Quantifying Distributed System Stability through Simulation: A Case Study of an Agent-Based System for Flow Reconstruction of DDoS Attacks

Omer Demir
Department of Computer Science
City University of New York Graduate Center
New York, U.S.A.
Email: odemir@gc.cuny.edu

Bilal Khan
Department of Mathematics and Computer Science
John Jay College
New York, U.S.A.
Email: bkhan@jjay.cuny.edu

***Abstract*—We investigate the stability properties of a novel agent-based system for the detection of network bandwidth-based distributed denial of service (DDoS) attacks. The proposed system provides a description of the structure of flows which comprise the DDoS attack. In doing so, it facilitates DDoS mitigation at or near attack traffic sources. The constituent agents within the system operate at the inter autonomous system (AS) level, comprising a distributed collection of IP-layer network taps which self-organize in response to attack flows. We formalize the notion of stability for the proposed system, and show how we can use simulation to identify regions of instability within the system's parameter space. We then modify our system design to circumvent the uncovered singularities, and demonstrate the efficacy and tradeoffs implicit in our redesigned system.**

***Keywords*-stability; simulation; agent-based; distributed denial of service; flow reconstruction.**

I. Introduction

Denial of service (DoS) occurs when legitimate users are prevented from getting access to shared resources or services. If DoS is originated from a large number of distributed attackers, the event is termed a Distributed Denial-of-service (DDoS) attack. DDoS attacks have been identified as the most urgent Internet security concern by Arbor Network's 2008 survey [1].

The roots of the DDoS problem lie in the design of the Internet architecture [2] itself: (1) In order to gain the most of the Internet, its network link resources are shared, but there is no enforcement of fair sharing; (2) The network core processes high volumes of traffic so core components can do very little processing per packet, thus all computations (e.g. those ensuring security) must be performed at the edge; (3) The Internet's constituent networks are managed by different authorities, and this heterogeneity makes widespread deployment of DDoS defense mechanisms difficult.

"Solving the DDoS problem" has many interpretations. The one we focus on here is the problem of finding the true sources and mechanics of attacks. Finding the source of attack packets is difficult because the source address field of IP header of packets is easy to forge or spoof. Moreover, current standards do not require network devices to maintain information about paths that packets take. We define Flow Reconstruction as "Actions taken to find the true source and route of packets". There have been different approaches to this problem, including actively interacting with network traffic [3], [4], probabilistic and packet marking techniques [5], [6], and hash based logging [7].

II. Related Work

Active Interaction is a strategy of interfering with attack traffic in order to deduce information about attack sources based on the systemic reaction to the interference. Backscatter is the prototypical example of this technique [4], operating at the level of BGP level routers. Backscatter finds the point of entry of the attack packets into BGP-level Internet backbone. While innovative, Backscatter has many drawbacks, most notably, a huge collateral effect by which legitimate traffic to the victim will also be blocked at the BGP level.

Packet Marking relies on routers adding identification information to the packets that they forward to that packets reveal the path they have taken [5]. Marking every packet is not feasible because of packet processing overhead introduced by checksum recalculation. Probabilistic packet marking (PPM) selects packets randomly and marks them with transit router address information (typically using the the IP identification field). The main limitation of packet marking is that the path convergence is slow. Park and Lee [8] showed that PPM is effective at localizing the attack origin in single-source attacks but that as the number of mounted attack sources increases, the traceback is rendered more difficult.

Hash-Based Traceback was introduced by Snoeren and Alex [7]. In their system the packets are stored in a space-efficient data structure called a Bloom filter [9] as follows: Periodically, a 2^n-sized bit-array is initialized to all zeroes. Whenever a packet arrives, k distinct n-bit packet digest functions are computed from the packet's immutable header fields; the array is marked at indices corresponding to these k values. When an IP Traceback request arrives, it specifies the time, and provides a copy of the packet to be traced. The

system then asks all the routers, starting from the last one, whether they have any record of the given packet. These queries are answered by consultation with the router's local Bloom filter databases. The main drawback of hash-based IP traceback is that in order to perform traceback, a copy of an attack packet must be presented to the system as soon as the attack starts–since otherwise Bloom filter tables might be discarded and the records lost.

III. Objectives

We shall develop scalable system for collecting information about the structure and dynamics of DDoS attacks. The system will (1) self-organize and require no centralized coordination, (2) effectively produce structural and temporal data concerning DDoS attack flows, (3) be effective even when system deployment levels are modest, (4) not significantly increase router processing load, (5) not require modification of router internals, and (6) be built using existing Internet protocols. The example of questions we seek to be able to answer using the system are: What does the attack flow (tree) structure look like during the attack?

We seek to instrument an agent-based system to be able to answer such a question. Because we do not wish to modify router internals, agents must be viewed as devices which reside on links. Before we design and describe how these agents operate, let us define the problem that they must collaborate to solve, and how we shall evaluate the solution.

IV. Mathematical Model

An instance of the **Flow Reconstruction Problem** (FRP) consists of a tuple (G, R, A, D, v) where $G = (V, E)$ is a network on nodes V (and $E \subset V \times V$ represents a set of undirected edges between nodes), $R : V \times V \rightarrow V$ is a routing table, where $R(u, d) = v$ represents the next hop v for u to go towards d. $D \subseteq V$ is a set of attackers, $v \in V$ is the victim, $A \subseteq E$ is the set of agents. We define **flow** $f(d, v, n)$ as $f(d, v, 0) = d$, and for every $n \geqslant 0$ $f(d, v, n + 1) = R(f(d, v, n), v)$. The sequence $F(d, v) = (f(d, v, i); i = 0, 1, ...)$ is called the **flow** from d towards v.

A **valid solution** to an instance (G, R, A, D, v) of FRP is a logical overlay network $L = (S, E_S)$ on a subset of agents $S \subset A$ where $E_S \subset S \times S$ and (i) Every agent in the solution set S lies on the flow from some attacker to the victim; (ii) If two agents appear successively in the flow from some attacker to the victim, then these two agents are connected by a logical link in L_S. A solution $L = (S, E_S)$ is said to be **maximum valid** if $e \in A \wedge \exists d \in D \wedge i \in \mathbb{N} \wedge (f(d, v, i) = e) \Rightarrow e \in S$.

A. Performance Measures

How do we know how "good" a particular solution to FRP is? We define performance measures to achieve this, which is to say real-valued functions of problem-solution pairs: (G, R, A, D, v), (S, E_S). We need some preliminary notation to define our performance measures. Let $d_{G,R,v} : V \times V \rightarrow \mathbb{N} \cup \{\infty\}$ be defined so that $d_{G,R,v}(x, y)$ equals the number of hops that a packet takes to reach y when it is sent by x to v in graph G according to routing table R. Note that $d_{G,R,v}$ is not generally symmetric or transitive, and hence does not define a metric on V. Now given any $Y \subset V$ and $x \in V$, we define the distance from x to Y as $d_{G,R,v}(x, Y) = \min_{y \in Y}\{d_{G,R,v}(x, y)\}$. The set of **undiscovered** attackers $U \subset D$ is taken as $\{u \in U \mid d_{G,R,v}(u, S) = \infty\}$. We now define our performance measure: The **undiscovered attacker rate** $M1 = |U|/|D|$.

V. System Design

We give an informal description of the architecture. Each agent lies on router link and (i) can passively listen to ingress/egress traffic on a switch port, sampling packet headers and aggregating statistics based on destination IP address; (ii) can inject new ICMP control traffic into the stream, and listens to all ICMP packets (header and payload) regardless of their destination. Whenever traffic volume to a destination IP triggers an alarm function (e.g. exceeds a system threshold) the agent creates a Alert to Downstrean (AD) message and sends it towards the victim. The purpose of this message is two-fold: (i) it informs downstream agents about a hypothesized attack on the victim, and (ii) it initiates the formation of a logical link in an agent overlay network specifically instantiated in response to the attack. The AD message is implemented as the payload of ICMP reply packet, whose destination address is the victims IP address; it is sent by starting with a TTL of 1, and incrementing the TTL gradually until an acknowledgment is received from the next downstream agent in the direction of the victim. Whenever an agent sees an AD message in an ICMP reply packet, it replies with an acknowledgment, revealing itself to be the next downstream agent in the direction of the victim, and causing the upstream agent to terminate its TTL-increasing search process. The two agents can then share information concerning the attack on the victim over this newly formed logical link. If we view each logical link as a directed edge from upstream agents to downstream agents, the resulting logical network yields a distributed representation of a tree that is a maximal solution of the flow reconstruction problem corresponding to the scenario at hand. Information about the structure of this overlay network can be queried in real time by sending a broadcast message in the overlay network. Agents store their incident logical adjacencies in a distributed database that can be queried to determine the structure and dynamics of DDoS attacks.

We illustrate a DDoS attack where v is the victim, clouds numbered 1, 2, 3, 4, 5, 6, 7, 8 represent ASs, circles represent routers, happy faces represent agents, straightlines represent inter-AS link between autonomous systems, and arrows represent the attack flow. Figure 1 shows the recon-

structed attack flow. The path to the attacker in AS-8 is fully reconstructed in full detail. Although the attack traffic from AS-3 comes through non participating AS-2, the path to the attacker is successfully reconstructed. The attacker from AS-7 can be traced only up to AS-5. Although some attackers are undiscovered, the system enables traceback to the extent possible with agent deployment. The reconstructed flows provide the victim with actionable information about both attack structure and dynamics.

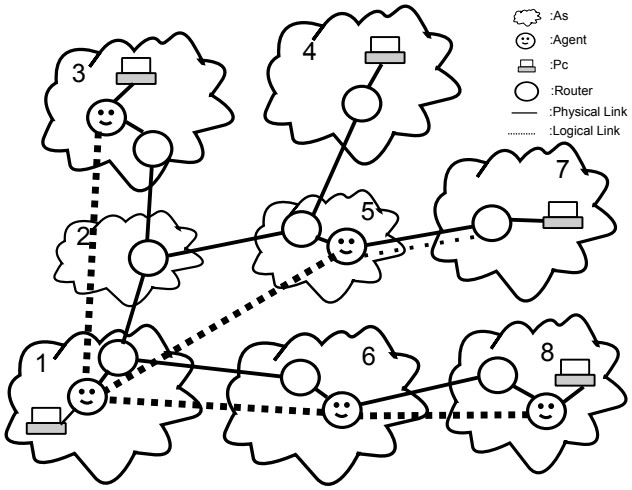

Figure 1. Sample Flow Reconstruction

VI. Agent Design

Here we present the protocols operating within each agent in our architecture, which we present as a Mealy machine [10] by describing transitions between states as a pair: event/action. Events in our agent protocol may be generated locally or may occur due to the receipt of network messages. Local events include Above High (AH), Below High (BH), and timer expiration ($Timeout$). Message related events occur when an agent sees an $ICMP$ echo reply message that was sent by another agent. All of our control messages are carried inside the ICMP reply; these include the Alert to Downstream (AD), No-Alert to Downstream (ND), and Alert to Downstream ACK (ADA). In the protocol these messages appear at times as events (when they are received) and at times actions (when they are to be sent).

Each agent counts the number of packets $c_v(t)$ it sees destined to each target v in a time window in the interval $(t-W,t]$. With periodicity W it updates a sliding window estimate of the traffic to v, as $X_v(iW) = C_v(iW)+X_v((i-1)W)\cdot r$; here r is called the **statistical history coefficient** and reflects the extent to which traffic history lingers in the assessment of traffic rates. In our initial system, an $AH(v)$ event is generated locally by the agent whenever $X_v(t)$ exceeds some fixed threshold T. Similarly, a $BH(v)$ event is generated when the traffic to v goes below threshold T.

Whenever an agent A gets an $AH(v)$ event it starts the process of self-organization by searching for other agents downstream towards v. The agent uses AD messages to do this, sending an AD message using connectionless transport, e.g. as the payload of an $ICMP$ reply packet. Initially A creates an AD message with $TTL = 1, Source = A, Destination = v$, sends this and sets a timer to wait for a response. If there is no response before the timer expires, a local $Timeout$ event is generated, causing A to try again with a higher TTL. This process continues until either a downstream agent B closer to v responds to A with an ADA, or the TTL value reaches 255. In the latter case, A knows that it is a **proxy to the victim**. Analogously, whenever the agent A gets an $BH(v)$ event it must tell its downstream agent B that it now no longer believes there is an attack on v. The agent A does this by sending an ND message concerning v to its downstream neighbor B.

Agents listen to traffic for $ICMP$ reply messages carrying AD or ND messages as their payload *regardless of the ICMP message's destination*. If an agent B sees an AD message from A to v it opens a network connection back to A (the source) using a connection oriented transport protocol (e.g. TCP)and replies with an ADA message. The ADA message from B is an acknowledgment message which tells the agent A sending AD message that there is an agent further downstream on the attack path to v. In this case, we say that (in the attack on v) A is a **parent** of B. Note that in the case of DDoS, an agent will frequently have more than one parent. Figure 2 shows the finite state machine (FSM) of the protocol. An agent with no parents is a **proxy to the attacker**.

VII. System Stability

We now define what we mean by stability of the aforementioned system. Let $G = (V, E)$ be a network where V is the set of network nodes, and E is the set of bidirectional network links between nodes. The **traffic history** is defined as a continuous function $c : V \times R^+ \rightarrow R^+$, where $c(v,t)$ is the traffic seen by v in time interval $[t, t+W)$. The traffic is **eventually stable at v** if $\exists t_0 \in R^+$ for which $\forall t > t_0, c(v,t) = c(v,t_0)$. If traffic is eventually stable at v, we can define **traffic convergence time at v**, $t_c(v) = min\{t_0 | \forall t > t_0, c(v,t) = c(v,t_0)\}$. The **stable traffic at v** is taken to be $\overline{c}(v) = \overline{c}(v) = c(v, t_c((v))$.

Consider a set of agents $A \subset V$ distributed in the network $G = (V, E)$ each of which is running a FSM having a set of states S. We codify the state of the agents at any point in time as follows:

The **state history** of a set of nodes A, each running an instance of a protocol with states S defines as a function: $\sigma = V \times R^+ \rightarrow S$. A vertex ($v$) at time t is in state $\sigma(v,t)$. We say that **state is eventually stable at** v if $\exists t_0 \in R^+$ such that $\forall t > t_0$, $\sigma(v,t) = \sigma(v,t_o)$. If state is eventually stable at v, we define: the state convergence time at v as

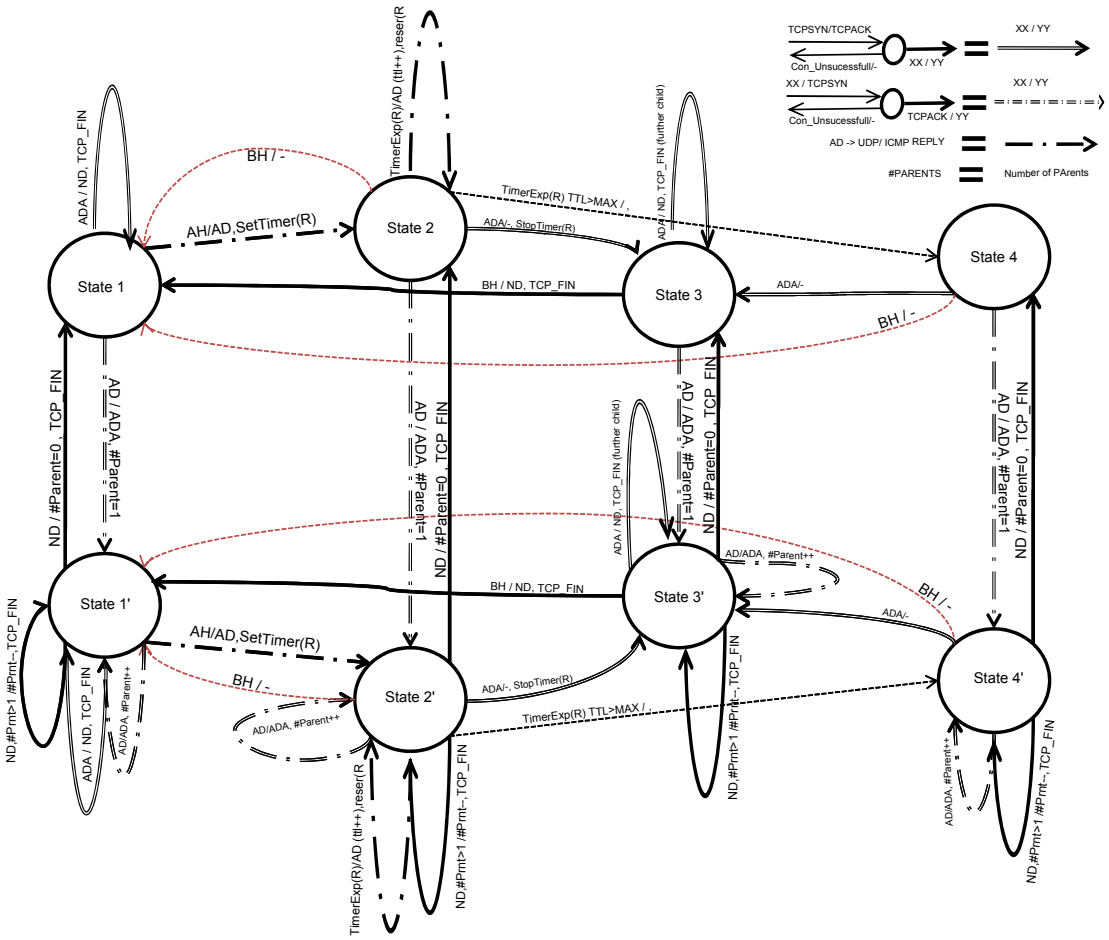

Figure 2. Agent Finite State Machine (one instance per victim)

$t_\sigma(v)$, and define it to be $t_\sigma(v) = min\{t_0 | \forall t > t_0, \sigma(v,t) = \sigma(v,t_0)\}$. The **stable state at v** is denoted $\overline{\sigma}(v)$ and is defined as $\overline{\sigma}(v) = \sigma(v, t_\sigma((v))$.

Given an instance of the attack flow reconstruction problem (G, R, D, v, A), where each of the agents is running a protocol, we define the **convergence time** (CT) to be the maximum difference between the traffic convergence time and state convergence time. Formally: $conv(P, G, R, D, v, A) = \max_{a \in A}\{t_\sigma(a) - t_c(v)\}$. A protocol is **stable** if $conv(P, G, R, D, v, A)$ is always finite.

VIII. Simulation Design

At the heart of any centralized discrete event simulator is a "Scheduler", which acts as the centralized focal point for the passage of time. The scheduler contains a linearly order set of events scheduled for the future, and is responsible for executing them sequentially and chronologically. An any given point in time, the event at the head of this list is being delivered "now". Events are defined to be directed interactions between two entities at a particular time. The entities in our simulation are represented in our code by a base abstract Java class called SimEnt. A SimEnt represents a unit of logic which responds to the arrival of Events. The SimEnts response can involve: (i) sending new Events to other SimEnts, (ii) creating new SimEnts, (iii) destroying existing SimEnts. The influence transmitted between one (source SimEnt) and another (target SimEnt) is embodied in Java classes implementing the Event interface. We note that this one-to-one directed model does not cover all types of influence. In particular, one-to-many and many-to-many influences are not covered directly. However, these more complex forms of interaction can be simulated using higher-level constructs built over the elementary one-to-one constructs. The timed delivery of Events between SimEnts is mediated by the Scheduler, and drives the discrete forward passage of time.

IX. Experimental Setup

For each experiment, we created a connected network by randomly placing nodes in a 2D plane and randomly selecting links between nodes with respect their Waxman probabilities (where the likelihood of having a link between

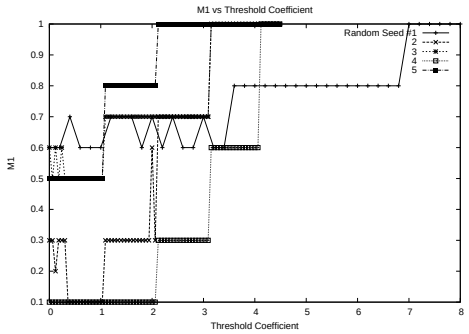

Figure 3. Performance M1 as traffic exceeds thresholds

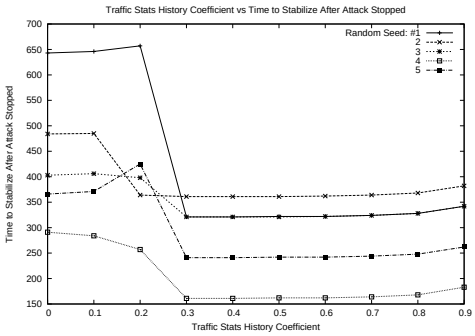

Figure 4. Hysteresis in detecting attack stop

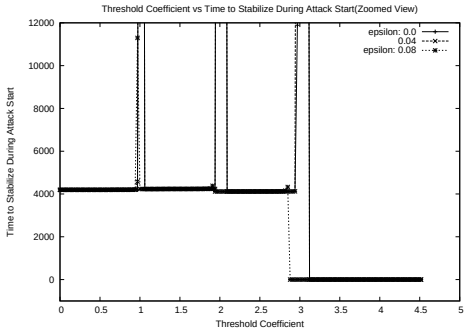

Figure 5. System convergence time at attack start

two nodes is inversely proportional to their Euclidean distance) and built the routing table for the network using the Bellman-Ford algorithm. Afterwards, we create a problem to solve by selecting a victim, attackers and agents randomly where victim and attackers are members of routers and agents are members of physical links. We implemented routers and agents as SimEnts and network packets as Events. We also created a report entity which serves as information collector.

We use HTTP traffic models for the attack traffic in our experiment, because attackers would likely attempt to hide a DDoS attack through mimicry of legitimate traffic to thward IDS sensors [11]. Towards this end, we followed Sun et.al [12] and modeled traffic using constant size attack packets with inverse Gaussian distribution (μ, λ) inter-arrival times[1]. We set $T \propto W/\mu(1-r)$ since this is the expected value of $X_v(t)$ during an attack. The constant of proportionality we denoted as ATC as the **attack threshold coefficient**. An ATC of 1 indicates settings in which the attack threshold T is exactly the expected value of X_v during an attack.

Once simulation starts, the agents start reporting their state changes to the experiment report entity. This report entity is checked regularly for any changes during the experiment. If there is no change for significant amount of time, we decide that the agent system became stable and we than use this to calculate the convergence time, CT.

Through our experiments we seek the relationship between $M1$ and ATC, CTs and ATC, and CTs and r. In order to do the experiment for CT versus r, we run the experiment by keeping everything fixed except the value of r; for the rest of the experiments everything except the ATC values were fixed. We repeated the experiments for different seeds which results in different set of victim, attackers and agents.

A. Results and Analysis of Performance

Figure 3 shows the relationship between $M1$ and ATC. As the value of ATC increases curves show stepping increase showing different patterns. An increase in the value of $M1$ means the number of undiscovered attackers has increased. This is a normal behavior because depending on their locations, the agents may experience traffic from different number of attackers and this may result in seeing different amount of attack traffic . Consequently, T will be greater than the attack traffic sent from single attacker and any agent seeing attack traffic from single attacker will not detect the attack anymore because the attack traffic it sees will be less than T. If there is no more agent down on the path of the attack, the corresponding attacker will not be discovered, so $M1$ will increase. Similar behavior is repeated as T passes the integer multiples of estimated attack traffic (i.e. as ATC crosses integer boundaries). At these points, the value of T is approximately equal to a multiples of estimated attack traffic to v. The results show that threshold selection has a direct effect on $M1$.

We also explored the relationship between CT and r. The figure 4 show that as the statistical history coefficient increases to the value of 0.3 the CT decreases to a stable value. The smaller the value of r the more we are affected by the traffic fluctuations.

B. Results and Analysis of Stability

Figure 5 shows the relationship between convergence time CT and ATC. Initially we will focus on just the curve labelled $epsilon = 0$, which represents the system as described; the other curves are refinements to the system

[1]Here μ represents the mean frequency of sending attack packets and λ represents the shape parameter.

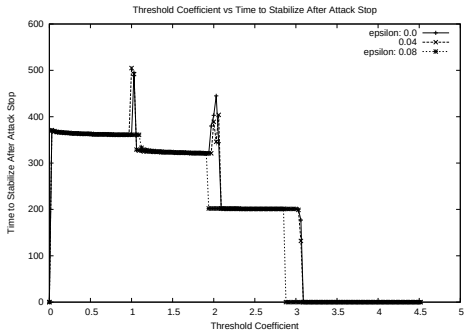

Figure 6. System convergence time at attack stop

and will be presented later.

The curve shows that the agent system gets stabilized in constant time,but that for certain values of ATC near integer values, it never stabilizes. For ATC sufficiently large, convergence is instantaneous since no attack is detected at such high thresholds.

Although it is expected that there is a decrease as the value of T increases, the timers within the agent FSMs force them to wait for certain amount of time to decide whether they are a parent (or not) before they change state to their final converged state. This results in not seeing a noticeable decrease in the value of CT during the attack.

Similarly, in Figure 6 we see the relationship between the CT and ATC, once the attack ceases. The $epsilon$=0 curve shows a stepping decrease as the value of T increases. This is because the number of agents involved in the attack decreases as the value of T increases, as describe above. The difference here is that there is no wait time; as soon as the traffic goes below T the agent changes it state.

In Figure 5 and Figure 6 the curves show peaks at certain points. The peaks goes to infinity for the first case and a finite high value in the latter setting. These peaks occur at integral values because the expected traffic and the value of the threshold are very close to each other and since we use a random process to model interpacket times the value of $X_v(t)$ fluctuates in the neighborhood of T, cauing AH and BH events and preventing state convergence.

Since this kind of behavior is undesired, we modified our system so it generates AH only when traffic $X_v(t)$ exceeds $T = ATC(W/\mu)(1 + epsilon)$ and generates BH only when $X_v(t)$ falls below $T = ATC(W/\mu)(1 - epsilon)$. After modification of the system we repeated the experiments for $epsilon$ values of 0.04 and 0.08.

In Figure 5 and Figure 6 we see that the spikes decreased as we used greater $epsilon$. This shows that the system can be made stable as a whole, over the entire parameter space by a minor modification of the local attack detection logic within its constituent agents.

X. Conclusion

We report on the use of simulation to characterize the performance and stability of a novel distributed system for DDoS flow reconstruction. Preliminary simulation results showed that in certain settings the system state was unstable. We were able to locate the parameter ranges with the help of simulations, and determine the cause of instability, based on which we modified our system design. Finally, using simulation once again, we verified that the augmented protocols exhibit both good performance and systemic stability.

Acknowledgments

The first author thanks the Turkish National Police for supporting his graduate studies at City University of New York, of which this research is a part.

References

[1] "Arbor networks worldwide infrastructure security report," 2008. [Online]. Available: http://www.arbornetworks.com

[2] O. Demir, "A survey of network denial of service attacks and countermeasures," City University of New York, Computer Science Department, Tech. Rep., 2009.

[3] Burch and Hal, "Tracing anonymous packets to their approximate source," in *LISA '00: Proceedings of the 14th USENIX conference on System administration*. Berkeley, CA, USA: USENIX Association, 2000, pp. 319–328.

[4] B. Gemberling, C. Morrow, and B. Greene, "ISP security-real world techniques. presentation, nanog." NANOG, 2001. [Online]. Available: www.nanog.org

[5] Bellovin, "ICMP traceback messages," RFC draft, September 2000. [Online]. Available: 'http://tools.ietf.org/draft/draft-bellovin-itrace/draft-bellovin-itrace-00.txt

[6] S. Savage, D. Wetherall, A. Karlin, and T. Anderson, "Practical network support for IP traceback," *SIGCOMM Comput. Commun. Rev.*, vol. 30, no. 4, pp. 295–306, 2000.

[7] Snoeren and A. C., "Hash-based IP traceback," in *SIGCOMM '01: Proceedings of the 2001 conference on Applications, technologies, architectures, and protocols for computer communications*. New York, NY, USA: ACM, 2001, pp. 3–14.

[8] Bellovin, "Cert advisory ca-1996-26," Cert Advisory, 1996. [Online]. Available: 'http://www.cert.org/advisories/CA-1996-26.html

[9] B. H. Bloom, "Space/time trade-offs in hash coding with allowable errors," *Commun. ACM*, vol. 13, no. 7, pp. 422–426, 1970.

[10] G. H. Mealy, "A method to synthesizing sequential circuits." *Bell Systems Technical Journal*, pp. 1045–1079, 1955.

[11] D. Wagner and D. Dean, "Intrusion detection via static analysis," 2001.

[12] Z. Sun, D. He, L. Liang, and H. Cruickshank, "Internet qos and traffic modelling," *IEE Proceedings - Software*, vol. 151, no. 5, pp. 248–255, 2004. [Online]. Available: http://link.aip.org/link/?IPS/151/248/1