCOMM Computer Communications Review, vol. 34, no. 2, pp. 39-53, April 2004.

[4] L. C. Chen, T. A. Longstaff, K. M. Carley, *Characterization of defence mechanisms against distributed denial of service attacks*, Computers and Security, vol. 23, no. 8, pp. 665-678, 2004.

[5] K. Lu, D. Wu, J. Fan, S. Todorovic, A. Nucci, *Robust and efficient detection of DDoS attacks for large-scale internet*, Computer Networks, vol. 51, no. 18, pp. 5036-5056, 2007.

[6] *A Roadmap for Cybersecurity Research*, Department of Homeland Security, 2009, [online] http://www.cyber.st.dhs.gov/docs/DHS-Cybersecurity-Roadmap.pdf

[7] S. T. Zargar, J. B. D. Joshi, D. Tipper, *DiCoTraM: A DDoS flooding attack tailored Distributed Coordinated Traffic flow Monitoring*, Elsevier Computers Security (*to be submitted*), December 2013.

[8] S. T. Zargar, A. Clemm, J. B. D. Joshi, *Network and Traffic Monitoring in Software Defined Networks*, IEEE Communications Magazine (Major revision), Submitted May 11, 2013, first round review October 22, 2013.

[9] P. J. Criscuolo, *Distributed Denial of Service*, Tribe Flood Network 2000, and Stacheldraht CIAC-2319, Department of Energy Computer Incident Advisory Capability (CIAC), UCRL-ID-136939, Rev. 1., Lawrence Livermore National Laboratory, February 14, 2000.

[10] S. Ranjan, R. Swaminathan, M. Uysal, and E. Knightly, *DDoS-Resilient Scheduling to Counter Application Layer Attacks under Imperfect Detection*, IEEE INFOCOM'06, 2006.

[11] R. K. C. Chang, *Defending against flooding-based distributed denial of service attacks: A tutorial*, Computer journal of IEEE Communications Magazine, Vol. 40, no. 10, pp. 42-51, 2002.

[12] R. Puri, *Bots and Botnet – an overview*, Aug. 08, 2003, [online] http://www.giac.org/practical/GSEC/Ramneek_Puri_GSEC.pdf

[13] B. Todd, *Distributed Denial of Service Attacks*, Feb. 18, 2000, [online] http://www.linuxsecurity.com/resource_files/intrusion_detection/ ddos–whitepaper.html

[14] J. Liu, Y. Xiao, K. Ghaboosi, H. Deng, and J. Zhang, *Botnet: Classification, Attacks, Detection, Tracing, and Preventive Measures*, EURASIP Journal on Wireless Communications and Networking, vol. 2009, Article ID 692654, 11 pages, 2009.

[15] *Yahoo on Trail of Site Hackers*, Wired.com, Feb. 8, 2000, [online] http://www.wired.com/news/business/0,1367,34221,00.html

[16] *Powerful Attack Cripples Internet*, Oct. 23, 2002, [online] http://www.greenspun.com/bboard/q–and–a–fetch–msg.tcl?msg _id=00A7G7

[17] *Mydoom lesson: Take proactive steps to prevent DDoS attacks*, Feb. 6, 2004, [online] http://www.computerworld.com/s/article/89932/Mydoom _lesson_Take_proactive_steps_to_prevent_DDoS _attacks?taxonomyId=017

[18] *Lazy Hacker and Little Worm Set Off Cyberwar Frenzy*, July 8, 2009, [online] http://www.wired.com/threatlevel/2009/07/mydoom/

[19] *New "cyber attacks" hit S Korea*, July 9, 2009, [online] http://news.bbc.co.uk/2/hi/asia-pacific/8142282.stm

[20] *Operation Payback cripples MasterCard site in revenge for WikiLeaks ban*, Dec. 8, 2010, [online] http://www.guardian.co.uk/media/2010/dec/08/operation–payback – mastercard–website–wikileaks

[21] T. Kitten, *DDoS: Lessons from Phase 2 Attacks*, Jan. 14, 2013, [online] http://www.bankinfosecurity.com/ddos-attacks-lessons-from-phase-2-a-54 20/op-1

[22] Forrester Consulting, *The Trends And Changing Landscape Of DDoS Threats And Protection*, A commissioned study conducted by Forrester Consulting on behalf of VeriSign, Inc., July 2009.

[23] *Worldwide Infrastructure Security Report: Volume VI, 2011 Report*, Arbor Networks, Feb. 1st, 2011, [online] http://www.arbornetworks.com/report

[24] Prolexic Technologies, [online] http://www.prolexic.com/index.php _knowledge–center/frequently–asked–questions/index.html

[25] T. Peng, C. Leckie, and K. Ramamohanarao, *Survey of network-based defense mechanisms countering the DoS and DDoS problems*, ACM Comput. Surv. 39, 1, Article 3, April 2007.

[26] U. Tariq, M. Hong, and K. Lhee, *A Comprehensive Categorization of DDoS Attack and DDoS Defense Techniques*, ADMA LNAI 4093, pp. 1025-1036, 2006.

[27] S. M. Specht, and R. B. Lee, *Distributed Denial of Service: Taxonomies of Attacks, Tools and Countermeasures*, in Proc. of the 17th International Conference on Parallel and Distributed Computing Systems, pp.543-550, 2004.

[28] RioRey, Inc. 2009-2012, *RioRey Taxonomy of DDoS Attacks*, RioRey_Taxonomy_Rev_2.3_2012, 2012. [online] http://www.riorey.com/x-resources/2012/RioRey_Taxonomy_DDoS _Attacks_2012.pdf

[29] C. Douligeris, and A. Mitrokotsa, *DDoS attacks and defense mechanisms: classification and state-of-the-art*, Computer Networks, Vol. 44, No. 5, pp. 643-666, April 2004.

[30] S. Ranjan, R. Swaminathan, M. Uysal, A. Nucci, and E. Knightly, *DDoS-Shield: DDoS-Resilient Scheduling to Counter Application Layer attacks*, IEEE/ACM Transactions on Networking, Vol. 17, No. 1, pp. 2639, February 2009.

[31] *Arbor Application Brief: The Growing Threat of Application-Layer DDoS Attacks*, Arbor Networks, Feb. 28, 2011, [online] http://www.arbornetworks.com/component/docman/doc_download/467 -the-growing-threat-of-application-layer-ddos-attacks?Itemid=442.

[32] BreakingPoint Labs, *Application-Layer DDoS Attacks Are Growing: Three to Watch Out For*, Oct. 4, 2011, [online] http://www.breakingpointsystems.com/resources/blog/application-layer -ddos-attacks-growing/

[33] ha.ckers.org, *Slowloris HTTP DoS*, Retrieved Oct. 19, 2012, [online] http://ha.ckers.org/slowloris/

[34] TrustWave SpiderLab, *(Updated) ModSecurity Advanced Topic of the Week: Mitigating Slow HTTP DoS Attacks*, Jul. 13, 2011, [online] http://blog.spiderlabs.com/2011/07/advanced-topic-of-the-week -mitigating-slow-http-dos-attacks.html

[35] K. J. Higgins, *Researchers To Demonstrate New Attack That Exploits HTTP*, Nov. 01, 2010, [online] http://www.darkreading.com/vulnerability-management/167901026 /security/attacks-breaches/228000532/index.html

[36] S. Shekyan, *Are you ready for slow reading?*, Jan. 5, 2012, [online] https://community.qualys.com/blogs/securitylabs/2012/01/05/slow-read

[37] E. Alomari, S. Manickam, B. B. Gupta, S. Karuppayah, and R. Alfaris, *Botnet-based Distributed Denial of Service (DDoS) Attacks on Web Servers: Classification and Art*, International Journal of Computer Applications, Vol. 49, no. 7, pp. 24-32, Jul., 2012.

[38] J. Lo et al., *An IRC Tutorial*, April, 2003, irchelp.com 1997, [online] http://www.irchelp.org/irchelp/irctutorial.html#part1.

[39] B. Hancock, *Trinity v3, a DDoS tool, hits the streets*, Computers & Security, Vol. 19, no. 7, pp. 574-574, Nov., 2000.

[40] Bysin, *knight.c sourcecode*, 2001, [online] http://packetstormsecurity.org/distributed/ knight.c.

[41] team-cymru Inc., *A Taste of HTTP Botnets*, July, 2008, [online] http://www.team-cymru.com/ReadingRoom/Whitepapers/2008/http- botnets.pdf

[42] J. Nazario, *BlackEnergy DDoS Bot Analysis*, Arbor Networks, 2007, [online] http://atlas-public.ec2.arbor.net/docs/BlackEnergy+DDoS+Bot+ Analysis.pdf

[43] Praetox Technologies *Low Orbit Ion Cannon*, 2010, [online] https://github.com/NewEraCracker/LOIC/

[44] E. Mills, *DOJ, FBI, entertainment industry sites attacked after piracy arrests*, 2012, [online] http://news.cnet.com/8301-27080_3-57362279-245/doj-fbi-entertainment -industry-sites-attacked-after-piracy-arrests.

[45] C. Wilson , *DDoS and Security Reports: The Arbor Networks Security Blog*, Arbor Networks, 2011, [online] http://ddos.arbornetworks.com/2012/02/ddos-tools/.

[46] [online] http://infosecisland.com/blogview/12395–DDoS–Attack –Utilizes–Self–Destructing–Botnet.html

[47] Cisco, *IPS 7.0 Global Correlation*, 2009, [online] http://www.cisco.com/en/US/docs/security/ips/7.0/configuration /guide/ime/ime_collaboration.html

[48] P. Ferguson, and D. Senie, *Network Ingress Filtering: Defeating Denial of Service Attacks that employ IP source address spoofing*, Internet RFC 2827, 2000.

[49] S. Kent, and R. Atkinson, *Security Architecture for the Internet Protocol*, IETF, RFC 2401, November 1998.

[50] S. Kent, and R. Atkinson, *IP Authentication Header*, IETF, RFC 2402, November 1998.

[51] J. Mirkovic, G. Prier, and P. Reiher, *Attacking DDoS at the Source*, in Proc. of the 10th IEEE International Conference on Network Protocols (ICNP '02), Washington DC, USA, 2002.

[52] J. Mirkovic, G. Prier, and P. Reihe, *Source-End DDoS Defense*, in Proc. of 2nd IEEE International Symposium on Network Computing and Applications, April 2003.

[53] T. M. Gil, and M. Poleto, *MULTOPS: a data-structure for bandwidth attack detection*, in Proc. of 10th Usenix Security Symposium, Washington, DC, pp. 2338, August 1317, 2001.

[54] S. Abdelsayed, D. Glimsholt, C. Leckie, S. Ryan, and S. Shami, *An efficient filter for denial-of-service bandwidth attacks*, in Proc. of the 46th IEEE Global Telecommunications Conference (GLOBECOM03), pp. 13531357, 2003.

[55] Mananet, *Reverse Firewall*, [online] http://www.cs3–inc.com/pubs/Reverse_FireWall.pdf

[56] A. John, and T. Sivakumar, *DDoS: Survey of Traceback Methods*, International Journal of Recent Trends in Engineering ACEEE (Association of Computer Electronics & Electrical Engineers), vol. 1, no. 2, May 2009.

[57] R. Chen, J. M. Park, and R. Marchany, *RIM: Router interface marking for IP traceback*, in IEEE Global Telecommunications Conference (GLOBECOM'06), 2006.

[58] B. Al–Duwairi, and G. Manimaran, *Novel Hybrid Schemes Employing Packet Marking and Logging for IP Traceback*, IEEE Trans. Parallel and Distributed Systems, vol. 17, no. 5, pp. 403- 418, May 2006.

[59] S. Savage, D. Wetherall, A. Karlin, and T. Anderson, *Practical Network Support for IP Traceback*, Technical report, Department of Computer Science and Engineering, University of Washington, 2000.

[60] H. Burch, and B. Cheswick, *Tracing anonymous packets to their approximate source*, in Proc. of the USENIX Large Installation Systems Administration Conference, pages 319–327, New Orleans, USA, Decemeber 2000.

[61] J. Glave, *Smurfing cripples ISPs*, in Wired TechnologyNews, 1998, [online] http://www.wired.com/news /news/technology/story/9506.html

[62] Y. C. Wu, H. R. Tseng, W. Yang, and R. H. Jan, *DDoS detection and traceback with decision tree and grey relational analysis*, Int. J. Ad Hoc Ubiquitous Comput., vol. 7, no. 2, pp. 121-136, 2011.

[63] B. Joao, D. Cabrera, and et al., *Proactive Detection of Distributed Denial of Service Attacks Using MIB Traffic Variables A Feasibility Study*, Integrated Network Management Proceedings, pp. 609-622, 2001.

[64] R. Jalili, F. ImaniMehr, *Detection of Distributed Denial of Service Attacks Using Statistical Pre-Prossesor and Unsupervised Neural Network*, ISPEC, Springer-Verlag Berlin Heidelberg, pp.192-203, 2005.

[65] M. Li, J. Liu, and D. Long, *Probability Principle of Reliable Approach to detect signs of DDOS Flood Attacks*, PDCAT, Springer-Verlag Berlin Heidelberg, pp.596-599, 2004.

[66] T. Peng, C. Leckie, and K. Ramamohanarao, *Protection from distributed denial of service attacks using history-based IP filtering*, ICC '03. May , Vol.1, pp: 482- 486, 2003.

[67] H. Wang, C. Jin, and K. G. Shin, *Defense Against Spoofed IP Traffic Using Hop-Count Filtering*, IEEE/ACM Trans. On Networking, vol. 15, no. 1, pp.40-53, February 2007.

[68] M. Abliz, *Internet Denial of Service Attacks and Defense Mechanisms*, University of Pittsburgh, Department of Computer Science, Technical Report. TR-11-178, March 2011.

[69] A. Yaar, A. Perrig, and D. Song, *Pi: A Path Identification Mechanism to Defend against DDoS Attacks*, in IEEE Symposium on Security and Privacy, pp. 93, 2003.

[70] Y. Kim, W. C. Lau, M. C. Chuah, and H. J. Chao, *PacketScore: A Statistics-Based Packet Filtering Scheme against Distributed Denial-of-Service Attacks*, IEEE Trans. On Dependable and Secure Computing, vol. 3, no. 2, pp. 141-155, 2006.

[71] E. Y. K. Chan et al., *Intrusion Detection Routers: Design, Implementation and Evaluation Using an Experimental Testbed*, IEEE Journal on Selected Areas in Communications, vol. 24, no. 10, pp. 1889 - 1900, 2006.

[72] K. Park, and H. Lee, *On the effectiveness of probabilistic packet marking for IP traceback under denial of service attack*, in Proc. of IEEE INFOCOM 2001, pp. 338347.

[73] K. Park, and H. Lee, *On the Effectiveness of Route-Based Packet Filtering for Distributed DoS Attack Prevention in Power-Law Internets*, in Proc. ACM SIGCOMM, August 2001.

[74] A. T. Mizrak, S. Savage, and K. Marzullo, *Detecting compromised routers via packet forwarding behavior*, IEEE Network, pp.34-39, 2008.

[75] K. A. Bradley, S. Cheung, N. Puketza, B. Mukherjee, and R. A. Olsson, *Detecting Disruptive Routers: A Distributed Network Monitoring Approach*, in Proc. of the 1998 IEEE Symposium on Security and Privacy, May 1998.

[76] J. R. Hughes, T. Aura, and M. Bishop, *Using Conservation of Flow as a Security Mechanism in Network Protocols*, in Proc. of the 2000 IEEE Symposium on Security and Privacy, May 2000.

[77] J. M. Gonzalez, M. Anwar, and J. B. D. Joshi, *A trust-based approach against IP-spoofing attacks*, in Proc. of the IEEE PST, pp. 63-70, 2011.

[78] P. Zhou, X. Luo, A. Chen, and R. K. C. Chang, *STor: Social Network based Anonymous Communication in Tor*, in The Computing Research Repository (CoRR), 2011.

[79] S. T. Zargar, and J. B. D. Joshi, *A Collaborative Approach to Facilitate Intrusion Detection and Response against DDoS Attacks*, the 6th Intl Conference on Collaborative Computing: Networking, Applications and Worksharing (CollaborateCom 2010), Chicago, IL, October 9-12, 2010.

[80] M. R. Sharma, and J. W. Byers, *Scalable Coordination Techniques for Distributed Network Monitoring*, in Proc. of PAM, pp. 349-352, 2005.

[81] B. Claise, *Cisco Systems NetFlow Services Export Version 9*, RFC 3954, 2004.

[82] R. Mahajan, S. M. Bellovin, S. Floyd, J. Ioannidis, V. Paxson, and S. Shenker, *Controlling high bandwidth aggregates in the network*, presented at Computer Communication Review, pp.62-73, 2002.

[83] D. Yau, J. C. S. Lui, and F. Liang, *Defending against distributed denial of service attacks using max-min fair server centric router throttles*, IEEE international conference on Quality of Service. 2002.

[84] R. Chen, and J. M. Park, *Attack Diagnosis: Throttling distributed denial-of-service attacks close to the attack sources*, IEEE Int'l Conference on Computer Communications and Networks (ICCCN'05), Oct. 2005.

[85] R. Chen, J. M. Park, and R. Marchany, *TRACK: A novel approach for defending against distributed denial-of-service attacks*, Technical Report TR-ECE-06-02, Dept. of Electrical and Computer Engineering, Virginia Tech, Feb. 2006.

[86] J. Mirkovic, P. Reiher, and M. Robinson, *Forming Alliance for DDoS Defense*, in Proc. of New Security Paradigms Workshop, Centro Stefano Francini, Ascona, Switzerland, 2003.

[87] C. Papadopoulos, R. Lindell, J. Mehringer, A. Hussain, and R. Govindan, *Cossack: Coordinated Suppression of Simultaneous Attacks*, in Proc. of the DARPA Information Survivability Conference and Exposition, Vol. 1, pp. 2 13, Apr. 2003.

[88] T. Anderson, T. Roscoe, and D. Wetherall, *Preventing Internet denial-of-service with capabilities*, SIGCOMM Comput. Commun. Rev., vol. 34, no. 1, pp. 39-44, 2004.

[89] B. Parno et al., *Portcullis: protecting connection setup from denial-of-capability attacks*, SIGCOMM Comput. Commun. Rev., vol. 37, no. 4, pp. 289-300, 2007.

[90] X. Yang, D. Wetherall, and T. Anderson, *TVA: a DoS-limiting network architecture*, IEEE/ACM Trans. Netw., vol. 16, no. 6, pp. 1267-1280, 2008.

[91] X. Yang, D. Wetherall, and T. Anderson, *A DoS-limiting Architecture*, in ACM SIGCOMM, Philadelphia, PA, USA, August 2005.

[92] A. Yaar, A. Perrig, and D. Song, *SIFF: a Stateless Internet Flow Filter to Mitigate DDoS Flooding Attacks*, in Proc. of the 2004 IEEE Symposium on Security and Privacy, pp. 130-143, May 2004.

[93] X. Liu, A. Li, X. Yang, and D. Wetherall, *Passport: secure and adoptable source authentication*, in Proc. of the 5th USENIX Symposium on Networked Systems Design and Implementation (NSDI'08), San Francisco, CA, USA, pp. 365-378, 2008.

[94] X. Liu, X. Yang, and Y. Lu, *To filter or to authorize: network-layer DoS defense against multimillion-node botnets*, in Proc. of the ACM SIGCOMM conference on Data communication (SIGCOMM '08), NY, USA, pp. 195-206, 2008.

[95] X. Yang, *A DoS Limiting Network Architecture*, [online] http://www.cs.duke.edu/nds/ddos/

[96] K. Argyraki, and D. R. Cheriton, *Scalable network-layer defense against internet bandwidth-flooding attacks*, in IEEE/ACM Trans. Netw., 17(4), pp. 1284-1297, August 2009.

[97] F. Huici, *Deployable Filtering Architectures Against Large Denial-of-Service Attacks*, Ph.D. dissertation, Department of Computer Science, University College London, December, 2009.

[98] X. Liu, X. Yang, and Y. Lu, *StopIt: Mitigating DoS Flooding Attacks from Multi-Million Botnets*, Technical Report 08-05, http://www.cs.duke.edu/ xinl/stopit-tr.pdf

[99] G. Kambourakis, T. Moschos, D. Geneiatakis, and S. Gritzalis, *Detecting DNS Amplification Attacks*, in Critical Information Infrastructures Security Lecture Notes in Computer Science, Vol. 5141, pp. 185-196, 2008.

[100] A. Rahul, S. K. Prashanth, B. S. kumarand , and G. Arun, *Detection of Intruders and Flooding In Voip Using IDS, Jacobson Fast And Hellinger Distance Algorithms*, IOSR Journal of Computer Engineering (IOSRJCE), Vol. 2, no. 2, pp. 30-36, July-Aug. 2012.

[101] S. Ranjan, R. Swaminathan, M. Uysal, A. Nucci, and E. Knightly, *DDoS-shield: DDoS-resilient scheduling to counter application layer attacks*, IEEE/ACM Trans. Netw., Vol. 17, no. 1, pp. 26-39, February 2009.

[102] Y. Xie, and S. Z. Yu, *A large-scale hidden semi-Markov model for anomaly detection on user browsing behaviors*, IEEE/ACM Transactions on Networking (TON), Vol. 17, no. 1, pp. 54-65, February 2009.

[103] H. I. Liu, and K. C. Chang, *Defending systems Against Tilt DDoS attacks*, Telecommunication Systems, Services, and Applications (TSSA), pp. 22-27, October 20-21, 2011.

[104] M. Walfish, M. Vutukuru, H. Balakrishnan, D. Karger, and S. Shenker, *DDoS defense by offense*, SIGCOMM Computer Communications Review, Vol. 36, no. 4, pp. 303-314, August 2006.

[105] J. Yu, Z. Li, H. Chen, and X. Chen, *A Detection and Offense Mechanism to Defend Against Application Layer DDoS Attacks*, the third International Conference on Networking and Services (ICNS'07), pp. 54, June 19-25, 2007.

[106] S. Kandula, D. Katabi, M. Jacob, and A. W. Berger, *Botz-4-sale: Surviving organized ddos attacks that mimic flash crowds*, in Proc. of Symposium on Networked Systems Design and Implementation (NSDI), Boston, May 2005.

[107] L. V. Ahn, M. Blum, N. J. Hopper, and J. Langford, *CAPTCHA: using hard AI problems for security*, in Proc. of the 22nd international conference on Theory and applications of cryptographic techniques (EUROCRYPT'03), Eli Biham (Ed.). Springer-Verlag, Berlin, Heidelberg, 294-311. 2003.

[108] G. Oikonomou, and J. Mirkovic, *Modeling human behavior for defense against flash-crowd attacks*, in Proc. of the 2009 IEEE international conference on Communications (ICC'09), pp. 625-630, 2009.

[109] M. Srivatsa, A. Iyengar, J. Yin, and L. Liu, *Mitigating application-level denial of service attacks on Web servers: A client-transparent approach*, ACM Transactions on the Web (TWEB), Vol. 2, no. 3, July 2008.

[110] J. Yu, C. Fang, L. Lu, and Z. Li, *A Lightweight Mechanism to Mitigate Application Layer DDoS Attacks*, in Proc. of Infoscale 2009, LNICST 18, pp. 175191, 2009.

[111] S. R. Devi, and P. Yogesh, *A hybrid approach to counter application layer DDoS attacks*, International Journal on Cryptography and Information Security(IJCIS), Vol. 2, no.2, June 2012.

[112] X. Geng, and A. B. Whinston, *Defeating Distributed Denial of Service attacks*, IEEE IT Professional, 2(4), pp. 36-42, 2002.

[113] *Strategies to Protect Against Distributed Denial of Service (DDoS) Attacks*, Retrieved Oct. 19, 2012, [online] http://www.cisco.com/en/US/tech/tk59/technologies/_white_paper09186a 0080174a5b.shtml

[114] A. D. Keromytis, V. Misra, and D. Rubenstein, *SOS: Secure Overlay Services*, in Proc. of SIGCOMM'02, 2002.

[115] D. G. Andersen, V. Misra, and D. Rubenstein, *Mayday: Distributed filtering for internet services*, in Proc. of USENIX'03, 2003.

[116] J. Yan, S. Early, and R. Anderson, *The XenoService - A Distributed Defeat for Distributed Denial of Service*, in Proc. of ISW 2000, October 2000.

[117] ICANN Report, *DNS Distributed Denial of Service (DDoS) Attacks*, Security and Stability Advisory Committee (SSAC), March 2006.

[118] Y. Huang, and J. M. Pullen, *Countering Denial of Service attacks using congestion triggered packet sampling and filtering*, in Proc. of the 10th International Conference on Computer Communiations and Networks, 2001.

[119] R. M. Mutebi, and I. A. Rai, *An Integrated Victim-based Approach against IP Packet Flooding Denial of Service*, International Journal of Computing and ICT Research, Special Issue Vol. 4, No. 1, pp. 70-80, October 2010.

[120] V. A. Siris, and F. Papaglou, *Application of anomaly detection algorithms for detecting syn flooding attacks*, in Proc. of the IEEE GLOBECOM, 2004.

[121] T. Peng, C. Leckie, and K. Ramamohanarao, *Detecting distributed denial of service attacks using source ip address monitoring*, 2003, [Online] http://www.cs.mu.oz.au/tpeng/mudguard/research/detection.pdf

[122] A. Dainotti, A. Pescape, and G. Ventre, *Wavelet-based detection of dos attacks*, in IEEE Global Telecommunications Conference, GLOBECOM, 2006.

[123] M. Kim, H. Kang, S. Hong, S. Chung, and J. W. Hong, *A flow-based method for abnormal network traffic detection*, in Network Operations and Management Symposium, vol. 1, pp. 599-612, April 2004.

[124] S. T. Zargar, H. Takabi, and J. B. D. Joshi, *DCDIDP: A Distributed, Collaborative, and Data-driven Intrusion Detection and Prevention Framework for Cloud Computing Environments*, the 7th Intl Conference on Collaborative Computing: Networking, Applications and Worksharing (CollaborateCom 2011), October 15-18, 2011, Orlando, FL.

[125] J. Carter, *The Internet of Things: how it'll revolutionise your devices*, techradar, July 2012, [online] http://www.techradar.com/news/internet/the-internet-of-things-how-itll - revolutionise-your-devices-958669

[126] D. Evans, *The Internet of Things [INFOGRAPHIC]*, Cisco Blog, July 2011, [online] http://blogs.cisco.com/news/the-internet-of-things-infographic/

[127] M. Chui, M. Lffler, and R. Roberts, *The Internet of Things*, McKinsey Quarterly, March 2010, [online] http://www.mckinseyquarterly.com/The_Internet_of_Things_2538

[128] R. E. Jurga, M. L. M. Hulboj, *Packet Sampling for Network Monitoring*, Technical report, CERN HP Procurve openlab project, December 2007.

[129] *Cisco NetFlow*, [online] http://www.cisco.com/en/US/products /ps6601/products_ios_protocol_group_home.html

[130] J. Mai, C. N. Chuah, A. Sridharan, T. Ye, and H. Zang, *Is Sampled Data Sufficient for Anomaly Detection?*, In Proc. of IMC, 2006.

[131] A. Kumar, M. Sung, J. Xu, and J. Wang, *Data Streaming Algorithms for Efficient and Accurate Estimation of Flow Distribution*, In Proc. of ACM SIGMETRICS, 2004.

[132] N. Hohn, and D. Veitch, *Inverting Sampled Traffic*, In Proc. of IMC, 2003.

[133] B. Li, J. Springer, G. Bebis, and M. H. Gunes, *A survey of network flow applications*, Journal of Network and Computer Applications, vol. 36, pp. 567-581, 2013.

[134] N. Hohn, and D. Veitch, *Inverting sampled traffic*, IEEE/ACM Transactions on Networking, vol. 14, no. 1, pp. 68-80, 2006.

[135] M. Canini, D. Fay, D. J. Miller, A. W. Moor, and R. Bolla, *Per Flow Packet Sampling for High-Speed Network Monitoring*, In Proc. of COMSNETS, 2009.

[136] V. Sekar, M. K. Reiter, W. Willinger, H. Zhang, R. R. Kompella, and D. G. Andersen, *cSamp: a system for network-wide flow monitoring*, NSDI'08: Proceedings of the 5th USENIX Symposium on Networked Systems Design and Implementation, San Francisco, CA, pp. 233-246, 2008.

[137] K. Bartos, and M.Rehak, *Towards Efficient Flow Sampling Technique for Anomaly Detection*, In Traffic Monitoring and Analysis, LNCS, vol. 7189, pp. 93-106, 2012.

[138] M. Lee, M. Hajjat, R. R. Kompella, and S. Rao, *RelSamp: Preserving application structure in sampled flow measurements*, In IEEE INFOCOM, pp. 2354-2362, 2011.

[139] Y. Zhang, M. Roughan, N. Duffield, A. Greenberg, *Fast Accurate Computation of Large-scale IP Traffic Matrices from Link Loads*, SIGMETRICS '03: Proceedings of the 2003 ACM SIGMETRICS international conference on Measurement and modelling of computer systems, New York, NY, pp. 206-217, 2003.

[140] A. Lakhina, M. Crovella, C. Ditot, *Diagnosing Network-Wide Traffic Anomalies*, In ACM SIGCOMM'04, New York, NY, pp. 219-230, 2004.

[141] A. Lakhina, K. Papagiannaki, M. Crovella, C. Diot, E. Kolaczyk, N. Taft, *Structural Analysis of Network Traffic Flows*, In ACM SIGMETRICS '04/Performance '04, New York, NY, pp. 61-72,2004.

[142] X. Li, F. Bian, H. Zhang, C. Diot, R. Govindan, W. Hong, G. Iannaccone, *MIND: A Distributed Multidimensional Indexing for Network Diagnosis*, In INFOCOM'06, Barcelona, Spain, pp. 1-12, 2006.

[143] A. Greenberg, G. Hjalmtysson, D. A. Maltz, A. Meyers, J. Rexford, G. Xie, H. Yan, J. Zhan, H. Zhang, *A Clean Slate 4D Approach to Network Control and Management*, ACM SIGCOMM Computer Communication Review, vol. 35, no. 5, pp. 41-54, 2005.

[144] H. Ballani, P. Francis, *CONMan: A Step Towards Network Manageability*, ACM SIGCOMM Computer Communication Review, vol. 37, no. 4, pp. 205-216, 2007.

[145] M. Caesar, D. Caldwell, N. Feamster, J. Rexford, A. Shaikh, J. Van der merwe, *Design and implementation of a Routing Control Platform*, NSDI'05: Proceedings of the 5th

USENIX Symposium on Networked Systems Design and Implementation, Berkeley, CA, pp. 15-28, 2005.

[146] M. R. , Sharma, and J. W. , Byers, *Scalable coordination techniques for distributed network monitoring*, In PAM '05: Proceedings of the 6th international conference on Passive and Active Network Measurement, Springer-Verlag, Berlin, Heidelberg, pp. 349-352, 2005.

[147] C. Chadet, E. Fleury, I. Lassous, Herve, M. E. Voge, *Optimal Positioning of Active and Passive Monitoring Devices*, In CoNeXT '05, NY, USA, pp. 71-82, 2005.

[148] K. Park, H. Lee, *On the Effectiveness of Route-Based Packet Filtering for Distributed DoS Attack Prevention in Power-Law Internets*, In ACM SIGCOMM Comput. Commun. Rev., vol. 31, no. 4, pp. 15-26, 2001.

[149] K. Suh, Y. Guo, J. Kurose, D. Towsley, *Locating Network Monitors: Complexity, heuristics and coverage*, In INFOCOM '05, pp. 351-361, 2005.

[150] A. Lakhina, M. Crovella, and C. Diot, *Mining anomalies using traffic feature distributions*, In Proc. ACM SIGCOMM, 2005.

[151] A. Kumar, M. Sung, J. Xu, and E. Zegura, *A data streaming algorithm for estimating subpopulation flow size distribution*, In Proc. of ACM SIGMETRICS, 2005.

[152] R. R. Kompella, S. Singh, and G. Varghese, *On scalable attack detection in the network*, In Proc. IMC, 2004.

[153] S. Venkataraman, D. Song, P. B. Gibbons, and A. Blum, *New Streaming Algorithms for Fast Detection of Superspreaders*, In Proc. NDSS, 2005.

[154] K. Greene, *TR10: Software-Defined networking*, MIT Technology Review, vol. 112, no. 2, Apr. 2009. [online] http://www.technologyreview.com/web/22120/

[155] Open Networking Foundation (ONF), *Software-Defined Networking: The New Norm for Networks*, ONF White paper, Apr. 2012. [online] https://www.opennetworking.org /images/stories/downloads/white-papers/wp-sdn-newnorm.pdf

[156] L. Yuan, C. N. Chuah, and P. Mohapatra, *ProgME: Towards Programmable Network MEasurement*, In Proc. SIGCOMM, 2007.

[157] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, J. Turner, *OpenFlow: enabling innovation in campus networks*, SIGCOMM Comput. Commun. Rev., vol. 38, no. 2, pp. 69–74, March 2008.

[158] K. Hyojoon, and N. Feamster, *Improving network management with software defined networking*, IEEE Communications Magazine, vol. 51, no. 2, pp.114-119, February 2013.

[159] L. Kekely, V. Pu, and J. Korenek *Software Defined Monitoring of Application Protocols*, In Proc. INFOCOM, 2014.

[160] Y. Chen, K. Hwang, W. S. Ku, *Collaborative Detection of DDoS Attacks over Multiple Network Domains*, IEEE Transactions on Parallel and Distributed Systems (TPDS), vol. 18, no. 12, pp. 1649-1662, 2007.

[161] A. Shaikh, A., Greenberg, *OSPF Monitoring: Architecture, Design and Deployment Experience*, In Proc. of Symposium on Networked System Design and Implementation (NSDI), 2004.

[162] G. Varghese, *Network Algorithmics*, Morgan Kaufman, 2005.

[163] C. Estan, G. Varghese, *New directions in traffic measurement and accounting*, SIGCOMM Comput. Commun. Rev., vol. 32, no. 4, pp. 323-336, 2002.

[164] Q. Zhao, J. Xu, Z. Liu, *Design of a novel statistics counter architecture with optimal space and time efficiency*, ACM SIGMETRICS, 2006.

[165] J. Garcia-Vidal, M. March, L. Cerda, J. Corbal, M. Valero, *A DRAM/SRAM memory scheme for fast packet buffers*, IEEE Transactions on Computers, vol. 55, no. 5, pp. 588-602, 2006.

[166] J. Hawkinson, T. Bates, *Guidelines for creation, selection, and registration of an Autonomous System (AS)*, [online] http://tools.ietf.org/html/rfc1930

[167] N. Spring, R. Mahajan, D. Wetherall, *Measuring ISP Topologies With Rocketfuel*, Proc. ACM SIGCOMM'02, Pittsburgh, PA, pp. 133-145, 2002.

[168] M. R. Garey, D. S. Johnson, *Computers and Intractability: A guide to the theory of NP-completeness*, W. H. Freeman and co., New York, 1979.

[169] *Gurobi Optimization Inc.*, Gurobi Optimizer Reference Manual, 2012. [online] http://www.gurobi.com

[170] *Network Simulator 2*, The Network Simulator ns-2: Documentation, 2012. [online] http://www.isi.edu/nsnam/ns/

[171] H. Wang, D. Zhang, K. G. Shin, *Change-Point Monitoring for the Detection of DoS Attacks*, IEEE Transactions on Dependable and Secure Computing, vol. 1, no. 4, pp. 193-208, 2004.

[172] T. Benzel, R. Braden, D. Kim, C. Neuman, A. Joseph, K. Sklower, R. Ostrenga, S. Schwab, *Experience with DETER: a testbed for security research*, 2nd International Conference on Testbeds and Research Infrastructures for the Development of Networks and Communities (TRIDENTCOM'06), 2006.

[173] S. Schwab, B. Wilson, C. Ko, A. Hussain, *SEER: a security experimentation EnviRonment for DETER*, in DETER Community workshop on Cyber Security Experimentation and Test on DETER Community Workshop on Cyber Security Experimentation and Test, 2007.

[174] J. Sommers, H. Kim, P. Barford, *Harpoon: a flow-level traffic generator for router and network tests*, in ACM SIGMETRICS Performance Evaluation Review, vol. 32, no. 1, 2004.

[175] *ISO 3166 Report*, AS Resource Allocations, 2006. [online] http://bgp.potaroo.net/iso3166/ascc.html

[176] H. Rahmani, N. Sahli, F. Kamoun, *DDoS flooding attack detection scheme based on F-divergence*, Elsevier Computer Communications, vol. 35, no. 11, pp. 1380-1391, 2012.

[177] Q. Zhao, Z. Ge, J. Wang, J. Xu, *Robust Traffic Matrix Estimation with Imperfect Information: Making use of Multiple Data Sources*, In Proc. of ACM SIGMETRICS, 2006.

[178] A. Solue, A. Lakhina, N. Taft, K. Papagiannaki, K. Salamatian, A. Nucci, M. Crovella, C. Diot, *Traffic Matrices: Balancing Measurements, Inference, and Modelling*, In Proc. of ACM SIGMETRICS, 2005.

[179] A. Shaikh, M. Goyal, A. Greenberg, R. Rajan, K. Salamatian, *An OSPF topology server: Design and evaluation*, IEEE Journal on Selected Areas in Communications, vol. 20, no. 4, 2002.

[180] H. Dreger, A. Feldmann, V. Paxson, R. Sommer, *Predicting the Resource Consumption of Network Intrusion Detection Systems*, In Proc. of RAID conference, 2008.

[181] P. Sharma, S. Banerjee, S. Tandel, R. Aguiar, R. Amorim, D. Pinheiro, *Enhancing network management frameworks with SDN-like control*, In Proc. of IFIP/IEEE International Symposium on Integrated Network Management (IM 2013), pp. 688,691, 2013.