BHP (Burst Header Packet) Flooding Attack Detection in Optimal Burst Switching Networks

Akshay Bhat
20/10/2022

Abstract

Optical burst switching (OBS) network is a fortunate switching technology for constructing the next generation of Internet backbone infrastructure. It operates by gathering UDP packets and transmitting a burst header packet (BHP) to accumulate the necessary network resources along the path before transmitting the related data burst. Suppose a source node (ingress) gets compromised by an attacker and floods the network with only BHPs to secure assistance without transmitting actual data. In that case, a refusal of assistance attack can ensue. This paper suggests and designs a unique security model that can be ingrained into an OBS core switch architecture to preclude BHP flooding attacks. The data accumulated from network traffic with the output approach is a class label of the BHP flooding attacks.
A few data mining approaches include data preprocessing, feature selection, and enforcing ML algorithms to analyse and illustrate a striking model to prevent BHP flooding attacks. For the classification phase, LDA, QDA, K-Neighbor classifier, Decision-based Tree, RandomForest, and Multinomial are utilised to create the model based on the chosen features of BHPs, diminishing the overfitting issue, and enhancing detection accuracy.
The model allows the OBS core switch to most accurately classify the ingress nodes founded on their conduct and the number of reserved resources that are not being used. The model suggested in this paper would block the malicious node that induces BHP flooding attacks and provide the network resources to the fair nodes.

Introduction

The Optimal Burst Switching (OBS) network primarily consists of three nodes: core nodes, ingress, and egress. The core nodes depict the intermediate nodes, devised to facilitate the processing and protection of the optical data burst using a management data packet with precise data such as BHPs (burst header packets) (T. Venkatesh and C. Murthy, 2010). Optimal Burst Switching (OBS) plays a vital role in packet switching with a more elevated level of essential elements than other living networks' switching techniques. Due to BHP flooding attacks, this switching class still suffers several challenges, such as security and quality of service (QoS). The function of Burst Header Packets (BHP) in OBS is to secure the new channel for the appearance of a data burst.
Attackers can manipulate this operation to send fake Burst Header Packets (BHP) without disclosure of Data Burst (DB). Such unnatural BHPs can impact the web and decrease performance by decreasing bandwidth utilisation and data loss, ultimately up to Denial-of-Service attacks (CISA. (2019, November 20), which is one of the considerable critical security hazards to network.
This paper presents an adequate and efficient method for ensuring the OBS networks. Thus, the primary purpose of the work is to design a light ML model for noticing BHP flooding attacks established on the forward feature selection method and classifier methods. Two questions are crafted to respond throughout this study to accomplish this goal. The first question is whether the feature selection method enhances the classifier model's efficacy in noticing BHP flooding attacks. The second question is whether the feature

selection method enhances the classifier model's efficiency for noticing BHP flooding attacks. The light effects of the model reach the point that only a few features are utilised to create the classifier.
The model will be assessed utilising the OBS dataset established on confusion metrics, accuracy, and cross-validation.

Data

The datasets extracted in this report were sourced from the UCI machine learning repository(https://archive.ics.uci.edu/ml/datasets/Burst+Header+Packet+%28BHP%29+flooding+attack+on+Optical+Burst+Switching+%28OBS%29+Network).

Table 1.1 Data Overview

Number of Instance	1075
Number of Attributes	22
Missing Values	N/A

Table 1.1 shows that the default pulled data does not have a column name for the 22 attributes, so the R-studio function (**colnames()**) has been utilised to name the attributes based on the information provided in the UCI machine learning repository. The extracted data has been stored in a variable called data to carry out the further investigation.

Methods

This section explains the methodology and components of the summary of tasks and processes to draw understanding into the Burst Header Packet Dataset. First, exploratory data analysis is carried out
Then splitting data into training and testing, 10-fold cross validation, and finally machine learning techniques like LDA, QDA, KNeighbour classifier, Tree based classifier, and Random forest with forward feature selections are used to compare which method classifies the class labels more precisely. This analysis has been performed using **R studio software** (Desktop mode with 2021.9.0.351 version.

Data Exploration and Visualisation

Let us begin with some fundamental knowledge of the variables, the arrangement of the data, and fundamental statistics about the data.

```
summary(data) # to check statistics
str(data) # to check structure
```

Let us do some data visualisations to comprehend the relationship between the variables.

```
pairs(data[,-c(14, 20,22)]) # retrieve numeric variables
```

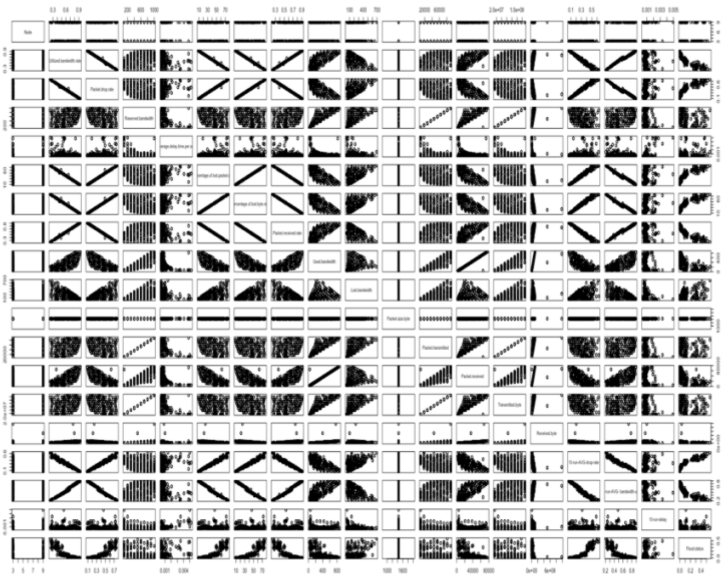


Figure 1. plots of all the numerical features

Figure 1 shows the relationship between each pair of variables and its associated connection with the target variable **class labels.**

Now, let us visualise the class label

```
data %>% ggplot(aes(x = Class.label, y = nrow(data), colour =Class.label))+geom_bar(stat = "identity")+labs(x = "class label",y = "")
```

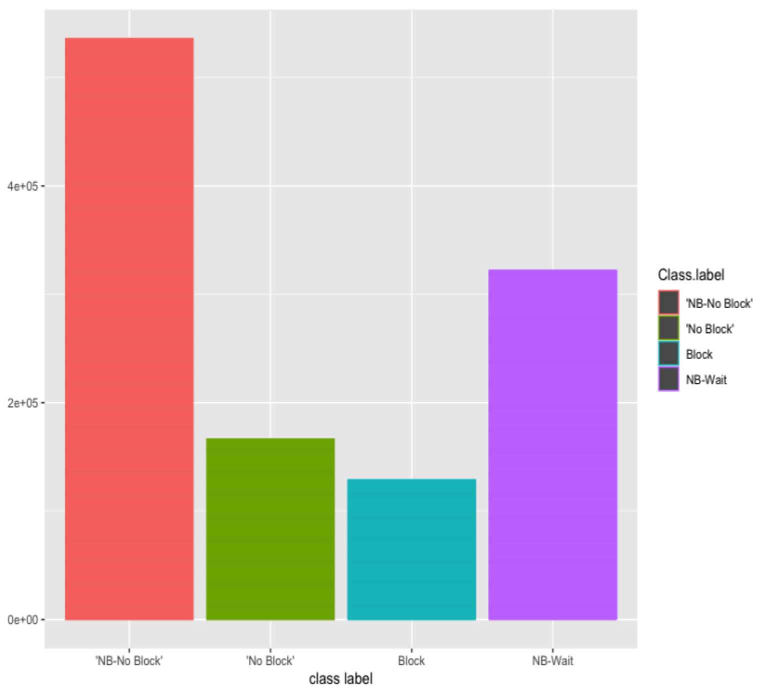


Fig 2: The distribution of instances for each class in the OBS network dataset

Table 1.2 The number of instances for each class in the OBS network dataset.

Class label	*NB-NO BLOCK*	*NO BLOCK*	*BLOCK*	*NB-WAIT*	*Total*
Number of Instances	*500*	*155*	*120*	*300*	*1075*

Figure 2 illustrates the allocation of instances over various types of BHP flooding attacks, and Table 1.2 indicates the number of samples for individual classes in the dataset.

Data Cleaning and Pre-processing

The attribute **Packet size byte** has been dropped because of its constant value of 1440 throughout all the accumulated samples in a dataset. Table 1.1 shows no missing values in an OBS dataset extracted from the UCI machine learning repository(data). A few variables are incredibly skewed, like Average delay time per sec, lost bandwidth, Received byte, 10-run-delay, and Flood status. Log transformation is applied when fitting ML models to the above-required attributes. Finally, the data is now ready for the validation method and ML model fitting.

Validation Set Approach

The data has been divided into training and validation sets, with 80% for the training and 20% for the validation set. The 20% validation data is held back and not exposed to the machine learning model until it is time to test the model.

```
Set.seed(123)
sample_split = sample.int(n = nrow(data), size = floor(.80*nrow(data)), replace = F) # data splitting into 80% and 20%
training_set = data[sample_split,] # training data with 80% samples
validation_set = data[-sample_split,] # test data with 20% samples
```

Model fitting

Considering the main objective of the paper with designing a lightweight ML model, based on the forward selection method, specific features that are most useful improve the model until adding a unique variable does not enhance the model's performance, which helps to reduce the test error rate significantly. In this investigation, the Class label is the response or dependent variable, and the other features are the independent variables. Several ML algorithms have been implemented, such as QDA, LDA, K-Neighbour, Decision Tree, and RandomForest model. Almaslukh achieved the random forest model with a 100% accuracy score in 2020 (Almaslukh, B. 2020)with **three features**. However, one of the most outbreaking results in this paper with the Randomforest, with only **two features** based on the feature importance, came up with an accuracy score of 100% while testing the model on the held-back 20% test set.

Results and Discussion

To summarise, our machine learning formula is class labels ~ features decided from the forward selection to improve the model performance with the lowest number of elements to align with the paper's objective.

Linear Discriminant Analysis (LDA), Quadratic Discriminant Analysis (QDA), K-Neighbor Classifier, Tree-Based classifier, and RandomForest are the machine learning algorithms used to classify the Class labels in the OBS networks. set.seed() function has been used to get the consistent results.

For LDA(Linear Discriminant Analysis), 14 features out of 22 have been utilised based on the forward selection process that could enhance the model performance and reduce the test error rate.

```
              'NB-No Block' 'No Block' Block NB-Wait
 'NB-No Block'            72          1     0      27
 'No Block'                6         25     0       0
 Block                     1          0    32       2
 NB-Wait                  17          0     0      32
```

Fig 3: LDA Confusion Matrix of classification for the 20% testing using 14 features

Fig 3 shows the confusion matrix with the **accuracy score of 74.88%** and **the test error rate of 25.11%** for LDA.

For QDA(Quadratic Discriminant Analysis), 13 features out of 22 have been utilised based on the forward selection process that could enhance the model performance and reduce the test error rate.

```
              'NB-No Block' 'No Block' Block NB-Wait
'NB-No Block'            51          0     0       6
'No Block'               14         26     0       0
Block                    10          0    32      16
NB-Wait                  21          0     0      39
```

Fig 4: QDA Confusion Matrix of classification for the 20% testing using 14 features

Fig 4 shows the confusion matrix with the **accuracy score of 68.83% and the test error rate of 31.16%** for QDA.

For KNN, 14 features out of 22 have been utilised based on the forward selection process. Choosing an optimal k value in KNN determines the number of neighbours we look at when we assign a value to any new observation. The model overfits the training data for a low value of k (suppose k=1), which **leads to a high error rate on the validation set.**

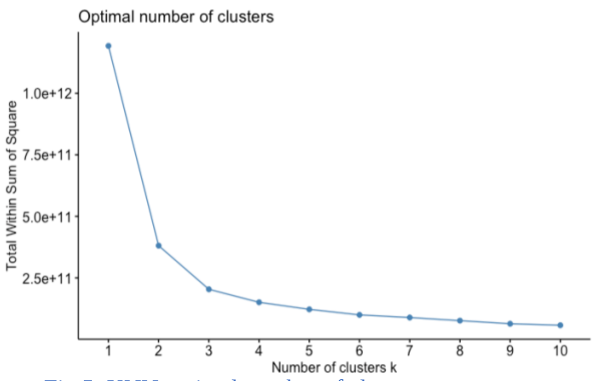


Fig 5: KNN optimal number of clusters

Figure 5 shows an "elbow" or bends at k = 3 neighbours. The total within the sum of squares begins to level off at this point. This tells us that the optimal number of neighbors to use in the KNN algorithm is 3

```
knn_model      'NB-No Block' 'No Block' Block NB-Wait
  'NB-No Block'           93          0     0       3
  'No Block'               0         26     0       0
  Block                    0          0    32       3
  NB-Wait                  3          0     0      55
```

Fig 5: KNN Confusion Matrix of classification for the 20% testing using 14 features.

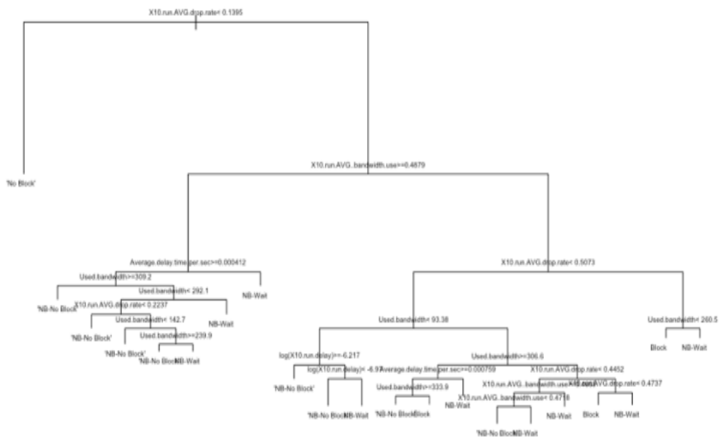
Fig 5 shows the confusion matrix with the **accuracy score of 95.81% and the test error rate of 4.18%** for KNN which eventually beats the model proposed by Almaslukh, B. (2020) with 21 features considering the ratio of the features along with its accuracy score and test error rate.

For Tree based classifier, 14 features out of 22 have been utilised based on the forward selection process.

```
tree_model     'NB-No Block' 'No Block' Block NB-Wait
  'NB-No Block'           93          0     0      15
  'No Block'               0         26     0       0
  Block                    0          0    32       0
  NB-Wait                  3          0     0      46
```

Fig 6: Tree based classifier Confusion Matrix of classification for the 20% testing using 14 features

Fig 6 shows the confusion matrix with the **accuracy score of 91.62% and the test error rate of 8.37%.**



X10.run.AVG.drop.rate< 0.1395
'No Block'
X10.run.AVG..bandwidth.use>=0.4879
Average.delay.time.per.sec>=0.000412
X10.run.AVG.drop.rate< 0.5073
Used.bandwidth>=309.2
Used.bandwidth< 292.1
NB-Wait
'NB-No Block'
X10.run.AVG.drop.rate< 0.2237
Used.bandwidth< 142.7
NB-Wait
'NB-No Block'
Used.bandwidth>=239.9
'NB-No Block'
'NB-No Block'
NB-Wait
Used.bandwidth< 93.38
Used.bandwidth< 260.5
Block
NB-Wait
log(X10.run.delay)>=-6.217
Used.bandwidth>=306.6
log(X10.run.delay)< -6.97
Average.delay.time.per.sec>=0.000759
X10.run.AVG.drop.rate< 0.4452
'NB-No Block'
Used.bandwidth>=333.9
X10.run.AVG..bandwidth.use
X10.run.AVG.drop.rate< 0.4737
X10.run.AVG..bandwidth.use< 0.4718
NB-Wait
'NB-No Block'
NB-Wait
'NB-No Block'
Block
NB-Wait
Block
NB-Wait
'NB-No Block'
NB-Wait

Fig 7: Tree based classifier

The tree-based classifier has utilised a 10-num-average drop rate as the primary root node for the classification task. If the **10-num-average drop** value is less than 0.1395, it will classify as **No block class label**, or it will go to **10-num-average bandwidth use** and so on with the different conditions as shown in figure 7 to classify the class labels on OBS networks. Initially, for RandomForest, certain features were utilised to fit the model and then, Flood status and '10-run-delay' were chosen to fit the RF model based on the feature importance plot shown in figure 8.

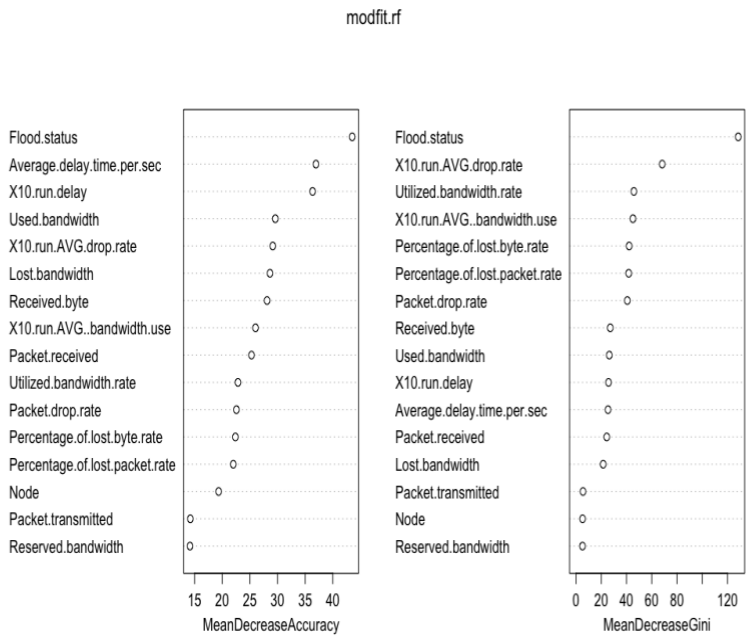


Fig 8: Random Forest Feature Importance

```
random_forest_model 'NB-No Block' 'No Block' Block NB-Wait
      'NB-No Block'            104          0     0       0
      'No Block'                 0         31     0       0
      Block                      0          0    23       0
      NB-Wait                    0          0     0      57
```

Fig 9: RandomForest Confusion Matrix of classification for the 20% testing using 2 features

Fig 9 shows the confusion matrix with the **accuracy score of 100% and the test error rate of 0% with only two features which is even more lightweight model proposed by Almaslukh, B. (2020)**. Table 1.2 displays the different ML approaches and their comparisons via the number of features, accuracy, and test error rate. It is clear from Table 1.2 RandomForest Model has topped

Table 1.2: Accuracy and number of features of the proposed BHP flooding attack detection approaches using OBS network dataset.

Approach	No of Features	Test Error rate (%)	Accuracy (%)
LDA	14	25.11	74.88
QDA	13	31.16	68.83
KNN	**14**	**4.18**	**95.81**
Decision Based Tree	14	8.37	91.62
RandomForest	**2**	**0**	**100**

with low bias and low variance with an **accuracy score of 100% with only two features directly connecting to the paper's primary objective of devising a lightweight ML model** to detect or classify the class labels in OBS networks. KNN comes close second to Random forest. More importantly, the benefit of the suggested method is that it decreases the number of features to two, which directs in a quicker running term than all features. This ensures that the method can catch attacks more efficiently, especially in overloaded networks with little computing help. We can infer that decreasing the features to two and using the validation and hold-out technique helped the suggested method to facilitate the overfitting problem and classify the OBS flooding attacks. Accordingly, all interpretation results explained the significance and efficiency of the models based on chosen components to classify BHP flooding attacks. This demonstrates that the suggested method is precise and appropriate for real-time detection in the limited computing ability of OBS switches.

Conclusion

This paper presents a promising approach using the forward feature selection method, 10-fold cross validation, hold-out technique, and a few ML techniques to reduce the overfitting problem and classify BHP flooding attacks on OBS networks. The process begins with selecting the most critical elements of the OBS network through the

forward feature selection to enhance the precision and efficiency of attack detection in OBS switches. The empirical outcomes indicate that the suggested technique can classify the class labels of OBS nodes with **100% accuracy by using only two features**.
The comparison with current corresponding results indicates that the suggested technique is appropriate for OBS network security in terms of significance and efficiency.
One of the constraints of the suggested technique is the scarcity of evaluation of more OBS datasets that can be altered in scope and classes of attacks due to inaccessible OBS datasets other than the dataset utilised in the investigations of this analysis.

References

- CISA. (2019, November 20). Understanding Denial-of-Service Attacks | CISA. Retrieved from www.cisa.gov website: https://www.cisa.gov/uscert/ncas/tips/ST04-015
- Almaslukh, B. (2020). An Efficient and Effective Approach for Flooding Attack Detection in Optical Burst Switching Networks. *Security and Communication Networks*, *2020*, 1–11. https://doi.org/10.1155/2020/8840058
- A. Rajab, C. T. Huang, M. Alshargabi, and J. Cobb, â€œCountering Burst Header Packet Flooding Attack in Optical Burst Switching Network,â€• In: International Conference on Information Security Practice and Experience, Springer International Publishing, pp. 315â€“329, Nov 16 2016
- T. Venkatesh and C. Murthy, An Analytical Approach to Optical Burst Switched Networks, Springer, Boston, MA, 2010.