Received: 16 August 2020 | Revised: 18 December 2020 | Accepted: 1 January 2021

DOI: 10.1002/cpe.6197

RESEARCH ARTICLE

WILEY

Distributed frameworks for detecting distributed denial of service attacks: A comprehensive review, challenges and future directions

Nilesh Vishwasrao Patil[1] | **C. Rama Krishna**[1] | **Krishan Kumar**[2]

[1]Department of Computer Science and Engineering, National Institute of Technical Teachers Training and Research (NITTTR), Chandigarh (Established by Ministry of Human Resource Development, Govt. of India), Chandigarh, India

[2]Department of Information Technology, University Institute of Engineering & Technology (UIET), Panjab University, Chandigarh, India

Correspondence
Nilesh Vishwasrao Patil, Department of Computer Science and Engineering, National Institute of Technical Teachers Training and Research (NITTTR), Chandigarh (Established by Ministry of Human Resource Development, Govt. of India), Chandigarh, India.
Emails: patil.nilesh38@gmail.com; nilesh.cse18@nitttrchd.ac.in

Abstract

A distributed denial of service (DDoS) attack is a significant threat to web-based applications and hindering legitimate traffic (denies access to benign users) by overwhelming the victim system or its infrastructure (service, bandwidth, networking devices, etc.) with a large volume of attack traffic. It leads to a delay in responses or sometimes a crash victim system. Even a few moments of pause in web-based applications lead to a huge monetary loss and a bad reputation in the market. Several approaches available in the literature to protect websites from different types of DDoS attacks. However, incidents and volume sizes of DDoS attacks are growing quarter by quarter. Further, various challenges in the traditional framework based defense mechanisms: itself becoming a victim of attacks while analyzing a massive amount of traffic, require more time for detection process, no coordination among the modules, etc. This paper presents a comprehensive DDoS defense deployment taxonomy and critically reviewed existing distributed frameworks based DDoS attack detection systems. Further, characterized several existing distributed processing frameworks to select an appropriate one for deploying DDoS attack detection mechanisms. Finally, several evaluation metrics, open issues, discussion on available datasets including their limitations, and future directions are presented.

KEYWORDS

Big data, Distributed denial of service (DDoS) attacks, DDoS defense deployment taxonomy, Distributed processing framework

1 | INTRODUCTION

Over the last two decades, every organization has been moving its business online for increasing revenue and is accessible 24/7 to consumers from anywhere. There is a huge growth in Internet users (approximately four billion six hundred forty-eight million peoples using the Internet[1]) and Internet of Things (IoT) devices (approximately every second 127 devices connected to the Internet[2]). However, this remarkable growth brought vulnerable network paths, less-secure IoT devices, increased risk of distributed denial of service (DDoS) attacks, etc. Therefore, attackers take this as an opportunity to compromise more number of devices for launching a large-scale DDoS attack.

1.1 | DDoS attack

A DDoS attack is the most influential threat to online services and immediately overwhelms victims by sending a huge volume of irrelevant traffic.[4] Therefore, several hurdles: recognizing DDoS attacks with higher accuracy, real-time traffic analysis, immediate recovery, etc. The primary objective of DDoS attacks is to hide victim services from legitimate users.[5] To perform this type of attack, an attacker firstly compromises numerous devices (such as computers, mobiles, IoT devices, etc.), then concurrently transfers attack traffic towards victims from each attack source. A typical DDoS attack setup is shown in Figure 1. In this, a master (a.k.a attacker) node compromises several devices (a.k.a slaves, zombies, bots). Handlers are intermediate programs between masters and slaves for controlling the attacker's army to perform large-scale DDoS attacks on the victim in a coordinated manner. Further, DDoS attacks need to distinguish from legitimate flash events. A flash event is a situation in which the web-server receives numerous requests from legitimate sources due to breaking news happens around the world. Both flash events and DDoS attacks have shared several similar features. Therefore, it is a challenging job to identify attacks during flash events. However, both situations require distinct actions: (i) For DDoS attacks, filter traffic from each attack source, and (ii) For the flash event, add extra servers for providing service to each user.

1.2 | Statistical information of DDoS attacks incidents

Human life has completely shifted to the online since January 2020 due to the COVID-19 (Corona Virus Disease) pandemic situation throughout the world. During this lockdown period, people's learn, work, fun, shopping, etc. activities are carried over in online mode. According to Kaspersky Q1-2020 DDoS attacks report,[6] attackers mostly targeted pharmaceuticals organizations, food delivery, distribution services, entertainment, and learning platforms. Popular incidents of Q1-2020 are:

1. In March 2020, attackers attempted to target the Health and Human Service (HHS) online service of the U.S. Department. However, they failed to shutdown the HHS service. It has been running before and after attacks, despite the immense load.
2. In this pandemic, attackers launched attacks on two online food delivery organizations: Lieferando (Germany) and Thuisbezorgd (Netherlands). During attacks, both services can accept orders but failed to process requests and refunded money. Attackers demanded 2 BTC (approximately 13,000 dollar) from Lieferando for halting ongoing DDoS attacks.
3. In February 2020, Amazon Web Services experienced a significant DDoS attack in history, which is almost 2.3 Tbps.[7] They have defended such a large-scale attack, previously Github sustained by 1.35 Tbps DDoS attack in February 2018.[8]

According to the statistical report of Q1-2020, it reveals that attackers also showing their interest to deploy Windows operating system-based botnets for launching attacks. The share of Windows botnets shifted from 2.6% (Q4-2019) to 5.64% (Q1-2020). However, 9 out of 10 DDoS attacks still proceed using Linux botnets (94.36%).

The 2019 Kaspersky statistics report[10] shows that the number of DDoS attacks has been grown by 84% compared to the Q4-2018 DDoS attacks. The Q2 is usually milder than the Q1 in case of DDoS attacks, but 2020 is an exception. On April 9, 2020, registered 298 DDoS attack

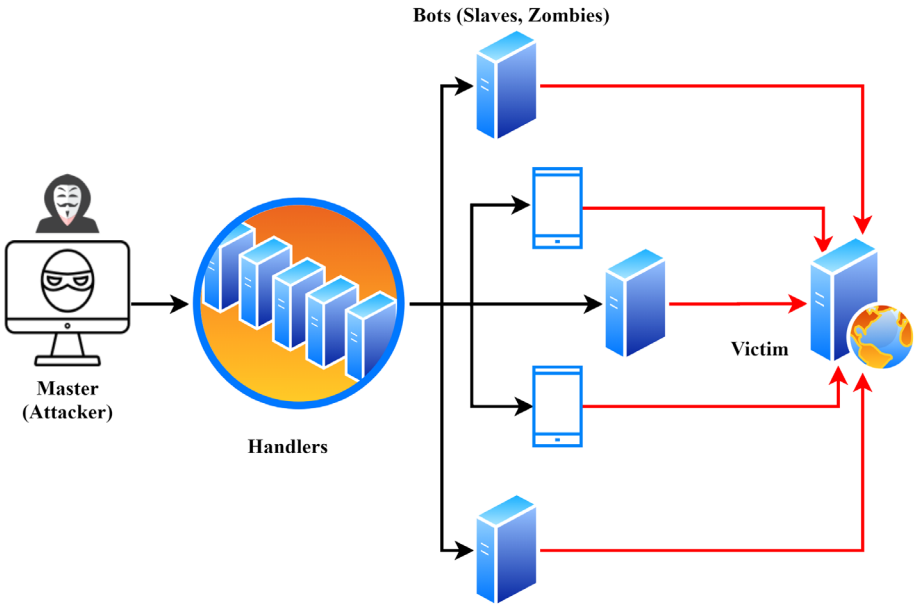

FIGURE 1 A typical distributed denial of service attack setup[3]

incidents in a single day.[9] The measurement of the total number of attacks and durations of Q2-2019, Q1-2020, and Q2-2020 shown in Figure 2. The long-lasting DDoS attack registered in Q4-2018[11] and resuming for nearly 329 h. The first incident of DDoS attack observed in June–July 1999 and reported in August 1999. According to recent statistical bulletins, DDoS attack events on websites signifies that attack incidents are not only increasing but also growing in the volume size (attack pattern shift from Gbps to Tbps). The year-wise growth of DDoS attacks shown in Figure 3.

1.3 | Challenges

In the literature, numerous mechanisms proposed by researchers to protect victims from different types of DDoS attacks. However, sometimes detection mechanisms itself becoming victims during a large-scale DDoS attack. Therefore, it is necessary to deploy a mechanism on distributed processing frameworks (such as Apache Hadoop, Spark, Storm, Kafka, etc.) for efficiently analyzing a large amount of traffic on a cluster of nodes. In the available literature, fewer mechanisms deployed on a distributed framework compared to the traditional framework. Therefore, several critical challenges in front of researchers are: (i) Identify DDoS attack with higher detection accuracy and low false-positive rate, (ii) Distinguish between DDoS attacks and flash events, (iii) Real-time analysis and response, (iv) Efficiently analyze a massive amount of traffic flows, etc.

1.4 | Contributions

The significant contributions of this paper summarized as follows:

1. Proposed a DDoS defense deployment taxonomy and examined each class of taxonomy.
2. Critical analysis of existing distributed processing frameworks (such as Apache Hadoop, Spark) based DDoS detection systems.

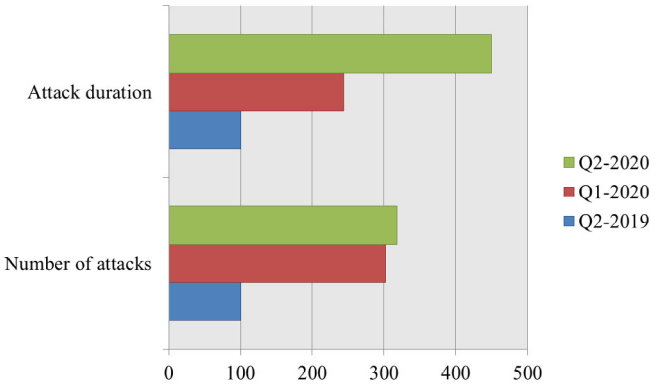

FIGURE 2 Comparison of the total number of distributed denial of service attacks and attack duration between Q2-2019, Q1-2020, and Q2-2020 (100% reference value considered for Q2-2019)[9]

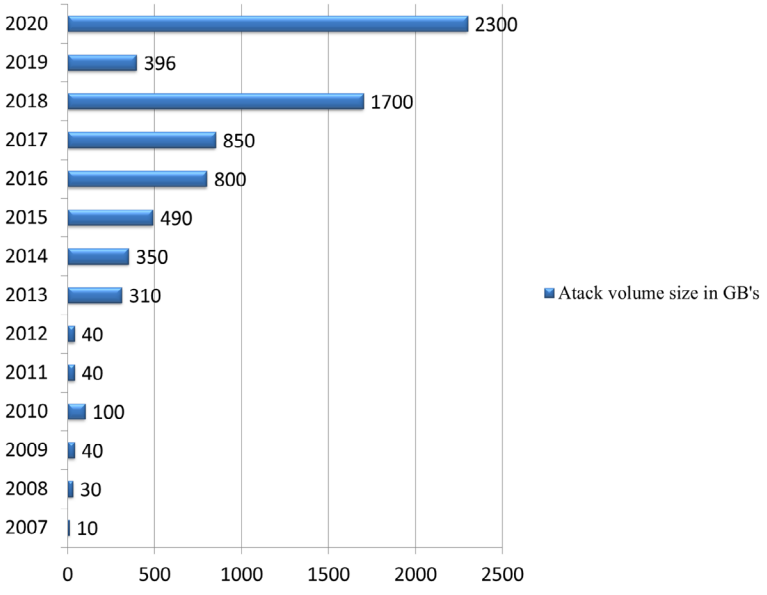

FIGURE 3 Year-wise large distributed denial of service attack w.r.t. volume size

TABLE 1 Summary of prior reviews

Authors (year)	Primary focus	DDoS taxonomy	Characterization of DPF	Evaluation metrics	Research gaps	DDoS datasets	Future directions
Prasad et al.[24]	Application level DDoS attacks	✓	×	✓	✓	✓	✓
Manavi et al.[13]	Application & Network level DDoS attacks	✓	×	×	✓	×	✓
Joelle et al.[25]	DDoS attacks in SDN	✓	×	×	✓	×	✓
Dong et al.[26]	DDoS attack in SDN & Cloud environments	✓	×	×	✓	×	✓
Tandon et al.[27]	Low-rate DDoS attacks	✓	×	×	✓	×	×
Singh et al.[29]	DDoS attacks in SDN	✓	×	×	✓	×	✓
Our work	Distributed Processing Framework- (DPF) based DDoS techniques	✓	✓	✓	✓	✓	✓

Abbreviations: DDoS, distributed denial of service.

3. Characterization of existing distributed processing frameworks to select an appropriate one for deploying the proposed DDoS detection approach.
4. Presented a set of standard metrics for evaluation of DDoS defense mechanisms.
5. Highlighted open issues and future research directions with respect to distributed frameworks based defense mechanisms.
6. Summarized available datasets with their limitations.

1.5 | Prior reviews

Several review papers are available in the literature,[3,12-29] but no one has specifically addressed distributed processing framework-based DDoS attack detection mechanisms. In this paper, we presented a comprehensive study with the review of distributed processing frameworks-based mechanisms, taxonomy, characterization of various distributed processing frameworks, set of evaluation metrics, discussion on available datasets, etc. In Table 1, we summarized a few review papers and compared them on several parameters with our paper.

1.6 | Organization of paper

The rest of the paper is discussed as follows: Section 2 presents a DDoS defense deployment taxonomy with the viability comparison of each class. Section 3 presents a review of the existing distributed processing framework based DDoS detection system. Section 4 presents characterization of the existing distributed processing frameworks for selecting an appropriate framework for deploying the DDoS attack detection mechanisms. A set of standard evaluation metrics presented in section 5. Section 6 highlights open issues and discussion on available datasets. Finally, section 7 concludes the paper with future directions.

2 | A DDOS DEFENSE DEPLOYMENT TAXONOMY

In the literature, several defense systems proposed by researchers to protect victim services and resources from different types of DDoS attacks. According to References 12,13, deployment locations for defense systems can be categorized into four types: source-end, intermediate, victim-end, and hybrid. Sachdeva et al.[30] recommended the distributed internet service provider (ISP) domain DDoS deployment because of several reasons: (i) Distributed environment, (ii) Practically feasible solution, (iii) Robust implementation, (v) Autonomous control, etc. Further, few researchers[31-43] proposed distributed processing framework-based DDoS detection systems. This type of deployment not only quickly recognizes attacks but also efficiently explores a huge volume of traffic by employing a cluster of slaves in a coordinated manner. Therefore, it is necessary to consider distributed processing frameworks-based deployment while proposing a taxonomy for DDoS defense deployment. In this section, we propose a DDoS defense deployment taxonomy and presented in Figure 4. We broadly classify DDoS defense deployment into two groups with their design frameworks: traditional and distributed processing frameworks-based DDoS defense deployment.

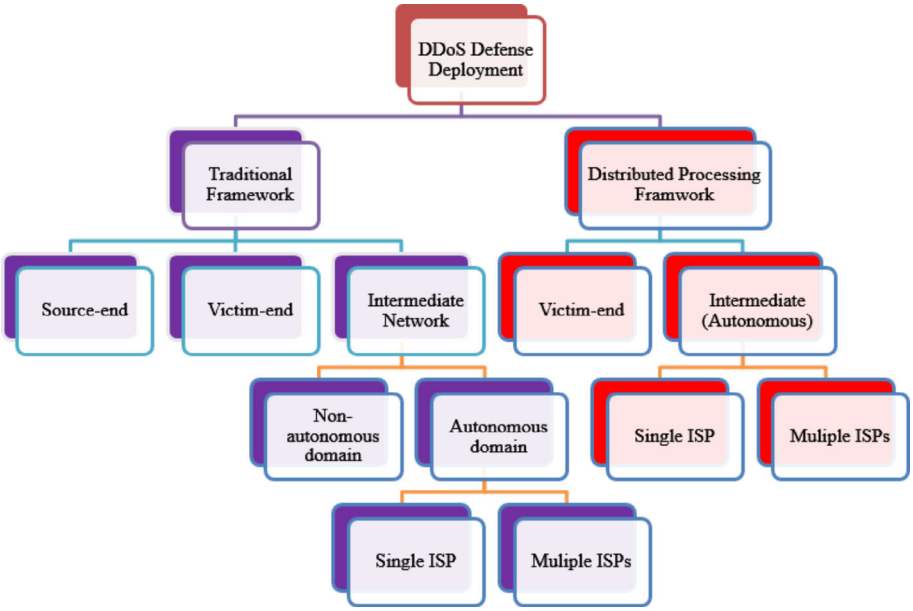

FIGURE 4 A proposed distributed denial of service defense deployment taxonomy

2.1 | Traditional framework-based DDoS defense deployment

In the case of traditional frameworks, the defense system deployed on a single machine to perform various jobs: filtering, detection, characterization, and mitigation. Sometimes, this type of defense system deployed on multiple machines, even though each machine performed its job without communicating between them while analyzing network traffic. The traditional framework-based DDoS defense deployment group can be categorized into three categories: victim-end, source-end, and intermediate (nonautonomous domain and autonomous domain). Each class of traditional frameworks-based DDoS defense deployment summarized in Table 2. There are several challenges in the traditional framework-based DDoS deployment model:[5,43,54,91,92]

1. The traditional framework-based DDoS defense system can deploy on single or multiple machines. However, no communication between them while analyzing network traffic flows.
2. Each machine performs its task separately. Therefore, it affects the detection accuracy and false-positive rate.
3. There is no central coordinator to employ each resource efficiently (no separate thread for resource allocation).
4. Sometimes, the traditional framework-based system itself becomes the victim of an attack while analyzing a massive volume of traffic (due to attack pattern shifted from Tbps to Gbps).
5. The source-end and intermediate network (nonautonomous domain) deployments are challenging tasks for real-time implementation. Further, these are expensive solutions.

2.2 | Distributed processing framework-based DDoS defense deployment

In the literature, few researchers proposed mechanisms deployed on distributed frameworks (Hadoop and Spark) to protect victims from DDoS attacks. A distributed processing framework itself has a distributed design for analyzing and storing a large amount of data (in this case, network traffic) on a cluster of slaves. This type of mechanism is deployed on a group of nodes (based on master–slave architecture) for performing detection jobs efficiently. The complete detection job systematically distributed between nodes and each node sufficiently communicates with each other while analyzing network traffic. It will help to enhance the detection accuracy and minimize false-positive rates. This type of deployment is classified into two classes, such as victim-end and intermediate network deployment. The source-end and nonautonomous domain deployments are not a feasible solution using this type of framework because it requires autonomy on networks. Table 3 presents the features of each class.

References 31-42, proposed Hadoop- and Spark-based distributed DDoS attack detection mechanisms. A victim-end deployment is one of the best choice to implement distributed processing frameworks-based mechanisms because of several reasons:

1. It is deployed near to the victim system or in victim networks. Therefore, it closely observed network traffic, which helps to improve detection accuracy.
2. Victim-end deployment is considerably easy to implement due to full control of the networks.

TABLE 2 Description of each class of the traditional framework-based distributed denial of service (DDoS) defense deployment (presented in Figure 4)

Traditional framework based DDoS deployment	Features
Source-end[44-47]	• Easy to control flooding DDoS attack traffic in attack source networks itself • Defense systems deployed in edge routers of autonomous systems (AS) in attack source networks • Help to perform defense actions on DDoS attacks traffic using filtering and rate-limiting techniques • Ability to detect DDoS attacks traffic before collateral damage • Practically challenging task to implement
Victim-end[4,5,45,48-54 55-64 65-73 74-83]	• Defense systems deployed in the victim network either on the detection server or edge router of AS • Closely observes the network traffic • Practically feasible and cost-effective solution • Easy to discriminate legitimate and DDoS attacks traffic flows • Sometimes, defense systems itself become the victim of an attack while analyzing massive traffic (attack pattern shifted from Gbps to Tbps) • Difficult to implement filtering and rate-limiting techniques
Intermediate network[84-90]	• Defense systems deployed in intermediate networks (single or multiple ISPs) on multiple routers of AS • Help to filter DDoS attacks traffic using various filtering techniques • Help to a trade-off between detection accuracy and bandwidth consumption • Further classified into Nonautonomous and Autonomous domain (A) Nonautonomous Domain: • Deploy at multiple locations in decentralized Internet architecture (Nonautonomous control). • Deployment in the nonautonomous domain is a challenging task in real-time • Easy to detect DDoS attack traffic with implementation of filtering techniques (B) Autonomous Domain: • Deploy at multiple locations (ISP domain or multiple ISP's) in autonomous control domain. • Further classified into single and multiple ISP domain (B1) Single ISP: • Deploy at multiple location in ISP domain • Practically feasible for real-time implementation & cost-effective • Provide higher detection accuracy (B2) Multiple ISP: • Deploy at multiple location in multiple ISP's • Practically challenging for real-time implementation & expensive • Provide higher detection accuracy and mitigate impact of ongoing attack

TABLE 3 Description of each class of distributed framework-based distributed denial of service (DDoS) defense deployment (presented in Figure 4)

DPF-based DDoS defense deployment	Features
Victim-end[31-42]	• Defense systems deployed in victim networks and it closely observes network traffic • Feasible for real-time implementation • Easy to identify malicious instances • Efficiently analyze a large volume of traffic instances on a cluster of nodes
Intermediate[43]	• Deploy on a cluster of nodes in intermediate networks • Further classified into Single ISP and Multiple ISPs (A) ISP level • Deploy on a cluster of nodes in a single ISP domain • Feasible for real-time implementation • Efficiently analyze a large volume of traffic instances on a cluster • Early detection of attack flows (B) Multiple ISPs • Deploy on a cluster of nodes in multiple ISPs • Analyze a big volume of traffic flows by employing cluster of nodes. • Mitigate impact of DDoS attack before damage network. • Practically challenging for real-time implementation & expensive.

Abbreviation: DPF: Distributed Processing Framework.

3. It is a cost-effective solution compared to others.
4. A feasible solution for real-time implementation.
5. It does not depend on the cooperation between different ISPs.
6. It receives aggregated network traffic flows for analysis, which helps to minimize false-positive rates.
7. Distributed processing framework (Apache Hadoop[93,94]) efficiently analyzes and stores a massive amount of data (network traffic flows) using a cluster of nodes. Further, it will not perform well while analyzing a small amount of data. Therefore, this type of framework is not suitable for source-end deployment.
8. In traditional framework-based victim-end DDoS detection received a massive (aggregated) amount of network traffic to analyze, and sometimes systems itself becomes a victim of an attack while analyzing such massive traffic. Therefore, one of the best distributed approach is to implement a victim-end DDoS mechanisms and most of researchers[31-42] have used this type deployment.

We present the viability of DDoS defense deployment with parameters: detection, filtering, robustness, implementation, and accuracy in Table 4.

TABLE 4 Viability of distributed denial of service defense deployment taxonomy (presented in Figure 4)

Framework	Deployment class	Detection	Filtering	Robustness	Implementation	Accuracy
TF	Source-end	Level 5	Easy	High	Level 5	Low
	Victim-end	Easy	Level 5	Low	Easy	High
	Intermediate: Nonautonomous	Difficult	Easy	High	Challenging	Low
	Intermediate:Single ISP (Auto.)	Easy	Challenging	Medium	Difficult	High
	Intermediate:Multiple ISPs (Auto.)	Difficult	Difficult	High	Challenging	Medium
DPF	Victim-end	Easy	Difficult	High	Easy	High
	Intermediate: Single ISP	Easy	Difficult	High	Difficult	High
	Intermediate:Multiple ISPs	Difficult	Easy	High	Challenging	Medium

Abbreviations: Auto., Autonomous; DPF, Distributed Processing Framework; TF, Traditional Framework.

TABLE 5 Examination of each class of proposed distributed denial of service (DDoS) defense deployment using performance measurement metrics

Framework	Performance measurement metrics	Strength	Compromise ability	Efficient traffic analysis	Resource utilization	Economical
TF	Source-end	Low	Low	×	×	×
	Victim-end	High	High	×	×	✓
	Intermediate:Nonautonomous	Low	Low	×	×	×
	Intermediate: Single ISP (Auto.)	High	Medium	×	×	✓
	Intermediate: Multiple ISPs (Auto.)	Medium	Low	×	×	×
DPF	Victim-end	High	Low	✓	✓	✓
	Intermediate: Single ISP	High	Low	✓	✓	✓
	Intermediate: Multiple ISPs	Medium	Low	✓	✓	×

Abbreviations: Auto: Autonomous; DPF: Distributed Processing Framework; TF, Traditional Framework.

2.3 | Performance measurement of proposed a DDoS defense deployment

Reference 88 presented the performance measurement parameters for DDoS defense deployment. The essential performance measurement metrics are listed as follows:

1. Strength: The system either identify or miss attacks from incoming traffic. Therefore, the strength of the systems is measured based on correctly or wrongly classified legitimate and attack instances. Four important parameters required for measuring strength: TP, FP, TN, and FN.
2. Compromise ability: Could the defense system itself become victim DDoS attacks while analyzing a large amount of network traffic flows?
3. Traffic analysis: As per recent DDoS attacks statistical reports, DDoS attacks volume size is regularly increasing. Therefore, there is necessary to analyze a large amount of network traffic efficiently for responding quickly.
4. Resource utilization: Defense systems deployed on multiple machines and concurrently performing their jobs. Therefore, it is necessary to employ each resource efficiently.
5. Economical:The deployment cost of defense systems should be affordable.

In Table 5, we examine each class of proposed DDoS defense deployment taxonomy based on performance measurement metrics.

3 | REVIEW OF EXISTING DISTRIBUTED PROCESSING FRAMEWORK-BASED DDOS DETECTION SYSTEM

In the literature, researchers proposed numerous mechanisms to protect victims from different types of DDoS attacks. Peng et al.[14] classified defense systems into four types: prevention, detection, traceback, and mitigation. However, they did not consider the deployment framework of the mechanisms while classifying systems. Therefore, it can be categorized into two classes on the basis of deployment framework: traditional and distributed frameworks. In this section, our primary focus is to analyze distributed frameworks-based mechanisms.

In Table 6, we summarize the existing distributed mechanisms. Several researchers[3,12-23] systematically analyzed the traditional framework-based mechanisms. However, no one specifically addressed the distributed processing framework-based DDoS mechanisms.

Few authors[31-43,91,92,95,96] proposed distributed framework-based mechanisms and most of them deployed their mechanism on the Apache Hadoop. Apache Hadoop distributed framework follows a two-step process to analyze data efficiently. The first step stores data into the Hadoop distributed file system (HDFS). While storing data in the HDFS, data split-up into multiple blocks with equal sizes: 64, 128 (default), or 256 MB and stores these blocks on a cluster of nodes. Further, the Namenode (master) manages a metadata file regarding replication and storing locations of each block. The second step is to analyze data using the MapReduce programming model. This step divided into two jobs: mapper and reducer job. In the mapper job, multiple mappers executed to process data blocks. Each block requires separate mappers. It can possible a single node can execute multiple mappers based on a number of related blocks stored on that node. Further, mappers are sharing intermediate results (shuffling) between them by putting results into the HDFS. In the reducer job, reducing (combining) results of each mapper and stored reducer results in HDFS. Therefore, this type of detection mechanism does not only protect victims from attacks but also analyze a large volume of traffic flows

TABLE 6 Review of existing distributed distributed denial of service (DDoS) attack detection systems

Authors	DPF-based deployment	Experimental setup	Detection metrics and methodology	Dataset used
Lee and Lee[31]	Victim-end	1+10 (Master+slaves) Configurations:Quad -core:2.93GHz, Intel i7 RAM: 16 GB, HDD:1TB	Detection metrics: Page request rate Time interval, Threshold value unbalanced ratio, and access pattern	Offline: 500 GB 1 TB traces 500 GB and used.
Khattak et al.[32]	Victim-end	—	Horizontal and Vertical threshold (number of ingress and egress requests)	MIT LLS-DDoS-1.0
Zaho et al.[33]	Victim-end	—	Neural Network: three parameters CPU usage, avg. packet size, number of TCP connections.	—
Dayama et al.[34]	Victim-end	1+10(Master+slaves) Config.: Quadcore:2.93GHz,Intel:i7 RAM: 16GB HDD:1TB	Access Pattern based method used. Detection metrics: time-interval, threshold, and unbalanced ratio.	—
Hameed et al.[35,36]	Victim-end	One-Capturing Server One-Detection Server 10-slaves.	Counter based algorithm used. Metric: number of flows per source IP, threshold value: 500 and 1000.	Mausezahn tool used.
Hsieh et al.[37]	Victim-end	—	Neural network: Packet. header features used. Train and test: real dataset used.	MIT 2000-DARPA LLS-DDoS-1.0
Alsirhani et al.[38,39]	Victim-end	01+03 (Master+slaves) Core i7/5,RAM:2/ 16GB,HDD:500GB	1. Classification algorithm used. 2. Fuzzy logic used to select best classification algorithm	CAIDA.
Maheshwari et al.[41]	Victim-end	01+02(Master+slaves) Configuration: Core i5, 8GB RAM.	number of requests per IP associated with unique protocol:HTTP, TCP, ICMP. Threshold value as detection metric	07 Physical host used to generate traffic.
Chhabra et al.[42]	Victim-end	01+04 (Master+slaves) Hadoop-2.7.1	Supervised Machine learning used. Train system using labeled data. Validate using real CAIDA dataset.	CAIDA dataset
Patil et al.[43]	Intermediate: Single ISP	01+30 (Master+datanodes) Hadoop-2.9.X	Shannon Entropy	CAIDA, FIFA MIT DDoS
Patil et al.[92]	Victim-end	01+03 (Master + datanodes) Hadoop-3.2.1	Shannon Entropy	CAIDA, MIT Lincoln DDoS, Synthetic traffic
Sharma et al.[95]	Victim-end	01+02 (Master + datanodes) Hadoop-2.9.X	Shannon Entropy	Synthetic traffic
Patil et al.[91]	Victim-end	Spark cluster:03 nodes (deploy),Hadoop cluster:02 nodes(design),Flume agent: 01 node (snipping)	Shannon Entropy, SD and Mean	Synthetic traffic
Vani et al.[96]	Victim-end	—	Modified deep learning algorithm	NSL-KDD
Ujjan et al.[101]	Victim-end	Two virtual machines Intel X5560 CPU, 2.88 GHz processor, 16 GB RAM Mininet emulator tool	sFlow and Adaptive pooling Deep learning algorithm	Synthetic traffic
Shafiq et al.[102]	Victim-end	—	TOPSIS, Shannon Entropy	BoT-IoT dataset
Kumar et al.[103]	Intermediate Multiple ISPs	—	Random Forest algorithm XGBoost	BoT-IoT dataset

Abbreviation: DPF, Distributed Processing Framework.

efficiently. However, the Hadoop-based detection mechanisms validated in an offline batch processing mode. We reviewed several existing distributed processing framework based work presented as follows:

1. Lee and Lee[31] proposed a Hadoop based DDoS detection system to protect a victim from DDoS attacks. They employed a counter-based approach by examining three parameters: time-interval, threshold, and unbalanced ratio for identifying DDoS attacks. They designed a testbed (one master and ten slave nodes) to validate this system by processing 500 GB/1 TB offline network traffic. However, this system failed to detect low rate-DDoS attacks. Further, it requires more time for analyzing network traffic, approximately 25 min for 500 GB and 47 min for 1 TB.
2. Khattak et al.[32] proposed a Hadoop-based victim-end DDoS forensics technique. They analyzed network traffic using the MapReduce programming model. This system employed two threshold values: "horizontal threshold" (number of incoming requests coming towards victim) and "vertical threshold" (number of incoming requests coming towards victim network) to identify DDoS attacks. This system validated using MIT Lincoln LLS-DDoS-1.0[97] dataset. However, the proposed system validated in offline batch processing mode and failed to identify low-rate DDoS attacks. Further, during a legitimate flash event, the false-positive rate exponentially increases because of vertical and horizontal thresholds.
3. Zaho et al.[33] proposed the Hadoop-[94] and HBase-[98] based distributed approach using neural network. They designed a testbed (consist of multiple attacker nodes, web-server, and detection framework) on the cloud platform[99] for validating the system. However, this distributed approach required more time for the detection job.
4. Dayama et al.[34] proposed a Hadoop-based DDoS detection mechanism to protect victims from DDoS attacks. They implemented a counter-based detection algorithm using the MapReduce programming to identify DDoS attacks. However, this system failed to detect attacks in real-time and not capable to identify low rate DDoS attacks.
5. Hsieh et al.[37] proposed Apache Spark-based DDoS detection framework. They designed a detection model using neural networks to classify DDoS attacks and legitimate traffic. This system trained and tested using the DDoS 2000 DARPA dataset. The detection accuracy of the proposed system is approximately 94%. However, this detection approach required more time for the detection job and designed a model using an obsolete dataset compared to today's high-speed network traffic.
6. Hameed et al.[35] proposed a Hadoop based DDoS detection framework. They employed a counter-based detection algorithm to identify DDoS attacks traffic. Further, they proposed a novel victim-end DDoS detection framework called HADEC.[36] It consists of one snipping node, detection node (Namenode), and multiple slaves (between 2 and 10). To validate the HADEC detection framework, They generate legitimate and attack flows using Mausezahn[100] for validating the HADEC detection framework. The methodology behind this approach to identify attack traffic uses static threshold values: 500 and 1000, which are the number of requests coming from a unique source IP. However, a snipping node requires more time (around 77% time of the detection process). Further, this system failed to detect a low rate DDoS attacks due to a static threshold value (500/1000).
7. Maheshwari et al.[41] proposed a distributed DDoS detection approach deployed on the Hadoop framework. This system used a counter-based detection algorithm to identify DDoS attack traffic. It maintains a count of incoming requests from unique source IP associated with protocols (such as HTTP, TCP, and ICMP) and compared with a static threshold value. However, this system could not detect low-rate DDoS attacks and validated using a small volume of data (Max. 200MB).
8. Alsirhani et al.[38] proposed a Hadoop-[93] and Spark-[104] based DDoS detection framework to protect victim server from DDoS attacks. This system used the classification algorithm (for discriminate DDoS attack and legitimate network flows), the fuzzy logic system (to select the most suitable algorithm). Alsirhani et al.[39] proposed a DDoS detection system similar to Reference 38 and applied Gradient Boosting to identify malicious traffic. However, both systems[38,39] required more time.
9. Chhabra et al.[42] proposed a victim-end DDoS forensics analytics technique deployed on the Apache Hadoop distributed framework. They employed a supervised random forest-based decision tree algorithm to design a detection model and claimed an approximate 99% accuracy. The proposed technique validated using CAIDA and university traffic samples. However, this system required more time for detection jobs and validated using an obsolete CAIDA dataset.
10. Patil et al.[43] proposed a Hadoop-based DDoS detection system called as E-Had. They employed a distributed approach based on Shannon Entropy to identify different types of DDoS attacks. This system detects low-rate DDoS attacks, high-rate DDoS attacks, and discriminate flash events from the DDoS attack. They validated the proposed system using the CAIDA, FIFA, MIT Lincoln datasets, and various scenarios of synthetic traffic flow. However, this system validated in an offline batch processing mode.
11. Sharma et al.[95] proposed an Apache Hadoop-based DDoS detection system. They evaluated this system by designing a testbed (01 Namenode and 02 Datanodes) and generated real-time traffic (ICMP) using several hosts. Patil et al.[92] proposed an Apache Hadoop-based DDoS detection system. They evaluated this system by designing a testbed (01 Namenode and 03 Datanodes). They used two DDoS datasets to validate this system: MIT DDoS and CAIDA. The methodology behind[92,95] approach is to employ a Shannon Entropy to identify malicious flows. However, both systems[92,95] validated in an offline batch processing mode.

12. Patil et al.[91] proposed a Spark-based victim-end DDoS detection system called S-DDoS to identify both high-rate and low-rate DDoS attacks. This system designed using a highly scalable H2O sparkling machine learning tool on the Hadoop framework. The methodology behind this approach is to formulate seven features from network attributes using statistical approaches to train and test a detection model. They designed a testbed (Spark cluster: three nodes, Hadoop cluster: two nodes, Flume agent: one node) to validate the S-DDoS. However, S-DDoS failed to classify attacks and flash events.
13. Vani et al.[96] proposed a Hadoop-based DDoS detection system using the DLMN algorithm. They modified a deep learning modified network and employed to identify malicious traffic flows. They designed this system using the NSL-KDD dataset and classify attack traffic into 04 classes: DoS, R2L, U2R, and probing. However, this system designed using an obsolete dataset and validated in an offline batch processing mode.
14. Ujjan et al.[101] proposed a sFlow and adaptive pooling-based DDoS attack detection approach in the SDN environment (at data-plane). They specifically designed this approach for IoT traffic flows. At the control-plane of SDN, they employed Snort IDS and Stacked Autoencoders approach to enhance the detection rate of the approach. The results show that the detection rate of the model is higher when used sFlow compared to adaptive pooling. However, this approach failed to distinguish between DDoS attacks and Flash events.
15. Shafiq et al.[102] proposed a novel approach to identify malicious flows from the Bot-IoT dataset. They introduced a novel feature selection metric and algorithm named CorrAUC for picking features from IoT flows. Then, they employed TOPSIS and Shannon Entropy-based metrics to select the best features for identifying malicious IoT flows. Finally, this approach is evaluated by implementing four machine learning algorithms on the BoT-IoT dataset instances. The results show that the detection rate of the model is more than 96%. However, this approach failed to detect low rate DDoS attacks.
16. Kumar et al.[103] proposed a distributed detection approach in the Fog computing environment for IoT applications. They employed Random Forest and XGBoost algorithms to decision making on traffic with autonomous control. Then, for load balancing of IoT traffic instances, they presented an interplanetary file system for storage. Finally, they deployed their model in the Fog computing environment and validated it using the BoT-IoT dataset traffic instances. The results show that the detection rate of the model is more than 99%. However, this approach failed to distinguish between DDoS attacks and Flash events.

3.1 | Discussion

It has been seen that several challenges in the literature for protecting victims from different types of DDoS attacks. However, our primary focused in this paper to draw some inferences with respect to distributed processing framework based detection systems. They are listed as follows:

- Most of the researchers efficiently analyzed a massive amount of network traffic on a cluster of nodes (master–slaves architecture). It will help to divide detection jobs and execute on multiple nodes in parallel. Further, this type of system doesn't become victims of attacks even a large number of attack sources and traffic volume increased.
- Most of the authors employed a counter-based detection algorithm (by calculating the number of requests from specific source IP in the time window) for recognizing the high-rate of DDoS attacks. Therefore, these types of systems failed to recognize low-rate DDoS attacks.
- Both DDoS attacks and flash events shared several similar properties. Therefore, it is a challenging task to identify attacks during flash events. Fewer systems addressed the discrimination job between attacks and flash events.
- Most of the distributed processing framework based approaches are deployed in victim-end location. Therefore, there is a necessity to explore the ISP level approach for identifying attacks at an early stage.
- Most of the existing distributed processing-based mechanisms validated in an offline batch processing mode. Therefore, there is a necessity to analyze real-time traffic with immediate responses to incoming stream traffic that means validate the system in real-time mode.

4 | CHARACTERIZATION OF DISTRIBUTED PROCESSING FRAMEWORKS

A distributed processing framework is a significant tool for Big data analytics. It analyzes a large volume of data by fetching it from distributed storage or as coming towards the system. In this section, we characterize seven open-source distributed frameworks, which represented in Figure 5. A primary objective of each distributed framework is handling (capture, analyze, store, etc.) a big volume of data in a distributed manner (master–slave architecture). Therefore, it will be interesting to design a detection mechanism using distributed processing frameworks for analyzing an enormous amount of network traffic on multiple systems in coordination. These seven frameworks categorized into three categories based on their processing ability: Batch processing framework, Stream processing framework, and Hybrid processing framework. However, questions come in mind which framework is suitable for our detection use-case. Therefore, we characterize each framework that will help to select an appropriate framework for designing experimental setup and case scenarios of detection mechanisms. A brief overview of each framework given as follows:

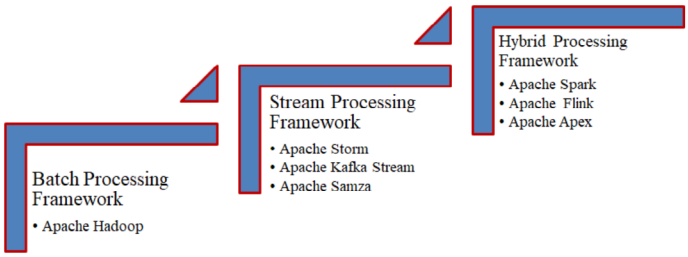

FIGURE 5 Classification of existing distributed processing frameworks

4.1 | Batch processing frameworks

The batch processing mode of distributed processing frameworks involves analyzing a massive amount of data or a large static dataset by reading it from permanent distributed storage and written outputs after completion of the task. In this, data can have bounded (a finite set of records) and persistent (stored in permanent storage) features. While analyzing a static dataset in batch processing mode treat as aggregate data instead of specific instances. The batch processing framework is a perfect choice for historical data analysis.

4.1.1 | Apache Hadoop

Apache Hadoop is one of the powerful distributed processing framework[94,105] for handling (capture, store, analyze, etc.). Big data on a cluster of nodes in batch processing mode. It is an open-source platform and implemented on commodity hardware. Apache Hadoop consists of HDFS (distributed storage), YARN (resource allocator for a cluster), and MapReduce programming model (processing model). It is a reliable, flexible, and low-cost implementation framework. The processing job involves three steps based on key-value pairs: map, shuffle and reduce. The step-by-step procedure for the data processing engine listed as follows:

- Store a massive amount of data/dataset into HDFS (on a cluster of nodes named as Datanodes) in *N* blocks (64, 128, 256 MB, etc.) and maintain metadata of blocks in master node (named as Namenode)
- Fetch *N* data blocks from HDFS
- *N* mappers executed on datanodes (that means each block requires one mapper)
- Each mapper stores intermediate results into HDFS
- Shuffle the results based on keys (key-value pairs)
- Reducer combining the results based on keys (One or more reducer can possible, its depend on configuration file)
- Store the final output into HDFS

It will be interesting to implement a DDoS detection system using Apache Hadoop. This type of detection system will: (i) Efficiently analyze and store a massive amount of network traffic, (ii) Network traffic analysis task executed on multiple nodes (a cluster) in a coordinated manner, (iii) Not becoming a victim of an attack even analyzing a large volume of traffic flows, (iv) Best tool for historical data analysis, etc.

4.2 | Stream processing frameworks

The stream processing framework analyzes data as it coming toward the system. In this, rather than analyzing complete data (datasets), it analyzes each instance (or multiple records in small time window) as it received. This type of framework is perfectly suitable where need to give immediate responses, take an action, refresh state, etc. based on incoming data.

4.2.1 | Apache storm

Apache storm[106] is an open-source, distributed, and streaming data processing platform for Big data analytics. It is one of the oldest and reliable streaming platforms from Apache family. It comes up with the simple architecture and supports various popular programming languages. A cluster (master node named as nimbus and slaves node named as supervisors) of Apache Storm manages by Apache Zookeeper. There are three important

components: topology, spouts, and bolts. A collection of spouts and bolts executing for analyzing real-time data called Storm topology. Spouts are capturing real-time data from various sources and given tuples (list of values) to bolts. Bolts analyze tuples received from spouts and perform various tasks: filter, combine, etc.

It will be interesting to explore the design DDoS detection system using Apache Storm streaming platform. This type of detection system will enable real-time traffic analysis with immediate responses to streaming attack traffic. When the requirements of the proposed detection approach are: (i) Analyze and response to streaming traffic flows, (ii) Handle a large amount of traffic in a cluster of nodes, (iii) Extremely low latency, etc. then Apache Storm is the best option for deploying a detection approach. Apache Storm is not a feasible solution when your use-case demands loosely coupled architecture.

4.2.2 | Apache Kafka Stream

Apache Kafka stream[107] is an open-source, distributed, and real-time streaming data processing platform for handling a large amount of data on single or multiple nodes (named as brokers). This framework usually used for developing streaming applications that respond in real-time. It is one of the best options for the Storm streaming platform. In this, no need to worry about a cluster manager, nimbus, supervisors, etc. Kafka can handle everything and need to concentrate on the business logic of application only. It comes up with a lightweight library (Kafka Producer, Consumer, Streams, and Connect) and supports loosely coupled features because of pub-sub architecture.

It will be interesting to explore the design DDoS detection system using the Apache Kafka streams platform. This type of detection system will enable real-time traffic analysis with immediate responses to streaming attack traffic. When the requirements of the proposed detection approach are: (i) analyze and immediate response to streaming traffic flows, (ii) handle a large amount of traffic on multiple nodes, (iii) extremely low latency, (iv) Support loosely coupled architecture, (v) highly scalable, etc. then Apache Kafka is the best option for deploying a real-time detection approach.

4.2.3 | Apache Samza

Apache Samza[108] is a distributed platform for developing stateful real-time data processing applications. Both Apache Kafka and Apache Samza implemented by the same developers and share several similar features. It is a scaled version of Apache Kafka which means it is highly scalable than Kafka. However, it executes analysis tasks on the YARN cluster which means latency higher than Kafka. It uses a Apache Kafka for fault-tolerance and guaranteed delivery of messages.

To do the real-time analysis of network traffic on the YARN cluster, one of the best ways to implement a DDoS detection system using the Apache Samza. This type of detection system will identify DDoS attacks from streaming traffic flows. When the requirements of the proposed detection approach are: (i) analyze and immediate response to streaming traffic flows, (ii) handle a large amount of traffic on multiple nodes, (iii) low latency, (iv) highly scalable, etc. then Apache Samza is the best option for deploying a real-time detection approach. Apache Samza is not a feasible solution when your use-case demands extremely low latency and exactly once message delivery guarantee.

4.3 | Hybrid processing frameworks

Hybrid frameworks handle both batch and stream processing jobs. It provided both types of APIs for analyzing bounded (static dataset) and unbounded (continuously incoming stream instances) data. There are various open-source hybrid processing frameworks available: Apache Spark, Flink, Apex, etc.

4.3.1 | Apache Spark

Apache Spark[104] is an open-source Big data analytics engine for processing data in both batch and stream processing mode. It can run on various platforms: Apache Hadoop, Apache Mesos, on Cloud platform, or a standalone system. It supports four essential libraries: Spark SQL, Spark Streaming, MLlib (machine learning), and GraphX features. Apache Spark performs a job faster than the Apache Hadoop framework due to in-memory processing capability and not reading/writing intermediate results on/from permanent storage.

It will be interesting to explore the design DDoS detection system using the Apache Spark. A Spark-based detection system will give: (i) analyzing traffic flows in real-time, (ii) historical data analysis in batch processing mode, (iii) low-latency and fault-tolerance, etc. However, it is not feasible where the detection model demands extremely low latency.

4.3.2 | Apache Flink

Apache Flink[109] is an open-source distributed framework for implementing a real-time application with performing a historical data analysis task. It also supports Lambda architecture[110] like Apache Spark. It runs on a YARN cluster for analyzing data.

It is a feasible solution for a use-case which demands to do major real-time processing tasks with minimum historical data analysis task.

4.3.3 | Apache Apex

Apache Apex[111] is a distributed processing framework for implementing a stateful, fault-tolerance, and secure Big data analytics application on YARN platform. It supports both batch and real-time processing.

If a DDoS detection system demands both batch and real-time network traffic analysis, then it is one of the good ways to deploy a detection model on the Apache Apex. This type of detection system will help to identify attacks in real-time as well as does historical data analysis.

These distributed frameworks would play a vital role in systematically analyzing network traffic on a cluster of nodes to identify DDoS attacks. Apache family frameworks are open-source, reliable, fault-tolerance, and highly scalable. We systematically summarized these tool in Table 7 by considering 15 parameters and some of them discussed as follows:

- **Delivery guarantee:** It means the reliable delivery of data and can have value: "at-least-once", "at-most-once", or "exactly-once." The "exactly-once" is desirable for distributed processing frameworks. However, it is challenging to achieve by tradeoffs with other performance parameters.
- **State management:** Maintaining a state is necessary while analyzing the data. A distributed framework can be able to keep its state with the help of checkpoints, snapshots, transaction updates, etc.

TABLE 7 Summary of distributed processing frameworks (Apache Hadoop ecosystem tools)

Parameter	Hadoop	Spark stream	Storm	Kafka stream	Samza	Flink	Apex
Version	3.1.2	2.4.2	1.2.2	2.2.0	2.11.1	1.8.0	3.7.0
Event size	Batch	Micro-batch	One	One	One	One	One
Available from	September 2007	February 14	September 2014	January 2016	July 2014	January 2015	April 2016
Supporter	Yahoo!	AMP Lab	Twitter	Confluent	LinkedIn	Artis	Torrent
Developed In	Java	Scala	Clojure	Java	Scala	Java	Java
Auto scale	Yes	Yes	No	Yes	No	No	Yes
Data-flow	Application	Application	Topology	Topology	Job	Stream	Stream
Delivery guarantee	Exactly once	Exactly once	At least once	At least once	At least once	Exactly once	Exactly once
State Management	Checkpoints	Checkpoints	Record acknowledgements	Local distributed snapshots	Local distributed snapshots	distributed snapshots	Check points
Fault tolerance	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Latency	Medium	Low	Extremely low	Extremely low	Low	Low	Very low
Windowing	Time-based	Time-based	Time/count based	Time-based	Time-based	Time/count based	Time-based
Scalability	High	High	High	High	High	High	High
Resource Management	YARN	YARN Mesos	YARN Mesos	YARN Mesos	YARN	YARN	YARN
Used in DDoS literature?	References 31,33,34	References 37-39	—	—	—	—	—

- **Fault tolerance:** A distributed framework must have to start again from where it left the analyzing process in case of failure. It can achieve by preserving snapshots of the ongoing process into persistent storage. Apache family frameworks supported the fault tolerance feature.
- **Latency:** Latency is the amount of time required to process a task, record, or topic. It should be as minimum as possible.
- **Windowing:** Windowing is an approach for selecting data based on time or event to analyze data on a framework. It can be either time-based, count-based, or event-based.

We suggested three use-cases with the selection of appropriate frameworks for designing DDoS detection system:

1. **Use-case 1:** A DDoS detection approach demands to analyze network traffic flows using batch processing mode that means historical data analysis of a large volume of network traces to get insight, then Apache Hadoop framework is one of the best tools for this use-case.
2. **Use-case 2:** A DDoS detection approach demands to analyze streaming network traffic flows in a real-time mode and immediate responses to incoming flows, then the Apache Storm, Samza, and Kafka Streams frameworks are the best tools for this type of use-case. Further, detection use-case demands extremely low latency then we can choose one from Apache Storm and Kafka Streams. Moreover, if the detection use-case requires loosely coupled architecture then Apache Kafka Streams is the best framework.
3. **Use-case 3:** A DDoS detection approach demands to analyze streaming network traffic flows in a real-time mode and historical data analysis too, then the Apache Spark, Flink, and Apex frameworks are the best tools for this type of use-case.

There is no restriction of using multiple tools in the use-case (for example, a combination of Apache Storm and Hadoop) if your use-case demands.

5 | PERFORMANCE EVALUATION METRICS: DISTRIBUTED PROCESSING FRAMEWORK BASED DDOS DETECTION SYSTEM

In Reference 88, provided several performance measurement metrics in 2013 and several mechanisms validated with some other metrics also. Therefore, there is a need to provide a new set of standard performance evaluation metrics for the proposed mechanisms. A set of performance evaluation metrics is summarized as follows:

1. **System strength:** The strength of the proposed system can measure using how it efficiently classify instances (benign and attacks). For that we need a confusion matrix by measuring values of four parameters: true positive (*True_Pos*), true negative (*True_Neg*), false positive (*False_Pos*), and false negative (*False_Neg*). A confusion matrix presented in Table 8. We can calculate twelve performance evaluation metrics using a confusion matrix, which are listed as follows with their mathematical representation:
   - Precision (P_R): $P_R = \frac{True\_Pos}{True\_Pos+False\_Pos}$
   - Detection rate/Recall (D_R): $D_R = \frac{True\_Pos}{True\_Pos+False\_Neg}$
   - False Positive Rate (FP_R): $FP_R = \frac{False\_Pos}{True\_Neg+False\_Pos}$
   - False Negative Rate (FN_R): $FN_R = \frac{False\_Neg}{True\_Pos+False\_Neg}$
   - True Positive Rate (TP_R): $TP_R = \frac{True\_Pos}{True\_Pos+False\_Neg}$

TABLE 8 Confusion matrix

		Predicted class: Attack	Predicted class: Benign	total
Actual class	Attack′	True Positive (*True_Pos*)	False Negative (*False_Neg*)	Attack′
	Benign′	False Positive (*False_Pos*)	True Negative (*True_Neg*)	Benign′
	total	Attack	Benign	

- True Negative Rate (TN_R): $TN_R = \frac{True\_Neg}{True\_Neg+False\_Pos}$
- F-Measure (F_M): $F_M = 2 * \frac{P_R * D_R}{P_R + D_R}$
- Negative Predictive Value (NP_V): $NP_V = \frac{True\_Neg}{True\_Neg+False\_Neg}$
- F-Measure Complement (F_{MC}): $F_{MC} = \frac{2 * TN_R * NP_V}{TN_R + NP_V}$
- Classification rate/detection accuracy (C_R): $C_R = \frac{True\_Pos+True\_Neg}{True\_Pos+True\_Neg+False\_Pos+False\_Neg}$
- Balance accuracy (B_{acc}): $B_{acc} = \frac{TP_R+TN_R}{2}$
- Misclassification rate (M_R): $M_R = \frac{False\_Pos+False\_Neg}{True\_Pos+True\_Neg+False\_Pos+False\_Neg}$

2. **Cost:** Cost is one more essential metric for the evaluation of the proposed system. The model should be economical. The detection system cost depends on various parameters such as infrastructure, various hardware devices, traffic transportation, analysis task, and deployment framework (traditional and distributed processing frameworks). However, the most cost affecting factor is the deployment location of the system. The cost-wise deployment locations listed in ascending order as victim-end, intermediate network, source-end, and hybrid locations.
3. **Deployment complexity:** Only proposed a robust mechanism for protecting victims from DDoS attacks is not sufficient, also it should be feasible for real-time implementation. Therefore, deployment/implementation complexity is an essential metric for comparison.
4. **Scalability:** A highly scalable mechanism does not become victims of attacks while analyzing a large amount of network traffic (benign, flash events, and DDoS attacks). That means, even attacker increased automated bots and traffic, the detection model should efficiently handle.
5. **Response time:** How much time required to analyze and make an immediate decision against incoming malicious flows? It should be in seconds to avoid the collateral damage of the victim network and continue for giving services to benign users.
6. **Deployment locations:** Several important parameters such as cost, accuracy, complexity, robustness, real-time implementation, etc. are depending on the deployment locations of the mechanism. Therefore, the selection of deployment locations for the proposed mechanism is a significant metric.
7. **Receiver operating characteristic (ROC) curve:** It is a graphical tool for the distinguishing ability of a binary classification problem. It generated by plotting values of true-positive rate (sensitivity) and false-positive rate (specificity) based on the range of threshold values. As we know, both terms sensitivity and specificity are inversely proportional to each other. With the help of the ROC curve, we easily get higher or lower detection accuracy of the system. If a curve is closer to the left-hand boundary (*Y*-axis) and top, then it shows higher detection accuracy that means the area under the curve is more. While if the curve line closer to a diagonal line, then it shows lower detection accuracy that means the area under the curve is less.
8. **Network traffic processing complexity (w.r.t. traditional and distributed framework-based systems):** In traditional framework-based systems, each traffic instance analyze at a single point. Therefore, the complexity of the traditional approach is $O(N)$, where N is the number of instances executed on a system. While in distributed framework-based systems, traffic instances analyzed on a cluster of systems (slaves/supervisors). Therefore, the complexity cost-shared between systems, say n (number of systems in a cluster). For calculating complexity for a distributed approach, assuming each system analyzed an equal number of traffic instances. Therefore, the time complexity of the distributed approach is $O(\frac{N}{n})$. Further, we have to consider one more variable while calculating the complexity of a distributed approach is intermediate results sharing cost between systems. Let us consider the intermediate result sharing cost is $O(c)$. Therefore, the distributed approach time complexity (DTC) is $DTC = O(\frac{N}{n}) + O(c)$. However, the intermediate results sharing the cost between systems is extremely small comparatively analyzing tasks and can negligible. Therefore the DTC of the distributed approach is $O(\frac{N}{n})$. From this, we can conclude that the complexity of the distributed approach is inversely proportional to the number of systems in a cluster(n).

6 | OPEN ISSUES: DISTRIBUTED PROCESSING FRAMEWORKS-BASED DDOS DETECTION SYSTEM

In the literature, numerous solutions have been proposed by fellow researchers to protect victims from different types of DDoS attacks. However, it is a challenging task when an attacker follows legitimate traffic patterns of victims and launches attacks during flash events. Further, sometimes the defense mechanism itself becomes a victim of an attack while analyzing a big volume of network flows (pattern changed to Tbps from Gbps). Therefore, there is a need to implement a robust mechanism using a distributed processing framework which: (i) protect victims and its resources from attack, (ii) efficiently analyze a big volume of network flows, (iii) analyze traffic flows in real-time mode, (iv) perform historical data analysis to regularly update a detection model for identifying zero-day attacks, (v) give continue access to benign users even during an attack, (vi) real-time response, etc. We identified several issues which need to address:

1. Most of the existing systems have designed and validated in simulation or emulation environments. However, it is a hurdle to model the actual behavior of the real network in the simulation or emulation environment. Therefore, there is a need to design and validate the proposed system in a real environment.

2. Further, most of the systems designed and validated using different datasets. However, this is not a guaranteed solution because each dataset captured with different topology, infrastructure, traffic speed, constraints, etc. Therefore, there is necessary to generate and use a well-balanced dataset that includes HR-DDoS, LR-DDOS, flash events, legitimate, etc. scenarios for systems.
3. It is essential to design a robust baseline network behavior for anomaly-based detection. In the literature, most of the systems employ outdated datasets (DARPA DDoS MITLL-2000, CAIDA DDoS-2007, FIFA-1998, etc.) to design. However, this is not a feasible solution compared to today's high-speed and high volume of Internet traffic. Therefore, there is a demand of new dataset, which captured recently to design the baseline behavior of the system.
4. DDoS attacks traffic immediately overwhelms a victim (resources/bandwidth). Therefore, there is necessary to analyze the minimum number of traffic features to identify DDoS attacks. It will help to analyze traffic faster.
5. In the literature,[31-43] most of the researchers proposed a distributed processing framework-based detection systems to recognize high-rate attacks by employing a counter-based detection approach on incoming traffic flows. However, this type of systems failed to recognize low-rate attacks.
6. In the literature,[31-42] most of the researchers proposed a distributed processing framework based systems to identify attacks. However, these systems failed to discriminate between flash events and DDoS attacks.
7. Most of researchers,[31-43] validated their distributed processing frameworks-based detection systems in offline batch processing or micro-batch processing mode. However, none of them focused on validating system using stream processing mode.
8. Several researchers deployed their detection systems at ISP-level, intermediate network, etc. in the case of traditional frameworks. However, in the case of distributed processing frameworks, most of the researchers deployed systems at victim-end only. Therefore, there is a demand to explore the ISP-level detection system using distributed processing frameworks.

6.1 | Datasets and their limitations

The biggest problem while designing a comprehensive DDoS attack detection mechanism is the lack of recent well-balanced DDoS datasets. The well-balanced dataset must have an appropriate balance between attacks and legitimate network flows.[112] Further, organizations that suffered from DDoS attacks don't like to share network traces (attack and legitimate) in the public domain due to their security policies. Most of the existing mechanisms are validated using outdated datasets such as MIT Lincoln Laboratory DDoS, CAIDA DDoS, and FIFA 1998 world cup. Popular DDoS datasets are summarized as follows and compared each dataset including limitations in Table 9:

1. **MIT Lincoln Laboratory (LLSDDoS Dataset 2000)[113]:** This dataset has two versions: LLDOS 1.0 (performed by novice attackers) and LLDOS 2.0.2 (performed by stealthier attackers) dataset. Both versions of the MIT laboratory DDoS dataset are grouped into five phases: (i) network

TABLE 9 Comparison of available datasets

Dataset	Year	Dataset class	Dataset scope	IP address	Limitations
MIT LLSDDoS 1.0[113] & LLSDDoS 2.0.2	1998, 2000	Synthetic	DDoS	Real IPs	• Asymmetric flows (not well-balanced) • Outdated dataset (compared to today's high speed network traffic)
CAIDA[114]	2007	Real	DDoS	Mapped IPs	• Asymmetric flows (not well-balanced) • Pseudonymized IPs (due to security) • Captured at network layer (hidden application specific details)
FIFA WorldCup98[115]	1998	Real	Flash	Mapped IPs	• Asymmetric flows and no distributed denial of service (DDoS) flows • Pseudonymized IPs (due to security) • Obsolete dataset
CIC DoS[116]	2017	Synthetic	DoS	Real IPs	• No DDoS traffic flows • Asymmetric flows (not well-balanced)
BoT-IoT[117]	2018	Synthetic	DDoS DoS	Real IPs	• Short length of captured packets • Not considered Flash event scenarios
CICDDoS2019[118]	2019	Synthetic	DDoS	Real IPs	• Asymmetric flows (attack traces 1500 times more than benign traces) • Not considered Flash event scenarios

scanning (ii) compromising nodes, (iii) downloads the mstream, (iv) install the mstream (trojan-based malicious program), and (v) begin the DDoS attack.

2. **CAIDA DDoS 2007 Dataset**[114]**:** The CAIDA DDoS dataset contains DDoS attack network flows for approximately an hour. This attack aimed to overwhelm both the computing and network resources of the victim server. Further, this dataset includes both high- and low-rate DDoS attack instances. The PCAP file size of the dataset is approximately 21 GB and removed benign flows from it.
3. **FIFA WorldCup98 Dataset**[115]**:** This dataset captures 1 billion 353 million incoming requests, which are recorded on the website of World Cup's 1998 between April to July 1998. This dataset is intended to provide flash events scenarios to distinguish between flash and DDoS traffic flows.
4. **CIC DoS Dataset 2017**[116]**:** This dataset combines legitimate traffic instances from the ISCX-IDS dataset. They provided four types of application-layer DoS attack instances. The duration of the recording dataset is approximately 24 h, and the storage capacity is 4.6 GB.
5. **BoT-IoT Dataset 2018**[117]**:** The BoT-IoT dataset comprises both IoT-related background traffic and different types of attack flows. This dataset is captured in a real testbed environment with more than 72 million traces. Then they have created several features from existing network traffic attributes and labeled each flow (legitimate/attack). It will help to design a model using traditional and non-traditional machine learning algorithms. This dataset available is PCAP (69.3 GB) and CSV (16.7 GB) files.
6. **CIC DDoS Dataset 2019 (CICDDoS2019)**[118]**:** The CICDDoS2019 dataset includes both legitimate and different types of DDoS attack traffic flows. This dataset is captured in a real testbed environment approximately for 2 days. On the first day, they executed seven types of attacks, such as "PortScan, NetBIOS, LDAP, MSSQL, UDP, UDP-Lag, and SYN" while on another day 12 types of attacks, such as "NTP, DNS, LDAP, MSSQL, NetBIOS, SNMP, SSDP, UDP, UDP-Lag, WebDDoS, SYN, and TFTP". This dataset is publicly available for researchers in both PCAP (without labeled) and CSV format (with 87 features including labeled).

We listed the consolidated limitations of these datasets as follows:

- Asymmetric (not well-balanced) network traffic scenarios (legitimate, attacks, and Flash events)
- Most of the popular datasets are outdated (compared to today's high speed and volume of network traffic)
- Pseudonymized IP address from the dataset (due to security)
- Short length of captured packets
- Many datasets are captured at the network layer and hence hide application-specific details from traces.
- Flash event dataset captured in 1998. Therefore, it is outdated.

7 | CONCLUSIONS AND FUTURE DIRECTIONS

A DDoS attack is a significant threat to cyber-security. Numerous mechanisms have been proposed to protect victims from DDoS attacks. However, attack incidents and volume sizes are regularly increasing. There are several challenges in front of researchers: improve detection accuracy, minimize false-positive rate, analyze a big volume of network traffic, real-time processing, etc. In this paper, we presented a DDoS defense deployment taxonomy and analyzed the existing distributed framework-based detection systems. Additionally, characterized open-source distributed processing frameworks to select an appropriate as per requirements of customer environment. Finally, we presented a set of standard evaluation metrics, open issues, and discussion on available datasets including their limitations.

We strongly believe that a comprehensive real-time distributed streaming platform (Apache Kafka streams, Storm, Spark streaming, etc.) based DDoS attack detection mechanism could be one of the most reliable approaches to detect DDoS attacks. Therefore, it will be interesting to design a distributed detection model on the Apache Hadoop framework using traditional and nontraditional (deep learning) machine learning algorithms. And deploy implemented model on the distributed streaming platform to classify incoming network traffic flows (legitimate, DDoS attack, and Flash Events) in real-time mode.

DATA AVAILABILITY STATEMENT

Data sharing not applicable - no new data generated.

ORCID

Nilesh Vishwasrao Patil https://orcid.org/0000-0002-1983-668X

REFERENCES

1. Internet users in the world by geographic regions 2020 Q1. https://www.internetworldstats.com/stats.htm.. Accessed July 01, 2019.
2. VXchange: comprehensive guide to IoT statistics you need to know in 2020. https://www.vxchnge.com/blog/iot-statistics. Accessed July 01, 2019.

3. Bhatia S, Behal S, Ahmed I. Distributed denial of service attacks and defense mechanisms: current landscape and future directions. *Versatile Cybersecurity*. New York, NY: Springer; 2018:55-97.
4. Sachdeva M, Kumar K. A traffic cluster entropy based approach to distinguish DDoS attacks from flash event using DETER testbed. *ISRN Commun Netw*. 2014;2014:1–15. https://dx.doi.org/10.1155/2014/259831.
5. Behal S, Kumar K, Sachdeva M. D-FACE: an anomaly based distributed approach for early detection of DDoS attacks and flash events. *J Netw Comput Appl*. 2018;111:49-63.
6. Kaspersky: DDoS attacks Q1-2020. https://securelist.com/ddos-attacks-in-q1-2020/96837/. Accessed June 26, 2020.
7. Re-Hash: the largest DDoS attacks in history. https://www.thesslstore.com/blog/largest-ddos-attack-in-history/. Accessed June 23, 2020.
8. Github DDoS attack 28 February 2018. https://www.a10networks.com/resources/articles/5-most-famous-ddos-attacks. Accessed March 21, 2019.
9. DDoS attacks in Q2-2020: kaspersky. https://securelist.com/ddos-attacks-in-q2-2020/98077/. Accessed August 13, 2020.
10. DDoS Q1 2019 report. https://www.kaspersky.com/about/press-releases/2019a-ddos-storm-has-come-number-of-attacks-grows-after-long-period-of-decline. Accessed September 9, 2019.
11. Kaspersky DDoS attack Q4 2018. https://securelist.com/ddos-attacks-in-q4-2018/89565/. Accessed March 04, 2019.
12. Zargar ST, Joshi J, Tipper D. A survey of defense mechanisms against distributed denial of service (DDoS) flooding attacks. *IEEE Commun Surv Tutor*. 2013;15(4):2046-2069.
13. Manavi MT. Defense mechanisms against distributed denial of service attacks: a survey. *Comput Electr Eng*. 2018;72:26-38.
14. Peng T, Leckie C, Ramamohanarao K. Survey of network-based defense mechanisms countering the DoS and DDoS problems. *ACM Comput Surv (CSUR)*. 2007;39(1):3.
15. Bhuyan MH, Bhattacharyya DK, Kalita JK. Network anomaly detection: methods, systems and tools. *IEEE Commun Surv Tutor*. 2014;16(1):303-336.
16. Douligeris C, Mitrokotsa A. DDoS attacks and defense mechanisms: classification and state-of-the-art. *Comput Netw*. 2004;44(5):643-666.
17. Mirkovic J, Reiher P. A taxonomy of DDoS attack and DDoS defense mechanisms. *ACM SIGCOMM Comput Commun Rev*. 2004;34(2):39-53.
18. Hoque N, Bhuyan MH, Baishya RC, Bhattacharyya DK, Kalita JK. Network attacks: taxonomy, tools and systems. *J Netw Comput Appl*. 2014;40:307-324.
19. Lee S. Distributed denial of service: taxonomies of attacks, tools and countermeasures. Paper presented at: Proceedings of the International Workshop on Security in Parallel and Distributed Systems, San Francisco, CA; 2004:543-550.
20. Mahjabin T, Xiao Y, Sun G, Jiang W. A survey of distributed denial-of-service attack, prevention, and mitigation techniques. *Int J Distrib Sens Netw*. 2017;13(12):1550147717741463.
21. Behal S, Characterization KK. Comparison of DDoS attack tools and traffic generators: a review. *Int J Netw Secur*. 2017;19(3):383-393.
22. Elejla OE, Anbar M, Belaton B. ICMPv6-based DoS and DDoS attacks and defense mechanisms. *IETE Tech Rev*. 2017;34(4):390-407.
23. Fenil E, Mohan Kumar P. Survey on DDoS defense mechanisms. *Concurr Comput Pract Exper*. 2019;32:e5114.
24. Praseed A, Thilagam PS. DDoS attacks at the application layer: challenges and research perspectives for safeguarding web applications. *IEEE Commun Surv Tutor*. 2018;21(1):661-685.
25. Joëlle MM, Park YH. Strategies for detecting and mitigating DDoS attacks in SDN: a survey. *J Intell Fuzzy Syst*. 2018;35(6):5913-5925.
26. Dong S, Abbas K, Jain R. A survey on distributed denial of service (DDoS) attacks in SDN and cloud computing environments. *IEEE Access*. 2019;7:80813-80828.
27. Tandon R. A survey of distributed denial of service attacks and defenses; 2020. arXiv preprint arXiv:2008.01345.
28. HM M, Kumar DRA. A survey on machine learning techniques used for detection of DDOS attacks; 2019.
29. Singh J, Behal S. Detection and mitigation of DDoS attacks in SDN: a comprehensive review, research challenges and future directions. *Comput Sci Rev*. 2020;37:100279.
30. Sachdeva M, Singh G, Kumar K. Deployment of distributed defense against DDoS attacks in ISP domain. *Int J Comput Appl*. 2011;15(2):25-31.
31. Lee Y, Lee Y. Detecting ddos attacks with hadoop. Paper presented at: Proceedings of the ACM CoNEXT Student Workshop, Tokyo Japan; 2011:7.
32. Khattak R, Bano S, Hussain S, Anwar Z. Dofur: Ddos forensics using mapreduce. *Frontiers of Information Technology (FIT)*. Islamabad, Pakistan: IEEE; 2011:117-120.
33. Zhao T, Lo DCT, Qian K. A neural-network based DDoS detection system using hadoop and HBase. Paper presented at: Proceedings of the 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, New York, NY: IEEE; 2015:1326-1331.
34. Dayama R, Bhandare A, Ganji B, Narayankar V. Secured network from distributed DOS through HADOOP. *Int J Comput Appl*. 2015;118(2):20–22.
35. Hameed S, Ali U. Efficacy of live ddos detection with hadoop. Paper presented at: Proceedings of the Network Operations and Management Symposium (NOMS), 2016 IEEE/IFIP, Istanbul, Turkey: IEEE; 2016:488-494.
36. Hameed S, Ali U. HADEC: Hadoop-based live DDoS detection framework. *EURASIP J Inf Secur*. 2018;2018(1):1–19.
37. Hsieh CJ, Chan TY. Detection DDoS attacks based on neural-network using apache spark. Paper presented at: Proceedings of the Applied System Innovation (ICASI), 2016 International Conference, Okinawa, Japan: IEEE; 2016:1-4.
38. Alsirhani A, Sampalli S, Bodorik P. DDoS attack detection system: utilizing classification algorithms with apache spark. Paper presented at: Proceedings of the 2018 9th IFIP International Conference on New Technologies, Mobility and Security (NTMS), Paris, France: IEEE; 2018:1-7.
39. Alsirhani S, Bodorik P. DDoS detection system: utilizing gradient boosting algorithm and apache spark. Paper presented at: Proceedings of the 2018 IEEE Canadian Conference on Electrical & Computer Engineering (CCECE), Quebec, Canada: IEEE; 2018:1-6.
40. Ahmad S, Yasin A, Shafi Q. DDoS attacks analysis in bigdata (hadoop) environment. Paper presented at: Proceedings of the 2018 15th International Bhurban Conference on Applied Sciences and Technology (IBCAST), National Centre for Physics Quaid-i-Azam University, Pakistan: IEEE; 2018:495-501.
41. Maheshwari V, Bhatia A, Kumar K. Faster detection and prediction of DDoS attacks using MapReduce and time series analysis. Paper presented at: Proceedings of the 2018 International Conference on Information Networking (ICOIN), Jeju, Korea; 2018:556-561.
42. Chhabra GS, Singh V, Singh M. Hadoop-based analytic framework for cyber forensics. *Int J Commun Syst*. 2018;31(15):e3772.
43. Patil NV, Krishna CR, Kumar K, Behal S. E-Had: a distributed and collaborative detection framework for early detection of DDoS attacks. *J King Saud Univ Comput Inf Sci*. 2019;1–15. https://dx.doi.org/10.1016/j.jksuci.2019.06.016.
44. Senie D, Ferguson P. Network ingress filtering: defeating denial of service attacks which employ IP source address spoofing. *Network*. 1998;1–8.
45. Wang F, Wang H, Wang X, Su J. A new multistage approach to detect subtle DDoS attacks. *Math Comput Model*. 2012;55(1-2):198-213.

46. Abdelsayed S, Glimsholt D, Leckie C, Ryan S, Shami S. An efficient filter for denial-of-service bandwidth attacks. Paper presented at: Proceedings of the IEEE Global Telecommunications Conference on GLOBECOM'03. (IEEE Cat. No. 03CH37489), San Francisco, CA: IEEE; vol. 3, 2003:1353-1357.
47. Devi SR, Yogesh P. Detection of application layer DDoS attacks using information theory based metrics. *CS & IT-CSCP*. 2012;10:213-223.
48. Peng T, Leckie C, Ramamohanarao K. Protection from distributed denial of service attacks using history-based IP filtering. Paper presented at: Proceedings of the IEEE International Conference on Communications, 2003. ICC'03, Anchorage, AK; 2003:482-486.
49. Kim Y, Lau WC, Chuah MC, Chao HJ. PacketScore: a statistics-based packet filtering scheme against distributed denial-of-service attacks. *IEEE Trans Depend Sec Comput*. 2006;3(2):141-155.
50. Zhang G, Jiang S, Wei G, Guan Q. A prediction-based detection algorithm against distributed denial-of-service attacks. Paper presented at: Proceedings of the 2009 International Conference on Wireless Communications and Mobile Computing: Connecting the World Wirelessly; 2009:106-110; ACM, New York, NY.
51. Muharish EYM. Packet Filter Approach to Detect Denial of Service Attacks [Master of Science, thesis]; 2016.
52. Bhatia S, Schmidt D, Mohay G. Ensemble-based ddos detection and mitigation model. Paper presented at: Proceedings of the 5th International Conference on Security of Information and Networks, Jaipur, India. 2012:79-86.
53. Tellenbach BM, . Detection Classification and Visualization of Anomalies using Generalized Entropy metrics [PhD thesis]. ETH Zurich; 2012.
54. Behal S, Kumar K, Sachdeva M. D-fac: a novel φ-divergence based distributed ddos defense system. *J King Saud Univ Comput Inf Sci*. 2018;1-12. https://dx.doi.org/10.1016/j.jksuci.2018.03.005.
55. Sachdeva M, Kumar K, Singh G. A comprehensive approach to discriminate DDoS attacks from flash events. *J Inf Secur Appl*. 2016;26:8-22.
56. Ma X, Chen Y. DDoS detection method based on chaos analysis of network traffic entropy. *IEEE Commun Lett*. 2014;18(1):114-117.
57. Jun JH, Lee D, Ahn CW, Kim SH. DDoS attack detection using flow entropy and packet sampling on huge networks of ICN; 2014:185-190.
58. Dorbala SY, Kishore R, Hubballi N. An experience report on scalable implementation of ddos attack detection. Paper presented at: Proceedings of the International Conference on Advanced Information Systems Engineering; 2015:518-529; Springer, New York, NY.
59. Feinstein L, Schnackenberg D, Balupari R, Kindred D. Statistical approaches to DDoS attack detection and response. Paper presented at: Proceedings of the DARPA Information Survivability Conference and Exposition, Washington, DC: IEEE; 2003:1-12.
60. Jin S, Yeung DS. A covariance analysis model for DDoS attack detection. Paper presented at: Proceedings of the 2004 IEEE International Conference on Communications, Paris, France: IEEE; 2004:1882-1886.
61. Chen Y, Hwang K, Ku WS. Collaborative detection of DDoS attacks over multiple network domains. *IEEE Trans Parall Distrib Syst*. 2007;12:1649-1662.
62. Jun JH, Oh H, Kim SH. DDoS flooding attack detection through a step-by-step investigation. Paper presented at: Proceedings of the 2011 IEEE 2nd International Conference on Networked Embedded Systems for Enterprise Applications, Perth, Australia: IEEE; 2011:1-5.
63. Ranjan S, Swaminathan R, Uysal M, Knightly EW. DDoS-resilient scheduling to counter application layer attacks under imperfect detection. *INFOCOM*. Barcelona, Spain: Citeseer; 2006.
64. Kumar K, Joshi R, Singh K. A distributed approach using entropy to detect DDoS attacks in ISP domain. Paper presented at: Proceedings of the 2007 International Conference on Signal Processing, Communications and Networking, Chennai, India: IEEE; 2007:331-337.
65. Lu K, Wu D, Fan J, Todorovic S, Nucci A. Robust and efficient detection of DDoS attacks for large-scale internet. *Comput Netw*. 2007;51(18):5036-5056.
66. Nychis G, Sekar V, Andersen DG, Kim H, Zhang H. An empirical evaluation of entropy-based traffic anomaly detection. Paper presented at: Proceedings of the 8th ACM SIGCOMM Conference on Internet Measurement; 2008:151-156; ACM, New York, NY.
67. Das D, Sharma U, Bhattacharyya D. Detection of HTTP flooding attacks in multiple scenarios. Paper presented at: Proceedings of the 2011 International Conference on Communication, Computing & Security; 2011:517-522; ACM, New York, NY.
68. Xiang Y, Li K, Zhou W. Low-rate DDoS attacks detection and traceback by using new information metrics. *IEEE Trans Inf Forens Secur*. 2011;6(2):426-437.
69. Behal S, Kumar K. Detection of DDoS attacks and flash events using novel information theory metrics. *Comput Netw*. 2017;116:96-110.
70. Saad RM, Anbar M, Manickam S, Alomari E. An intelligent icmpv6 ddos flooding-attack detection framework (v6iids) using back-propagation neural network. *IETE Tech Rev*. 2016;33(3):244-255.
71. Baishya RC, Bhattacharyya D. A complete detection and mitigation framework to protect a network from DDoS attacks. *IETE J Res*. 2019;1-18.
72. Bhatia S, Mohay G, Schmidt D, Tickle A. Modelling web-server flash events. Paper presented at: Proceedings of the 2012 IEEE 11th International Symposium on Network Computing and Applications, Cambridge, MA: IEEE; 2012:79-86.
73. Beitollahi H, Deconinck G. Tackling application-layer DDoS attacks. *Proc Comput Sci*. 2012;10:432-441.
74. Shiaeles SN, Katos V, Karakos AS, Papadopoulos BK. Real time DDoS detection using fuzzy estimators. *Comput Secur*. 2012;31(6):782-790.
75. Ni T, Gu X, Wang H, Li Y. Real-time detection of application-layer DDoS attack using time series analysis. *J Control Sci Eng*. 2013;2013:4.
76. Sangkatsanee P, Wattanapongsakorn N, Charnsripinyo C. Practical real-time intrusion detection using machine learning approaches. *Comput Commun*. 2011;34(18):2227-2235.
77. Akbar A, Basha SM, Sattar SA. Leveraging the SIP load balancer to detect and mitigate DDos attacks. Paper presented at: Proceedings of the 2015 International Conference on Green Computing and Internet of Things (ICGCIoT), Greater Noida, Delhi: IEEE; 2015:1204-1208.
78. Saleh MA, Abdul MA. A novel protective framework for defeating HTTP-based denial of service and distributed denial of service attacks. *Sci World J*. 2015;2015:1–19.
79. Kalkan K, Alagöz F. A distributed filtering mechanism against DDoS attacks: ScoreForCore. *Comput Netw*. 2016;108:199-209.
80. Thangavel S, Kannan S. Detection and trace back of low and high volume of distributed denial-of-service attack based on statistical measures. *Concurr Comput Pract Exper*. 2019;e5428.
81. Santos R, Souza D, Santo W, Ribeiro A, Moreno E. Machine learning algorithms to detect DDoS attacks in SDN. *Concurr Comput Pract Exper*. 2020;32(16):e5402.
82. Vetha S, Vimala Devi K. A trust-based hypervisor framework for preventing DDoS attacks in cloud. *Concurr Comput Pract Exper*. 2019;e5279.
83. Jing H, Wang J. DDoS detection based on graph structure features and non-negative matrix factorization. *Concurr Comput Pract Exper*. 2020;e5783.
84. Liu X, Yang X, Lu Y. *StopIt: Mitigating DoS Flooding Attacks from Multi-million Botnets. Technical Report*. Irvine: Citeseer; 2008.
85. Saifullah A. Defending against distributed denial-of-service attacks with weight-fair router throttling; 2009.
86. François J, Aib I, Boutaba R. FireCol: a collaborative protection network for the detection of flooding DDoS attacks. *IEEE/ACM Trans Netw (TON)*. 2012;20(6):1828-1841.

87. Akella A, Bharambe A, Reiter M, Seshan S. Detecting DDoS attacks on ISP networks. Paper presented at: Proceedings of the Workshop on Management and Processing of Data Streams; 2003; Citeseer.
88. Zargar ST, Joshi J. DiCoDefense: distributed collaborative defense against ddos flooding attacks. Paper presented at: Proceedings of the 34th IEEE Symposium on Security and Privacy (S&P â€~13)(Poster), San Francisco, CA: Citeseer; 2013.
89. Tao Y, Yu S. DDoS attack detection at local area networks using information theoretical metrics. Paper presented at: Proceedings of the 2013 12th IEEE International Conference on Trust, Security and Privacy in Computing and Communications, Melbourne, Australia: IEEE; 2013:233-240.
90. Kongshavn M, Haugerud H, Yazidi A, Maseng T, Hammer H. Mitigating DDoS using weight-based geographical clustering. *Concurr Comput Pract Exper*. 2020;32(11):e5679.
91. Patil NV, Rama Krishna C, Kumar K. S-DDoS: apache spark based real-time DDoS detection system. *J Intell Fuzzy Syst*. 2020;38:1-9.
92. Patil NV, Krishna CR, Kumar K. Apache hadoop based distributed denial of service detection framework. Paper presented at: Proceedings of the International Conference Information, Communication and Computing Technology; 2019:25-35; Springer, New York, NY.
93. Hadoop framework: HDFS and MapReduce; 2019. https://www.edureka.co/blog/what-is-hadoop/.
94. Apache Hadoop. https://hadoop.apache.org/. Accessed March 04, 2019.
95. Sharma A, Agrawal C, Singh A, Kumar K. Real-time DDoS detection based on entropy using hadoop framework. *Computing in Engineering and Technology*. Singapore, Asia: Springer; 2019:297-305.
96. Vani YK, Ranjana P. Detection of distributed denial of service attack using dlmn algorithm in hadoop. *J Critical Rev*. 2020;7(11).1011–1017.
97. LLDOS 1.0 - scenario one. https://www.ll.mit.edu/r-d/datasets/2000-darpa-intrusion-detection-scen%ario-specific-datasets. Accessed January 24, 2019.
98. Hbase tutorials. https://data-flair.training/blogs/hbase-tutorial/. Accessed January 01, 2019.
99. Cloud Computing. https://azure.microsoft.com/en-in/overview/what-is-cloud-computing/. Accessed January 01, 2019.
100. Mausezahn- fast traffic generator; 2019. http://man7.org/linux/man-pages/man8/mausezahn.8.html. Accessed January 19, 2019
101. Ujjan RMA, Pervez Z, Dahal K, Bashir AK, Mumtaz R, González J. Towards flow and adaptive polling sampling for deep learning based DDoS detection in SDN. *Futur Gener Comput Syst*. 2020;111:763-779.
102. Shafiq M, Tian Z, Bashir AK, Du X, Guizani M. Corrauc: a malicious bot-iot traffic detection method in iot network using machine learning techniques. *IEEE IoT J*. 2020. https://dx.doi.org/10.1109/JIOT.2020.3002255.
103. Kumar P, Kumar R, Gupta GP, Tripathi R. A distributed framework for detecting DDoS attacks in smart contract-based blockchain-IoT systems by leveraging fog computing. *Trans Emerg Telecommun Technol*. 2020;e4112.
104. Apaceh Saprk. https://spark.apache.org/. Accessed January 19, 2019
105. Bhardwaj A, Singh VK, Narayan Y. Analyzing BigData with Hadoop Cluster in HDInsight azure cloud. Paper presented at: Proceedings of the 2015 Annual IEEE India Conference (INDICON), New Delhi, India: IEEE; 2015:1-5.
106. Apache Storm. https://storm.apache.org/. Accessed May 04, 2019.
107. Apache Kafka. https://kafka.apache.org/. Accessed May 06, 2019.
108. Apache Samza. https://samza.apache.org/. Accessed May 06, 2019.
109. Apache Flink. https://flink.apache.org/. Accessed May 06, 2019.
110. Lambda Architecture. http://lambda-architecture.net/. Accessed March 02, 2019.
111. Apache Apex. https://apex.apache.org/. Accessed May 06, 2019..
112. Bhatia S, Schmidt D, Mohay G, Tickle A. A framework for generating realistic traffic for distributed denial-of-service attacks and flash events. *Comput Secur*. 2014;40:95-107.
113. 2000 Darpa intrusion detection scenario specific datasets. https://www.ll.mit.edu/r-d/datasets/2000-darpa-intrusion-detection-scenario-specific-datasets. Accessed October 26, 2020.
114. The CAIDA DDoS Attack 2007 dataset. https://www.caida.org/data/passive/ddos-20070804_dataset.xml. Accessed October 26, 2020.
115. FIFA World Cup; 1998. http://ita.ee.lbl.gov/html/contrib/WorldCup.html. Accessed August 25, 2019.
116. CIC DoS Dataset 2017. https://www.unb.ca/cic/datasets/dos-dataset.html. Accessed October 27, 2020.
117. Koroniotis N, Moustafa N, Sitnikova E, Turnbull B. Towards the development of realistic botnet dataset in the internet of things for network forensic analytics: Bot-iot dataset. *Futur Gener Comput Syst*. 2019;100:779-796.
118. CIC DDoS Dataset; 2019. https://www.unb.ca/cic/datasets/ddos-2019.html. Accessed October 27, 2020.

How to cite this article: Patil NV, Rama Krishna C, Kumar K. Distributed frameworks for detecting distributed denial of service attacks: A comprehensive review, challenges and future directions. *Concurrency Computat Pract Exper*. 2021;33:e6197. https://doi.org/10.1002/cpe.6197