Can We Beat DDoS Attacks in Clouds?  
(Supplementary Material)

Shui Yu, Senior Member, IEEE, Yonghong Tian, Senior Member, IEEE, Song Guo, Senior Member, IEEE, and Dapeng Oliver Wu, Fellow, IEEE

Keywords—Cloud computing, DDoS attacks, Mitigation, System modelling, Resource Investment.

1 RELATED WORK

DDoS attacks aim to exhaust the resources of victims, such as network bandwidth, computing power and operating system data structures. Early DDoS attacks emerged around the year 2000, and well-known web sites, such as CNN, Amazon and Yahoo, have been the targets of hackers since then. The purpose of early attacks was mainly for fun and curiosity about the technique. However, recently we have witnessed an explosive increase in cyber attacks due to the huge financial or political rewards available to cyber attackers [1].

Botnets are the engines behind major DDoS attacks. Hackers exploit the vulnerability of computers connected to the Internet, and establish an overlay network of compromised computers to commit malicious activities, such as DDoS attacks or information phishing. This kind of malicious network is what we call a botnet [2], [3]. A DDoS attack can be carried out in various forms, such as flooding packets or synchronization attacks [2]. Flooding packets is the most common and effective DDoS attack strategy amongst all the available attack weapons.

It is critical for defenders to understand the size of botnets, which helps us to estimate the possible attack volume. There has been plenty of work completed on this issue, such as [1] and [4]. Rajab et al. [5] found the number of active bots a botmaster could manipulate was usually at the hundreds or a few thousands level. This means the resources a botnet owner can use is limited. Based on this fact, Yu et al. [6] proposed a similarity based DDoS detection method to beat flash crowd mimicking attacks.

Traditionally, a potential victim would be vulnerable if they were left to deal with a DDoS flooding attack by themselves. As nature of the Internet is anarchical, potential victims, such as popular web sites, are usually left on their own to deal with attacks. A firewall is usually a solution to block attack packets. When a DDoS attack is ongoing, we need to not only block attack packets, but also sustain the services to benign users. However, detecting DDoS attack packets draws on a large amount of resources, such as bandwidth and computing power. As a result, a traditional single web site is rarely able to beat DDoS attacks. A passive defence strategy is to increase investment in resources, e.g. bandwidth and computing power, to confront possible DDoS attacks. However, this solution is very expensive as most of the time the extra resources are idle.

For a non-cloud environment, the current DDoS mitigation techniques depend on either sufficiency of resource or collaboration among different organizations. On top of a large number of attack packets, a DDoS attack may be carried out in various forms. As a result, our detection has to go through many different possible detection methods, such as IP spoofing [7], hop-count [8], packet score [9], and flash crowd mimicking [6], [10]. Therefore, DDoS detection is expensive in terms of computing power.

It is a resource management problem in DDoS attack and defence, and the winner is the one who possesses more resources than his opponent. Yau et al. [11] viewed DDoS attacks as a resource management issue in their DDoS defence proposal. They proposed the installation of a router throttle at selected upstream routers of a possible victim. The participant routers regulate the traffic flows to the protected server in a proactive way using a level-k max-min fairness strategy. They target was to constrain the number of attack packets far away from the protected server. If we have sufficient resources, such as bandwidth and computing power, we can then perform a deep packet inspection, and filter out attack packets. However, this is not usually possible for a non-cloud platform. Chen et al. [12] proposed a DDoS attack mitigation scheme, attack diagnosis (AD), using a divide-and-conquer strategy to address the problem. They used the push back strategy to enable an AD to work as close as possible to an individual attack source. As a result, attack sources were isolated, and then throttled. Another method is to establish an ally among multiple network domains to protect a potential victim. In [13], distributed

---

1. S. Yu is with the School of IT, Deakin University, Victoria, 3125, Australia. E-mail: syu@deakin.edu.au.
2. Y. Tian, corresponding author, is with the School of EECS, Peking University, Beijing, China. E-mail: yhtian@pku.edu.cn.
3. S. Guo is with the School of CSE, The University of Aizu, Aizuwakamatsu, Japan. E-mail: sguo@u-aizu.ac.jp.
4. D. Wu is the Department of ECE, University of Florida, Florida, USA. E-mail: wu@ece.ufl.edu.
change-point detection architecture was proposed using a change aggregation tree (CAT). Each CAT works at one network domain, and all CATs report their traffic fluctuation to a server, with the server overlooking all the reports to make a final decision on DDoS attacks. The authors of [14] proposed FireCol, a distributed intrusion prevention system at the ISP level to mitigate DDoS attacks from large botnets. The cooperative ISPs establish virtual protection rings around potential victims to defend and collaborate through the exchange of selected traffic information. However, it is difficult to obtain either sufficient resources or collaboration among multiple network domains in a non-cloud environment at the Internet level.

With the boom in cloud computing, it is certain that individual cloud customers will be victims of DDoS attacks as we have discussed in the Introduction of this paper. A variation of a DDoS attack in cloud computing is the Economic Denial of Sustainability (EDoS) attack [15] or the Fraudulent Resource Consumption (FRC) attack [16]. If the billing mechanism for cloud customers is “pay-as-you-use”, botnet owners can create a large number of fake users to intensively consume the service of the targeted cloud customer. For example, the existing flash crowd mimicking attacks [6], [17] on an e-business web site is an excellent example. As a result, the bill for the targeted cloud customer will increase dramatically until the victim suspends her service or is bankrupted. On the other hand, if a cloud customer fixes her cost for renting the resources of her hosted services, then an effective DDoS attack will disturb, or even shut her services down.

There has been some work on mitigating DDoS attacks in a cloud computing environment. Lua and Yow [18] proposed the establishment of a large swarm network to mitigate DDoS attacks on a cloud, with an intelligent fast-flux technique used to transparently maintain connectivity between nodes in a swarm network, cloud clients and servers. Their software simulation indicated they can maintain a high percentage of benign request delivery rates while successfully blocking attack packets. Chen et al. [19] proposed on-demand security architecture to offer different services for different needs in cloud environments. This includes three factors: risk of network access, service type and security level. Based on the mechanism of cloud computing, this is a good idea as it meets the different requirements of users. In order to deal with EDoS attacks, Squali et al. [15] proposed a white and black list based filtering scheme to block malicious service requests. Amazon developed cloudWatch [20], a tool to monitor the company’s cloud resources and mitigate EDoS attacks on their cloud customers.

Cloud platforms possess the unique feature of cloning virtual machines on the fly, but there is a cost for performing this function. In general, there are two categories for cloning virtual machines in a cloud: the network-driven approach and non-network efforts. For the first group, researchers usually take a BitTorrent-like strategy to treat an image of a virtual machine as one file, and distribute the entire file as demanded [21]. In the second category, researchers try to take advantage of non-network techniques, such as reducing the size of a virtual machine image, prediction and partial page launch to speed up the initialization of virtual machine instances [22]. Peng et al. [23] observed six production cloud data centers for a long period of time, and proposed a chunk-level, network topology-aware virtual machine image distribution network. The proposed method can reduce the cloning time to the one minute level.

As a new business model, it is critically important to understand the economics of a cloud, such as billing models, and resource allocation metrics. Cao et al. [24] proposed a cloud service charge model, which includes several factors, such as the amount and quality of service, the workload of an application environment, the configuration of a service, the cost of renting, the cost of energy consumption, and the service provider’s margin and profit. They established an M/M/m queueing model for their analysis, and targeted maximization the profit for service providers. Wang et al. [25] discussed the optimal resource allocation issue in the customer-provided cloud platform, SpotCloud. He et al. [26] studied the bandwidth-latency tradeoff for job migration in a cloud environment.

2 DDoS Mitigation Algorithm for a Cloud

2.1 DDoS Detection Methods

In this subsection, we only list a few DDoS detection methods that could be implemented in cloud.

A common strategy to disguise attack sources is IP spoofing. In order to fight against source IP spoofing, a hop-count filter is an effective method: it is easy for a server to establish a table of hops for each of its legitimate clients, who are identified by their IP addresses [8].

Packet score is another DDoS detection method at a potential victim side. The idea is to prioritize a packet based on a score that estimates its legitimacy given the attribute values it carries. Once the score of a packet is computed, the scheme performs score-based selective packet discarding where the dropping threshold is dynamically adjusted based on the score distribution of recent incoming packets and the current level of system overload [9].

Sophisticated attackers now mimic legitimate flash crowds to fly under the radar. However, compared with legitimate network flows, DDoS attack flows possess higher similarity [6]. A measure on flow similarity can be used to detect the attack [27].
It demands considerable resources and time for a server to conduct various detection methods when it faces a DDoS attack.

DDoS attacks are carried out in various forms, and there exist many different detection methods in practice, such as detection methods based on IP spoofing [7], hop-count [8], packet score [9], low rate attacks [28], and flash crowd mimicking [6]. As mentioned before, we focus on how to beat DDoS attacks in clouds from resource competition aspects in this paper. We therefore do not involve specific DDoS detection methods. All of the detection methods involved in this paper are represented by the system service rate in our algorithm.

### 2.2 DDoS Mitigation Algorithm in Cloud

The details of the dynamic resource allocation algorithm against DDoS attacks on a cloud customer is presented in Algorithm 1.

**Algorithm 1:** The DDoS Mitigation Algorithm for Cloud

```plaintext
//Initialize the system;
1. Identify the available resource \( R_c \) of the cloud and the resources for one IPS, \( R_{IPS} \);
2. Observe the non-attack scenarios, extract parameter \( \mu \) and \( \lambda \), and obtain \( T_a \) according to equation (11);
3. \( m=1; \)
4. while \{a DDoS attack has been detected\} do
   5. clone one IPS;
   6. \( m = m + 1; \)
   7. \( R_c = R_c - R_{IPS}; \)
   8. while \{true\} do
      Calculate \( T_a(t,m) \) according to equation (18);
      if \( T_a(t,m) > T_a \) then
         if \( R_c \geq 0 \) then
            Clone one IPS;
            \( m = m+1; \)
         else
            Declare mitigation failure;
      end
   end
9. Calculate \( T_a(t,m-1) \) according to equation (18);
   if \( T_a(t,m) < T_a(t,m-1) \) then
      Reduce one IPS;
      \( R_c = R_c + R_{IPS}; \)
   end
```

### References


