The Journal of Supercomputing (2022) 78:8946–8976
https://doi.org/10.1007/s11227-021-04241-1

KS-DDoS: Kafka streams-based classification approach for DDoS attacks

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

Accepted: 30 November 2021 / Published online: 16 January 2022

Abstract

A distributed denial of service (DDoS) attack is the most destructive threat for internet-based systems and their resources. It stops the execution of victims by transferring large numbers of network traces. Due to this, legitimate users experience a delay while accessing internet-based systems and their resources. Even a short delay in responses leads to a massive financial loss. Numerous techniques have been proposed to protect internet-based systems from various kinds of DDoS attacks. However, the frequency and strength of attacks are increasing year-after-year. This paper proposes a novel Apache Kafka Streams-based distributed classification approach named KS-DDoS. For this classification approach, firstly, we design distributed classification models on the Hadoop cluster using highly scalable machine learning algorithms by fetching data from Hadoop distributed files system (HDFS). Secondly, we deploy an efficient distributed classification model on the Kafka Stream cluster to classify incoming network traces into nine classes in real-time. Further, this distributed classification approach stores highly discriminative features with predicted outcomes into HDFS for creating/updating models using a new set of instances. We implemented a distributed processing framework-based experimental environment to design, deploy, and validate the proposed classification approach for DDoS attacks. The results show that the proposed distributed KS-DDoS classification approach efficiently classifies incoming network traces with at least 80% classification accuracy.

Keywords Big data · Apache hadoop · Kafka streams · DDoS attacks · Distributed processing frameworks · Distributed H2O machine learning algorithms · Distributed streaming platform · CICDDoS2019 dataset

✉ Nilesh Vishwasrao Patil
nilesh.cse18@nitttrchd.ac.in

[1] Computer Science and Engineering, National Institute of Technical Teachers Training and Research, Panjab University, Chandigarh, India

[2] University Institute of Engineering and Technology, Panjab University, Chandigarh, India

1 Introduction

In this modern era, each enterprise has been shifting its sales services to online mode for growing profits and making it available anytime-anywhere for clients. There is an immense rise in Internet subscribers (4-billion+ users [1]) and IoT devices (100+ IoT devices connected to the Internet per second [2]). However, this significant improvement caused unprotected Internet routes, less-safe nodes, etc. Therefore, attackers use these opportunities for performing large-scale DDoS attacks on internet-based systems.

1.1 DDoS attack

A DDoS attack is one of the lethal attacks for internet-based systems, which immediately stop the working of victim systems by sending large numbers of unnecessary network traces [4]. The primary objective of this attack is to hide victim systems and their resources from benign requests [5]. For the execution of this type of attack: (1) Compromise large numbers of devices and (2) Perform a large-scale DDoS attack by transmitting large numbers of network traces towards the victim system in a coordinated manner using these compromised nodes. A typical example for performing a DDoS attack on the victim system is presented in Fig. 1. In this, an attacker compromised multiple devices via handlers. The handler is an intermediate program between attackers and compromised nodes for performing a large-scale attack.

1.2 Recent events information of attacks

Most of the countries have been fighting the COVID-19 pandemic situation since Jan. 2020. In Q1:2021 [6], DDoS attack incidents are declined by 28%+ when compared with Q1:2020. However, it increased by 46%+ compared to Q4:2020. Further,

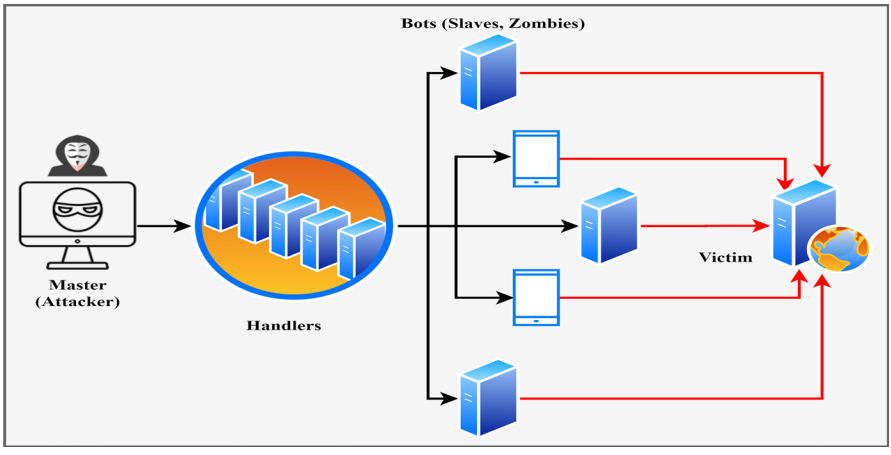

Fig. 1 An example of DDoS attack execution [3]

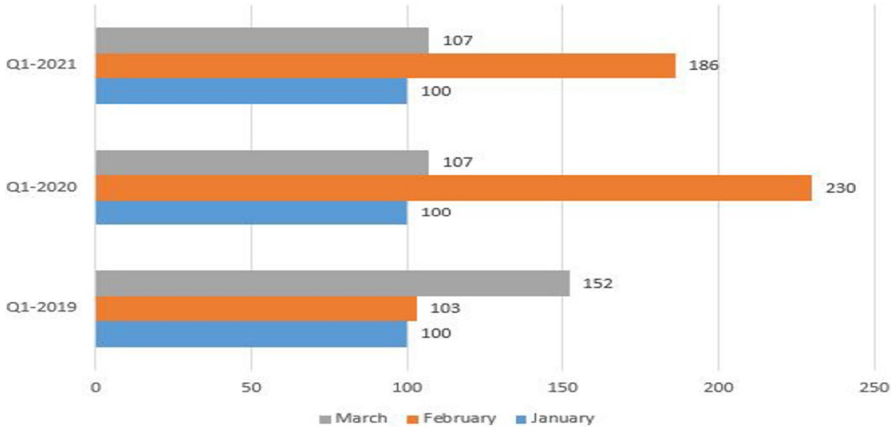

Fig. 2 No. of attack events in Q1:2019, Q1:2020, and Q1:2021 (100% ref. value assumed for Q1:2019) [6]

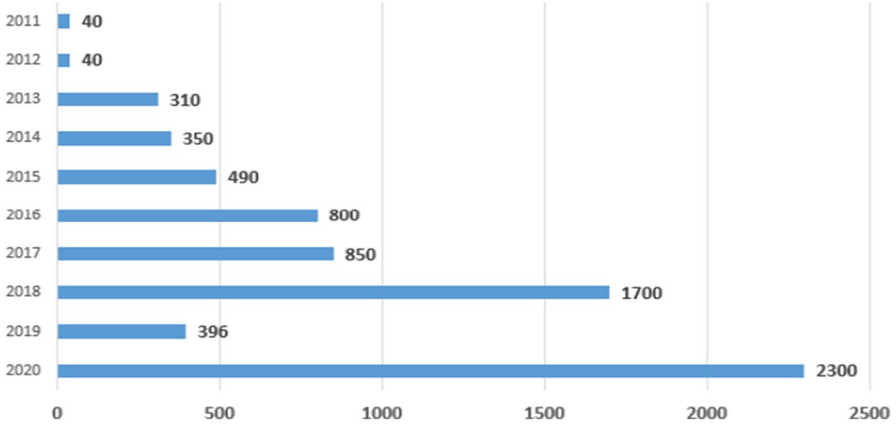

Fig. 3 Biggest DDoS attacks w.r.t. strength (in gigabytes) between 2011 and 2020

more than 40% of DDoS attack incidents are performed on internet-based systems in Jan. 2021, which means attack incidents in the other two months remain low.

The comparison of DDoS attack incidents between Q1-2019, Q1-2020, and Q1-2021 are presented in Fig. 2. Recent statistical data shows that both incidents and the strength of DDoS attacks are increasing year-after-year. The Q1-2019 DDoS attacks statistical data [7] shows that the number of attack incidents increased by 80%+ opposed to the Q4:2018. The year-wise largest DDoS attack for 2011–2020 is shown in Fig. 3.

1.3 Challenges

In this big data environment, the traditional framework-based classification approaches themselves became victims of large-scale DDoS attacks. This type of approach has several issues: (1) Requires more time for traffic analysis, (2) Explicit communication protocol needs to implement, (3) Failed to analyze network traffic in real-time, etc. A DPF systematically handles large numbers of network packets by employing multiple nodes. Further, intermediate communication between nodes, communication protocol, and metadata of data is managed by DPF. In the literature, few classification approaches are deployed on DPF compared to the traditional framework. Further, most of the DPF-based classification approaches for DDoS attacks are deployed on the Hadoop framework and fail to analyze incoming network traces in real-time. Additionally, existing DDoS attack classification approaches are designed and validated using outdated datasets. Therefore, it needs to implement a distributed classification approach using the recent CICDDoS2019 dataset on the DPF, such as Hadoop, and deploy it on distributed streaming platforms (DSP), such as Kafka Streams.

1.4 Apache Kafka streams, H2O machine learning library, and CICFlowMeter tool

A good DSP must have the following features:

1. To analyze the streaming data such as network traffic flows as it receives.
2. To support the producer-consumer architecture for real-time applications, which forms a loosely-coupled architecture. Therefore, multiple producers/consumers can publish/consume data concurrently and independently without delay.
3. To have features: analyze data in a distributed manner, extremely low latency, reliability, scalability, etc.

Apache Kafka Streams, H2O library, and CICFlowMeter are discussed in the following sub-sections.

1.4.1 Apache Kafka streams

Apache Kafka Streams [8, 9] is a distributed streaming environment for implementing real-time applications. The publishing/consuming feature of the Kafka platform helps to form a loosely coupled prototype between multiple sources (publishers) and sinks (consumers) in the big data [10] environment. The Apache Kafka streaming platform is commonly used:

1. To design real-time data applications, which receive streaming data such as incoming network traces from any number of sources and can be aggregated, analyzed, and consumed by any number of sinks.

2. To immediately respond to the streaming data such as network traces.

It has four primary APIs, such as Producer, Consumer, Streams, and Connector APIs. The Producer API help to distribute streaming data on multiple topics or sinks, while the Consumer API permits consuming the published data from multiple topics or sources. The Stream API provides a way to transfer the streaming data between topics, sinks, and sources. Finally, the Connector API provides a way to connect sources and sinks. The key features of Apache Kafka Streams (supported by Confluent) are: (1) High scalability, (2) Streaming data analysis, (3) High fault-tolerant, (4) Reliable, (5) Durable, (6) High performance, (7) Zero-downtime, and (8) Replication of data [9].

1.4.2 H2O machine learning library

The H2O [11] is an open-source and distributed in-memory machine learning platform for creating distributed models. It provides:

1. A way to design distributed models on DPF, DSP, or standalone machine.
2. Robust APIs to convert Spark/Python data frames to H2O frames and vice-versa.
3. High-scalability features for generated models when deployed on DPF or DSP.
4. Support for various programming languages: Python, R, Java, etc.
5. Support to generate highly scalable POJO (Plain old java object) code that is deployed in the production environment on DPF or DSP.

Multiple ways are available to design traditional (shallow) and non-traditional (deep) machine learning models, such as Scikit-learn, Theano, TensorFlow, Keras, PyTorch, etc. However, models built using these techniques suffer from scalability issues when deployed on DPF/DSP. The highly scalable H2O library provides a way to implement distributed and in-memory machine learning models. This type of a distributed model is specially implemented to deploy on distributed platforms: Apache Hadoop, Kafka Streams, Spark Streaming, etc. Hence, it is interesting to design a distributed classification model using H2O machine learning algorithms on the Hadoop cluster and deploy it on the Kafka Streams to classify incoming network traces into nine classes in real-time. Therefore, this distributed classification approach will provide various features, such as analyze network traces in real-time, loosely-coupled architecture, highly scalable, etc.

1.4.3 CICFlowMeter-V4.0

CICFlowMeter-V4.0 [12] is an open-source network flow generator tool. It creates network flows in offline (from PCAP) and online (from network interfaces) mode. It generates network flows with 83 attributes and stores them in a CSV file for designing models. A pictorial view of CICFlowMeter-V4.0 for capturing raw packets and generating network flows from live network traffic is shown in Fig. 4.

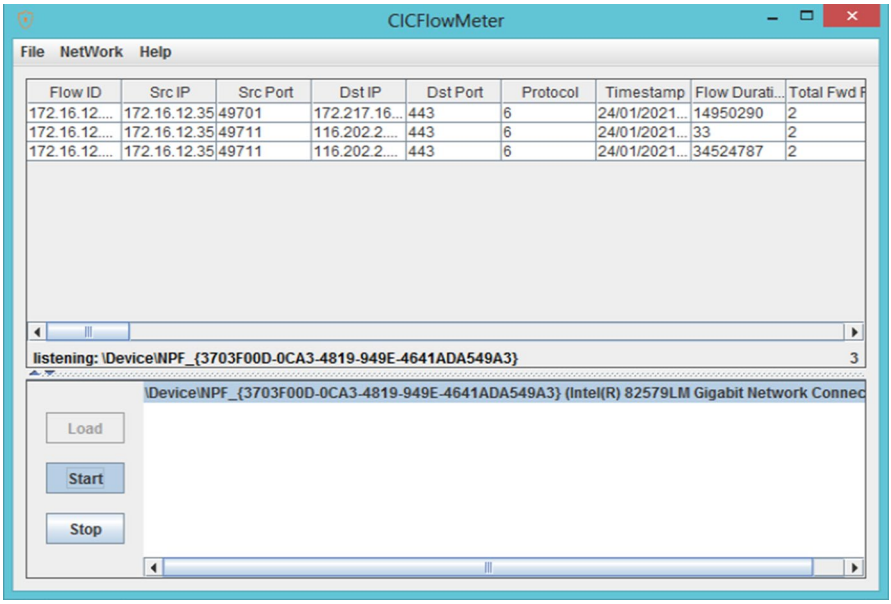

Fig. 4 CICFlowMeter-V4.0: Generating network flows

1.5 Contributions

The significant contributions of this paper are listed in the following:

1. Proposed a Kafka Streams-based distributed classification approach for DDoS attacks called KS-DDoS, which classify incoming network traces into nine classes in real-time.
2. Designed a distributed classification model on three nodes Hadoop cluster using highly scalable H2O machine learning algorithms by fetching designing instances (data) from HDFS.
3. Deployed an efficient distributed classification model for DDoS attacks on the Kafka Streams cluster to classify incoming network traces in real-time.
4. Proposed distributed KS-DDoS classification approach operates in an automated way: incoming network traces are immediately consumed by CICFlowMeter-V4.0 and publishes generated network flows on Kafka topics. After that, it extracts significant network traffic features, normalizes them, and gives to the distributed classification model. Finally, it analyzes preprocessed network traces and publishes the prediction on the Kafka topic to take action.
5. Proposed KS-DDoS distributes the workload (handling network traces, generating network flows, extracting highly discriminative features, classifying traces, storing significant features with predicted classes into HDFS, etc.) between multiple nodes of Kafka Stream clusters (‘KSC-1’ and ‘KSC-2’).
6. Designed and validated the proposed distributed KS-DDoS classification approach for DDoS attacks using the recent CICDDoS2019 dataset.

1.6 Structure of the article

This article is organized as: existing DPF-based classification approaches summarized in Sect. 2. Section 3 presents the working of the proposed distributed KS-DDoS classification approach. Section 4 provides experimental setup of the proposed KS-DDoS classification approach, results and discussion are presented in Sect. 5. Finally, Sect. 6 presents concluding remarks.

2 Related work

Several classification approaches have been proposed in the literature to protect internet-based systems and their resources from different kinds of DDoS attacks. Patil et al. [13] have systematically categorized DDoS attack classification approaches into two classes based on their deployment framework: traditional and DPF (Apache Hadoop, Spark, Kafka, etc.) based classification approaches. In the literature [3, 14–26], several researchers have systematically analyzed traditional framework based approaches and few [13] analyzed DPF-based classification approaches for DDoS attacks. A distributed framework itself has distributed archetype to collect, save, and analyze a massive amount of data (in this example, network traffic traces) on a cluster of machines. Therefore, the DPF-based classification approach is deployed on a cluster of machines to execute its classification job. In a cluster, each node communicates with others for sharing intermediate results during analysis tasks. It will help to improve the classification accuracy of approaches.

In this section, the primary purpose is to summarize existing DPF-based classification/detection approaches. In the literature, several researchers [27–30, 32–47], proposed DPF-based classification approaches. Most of them are deployed on the Apache Hadoop [48, 49] and stored data in HDFS [50]. Therefore, this type of classification approach efficiently analyzes large numbers of packets and classifies them quickly. However, the Hadoop framework is particularly designed for historical data analysis using offline batch processing mode. Therefore, Hadoop-based classification approaches for DDoS attacks are not capable of classifying network traces in real-time. In the DDoS attacks research area, the Hadoop framework is helpful for historical network traces analysis, designing distributed classification models, and retraining existing classification models for DDoS attacks. Further, few researchers [43] addressed classifying incoming network traces in real-time. Therefore, to analyze and classify incoming network traces in real-time, need to deploy the proposed classification model on DSP, such as Kafka Streams. Existing DPF-based classification approaches are summarized in Table 1.

It has been observed that few challenges still exist in the literature to protect internet-based systems. However, in this paper, the primary focus is on DPF/DSP-based classification approaches for DDoS attacks. Challenges are listed in the following:

– Most of the existing approaches are designed, deployed, and validated in a batch or micro-batch processing mode. Therefore, there is a need to deploy a classifi-

Table 1 Summary of DPF/DSP-based classification approaches

Authors	Parameters	Dataset	Testbed details
Lee and Lee [27]	"Time-interval", "threshold" and "unbalanced ratio"	Offline traffic: 500GB,1TB	Cluster: 1+10(Namenode+Datanodes)
Khattak et al. [28]	"Timestamp", "source and dest. (IP and port)", "flag", "protocol length"	MIT Lincoln DARPA2000	Karmasphere used to Implement Hadoop cluster
Zaho et al. [29]	"Avg. CPU usage", "avg. packet length", and "no. of TCP connections".	Attack flows: Five nodes	Hadoop cluster: 1+2+1 (Namenode+Datanodes+Server)
Dayama et al. [30]	"Time-interval","threshold" and "unbalanced ratio"	Offline traffic	Hadoop cluster:1+10 (Namenode+Datanodes)
Zhang et al. [31]	"Source and dest. (IP and port)"	Offline traffic	Not provided
Hameed et al. [32, 33]	"Timestamp", "source IP"	Mausezahn tool	Hadoop cluster: 1+10 (Namnode+Datanodes)
Alsirhani et al. [34, 35]	"Source and dest.(IP and port)", "packet length"	CAIDA07	Cluster: 01+03 (Master+Slaves)
Chhabra et al. [36]	Not provided	CAIDA07	Hadoop cluster:01+04 (Namenode+Datanodes)
Maheshwari et al. [37]	"Source IP"	Attack traffic use 7 nodes	Hadoop cluster:01+02 (Namenode+Datanodes)
Chen et al.[38]	"Query Rate", "Source IP space", "length", etc.	–	Spark:01 node
Patil et al. [39]	"Source and dest.	CAIDA,MIT,	Hadoop cluster: 01+30

Table 1 (continued)

Authors	Parameters	Dataset	Testbed details
	(IP and port)"	FIFA98	(Namenode+Datanodes)
Sharma et al. [40]	"Source and dest.	Attack traffic:	Hadoop cluster:01+02
	(IP and port)"	20-40 nodes	(Namenode+Datanodes)
Patil et al. [41]	"Source and dest.	CAIDA07, MIT-DDoS98	Hadoop cluster: 01+03
	(IP and port)"		(Namenode+Datanodes)
Gumaste et al. [42]	"Source and dest.	Captured flows	Open-stack based testbed
	IP", "Protocol"		01+01+03 (Controller+
			neutron+compute nodes)
Patil et al. [43]	"Source IP", "Timestamp",	Synthetic data	02 nodes Hadoop cluster
	"Destination IP"		03 nodes Spark Streaming cluster
Ahmed et al. [44]	"Source and dest. (IP and port)", "protocol"	UNSWNB15	Two testbeds: (1)High-end:02+01 (Worker+master)16GB RAM, 1TB HDD, and 3.5GHz Processor (2)Low-end:03+01 (Worker+master) 8GB RAM, 500GB HDD, and 2.7GHz Processor

cation approach on DSP such as Apache Kafka Streams that analyze incoming network traces in real-time.
- Few researchers designed their classification models for DDoS attacks using shallow and deep machine learning algorithms. This type of model is run efficiently on a single node. However, they faced the scalability issue after deploying on DPF/DSP. Therefore, there is a need to create a highly scalable classification model using distributed machine learning algorithms that will perform well even deployed on DPF/DSP.
- Most of the DPF-based approaches efficiently analyzed large numbers of network traces on multiple nodes. Therefore, it will help to divide the workload of the classification process. Further, it perfectly handles large-scale attacks.
- Most of the existing DPF-based approaches used a "counter-based detection" methodology to identify High-rate DDoS attacks. Therefore, this type of approach is failed to protect intenet-based systems from low-rate DDoS attacks.
- Most of the existing classification approaches for DDoS attacks are designed and validated using outdated datasets. Therefore, there is a need to design and validate the classification approach for DDoS attacks using recent and standard datasets, such as CICDDoS2019.

In this section, we explored the existing DPF/DSP-based DDoS attacks classification approaches. Further, listed challenges in the literature. In the next section, we present the working of the proposed distributed classification approach, such as KS-DDoS.

3 KS-DDoS: Kafka streams-based classification approach For DDoS attacks

This section presents the detailed working of the proposed distributed KS-DDoS classification approach for DDoS attacks. The logical framework of the distributed KS-DDoS classification approach is presented in Fig. 5. The distributed KS-DDoS performs two tasks:

- The first function of the proposed distributed KS-DDoS classification approach is to execute preprocessing and classification jobs on incoming network traces. To perform this task, we implemented two Kafka Streams clusters: 'KSC-1' and 'KSC-2'. The job of the 'KSC-1' cluster is to consume network flows from the "streaming-flows" topic, extract significant network traffic features from network flows, normalize extracted network traffic features, and publish them on the "highly-discriminative-features" topic. On the other hand, the 'KSC-2' cluster is continuously consuming features from "highly-discriminative-features" (replicated by 'KSC-1'), analyzing them, and publishing predictions on the "classification-outcome" to take action.
- The second function of the proposed distributed KS-DDoS classification approach is to store highly discriminative features with predicted classes

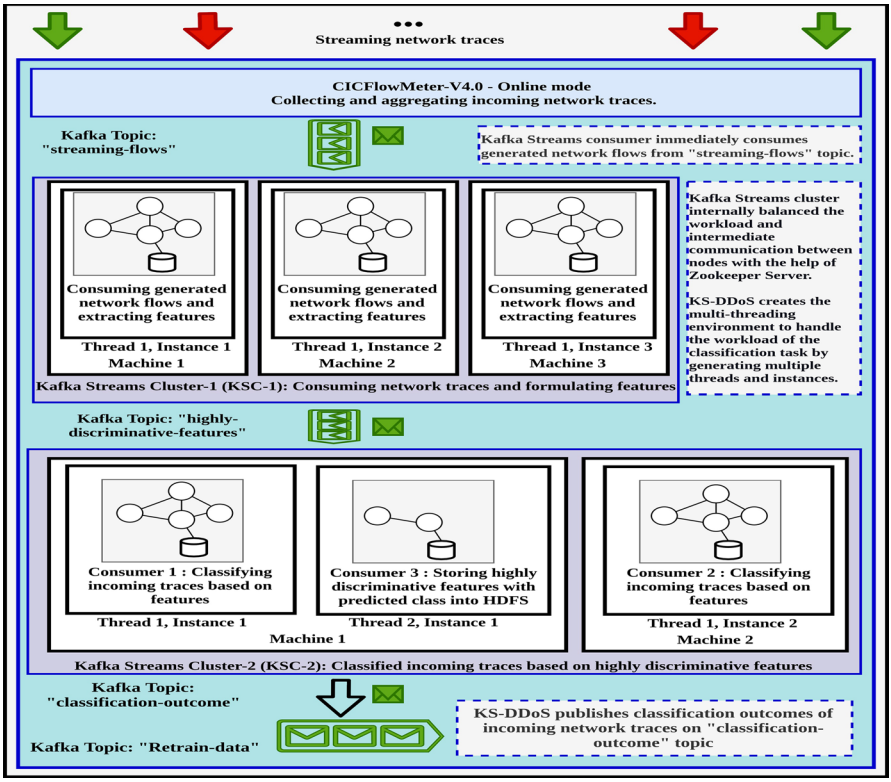

Fig. 5 Logical workflow of distributed KS-DDoS classification approach for DDoS attacks

into the HDFS from the "Retrain-data" topic. It helps to create new distributed classification models and retrain existing classification models for DDoS attacks.

Both functions of distributed KS-DDoS classification approach for DDoS attacks are independently running on two Kafka Streams clusters: 'KSC-1' and 'KSC-2'. Producers continuously publish messages on "streaming-flows" generated by CICFlowMeter-V4.0. The 'KSC-1' cluster consumes network flows from "streaming-flows". Then extract significant network traffic features from network flows, normalize extracted network traffic features, and publish them on the "highly-discriminative-features" topic. A distributed classification model for DDoS attacks runs on the 'KSC-2' by executing multiple threads and instances based on incoming network traces workload. The Zookeeper [51] server divides the workload of the KS-DDoS classification approach on a cluster of machines. An example of workload management on a two-node 'KSC-2' cluster is shown in Fig. 5: classification job is running on machine-1 and machine-2 while storing highly discriminative features with predicted outcomes into HDFS is running on machine-1. Highlights of the proposed distributed KS-DDoS classification approach are listed in the following:

- Supports loosely-coupled architecture as it follows the publisher-subscriber paradigm (distributed Kafka messaging system used for communication).
- Real-time preprocessing and classification jobs on incoming network traces using Kafka Streams API.
- Distributes the workload of preprocessing, classification, and storing jobs on clusters/nodes.
- Systematically stores highly discriminative network traffic features for creating new and updating the existing distributed DDoS attacks classification models.
- Supports features like high scalability, low latency, etc.

This section is divided into three sub-sections: (1) Preprocessing on incoming network traces, (2) Classification on preprocessed network flows, and (3) Stores significant features with predicted classes.

3.1 KS-DDoS: preprocessing on incoming network traces

The preprocessing job of the distributed KS-DDoS classification approach on incoming network traces is executed on the 'KSC-1' cluster. Therefore, the work of the 'KSC-1' cluster: capture incoming network traces, create network flows from network traces using CICFlowMeter-V4.0, extract significant network traffic features, normalize extracted features on the same scale, and finally publish features on the "highly-discriminative-features" topic. Further, we divided this section into three sub-sections: generate network flows from incoming network traces, extract significant network traffic features, and normalization of extracted features on the same scale.

3.1.1 Generate network flows from incoming network traces

The CICFlowMeter-V4.0 generates network flows from incoming network traces and continuously puts them in CSV files. The current version of CICFlowMeter-V4.0 creates 83 network traffic features for each network flow. The Kafka connect agent is authorized to consume network flows from CSV and publish them on the "streaming-flows" topic of the 'KSC-1' cluster.

3.1.2 Extract significant network traffic features

In this step, the 'KSC-1' cluster extracts 23 highly discriminative network traffic features from each network flow. In [52], 24 highly discriminative features are used to classify network flows. The RandomForestRegressor is employed to measure the significance of features. However, two network traffic features, such as "Fwd Header Length" and "Fwd Header Length.1" have almost the same values for each network flow. Further, the current version of CICFlowMeter-V4.0 removed the "Fwd Header Length.1". Therefore, we extract twenty-three significant network traffic features from consumed network flows, which are summarized in Table 2.

Table 2 Extracted twenty-three significant network traffic features from network flows

Features	Description
"Max packet length"	Maximum length of packet
"Min packet length"	Minimum length of packet
"Fwd packet length min"	Min. packet size in the forward direction
"Fwd packet length max"	Max. packet size in the forward direction
"Average packet size"	Avg. size of packets
"Packet length std"	Standard deviation of packet length
"Fwd packet length std"	Standard deviation of packet length in forward
"Total length of fwd packets"	Packet size in forward
"Fwd packets/s"	No. of packets forwarded in a second
"Flow IAT min"	Min. time between two packets in flow
"Flow IAT mean"	Mean time between two packets in flow
"Flow IAT max"	Max. time between two packets in flow
"Fwd IAT total"	Total time between two packets in forward
"Fwd IAT mean"	Mean time between two packets in forward
"Fwd IAT max"	Max. time between two packets in forward
"Min seq size forward"	Minimum seqment size in the forward direction
"Sub flow fwd bytes"	Avg. number of bytes in a sub-flow in the forward direction
"Destination port"	Port number which receives packets
"ACK flag count"	Number of packets with ACK
"Init win bytes forward"	Number of bytes initially in the window in the forward direction
"Fwd header length"	Header length of forwarded packets
"Protocol"	Protocol of flow
"Flow duration"	Duration of flow

3.1.3 Normalize values of extracted features

The next step is to normalize these twenty-three significant network traffic features on the same scale. It can be achieved using the "MinMax scaling technique provided by the sklearn.preprocessing". After this process, data values of network traffic features lie between zero and one. The mathematical equation for the scaling of data values is given as:

$$N\_\text{Value}_i = \frac{\text{DataValue}_i - \min(\text{DataValue})}{\max(\text{DataValue}) - \min(\text{DataValue})} \tag{1}$$

After normalization, the next task of the 'KSC-1' cluster is to publish normalized network traffic features on the "highly-discriminative-features" topic. Practically, we replicated these messages from one Kafka cluster ('KSC-1') to another ('KSC-2'). Therefore, the next task is to consume these messages from the "highly-discriminative-features" topic by distributed classification model and classify them into nine classes.

3.2 KS-DDoS: classification on preprocessed network traces

In this section, we present a novel distributed classification approach for identifying eight attacks: DDoS_DNS, DDoS_LDAP, DDoS_MSSQL, DDoS_NetBIOS, DDoS_UDP, DDoS_SYN, DDoS_NTP, and DDoS_SSDP. The distributed classification approach for DDoS attacks is designed using network traces from PCAPs of the recent CICDDoS2019 dataset based on distributed machine learning algorithms such as Gradient Boosting Machine (GBM), Naive Bayes (NB), and Distributed Random Forest (DRF). After that, an efficient distributed classification model (DRF-based) has been deployed on the 'KSC-2' cluster for classifying incoming network traces into nine classes: Benign (1), DDoS_DNS (2), DDoS_LDAP (3), DDoS_MSSQL (4), DDoS_NetBIOS (5), DDoS_UDP (6), DDoS_SYN (7), DDoS_NTP (8), and DDoS_SSDP (9).

The main objective of this distributed classification model for DDoS attacks is to classify incoming network traces into nine classes in real-time. We divided the proposed distributed classification approach into two parts: (i) Design process of distributed classification models for DDoS attacks and (ii) Real-time classification process after deploying an efficient distributed model on the 'KSC-2' cluster. The step-by-step flow of the proposed distributed classification model for DDoS attacks is presented in Fig. 6 (designing process) and Fig. 7 (real-time classification

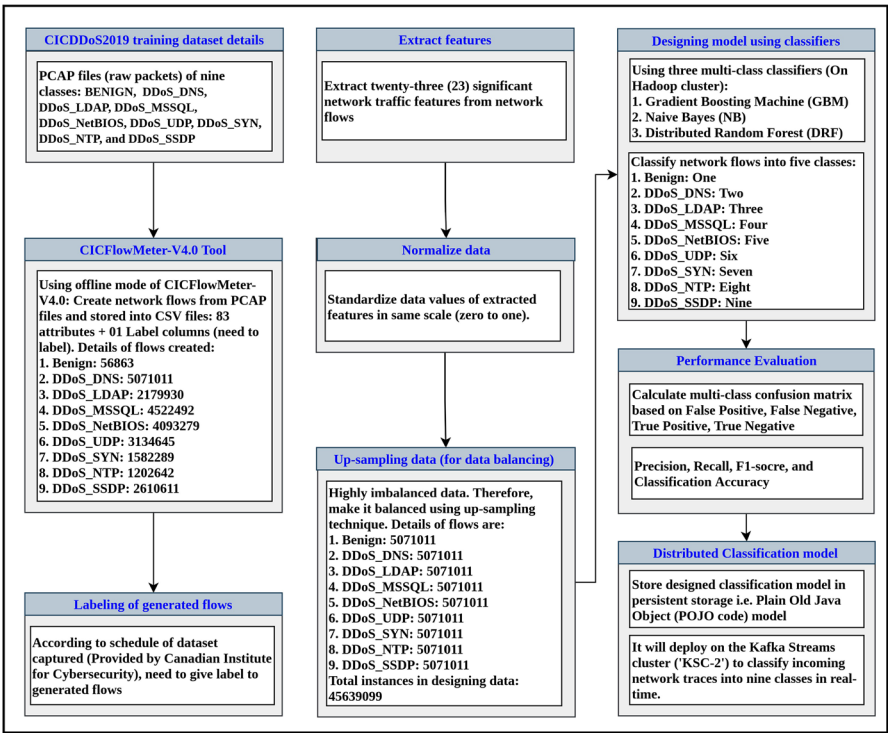

Fig. 6 Design flow of distributed classification model

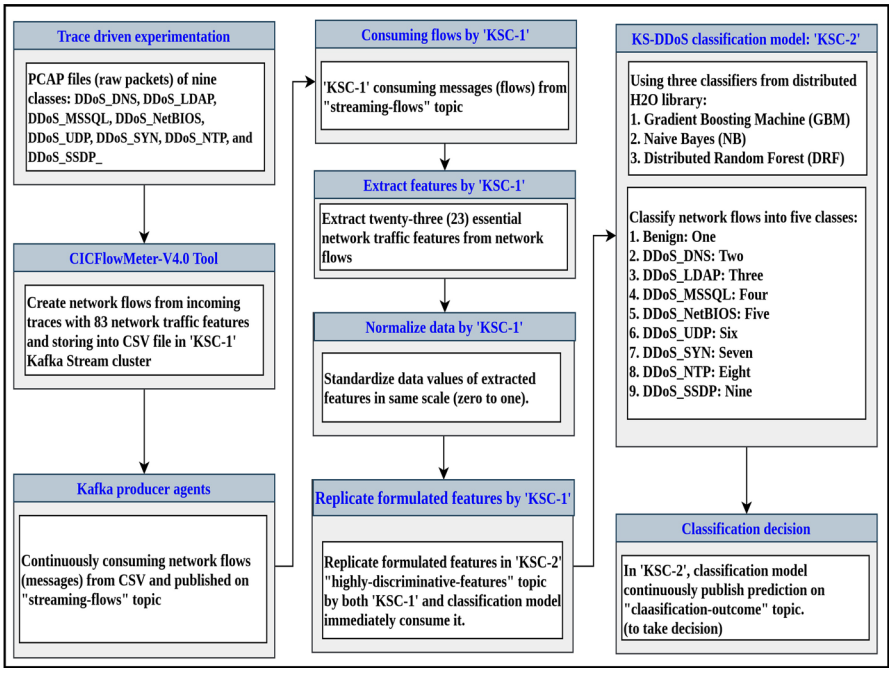

Fig. 7 Real-time classification workflow of distributed classification model

process). Therefore, we classify this section into three sub-sections: summary of the CICDDoS2019 dataset, designing process, and real-time classification process of the distributed classification model for DDoS attacks.

3.2.1 CICDDoS2019 dataset

The CICDDoS2019 [52] dataset is a combined project of the "Canadian Communications Security Establishment (CSE) and Canadian Institute for Cybersecurity (CIC)". This dataset comprises both legitimate and multiple DDoS attack scenarios. It has been provided in both PCAPs (network traces) and CSVs (flows with label). Further, they have supplied a schedule of the execution of each scenario. For implementing machine/deep learning-based models, feature data with labels are enough to design models. However, provided CSVs have few issues: (1) Imbalanced data that influences the accuracy, (2) Few columns have 50+% null values, which skip essential facts, (3) Features naming convention is different as given on the portal, and so on. Therefore, instead of using available CSVs, we created network flows from PCAPs for various scenarios, such as Benign, DDoS_DNS, DDoS_LDAP, DDoS_MSSQL, DDoS_NetBIOS, DDoS_UDP, DDoS_SYN, DDoS_NTP, and DDoS_SSDP using the current version of the CICFlowMeter-V4.0. The created network flows comprise 83 network traffic features with one empty label column, which need to be updated as per the schedule of PCAPs given on the dataset portal.

3.2.2 KS-DDoS: design workflow

The step-by-step process to implement distributed classification models for DDoS attacks is presented in Fig. 6. For this process, we collected various PCAPs of DDoS_DNS, DDoS_LDAP, DDoS_MSSQL, DDoS_NetBIOS, DDoS_UDP, DDoS_SYN, DDoS_NTP, DDoS_SSDP and Benign scenarios. The first step is to create network flows from PCAPs using the CICFlowMeter-V4.0 and label created flows according to the schedule given on the dataset portal. The number of network flows available in each class is Benign: 56863, DDoS_DNS → 5071011, DDoS_LDAP → 2179930, DDoS_MSSQL → 4522492, DDoS_NetBIOS → 4093279, DDoS_UDP → 3134645, DDoS_SYN → 1582289, DDoS_NTP → 1202642, and DDoS_SSDP → 2610611. The second step is to extract twenty-three significant network traffic features from each network flow. The third step is normalized data values of twenty-three network traffic features on the same scale ranging from zero to one.

As we have seen, created network flows for each class are highly imbalanced (particularly, benign scenario). It impacts badly on the classification accuracy. Therefore, we have to take an essential step on each class of the sample that is up-sampling. The most number of instances belongs to DDoS_DNS, so we up-sampled the remaining classes to 5071011. Therefore, the total number of instances in the sample is 45 million+ and store these up-sampled instances in the HDFS. The next step is to design a distributed classification model for DDoS attacks. We implemented this use-case using three distributed shallow machine learning algorithms: GBM, NB, and DRF. Further, it needs to implement these distributed classification models on the Hadoop cluster ('HC-1') because we have to deploy an efficient distributed classification model on the Kafka Streams platform.

Hence, we designed distributed classification models for DDoS attacks on the 'HC-1' cluster using a highly scalable distributed H2O machine library by fetching designing instances from the HDFS. The next task is to evaluate the performance parameters: Precision, Recall, and F1-score. The performance evaluation of these algorithms is discussed in Sect. 5. The final step is to store this distributed classification model in the persistent storage for deploying an efficient model on the 'KSC-2' cluster. We will get a POJO code and deploy it in the production environment.

3.2.3 KS-DDoS: real-time classification workflow (after deployment)

The second part of the distributed classification approach starts with deploying an efficient distributed classification model on the 'KSC-2' Kafka Streams cluster. Figure 7 presents the step-by-step process of the real-time classification for DDoS attacks. In trace-driven experimentation, the CICFlowMeter-V4.0 tool creates network flows from incoming network traces. Then, Kafka producer agents in the 'KSC-1' cluster continuously publish these network flows on the "streaming-flows" topic. The 'KSC-1' cluster immediately consumes these messages and extracts twenty-three significant network traffic features from each message. The next step is to normalize extracted features on the same scale ranging from 0 to 1 with a time window of 1 s and publish them on the "highly-discriminative-features" topic.

Practically, 'KSC-1' replicates messages into the 'KSC-2'. After that, distributed KS-DDoS classification model immediately consumes messages from the "highly-discriminative-features" topic, analyzes them, and classifies them into nine classes in real-time. Finally, the classification approach publishes the prediction results on the "classification-outcome" topic to take action.

3.3 Storing features with outcomes in HDFS

Another function of the distributed KS-DDoS classification approach is to stores twenty-three highly discriminative features with predicted outcomes into HDFS. A Kafka agent "store-data-agent" is consumed significant features from the "highly-discriminative-features" topic and predicted outcomes from the "classification-outcome" topic. After that, the "store-data-agent" is combined each instance of features with their predicted outcomes and store them in HDFS. It helps to design new and update existing distributed classification models for DDoS attacks using a set of new samples.

In this section, we explored the detailed working of the proposed distributed KS-DDoS classification approach. In the next section, we present the experimental setup of the proposed distributed KS-DDoS classification approach. It helps to design, deploy, and validate the proposed KS-DDoS classification approach.

4 Experimental setup

In this section, we present the experimental setup details of the proposed distributed KS-DDoS classification approach. The experimental setup of the proposed KS-DDoS classification approach for DDoS attacks is shown in Fig. 8. In this, we considered three source networks and one incoming edge point towards the victim network. Therefore, we implemented one Kafka cluster 'KSC-1' to capture incoming network traces and extract significant features. Further, we designed one Kafka cluster 'KSC-2' for deploying the proposed distributed classification model for DDoS attacks to classify incoming network traces into nine classes in real-time. Additionally, we implemented the Hadoop cluster 'HC-1' to store significant features with predicted outcomes and design distributed classification models. The details for clusters/nodes are given in the following:

- Three source networks ('SN-1', 'SN-2', and 'SN-3'):
  Benign and DDoS attacks traffic traced towards victim network, i.e., CICDDoS2019 dataset.
- One Kafka Streams cluster ('KSC-1'):
  Implemented a three-node Kafka Stream cluster ('KSC-1') to capture incoming network traces, extract network traffic features, and normalize features.
- Hadoop cluster ('HC-1'):

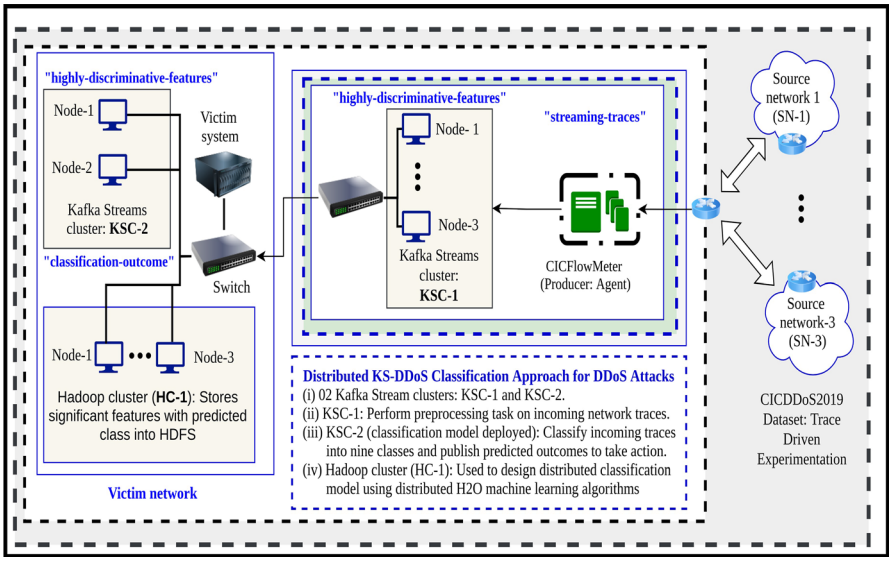

Fig. 8 Experimental steup for the proposed distributed KS-DDoS classification approach

Deployed three nodes Hadoop cluster ('HC-1') to store twenty-three features with predicted outcomes and design distributed classification models using highly scalable distributed H2O machine learning algorithms.

– One Kafka Streams cluster ('KSC-2'):

Implemented two nodes Kafka Stream cluster 'KSC-2' for deploying an efficient distributed classification model to classify incoming network traces into nine classes in real-time.

Two Kafka distributed messaging system topics have been created to publish and consume messages. In 'KSC-1', two topics are created:

1. "streaming-flows": It is used to publish generated network flows CICFlowMeter-V4.0 from incoming network traces.
2. "highly-discriminative-features": It is used to publish normalized twenty-three significant network traffic features.

Further, in the 'KSC-2' cluster, three Kafka topics are created to publish and consume messages:

1. "highly-discriminative-features": It consumes network traffic features and messages are replicated from 'KSC-1' to 'KSC-2'.
2. "classification-outcome": It is used to publish predicted outcomes of the proposed distributed classification model to take action on incoming network traces.
3. "Retrain-data": The messages (twenty-three features with predicted outcomes) published on this topic are stored into HDFS by "store-data-agent".

In this section, we summarized the experimental setup of the proposed distributed KS-DDoS classification approach. In the next section, we evaluate the performance of the proposed KS-DDoS classification approach by executing various scenarios.

5 Results and discussion

In this section, we evaluate the performance of the proposed distributed KS-DDoS classification approach for DDoS attacks. The proposed KS-DDoS classification approach is deployed on 'KSC-1', 'KSC-2', and 'HC-1' in a distributed manner. The job of 'KSC-1', 'KSC-2', and 'HC-1' is to perform preprocessing on incoming network traces, classify incoming network traces in real-time, and store features with predicted classes in HDFS, respectively.

We have considered two cases for evaluating the performance of this KS-DDoS classification approach, such as Case-I: While designing distributed classification models and Case-II: After deployment of an efficient distributed classification model on 'KSC-2'. In this paper, we calculated three performance metrics. A mathematical representation of these metrics for multi-class classification is given in the following: Precision ($P_{\text{m\_class}}$), Recall ($R_{\text{m\_class}}$), and $F1$-score ($F1_{\text{m\_class}}$):

1. $P_{\text{m\_class}} = \frac{\sum_{i=1}^{n} \frac{\text{True\_Positive}_i}{(\text{True\_Positive}_i + \text{False\_Positive}_i)}}{n}$, *where n = number of classes (in this instance, nine classes)*
2. $R_{\text{m\_class}} = \frac{\sum_{i=1}^{n} \frac{\text{True\_Positive}_i}{(\text{True\_Positive}_i + \text{False\_Negative}_i)}}{n}$
3. $F1_{\text{m\_class}} = \frac{2 * \text{Precision}_{\text{multiclass}} * \text{Recall}_{\text{multiclass}}}{(\text{Precision}_{\text{multiclass}} + \text{Recall}_{\text{multiclass}})}$

We designed and tested the proposed distributed classification model using the recent CICDDoS2019 dataset. For evaluating the performance of the proposed distributed classification model, the details of the sample dataset for Case-I are presented in Table 3. However, the class-wise network traces are highly imbalanced. Therefore, we up-sampled network flows and designed this classification model using three highly scalable machine learning algorithms: GBM, DRF, and NB. After testing this classification model using these algorithms for Case-I, we visualized multi-class confusion matrices in Figs. 9, 10, and 11, and standard evaluation parameters in Table 4.

DRF follows the bagging approach. It creates multiple trees independently and combines results at the end of the process (by average) to solve a problem. In this use-case, DRF has given better accuracy than GBM and NB. The average classification accuracy of classifiers for nine classes is GBM (79.94%), NB (59.55%), and DRF (80.07%).

For evaluation of the classification model after deployment on the 'KSC-2' for Case-II, we considered four scenarios of the CICDDoS2019 dataset network traffic flows based on different combinations of the target classes. The details of each scenario are presented in Table 5. We deployed the distributed DRF-based classification

Table 3 Network traffic traces information to evaluate the proposed KS-DDoS: Case-I (during implementation)

Scenario	Traffic classes	No. of flows	No. of up-sampled flows	No. of validating flows	Correctly classified flows by		
					GBM	NB	DRF
Designing	Benign	56863	5071011	1672909	1672039	346078	1671196
	DDoS_DNS	5071011	5071011	1673500	494335	24747	505086
	DDoS_LDAP	2179930	5071011	1673538	1550755	1652913	1553900
	DDoS_MSSQL	4522492	5071011	1673243	1611971	1501213	1612710
	DDoS_NetBIOS	4093279	5071011	1673390	1647724	1581554	1650067
	DDoS_UDP	3134645	5071011	1673525	1337334	298	1435797
	DDoS_SYN	1582289	5071011	1673497	1672673	1672452	1673193
	DDoS_NTP	1202642	5071011	1673640	1668642	1308906	1669306
	DDoS_SSDP	2610611	5071011	1673384	383293	880610	287611

GBM gradient boosting machine, *NB* naive bayes, *DRF* distributed random forest

Benign → 1, DDoS_DNS → 2, DDoS_LDAP → 3, DDoS_MSSQL → 4, DDoS_NetBIOS → 5

DDoS_UDP → 6, DDoS_SYN → 7, DDoS_NTP → 8, DDoS_SSDP → 9

Table 4 Performance evaluation of KS-DDoS: For Case-I, while designing a distributed classification model on 'HC-1' by fetching designing data from HDFS

Classifier	Metrics	Target classes								
		1	2	2	4	5	6	7	8	9
GBM	Precision	1.00	0.79	0.57	0.92	1.00	0.52	1.00	0.99	0.55
	Recall	1.00	0.29	0.93	0.96	0.98	0.80	1.00	1.00	0.23
	F1 Score	1.00	0.43	0.70	0.94	0.99	0.63	1.00	0.99	0.32
	Average classification accuracy: 79.94%									
NB	Precision	0.99	0.34	0.51	0.40	0.98	0.26	0.92	0.95	0.31
	Recall	0.20	0.02	0.99	0.90	0.94	0.00	1.00	0.78	0.53
	F1 Score	0.34	0.03	0.67	0.56	0.96	0.00	0.96	0.86	0.39
	Average classification accuracy: 59.55%									
DRF	Precision	1.00	0.80	0.57	0.92	1.00	0.53	1.00	0.99	0.57
	Recall	1.00	0.30	0.93	0.96	0.99	0.86	1.00	1.00	0.17
	F1 Score	1.00	0.44	0.71	0.94	0.99	0.64	1.00	1.00	0.26
	Average classification accuracy: 80.07%									

GBM gradient boosting machine, *NB* naive bayes, *DRF* distributed random forest

Benign → 1, DDoS_DNS → 2, DDoS_LDAP → 3, DDoS_MSSQL → 4, DDoS_NetBIOS → 5

DDoS_UDP → 6, DDoS_SYN → 7, DDoS_NTP → 8, DDoS_SSDP → 9

model for DDoS attacks on the 'KSC-2'. For example, the DDoS_SYN raw packets are traced, then generate network flows using CICFlowMeter and publishes on "streaming-traces" topic in 'KSC-1'. After that, extracted essential network traffic features, normalized them, and published them on "highly-discriminative-features"

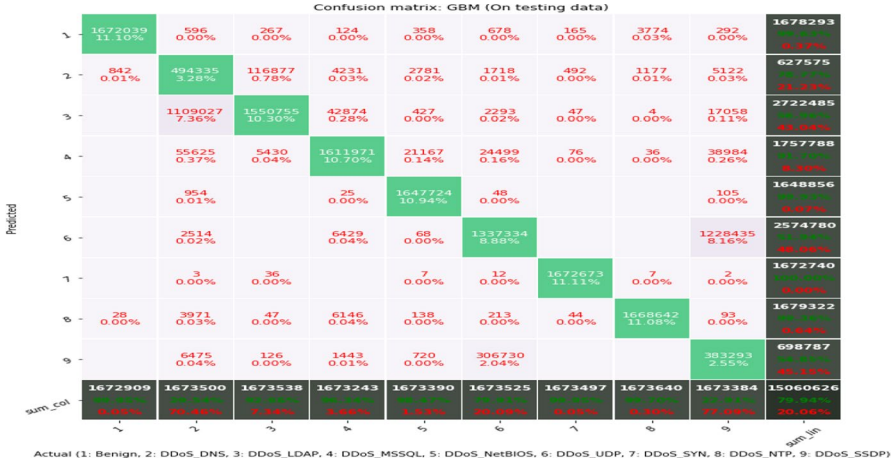

Fig. 9 Multi-class confusion matrix for GBM-based distributed classification approach

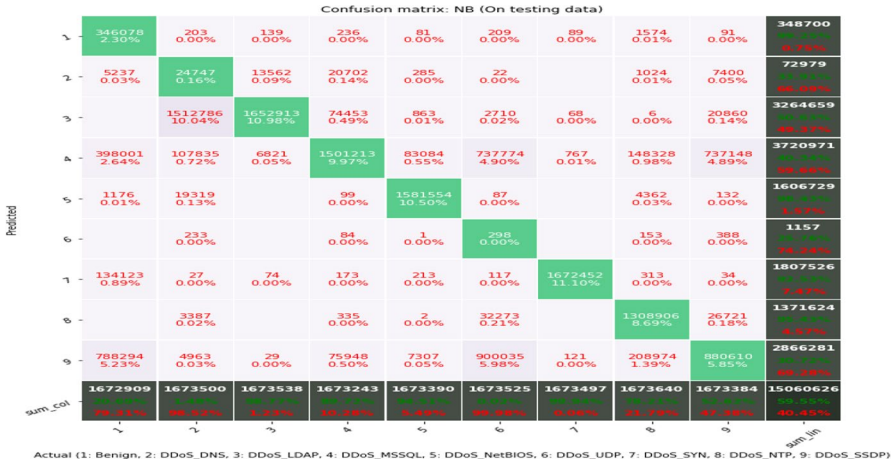

Fig. 10 Multi-class confusion matrix for NB-based distributed classification approach

in 'KSC-2'. Finally, the DRF-based distributed classification model immediately analyzes and publish the prediction on "classification-outcome" topic to take action. The performance evaluation of four scenarios is presented in Table 6 (case-II) and visualized their multi-class confusion matrices in Figs. 12, 13, 14, and 15.

For scenario-I, we traced three classes traffic: Benign, DDoS_UDP, & DDoS_SYN. For scenario-II, traced six classes traffic: Benign, DDoS_LDAP, DDoS_MSSQL, DDoS_NetBIOS, DDoS_UDP, & DDoS_SYN. After that, scenario-III, traces five classes: Benign, DDoS_DNS, DDoS_MSSQL, DDoS_UDP, & DDoS_SYN. Finally, scenario-IV traced five classes: Benign, DDoS_MSSQL, DDoS_SYN, DDoS_NTP, & DDoS_SSDP.

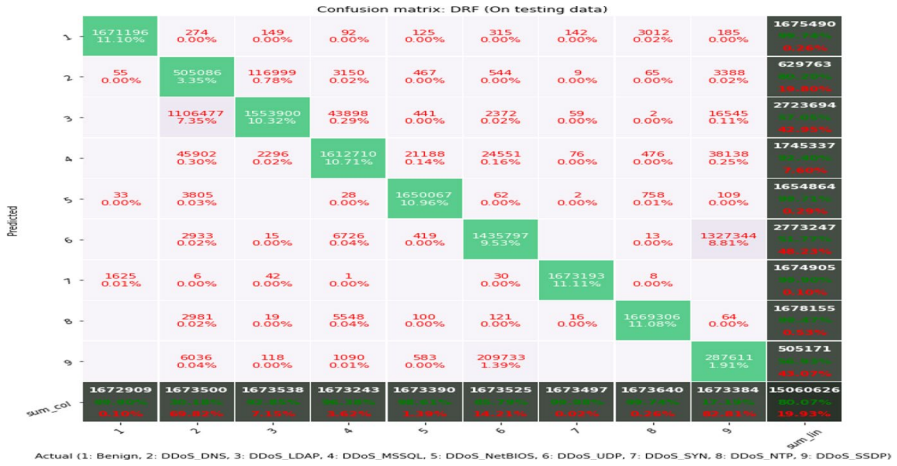

Fig. 11 Multi-class confusion matrix for DRF-based distributed classification approach

Table 5 Network traces details to evaluate the performance of KS-DDoS: Case-II (After deployment)

Scenario	Deployed	Classes	Predicting traces	Traces correctly identified
Scenario-I	DRF	Benign (1)	56863	56859
		DDoS_UDP (6)	3134645	2370013
		DDoS_SYN (7)	1582289	1581938
		Benign (1)	56863	56856
Scenario-II	DRF	DDoS_LDAP (3)	2179930	2028369
		DDoS_MSSQL (4)	4522492	4358305
		DDoS_NetBIOS (5)	4093279	4036555
		DDoS_UDP (6)	3134645	2141484
		DDoS_SYN (7)	1582289	1581970
		Benign (1)	56863	56856
Scenario-III	DRF	DDoS_DNS (2)	5071011	1476441
		DDoS_MSSQL (4)	4522492	4358381
		DDoS_UDP (6)	3134645	1913171
		DDoS_SYN (7)	1582289	1581969
		Benign (1)	56863	56860
Scenario-IV	DRF	DDoS_MSSQL (4)	4522492	4358385
		DDoS_SYN (7)	1582289	1581961
		DDoS_NTP (8)	1202642	1199986
		DDoS_SSDP (9)	2610611	1709281

Table 6 Performance evaluation of KS-DDoS: For Case-II, after deployment of DRF-based classification model on 'KSC-2'

Scenario	Metrics	Target classes								
		1	2	2	4	5	6	7	8	9
Scenario-I	Precision	0.99	0.00	0.00	0.00	0.00	1.00	1.00	0.00	0.00
(Benign, DDoS_UDP, DDoS_SYN)	Recall	1.00	0.00	0.00	0.00	0.00	0.76	1.00	0.00	0.00
	F1 Score	0.99	0.00	0.00	0.00	0.00	0.86	1.00	0.00	0.00
	Average classification accuracy: 83.98%									
Scenario-II	Precision	0.97	0.00	0.94	0.98	1.00	0.99	1.00	0.00	0.00
(Benign, DDoS_LDAP, DDoS_MSSQL,	Recall	1.00	0.00	0.93	0.96	0.99	0.68	1.00	0.00	0.00
DDoS_NetBIOS, DDoS_UDP, DDoS_SYN)	F1 Score	0.99	0.00	0.94	0.97	0.99	0.81	1.00	0.00	0.00
	Average classification accuracy: 91.23%									
Scenario-III	Precision	0.97	0.99	0.00	0.96	0.00	0.99	1.00	0.00	0.00
(Benign, DDoS_DNS, DDoS_MSSQL,	Recall	1.00	0.29	0.00	0.96	0.00	0.61	1.00	0.00	0.00
DDoS_UDP, DDoS_SYN)	F1 Score	0.98	0.45	0.00	0.96	0.00	0.75	1.00	0.00	0.00
	Average classification accuracy: 65.33%									
Scenario-IV	Precision	0.95	0.00	0.00	0.98	0.00	0.00	1.00	0.99	0.99
(Benign, DDoS_MSSQL, DDoS_SYN,	Recall	1.00	0.00	0.00	0.96	0.00	0.00	1.00	1.00	0.65
DDoS_NTP, DDoS_SSDP)	F1 Score	0.97	0.00	0.00	0.97	0.00	0.00	1.00	0.99	0.79
	Average classification accuracy: 89.29%									

Benign → 1, DDoS_DNS → 2, DDoS_LDAP → 3, DDoS_MSSQL → 4, DDoS_NetBIOS → 5, DDoS_UDP → 6,
DDoS_SYN → 7, DDoS_NTP → 8, DDoS_SSDP → 9

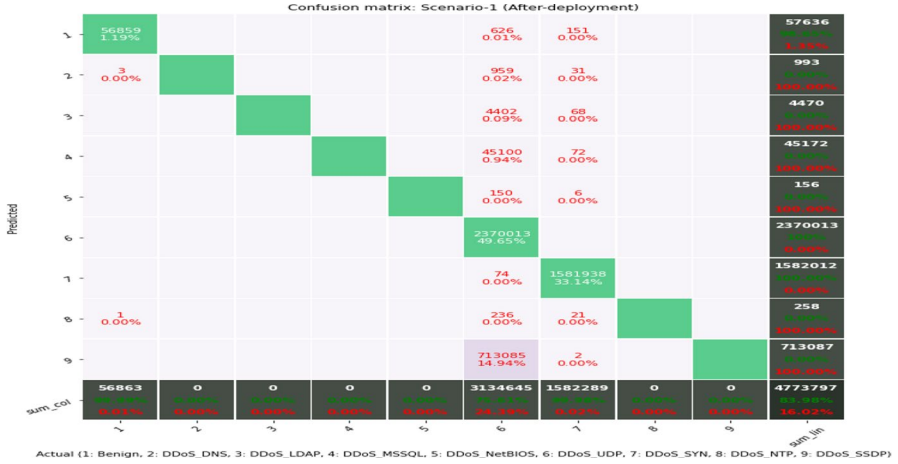

Fig. 12 Scenario-I: Benign, DDoS_UDP, DDoS_SYN

Fig. 13 Scenario-II: Benign, DDoS_LDAP, DDoS_MSSQL, DDoS_NetBIOS, DDoS_UDP, DDoS_SYN

5.1 Performance improvement after implementation of the proposed KS-DDoS classification system

For this case study-based demonstration, we have executed four scenarios for determining the classification accuracy of the proposed distributed DRF-based classification model. The proposed distributed scheme gives a better classification accuracy for the following four scenarios: Scenario-I: 83.98%, Scenario-II: 91.23%, Scenario-III: 65.33%, and Scenario-IV: 89.29%. In general, attackers do launch one type of DDoS attack at a time. Therefore, the proposed system will provide at least 83% accuracy when deployed in the production environment.

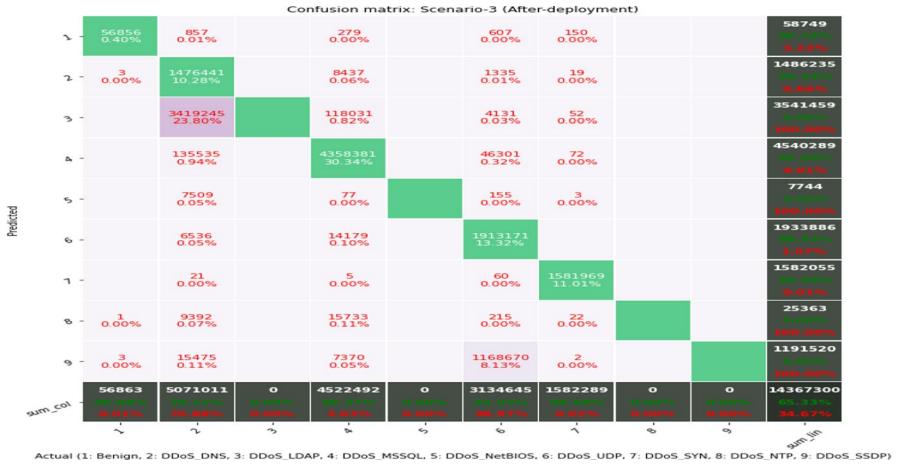

Fig. 14 Scenario-III: Benign, DDoS_DNS, DDoS_MSSQL, DDoS_UDP, DDoS_SYN

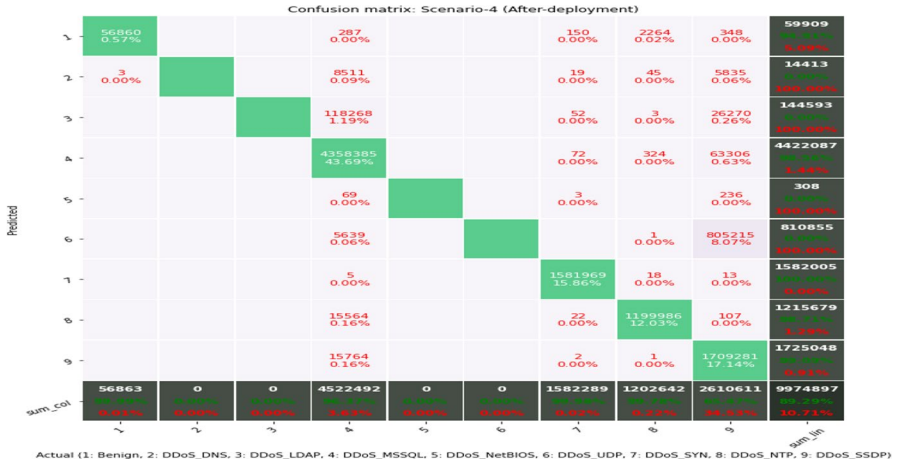

Fig. 15 Scenario-II: Benign, DDoS_MSSQL, DDoS_SYN, DDoS_NTP, DDoS_SSDP

5.2 Complexity analysis

In traditional framework-based DDoS attack classification approaches, network traces are analyzed at the same node. Therefore, the time complexity of such classification approaches is $O(\mathrm{NT})$, where NT is the number of network traces analyzed by approaches [53]. However, in DPF/DSP-based classification approaches, the analysis task of network traces is distributed between multiple machines and executed in parallel. Therefore, the complexity of the DPF/DSP-based classification approaches is also distributed between the number of machines, say m. To measure the complexity of the proposed DSP-based classification approach, we assume each

machine of a cluster equally analyzes network traces. Therefore, the complexity of DSP-based classification approach is $O\left(\frac{\mathrm{NT}}{m}\right)$. Further, in this case, we have to consider one more parameter, i.e., the intermediate communication cost between machines while communicating intermediate results between machines.

Let us assume the intermediate communication cost is $O(\mathrm{CC})$. Therefore, the Total combined complexity (TCC) of the proposed system is $\mathrm{TCC} = O\left(\frac{\mathrm{NT}}{m}\right) + O(\mathrm{CC})$. However, distributed frameworks are specially designed to analyze a massive volume of data. Hence O(CC) is negligible when compared with $O(\mathrm{NT})$. Therefore, the TCC of the DFP/DSP-based classification approach for DDoS attacks is $O\left(\frac{\mathrm{NT}}{m}\right)$. It reveals that the time complexity of the DPF/DSP-based classification method is inversely proportional to no. of machines (m) in a cluster.

5.3 Comparison with existing approaches

In this section, we presented the comparison of proposed distributed KS-DDoS classification approach with existing approaches [27, 28, 30, 32–37, 40, 41, 41, 43, 46, 54] implemented using DPF/DSP and traditional frameworks in Tables 7 and 8 respectively.

Table 7 Comparison of KS-DDoS with DPF/DSP-based classification approaches

System/Ref. No. (Year)	Deployed on	Public dataset	DDoS attacks	Historical analysis	Real-time analysis	Real-time action
[27] (2011)	Hadoop	–	✓	×	×	×
[28] (2011)	Hadoop	–	✓	×	×	×
[30] (2015)	Hadoop	–	✓	×	×	×
[54] (2017)	Spark	–	✓	×	✓	×
HADEC [32, 33] (2018)	Hadoop	–	✓	×	×	×
[37] (2018)	Hadoop	–	✓	×	×	×
[34, 35] (2018)	Spark	–	✓	×	×	×
[36] (2018)	Hadoop	CAIDA	✓	×	×	×
E-Had[39] (2019)	Hadoop	CAIDA, MITLincoln	✓	×	×	×
[41] (2019)	Hadoop	CAIDA, MITLincoln	✓	×	×	×
[40] (2020)	Hadoop	–	✓	×	×	×
S-DDoS[43] (2020)	Spark, Hadoop	–	✓	×	✓	×
SAD-F[44] (2020)	Hadoop, Spark	UNSW-NB2015	✓	×	✓	×
KS-DDoS (Proposed)	Kafka, Hadoop	CICDDoS2019	✓	✓	✓	✓

Table 8 Comparison of KS-DDoS with traditional framework-based classification approaches

System/Ref. No. (Year)	DDoS attacks	Historical analysis	Real-time analysis	Real-time action	Handle massive data (traces)
TIDS [55] (2016)	✓	×	×	×	×
D-FACE [5] (2018)	✓	×	×	×	×
D-FAC [56] (2018)	✓	×	×	×	×
FL-DDoS [57] (2018)	✓	×	×	×	×
[58] (2019)	✓	×	×	×	×
[59] (2019)	✓	×	×	×	×
[60] (2019)	✓	×	×	×	×
[61] (2020)	✓	×	×	×	×
KS-DDoS (Proposed)	✓	✓	✓	✓	✓

Most of the DPF-based classification approaches [27, 28, 30, 32, 33, 36, 37, 40, 41, 41] deployed on the Apache Hadoop framework. This type of system systematically stores and analyzes large numbers of network packets on a group of machines. However, the Apache Hadoop-based use-cases are specially designed for analyzing a massive amount of data in offline mode. Therefore, the Hadoop-based classification approaches for DDoS attacks are failed to classify network traces in real-time.

Few [34, 35, 43, 46, 54] researchers proposed Apache Spark-based classification approaches for DDoS attacks. This type of approach analyzes network streams in micro-batch processing mode (near-to-live). However, the Spark-based approaches are failed to provide an automated way. However, in the proposed distributed KS-DDoS classification approach, the classification model is designed on the Hadoop cluster and deployed on the Kafka Streams cluster. Further, employed distributed messaging system for providing loosely-coupled architecture with an automated way to the proposed KS-DDoS. The KS-DDoS classifies incoming network traces into nine classes in real-time: Benign (1), DDoS_DNS (2), DDoS_LDAP (3), DDoS_MSSQL (4), DDoS_NetBIOS (5), DDoS_UDP (6), DDoS_SYN (7), DDoS_NTP (8), and DDoS_SSDP (9).

Sharafaldin et al. [52] have generated a realistic dataset and proposed a DDoS attacks taxonomy based on different types of attacks captured in the CICDDoS2019 dataset. Further, they proposed a classification approach for DDoS attacks. According to their performance evaluation, precision values for classifiers ID3, RF, NB, and LR is 0.78, 0.77, 0.41, and 0.25, respectively. When we compared these values with the proposed distributed classification approach precision value while designing the model, the DRF-based classification model given a better precision value (0.8).

6 Conclusions

A DDoS attack is the most destructive threat for internet-based systems and their resources. It stops the execution of victims by sending large numbers of network traces. Due to this, legitimate users experience a delay while accessing internet-based systems. Few challenges exist in the traditional framework-based detection

approaches: defending approach itself became the victim of attacks while analyzing large numbers of packets, fails to examine network traffic in real-time, etc. Further, most of the existing DPF-based classification approaches are deployed on Apache Hadoop. Therefore, they are not capable of classifying incoming network traces in real-time. In this paper, we proposed a novel Apache Kafka Streams-based distributed classification approach named KS-DDoS. Firstly, we designed three distributed classification models on the Hadoop cluster using highly scalable machine learning algorithms by fetching training and validating instances from HDFS. Secondly, we deployed an efficient distributed classification model on the Kafka Stream cluster to classify incoming network traces into nine classes in real-time. Further, this classification approach stored highly discriminative features with predicted outcomes into HDFS. It helps in creating new or updating existing classification models using a new set of instances. We designed, deployed, and validated the proposed KS-DDoS classification approach on the DPF-based testbed. The results show that the proposed distributed KS-DDoS classification approach efficiently classified incoming network traces into nine classes: Benign (1), DDoS_DNS (2), DDoS_LDAP (3), DDoS_MSSQL (4), DDoS_NetBIOS (5), DDoS_UDP (6), DDoS_SYN (7), DDoS_NTP (8), and DDoS_SSDP (9). The average classification accuracy of the proposed distributed DRF-based classification approach is 80.07% while designing a model. After deployment, the average accuracy for Scenario-I → 83.98%, Scenario-II → 91.23%, Scenario-III → 65.33%, and Scenario-IV → 89.29%. The proposed distributed KS-DDoS classification system will fail to distinguish between DDoS attacks and Flash Events. Therefore, it will be interesting to provide a real-time distributed mechanism for classifying incoming traces into different types of DDoS attacks, legitimate flows, and Flash Events. Further, in the case of resource constraint devices/networks such as IoT devices/networks, it will be interesting to deploy this approach in the fog/edge computing environment.

Data availability Data available in a public (UNB-Canadian Institute for Cybersecurity, CICDDoS2019) repository that issues datasets with DOIs (https://www.unb.ca/cic/datasets/ddos-2019.html)

Declarations

Conflict of interest The authors declare that they have no conflict of interest

References

1. Internet users in the world geographic regisons 2020 q1 (2020) https://www.internetworldstats.com/stats.htm
2. Vxchange comprehensive guide to iot statistics you need to know in 2020 (2020) https://www.vxchange.com/blog/iot-statistics
3. Bhatia S, Behal S, Ahmed I (2018) Distributed denial of service attacks and defense mechanisms: current landscape and future directions. In: Versatile cybersecurity. Springer, pp. 55–97
4. Sachdeva M, Kumar K (2014) A traffic cluster entropy based approach to distinguish ddos attacks from flash event using deter testbed. ISRN Commun Netw 2014

5. 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
6. Ddos kaspersky q1 (2021) https://www.kaspersky.co.in/about/press-releases/2021_back-to-normal-despite-a-spike-in-january-ddos-attacks-in-q1-2021-return-to-pre-lockdown-numbers, May 2021
7. Ddos kaspersky q1 (2019) https://www.kaspersky.com/about/press-releases/2019a-ddos-storm-has-come-number-of-attacks-grows-after-long-period-of-decline, May 2021
8. Apache Kafka (2020) https://kafka.apache.org/
9. Confluent Kafka (2020) https://www.confluent.io/
10. Oussous A, Benjelloun F-Z, Lahcen AA, Belfkih S (2018) Big data technologies: a survey. J King Saud Univ-Comput Inf Sci 30(4):431–448
11. Aiello S, Click C, Roark H, Rehak L, Lanford J (2016) Machine learning with python and h20. H2O. ai Inc
12. Lashkari AH, Draper-Gil G, Mamun MSI, Ghorbani AA (2017) Characterization of tor traffic using time based features. In: ICISSp. pp. 253–262
13. Patil NV, Rama Krishna C, Kumar K (2021) Distributed frameworks for detecting distributed denial of service attacks: a comprehensive review, challenges and future directions. Concurr Comput: Pract Exp 33(10):e6197
14. Mirkovic J, Reiher P (2004) A taxonomy of ddos attack and ddos defense mechanisms. ACM SIGCOMM Comput Commun Rev 34(2):39–53
15. Zargar ST, Joshi J, Tipper D (2013) A survey of defense mechanisms against distributed denial of service (ddos) flooding attacks. IEEE commun Surv Tutor 15(4):2046–2069
16. Manavi MT (2018) Defense mechanisms against distributed denial of service attacks: A survey. Comput Electr Eng 72:26–38
17. Peng T, Leckie C, Ramamohanarao K (2007) Survey of network-based defense mechanisms countering the dos and ddos problems. ACM Comput Surv (CSUR) 39(1):3
18. Bhuyan MH, Bhattacharyya DK, Kalita JK (2014) Network anomaly detection: methods, systems and tools. IEEE commun Surv Tutor 16(1):303–336
19. Douligeris C, Mitrokotsa A (2004) Ddos attacks and defense mechanisms: classification and state-of-the-art. Comput Netw 44(5):643–666
20. Hoque N, Bhuyan MH, Baishya RC, Bhattacharyya DK, Kalita JK (2014) Network attacks: taxonomy, tools and systems. J Netw Comput Appl 40:307–324
21. Lee S (2004) Distributed denial of service: taxonomies of attacks, tools and countermeasures. In: Proceedings of the International Workshop on Security in Parallel and Distributed Systems. pp. 543–550
22. Mahjabin T, Xiao Y, Sun G, Jiang W (2017) A survey of distributed denial-of-service attack, prevention, and mitigation techniques. Int J Distrib Sens Netw 13(12):1550147717741463
23. Behal S, Kumar K (2017) Characterization and comparison of ddos attack tools and traffic generators: a review. Int J Netw Secur 19(3):383–393
24. Elejla OE, Anbar M, Belaton B (2017) Icmpv6-based dos and ddos attacks and defense mechanisms. IETE Tech Rev 34(4):390–407
25. Fenil E, Kumar PM (2019) Survey on ddos defense mechanisms. Concurr Comput: Pract Exper 32(4):e5114
26. Singh J, Behal S (2020) Detection and mitigation of ddos attacks in sdn: a comprehensive review, research challenges and future directions. Comput Sci Rev 37:100279
27. LeeY, Lee Y (2011) Detecting ddos attacks with hadoop. In: Proceedings of the ACM CoNEXT Student Workshop. ACM. p. 7
28. Khattak R, Bano S, Hussain S, Anwar Z (2011) Dofur: ddos forensics using mapreduce. In: Frontiers of information technology (FIT). IEEE 2011, pp. 117–120
29. Zhao T, Lo DCT, 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, 2015, pp. 1326–1331
30. Dayama R, Bhandare A, Ganji B, Narayankar V (2015) Secured network from distributed dos through hadoop. Int J Comput Appl 118(2)
31. 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/I? SPA, 2016 IEEE. IEEE. pp. 1976–1983

32. Hameed S, Ali U (2016) Efficacy of live ddos detection with hadoop. In: Network Operations and Management Symposium (NOMS), IEEE/IFIP. IEEE 2016, pp. 488–494
33. Hameed S, Ali U (2018) Hadec: hadoop-based live ddos detection framework. EURASIP J Inf Secur 2018(1):1–9
34. 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
35. Alsirhani S, Sampalli A, Bodorik P (2018) Ddos detection system: utilizing gradient boosting algorithm and apache spark. In: 2018 IEEE Canadian Conference on Electrical and Computer Engineering (CCECE). IEEE, pp. 1–6
36. Chhabra GS, Singh V, Singh M (2018) Hadoop-based analytic framework for cyber forensics. Int J Commun Syst 31(15):e3772
37. 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 on. IEEE, pp. 556–561
38. Chen L, Zhang Y, Zhao Q, Geng G, Yan Z (2018) Detection of dns ddos attacks with random forest algorithm on spark. Procedia Comput Sci 134:310–315
39. Patil NV, Krishna CR, Kumar K, Behal S (2019) E-had: a distributed and collaborative detection framework for early detection of ddos attacks. J King Saud Univ-Comput Inf Sci. p. in press
40. Sharma A, Agrawal C, Singh A, Kumar K (2019) Real-time ddos detection based on entropy using hadoop framework. In: Computing in engineering and technology, Springer, pp. 297–305
41. Patil NV, Krishna CR, Kumar K (2019) Apache hadoop based distributed denial of service detection framework. In: International Conference on Information, Communication and Computing Technology, Springer, pp. 25–35
42. Gumaste S, Narayan D, Shinde S, Amit K (2020) Detection of ddos attacks in openstack-based private cloud using apache spark. J Telecommun Inf Technol 4:62–71
43. Patil NV, Krishna CR, Kumar K (2020) S-ddos: apache spark based real-time ddos detection system. J Intell Fuzzy Syst, no. Preprint, pp. 1–9
44. Ahmed A, Hameed S, Rafi M, Mirza QKA (2020) An intelligent and time-efficient ddos identification framework for real-time enterprise networks sad-f: spark based anomaly detection framework. arXiv, pp. arXiv–2001
45. Hsieh C-J, Chan T-Y (2016) Detection ddos attacks based on neural-network using apache spark. In: Appl Syst Innov (ICASI), 2016 International Conference on. IEEE. pp. 1–4
46. Ahmad S, Yasin A, Shafi Q (2018) Ddos attacks analysis in bigdata (hadoop) environment. In: Applied Sciences and Technology (IBCAST), 2018 15th International Bhurban Conference on. IEEE, pp. 495–501
47. Vani YK, Ranjana P (2020) Detection of distributed denial of service attack using dlmn algorithm in hadoop. J Crit Rev 7(11):1011–1017
48. Bhardwaj A, Singh VK, Narayan Y (2015) Analyzing bigdata with hadoop cluster in hdinsight azure cloud. In: et al (2015) Annual IEEE India Conference (INDICON). IEEE 2015. pp. 1–5
49. Lucas Filho ER, de Almeida EC, Scherzinger S, Herodotou H (2021) Investigating automatic parameter tuning for sql-on-hadoop systems. Big Data Res 25:100204
50. Bauer D, Froese F, Garcés-Erice L, Giblin C, Labbi A, Nagy ZA, Pardon N, Rooney S, Urbanetz P, Vetsch P et al (2021) Building and operating a large-scale enterprise data analytics platform. Big Data Res 23:100181
51. Apache Zookeeper (2020) https://zookeeper.apache.org/
52. Sharafaldin I, Lashkari AH, Hakak S, Ghorbani AA (2019) Developing realistic distributed denial of service (ddos) attack dataset and taxonomy. In: 2019 International Carnahan Conference on Security Technology (ICCST). IEEE, pp. 1–8
53. Brent RP, Zimmermann P (2010) Modern Comput Arith 18:1–239
54. Han D, Bi K, Liu H, Jia J (2017) A ddos attack detection system based on spark framework. Comput Sci Inf Syst 14(3)
55. Joldzic O, Djuric Z, Vuletic P (2016) A transparent and scalable anomaly-based dos detection method. Comput Netw 104:27–42
56. Behal S, Kumar K, Sachdeva M (2018) D-fac: Aanovel ϕ-divergence based distributed ddos defense system. J King Saud Univ-Comput Inf Sci 33(3):1–12
57. Şimşek M, Şentürk A (2018) Fast and lightweight detection and filtering method for low-rate tcp targeted distributed denial of service (lddos) attacks. Int J Commun Syst 31(18):e3823

58. Lima Filho FSD, Silveira FA, de Medeiros Brito Junior A, Vargas-Solar G, Silveira LF (2019) Smart detection: an online approach for dos/ddos attack detection using machine learning. Secur Commun Netw 2019
59. Priyadarshini R, Barik RK (2019) A deep learning based intelligent framework to mitigate ddos attack in fog environment. J King Saud Univ-Comput Inf Sci:1–7
60. Aamir M, Zaidi SMA (2019) Clustering based semi-supervised machine learning for ddos attack classification. J King Saud Univ-Comput Inf Sci 33(4):436–446
61. Marvi M, Arfeen A, Uddin R (2020) A generalized machine learning-based model for the detection of ddos attacks. Int J Netw Manage 200:e2152