BitTorrent Traffic Classification

Atwin O. Calchand, Van T. Dinh, Philip Branch, Jason But
Centre for Advanced Internet Architectures, Technical Report 090227A
Swinburne University of Technology
Melbourne, Australia
pbranch@swin.edu.au, jbut@swin.edu.au

***Abstract*—This paper describes the research undertaken to classify BitTorrent traffic from "Other" traffic. BitTorrent is a Peer-to-Peer (P2P) file sharing protocol. According to various research papers, P2P traffic makes up a large portion of today's Internet. P2P technology has been evolving to take into consideration and bypass methods of detection that could identify it. As such, P2P applications, like BitTorrent, use various obfuscation techniques to hide themselves. As P2P applications continue to evolve, powerful techniques need to be found to effectively identify P2P traffic. This paper outlines some research that we have been doing using Machine Learning techniques to identify BitTorrent traffic based on its traffic-flow characteristics.**

I. Introduction

Recent traffic measurement studies carried out by various institutions indicate that P2P occupies a large volume of the Internet, around 35% [1]. A proper mechanism that is able to identify P2P traffic on today's modern Internet is necessary since P2P applications are developing rapidly to adapt to newer detection techniques. Many P2P applications make use of obfuscation techniques, such as multiple TCP connections, port hopping, dynamic random port numbers, HTTP masquerading and encrypted payloads, to conceal their presence on the Internet.

Proper identification of BitTorrent is particularly useful to Internet Service Providers (ISP). One of the concerns to ISPs is the issue of P2P applications being bandwidth-intensive. When these applications start using a large proportion of the ISP's network, it slows down other basic services like web browsing. Excessive network congestion could hence lead to dissatisfied customers. A solution to this problem that ISPs can consider is to increase their network capacity by dedicating more resources but this is not always feasible since doing so adds to the operating costs of the company - adding new equipments is expensive, and so is their maintenance. Moreover increasing network capacity would only make a difference for a short amount of time as P2P traffic may soon expand to consume the increased capacity.

One other major concern to ISPs is the contents of files shared in P2P applications. Many of these files contain illicit material, such as games, video clips, movies, software and eBooks, which disregard copyright laws. Companies which produced the aforementioned contents have on various occasions held ISPs responsible for the copyright infringement [1].

Since BitTorrent makes use of dynamic ports, port-based analysis cannot be used to map known port numbers to BitTorrent. Also, there is an increasing amount of applications on the Internet which are making use of non-standard port numbers, in the case of running a web server on a random port instead of the usual port 80. Research suggests that 30-70% of Internet traffic cannot be accurately identified using port numbers. The other method that could be used to analyse network traffic relies on examining the contents of the payload in packets. The payload information contained would reveal a specific signature - each application has its own signature - which would help determine what software generated the packet. Payload analysis is difficult for various reasons. First, there is the issue of the payload being encrypted. Analysis of encrypted payloads is not only time-consuming, but it is also difficult. Secondly, there are the concerns about whether it is legal or not. Since examining the payload would mean that someone would know what data it contains, this constitutes a breach of the receiver's privacy.

A promising method that can effectively help us identify traffic generated by BitTorrent without having to worry about the legal ramifications and which would not take port number into consideration is the use of Machine Learning techniques. Machine Learning would allow the identification of specific patterns in traffic flow and then classify those flows.

This paper describes research in finding features that best identify BitTorrent traffic from "Other" Traffic - such as DNS and HTTP. The rest of the paper is structured as follows. Section II describes the setup of the experiment and our datasets. Section III discusses the results we obtained and provides some graphs to contrast BitTorrent from Other traffic.

II. Setup of the Experiment

This section explains how the experiment was set up and the process of analysing the traffic characteristics and behaviour in order to build the best feature set to classify BitTorrent traffic amongst other traffic. In particular, what these experiments hoped to achieve is a set of key features that could later be used to build a classifier model using Machine Learning (ML) techniques. In order to accomplish this task, the experiments were designed so that we could generate as many features as possible, and make use of multiple ML algorithms to sort out the best feature set and also determine the algorithm that gave us better results.

The experiment used two classes of trace files in pcap format [2]. One of the classes contained only BitTorrent traffic and the second trace contained a mix of other traffic - Hello packets, HTTP, DNS, QUAKE3 and more (with no payload). BitTorrent trace files were captured when simulating BitTorrent traffic over a FreeBSD jail system at the Centre for Advanced Internet Architectures, Swinburne University of Technology. The other trace was obtained from the University of Twente in the Netherlands.

A. Tools

Two important tools used in the experiment were Netmate and WEKA (Waikato Environment for Knowledge Analysis).

- Netmate [3] was used to group the raw packets into flows and generate statistics for those flows. The way netmate generates the statistics is through its modules. Netmate provides some standard modules, and each module generates a specific range of statistics. Because the experiments focused on flow classification, only a few essential modules were selected: `bandwidth, pktlen` and `jitter`. Using other modules does not seem significant in this case since they generate statistics for single packets or overlap other module output. However, they are useful if there is need to gather information about individual packet information.
- WEKA [4] - Selecting out relevant features is the next important part of the process. The experiment made use of WEKA's classifiers to build the model and indicate how effective each Machine Learning algorithm was at classifying the flows.

TABLE I
Attributes Used

`proto`	Protocol
`packets`	Total packet in the flow
`bytes`	Total byte in the flow
`diff_min`	Minimum inter-arrival time
`diff_avg`	Average inter-arrival time
`diff_max`	Maximum inter-arrival time
`diff_var`	Inter-arrival time variation
`minlen`	Minimum packet length in the flow
`maxlen`	Maximum packet length in the flow
`avglen`	Average packet length in the flow

B. Process

Fig. 1. illustrates how we used trace files to make the training and testing datasets. The same set of netmate modules (`bandwidth, pktlen` and `jitter`) were applied for all the trace files. The amount of output flows are not the same for every traces. However, to achieve more accuracy when using the ML technique later in the process, the training data file had been generated to contain comparative flows of BitTorrent and other traffic. The training files were formatted as `arff` files which are readable by WEKA. Before running WEKA classifiers, a preliminary step took place to remove the features that have been known to be irrelevant.

The following features were sorted out of the All-Feature-Training-File:

- `srcip` - Source IP Address
- `desip` - Destination IP Address
- `srcport` - Source Port Number
- `desip` - Destination Port Number

The reason to filter out these features is because IP addresses tell us nothing about the traffic behaviour and provides no help for classification and BitTorrent clients tend to use dynamic ports.

After generating the proper `arff` files that would be used for training and testing data, WEKA was run to build the model. Datasets A and B can be either used for training or for testing. The experiment ran all possible tests in order to get the best understanding of the data set. The steps can be listed as:

- Use data set A for both training and testing
- Use data set B for both training and testing
- Use data set A for training and B for testing
- Use data set B for training and A for testing
- Use data set A and cross-validation technique for testing

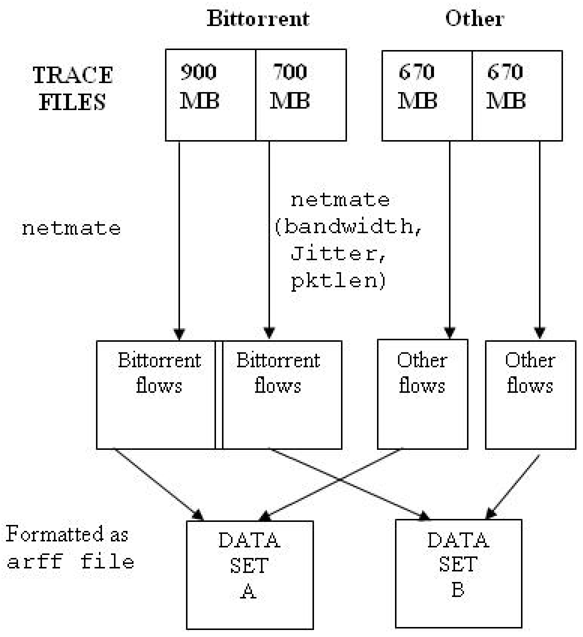


Fig. 1. Dataset Illustration

- Use data set B and cross-validation technique for testing

The next section describes the results for these experiments.

III. Results

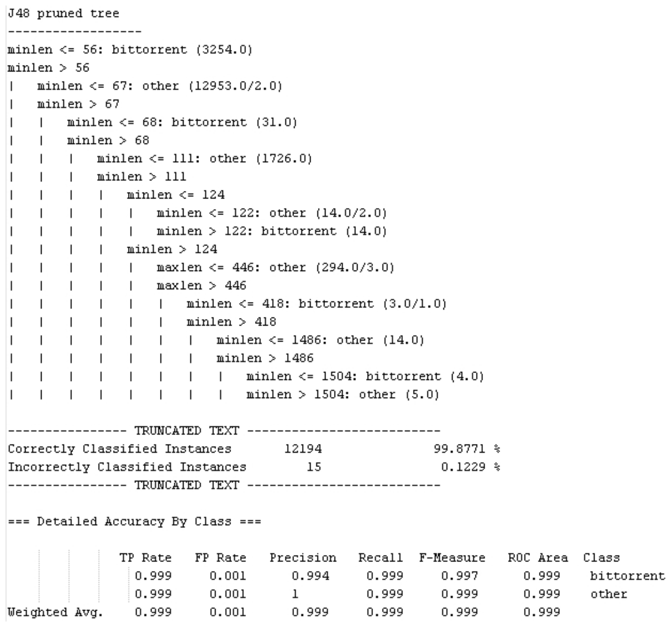
Once we had netmate's output from each of its packet processing modules we used them to build our training and testing datasets. The datasets were then used with WEKA [4] to run simulations with different Machine Learning algorithms to see how they compared and how well they could classify BitTorrent traffic from other traffic.

The main algorithms we used were the Naive Bayes, Bayes Net, J48 (C4.5) and NBTree. These algorithms have been showed to work well at classifying IP traffic flows [5]. Table II shows the results that we obtained. Once we had identified the attributes that gave a us a higher recall and precision, we built the best-features training and testing `arff` files. We have included the output from two tree algorithm which have worked well and since they are easy to understand as they provide a graphical representation of how the classifier is working on the dataset.

From Table II, we can see that packet length and packet inter-arrival time attributes gave us a higher accuracy in correctly identifying instances from each of the classes. Once we had this information, we built our best-features set using attributes from those two packet-processing modules only and ran more simulations to compare it with the results obtained when having attributes from all three modules. We found out that using bandwidth attributes did not contribute to classification and that we obtained better results using only the best-features set.

TABLE II
Percentage of correctly identified instances

Algorithm	**Bandwidth**	**Jitter**	**Packet Length**
Naive Bayes	43	88	94.3
Bayes Net	41	99.2	99.9
J48 (C4.5)	61	99.9	99.9



```
J48 pruned tree
------------------
minlen <= 56: bittorrent (3254.0)
minlen > 56
|   minlen <= 67: other (12953.0/2.0)
|   minlen > 67
|   |   minlen <= 68: bittorrent (31.0)
|   |   minlen > 68
|   |   |   minlen <= 111: other (1726.0)
|   |   |   minlen > 111
|   |   |   |   minlen <= 124
|   |   |   |   |   minlen <= 122: other (14.0/2.0)
|   |   |   |   |   minlen > 122: bittorrent (14.0)
|   |   |   |   minlen > 124
|   |   |   |   |   maxlen <= 446: other (294.0/3.0)
|   |   |   |   |   maxlen > 446
|   |   |   |   |   |   minlen <= 418: bittorrent (3.0/1.0)
|   |   |   |   |   |   minlen > 418
|   |   |   |   |   |   |   minlen <= 1486: other (14.0)
|   |   |   |   |   |   |   minlen > 1486
|   |   |   |   |   |   |   |   minlen <= 1504: bittorrent (4.0)
|   |   |   |   |   |   |   |   minlen > 1504: other (5.0)

---------------- TRUNCATED TEXT --------------------------
Correctly Classified Instances       12194              99.8771 %
Incorrectly Classified Instances        15               0.1229 %
---------------- TRUNCATED TEXT --------------------------

=== Detailed Accuracy By Class ===

                 TP Rate   FP Rate   Precision   Recall  F-Measure   ROC Area  Class
                   0.999     0.001      0.994     0.999     0.997      0.999    bittorrent
                   0.999     0.001      1         0.999     0.999      0.999    other
Weighted Avg.      0.999     0.001      0.999     0.999     0.999      0.999
```

Fig. 2. J48 algorithm (truncated output from WEKA)

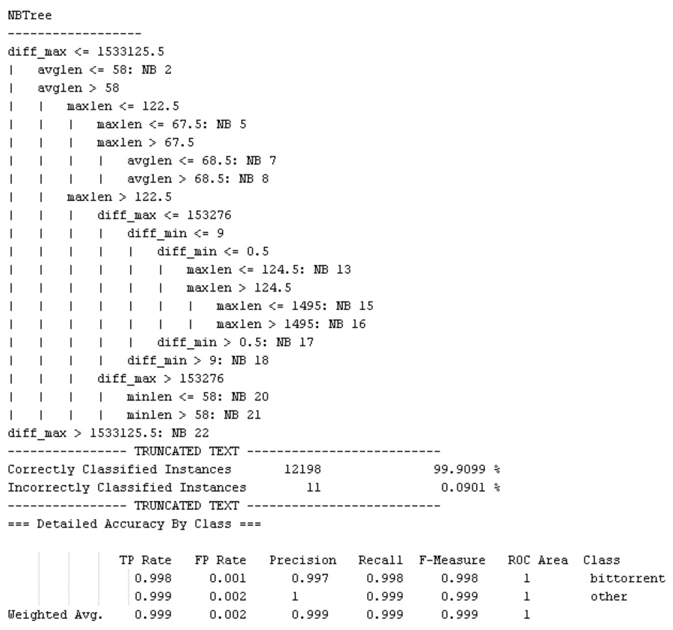
A. *J48 Algorithm Results*

The J48 algorithm is a decision-tree algorithm which is available in WEKA and is an implemetation of Quinlan's **C4.5** algorithm [6].

Fig. 2. shows the results that we obtained after running the J48 classifier on our training and testing datasets. The output from this algorithm gave us something to think about. As can be observed, the J48 classifier is making its decisions based mainly on the minimum packet length of each flow.

B. *NBTree Algorithm Results*

The NBTree (Naive Bayes Tree) algorithm is a hybrid algorithm combining the reliability and robustness of the Naive Bayes algorithm and the swiftness of decision tree



```
NBTree
------------------
diff_max <= 1533125.5
|   avglen <= 58: NB 2
|   avglen > 58
|   |   maxlen <= 122.5
|   |   |   maxlen <= 67.5: NB 5
|   |   |   maxlen > 67.5
|   |   |   |   avglen <= 68.5: NB 7
|   |   |   |   avglen > 68.5: NB 8
|   |   maxlen > 122.5
|   |   |   diff_max <= 153276
|   |   |   |   diff_min <= 9
|   |   |   |   |   diff_min <= 0.5
|   |   |   |   |   |   maxlen <= 124.5: NB 13
|   |   |   |   |   |   maxlen > 124.5
|   |   |   |   |   |   |   maxlen <= 1495: NB 15
|   |   |   |   |   |   |   maxlen > 1495: NB 16
|   |   |   |   |   diff_min > 0.5: NB 17
|   |   |   |   diff_min > 9: NB 18
|   |   |   diff_max > 153276
|   |   |   |   minlen <= 58: NB 20
|   |   |   |   minlen > 58: NB 21
diff_max > 1533125.5: NB 22
---------------- TRUNCATED TEXT --------------------------
Correctly Classified Instances       12198               99.9099 %
Incorrectly Classified Instances        11                0.0901 %
---------------- TRUNCATED TEXT --------------------------
=== Detailed Accuracy By Class ===

               TP Rate   FP Rate   Precision   Recall  F-Measure   ROC Area  Class
                 0.998     0.001      0.997     0.998     0.998      1        bittorrent
                 0.999     0.002      1         0.999     0.999      1        other
Weighted Avg.    0.999     0.002      0.999     0.999     0.999      1
```

Fig. 3. NBTree algorithm (truncated output from WEKA)

algorithms. These combined features make the NBTree outperform the other algorithms and NBTree has been shown to scale well in large databases [7].

Fig. 3. shows the results that we obtained after running the NBTree classifier on our training and testing datasets. The results obtained from the NBTree output tells us that a more broad range of features are being used to correctly identify each instance in the dataset including both packet-length and inter-arrival time modules.

C. *Further Investigation*

As we mentioned earlier the output from the J48 algorithm was surprising. We were not expecting the classifier to base its decisions mainly on one attribute, in that case the minimum packet length. Moreover, for that algorithm to reach a high level of precision (0.994) and recall (0.999) using that one attribute was even more confusing. This led us to analyse the distribution of packets in the whole PCAP files more accurately.

We used the "show_ascii" module supplied with the netmate package to get the size of each individual packet. Using a stream editor, we modified netmate's output to include only the packet sizes. We then developed a simple shell script which did a word-count on that output and gave us the frequency of each packet size. We ran this script on the output obtained for both the BitTorrent traffic and the other traffic.

Fig. 4. shows a plot of the cumulative packet sizes for both datasets. The minimum packet size for BitTorrent was 40 bytes and the one for Other was 28 bytes whereas their maximum packet size were 1500 bytes for each.

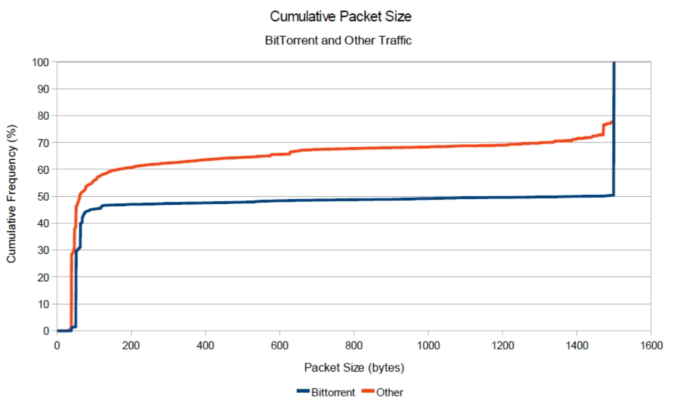


Fig. 4. Cumulative Packet Size Distribution of BitTorrent and Other traffic

Figures 5 and 6 show how the minimum and maximum packet sizes compare in each dataset. We have only included the range of statistics which gave a more representative idea of the distribution. The packets that were discarded in the graphs only accounted for between 0 - 0.01% of the whole distribution.

From the cumulative graph, we can observe that the minimum packet length and the maximum packet length are roughly the same. But if we take a closer look at the data in between the two extremities, we can see that the middle portion for BitTorrent accounts for around 4% of the total distribution whereas the one for the Other traffic amounts to around 12%. This information tells us that the average packet length plays an important role in distinguishing between BitTorrent and other traffic. The NBtree algorithm can be seen to be making use of the average packet length (avglen) attribute to classify the instances in the dataset. The mean packet length for BitTorrent was 800.18 bytes and the one for Other traffic was 537.15 bytes.

From these results, we deduced that we need to undertake some more work towards understanding the significance of the average packet lengths and why the J48 algorithm omits the 40 byte packets which constitute an important part of the Other traffic.

IV. Conclusion

The research conducted on BitTorrent and Other traffic has helped us to understand how packets are distributed in both datasets. We also found that there are enough distinct characteristics between BitTorrent and Other traffic that would make the former's classification possible using Machine Learning Techniques. We have also noted

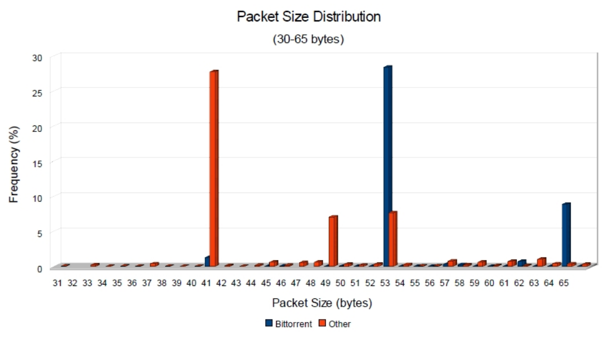


Fig. 5. BitTorrent and Other Traffic Minimum Packet Size Distribution

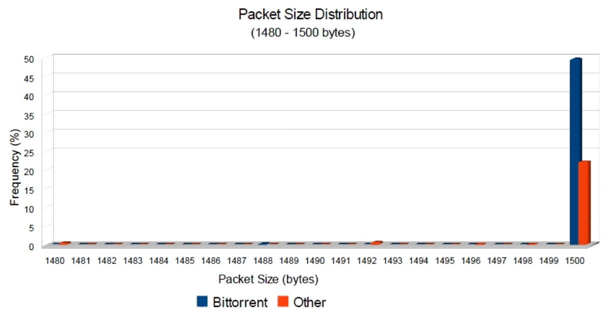


Fig. 6. BitTorrent and Other Traffic Maximum Packet Size Distribution

that using the J48 algorithm we reached an accuracy at correctly identifying instances of 99.8771%, with a precision and recall margin of 0.994 and 0.999 respectively. On the other hand, using the NBTree algorithm we reached an accuracy at correctly identifying instances of 99.9099%, with a precision and recall margin of 0.997 and 0.998 respectively.

References

[1] A. Madhukar and C. Williamson, "A longitudinal study of p2p traffic classification," 2006.
[2] Wikipedia, "Pcap — Wikipedia, the free encyclopedia," 2009, [Online; accessed 27-January-2009]. [Online]. Available: http://en.wikipedia.org/wiki/Pcap
[3] C. Schmoll and S. Zander, "Netmate — user and developer manual," 2004.
[4] T. U. of Waikato, "Weka 3: Data mining software in java," 2009, [Online; accessed 19-January-2009]. [Online]. Available: http://www.cs.waikato.ac.nz/~ml/weka/
[5] S. Zander, N. Williams, and G. Armitage, "A preliminary performance comparison of five machine learning algorithms for practical ip traffic flow classification," October 2006.
[6] G. T. Shed, "J48 decision trees," 2002, [Online; accessed 21-December-2008]. [Online]. Available: http://grb.mnsu.edu/grbts/doc/manual/J48_Decision_Trees.html
[7] D. Morrison, "Nbtree: A naive bayes/decision-tree hybrid," April 17, 2006.