DECENTRALIZED INFORMATION SHARING FOR DETECTION AND PROTECTION AGAINST NETWORK ATTACKS

BY GUANGSEN ZHANG

A dissertation submitted to the

Graduate School—New Brunswick

Rutgers, The State University of New Jersey

in partial fulfillment of the requirements

for the degree of

Doctor of Philosophy

Graduate Program in Electrical and Computer Engineering

Written under the direction of

Professor Manish Parashar

and approved by

_____

_____

_____

_____

_____

New Brunswick, New Jersey

January, 2006

ABSTRACT OF THE DISSERTATION

Decentralized Information Sharing for Detection and Protection against Network Attacks

by Guangsen Zhang

Dissertation Director: Professor Manish Parashar

Over the last two decades the computing infrastructure has grown dramatically in size, functionality and complexity, and has become an integral part of our lives. However, its pervasiveness and increased visibility have also made it vulnerable and a target of malicious attacks. Current attacks such as distributed denial of service (DDoS) and Internet worms are highly distributed, well coordinated, offensive assaults on services, hosts, and the infrastructure of the Internet, and can have disastrous effects including financial losses and disruption of essential service. As a result, protecting the computing infrastructure from such attacks has become a critical issue that needs to be urgently addressed.

In this thesis, we investigate techniques for decentralized cooperative attack detection and countermeasures. Our objective is to enable early and accurate detection of and reaction to attacks in the network. The key underlying concept is the use of scalable decentralized epidemic algorithms for information sharing and achieving quasi-global knowledge of network attacks. Our proposed distributed

framework for network infrastructure protection builds on a self-managing, robust and resilient peer-to-peer overlay composed of local detection and protection agents that are placed at "strategic" locations in the Internet such as a domain gateway. These agents non-intrusively monitor the immediate network around them for possible attacks. Locally detected network anomalies are used to generate attack alert messages, which are disseminated across the network using gossip mechanisms. A decentralized cooperative detection algorithm is used to aggregate these alert messages to estimate a quasi-global view of the anomalous network behavior, and to detect and react to attacks, both early and effectively.

This thesis first presents a conceptual model that defines the relationships between the level of knowledge in the distributed system and attack detection accuracy. The analysis presented demonstrates the feasibility and effectiveness of gossip based communication mechanisms for cooperative attack detection. A prototype simulation of the framework and its key concepts are presented and applied to detect and defend against DDoS attacks and Internet worms. Results using this simulation demonstrate that the proposed approach is feasible and effective against network attacks.

Acknowledgements

First and foremost, I thank Professor Manish Parashar, my advisor, for his guidance, patience, and encouragement during this research. I am very thankful to Professor Ivan Marsic, Professor Hoang Pham, Professor Wade Trappe, and Professor Yanyong Zhang for being on my thesis committee and for their advice and suggestions regarding the thesis and beyond.

I owe much gratitude to my parents for their unconditional support and love. I am especially grateful to my dear wife, Xiaokun Wang, for her love at all times. She is always emotionally supportive.

Moreover, I would like to thank my colleagues at The Applied Software Systems Laboratory (TASSL) and other friends at Rutgers for their friendship and help, which makes my study at Rutgers enjoyable and fruitful. I am also thankful to staff at the Center for Advanced Information Processing (CAIP) and Department of Electrical & Computer Engineering for their assistance and support.

Dedication

To my Parents, my Wife and my Daughter

Table of Contents

List of Tables

List of Figures

Chapter 1
Introduction

1.1 Motivation

As the Internet grows in size and complexity, its increased visibility and its diversity have attracted a variety of highly damaging attacks. These attacks can have catastrophic affects, including stolen or corrupted data, huge financial losses and even disruption of essential services. For example, large scale epidemics caused by the unleashing of recent Internet worms, such as Code Red [10], have resulted in millions of host computer infections and over 2 billion dollars in financial loss [31]. In recent years, Internet attacks have been appearing with alarmingly regularity. Given their profound impact, protecting the computing infrastructure from such attacks has become a critical issue that needs to be urgently addressed. To address these problems, a viable and efficient network attack detection technique should meet the following requirements:

- *Accurate attack detection.* Attack detection should be as accurate as possible. False positives can lead to inappropriate responses that cause denial of service to legitimate users. False negatives result in attacks going unnoticed.

- *Efficiency.* When an attack is detected, the system should engage in a response that significantly reduces the effectiveness of the attack, regardless of the attack characteristics. Further, the response should be quick, automatic and effective in as many domains on the attack path as possible.

- *Intelligence.* Attack response should employ intelligent packet filtering and

discard mechanisms to reduce the downstream impact of the flood, while preserving and routing non-attack packets.

- *High security.* An attack defense system must ensure that it cannot be misused to degrade or deny service to legitimate clients. It also must be resistant to attempts by attackers to bypass or to disable it.

Recent years have seen a large amount of research in local attack detection and defense mechanisms. Most of them can be classified as either signature based (misuse detection) or abnormal behavior based (anomaly detection) mechanisms. *Misuse detection* techniques perform pattern matching against a database of known attack signatures. *Anomaly detection* techniques establish statistical profiles of network traffic and flag any traffic deviating from the profile as anomalous. Although these local detection techniques have been widely used to detect attacks, protecting networks from intrusions and attacks remains a significant challenge for a number of reasons. First, and perhaps the foremost, is the easy access to new attack tools and the Internet's basic vulnerability to widespread intrusions. These attacks have no common attack signatures and a sophisticated or experienced attacker may change attack patterns frequently, *misuse detection* techniques will not be effective. Second, modern network attacks frequently span multiple sites that are not under a central adminstration, and experienced intruders can often launch simultaneous sessions to attack a system or set of systems. The high variability common in network packet traffic limits the effectiveness of *anomaly detection* techniques. Third, current Internet firewall or Intrusion Detection Systems (IDS) are located at the edges of the local network and have to detect attacks using only partial information about the observed attack.

1.2 Cooperative Network Defense

Local detection mechanisms can have a high rate of false positives (legitimate connections that are misclassified as an attack). In general, current IDS that are located at the local network can not be used to efficiently and accurately detect distributed or coordinated attacks. Instead, various isolated IDS should cooperate and share information about networks to defense against those attacks. Recently, researchers have used data sharing mechanism to improve the accuracy and effectiveness of network detection systems in their work. The design of an effective cooperative attack detection and defense systems that can successfully detect and stop attacks presents significant challenges, while at the same time preserving current performance guarantees to legitimate traffic. Key requirements of such system include:

- *Resilience.* Since all networks are prone to system failures, congestion and attacks, the protection infrastructure must be resilient to temporary network instabilities. Furthermore, it is crucial that the detection and defense infrastructure remain available in the face of attacks such as worm outbreaks, denial of service attacks and other Internet catastrophes.

- *Decentralization.* A decentralized architecture provides for greater flexibility and eliminates single points of failure.

- *Cooperative attack detection and response.* Cooperation among multiple domains in attack detection and response offers several benefits. However, the communications required for cooperation and information sharing must not result in significant network loads.

- *Low performance cost.* Resource requirements of the detection and protection system must not degrade the performance of the deploying network.

Currently, most existing distributed intrusion detection systems process data centrally, despite distributed data collection [64]. This limits their scalability and fault tolerance, and as a result, they still can not efficiently and accurately defend against network attack. To circumvent these shortcomings, hierarchical designs have been introduced [64]. Systems such as Emerald [59] and NetSTAT [74] have a layered structure where attacking events are filtered and preprocessed before they are forwarded to higher levels of the control hierarchy. However, these systems still use dedicated nodes that act as central points of control. As a result, the systems remain vulnerable to faults and have limited scalability. Recently, cooperative mechanisms have been designed to specifically defend against DDoS attacks and Internet worms. Cossack [54] addresses cooperative DDoS detection and Netbait [14] provides a structured overlay for sharing Internet worm information. Although such techniques have improved DDoS and Internet worm detection capability over local attack detection techniques, they don't meet most of the requirements of an efficient defense systems as we discussed above. For example, they are not scalable and resilient to network congestions.

1.3 Research Objectives

In this thesis, we present a decentralized cooperative attack detection and countermeasure infrastructure that addresses the addresses the requirement outlined above. The object of this research is not to investigate specific network attack detection techniques. Our work complements current available network attack detection techniques by integrating them into a decentralized, scalable and resilient cooperative framework. Attack detection nodes are deployed at strategic points (e.g. hosts, routers, gateways) potentially sensitive to attack behavior. These nodes collaborate and exchange information in a peer-to-peer fashion without a

centralized coordinator. Primary innovation is to enable early and accurate detection of and reaction to attacks in the network based on a quasi-global view of the overall network behavior which can be practically achieved in a real distributed system. Key components of this research are:

- *Conceptual Model.* How well an attack detection framework can detect an attack will depend on the amount of knowledge about the system that can be acquired. We conceptually analyze the attack detection accuracy we can achieve through information sharing in real distributed systems where gathering complete knowledge is unfeasible.

- *Gossip Based Communication Mechanism.* Given the large scale of the Internet and critical nature of many of its applications, we need a resilient and scalable communication mechanism to exchange attack information. We design directional gossip mechanisms to fulfill this purpose, while reducing the overhead of information sharing.

- *Distributed Coordination Algorithm.* Instead of detecting attacks at central point of control, each detection node detects the network attacks individually. By using a distributed coordination algorithm, the detection nodes in the countermeasure framework can more accurately and collectively detect and react to network attacks.

- *Detection Overlay.* This distributed decentralized framework presented is built as a peer-to-peer overlay network on top of the Internet and enables isolated local network detection agents at diverse locations to securely share intrusion information. The overlay design facilitates scalable information sharing, manages system heterogeneity participation and is resilient to failure.

- *Case Study.* To demonstrate the feasibility and effectiveness of the proposed

decentralized attack detection framework, we use it to detect DDoS attacks and Internet worms as case studies. Simulation results demonstrate that the proposed mechanism can efficiently detect and protect against these attacks.

Designing an effective and feasible framework for distributed attack countermeasure is a challenging task that involves many algorithms and engineering design issues. In this thesis, we will formalize a conceptual model for this framework, investigate algorithms for the individual detection nodes to acquire quasi-global knowledge about attacks, and discuss design issues for the detection overlay. Other issues investigated include the design of individual detection nodes, attack countermeasures and the communication mechanisms among them. The performance and effectiveness of the designed system is evaluated using simulation.

1.4 Organization of Thesis

The rest of the thesis is organized as follows. Chapter 2 gives an overview of network attacks, discusses related work on DDoS and Internet worm countermeasures, and introduces background information on gossip mechanism and overlay network. Chapter 3 provides a conceptual model for the proposed distributed decentralized attack detection framework. Chapter 4 explains the distributed decentralized network attack detection framework in detail. Chapter 5 describes use of the framework for detecting DDoS attacks. Chapter 6 describes the use of framework to detect Internet worms. Chapter 7 concludes this thesis and outlines future research directions.

Chapter 2
Background and Related Work

In this thesis, we design a distributed decentralized information sharing framework to defend against network attacks. To motivate our cooperative approach, in this chapter we first analyze the issues and limitations of current available network detection approaches. Specifically, we focus our discussion on two kind of network attacks: Distributed Denial of Service and Internet worms. Then the challenges and requirements of a viable cooperative network defense approach are presented. Finally, we provide some background information about concepts and technologies that are used by the cooperative attack detection and countermeasure infrastructure developed in the thesis.

2.1 Network Attacks

Network attacks are primarily a result of software vulnerability, protocol vulnerability or infrastructure vulnerability. Software vulnerabilities are due to the buggy implementation of the network software (e.g., DNS Bind, HTTP Apache, Telnet, SMTP etc.) [36]. Protocol vulnerability is the exploitation of logical loopholes in protocol state models. Infrastructure vulnerabilities are caused by exploiting the inter dependencies among network components to attack network resources that provide critical services. In this thesis, we discuss two types of network attacks to demonstrate the feasibility and efficiency of our decentralized information sharing framework:

- Denial of Service (DoS) Attacks. DoS attacks aim at consuming all the

available network resources such that no more capacity is left to carry the normal traffic and thus normal service is denied. Distributed Denial of Service (DDoS) is a distributed version of DoS where many compromised hosts participate in the attack in a coordinated manner.

- Internet Worm Attacks. In Internet worm attacks, the attackers first probe network information and system resources to identify known vulnerabilities that can be exploited later. Internet worms are launched from inside the machine and replicate themselves aggressively on different machines. A worm is a program that self-propagates across a network exploiting security or policy flaws in widely used services.

As these network attacks are highly coordinated and widely distributed across the global Internet, effective attack detection and protection strategies should be similarly distributed and cooperative. The key challenge is scalably and effectively collecting network attack information to enable accurate and timely detection and protection. In this research, we present a conceptual framework and prototype implementation of a decentralized cooperative detection and protection system that provides this capability. Furthermore, we demonstrate how this mechanism can be used to protect network infrastructure against DDoS attacks and Internet worms.

2.2 Distributed Denial of Service: Background and Related Work

Distributed denial of service attacks (DDoS) pose a great threat to the Internet. A recent DDoS attack occurred on October 20, 2002 against the 13 root servers that provide the Domain Name System (DNS) service to Internet users around the world. Although the attack only lasted for an hour and the effects were hardly

noticeable to the average Internet user, it caused 7 of the 13 root servers to shut down demonstrating the vulnerability of the Internet to DDoS attacks [31]. As one of the most difficult problems in network security, DDoS attacks have become a serious threat to the availability of Internet Services. Distributed denial of service attacks occur when numerous subverted machines (zombies) generate a large volume of coordinated traffic toward a target, overwhelming its resources. DDoS attacks are advanced methods of attacking a network system to make it unavailable to legitimate network users. An example is presented in Figure 2.1. The handler is a host running applications that can be used to initiate attacks by sending commands to agents. The agent is a host that runs processes responsible for receiving and carrying out commands issued by the handler. The communication between the attacker and handler and between the handler and the agents is called the control traffic, while the communication between the agents and the victim is called the flooding traffic. DDoS attacks are becoming an increasing threat to the Internet due to the easy availability of user friendly attack tools, which help to coordinate and execute a large scale DDoS attack. Even an unsophisticated individual can launch a devastating attack with the help of these tools. Available tools (e.g., Trinoo, TFN, TFN2K, Shaft, and Stacheldraht) have been used in DDoS attacks against well-known commercial web sites, such as Yahoo, Amazon and Ebay [20]. Furthermore, attackers need not fear punishment as it is extremely difficult to trace back the attack and locate even the agent machines, let alone the attacker who infected them.

2.2.1 DDoS Attack Strategy

The mechanism of DDoS attacks works as follows: the master sends control packets to previously compromised slaves, instructing them to target a specified victim. The slaves then generate and send high volume streams of messages to the

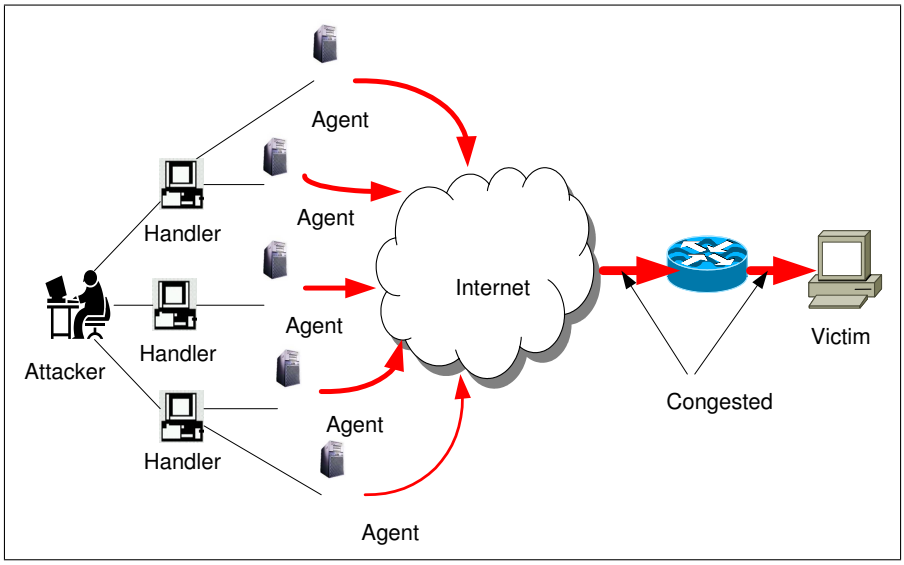

Figure 2.1: Distributed denial of service attack

victim using fake or randomized source addresses, so that the victim cannot locate the attackers. There are several steps in preparing and conducting a DDoS attack:

- *Selection of agents.* The attacker chooses the agents that will perform the attack. These machines need to have some vulnerability that the attacker can use to gain access to them [20]. They should also have abundant resources that will enable them to generate powerful attack streams. At the beginning this process was performed manually, but it is currently automated using scanning tools.

- *Compromise.* The attacker exploits the security holes and vulnerabilities of the agent machines and plants the attack code. Furthermore the attacker tries to protect the code from discovery and deactivation. Self-propagating tools such as the Ramen worm [79] and Code Red [10] have automated this phase. The owners and users of the agent systems typically have no knowledge that their system has been compromised and that they will be taking part in a DDoS attack. When participating in a DDoS attack, each agent program uses only a small amount of resources (both in terms of

memory and bandwidth), so that the users of computers experience minimal change in performance.

- *Communication.* The attacker communicates with any number of handlers to identify which agents are up and running, when to schedule attacks, or when to upgrade agents. Depending on how the attacker configures the DDoS attack network, agents can be instructed to communicate with a single handler or multiple handlers. The communication between the attacker and the handler and between the handler and agents can be via TCP, UDP, or ICMP protocols.

- *Attack.* In this step, the attacker commands the onset of the attack. The victim, the duration of the attack, as well as special features of the attack such as the type, length, TTL, port numbers etc., can be adjusted. The large variability of the properties of attack packets can be beneficial to the attacker, enabling them to avoid detection. IP spoofing is used to mask the source address, and packet type, header fields (all except the destination IP address) and communication channel can all be changed during the attack.

2.2.2 DDoS Attack Classification

There are two main classes of DDoS attacks: bandwidth depletion and resource depletion attacks. A bandwidth depletion attack is designed to flood the victim network with unwanted traffic that prevents legitimate traffic from reaching the victim. It leverages the power of many distributed machines to make a very large number of legitimate requests for a service from a single host. Since these requests are legitimate, the victim cannot refuse to service them, nor can it recognize them as part of the attack before it is too late and its resources are exhausted.

Protocol implementations may have bugs that allow a few malformed packets to severely degrade server or network performance. A resource depletion attack is

an attack that is designed to exploit such vulnerabilities at the victim. This type of attack targets a server or process at the victim making it unable to process legitimate requests for service [13]. The victim can frequently prevent these attacks by implementing various patches to fix the vulnerability, or by filtering malformed packets.

In this thesis we will use the term DDoS attack to refer to packet flooding attacks and not to logical DDoS attacks that exploit certain OS or application vulnerabilities.

2.2.3 DDoS Attacks Tools

There are several known DDoS attack tools. The architecture of these tools is very similar and in fact some tools have been constructed as minor modifications of other tools. In this section, we present the functionality of some of these tools. For presentation purposes we divide them in agent-based and IRC-based DDoS tools.

Agent based DDoS tools

- **Trinoo** [21] is a simple tool used to launch coordinated UDP flood attacks against one or many IP addresses. The attack uses constant-size UDP packets to target random ports on the victim machine. The handler uses UDP or TCP to communicate with the agents. This channel can be encrypted and password protected as well. Trinoo does not spoof source addresses although it can easily be extended to include this capability.

- **Tribe Flood Network** (TFN) [24] can generate UDP and ICMP echo request floods, TCP SYN floods and ICMP directed broadcast (e.g., Smurf). It can spoof source IP addresses and also randomize the target ports. Communication between handlers and agents occurs exclusively through

ICMP_ECHO_REPLY packets.

- **Stacheldraht** [23] combines features of Trinoo (handler/agent architecture) with those of the original TFN (ICMP/TCP/UDP flood and Smurf style attacks). It adds encryption to the communication channels between the attacker and Stacheldraht handlers. Communication is performed using TCP and ICMP packets. Stacheldraht allows automated update of agents using *rpc* and using a stolen account at some site as a cache. New program versions will have more features and different signatures to avoid detection.

- **TFN2K** [11] is the variant of TFN that includes features designed specifically to make TFN2K traffic difficult to recognize and filter. Targets are attacked using UDP, TCP SYN, ICMP_ECHO flood or Smurf attack, and the attack type can be varied during the attack. Commands are sent from the handler to the agent via TCP, UDP, ICMP, or all three at random. The command packets may be interspersed with any number of decoy packets sent to random IP addresses to avoid detection. In networks that employ ingress filtering as described in [30], TFN2K can forge packets that appear to come from neighboring machines. All communication between handlers and agents is encrypted and base-64 encoded.

- The **mstream** [22] tool uses spoofed TCP packets with the ACK flag set to attack the target. Communication is not encrypted and is performed using TCP and UDP packets. Access to the handler is password protected. This program has a feature not found in other DDoS tools. It informs all connected users of access, successful or not, to the handler (s) by competing parties.

- **Shaft** [19] uses TCP, ICMP or UDP flood to perform the attack, and it can deploy all three styles simultaneously. UDP is used for communication

between handlers and agents, and messages are not encrypted. Shaft randomizes the source IP address and the source port in packets. The size of packets remains fixed during the attack. A new feature is the ability to switch the handler's IP address and port during the attack.

- The **Code Red** [10] worm is a self-propagating malicious code that exploits a known vulnerability in Microsoft IIS servers for propagation. It achieves a synchronized attack by pre-programming the onset and abort time of the attack, attack method and target addresses (i.e., no handler/agent architecture is involved).

IRC-based DDoS attack tools

IRC-based DDoS attack tools were developed after agent handler attack tools. As a result, many IRC-based tools are more sophisticated and include typical features that can be found in many agent handler attack tools.

- **Trinity v3** [34] besides the up to now well-known UDP, TCP SYN, TCP ACK, TCP NUL packet floods, introduces TCP fragment floods, TCP RST packet floods, TCP random flag packet floods, and TCP established floods, while randomizing all 32 bits of the source IP address. It also generates TCP flood packets with random control flags set, and supports a wider set of TCP based attacks .

- **Knight** is an IRC-based DDoS attack tool. It is very lightweight and powerful and was first reported in July 2001 [9]. The Knight DDoS attack tool provides SYN attacks, UDP Flood attacks, and an urgent pointer flooder. It is designed to run on the Windows operating systems and has features such as an automatic updater using http or ftp, a checksum generator. The Knight tool is typically installed by using a Trojan horse program called Back Orifice.

2.2.4 DDoS Characteristics and Challenges

There are several features of DDoS attacks that hinder their successful detection and defense:

- DDoS attacks generate a large volume flow to overwhelm the target host. The victim can not protect itself even if it detects this event. So the detection and defense of DDoS should ideally be near the source of the attack or somewhere in the network.

- It is difficult to distinguish attack packets from legitimate packets. Attack packets can be identical to legitimate packets, since the attacker only needs volume, not content, to inflict damage. Furthermore, the volume of packets from individual sources can be low enough to escape notice by local administrators. Thus, a detection system based on single site will have either high false positive or high false negative rates.

- Most DDoS attacks use spoofed IP source addresses. This is done primarily to disguise agent machines, but it can also be used as a means to perpetrate reflector attacks (an indirect attack where intermediary nodes are innocently used as attack launchers). Due to the large scale of a DDoS attack, it is impossible to locate agent machines [13].

- DDoS traffic generated by available tools often has identifying characteristics, making the detection based on statistics analysis possible. However, given the inherently busty nature of Internet, detecting DDoS attacks is error prone.

- DDoS attack code and automated tools for propagation and deployment can be easily downloaded from the Internet. Thus, even novice attackers can launch powerful attacks.

2.2.5 A Taxonomy of the DDoS Detection and Defense

Detection Classification	Related Efforts	Issues
Proactive Mechanism	Secure Overlay Service	Cannot be used for general DDoS
	Ingress/Egress Filtering	Per-flow filtering
	Router based Packet Filtering	Drops legitimate packet
Reactive Mechanism	D-WARD	High false detection rate
	Center Track	High overhead
	History Based IP Filtering	Drops legitimate packets
	MULTOPS	High false detection rate
Tolerance Mechanism	Pushback	Requires new router capability.
	Max-min Fair Server Centric Throttling	Drop legitimate packets
Post Attack Analysis	Trace back	Efficient for DoS, infeasible for large scale DDoS Attacks

Figure 2.2: Taxonomy of DDoS Detection and Defense

The increase in DDoS attacks in the Internet has resulted in many DDoS detection and defense mechanisms. These mechanisms can be classified as: Proactive Mechanism, Reactive Mechanism, Tolerance Mechanism and Post Attack Analysis, as listed in Table 2.2

Proactive Mechanisms

The motivation for proactive defense mechanisms is based on the observation that it is hard to detect DDoS attacks. So instead of detecting the attacks using signatures (attack pattern) or anomaly behavior, these approaches try to improve the reliability of the global Internet infrastructure by adding extra functionality to Internet components to prevent attacks and vulnerability exploitation. The primary goal is to make the infrastructure immune to the attacks and to continue to provide service to normal users under extreme conditions.

Angelos et al. [16] propose Secure Overlay Services that prevent denial-of-service attacks on critical servers by routing requests from previously authenticated clients to the servers via an overlay network. Unlike reactive DDoS defense

approaches, which go into effect when the attacks were detected, this approach is proactive. There are two key ideas behind this approach. First, it distributes the filtering function of the target host across the overlay network, so that the filtering service itself is hard to attack. Second, it introduces randomness and anonymity into the path to the target host, which makes it hard to attack the vulnerability of the routing infrastructure. The assumption behind this approach is that the nodes of the overlay have trust relationships with each other and the attacker does not know all the information about the overlay infrastructure. This work uses the Chord structured peer-to-peer overlay [69]. However, due to the assumptions made, this mechanism can not be used for general DDoS attacks.

Filtering [29] mechanisms use the characterization provided by a detection mechanism to filter out the attack stream completely. In ingress filtering, routers check a packet for its source IP address and block packets that come from an address beyond the routers' possible ingress address range. This requires a router to accumulate sufficient knowledge to distinguish between legitimate and illegitimate addresses. As a result, this approach is most feasible in customer networks or at the border of Internet service providers (ISP) where address ownership is relatively unambiguous. Packet flow filtering is an effective means to counterattack DDoS flows. However, since filtering is rendered per-flow, routers must have sufficient computational power to process large number of flows simultaneously.

The router-based packet filtering (RPF) [56] approach, proposed by Park and Lee, essentially extends the ingress packet filtering function to the Internet core. This approach employs a number of distributed packet filters to examine whether each received packet comes from a correct link according to inscribed source and destination addresses, and BGP routing information. A packet is considered as an attack packet if it is received from an unexpected link and is therefore dropped. However, note that the dropped packet may still be legitimate due to a recent route change. Moreover, the effectiveness of the approach is quite sensitive to the

underlying Internet topology.

Reactive Mechanisms

These mechanisms typically deploy third-party Intrusion Detection Systems (IDS) to obtain attack information and take action based on this information. Consequently their usefulness depends on the capability of the IDS systems. Different strategies are used based on the assumptions made by the IDS systems. If the IDS system can detect the DDoS attack packets accurately, filtering mechanism are used, which can filter out the attack stream completely, even at the source network. If the IDS can not detect the attack stream accurately, rate limiting is used. This mechanism imposes a rate limit on the stream that is characterized as malicious by the IDS.

Tao Peng et al. [57] observed that during DDoS attacks, the IP addresses of the attack packets seldom appears during normal situation. They proposed a mechanism called History-based IP Filtering (HIF) at the edge router to admit the incoming packets based on a pre-built IP address database, which is built using the previous connection history. The key issue in this approach is building an accurate IP address database (IAD), and filtering the incoming packets according to this IAD. In the approach, legitimate requests whose IP addresses are not in IAD will be refused and the mechanism itself is brittle to DDoS attack.

Attacking DDoS at the source is proposed by J. Mirkovic et al. [47]. The proposed system is located at the source network router (either LAN or border router) autonomously detects and suppresses DDoS flows originating at this network. This system observes the outgoing and incoming traffic and gathers lightweight statistics on the flows, classified by destination. These statistics, along with built-in traffic models, define legitimate traffic patterns. Any discrepancy between observed traffic and a legitimate traffic pattern for a given destination is considered to be the signal of a potential DDoS attack. For example, TCP

traffic is monitored and compared to a TCP congestion control model. A TCP stream that is observed violating the behavior of the model is marked as an attack and is subsequently throttled by the edge network's egress router. The amount of throttling is proportional to the flow's deviation from the expected behavior. Similar approaches are applied to other transport protocols. The source router decides to throttle all traffic to the suspected target of the attack and at the same time attempts to separate attacking flows from legitimate flows and identify the attacking machines. This approach has the benefit of preventing malicious flows from entering the network and consuming resources. As DDoS attack flows are not sufficiently aggregate at the source network, this approach has high false positive rates.

Thomer M. Gil et al. propose MULTOPS [32], a heuristic and data structure that network devices can use to detect DDoS attacks. It is based on the assumption that during normal operations on the Internet, the packet rate of traffic going in one direction is proportional to the packet rate of traffic going in the opposite direction. So, a significant disproportional difference between the packets in these two directions indicates an attack. Each network device maintains a multi-level tree, monitoring certain traffic characteristics and storing data in nodes corresponding to subnet prefixes. The tree expands and contracts within a fixed memory budget. An attack is detected by abnormal packet ratio values and offending flows are rate limited. MULTOPS uses only the aggregate packet ratio to model normal flows. Non-TCP flows in a system using MULTOPS can either be misclassified as attack flows, or recognized as special and rate limited to a fixed value. In the first approach, legitimate flows may be harmed, while in the second approach, sufficiently distributed attacks can successfully make use of the allowed transfer rate.

CenterTrack [70] is an architecture proposed by R. Stone, which creates an overlay network of IP tunnels by linking all edge routers to central tracking

routers, and all suspicious traffic is rerouted from edge routers to these tracking routers. When a DoS attack is detected using available attack detection system, routers at the edge of the backbone network are instructed to reroute packets that are addressed to the attack target. The tracking routers can then identify the ingress points of the main attack traffic flows. Edge routers do not have to support input debugging. On the other hand, there is a high storage and processing overhead at the edge routers because they have to log packets in order to identify the attack traffic.

In the MIB variable correlation method [8], network management information is used to detect DDoS attacks. SNMP is a network management protocol that stores information about network devices in a local database called Management Information Base (MIB). Local SNMP agents periodically update variables in MIB. Network administrators can view MIB variables for the traffic sent to local network devices. The assumption is that some MIB variables may indicate attacks if these variables from receiver machines and from sender machines are correlated on a sequential time line. For example, in ICMP ping flood, attackers send out ICMP Echo requests with IP variable "ipOutRequest" in MIB, and later the receivers will reply with an ICMP Echo in which the same set of variables contains "icmpInEchos". The detection algorithm queries the values of several specific MIB variables from local network devices periodically and correlates the relationship of these values. The purpose of the correlation is to reduce false positives in identifying attack traffic. This method is effective for a local test bed and controlled traffic load, however it needs to be evaluated for realistic traffic environments.

Tolerance Mechanisms

Observing that it is hard to distinguish DDoS attack packets from legitimate ones, researchers have proposed attack tolerance mechanisms. These approaches

are based on rate control to enforce fairness in bandwidth allocation and thus minimizes the damage caused by DDoS attacks.

Pushback [37, 45] augments routers with the capability to detect and control flows that create congestion using a Pushback daemon. The routers monitor the packets dropped due to queue restrictions and send them to the Pushback daemon. The Pushback daemon determines whether there is an indication of any attacks by running a detection algorithm. A rate limit is then imposed on the suspicious aggregate traffics. If the router can not control the aggregate itself, it then asks upstream routers to rate-limit or drop the traffic identified as part of DDoS attack. Pushback requires augmentation of routers to perform input debugging. The approach also requires cooperation of different administrative domains.

Throttling [80] is a mitigation approach against DDoS attacks, which prevents servers (web servers in particular) from going down. This approach uses max-min fair server-centric router throttles and involves a server under stress installing rate throttles at a subset of its upstream routers. On installing such throttles, all the traffic passing through the router to the source is rate limited to the throttle rate. This scheme can distribute the total capacity of the server in a max-min fair way among the routers servicing it. This means that only aggressive flows that do not respect their rate shares are punished and not the other flows. This method is still in the experimental stage, however, similar techniques to throttling are being implemented by network operators. The difficulty with implementing throttling is that it is still hard to decipher legitimate traffic from malicious traffic. In the process of throttling, legitimate traffic may sometimes be dropped or delayed and malicious traffic may be allowed to pass to the servers.

Post Attack Analysis

The purpose of post attack analysis is to either look for attack patterns that will be used by an IDS, or to identify attackers using packet tracing. Since the source addresses of flooding packets are faked, various traceback techniques have been proposed to find out the origin of a flooding source. The goal of packet tracing is to trace Internet traffic back to the true source (not spoofed IP address).

Traceback [78, 67, 65, 55, 4, 17] uses router-generated trace back messages to reconstruct the attack paths. Source traceback does not directly address the DDoS problem. It serves as a post attack analysis mechanism against DoS. Packet-based marking is the typical traceback method. Packet-based marking normally comprises of two complementary components: a marking procedure executed by routers in the network and a path reconstruction procedure implemented by the victim. The routers augment IP packets with address marks en-route. The victims can then use information embedded in the IP packets to trace the attack back to the actual source. As attackers change their strategy frequently, analyzing huge amount of traffic logs is time consuming and often useless in detecting new attacks.

Trace back mechanism can help to identify zombies in some situations, however, it is impractical for defending against DDoS attacks for the following reasons. First, during a DDoS attack, the attacker will control thousands of zombies (numbers will increase in the future) to launch an attack. As a result, identifying these zombies is expensive and infeasible. Second, since different network administrators control different section of the global Internet, it would be difficult to determine who would be responsible for providing trace back information. Finally, even if the attacks sources can be successfully traced, stopping them from sending attack packets is another very difficulty task, especially when they are scattered across various autonomous systems(AS).

All the traceback approaches have serious deployment and operational challenges. A sufficient number of routers need to support traceback before it can be effective. Attackers can generate false traceback messages too and some form of authentication of traceback messages is necessary. The victim of a bandwidth attack might not receive significant traceback messages as they may be dropped by overloaded edge routers. In addition, if an attack is very distributed, there may not be enough traceback information to find the attackers.

The mechanisms described so far can be used to protect Internet from particular DDoS attacks. However, given the dynamic nature of Internet as well as the diversity patterns of DDoS attacks, single deployment of these detection systems can not detect general DDoS attacks with high accuracy. To improve the defense efficiency, we need a paradigm shift. Instead of building detection systems that operate in isolation, we present a distributed framework of detection nodes where heterogeneous systems can plug in and cooperate to achieve a better overall protection. Our work complements current available DDoS mechanisms.

2.3 Internet Worms: Background and Related Work

Internet Worms are another major network attacks against Internet. In addition to their primary objective of replicating and propagating themselves, Internet worms can also achieve DDoS attacks by flooding a selected target. David Moore and et. al. provide a nice introduction about Internet worms in their paper [52]. We used some of their material to provide background information about Internet worms. The term "worm" was fist discussed in 1982 by Shoch and Hupp of Xerox PARC [61]. Inspired by the "tapeworm" program described in John Brunner's 1972 novel, "The Shock wave Rider", Schoch and Hupp used the term to describe a collection of benign programs that propagated through a local area

network performing system maintenance functions on each workstation they encountered. The security implications of self-replicating code were not explored by researchers until 1984, when Fred Cohen described the initial academic experiments with computer viruses in his 1984 paper "Computer Viruses - Theory and Experiments" [15]. However, the Internet worm of 1988 was the first well-known replicating program that self-propagated across a network by exploiting security vulnerabilities in host software. This program, which infected several thousand hosts and disrupted Internet wide communication due to its high growth rate, is the modern archetype for contemporary Internet worms [26]. In this section, we provide background information on the major Internet worms released over the last few years.

2.3.1 Internet Worms

Code-Red

On June 18th, 2001, a serious Windows IIS vulnerability was discovered. After almost one month, the first version of Code Red worm that exploited this vulnerability emerged on July 13th, 2001. Due to a code error in its random number generator, it did not propagate well. The truly virulent strain of the worm (Code Red version 2) began to spread around 10:00 UTC on July 19th, 2001. This new worm had implemented the correct random number generator. It generated 100 threads. Each of the first 99 threads randomly chose one IP address and tried to set up connection on port 80 with the target machine. If the system was an English version of Windows 2000, the 100th worm thread would deface the infected systems web site, otherwise the thread was also used to infect other systems. If the connection was successful, the worm would send a copy of itself to the victim web server to compromise it and continue to find another web server. If the victim was not a web server or the connection could not be setup, the worm

thread would randomly generate another IP address to probe. The timeout of the Code Red connection request was programmed to be 21 seconds. Code Red could exploit only Windows 2000 with IIS server installed(it could not infect Windows NT because the jump address in the code is invalid under NT).

Code Red has inspired studies of computer worms. Moore and Shannon have published an empirical analysis of Code-Red's growth, repair, and geography distribution based on observed scan probes to a dedicated class A network [50]. One important aspect of an Internet worm's replication algorithm is how it selects target machines to infect. In the case of Code Red v1, target machines are selected using a random IP address. Due to its use of a static seed in its pseudo random number generator [50], Code Red v1 actually selected target IP addresses in precisely the same sequence at all infected machines. Code Red v2, released shortly thereafter, corrected this by using a better seed and subsequently achieved much greater infection rates. Besides Code Red variants, there is another class of worms which use more sophisticated machinery for selecting potential victims. These are briefly described below:

Nimda

Nimda, uses the following strategy to select victims. 50% of the time an address with the same first two octets is chosen, 25% of the time an address with the same first octet is chosen, and 25% of the time, a random address is chosen. Worms which have affinity for "nearby" nodes, both geographically and in IP address space, have the potential to achieve much higher infection rates due to the performance benefits of network locality.

SQL-Snake

The SQL-Snake was detected on May 20th 2002. This worm scanned for open MS-SQL 7 Servers running on port 1433 by default and exploits machines that

have the default SA (Admin) account without an associated password. The worm scans for IPs of the form A.B.C.D randomly in the following IP ranges where: A = random number not equal to 10,127,172 or 192, B = 0-255, C = 1-255 and D = 1-254. The primary function of the worm is to email passwords and related system information to ixltd@postone.com [7].

SQL-Sapphire

The SQL-Sapphire worm, also known as SQL-Slammer, was released in January 2003, and wreaked significant havoc on the networking infrastructures in under ten minutes. The worm works by generating pseudo-random IP addresses to try to infect with its payload. The worm distinguished itself from its predecessors by its small payload size (single UDP packet of 404 bytes) that enabled a rapid propagation rate in spite of a small susceptible population (75000) [49].

2.3.2 Scan Methods Used by Internet Worms

Probing is the first task performed by worms to find vulnerable hosts. Depending on how worms choose their destinations from a given address space, scan methods can be classified as random scan, routable scan, hitlist scan and divide-conquer scan. In this section, we present a selection of scan methods used by Internet worms.

Selective Random Scan

Instead of scanning the whole IP address space at random, selective random scan attempts to select a set of addresses that are more likely to belong to existing machines can be selected as the target address space. The selected address list can be obtained from the global or the local routing tables. Care needs to be taken to ensure that unallocated or reserved address blocks in the IP address space

are not selected for scanning. Worms can avoid using addresses within these address blocks. Information about such address blocks can be found in several ways. For example, the Bogon list [71] contains around 32 address blocks and the addresses in these blocks should never appear in the public network. IANA's IPv4 address allocation map [1] is a similar list that shows the 8 address blocks which have been allocated. The Slapper [12] (or Apache, OpenSSL) worm made use of these lists to spread rapidly. Worms using the selective random scan need to carry information about the selected target addresses, which can be hundreds of bytes. Carrying more information enlarges the worm's code size and slows down its infection process.

Routable Scan

In addition to the reduced scanning address space, if a worm also knows which of the addresses are routable or are in use, then it can spread faster and more effectively and can avoid detection. This type of scanning technique, where unassigned IP addresses that are not routable on the Internet are removed from the worm's database, is called routable scan. One problem with this type of scan method is that the size of the worm has to be increased as it needs to carry a routable IP address database. This leads to a long infection time resulting in a slow down of worm propagation.

Divide Conquer Scan

In the divide and conquer scan, instead of scanning the complete database, an infected host divides its address database among its victims. For example, after machine A infects machine B, machine A will divide its routable addresses into half and transmit one half to machine B. Machine B can then scan half of routable address database. Using Divide-Conquer scan, the code size of the worm can be further reduced, as each victim will scan a smaller address space. In addition, the

scan traffic generated by the victims is also reduced and more difficult to detect. One weak point of the Divide-Conquer scan is its "single point of failure". During worm propagation, if one infected machine is turned off or crashes, its part of the database will be lost. The earlier this happens, the larger the impact. Several solutions have been used to overcome this problem. One possible solution is the generation of a hitlist, where a worm infects a large number of hosts before passing on the database. Another solution is the use of a generation counter. Each time the worm program is transferred to a new victim a counter is incremented. The worm program decides to split the database based on the value of the counter. A third possible solution is to randomly decide on whether to pass on the database or not.

2.3.3 Worm Defense Classification

In response to recent Internet worm attacks, researchers have proposed different defense mechanisms. According to the methods used to stop the propagation of the worm, these mechanisms essentially can be classified into three classes: Prevention, treatment and containment.

Prevention

Prevention technologies are those that reduce the size of the vulnerable population, thereby limiting the spread of a worm outbreak. In the Internet context, the vulnerability of the population is a function of the software engineering practices that produce security vulnerabilities as well as the social and economic conditions that ensure the homogeneity of the software base. For example, a single vulnerability in a popular software system can translate into millions of vulnerable hosts. While there is an important research agenda, initiated in [66], to increase the security and heterogeneity of software systems on the Internet, widespread software vulnerabilities will persist for the foreseeable future. Therefore, pro-active

prevention measures alone are unlikely to be sufficient to counter the worm threat.

Treatment

Treatment technologies, as exemplified by the disinfection tools found in commercial virus detectors and the system update features in popular operating systems, are an important part of any long-term strategy against Internet pathogens. By deploying such measures on hosts in response to a worm outbreak, it is possible to reduce the vulnerable population (by eliminating the vulnerability exploited by the worm) and reduce the rate of infection (by removing the worm itself from infected hosts). However, for practical reasons, these solutions are unlikely to provide short-term relief during an acute outbreak. The time required to design, develop and test a security update is limited by human time scales, which is usually measured in days and far too slow to have significant impact on an actively spreading Internet worm. Worse, if the installation of such updates is not automated, the response time can be substantially longer. For example, during the Code-Red epidemic, it took sixteen days for most hosts to eliminate the underlying vulnerability and thousands had not patched their systems six weeks later [51]. Finally, creating a central authority for developing, distributing, and automatically installing security updates across hundreds of thousands of organizations will require a level of trust and coordination that does not currently exist [53].

Containment

Finally, containment technologies, as exemplified by firewalls, content filters, and automated routing blacklists, can be used to block infectious communication between infected and uninfected hosts [14, 81]. In principal, this approach can quickly reduce, or even stop, the spread of infection, thereby mitigating the overall threat and providing additional time for more heavy-weight treatment measures to

be developed and deployed. During the Code-Red epidemic, ad-hoc containment mechanisms were the primary means used to protect individual networks (e.g., by blocking inbound access to TCP port 80, or content filtering based on Code-Red specific signatures), or isolating infected hosts (e.g., by blocking the hosts outbound access to TCP port 80). These solutions were implemented manually using existing routers, firewalls, and proxy servers. While these limited quarantines did not halt the spread of the worm, they provided limited protection to portions of the Internet.

There are strong reasons to believe that containment is the most viable of these strategies. First, there is hope that containment can be completely automated, since detecting and characterizing a worm - required before any filtering or blocking can be deployed - is far easier than understanding the worm itself for the vulnerability being exploited, let alone creating software to patch the problem. Second, since containment can potentially be deployed in the network it is possible to implement a solution without requiring universal deployment on every Internet host.

2.3.4 Worm Containment Related Projects

The Early Bird System at UCSD [63] uses a content sifting approach to detect content prevalence and use scaled bitmaps to estimate address dispersion. Sensors are used to sift through traffic on configurable address space zones and report signatures. An aggregator coordinates real time updates from the sensors, coalesces related signatures, and activates network/host level blocking services.

Autograph at Carnegie Mellon University [41] automatically generates signatures from worms propagating with TCP transport. The system analyzes the prevalence of partial flow payloads and produces signatures that exhibit high true positives and low false positives. The system shares port-scan reports among distributed monitors.

Fast Scan Detection at Berkeley ICSI [77] is a scan detection and worm suppression algorithm based on the Threshold Random Walk (TRW) for both hardware and software implementations. The project enhances containment through collaboration among containment devices.

Network Worm Vaccine at Columbia [62] is a reactive mechanism to patch vulnerable software using a collection of sensors that detect and capture potential worm infection vectors based on a set of heuristics to test the resistance of patched application against these infection vectors.

The Shield project at Microsoft [76] installs vulnerability specific and exploit generic network filters in end systems. These filters examine the incoming or outgoing traffic of vulnerable applications, and drops or corrects traffic that exploits vulnerabilities. The system is resilient to polymorphic or metamorphic variations of exploits.

Although these projects have presented efficient worm defense mechanisms, most of them use the local information only to identify the Internet worm attacks. As the Internet worms propagate across multiple geographically distributed domains, local detection based mechanism can not obtain a complete view of the extent and nature of Internet worm epidemics. We leverage the fact that worms typically replicate themselves through remote system exploits which have well-known network signatures that are easily detected by modern intrusion detection systems (IDSs). We use a set of geographically distributed machines to collect probe information using local IDSs then share that information by building an efficient distributed overlay.

2.4 Cooperative Defense Mechanism: Challenges, Requirements and Related Work

As we discussed in Section 2.2 and Section 2.3, network attacks such as DDoS attacks and Internet worms are large scale network attacks that are distributed across the dynamic and heterogeneous Internet. To effectively defend against such cooperative malicious behavior, we need a cooperative mechanism. Isolated local defense systems using various attack detection techniques should cooperate with each other to share their view of network attacks. In this research, we investigate a decentralized information sharing framework which can be used to defend against large scale attacks such as DDoS and Internet worms. According to the discussion in the previous sections, key challenges and requirements include:

- *Reliable and scalable attack information sharing.* When the network infrastructure is under the attack of DDoS or Internet worms, network links are usually congested and failures are common. The cooperative information sharing mechanism should be scalable and reliable to network link failures.

- *Decentralized attack defense.* As DDoS and Internet worms are large scale attacks distributed across the whole Internet, the cooperative defense systems should detect and defend against them in a decentralized manner.

- *Attack and signature identification with short delay.* As discussed in Section 2.3, some aggressive worms propagate quickly to compromise large amount of hosts on the Internet. The individual detection nodes of a cooperative defense system should respond in a quick and timely manner.

Several researchers have started investigating cooperative network defense systems. Systems closely related to this research are discussed below.

DIDS [64] is a distributed intrusion detection system consisting of host managers and LAN managers that are responsible for distributed data monitoring

and sending notable events to the director. The centralized director then analyzes these events to determine the security state of the system as a whole. The centralized director is clearly the bottleneck to the distributive approach of DIDS. Furthermore as there is only one level in the hierarchy with all host and LAN managers reporting to a single director, DIDS thus lacks scalability and has a single point of failure.

Emerald [59] is a system targeted towards the exchange of security incident related information between different domains or large networks. It consists of a layered architecture that provides a certain abstraction, and requires the adjustment of parameters relevant to the trust relationships between cooperative parties. The hierarchical information sharing architecture limits its scalability.

Dshield [25] is a platform that allows firewall users to share intrusion detection information. Users contribute intrusion detection information by running client programs that collect local firewall log files and submit them to the DShield team via email or a web-based interface. Contributed log files are stored in a centralized database which can then be queried to determine if a particular machine has been compromised and to help produce summary reports of various Internet worm epidemics. The system collects information in a centralized database and can not respond to attacks in real time.

NetBait [14] is a distributed system that provides detailed information about network intrusions. It collects data from geographically located machines, which use traditional intrusion detection systems (such as Snort [60]) to discover worm attacks. The goal of NetBait is to provide accurate information to identify infected hosts and expedite the process of worm containment and cleanup.

The cooperative mechanisms listed above can not be used to efficiently defend against large scale network attacks such as DDoS and Internet worms. Despite collecting network attack information from distributed sites, these systems process information and detect attacks at a centralized node. As a result, these systems

are not scalable and resilient. The framework presented in this thesis seeks to address the problems with these approaches. In the following sections, we provide some background information about underlying concepts and technologies.

2.5 Gossip Based Mechanism for Decentralized Information Sharing

To design an effective distributed decentralized network attack detection framework in a large distributed environment, a scalable and resilient information communication and sharing mechanism is necessary. This mechanism should be easy to deploy, robust, and highly resilient to failures. Gossip based mechanisms have been proposed as a potentially effective solution for this purpose.

Gossip based mechanism build upon epidemic algorithms, which mimic the spread of a contagious disease. One may consider dissemination of information in a network to be similar to the spread of a rumor or of an infectious disease in a society, which have been extensively studied by applied mathematicians. Once it has started, an epidemic is hard to eradicate: it only takes a few people to spread a disease, directly or indirectly, to the community at large. An epidemic is also highly resilient in that even if many infected people die before they transmit the contagion or they are immunized, the epidemic continues to propagate through the population. It is possible to adjust the parameters of an epidemic algorithm to achieve high reliability despite process crashes and disconnections, packet losses, and a dynamic network topology.

Epidemic algorithms have been applied to solving several practical problems like database replication, failure detection and resource monitoring. A large body of theoretical work is also available due to the general importance of understanding epidemics and its close relation to random graph theory. Alan Demers et al. [18] have designed an epidemic algorithm for spreading updates of database

that is replicated at many sites in a large, heterogeneous, slightly unreliable and slowly changing network. Compared with a deterministic algorithm, this randomized algorithm is very simple and requires few guarantees from the underlying communication system, yet it ensures that the effect of every update is eventually reflected in all replicas. Implementations of this algorithm demonstrate that it can solve long standing problems of high traffic and database inconsistency. Robbert van Renesse et al. [73] have presented a failure detection service based on a gossip protocol. The service provides accurate failure detection with known probability of a false detection, and is resilient to both transient message loss and permanent network partitions, as well as host failures. The service uses two separate protocols that automatically take advantage of the underlying network topology, and scale well in terms of the number of members. Astrolabe [72] is an information management service developed at Cornell University. By combining gossip based communication mechanisms with an explicit hierarchy at the nodes of a distributed applications, Astrolabe collects large-scale system state, permitting rapid updates and providing on-the-fly attribute aggregation. These approaches have been observed to scale well and self-organize, which are key requirements of a large scale, highly dynamic, distributed applications.

The power of gossip is that information reaches its destination by following a diversity of paths. In effect, every process is a potential source of data for every other process, and over time the number of possible routes by which information from process p might travel to process q increases exponentially. For example, suppose that process p has detected an event of interest, after t time units, $O(2^t)$ processes will know that event [72]. When a system comes under attack, an adversary may manage to corrupt a few messages or disrupt a region of the network. However, if any connectivity remains at all, the gossip exchange of data will eventually prevail, and data stored within the infrastructure will reach all sites in the system. Thus, gossip protocols are relatively tolerant of denial of

service attacks.

Given these features of the gossip based communication mechanism, it is potentially valuable to be used in the decentralized network attack detection framework for information sharing that is investigated in this research. In this thesis, we will theoretically discuss the detection capability that can be achieved using this mechanism. We then design a framework using the mechanism to achieve effective and accurate cooperative attack detection. The section below will provide mathematics foundation of this mechanism. Several mathematics models exist for epidemic, our discussion is based on the infect forever model [28].

Information Dissemination Model

The epidemic algorithm we used in this thesis is similar to the infect forever model, in which infected individuals remain infectious throughout. A quantity of interest is the number Z_r of individuals infected prior to round r. Two key measures of the "success" of an epidemic dissemination are proportion of infected processes defines as $Y_r = Z_r/n$ and the latency for a rumor to reach other processes defined as number of rounds R necessary to infect the entire system. In this research we use the analysis result of [28] as follows:

- The number Z_r of individuals infected prior to round r, assuming that infectious individuals try to contaminate f other members in each round, the approximate formula for the first measure - the expected fraction of infected members after r rounds is:

  $$Y_r \approx \frac{1}{1 + ne^{-fr}}.$$

  Thus, the ratio of infected individuals to uninfected individuals increases exponentially on average, by a factor of e^f in each round.

- The latency for a rumor, R, to reach every process, assuming that each infectious process tries to contaminate f other processes in each round,

Boris Pittel [58] showed that this number R satisfies:

$$R = \log_{f+1}(n) + \frac{1}{f}\log(n) + O(1).$$

Thus the epidemic spreads quickly, taking at most a logarithmic number of steps to reach every process.

2.6 Overlay Networks

An overlay network is an isolated virtual network deployed over an existing network [6]. It is composed of individual nodes (e.g. hosts, routers, gateways), and tunnels. Tunnels are paths in the base network, and links in the overlay network. Individual nodes can participate in more than one overlay at the same time or in a single overlay in multiple ways. As a result, wherever there is a physical path in the underlying network, there can exist a link in the overlay network. Having multiple available links increases the flexibility of the network, and a more flexible network is less likely to be susceptible to attacks. Furthermore, by building a large-scale, self-organizing and resilient overlay network on top of the Internet, the peer nodes of the overlay network can deliver attack information with speed and reliability [42]. As overlay networks provide multi path routing other than the underlying physical network routing, they are highly resilient to disruption and posses the ability to deliver messages even during large scale failures and network partitions.

Overlay networks have been widely used in peer-to-peer data sharing and content distribution applications, which are used by millions of users and represent a large fraction of the traffic in the Internet. There are two basic types of overlays: unstructured and structured.

Unstructured overlays, in which peers are connected in an uncontrolled fashion, do not impose any constraints on the node graph. For example, each node can

choose any other node to be its neighbor in the overlay. One example application of unstructured overlay is Gnutella [2].

Structured overlays, in which peers are organized in a controlled manner, impose constraints on the node graph. Structured overlay networks provide location-independent routing mechanism as the basic component, on top of which high level services can be built. One such example is Chord [69].

In this thesis, we build an unstructured overlay network consisting of network attack detection nodes across the whole Internet. This overlay is used and maintained to provide an effective network attack countermeasure infrastructure.

2.7 Summary

Network attacks have already become a major threat to the stability of the Internet. As current attacks are highly distributed, isolated attack detection systems can not detect them efficiently and accurately. In this chapter, we discussed the characteristic of the DDoS attacks and Internet worms, currently used approaches to address these attacks, and the limitations of these approaches which prevent them from efficiently and effectively detecting and defending against these attacks. A distributed and decentralized attack detection and protection framework requires a resilient and scalable communication mechanism to share attack information. We introduced the gossip based communication mechanism, which has significant advantages for this purpose. Finally, the overlay network technology used to build this framework was discussed. In Chapter 3, we will discuss the conceptual model underlying the proposed detecting and defense framework.

Chapter 3
Decentralized Information Sharing

As we discussed in Chapter 1, in order to scale with exploding network sizes, it is imperative that attack detections are distributed and cooperative. In this thesis, we investigate a distributed decentralized attack detection framework. A critical issue is acquiring sufficient knowledge about the network attacks. Therefore, a conceptual model that defines the relationships between the level of knowledge in the cooperative detection framework and attack detection capability that can be achieved, is necessary and helpful. In this chapter, we describe such a decentralized information sharing model for a cooperative network defense overlay that aims to detect a wide range of network attacks ranging from distributed denial of service attacks to large scale Internet worms. We first define the knowledge requirements for cooperative attack detection. We then describe the relationship between the level of knowledge and associated attack detection capability. We then discuss the nature of the attack detection capability that can be achieved with knowledge acquired using a gossip based communication mechanism. Finally, we discuss aggregation latencies when using gossip based communication mechanisms to achieve the knowledge for cooperative attack detection purpose.

3.1 Knowledge Requirement for Attack Detection

The key requirement for a decentralized distributed cooperative attack detection framework is the capability to acquire sufficient knowledge about the distributed

attacks. If all the nodes in this distributed framework have common knowledge [33] about the network attack behaviors, which means that every detection node knows the overall network attack behavior, and it also knows that all other nodes have same knowledge about the network attack behaviors as itself, then network attacks can be perfectly detected. However, to achieve common knowledge, distributed detection nodes should coordinate using synchronized and reliable communication, which is not possible in a practical distributed system. Instead of achieving perfect common knowledge, we investigate the level of knowledge that can be practically achieved given different communication and coordination mechanisms, and the associated attack detection capability that can be achieved given these different levels of knowledge. Therefore, a formal definition of knowledge hierarchy will give us a basis for discussing these relations. In this section, we define these levels of knowledge.

In a decentralized distributed intrusion detection system, each detection node i will only have a partial view of the system. We assume that this view is correct and can be trusted. This partial view is a fact represented as p_i. We define the overall knowledge as $p \equiv \wedge_{i \in G} p_i$, where G represent all the attack detection nodes. Based on the discussions in [33], the different levels of knowledge in the system can be defined as the follows:

- Local Knowledge. Each detection node i only has its own observation, i.e., p_i. In this case, the knowledge is partial and therefore the detection based on this knowledge will be error prone.

- Distributed Knowledge. Each detection node not only has its own observation p_i, it also has some knowledge p_j, where $j \neq i$. However, there is knowledge p_k which is not known to i, ($k \in G$ and $k \neq i \neq j$).

- Quasi-global Knowledge. Every detection node has knowledge p, but this is the knowledge about the system from t seconds before. If the attacks take

a time longer than t, this Quasi-global knowledge can be used by network defense system to make decisions to efficiently defend against these attacks.

- Global Knowledge. Every detection node has knowledge p.
- Common Knowledge. Every detection node has knowledge p, and this node also knows that other nodes have the knowledge p, know that other nodes know other nodes know the knowledge p.

3.2 Levels of Knowledge and Attack Detection Capability

In Table 3.1, we list the relationships between attack detection capabilities that can be achieved and knowledge level attainable using different communication mechanisms. In a distributed decentralized attack detection system, initially, each individual detection node only has *local knowledge* about network behaviors. As a result, such systems will let a large number of distributed network attacks to pass unnoticed. Using only unreliable communication mechanism between these detection nodes, system can achieve *distributed knowledge*. In this case, each detection node only gets a partial view of the distributed attacks, and therefore, the attack detection has high false rates. Using asynchronous, reliable communication mechanism, system can achieve *quasi-global knowledge*. With this knowledge, every detection node can acquire sufficient information about attacks and as a result, the attacks can be detected effectively. However, there is a delay between the time an attack occurs and the time it is detected. Better attack detection capability requiring higher levels of knowledge in the knowledge hierarchy. To acquire *global knowledge*, we need to build a synchronous framework. Such a framework will incur significant cost but it can achieve a better attack detection. Ideally, if the system can achieve *common knowledge*, then all the network attacks could be perfectly detected. However, it has been shown that this can not be achieved in real distributed systems [33].

From the discussion above, we see that the capability of the decentralized cooperative network attack defense framework depends on the communication mechanism used to acquire the knowledge about network attacks. When the network is under attacks, such as DDoS or Internet worms, the network bandwidth is critical resource. The communication mechanism selected for this defense framework should introduce minimum overhead while providing sufficient information to enable attack detection.

3.3 Attack Detection Using Gossip Based Communication Mechanism

Observing from the Table 3.1, we see that effective attack detection and defense still can be achieved with quasi-global knowledge (Refer Table 3.1). Asynchronous communication mechanisms can be used to achieve this with low overhead. Therefore, it is feasible and efficient to design a distributed decentralized attack detection framework to achieve quasi-global knowledge. The key is to select the appropriate communication mechanism. As both the DDoS attacks and Internet worms only need a short time to achieve significant damage to the network infrastructure, the delay δ in acquiring quasi-global knowledge should be less than the short duration of these network attacks.

Based on the discussion in Section 2.5, gossip based communication mechanism is resilient and scalable, which makes it suitable for information sharing purpose in large scale and failure prone systems.

3.4 Latency of the Network Attack Information Aggregation Using Gossip Mechanisms

In addition to scalability and resilience, another key requirement for a cooperative network defense infrastructure is fast abnormal behavior detection and defense. If the communication mechanism used to share information requires a long time for every defense node to acquire knowledge about the network attack behavior, the attacks may have already caused significant damage to the network infrastructure before that nodes can initiate a defense. In this Section, we investigate the aggregation speed when using gossip based communication mechanism to collect network attack information. The gossip based attack information aggregation mechanism(discussed in more detail in Chapter 4) is quite similar to that of broadcasting by means of the push-pull anti-entropy epidemic protocol presented in [18]. Our analysis is based on a modified and generalized push-pull anti-entropy protocol.

```
// the active process of the protocol on node ni
do forever
wait(getWaitingTime())
nj = selectRandomNeighbor()
// perform elementary aggregation step
send xi to nj
receive xj from nj
xi = aggregate(xi ; xj )
// reply on node nj
receiveApproximation(xi; ni)
send xj to ni
xj = aggregate(xj ; xi)
```

Figure 3.1: Push-pull anti-entropy aggregation protocol

In the push-pull anti-entropy gossip model, every node participating in group

communication either periodically sends its own information to its neighbors or queries its neighbors to acquire up to date information. Let us assume that each network defense node n_i maintains a neighbor list, which is a random and uniform sample of the whole overlay network. This node has a numeric attribute a_i, representing confidence with which the detection node suspects an attack. Aggregation is performed over the set of these values. Node n_i also stores an approximation x_i of the aggregate. The algorithm for the push-pull anti-entropy gossip is illustrated in Figure 3.1. We introduce the following notations:

$$\mu_i = \mu_{\alpha_i} = \frac{1}{N} \sum \alpha_{i,k}$$

$$\delta_i^2 = \delta_{\alpha_i}^2 = \frac{1}{N} \sum (\alpha_{i,k} - \mu_i)^2$$

Here, μ_i is the target value of the protocol in round i, k is the index of the node, α_i is the information to be aggregated, N is the number of nodes in the attack information sharing overlay network, and δ_i^2 is a variance. Without loss of generality we will assume that the common expected value of the elements of α_0 is zero. The purpose of this assumption is to simplify our expressions. In particular, for any vector α, if the elements of α are independent random variables with zero expected value then

$$E(\delta_\alpha^2) = \frac{1}{N} \sum E(\alpha_k^2)$$

Furthermore, the elementary variance reduction step in which both selected elements are replaced by their average does not change the sum of the elements in the vector, so $\mu_i \equiv \mu_0$ for all cycles $i = 1, 2, ...$ This property is very important because it guarantees that the algorithm does not introduce any errors into the approximation. This means that from now on we can focus on variance. Clearly, if the expected value of δ_i^2 tends to zero with i tending to infinity, then the variance of all vector elements will tend to zero as well, so the correct average μ_i will be

approximated locally with arbitrary accuracy by each node. As proved in [3], the expected value of variance reduction during one cycle is given by

$$E(\delta_{i+1}^2) \approx E(2^{-\phi})\frac{1}{N}\sum E(\alpha_{i,k}^2) = E(2^{-\phi})E(\delta_i^2)$$

where

$$E(2^{-\phi}) = \frac{1}{2\sqrt{e}}$$

The formulae above tell us that the gossip based information sharing mechanism will converge exponentially toward global knowledge of network attack behavior distributed across the network. In a real deployment of this communication mechanism, we need to tune some parameters to make the gossip based communication mechanism both efficient and low cost. If the attacks are aggressive and propagate very quickly, the information aggregation interval should be very short. Furthermore, for a short aggregation interval, the mechanism will occupy a bigger portion of network bandwidth than for a longer information aggregation interval. In a real design, we should select these parameters to balance the speed of the information aggregation convergence and the cost of the communication. Chapter 5 and Chapter 6 have detailed discussion about how these parameters are selected to defend against DDoS and Internet worms. The analysis described in this section is based on the assumption that the underlying overlay is "sufficiently random". More formally, this means that the neighbor selected by a node when initiating communication is a uniform random sample among its peers. The discussion of impact of the generic overlay network topology on the aggregation scheme is presented in [3]. It shows that the aggregation scheme have same performance with proper selected parameters.

3.5 Quasi Global Knowledge about the Network Attacks

Since common knowledge cannot be attained in practical distributed systems, it is natural to ask what states of knowledge can be obtained by the gossip based communication mechanism. From the previous Section, the messages sent out using gossip based communication mechanism are not guaranteed to be received by all the network defense nodes of the overlay network. There are variables $\varepsilon(i)$ and p_i, where i is the number of the gossip, such that all the defense nodes get the knowledge of the network attacks between t_0 and $t_0 + \varepsilon(i)$ time units with the probability p_i. As it is hard to compute the p_i, let's consider the variable q_i, which denotes the probability that one overlay node does not get the global network attack information in round i. Then we have $q_0 = 1 - 1/N$. Here, we express q_{i+1} as a function of q_i. Clearly, a defense node will not know the global network attack information in round $i + 1$ if the following is true:

1. It did not know it in round i.

2. The neighbor node it chose did not know it either.

Formally, we can get the following equation:

$$q_{i+1} = q_i q_i (1 - \frac{1}{N})^{N(1-q_i)}$$

As $(1 - \frac{1}{N})^{N(1-q_i)} < 1$, from this equation it follows that

$$q_{i+1} < q_i^2 < q_0^{2^{i+1}} = (1 - \frac{1}{N})^{2^{i+1}}$$

This result suggests that q_i decreases super-exponentially.

Thus we can consider the state of knowledge of the cooperative defense overlay where the message m is broadcast using the gossip based communication mechanism. The communication mechanism will guarantee that every messages will

eventually reach all the defense nodes. In each round of the gossip based aggregation of the network attack information, an individual defense nodes knows that every other defense node either has already received the global network attack information or will eventually get the global network attack information. Further, in the time interval $\varepsilon(i)$, all the overlay defense nodes will get the global network attack behavior with probability p_i. In Figure 3.2, we illustrate the speed of convergence of the gossip based aggregation mechanism for different N, the number of overlay nodes. The X axis is the number of gossip round and the Y axis is the probability $q_i = 1 - p_i$. For an overlay network of size 10000 nodes, it takes around 15 rounds for the mechanism to converge. For example, if we select the interval between two rounds as 10 seconds, then we need about 2.5 minutes for all the nodes to get the attack information. The interval between two gossip rounds can be tuned to balance the performance costs and defence efficiency. According to David Moore's work [53], as shown in Figure 3.3, most attacks are longer than 5 minutes in duration. As a result, given appropriate parameters, the gossip based mechanism can be effective against network attacks.

3.6 Summary

In this chapter we presented a conceptual model for decentralized information sharing for a cooperative network defense framework. We discussed the knowledge requirement for such a framework to efficiently defend against DDoS attacks and Internet worms. Based on the analysis of the relationship between the knowledge level and associative attack detection capability, gossip based communication mechanism can be used to achieve proper knowledge level for efficient attack detection. Finally, we discussed the latency of the gossip based communication mechanism and defined the knowledge achieved using this mechanism.

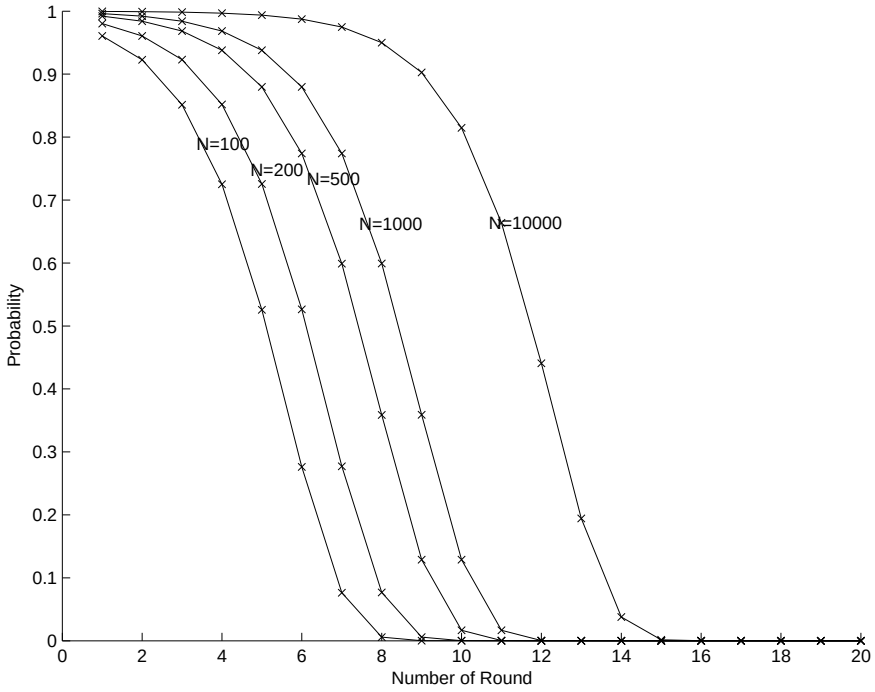

Figure 3.2: Convergence speed of aggregations

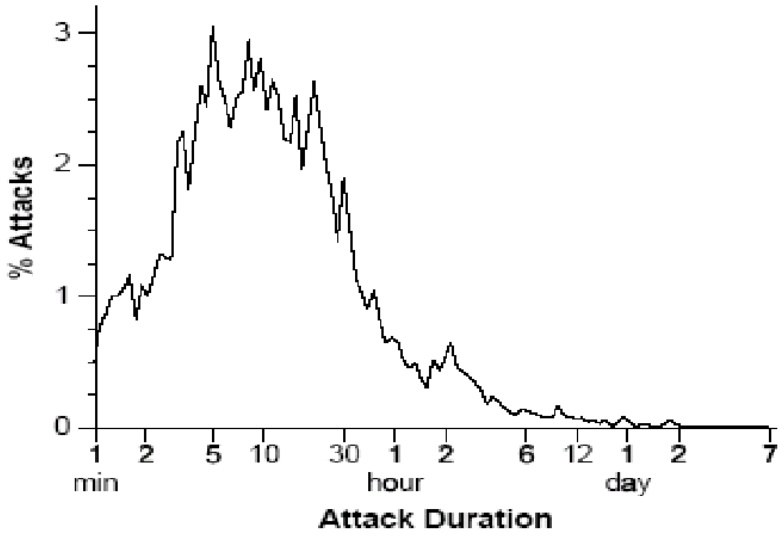

Figure 3.3: Probability density of attack durations

Table 3.1: Relation between attack detection and knowledge

Hierarchy of knowledge	Type of detection that can be achieved	Detection accuracy and cost
Common Knowledge	Using synchronized, timely, coordinated communication between individual detection nodes, the system can know all the information about attacks. As a result, the system can achieve perfect detection and defense.	Perfect detection, but it is impossible in real system.
Global Knowledge	Using synchronized, but not coordinated communication between individual detection nodes, every detection node still has all the information about attacks. However as they are not coordinated, attacks are not detected simultaneously.	High detection accuracy, high cost on information collection.
Quasi-Global Knowledge	Using asynchronous communication between every detection node, the system still can acquire the full information about attacks. However, the delay δ in acquiring quasi-global knowledge should be less than the durations of the attacks.	Effective detection, communication cost is acceptable if system well designed with proper parameters.
Distributed Knowledge	Using unreliable communication mechanism between detection nodes, every detection node can only get partial view of the distributed attacks. As a result, the detection has a high false detection rate.	High false detection rate, communication cost depends on the system design.
Local Knowledge	Based on local observations of the network attack behaviors thus has high false rate when used to detect distributed network attacks,	Can not effectively detect distributed network attacks.

Chapter 4

A Framework for Decentralized Cooperative Detection and Protection for Network Attacks

Based on the conceptual model discussed in Chapter 3, it is feasible to build a cooperative defense systems to defend against network attacks. In this chapter, we present a design of such a distributed decentralized detection and protection framework that meets the requirement discussed in Section 2.4. In this framework, a number of local detection nodes are placed at "strategic" locations in the Internet, and they non-intrusively monitor and analyze the passing traffic for possible attacks. The attack detection mechanism using this framework includes two key stages. In the first stage, each local detection node detects traffic anomalies using various intrusion detection mechanisms. Due to the dynamic and distributed nature of Internet attacks, detections based on these mechanisms alone will have high false detection rates. In the second phase, we enhance the accuracy of the detection by using gossip based communication mechanism to share information among individual detection nodes. To enhance the security and reliability of information sharing, our system is built on an overlay network composed of local detection nodes, which are routers with attack detection and attack packets filtering functionality.

We discuss different aspects of this framework in detail in the following sections. In Section 4.1, we describe the components of this framework. The procedure to detect attacks using this framework is discussed in Section **??**. A detail discussion about the local detection and information sharing mechanism is presented in Section 4.2 and Section 4.3. The advantages and concerns of the

presented framework are discussed in Section 4.4. The uses of the framework for defending against DDoS and Internet worms are presented in Chapter 5 and Chapter 6.

4.1 Decentralized Information Sharing Overlay Framework

As discussed in Section 2.6, overlay networks have been shown to be highly resilient to disruption, and possess the ability to deliver messages even during large scale failures and network partitions [6]. Therefore, our decentralized cooperative detection framework is designed as an overlay. The detection overlay is a dynamic infrastructure composed of a diverse collection of nodes located at critical locations that can monitor network attacks and collect meaningful information to detect network attacks locally. The framework then enables information sharing aimed at improving attack detection capability for all participants. The overall architecture is illustrated in the Figure 4.1. The key functionalities of a detection node in the overlay include:

- Local Network Attack Detection: Each overlay node monitors the immediate network around it for possible attacks. Alert messages are generated when abnormal network behaviors are detected.
- Information Sharing and Global Detection: Alert messages are disseminated using a gossip protocol based on the epidemic algorithm across the Internet. Each overlay node aggregates these alert messages to make a global decision on the occurrence of network attacks.
- Cooperative Defense. Finally, overlay nodes cooperate with each other to defend against confirmed network attacks.

In our approach, the individual detection nodes of the detection framework coordinate with each other to provide the information necessary to detect and

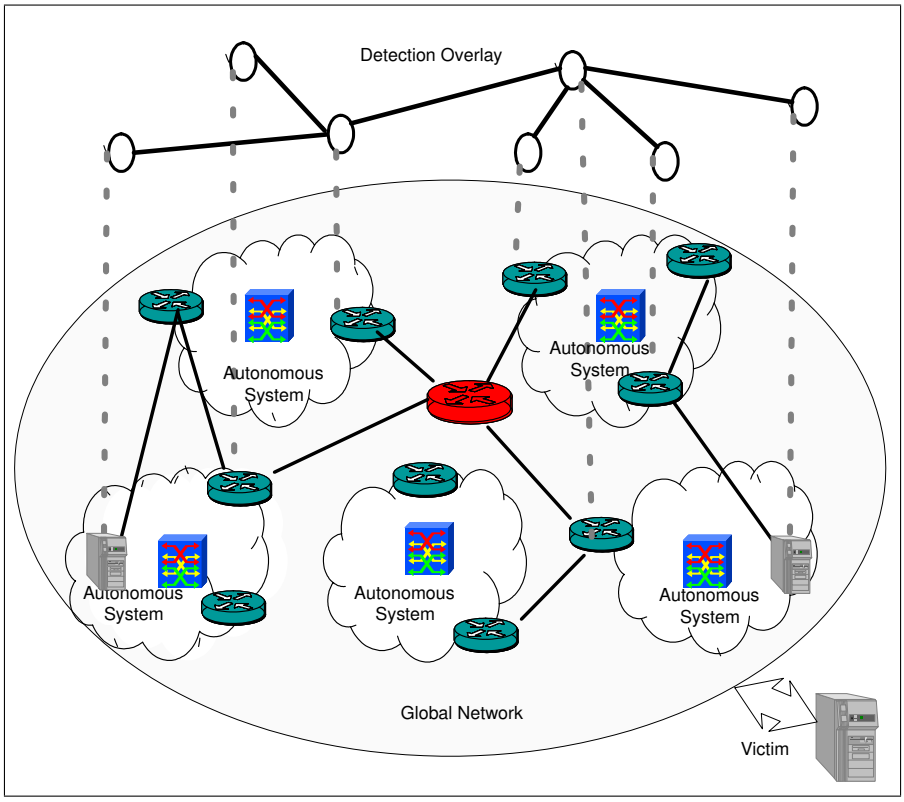

Figure 4.1: Detection overlay architecture

respond to an attack. This can improve the accuracy and speed of detection of network attacks. Here, we assume that each local detection system is trusted. The operations at the individual detection nodes are described below:

- The local detection node detects attacks using various mechanism. To detect DDoS attacks, it keeps traffic statistics for high-traffic destinations using sample-and-hold algorithms. If the traffic statistics deviate from the normal profile, the local node will raise an alarm to report attacks. To detect Internet worms, the local detection node can either use a database of worm signatures or monitor packets toward unused IP addresses to identify malicious packet flows.

- When each individual node detects a possible network attack, it will share this information with other nodes using a gossip mechanism. The information shared can be either the confidence that certain target machines are

under DDoS attack, or the signatures identified as Internet worm attacks. According to the discussion in Chapter 3, the presented gossip information sharing mechanism will converge exponentially. After this time period, each node will acquire sufficient information about the network attacks and will know that other nodes have the same information, and can make decisions about the attacks.

- When an individual node confirms the network attack, it will deploy countermeasures to prevent continuance of the attack, and communicate with other peers about the attack. The peers then perform similar actions in response. The process continues until the attack traffic is effectively blocked.

Our approach can be combined with available mitigating or rate limit technologies to eliminate the attack before it does significant damage.

4.2 Local Network Attack Detection

There are several techniques for local network attack detection, such as misuse detection, statistical anomaly detection, information retrieval, data mining and inductive learning. The internals of an individual local detection node can be fairly complex, but conceptually it can be structured into six components, as shown in the Figure 4.2. The traffic measurement module is responsible for measuring local traffic. Next, the local detection mechanism will use this data to detect any local anomaly. This local decision will be sent to the cooperative detection engine, which will combine this local decision with the decisions from neighboring nodes using the message dissemination module, to make a global detection decision. Finally, the detection decision module will inform the attack defense module to take action to defend against the attack.

When the local detection node detects a network attack, it will generate an

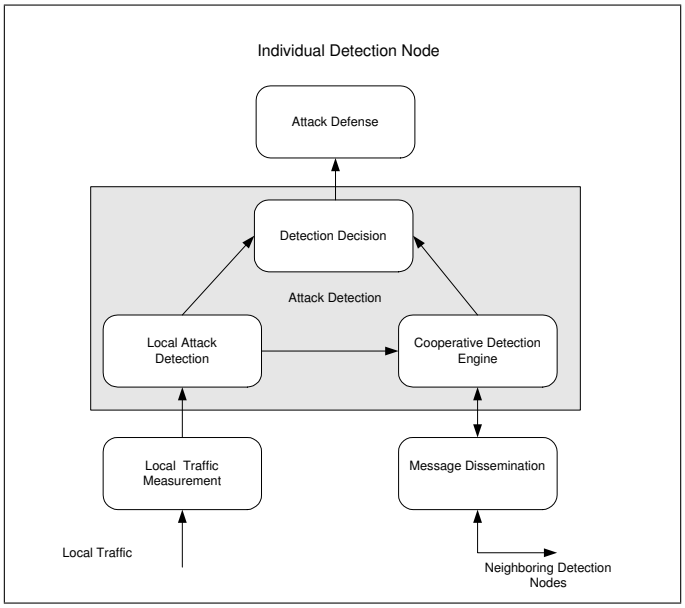

Figure 4.2: A conceptual architecture for individual detection node

alert message in the form of a tuple $(conf, dest)$, where the $conf$ is the confidence of the detection node about this alert message, and $dest$ is the target of the attack. This message will be aggregated using the decentralized attack information sharing mechanism.

4.3 Attack Information Sharing

A key requirement for network attack detection is low false positive rates, calculated as the percentage of normalcy variations detected as anomalies, and low positive rate, calculated as the percentage of anomalies detected as normalcies. In our approach, there are two factors that will affect system performance: the overhead of the information sharing mechanism, and the level of knowledge acquired about the network attack. Communication bandwidth is often a scarce resource during the network attack, so the attack information sharing should involve only a small number of messages. In particular, any protocol collecting all local data at a single node will create communication bottlenecks or a message implosion at that node. According to the discussion in Chapter 3, gossip based protocols are

resilient and scalable while providing sufficient information for attack detection. We use gossip based communication for the information sharing purpose. The structure of the gossip protocol running at each node n is shown in Figure 4.3.

```
when  ( node n builds a new  (conf, dest) pair)
{       while  ( node n believes that not enough of its
        neighbors have received  the (const, dest) pair)
        {
                m =a neighbor node of p;
                send  (conf, dest) pair to m;
        }
}
```

Figure 4.3: Gossip protocol for cooperation

Compared to multicast or broadcast protocols, the gossip protocol has a smaller overheads. However, it requires a longer time for each node to get the message. While reducing message dissemination overhead, we still want to maintain the speedy information delivery provided by multicast or broadcast. A possible variant is directional gossip [43]. Directional gossip is primarily aimed at reducing the communication overhead of traditional gossip protocols. Here we present a modified directional gossip strategy, which can efficiently defend against DDoS attacks. The application of this strategy to defend against Internet worms will be presented in Chapter 6 in detail. We assume that the individual node knows its immediate neighbors in the overlay network. Our gossiping protocol is as follows: An individual node sends the $(conf, dest)$ pair to the node on its path to the destination target node with probability 1. It forwards the $(conf, dest)$ pair to all other nodes at random with probability p.

At anytime t, each node i maintains a list of $(conf_k, dest_k)$ pairs. The algorithm is described below:

1. *Every node in the overlay network runs the aggregation protocol and makes a global decision in the period T_g.*

   (a) *Let $(conf_{r,k}, dest_{r,k})$ be all pairs sent to node i in round t, where $t \leq N$ (N is total number of rounds in the period T_g, k is the index of node and r is the gossip round).*

   (b) *Let $d_{t,i} = \frac{\Sigma_r conf_{r,k}}{m}$, where m is the number of messages received and $d_{t,i}$ is the aggregated information.*

2. *Query the routing table, find out the next hop to $dest_{t,i}$, send the pair $(d_{t,i}, dest_{t,i})$ to that node with probability 1. Send the pair to other neighbors with probability p.*

3. *At round N, Compare $d_{N,i}$ with $Threshold_i$. If $d_{N,i} > Threshold_i$, then $dest_{N,i}$ is under attack. Otherwise, set $d_{N,i}$ =new local detection confidence value. Begin new round of attack information aggregation.*

The algorithm is illustrated in the flowchart in Figure 4.4.

The algorithm described so far is based on the assumption that all nodes synchronize on aggregation. This assumption cannot be satisfied given the dynamic and heterogeneous nature of Internet. In [3], it has been discussed that even the nodes are not synchronized on aggregation, the algorithm still can work efficiently.

The advantage of the strategy is illustrated in Figure 4.5. Suppose node X and node Y suspect that the destination host A is under attack, and both of them use node Z to forward packets to destination A. Obviously, it is better to send the $(conf, dest)$ pair with a higher priority to Z than to other neighbors. The rationale behind this scheme is as follows. Sending the detection information with higher probability to critical nodes allows them to make decision early, allowing the network attacks to be mitigated earlier.

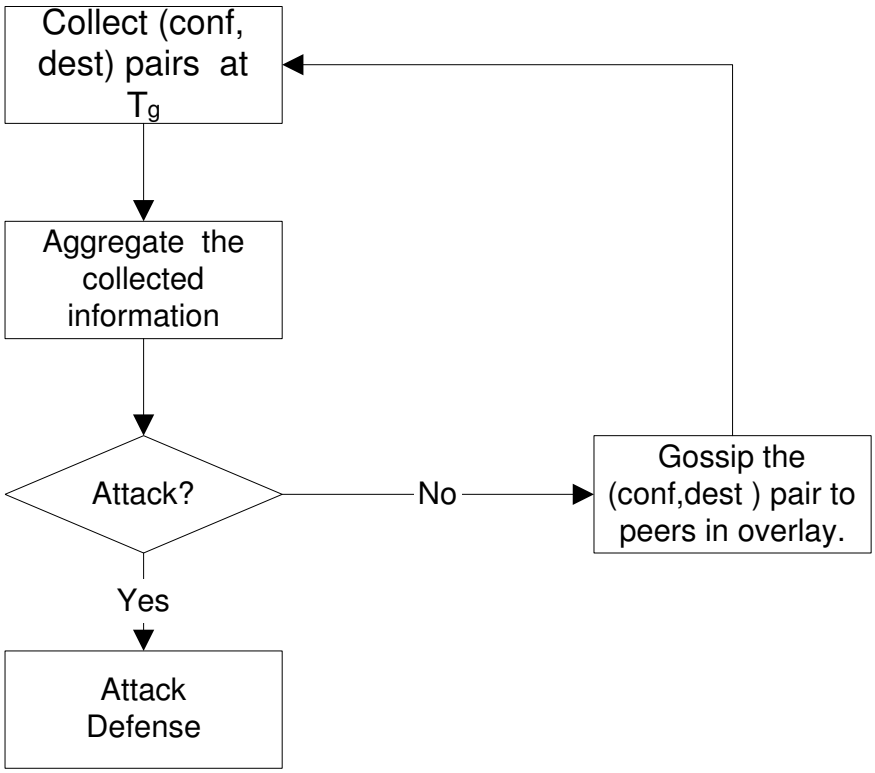

Figure 4.4: The flow chart for the gossip based coordination algorithm

For each destination (we monitor detections of sampled big flow only) with $conf > 0$, each individual node in the overlay network sends the $(conf, dest)$ pair to its neighbors. On receiving such a message, the neighbors discard duplicates, compute the aggregate ($Aggr$) of the $conf$ values received per destination, and forward non-duplicate values to their neighbors. If, for any destination, $Aggr$ exceeds a pre-defined threshold, the individual node concludes that the destination is under attack. This cooperation stage helps reduce errors in the identification of attacks.

4.4 Analysis of Cooperative Defense Framework

In this section, we discuss both the advantage of this framework and related issues.

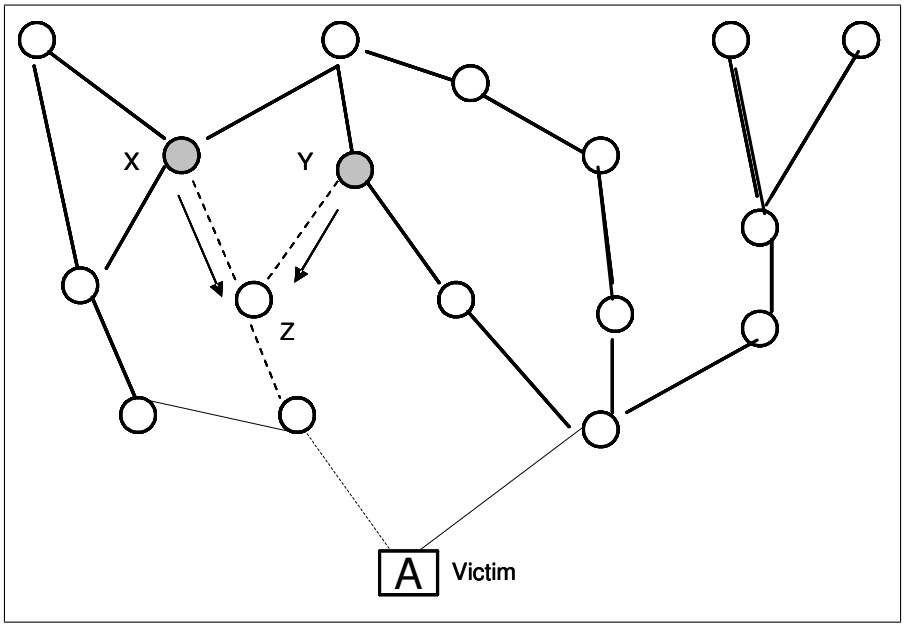

Figure 4.5: Gossip strategy illustration

4.4.1 Advantage of Cooperative and Information Sharing

In addition to improving the accuracy of countermeasure against network attacks, cooperation among overlay nodes have the following advantages:

- Accurate Detection. As discussed in Chapter 3, each detection node can only partially observe network attacks. As a result, detection based on this information has a high false positive rate, and legitimate traffic will be affected. The presented cooperative framework can improve a detection node's knowledge about the attacks and thus enable accurate attack detection.

- Quick Response. Any detection node that observes network attacks can immediately inform other nodes in the framework, which can then take appropriate actions.

- Optimization. Reducing the bandwidth consumption due to the attack traffic is our basic approach to network flooding attacks. Different attack traffic may have different targets with different paths and accordingly, have different potential bandwidth consumption. Generally speaking, the attack

traffic with a longer path will consume more bandwidth and cause more damage than the traffic with a shorter path. Hence, it is beneficial for the overlay nodes to process more packets with longer remaining paths to their destination, as compared to ones with shorter remaining.

- Distributed Load. A detection node can either rate limit or filter the attack traffic based on aggregated information. With this approach, the load of defending against attacks is distributed among the nodes of the cooperative defense framework.

4.4.2 Defense Infrastructure Overhead

Defenses mitigate the impact of the attack traffic on the network but may impose an additional overhead on the networks that implements them. The additional overhead includes computational overhead imposed by attack detection and attack response enforcement, storage requirement to save logs for attack detection, and communications overhead used to send control messages to distributed locations in a network. These overheads are described below.

First, attack responses may impose a computational overhead on network devices. Once filtering rules are enforced to examine network packets, a per-packet delay will be incurred for executing the filtering rules. Minimizing the per-packet delay is a packet classification problem in router performance optimization. Although most commercial routers are optimized for routing, the per packet delay of matching filtering rules depends on the number of filtering rules, the number of characteristics used to identify attacks, and the update frequencies of the filtering rules.

Second, attack detection algorithms impose a storage requirement for saving

network information to determine attack characteristics. This storage requirement is usually very large for monitoring high speed network links. Current technology can scale up to 10Gbps link speed without losing much information on IP packets. To reduce the storage requirement and to catch network packets from high throughput routers, sampling and processing of packet data dynamically will be needed.

Third, gossip messages to coordinate attack detection among the proposed detection framework are an additional overhead to network transmission. If communication occurs between network routers, it is important to know if such communication will result in abnormal behavior at the routers. Since most commercial routers are optimized for routing, it is not certain if additional communications among routers will impose additional delay at routers or not. Future work will explore how communication overhead impacts system performance. For example, a DDoS flood could overwhelm systems and limit the use of in-band control protocols to detect and respond to the trouble [54]. This is a limitation of distributed cooperative detection technology, and lends credence to more local intelligence for throttling attacks. However, given that distributed cooperative detection is used, gossip based communication mechanisms provides reduced overhead.

To accurately measure the cost of cooperative mechanism as have discussed so far, we need a test bed in the Internet scale. Currently, researchers have proposed to build such a test bed in the expense of tens of millions of dollars [35]. We will leave this as future work when a test bed is available.

4.4.3 Miscellaneous Infrastructure Issues

Trust Trust is an important issue in such a system, more so in the absence of a centralized trusted authority to provide digital certificates. The usual decentralized alternate to central CA is the web-of-trust model, where certifying happens among peers rather than from a central authority. We believe, the overlay nodes

can build trust relationships based on this model. Ideally, every overlay nodes should digitally sign their messages sent to other nodes in a manner that allows other nodes to validate the authenticity of the sending nodes. Current available technologies should suffice for this purpose.

Multiple wrong decisions There is a possibility that multiple nodes of the cooperative defense overlay will make wrong decisions at same time. As a result, the cooperative defense will drop legitimate packets. However, given the state of art available local detection techniques, the false rate p_i of each detection node will be a very small value [75]. The probability q that multi nodes make wrong decisions at same time can be approximated as $p_1 * p_2 * .....p_n$, which is a very small value.

Attack against the infrastructure Another issue that must be addressed is how to protect the communications of the detection nodes when the links are completely saturated during a DDoS attack or Internet worm propagation. In the event of standard packet flood attacks, it is certainly possible that some set of nodes could be effectively removed from the infrastructure. Yet, if any connectivity remains at all, the gossip exchange of data will eventually prevail, and data stored within the infrastructure will reach all sites in the system. Also, the distributed and coordinated nature of the infrastructure makes it robust to the removal of nodes through failures or attacks. Thus, the infrastructure is relatively tolerant to attacks. In the case that a compromised overlay node sends large amounts of data to flood other overlay nodes, the overlay node can apply filters to incoming data such that data sent by any nodes or set of nodes can not exceed a specified threshold.

4.4.4 Limitations of the Approach

The approach of detecting DDoS attacks and Internet worms in a distributed manner based on traffic anomalies has its own limitations. On one hand, there

are a set of theoretical issues related to the detection algorithms, such as the choices of local and global thresholds, traffic modeling, and admitting multilevel local detection results. On the other hand, since the large scale distributed cooperative mechanism induces a certain amount of delay to reach a global detection decision, this defense infrastructure is not very useful for DDoS attacks of very short durations. For example, as discussed in a recent study [53], the infrastructure should target to handle DDoS attacks longer than 5 minutes, which is around 75 percent of all the attacks measured.

4.5 Summary

In this chapter we presented the framework of the decentralized information sharing framework. We first described the architecture of the framework and the internal components of each overlay node. Then we introduced the procedure of using this framework to detect and defend against attacks. The gossip based information aggregation protocol used by this procedure to acquire quasi-global view of network attack behavior was discussed in detail. Finally, we analyzed the advantages and limitations of this framework.

Chapter 5
Cooperative DDoS Defense

During a distributed denial of service (DDoS) attack, traffic transmits across the Internet towards the victim and the victim can easily detect the attack by observing its degraded service. However, it is too late to defend against DDoS attacks near the victim as the victim resources would be heavily loaded and would not be able to react to the DDoS attacks. The attacks should ideally be stopped as close to the sources as possible, saving network resources and reducing congestion. However, there are no common characteristics of DDoS streams that can be used to detect and filter them near the source. Our strategy is to defend the DDoS attacks in the intermediate network. We make the assumption that in the intermediate network, the aggregated attack flows toward the victim consume more bandwidth than aggregated normal flows to the victim. As the aggregate does not cause congestion in the network, and it is hard to detect the DDoS attacks in a single domain, we propose that by sharing information across domains distributed in the network, we can detect the DDoS attacks early. Based on the framework discussed in Chapter 4, we present a DDoS defense mechanism in this chapter.

5.1 Cooperative DDoS Defense System Modules

We assume that the Internet is a set of Autonomous Systems (AS) as discussed in Chapter 4. Individual detection nodes are located at the egress routers of the Autonomous System, and collect meaningful information and detect DDoS

attacks locally. These detection nodes form an overlay and the overlay is used to share the detection information using the gossip protocol. The functions of each individual DDoS detection node is discussed below.

5.1.1 Traffic Measurement Module

The traffic measurement module monitors all traffic passing through the detection nodes. Each packet is classified as incoming or outgoing based on its arriving interface. Information in the packet header is then used to update statistics on current flows. Periodically, statistics are compared with a model of normal traffic, and flows are characterized as being normal, transient or attack flows. Normal flows are those flows whose parameters match those of the model and that have not been recently classified as attack flows. Attack flows are those flows whose parameters are outside of the model boundaries. Transient flows are those whose parameters match those of the model but that have been recently classified as attack flows.

During normal operation, the traffic measurement module keeps packet rate statistics for different flows grouped by address prefix. It constructs an address prefix tree data structure, which allows for quick aggregation of flow information. To keep the tree from growing in an unbounded manner, periodic garbage collection is performed when the tree attempts to grow beyond a certain size. The traffic measurement module supports hundreds of simultaneous packet flows by dynamically building an aggregation tree based on flow information. The architecture of the traffic measurement module is shown in Figure 5.1.

A disproportional increase in the relative frequency of a particular packet attribute value is an indication that the attacking packets also share the same value for that particular attribute. The greater the disproportional increase, the stronger the indication. The more “abnormal” attribute values a packet possesses,

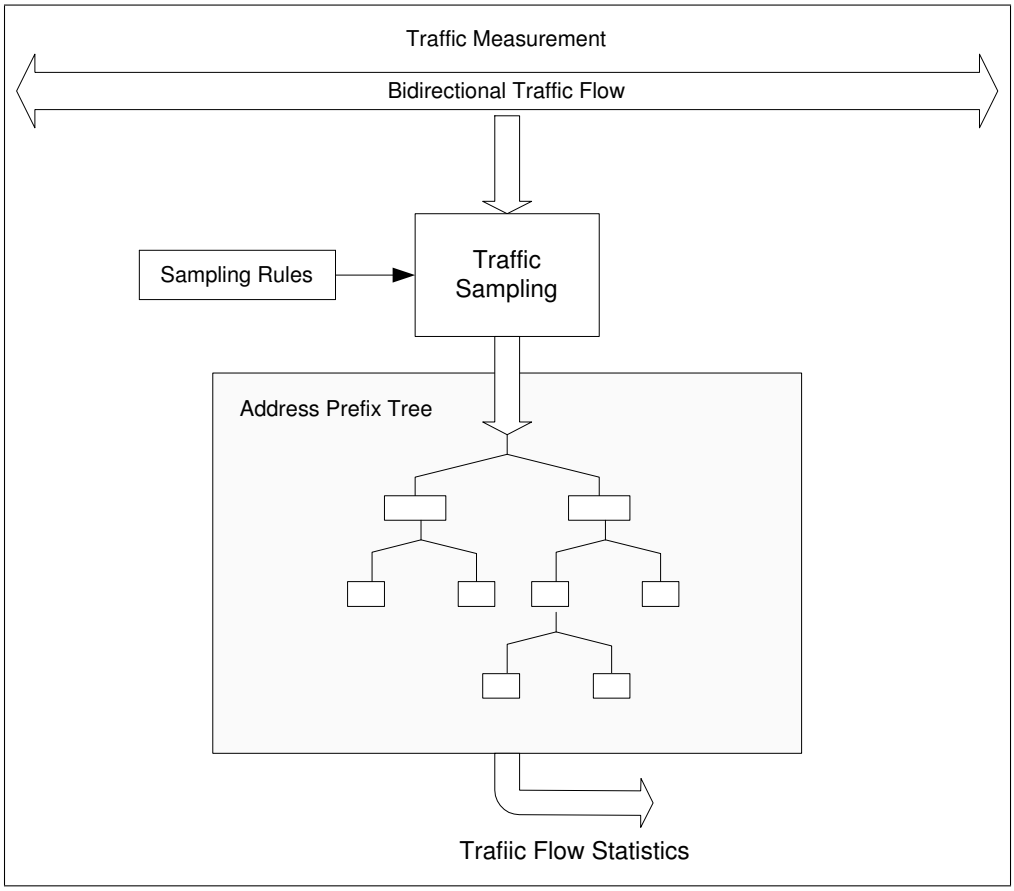

Figure 5.1: Traffic measurement module

the higher the probability that the packet is an attack packet. For example, if it is found that the suspicious packet flows contain an abnormally high percentage of (1) UDP packets, (2) packets of size S, (3) packets with TTL value T, then UDP packets of size S and TTL value T destined to the DDoS victim should be treated as prime suspects and given lower priority during selective packet filtering when there is an overload.

Candidate packet attributes considered for traffic profiling include: the marginal distributions of the fraction of recently arrived packets having various (1) IP protocol-type values, (2) packet size, (3) server port numbers, (4) source/destination IP prefixes, (5) Time-to-Live (TTL) values, (6) IP/TCP header length and (7) TCP flag patterns. We are also interested in the fraction of packets which (8) use

IP fragmentation and (9) bear incorrect IP/TCP/UDP checksums. It is worthwhile to consider the joint distribution of the fraction of packets having various combinations of (10) packet-size and protocol type, (11) server port number and protocol-type, as well as (12) source IP prefix.

5.1.2 Traffic Models

Internet traffic models have been developed for attack detection in several projects. The discussions below are based on the work in [36, 47, 46].

TCP normal traffic model. There are two special characteristics in TCP semantics. One is that a TCP flow experiences a three-stage hand shake during flow establishment. An unresponsive attack flow with a spoofed source, although marked as a TCP flow, cannot establish a real TCP flow. The reason is that its source is unlikely to get the SYN-ACK packet from the receiver, which is sent to the spoofed source rather than the real source. Unfortunately, it will be very difficult for the detection node to monitor the three stage connection establishment for individual flows. The other special point in TCP semantics is that during a TCP session, the data flow from the source to destination host is controlled by the constant flow of acknowledgement in the reverse direction. Our TCP flow model defines TCP_{rto} - the maximum allowed ratio of the number of packets sent and received in the aggregate TCP flow. The flow is classified as an attack flow if the packet ratio is above the threshold; otherwise, it is considered a compliant flow.

ICMP normal traffic model. The ICMP protocol specifies many different message types. During normal operation the "timestamp," "information request," and "echo," messages should be paired with corresponding replies. Using this observation, the normal ICMP flow model defines $ICMP_{ratio}$ - the maximum allowed ratio of the number of echo, time stamp, and information request and reply packets sent and received in the aggregate flow to the peer. The frequency

of other ICMP messages, such as "destination unreachable," "source quench," "redirect," etc., is expected to be small and a predefined rate limit can be used to control this portion of the traffic.

UDP normal traffic model. The UDP protocol is used for unreliable message delivery and in general does not require any reverse packets for its proper operation. Many applications that communicate through UDP packets generate a relatively constant packet rate, but the maximum rate depends heavily on the application. On the other hand, UDP traffic usually occupies a small percentage of overall network traffic and is conducted via a few connections. We use this observation to define the UDP flow model as a set of thresholds: n_{conn} - an upper bound on the number of allowed connections per destination, p_{conn}- a low bound on the number of allowed packets per connection, and UDP_{rate}- a maximum allowed sending rate per connection. The model classifies a flow as an attack when at least one of these thresholds has been breached. The first two thresholds help identify a UDP attack through spoofed connections, while the third identifies a UDP attack through a few very aggressive, non-spoofed connections. An attacker can still get enough traffic past the thresholds to perpetrate an attack if she/he chooses to spoof a small number of addresses consistently and distributes the attack sufficiently so that each source network sees only a small portion of the traffic.

5.1.3 Attack Detection Module

The objective of the detection module is to detect the onset of an attack and identify the victim by monitoring traffic statistics. Every detection node maintains a local and global view of intrusion and attack activity. The local view considers activity in the node's own network. The detection nodes periodically receive summaries from their peers, which are then used to create a global view. Each detection nodes can employ its own strategy for data aggregation to create local

and global views. Figure 5.2 shows the state diagram for the attack detection module.

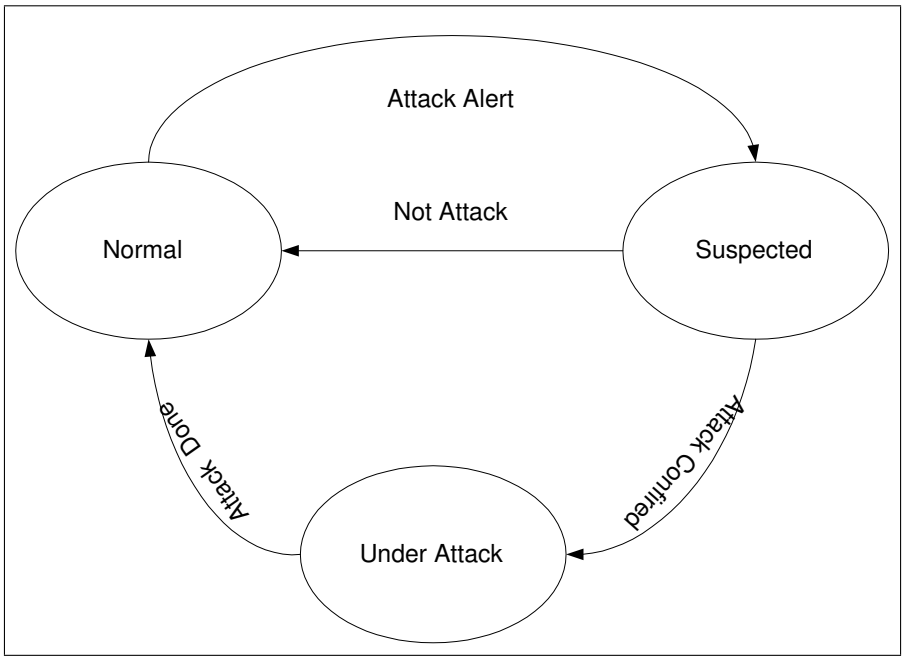

Figure 5.2: A state diagram for the attack detection module

To enable local abnormal behavior detection, we need to define the data that routers will collect using a statistical measurement method. As the high-traffic destinations are most likely to be under attack, it is reasonable to keep traffic statistics only for those high traffic flows that have the same destination IP addresses. We can use a *sample-and-hold* [27, 5] algorithm to let the egress routers keep track of destinations whose traffic occupies greater than a fraction r of the capacity C of the outgoing link. We call these destinations popular and destinations not in this list as unpopular.

Traffic profiles at each router are essentially a set of metrics $\{M_i\}$ for the traffic to popular destinations. An effective choice of such metrics is key to characterizing traffic streams. However, computing arbitrary fingerprints might require excessive memory and computation. Several metrics have been proposed by the research community. Some of them are:

- The fraction of new source IP addresses.
- The ratio of traffic between the two directions.
- An approximation of the flow-length distribution of traffic to the destination.

Based on these metrics, there are different abnormal behavior based detection approaches, such as those described in [36, 32]. In this thesis, we use CUSUM to detect abnormal behavior [75]. Let X_n represent one of these metrics during time interval Δ_n. The main idea is that, during an attack, for the random sequence X_n, there is a step change in the mean value $E(X_n)$. The non-parametric CUSUM is asymptotically optimal for such Change Point Detection problems. This general approach is based on the model presented in Wang et al. [75] for attack detection using CUSUM. One of the assumption for the nonparametric CUSUM algorithm is that the mean value of the random sequence is negative during normal conditions, and becomes positive when a change occurs. In general, $E(X_n) = c \ll 1$. We choose a parameter γ that is the upper bound of c, i.e., $\gamma > c$. Thus without loss of any statistical feature, X_n is transformed into another random sequence Y_n with negative mean b during normal operation, i.e., $Y_n = X_n - \gamma$. When an attack happens, Y_n will suddenly become large and positive. Suppose, during an attack, the increase in the mean of $E(Y_n)$ can be lower bounded by h. Our change detection is based on the observation that $h \gg c$.

We use the recursive version of the non-parametric CUSUM algorithm [75], which is as follows:

$$z_n = (z_{n-1} + Y_n)^+,$$
$$z_0 = 0, \tag{5.1}$$

where $(z_{n-1} + Y_n)^+$ is equal to $(z_{n-1} + Y_n)$ if $(z_{n-1} + Y_n) > 0$ and 0 otherwise. z_n represents the continuous increment of Y_n. A large z_n is a strong indication of

an attack.

Let $d_N(.)$ be the decision at time n: '0' for normal operation and '1' for attack. The decision function can be described as follows:

$$d_N(z_n) = \begin{cases} 0 & \text{if } z_n \leq N; \\ 1 & \text{if } z_n \geq N. \end{cases}$$

Here N represent the threshold for local attack detection. Let *conf* denote the confidence with which the individual detection node suspects an attack. We set $conf = \sum_i \delta(M_i) * d_N(M_i)$. δ assigns "weights" to a metric, depending on the extent to which the metric contributes to errors (false positive or negatives): $\delta(M_i) \propto \frac{1}{err(M_i)}$ where $err(M_i)$ is the sum of the false positive and negative rates for M_i. The appropriate δ can be configured from measurements.

When a local detection node detects an attack, it will sends the $(conf, dest)$ pair to its neighbor nodes in the overlay network infrastructure for correlation purpose.

Each overlay node independently consolidates and analyzes its local detection results with attack alerts received from other overlay nodes to make a global decision. The most straight forward way to merge information from multiple site is through a simple addition or average across the whole domain. While this approach provides a simple means for organizing and summarizing data, it also has the risk of inaccuracy. We currently performs aggregation through computing average of attack information, and will design advanced mechanisms with better performance in the future research.

5.1.4 Message Dissemination Module

This module exchanges attack information with other overlay nodes using the gossip mechanism. In terms of the communication protocols and intrusion detection language specification, possible candidates are the Intrusion Detection

Exchange Protocol and Intrusion Detection Message Exchange Format. To foster inter-operability and maximize extensibility, we represent messages using XML.

There are three kind of messages: alert, heartbeat, and cancel. The alert message is shown in Table 5.1.

Table 5.1: Detection Information Alert

Alert Identifier
Destination address prefix
Confidence Level
Timestamp
Attack traffic signature

Each alert message has a *Alert Identifier*, which is used to let other overlay nodes to know the sender of this alert. The *Destination address prefix* represents the targets that are under DDoS attack. The *Confidence Level* denote the confidence with which the detection nodes suspect an attack. The *Timestamp* is used to denote the generation time of this alert, so other nodes can discard a message when it has expired. Finally, the *Attack traffic signature* is a list of attributes of the attack traffic.

In the system, the overlay detection nodes periodically exchange heartbeat message with other nodes. Based on these heartbeat message, the overlay node will adapt their gossip strategy to improve the message dissemination performance. One example of such strategy is changing the gossip interval according to the network bandwidth usage.

The cancel message is used to inform other overlay detection nodes that the Alert message has expired. Each detection node will evaluate a current attack status based on this message.

5.1.5 Attack Defense Module

Attack detection itself is not the final goal of the defense system. Once a DDoS attack is detected, the next step is to distinguish the attacking packets from the legitimate ones amongst the suspicious traffic. Our approach is to perform online profiling of the suspicious traffic and compare the findings with the nominal traffic profile of the victim. The viability of this approach is based on the premise that there are some traffic characteristics that are inherently stable during normal network operation of a target network, in the absence of DDoS attacks. The objective is to maximize friendly traffic throughput while reducing attack traffic as much as possible [38].

The architecture of control module is shown in Figure 5.3.

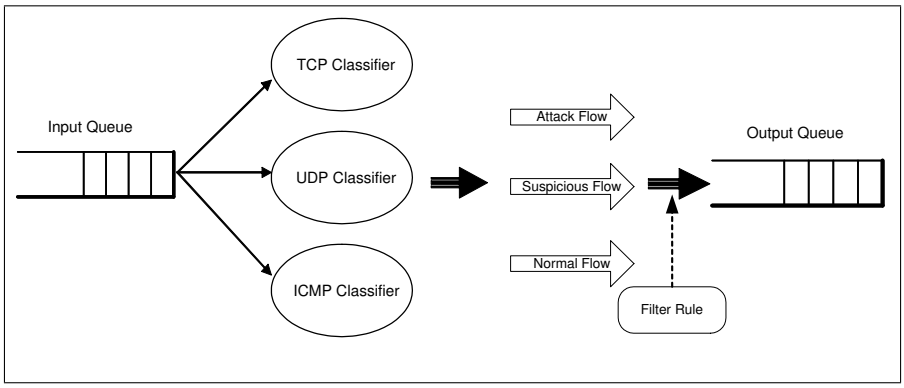

Figure 5.3: Traffic control module architecture

According to the confidence of the attack signature, the traffic with the identified attack signature(A_i) will be rate-limited according to the formula below:

$$rate_{out}(A_i) = rate_{in}(A_i) * \lambda(conf_i)$$

Where $\lambda(conf_i) < 1$ is a factor defined by the confidence level of the attack signature identified.

5.2 Simulation Results: DDoS Case Study

To further examine system performance, under detailed network models, we conduct experiments using the Emulab testbed. The objective of the emulation is to illustrate that our approach can effectively defend against DDoS attack with high accuracy and reasonable overheads.

5.2.1 Performance Metrics

In our model, we assume that it is not easy to distinguish DDoS attack traffic from legitimate traffic. Therefore, our rate limiting mechanism will block legitimate traffic as well. To evaluate performance of the proposed defense mechanism, we adopt the following measurement:

- Measure the legitimate traffic drop rate and the malicious traffic drop rate under different patterns of DDoS attacks. Since the algorithm dynamically adjusts rate limiting to the suspicious traffic based on the aggregated information, a legitimate user should be adequately served.

- Measure the affect of deployment ratio (the ratio of overlay defense nodes to all the network routers) of defense nodes on the effectiveness of the cooperative defense framework. One advantage of distributed cooperative defense mechanism lies in the fact that with partial converge or deployment a synergistic defense effect can be achieved. Since not every router or gateway in the Internet will be a defense node, the cooperative defense mechanism is designed to be effective in partial deployment. This feature is supported by an overlay network topology in which only nodes that have established direct peering relationship are aware of each other. The system provides a significant level of defense for potential targets with only a few defense nodes deployed, and becomes more effective as more defense nodes are added, protecting a larger community.

- Measure the reliability and scalability of gossip mechanism. We analyze message dissemination convergence rates and the overhead introduced by this distributed cooperation mechanism.

5.2.2 Results

We implemented our distributed cooperative defense mechanism in a Linux router and tested it with live traffic in the Emulab testbed. As mentioned earlier, we rely on existing intrusion detection systems to detect attacks at each individual detection node. In our experiment, we use Snort [60] for this purpose. We also implemented the gossip based communication mechanism within the Linux router, which dynamically adjusts the rate limiting parameters according to the information aggregated from detection nodes in the peer to peer defense overlay network.

We use a simple HTTP client-server as the simulated application. Further we use the GT-ITM topology generator to generate the Internet topology. GT-ITM can generate a random transit-stub graph based on input parameters. This graph closely resembles the Internet topology. Figure 5.4 shows the experimental topology with 100 nodes. The attack is simulated using a given number of compromised nodes in different sub networks. Detection agents are deployed at selected nodes and execute the algorithm described in Section 4. The communication agents use gossip to share information. In these experiments, there are 10 attackers, each of them sends out 1.3Mbps UDP traffic to the victim. The legitimate user makes request with traffic rates chosen randomly and uniformly from the range [2Kbps, 6Kbps]. If a request successfully arrives at the server, the server will return the requested document after a random processing time, chosen according to collected empirical distributions [44].

In the first set of experiments, we performed test runs for normal use, under attack without response, and under attack with distributed cooperative response.

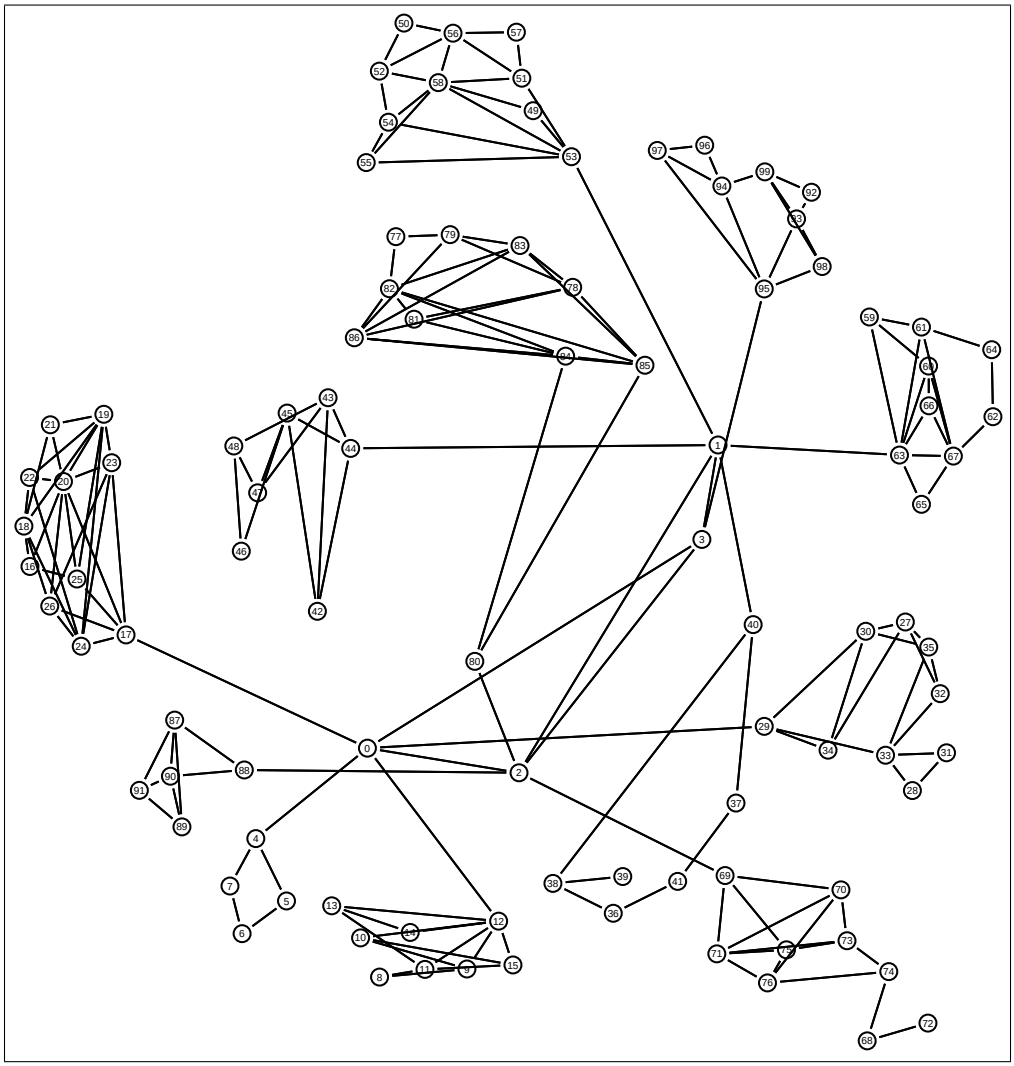

Figure 5.4: Simulated network topology

In each case, we measured the packets rate of a selected client at the HTTP server. Figure 5.5 shows the result from the experiment. The X axis represents time intervals in seconds; the Y axis represents the number of packets received at the server. The attack starts 50 seconds after the start of legitimate traffic and last for 500 seconds. Compared with the packet rate of normal run, the selected legitimate client's packet rate at the server drop dramatically under attack without response. For the experiment with cooperative defense mechanism enabled, we can notice a gradual increase in the legitimate packet rate. The ramp-up behavior is due to the false detection at local defense node. As a result, some legitimate traffic will be

dropped by the rate limiting mechanism as well. As the algorithm converges, each defense node get more precise information about the global attack information and thus can more accurately rate-limit attack traffic.

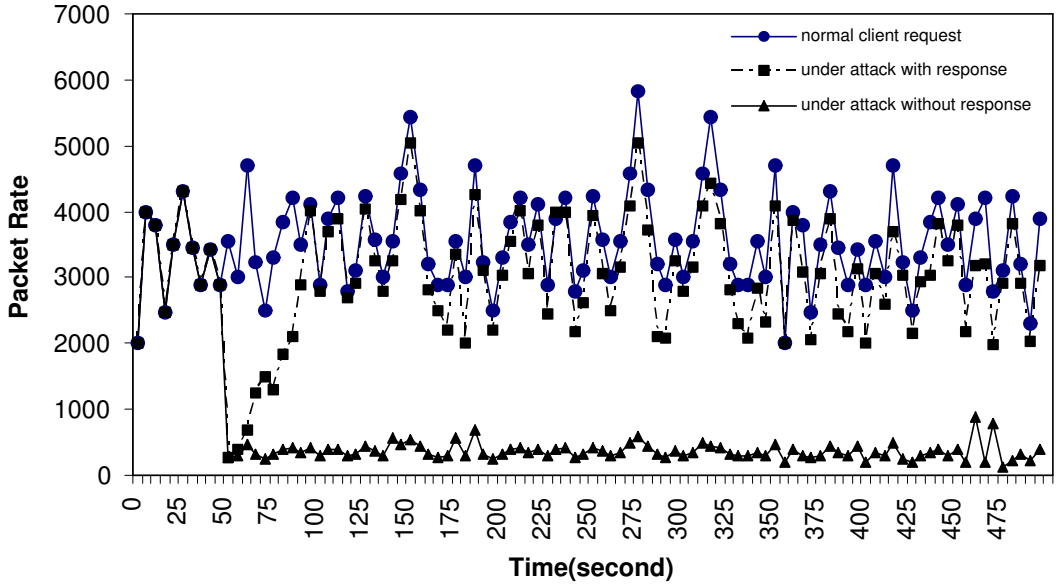

Figure 5.5: User packet rates for legitimate traffic under different test conditions

In the second set of experiments, we investigate the benefit of an increased deployment of defense nodes. Figure 5.6 plots the average number of false alarm rates, which decrease gradually as more nodes join the peer to peer defense overlay network. By adding sufficient nodes to the defense overlay, attack traffic is dropped efficiently and the amount of the attack packets that reach the victim server decreases. The decrease of legitimate packet drop rate stabilizes when the deployment ratio is great then 20% in this experiment. This is because, as we add more routers as local defense nodes into the cooperative defense network, more of them will be on the path from the attack traffic source to the victim destination. As a result, sharing attack information among them will not increase the overall knowledge about the network attack very much.

In the third set of experiments, we vary the parameters of the gossip mechanism to investigate the relationship between the overhead of information sharing

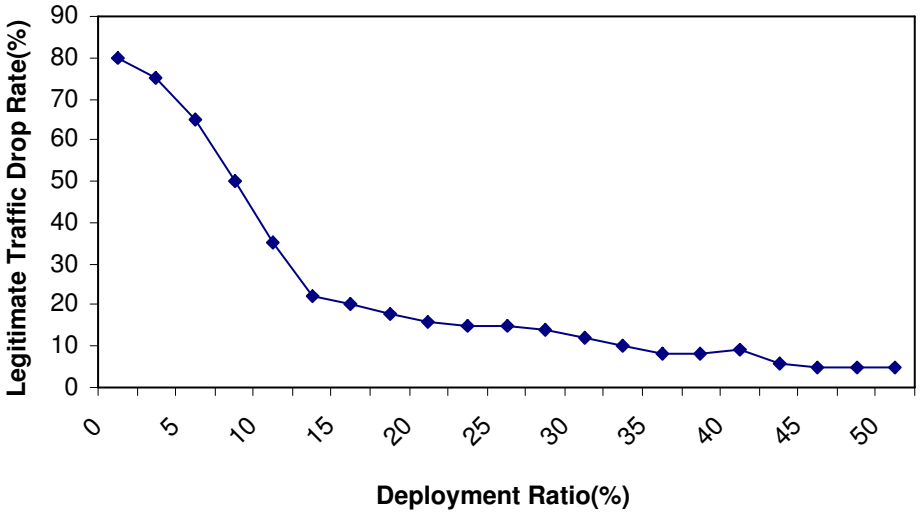

Figure 5.6: Reduction in false detection rates with increased deployment

and defense efficiency. Let p represent the probability that each detection node in the detection overlay network sends the local attack information to its neighbor nodes. We vary the gossip probability p between 0.2, 0.4, 0.6, 0.8, 1.0. The performance of the approach for different gossip probabilities p used are shown in Table 5.2. The *false positive rate* measures the percentage of legitimate packets dropped by the rate limiting mechanism, and the *false negative rate* measures the percentage of attack traffic that passes the defense node.

Table 5.2: Cooperative defense performance for different directional gossip probabilities

Gossip Prob.	False Positive	False Negative
0.2	12.12%	5.2%
0.4	10.03%	4.13%
0.6	8.32%	4.32%
0.8	8.15%	3.56%
1.0	7.67%	3.12%

As we can see from the simulation results, our algorithm can detect and defend against DDoS attacks with high accuracy. With $p = 0.4$, we have low false positive and low false negative packet drop rate respectively. The false positive rate is

relatively higher than the false negative rate. This is because we adopt a high initial drop rate when the local defense node detects an attack and as a result legitimate packets will be dropped in the case of a false detection.

Defenses mitigate the impact of the attack traffic at the victim network but may impose an additional overhead on the networks that implements them. We measure the overhead introduced by distributed cooperative information sharing in this experiment as well. Figure 5.7 shows the per-node overhead with different number of nodes in the system. The number of bytes per second processed by each node for cooperative defense purposes do not increase significantly as we add more nodes to the defense overlay. As a result, the gossip based information sharing mechanism is scalable and can be used in larger networks. When the gossip probability increases, the overhead will increases as well. This parameter can be tuned to adapt to different applications to achieve the best performance.

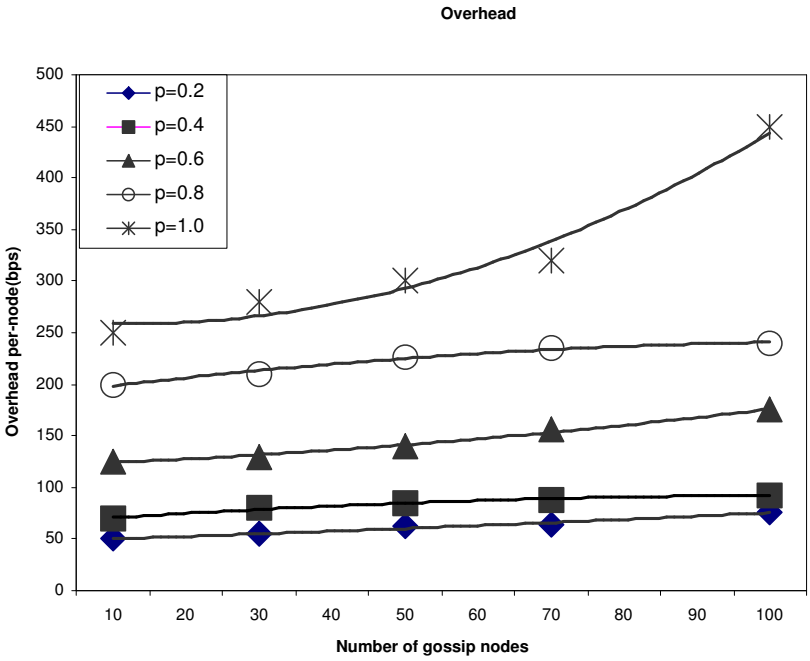

Figure 5.7: Overhead of information sharing using gossip

5.3 Summary

In this chapter we presented the use of the decentralized information sharing framework to defend against DDoS attacks. The specific DDoS detection and defense modules within the framework were described in detail. The traffic measurement module maintains track of statistics about the traffic passing through the detection nodes. The traffic detection module then identifies the attack traffics using this information. The message dissemination module shares this information using the algorithm discussed in Chapter 4. Finally, the attack defense module either filters or rate limits identified attack traffics using the aggregated information. We implemented and evaluated the cooperative defense using emulator. The evaluation demonstrates that it is effective against DDoS attacks.

Chapter 6
Cooperative Internet Worm Containment

Internet worms are another critical threat to the Internet. They generally use various scanning methods to probe the vulnerabilities of Internet services and propagate themselves rapidly. Unfortunately, many existing defense mechanisms, which employ local detection to detect infections, hold out little hope in containing novel and fast worms. These mechanisms will have high false alarm rates. A false alarm, whether malicious or unintended, can trigger an effective denial of service attack by the response mechanism itself. In this Chapter, we present a decentralized solution in which firewalls at various access networks exchange information amongst themselves to defend against worm attacks using the framework discussed in Chapter 4. In this chapter, we first analyze the mathematical models of worm propagation. We then discuss a worm defense framework that monitors malicious worm activities and defends against them. Finally, we evaluate the framework using simulation.

6.1 Worm Model Analysis

Distributed response mechanisms require some degree of trust between the automated agents cooperating in the response. In the best of times there are at most an insignificant minority of nodes in the Internet that any given node expects to be trustworthy; during a virus attack it is unreasonable to trust any particular node. Thus any proposed defense mechanism must be robust in the face of inaccurate information from some of its peers. These observations expose several

significant problems that must be dealt with. Any node that responds to a potential virus carries a cost: a node has finite resources and therefore can only actively engage a limited number of viruses at a time. Further, filtering packets containing potential threats carries the possibility of false positives: legitimate traffic may be blocked. In the absence of cost, the best response to a potential virus attack is to flood the network as rapidly as possible, causing as many cooperative agents to respond at once.

6.2 Mathematical Models for Virus/Worm Propagation

Computer worms are similar to biological viruses in their self-replicating and propagation behaviors. Thus the mathematical techniques developed for the study of biological infectious diseases can be adapted to the study of computer viruses and worm propagation. For simple worm mitigation strategies, mathematical models with closed form solutions are possible. Considerable work has been done on models for worm propagation in the absence of any coherent mitigation measures; some of these will be described below. Both of these models will be used as theoretical foundation for our simulation.

1. **Stanifords Virus Propagation Model** S. Staniford, et al. in their paper [68] analyze the Code Red worm by developing a quantitative model of its propagation. Their model is as follows:

   - N : Initial total # of vulnerable hosts
   - a : proportion of infected hosts
   - K : # of hosts that each infected host can find and compromise

   Since the infection ability of a worm is proportional to the density of the target hosts, the successful infection in time dt is $K(1-a)dt$. There are Na

worms in total, thus, the rate at which infected hosts increase during the time period dt is

$$Nda = (Na)K(1-a)dt.$$

Dividing by N,

$$\frac{da}{dt} = aK(1-a)$$

resulting in:

$$a = \frac{e^{K(t-T)}}{1+e^{K(t-T)}}$$

where T is a constant that fixes the time position of the incident (the time that worm starts propagation). This equation produces the S-curve growth behavior typically seen in population growth models with limited environmental carrying capacity.

2. **Kephart's virus infection model [40]**

   J. Kephart and S. White created another mathematical model by representing an individual system as a node in a graph. Directed edges from a given node j to other nodes represent the set of individuals that can be infected by j. We briefly introduce this model here, and refer to the original paper [40] for details.

   - i : The proportion of infected hosts
   - $\overline{b}$: The expected number of nodes neighboring a node
   - β : The infection rate of the virus
   - δ : The cure rate of each node

   J. Kephart et. al. derived a deterministic differential equation describing the time evolution of i(t):

$$\frac{di}{dt} = \beta\overline{b}i(1-i) - \delta$$

Note that if the second term, which describes the cure rate of a host, is taken out, this becomes the same as Staniford's model.

These functions have the characteristics that, for small values of time t, the incidence grows exponentially until a majority of individuals are infected. At this point the incidence slows exponentially, reaching zero as all individuals are infected.

6.3 Cooperative worm Defense Approach

In Chapter 4, we discussed that the gossip based information sharing framework can be used to efficiently defend against network attacks. Based on this framework, in this Section, we present a dynamic infrastructure for worm detecting and defense, which composed of a diverse collection of nodes located at ingress or egress routers of local network. The objective of this framework is to share information to improve worm defense capability for all participants. When the Internet is under a worm attack, the network links will be congested. To quickly and scalably share information in such a situation before the worm can compromise a significant amount of hosts, the system must overcome the challenges discussed in Section 2.4.

The discussion in Chapter 3 has analytically shown that our decentralized cooperative information sharing framework can collect quasi-global knowledge about the network attacks within acceptable delay. Further the gossip based information sharing mechanism is resilient and scalable. As a result, the framework can be used to efficiently defend against network attacks. The internal architecture of individual detection node is described in detail in Section 4.2. We have given a detailed discussion of the functionality of each module within the detection node in Chapter 5. For Internet worm detection, the modules within each detection node have similar functionality except that they need to be customized to address

requirements of Internet worms. In the following sections, we will describe how this framework can be used to defend against Internet worms.

6.3.1 Architecture

Recent seminal work by Autograph [41] suggests that it is promising to automatically generate worm signatures by analyzing the prevalence of payload contents and their dispersion. However, monitoring traffic towards a single network is often not enough to identify a worm attack. The traffic pattern could appear normal during a worm attack because the worm has not yet infected the network or will not infect it at all. Therefore, we have to monitor the network behavior at as many places as possible in order to reduce false alarms. Multiple vantage points offer more information, and exchange of observed information will improve observation accuracy. Our goal is to quickly get global information about worm attack signatures on large enterprize networks or the Internet, while ensuring that the false alarm probability is as low as possible. We have discussed the framework for decentralized cooperative detection and protection for network attacks in Chapter 4, which can be used to fulfill this purpose. In this chapter, we present how this framework can be customized to defend against Internet worms.

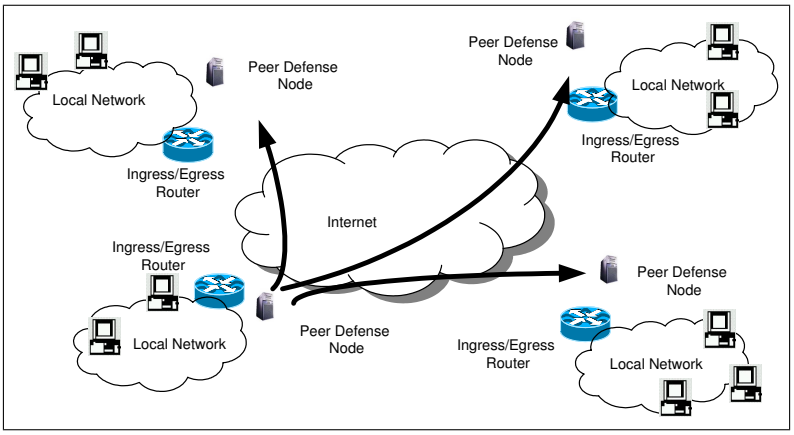

Figure 6.1: System architecture

As shown in Figure 6.1, our defense system is composed of peer defense nodes that are located at the border routers (ingress routers or egress routers) of the Internet. These nodes use available Intrusion Detection Systems (IDS) to get the local information of worm behavior of their own local network. This information includes the blacklist of the compromised nodes in the local network and worm signatures dynamically generated by local IDS. In addition to the local view of worm activity, each individual detection node also maintains a global view of the attack.

6.4 Automatic Worm Signature Generation and Aggregation

Various worm signature generation mechanisms can be used in our cooperative worm containment framework. In this thesis, we present how the Rabin footprint algorithm in Autograph [41] works within the framework. Each overlay node uses this algorithm to compute the content blocks in packet payloads. Then, it updates the local prevalence L(i,j) for each content block j, where i is the index of overlay node. Once L(i,j) is greater than a local prevalence threshold and the address dispersion is greater than a local address dispersion threshold, the overlay defense nodes begin to filter or rate limit the traffic with this signature.

As presented in Chapter 4, gossip based aggregation protocol is used to aggregate information within the cooperative defense framework. To defend against Internet worms, we use a gossip strategy to get the aggregated prevalence of the content block, which is identified by each individual overlay node using Rabin footprint algorithm [41] . At any time t, each node i maintains a list of $(content_k, count_k)$ pairs, where $content_k$ represents the signature identified by the signature generation mechanism and $count_k$ represents the prevalence of the signature. Here the k is the index for the content blocks. Our gossiping protocol

is described as the following:

1. *Let* $(content_{t-1,k}, count_{t-1,k})$ *be all pairs sent to node i in round t-1 (i is the index of node).*

2. *Let* $d_{t,i} = \Sigma_r count_{t-1,k}$ *represent the sum of the prevalence values of the signature* $content_k$ *received by node i at round t for one particular content block k.*

3. *Compare* $d_{t,i}$ *with* $Threshold_i$*. If* $d_{t,i} > Threshold_i$*, then* $content_k$ *is identified as a worm signature.*

4. *Choose target* $target_t(i)$ *from the neighbors of i uniformly at random.*

5. *Send the pair* $(content_k, \frac{1}{2}d_{t,i})$ *to* $target_t(i)$ *and i(itself).*

6.5 Simulation Results: Internet Worm Case Study

As a first step, we examine the effectiveness of this gossip based cooperative defense system on a Code-Red style worm. While future worms are likely to be more severe, we argue that any containment system must at least mitigate a worm of this magnitude.

6.5.1 Simulated Worm Spreading Experiments

To demonstrate the ability of the cooperative defense framework to defend against Internet worms, we evaluated our system with a large scale simulation using a method similar to that used by Zou et. al. [82]. We simulated the worm propagation under the defense of cooperative defense mechanism using a stand alone simulator in Windows environment. The worm propagation model used in this simulation is an extension of the model described in Section 6.2. We simplify the Internet topology by considering it as a flat network. As discussed

in [50], more than 359,000 Code Red infected hosts were observed on July 19th, 2001 by CAIDA. In our simulation we set up 360,000 vulnerable hosts and 1500 local networks. One node of the selected local networks will be set as node of the cooperative defense overlay. When the number of worm signature received exceeds a certain threshold, the worm signature is confirmed and defense overlay nodes begin to filter the packets with this worm signature. Each vulnerable host stays in one of three states at any time: susceptible, infectious, or immunize. A host is in the "immunize" state when it is immunized, no matter whether it is previous infected or susceptible. At the beginning of simulation, several hosts are initially in "infectious" state and the others are all in "susceptible" state. And the defense node has two states: monitoring or alerted.

Each defense node at the edge network can watch both traffic coming in from the ISP network and traffic going out from the local network. The defense node has two states: monitoring and alerted. In the monitoring states, it counts the number of worm infection probes observed and aggregates this information with other defense nodes to acquire the global worm probe behavior, the total number of worm probes. As we have discussed in Section 4, using the Rabin footprint algorithm [41], we can compute the worm signatures given enough number of worm probes have been analyzed (when the total number of worm probed greater than a threshold). When the overlay node has acquired signatures of the worms, it changes its state from "monitoring" to "alerted". In our experiment, the threshold of worm probes to change state of the overlay node is an adjustable parameter.

6.5.2 Experimental Results

Figure 6.2 shows the progress of infections for simulated worms with and without cooperative defense mechanism enabled. We use the infection progress model of Code Red worm as discussed in [82] for our simulation. In this experiment, 60% of the vulnerable edge networks (900 networks out of 1500) are deployed with the

cooperative defense overlay nodes. The impact of different number of cooperative nodes on defense accuracy has been discussed in [81], so we just use this specific case to demonstrate the feasibility of our approach. The global worm signature count threshold is 100.

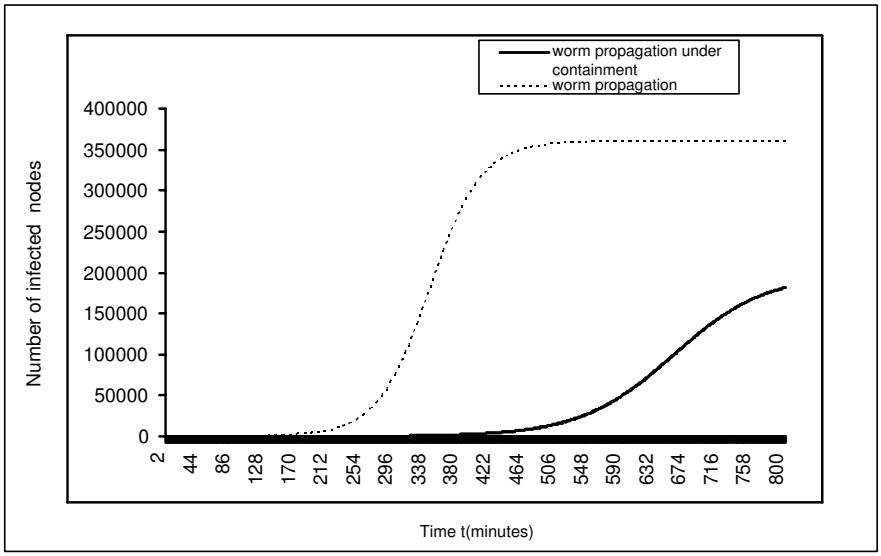

Figure 6.2: Infection progress of worms

Figure 6.3 shows the propagation when the cooperative defense overlay nodes share worm signature using different gossip intervals. We set the gossip interval to be 0.1, 0.2, 1, 10 minutes respectively. When the gossip frequency is increased, the number of infected hosts will decrease as expected. Increasing the gossip frequency of the worm information sharing will increase the communication overhead as well. As discussed in [39], the gossip based information sharing mechanism is scalable and resilient with proper selected parameters. However, we can find even when we aggregate information using gossip interval of 10 minutes, the containment effect against worm propagation is still acceptable.

Figure 6.4 shows the propagation of worm when independent containment nodes and cooperative overlay defense nodes are deployed. For both cases, we assume the individual nodes need to collect 100 worm signatures with the same

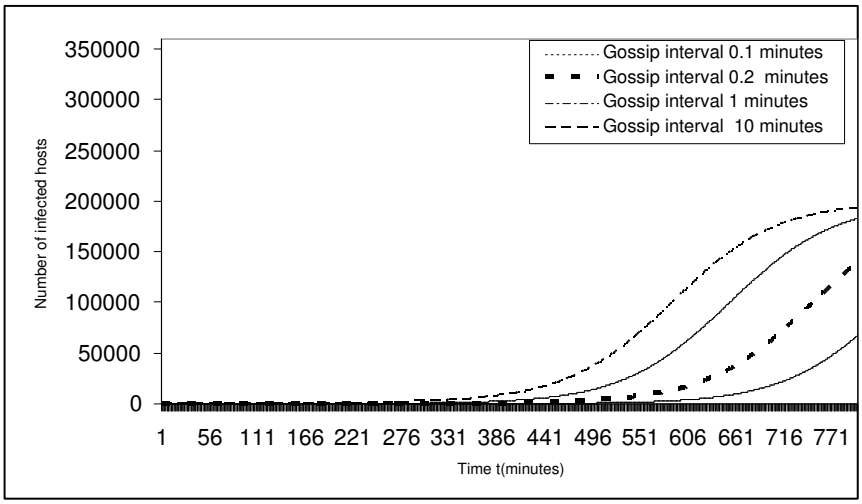

Figure 6.3: Global threshold effects

pattern before it filter the packets with that worm signature. For independent containment nodes, the worm propagation behavior is not constrained while compared with the situation without any containment. By contrast, for cooperative overlay defense nodes, the worm propagation behavior is largely constrained.

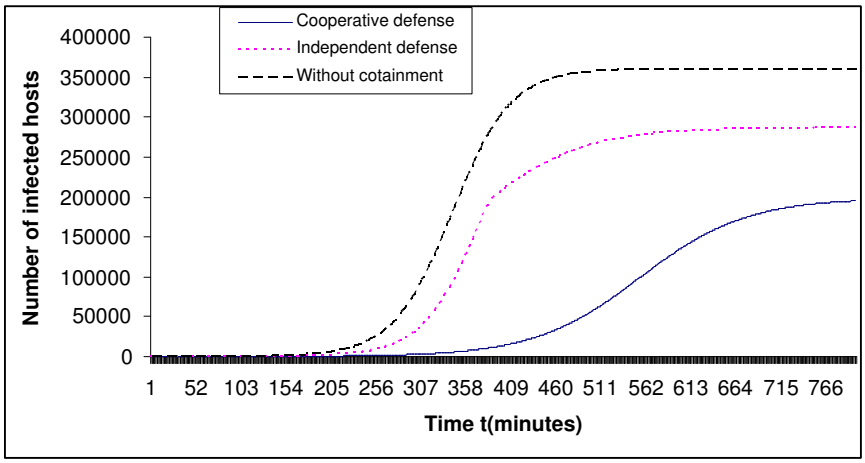

Figure 6.4: Benefit of cooperative defense

6.6 Summary

In this chapter we presented the use of the decentralized information sharing framework to defend against Internet worms. As the internal architecture and individual modules of each detection node has been discussed in Chapter 4 and Chapter 5, we focus our discussion on the customizing of the framework to defend against Internet worms. A prototype simulation of the framework and its key concept was presented and applied to detect and defend against Code Red worms. Results using the simulation demonstrate that the proposed approach is feasible and effective against worms.

Chapter 7
Conclusion and Future Work

7.1 Conclusion

Distributed denial of service and Internet worms are major threats to the global network, which cannot be addressed through isolated actions of sparsely deployed defense nodes. Instead, various defense systems must organize into a framework and inter-operate, exchanging information and services, and acting together, against the threat [48, 54]. In this dissertation, we designed a decentralized cooperative defense mechanism to protect the network infrastructure against network attacks. The protection infrastructure is a overlay network composed of individual network attack detection and defense nodes that are deployed at critical points of the global network infrastructure. Each overlay node monitors the local network attack behavior and uses a gossip based overlay infrastructure to aggregate attack information. Our design is distributed and decentralized, where each defense node collects the global network attack information independently and makes decisions on its own. As a result, the mechanism is resilient and scalable. As the aggregated information provides quasi global view of the network attacks, our mechanism can more effectively and efficiently defend against network attacks than isolated approach as long as the length of the attack is greater than the information aggregation delay. We have presented a conceptual model that defines the relationships between the level of knowledge in the distributed system and attack detection accuracy. The analysis presented demonstrates the feasibility of gossip based communication mechanisms for cooperative attack detection.

A prototype simulation of the framework and its key concepts is presented and applied to detect and defend against DDoS attacks and Internet worms. Results using this simulation demonstrate that the proposed approach is effective against network attacks.

7.2 Research Contributions

The main contributions of this thesis are as follows:

- We defined the knowledge about the network attacks in distributed system and conceptually analyzed the attack detection accuracy we can achieve through information sharing in real distributed systems.

- We presented a framework that builds on a self-managing, robust and resilient peer-to-peer overlay. This framework composed of local detection and protection agents that are placed at "strategic" locations in the Internet such as a domain gateway. These agents non-intrusively monitor the immediate network around them for possible attacks. By correlating the detection information of each individual nodes, our scheme can greatly improve the accuracy of the detection.

- We designed gossip based communication mechanism to share information about attacks within the proposed framework. This mechanism is scalable and resilient to failure.

- To demonstrate the feasibility and effectiveness of the proposed decentralized attack detection framework, we used it to detect DDoS attacks and Internet worms as case studies. Simulation results demonstrated that the proposed mechanism can efficiently detect and protect against these attacks.

7.3 Future Directions

There continues to exist many insecure areas in the Internet today that can be compromised to launch large scale network attacks. This situation will perhaps last for a long while, if not forever. Coupled with the fact that attack mechanisms and tools continue to improve and evolve, more effective detect and filter approaches must be developed in addition to the use of ingress packet filtering and other existing defense mechanisms and procedures.

Future work will fold in more topology information and vulnerability information gleamed from automated scanning and mapping tools. When the nodes know more topology information of the global Internet, they can use more intelligent gossip strategies to reduce the information sharing overhead while trying to detect attacks. Armed with these more sophisticated methods, our approach can detect attacks more efficiently. We are investigating several important questions that still need to be addressed. These include the consensus algorithm and optimal gossip periods. We also plan to validate this scheme by running them on real attack data sets.

Furthermore, a relatively homogeneous software base coupled with high speed Internet connections facilitates the widespread of Internet worms. The increasing outbreaks of Internet worms pose an immediate risk to the overall security of the Internet. When the most recent Sapphire/Slammer worm began spreading throughout the Internet, it doubled in size every 8.5 seconds. It infected more than 90 percent of vulnerable hosts within 10 minutes. Each infected machine was compromised, and could be used as a flooding source in a massive DDoS attacks later on. So far, the Internet worms have been mostly nuisances, e.g., the analysis of the Sapphire/Slammer worm revealed no intent to harm its infected hosts. However, in the future the Internet worms coupled with DDoS attacks will be move virulent, and thus, result in a chaos in the Internet. How to detect and

contain such fast spread of Internet worms in real time is an open issue.

References

[1] Internet protocol v4 address space. http://www.iana.org/assignments/ipv4-address-space/.

[2] The gnutella 0.4 protocol specification, 2000. http://dss.clip2.com/GnutellaProtocol04.pdf.

[3] Gossip-based aggregation in large dynamic networks. (3):219252, August 2005.

[4] Micah Adler. Tradeoffs in probabilistic packet marking for IP traceback. In *Proceedings of the thiry-fourth annual ACM symposium on Theory of Computing*, pages 407–418, Montreal, Quebec, Canada, 2002.

[5] Aditya Akella, Ashwin Bharambe, Mike Reiter, and Srinivasan Seshan. Detecting DDoS attacks on ISP networks. In *ACM SIGMOD Workshop on Management and Processing of Data Streams*, pages 20–23, San Diego, CA, 2003.

[6] David Andersen, Hari Balakrishnan, Frans Kaashoek, and Robert Morris. Resilient overlay networks. In *Proceedings of 18th ACM Symposium on Operating Systems Principles*, pages 131–145, Banff, Canada, October 2001.

[7] George Bakos. Sqlsnake code analysis, 2002. http://www.incidents.org/diary/diary.php?-id=157.

[8] Joao B. D. Cabrera, Lundy Lewis, Xinzhou Qin, Wenke Lee, Ravi K. Prasanth, B. Ravichandran, and Raman K. Mehra. Proactive detection of distributed denial of service attacks using mib traffic variables, a feasibility study. In *IEEE IFIP International Symposium on Integrated Network Management*, pages 609 – 622, Seattle, WA, June 2001.

[9] CERT Coordination Center. Cert advisory ca-2001- 20 continuing threats to home users, 2001. http://www.cert.org/advisories/CA-2001-20. html.

[10] CERT Coordination Center. Cert advisory ca-2001-19 code red worm exploiting buffer overflow in iis indexing service dll, 2001. http://www.cert.org/advisories/CA-2001-19.html.

[11] "CERT Coordination Center". Cert advisory ca-1999-17 denial-of-service tools, 2004. http://www.cert.org/advisories/CA-1999-17.html.

[12] Internet Storm Center. Openssl vulnerabilities, Sept. 2002. http://isc.incidents.org/analysis.html?id=167.

[13] Rocky K. C. Chang. Defending against flooding-based, distributed denial-of-service attacks: A tutorial. *IEEE Communications Magazine*, 40(10):42–51, 2002.

[14] Brent Chun, Jason Lee, and Hakim Weatherspoon. Netbait: a distributed worm detection service, 2003.

[15] Fred Cohen. Computer viruses theory and experiments. 6:2235, 1987.

[16] Keromytis Angelos D., Misra Vishal, and Rubenstein Daniel. Using overlays to improve network security. In *Proceedings of the ITCom Conference, special track on Scalability and Traffic Control in IP Networks*, pages 245–254, Boston, MA, August 2002.

[17] Drew Dean, Matt Franklin, and Adam Stubblefield. An algebraic approach to IP traceback. *Information and System Security*, 5(2):119–137, 2002.

[18] Alan Demers, Dan Greene, Carl Hauser, Wes Irish, and John Larson. Epidemic algorithms for replicated database maintenance. In *Proceedings of the sixth annual ACM Symposium on Principles of distributed computing*, pages 1–12, Vancouver, British Columbia, Canada, August 1987.

[19] Sven Dietrich, Neil Long, and Dave Dittrich. An analysis of the "shaft" distributed denial of service tool, 2004.

[20] Dave Dittrich. Distributed denial of service (DDoS) attacks/tools, 2004. http://staff.washington.edu/dittrich/misc/ddos/.

[21] Dave Dittrich. The DoS project's 'trinoo' distributed denial of service attack tool, 2004. http://staff.washington.edu/dittrich/misc/trinoo.analysis.

[22] Dave Dittrich. The 'mstream' distributed denial of service attack tool, 2004. http://staff.washington.edu/dittrich/misc/mstream.analysis.txt.

[23] Dave Dittrich. The 'stacheldraht' distributed denial of service attack tool, 2004. http://staff.washington.edu/dittrich/misc/stacheldraht.analysis.txt.

[24] Dave Dittrich. The 'tribe flood network' distributed denial of service attack tool, 2004. http://staff.washington.edu/dittrich/misc/tfn.analysis.txt.

[25] DShield.org. Distributed intrusion detection system, 2000. http://www.dshield.org, November.

[26] Mark W. Eichin and Jon A. A. Rochlis. With microscope and tweezers: An analysis of the internet virus of november 1988. In *Proceedings of the 1989 IEEE Computer Society Symposium on Security and Privacy*, Oakland, Ohio, 1989.

[27] Cristian Estan and George Varghese. New directions in traffic measurement and accounting. In *Proceedings of SIGCOMM 2002*, pages 270–313, Pittsburgh, PA, USA, 2002.

[28] Patrick T. Eugster, Rachid Guerraoui, Anne-Marie Kermarrec, and Laurent Massoulieacute. Epidemic information dissemination in distributed systems. *IEEE Computer*, 37(50):60–67, 2004.

[29] Peter Ferguson and Dave Senie. Network ingress filtering: Defeating denial of service attacks which employIPsource address spoofing, 2000.

[30] Peter Ferguson and Dave Senie. Network ingress filtering: Defeating denial of service attacks which employ IP source address spoofing, 2002. IETF RFC 2827.

[31] Associated Press for Fox News. Powerful attack cripples internet, 2002.

[32] Thomer M. Gil and Massimiliano Poletto. Multops: a data-structure for bandwidth attack detection. In *Proceedings of 10th Usenix Security Symposium*, pages 23–28, Washington, D.C., USA, August 2001.

[33] Joseph Y. Halpern and Yoram Moses. Knowledge and common knowledge in a distributed environment. In *Symposium on Principles of Distributed Computing*, pages 50–61, 1984.

[34] Brain Hancock. Trinity v3, a DDoS tool, hits the streets. *Computers Security*, 19(7), 2000.

[35] Wes Hardaker, Darrell Kindred, Ron Ostrenga, Dan Sterne, and Roshan Thomas. Justification and requirements for a national ddos defense technology evaluation facility, 2005. http://www.isi.edu/deter/docs.

[36] Salim Hariri, Tushneem Dharmagadda, Modukuri Ramkishore, Guangzhi Qu, and C.S Raghavendra. Vulnerability analysis of faults/attacks in network centric systems. In *Proceedings of Parallel and Distributed Computing Systems*, pages 256–261, Reno, Nevada, USA, 2003.

[37] John Ioannidis and Steven M. Bellovin. Implementing pushback: Router-based defense against DDoS attacks. In *Proceedings of Network and Distributed System Security Symposium, NDSS '02*, pages 100–108, Reston, VA, USA, February 2002.

[38] John Ioannidis and Steven M. Bellovin. Implementing pushback: Router-based defense against DDoS attacks. In *Proceedings of Network and Distributed System Security Symposium, NDSS '02*, pages 100–108, Reston, VA, USA, February 2002.

[39] David Kempe, Alin Dobra, and Johannes Gehrke. Computing aggregate information using gossip. In *in Proceedings of the 44th Annual IEEE Symposium on Foundations of Computer Science*, Cambridge, MA, October 2003.

[40] Jeffrey O. Kephart and Steve R. White. Directed-graph epidemiological models of computer viruses. In *Procedings of the 1991 IEEE Computer Society Symposium on Research in Security and Privacy*, Oakland,California, 1991.

[41] Hyang-Ah Kim and Brad Karp. Autograph: Toward automated distributed worm signature detection. In *Proceedings of USENIX Security Symposium*, 2004.

[42] Jun Li, Peter Reiher, and Gerald Popek. Resilient self-organizing overlay networks for security update delivery. *IEEE Journal on Selected Areas in Communications, special issue on Service Overlay Networks*, 22(1), January 2004.

[43] MengJang Lin and Keith Marzullo. Directional gossip: gossip in a wide area network. In *Proceedings of Dependable Computing - Third European Dependable Computing Conference*, pages 364–379, Berlin, Germany, 1999.

[44] Bruce A. Mah. An empirical model of HTTP network traffic. In *Proceedings of the IEEE INFOCOM*, pages 592–600, 1997.

[45] Ratul Mahajan, Steve Bellovin, Sally Floyd, John Ioannidis, Vern Paxson, and Scott Shenker. Aggregate-based congestion control, 2003. http://citeseer.nj.nec.com/530614.html.

[46] Allison Mankin, Dan Massey, Chie Lung Wu, S. Felix Wu, and Lixia Zhang. On design and evaluation of intention-driven icmp traceback. In *10th International Conference on Computer Communications and Networks*, Arizona, October 2001.

[47] Jelena Mirkovic, Gregory Prier, and Peter Reiher. Attacking DDoS at the source. In *Proceedings of ICNP 2002*, pages 312–321, Paris, France, November 2002.

[48] Jelena Mirkovic, Gregory Prier, and Peter Reiher. Alliance formation for DDoS defense. In *Proceedings of the New Security Paradigms Workshop, ACM SIGSAC*, pages 11–18, Ascona, Switzerland, August 2003.

[49] David Moore, Vern Paxson, Stefan Savage, Colleen Shannon, Stuart Staniford, and Nicholas Weaver. The spread of the sapphire/slammer worm, 2003. http://www.caida.org/outreach/papers/2003/sapphire/sapphire.html.

[50] David Moore, Colleen Shannon, and Jeffery Brown. Code-red: a case study on the spread and victims of an internet worm. In *Proceedings of the Internet Measurement Workshop (IMW)*, 2002.

[51] David Moore, Colleen Shannon, and Jeffery Brown. Code-red: a case study on the spread and victims of an internet worm. In *ACM/USENIX Internet Measurement Workshop*, France, November, 2002.

[52] David Moore, Colleen Shannon, Geoffrey M. Voelker, and Stefan Savage. Internet quarantine: Requirements for containing self-propagating code. In *INFOCOM*, 2003.

[53] David Moore, Geoffrey Voelker, and Stefan Savage. Inferring internet denial of service activity. In *Proceedings of the USENIX Security Symposium*, pages 9–22, Washington, DC, USA, August 2001.

[54] Christos Papadopoulos, Robert Lindell, John Mehringer, Alefiya Hussain, and Ramesh Govindan. Cossack: Coordinated suppression of simultaneous attacks. In *DARPA Information Survivability Conference and Exposition*, volume 1, pages 2–13, Washington, DC, April 2003.

[55] Kihong Park and Heejo Lee. On the effectiveness of probabilistic packet marking for IP traceback under denial of service attack. In *Proceedings of IEEE INFOCOM*, pages 338–347, Anchorage, Alaska, USA, 2001.

[56] Kihong Park and Heejo Lee. On the effectiveness of route-based packet filtering for distributed DoS attack preventation in power-law internets. In *Proceedings of ACM SIGCOMM*, pages 15–26, San Diego, CA, USA, August 2001.

[57] Tao Peng, Christopher Leckie, and Kotagiri Ramamohanarao. Protection from distributed denial of service attack using history-based IP filtering. In *Proceedings of IEEE International Conference on Communications*, volume 1, pages 482–486, Anchorage, Alaska, USA, May 2003.

[58] Boris Pittel. On spreading a rumor. *SIAM Journal on Applied Mathematics*, 47(1):213–223, February 1987.

[59] Phillip A. Porras and Peter G. Neumann. EMERALD: Event monitoring enabling responses to anomalous live disturbances. In *Proc. 20th NIST-NCSC National Information Systems Security Conference*, pages 353–365, 1997.

[60] Martin Roesch. The snort network intrusion detection system, 2002. http://www.snort.org.

[61] John Shoch and Jon Hupp. The "worm" programs - early experience with a distributed computation. 25(3), March 1982.

[62] Stelios Sidiroglou and Angelos D. Keromytis. Countering network worms through automatic patch generation. In *IEEE Security and Privacy*, 2005.

[63] Sumeet Singh, Cristian Estan, George Varghese, and Stefan Savage. Automated worm fingerprinting. In *Proceedings of the USENIX Symposium on Operating System Design and Implementation*, San Francisco, December 2004.

[64] Steven R. Snapp, James Brentano, Gihan V. Dias, Terrance L. Goan, L. Todd Heberlein, Che lin Ho, Karl N. Levitt, Biswanath Mukherjee, Stephen E. Smaha, Tim Grance, Daniel M. Teal, and Doug Mansur. DIDS (distributed intrusion detection system) - motivation, architecture, and an early prototype. In *Proceedings of the 14th National Computer Security Conference*, pages 167–176, Washington, DC, 1991.

[65] Alex C. Snoeren, Craig Partridge, Luis A. Sanchez, Christine E. Jones, Fabrice Tchakountio, Stephan T. Kent, and W. Timothy Strayer. Hash-based IP traceback. In *Proceedings of Sigcomm*, pages 3–14, San Diego, California, United States, August 2001.

[66] Anil Somayaji, Steven Hofmeyr, and Stephanie Forrest. Principles of a computer immune system. In *Meeting on New Security Paradigms, 23-26 Sept. 1997, Langdale, UK*, pages 75–82. New York, NY, USA : ACM, 1998.

[67] Dawn Xiaodong Song and Adrian Perrig. Advanced and authenticated marking schemes for IP traceback. In *Proceedings of IEEE Infocomm*, volume 2, pages 878–886, Anchorage, Alaska, USA, 2001.

[68] Stuart Staniford, Vern Paxson, and Nicholas Weaver. How to 0wn the internet in your spare time. In *To Appear in the Proceedings of the 11th USENIX Security Symposium (Security '02)*, 2002.

[69] Ion Stoica, Robert Morris, David Karger, Frans Kaashoek, and Hari Balakrishnan. Chord: A scalable Peer-To-Peer lookup service for internet applications. *IEEE Transactions on Networking*, 11(1):17–32, February 2003.

[70] Robert Stone. Centertrack: An IP overlay network for tracking DoS floods. In *Proceedings of the 9th USENIX Security Symposium*, pages 199–212, Denver, CO, August 2000.

[71] Rob Thomas. Bogon list v1.5, 07 Aug 2002. http://www.cymru.com/Documents/bogon-list.html.

[72] Robbert van Renesse, Kenneth Birman, and Werner Vogels. Astrolabe: A robust and scalable technology for distributed system monitoring, management, and data mining. *ACM Transactions on Computer Systems*, 21(2):164–206, May 2003.

[73] Robbert van Renesse, Yaron Minsky, and Mark Hayden. A gossip-based failure detection service. In *Proceedings of Middleware '98, the IFIP International Conference on Distributed Systems Platforms and Open Distributed Processing*, pages 55–70, England, September 1998.

[74] Giovanni Vigna and Richard A. Kemmerer. Netstat: A network-based intrusion detection system. *Journal of Computer Security*, 7(1), 1999.

[75] Haining Wang and Danlu Zhang Kang G. Shin. Detecting syn flooding attacks. In *Proceedings of IEEE Infocom*, pages 1530–1539.

[76] Helen J. Wang, Chuanxiong Guo, Daniel R. Simon, and Alf Zugenmaier. Shield: Vulnerability-driven network filters for preventing known vulnerability exploits. In *Proceedings of ACM SIGCOMM*, Portland, Oregon, August 2004.

[77] Nicholas Weaver, Stuart Staniford, and Vern Paxson. Very fast containment of scanning worms. In *Proceedings of the 13th USENIX Security Symposium*, San Diego, USA, August 2004.

[78] Stefan Savageand David Wetherall, Anna Karlin, and Tom Anderson. Practical network support for IP traceback. In *Proceedings of the ACM SIGCOMM Conference*, pages 295–306, Stockholm, Sweden, August 2000.

[79] The Ramen Worm. Ciac information bulletin, 2001. http://www.ciac.org/ciac/bulletins/l-040.shtml.

[80] David K. Y. Yau, John C. S. Lui, and Feng Liang. Defending against distributed denial of service attacks with max-min fair server centric router throttles. In *Proceedings of the Tenth IEEE International Workshop on Quality of Service*, pages 35–44, Miami Beach, FL, 2002.

[81] Vinod Yegneswaran, Paul Barford, and Somesh Jha. Global intrusion detection in the DOMINO overlay system. In *The 11th Annual Network and Distributed System Security Symposium (NDSS)*, Feburary 2004.

[82] Cliff Changchun Zou, Weibo Gong, and Don Towsley. Code red worm propagation modeling and analysis. In *In Proceedings of the 9th ACM Conference on Computer and Communications Security*, pages 138–147, November 2002.

Curriculum Vita

Guangsen Zhang

2006 PhD, Electrical & Computer Engineering, Rutgers University, USA.

2003 MS, Electrical & Computer Engineering, Rutgers University,USA.

1997 MEng, Information Engineering, Beijing University os Posts and Telecommunication, PRC

1994 BEng, Information Science and Technology, Xian Jiaotong University, PRC

2001-2005 Graduate Research Assistant, The Applied Software Systems Lab, Center for Advance Information Processing, Rutgers University, USA

2001-2001 Senior Software Engineer, Agilent China Software Design Center, Agilent Technologies, PRC

1998-2001 Staff R&D Engineer, Bell Labs China, Lucent Technologies, PRC

1997-1998 Staff R&D Engineer , Putian Telecom, PRC

1994-1997 Research Assistant, The Applied Communication Systems Lab, Beijing University of Posts and Telecommunication, PRC

Publications

G. Zhang and M. Parashar, "Dynamic Context-aware Access Control for Grid Applications", *Proceedings of the 4th International Workshop on Grid Computing (Grid 2003)*, Phoenix, AZ, USA, November 2003.

G. Zhang and M. Parashar, "Context-aware Dynamic Access Control for Pervasive Computing", *2004 Communication Networks and Distributed Systems Modeling and Simulation Conference (CNDS'04)*, San Diego, CA, USA, January 2004.

G. Zhang and M. Parashar, "Environment Sensitive Access Management for Pervasive Grid Applications", *Cluster Computing: The Journal of Networks, Software Tools, and Applications*, Kluwer Academic Publishers, Vol. 9, No. 2, 2006.

M. Parashar, H. Liu, Z. Li, V. Matossian, C. Schmidt, G. Zhang and S. Hariri, "AutoMate: Enabling Autonomic Grid Applications", *Cluster Computing: The Journal of Networks, Software Tools, and Applications, Special Issue on Autonomic Computing*, Kluwer Academic Publishers, Vol. 9, No. 1, 2006.

G. Zhang and M. Parashar, "Cooperative Defense against Network Attacks", *Proceedings of the 3rd International Workshop on Security In Information Systems (WOSIS 2005), 7th International conference on Enterprise Information Systems (ICEIS 2005)*, Miami, FL, USA, May 2005.

G. Zhang and M. Parashar, "Cooperative Defense against DDoS Attacks", *Proceedings of the 2005 International Conference on Security Management (SAM 2005)*, Las Vegas, NV, USA, CSREA Press, June 2005.

G. Zhang and M. Parashar, "Cooperative Defense against DDoS Attacks", *Journal of Research and Practice in Information Technology (JRPIT)*, Australian Computer Society Inc., Vol. 38, No. 1, February 2006.

M. Agarwal, V. Bhat, Z. Li, H. Liu, B. Khargharia, V. Matossian, V. Putty, C. Schmidt, G. Zhang, S. Hariri and M. Parashar,"AutoMate: Enabling Autonomic Applications on the Grid", *Proceedings of the Autonomic Computing Workshop, 5th Annual International Active Middleware Services Workshop (AMS2003)*, June 2003.