Distributed Detection of DDoS Attack

Santhosh Kumar Karre

***Abstract*—Denial of Service (DoS) and Distributed Denial of Service (DDoS) attacks are attempts to make a server resources unavailable to its intended users. SYN flooding attack is one type of DDoS attack. In SYN flooding attack, the attacker sends flood of SYN packets to victim server. This paper focus on effective detection of SYN flooding attack. The aim of this paper is to compare the results of detection of DDoS attack in both centralized and distributed approaches. The proposed approach is distributed detection of DDoS attack which reduces traffic in the network and load on server which is very high in centralized detection. The experiments are conducted in Network Simulator 2 (NS2) to validate our distributed detection of DDoS attack . The experiments are conducted in Centralized and Distributed approaches. The total actual victims found by centralized approach is 78.75 % and the total number of actual victims found by our distributed approach is 77.5 %. In both centralized and distributed approaches the results matched well. The traffic in network and load on the central DDoS monitor in our distributed approach is less, which encourages research in distributed detection of SYN flooding attack instead of centralized approach.**

***Index Terms*—DDoS Attack and SYN flooding attack.**

I. Introduction

Denial of Service (DoS) and Distributed Denial of Service (DDoS) attacks are attempts to make a server resources unavailable to its intended users. Information Security have three fundamental objectives: they are Information Integrity, Confidentiality and Availability. DoS attack is an attack on availability. In this attack the attacker makes the server busy in processing illegitimate requests thereby making server resources unavailable for legitimate clients. In DDoS attack, multiple DoS attacks are carried out from several agents (Zombies) at a time on the victim (target server).

A. DoS Attack

In DoS attack attacker sends flood of requests to the victim, thereby making the victim(target server) in a position to not serve for legitimate clients. The DoS attack can be carried out in various forms such as crashing servers, crashing routers, overwhelming the network with high traffic, damaging server critical resources (processing time, memory) etc. DoS attack victim can be either server, operating system, protocol which is used in network communication, network bandwidth, disk space, routing information etc. If the victim is server then we can have various types of DoS attacks like smurf attack [1], ping of death [2] etc. If the operating system is the victim then the attack is carried out by knowing the vulnerabilitys in the design or implementation of operating system. DoS attack can be performed by knowing the vulnerabilitys in the protocol thereby making protocol not working. SYN flooding attack is a result of weakness in TCP 3-way handshake procedure.

Manuscript received January 15, 2013; revised April 2, 2013.

Santhosh Kumar Karre is with IBM India Software Labs, Bangalore, India. As part of my job at IBM, I am involved in Network management product development.

B. DDoS Attack

DDoS attack is an extension of DoS attack. Wherein the attackers starts attacking the victim with DoS attacks at the same time in coordination.

In DDoS attack there is one master(who is actual attacker) and number of attacking agents (zombies). Master is responsible for issuing control commands for zombies, and the zombies are responsible for generating actual attack traffic. The Fig. 1 shows the schematic diagram of DDoS attack. The more number attacking agents makes that when compared to DoS attack there is less probability for the zombies to get detected by victim.

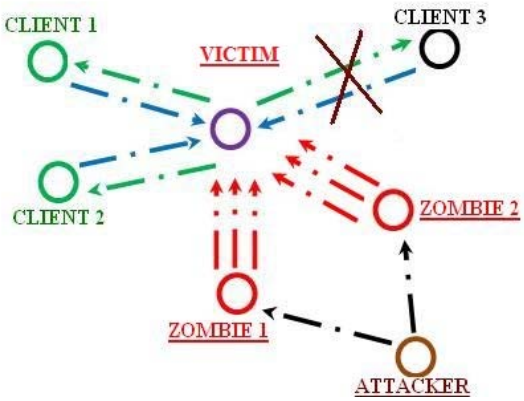

Fig. 1. DDoS attack

There are a number of mechanisms to carry out DDoS attack, such as TCP SYN flooding [3], UDP flooding attack [4], Ping of death [1] and DNS attack [5]. In this paper we will concentrate only on SYN flooding attack.

C. SYN Flooding Attack

The SYN Flooding attack is result of weaknesses in TCP Protocol design. It uses the flaws in the TCP 3-way handshake mechanism [5]. The TCP 3-way handshake mechanism is shown in Fig. 2. In SYN flooding attack, the attacker sends flood of SYN packets to victim server with spoofed source IP addresses [3]. Server stores the state information of each of these attack connections. State information includes, the source IP address, source port, destination IP address and destination port etc. Server responds with SYN-ACK packets which are destined for spoofed IP addresses, so attacker does not receive SYN-ACK packets. It causes the wastage of server resources in storing connection information of half open connections (Half open connection is a connection which is established from only one side). The victim server is

DOI: 10.7763/IJFCC.2013.V2.241

busy in processing SYN packets which are originated from attacker, thus server is not in a position to serve legitimate clients.

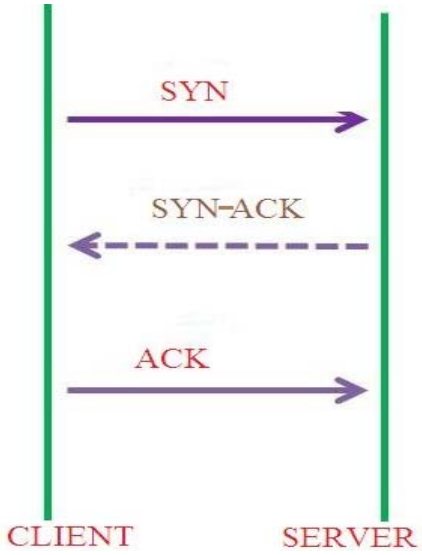

Fig. 2. TCP 3-way handshake

D. Prior Work for Detection of DDoS Attack

We can fight DDoS attacks either by detecting them or defense by preventing them in the first phase. In this paper, we focus only on DDoS detection mechanism, in particular detecting SYN flooding attacks.

In order to detect attack traffic, the detection system should be able to distinguish between the normal traffic and attack traffic. DDoS attack detection can be done in various ways such as, Pattern Detection [6], Anomaly Detection [7], Third party detection [8] and Packet Marking [9] etc. For example in [10] the authors I.B.Mopari, S.G.Pukale and M.L.Dhone assumed that attackers can not modify hop count. In their approach, they used TTL field for calculating hop count. But in fact if the attacker spoofs the initial TTL values, then the proposed algorithm may not work properly. In [11] Yoohwan Kim, d Ju-Yeon Jo, Jonathan Chao and Frank Merat the blocking of TCP SYN flooding attack is done in a centralized way, so even if the attacker want to knockout the router, attacker can overwhelm the router with flood of spoofed IPs, thereby making the router to not work properly, so the detection system for DDOS attack detection may fail. In [12] Yonghua You, Mohammad Zulkernine and Anwar Haque, detection of DDOS attack is achieved by calculating distance from TTL value, but it is always not possible to calculate true distance from TTL. In [13] Tao Peng, Christopher Leckie and Kotagiri Ramamohanarao, detection of DDOS attack is done by considering anomalies in source IP address. Anomalies in source IP addresses Can Not Reflect the SYN flooding DDoS attack, so it can not detect syn flooding DDoS attack.

The Ganguly *et al*. [14] proposed a solution for detection of DDoS attack, our proposed solution is an extension of Ganguly *et al*. work. In their approach they used data stream algorithm to detect DDoS attack. The advantage of data stream algorithms is that they uses less memory and it requires processing time to get information from large pool of data. The advantages of their proposed solution are we can monitor many servers at a time for DDoS attack, it requires less processing time and less memory. The only problem with their *et al*. work is that it causes more traffic in the network and it leads to more processing overhead on central DDoS Monitor System.

II. Proposed Method

In this paper, we propose a technique that reduces both the traffic in the network and load on the DDoS attack monitoring server which is very high in [14] approach. Our proposed solution is an extension of Ganguly *et al.* work. The problem in Ganguly *et al*. is that, tuples (Source, destination,#1) for all connections in the ISP have to be sent to a centralized DDoS monitor where the DDoS detection algorithm is run. This results in a high network overload, and high processing overhead at the DDoS monitor.

We proposed a solution which can be used for tracking SYN flood attack effectively. Our solution is distributed detection of DDoS attack. In our approach we used count min sketch algorithm to keep track of top-K destinations which are having maximum number half open connections. The advantage of count min sketch is that it allows deletion of legitimate connections from the pool of observation. Our approach is as follows: for each of incoming SYN packet the count for that destination is incremented by 1 and for each of the ACK from a destination count for the destination is decrements by 1. Each router in the network stores (Source, Destination, count of half open connections) . The Count min Sketch algorithm runs on each router in the network. The input for count min sketch is collection of such tuples. The output is top K destinations which having maximum number of half open connections.

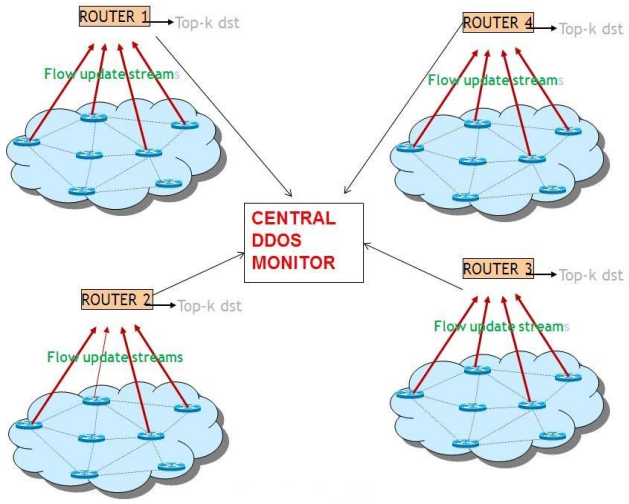

Fig. 3. TCP 3-way handshake

Outputs of all routers send to one main DDOS attack monitoring server. DDOS attack monitoring server merges outputs from all routers in the network and it runs DDOS attack detection algorithm on merged output. As the size of output produced by router's DDOS attack detection algorithm is very low when compared with total size of all tuples for each connection in the network, the traffic in our approach is reduced and the processing overhead on the main DDoS attack monitor is also reduced. The block diagram of our proposed method is shown in Fig. 3.

We believe that a frequency of half open connection for a

destination IP address provides a very robust indicator of potential DDoS activity. So the problem, is seeks to find the top-k destinations connected to the most of the half open connections. As Count min Sketch synopses incur a small (logarithmic) number of steps to process each streaming update. Our algorithm is guaranteed logarithmic time to find an approximate set of top-k destinations (and, corresponding distinct frequencies) that is provably close (with high probability) to the actual top-k set. As it is taking less time it can be readily deployed to monitor large networks transmitting large volumes of IP packet data.

In our approach we used data stream algorithms such as count min sketch and Misra Gries Algorithm. In addition to these algorithms we used heap sort algorithm to maintain top-K destinations. In this next section we discuss count min sketch and Misra Gries algorithm.

A. Count Min Sketch

Count min sketch was developed by Graham Cormode and S.Muthu Krishnan [15]. It is used to answer frequency related queries in the data stream processing. Count min sketch is a variant of Bloom Filter [16]. It is used to store and retrieve the frequencies (counts) of input elements efficiently. The name of count min sketch is derived based on the two operations which are performed in count min sketch. The first operation is counting frequency of input elements. The second operation is computing minimum. Count min sketch maintains a 2-D array of size w # d, where w is the width (number of columns in array) of array and d is depth (number of rows in array) of array. Count min sketch maintains d hash functions H_1, H_2, H_3... H_d. Each of these d hash functions is associated with one row in count min sketch array as follows: function H_1 is associated with first row, function H_2 is associated with second row, function H_3 is associated with third row etc. The Fig. 4 shows the schematic diagram of count min sketch. The input parameters for count min sketch are accuracy (ε), certainty (δ). The hash table size and number of hash functions required for count min sketch are calculated from the input parameters ε and δ as follows:

Hash table size (w) = e/ε.

Number of hash functions (d)= ln 1/δ.

Initially all entries in the count min sketch array are initialized to zero, which means that counts of all input elements is initialized to zero. The d hash functions are used to insert frequencies of elements into array and deletion of frequencies of elements from array. The update procedure for inserting (or deleting) an element to (from) count min sketch is as follows:

1) Update Procedure: To insert an element X into count min sketch array, the procedure is as follows: we calculate hash values of X *w. r. t* all hash functions $H_1(X)$, $H_2(X)$, $H_3(X)$, ... $H_d(X)$. The count of counters which are at ithrow, $H_i(X)^{th}$ column of (where i=1, 2, 3, ..., d) count min sketch array are incremented by 1. To delete an element X from count min sketch array, we follow same procedure as insertion procedure of count min sketch, but instead of incrementing value of counters we decrement the value of counters by 1.
2) Procedure To Answer A Query: The query for count min sketch is finding the frequency of an input element. The procedure to find frequency of an element X is as follows: First we calculate the hash values of X w.r.t all d hash functions. Next the minimum count among all counts which are at calculated hash values is taken as true count for that element. The proof of algorithm correctness can be found in [15].

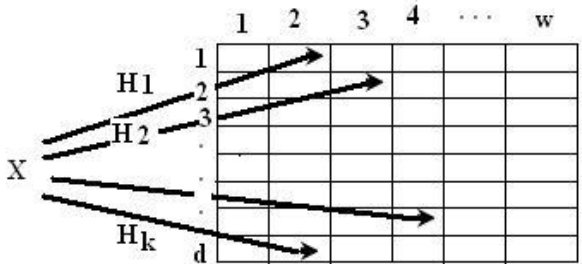

Fig. 4. Count min sketch

B. Misra Gries Algorithm:

Misra Gries algorithm is used to find the frequent elements in an input data stream or input array. The input for the algorithm is set of elements in a data stream or an array of elements. The parameter for Misra Gries algorithm is a constant number K. The output of the algorithm is set elements which are having frequency greater than N/K, where N is number of elements already inserted into algorithm. The basic idea of Misra Gries algorithm is Pigeon hole principle. Lets recall the Pigeonhole principle. There are n pigeons, m holes and $m < n$. This implies that there exists at-least one hole which is having ≥ 2 pigeons in it. The basic idea of this algorithm can be generalized for $K > 2$. The detailed proof of the Misra Gries Algorithm can be found at [17], [18].

III. Experiments and Results

We have conducted two experiments to check validity of our approach, of which one experiment is done in Network Simulator 2 (NS2) simulator [19]. The experiments are conducted by considering various network topologies each with different sizes. In each trial of experiment network is built randomly, The random parameters in our experiments are as follows: topology of network, number of agents attached to every node, start time of each traffic generator and end time of each traffic generator.

A. Experiment 1

We have conducted this experiment to compare results in both distributed approach and centralized approach for detecting top destinations with maximum number of half open connections. This experiment is conducted to simulate the idea (insertion of all connections into pool of observation and deletion of legitimate connections from pool of observation) of Ganguly *et al*. [14]. In this experiment to find top destinations we used count min sketch algorithm. We conducted this experiment using NS2 Simulator [19]. The experimental setup is as follows: In each trial of experiment, we built network with various random parameters. The random parameters in this experiment are as follows: The links between nodes in the network, number of agents attached to a node, the start time and end time of traffic generator. The sizes of network topologies used in this experiment are 60, 120, 180 and 210 nodes. In this experiment, at each node in the net-

work we have used a 2 D array of size w×d (count min sketch array). The count for destination is incremented by 1 for each of SYN packet towards the destination, we decrement the count of a destination by 1 for each ACK towards destination. The procedure for updating count of a destination is given in section 2.1. The count of the count min sketch is given as input for the heap sort algorithm, heap sort algorithm takes destination id and count of the estimation as input. The tuple (destination id, destination count) is treated as a single data structure. The heap tree is constructed by taking these data structures as nodes of the tree. The heap sort algorithm always maintains the max heap property on the destination count.

The experiment procedure is as follows: Initially we selected 4 nodes as victim nodes of DDOS attack. We noted the nodes which are chosen as attack victims, then we ran detection of DDOS Attack algorithm in both distributed approach and centralized approaches independently. The output of detection algorithm is 4 nodes which are victims of DDOS Attack. The nodes which are reported by detection algorithm are may or may not be actual victim of DDOS Attack. We noted down victims reported by algorithm. We calculated the number nodes which are actual victims of DDOS Attack. The value for number nodes which are actual victims of DDOS Attack can be 4 in best case (where all the reported nodes are actual victims of DDOS Attack), and it is 0 in worst case (where none of reported nodes are victims of DDOS Attack). To get results of detection approach accurately, for each size of we ran experiment in 5 trails. The sum of results of 5 trails is taken as result of that size input. In Fig. 5 X-Axis represents the size network and Y- Axis represents total number actual victims found by detection algorithm for that particular size. So in best case the value of sum of actual victims reported by algorithm is 20 and in worst case it 0. The experiments are conducted for 4 different sizes of topologies, so the maximum number actual victims found by algorithm in either of the approach can be 80 (4× 20) in the best case. The minimum number actual victim found by either of the approach is 0 in worst case (which means that our algorithm can not found any victims correctly).

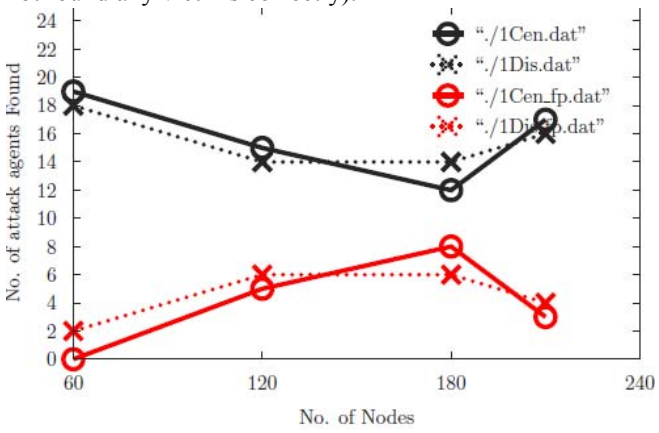

Fig. 5. Comparison of centralized and Distributed detection of DDoS attack using count min sketch

The above experiment is conducted both in centralized approach and distributed approach independently. We plotted the results of detection of DDOS attack in both approaches in Fig. 5. In Fig. 5 the thick lines represents results pertaining to centralized approach and dashed lines represents results pertaining to distributed approach. The total actual victims found by centralized approach are 63, which is equal to 78.75 % and the total number of actual victims found by our distributed approach is 62 which is equal to 77.5 %. The results in both approaches match in most of the cases as shown in Fig. 5. In Fig. 5 above two lines represents the detection values of centralized and distributed approaches, and below lines represents false positives of centralized and distributed detection approaches. The traffic in our approach is less, because we are sending results of routers to the Central DDoS monitor.

B. Experiment 2

This experiment is a static one, where we generate only arrays as input for the Misra Gries algorithm. To simulate the event that some legitimate connections might also be open at the time of making the measurement in real, we deliberately put every tuple in the array with a small probability and the attack connections are inserted with probability 1. This serves as false positives. we have used Misra Gries algorithm to detect victims of DDoS Attack.

The setup for this experiment is as follows: We constructed network by using various random network parameters. The random features in this experiment are topology of network, number of connections to each node and the number of open connections sent from node, source and destinations of connection. We have conducted this experiment by varying size of topology in the network. For a fixed size topology we varied percentage of legitimate that goes into Misra Gries algorithm. We have considered three different sizes of topology (60, 120, 240 nodes). We considered three different percentages (5, 40, 80) by which legitimate connections are inserted into Misra Gries algorithm. To validate our distributed detection approach we conducted the same experiment with various possible combinations of size of topology vs percentage of legitimate connections. So we will get 9 different combinations of size of topology vs percentage of legitimate connections. In Fig. 6 *X*-axis represents the size of topology, *Y* - Axis represent percentage of legitimate connections and each point in Fig. 6 resembles a combination of size vs percentage of experiment.

The experiment procedure is as follows: Initially we selected 3 nodes as victim nodes of DDOS attack. We noted the nodes which are chosen as attack victims, and then we ran detection of DDOS Attack algorithm in both distributed approach and centralized approaches independently. The output of detection algorithm is 3 nodes which are victims of DDOS Attack. The nodes which are reported by detection algorithm are may or may not be actual victim of DDOS Attack, it is because of false positives which we inserted wanted. We noted down victims reported by algorithm. We calculated the number nodes which are actual victims of DDOS Attack. The value for number nodes which are actual victims of DDOS Attack can be 3 in best case (where all the reported nodes are actual victims of DDOS Attack), and it is 0 in worst case (where none of reported nodes are victims of DDOS Attack). To get results of detection approach accurately, for each combination of size vs percentage we ran experiment in 5 trails. The sum of results of 5 trails is taken as

result of that combination. In Fig. 6 Z - Axis represents the sum of number of actual victims reported by detection algorithm. So in best case the value of sum of actual victims reported by algorithm is 15 and in worst case it 0.

The above experiment is conducted both in centralized approach and distributed approach independently. We plotted the results of detection of DDOS attack in both approaches in Fig. 6. In Fig. 6 the thick lines represents results pertaining to centralized approach and dashed lines represents results pertaining to distributed approach. The results in both approaches match in most of the cases as shown in Fig. 6. As expected, for high percentage insertion of legitimate connections the detection of victims which are under DDOS attack is low. For low percentage insertion of legitimate connections the detection of victims which are under DDOS attack is high. The plotted results shows that when the percentage of legitimate connections having half open connection are less, the centralized approach of detection of DDOS Attack algorithm works well, but when percentage of legitimate connections having half open connection are increases, the distributed approach of detection of DDOS Attack algorithm works well when compared with centralized approach.

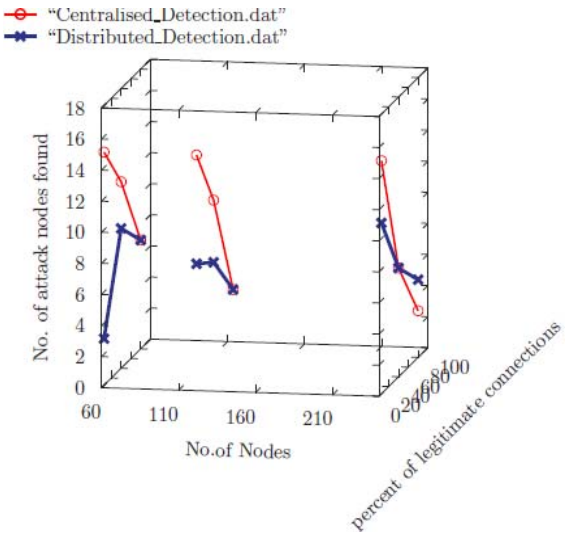

Fig. 6. Comparison of centralized and Distributed detection of DDoS attack using Misra Gries algorithm

IV. Experiments and Results

Our proposed detection approach aims to reduce traffic in the network and load on the DDOS monitor which is very high in centralized detection approach. We conducted experiments in NS2 simulator to compare results in centralized and distributed approaches for detecting DDoS attack. The total actual victims found by centralized approach is 78.75 % and the total number of actual victims found by our distributed approach is 77.5 %. The traffic in our distributed approach is less, because we are sending results of routers to the Central DDoS monitor instead of sending all tuples to Central DDoS monitor. The distributed approach decreases traffic on the network and computational overheads at the central DDoS monitor, we believe that it could be an interesting alternative to explore.

Acknowledgment

I am deeply indebted to my professor Dr. M.V. Panduranga Rao for his constant guidance and support right from my problem selection to experimentation. He patiently listened to all the problems I faced during the course of this work and appropriately guided me with his full support.

I thank to my friend M.Srinivas for his friendly support, for his help in writing this paper and who provided valuable suggestions in improving this paper.

References

[1] S. Kumar, "Smurf-Based Distributed Denial of Service (DDOS) Attack Amplification in Internet," Internet Monitoring and Protection, International Conference on, pp. 25, 2007.
[2] Ping of Death. [Online]. Available: http://linuxmafia.com/faq/Security/ping-of death.html
[3] S. Kumarasamy and A. Gowrishankar, "An active defense mechanism for Tcp Syn Flooding Attacks," *CoRR*, vol. abs/1201.2103, 2012.
[4] R. Xu, W. L. Ma, and W. L. Zheng, "Defending Against Udp Flooding by Negative Selection Algorithm Based on Eigenvalue Sets," in P*roc. of the Fifth International Conference on Information Assurance and Security*, 2009, pp. 342–345.
[5] J. Mirkovic and P. Reiher, "D-ward: A Source-end Defense Against Flooding Denial-of-Service Attacks," *IEEE Transactions on Dependable and Secure Computing*, vol. 2, pp. 216–232, 2005.
[6] The Open Source Network Intrusion Detection System. [Online]. Available: http://www.snort.org/
[7] J. Yan, S. Early, and R. Anderson, "The xenoservice a distributed defeat for distributed denial of service," in *Proceedings of ISW 2000*, 2000.
[8] B. S. L. M and T. Taylor. Icmp Traceback Messages. (2003). [Online]. Available: http://ietf.org/ internet-drafts/draft-ietf-itrace-01.txt.
[9] S. Savage, D. Wetherall, A. Karlin, and T. Anderson, "Practical Network Support for IP Trace back," 2000, pp. 295–306.
[10] I. B. Mopari, S. G. Pukale, and M. L. Dhone, "Detection and Defense Against Ddos Attack with IP Spoofing," in *Proc. ICAC3 '09*, 2009.
[11] Y. Kim, J.-Y. Jo, J. Chao, and F. Merat, "High Speed Router Filter for Blocking Tcp Flooding Under Ddos Attack," 2003.
[12] Y. You, M. Zulkernine, and A. Haque, "A distributed defense framework for flooding-based ddos attacks," *ARES. IEEE Computer Society*, 2008, pp. 245–252.
[13] T. Peng, C. Leckie, and K. Ramamohanarao, "Detecting Distributed Denial of Service Attacks Using Source IP Address Monitoring," in *Proc. of the Third International IFIP-TC6 Networking Conference*, 2002, pp. 771–782.
[14] S. Ganguly, M. N. Garofalakis, R. Rastogi, and K. K. Sabnani, "Streaming Algorithms for Robust, Real-Time Detection of Ddos Attacks," in *Proc. ICDCS*, 2007, pp. 4.
[15] G. Cormode and S. Muthukrishnan, "An improved data stream summary: the count-min sketch and its applications," *J. Algorithms*, vol. 55, no. 1, pp. 58–75, 2005.
[16] B. H. Bloom, "Space/time trade-offs in hash coding with allowable errors," *Commun. ACM*, vol. 13, pp. 422–426, July 1970.
[17] A. Chakrabarti, *Cs85: Data Stream Algorithms Lecture Notes*, Fall 2009.
[18] J. Misra and D. Gries, "Finding repeated elements," *Sci. Comput. Program*, vol. 2, no. 2, pp. 143–152, 1982.
[19] T. Issariyakul and E. Hossain, *Introduction to Network Simulator NS2*, 1st ed. Springer Publishing Company, Incorporated, 2008.

Santhosh Kumar Karre did M. Tech in Computer Science and Engineering from Indian Institute of Technology Hyderabad, Andhra Pradesh, India in 2011. I did my B.Tech in Computer Science and Engineering from Srinidhi Institute of Science and Technology Hyderabad, Hyderabad, Andhra Pradesh, India in 2009.

He is currently working in IBM India Software Labs, Bangalore, India. As part of his job at IBM, he is involved in Network management product development.