Journal of King Saud University – Computer and Information Sciences 34 (2022) 1373–1387

Contents lists available at ScienceDirect

Journal of King Saud University – Computer and Information Sciences

journal homepage: www.sciencedirect.com

E-Had: A distributed and collaborative detection framework for early detection of DDoS attacks

Nilesh Vishwasrao Patil [a,1], C. Rama Krishna [b,2], Krishan Kumar [c,3], Sunny Behal [d,*,4]

[a] NITTTR, Chandigarh, India
[b] Department of Computer Science, NITTTR, Chandigarh, India
[c] Department of Information Technology, University Institute of Engineering and Technology (UIET), Panjab University, Chandigarh, India
[d] Department of Computer Science and Engineering, SBS State Technical Campus, Punjab, India

ARTICLE INFO

Article history:
Received 3 April 2019
Revised 20 May 2019
Accepted 28 June 2019
Available online 2 July 2019

Keywords:
DoS attack
DDoS attack
Apache Hadoop
Hadoop Distributed File System (HDFS)
MapReduce
Entropy
Big Data

ABSTRACT

During the past few years, the traffic volume of legitimate traffic and attack traffic has increased manifolds up to Terabytes per second (Tbps). Because of the processing of such a huge traffic volume, it has become implausible to detect high rate attacks in time using conventional DDoS defense architectures. At present, the majority of the DDoS defense systems are deployed predominantly at the victim-end domain But these victim-end defense systems themselves are vulnerable to HR-DDoS attacks as the mammoth volume of attack traffic is generated by such type of attacks. The insufficient computational resources further make the problem more crucial at the victim-end. This paper proposed a distributed and collaborative architecture called E-Had that is capable of efficiently processing a large amount of data by distributing it among a number of mappers and reducers in a Hadoop based cluster. The proposed E-Had system has been comprehensively validated using various publicly available benchmarked datasets and real datasets generated in HA-DDoS testbed in terms of various detection system evaluation metrics. The experimental results clearly show that the proposed detection system is capable of early detection of different scenarios of DDoS attacks along with differentiating them from flash crowds.

1. Introduction

In the recent times, thousands of organizations such as domain name servers, payment gateways, banks, search engines, educational institutes, social websites such as Whatsapp, Twitter, Facebook, stock trades, commercial cloud based service providers servers such as Amazon, Google, weather forecasting, etc. are growing exponentially to provide Internet-based services and applications to the end users. It has lead to the increase in network traffic over the Internet manifolds. This increasing usage of Internet based applications has further induced rise in the misuse of Internet (Hoque et al., 2014). Out of the various types of malware present on the Internet, DDoS attacks pose an austere menace to Internet and its architecture. According to the traffic rate, DDoS attacks can be classified into low rate (LR-DDoS) and high rate (HR-DDoS) DDoS attacks. As per WISR (2018), HR-DDoS having traffic volume more than 1 Terabits per second (Tbps) are frequent nowadays. In 2018, Github suffered from the largest DDoS attack in the history having traffic volume around 1.35 Tbps, however, Github recovered from it within 8 min (Github, 2018). Such DDoS attacks can cripple down the services of organizations in no time and can lead to devastating impacts of huge financial losses. So, it is very critical to detect and react in such situations in a short time so as to ensure uninterrupted delivery of various Internet-based services and applications. According to Wang et al. (2012), HR-DDoS attacks are often variegated with LR-DDoS attacks nowadays so as to remain undetected but often lead to devastating impacts.

Further, HR-DDoS attacks needs to be differentiated from similar looking flash crowd (Behal and Kumar, 2017). A flash crowd (FC) is an event that also causes a denial of service to extensively used Internet-based services and applications. An FC is similar to

* Corresponding author.
E-mail address: sunnybehal@sbsstc.ac.in (S. Behal).
[1] orcid: 0000-0002-1983-668X.
[2] orcid: 0000-0002-4647-2842.
[3] orcid: 0000-0001-9877-0238.
[4] orcid: 0000-0002-9701-4358.

Peer review under responsibility of King Saud University.

Production and hosting by Elsevier

https://doi.org/10.1016/j.jksuci.2019.06.016
1319-1578/

a HR-DDoS attack wherein millions of clients try to access a specific web page of a website simultaneously (Bhandari et al., 2016). This sudden increase in normal network traffic is primarily because of some breaking news happening around the world like launching of a latest product by trend setters like Apple, Samsung, etc or may be due to the sudden death of a famous celebrity. It leads to delayed responses from a web service similar to the case of a HR-DDoS attack and thus, requires immediate attention and action. The problem turns more crucial when sophisticated attackers launch DDoS attacks during the high rate legitimate hours such as in the case of FCs. Such kind of attack is called a flash crowd attack.

To combat from such types of HR-DDoS attacks, a DDoS detection system can be deployed at source-end, intermediate network, or victim-end (Bhuyan et al., 2016; Kumar et al., 2007; Behal et al., 2018). However, each of these deployment location has its own merits and demerits. A victim-end deployment is the defined as the location where the web server is installed and hosted. The nodes near this location can closely observe the complete attack traffic, model its behavior and detect anomalies efficiently whereas the mechanisms deployed elsewhere (Source-end and Intermediate) can see only a partial attack traffic and might need to take action based on this incomplete attack information. So, it is vital to deploy and install a DDoS defense system at proper location (Gulisano et al., 2015). A victim-end deployment is the best location where whole of the network traffic is converged. The defense system deployed at this location can produce high detection accuracy. However, during a typical DDoS attack, various network and server resources like bandwidth, CPU cycles, memory, etc. often gets overloaded. In such cases, it may not be possible for the victim-end based deployment solution to detect and characterize the legitimate and attack traffic efficiently. It is because of the lack of sufficient computational resources at this deployment location. In such cases, it may start dropping legitimate packets instead of dropping attack packets, results in the increase of false positives. Authors in Kumar et al. (2007), Behal et al. (2018), and Sachdeva et al. (2016) have clearly highlighted the main reasons for the failures of the current victim-end based deployments. Some of the key reasons are the decentralized architecture of Internet, unplanned and automated infrastructure changes, collateral damage, lack of collaboration among ISPs, non-availability of latest real datasets for forensics and deployment issues, etc. Even the traditional measures such as router ACLs, firewalls, IDS/IPS systems are not able to defend against HR-DDoS attacks effectively. Therefore, the availability of perfect solution of detecting HR-DDoS attacks is still an open research issue (Cui et al., 2016).

Many fellow researchers have proposed efficient Hadoop-based DDoS defense systems (Lee et al., 2011; Khattak et al., 2011; Zhao et al., 2015; Dayama et al., 2015; Zhang et al., 2016; Hameed and Ali, 2016; Hameed and Ali, 2018; Hsieh and Chan, 2016; Alsirhani et al., 2018; Alsirhani et al., 2018; Maheshwari et al., 2018; Chhabra et al., 2018). but majority of them have used counter based approaches to detect HR-DDoS attacks which fails in nowadays high rate traffic scenarios and can be easily deceived by sophisticated attackers. In this paper, we have proposed a novel hadoop based distributed victim-end framework called E-Had that is capable of efficiently analyzing a large volume of network traffic flows. E-Had makes use of an open source Apache Hadoop technology to process huge amount of data in parallel by dividing the tasks among cluster of datanodes (Apache Hadoop, 2018). There are two types of nodes in hadoop- Namenode and Datanode. The Namenode is used to map the large tasks among n number of datanodes also called mappers. The network traffic is processed at all of these mapper nodes. Out of these mappers, one mapper is designated as reducer. All mapper nodes send their intermediate results to reducer node. The prime objective of this reducer node is to summarize the intermediate results received from mappers. The aggregated detection metric is then computed using our proposed mathematical model (Eq. (2)) at this reducer node. We are the first one to propose such a shannon entropy based distributed mathematical model on hadoop based clusters to dispense the computational and storage costs of a victim-end solution. This proposed approach also makes the existing victim-end DDoS defense solutions more capable to handle the large traffic volume of DDoS attacks where most of the existing solutions have failed. As part of the work, a real hadoop based DDoS testbed (HA-DDoS) has been developed to replay the traffic traces of benchmarked datasets to validate the proposed E-Had framework. The E-Had framework splits the large volume network traffic files into multiple parts of fixed size (64 MB,128 MB,256 MB) and has the capability to analyze them concurrently by sub dividing the task among several mappers. It is also possible that one datanode may process more than one mapper job simultaneously depending on the number of data blocks present on that datanode. It has lead to low computational cost and fast response time of the proposed E-Had distributed detection system.

The major contributions of this paper are summarized as:

- This paper proposes a Hadoop based distributed, automated and collaborative DDoS detection framework system for ISP networks, called E-Had that is capable of efficiently detecting different scenarios of DDoS attacks and Flash Crowds.
- E-Had distributes the computational and memory overhead of the detection metric to multiple datanodes called Mappers. The partial results of the mappers are then sent to a central datanode called a Reducer. Reducer node then aggregates the partial results obtained from various Mappers using the proposed distributed mathematical formula (Eq. (2)) using information theory based shannon entropy detection metric.
- The proposed E-Had framework works in a automated way as network flows are assigned to various mapper nodes of Hadoop framework on the fly without storing in the file system (HDFS) of Hadoop.
- E-had works in a collaborative way as various mapper nodes collaborate with each other to send their partial computed results to a designated reducer node for final detection metric computation.
- The proposed E-Had detection system uses the same information theory based information distance (ID) metric for detecting attack traffic as well as characterizing it from similar looking high rate legitimate traffic called, flash crowds.
- The design of the proposed E-Had detection framework is robust as it can continue its working even in case some of the datanodes of Hadoop, so called, Mappers do not respond in-time.
- The proposed E-Had detection system takes less computational time and has fast response time as compared to the traditional single victim-end solution.
- The proposed E-Had detection system has been validated using real datasets of CAIDA and MIT Lincoln in a Hadoop based experimental (HA-DDoS) testbed in terms of various performance evaluation metrics such as detection accuracy, ROC curve, FPR-FNR curve, classification rate, precision and processing time.

The remaining part of the paper is organized as follows. Section 2 presents the literature review, Section 3 describes the architecture of proposed E-Had framework, Section 4 describes the proposed methodology, Section 5 discusses the experimental setup and performance evaluation of E-Had and finally Section 6 conclude the paper by highlighting the scope for future work.

2. Related work

Fellow researchers have proposed many capable solutions using latest Hadoop based technology to combat against this research gap but most of these approaches are implemented in offline mode. Lee et al. (2011) proposed a DDoS attack detection system based on Hadoop framework (Hadoop Framework, 2019). They implemented a counter based detection algorithm to HR-detect attacks using the MapReduce programming model of the Hadoop. They validate their proposed offline approach in a testbed consisting of one master (Namenode) and 10 slaves (Datanodes). The authors investigated that their proposed system require 25 min for processing 500 GB and 47 min for processing 1 TB attack traffic files but it takes only a few seconds to flood the victim network and render its services unavailable to legitimate users. The authors further extended their idea in Lee and Lee (2013) to reduce this time duration. They implemented a more powerful traffic analysis algorithm by implementing two testbeds with 30 nodes cluster and commodity configuration with 200 nodes cluster respectively. The authors claimed that their approach can process 100% packets successfully in a short span of time.

Khattak et al. (2011) proposed an offline DDoS forensics system using MapReduce programming model of Hadoop framework. They used "horizontal threshold" (number of ping requests allowed on victim host) and "vertical threshold" (number of ping requests allowed to external host from internal host) within the specified time window. Authors validate their proposed defense system using real dataset of MIT Lincoln LLS-DDoS-1.0.

Zhao et al. (2015) proposed a Hadoop based DDoS defense system using a neural network. Their proposed system consists of a cluster of zombies, victim server and Hadoop-Hbase DDoS detection system implemented on a cloud platform. However, the limitation of neural network based system is the duration of training period required to train the detection system. Such kind of detection systems are error prone as they can be easily deceived if sophisticated attackers observed and followed the legitimate traffic patterns. In this system, the master node perform the reducer task but according to the Hadoop framework the job of master node is of only monitoring, maintaining the metadata of data blocks and the resource management whereas the slave nodes perform the mapper and reducer tasks.

Dayama et al. (2015) proposed a counter based approach implemented in Hadoop to protect DDoS attacks. However, the sophisticated attackers can easily bypass such defense systems. The legitimate traffic can be treated as an attack traffic that can reduce the detection accuracy and false positive rates. Further, Zhang et al. (2016) proposed a defense system to fight against IP spoofing based DDoS attacks. They proposed many inbound and outbound rules for the abnormal traffic detection mechanism and used non-parametric CUSUM algorithm to efficiently detect DDoS attacks. However, authors did not provide any details about the Hadoop testbed configuration. However, nowadays attackers can easily compromise millions of devices to launch a sophisticated attack towards victim without IP spoofing.

In some recent works, Hameed and Ali (2016) proposed a Hadoop based HADEC framework to combat against HR-DDoS attacks. Authors claimed that HADEC is a near to live detection framework that implement counter based DDoS detection algorithm for four major flooding attacks (ICMP, UDP, TCP-SYN, HTTP-GET). To evaluate the framework, they used Mausezahn (2019) attack tool to generate attack traffic. HADEC framework consists of capturing server, detecting server (Namenode) and 10 data nodes. They extended their work in Hameed and Ali (2018) by adding more traffic scenarios for the validation purpose. They used the threshold value of 500 and 1000 packets per second for differentiating attack and legitimate traffic. We also used the same threshold values in our proposed E-Had framework so as to compare the results with HADEC.

Hsieh and Chan (2016) presented an Apache Spark based DDoS attack detection system using a neural network. The detection system is trained and validated using real dataset of 2000 DARPA LLS-DDoS 1.0. The authors claimed the detection accuracy over 94%. Bachupally et al. (2016) proposed a Hadoop based security analysis system using Apache Hive (2019) application. They validated their proposed approach using first-day data from NCCDC provided by the PREDICT (Predict, 2019) cyber security dataset repository. However, the scope of this system is security analysis of traffic only.

Some authors like Alsirhani et al. (2018) combined Apache Spark (2019) to propose a DDoS attack defense system. They used a classification algorithm to differentiate attack traffic from legitimate traffic and a fuzzy logic system to choose best classification algorithm whereas in Alsirhani et al. (2018), they used Gradient Boosting algorithm used to differentiate attack flows from legitimate flows. However, this proposed system require more time for training and testing phases.

Maheshwari et al. (2018) proposed a faster DDoS detection system using a Hadoop cluster. They implemented a counter based algorithm using MapReduce programming model to count the number of requests per source IP related to different protocols such as HTTP, TCP, and ICMP in each time slot. If the count exceeds the threshold value then declare the source IP is of an attacker and block it. If the attacker trains the system so that it never cross the threshold limits, then it will consequently bypasses the detection system. However, the authors validated their proposed system on a very small amount of data (200 MB).

Many authors have proposed DDoS analytic systems using machine learning approach implemented on Hadoop based frameworks. Chhabra et al. (2018) presented an offline forensics analytic system for DDoS attacks using Hadoop framework and implemented using supervised machine learning algorithm. They validated their proposed framework using CAIDA dataset and claimed 99.34% detection accuracy. Rathore et al. (2016) proposed a real-time IDS system using Hadoop framework for processing Big Data. They compared the performance of several machine learning algorithms such as SVM, RFTree, REPTree, J48, and Naive Bayes etc. Authors claimed that J48 and REPTress perform better than other classifiers for processing large amount of data. Cui and He (2016) uses the idea presented in Rathore et al. (2016) to identify the network anomalies by processing huge amount of real-time traffic. They also compared the performance of multiple machine learning algorithms using MapReduce programming model and WEKA tool, and claimed the detection accuracy of over 90%.

However, only a few authors have used information theory based metrics to detect DDoS attacks on Hadoop based frameworks. In Tian et al. (2015), Tian et al. used information theory based adjustable piece-wise Shannon entropy(APSE) for network anomaly detection system on Hadoop based detection system and they claimed that APSE performed better than traditional entropy.

Some authors like Zhou et al. (2014) and Terzi et al. (2017) used publicly available real datasets to validate their Hadoop based detection systems. They analyzed NetFlow dataset, CTU-13 dataset and CAIDA dataset. Authors in Terzi et al. (2017) claimed that their proposed system perform better than the traditional analysis of netflows as done by Zhou et al. (2014).

2.1. Discussion

In this section, we extensively reviewed the existing literature (summarized in Table 1) related to DDoS attack detection using

Table 1
Comparison of related work in Hadoop based DDoS defense systems.

Authors	Experimental Setup	Configuration	Datasets and tools used	Detection Metrics Detection Methodology used	Performance Measurements
Lee et al. (2011)	Hadoop Cluster 1-Master (Namenode) 10-Slaves (Datanodes)	Quad-core:2.93 GHz, Intel i7 RAM: 16 GB, HDD:1 TB, Ethernet: 1Gbps	Offline 500 GB & 1 TB data	Page request rate Time interval, Threshold value unbalanced ratio and access pattern	25 Mins for 500 GB and 47 Mins for 1 TB data
Khattak et al. (2011)	–	–	MIT Lincoln LLS-DDoS-1.0	HDFS and MapReduce method based on # of ping requests from victim # of ping requests to victim	Performance compared with java program Case I: 911 MB Proposed system takes 50% less time Case II: 6.9 GB proposed system takes 65% less time
Suryawanshi and Mande (2013)	–	–	–	HDFS and MapReduce	100% traffic monitored
Fontugne et al. (2014)	Hadoop cluster on cloud: 1-master(namenode + datanode) 5-slaves(datanode)	No details	CAIDA	HDFS and MapReduce Apache Hive and Mahot	Performance metrics: False positive & False negative.
Zhao et al. (2015)	–	–	–	Neural network used	–
Dayama et al. (2015)	Hadoop Cluster 1-Master (Namenode) 10-Slaves (Datanodes)	Quad-core:2.93 GHz, Intel i7 RAM: 16 GB, HDD:1 TB, Ethernet: 1Gbps	–	Page access pattern Number of Requests unbalanced ratio and access pattern	–
Tian et al. (2015)	Hadoop cluster: 5-physical servers used. 1-master (namenode + datanode) 4-slaves(datanode).	Each server integrates: Two 2.60 GHz, RAM:32 GB CPU Intel Xeon E5-2360, HDD:9 TB	Data collected at edge router of university campus network in IPFIX format.	Collect traffic at edge router. HDFS and MapReduce & Evaluate anomaly detection system using experiment & mathematical modeling.	Claimed:APSE perform better than SE.
Zhang et al. (2016)	No details provided.	No details provided.	No details provided.	HDFS and MapReduce non-parametric CUSUM algorithm	Better than Cheng Jin
Hameed and Ali (2016)	Hadoop Cluster 1-Master (Namenode) 10-Slaves (Datanodes)	CPU 2.60 GHz, Intel i5 RAM: 16 GB, HDD:1 TB, Ethernet: 1Gbps	Mausezan 3 attackers and 2 legitimate nodes	HDFS and MapReduce Number of Requests	varying Cluster size
Hsieh and Chan (2016)	–	Apache Spark Apache Spark is used.	2000 DARPA LLS-DdoS-1.0 2000-DARPA	Neural network Time Window = 1000 Seconds	Accuracy 94% accuracy
Network security Bachupally et al. (2016)	–	Apache Hadoop, Hive	First day data from NCCDC	HDFS, Hive # of Requests	Detection Time: Dataset1: 42 s Dataset2: 46 s Dataset3: 109 s
Rathore et al. (2016)	Hadoop Cluster multiple master and slaves nodes. Evaluated on 1-host	-	DARPA,KDD99 & NSL-KDD.	Five Machine Learning techniques are used: SVM, RFTree,REPTree, J48 & Naive Bayes.	Claimed: REPTree & J48 perform better.
Cui et al. (2016)	–	–	KDD'99.	3 Machine Learning algorithms are used: SVM, Decision Tree and Naive Bayes HDFS, MapReduce and WEKA	Detection accuracy claimed: more than 90% and decision tree performed better than other two
Terzi et al. (2017)	–	Apache spark.	CTU-13.	1. NetFlows data into intervals. K-means algorithm Euclidean distance	Detection accuracy: claimed 96%
Alsirhani et al. (2018)	Hadoop Cluster 1-Master (Namenode) 3-Slaves (Datanodes)	CPU 2.60 GHz, Intel i7 RAM: 16 GB, HDD:500 GB, Ethernet: 1Gbps	Spark	Classification algorithms Fuzzy Logic HDFS and MapReduce	Detection accuracy FPR, Recall Precision and F-Score
Alsirhani et al. (2018)	Hadoop Cluster 1-Master (Namenode) 3-Slaves (Datanodes)	CPU 2.60 GHz, Intel i7 RAM: 16 GB, HDD:500 GB, Ethernet: 1Gbps	CAIDA attack dataset	HDFS and Apache Spark Gradient Boosting Algorithm	Detection accuracy FPR, Recall, Precision F-Score and ROC Curve

Table 1 (*continued*)

Authors	Experimental Setup	Configuration	Datasets and tools used	Detection Metrics Detection Methodology used	Performance Measurements
Maheshwari et al. (2018)	Hadoop Cluster 1-Master (Namenode) 2-Slaves (Datanodes)	CORE Intel i5 RAM: 8 GB, HDD:500 GB,	Attack Cluster of 7 physical hosts	HDFS and MapReduce # of Requests	Varying Cluster size Varying File Sizes
Chhabra et al. (2018)	Hadoop Cluster 1-Master (Namenode) 3-Slaves (Datanodes)	Hadoop RAM: 8 GB, HDD:500 GB,	CAIDA Attack dataset 7 physical hosts	Random Forest Supervised Machine Learning Algo University Network Samples	Detection Accuracy claimed 99.34%

Hadoop based frameworks and observed that majority of the researchers have used counter based offline detection algorithms to detect anomalies in the network traffic but such kind of deployments can be easily deceived by the sophisticated attackers. Further, majority of the researchers have used MIT Lincoln dataset to represent legitimate traffic and CAIDA dataset to represent attack traffic. Only a few of the authors have used information theory based detection metrics in Hadoop based frameworks. Therefore, it motivate us to propose an information theory based realtime distributed detection model using Hadoop based framework called E-Had so as to reduce the detection time. Further, the proposed E-Had hadoop based detection framework is capable of detecting different scenarios of HR-DDoS attacks along with differentiating it from similar looking flash crowds which the fellow researchers have not done. It has also been observed that the fellow researchers have used only a few number of datanodes (mappers) for validating their approaches but in the case of HR-DDoS attacks, the detection system needs to be validated under high volume of attack traffic which we did by replaying the traffic traces of different scenarios of DDoS attacks (constant rate attack, increasing rate attack, pulsating attack and subgroup attack) from CAIDA dataset.

3. Hadoop based distributed approach using Shannon entropy

This section briefly introduces information theory based Shannon entropy detection measure and extend the concept to propose a Hadoop based distributed detection system called E-Had to detect HR-DDoS attacks.

3.1. Shannon entropy

Claude Shannon in 1948 defined Shannon entropy ($H(x)$) as:

$$H(x) = -\sum_{i=1}^{n} p_i log_2 p_i \quad (1)$$

where p_i is the probability of the occurrence of an event x.

3.2. Hadoop based distributed detection approach

This section presents the proposed hadoop based distributed approach to detect HR-DDoS attacks using Shannon entropy metric. An abstract architectural view of the proposed Hadoop based detection framework, E-Had is shown in Fig. 1. As shown, there are two types of nodes in Hadoop. The namenode is used to distribute the tasks among n datanodes also called mappers. Out of the n mappers, one node is designated as reducer. The reducer node aggregates the partial results (entropy values in our case) obtained from mapper datanodes using the proposed distributed mathematical model (Eq. (2)).

In the case of a traditional DDoS detection systems deployed at the victim-end, the huge network traffic volume generated during HR-DDoS attacks and lack of sufficient computational resources (memory, processing power) makes detection system itself vulnerable to DDoS attacks. To address this issue, we distribute the costs of traffic monitoring and detection logic to cluster of nodes called mappers in Hadoop based framework.

To understand this situation, kindly refer to the statistics given in 2. Here, $M_1, M_2, \ldots M_n$ are the mapper nodes of a victim network where n is the total number of data nodes. Let $N_1, N_2, \ldots, N_n$ be the number of traffic flows in M_i. Let $H_1, H_2 \ldots H_n$ be the frequency histograms associated with mappers M_i in time window Δt. Each H_i represent $X_{i1}, X_{i2} \ldots, X_{iNi}$ where X_{ij} represent number of packets for M_i for network flow j where $S_i = \sum_{j=1}^{N_i} X_{ij}$. Let She_i represent entropy for mapper M_i. Entropy She_i computed at M_i along with

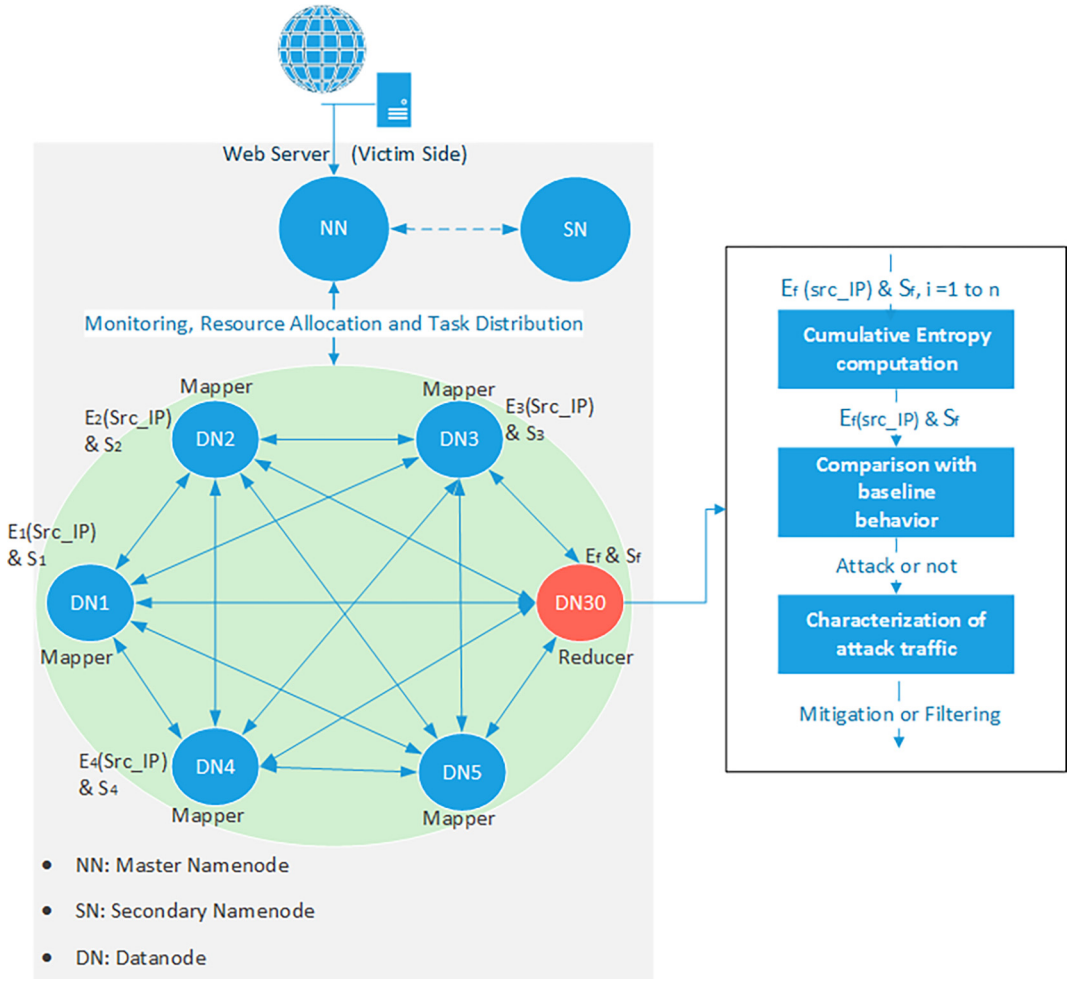


Fig. 1. E-Had: Hadoop based distributed DDoS detection framework.

Table 2
Statistics collected at all mappers of Hadoop architecture.

	1	2	3	4	5	⋯	⋯	N_i	Total number of arrived packets (S_i)	Shannon Entropy (She_i)
1	X_{11}	X_{12}	X_{13}	X_{14}	X_{15}	⋯	⋯	X_{1N_1}	S_1	She_1
2	X_{21}	X_{22}	X_{23}	X_{24}	X_{25}	⋯	⋯	X_{2N_2}	S_2	She_2
3	X_{31}	X_{32}	X_{33}	X_{34}	X_{35}	⋯	⋯	X_{3N_3}	S_3	She_3
4	X_{41}	X_{42}	X_{43}	X_{44}	X_{45}	⋯	⋯	X_{4N_4}	S_4	She_4
.	.	.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.	.	.
n	X_{n1}	X_{n2}	X_{n3}	X_{n4}	X_{n5}	⋯	⋯	X_{nN_n}	S_n	She_n

S_i is sent to M_s (data node that connects web server to victim network) where final computation of entropy She_f is calculated using

$$She_f(x) = \left(\frac{1}{S_f}\right)\sum_{i=1}^{n} S_i(She_i - log(S_i)) + log(S_f) \quad (2)$$

where $S_f = \sum_{i=1}^{n} S_i$ is total number of packets observed at all mappers in a time window.

Table 2 shows the statistics collected at all mappers of the victim-side network. The sample Entropy She_i at mapper M_i is computed using equation (log is used in place of $\log_2$ for simplicity) as follows:

– $She_i = \sum_{j=1}^{N_i}\left(\frac{X_{ij}}{S_i} log\left(\frac{X_{ij}}{S_i}\right)\right)$

For a specific mapper, say, M_1, the equation becomes:

– $She_1 = \sum_{j=1}^{N_1}\left(\frac{X_{1j}}{S_1} log\left(\frac{X_{1j}}{S_1}\right)\right)$

Expand the summation, we get:

– $She_1 = \frac{X_{11}}{S_1} log\left(\frac{X_{11}}{S_1}\right) + \frac{X_{12}}{S_1} log\left(\frac{X_{12}}{S_1}\right) + \cdots + \frac{X_{1N_1}}{S_1} log\left(\frac{X_{1N_1}}{S_1}\right)$

Rearranging the multiplication factor

– $She_1 = log\left(\frac{X_{11}}{S_1}\right)^{\frac{X_{11}}{S_1}} + log\left(\frac{X_{12}}{S_1}\right)^{\left(\frac{X_{12}}{S_1}\right)} + \cdots + log\left(\frac{X_{1N_1}}{S_1}\right)^{\left(\frac{X_{1N_1}}{S_1}\right)}$ which is equivalent to:

– $She_1 = log\left(\left(\frac{X_{11}}{S_1}\right)^{\frac{X_{11}}{S_1}}\left(\frac{X_{12}}{S_1}\right)^{\frac{X_{12}}{S_1}} \cdots \left(\frac{X_{1N_1}}{S_1}\right)^{\frac{X_{1N_1}}{S_1}}\right)$

By using the property log(x + y) = log(xy).

$2^{-She_1} = \left(\frac{X_{11}}{S_1}\right)^{\frac{X_{11}}{S_1}} \left(\frac{X_{12}}{S_1}\right)^{\frac{X_{12}}{S_1}} \cdots \left(\frac{X_{1N_1}}{S_1}\right)^{\frac{X_{1N_1}}{S_1}}$ using the property $a = log_b(x) = b^a = x$

As $S_f = \sum_{i=1}^{n} S_i$, considering all flows are observed at name node. The traffic destined to only protected server is monitored at all mappers of the E-Had framework. Moreover, we have also assumed that flows observed by all mappers directed to protected server are different at different mappers. On the similar pattern,

$$2^{-She_f} = \left(\frac{X_{11}}{S_f}\right)^{\frac{X_{11}}{S_f}} \left(\frac{X_{12}}{S_f}\right)^{\frac{X_{12}}{S_f}} \cdots \left(\frac{X_{1N_1}}{S_f}\right)^{\frac{X_{1N_1}}{S_f}} \left(\frac{X_{21}}{S_f}\right)^{\frac{X_{21}}{S_f}} \left(\frac{X_{22}}{S_f}\right)^{\frac{X_{22}}{S_f}} \cdots \left(\frac{X_{2N_2}}{S_f}\right)^{\frac{X_{2N_2}}{S_f}} \left(\frac{X_{31}}{S_f}\right)^{\frac{X_{31}}{S_f}} \left(\frac{X_{32}}{S_f}\right)^{\frac{X_{32}}{S_f}} \cdots \left(\frac{X_{3N_3}}{S_f}\right)^{\frac{X_{3N_3}}{S_f}} \cdots \left(\frac{X_{n1}}{S_f}\right)^{\frac{X_{n1}}{S_f}} \left(\frac{X_{n2}}{S_f}\right)^{\frac{X_{n2}}{S_f}} \cdots \left(\frac{X_{nN_n}}{S_f}\right)^{\frac{X_{nN_n}}{S_f}} \quad (3)$$

Multiply both numerator and denominator by S_i where i varies from 1 to n.

$$\Rightarrow 2^{-She_f} = \left(\left(\frac{X_{11}}{S_f}\right)\left(\frac{S_1}{S_1}\right)\right)^{\left(\frac{X_{11}}{S_f}\right)\left(\frac{S_1}{S_1}\right)} \left(\left(\frac{X_{12}}{S_f}\right)\left(\frac{S_1}{S_1}\right)\right)^{\left(\frac{X_{12}}{S_f}\right)\left(\frac{S_1}{S_1}\right)} \cdots \left(\frac{X_{1N_1}}{S_f}\right)\left(\left(\frac{S_1}{S_1}\right)\right)^{\left(\frac{X_{1N_1}}{S_f}\right)\left(\frac{S_1}{S_1}\right)} \left(\left(\frac{X_{21}}{S_f}\right)\left(\frac{S_2}{S_2}\right)\right)^{\left(\frac{X_{21}}{S_f}\right)\left(\frac{S_2}{S_2}\right)} \left(\left(\frac{X_{22}}{S_f}\right)\left(\frac{S_2}{S_2}\right)\right)^{\left(\frac{X_{22}}{S_f}\right)\left(\frac{S_2}{S_2}\right)} \cdots \left(\frac{X_{2N_2}}{S_f}\right)\left(\left(\frac{S_2}{S_2}\right)\right)^{\left(\frac{X_{2N_2}}{S_f}\right)\left(\frac{S_2}{S_2}\right)} \cdots \left(\left(\frac{X_{n1}}{S_f}\right)\left(\frac{S_n}{S_n}\right)\right)^{\left(\frac{X_{n1}}{S_f}\right)\left(\frac{S_n}{S_n}\right)} \left(\left(\frac{X_{n2}}{S_f}\right)\left(\frac{S_n}{S_n}\right)\right)^{\left(\frac{X_{n2}}{S_f}\right)\left(\frac{S_n}{S_n}\right)} \cdots \left(\frac{X_{nN_n}}{S_f}\right)\left(\left(\frac{S_n}{S_n}\right)\right)^{\left(\frac{X_{nN_n}}{S_f}\right)\left(\frac{S_n}{S_n}\right)} \quad (4)$$

After rearranging the multiplication factor:

$$\Rightarrow 2^{-She_f} = \left(\left(\frac{X_{11}}{S_1}\right)\left(\frac{S_1}{S_f}\right)\right)^{\left(\frac{X_{11}}{S_1}\right)\left(\frac{S_1}{S_f}\right)} \left(\left(\frac{X_{12}}{S_1}\right)\left(\frac{S_1}{S_f}\right)\right)^{\left(\frac{X_{12}}{S_1}\right)\left(\frac{S_1}{S_f}\right)} \cdots \left(\frac{X_{1N_1}}{S_1}\right)\left(\frac{S_1}{S_f}\right)^{\left(\frac{X_{1N_1}}{S_1}\right)\left(\frac{S_1}{S_f}\right)} \left(\left(\frac{X_{21}}{S_2}\right)\left(\frac{S_2}{S_f}\right)\right)^{\left(\frac{X_{21}}{S_1}\right)\left(\frac{S_2}{S_f}\right)} \left(\left(\frac{X_{22}}{S_2}\right)\left(\frac{S_2}{S_f}\right)\right)^{\left(\frac{X_{22}}{S_2}\right)\left(\frac{S_2}{S_f}\right)} \cdots \left(\frac{X_{2N_2}}{S_2}\right)\left(\left(\frac{S_f}{S_f}\right)\right)^{\left(\frac{X_{2N_2}}{S_2}\right)\left(\frac{S_2}{S_f}\right)} \left(\left(\frac{X_{n1}}{S_n}\right)\left(\frac{S_f}{S_f}\right)\right)^{\left(\frac{X_{n1}}{S_n}\right)\left(\frac{S_n}{S_f}\right)} \left(\left(\frac{X_{n2}}{S_n}\right)\left(\frac{S_n}{S_f}\right)\right)^{\left(\frac{X_{n2}}{S_n}\right)\left(\frac{S_n}{S_f}\right)} \cdots \left(\frac{X_{nN_n}}{S_n}\right)\left(\left(\frac{S_n}{S_f}\right)\right)^{\left(\frac{X_{nN_n}}{S_n}\right)\left(\frac{S_n}{S_f}\right)} \quad (5)$$

Replacing 2^{-She_i} for $\left(\frac{X_{i1}}{S_i}\right)^{\frac{X_{i1}}{S_i}} \left(\frac{X_{i2}}{S_i}\right)^{\frac{X_{i2}}{S_i}} \cdots \left(\frac{X_{iN_i}}{S_i}\right)^{\frac{X_{iN_i}}{S_i}}$ for all i = 1 to n, we get

$$\Rightarrow 2^{-She_f} = \left(2^{-She_1}\right)^{\left(\frac{S_1}{S_f}\right)} \left(2^{-E_2}\right)^{\left(\frac{S_2}{S_f}\right)} \cdots \left(2^{-She_n}\right)^{\left(\frac{S_n}{S_f}\right)} \left(\frac{S_1}{S_f}\right)^{\left(\frac{S_1}{S_f}\right)} \left(\frac{S_2}{S_f}\right)^{\left(\frac{S_2}{S_f}\right)} \cdots \left(\frac{S_n}{S_f}\right)^{\left(\frac{S_n}{S_f}\right)} \quad (6)$$

Taking log on both side, we get

$$\Rightarrow -She_f = log\left(\left(2^{-She_1}\right)^{\left(\frac{S_1}{S_f}\right)} \left(2^{-E_2}\right)^{\left(\frac{S_2}{S_f}\right)} \cdots \left(2^{-She_n}\right)^{\left(\frac{S_n}{S_f}\right)}\right) + log\left(\left(\frac{S_1}{S_f}\right)^{\left(\frac{S_1}{S_f}\right)} \left(\frac{S_2}{S_f}\right)^{\left(\frac{S_2}{S_f}\right)} \cdots \left(\frac{S_n}{S_f}\right)^{\left(\frac{S_n}{S_f}\right)}\right) \quad (7)$$

implies

$$\Rightarrow -She_f = -She_1\left(\frac{S_1}{S_f}\right) - She_2\left(\frac{S_2}{S_f}\right) - \cdots - She_n\left(\frac{S_n}{S_f}\right) + \left(\frac{S_1}{S_f}\right)log\left(\frac{S_1}{S_f}\right) + \left(\frac{S_2}{S_f}\right)log\left(\frac{S_2}{S_f}\right) \cdots + \left(\frac{S_n}{S_f}\right)log\left(\frac{S_n}{S_f}\right)$$

implies

$$\Rightarrow She_f = E_1\left(\frac{S_1}{S_f}\right) + She_2\left(\frac{S_2}{S_f}\right) \cdots + She_n\left(\frac{S_n}{S_f}\right) - \left(\frac{S_1}{S_f}\right)log\left(\frac{S_1}{S_f}\right) - \left(\frac{S_2}{S_f}\right)log\left(\frac{S_2}{S_f}\right) \cdots - \left(\frac{S_n}{S_f}\right)log\left(\frac{S_n}{S_f}\right)$$

implies

$$\Rightarrow She_f = \tfrac{1}{S_f}(She_1S_1 + She_2S_2 \cdots + She_nS_n) - \tfrac{1}{S_f}\left(S_1log\left(\frac{S_1}{S_f}\right) + S_2log\left(\frac{S_2}{S_f}\right) + S_nlog\left(\frac{S_n}{S_f}\right)\right)$$

implies

$$\Rightarrow She_f = \tfrac{1}{S_f}((She_1S_1 + She_2S_2 \cdots + She_nS_n) - \left(S_1log\left(\frac{S_1}{S_f}\right) + S_2log\left(\frac{S_2}{S_f}\right) + S_nlog\left(\frac{S_n}{S_f}\right)\right))$$

implies

$$\Rightarrow She_f = \tfrac{1}{S_f}(S_1(She_1 - log(S_1)) + S_2(She_2 - log(S_2)) + \cdots + S_n(She_n - log(S_n)) + S_1log(S_f) + S_2log(S_f) + \cdots S_nlog(S_f))$$

implies

$$\Rightarrow She_f = \tfrac{1}{S_f}(S_1(She_1 - log(S_1)) + S_2(She_2 - log(S_2)) + \cdots + S_n(She_n - log(S_n)) + log(S_f)(S_1 + S_2 + \cdots S_n))$$

As $S_f = S_1 + S_2 \cdots + S_n$, it implies.

$$She_f = \tfrac{1}{S_f}(S_1(She_1 - log(S_1)) + S_2(She_2 - log(S_2)) + \cdots + S_n(She_n - log(S_n)) + log(S_f)(S_f))$$

implies

$$\Rightarrow She_f(x) = \tfrac{1}{S_f}\left(\sum_{i=1}^{n} S_i(She_i - log(S_i))\right) + log(S_f)$$ which is same as Eq. (2).

4. Hadoop framework with HDFS and MapReduce

Apache Hadoop is an open-source distributed computing framework implemented in Java to provide with MapReduce as the programming model. Hadoop Distributed File System (HDFS) is used as the underlying distributed file system. Hadoop offers fault-tolerant computing and storage mechanisms for the large-scale cluster environment. Hadoop framework is made up of two components namely Hadoop Distributed File System (HDFS) (HDFS, 2019) and Yet Another Resource Negotiator (YARN) (Apache YARN, 2019; Lin and Lee, 2016). It allows to store a large amount of data on a cluster of nodes (HDFS) and a YARN module is provide resources to process Big Data (Kune et al., 2016) in parallel (using MapReduce programming model).

A MapReduce is a data processing layer of the Hadoop framework. It consist of two components namely mapper and reducer. The Fig. 2 shows the execution process of MapReduce programming model. We can write our own business logic in mapper and reducer program. It is also possible that one or multiple mappers (depending on the number of blocks stored at a datanode) are running on single datanode to process data blocks. However, one mapper can process one block at a time, stored its output at the local disk (called intermediate output). The sorting & shuffling process is executed between all mappers of a particular user job. The inter-

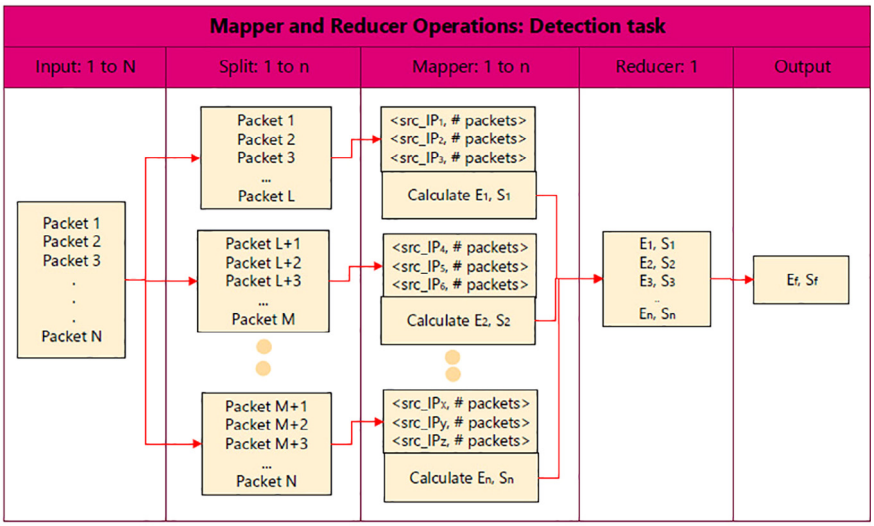


Fig. 2. Implementation of MapReduce approach of Hadoop in E-Had.

mediate output (mappers output) is, then, given as input to the reducer for further processing and the final output called reducer output. which is then stored on HDFS.

5. Proposed methodology

The detection framework comprises of two phases: (i) Traffic capturing and storing phase (ii) Resource allocation and data processing phase. Traffic capturing and storing phase captures the near to live incoming traffic target towards the victim and store on HDFS by splitting the large data files into multiple blocks with size 128 MB each (replication factor is 2). In resource allocation and data processing phase, YARN allocates the required number of resources (NodeManagers) to MapReduce which process the data on the cluster of nodes. After storing data into multiple Datanodes, next is processing task using MapReduce programming model by taking resources from YARN resource manager. The mapper programs run on multiple datanodes (the node which contains related data block, if one datanode has 3 blocks of processing job file then three mappers will run on it) and stored their an output on local disk and generally called an intermediate output. The next is sort and shuffling task, its perform over the network (buffer memory) and given as input to reducer program. There can be one or more reducer to reducing and combine an output of all mappers, store the final result on HDFS.

The proposed E-Had detection uses information theory based Shannon entropy (She) metric to detect various types of DDoS attacks along with differentiating them from similar looking flash crowds (FCs). The proposed Hadoop based distributed detection flowchart is shown in figure. To characterize different types of network traffic flows, we defined traffic flow as, "For a given mapper M_i in a network, a traffic flow is defined as a 4-tuple {srcIP, dstIP, protocol, arrival time} where srcIP/dstIP denotes source/destination IP address, protocol is traffic flow type, arrival time is the time when a packet reach the respective mapper node". We used the generalized detection system for information theory metrics as developed by Behal et al. (2018) for detecting different types of DDoS attacks. The authors used the concept of information distance to differentiate among various types of network traffic flows. We subtract the entropy value of current time window from the entropy of normal traffic flow (without attack scenario) to compute the information distance. i.e. ID = $|She_C - She_N|$. Based on this concept, the following mathematical models of different types of traffic flows are derived (Behal et al., 2018):

- Legitimate traffic flow, $n_C <= n_N + a * d_n \wedge ID_C <= ID_N + k * z_{ID_N}$, i.e. the traffic rate of current time window is less then equal to traffic rate of normal traffic, and information distance of current traffic flow is also less than or equal to the information distance between normal flows.
- LR-DDoS attack traffic flow, $n_C <= n_N + a * d_n \wedge ID_C > ID_N + k * z_{ID_N}$
- HR-DDoS attack traffic flow, $n_C > n_N + a * d_n \wedge ID_C > ID_N + k * z_{ID_N}$
- FE Traffic Flow, $n_C > n_N + a * d_n \wedge ID_C <= ID_N + k * z_{ID_N}$

Here, a, k are design parameters that denote tolerance factors of number of packets received and information distance between normal traffic flows per time window respectively. d_n is the standard deviation in the rate of network traffic. z_{ID_N} represent the standard deviation of ID values between normal flows. n_C and n_N represents incoming packet rate of current time window and normal traffic respectively. The threshold σ_1 is computed using $n_N + a * d_n$, whereas the threshold σ_2 is computed using $ID_N + k * z_{ID_N}$.

The flowchart of the proposed detection process is given in Fig. 3. All the mappers (M_i) compute their detection metric on the traffic destined to them by the namenode of the E-Had framework where webserver is attached. In each time window T_w, E-Had continuously send the calculated She_i entropy values and number of packets received per time window i.e. S_i at each mapper to the designated reducer node, which then, decides whether the system is under DDoS attack or not. In there is an attack, it activates the characterization and mitigation module of the framework whereas in the case of FC, it activates the captcha module to slow down the rate of incoming traffic. The proposed approach is robust as the framework can continue to work even some of the mappers does not respond in time or fail.

6. Experimental setup

As part of the work, a real Hadoop based DDoS detection testbed (HA-DDoS) has been designed as shown in Fig. 5. HA-DDoS testbed works in three phases (1) Traffic generation: both legitimate and attack live traffic, (2) Sniffing traffic, and (3) Hadoop based DDoS

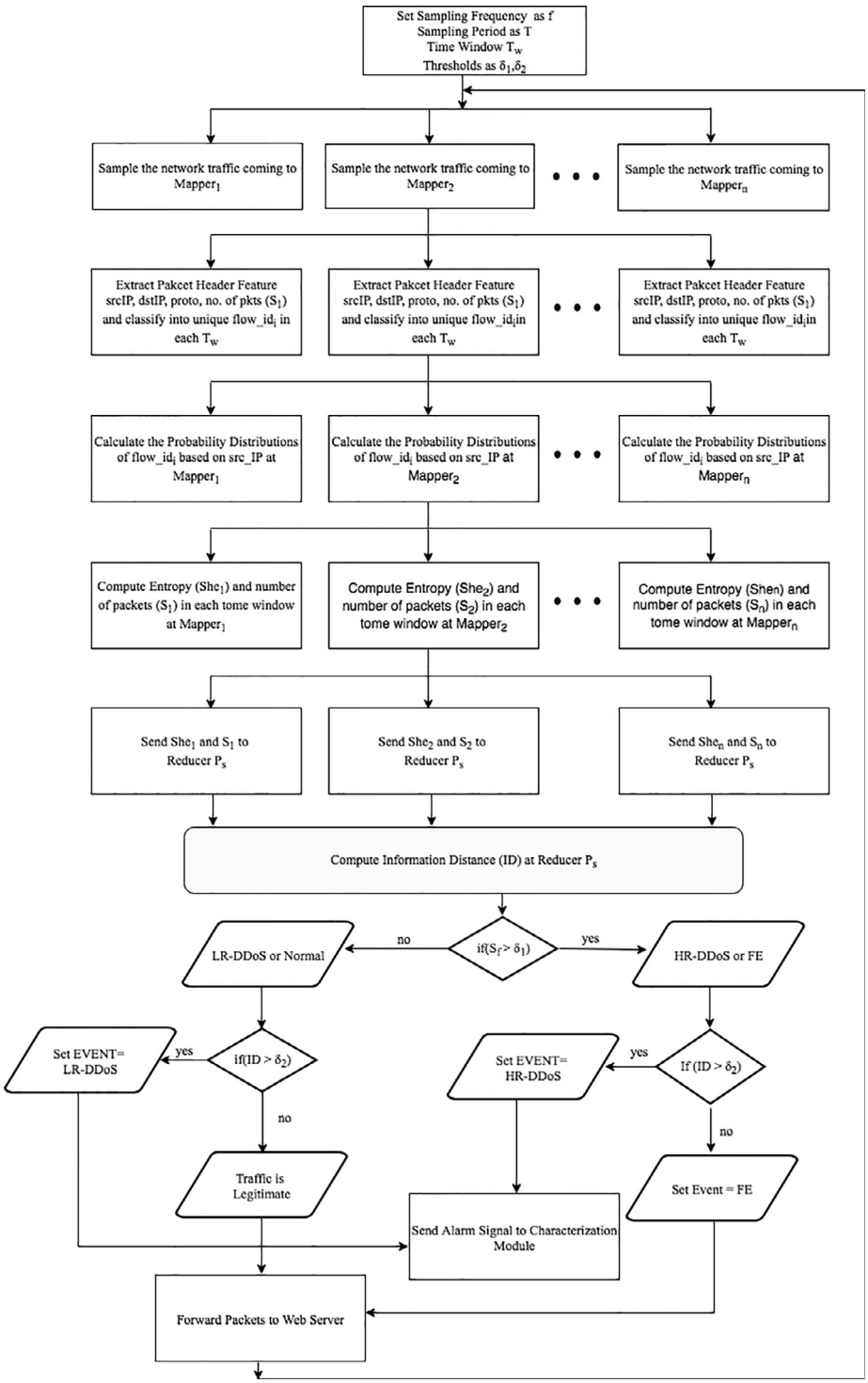
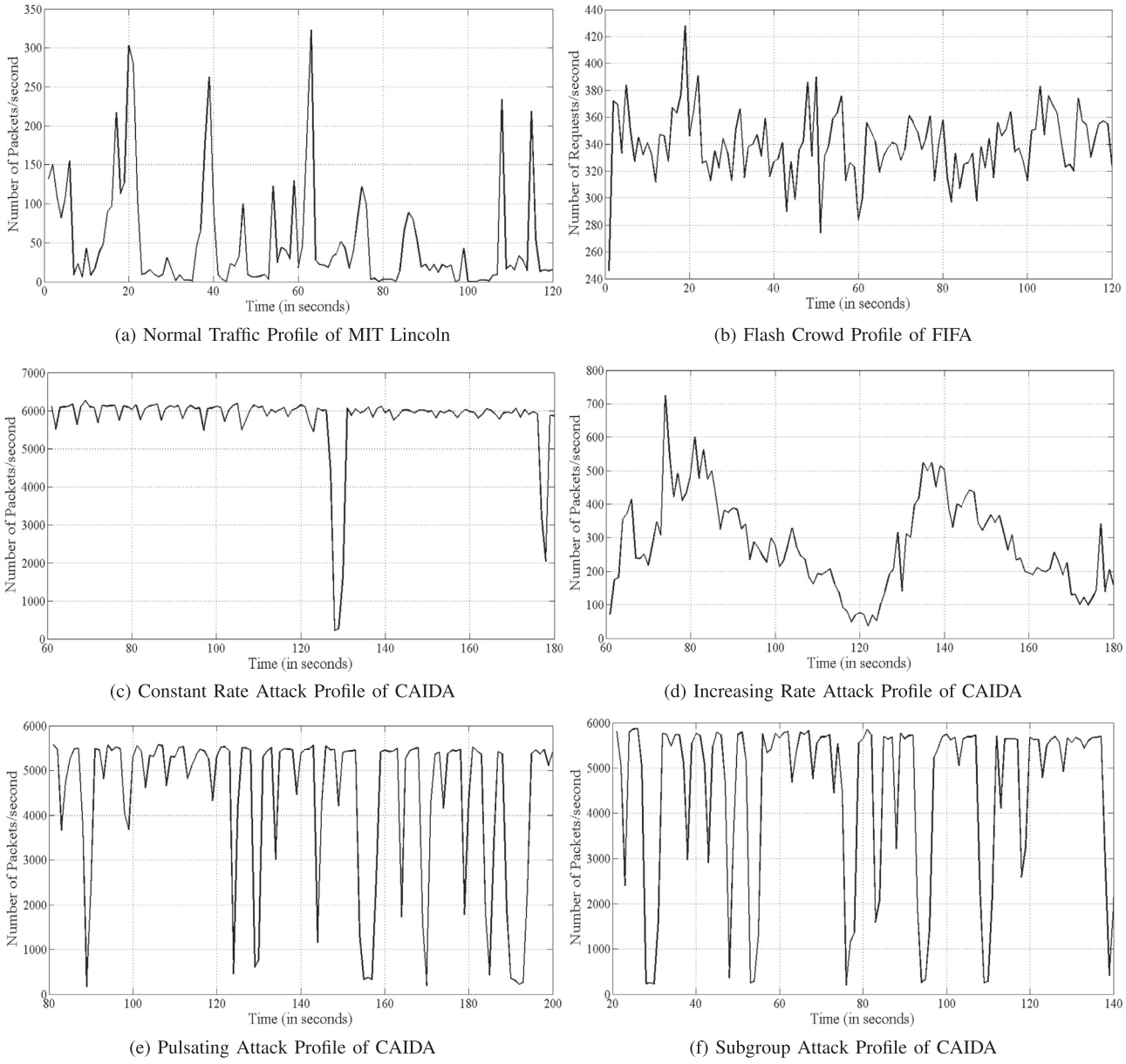


Fig. 3. Distributed DDoS detection methodology of E-Had.

detection framework. For traffic generation, we have used 30 real computers to generate legitimate and attack traffic. E-Had transfers the data on the fly to HDFS and various mapper nodes. The Hadoop framework, then, splits the data into multiple blocks of 128 MB size and assign computational task to multiple datanodes called mappers in parallel. HA-DDoS consists of 01 master Namenode, 01 secondary Namenode (Backup node if the master node fails during processing), and 30 slave nodes also called datanodes. The hardware configuration of each real computer system is Intel core i7 CPU 3.60Ghz with 8 cores, 500 GB HDD, 8 GB RAM and two network cards.

7. Performance evaluation

We used the legitimate traffic profiles of MIT (1998) (Fig. 5 and FIFA (1998) (Fig. 4b) datasets to represent the legitimate baseline traffic behavior of E-Had detection system. We used different DDoS attack profiles of CAIDA dataset to evaluate the proposed E-Had



(a) Normal Traffic Profile of MIT Lincoln

(b) Flash Crowd Profile of FIFA

(c) Constant Rate Attack Profile of CAIDA

(d) Increasing Rate Attack Profile of CAIDA

(e) Pulsating Attack Profile of CAIDA

(f) Subgroup Attack Profile of CAIDA

Fig. 4. Partial Traffic Profiles of various Datasets used.

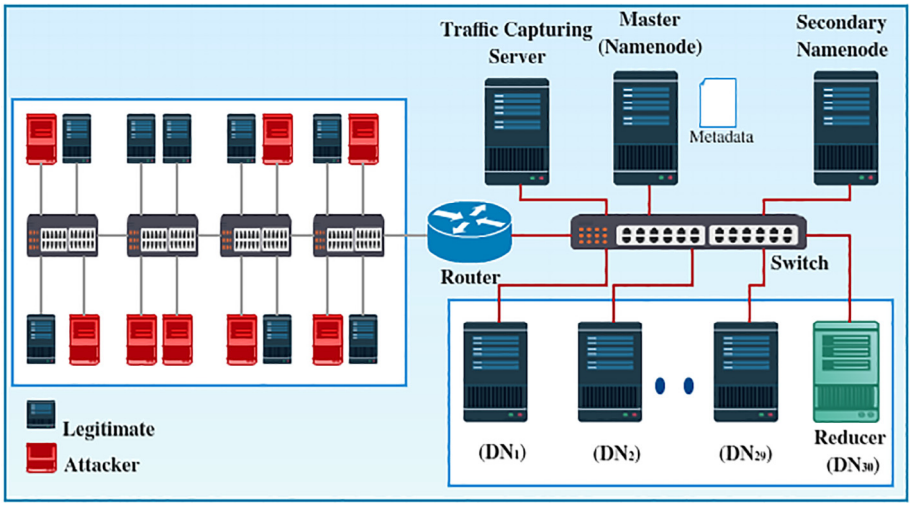


Fig. 5. Hadoop based DDoS Testbed (HA-DDoS).

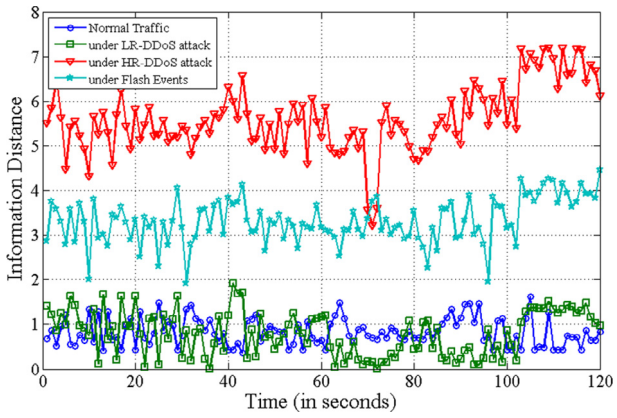


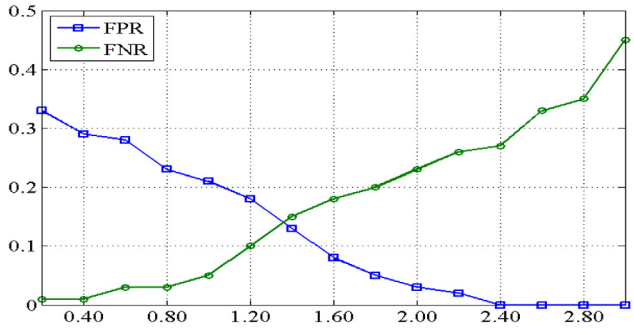
Fig. 6. Temporal Variation in Information Distance values of different network traffic flows in HA-DDoS Testbed.

detection framework. We used four different scenarios of CAIDA dataset namely: increasing rate attack scenario (134936) (Fig. 4d) d) to represent LR-DDoS attack scenario whereas constant rate attack (142936) (Fig. 4c), pulsating attack (143436) (Fig. 4e), and subgroup attack (143936) (Fig. 4e) represent HR-DDoS attack scenario for validating the proposed E-Had detection system. Many authors Bhuyan et al. (2016), Behal et al. (2018), Behal et al. (2018) have used benchmarked attack tools (Kaur et al., 2015) also to generate attack and legitimate traffic in a experimental testbed.

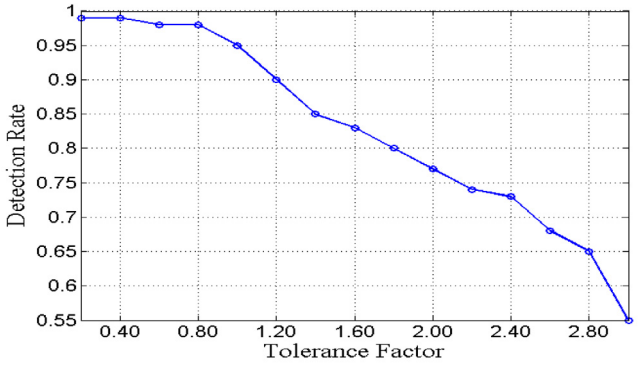
For detecting the variations of network traffic features, we have used information theory based Shannon entropy (She) metric. We compute She values based on the probability distributions of source IP. We observed that the entropy value goes down as compared to normal traffic when number of attack sources are less but they generate a large amount of attack traffic. Whereas it remains in high range, when a large number of attack sources generates small attack traffic.

Table 3
Detection metrics.

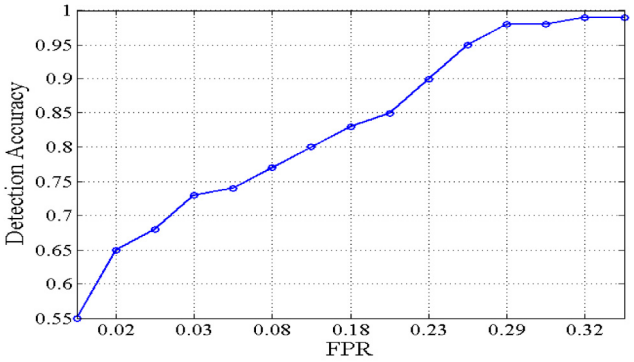
Sr No	Detection Metric	Formula	Description
1	Precision	$P = \frac{TP}{TP+FP}$	measure of fraction of test data being truly detected as attacks
2	Detection Rate	$D_R = \frac{TP}{TP+FN}$	measure of fraction of attack events which are detected correctly
3	FPR	$FPR = \frac{FP}{TN+FP}$	%age of negative events i.e. not attacks that were mistakenly reported as being positive events i.e. attacks
4	True Negative Rate	$TNR = \frac{TN}{TN+FP}$	percentage of the attack packets being dropped
5	Negative Predictive Value	$NPV = \frac{TN}{TN+FN}$	percentage of dropped packets that were reported as attack events
6	F-Measure	$F_m = \frac{2 * P * R}{P+R}$	harmonic mean of detection accuracy and precision
7	F-Measure Complement	$FMC = \frac{2 * TNR * NPV}{TNR+NPV}$	system's success in dealing with attack events
8	Classification Rate	$C_R = \frac{TP+TN}{TP+TN+FP+FN}$	ratio of truly classified events to the total occurred events



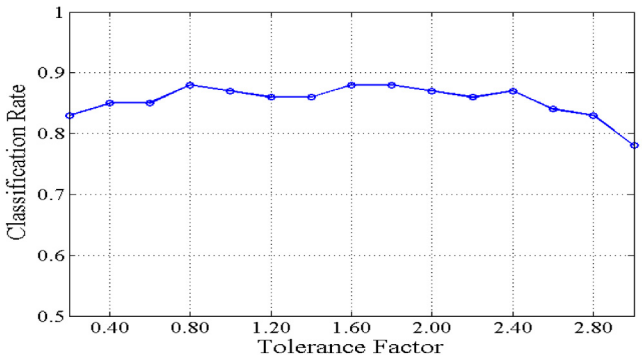
(a) Selection of Optimal Threshold



(b) Variation of Detection rate on different Tolerance Factors



(c) ROC Curve of E-Had



(d) Variation of Classification rate on different Tolerance Factors

Fig. 7. Performance Evaluation of E-Had framework on different Tolerance Factors.

Table 4
Detection Metrics results of E-Had on different scenarios of DDoS attacks.

Attack Scenario	Tolerance Factor	Threshold	DR	P	FPR	TNR	NPV	FM	FMC	CR
LR-DDoS Attack	0.20	1.34	0.99	0.75	0.33	0.68	0.99	0.86	0.80	0.83
	0.40	1.44	0.99	0.77	0.29	0.71	0.99	0.87	0.83	0.85
	0.60	1.54	0.98	0.78	0.28	0.73	0.97	0.87	0.83	0.85
	0.80	1.64	0.98	0.81	0.23	0.78	0.97	0.89	0.86	0.88
	1.00	1.74	0.95	0.82	0.21	0.79	0.94	0.88	0.86	0.87
	1.20	1.84	0.90	0.83	0.18	0.82	0.89	0.86	0.85	0.86
	1.40	1.95	0.85	0.87	0.13	0.88	0.85	0.86	0.86	0.86
	1.60	2.05	0.83	0.92	0.08	0.93	0.84	0.87	0.88	0.88
	1.80	2.15	0.80	0.94	0.05	0.95	0.83	0.86	0.88	0.88
	2.00	2.25	0.77	0.97	0.03	0.98	0.81	0.86	0.88	0.87
	2.20	2.35	0.74	0.98	0.02	0.98	0.79	0.84	0.88	0.86
	2.40	2.45	0.73	1.00	0.00	1.00	0.79	0.85	0.88	0.87
	2.60	2.55	0.68	1.00	0.00	1.00	0.75	0.81	0.86	0.84
	2.80	2.65	0.65	1.00	0.00	1.00	0.74	0.79	0.85	0.83
	3.00	2.75	0.55	1.00	0.00	1.00	0.69	0.71	0.82	0.78
Constant Rate DDoS	0.20	1.34	1.00	0.75	0.33	0.68	1.00	0.86	0.81	0.84
	0.40	1.44	1.00	0.77	0.29	0.71	1.00	0.87	0.83	0.85
	0.60	1.54	1.00	0.78	0.28	0.73	1.00	0.88	0.84	0.86
	0.80	1.64	1.00	0.82	0.23	0.78	1.00	0.90	0.87	0.89
	1.00	1.74	1.00	0.83	0.21	0.79	1.00	0.91	0.88	0.90
	1.20	1.84	1.00	0.85	0.18	0.82	1.00	0.92	0.90	0.91
	1.40	1.95	1.00	0.89	0.13	0.88	1.00	0.94	0.93	0.94
	1.60	2.05	1.00	0.93	0.08	0.93	1.00	0.96	0.96	0.96
	1.80	2.15	1.00	0.95	0.05	0.95	1.00	0.98	0.97	0.98
	2.00	2.25	1.00	0.98	0.03	0.98	1.00	0.99	0.99	0.99
	2.20	2.35	1.00	0.98	0.02	0.98	1.00	0.99	0.99	0.99
	2.40	2.45	1.00	1.00	0.00	1.00	1.00	1.00	1.00	1.00
	2.60	2.55	1.00	1.00	0.00	1.00	1.00	1.00	1.00	1.00
	2.80	2.65	1.00	1.00	0.00	1.00	1.00	1.00	1.00	1.00
	3.00	2.75	1.00	1.00	0.00	1.00	1.00	1.00	1.00	1.00
Pulsating DDoS attack	0.20	1.34	1.00	0.75	0.33	0.68	1.00	0.86	0.81	0.84
	0.40	1.44	1.00	0.77	0.29	0.71	1.00	0.87	0.83	0.85
	0.60	1.54	1.00	0.78	0.28	0.73	1.00	0.88	0.84	0.86
	0.80	1.64	1.00	0.82	0.23	0.78	1.00	0.90	0.87	0.89
	1.00	1.74	1.00	0.83	0.21	0.79	1.00	0.91	0.88	0.90
	1.20	1.84	1.00	0.85	0.18	0.82	1.00	0.92	0.90	0.91
	1.40	1.95	1.00	0.89	0.13	0.88	1.00	0.94	0.93	0.94
	1.60	2.05	1.00	0.93	0.08	0.93	1.00	0.96	0.96	0.96
	1.80	2.15	1.00	0.95	0.05	0.95	1.00	0.98	0.97	0.98
	2.00	2.25	1.00	0.98	0.03	0.98	1.00	0.99	0.99	0.99
	2.20	2.35	1.00	0.98	0.02	0.98	1.00	0.99	0.99	0.99
	2.40	2.45	1.00	1.00	0.00	1.00	1.00	1.00	1.00	1.00
	2.60	2.55	1.00	1.00	0.00	1.00	1.00	1.00	1.00	1.00
	2.80	2.65	1.00	1.00	0.00	1.00	1.00	1.00	1.00	1.00
	3.00	2.75	1.00	1.00	0.00	1.00	1.00	1.00	1.00	1.00
Subgroup DDoS attack	0.20	1.34	1.00	0.75	0.33	0.68	1.00	0.86	0.81	0.84
	0.40	1.44	1.00	0.77	0.29	0.71	1.00	0.87	0.83	0.85
	0.60	1.54	1.00	0.78	0.28	0.73	1.00	0.88	0.84	0.86
	0.80	1.64	1.00	0.82	0.23	0.78	1.00	0.90	0.87	0.89
	1.00	1.74	1.00	0.83	0.21	0.79	1.00	0.91	0.88	0.90
	1.20	1.84	1.00	0.85	0.18	0.82	1.00	0.92	0.90	0.91
	1.40	1.95	1.00	0.89	0.13	0.88	1.00	0.94	0.93	0.94
	1.60	2.05	1.00	0.93	0.08	0.93	1.00	0.96	0.96	0.96
	1.80	2.15	1.00	0.95	0.05	0.95	1.00	0.98	0.97	0.98
	2.00	2.25	1.00	0.98	0.03	0.98	1.00	0.99	0.99	0.99
	2.20	2.35	1.00	0.98	0.02	0.98	1.00	0.99	0.99	0.99
	2.40	2.45	1.00	1.00	0.00	1.00	1.00	1.00	1.00	1.00
	2.60	2.55	1.00	1.00	0.00	1.00	1.00	1.00	1.00	1.00
	2.80	2.65	1.00	1.00	0.00	1.00	1.00	1.00	1.00	1.00
	3.00	2.75	1.00	1.00	0.00	1.00	1.00	1.00	1.00	1.00

As whole of the network traffic reached out at the victim web-server, the corresponding ID values ($|She_C - She_N|$) of different network flows are shown in Fig. 6. The proposed approach is capable enough to differentiate these network flows based on the concept of ID. It has been observed that the ID values of LR-DDoS attack remain in range from 0 to 1.9, ID values of HR-DDoS attack remain in range from 3.14 to 7.17, and ID values of FE traffic remain in range from 1.85 to 4.15.

Further, we measure the effectiveness of the E-Had detection system using various detection system evaluation metrics as given by Sachdeva et al. (2016), Kalkan and Alagöz (2016), Tavallaee et al. (2010) (see Table 3). It is to be noted that a detection system classifies two types of events. If attack is detected, it is referred to as a positive event and if legitimate traffic is detected, it is called a negative event. Based on these two events, different types of detection metrics are derived as shown in Table 3.

The tradeoffs between FPR and detection rate at various tolerance factors is depicted in Fig. 7b. False positive rate (FPR) denotes the detection system effectiveness in detection various anomalies whereas False negative rate (FNR) (1-detection rate) measures

Table 5
Performance E-Had framework in detecting different scenarios of DDoS attacks on optimal threshold.

Attack Scenario	DR	P	FPR	TNR	NPV	FM	FMC	CR
LR-DDoS attack	0.85	0.87	0.13	0.88	0.85	0.86	0.86	0.86
Constant Rate DDoS attack	1.00	0.89	0.13	0.88	1.00	0.94	0.93	0.94
Pulsating DDoS attack	1.00	0.89	0.13	0.88	1.00	0.94	0.93	0.94
Subgroup DDoS attack	1.00	0.93	0.08	0.93	1.00	0.96	0.96	0.96

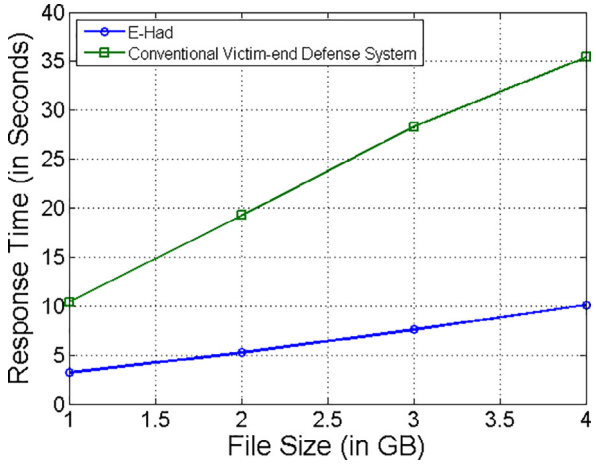


Fig. 8. Comparison of Conventional victim-end and E-Had on processing large files.

the system reliability. We chose the threshold value of the network where FPR and FNR curves intersect. As shown in Fig. 7a, FPR and FNR curves intersect at tolerance factor of 1.40.

Further, the tradeoff between FPR and detection rate (D_R) is shown in receiver operating characteristic curve (ROC) (Fig. 7c). FPR is around 13% at tolerance factor of 1.40. It can be observed that with an increase in FPR, D_R increases i.e. if we compromise on FPR, better D_R can be achieved and vice versa. We chose the threshold of 1.95 at the tolerance factor of 1.40. The values of other parameters used to verify the effectiveness of proposed metrics are shown in Table 4. It has been observed that the proposed E-Had detection system is able to achieve 100% detection accuracy, 89% precision, 88% TNR, 100% NPV, 94% F-Measure, and 94% classification rate at FPR of 13%. (See Table 5).

E-Had framework splits the large volume network traffic files into multiple parts of fixed size (64 MB,128 MB,256 MB) and has the capability to analyze them concurrently by sub dividing the task among several mappers. It is also possible that one datanode may process more than one mapper job simultaneously depending on the number of data blocks present on that datanode. It leads to fast response time and low computational cost to analyze same amount of data as compared to conventional single victim-end defense system. In the present case, we processed network traffic files of size around 1 GB on E-Had. It has been observed that E-Had took only 3.21 s as compared to 10.38 s taken by the conventional victim-end solution to analyze same volume of network traffic flows. Further, the response time of E-Had is also fast as compared with conventional victim-end DDoS defense system as shown in Fig. 8.

7.1. Comparison with existing work

Our proposed Hadoop based victim-end DDoS detection system, E-Had, differs from similar kinds of existing work in many ways.

- Hameed and Ali (2016) and Hameed and Ali (2018) proposed hadoop based HADEC framework for detecting HR-DDoS attacks. Authors proposed a counter based DDoS detection algorithm for detecting four classes of flooding DDoS attacks namely: ICMP, UDP, TCP-SYN, and HTTP-GET flood. They also implemented their detection algorithm using Map-Reduce programming model of Hadoop. The authors chose thresholds of 500 and 1000 packets per seconds to represent legitimate and attack traffic respectively. But nowadays, the volume of network traffic has been increased manifolds. We have extended this work by proposing a distributed detection algorithm using Shannon entropy. Additionally, we have proposed a mathematical model (Eq. (2)) to summarize the results of various mappers at one central reducer node. Our proposed E-Had system is more robust and fault tolerant as it can continue to work even some of the mapper does not provide the partial results to the reducer node in time. Further, E-Had has been evaluated comprehensively on a set of detection system evaluation metrics such as FPR-FNR curve, ROC curve, F-measure, FMC, NPV, TNR and classification rate which the authors of HADEC have not done.
- Choi et al. (2013) proposed a Hadoop based framework to detect HTTP GET flooding attack. The authors used signature based detection system to detect HR-DDoS attacks in the network but signature based detection systems have their own demerits. They can not detect novel or zero day attack whereas our proposed E-Had is anomaly based detection system which can be easily performance tuned to detect novel or zero day attacks. Authors used an experimental setup of around 8 nodes whereas the design of E-Had is more scalable and robust. HA-DDoS testbed is composed of one two-processor 8-core server with 30 datanodes used as mappers.
- Hsieh and Chan (2016) proposed a hadoop based DDoS attack detection system framework using Neural Networks. Their detection system is trained and validated using real dataset of 2000 DARPA LLS-DDoS 1.0 whereas E-Had is validated using more robust experimental setup. We have four different HR-DDoS attack scenarios of benchmarked CAIDA dataset whereas DARPA contains traffic from few attack sources only which are easy to detect. The authors claimed the detection accuracy over 94% to detect HR-DDoS attacks whereas the detection accuracy of E-Had is around 99%.
- Bhuyan et al. (2016) proposed a light-weight detection system (E-LDAT) capable of detecting HR-DDoS attacks. However, they did not detect LR-DDoS attacks and FCs. Besides this, E-LDAT system is deployed at victim-end which has obvious limitations of processing large volume of traffic whereas E-Had can work efficiently even in the case of high-rate network traffic as it distributes the memory and computational overhead of victim-end deployment among n mappers of Hadoop. Further, the detection accuracy of E-LDAT is around 91% whereas detection accuracy of E-Had is 99%.
- Joldzic et al. (2016) proposed a distributed defense system called TIDS (transparent intrusion detection system) for detecting network layer HR-DDoS attacks. To process high rate of traffic, the authors implemented load balancing algorithm in software defined networking (SDN) technique to which most of the companies have not switched over to till date. Whereas E-Had is more practically implementable using the existing infrastructures. TIDS also uses Shannon entropy metric for the

detection of malicious traffic similar to E-Had but additionally, we proposed a mathematical model to automatically summarize the results of various mappers. E-Had distributes the computational overheads to the multiple datanodes called mappers instead of layer-2 switches as done in TIDS that makes the E-Had more computationally efficient.

- Wang et al. (2018) proposed a novel anomaly detection system called SkyShield for detecting HR-DDoS attacks. They handled the large volume of attack traffic by making use of sketch data structure. Authors performed offline analysis of real datasets to validate SkyShield with detection accuracy of 61% whereas E-Had produce detection accuracy of around 99%. SkyShield uses modified Hellinger (HL) distance detection metric which is computationally very expensive as compared to Shannon entropy metric. E-Had distributes the computational complexity to mapper nodes of Hadoop whereas SkyShield is deployed at the victim-end which has the obvious limitations as mentioned before. Furthermore, both SkyShield and E-Had are capable of detecting HR-DDoS attacks launched during FCs with efficacy. SkyShield uses whitelists and blacklists to mitigate the impact of DDoS attacks whereas E-Had uses a captcha module to mitigate the impact of FCs.
- Behal et al. (2018) also proposed a distributed, flexible, automated, and collaborative (D-FACE) defense system which not only distributes the computational and storage complexity to the nearest point of presence (PoPs) routers but also leads to an early detection of DDoS attacks and flash crowds. But the main limitation of D-FACE is from implementation point of view. Because of the non-cooperation among various ISPs and the decentralized nature of Internet, it is not possible to modify the existing network infrastructure such as routers to embed detection logic of D-FACE in realtime whereas E-Had has been implemented using existing infrastructure.

8. Conclusion

One of the biggest challenges in front of the researchers are to detect different types of DDoS attack with high detection rate and quicker response. In this paper, we proposed a Hadoop based distributed, automated and collaborative detection system. To validate the proposed framework, we replayed the network traffic traces of four different attack profiles viz: constant rate attack, increasing rate attack, pulsating attack and subgroup attack of CAIDA dataset in a Hadoop based HA-DDoS testbed. As a part of the work, we proposed a mathematical model to summarize the processed distributed partial results from multiple mapper nodes of Hadoop architecture. Information theory based Shannon metric has been used to differentiate, characterize and classify different types of network traffic flows. The results show that the proposed E-Had detection system is capable of detecting different types of LR-DDoS and HR-DDoS attacks along with differentiating them from similar looking flash crowds. E-Had works in (1) automated way as network flows are assigned to various mapper nodes on the fly without storing in file system (HDFS) Hadoop, (2) collaborative way as various mapper nodes collaborate with each other to send their partial computed results to a designated reducer node for final detection metric computation.

E-had produce 100% detection accuracy, 89% precision, 88% TNR, 100% NPV, 94% F-Measure, and 94% classification rate at FPR of 13%. As part of the future work, the authors shall (1) explore the use of more advanced ϕ-Entropy and other information theory based divergence measures in detecting different types of DDoS attacks, (2) analyze high volume network traffic using Apache Spark and Apache Kafka application tools on Hadoop.

Declaration of Competing Interest

None.

References

Alsirhani, A., Sampalli, S., Bodorik, P., 2018. DDoS attack detection system: utilizing classification algorithms with Apache Spark. In: New Technologies, Mobility and Security (NTMS), 2018 9th IFIP International Conference on. IEEE, pp. 1–7.

Alsirhani, A., Sampalli, S., Bodorik, P., 2018. Ddos detection system: utilizing gradient boosting algorithm and apache spark. In: 2018 IEEE Canadian Conference on Electrical & Computer Engineering (CCECE). IEEE, pp. 1–6.

Apache Hadoop: open-source software for reliable, and distributed computing. URL: https://hadoop.apache.org/ (accessed 30.12.2018)..

Apache Hive. URL:https://hive.apache.org/..

Apache Spark. URL:https://spark.apache.org/..

Apache YARN components. URL:https://www.oreilly.com/library/view/apache-hadooptm-yarn/9780133441925/ch04.html..

Bachupally, Y.R., Yuan, X., Roy, K., 2016. Network security analysis using Big Data technology. In: SoutheastCon, 2016. IEEE, pp. 1–4.

Behal, S., Kumar, K., 2017. Detection of DDoS attacks and flash events using novel information theory metrics. Comput. Netw. 116, 96–110.

Behal, S., Kumar, K., Sachdeva, M., 2018. A generalized detection system to detect distributed denial of service attacks and flash events for information theory metrics. Turk. J. Electr. Eng. Comput. Sci. 26 (4), 1759–1770.

Behal, Sunny, Kumar, Krishan, Sachdeva, Monika, 2018. D-FAC: a novel ϕ-divergence based distributed DDoS defense system. J. King Saud Univ.-Comput. Inf. Sci.

Behal, S., Kumar, K., Sachdeva, M., 2018. D-FACE: an anomaly based distributed approach for early detection of DDoS attacks and flash events. J. Netw. Comput. Appl. 111, 49–63.

Bhandari, A., Sangal, A.L., Kumar, K., 2016. Characterizing flash events and distributed denial-of-service attacks: an empirical investigation. Secur. Commun. Netw. 9 (13), 2222–2239.

Bhuyan, M.H., Bhattacharyya, D., Kalita, J.K., 2016. E-LDAT: a lightweight system for DDoS flooding attack detection and IP traceback using extended entropy metric. Secur. Commun. Netw. 9 (16), 3251–3270.

Chhabra, G.S., Singh, V., Singh, M., 2018. Hadoop-based analytic framework for cyber forensics. Int. J. Commun. Syst. 31, (15) e3772.

Choi, J., Choi, C., Ko, B., Choi, D., Kim, P., 2013. Detecting web based DDoS attack using mapreduce operations in cloud computing environment. J. Internet Serv. Inf. Secur. 3 (3/4), 28–37.

Cui, B., He, S., 2016. Anomaly detection model based on hadoop platform and weka interface. In: Innovative Mobile and Internet Services in Ubiquitous Computing (IMIS), 2016 10th International Conference on. IEEE, pp. 84–89.

Cui, Y., Yan, L., Li, S., Xing, H., Pan, W., Zhu, J., Zheng, X., 2016. SD-Anti-DDoS: fast and efficient DDoS defense in software-defined networks. J. Netw. Comput. Appl. 68, 65–79.

Dayama, R., Bhandare, A., Ganji, B., Narayankar, V., 2015. Secured network from distributed DoS through hadoop. Int. J. Comput. Appl. 118 (2).

FIFA world-cup dataset, 1998. URL:http://ita.ee.lbl.gov/html/contrib/worldcup.html..

Fontugne, R., Mazel, J., Fukuda, K., 2014. Hashdoop: a mapreduce framework for network anomaly detection. In: Computer Communications Workshops (INFOCOM WKSHPS), 2014 IEEE Conference on. IEEE, pp. 494–499.

Github survived biggest DDoS attack ever recorded, wired.com, 2018. URL:https://www.wired.com/story/github-ddos-memcached/ (accessed 29.11.2018)..

Gulisano, V., Callau-Zori, M., Fu, Z., Jiménez-Peris, R., Papatriantafilou, M., Patiño-Martínez, M., 2015. Stone: a streaming DDoS defense framework. Expert Syst. Appl. 42 (24), 9620–9633.

Hadoop Framework: Hdfs and mapreduce. URL:https://www.edureka.co/blog/what-is-hadoop/ (accessed 05.01.2019)..

Hameed, S., Ali, U., 2016. Efficacy of live DDoS detection with hadoop. In: Network Operations and Management Symposium (NOMS), 2016 IEEE/IFIP. IEEE, pp. 488–494.

Hameed, S., Ali, U., 2018. Hadec: Hadoop-based live DDoS detection framework. EURASIP J. Inf. Secur. 2018 (1), 11.

Hadoop distributed file system (HDFS). URL:https://opensource.com/life/14/8/intro-apache-hadoop-big-data..

Hoque, N., Bhuyan, M.H., Baishya, R.C., Bhattacharyya, D., Kalita, J.K., 2014. Network attacks: taxonomy, tools and systems. J. Netw. Comput. Appl. 40, 307–324.

Hsieh, C.-J., Chan, T.-Y., 2016. Detection DDoS attacks based on neural-network using Apache Spark. In: Applied System Innovation (ICASI), 2016 International Conference on. IEEE, pp. 1–4.

Joldzic, O., Djuric, Z., Vuletic, P., 2016. A transparent and scalable anomaly-based DoS detection method. Comput. Netw. 104, 27–42.

Kalkan, K., Alagöz, F., 2016. A distributed filtering mechanism against DDoS attacks: Scoreforcore. Comput. Netw. 108, 199–209.

Kaur, H., Behal, S., Kumar, K., 2015. Characterization and comparison of distributed denial of service attack tools. In: 2015 International Conference on Green Computing and Internet of Things (ICGCIoT). IEEE, pp. 1139–1145. https://doi.org/10.1109/ICGCIoT.2015.7380634.

Khattak, R., Bano, S., Hussain, S., Anwar, Z., 2011. Dofur: DDoS forensics using mapreduce. In: Frontiers of Information Technology (FIT). IEEE, pp. 117–120.

Kumar, K., Joshi, R., Singh, K., 2007. An isp level distributed approach to detect DDoS attacks. In: Innovative Algorithms and Techniques in Automation, Industrial Electronics and Telecommunications. Springer, pp. 235–240.
Kune, R., Konugurthi, P.K., Agarwal, A., Chillarige, R.R., Buyya, R., 2016. The anatomy of big data computing. Software: Practice Exp. 46 (1), 79–105.
Lee, Y., Lee, Y., 2011. Detecting DDoS attacks with hadoop. In: Proceedings of The ACM CoNEXT Student Workshop. ACM, p. 7.
Lee, Y., Lee, Y., 2013. Toward scalable internet traffic measurement and analysis with Hadoop. ACM SIGCOMM Comput. Commun. Rev. 43 (1), 5–13.
Lin, J.-C., Lee, M.-C., 2016. Performance evaluation of job schedulers on Hadoop YARN. Concurr. Comput.: Practice Exp. 28 (9), 2711–2728.
Maheshwari, V., Bhatia, A., Kumar, K., 2018. Faster detection and prediction of DDoS attacks using mapreduce and time series analysis. In: Information Networking (ICOIN), 2018 International Conference. IEEE, pp. 556–561.
Mausezahn – fast traffic generator. URL:http://man7.org/linux/man-pages/man8/mausezahn.8.html..
MIT Lincoln Laboratory LLSDDoS 1.0 dataset, 1998. URL:https://www.ll.mit.edu/ideval/data/2000/llsddos1.0.html..
Predict protected repository for the defense of infrastructure against cyber threats. URL:https://apps.dtic.mil/docs/citations/AD1050331 (accessed 03.01.2019)..
Rathore, M.M., Ahmad, A., Paul, A., 2016. Real time intrusion detection system for ultra-high-speed big data environments. J. Supercomput. 72 (9), 3489–3510.
Sachdeva, M., Kumar, K., Singh, G., 2016. A comprehensive approach to discriminate DDoS attacks from flash events. J. Inf. Secur. Appl. 26, 8–22.
Suryawanshi, D., Mande, U., 2013. Traffic measurement and analysis with Hadoop. Int. J. Eng. Res. Technol. 2 (10).
Tavallaee, M., Stakhanova, N., Ghorbani, A.A., 2010. Toward credible evaluation of anomaly-based intrusion-detection methods. IEEE Trans. Syst. Man Cybern. Part C 40 (5), 516–524.
Terzi, D.S., Terzi, R., Sagiroglu, S., 2017. Big data analytics for network anomaly detection from netflow data. In: Computer Science and Engineering (UBMK), 2017 International Conference. IEEE, pp. 592–597.
Tian, G., Wang, Z., Yin, X., Li, Z., Shi, X., Lu, Z., Zhou, C., Yu, Y., Guo, Y., 2015. Mining network traffic anomaly based on adjustable piecewise entropy. In: Quality of Service (IWQoS), 2015 IEEE 23rd International Symposium on. IEEE, pp. 299–308.
Wang, F., Wang, H., Wang, X., Su, J., 2012. A new multistage approach to detect subtle DDoS attacks. Math. Comput. Modell. 55 (1–2), 198–213.
Wang, C., Miu, T.T., Luo, X., Wang, J., 2018. Skyshield: a sketch-based defense system against application layer DDoS attacks. IEEE Trans. Inf. Forensics Secur. 13 (3), 559–573.
Arbor Network's Worldwide Infratsructure Security Report (WISR), 2018..
Zhang, J., Liu, P., He, J., Zhang, Y., 2016. A Hadoop based analysis and detection model for IP spoofing typed DDoS attack. In: Trustcom/BigDataSE/ISPA, 2016 IEEE. IEEE, pp. 1976–1983.
Zhao, T., Lo, D.C.-T., Qian, K., 2015. A neural-network based DDoS detection system using Hadoop and Hbase. In: High Performance Computing and Communications (HPCC), 2015 IEEE 7th International Symposium on Cyberspace Safety and Security (CSS), 2015 IEEE 12th International Conferen on Embedded Software and Systems (ICESS), 2015 IEEE 17th International Conference on. IEEE, pp. 1326–1331.
Zhou, X., Petrovic, M., Eskridge, T., Carvalho, M., Tao, X., 2014. Exploring Netfow data using Hadoop. In: Proceedings of the Second ASE International Conference on Big Data Science and Computing.

Mr. Nilesh Vishwasrao Patil has done Bachelor of Engineering in Computer Engineering from North Maharashtra University, Jalgaon in 2008. He finished his masters in Computer Engineering from Savitribai Phule Pune University, Pune in 2015. He is full-time Ph.D. Research Scholar of Panjab University, Chandigarh at National Institute of Technical Teachers Training and Research, Chandigarh, India and have published 06 papers in different International Journals and Conferences.

Dr. Rama Krishna received B.Tech. from JNTU, Hyderabad, M.Tech. from Cochin University of Science & Technology, Cochin, and Ph.D from IIT, Kharagpur. He is Senior Member, IEEE, USA. Since 1996, he is working with Department of Computer Science & Engineering, National Institute of Technical Teachers' Training & Research, Chandigarh and currently holding the position of Professor. His areas of research interest include Computer Networks, Wireless Networks, Cryptography & Cyber Security, and Cloud Computing. To his credit, he has more than 80 research publications in referred International and National Journals and Conferences. He acted as Associate Editor for International Journal of Technology, Knowledge and Society. He is a member in Advisory/Technical committees of many national and international journals and conferences and also chaired many technical sessions. He is reviewer of Elsevier Journal of Vehicular Communications, Elsevier Journal of Computers & Security, Elsevier Journal of Information and Software Technology, IEEE Access. He has 22 years of experience in organizing more than 100 training programmes in the upcoming areas of CSE and IT for the faculty of engineering colleges, polytechnics and industry professionals. He is instrumental in launching various initiatives at NITTTR Chandigarh towards paperless office.

Dr. Krishan Kumar has done Bachelor of Technology in Computer Science and Engineering from National Institute of Technology, Hamirpur in 1995. He finished his Masters in Software Systems from BITS Pilani in 2001. He finished his Ph. D. from Department of Electronics and Computer Engineering at Indian Institute of Technology, Roorkee in 2008. His general research interests are in the areas of Information Security and Computer Networks. He has published around 200+ research papers in different International Journals and Conferences of Repute.

Dr. Sunny Behal has done Bachelor of Technology in Computer Science and Engineering from SBS State Technical Campus, Ferozepur, Punjab, India in 2002. He finished his Masters in Computer Science and Engineering from Guru Nanak Dev Engineering College, Ludhiana, Punjab, India in 2010. He completed his PhD in 2018 and currently, working as Assistant Professor at SBS State Technical Campus, Ferozepur, India. His research interests include Botnet detection, DDoS Defense, Information and Network Security, Digital Information processing. He has published around 60+ Research papers in different International Journals and Conferences of repute. He is reviewer of Elsevier ADHOC Networks, Elsevier Computers and Security, IEEE Communication Surveys and Tutorials, Wiley Security and Communication Networks, Thomsen Reuters, Australasian Journal of Information Systems, Thomsen Reuters, E-SCI \item International Journal of Network Security, SCOPUS, DBLP Indexed \item Journal of Bulletin of Electrical Engineering and Informatics, Indonesia, SCOPUS Indexed \item International Journal of Electrical and Computer Engineering, Indonesia, SCOPUS Indexed.