Entropy-Score: A Method to Detect DDoS Attack and Flash Crowd

Efi Korenfeld

Saar Ben Yochana

Contents

1. Introduction

1.1 Definitions and abbreviations

Abbreviations:

DDoS – Distributed Denial of Service

FC – Flash Crowd

Definitions:

Entropy – Measurable property that describes the degree of disorder, randomness or uncertainty of the system.

DDoS Attack – Distributed Denial of Service Attack is a cyber-attack in which the attacker can make the server unavailable for serving legitimate clients by flooding requests from multiple sources.

When the attack is activated, all the sources of the attack generate massive traffic towards the server simultaneously, therefore, if the server cannot distinguish between legitimate requests and malicious ones it is impossible to stop this attack simply by blocking its source in contrast to a regular DoS attack.

Flash Crowd – A flash crowd event is a large increase in traffic to a particular common server causing a dramatic growth in server load and service time. Usually occurs when a new launched feature is causing worldwide interest.

1.2 Overview

The article presents a problem that many popular servers are constantly faced with, that is the inability to distinguish between legitimate flash crowd traffic as a result of a new feature being uploaded and massive incoming traffic because of a DDoS attack.

The need to differentiate between the two is necessary because early detection of a DDoS attack can help prevent it. Moreover, the service for flash crowd traffic is often very important, therefore false-positive detection might not be applicable.

To solve this problem, the article proposes a combined method consisting of packet scoring and entropy calculation, with the help of which it will be possible to distinguish between malicious network traffic and mass crowd traffic.

1.3 Goals of the simulation

The main goal of the simulation is to test the correctness and efficiency of the suggested method to differentiate between DDoS Attack traffic and Flash Crowd traffic.

Another goal is to simulate the most general scenario possible and record statistics to compare against the analytical expressions (entropy calculations, packet scoring, group numbers).

1.4 Assumptions

IP addresses – According to Equation (2) in the article:

$$H_{no\ attack} < H_{DDoS\ attack} < H_{Flash\ crowd} \quad (2)$$

Assuming that the randomness of normal traffic is the lowest, each normal client will initialize his IP address by getting 3-octet pattern and extend it by a unique 4th-octet which is chosen randomly.

In the same way DDoS clients achieve higher randomness by generating both 3rd and 4th for octets. Flash Crowd clients achieve most randomness by generating 2nd, 3rd and 4th octets.

Center of group – The way of choosing the center of the group is not specified in the article, hence we decided to calculate it in a dynamic way:

- New Group – If the IP address does not match any existing group, a new group is created and the address will be picked as the center of the group.
- Existing Group – If the IP address matches an existing group, then we will calculate the median of the group and set it as the new center of the group.

Distance – The way of calculating the distance is not specified in the article, so after several tests (using our randomness implementation) we have found that 29 was the optimal distance in terms of the article results.

Entropy Threshold – The value of this parameter was not explicitly declared.
We decided to set the entropy threshold as the upper bound entropy value for the number of normal groups created. Given N normal groups, threshold would be:

$$Threshold = \max \sum_{i=1}^{N} -p(x_i) \cdot log_2(p(x_i)) \underbrace{=}_{Uniform\ Distribution} -N \cdot \frac{1}{N} \cdot log_2\left(\frac{1}{N}\right) = log_2(N)$$

Packet Score Threshold – The value of this parameter was shown in figure 5 of the article, so we assumed it was assigned deterministically, so we have set it accordingly to be 15.

Number of requests – We calculated for each host's type (Normal/DDoS/FC) the average number of requests by the formula: (rounded to the closest integer)

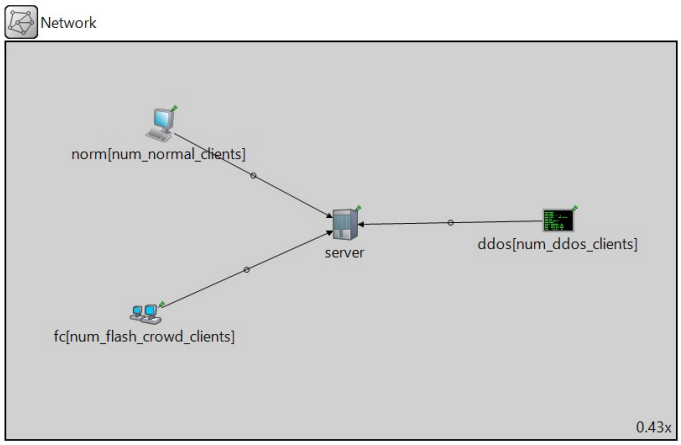
$$\frac{Packets\ Generated}{Number\ of\ Hosts}$$ **

** The values of the parameters were taken from TABLE II. in the article.

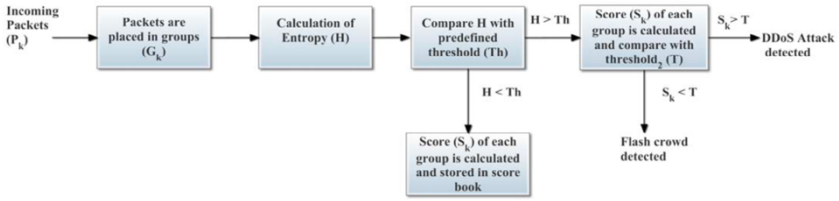
2. Design and Flow

2.1 Overview

To implement and test the algorithm's correctness and efficiency, first a general topology should be built to enable simulating the incoming requests distribution from all types of clients to the server:



The next step is implementing the algorithm itself at the server-side as described in Figure 2. of the article:



Phase 1: Packets will be divided into groups according to 'Distance' parameter and their IP address.

Phase 2: Entropy calculation of the system which is done by Equation 1. In the article:

$$Entropy(H) = -\sum_{X}\{p(x_i)\log_2(p(x_i))\} \quad (1)$$

**Where X is the total sample space and $p(x_i)$ is the probability of i^{th} sample occurrence.

Phase 3: Compare entropy value against the threshold:

- If the entropy is less than the threshold, then traffic is classified as normal.
- Else, further distinguishing between DDoS and FC traffic is required.

Phase 4: In any of the cases in the previous phase, the server calculates the packet score value and store it in the score book (database) only if it is classified as normal traffic.

The scoring method is based on IP addresses scattering among the groups while each group contains two dynamic consecutive probabilities: $p(x_i), p(y_i)$.

$p(x_i)$ is the same as described in phase 2 and calculated by next formula:

$$p(x_i) = \frac{Requests\ of\ Group\ i}{Total\ Requests}$$

$p(y_i)$ is the previous $p(x_i)$, this value is set every time the score is updated.

The score calculation of a group is as described in the article in Equation 3:

$$Score\ (S) = \frac{p(x_i)}{p(y_i)} \qquad (3)$$

The threshold for packet score method is calculated by load shedding algorithm and its value is given by Equation 4. in the article:

$$T = 1 - \frac{\psi}{\phi} \qquad (4)$$

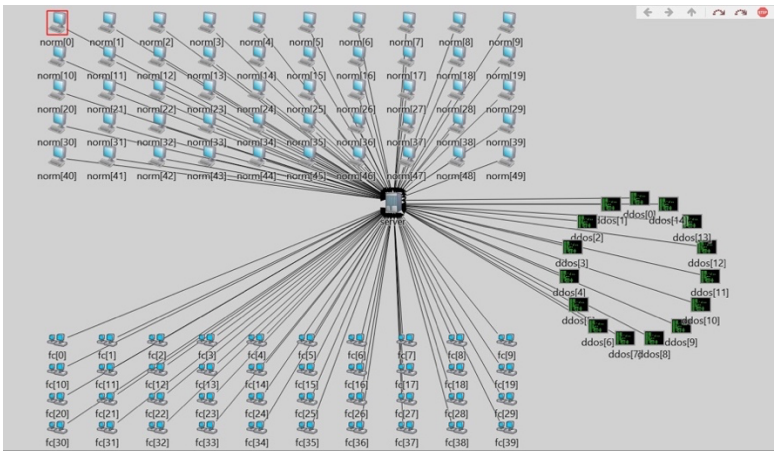
Where ψ is the probability of current traffic and ϕ is the probability of excitable traffic.

While entropy calculation is used to separate between normal traffic and DDoS attack or Flash Crowd scenarios, then, by comparing the score calculation results versus the score threshold, an indication is achieved about which kind of traffic was received.

The previous claim is based on the fact that Flash Crowd traffic is characterized by a relatively low score while DDoS attack traffic is characterized by a sharp increase of the score value.

2.2 Modules overview

The layout of the modules:



Simple Module: Client

Description: An object that generates requests to the server.

Variables: **client_type** – used to define the behavior of the client (Normal/DDoS/FC) and assign his IP address accordingly.

IP_addr – address of the client (assigned dynamically based on template in ini).

Statistics Gathering: **None**

Simple Module: Server

Description: Receives requests from clients, initiates the algorithm to try and distinguish between different client's traffic**.**

Variables: **thresh_entropy**– Threshold for entropy score

thresh_packet_score– Threshold for packet scoring

Statistics Gathering: All calculations and statistics will be recorded on the server side, based on the described algorithm.

Compound Module: Network

Description: The network of the simulation - connects the clients to the server

Variables: **num_normal_clients** – the number of Normal clients.

num_ddos_clients – the number of DDoS attackers.

num_flash_crowd_clients – the number of Flash Crowd clients.

Statistics Gathering: **None**

2.3 Implementation

Initialization: Clients are divided to three different types – Normal, DDoS and FC. Based on their type and our randomness implementation, which was described in the assumptions section, each client generated a unique IP address and scheduled his first request. In order to test the algorithm correctness, we tried to verify that the algorithm's classification would match the client's pre-defined types.

To do so we have generated requests in 3 different phases:

First Phase - Normal Operation:

Requests are generated from all normal clients towards the server. There are total of 50 normal clients, each one of them generating 8 requests resulting in total of 400 requests. Groups were created dynamically as we described at the assumptions section. At the end of this phase, we calculated and recorded the system's entropy value and assigned the entropy threshold based on the number of normal groups created.

Second Phase - DDoS Attack:

In this Phase 15 DDoS clients generate 54 requests each, resulting in a total of 810 requests.
At the end of this phase, we calculated and recorded the system's entropy value and packet score values for each DDoS group that was created.

Third Phase - FC Scenario:

40 FC clients generate 20 requests each, resulting in a total of 800 requests. Each one of the clients causes a formation of a unique group – based on the assumption that their addresses are completely random. At the end of this phase, we calculated and recorded the system's entropy value and packet score values for each FC group that was created.

Classification:

During the simulation we used phase 1 as the learning phase of the system – since the entropy threshold should be based on normal traffic. Therefore, in real life scenario normal traffic should not cross the system's entropy threshold.

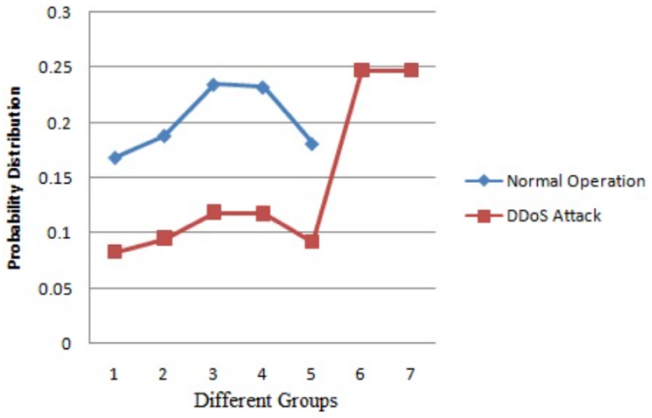
For the next 2 phases, if the system's entropy will be higher than the threshold, then it should be either DDoS attack or FC scenario. We will differentiate between them using the packet score calculated for each group and set the classification tag accordingly.

Then we compare between the classification of each group and the pre-defined type of each client in the group to check the algorithm's correctness.

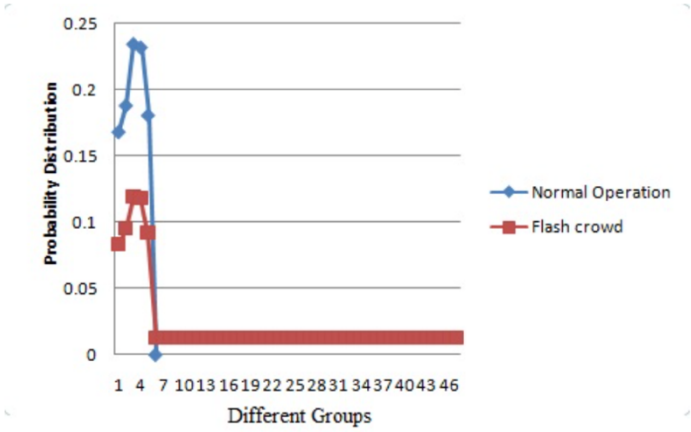
2.4 Results and Test measurements

Probability Distributions:

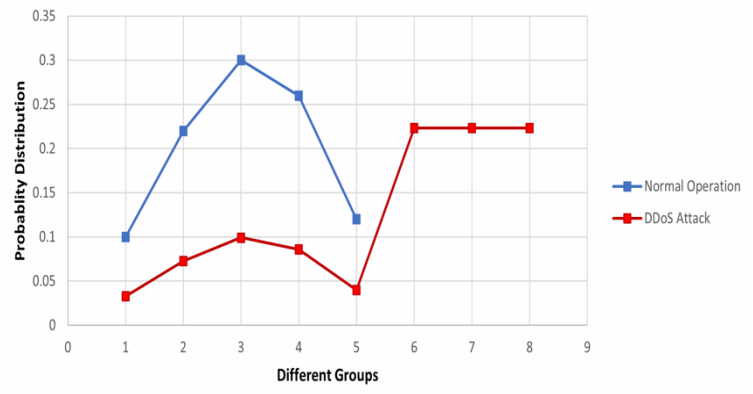
The expected results (taken from the article):

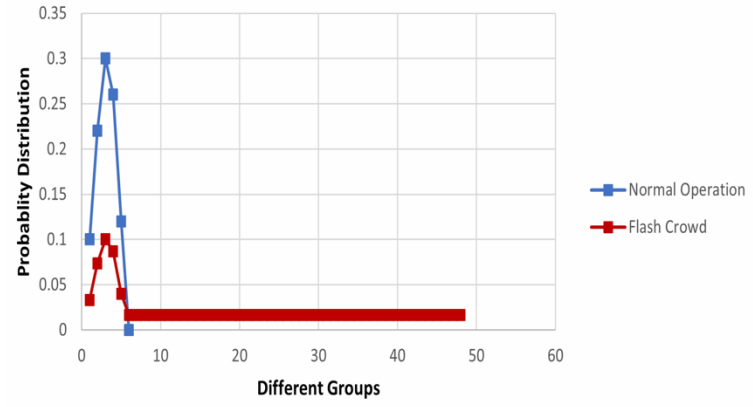


(a)



Our results:



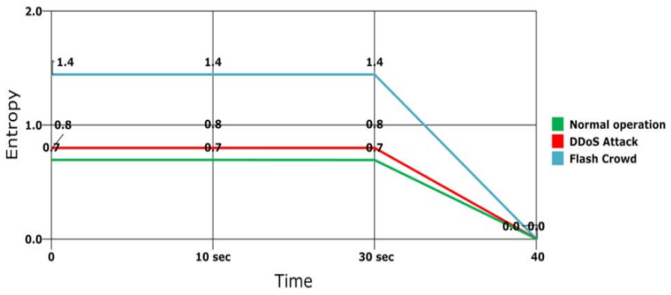


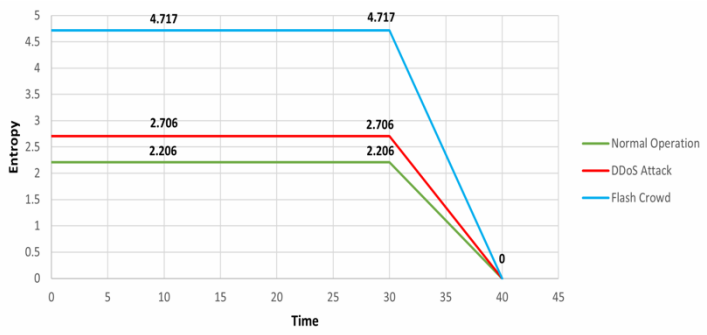
There are slight differences in groups probability distribution, we assume that this is a result of our implementation and the fact that the creation of groups and randomness of IP addresses are not specified in the article. In our simulation, 3 DDoS groups were created instead of 2 in the article results.

We would like to note that one of the article's assumptions (equation 2) is that DDoS attackers have IP addresses that are more random than normal clients, so we assume that 3 uniform distributed groups created are reflecting more randomness and not necessarily contradict the article's results. In addition, the behavior of our graphs matches the article's results.

System Entropy Values:

The expected results (taken from the article): Our results:

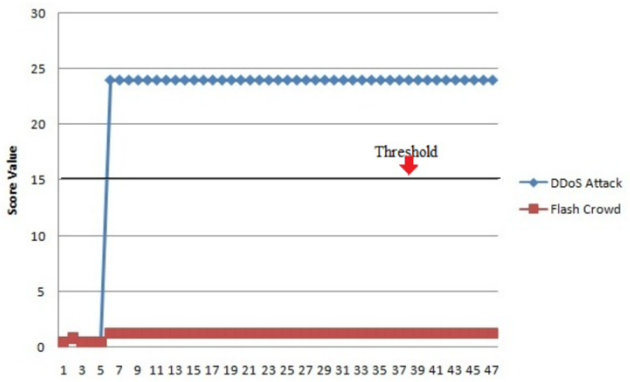


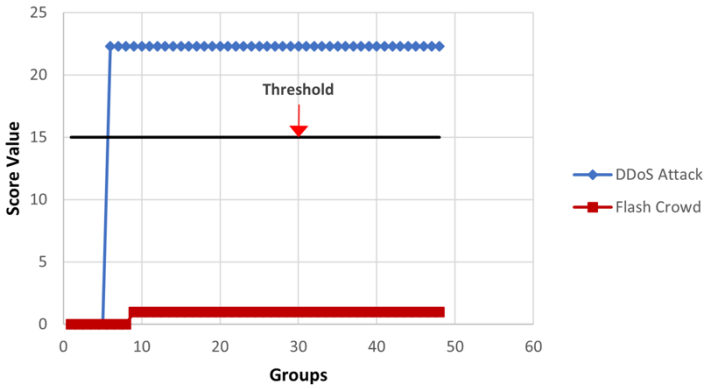


From the graphs it can be shown that the entropy values are not equal.
Since the amount and type of the requests during the 30sec interval was not specified, we decided to measure the entropy values at the end of each phase. In our opinion, this is the cause of the difference in value, although the behavior of the graphs is the same.

Packet Score:

The expected results (taken from the article): Our results:





As we expected, our results fit the article's results. The DDoS groups acquire a high packet score whereas the FC groups scores remains low.

Efficiency:

The article's analysis:

Different Detection Methods	Qualities Considered			
	Detects DDoS attack	Detect Flash Crowd	Space Complexity	False Positive Rate
Packet Score [14]	YES	NO	O(g)	LOW
Entropy Based Methods [16-21]	YES	YES	2O(n)	High
Entropy-score	YES	YES	O(g)	LOW

In our implementation, the classification is done in the end of each phase, by traversing all the groups in the system, resulting in a space complexity of O(g). There were no false positive classifications in our simulation, matching the low false positive rate shown above.

Conclusion:

Since there are missing details about the simulation, it was not possible to re-conduct exactly the same experiment shown in the article.
Even so, by implementing the algorithm concept described and using the given simulation parameters, we achieved results that verify the correctness and efficiency of the algorithm in the ability to distinguish between normal traffics, DDoS attacks and FC scenarios.

In order to implement the algorithm correctly, it is required to know the system's normal behavior and set the right threshold parameter.

In real time scenario the algorithm can detect DDoS attacks in a relatively short time interval and can be used to prevent it without blocking Flash Crowd traffic and harm the popularity of the server and it services.