Journal of Scientific & Industrial Research
Vol. 83, October 2024, pp. 1075-1085
DOI: 10.56042/jsir.v83i10.4981

Deep Learning based Bursty Traffic Discrimination and Management using Sandpile Model

Bharathi N A[1]*, Ranjani Parthasarathi[1] & V Vetriselvi[2]

[1]Department of Information Science and Technology, [2]Department of Computer Science and Engineering, College of Engineering, Anna University, Guindy, Chennai 600 025, Tamil Nadu, India

Received 18 August 2023; revised 10 September 2024; accepted 18 September 2024

The significant advancements in internet technologies and applications have resulted in a substantial increase in network traffic volume, presenting considerable challenges for network management. The management of bursty traffic, in particular, poses difficulties, as it can originate from both legitimate and malicious sources. To ensure the continuity of normal network operations, it is critical to distinguish between genuine and attack traffic, preventing the blockage of legitimate traffic. This study proposes a framework for detecting and managing bursty traffic within Software-Defined Networking (SDN) environments. A deep learning-based approach is applied to differentiate between Distributed Denial of Service (DDoS) and flash traffic, utilizing the BiLSTM algorithm for its high classification accuracy. This approach uses the Markov Modulated Poisson Process (MMPP) to generate flash traffic, which is then integrated with the CIC-DDoS2019 dataset. For traffic management, a drop mechanism is applied to DDoS traffic, while the Bak-Tang-Wiesenfeld (BTW) Sandpile load balancing algorithm is utilized for managing flash traffic. The proposed Sandpile-based load balancing approach significantly reduces round-trip time by 93% and packet loss by 98.4%, while improving bandwidth availability by 94.5%. Thus the proposed approach combines deep learning for precise traffic classification with a dynamic, self-organizing load-balancing mechanism, offering an efficient and novel solution for managing bursty traffic in real-time network environments.

Keywords: Bursty traffic, Flash, Load balancing, Markov modulated poisson process, Sandpile

Introduction

SDN enables flexible network management, and many network environments such as enterprise networks and data centers deploy it for its dynamic and cost-effective nature. The capabilities of this architecture, such as programmability, centralized control, and network visibility, ease the work of network administrators. The primary focus in this work is to manage the overwhelming traffic in the networks using SDN architecture. One of the significant challenges that any enterprise has to handle is a surge in traffic. Particularly, there may be short periods with an overwhelming number of requests coming in, such as flash traffic, which have to be handled properly using appropriate load-balancing approaches.

To add to this challenge, networks are prone to different attacks, which become a significant concern in managing and providing continued network services. Specifically, Distributed Denial of Service attacks (DDoS) disrupt network services by creating maximum congestion and affecting network availability. These attacks mimic legitimate users and flash traffic. Hence, detecting and eliminating DDoS traffic is imperative for effectively managing flash traffic.

Hence, this paper presents an approach to discriminate DDoS from flash traffic using Machine Learning (ML) based approaches. Once the flash traffic is identified, the use of a sandpile model[1] is proposed to balance the traffic. The sandpile model presents the theory of Self-Organized Criticality (SOC) which has been used for dynamic load balancing of tasks.[2,3] Since bursty traffic must be balanced among switches in an SDN network, this research focuses on exploring the SOC property of the sandpile model for load-balancing traffic. This work provides a novel approach to load-balance a Self-Organized Criticality (SOC) inspired load-balancing mechanism with machine learning-based classification for managing bursty traffic and differentiating between flash traffic and DDoS attacks in SDN architecture. The following are the main contributions and novel characteristics of this study:

*Author for Correspondence
Email: baru.cheers@gmail.com

- Comprehensive Framework for Bursty Traffic Management in SDN: A framework is built that detects and handles incoming bursty traffic in an SDN context. This integrated solution ensures effective network performance even during high traffic volume by addressing both the problems of load balancing and traffic classification.
- ML-Based Traffic Classification: The proposed framework classifies incoming network traffic into three categories: flash, DDoS, or normal traffic, using advanced machine learning models. This provides enhanced detection of attacks that closely resemble flash traffic with improved classification accuracy.
- Sandpile Model for Dynamic Load Balancing: The Bak-Tang-Wiesenfeld (BTW) Sandpile model for traffic load balancing in SDN systems is being applied as a novel approach. A dynamic and self-organizing mechanism is used to distribute flash traffic across network switches by utilizing the SOC property of the sandpile model. By enabling the network to automatically balance load during traffic spikes, this method enhances resilience and reduces congestion.
- Novel Flash Traffic Dataset Creation using MMPP Model**:** It may be noted that while sufficient datasets are available for DDoS traffic, not many datasets that characterize flash traffic are available. Hence, a probability model, the MMPP model which characterizes internet bursty traffic[4], to generate flash traffic is used and combined with the CIC-DDoS2019 dataset.[5] This combined dataset is used to learn the ML models for classification.
- Evaluation of the framework based on two aspects: The framework is evaluated with a dataset that combines synthetic flash traffic produced by the MMPP model with DDoS traffic from CIC-DDoS2019. The effectiveness of the load-balancing techniques as well as the ML classification algorithms is evaluated.

Related Work

This section reviews the existing literature on bursty traffic focusing on flash and DDoS traffic under three topics: (i) Classification of bursty traffic; (ii) Bursty traffic dataset models, and (iii) Bursty traffic management in SDN.

Bursty Traffic Classification

The issue of distinguishing DDoS and flash traffic has been the focus of numerous researches. The features such as, the amount of incoming traffic, inter-arrival time, the number of requests made per IP address, and other factors are applied to characterize and distinguish the traffic. Both statistical and ML-based approaches have been proposed as discussed below.

Gupta *et al.*[6] investigate the metrics and methods used to define flash and DDoS. David and Thomas[7] devised a thresholding algorithm to discriminate between DDoS and flash traffic. In their work, tsallis entropy is used for distinguishing attacks from flash crowds, for the network attributes such as packet size and destination IP, and the threshold is set dynamically to detect the traffic. Patil *et al*.[8] proposed a model to discriminate DDoS and flash based on the behavior and the techniques used by the attacker in the SDN environment. The suggested model particularly targets traffic created by spoofing source IP/MAC addresses to detect and mitigate Multi-Destination (MD) DDoS attacks against SDN controllers. This approach is created for the attacks targeting the SDN environments rather than generalized.

Silva *et al*.[9] utilize Poisson, BINEG, ZINB, and ZIP models to differentiate between regular traffic, DDoS attacks, flash crowd events, and combined flash crowd-DDoS traffic, addressing the challenge of distinguishing these similar traffic patterns. The models are evaluated in a particular situation, so more validation in different settings and with different variables is required to ensure wider applicability. Shalini *et al*.[10] use the Cumulative Sums (CUMSUM) algorithm to distinguish between flash and DDoS traffic and employ a drop policy to lessen the attack volume. Finding the precise threshold value to distinguish between DDoS and flash is still a challenge, although many studies use statistical characteristics to do so. In response to this, researchers have selected ML-based detection techniques.

Gong *et al*.[11] propose a hybrid DDoS detection framework named the Intelligent Trust Model (ITM). This framework uses an Extreme Learning Machine (ELM) algorithm to classify the attacks. Agha and Rehman.[12] use the random forest algorithm to improve the accuracy of classifying DDoS and flash flows. Kumar *et al*.[13] employed deep learning techniques to identify DDoS using the CIC-DDoS2019 dataset and stacked LSTM. Gebremeskel *et al.*[14] propose a Long Short-Term Memory (LSTM) model to classify and detect DDoS attacks with a high

accuracy of up to 99.42%. However, the approach does not address the identification of flash crowd traffic, which could make it difficult to distinguish DDoS attacks from legitimate high-traffic events.

Bursty Traffic Dataset Models

The insight of different authors in exploring the bursty nature of traffic is discussed here. Anteneodo *et al.*[15] investigate a variety of bursty and non-homogeneous Poisson process models to capture the overall rate at which individuals send an email.[15] Similarly, Scott and Smyth analyze models for the telephone call and web navigation data.[4]

Navaroli *et al.*[16] use latent piecewise-constant Poisson processes for modeling temporal variations in an individual's communication rate. Abu-Shareha *et al*. introduce an innovative internet traffic modeling approach using Multiple Markov Modulated Bernoulli Processes (MMBPs) to capture the burstiness and characteristics of packet arrivals, enhancing the effectiveness of Active Queue Management (AQM) techniques.[17] Heyman and Lucantoni examine the suitability of the Markov Modulated Poisson Process (MMPP) model for fitting internet traffic.[18] Shah-Heydari and Le-Ngoc model multimedia traffic using the MMPP model.[19] The authors create a two-state model to represent the bursty nature of multimedia internet traffic. Rajabi and Wong assess MMPP's performance as a traffic model for resource provisioning in web applications.[20] By varying the burstiness level, they create a synthetic trace of web application traffic and then fit the data using the MMPP model.

Muscariello *et al.,* have proposed a simplified MMPP internet traffic model that can reasonably approximate the traffic patterns.[21] Utilizing traffic parameters such as the average flow arrival rate, average number of packets per flow, and average packet arrival rate, they create synthetic traffic and compare it to real traffic traces. Abu-Shareha *et al*.[17] introduce an innovative internet traffic modeling approach using Multiple Markov Modulated Bernoulli Processes (MMBPs) to capture the burstiness and characteristics of packet arrivals, enhancing the effectiveness of Active Queue Management (AQM) techniques.

From the study, it is noted that although earlier models, such as the latent piecewise-constant Poisson processes, MMBP and MMPP, and bursty and non-homogeneous Poisson process models, have been used for a variety of traffic modeling applications, such as email rates, multimedia traffic, and web applications, they did not work on capturing the burstiness and real-time dynamics of internet traffic. Notably, MMPP is suitable for time-varying intensities but was not optimally used for creating flash traffic datasets. This study improves the relevance and accuracy of traffic modeling in the present network conditions by efficiently modeling bursty traffic and creating new flash traffic datasets.

Bursty Traffic Management

Faraj *et al*. investigate and implement the SDN-based Least Packet Load Balancing method.[22] This method dynamically selects the server based on the number of packets forwarded to that server. The number of packets at each port is analyzed, and load balancing is done based on the least loaded server. Jadhav *et al*.[23] implemented load balancing in SDN networks using round-robin and random approaches. Round robin proved to be more effective, according to the authors, as it reduces packet loss and increases network throughput.[23] Yan *et al.* introduced a two-stage dual-threshold random early detection (TD-RED) control method for heterogeneous cognitive radio networks (Het-CRNs) that successfully handles network congestion caused by burst service flows.[24]

The study highlights various strategies researchers have employed to address network congestion due to bursty traffic. Methods like round-robin and TD-RED often lack the adaptability needed to handle sudden and significant shifts in traffic patterns, potentially resulting in suboptimal performance during traffic spikes or abrupt changes. In contrast, the sandpile model offers a more flexible solution for managing network congestion caused by bursty traffic. This model's adaptive nature makes it well-suited for effectively detecting and managing traffic fluctuations. The following section outlines a framework that integrates these capabilities for improved handling of bursty traffic.

Proposed Framework for Bursty Traffic

The proposed framework, shown in Fig. 1, is based on the SDN architecture, with a clear separation of the data plane, and the control plane. The proposed traffic management framework is built on the top of the control plane.

It consists of three main components, namely traffic monitoring, an ML-based flow classifier, and a traffic management application in the SDN controller. The incoming flows from the data-plane devices are

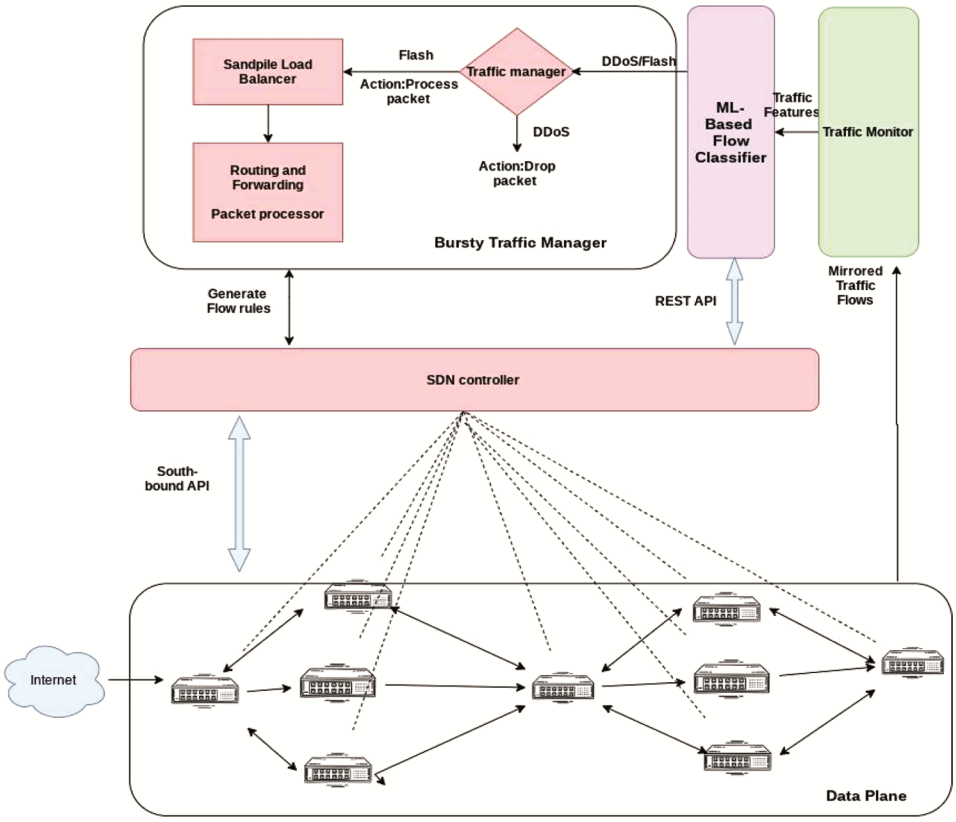

Fig. 1 — Proposed framework for traffic management

periodically mirrored and sent to the traffic classifier. The traffic flow classifier classifies the flows as one of the following classes: normal, DDoS, and flash. If DDoS or flash flows are detected, the classifier notifies the traffic management application in the SDN controller. The traffic management application decides the strategy to handle DDoS and flash. The functionalities of each component in the framework are discussed below.

Traffic Monitoring

The monitoring node captures all the packets that enter the network and sends them to the flow classifier. The traffic from ingress and egress nodes is mirrored and sent to this module.

ML-based Flow Classifier

An ML model is constructed to classify traffic as DDoS, flash, or normal traffic in the network. The training dataset for the model consists of combined instances of DDoS and flash. The CIC-DDoS2019 dataset is used for DDoS, while flash traffic is generated using a statistical model to create the flash dataset. The data is pre-processed, and a subset of features is selected based on decision trees. These features are fed to different ML algorithms, and the model providing the best accuracy is chosen as the final model.

Enhancing the Dataset

The DDoS dataset is chosen from the CIC, which provides various benchmark datasets for network security. The CIC-DDoS2019 dataset contains 12 DDoS attacks: NTP, DNS, LDAP, MSSQL, NetBIOS, SNMP, SS64DP, UDP, UDP-Lag, WebDDoS, SYN, and TFTP and benign (normal) network traffic. Bursty flash traffic was generated using the MMPP model and add flash traffic to the CIC-DDoS2019 dataset. This section gives the details of the traffic model and flash traffic generation.

(A) nMMPP Traffic Modeling

An MMPP is a doubly stochastic process where arrivals are modeled using the Poisson process, and the states are maintained using the Markov sequence. This model is governed by a Markov chain and is memory less, where the current state is not dependent on the previous state. Each state is associated with a particular arrival rate. As the arrival rate changes the Markov chain changes correspondingly. A Markov chain with a state space S is considered to be MMPP

if and only if arrivals follow a Poisson process as in Eq. (1)

$$P\{F(t) = n\} = \frac{e^{-\lambda t}\lambda t^{n}}{n!} \quad \ldots (1)$$

The probability of 'n' arrivals in unit time t with an average arrival rate of λ per unit time is given in Eq. (2)

$$\sum_{n=0}^{\infty} P_x(n) = \sum_{n=0}^{\infty} \frac{e^{-\lambda t}\lambda^{n}}{n!} \quad \ldots (2)$$

The inter-arrival times x_1, x_2.....x_n are exponentially distributed using Eq. (3).

$$P_x(t) = \lambda e^{\lambda t} \quad \ldots (3)$$

The arrival times t_1, t_2.....t_n are calculated by summing up the inter-arrival times ($t_n = x_1 + x_2.....x_n$). A three-class model is created with traffic arrival rates for transition. MMPP is parameterized by a 3-state Markov chain with arrival rate matrix (R) and transition probability matrix of the Markov chain (T). Transition Probability Matrix T is a matrix of one-step transition probabilities P_{ij} and is defined as in Eq. (4)

$$P_{ij} = P(S_n + 1 = j || S_n = i) \forall n \quad \ldots (4)$$

The state space for the Markov process, S = (Idle, Normal, Bursty), is defined for the arrival of traffic flow. As shown in Fig. 2, the Markov chain regulates the state transitions, and traffic is modulated according to a particular state sequence using the arrival rate. The idle state, S_0 is the phase of network traffic where the arrival rate is zero. In a normal state, S_N follows a constant rate of arrival. The bursty state, S_B is a sudden increase in the arrival rate with less inter-arrival time. λ_0, λ_N, λ_B are the arrival rates of initial, normal, and bursty traffic correspondingly. The following subsection used the parameters to generate flash traffic in the SDN environment.

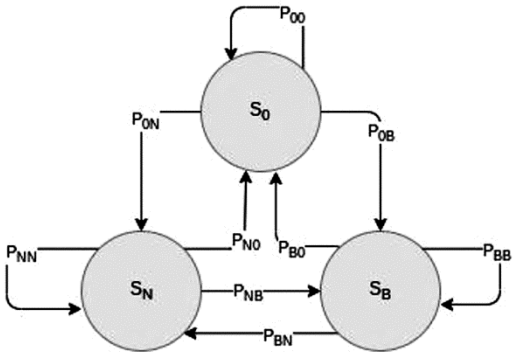

Fig. 2 — Markov chain of states

(B) Synthetic Flash Traffic Generation

The steps to create traffic using the model are defined in Algorithm 1. The arrays for states S, transition matrix T, and rate matrix R are initialized. The probability transition matrix is generated using a random function where the sum of the rows in the matrix is always 1. Initially, the random state is chosen from a state array. Then for 'n' iterations, the states are selected using a probability matrix. Packets are forwarded according to the state's rate using the transition matrix.

The flash traffic is generated by simultaneously executing the MMPP model in many clients using HTTP, HTTPS, FTP, and email protocols similar to realistic background traffic. The traffic was generated using multiple clients, as shown in Fig. 3. The CICFlowMeter[25] tool captures the simulated data traffic. The generated flash traffic data is stored and labeled. Our dataset comprises some significant features that differentiate flash from DDoS traffic. These features include Src & Dst IP, flow packets, flow bytes, forward packets, backward packets, flow packet rate, flow Inter Arrival Time (IAT) max, backward (bwd) IAT total, the total length of the forward (fwd) packets, flow bytes, flow IAT mean, Acknowledgement (ACK) flag count, Down/Up ratio, Total fwd packets, Total bwd packets, fwd bytes average, bwd bytes average, etc. help to identify and discriminate traffic characteristics.

Algorithm 1: Traffic Generation using MMPP

1. **Procedure** MMPP(*T, R*)
2. Initialize the number of states, N = 3
3. Define state space array =[Idle,Normal,Bursty]
4. Define the rate array R=[λ_0,λ_N,λ_B]
5. State probability transition matrix array
6. T = [[P_{00}, P_{0N}, P_{0B}],[P_{N0}, P_{NN}, P_{NB}][P_{B0}, P_{BN}, P_{BB}]]
7. Generate probability transition matrix T randomly
8. Verify that the sum of the rows in the matrix T is always 1

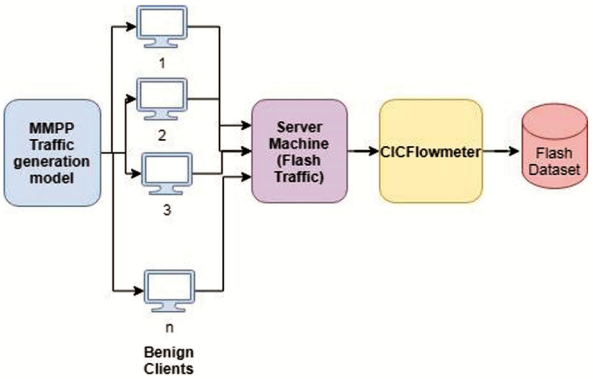

Fig. 3 — Flash traffic generation

9. Initialize the state as Idle
10. **for** 0 to n **do**
11. Choose a state randomly from P and fix the transition array element
12. **for** all the transition states in the matrix **do**
13. **if** state ="Idle" **then**
14. Packet generated with $\lambda_0 r$ate
15. **end if**
16. **if** state ="Normal" **then**
17. Packet generated with the random packet rate range λ_N
18. **end if**
19. **if** state="Bursty" **then**
20. Packet generated with λ_B packet rate range
21. **end if**
22. **end for**
23. **end for**
24. **end procedure**

The total fwd Packet and Total bwd Packet parameters in the dataset represent a two-way communication response: the total forward and backward packets. IP address parameter helps to find changes in the source IP address rate and the number of requests sent from one particular IP address. The flow packet rate parameter gives the number of packets transferred per second which indicates the rate of incoming traffic. The flow IAT mean, flow IAT max, and bwd IAT total provide the time taken between the flows. ACK flag count parameter gives the count of the number of packets acknowledged. Overall there are 84 features in the flash dataset similar to the CIC-DDoS2019 dataset. The combined dataset is used for training and validating the model.

Data Pre-processing

The data is pre-processed by removing irrelevant data, categorizing and normalizing data, label conversion, and handling missing values. Features in the data set like Unnamed, Flow ID, and Inbound are irrelevant for classification. Then the columns with missing values such as 'Not a Number' and infinity values are corrected. The categorical labels are transformed and encoded. Encoding is done for benign traffic as 0, all DDoS attack instances as 1, and flash traffic instances as 2.

Selection of Classification Model

The training dataset with the selected features is given as the input to the conventional ML-based classification algorithms: Naive Bayes (NB), Decision Tree (DT), Random Forest (RF), Extreme Gradient Boosting (XGBoost), Support Vector Machines (SVMs) and deep learning methodologies: Long Short-Term Memory (LSTM), Bidirectional LSTM (BiLSTM), Gated-Recurrent Unit (GRU). Initially, the performance parameters are examined for each model. The selection of the classification algorithm is based on the accuracy of prediction.

Bursty Traffic Manager

Traffic manager make decisions based on DDoS or flash from the ML model. It decides whether or not to forward packets based on the type of traffic. If DDoS flows are identified then the packets are dropped. In the case of normal traffic, the packets are forwarded. In the case of flash traffic, to maintain the stability of the network, a sandpile-based load balancer was used. Sandpile models have been used in different applications. Laredo *et al.* use the Sandpile model to schedule tasks in a network. Gasior and Seredynski use it for scheduling in highly parallel and distributed environments in cloud systems.[26] Chilingirian used the Abelian Sandpile Model (ASM) to solve the latency minimization problem.[1]

The sandpile load balancer is based on the Bak-Tang-Wiesenfeld (BTW) Sandpile model, which was proposed by Per Bak, Chao Tang, and Kurt Wiesenfeld.[27] It was widely used in the areas of mathematics and physics. This model follows Self Organized Criticality (SOC), which is a cellular automata model. SOC is a property of complex systems with critical, unstable states where the events follow the power-law relationship. The BTW sandpile model is explained below.

BTW Sandpile Model

The BTW Sandpile Model is initiated as an example of a self-organized criticality state exposed by a system. A real sandpile is used on a finite base that has attained stability with the slow addition of grains and lost grains at the boundary. The sandpile model developed primitively defines a one-dimensional cellular automaton for simulating a two-dimensional heap of sand. The model defines a two-dimensional (2d) square lattice as follows. At any site (x, y), a variable for height $h(x, y)$is defined as the number of grains at that site. At first, adding a grain at a site (x, y) simply modifies the site's height *h(x, y)* as shown in Eq. (5)

$$h(x, y) \geq h = 2d \quad \ldots (5)$$

If a node's height is greater than a certain threshold value ($h(x, y) \geq th$), an avalanche happens. However, according to the model, the site collapses and loses excess grains (e) which are distributed randomly onto the neighboring sites as given in Eq. (6).

$$h(x,y) = h(x,y) + 1 \quad \ldots (6)$$

The state of the neighbors changes as a result of a reassignment as shown in Eqs (7 & 8). The toppling continues for each neighboring grid until the number of grains in all grids is less than the threshold value, and the entire system reaches a stable state.

$$h(x,y) = h(x,y) - Th \quad \ldots (7)$$

$$h(x \pm 1, y) = h(x \pm 1, y) + 1\, h(x, y \pm 1) = h(x, y \pm 1) + 1 \quad \ldots (8)$$

This paper has proposed load balancing using the sandpile model and reduction of the network unavailability probability due to flooding of flash traffic. According to the interpretation of the sandpile load balancer, incoming flows within a specified topology are managed by a centralized SDN controller. The network flow arrival is related to sand grains arrival. In the framework, switches receive network flows where they pile up like sand grains. An avalanche is considered a situation in which the network switch is about to exhaust its bandwidth due to an unexpected rise in traffic flows. When a controller detects that a pile in a switch is larger than its neighboring switches, it collapses the pile, triggering an avalanche that could spread the traffic across the whole network until equilibrium is reached.

Load Balancing based on the Sandpile Algorithm

Load balancing algorithms use the routing and forwarding information given by the controller in the SDN network. The critical factors considered in designing the load balancing algorithm are: evaluating the necessary conditions to transfer the load, calculating the neighbouring paths based on available bandwidth, and transferring the load to eligible paths. Algorithm 2 depicts sandpile-based load balancing the paths for flash flows in the network. Let C_i be the controller which calculates a set of paths (ρ_1 to ρ_p) for the incoming flows η_1 to η_l) and program rules for switches (σ_1 to σ_m) in the path.

It is considered that every switch in a path is capable of supporting a certain load (threshold bandwidth, th).). When the load of a flow ($\eta.bw$) in a switch exceeds the threshold limit (th), then an avalanche has occurred. So, the load (flow) needs to be transferred to the switches in the neighbouring paths. The paths for the source and destination of the flows are calculated and sorted according to the available bandwidth. Flows that exceed the threshold bandwidth in a switch are selected. For each neighbour switch (σj), the load is balanced. If there exists an imbalance due to the transfer of flows from the previous switch then the process is repeated again and again until a stable state is obtained.

Algorithm 2: Sandpile-Based Load Balancing

Procedure LOAD BALANCE(paths)

Paths = { $\rho_1, \rho_2, \rho_3, \ldots \rho_p$}
Switches = {$\sigma_1, \sigma_2, \sigma_3, \ldots\ldots \sigma_l$}
Flows= { $\eta_1, \eta_2, \eta_3, \ldots\ldots \eta_m$}
$\sigma i = current\ switch$
$\sigma j = target\ switch$
$th = threshold\ bandwidth(bw)\ of\ switch$
$O = excess\ bw\ in\ a\ flow\ than\ the\ threshold\ bw$
$T = highest\ bw\ of\ a\ flow\ in\ the\ list$
$for\ each\ \rho i\ in\ Paths\ do$
$Compute\ the\ cost\ of\ the\ path$
$Find\ the\ neighbor\ candidate\ paths$
$for\ all\ Switches\ in\ a\ path\ do$
$Sort\ the\ switches\ according\ to\ the\ bw\ used$
$for\ all\ Flows\ in\ \sigma i\ do$
$O = nil$
$Sort\ all\ the\ flows\ in\ \sigma i\ according\ to\ the\ bw$
$if\ \eta.bw[\sigma i] < th\ then$
a. $if\ th - \eta.bw[\sigma i] \geq O\ then$
b. $T = O$
c. $endif$
$endif$
$if\ \eta.bw[\sigma i] > th\ then$
$O = \eta.bw[\sigma i] - th$
$Fetch\ the\ flow\ with\ more\ bw\ from\ the\ sorted\ list$
$O = O - th$
$end\ if$
$end\ for$
$for\ all\ \eta.bw[\sigma j]\ except\ \sigma j = \sigma i\ do$
$O = \eta.bw[\sigma j] - th$
$if\ O = 0\ then$
$T = th - \eta.bw[\sigma j]$
$if\ O < T\ then$
$\sigma j\ has\ space\ to\ transfer$
$else$
a. $Choose\ the\ next\ switch\ in\ the\ sorted\ list$
$end\ if$
$end\ if$
$end\ for$
$\eta.bw[\sigma i] = \eta.bw[\sigma i] - T$
$\eta.bw[\sigma j] = \eta.bw[\sigma j] - T$
$Begin\ to\ topple\ from\ \sigma i\ to\ \sigma j$
$end\ for$
$if\ all\ the\ switches\ in\ the\ network\ are\ within\ the\ threshold\ th\ then$
$Network\ has\ attained\ Equilibrium\ state$
$Stop\ the\ transfer\ and\ return\ to\ the\ Traffic\ manager()$
$end\ if$
$end\ for$
$End\ procedure$

Simulation and Performance Evaluation

A comprehensive overview of the simulation-based assessment of the suggested framework is provided in this section. The ML model findings are evaluated before discussing traffic detection and management strategies.

SDN Simulation Environment

The proposed framework is evaluated with different traffic rates for flash and DDoS using the GNS3 emulator.[28] The network is emulated with an ONOS controller[29], management switches, OpenFlow switches, botnets, and benign users on a single machine with different topologies. The botnets are used to generate DDoS attacks. Benign traffic like TCP, UDP, DNS, and SYNC is generated using the Iperf3 tool[30], and flash traffic is simulated using multiple clients simultaneously. The traffic from each scenario is monitored and sent to the flow classifier. A flow classifier is an external application that interacts with the controller using REST API. If DDoS or flash flows are detected, it notifies the traffic management application in the controller. Different traffic scenarios are created and tested to analyze the behaviour of the network and to calculate the system's performance. The experimental set-up parameters are given in Table 1.

Performance of ML Classification Algorithms

The purpose of executing different classification algorithms is to ensure that an efficient model suitable for the task is selected. This combined dataset used for training consists of 50006249 DDoS attacks, 859654 flash traffic, and 458771 normal traffic instances. The model is trained with different classification algorithms and uses 10-fold cross-validation. The metrics chosen for evaluating the classification algorithms are accuracy, precision, recall, and F1 score.

The average values of performance measures for all classes using various models are displayed in Table 2. The analysis shows that Naive Bayes has an accuracy rate of only 75% whereas the accuracy rates of all the other algorithms used are above 94%. Additionally, BiLSTM has a precision rate of 99% and a recall rate greater than 98%. According to the findings, the BiLSTM algorithm is the most accurate classifier for the provided features. The BiLSTM is used as a final model to classify flows.

Table 1 — Experimental set-up parameters

Parameters	Description
Operating system	Linux
RAM	16Gb
Network emulator	GNS3
Controller	ONOS
Switch type	OpenVSwitch
Number of switches	14
Number of hosts	24
Normal traffic generation tool	Iperf3
Attack traffic generation	Botnet

Performance of BiLSTM Classifier in SDN

In the SDN context, the performance of the BiLSTM model is assessed using the metrics: detection rate, false alarm rate, and classification rate. The classification rate depicts the ratio of the number of correctly detected DdoS, flash, or normal flows to the total number of generated traffic flows during the simulation. The percentage of real traffic classes that are incorrectly labeled as other traffic classes is known as the false alarm rate. The detection rate depicts the model's accuracy in detecting the relevant traffic flows. These metrics are chosen for evaluation because they indicate the algorithm's efficiency classification and misclassification of the actual traffic in the network.

According to Table 3, normal traffic is detected at a rate of above 94% for all the traffic classes. It is observed that the false alarm rate of all three traffic classes is less than or equal to 5%. The results demonstrate that the model performs with a classification rate of more than 93% for all the traffic classes in the simulation environment.

Performance of Traffic Management Strategies

The effectiveness of the proposed technique is assessed by comparing the Sandpile-based load-

Table 2 — Performance of ML Models

Model	Accuracy	Precision	Recall	F1-score
Naive Bayes	75	76	74	75
Decision tree	93	92	94	93
SVM	89	87	90	88
Random Forest	95	97	95	93
XG boosting	93	96	95	93
LSTM	97	96	95	93
BiLSTM	98	99	98	98
GRU	95	96	96	96

Table 3 — Performance of BiLSTM in SDN

	Detection rate	False alarm rate	Classification rate
Normal	98%	5%	95%
Flash	94%	3.8%	93%
DDoS	95.5%	5%	94.5%

balancing algorithm with four other widely used existing load-balancing methods

Round-Robin: This algorithm continuously alternates the paths in order, starting with the first and ending with the last.

Weighted Round-Robin: This approach is a variant of round-robin where weights are given to each path.

Least utilization: This approach distributes the flows to the paths that have the least link load.

Random: This approach randomly selects a path without imposing any restrictions. The network controller randomly selects paths without taking the status of the path into account.

To evaluate the proposed algorithm packet loss rate, bandwidth utilization, and Round Trip Time (RTT) as performance metrics were used.

RTT: RTT is the time taken for the packets to reach from source to destination and back to the source. The RTT taken before and after management in DDoS and flash scenarios were compared. The RTT of the DDoS flows could be seen in Fig. 4. In the flash scenario, the performance of the sandpile is compared with four other load-balancing algorithms. From Fig. 4, it can observed that RTT is greater (average of 3ms) initially because the network is overloaded with DDoS traffic. When the approach stops dropping DDoS packets and just forwards normal traffic through the network, the RTT reduces to 0.3 ms. In the absence of traffic management, the RTT is 3 ms in the case of flash traffic. It decreases when using the load balancing strategies, as demonstrated in Fig. 5. For the RR algorithm, the RTT is up to 2 ms, however when utilizing the Sandpile approach, it is only 0.2 ms. The round robin and WRR have limitations because the traffic flow is dispersed sequentially regardless of the path load state. On the other hand, the Sandpile algorithm equally distributes traffic load until the whole network finds equilibrium. An improvement in RTT is realized while using the sandpile approach since it minimizes congestion.

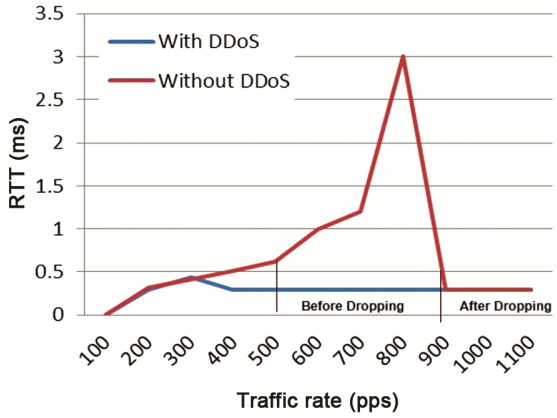

Fig. 4 — RTT measure in case of DDoS

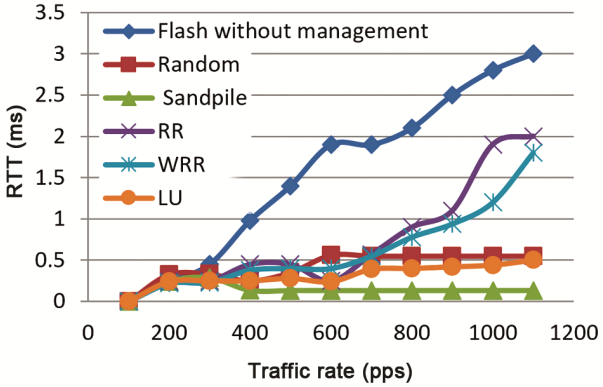

Fig. 5 — RTT measure in case of flash

***Packet Loss Rate*:** The packet loss rate is the ratio of the number of packets lost to the total number of packets sent. This metric is preferred for the evaluation of flash traffic because it indicates the reliability of network communication. To provide proper quality of network connectivity, packet loss should be minimal. The results of packet loss with and without load balancing techniques are shown in Fig. 6. The percentage of packet loss is lowest when the rate of traffic flow is moderate, but when flash traffic is prevalent, packet loss reaches to 50% or higher in the absence of traffic management.

The packet loss is an average of 35% and 40% for RR and WRR respectively, because the number of packets transmitted in a round is decreased in response to bursty traffic. While using the Sandpile approach, packet loss drops to 0.8%. Packet loss is significantly reduced by the proposed approach through continuous balancing of traffic to the adjacent paths with available bandwidth without causing any network congestion.

Network Utilization and Bandwidth Availability: Network utilization is the ratio of the actual bandwidth received to the maximum bandwidth that a

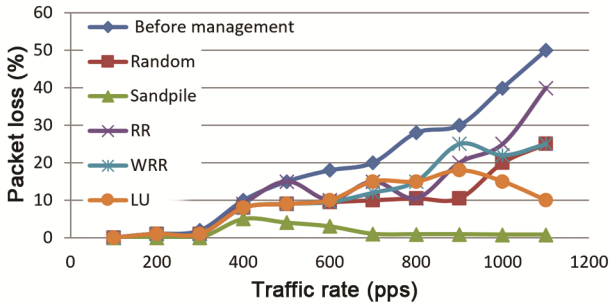

Fig. 6 — Packet loss for flash traffic scenario

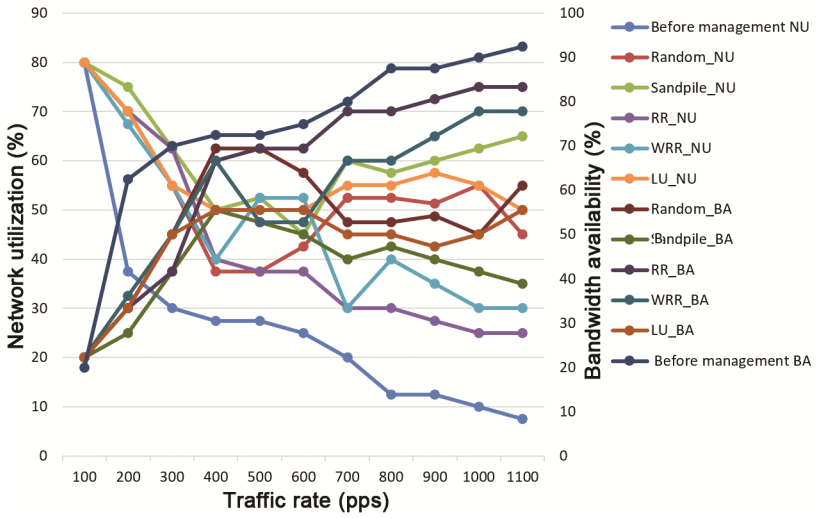

Fig. 7 — Network Utilization (NU) and Bandwidth Availability (BA) in the case of Flash

port can handle. The average bandwidth consumption and availability for each load-balancing algorithm are measured to evaluate our work. In Fig. 7 the impact of load-balancing techniques on network utilization and bandwidth availability both before and after implementation is shown.

In the case of flash traffic, the network utilization before load balancing is 92%. It gradually falls when the load balancing methods are used. It is noted that WRR (avg. 75%) exhibits the highest network utilization. The utilization rate drops to 35% while using the suggested strategy. Additionally, Fig. 7 shows the bandwidth availability, for which the sandpile method has a maximum availability of 65%.

The evaluations discussed in this section show that our proposed framework detects bursty traffic and manages traffic efficiently. With the dataset, including the DDoS and flash traffic flow examples, our traffic classification approach provides good accuracy. DDoS traffic aims to bring down the network. Therefore, this study drop packets rather than load balancing which improves the network performance. Flash traffic, however, must be forwarded and cannot be dropped, so traffic load balancing improves network availability. The simulation results show that the sandpile technique significantly lowers RTT from 3 ms to 0.2 ms and packet loss from 50% to 0.8%. In addition, it increases bandwidth availability from 35% to 65% by properly utilizing the SDN controllers.

Conclusions

This research proposes a framework for discriminating and managing bursty traffic within an SDN environment, demonstrating effective traffic classification and management. The Bi-LSTM model achieves over 99% accuracy in identifying traffic types, outperforming other methods such as Naive Bayes and Random Forest. A notable enhancement to the DDoS dataset includes integrating flash traffic via an MMPP model. Simulation results demonstrate that round-trip time is reduced to 0.2 ms, packet loss is minimized to 0.8%, and bandwidth availability increases to 65%. However, one should address the complexity of the model which may increase when applying the suggested work to large networks to avoid scaling issues. Future work could explore enhancements to the sandpile model to improve its efficiency and accuracy across diverse network conditions. Enhancements could include expanding the framework to address low-rate DDoS attacks distinguishing them from normal traffic and integrating more advanced machine learning techniques to refine detection and classification capabilities.

References

1 Chilingirian B, *The Sandpile Load Balancer and Latency Minimization Problem*, (2016).

2 Laredo J J, Jiménez J, Guinand F, Olivier D & Bouvry P, Load balancing at the edge of chaos: How self-organized criticality can lead to energy-efficient computing, *IEEE Trans Parallel Distrib Syst*, **28(2)** (2016) 517–529, https://doi.org/10.1109/tpds.2016.2582160.

3 Laredo J J, Dorronsoro B, Pecero J, Bouvry P, Durillo J J & Fernandes C, Designing a self-organized approach for scheduling bag-of-tasks, *7th Int Conf P2P Parallel Grid Cloud Internet Comput* (IEEE) 2012, 315–320, https://doi.org/10.1109/3pgcic.2012.28.

4 Scott S L & Smyth P, The Markov modulated Poisson process and Markov Poisson cascade with applications to web traffic data, Bayesian statistics **7,** *Oxford University Press* 2003, 671–680, https://doi.org/10.1093/oso/9780198526155.003.0047.

5 Sharafaldin I, Lashkari A H, Hakak S & Ghorbani A A, Developing realistic distributed denial of service (DDoS) attack dataset and taxonomy, *Int Carnahan Conf Security Tech* (IEEE) 2019, 1–8, https://doi.org/10.1109/ccst.2019.8888419

6 Gupta S, Grover D & Singla J, Characterization of flash events and DDoS attacks: a survey, *7th IEEE Int Conf Adv Comput Commun Syst*, **14** (2021) 1442–1447, https://doi.org/10.1109/icaccs51430.2021.9441793.

7 David J & Thomas C, Discriminating flash crowds from DDoS attacks using efficient thresholding algorithms, *J Parallel Distrib Comput*, **152** (2021) 79–87, https://doi.org/10.1016/j.jpdc.2021.02.019.

8 Patil J, Vrinda T, Rajan A & Rawat A, Discriminate, locate and mitigate DDoS traffic in presence of Flash Crowd in Software Defined Network, *J Supercomput*, **78(15)** (2022) 16770–16793, https://doi.org/10.1007/s11227-022-04538-9.

9 Alves da Silva A A, Silva L S, Bezerra E L, Guelfi A E, de Armas C, Teixeira de Azevedo M & Kofuji S T, A proposal to distinguish DDoS traffic in flash crowd environments, *Int J Inf Secur Privacy*, **1** (2021) 1–16, https://doi.org/10.4018/ijisp.2022010104.

10 Shalini P V, Radha V, Sriram G & Sanjeevi, DOCUS-DDoS detection in SDN using modified CUSUM with flash traffic discrimination and mitigation, *Comput Netw*, **217** (2022) 109–361, https://doi.org/10.1016/j.comnet.2022.109361.

11 Gong C, Yu D, Zhao L, Li X & Li Xi, An intelligent trust model for hybrid DDoS detection in software-defined networks, *Concurr Comput Pract Exp*, **32(16)** (2020) e5264, https://doi.org/10.1002/cpe.5264

12 Agha S & Rehman O, Improving discrimination accuracy rate of DDoS attacks and flash events, *Int Conf Cyber Warfare Security* (IEEE), **6** (2020) 1–6, https://doi.org/10.1109/iccws48432.2020.9292377.

13 Kumar K & Behal S, Distributed denial of service attack detection using deep learning approaches, *8th Int Conf Comput Sustainable Dev* (IEEE) 2021, 491–495.

14 Gebremeskel T G, Gemeda K A, Krishna T G & Ramulu P J, DDoS attack detection and classification using hybrid model for multi-controller SDN, *Wirel Commun Mob Com*, **1** (2023) 1–18, https://doi.org/10.1155/2023/9965945.

15 Anteneodo C, Malmgren R D & Chialvo D R, Poissonian bursts in e-mail correspondence, *Eur Phys J B*, **75(3)** (2010) 389–394, https://doi.org/10.1140/epjb/e2010-00139-9.

16 Navaroli N, Christopher D & Smyth P, Modeling individual email patterns over time with latent variable models, *Mach Learn*, **92(2–3)** (2013) 431–455, https://doi.org/10.1007/s10994-013-5348-5.

17 Abu-Shareha A A, Abualhaj M M, Alshahrani A & Al-Kasasbeh B, A four-state Markov model for modelling bursty traffic and benchmarking of random early detection, *Int J Netw Sci*, **8(2)** (2024) 1151–1160, https://doi.org/10.5267/j.ijdns.2023.11.019.

18 Heyman D P & Lucantoni D, Modeling multiple IP traffic streams with rate limits, *IEEE/ACM Trans Netw*, **11(6)** (2003) 948–958, https://doi.org/10.1109/tnet.2003.820252.

19 Shah-Heydari S & Le-Ngoc T, MMPP models for multimedia traffic, *Telecommun Syst*, **15(3)** (2000) 273–293.

20 Rajabi A & Wong J W, MMPP characterization of web application traffic, *20th Int Symp Model Anal Sim Comput Telecomm Syst* (IEEE), **30** (2012) 107–114, https://doi.org/10.1109/mascots.2012.22.

21 Muscariello L, Meillia M, Meo M, Marsan M A & Cigno R L, An MMPP-based hierarchical model of Internet traffic, *IEEE Int Conf Commun* (IEEE), **(4)** (2004) 2143–2147, https://doi.org/10.1109/icc.2004.1312897.

22 Faraj M K, Al-Saadi A & Albahadili R J, An investigation of using traffic load in SDN based load balancing, *Iraqi J Comput Commun Control Syst Eng*, **20(3)** (2020) 65–74, https://doi.org/10.33103/uot.ijccce.20.3.6.

23 Jadhav K A, Mohammed M M & Narayan D G, An efficient load balancing mechanism in software-defined networks, *12th Int Conf Comput Intell Commun Networks* (IEEE) 2020, 116–122, https://doi.org/10.1109/cicn49253.2020.9242601.

24 Yan J, Zhang B, Wang S, Lan H & Wang X, Burst traffic: Congestion management and performance optimization strategies in heterogeneous cognitive radio networks, *IET Commun*, **18(8)** (2024) 523–533, https://doi.org/10.1049/cmu2.12753.

25 https://www.unb.ca/cic/research/applications.html#CICFlow Meter (accessed on 17 Sep 2024)

26 Gasior J & Seredynski F, Dynamic job scheduling in the cloud using slowdown optimization and sandpile cellular automata model, Int Parallel Distrib Process Symp Workshop (IEEE) 2015, 276–285, https://doi.org/10.1109/ipdpsw.2015.139.

27 Bak P, Tang C & Wiesenfeld K, Self-organized criticality: An explanation of the 1/f noise, Phys Rev Lett, 59(4) (1987) 381, https://doi.org/10.1103/physrevlett.59.381.

28 Graphical Network Simulator (GNS3): https://docs.gns3.com/docs/, (accessed on 17 Sep 2024)

29 ONF: https://opennetworking.org/; 2019, (accessed on 10 Sep 2024)

30 Silva P H & Alves N, IPERF tool: generation and evaluation of TCP and UDP data traffic, Notas técnicas, 4(2) (2014) 1–13, https://doi.org/10.7437/nt2236-7640/2014.02.003.