An Intelligent Fast-Flux Swarm Network for Minimizing DDoS attacks in Cloud Computing

R. Kesavamoorthy  
Asst. Prof.  
Kalaivani College of Technology  
Coimbatore  
kesavamoorthycse@gmail.com

Dr.K.Ruba Soundar  
Prof. & Head / CSE  
P.S.R. Engineering College  
Sivakasi  
rubasoundar@yahoo.co.in

Abstract

One of the great threat to service availability in cloud computing is Distributed Denial of Service. Here a novel approach has been proposed to minimize DDoS attacks. This has been planned to achieve by an intelligent fast-flux swarm network. An intelligent swarm network is required to ensure autonomous coordination and allocation of swarm nodes to perform its relaying operations. Intelligent Water Drop algorithm has been adapted for distributed and parallel optimization. The fast-flux technique was used to maintain connectivity between swarm nodes, clients, and servers. We have planned to simulate this as software consisting of various client nodes and swarm nodes.

Distributed denial of service (DDoS) causes performance of a website or cloud service to be severely degraded. Using DDoS attacks, attackers can control the freedom of information by making certain information available at certain times to determine what information is and is not relayed to public. According to a recent survey, DDoS attacks are viewed as the number one security threat. Combating DDoS also requires the most resources. The proliferation of DDoS applications such as Low Orbit Ion Cannon (LOIC) [1] by 4Chan-affiliated hackers allows anyone to coordinate and launch DDoS attacks on any target. Recently pro-Wikileaks activists used LOIC to great effect.

Our article proposes using a swarm network to extend the serviceability of cloud services under DDoS attacks. In the event of an incoming DDoS attack on a cloud service, the network will re-organize itself to provide the highest available service level for the clients and servers under adverse network conditions.

Many solutions have been proposed [2–9]; however, they have met with limited success. We make use of a swarm network to coordinate and relay messages between clients and servers. This swarm system has the ability to self-organize and conduct large-scale parallel search for optimal relay solutions. We have also explored how swarm intelligence can be incorporated into the existing Internet infrastructure. Using established communication protocols, we have reduced the need for modifications of existing networks for easy adoption. This approach presents the first attempt at creating non-stationary swarm transport for cloud services.

RELATED WORKS

Malicious Traffic Detection and Rate Limiting

Pushback by Mahajan et al. [5] uses two techniques: aggregate congestion control (ACC) and pushback. Local ACC detects congestion at the router level and devises an attack/congestion signature that can be translated into a router filter. The signature defines a high bandwidth aggregate, a subset of network traffic, and local ACC determines an appropriate rate limit for this aggregate. Pushback propagates this rate limit for the aggregate to the intermediate upstream neighbors that contribute the largest amount of the traffic. This mechanism works best against
flooding-based attacks as they are treated as congestion phenomena.

D-WARD by Mirkovic et al. [6] aims to detect attacks as they leave the network. It gathers two-way traffic statistics from the outer border at the source network and compares them to network traffic models, reflecting normal/legitimate, transient (suspicious) and attack behavior. Based on these models, D-WARD applies rate limits at the router on all the outgoing routers, slowing down the attack connections. DWARD has the ability to detect attacks; however, it stops attacks at source networks, and thus requires widespread deployment to achieve the desired effectiveness.

Overlay Networks and Indirection Infrastructures

Keromytis et al. proposed SOS [4] with the goal of routing only good traffic to servers. Only traffic from good users/clients is allowed. Clients make use of an overlay network to reach the servers. The built-in redundancy of SOS, as well as the secrecy of how packets are forwarded in the network, contributes to its resistance against DDoS attacks. However, this is a purely target-side solution, which can still be potentially overwhelmed by a bandwidth attack at its access points.

Stavrou et al. [8] suggest a spread-spectrum-like communication model. They proposed spreading their packets randomly across all access points. Alongside these packets, a token is used for authentication. The system intends to prevent the attacker from attacking specific overlay nodes by eavesdropping on clients.

Stoica et al. [9] propose a multipurpose Internet Indirection Infrastructure (I3) by providing rendezvous-based communication. Stoica et al. pairs client's data with an identifier. The servers make use of a trigger to indicate their interest in packets. This infrastructure does not require the sender to be aware of the receiver and vice versa.

Dixon proposes Phalanx [2] consisting of three main components. Phalanx relies on utilizing a swarm that can match an attacking botnet in strength. It makes use of simple packet mailboxes to allow users to store and pick up messages. The servers in the network must explicitly request a packet from the mailbox for it to be delivered. Lastly, Phalanx uses authentication mechanisms to ensure that data stored in its network is legitimate. However, Phalanx’s message interchange mechanism requires extensive modification of existing services.

Extensions to Server Applications

Oikonomou et al. [7] model human behavior as a means to differentiate between human and bots. They suggest modeling request dynamics of legitimate users. They use a particular server’s history to train a detection model. They predominately use the timings of requests as input. They also make use of the links accessed to determine if the client is a human or a bot. They model popularity of the links as a request graph. This graph is then matched to the access patterns for the system to discriminate bots. They have also proposed embedding invisible objects with hyperlinks. These objects will only attract the attention of bots, and this is then used to flag the malicious clients.

Kandula et al. [3] propose a kernel extension that provides multilayer authentication and admission control. Kill-Bots provides authentication through graphical tests. When bots ignore or are unable to solve the test, they are blacklisted.

Approach and Strategy

An effective solution to extend serviceability of cloud services under DDoS attacks is to implement a decentralized swarm transport system. We can relay messages between client and servers through the use of the swarm transport system.
The swarm network is built on two underlying concepts:

- Fast-flux technique in domain name servers
- Organization of swarm Fast-flux hosting utilizes a great number of nodes to provide high availability to a certain resource. A fast-flux service does not provide an attacker with a single IP address as a target. The objective of any DDoS attack is to focus malicious traffic from multiple sources to a bottleneck in a victim’s network. This bottleneck is often the targeted server’s link to the Internet. In a fast-flux network, there is no obvious bottleneck, thus making it harder for an attacker to bring down the entire service network.

Fast-flux hosting allows a fully qualified domain name to have many IP addresses assigned to it. It uses a very short time-to-live (TTL) parameter for any particular name record. The hostnames will be reassigned at very high frequency.

A load-balancing scheme is usually in place to distribute the load to faster responding hosts. It will also check the health of its nodes, and removes those that are unresponsive. Using fast-flux, websites can maintain extremely high availability with many unreliable hosts. Figure 1 shows how fast-flux operate.

However, a fast-flux hosting technique in itself is not robust enough to cope with sophisticated DDoS attacks that exploit fast-flux service networks. Attacker can simply instruct their DDoS attacks to overload the known hosts of the network. This is because fast-flux networks provide a different but static route to the designated host. Even though connections are routed differently at any time, there is no means for it to reorganize as the network environment changes. Hence, by sniffing a fast-flux network, an attacker will still be able to devise means to cripple the network by targeting bottlenecks.

We apply techniques associated with swarm intelligence to imbue the network with autonomy to address the above fallacy. The self-organizing capability of the swarm network provides simple directives for it to optimize its own network. This allows it to always provide the most efficient relaying of messages, even when part of the network becomes unserviceable under an attack.

The movement of bodies of water in nature inspires the Intelligent Water Drop (IWD) algorithm [10]. Water always finds the path of least resistance. This is well suited to designing a relay system for the swarm network. The IWD algorithm has two main parameters: velocity and soil. These two parameters vary during the lifetime of the IWD algorithm. As the IWD moves from the source to the destination, it affects the soil it passes through and may gain or lose speed. As the IWD moves from one place to another, its velocity increases at a rate that is nonlinearly inversely proportional to the amount of soil between the two points. Hence, IWD moving through less soil will gather more speed. The IWD also carries soil along with it, and the amount is determined by the time required for the IWD to move between locations. The movement of soil between the nodes forms the memory of the network. The IWD will tend toward choosing paths of least resistance (or soil). Hence, the IWD algorithm can be seen as an optimization solution.

Figure 1. Operation of a fast-flux service network.
IWD can perform partial optimization depending on the parameters it can sense. It is also highly resistant to sudden network changes due to its distributed nature. To manage the large number of swarm nodes, a hierarchical community approach is used.

Most DDoS mitigation approaches require extensive modification of the network it wishes to protect. This results in an economical barrier to practical implementation. Our swarm acts as a transparent transport layer. It allows transmission of common well established protocols (e.g., HTTP, SMTP) through its network. No modification on the server or client side is required. Servers who wants to benefit from the swarm network only have to register their IP addresses and domain names with the swarm network. The swarm’s name servers will route any request to the specified domain name to the server. Any response from the server will also be relayed without modification to the client. Applying our approach to existing services will not be discernable to the users (both servers and clients), and is hence easily adoptable by cloud services.

IMPLEMENTATION

[Diagram showing the operation of the relay mechanism]

Relay Mechanism

Conventional communications today forward requests and responses through a stationary network. Using fast-flux techniques we are able to develop a non-stationary and variable network infrastructure.

In conventional communications such as traditional server/client architecture, requests from the client are forwarded along the routers towards the server. The proposed mechanism adds a layer to the flow of communications. All incoming requests are registered at the swarm, and the swarm coordinates and forwards the requests to the designated server. The server’s response will be sent back to the swarm, which will forward it to the requesting client through the use of session binding. This process is illustrated in Fig. 2. The hostnames will be reassigned at very high frequency.

The flux-capable domain name server returns a series of IP addresses of its swarm participants. The IWD algorithm determines the nodes chosen to relay messages between the client and server. The IWD treats the user and server as the start and end nodes and vice versa. The name server only needs to return the entry address of the relay node.
node of any particular community. The community of nodes would have individually contained local solutions to the relay problem. The flux-capable domain name server also acts as a leader node that remains in constant contact with the various swarm communities.

**Network Building**

Fast fluxing provides a convenient strategy for swarm building. As a node becomes active, it performs a query on its designated domain name. If no nodes are found, the node assumes a leadership or seed role. If a node is able to contact another node when resolving the designated name, it will negotiate and subscribe to that particular community. Through performing latency checks on neighboring nodes, each node will determine if it is in its optimal community. The optimal community is determined from the collection of nodes that provides the highest throughput in its current configuration. Multiple communities are formed when a community has reached an arbitrary maximum cap. When this cap has been reached, the community is divided in two; and from each community, the node with the highest network performance will be promoted to the leadership role. Communities trade network profile information through their leader nodes. This information includes the number of nodes and its addresses. Leader nodes are then responsible for managing and disseminating this information to its followers. This process is illustrated in Fig. 3.

Crossovers are encouraged to ensure that each individual community remains vibrant. Peer information will be disseminated to each node from the leader node. Latency tests will then be conducted between the nodes to determine suitable breakaways. By allowing a group of nodes to cross over, we can discover better solutions. This process is inspired by genetic algorithms, whereby the selection to break from the community is based on a fitness function.

To determine the network terrain of the swarm, we assume the network as a graph. By performing latency test between nodes, the nodes will form their own graph of their surroundings. Periodically, each node will share its own knowledge with other nodes to allow this information to fuse together to form a wider view of the network.

This network terrain graph is used as the basis for calculating the soil parameter of each particular node.

**Swarm Directive**

The swarm is made up of similar nodes distributed in the network. The nodes communicate with one another, performing decentralized coordination between them.

The swarm makes use of the IWD methodology [10] to determine the fastest and most efficient route to relay messages between the client and servers. The speed and latency of the nodes are used as
The applied IWD algorithm can be described as follows:

1. Initialization of the parameters. The graph of the local community is initialized in each node. The quality of the global best solution is set to an arbitrary large negative number. The number of water drops is set to the number of nodes in the community. Every node has a visited node list \( Vc(IWD) \), which is initially empty. The velocity of the IWD is set to an initial value.

2. As all the nodes are distributed into various communities, each will hold on to a partial solution to the global optimization problem. At each node \( i \), it will choose the next node \( j \), within the constraints of the problem and not in the visited node list with the following probability model.

3. For each IWD moving from one node to another, the velocity is updated.

   \[
   \text{velocity}_{IWD}^{(t+1)} = \text{velocity}_{IWD}^{(t)} + \frac{a_x \cdot y}{b_y + c_y \cdot \text{soil}^2(i,j)}
   \]

   \( a_x \), \( b_y \), and \( c_y \) are parameters that determine the viscosity of the water droplet.

4. For every IWD that moves from one node to another, the amount of soil change is correlated with the velocity of the moving water drop. This correlation can be adjusted to generate alternate solutions for optimizations.

5. The soil parameter is then updated accordingly.

6. The optimal path is computed using a quality function. The global best solution is retrieved from the pool of solutions.

7. The soil on the path of the optimal solution is then updated.

8. This algorithm is repeated as the system operates and up-to-date optimal paths will be generated.

The convergence property of the IWD was shown in [10]. In the above, the IWD algorithm uses a swarm of water drops to conduct a large parallel search for the optimal solutions.

**Simulation and Results**

The simulation consists the swarm nodes, the legitimate clients, servers and the attackers. A script was used to create up to 25,000 concurrent agents in the computing cluster. The client agents mimics human activity and are adapted from “curl-loader.” We then simulate 400,000 clients connecting to the swarm, transmitting approximately 200 Mbytes of requests to the servers through the swarm. To understand the robustness of the swarm network, we simulate 10,000 attacking nodes to generate peak attack traffic of 45.77 Mb/s. A total of 5.59 Gbytes of malicious packets were sent to the network.

To demonstrate how our proposed swarm solution can be used to improve network robustness, a simple client-server framework and two implementations of fast-flux hosting are created. In the simple clientserver framework, as the attack packets arrive at the server, the latency of the network increases dramatically and soon becomes unserviceable.

**Swarm Network**

We then apply the same attack pressure as used in our conventional server client framework on our swarm. In Fig. 4, latency below the threshold value indicates
that the responsiveness of the swarm service is not affected. Spikes above the threshold indicate that there is detectable congestion at the network.

Figure 4. Latency performance of swarm under DDoS.

The results illustrate the swarm of 10,000 nodes under attack by a similarly sized botnet. While servicing up to 400,000 clients connecting to the network, the network received a concerted attack of 10,000 nodes. During this simulation, approximately 1,584,000 sessions were created. We consider a connection to be unsuccessful if it requires a retransmission or dropped from the network. This test was repeated over four separate occasions. We have determined that 516–4800 connections were unsatisfactory. This results in a delivery ratio of more than 99.5 percent. Hence, our approach shows that a multi-path transport system such as the swarm network was able to provide a highly robust and reliable service.

Effectiveness of Intelligent Water Drop

To evaluate the performance of IWD as a distributed optimization algorithm, we created two smaller setups and introduced a DDoS attack on both of them. Figure 5 illustrates an implementation without the IWD algorithm. It consists of only the swarm network, and traffic is routed using fastest peer strategy. Figure 6 shows the swarm network with IWD as the routing protocol.

Each spike in Figs. 5 and 6 (of latency measure arbitrarily set to 106 ms) indicates a timeout, and hence an unsuccessful attempt. As shown in Fig. 5, throughout the duration of the attack, the latency timeouts experienced by the clients are high. In this particular scenario, each of the 4000 clients makes 100 connections with the servers through the 10,000 swarm nodes. This results in 400,000 connection attempts. However, a portion of these messages is used to query the FDNS for the route to be taken. Therefore, only 394,250 connections are made directly to the server.

Figure 5. Network without intelligence under DDoS attack.

Out of these connections, only 1883 connections were unsuccessful. This represents a delivery ratio of 99.522 percent.

Figure 6 shows that with the application of the proposed IWD algorithm for self-organization, the number of unsuccessful connections has fallen significantly. 395,871 connections were made, and only 129 connections were unsuccessful. This represents a delivery ratio of 99.967 percent. Hence, we are able to obtain an increase in delivery ratio from 99.522 percent to 99.967 percent through the implementation of selforganizing swarm networks.
Through the above observations, we have shown how IWD is a more effective protocol when applied to a large distributed system such as the swarm network.

**Discussion**

**Comparisons to Existing Work**

The system imbues swarm intelligence into a transport network in a manner similar to an indirection infrastructure. Our system mitigates DDoS by providing a resilient and robust transport service. It does not explicitly detect malicious traffic as do Pushback [5] and D-WARD [6]. Malicious traffic detection and rate limiting requires service providers’ commitment and cooperation. This faces economic barriers as service providers seldom view protecting competitors’ networks as being in their own self-interest. These mechanisms also require routers in the networks to be modified to support such features.

Approaches such as modeling human behavior [7] and killbots [3] extend the capability of existing server applications to mitigate DDoS attack. However, these approaches do not address resource exhaustion attacks on a single target – Server. Optimization and detection of bots reduces the effectiveness of a DDoS attack. However, it lacks the ability to adapt defensive network configurations to reduce the impact of such the attacks on their service level.

Inspired by the growing trend of decentralized peer-to-peer systems, distributed solutions such as overlay networks and indirection infrastructure have great potential as solutions to the DDoS challenge. Secure Overlay Services [4] provides a form of virtual network to facilitate message passing in a safe environment. IONS [9] as an evolution of overlay networks also make use of stationary safe nodes to perform message passing.

Our swarm-based approach represents a novel attempt to create a non-stationary transport network. In comparison, the static nature of existing overlay networks prevents the network from reconfiguring itself when faced with ever changing network terrain. This inflexibility prevents existing solutions from exploiting the potential of large distributed systems. Such overlay networks assume ownership over the overlay nodes. This provisioning of dedicated hardware resources might be cost prohibitive for any single party. The swarm network can be deployed on multiple platforms. They can also be woven into existing Internet applications to further the common good. This is made possible by the robust nature of the swarm network. The fast-flux capabilities do not require any particular host to be serviceable at all times. This reduces the dependency on dedicated hardware for its continued operation.

**Limitations**

The proposed architecture is designed for stateless packet relay through a non-stationary network. The swarm network binds each client to a server though a unique route that exists only for a particular connection. Each message will pass through a unique route that is not guaranteed for the next transmission. The message passing service can be viewed as a one-time-use-only trip. The route...
changes according to the varying network terrain. The client will view the entry node as the server. The server assumes that the exit node of the swarm as the requesting client. This mechanism prevents a direct host-to-host connection. While this provides the location anonymity that increases the security for the server, it does not allow certain connections such as SSL to be possible. Future work will address this concern to allow virtual persistent and secure connection to take place.

It is important to note that there are many factors that contribute to the overhead needed maintain a swarm network. TTL values of typical domain names last for a long period of time. TTLs for fast-flux networks are very short and also prone to disruptions. Depending on the volatility of the network, rapid changes in the swarm composition may result in inactive name servers listed. This problem can be addressed by having a large deployment of name servers to improve robustness. However, increasing the allocation of name servers will result in a drain on the swarm’s resources.

Communication between nodes takes up the swarm’s bandwidth. The nodes frequently contact each other for latency tests and trading of peer information. While these messages are limited to a percentage of available bandwidth, they represent an overhead required to maintain the swarm integrity.

Conclusion

We have presented an effective DDoS mitigation technique consisting of nodes in a swarm network. The client tries to reach the server through a fully qualified domain name. The client is directed to the server and then sends its request to Figure 5. Network without intelligence under DDoS attack. the designated server via the community’s exit node. The server then responses accordingly, and the result is forwarded through the swarm network back to the client. The swarm network is made accessible through fast-flux hosting, hence demonstrating high robustness. The swarm network constantly reconfigures itself through the use of a parallel optimization algorithm such as the Intelligent Water Drop mechanism. All these techniques are used to extend the serviceability of a cloud service under DDoS attack.

To demonstrate the feasibility of such a swarm network, a computer simulation using a high-performance computing cluster was implemented. Up to 400,000 clients are used to interact with the 10,000-node swarm network. Simulated DDoS attacks from 10,000 dedicated attack nodes are conducted against the swarm network. Results have shown that damages sustained by DDoS attacks are greatly reduced with the use of a swarm transport layer.

References


