University of South Carolina

Scholar Commons

Theses and Dissertations

2017

A Machine Learning Approach For Enhancing Security And Quality Of Service Of Optical Burst Switching Networks

Adel Dabash A. Rajab
University of South Carolina

Follow this and additional works at: https://scholarcommons.sc.edu/etd

Part of the Computer Engineering Commons, and the Computer Sciences Commons

Recommended Citation
Rajab, A. A.(2017). *A Machine Learning Approach For Enhancing Security And Quality Of Service Of Optical Burst Switching Networks.* (Doctoral dissertation). Retrieved from https://scholarcommons.sc.edu/etd/4402

This Open Access Dissertation is brought to you by Scholar Commons. It has been accepted for inclusion in Theses and Dissertations by an authorized administrator of Scholar Commons. For more information, please contact digres@mailbox.sc.edu.

A Machine Learning Approach for Enhancing Security and Quality of Service of Optical Burst Switching Networks

by

Adel Dabash A. Rajab

Bachelor of Science
University of South Carolina, 2005

Master of Engineering
University of South Carolina, 2007

Submitted in Partial Fulfillment of the Requirements

For the Degree of Doctor of Philosophy in

Computer Science and Engineering

College of Engineering and Computing

University of South Carolina

2017

Accepted by:

Chin-Tser Huang, Major Professor

Manton M. Matthews, Committee Member

Csilla Farkas, Committee Member

Marco Valtorta, Committee Member

Mohammed Al-Shargabi, Committee Member

Cheryl L. Addy, Vice Provost and Dean of the Graduate School

© Copyright by Adel Dabash A. Rajab, 2017
All Rights Reserved

ACKNOWLEDGEMENTS

First, I would like to express my deepest appreciation to my super-advisor Dr. Chin-Tser Huang for his patient, kindness, friendly, and unfailing support over the past few years. Without his invaluable directions, enlightenment, advices and constant supports, this work could never have been completed. Also I would like to thank Dr. Manton M. Matthews, Dr. Csilla Farkas, Dr. Marco Valtorta and Dr. Mohammed Al-Shargabi for their guidance, their valuable time to serve on my proposal/dissertation committee and for their valuable comments and advice to help improve my dissertation. Finally, I would like to express my utmost appreciation to my parents, my brother Dr. Khairan D. Rajab, my wife, Wafa M. Alyami and my kids for their love and constant support which inspired me to continue this journey.

ABSTRACT

The Optical Bust Switching (OBS) network has become one of the most promising switching technologies for building the next-generation of internet backbone infrastructure. However, OBS networks still face a number of security and Quality of Service (QoS) challenges, particularly from Burst Header Packet (BHP) flooding attacks. In OBS, a core switch handles requests, reserving one of the unoccupied channels for incoming data bursts (DB) through BHP. An attacker can exploit this fact and send malicious BHP without the corresponding DB. If unresolved, threats such as BHP flooding attacks can result in low bandwidth utilization, limited network performance, high burst loss rate, and eventually, denial of service (DoS). In this dissertation, we focus our investigations on the network security and QoS in the presence of BHP flooding attacks. First, we proposed and developed a new security model that can be embedded into OBS core switch architecture to prevent BHP flooding attacks. The countermeasure security model allows the OBS core switch to classify the ingress nodes based on their behavior and the amount of reserved resources not being utilized. A malicious node causing a BHP flooding attack will be blocked by the developed model until the risk disappears or the malicious node redeems itself. Using our security model, we can effectively and preemptively prevent a BHP flooding attack regardless of the strength of the attacker. In the second part of this dissertation, we investigated the potential use of machine learning (ML) in countering the risk of the BHP flood attack problem. In particular, we proposed and developed a new series of rules, using the decision tree method to prevent the risk of a BHP flooding attack. The proposed classification rule models were evaluated using different metrics to measure the overall performance of this approach. The experiments showed that using rules derived from the decision trees did indeed counter BHP flooding attacks, and enabled the automatic classification of edge nodes at an early stage. In the third part of this dissertation, we performed a comparative study, evaluating a number of ML techniques in classifying edge nodes, to determine the most suitable ML method to prevent this type of attack. The experimental results from a preprocessed dataset related to BHP flooding attacks showed that rule-based classifiers, in particular decision trees (C4.5), Bagging, and RIDOR, consistently derive classifiers that are more predictive, compared to alternate ML algorithms, including AdaBoost, Logistic Regression, Naïve Bayes, SVM-SMO and ANN-MultilayerPerceptron. Moreover, the harmonic mean, recall and precision results of the rule-based and tree classifiers were more competitive than those of the remaining ML algorithms. Lastly, the runtime results in ms showed that decision tree classifiers are not only more predictive, but are also more efficient than other algorithms. Thus, our findings show that decision tree identifier is the most appropriate technique for classifying ingress nodes to combat the BHP flooding attack problem.

TABLE OF CONTENTS

ACKNOWLEDGEMENTS iii

ABSTRACT iv

LIST OF TABLES ix

LIST OF FIGURES x

CHAPTER 1 INTRODUCTION 1

1.1 MOTIVATION 1

1.2 PROBLEM OVERVIEW 3

1.3 OVERVIEW OF DISSERTATION 6

CHAPTER 2 COUNTERING BURST HEADER PACKET FLOODING ATTACK IN OPTICAL BURST SWITCHING NETWORKS 9

2.1 BACKGROUND 9

2.2 DESIGN OF THE PROPOSED SECURITY MODEL 13

2.3 IMPLEMENTATION 15

2.4 EVALUATION AND ANALYSIS 22

2.5 RELATED STUDIES 27

2.6 SUMMARY 32

CHAPTER 3 DECISION TREE RULE LEARNING APPROACH TO COUNTER BURST HEADER PACKET FLOODING ATTACK IN OPTICAL BURST SWITCHING NETWORKS 33

3.1 BACKGROUND 33

3.2 BHP FLOODING ATTACK AS CLASSIFICATION PROBLEM 38

3.3 THE PROPOSED RULE-BASED MODEL FOR ANTI-BHP-FLOOD ATTACK 41

3.4 EXPERIMENTATION AND RESULT ANALYSIS 59

3.5 RELATED STUDIES 75

3.6 SUMMARY 78

CHAPTER 4 DETECTING BHP-FLOODING ATTACK IN OBS NETWORKS: A MACHINE LEARNING PROSPECTIVE 81

4.1 BACKGROUND 81

4.2 THE CONSIDERED MACHINE LEARNING TECHNIQUES 85

4.3 EXPERIMENTAL SETTINGS AND DATA 95

4.4 RESULTS ANALYSIS 100

4.5 STUDIES RELATED TO APPLICATION OF MACHINE LEARNING IN DETECTING AND CLASSIFICATION TASKS.. 107

4.6 SUMMARY 111

BIBLIOGRAPHY 113

LIST OF TABLES

Table 2.1 NCTUns Network Simulator parameter of the OBS Network configuration in evaluation 23

Table 3.1 Sample four iteration of two edge nodes 44

Table 3.2 CHI feature selection score generated from the binary-class dataset.....48

Table 3.3 Sample four iterations of the processed training dataset 49

Table 3.4 CHI feature selection score generated from the multi-class dataset 50

Table 3.5 Sample of five iterations of the processed multi-class dataset 51

Table 3.6 Confusion matrix for classification task in ML 72

Table 4.1 Sample of five iterations of the processed multi-class training dataset 97

LIST OF FIGURES

Figure 2.1 (a) Assembling of packets at an ingress node; (b) BHP (O-E-O) conversion at a core switch to allocate the resources for the incoming data burst in OBS networks 10

Figure 2.2 BHP Flooding attack on core switches in an OBS networks 12

Figure 2.3 The classification process of the proposed model 14

Figure 2.4 The proposed data structure component of the proposed security model 17

Figure 2.5 (a) Number of dropped DBs in each slot or cycle; (b) number of dropped DBs for one second; 0 indicates no DBs has been dropped 19

Figure 2.6 The algorithm process of classifying nodes 21

Figure 2.7 OBS network topology used in evaluation 22

Figure 2.8 Comparison of percentage of lost packets number in the presence of 0.2 Gbps load of malicious BHPs 24

Figure 2.9 Comparison of percentage of lost packets number in the presence of 0.5 Gbps load of malicious BHPs 26

Figure 2.10 Comparison of percentage of lost packets number in the presence of 1.0 Gbps load of malicious BHPs 27

Figure 3.1 The proposed rule based classification architecture for BHP flood Attack 42

Figure 3.2 Build_Tree algorithm for constructing a classification model 53

Figure 3.3 Decision tree model example 57

Figure 3.4 The initial binary classification model 65

Figure 3.5 The multi-class classification model with pruning 71

Figure 3.6 The confusion matrix of the decision tree model derived from the multi-class dataset...73

Figure 4.1 Steps of ML classification technique..87

Figure 4.2 Classification accuracies in % derived by the ML algorithms 101

Figure 4.3 The classifiers' sizes of RIDOR, Bagging and C4.5 algorithms............102

Figure 4.4 (a) TPs; (b) FPs; (c) TNs and (d) FNs of the ML algorithms 103

Figure 4.5 The Precision, Recall and F1 scores of the ML algorithms 106

Figure 4.6 The time in ms needed to build the classifiers of the ML algorithms 107

CHAPTER 1

INTRODUCTION

1.1 Motivation

The Optical Burst Switching (OBS) network is a promising switching technology for building the next generation of Internet infrastructure. It typically represents a trade-off between two switching technologies: optical circuit switching (OCS) and optical packet switching (OPS). Even with all the OBS's network merits, such as resiliency, bandwidth/resources efficiency, and overall economic advantages, OBS networks still suffer from several quality of service (QoS) and security issues, such as burst loss due to bursts contention, bursts scheduling, and most importantly, Denial of Service (DoS) due to Burst Header Packet (BHP) flooding attacks.

A BHP flooding attack can subjugate the core switches by sending a large number of BHPs into the network. Normally, when a core switch receives a BHP, it reserves a WDM channel for it and changes the status of the reserved channel from unoccupied to occupied. Attackers use this to flood the network with malicious BHPs without sending the corresponding data. The target node will blindly reserve a new WDM channel for each incoming malicious BHP without checking if the corresponding data arrives. When a legitimate request comes to the compromised core switch, there will be no unoccupied WDM channels available. This leads to preventing legitimate nodes from reserving the required network resources at the intermediate core switch [2], eventually causing a DoS attack.

The main motivation for this research is to examine the behavior of edge nodes to counter the risks associated with BHP flood attacks in OBS networks. This can develop efficient security techniques that can manage the problem, as well as providing contributions to enhance the QoS in OBS network. Furthermore, we aim to examine the applicability of machine learning (ML) to the problem of BHP flooding attacks in OBS networks. This is to develop a more efficient detection system enabling the core switches to classify ingress nodes in an automated manner, and identify misbehaving ones as early as possible. To achieve this, we extensively investigated various ML techniques that adopt different learning approaches to the research problem considered. We seek to identify the most relevant ML technique(s) to solve the issue of BHP flooding attacks, in addition to revealing the reasons behind the relevancy.

1.2 Problem Overview

1.2.1 Countering Burst Header Packet Flooding Attacks in Optical Burst Switching Networks

The study of detection techniques of BHP flooding attack is very limited for OBS networks. This type of attack, which relies on the flooding approach, has been studied in traditional DoS against the TCP protocol [9, 10]. For instance, the SYN flooding attack intends to exhaust the resources of the TCP/IP stack (e.g. the backlog) of the victim host by generating enormous numbers of SYN requests towards it without completing a connection setup. The victim host will be unable to accept legitimate connection requests if its backlog is fully occupied by all the fake half-opened connections [11]. However, in the context of UDP protocol over an OBS network, there is no such study on preventing or even limiting BHP flooding attacks that we are aware of. Therefore, it is important to be able to monitor or study the behavior of edge nodes in an OBS network in the likelihood of dealing with such threats.

In this part of the study, we propose and develop a new security system that can be added to OBS core node architecture to prevent BHP flood attacks. The countermeasure security system allows the OBS core node to classify the ingress nodes, based on both their behavior, and the amount of reserved resources not being utilized. A malicious node causing a BHP flooding attack will be blocked by the system until the threat is resolved. The system is implemented, tested and verified using a modified NCTUNS simulator. The analysis shows that this is highly effective in preventing BHP flooding attacks, as well as in providing the network resources for the legitimate nodes.

1.2.2. Decision Tree Rule Learning Approach to Counter Burst Header Packet Flooding Attacks in Optical Burst Switching Networks

Machine Learning (ML) is a widely adopted and powerful data analysis technique which has displayed a highly predictive performance in multiple application domains, due to its ability to discover useful hidden knowledge that can be beneficial for decision making. However, a key challenge in adopting ML to counter BHP flood attacks is the unavailability of the training datasets. Therefore, this study is two-fold. Firstly, it determines and develops a dataset from thousands of simulation runs that can be converted into a classification task. Secondly, it investigates the use of a predictive model using ML to counter the risk of BHP flooding attack dilemmas experienced in OBS networks, proposing a tree-based decision architecture as an appropriate solution.

Few related studies have used ML techniques within OBS networks [40, 50]. These studies centred on data traffic identification, whereas our study is concerned with an entirely different issue – BHP flooding attacks. To the best of our knowledge, this study is the first to offer proposals and to develop a decision-tree method of classification to solve the issue of BHP flooding attacks in OBS networks. Since previous studies have not used ML as a way of blocking misbehaving edge nodes which send DBs in OBS networks, we believe a solution is needed in order to address this critical issue in the initial phases of BHP flooding attacks.

1.2.3 Detecting BHP-Flooding Attacks in OBS Networks: A Machine Learning Prospective

A powerful and promising approach in identifying misbehaving edge nodes causing BHP flooding attacks is Machine Learning (ML), and in particular, classification techniques. A classification technique learns models by applying them to a large historical dataset derived from an edge node's performance during a simulation run. The dataset contains behavior traces from a number of edge nodes, with respect to input data characteristics, sensitivity, efficiency performance, predictive performance, and model content. The learned model can then be utilized to single out (classify) misbehaving edge nodes based on their future performance as accurately as possible, hence disciplining them. Therefore, this part of the study investigates the BHP problem by evaluating a number of ML techniques in classifying edge nodes, and determines the most suitable method to prevent this type of attack.

This study evaluates Decision Tree (C4.5), Bagging, Boosting (AdaBosst), Probabilistic (Naïve Bayes), Rule Induction (RIpple DOwn Rule Learner-RIDOR), Neural Network (NN-MultilayerPerceptron), Logistic Regression, and Support Vector Machine-Sequential Minimal Optimization (SVM-SMO) on a real dataset to identify the most appropriate method(s) to combat the BHP flood attack problem in OBS networks.

1.3 Overview of Dissertation

We will address some of the problems facing OBS networks. The organization of this dissertation is as follows:

Chapter 2 provides a background on BHP flooding attacks, and proposes a new security model, the node classifier, to counter BHP flooding attacks. We implemented a sophisticated data structure comprised of two layers, with the use of an adaptive sliding range window to classify ingress nodes into three varieties. This classification is based on the number of lost bursts from each ingress node during the time frame, to measure the performance of nodes and detect BHP flooding attacks at preliminary classes. The simulation results show that our proposed classifier is effective in preventing BHP flooding attacks.

Chapter 3 investigates the applicability of the predictive model, using ML to counter the risk of BHP flooding attacks experienced in OBS networks, and proposing a decision-tree based architecture as an appropriate solution. This architecture contains a learning algorithm that extracts novel rules from tree models using data processed from several simulation runs. Our simulation results show that the rules derived from our learning algorithm are able to accurately classify 93% of the BHP flooding attacks into either Behaving (B) or Misbehaving (M) classes. Moreover, the rules can further categorize the Misbehaving edge nodes into four sub-class labels of Misbehaving-Block (Block), Behaving-No Block (No Block), Misbehaving-No Block (M-No Block), and Misbehaving-Wait (M-Wait) with 87% accuracy. The results clearly show that our proposed decision-tree model is a viable solution in comparison to decisions undertaken by expert domains or human network administrators.

Chapter 4 builds on our previous work from Chapter 3. One method to prevent a BHP flood attack is to detect the misbehaving edge nodes overloading the network with malicious BHPs, and take the proper action to secure and sustain the QoS performance in an OBS network using ML. This chapter investigates the BHP flood attack problem by evaluating a number of ML techniques in classifying edge nodes, and determines the most suitable method to prevent this type of attack. To be precise, we evaluate Decision Tree (C4.5), Bagging, Boosting (AdaBosst), Probabilistic (Naïve Bayes), Rule Induction (RIpple DOwn Rule Learner-RIDOR), Neural Network (NN-

MultilayerPerceptron), Logistic Regression, and Support Vector Machine-Sequential Minimal Optimization (SVM-SMO) on a real dataset to identify the most effective method(s) to combat the BHP flood attack problem in OBS networks.

CHAPTER 2

COUNTERING BURST HEADER PACKET FLOODING ATTACK IN OPTICAL BURST SWITCHING NETWORKS

2.1 Background

Optical network is a modern network technology for transmitting information from one place to another by sending light through an optical fiber. The light forms an electromagnetic carrier wave that is modulated to carry information [1]. These features of optical networks provide high speed and huge bandwidth, which make optical networks a viable choice of the Internet backbone infrastructure [1]. The popularity of optical networks has led to the replacement of traditional copper wires by optical network fibers, and has also motivated many enterprises to invest in optical burst switching (OBS) network in particular within the past few years.

OBS network is a promising switching technology for building the next-generation Internet infrastructure [2, 3, 4]. It represents a trade-off between two switching technologies: optical circuit switching [2] and optical packet switching [3]. It uses one-way signaling scheme with an out-of-band method, which mean the burst header packet (BHP) is sent in a separate channel from the data burst (DB) channel. OBS is designed for a better utilization of wavelengths in order to minimize the latency (setup delay) and avoid the use of the optical buffers [4].

OBS transmission technique keeps the data in the optical domain and allows for sophisticated electronic processing of control header information at another domain. As illustrated in Figure 2.1 (a), the transmission works by assembling the incoming data traffic from clients at the edge node (called *ingress*) of the OBS network into what is called data burst (DB). Then a BHP, which contains the information about the DB packets, including the burst length, arrival

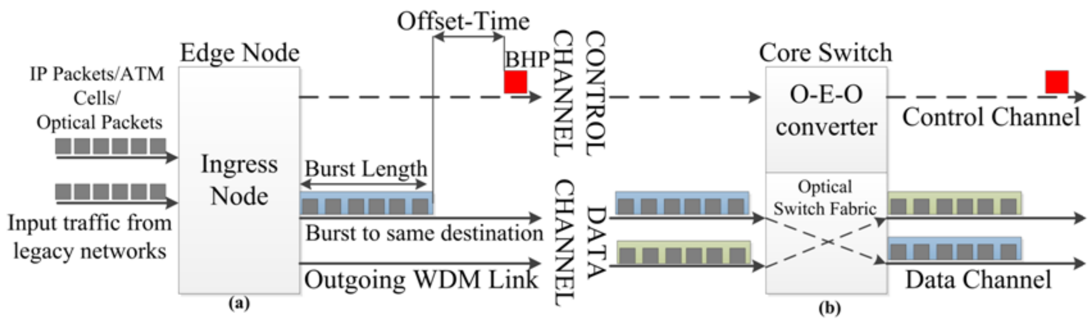


Figure 2.1 (a) Assembling of packets at an ingress node; (b) BHP (O-E-O) conversion at a core switch to allocate the resources for the incoming data burst in OBS networks.

time, offset time, etc., is transmitted ahead over a devoted Wavelength Division Multiplexing (WDM) channel (out-of-band). The BHP precedes the DB by a time known as offset time in order to reserve the required resources and to set up the path configuration for the DBs in the core switches [5]. The BHP goes through the Optical-Electronic-Optical (O-E-O) conversion at each intermediate node and is processed electronically to allocate the resources for the incoming data burst into the optical domain [6, 7] as shown in Figure 2.1 (b). OBS data bursts may have different lengths, and encompass many types of traffic (IP packets, ATM cells, optical packets, etc.). The ingress sends the data in the form of bursts which will be disassembled at the destination edge router (called *egress*).

Even with all its merits, OBS networks like any other communication networks can suffer from several threats. Some of the known threats are orphan bursts, redirection of data bursts, replay, BHP flooding attack, fake burst header attack and denial of service attack [8].

In this work, we are interested in the denial of service (DoS) that can be caused by BHP flooding attack, and aim to prevent a legitimate BHP from reserving the required network resources at the intermediate core switch. This type of attack relies on the flooding approach that has been studied in traditional DoS against the TCP protocol [9, 10]. For instance, the SYN flooding attack intends to exhaust the resources of the TCP/IP stack (e.g. the backlog) of the victim host by generating enormous numbers of SYN requests toward the victim host without completing connection setup. The victim host will be unable to accept legitimate connection requests if its backlog is fully occupied by all the fake half-opened connections [11].

In a similar way, the BHP flooding attack can subjugate the core switches when a malicious node sends large numbers of BHPs into the network without transmitting the actual DBs. When a core switch reserves WDM channels for the incoming BHPs, it changes the status of the reserved channels from unoccupied to occupied. Figure 2.2 demonstrates that when the target node (a core switch) receives malicious BHPs, the target node starts reserving new WDM channel for

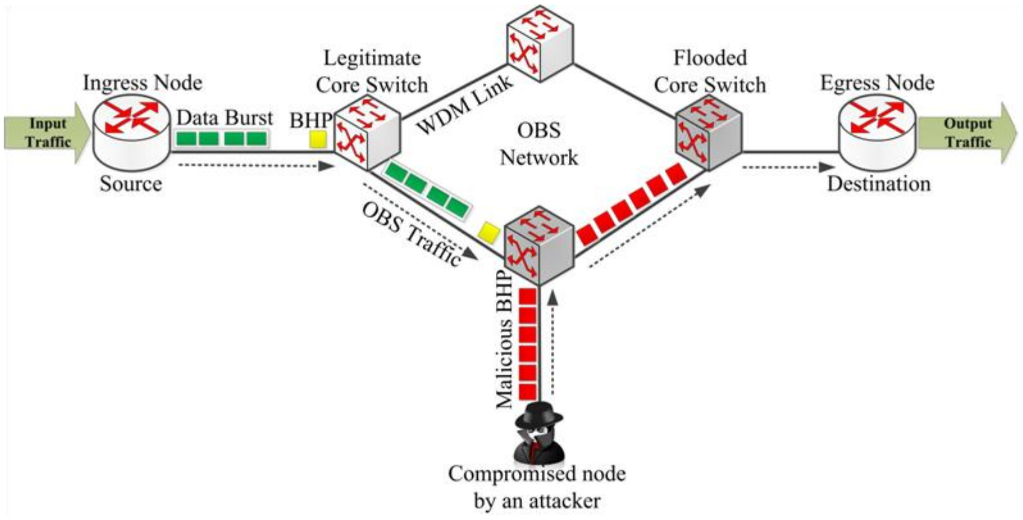


Figure 2.2 BHP Flooding attack on core switches in an OBS networks.

each malicious BHP. This prevents a legitimate BHP from reserving the required network resources at the intermediate core switch [2]. When a legitimate DB arrives and there are no unoccupied WDM channels available, the arrived DB will be dropped by the core switch and the reserved channels will be waiting for unidentified bursts which may never arrive [12].

This paper proposes a new security model, called the *node classifier*, which is designed to counter BHP flooding attacks. The proposed model has an adaptive sliding range window to classify ingress nodes into three *classes*. This classification will be based on the number of lost burst from each ingress node during time window to measure the performance of nodes and detect BHP flooding attack at preliminary classes.

2.2 Design of the Proposed Security Model

In this section, we present our proposed security model designed for BHP flooding countermeasure, and it is illustrated in Figure 2.3. In order to combat BHP flooding attacks, we study and analyze the behavior of each node to discover the point when the node is misbehaving. This can be considered an alert to prevent the malicious BHPs from reserving the network resources. The proposed security model has several merits summarized as follows:

– It only requires software modification and implementation, and does not require additional hardware.

– It is easy to be integrated with existing core switches architecture.

– It is not necessary to modify all the core switches at once for the model to effectively work. Incremental deployment of the model can still enhance the security of the OBS network.

The model works by classifying all the ingress nodes into three possible classes, namely *Trusted*, *Suspicious*, and *Blocked*. Initially, the model classifies all the nodes into the Trusted class. As time goes on, the classifier changes the class assignment of each ingress node based on its observed performance such as

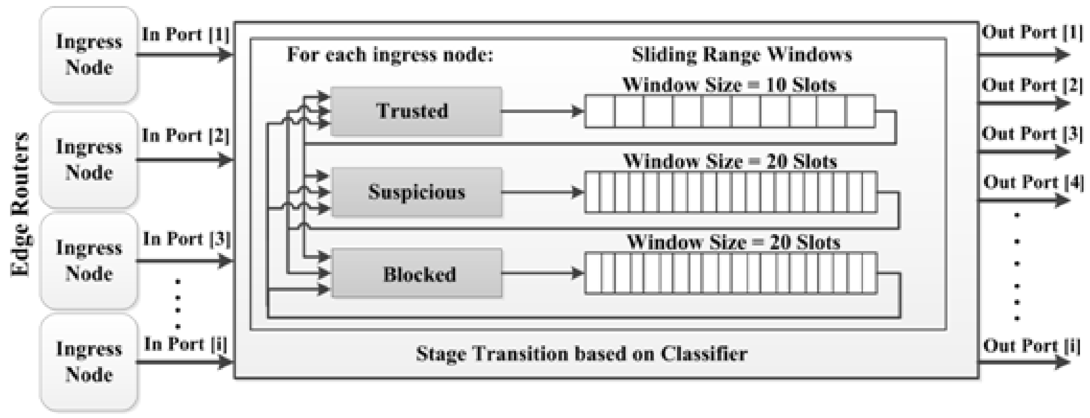


Figure 2.3 The classification process of the proposed model.

packet arrival rate and packet dropping rate using a sliding range window. For example, if a node is acting normally by sending the BHP with its corresponding DB on the expected time (BHP arrival time + offset time), the node will be assigned to the Trusted class. However, when the ingress node at some point does not send a predefined number of corresponding DBs within the expected time, the classifier assigns Suspicious class to the node. In cases when the transmitted data do not arrive at all and the packet dropping rate keeps increasing, the ingress node will then be assigned to the Blocked class, hence subsequent BHPs from this node will no longer be accepted and none of the available resource will be reserved for this node. Lastly, in cases of any BHP flooding attack, the classifier will eventually add the compromised node to the blocked list.

An ingress node can redeem itself from the Blocked and Suspicious classes back to Trusted by improving its throughput and lowering the packet dropping rate, i.e. stopping the BHP flooding attacks. In typical BHP attack, the attacking ingress node keeps sending the bogus BHPs. In this case, the core switches that place this attacking node in the Blocked class will not be able to forward its BHPs and will not allow the node to be allocated network resources. However, when the blocked node stops sending bogus BHPs and starts sending legitimate DBs, the arrived DBs will be used to redeem the node from the Blocked class.

2.3 Implementation

In this section, we discuss the implementation of our classification model in detail, and introduce its three main components (data structure, sliding range window, and classifier).

2.3.1 Data Structure

The model's data structure is composed of two layers. The first layer allows a core switch to store and maintain information about each connected port (representing an ingress node) including the following fields:

1) *Port ID*.

2) *Class*: The class currently assigned to this ingress node (i.e. Trusted, Suspicious, Blocked)

3) *Ingress node array size*: The size of the array for each ingress node. The size will be incremented by each received BHP and decremented by each dropped BHP from the array.

4) *The number of dropped BHPs*: This parameter keeps account of how many BHPs from each ingress node have been dropped based on the sliding range window.

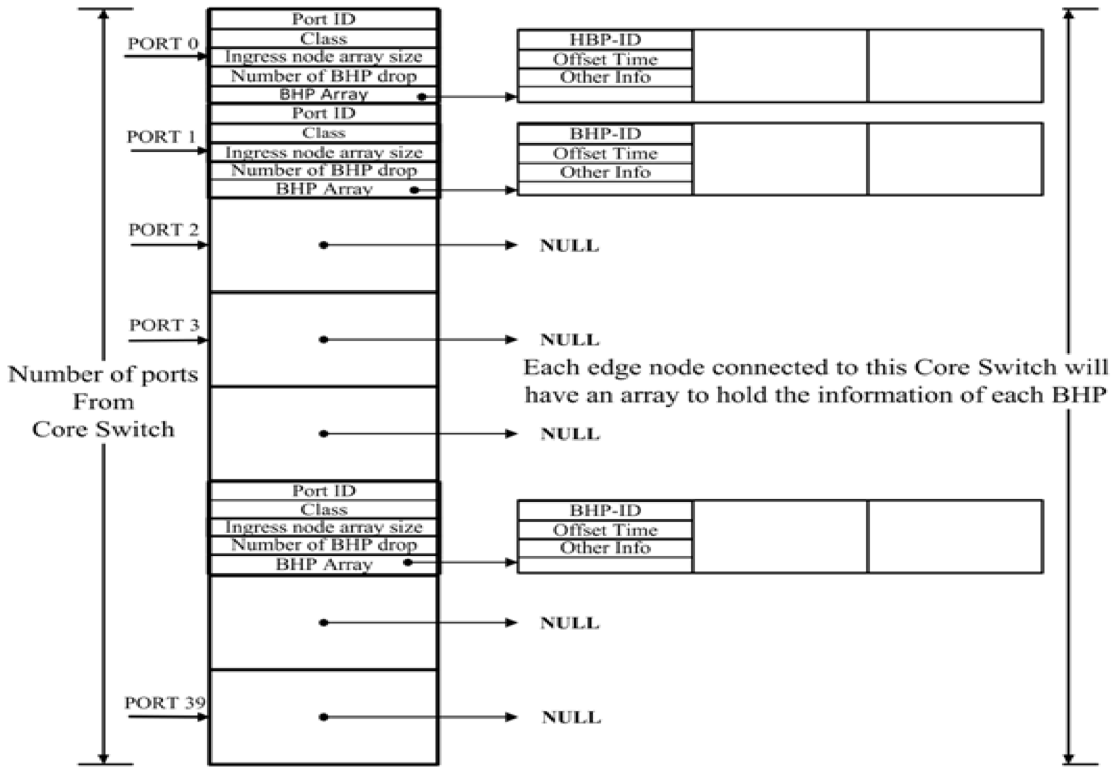
5) *BHP Array*: A pointer to the array of the BHPs. The array will be created dynamically for memory management purpose.

The second layer of the model's data structure is used to store information about the BHPs received from each incoming node (ingress node or core switch), including the following fields:

1) *BHP_ID*: This item is used to check which BHP does and does not have corresponding data burst received.

2) *Offset Time*: This is the time after which a BHP is considered part of a flooding attack when no more data arrive

The primary reason of using this data structure is to efficiently manage and store the data regarding each connected node. Figure 2.4 depicts a bird's eye view of this data structure management process.



PORT 0
Port ID
Class
Ingress node array size
Number of BHP drop
BHP Array
HBP-ID
Offset Time
Other Info
PORT 1
BHP-ID
PORT 2
NULL
PORT 3
Number of ports From Core Switch
Each edge node connected to this Core Switch will have an array to hold the information of each BHP
PORT 39

Figure 2.4 The proposed data structure component of the proposed security model.

2.3.2 Sliding Range Window

The proposed classification model utilizes a sliding range window scheme that is implemented as a circular queue. The window enables the classifier to monitor the behavior of each connected node over short and long periods to assign the appropriate and accurate class to each node.

The size of the window and the number of slots within the window need to be considered and configured carefully. Since most network performance metrics such as throughput or dropping rate are usually calculated in the unit of seconds, e.g. transmitted bytes per second, a natural choice of the window size is one second. However, a congestion or unexpected high dropping in data traffic may happen wherein the number of dropped DB may fluctuate in each slot. For example, consider the following worst case scenario in which only one BHP is transmitted in one second and the corresponding DB has not arrived, the result is 100% dropped packet rate in this period. Our classifier will block the node since the expected DB did not arrive. For this reason, we have to monitor the behavior of the edge nodes over short and long periods of time by computing packet dropping rate in each time slot using a sliding range window.

Moreover, the time range threshold cannot be set too long such that the attacker can flood the network within a short period of time and then discontinue doing so without being detected. Further, the time range cannot be set too short either, otherwise we cannot accurately determine the behavior of the node. Hence, in the case of Trusted class, we divide the window (one second) into 10 slots (one tenth of a second for each slot) during experimentations, whereas in the cases of Suspicious and Blocked classes, we double the number of time slots to 20 to closely monitor the node behavior.

Within the sliding range window, there are multiple counters for calculating the numbers of transmitted and dropped BHPs. We define W_S and

W_E as the start and end of the sliding range window, respectively. The sliding range window method often finds the total number of dropped and arrived DB packets per slot using transmitted BHPs per slot, and it calculates the dropped packets rate per slot or over the entire window. Our model considers each time slot and the entire window range W (one second) to monitor the behavior of the ingress nodes. Subsequently, each ingress node will be assigned a class based on its packet dropping rate.

Figure 2.5(a) shows an example of counting the number of harmful BHPs that its corresponding DB have not been received for a short period (i.e. per slot), such as ((0 = slot 1), (6 = slot 2), (5 = slot 3)). Figure 2.5(b) also illustrates the number of harmful BHPs for a long period (i.e. per second), such as (S1 through S10).

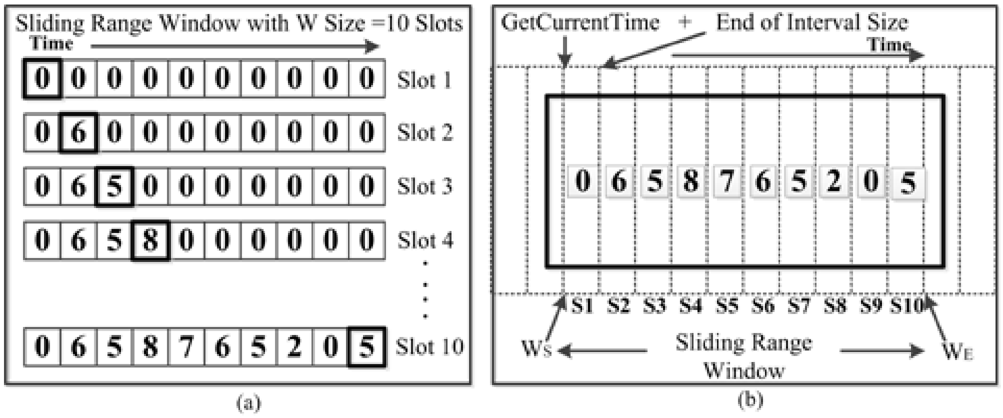

Sliding Range Window with W Size =10 Slots
Time
0 0 0 0 0 0 0 0 0 0 Slot 1
0 6 0 0 0 0 0 0 0 0 Slot 2
0 6 5 0 0 0 0 0 0 0 Slot 3
0 6 5 8 0 0 0 0 0 0 Slot 4
0 6 5 8 7 6 5 2 0 5 Slot 10
(a)
GetCurrentTime + End of Interval Size
Time
0 6 5 8 7 6 5 2 0 5
S1 S2 S3 S4 S5 S6 S7 S8 S9 S10
Ws
Sliding Range Window
WE
(b)

Figure 2.5 (a) Number of dropped DBs in each slot or cycle; (b) number of dropped DBs for one second; 0 indicates no DBs has been dropped.

2.3.3 Classifier

The basic idea of the classification model is to detect harmful ingress node at preliminary classes. However, what is the appropriate criterion for judging whether a node is under a BHP flooding attack? Based on previous research studies, i.e. [32, 33], a consistent high utilization of the network resources normally greater than 40% is an indication of network's performance deterioration. Moreover, link utilization ratio (the link's bandwidth being currently utilized by the network traffic) is another indicator of possible threats to the node. The node utilization can be calculated according to [33] using equation (1)

$$\textit{Utilization \%} = (\textit{data bits} \times 100) / (\textit{bandwidth} \times \textit{interval}) \quad (1)$$

The above two observations are typically used as indicators when a node is under a possible threat. In our model, we use 40% BHPs that do not have corresponding DB packets received as a threshold for blocking attacks. This is since 40% of the resources are reserved by malicious BHPs and are unused. This is a condition where we can be confident that the network is under BHP flooding attack. Note that this condition is distinguishable from network congestion, since in a congested network not only DB packets will be dropped, but BHPs as well. When using the 40% utilization as the single boundary of judging whether BHP flooding attack this may risk ignoring normal packet dropping cases such as network congestion. Therefore, we split the 40% threshold into two ranges, in which the first 20% is considered trustworthy, and the second 20% is considered suspicious but allowing the node a chance to redeem itself as trustworthy again once the abnormality disappears. We define the class assignment value of the node using the following rules:

- Trusted if: $80\% \leq ArrivedRate \leq 100\%$.
- Suspicious if: $60\% \leq ArrivedRate < 80\%$.
- Blocked if: $ArrivedRate < 60\%$.

```
Algorithm: Assign Node Class
Input:          Edge Router Number
Output :        Node Class
Preprocessing:  Data Structure and Sliding Window are populated with
                edge router information

STEP 1      Check Class
            IF (Edge Router has NO Class) THEN RETURN
            TRUSTED

STEP 2      Calculate from each slot in the sliding window
              Total number of dropped packets (DP)
              Total number of arrived packets (AP)

STEP 3      Calculate percentage of packet drop rate
              PDR ← (DP / DP+AP) * 100

STEP 4      Assign node class by checking
              IF       (PDR ≤ 20) THEN  Class ← TRUSTED
              ELSE IF  (PDR ≤ 40) THEN  Class ← SUSPICIOUS
              ELSE                      Class ← BLOCKED

STEP 5      RETURN Class
```

Figure 2.6 the process of classifying nodes

Figure 2.6 shows the algorithm process of classifying nodes. The procedure uses the sliding range window explained earlier and the classifier to assign each node its appropriate value.

2.4 Evaluation and Analysis

In this section, we explain the simulation setup and experimental results of our model. The simulation is conducted on a modified version of NCTUns network simulator to evaluate the performance of the proposed classifier [32]. The topology used in the simulation is shown in Figure 2.7, which contains eight core switches (3, 4, 5, 6, 7, 8, 9, 10) to simulate an OBS network, two ingress edge routers (2, 11), one egress edge router (12), one legitimate sender (1), one receiver

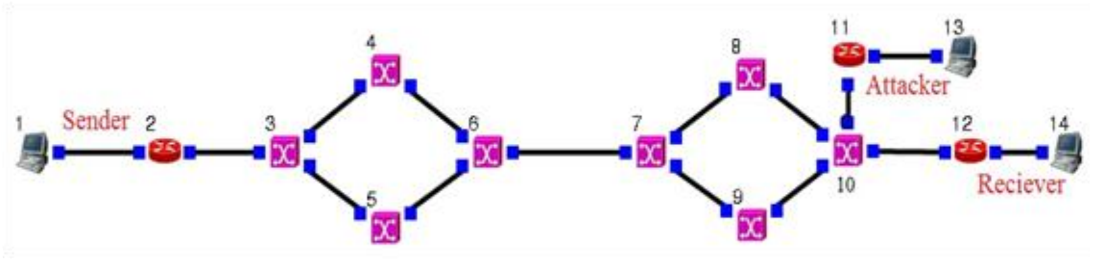


Figure 2.7 OBS network topology used in evaluation.

(14) and one attacker (13). It is worth to note that the attacker node can be located in different places of the topology, but we choose to place it near the destination in order to emphasize its effect and because the probability of remaining undetected is high. Moreover, although our classifier can handle any number of

ingress nodes and any number of attackers, in our experiments we use only one legitimate ingress node and one attacker. This is because we are interested in testing our classifier against the BHP flooding attack rather than testing the possible congestion in this topology.

Table 2.1 shows the simulation parameters for the OBS network configuration. As for the traffic files, we created ten trace files with incremental traffic load rate (0.1 Gbps, 0.2 Gbps, 0.3 Gbps, …, 1 Gbps respectively, where 1 Gbps is the maximum rate allowed by the simulator for each node) which represent the traffic transmitted by the legitimate sender.

Table 2.1

NCTUns Network Simulator parameter of the OBS Network configuration in evaluation

Parameter	**Value**
Link bandwidth	1000Mb/s
Propagation delay	1 μs
Bit error rate	0
Maximum burst length	15000 bytes
Number of BHP channels	1
Number of DB channels	1
Use of Wavelength Conversion	No
Use of Fiber Delay Line (FDL)	No
Transport Layer Protocol	UDP

We conducted experiments based on a BHP flooding attacker of varied strengths to evaluate and compare our classifier with the default scheme which has no security measures. The objectives of these experiments are twofold:

1. Firstly, we want to observe the impact of BHP flooding attack on legitimate traffic when no security measure is employed;
2. Secondly, we want to evaluate the effectiveness of our classifier in preventing the BHP flooding attack.

We start with a lightweight attacker with 0.2 Gbps load. By attacker's load we refer to the network resources collectively requested by the harmful BHPs sent by the attacker. Lightweight is relative to the traffic loads of other trace files used in our experiments. To increase the difficulty of detection, we make the attacker randomly flood the intermediate core switch with a random load of malicious BHPs with different interval time, and let the average attacker load reaches 0.2 Gbps. We test this lightweight attacker against all 10 trace files, with each trace file run three times and calculate the average. The results in terms of

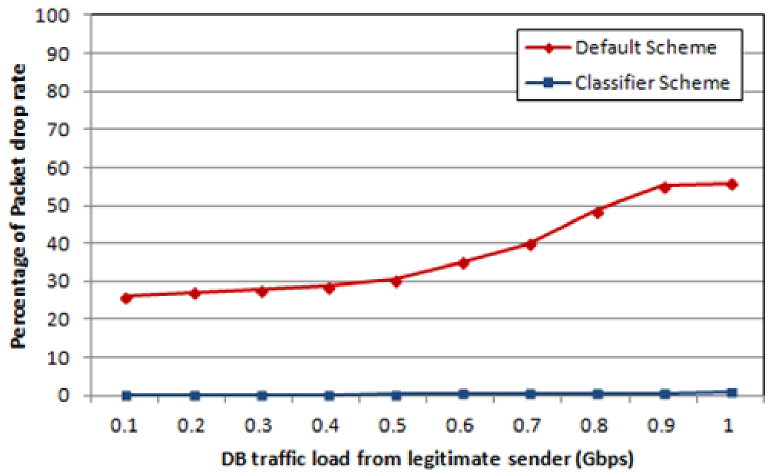


Figure 2.8 Comparison of percentage of lost packets number in the presence of 0.2 Gbps load of malicious BHPs.

packet dropping rate are shown in Figure 2.8. From this figure we can see that at the beginning when the legitimate traffic load is not very high, the packet dropping rate for the default scheme is not high. This is because the attacker load is relatively low which still leaves much bandwidth available for the legitimate traffic. The dropping rate of legitimate traffic starts at 26% and stabilizes to around 55% as the legitimate traffic load increases to 1 Gbps. This is expected since the legitimate traffic load becomes gradually higher than the attacker load and will request more bandwidth, which, however, has been falsely reserved by the attacker.

The packet dropping rate of our classifier remains low, only around 1%. This is because our classifier detects the misbehaving node and assigns it to the Blocked class. Once the system blocks the attacking node, all the resources requested by the legitimate ingress node are granted and hence the packet dropping rate becomes low even for high traffic loads. The 1% of dropped packets is due to the period when the attacker was not yet classified into the Blocked class at the beginning of the simulation and was granted the resources requested by the bad BHPs, which leads to the slight dropping of legitimate packets at the initial phase.

We continue testing with a medium-strength attacker with a load of 0.5 Gbps, and a powerful attacker with a load of 1 Gbps, which is the maximum load allowed by the simulator for each node. The results for these two cases are shown in Figures 2.9 and 2.10 respectively. For the default scheme, the packet dropping rate demonstrates similar trend as in Figure 2.8, except that the stable

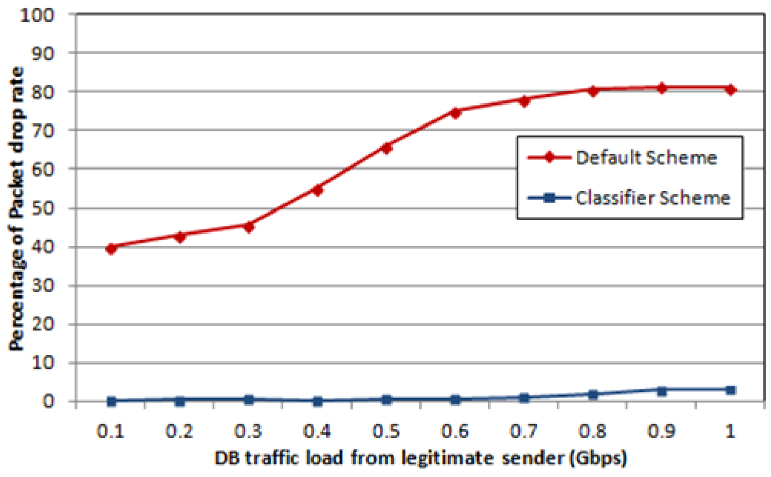


Figure 2.9 Comparison of percentage of lost packets number in the presence of 0.5 Gbps load of malicious BHPs.

packet dropping rate is around 80% for the medium-strength attacker, and around 90% for the powerful attacker. These results are reasonable since higher attacker load gives the attacker a better chance to reserve the DB channel for longer time and may result in higher packet dropping rate for the legitimate traffic. By contrast, both Figures 2.9 and 2.10 show that for our classifier, the packet dropping rate remains as low as between 1% and 5%, which clearly demonstrates the effectiveness of our classifier in stopping the BHP flooding attack.

Overall, the experimental results lead us to reach the following two conclusions. Firstly, if the BHP flooding attacker is more powerful to transmit its bad BHPs to request network resources at a higher rate, it can cause more legitimate DB packets to be dropped. Secondly, our classifier can effectively

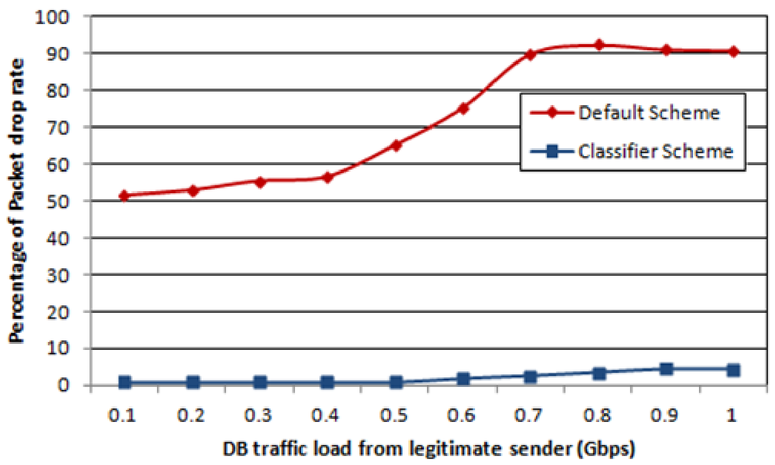


Figure 2.10 Comparison of percentage of lost packets number in the presence of 1.0 Gbps load of malicious BHPs.

prevent the BHP flooding attack regardless of the strength of the attacker. Furthermore, the model relies on detecting/preventing the BHP flooding attack in time which makes our classifier model perform better.

2.5 Related Studies

In OBS network, there are several potential threats including traffic analysis, eavesdropping, spoofing, data burst redirection attack, burst duplication attack, replay attack, burstification attack, land attack and BHP flooding attack [8]. In this section, the focus will be on discussing security issues related to OBS network and present common threats particularly DoS flooding attacks based on the protocol level.

In traffic analysis or eavesdropping attack, the attacker attempts to gain or access some unauthorized information about the target node by passively listening to the communication. The attacker in OBS can scan for an open vulnerability, and then intercepts active BHPs in order to compromise the corresponding data burst. When BHP gets compromised, the attacker will be able to analyze and monitor the transmitted information from the compromised BHPs which may expose him to the transparent DBs that contain the critical information. Passive attackers are hard to detect and can be seen a true troubling threat in OBS networks. In [13, 14, 15], the authors propose prevention techniques to overcome this type of attacks.

In data burst redirection attack, the attacker injects a malicious BHP into the OBS network, causing the corresponding DB to be redirected to unauthorized destination. In OBS network, a DB is configured to follow the optical routing path set up by its associated BHP, but it is not able to authenticate the routing path of the BHP. If a malicious BHP is injected into the OBS network at a time such as offset time, any active DB can be misdirected to an unauthorized destination. The authors of [2], [12], [16] developed solutions to fight data burst redirection attacks.

In burstification attack, the attacker can compromise the ingress node by changing the DB size value that was originally recorded in the BHP. Then the actual DB could be mishandled as a different DB value according to the modified BHP, in which the receiving node will have to inquire for retransmission of the burst. This attack can happen at both edge (including ingress and egress) and core switches. The attacker will compromise the ingress node and modifies the burst size value to a larger size, such that the burst reservation time will increase, resulting in longer propagation delay and increased burst setup latency. In [17], the authors thoroughly discussed the burstification besides other threats that may occur on optical nodes.

In land attack, the attacker compromises a node by making a copy of the BHP, modifying its destination address to the source address, and injecting the modified BHP into the OBS network. The result is that the corresponding data burst will reach the intended destination and the source itself. Due to this attack, some network resources will be wasted in sending the data burst back to the source, which in turn will cause some restriction on the sending resource in the best possible behavior. In [18], the authors discussed in details this type of attack.

Research works more relevant to ours include [9, 10], [19], whose authors also addressed the problem of preventing BHP flooding attacks that may cause DoS. For instance, the authors of [9] proposed a new flow filtering architecture that operates at the optical layer to filter out flooding attacks at early stages. The filtering process is performed based on comparing the offset time included in the BHP and the actual delay between this BHP and the associated DB. However, due to the high traffic rates in optical networks, the proposed flow filtering mechanisms cannot be effectively applied.

In [10], the authors study the denial of service attack resulting from BHP flooding attack in the resources reservation protocols. The proposed countermeasure module uses the concept of optical codewords to optically filter the fake BHP and identify the compromised source node in the network. This module can work at the edge node but it cannot optically filter the fake BHP at the core switch. Moreover, the module does not perform any system validation at the core switch to evaluate the performance of each connected node in the network based on packet arrival rate/packet dropping rate and allowing/blocking security rules.

In [19], the authors proposed a prevention mechanism to detect BHP flooding attack in TCP over OBS network. This mechanism is limited based on the statistical data collected from packets, and the threshold is not well defined to justify whether the behavior of the node is normal or under an attack. Moreover, the solution proposed by the authors increases the end-to-end delay which reduces the performance of the computer network with respect to its associated Quality of service (QoS) variables. [19]'s prevention mechanism only reduces the trust value of the node until it reaches a value below the threshold. However, there is no real or immediate action to stop the attacks before they occur.

It is worthy to note that flooding is a very common way to launch distributed denial of service (DDoS) attacks, in which the distributed attacking sources simultaneously transmit an overwhelming amount of malicious unwanted traffic toward the victim machine to congest the victim's network and drain the victim's communication and computation resources. Many approaches have been proposed to address DDoS flooding attacks, such as rate-limiting schemes [20,21,22,23,24,25] and IP traceback schemes [26,27,28,29,30,31]. However, the main purposes of these schemes are to identify the attacking sources and restrain them from sending excessive traffic. By contrast, the problem with BHP flooding attacks is that the attacking sources, whose identities are already known to the core switches, do not send out the corresponding data burst traffic after sending BHPs to reserve network bandwidth. This major difference deems the rate-limiting and IP traceback schemes unfit for addressing BHP flooding attacks.

2.6 Summary

In this work, we proposed a new security classification model for countering BHP flooding attack with an adaptive sliding range window to detect nodes based on their behavior. The classifier enables core switches to measure the performance of incoming nodes and detect BHP flooding attack. The simulation results show that our proposed classifier is effective in preventing BHP flooding attack. They show that the overall packet dropping rate when the classifier is used is less than 5% in all traffic load cases under BHP flooding attack. This is a remarkable improvement over the default scheme that employs no security measures, which results in up to 90% packet dropping rate. The proposed classifier has been studied with various scenarios with different cases to demonstrate its capability of securing the OBS network from BHP flooding attack, such as critical links in the network. We note during experimentations that our classifier not only can secure the core switches in the OBS network, but also has the potential to improve the QoS performance of the OBS network. In the near future, we will extend the solution to increase the performance of our current model and add QoS improvement features for OBS networks based on the node classification.

CHAPTER 3

DECISION TREE RULE LEARNING APPROACH TO COUNTER BURST HEADER PACKET FLOODING ATTACK IN OPTICAL BURST SWITCHING NETWORKS

3.1 Background

An Optical Network (ON) is a common network for transmitting data from source to destination via an optical fibre medium using light [1]. Featuring efficient quality performance indicators such as bandwidth and speed in contrast to traditional networks, ON is the preferable option for Internet infrastructure [2]. In order to make use of the huge bandwidth of ON, Optical Burst Switching (OBS) was proposed in [3] as the next generation of optical switching technology. Once it has obtained the User Datagram Protocol (UDP) packets, OBS network will assemble the packets from the clients at the edge nodes (ingress node) into a data burst (DB) and a burst header packet (BHP) will be transmitted in advance in order to preserve the network resources required before the DB is actually sent. However, an attacker can exploit this fact and can make an ingress node (source node) overloads (flood) the network with BHPs that reserve the resources without transmitting the actual DB [4]. It is important therefore, to ensure that prevention of BHP flooding attacks is set as a high priority in OBS, as it could severely reduce the performance of the entire network and eventually cause a Denial of Service (DoS) [5]. Despite the many advantages of the OBS network such as resiliency, bandwidth efficiency as well as its economic benefits, QoS and security can become issues for these networks, with consequences including burst loss as a result of BHP flood attacks [35, 10]. Attacks of this type are reliant on the flooding approach which has been examined in traditional DoS against the TCP protocol [11].

A limited number of studies exist in relation to dealing with and preventing issues caused by BHP flood attacks within OBS networks (such as, in [9, 10, 35, 36]). For example, [9] proposed a flow filtering architecture operating at the optical layer to filter out BHP flood attacks at an early stage. The filtering process is performed through a comparison between the offset time included in the BHP and the actual delay between this BHP and the associated DB. In [10], the authors examined the issue of DoS within the resources' reservation protocols. This countermeasure adopts the method of using optical code words which filter out fake BHP in order to identify the compromised source node within the network. In [35] meanwhile, the authors proposed a prevention method which was built by gathering statistical data from packets used to detect a BHP flood attack in TCP within an OBS network. In [36], the authors proposed and developed a new security model which could be integrated within an OBS core switch architecture to prevent BHP flooding attacks. Using a countermeasure security model enables the core switch of the OBS to classify the ingress nodes according to their behaviour in addition to the amount of reserved resources not being used. A malicious node which causes a BHP flooding attack can be blocked using this model until the risk is disappeared. The issues with these methods however is that they do not use the machine learning (ML) techniques which are available to identify edge node behaviour in order to counter the risk of BHP flooding attacks. These methods also contain their own faults and can be limited in their execution and therefore need further authentication.

This study will examine the problems of DoS resulting from BHP flooding attacks in which legitimate BHPs are prevented from preserving network resources for legitimate DBs. The ML architecture we have developed features sets of beneficial learnings gathered from past simulations carried out using a reduced number of features including the bandwidth used, the average packet drop rate as well as the average delay time per second and other factors (further details are presented in Section 3.3). One of the most encouraging data analysis methods used by researchers for prediction is ML [37]. It features intelligent techniques in order to complete a specific task which is usually linked to knowledge building or isolating patterns which are concealed [38]. Little research exists around the classification of edge nodes in preventing BHP flooding attacks within OBS networks. A large proportion of this research related to classifying traffic of computer networks through the use of ML, for example [39, 35, 40, 41], are not solely focused on the classification of BHP flood attack within OBS networks. Through our study, we find that ML can have a vital function in preventing BHP flood attacks, resulting in improved QoS. These results from the prediction decision used in ML techniques rely on automated knowledge learnt, which can impact the predisposed decisions made by users, leading to improved predictive accuracy.

Classification is one of the most frequent ML tasks which require the prediction of a target attribute [42]. Classification related to building a model (the classifier) using a recorded set of data along with a number of variables using data processing, and then using the classifier in order to predict precise attributes (the class) within the hidden dataset [43]. In relation to the issue of BHP flooding attacks, the edge nodes can be classified into pre-defined classes, e. g. Behaving or Misbehaving, meaning that this issue can be classified within predictive tasks, and of course, classification. The key objective therefore will be to forecast the type of edge nodes involved in order to prevent the risks associated with occupying network resources with improper use.

Of the different classification approaches used within various domains, the rule extraction from decision tree models is one of the most effective [44]. Decision tree models are a type of classification method which is cleverly built using an information theory approach [45]. For example, [46] used Entropy and Information Gain methods to build decision trees within an algorithm labelled C4.5. Using the training dataset to form the tree, every individual path stemming from the root node to the leaf represents a corresponding If-Then rule.

In this Chapter, we have investigated the potential of machine learning in countering the risk of BHP flood attack problem. In particular, we have proposed and developed a new series of rules using the decision tree method as a way to prevent the risks of the BHP flood attack problem. Firstly, the decision tree method proposed will build a binary classification model which can classify the edge nodes into two classes (Behaving and Misbehaving). Hundreds of simulation runs were used in order to collect the data to build the binary models, gathering the various attributes associated with how the edge nodes perform. It worth to note that the use of only two types of classes (binary) is intended for the researchers and developers who are interested in security of OBS networks and want only to identify suspicion nodes (Misbehaving). Next, the classification models were improved through splitting the Misbehaving class into further sub-class labels as we desired to establish a priority procedure for data transmission from the edge nodes (further details provided in Section 3.4). The proposed classification rule models were evaluated using different metrics to measure the overall performance of this approach. The experiments showed that using rules derived from the decision trees did indeed counter the BHP flooding attack problem and enabled the automatic classification of edge nodes at an early stage.

3.2 BHP Flood Attack as Classification Problem

In the field of artificial intelligence, ML is regarded as one of the most popular areas of research, and used for data analysis in various applications [42]. In recent years, researchers in the field of artificial intelligence and ML have developed several analytical methods which are directly concerned with the classification of datasets generated within various business areas. Some of the ML methods include Decision Trees [46], Greedy Induction [47], Associative Classification [43], Neural Network [47], Probabilistic [48], and Support Vector Machines [49]. Common ML methods tend to separate classification into three different sub-tasks, which are:

- ***Learning:*** Learning involves the tasks of executing data processing on the dataset entered. One example is the associative classification method, in which class association rules, if they exist, are identified using an association rule algorithm. Meanwhile, decision trees will single out the different input variable using Entropy (see Equation 4) in addition to a support vector machine method tries which will recognize a clear line within a hyper plane to identify the possible characteristics of the target class.

- ***Model Building:*** This task is related to the building of a predictive model using the results of the previous task. The ML method will sometimes change the results of the learning task through retaining key elements to build the predictive model. For example, decision trees will prune pointless branches to cut down overfitting.
- ***Class Forecasting:*** This task will use the ML method to measure the built model's success rate in predicting the test data's class values. The results of this step are used to serve as the model's error rate or establish the negatives, model efficiency, classification accuracy and other factors.

When a core switch within an OBS network reserves the wavelength division multiplexing (WDM) channels for the incoming BHPs, it will alter the status of these channels from unoccupied to occupied. If a core switch becomes the target of a BHP flooding attack and receives malicious BHPs, it will begin to reserve new WDM channels in correspondence with each received malicious BHP. This will prevent a legitimate BHP from reserving the essential network resources at this intermediate core switch [2]. If a legitimate DB becomes present and no WDM channels are available, the core switch will drop the arrived DB, while the reserved channels will be waiting for unidentified bursts which may never arrive [50]. This research will focus on the BHP flooding attacks in terms of a classification issue as it is concerned with classifying edge nodes into a single type out of a limited number of possible class values such as Behaving or Misbehaving. Misbehaving nodes are defined as those send BHPs at a substantially higher rate without sending the corresponding DB.

The definition of the classification problem is based on [51]. The training dataset is labelled as (X_i, C_i), where X_i represents the combination of variables within the training dataset aside from the class variable, and C_i represents the class variable. A vector $x_i \in R^m$ can be assigned one of two disjoint point sets C_1 or C_2 within an m-dimensional feature space. $x_i \in R^m$ is the ith training data row and $c_i \in \{1, \dots, K\}$ represents the ith class value. The aim is to derive a function, F, that maximizes the chance that $F(x) = c_i$ for each test data. Two-class problem (binary classification) is the simplest form of the classification; c_i can be either -1 (Misbehaving) or 1 (Behaving). This means Function $F(x)$ is:

$$F(x) = \begin{Bmatrix} -1 \; x \in \; C_1 \\ 1 \; x \in \; C_2 \end{Bmatrix} \qquad (1)$$

In relation to an OBS network, BHP flooding attacks can be prevented if edge nodes are correctly classified into the appropriate types, with misbehaving nodes being identified. More specifically, if the determination of the edge nodes that reserve network resources without proper use can occur in an early phase, then these nodes can be blocked. The blocking of the edge nodes means that the other 'behaving' edge nodes will be able to effectively reserve network resources, improving the resource management and QoS of the network. To deal with this issue, it is possible to build a model created from the edge nodes' previous behavior displayed during the simulations. This enables the model to be used to automatically place the edge nodes into the right classifications in future simulations.

3.3 The Proposed Rule-based Model for Anti-BHP-Flood Attack

In this section, we will present the proposed classification architecture and its different phases. The various phases which must be included to promote classification rules necessary to counter BHP flooding attacks can be seen in Figure 3.1. In the initial phase, simulations were carried out to gather data relevant to the edge nodes performance indicators, which was then catalogued as the training dataset. Next, the training data set was pre-processed to remove noise or statistical biases. Once the data is cleaned, a filtering process takes place to detect the features which are the most influential, as well as reduce the dimensionality of the data. After selecting the features, the decision tree algorithm will process the dataset to build a classification model that will then be transformed into rule sets. Any unnecessary rules are then be pruned to eliminate any redundant rules. The rule sets are finally applied to classify the edge nodes into a number of pre-defined classes against new test cases that had the same performance indicators. The following subsequent sections will explain the proposed classification architecture in further detail.

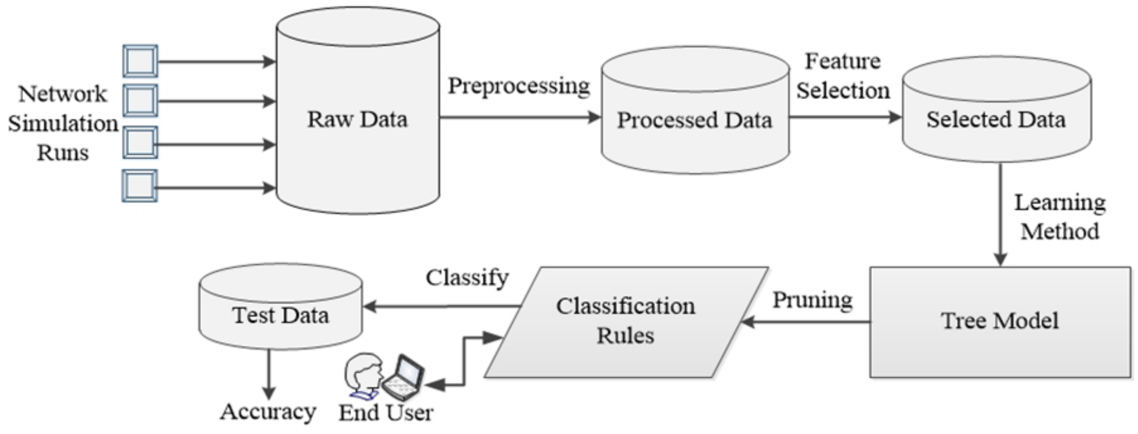


Figure 3.1 The proposed rule based classification architecture for BHP flood Attack

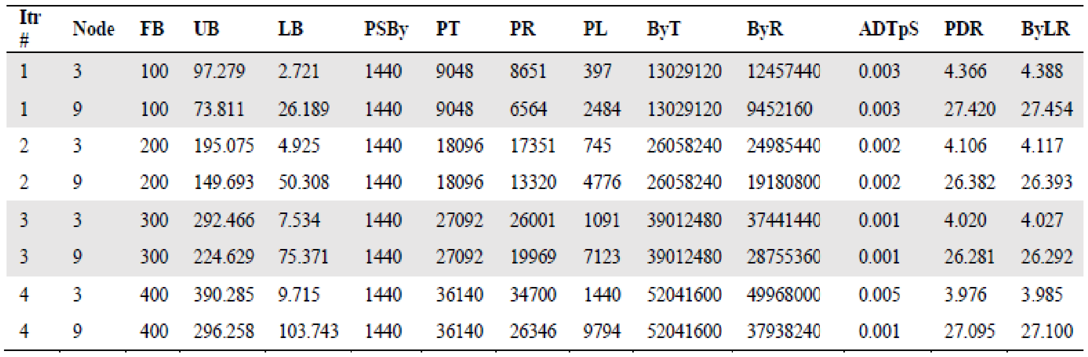
3.3.1 Understanding the Dataset

When using ML to prevent BHP flood attacks, establishing the right training dataset from various nodes through the simulations remains a significant challenge. The domain expert will typically identify many variables that will directly and indirectly affect the OBS network which are also relevant to the sending performance of the nodes. The first challenge faced is creating the training dataset and determining the needed variables that ensure the ML method can be used in order to construct a predictive model. Identifying the necessary mechanism for converting the training dataset to the task of classification is another challenge. The initial challenge can be addressed by recording as many variables linked to the sending nodes as possible while the simulator is working. On the other hand, the second challenge can be addressed by establishing a new variable, the 'class' within the training dataset. Values of classes can be assigned using a knowledge base to the variable which is entered by domain experts, which will reduce biased data cases and make the building of the training dataset more legitimate. A comparison will be made between this class and the proposed ML's predicted class to establish the effectiveness of the model in terms of accurate classification

3.3.2 Training Dataset Preparation

The right training dataset is very crucial step in order for ML algorithms to work as indented. To prepare the training dataset for the purpose of identifying misbehaving nodes that are causing BHP flooding attack, numbers of simulations were carried out in order to collect the different variables related to the OBS network performance. Significant variables recorded include the sending node number, allocated bandwidth, bandwidth used, bandwidth lost, packet transmitted, packet dropping rate, packet received, transmitted byte, received byte, average delay time per second, and the percentage of BHP flooding attack. For illustration purposes, Table 3.1 lists just four iterations for two of the edge nodes.

Table 3.1
Sample four iterations of two edge nodes



Itr #	Node	FB	UB	LB	PSBy	PT	PR	PL	ByT	ByR	ADTpS	PDR	ByLR
1	3	100	97.279	2.721	1440	9048	8651	397	13029120	12457440	0.003	4.366	4.388
1	9	100	73.811	26.189	1440	9048	6564	2484	13029120	9452160	0.003	27.420	27.454
2	3	200	195.075	4.925	1440	18096	17351	745	26058240	24985440	0.002	4.106	4.117
2	9	200	149.693	50.308	1440	18096	13320	4776	26058240	19180800	0.002	26.382	26.393
3	3	300	292.466	7.534	1440	27092	26001	1091	39012480	37441440	0.001	4.020	4.027
3	9	300	224.629	75.371	1440	27092	19969	7123	39012480	28755360	0.001	26.281	26.292
4	3	400	390.285	9.715	1440	36140	34700	1440	52041600	49968000	0.005	3.976	3.985
4	9	400	296.258	103.743	1440	36140	26346	9794	52041600	37938240	0.001	27.095	27.100

Variables' Descriptions

- **Itr**: The iteration number.
- **Node**: The edge node label.
- **FB**: Initial Bandwidth assigned (given) to each node, the user (usr) in the experiments assign these values.
- **UB**: This is what each node could reserve from the assigned Bandwidth from FB column. The drops here are due to congestions.
- **LB**: The amount of lost Bandwidth by each node from the assigned Bandwidth at column FB.
- **PSBy**: Packets size in Byte assigned specifically for each node to transmit. Note: 60 Byte will be added to the 1440 for the IP Header and the UDP Header ((Data size 1440 Byte) + (IP Header 40 Byte) + (UDP Header 20 Byte)) =1500 Byte.
- **PT**: Total transmitted packets (per second) for each node based on the assigned Bandwidth.
- **PR**: Total received packets (per second) for each node based on the reserved Bandwidth.
- **PL**: Total lost packets (per second) for each node, which based on the lost Bandwidth.
- **ByT**: Total transmitted Byte (per second) for each node.
- **ByR**: Total received Byte (per second) for each node based on the reserved Bandwidth.
- **ADTpS**: Average Delay Time (per second) for each node. This is (End-to End Delay).

- **PDR**: Percentage of Packets Drop Rate for each node.
- **ByLR**: Percentage of Lost Byte Rate for each node.

Initially, the edge nodes were classified into only two classes, Behaving (B) and Misbehaving (M), as the issue was to identify the misbehaving nodes that reserve resources without the right usage and secure the network by the right action such as blocking it. We refer to this dataset as the 'binary dataset' since we have only the two classes B and M. However, to improve the presentation of data further, and to better demonstrate the BHP flood attack scenario during the simulations, the binary dataset was augmented and a new dataset (multi-class dataset) was created. The target classes of the multi-class dataset was linked with four possible sub-class labels, i.e. Behaving-No Block (No Block), Misbehaving-No Block (M-No Block), Misbehaving-Wait (M-Wait), and Misbehaving-Block (Block). These were assigned to a new column called "New Class: Action" based on the level of BHP flood attacks for 200 runs and using two edge nodes.

In order to increase the statistical significance of the variables by smoothing their values and reducing the data variations or sudden drops in an iteration for each variable, the initial dataset (binary) and the augmented dataset (multi-class) were pre-processed. This was done by computing the average for 10 consecutive iterations per variable, and for each node as one new data instance.

More specifically, for each node variable, the value of the data instance at its first iteration was the first 10 consecutive values' averages in iteration 1 to 10, while at iteration 2, the second new instance value is iteration 2-11's average values, etc. Finally, the dataset's class values were assigned by a domain expert, i.e. (B and M) for the binary dataset and (No Block, M-No Block, M-Wait, and Block) for the multi-class dataset. The class assignments were based on a rule of thumb on two of the variables: the premeditated false resource utilization rate (percentage of BHP flooding attack) and the actual packet drop rate. (For more details about the simulation setup and the training dataset see section 3.4.1)

3.3.3 Feature Selection

Choosing which features should be used to distinguish between behaving and misbehaving edge nodes is another challenge. However, this can be addressed through employing feature selection methods to filter the initial data. To determine the most relevant features in relation to the problem, a filtering method called Chi-square testing (CHI) was applied [52] with the Correlation Feature Set (CFS) [47] for verification.

CHI is often used as a statistics metric and has been employed for use in both supervised and unsupervised learning applications to assess input data features' validity. The CHI metric will test two chosen variables to measure their level of independence. In many cases, the target class of the input dataset will be one of the variables, while the normal feature will be the other. In relation to the classification problem which is under discussion, the type of node (B, M) for the binary dataset and (Block, No Block, M-No Block, and M-Wait) for the multi-class dataset, will be the class, while the feature could be anything from packet drop rate, bandwidth used, packet received, or other feature. Either way, the selected features will be unrelated to the other, although they will have significant links to the other classes being obtained. The CHI's correlation score is given as:

$$CHI(f,c) = \frac{N\times(XZ-YW)}{(X+W)\times(Y+Z)\times(X+Y)\times(W+Z)}. \tag{2}$$

Where X is the frequency feature f and class c appearing together, Y is the frequency feature, f appears without class c, W is the frequency class c appears without feature f, Z is the frequency neither f or c appears, and N is the number of instances in the training dataset.

The result of the CHI was verified using CFS which is known for being a more pessimistic form of filtering. CFS is used for verification because it has continuous input features (numbers and decimals), as opposed to the categorical features of CHI. Therefore, the input dataset needed to be discretized before CHI feature selection could be used. As CFS is able to work with both categorical and continuous features, it is important to use CFS to verify the dataset.

3.3.3.1 Feature Selection on Binary Dataset

Table 3.2 shows the CHI scores from the processed binary dataset. It is important to note that four input features have been disregarded as part of the discretization process (where numeric variables are transformed into discrete variables) – the iteration number, node number, FB, and PSBy (abbreviation details can be found in Table 3.1). Once the CHI was applied, acceptable scores and high correlation against the class variables were noted in the features of

Table 3.2
CHI feature selection score generated from the binary training dataset

CHI Score	Feature Name
161.236	Lost_Bandwidth
153.442	Packet_Dropping_Rate
53.553	Packet_Received
53.553	Used_Bandwidth
53.553	Received_Byte
51.818	Full_Bandwidth
51.818	Packet_Transmitted
51.818	Transmitted_Byte
33.324	Average_Delay_Time_Per_Sec

"Lost Bandwidth" and "Packet Dropping Rate". However, when applying CFS filtering method for verification before final results are recorded, three features seem to survive without the discretization process: "Used Bandwidth," "Average_Delay_Time_Per_Sec," and "Packet_Dropping_Rate". Both filtering methods have highlighted "Packet_Dropping_Rate", making the feature appear significant and so it has been retained. The "Average_Delay_Time_Per_Sec" and "Used Bandwidth" features have also been retained. The "Lost Bandwidth" was discarded since it is the complement for "Used Bandwidth" and it would be ineffective if both were kept. Finally, both the iteration number and the sending node were also noted.

From the initial input dataset, three features (columns 3-5 in Table 3.3) have been chosen in addition to the iteration#, Node#, and the target class, as shown in Table 3.3. Two possible values exist for the target class: Behaving (B) or Misbehaving (M), which can be seen in the final column in the table. Table 3.3 shows only a sample of four iterations for two of the edge nodes (9, 3) which are sending data. Each value in the selected features represents an average value computed from 10 sequential iterations. This is a vital step to ensure that the statistical power is retained and bias is minimized within the node performance results.

Table 3.3
Sample four iterations of the processed binary training dataset

Iteration	Node#	10-Run-AVG-Drop-Rate	10- Run-AVG-Bandwidth-Use	10-Run-Delay	Target Class
1	9	0.412	0.40	0.0034	M
1	3	0.169	0.64	0.0009	B
2	9	0.414	0.39	0.0034	M
2	3	0.168	0.64	0.0008	B
3	9	0.404	0.40	0.0034	M
3	3	0.174	0.63	0.0007	B
4	9	0.403	0.40	0.0032	M
4	3	0.179	0.63	0.0008	B

3.3.3.2 Feature Selection on Multi-Class Dataset

Table 3.4 below details the CHI scores for key variables for the multi-class dataset. The BHP flooding attack variable however has been ignored in the data processing phase since it has been used to construct the dataset, and thus there is concern about overfitting the model with biased results. An over-fitted model can generate great performance on the training dataset but a poor performance on any unseen datasets; therefore it has been ignored in the data processing phase. The CFS filtering method was used to verify the feature selection, meaning that the 10-Run-AVG-Drop-Rate remained the CHI and CFS's most common variable. As a result, the selected variables were the 10-Run-AVG-Drop-Rate and two variables selected through CHI filtering which were the 10-Run-AVG-Bandwidth-Use and 10-Run-Delay.

Table 3.4
CHI feature selection score generated from the multi-class dataset

CHI Score	Feature Name
796	BHP Flood
503.133	10-Run-AVG-Drop-Rate
320.271	10-Run-AVG-Bandwidth-Use
28.117	10-Run-Delay

For illustration purposes, Table 3.5 shows how the edge nodes for every iteration was used to send data, and included the iterations which predominately featured misbehaving actions from both edge node. Iteration #5, for example, shows that node 3 has been assigned class = Block, as the BHP sent by this node did not contain data (high BHP flooding) meaning that a significant proportion of the BHPs had been reserved without use. However, although the same iteration showed misbehaving actions in node 9, the node was still permitted to transfer data (misbehaving but no block). Iteration #4 appears more complicated, as each of the nodes shows misbehaving actions, but not at the level of a significant BHP flood attack. As its dropping rate was smaller, node 9 was given higher priority over node 3. Nodes with low packet dropping rates (less than 0.39 as a result of the rule already produced by the decision tree) would not be blocked and would be allowed to send data.

Table 3.5
Sample of five iterations of the processed multi-class training dataset

Iteration	Node#	10-Run-AVG-Drop-Rate	10-Run-AVG-Bandwidth-Use	10-Run-AVG Delay	Node Status	BHP Flood	New Class: Action
1	3	0.45	0.559	0.0005	M	0.369	M-No Block
1	9	0.42	0.589	0.0007	M	0.3697	M-Wait
2	3	0.466	0.543	0.0005	M	0.2887	M-Wait
2	9	0.416	0.593	0.0006	M	0.2541	M-No Block
3	3	0.468	0.541	0.0005	M	0.3041	M-Wait
3	9	0.419	0.591	0.0007	M	0.2972	M-No Block
4	3	0.476	0.532	0.0006	M	0.3621	M-Wait
4	9	0.415	0.594	0.0008	M	0.3112	M-No Block
5	3	0.482	0.526	0.0005	M	0.4337	Block
5	9	0.414	0.596	0.0008	M	0.3848	M-No Block

After the datasets are created and pre-processed, the CHI filtering method was used to select the important features and verified using CFS filtering method, the next stage is to build the classification model.

3.3.4 Construction of Classification Models

We propose an anti-BHP flooding attack classification algorithm based on decision trees. The proposed classification approach is shown in Figure 3.2. It separates data instances into subsets, which are further divided into smaller divisions until the subsets are homogenous or the termination condition has been met. If at the initial stage the data instances fit into one class (known as a pure dataset), a single rule will result to enable prediction of that class (lines 1-3). If, however, more than one label is associated with the training data, the Information Gain (IG) metric (Equation 3) will be used to determine the tree's root node. This means the algorithm will iterate over the different attributes' values in the training dataset to determine the attribute that has the largest information gain in splitting the training data per available class label (Lines 4-5). The information gain for each attribute is computed based on Entropy (Equation 4). Once the largest gained attribute is identified then it be assigned as the root node (Lines 7-8), and the training cases are then clustered based on the splitting of this root variable. In other words, subsets of the training data are formed based on the root node's possible values (line 9). Then, the same building tree function is invoked to on these subsets repeatedly, and the decision tree algorithm will keep dividing and clustering the data cases, while leaving heterogeneous data cases (conquer) out (lines 10-12). Each of the output tree's nodes represents the features needing classification, while the values of the features are represented by the branches. Once the initial tree is built and the algorithm terminates the training phase (lines 1-12) the algorithm invokes a pruning procedure that prunes unnecessary branches in the tree without hindering the overall tree forecasted accuracy.

```
Input: Training dataset ($T$) and $Tree$ { }
Output: Classifier with rules $CL$
Build_Tree ($T$)
    1) If $T$ has one class
    2)   stop
    3) End if
    4) For each variable $v \in T$ Do
    5)  calculate the information gain from splitting on $v$
    6) End for
    7) $v\_Max$ = The variable with the largest calculated
       information gain
    8) $Tree$ = $v\_Max$ is added as a root node
    9) $T_v$ = subsets of $T$ that includes $v\_Max$
    10) For each example in $T_v$ Do
    11)   $Tree_v$= build_Tree ($T_v$)
    12)   $Tree \leftarrow Tree_v$
    13)   Pruning function
    14)   Predict test cases
    15) End for
```

Figure 3.2. Build_Tree algorithm for constructing a classification model

$$Gain\ (T, f) = Entropy\ (T) - \sum ((|\ T_f\ |\ /\ |\ T\ |) \ast \text{Entropy } T_f)) \tag{3}$$

$$Entropy\ (T) = \sum -P_c \log_2\ P_c \tag{4}$$

where P_c = Probability that T belongs to class l, T_f = Subset of T for which feature F has value $f_{a.}$, $|T_f|$ = Number of examples in T_f, and $|T|$ = Size of T.

Once the tree reaches a point where it cannot grow further, the building process will stop. One potential condition for termination involves growing the node continuously until every data instance is connected to the same class, or that similar values are shared by the data instances. The algorithm deals with many issues. The nominal and continual variables are handled by the classification algorithm, which means it is effective for noise tolerance. The algorithm also deals with the missing values and considers them important, and thus they can be estimated using probabilistic weights. For a specific variable (feature) with a missing value, each of its branches is given a weight based on its corresponding estimated probability after splitting that variable. For instance, if we have one variable "Gender" which has two possible classes in a problem (Yes, No) with missing values. If, through branching Gender, we have 100 instances - 70 associated with Female and 30 associated with Male, the weights assigned to the classes for this variable are 70/100 and 30/100. Now, when we get a data example without gender (missing value), this can be estimated at being weighted 70% (female) and 30% (male). This enables us to allocate more than one class with this data example, i.e. Yes and NO with weights 0.7 and 0.3 respectively.

The classification algorithm deals with overfitting by preventing the tree from growing once it reaches a certain point; to prune redundant partial trees does not contribute to overall predictive performance. The algorithm employs sub-tree replacement by replacing these unnecessary nodes with leaves. A post-pruning procedure involves testing pairs of nodes with the common parent to verify whether joining them together would possibly improve the Entropy by less than a predetermined value. If so, the leaves are merged into a single node with all possible outcomes. Equations 5 and 6 give more insight about sub-tree pruning.

$$q(v) = \frac{N_v - N_{v,c} + 0.5}{N_v} \tag{5}$$

where N_v is the number of training cases at node v and $N_{v,c}$ is the number of training cases belonging to the largest frequency class at node v. The error rate at sub-tree T is calculated as

$$q(T) = \frac{\sum_{l \in leafs(T)} N_l - N_{l,c} + 0.5}{\sum_{l \in leafs(T)} N_l} \tag{6}$$

After constructing the decision tree out of the training dataset, every path from the root node to the leaf of the tree will then be converted into If-Then classification rules. There are three reasons for this conversion:

1) The classification system will be easy to understand, making the rules manageable to allow network administrators to comprehend how BHP flooding attacks work, as well as the possible issues arising.
2) Data priority and blocking policies will be developed as a result of determining the sending behaviour of the edge nodes during the primary stages.
3) Overlapping and redundancy within the rules can be identified by the network administrator, allowing processes for rule pruning to be developed.

For illustration purposes, the process of converting the tree into rules is explained in Figure 3.3, which represents a simple decision tree for the binary classification of nodes. In the figure, the nodes are represented as rectangles while leaves are shown as circles. The root node represents the feature that best divides the cases, i.e. "10-Run-AVG-Drop-Rate". Classification based on the best features continues until a parsimonious representation is obtained. In this example, once "10-Run-AVG-Drop-Rate" was chosen, the first branch was based on values less than or equal to 0.38. This split leads to the second variable, "10-Run-AVG-Bandwidth-Use". The variable then splits on values larger than/equal to 0.80 and values less than 0.80. Each of these splits reaches a leaf node that denotes the possible class. In Figure 3.3, the tree consists of two variables, the class (B, M), and three possible rules. Each of these rules denotes a path from the root node to the leaf node. Decision trees are useful given their representation for classification problems, which constitutes a large portion of everyday applications.

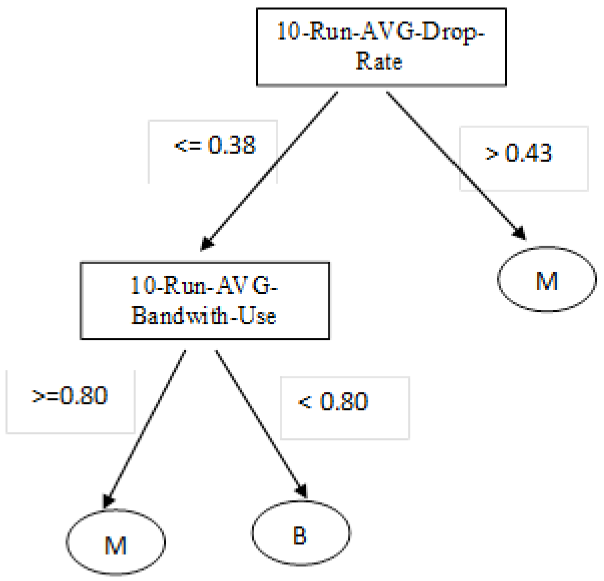


Figure 3.3 Decision tree model example.

The following reasons demonstrate why a decision tree was chosen as a way to create the models:

1) Decision tree models have been previously used by many researchers to solve classification problems, i.e. [53, 54], etc. They offer high performance in terms of classification accuracy and in different application fields.

2) Decision tree models are easy to understand since they can easily transform into a knowledge base that contains a set of If-Then rules. Recall then, that each path from the root to the leaf denotes a rule. The novice

user, or even domain experts, will be more interested in the rationale behind any predictive decision rather than just the predictive decision itself. Since the models contain rules that are easy to interpret and manage, these models can be extremely useful in decision making by end-users when compared with models derived by other ML approaches such as support vector machines, probabilistic, or neural networks.

3) The availability of decision tree models in ML tools such as R [55] and WEKA [56] means that researchers do not have to re-develop them, which saves a significant amount of time.

3.3.5 Prediction of Test Cases

Finally, the proposed classification model will use the established rule sets to predict the value of each class. We proposed a basic prediction method that takes into account the first rule that completely matches the test case variables' values. Alternatively, the algorithm will search for the rule corresponding to all attribute values inside the test case, in order for the actual test case to be classified. To accomplish the task of prediction, our prediction method covers the discovered rules in a top-down fashion and assigns the class of the rule that matches the test case variables' values. If there are no rules fully matching the test case, then the class label of the first partly matching rule will be assigned to the test case. If no rules partly match the test case, then the majority class in the training dataset is given to the test case.

3.4 Experimentation and Results Analysis

3.4.1 Preliminaries

In this section, we evaluate the performance of the proposed decision tree technique on a dataset that consists of a number of simulation runs aiming to improve the performance of UDP over OBS networks. All of the variables related to the network's performance were collected by running the NCTUns simulator for hundreds of runs on NSFNET topology [32]. One of the advantages of using NSFNET topology is that we will have the ability to add and simulate any number of nodes in the OBS network. The second advantage is that we will have the ability to record all the different cases that might take a place in the normal scenario where a single and multiple nodes are placed at different location. Furthermore, we will have the ability to position the attacker at any location, observe the behavior of the attacker and record its effect on other legitimate nodes based on its location. Using the NCTUns simulator with the necessary OBS modules [32], we collect the data used for classification which resemble the real world OBS network for training purpose. Therefore, for the normal scenario, the topology used to establish the training dataset in the simulation contains fourteen core switches which linked to an ingress or egress (edge routers) and then linked to a legitimate senders or receivers (Host-PC). As mentioned earlier, the location of the ingress and egress edge routers are randomly chosen in order to examine the network's performance of multiple nodes at different location in the OBS network. The following strategy has been developed to create the training dataset.

1) Set the duration of the simulation to 10 minutes and the number of edge nodes to *M*.
2) Record the different variables (see Table 3.1 for details).
3) The edge nodes' bandwidth capacity varied during the simulations in order to assess different situations. This is done to ensure all possible cases – normal, contention and congestion are covered.
4) Repeat for *N* number of the simulations.

Initially, we record the performance of each individual node with only one sender and one receiver. For each simulation run, the bandwidth of the node was assigned to 100 Mbps and then incrementally increases to 200 Mbps, 300 Mbps, 400 Mbps, until we reach the maximum bandwidth of the simulator which is 1000 Mbps. This is in turn to record and observe how much traffic each node can transmit based on its distance from the receiver when assigned different bandwidth. Afterwards, we start by randomly adding more senders and receivers, increasing the bandwidth of nodes at each simulation run (100 Mbps, 200 Mbps, 300 Mbps ...1000 Mbps) and simultaneously making the nodes transmit as much packets as it can in one second. This is in order to observe all the variables related to the node's performance i.e. (packets drop rate), the amount of traffic which has been transmitted by each node and also to distinguish between the normal, contention, and congestion scenarios during the simulation for each run. On the other hand, for the attacker scenario, we duplicate the same topology but this time by randomly placing the attacker at different location. This is for the aim to record the node's performance and observe the affect of the attacker on the legitimate node when placed on different locations.

During the simulations, the network load was adjusted in each simulation run, featured random (attacker node) and static (legitimate node) traffic for the number of nodes to assess the classifiers' effectiveness. In addition, the attacker node can be located in different places of the topology; hence we randomly place it at different locations to seek its true performance on the OBS network. Moreover, although our topology can handle any number of ingress nodes and any number of attackers, in the experiments we used single, multiple legitimate ingress nodes and one attacker in each simulation run since we are interested in testing the classifiers against the BHP flooding attack rather than testing the possible congestion in this topology. Every wavelength channel has 1 Gbps of bandwidth capacity (1 Gbps is the maximum rate allowed by the simulator for each node) which represent the traffic transmitted by the legitimate sender. Each WDM link has one control channel for BHPs and two data channels for DBs. The wavelength conversion capability was not assumed at the core switch.

Table 2.1 shows the simulation parameters for the OBS network configuration. As for generating the traffic, the UDP traffic was transmitted using the greedy mode (which transmits the maximum number of packets) with an average packet size of 1500 bytes and duration of one second for each run. On the other hand, the attacker's traffic have been generated using the simulator, but without pre-setting values. Situations for the edge nodes that (in simulation runs) would end up in random levels of BHP flood attacks were created. A point has been made to show scenarios in which there are occupied resources in the OBS network without utilization with different occupancies. The simulator may run for several minutes to achieve the result for "one second" depending on the load assigned.

For the learning process, we used supervised learning approach. This is since we are aiming to build a predictive model to counter the BHP flooding attack problem using the data collected from previous performance results of the edge nodes during a number of simulation runs. Supervised learning involves processing datasets to learn the classification models that are then utilized or used to automatically classify edge nodes during a future simulation run to the right class. The learning during the data processing phase is often guided toward a target variable called the class label and hence the model generated only predicts the target class variable. A common example of supervised learning is loan approval application in banking which aims to either approve or reject loans submitted by clients.

Since we are seeking to identify the behaviour of UDP over OBS networks by predicting the behaving ingress nodes from the misbehaving ones, supervised learning approach was adopted to accomplish this task. In supervised learning, an input data with several input variables plus a target class label is needed. The input data in our case consists of different performance indicators related to the UDP over OBS networks that have been formed and a target class label (as previously discussed in section 3.3.3). This data is basically called a training dataset and it is used as an input to the supervised learning algorithm to

a) Discover useful correlations between the performance indicators and the class label

b) Construct a classification model (classifier) that can be utilized to forecast the class label value in test cases

The supervised learning algorithm utilized to build the predictive model is based on useful If-Then knowledge base derived by a proposed decision tree algorithm (Sections 3.4.2 and 3.4.3 gives further details). To test the proposed classification algorithm and its types of feature selection, the WEKA ML tool was used [56]. WEKA is an open source Java platform implemented at Waikato University of New Zealand. The platform features various techniques for data analysis and can be used to perform a number of tasks including visualization, predictive and descriptive tasks. Classification, rule association, clustering, time series, regression and feature selection are some of WEKA's techniques.

In order to calculate the measure of evaluation during the building of the predictive models, a ten-fold cross validation evaluation approach was adopted [37]. In ML testing, this is a popular method as it can help to reduce overfitting within the training dataset. Overfitting typically occurs when the learning method over-trains on the input dataset to maximize the predictive performance of the resulting models. This leads to the serious issue of displaying effective performance on the training data, but poor performance on any test data, and thus the models' performance cannot be generalized, with the models being rejected. The ten-fold cross validation process works through splitting the training data set into ten partitions. The classification algorithm is trained on nine partitions and then evaluated on the remaining partition. The procedure is repeated 10 times and accuracies derived at each run are averaged.

All experiments have been conducted using a computing machine with a 2.3 GHz processor. C4.5 decision tree algorithms have been used for data processing to derive the predictive models for the BHP flooding attacks [46]. Finally, as discussed previously, CHI and CFS WEKA filters were used for feature selection [52, 57].

3.4.2 Results on the Binary-Class Dataset

After the training dataset has been processed and the features have been selected, the decision tree algorithm was applied to the binary-class dataset in

```
10-Run-AVG-Drop-Rate < 0.395560763
|  10-Run-AVG-Bandwidth-Use < 0.8003310674999999: B
|  10-Run-AVG-Bandwidth-Use >= 0.8003310674999999: M
10-Run-AVG-Drop-Rate >= 0.395560763
|  10-Run-AVG-Drop-Rate < 0.410186542
| |  10-Run-AVG-Drop-Rate < 0.4060369875
| | |  10-Run-Delay < 5.5215E-4
| | | |  10-Run-Delay < 5.237E-4: M
| | | |  10-Run-Delay >= 5.237E-4: B
| | |  10-Run-Delay >= 5.5215E-4: M
| |  10-Run-AVG-Drop-Rate >= 0.4060369875: B
|  10-Run-AVG-Drop-Rate >= 0.410186542: M
```

Figure. 3.4. The initial binary classification model.

order to generate a predictive model for classifying the edge nodes into the appropriate categories. Two categories, the Behaving node (B) and Misbehaving node (M) were featured in the primary dataset. The primary predictive model (featuring three variables and the class) can be seen in Figure 3.4.

The classification model's effectiveness is typically measured using either the classification accuracy or the error rate, which can be seen in equations 7 and 8. Using these metrics, the test data is allocated a predicted class by the model. When the test data class results are similar to a models' predicted class, a correct classification is recorded. Otherwise, a misclassification is counted. For test data with *N* data instances, the classification accuracy denotes the proportion of the correctly classified data instances from *N*, whereas the error rate is the proportion of misclassified data instances from *N*. A 93% accuracy of the binary decision tree model was recorded, meaning that centred on the nodes behaviour dataset, it could correctly assign almost 93% of the data instances into the correct categories, misclassifying 16 instances. The reason why many of the misclassifications occurred was due to a behaving/misbehaving overlap in the dropping packet average rate. More specifically, an average packet dropping rate of between 32% - 40% was observed in 75% of the misclassified instances. The rest of the misclassified instances resulted from exceptional cases, such as when there was a high dropping rate caused by congestion rather than BHP flood attacks. It can also be due to uncertain behaviours in sending packets from the edge nodes, including packet delays or bandwidth usage.

The binary classification model produced from the network dataset contains three features. These features helped in the generation of several automated rules that have been extracted from the tree model. After rule-pruning, these rules are described as follows:

1) IF 10-Run-AVG-Drop-Rate < 0.395560763 AND 10-Run-AVG-Bandwidth-Use< 0.8003310674999999

   THEN Class = B (205 cases classified correctly and 6 cases incorrectly)

2) IF 10-Run-AVG-Drop-Rate < 0.395560763 AND 10-Run-AVG-Bandwidth-Use >0.8003310674999999

   THEN Class = M (2 cases classified correctly and 1 case incorrectly)

3) IF 10-Run-AVG-Drop-Rate is between (0.395 & 0.41) AND 10-Run-Delay is between (5.5215E-4 and 5.237 E-4) THEN Class = B (2 cases classified correctly and 2 cases incorrectly)

4) IF 10-Run-AVG-Drop-Rate is between (0.395 & 0.41) AND 10-Run-Delay > 5.237 E-4)

   THEN Class = M (22 cases classified correctly and 3 cases incorrectly)

5) IF 10-Run-AVG-Drop-Rate is between (0.4060 & 0.4101)

   THEN Class = M (4 cases classified correctly and 0 case incorrectly)

6) IF 10-Run-AVG-Drop-Rate > 0.41

   THEN Class = M (147 cases classified correctly and 4 cases incorrectly)

The above shows that rules 5 and 6 can be combined into one rule showing that "IF 10-Run-AVG-Drop-Rate > 0.41 Then Class= M Otherwise Class= B." However, the number of errors when merging this rule increases from four misclassified instances to 10. To remove noisy feature-class correlations, pruning was used to improve the initial decision tree, which produced the following important rules:

- IF 10-Run-AVG-Drop-Rate <= 0.395522 Then Class = B (214 cases classified correctly and 6 cases incorrectly)
- IF 10-Run-AVG-Drop-Rate > 0.395522 Then Class = M (184 cases classified correctly and 10 cases incorrectly)

Just 16 misclassifications occurred within the above rules, equating to a 4.27% error rate using just one feature: 10-Run-AVG-Drop-Rate. While the classification accuracy has been improved using two basic rules from the decision tree algorithm, the first rule of the initial model has not led to a loss of knowledge. Using decision trees and other ML predictive models can help end-users by providing edge nodes with a binary classification, uncovering otherwise-hidden knowledge. This gained knowledge can help automate the classification of edge nodes as well as assisting the Network Administrators and other domain experts to determine the performance of edge nodes themselves.

3.4.3 Results on the Multi-Class Dataset

Next, a more comprehensive model that contained multiple categories is introduced to better reflect the reality of the BHP flood attacks. Following the processing of the training dataset and selection of the features, the decision tree algorithm was applied to the multi-class dataset in order to provide the predictive model for classifying the edge nodes into their correct categories. Four categories, the Misbehaving-Block (Block), Behaving-No Block (No Block), Misbehaving-No Block (M-No Block), and Misbehaving-Wait (M-Wait) were featured in the new dataset.

The decision tree-rule method was applied for data processing in the new multi-class dataset, and seven rules emerged. These rules are displayed as follows:

1. IF 10-Run-AVG-Drop-Rate <= 0.4 Then No Block

   (225 cases classified correctly and 0 cases incorrectly)
2. IF 10-Run-AVG-Drop-Rate <= 0.509 AND 10-Run-AVG-Drop-Rate <= 0.416 Then M-No Block

   (33 cases classified correctly and 3 cases incorrectly)
3. IF 10-Run-AVG-Drop-Rate > 0.515 Then Block

   (58 cases classified correctly and 22 cases incorrectly)
4. IF 10-Run-AVG-Bandwidth-Use <= 0.53 Then M-No Block

(47 cases classified correctly and 9 cases incorrectly)

5. IF 10-Run-Delay > 0.0009 Then M-No Block

   (17 cases classified correctly and 2 cases incorrectly)

6. IF 10-Run-Delay <= 0.0007 AND 10-Run-Delay <= 0.0006 AND 10-Run-AVG-Bandwidth-Use > 0.545 Then M-No Block

   (8 cases classified correctly and 3 cases incorrectly)

7. Otherwise M-Wait (10 cases classified correctly and 4 cases incorrectly)

From the new multi-class model, the packet drop rate variable remains the most critical for preventing BHP flooding attacks, as it appears in the tree model's first three rules. There is no error rate within the first rule while it also encompasses a spread of data instances (i.e. 225), signifying a strong rule when trying to identify behaving and misbehaving edge nodes, which can then be separated further to isolate the edge nodes which may be leading to BHP flooding attacks. Meanwhile, the next rule identified 33 instances correctly versus 3 incorrect ones. Again, this is a significant rule which showed that data was still able to be transmitted, despite the misbehaving edge nodes and therefore was a reliable QoS indicator in terms of the average packet drop rate variable of the nodes. It is clear in the third rule that as the average packet drop rate goes above 51.5% in misbehaving nodes, it shows signs of reserving unused resources as a result of the BHP flood attacks causing the large part of the packet dropping.

This node therefore will be prevented from sending further data. The remaining rules generated in the model (rules 4-7) are useful since two additional variables have been used: average bandwidth used and delay per second. However, these rules are associated with larger error rates and cover a limited number of data instances. For example, the last two rules only cover 25 instances, 7 of which are misclassified.

The main feature contributing to BHP flooding attacks remains the 10-Run-AVG-Drop-Rate with 10-Run-AVG-Bandwidth-Use and 10-Run-Delay following next. The multi-class models had a fair predictive accuracy of 83.66%. Interestingly, the accuracy increases to nearly 87% if the pruning method is used. Moreover, the number of rules shrinks to just three rules in the set, which can be seen in Figure 3.5. Just one variable is used within the newly pruned model: 10-Run-AVG-Drop-Rate. According to Figure 3.5, 225 instances are accurately covered by the first rule, making it appear as an effectively predictive rule. There were 23 misclassified instances in the second rule out of a total of 135, while the highest error rate versus lowest data coverage was found in the third rule.

1. IF 10-Run-AVG-Drop-Rate <= 0.4 Then No Block (225.0)
2. IF 10-Run-AVG-Drop-Rate > 0.4
   AND 10-Run-AVG-Drop-Rate <= 0.509 Then M-No Block (112.0/23.0)
3. IF 10-Run-AVG-Drop-Rate > 0.4
   AND 10-Run-AVG-Drop-Rate > 0.509: Block (61.0/24.0)

Figure 3.5. The multi-class classification model with pruning

To better visualize the performance of the algorithm, confusion matrix (Table 3.6), also known as error matrix, is used [58]. Figure 3.6 shows a decision tree model confusion matrix derived from the multi-class dataset. It examines both the predicted and accurate class values from the classification as shown in Table 3.6, where the different rows represent the actual class instances and the predicted classes are presented in the columns [58]. The performance of the classification model is usually measured using either the error rate (Equation 7) or the classification accuracy (Equations 8).

Table 3.6
Confusion matrix for classification task in ML

		Predicted Class	
		YES	**NO**
Actual Class	**YES**	True Positive (TP)	False Negative (FN)
	NO	False Positive (FP)	True Negative (TN)

$$error\ (\%) = 1 - Accuracy \tag{7}$$

$$Classification\ Accuracy\ (\%) = \frac{|TP+TN|}{TP+TN+|FP+FN|} \tag{8}$$

Where TP (True Positive) represents data instances that were predicted “Yes” and their actual class is “Yes”, the FP (False Positive) represents the data instances that are incorrectly predicted “Yes” and their actual class is “No,” FN (False Negative) denotes data instances that incorrectly predicted as “No” but have actual class “Yes,” and TN (True Negative) denotes data instances that are correctly predicted “No” and their actual class is “No.

Figure 3.6 shows that the behaving edge nodes have been identified by the decision tree method without any errors (224 instances have been correctly predicted "No Block," and they are behaving edge nodes). The misbehaving edge nodes were the ones which posed a problem. More specifically, 24 of the 113 actual misbehaving nodes (not at BHP flood attack levels) were blocked. Meanwhile, 21 of the 51 instances were identified as misbehaving, but were not blocked and instead were predicted as "M-No Block". An explanation for the misclassifications could be due to the high packet drop rates, but not at a significant level for them to be blocked. This means that in some iterations, the edge nodes will have reserved network resources, leaving the other portion of resources unused as a result of the BHP flooding attack.

```
 a    b   c    d  <-- classified as
224  1    0    0 |  a = No Block
 0   89  24    0 |  b = M-No Block
 0   21  30    0 |  c = Block
 0    9   0    0 |  d = M-Wait
```

Figure 3.6. The confusion matrix of the decision tree model derived from the multi-class dataset

Another notable result in the confusion matrix is that all of the data instances that should belong to class "M-Wait", have been misclassified to class "M-No Block". This is due to two reasons. The first reason is that a node gets classified as M-Wait only if there is a competition on reserving the resources between two or more nodes that have high drop rates but not to the point of blocking. In such case, the node with higher drop rate has to wait since it has less priority due to its misbehaviour. In the experiments, we minimized these scenarios having two data channels in the simulator setting since we are more interested in the other three sub-class labels; more specifically, the blocking. The second reason is the overlapping data between the two sub-class labels (M-No Block and M-Wait). While misbehaving behaviour is present within both sub-class labels and very likely has a high drop rate, the rate is not high enough to ensure that the instances are blocked. In this instance therefore, the decision tree failed to identify 9 challenging data instances, and instead misclassified them into a class which had some similar data instances. To overcome this issue, more simulation runs can be performed to have larger data representations for sub-class labels M-Wait in order to allow the learning algorithm to differentiate among the sub-class labels in the resulting tree models.

It is clear from the confusion matrix that the misbehaving edge nodes are the most difficult cases to predict. As a matter of fact, the false positives and true negatives presented in the confusion matrix by the decision tree method proves that the models derived are not overfitted. This is because during the simulation run, random levels of reserving network resources by the sending nodes were ensured. By separating the misbehaving class into three possible sub-class labels, the reality of the BHP flooding attack issue is exposed, creating errors as well as providing more specific classification for the edge nodes. Transmitting data through automatic classification is a key method for ensuring better quality of service (QoS) through reserving resources.

3.5 Related Studies

In recent years, researchers have been drawn to the significant issue of data flow management within computer networks. The problem rested in the inability of packet headers to hold enough information to enable automatic classification, leading to a low accuracy in labelling traffic flow. Previous studies [38, 39] have employed ML methods in order to classify packets and flow within the Internet. Meanwhile, few related studies have used ML techniques within OBS networks [40, 50].

In [40], the authors differentiated between contention and congestion problems in OBS networks by classifying data burst losses. The authors developed a measure called the 'number of bursts between failures' (NBBF), which is designed to accurately identify the types of losses. The method applied both expectation maximisation (EM) algorithm [59] and Hidden Markov Chain (HMC) [60] to a sample of data gathered from burst losses. More specifically, the NBBF at egress nodes between two lost bursts is recorded so that the losses can be categorized, followed by the use of a HMM classification algorithm in order to identify the kind of loss. The studies carried out show that this hybrid ML approach can single out both congestion and contention losses.

The authors of [50] employed a new form of routing mechanism for JET-based OBS networks called 'graphical probabilistic routing model', which identifies less commonly-used links on a hop-by-hop basis through adopting a Bayesian network [61]. The algorithm uses neither FDL nor wavelength converters at the OBS' core switch. The simulation results of the study show that the adaptive routing algorithm suggested is more effective at reducing the Burst Loss Ratio (BLR) in comparison to more fixed methods.

The authors of [35] proposed one of the earliest uses of ML in order to classify Internet data traffic. They used a Naïve Bayes probabilistic method of classification [48]. The early training dataset was built using various traffic flow identifiers such as flow length, port ID, the time elapsed between two consecutive flows as well as other identifiers. Traffic flow was manually assigned by a domain expert while the dataset was prepared. Next, the training dataset underwent the Naïve Bayes algorithm in order to produce a system of classification which would be able to instinctively predict approaching traffic flows. Large volumes of data were tested and displayed a 65% accuracy rate using the Naïve Bayes method, rising to 95% after some adjustments to the dataset, such as the use of feature selection.

The authors of [39] examined the issue of data traffic in order to strengthen network resource management, doing so through permitting the network manager to recognize different data traffic types. More specifically, the authors aimed to determine a key issue in relation to network performance in terms of source, destination, traffic quantities and so on, with recommended actions that the network manager should take. The study involved the collection of data traffic features including byte counts, connection duration, packet size, interarrival statistics amongst as well as others. Next, they applied the EM clustering algorithm to the dataset in order to cluster the traffic flows into a set of groups. The results recorded demonstrated that six key clusters, based on single and bulk transactions, could distinguish between the traffic flows to allow more in depth flow examination. They did not provide information on the categorization of the data flows following the clustering process.

In [41], the authors surveyed the various ML approaches which had been used between 2004 and 2007 to classify and manage IP traffic within different forms of computer networks. They based their research around the existing ML classification methodology, feature selection, model evaluation and type learning. Both supervised and unsupervised learning approaches were examined using them to complete traffic flow classification across various typical computer networks. They examined unsupervised learning using clustering as well as other tested approaches including EM and K-Means. Methods discussed for supervised learning included probabilistic (Naïve Bayes), K-Nearest Neighbour (KNN) [62], and Genetic Algorithm (GA) [63]. Finally, they examined the various evaluation approaches employed in order to judge whether or not the ML approaches were effective.

The ML-based academic studies discussed above have centred on data traffic identification, whereas this study is concerned with a completely different issue – BHP flooding attack. To the best of our knowledge, this study is the first to offer proposals and to develop a decision tree method of classification to provide a solution to the issue of BHP flooding attack in OBS networks. Since previous studies have not used ML as a way to block misbehaving edge nodes which send DBs in OBS networks, we believe a solution is needed in order to address this critical issue in the initial phases of BHP flooding attacks.

3.6 Summary

Although serving as one of the most promising optical switching technology for optical networks, OBS network is a technology that has not matured enough for it to be implemented and deployed. The basic idea of OBS relies mainly on the concept of sending BHP in advance to reserve the resources as well as setting the path for the packets that are aggregated into data bursts at the edge nodes. Compromising an edge node by an attacker and sending BHPs in large volumes without sending corresponding data bursts can lead to a serious security issue called BHP flooding attack. Network performance as well as QoS can be severely affected by BHP flooding attacks, which is why it is important to develop new classification strategies for identifying edge nodes with misbehaving sending behaviours at the initial stages.

This research proposes to use ML to develop a new architecture based on decision tree that will accurately and effectively classify edge nodes of OBS networks using straightforward If-Then rules. These rules are discovered from real data related to multiple performance indicators (variables) recorded by a simulator. Initially, the proposed rules are able to classify the sending nodes into Behaving and Misbehaving with 93% accuracy. The models that are more realistic are then produced after dividing the Misbehaving class into four sub-class labels in order to further classify this type of node based on data priority. Experimentations using multiple edge nodes on a large dataset collected from a number of simulation runs revealed that the rules generated by the classification algorithm are highly effective in preventing misbehaving nodes from sending data, and therefore able to counter the BHP flooding attack problem. More specifically, the tree models generated from the binary dataset were able to classify edge nodes with 93% accuracy. In addition, the modified decision tree models for the misbehaving nodes (multi-class dataset), had an 87% accuracy when splitting the Misbehaving class into four sub-class labels: Misbehaving-Block (Block), Behaving-No Block (No Block), Misbehaving-No Block (M-No Block), and Misbehaving-Wait (M-Wait). This breakdown meant that we could develop a fine-grained strategy for data priority for the edge nodes using the information gathered from the classification models. In the near future, we intend do an experimental study to evaluate several ML algorithms in order to seek the one which has the best performance for the BHP flooding attack problem. Also, we hope to further improve the classification architecture through being able to add further volumes of nodes, while also building the new learning algorithm into the simulator.

CHAPTER 4

DETECTING BHP-FLOODING ATTACK IN OBS NETWORK: A MACHINE LEARNING PROSPECTIVE

4.1 Background

An optical network (ON) is a known medium for data transmission, adopting an Optical Burst Switching (OBS) network for the Internet [64]. In an OBS network, burst header packets (BHPs) are transmitted in advance to allocate enough resources prior to sending the actual data bursts (DBs), ensuring network management and Quality of Service (QoS). This enables attackers to flood the network with malicious BHPs, reserving the network resources without proper use. In this case, malicious BHPs continue to reserve the network resources without sending the actual DBs, hindering the performance of the OBS network, in some cases causing Denial of Service (DoS) [65]. Therefore, it is essential to prevent BHP flooding attacks in OBS networks by blocking misbehaving ingress nodes that continuously transmit malicious BHPs, and preventing the legitimate BHPs from reserving the required resources at the intermediate core switch.

Limited research works detecting BHP flooding attacks in OBS networks exist, e.g. [9, 10, 65]. In [9], a data flow classification architecture was implemented at the optical layer to combat BHP flooding attacks. This method distinguishes between the offset time inside the BHP and the recorded delay between this BHP and its related DB. [10] utilized optical code words to single out malicious BHPs sent by ingress nodes in an OBS network. The authors used statistical data analysis related to packets sent and dropped to detect the possibility of BHP flooding attacks. [65] developed a new security model to be implemented into the OBS core switch to prevent BHP flooding attacks. The countermeasure security model can detect malicious ingress nodes based on their behavior, alongside the amount of reserved resources that are not being utilized, and block any malicious ingress nodes until the threat ceases. The reported results using the NCTUns network simulator showed that the security method of [65] was able to effectively differentiate among legitimate and malicious ingress nodes, thus maintaining good network performance.

Despite the few recent studies on BHP flooding attacks, the detection rate is still low. Further, the entire process relies on the domain experts' knowledge and experience. Therefore, there is a need for a more efficient detection system that can engage the core switch in OBS network, thus identifying misbehaving ingress nodes in an automated manner as early as possible. One promising approach to accomplish this is the Machine Learning (ML) method. This uses the historical performance of source nodes during data transmission to construct classification models known as classifiers. The classifiers then predict whether the source nodes are sending legitimate BHPs or not, and filter out malicious BHPs that might cause flooding attacks. The outcomes of the ML method will enable security administrators to quickly block misbehaving ingress nodes until they change their behaviors. (It is the firm belief of the authors) that classifying ingress nodes using ML to counter BHP flooding attacks is yet to be studied within an OBS network.

This study examines the performance of ML methods to counter the risks associated with BHP flood attacks in OBS networks. The problem studied is a typical predictive task in classification, in which different variables linked with ingress nodes' performances are collected whilst sending BHPs (in simulation runs), and are saved in a training dataset. Examples of variables are not limited to iteration number, but can include the sending node label, packets sent, packets dropped, delay time, and so on. More details on the complete dataset of variables can be found at [66], and are briefly explained in Section 4.3. The ML role involves processing the different variables in the dataset to obtain concealed information useful for prediction (classifier). This classifier is then used to categorize ingress nodes in certain future scenarios as accurately as possible, improving the manual classification which indeed requires care, time and experience.

The ultimate aim of this study is to examine the applicability of ML to the problem of BHP flooding attacks in OBS networks. To achieve this, we extensively investigated various ML techniques that adopt different learning approaches to the research problem considered. We seek to identify the most relevant ML technique(s) for solving the issue of BHP flooding attacks, in addition to revealing the reasons behind the relevancy. Thus, we endeavor to answer the following research questions:

- Can ML be used as a BHP detection approach in an OBS network?
- Which ML techniques improve detection rate and time performance?
- Which ML technique is more suitable to end-users, and why?

The ML approaches considered in this study are Logistic Regression, Naïve Bayes, RIDOR, SVM-SMO, NN-MultilayerPerceptron, C4.5, AdaBoost, and Bagging [45, 48, 67, 68, 69, 70, 71, 72]. The diversity of the ML approaches strengthens the confidence in the results, hence our recommendations (see Sections 4.3, 4.4 & 4.6). The performance of the wide range of ML techniques has been measured using different metrics, against a published dataset at UCI (University of California-Irvine) repository [73]. Specifically, we utilized classification accuracy, classifiers' construction time in milliseconds (ms), precision, recall, and the harmonic mean among other measures (Section 4.3 & 4.4 give further details) [74].

4.2 The Considered Machine Learning Techniques

Since the BHP flooding attack is a typical prediction problem, classification methods in ML seems appropriate to identify malicious and legitimate edge nodes. In classification problems, a model called the classifier is constructed from historical labelled dataset(s). The learned classifier is then employed to forecast the class label in datasets that are unlabeled, known as test datasets [43, 75]. The quality of the classifiers extracted by ML methods rely primarily on the classification accuracy, as well as other known evaluation metrics such as recall, precision, and harmonic mean [38]. In addition, classifiers formed after data processing differ based on the ML techniques used. For instance, rule induction classifiers contain rules, and Naïve Bayes classifiers hold just class memberships in a probability format [76]. In this section, we highlight eight different ML techniques that generate different type of classifiers. Specifically, we investigate classifiers extracted by Logistic Regression, Probabilistic-Naïve Bayes, Rule Induction- RIDOR, Support Vector Machine - Sequential Minimal Optimization (SVM-SMO), Neural Network-NN-MultilayerPerceptron, Decision Tree-C4.5, Boosting-AdaBoost, and Bagging [45, 48, 67, 68, 69, 70, 71, 72]. The choices of these techniques are mainly based on the following facts:

1) Different learning methodologies are employed for data processing
2) Different classifier formats are presented to the end-user
3) Applicability and usage in previous domains in particular computer networks, computer security among others, i.e. [43, 75, 77, 47, 78, 79].

Steps of machine learning are shown in Figure 4.1, and are briefly explained below.

1) Data pre-processing (Optional): In this step, any noise related to the training dataset, such as missing values, duplications, and feature selection are completed. The output of this step is a processed dataset.
2) Training: In this step, the ML technique processes the data for knowledge or patterns. In classification techniques, the classifier is constructed in this step.
3) Evaluation: The classifier is evaluated on a test dataset to measure its effectiveness. This step results in different evaluation metrics.
4) Pattern Visualization (Optional): In this step, the outcomes as well as its quality measures are presented to the end-user in a non-technical manner to ease decision making.

The next section briefly summarizes known ML learning approaches that this study investigates to be utilized in solving the BHP flood attacks problem.

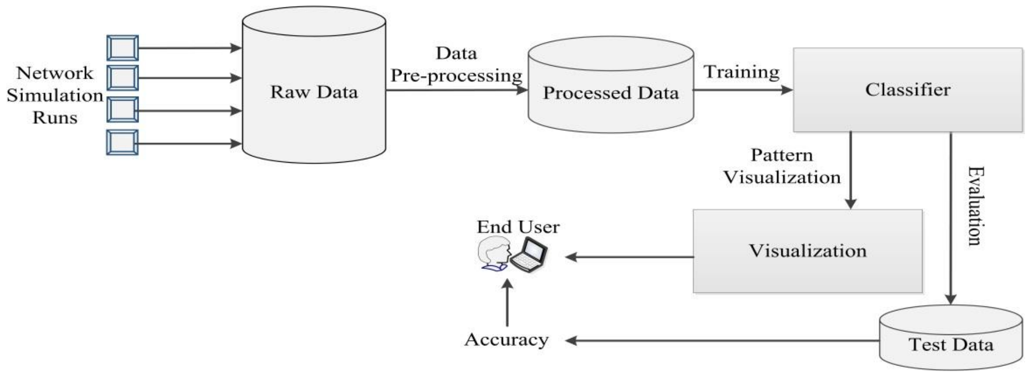


Figure 4.1. Steps of ML classification technique

4.2.1 Rule Induction - RIDOR

Rule induction is a classification approach that normally extracts If-Then rules in a sequential fashion [76]. Typically, a rule induction technique divides the input dataset into splits according to the available class values. Then, for each class split, the induction technique learns and derives If-Then rules based on mathematical metrics, such as a rule's expected accuracy (Equation (9)). Data examples in a split, for instance A, are positive examples for the class of A, and are considered negative examples for the other class labels in the other data splits. For a data split, the induction technique builds an empty rule, and then adds items to the rule's antecedent (left hand side/body) until the rule meets a termination condition. When this occurs, the rule is generated, and all data examples that the rule classifies are discarded. Then, the induction technique learns the next rule from the same split until the data split becomes empty. Following this, the induction technique moves to the next data split until all data splits become empty, or no more rules with acceptable accuracies can be discovered [78]. Common rule induction techniques are RIDOR [67] and RIPPER [80].

RIDOR, for example, derives a default rule class, and then learns all the exceptions for that default rule using Incremental Reduced Error Pruning (IREP) [81], a learning method. An exception is a rule able to forecast the class label other than the default class. IREP eliminated one exhausting phase of an earlier rule induction technique called Reduced Error Pruning (REP), saving substantial training time. In RIDOR, the training dataset is divided into pruning (1/3) and growing (2/3) subsets. Then, RIDOR builds incremental rules one at a time. When a rule is about to be evaluated for possible pruning, its training data examples in the pruning and growing subsets are removed, and the rule gets extracted. During pruning, RIDOR considers deleting items from the rule's body and terminates the pruning phase when removing an item from a rule cannot improve the rule's accuracy.

$$r's\ Expected\ Accuarcy = (P/T) \tag{9}$$

where P = the # of positive instances covered by a rule r (both antecedent and consequent)

T= the total # of instances covered by r's antecedent

4.2.2 Decision Tree Rules – C4.5

C4.5 [70] is a decision technique utilizing Entropy and Information Gain (IG) (Equations 10-11 below) to construct tree based classifiers for prediction. To build a classifier, initially, the IGs for all variables in the training dataset, other than the class variable, are computed, and a root with the highest IG is selected. The IG is calculated based on how informative a data variable is in dividing the examples in the training dataset with respect to the class label. When a root is chosen, the algorithm excludes it in the next iteration and repeatedly calculates the IGs for the other available variables, until the tree cannot be built any further or the remaining data examples are linked with just a single class. In the formed decision tree, a path from the root node to any leaf denotes a rule, and the leaf denotes a decision (class label).

$$Gain\ (T, f) = Entropy\ (T) - \sum ((|\ T_f\ |\ /\ |\ T\ |)\ *\ \text{Entropy}\ \text{T}_f)) \qquad (10)$$

$$Entropy\ (T) = \ \sum -P_c\ \log_2\ P_c \qquad (11)$$

where P_c = Probability that T belongs to class l, T_f = Subset of T for which feature F has value $f_{a.}$, $|T_f|$ = Number of examples in T_f, and $|T|$ = Size of T.

4.2.3 Probabilistic Methods- Naive Bayes

In classification, when a test example requires a class label, an efficient way to classify the test example is to use NB technique, which is based on Bayes theorem. NB calculates the probability of the test example with respect to each class label using prior knowledge of the test example's variables, and their appearances with each class in the training dataset. The frequency of each variable and the class in the training dataset is obtained in addition to the frequency of each class label. Then, all probabilities are multiplied by each other and the test data example is given the class with the highest probability score (Equation 12 below). NB predicates independent assumptions for variables and the class, which is not necessarily true in real application data [82]. Nevertheless, this probabilistic technique is highly efficient in deriving classifiers in contrast to other ML techniques [83].

Given a test data example as a vector A = (a_1, a_2, …, a_m) where each a is a variable, using NB, the conditional probability can be obtained as:

$$P(C_n|A) = \frac{P(C_n).(A|C_n)}{P(A)} \tag{12}$$

The test data example will be given the class with the greatest probability $P(C_n|A)$.

4.2.4 Boosting and Bagging

Bagging and Boosting learning approaches use the training dataset in multiple trails to produce numbers of weak classifiers, that are then merged to form a global classifier [84]. The idea is to utilize both the weak and the strong classifiers in predicting the class label of test data.

In Boosting, a weak classifier is simply built from the input dataset, and then utilized to assign class labels to the training data examples. The next weak classifier is built from the training data, and training examples that have not been correctly classified by the previous weak classifier are selected more often to be re-classified by the current weak classifier, improving the model's predictive accuracy. The below steps clarify how Boosting algorithms, such as AdaBoost [71, 85], work:

1) Select a base ML algorithm for learning such as a rule based classifier
2) The base algorithm learns a weak classifier from the training dataset and assigns an equal weight for each training data example
3) When there are misclassification cases (incorrectly classified data examples), we re-apply the base ML algorithm, and pay more attention to the unclassified data examples to improve the predictive performance
4) Repeat steps 2-3 until the intended accuracy has been derived

5) Merge the weak classifiers to produce a strong classifier
6) When a test data needs to be classified, use a voting mechanism to assign the class label from the strong classifier and the weak classifiers.

In the Bagging classification approach [72], sample data examples are generated for each trail (iteration) from the original training dataset (often with the same size of the original training dataset). Then, a base ML algorithm is used to generate a classifier from the sample, and the process is repeated a number of times. Finally, all derived classifiers are aggregated together to form a global (strong) classifier. When test data is about to be classified in the Bagging approach, the class is assigned based on a voting mechanism using both the global and weak classifiers, similar to the Boosting approach. The difference between Bagging and Boosting approaches is that in Bagging, when the data sample is produced from the training dataset, the resembling process is not reliant on the performance of any previously derived classifiers, as it is in Boosting.

4.2.5 ANN

An Artificial Neural Network (ANN) consists of interconnected neurons that transform a set of input examples into desired output (class) without having to reveal the transformation details [47]. The ANN advantage comes from choosing the right numbers of the hidden neurons, and the results often rely on the input variables features and weights associated with their interconnections. Nevertheless, determining the numbers of hidden neurons and other important thresholds prior to data processing is fundamental to the quality of the outcome in ANN algorithms. Questions such as, what is the right number of hidden layers, epoch size, and acceptable learning rate, among others, need to be set by a domain expert in order to generate fair and acceptable classifiers. Overall, researchers still utilize train-and-error methods to tune the aforementioned parameters since there is no clear methodology for setting these up [86]. ANNs utilize sigmoid functions during constructing classifiers, in which weights are repeatedly amended to come up with the desired error rate that the domain expert had set prior to the beginning of the learning phase.

4.2.6 SVM

SVM is a classification approach proposed to enhance the predictive performance of classic classification techniques [87]. This approach depends on hyperplanes, which divide data examples based on class memberships. The SVM learning mechanism sorts data examples using mathematical functions known as kernels. A kernel computes the similarity of data examples using the available classes in the training dataset [88]. Often, kernels are determined by SVM experts, and then utilized for the classification phase.

SMO trains SVM on a large quadratic programming (QP) optimization problem [68]. SMO decomposes the QP problem into a number of smaller problems, and then solves them by avoiding a numerical QP inner loop. The computing resource needed in the particular memory for SMO is linear in the training dataset size, which permits the SMO algorithm to process larger input datasets. Reported experimental results revealed that SVM algorithms such as SMO generate high predictive classification systems in multiple domains, especially text categorization rather than probabilistic, and induction [87, 89].

4.2.7 Logistic Regression

When the target variable in classification dataset is continuous, (numeric) classic ML methods such as rule induction, decision trees, and covering are not able to produce a classifier. Linear regression can solve such a problem by offering methods describing the training dataset in the context of a predictive task, by revealing the relationships between independent variables and the class variable (dependent). Unlike linear regression, in Logistic regression, the class variable is not continuous, but is rather categorical (predefined possible values) [45, 90].

Logistic regression is formulated based on Equation 5 below:

$$p = \frac{e^{a+bX}}{1+e^{a+bX}} \tag{13}$$

where p = probability of Y = 1

e = base of the natural logarithm (around 2.718)

a and b = inputs parameters of the logistic model

Due to the curvilinear correlation between p and X, b in (Equation 13) is different than b in a typical linear regression model. We can linearize the logistic regression model by converting the dependent variable from a likelihood (probability) to a logit, as shown in Equation 14.

$$ln\left(\frac{p}{1-p}\right) = a + bX \tag{14}$$

$$ln\left(\frac{p}{1-p}\right) = \text{logit (log odds) of } Y = 1 \tag{15}$$

where

a and b = inputs of the logistic model

The logit (Equation 15) is often named a *link* function, because it gives a linear conversion of the logistic regression model.

4.3 Experimental Setting and Data

This section investigates the ML algorithm's performance on a simulated dataset generated by the NCTUns simulator for over a thousand runs on NSFNET topology [32]. The aim is to enhance the performance of UDP on OBS networks by automatically detecting misbehaving ingress nodes that may cause

BHP flood attacks, helping to manage the network's resources. By employing NSFNET topology, we can insert and simulate with any number of nodes, in order to investigate different scenarios. The simulation parameters for the OBS network configuration are displayed in Table 2.1. The simulator may need to run for 15 to 45 minutes to obtain the result for just "one second" depending on the load assigned.

All experiments have been conducted utilizing a recently developed simulated dataset that belongs to the authors. This can be obtained from the UCI Machine Learning Repository (University of California-Irvine) dataset [73]. This contains twenty-two variables related to flooding attacks, including the class variable. The variables collected during the NCTUns simulator directly associate with the OBS network's performance. The dataset size consists of 1075 examples, and each example denotes one iteration (a simulation run) in which an ingress node is sending data over the OBS network. Different scenarios, including BHP flood attacks without pre-setting values, have been generated during the simulation, ensuring that ingress nodes have random levels of BHP flood attacks. This is essential to show situations of occupied network resources without proper utilization and with different occupancies. During the simulated runs, two ingress nodes were used. In addition, for each simulation run, the bandwidth of the node was initially assigned to 100 Mbps, and then incrementally increased to 200 Mbps, 300 Mbps, 400 Mbps, and so forth, until the maximum bandwidth, i.e. 1000 Mbps, is reached.

For illustration purposes, Table 4.1 depicts eight variables with five iterations exhibiting how the ingress nodes for every simulation run were used to transmit data. The table displays iterations that demonstrated behaving and misbehaving edge nodes. The dataset contains four possible class labels (Block, No Block, Misbehaving-No-Block, Misbehaving-Wait), and thus the problem is a

Table 4.1
Sample of five iterations of the processed multi-class training dataset

Iteration	Node#	Drop-Rate	Bandwidth-Use	Delay	Node Status	BHP Flood	Class
1	3	0.11	0.793	0.00009	B	0.000	No Block
1	9	0.22	0.703	0.00009	B	0.008	M-No Block
2	3	0.45	0.559	0.0005	M	0.369	M-No Block
2	9	0.42	0.589	0.0007	M	0.3697	M-Wait
3	3	0.466	0.543	0.0005	M	0.2887	M-Wait
3	9	0.416	0.593	0.0006	M	0.2541	M-No Block
4	3	0.476	0.532	0.0006	M	0.3621	M-Wait
4	9	0.415	0.594	0.0008	M	0.3112	M-No Block
5	3	0.482	0.526	0.0005	M	0.4337	Block
5	9	0.414	0.596	0.0008	M	0.3848	M-No Block

multi-class classification. In Table 4.1, at iteration #1, ingress node 3 was permitted to send data, since it was classified as a behaving node. Ingress node 9, associated with a low BHP flooding rate, was slightly misbehaving, yet because of its low packet dropping rate, it was not blocked. However, at iteration #5, ingress node 3 was blocked, since this node was causing high BHP flooding, its BHPs reserving bandwidth without utilization. At the same iteration, despite node 9 misbehaving, it was still permitted to send data (misbehaving but no block). A trickier scenario is illustrated at iteration #4, in which both ingress nodes are misbehaving, yet are not reaching a BHP flooding attack. Therefore, node 3, due to its higher BHP flood rate, delays until node 9 transmits its data.

The Waikato Environment for Knowledge Analysis (WEKA) tool was adopted to process the dataset using ML [56]. This tool is a Java-based open source, containing various methods related to ML, data mining, visulization, data filtering, and variable selection among others. For all considered ML algorithms, a 10-fold cross validation (10 fold-CV) method was employed during the training phase [56]. 10 fold-CV is a common testing method in ML that ensures the input dataset splits into 10 folds. The algorithm is then trained on 9 folds, and evaluated against the remaining fold to generate the error rate. This procedure is repeated ten times, and all error rates are averaged to show the overall performance of the learning algorithm. The machine used to run all experiments is Intel® Xeon with 3.72 GHz 2 processors.

A number of ML algorithms have been selected to counter the risk of BHP flood attacks by detecting misbehaving ingress nodes. In particular, Simple Logistic Regression, Naïve Bayes, RIDOR, SVM-Sequential Minimal Optimization (SVM-SMO), NN-MultilayerPerceptron, C4.5, AdaBoost, and Bagging [45, 48, 67, 68, 69, 70, 71, 72]. We would like to evaluate the classification systems' predictive accuracies derived from the aforementioned ML algorithms on the BHP flood attack problem. The main metrics used in the ML algorithms' comparisons are:

1) Classification accuracy in %
2) True Positives (TPs) and False Positives (FPs)
3) *P*recision, *R*ecall and Harmonic Mean (*F*-measure)
4) Training time measured in milliseconds (ms) to build the classifiers
5) Classifiers content for the rule induction, Bagging and tree based algorithms

These evaluation measures mathematical descriptions are given below:

$$P = \frac{TP}{TP+FP} \tag{16}$$

$$R = \frac{TP}{TP+FN} \tag{17}$$

$$F1 = 2 \cdot \frac{P \, . \, R}{P + R} \tag{18}$$

$$Accuracy = \frac{TP+TN}{TP+FP+TN+FN} \tag{19}$$

where *TP* is the number of data examples correctly classified by class A, *TN* is the number of data examples correctly classified by class -A, *FP* is the number of A's examples incorrectly classified as -A, and *FN* is the number of -A examples incorrectly classified as A.

Prior to running the ML learning algorithms against the BHP flooding attacks dataset, we pre-processed the dataset using Correlation Features Sets (CFS) to determine the most influential features [57]. CFS is a well-known feature selection method which heuristically examines the correlation of each feature with the class label in order to discard any redundant or low correlated features. After running the CFS on the initial dataset, three features (Drop-Rate, Bandwidth-Use, BHP-Flood) were identified to be more effective to combat the BHP flood attack problem. Hence, we will utilize these features during the training phase for the classifiers.

4.4 Results Analysis

Figure 4.2 highlights the classification accuracies derived by the ML classifiers from the dataset. It is clear from the figure that the Bagging, rule induction (RIDOR), and decision tree (C4.5) classifiers have higher prediction rates than that of the remaining classifiers. Noticeably, the C4.5 algorithm outperformed the remaining algorithms when it comes to predictive accuracy. To be exact, its prediction accuracy is 4.66%, 14.52%, 20.84%, 1.68%, 26.42%, 39.07%, 18.05% higher than those of RIDOR, Naïve Bayes, Simple Logistic Regression, Bagging, SVM-SMO, AdaBoost, and NN- MultilayerPerceptron, respectively. The superiority of C4.5 may be due to the intensive backward and forward pruning implemented after constructing the tree. C4.5 trims sub-trees that lead to larger errors, replacing them with more accurate leaves, resulting in concise, yet highly predictive, classifiers. In addition, the C4.5 algorithm triggers an implicit discretization procedure based on Entropy, converting continuous variables into discrete ones prior to the training phase. This ensures small intervals for each continuous attribute, easing the data processing, and ensuring its efficiency. Finally, Bagging and RIROD classifiers seem competitive in the decision tree, both algorithms using effective pruning procedures to cut down the number of rules produced.

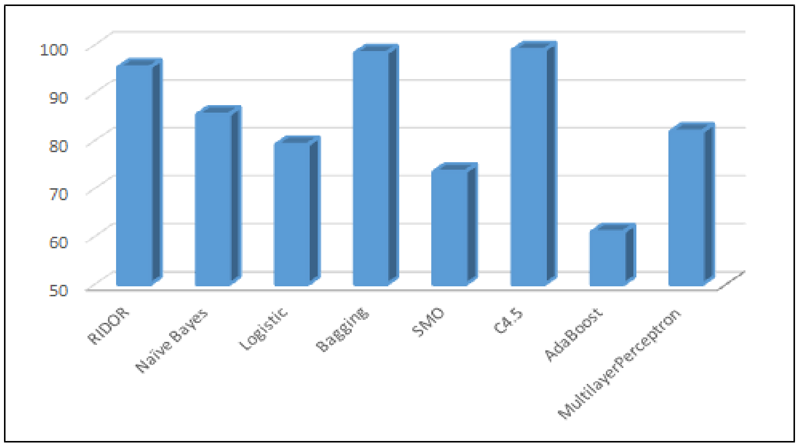


Figure 4.2 Classification accuracies in % derived by the ML algorithms

Figure 4.3 displays the classifiers' sizes for the top three predictive classifiers (C4.5, Bagging, RIDOR). It is clear from the figure that Bagging derives larger classifiers compared to both RIDOR and C4.5 algorithms. This is due to the generation of multiple local classifiers, and the integration step forming a final tree structure, which may lead to many branches and leaves. The classifier presented by RIDOR is the least predictive among those of the three algorithms, yet it contains a concise set of rules. For the user's perspective, a more concise set of rules could make it easier for network administrators to understand and manually control the BHP flooding attack problem. C4.5, on the other hand, offers moderate-sized classifiers that have superiority in classification accuracy over RIDOR and Bagging respectively. In fact, C4.5 covered more training examples than RIDOR, discovering more rules that may contribute to the increase in predictive performance.

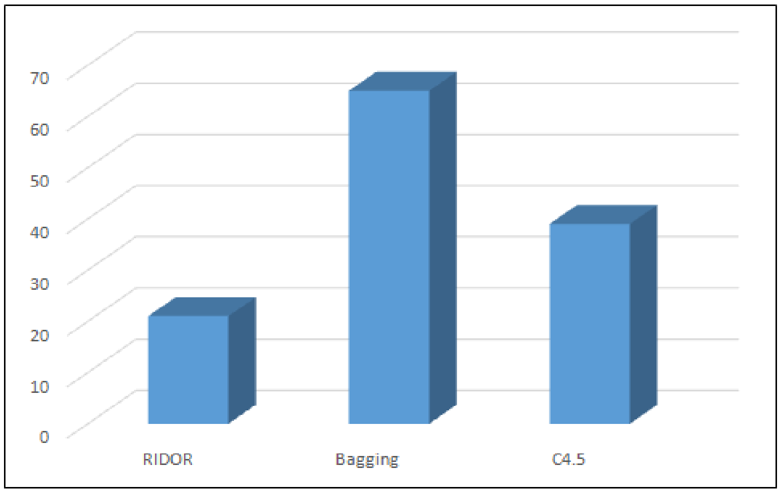


Figure 4.3 The classifiers' sizes of RIDOR, Bagging and C4.5 algorithms

Figures 4.4a - 4.4d show the true positives (TPs), false positives (FPs), true negatives (TNs) and false negatives (FNs) respectively for the considered algorithms on the BHP flood attack dataset. The TPs and TNs are consistent with the classification accuracy rates derived beforehand, in which C4.5, Bagging and

RIDOR achieved higher TPs than that of the remaining algorithms. For example, RIDOR correctly classified “Block”, “No Block” and “M- No Block” class labels without any error. However, for the hard-to-detect cases, i.e. the ones which

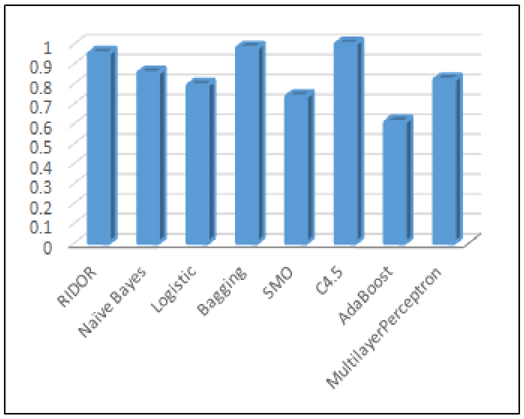


Figure 4.4a The TPs of the ML algorithms

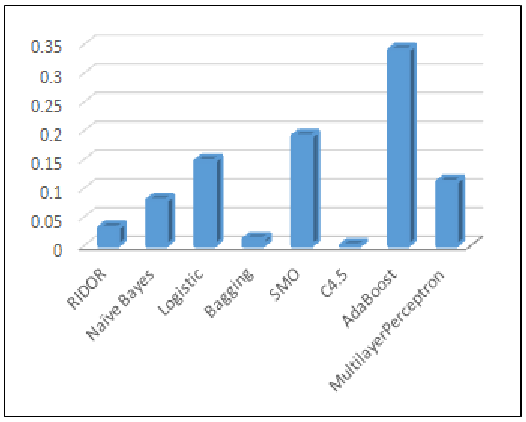


Figure 4.4b The FPs of the ML algorithms

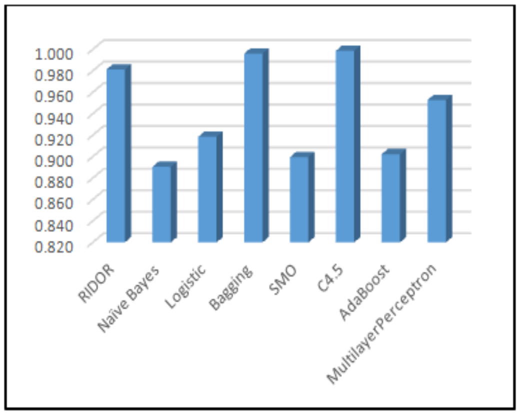


Figure 4.4c The TNs of the ML algorithms

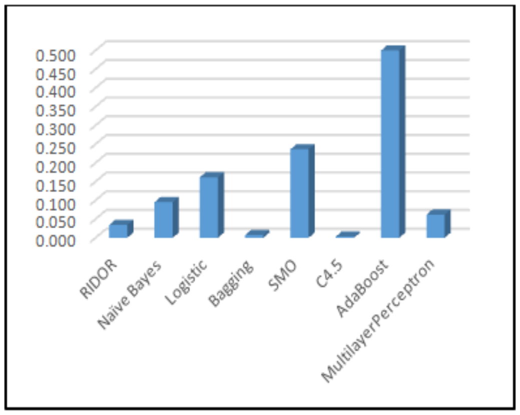


Figure 4.4d The FNs of the ML algorithms

belong to the “M-Wait” class, 28 instances have been misclassified by RIDOR as the “M=No Block” label. For the TNs results, AdaBoost algorithm seems have the rates because it was unable to clearly differentiate among the four class labels in particular M-Wait, which its instances have been completely misclassified to M-No Block class label.

The results of the TPs, TNs, FNs and FPs show that “Block” and “No Block” cases are easy to detect by the ML algorithms, but cases that belong to class labels “M-Wait” and “M-No Block” are harder to be detected, due to overlaps between these two class labels. To be precise, in terms of FPs, the three least performed algorithms (AdaBoost, SVM-SMO, Logistic) are associated with 300, 204, and 254 misclassifications respectively. These figures clearly reveal the reasons behind the low predictive rates of these three algorithms in detecting difficult-to-classify cases of “M-Wait” and “M-No Block”. To overcome this issue of overlapping between class labels, more data cases covering “M-No Block” and “M-Wait” are needed, so the ML algorithms can further distinguish between them during the learning phase. This is due to the fact that the misbehaving nodes are further decomposed in the dataset into three sub-class labels, in order to reflect the true nature of the problem and reduce overfitting during the learning phase. Moreover, and in terms of FNs, decision tree and Bagging algorithms consistently derived good results when compared with the remaining algorithms. To be exact, Bagging algorithm only wrongly classified 11 instances 8 of which belong to the hard to classify class M-Wait. Typically, we do not desire to end up with a binary classification problem in which the ML algorithm decides whether the ingress node is behaving or misbehaving. However, we do aim to understand to which degree the node is misbehaving, and if two nodes are misbehaving, which may be allowed to transmit data, and which should delay in using their flooding or network utilization rates. Therefore, it was necessary to further split the misbehaving class into multiple class labels during the data collection phase.

Figure 4.5 shows three more types of measures: precision, recall, and F-measure. The precision results displayed in Figure 4.5 shows a consistency with classification accuracy rates, and highlights that malicious ingress nodes are harder to be detected than behaving ingress nodes, at least for the dataset and algorithms used. Usually, high precision rates, such as in C4.5, RIDOR and Bagging, relate to their low FPs. C4.5 achieved the largest precision and AdaBoost the lowest. In the precision results, seven out of eight algorithms have consistent results when compared to their accuracies, except for the AdaBoost algorithm. The precision of AdaBoost declined significantly to 0.397 (39%) due to a large number of FP cases, as shown in Figure 4.5. Precision shows the number of correctly classified cases from all that have been classified. On the other hand, recall results in the same figure denotes the number of correctly classified cases in all cases intended to be correctly classified. In the recall results, all the ML algorithms have consistent results when compared to their predictive accuracies.

To have a clearer insight into precision and recall alongside one another, we generated the scores when using the F1 measure. The F1 score takes the weighted average of recall and precision (false negatives and false positives) into consideration, especially when involving data such as our four unevenly distributed class labels. In our study, we can observe that C4.5, RIDOR and Bagging still generate highly competitive F1 scores compared to the remaining considered algorithms on the BHP flood attack dataset.

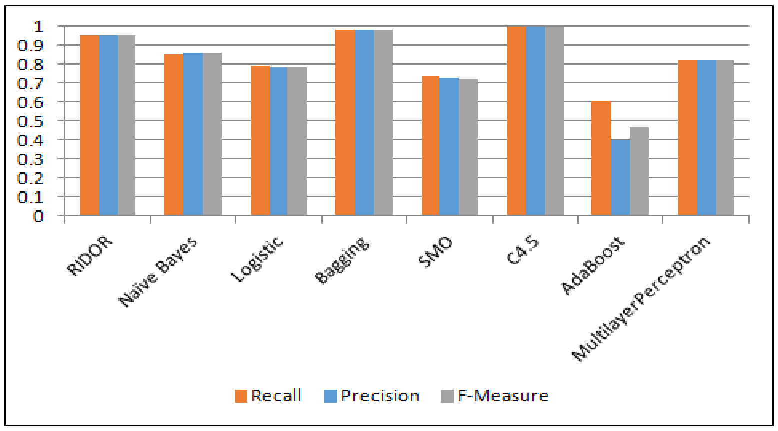


Figure 4.5 The Precision, Recall and F1 scores of the ML algorithms

Lastly, Figure 4.6 depicts the runtime in millisecond (ms) taken from the ML algorithms in constructing the classifiers. Here, the fastest algorithms were Naïve Bayes and C4.5. Naive Bayes uses simple likelihood calculations for all variables in the test dataset using their frequencies in the training dataset, hence no rule learning being involved. Alternately, the C4.5 algorithm employs fast learning based on computing Entropy for the variables in the training dataset to build tree based classifiers. Hence, these two algorithms are quite efficient in building predictive classifiers in contrast to alternative ML algorithms. The MultilayerPerceptron NN algorithm was the slowest algorithm in building the classifier due to the exhaustive search this algorithm employs, which is based on pre-setting the desired expected error achieved. This often necessitates repetitive training dataset scans.

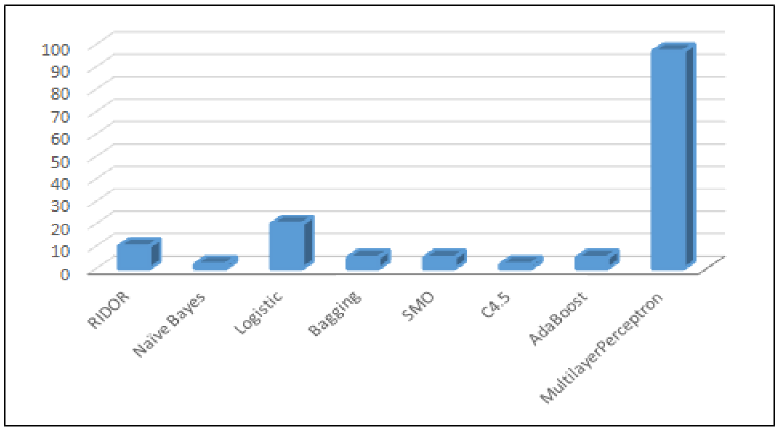


Figure 4.6 The time in ms needed to build the classifiers of the ML algorithms

4.5 Studies Related to Application of Machine Learning in Detection and Classification Tasks

Despite the scarcity of literature, this section highlights these studies and others related to primarily utilizing ML in different types of computer networks [40, 91, 92, 93, 35, 41].

[40] investigated the problems of BHP flood attacks in OBS networks to differentiate the types of data bursts, i.e. congestion or contention. A new metric named "number of bursts between failures" (NBBF) was proposed to detect which type of data bursts losses occur. In the process of classifying these data bursts, the authors applied two methods: unsupervised expectation maximization (EM) and a supervised Hidden Markov Chain (HMC). Reported

results showed that when both methods are integrated, the accuracy of distinguishing among types of bursts losses is increased.

[91] investigated the Distributed Denial-of-Service (DDoS) flood attacks on the transport and application layers, and developed a detection mechanism that analyzes the traffic according to types of packets, packet arrival rate and server capacity. The detection mechanism relies on recording and monitoring information related to address pair (source and destination), the type of packet, the port addresses of the source and destination among others. The key to success of [91]'s method is the predefined setting value of the server capacity. No experiments have been conducted to reveal the pros and cons of the detection method of DDoS flood attacks.

[92] investigated the problems of reducing flood attacks and other service attacks in computer networks using ML. These types of attacks normally belong to DDoS flooding attacks, and other risk that impair Internet security. The aim was to identify the misbehaving sources (nodes) in order to block their messages from their intended destinations. In the learning model proposed, elements of the network share behavior information about the network's performance, so the classifier may amend or enhance the model's behavior by blocking potentially detrimental messages. Reported experimental results revealed a 95% detection rate using a probabilistic classifier.

[93] reviewed different learning mechanisms utilized to detect DDoS flooding attacks, in particular, SYN flooding. This type of flooding attack harms the network performance: when packets flood the network, many users may suffer server access delays. In some cases, the server shuts down entirely from SYN flooding attacks. The authors of [93] critically analyzed different approaches related to ML, statistical analysis, and router based among others.

[35] adopted the Naïve Bayes (NB) probabilistic classification algorithm [48] to detect the type of Internet traffic. Before applying NB, features related to traffic flow such as port identification, elapsed time between two consecutive flows, and the flow length among others, were collected. The type of traffic flow variable was assigned by a domain expert in the dataset, and NB was applied to generate probabilistic classification systems to predict the traffic flow variable. The classification system derived by NB shows low predictive rates, but when the authors utilized feature selection methods prior to the training phase, the accuracy rate of the classification systems was improved.

The IP traffic classification problem was studied in the context of ML by [41]. The authors surveyed and compared the performance of supervised and unsupervised ML algorithms, and highlighted the role of feature assessment in pre-processing the IP traffic dataset. Results showed that NB, EM and decision tree algorithms often produce consistent results, with high classification accuracy for the IP Internet traffic problem. Moreover, a number of recommendations have been highlighted based on the survey, such as:

1) ML algorithms generate different results for the IP traffic problem because of the different learning mechanisms they employ in deriving the classification systems. Hence, hybrid learning seems appropriate for future investigation
2) Different requirements are sought by ML algorithms because learning environments differ from one algorithm to another, as well as configurations
3) It is essential to investigate real time learning, at least for the IP Internet traffic classification problem, in which the ML will, while in progress, derive the classifiers rather than using static datasets
4) Feature selection methods can be useful in some Internet application problems such as IP Internet traffic classification

The majority of recent research contends that utilizing ML techniques in computer networks relates to DDoS flood attacks using primarily adaptive distributed mechanisms, while other studies investigated data traffic analysis. This study investigates an entirely new issue – BHP flood attacks in OBS networks. We believe that ML has not yet been adopted to develop predictive models to counter BHP flood attacks in OBS networks.

4.6 Summary

In spite of the many benefits of an OBS network, such as bandwidth efficiency, economic values and resiliency, OBS networks can become vulnerable when burst loss occurs during ingress nodes sending data, causing BHP flood attacks. This problem may deteriorate the overall network's performance, due to the allocating of resources without proper usage. BHP flood attacks hinder the QoS of the OBS network, hence potentially causing a severe problem – the Denial of Service (DoS). This paper investigated the aforementioned issue by applying Machine Learning to automatically detect misbehaving ingress nodes, and blocking them in a preliminary stage. We evaluated various ML algorithms via simulation data, involving more than two ingress nodes and over 530 runs. The aim was to classify ingress nodes as accurately as possible, using variables related to their performance, such as packet drop rate, bandwidth used, and average delay time among others. Experimental results from a processed dataset related to BHP flood attacks showed that rule based classifiers, in particular decision trees (C4.5), Bagging, and RIDOR, consistently derive high predictive classifiers compared to alternate ML algorithms, including AdaBoost, Logistic Regression, Naïve Bayes, SVM-SMO and NN-MultilayerPerceptron. Moreover, the harmonic mean, recall and precision results of the rule based and tree classifiers were more competitive than those of the remaining ML algorithms.

Lastly, the runtime results measured in terms of millisecond showed that decision tree classifiers are not only more predictive, but are also more efficient than the rest of the algorithms. Thus, this is the most appropriate technique for classifying ingress nodes to combat the BHP flood attack problem. This paper is one of the initial attempts on adopting ML techniques to automatically classify ingress nodes in OBS networks.

In the near future, we intend to build a new rule-based classifier using the decision tree, and embed it inside the simulator to detect misbehaving nodes during the simulation phase.

BIBLIOGRAPHY

[1] Chatterjee, S., Pawlowski, S.: All-optical networks. Comm. ACM, vol. 42, pp. 74-83. (1999).

[2] Chen, Y., Verma, P. K..: Secure optical burst switching: framework and research directions. In: IEEE Communication Magazine, vol. 46, no. 8, pp. 40-45. (2008).

[3] Qiao, C. and Yoo, M.: Optical burst switching (OBS) - a new paradigm for an optical Internet. Journal of High Speed Networks, vol. 8, no.1, 69-84. (1999).

[4] Turner, J.: Terabit burst switching. Journal of High Speed Networks, vol. 8, pp. 3-16. (1999).

[5] Jue, J. P., Vokkarane, V. M.: Optical burst switched networks. Springer, (2006).

[6] Blumenthal, D. J., Prucnal, P. R., Sauer, J. R.: Photonic packet switches: architectures and experimental implementations. Proceedings of the IEEE, vol. 82, pp. 1650–1667. (1994).

[7] Chang, G.-K., Ellinas, G., Meagher, B., Xin, W., Yoo, S.J., Iqbal, M.Z., Way, W., Young, J., Dai, H., Chen, Y.J., Lee, C.D., Yang, X., Chowdhury, A., Chen, S.: Low Latency Packet Forwarding in IP over WDM Networks Using Optical Label Switching Techniques, : in IEEE LEOS 1999 Annual Meeting, pp. 17–18. (1999).

[8] Sreenath, N., Muthuraj, K.., Kuzhandaivelu, G. V.: Threats and vulnerabilities on TCP/OBS networks. in Proceedings of the International Conference on Computer Communication and Informatics (ICCCI '12), pp. 1–5. (2012).

[9] Sliti, M., Hamdi, M., Boudriga, N.: A novel optical firewall architecture for burst switched networks. in Proc. 12th Intl. Conference on Transparent Optical Networks (ICTON), pp. 1-5. (2010).

[10] Sliti, M., Boudriga, N.: BHP flooding vulnerability and countermeasure. Photonic Network Communications, 29(2), pp.198-213. (2015).

[11] Eddy W.: TCP SYN Flooding Attacks and Common Mitigations. RFC 4987. (2007).

[12] Chen, Y., Verma, P. K.., and Kak S.: Embedded security framework for integrated classical and quantum cryptography services in optical burst switching networks. Security and Communication Networks, vol. 2, no. 6, pp. 546–554. (2009).

[13] Chouhan, S. S., Sharma, S.: Identification of current attacks and their counter measures in optical burst switched (obs) network. Intl. Journal of Advanced Computer Research, vol. 2, no. 1, (2012).

[14] Kahate, A.: Cryptography and Network Security, 2nd edition, McGraw-Hill. (2008).

[15] Yuan, S., Stewart D.: Protection of optical networks against inter-channel eavesdropping and jamming attacks. Proc. Intl. Conference on Computational Science and Computational Intelligence (CSCI), Las Vegas, pp. 34–38. (2014).

[16] Stallings W.: Cryptography and Network Security, Prentice Hall. (2006).

[17] Terrance Frederick Fernandez, B. K. B. A., Sreenath, C. N.: Burstification threat in optical burst switched networks. pp. 1666–1670. (2014).

[18] Sreenath, N., Muthuraj, K., Sivasubramanian, P.: Secure optical internet: Attack detection and prevention mechanism. IEEE, pp. 1009–1012. (2012).

[19] Muthuraj, K., Sreenath, N.: Secure optical internet: an attack on OBS node in a TCP over OBS network. International Journal of emerging trends and technology in Computer Science, vol.1, no.4, pp. 75–80. (2012).

[20] Devi, B. S. K., Preetha, G., Shalinie, S. M.: DDoS detection using host-network based metrics and mitigation in experimental testbed. IEEE International Conference on Recent Trends in Information Technology (ICRTIT), MIT, Anna University, Chennai, pp. 423-427. (2012).

[21] Patil, R. Y., & Ragha, L.: A rate limiting mechanism for defending against flooding based distributed denial of service attack. In *Information and*

Communication Technologies (WICT), 2011 World Congress on pp. 182-186. IEEE (2011).

[22] Sharma, R., Kumar, K., Singh, K., & Joshi, R. C.: Shared based rate limiting: An ISP level solution to deal DDoS attacks. In *2006 Annual IEEE India Conference*, pp. 1-6. (2006).

[23] Patil, R. Y., & Ragha, L.: A dynamic rate limiting mechanism for flooding based distributed denial of service attack. In *Communication and Computing (ARTCom2012), Fourth International Conference on Advances in Recent Technologies in* (pp. 135-138). IET. (2012).

[24] Wang, F., Hu, X., & Su, J.: Mutual-aid team: Protect poor clients in rate-limiting-based DDoS defense. In *Communication Technology (ICCT), IEEE 14th International Conference on* pp. 773-778. (2012).

[25] Udhayan, J., & Anitha, R. (2009, March). Demystifying and rate limiting ICMP hosted DoS/DDoS flooding attacks with attack productivity analysis. In *Advance Computing Conference, IACC 2009. IEEE International,* pp. 558-564. (2009).

[26] S. Savage, D. Wetherall, A. Karlin, and T. Anderson.: Practical Network Support for IP Traceback. In *Proceedings of ACM SIGCOMM 2000,* Stockholm, Sweden, pp. 295-306. August (2000).

[27] Snoeren, A. C., Partridge, C., Sanchez, L. A., Jones, C. E., Tchakountio, F., Kent, S. T., Strayer, W. T.: Hash-Based IP Traceback". In *Proceedings of ACM SIGCOMM 2001*, San Diego, CA, USA, pp. 3–14. (2001).

[28] Gupta, B. B., Misra, M., & Joshi, R. C.: An ISP level solution to combat DDoS attacks using combined statistical based approach. *arXiv preprint arXiv:1203.2400*. (2012).

[29] Rajam, V. S., Selvaram, G., Kumar, M. P., & Shalinie, S. M.: Autonomous system based traceback mechanism for DDoS attack. In *2013 Fifth International Conference on Advanced Computing (ICoAC),* pp. 164-171. (2013).

[30] Kumar, K., Sangal, A. L., & Bhandari, A.: Traceback techniques against DDOS attacks: a comprehensive review. In *Computer and Communication Technology (ICCCT), 2011 2nd International Conference on,* pp. 491-498. (2011).

[31] Wei, J., Chen, K., Lian, Y. F., & Dai, Y. X.: A novel vector edge sampling scheme for IP traceback against DDoS attacks. In *2010 International Conference on Machine Learning and Cybernetics,* Vol. 6, pp. 2829-2832. (2010).

[32] [Online]. Available: http://nsl.csie.nctu.edu.tw/nctuns.html.

[33] Utilization, HP TopTools for Hubs & Switches, Hewlett-Packard Company 1999, http://hp.com/rnd/device_help/help/hpwnd/webhelp/HPJ4093A/utilizatio n.htm [online].

[34] Shakhov, V.: DDoS flooding attacks in OBS networks. IEEE. (2012).

[35] A.W. Moore, D. Zuev, Internet traffic classification using Bayesian analysis techniques, In: ACM SIGMETRICS Performance Evaluation Review. vol. 33, no. 1, 2005, pp. 50-60.

[36] A. Rajab, C.T. Huang, M. Alshargabi, and J. Cobb, Countering Burst Header Packet Flooding Attack in Optical Burst Switching Network, In: International Conference on Information Security Practice and Experience. Springer International Publishing. Nov 16 (2016) pp. 315–329.

[37] Witten I. H. and Frank E. (2005). Data Mining: Practical Machine Learning Tools and Techniques.

[38] F. Thabtah, W. Hadi, N. Abdelhamid, A. Issa, Prediction Phase in Associative Classification, In: *Journal of Knowledge Engineering and Software Engineering*. 21(6): (2011) pp. 855-876.

[39] A. McGregor, M. Hall, P. Lorier, J. Brunskill, Flow Clustering Using Machine Learning Techniques, In Proceedings of the Fifth Passive and Active Measurement Workshop. Springer Berlin Heidelberg. (2004) pp. 205-214.

[40] A. Jayaraj, T. Venkatesh, C. Murthy, Loss classification in optical burst switching networks using machine learning techniques: improving the performance of tcp, IEEE Journal on Selected Areas in Communications. 26(6) (2008) pp. 45 –54.

[41] T.T. Nguyen, G. Armitage, A survey of techniques for internet traffic classification using machine learning, IEEE Communications Surveys and

Tutorials. 10(4), (2008) pp. 56-76.

[42] J. Han, M. Kamber, J. Pei, *Data Mining: Concepts and Techniques*. Elsevier (2011).

[43] N. Abdelhamid, F. Thabtah, Associative Classification Approaches: Review and Comparison, *Journal of Information and Knowledge Management (JIKM).* 13(03), (2014) pp. 145-227.

[44] K.M. Leung, Decision trees and decision rules, Polytechnic University, Department of Computer Science, Finance and Risk Engineering. (2007).

[45] M. Sumner, E. Frank, M. Hall, Speeding up logistic model tree induction, In: PKDD. (2005), pp. 675–683.

[46] J.R. Quinlan, *C4.5: Programs for machine learning*, Elsevier. (2014).

[47] R.M. Mohammad, F. Thabtah, L. McCluskey, An Improved Self-Structuring Neural Network, in: Pacific-Asia Conference on Knowledge Discovery and Data Mining. (2016) pp 35-47.

[48] Ro. Duda, Pe. Hart, Pattern classification and scene analysis, John Wiley & Son. (1973).

[49] C. Cortes, V. Vapnik, Support-Vector Networks, In *Machine Learning*. 20 (3) (1995) pp. 273 - 297.

[50] M. Lévesque, H. Elbiaze, Graphical probabilistic routing model for OBS networks with realistic traffic scenario, In: IEEE *Global Telecommunications Conference. GLOBECOM (2009)* pp. 1-6.

[51] PS. Bradley, UM. Fayyad, OL. Mangasarian, Data mining: Overview and optimization opportunities, INFORMS: Journal of Computing 10. (1998).

[52] H. Liu, R. Setiono, Chi2: Feature Selection and Discretization of Numeric Attribute. In *Proceedings of the Seventh IEEE International Conference on Tools with Artificial Intelligence*. November 5-8 1995 pp. 388.

[53] S. Hashim, G. Esmat, W. Elakel, S. Habashy, S. Abdel Raouf, S. Darweesh, M. Soliman, M. Elhefnawi, M. El-Adawy, M. ElHefnawi, Accurate Prediction of Advanced Liver Fibrosis Using the Decision Tree Learning Algorithm in Chronic Hepatitis C Egyptian Patients, In: *Gastroenterology Research and Practice*. (2016).

[54] K. Doubravský, R. Doskočil, Determination of Client Profitability under Uncertainty Based on Decision Tree, In *Journal of Eastern Europe Research in Business & Economics*. (2016).

[55] R Development Core Team R: A Language and Environment for Statistical Computing, R Foundation for Statistical Computing. (2008).

[56] M. Hall M, E. Frank, G Holmes, B. Pfahringer, P. Reutemann, I. Witten, The WEKA Data Mining Software: An Update, In *SIGKDD Explorations*. 11(1) (2009).

[57] M. Hall, Correlation-based Feature Selection for Machine Learning. Thesis, department of computer science, Waikaito University, New Zealand. Morgan Kaufmann Publishers Inc. San Francisco, CA, USA (2011).

[58] D.M. Powers, Evaluation: From Precision, Recall and F-Measure to ROC, Informedness, Markedness & Correlation. In *Journal of Machine Learning Technologies.* 2 (1): (2011) pp. 37–63.

[59] A. DEMPSTER, N. LAIRD, D. RUBIN, Maximum likelihood from incomplete data via the EM algorithm, J. Royal Statist. Soc. B 39 (1977) pp. 1-38.

[60] L.E. Baum, T. Petrie, Statistical Inference for Probabilistic Functions of Finite State Markov Chains, The Annals of Mathematical Statistics. 37 (6) Retrieved 28 November (2011) pp. 1554–1563.

[61] N. Friedman, D. Geiger, M. Goldszmidt, Bayesian network classifiers, Machine learning. (1997).

[62] TM. Cover, PE. Hart, Nearest neighbor pattern classification, IEEE Transactions on Information Theory. 13 (1): (1967) pp. 21–27.

[63] D.E. Goldberg, Genetic Algorithms in Search, Optimization & Machine Learning, Addison-Wesley. (1989).

[64] Chen, Y., Qiao, C., and Yu, X.: Optical burst switching: A new area in optical networking research. IEEE Network 18(3), 16–23 (2004).

[65] Rajab, A., Huang CT, Al-Shargabi M., Cobb J (2016) Countering Burst Header Packet Flooding Attack in Optical Burst Switching Network.

ISPEC 2016: Information Security Practice and Experience pp 315-329.

[66] Rajab, A., (2017) Burst Header Packet (BHP) flooding attack on Optical Burst Switching (OBS) Network Data Set. University of California Irvine Data Repository, 2017.

[67] Gaines B. R., and Compton P., (1995). Induction of Ripple-Down Rules Applied to Modeling Large Databases. J. Intell. Inf. System. 5(3):211-228.

[68] Platt, J., (1998) Fast training of SVM using sequential optimization, (Advances in kernel methods – support vector learning, B. Scholkopf, C. Burges, A. Smola eds), MIT Press, Cambridge, 1998, pp. 185-208

[69] Rumelhart, D. E., Hinton, G. E., and Williams, R. J., (1986). Learning representations by back-propagating errors. Nature 323 (6088): 533–536.

[70] Quinlan, J., (1993) C4.5: Programs for machine learning. San Mateo, CA: Morgan Kaufmann.

[71] Freund, Y., and Schapire, R.E., (1997) A Decision-Theoretic Generalization of On-Line Learning and an Application to Boosting. Journal of Computer and System.

[72] Breiman, L., (1996) Bagging predictors. Machine Learning, 24 (1996), pp. 123-140.

[73] Lichman, M., (2013). UCI Machine Learning Repository [http://archive.ics.uci.edu/ml]. Irvine, CA: University of California, School of Information and Computer Science.

[74] Thabtah, F., (2007) A review of associative classification mining. The Knowledge Engineering Review 22 (1), 37-65.

[75] Abdelhamid, N., Thabtah, F., and Ayesh, A., (2014) Phishing detection based associative classification data mining. Expert systems with Applications Journal. 41 (2014) 5948–5959.

[76] Qabajeh, I., Thabtah, F., Chiclana, F. (2015) A dynamic rule induction method for classification in data mining. Journal of Management Analytics, 2(3): 233–253

[77] Abdel-Jaber, H., Thabtah, F., Woodward, M., Jaffar, A., and Al Bazar, H., (2014) Random Early Dynamic Detection Approach for Congestion

Control Baltic J. Modern Computing 2, 16-31.

[78] Thabtah, F., Qabajeh, I., and Chiclana, F., (2016) Constrained dynamic rule induction learning. Expert Systems with Applications 63, 74-85.

[79] Bunker, R., and Thabtah, F., (2017) A Machine Learning Framework for Sport Result Prediction. Applied Computing and Informatics Journal, pp. 1-21. Elsevier.

[80] Cohen, W., 1995. Fast Effective Rule Induction. In Proceedings of the Twelfth International Conference on Machine Learning. Tahoe City, California, 1995. Morgan Kaufmann.

[81] Fürnkranz J., and Widmer, G., (1994). Incremental reduced error pruning. In Machine Learning: Proceedings of the Eleventh Annual Conference, New Brunswick, New Jersey,1994. Morgan Kaufmann.

[82] Zaidi, N. A., Cerquides, J., Carman, M. J., and Webb, G. I., Alleviating naive bayes attribute independence assumption by attribute weighting. Journal of Machine Learning Research 14 (2013), 1947 – 1988.

[83] Zhang, J., Chen, C., Xiang, Y., Zhou, W., and Xiang, Y., Internet traffic classification by aggregating correlated naive bayes predictions. IEEE Transactions on Information Forensics and Security 8 (2013), 5–15.

[84] Quinlan, R., (2006) Bagging Boosting and C4.5.

[85] Schapire, R., 1990. The strength of weak learnability Machine Learning, 5(2), 197-227.

[86] Mohammad, R., Thabtah F., and McCluskey L., (2014A) Predicting Phishing Websites based on Self-Structuring Neural Network. Journal of Neural Computing and Applications, 25 (2). pp. 443-458. ISSN 0941-0643. Springer.

[87] Joachims, H., (1999) Large-scale support vector machine learning practical, Advances in kernel methods: support vector learning, MIT Press, Cambridge, MA, 1999.

[88] Shawe-Taylor J., and Sun S., (2011) A review of optimization methodologies in support vector machines, Neurocomputing 74 (17) (2011) 3609–3618.

[89] Joachims, T., Text categorization with support vector machines. Technical report, LS VIII Number 23, University of Dortmund, 1997. ftp://ftp-ai.informatik.uni-dortmund.de/pub/Reports/report23.ps.Z

[90] Le Cessie, S., Van Houwelingen, J.C., (1992). Ridge Estimators in Logistic Regression. Applied Statistics. 41(1):191-201.

[91] Patani1, N., Patel, R., (2017) A Mechanism for Prevention of Flooding based DDoS Attack. International Journal of Computational Intelligence Research ISSN 0973-1873 Volume 13, Number 1 (2017), pp. 101-

[92] Berral, J.L., Poggi, N., Alonso, J., Gavaldà, R., Torres, J., and Parashar, M., (2008) Adaptive distributed mechanism against flooding network attacks based on machine learning, Proceedings of the 1st ACM Workshop on Workshop on AISec, October, pp.43–50.

[93] Prathibha, R. C., and Rejimol Robinson, R. R., (2014)A Comparative Study of Defense Mechanisms against SYN Flooding Attack. International Journal of Computer Applications (0975 – 8887). Volume 98– No.18, July 2014.