Bittorrent traffic classification

Tung M Le[1], Jason But
Centre for Advanced Internet Architectures. Technical Report 091022A
Swinburne University of Technology
Melbourne, Australia
jbut@swin.edu.au

Abstract- **Bittorrent currently makes up a large proportion of Internet traffic. The main purpose of my project is to attempt to identify Bittorrent traffic from other kinds of traffic (with particular emphasis on FTP which provides similar functionality to BitTorrent in that it is primarily used for large file downloads) using statistical classification techniques. I will show that Bittorrent traffic can be identified efficiently and my method has great potential to classify Bittorrent in real time.**

Keywords- Bittorrent, ftp, other, traffic, flow, subflow

I. Introduction

Bittorrent is increasing in popularity due to high demand for large file distribution and its advantages compared with a client-server model. The issue arising for network management is how network managers can identify Bittorrent traffic. The current classification methods are mainly based on port number have decreased accuracy because Bittorrent applications nowadays can use random port numbers. This report introduces a new approach to identify Bittorrent traffic by using machine learning techniques.

Machine learning is a method that lets computers learn from data and make intelligent decisions. Previous research shows that machine learning techniques demonstrate high accuracy in classifying real-time IP traffic [1]. In our work, the computer learned from Bittorrent sample data. After that, the computer could automatically recognise Bittorrent traffic from arbitrary network traffic.

We mostly focused on the packet length statistics of Bittorrent traffic because these features are stable over different network conditions such as packet loss, delay and bandwidth. The statistice were minimum payload, the ratio of small packets, the ratio of big packets and the standard deviation of payload.

The purpose of this project is to classify Bittorrent traffic in real-time. The classifier should have both high accuracy and timely classification. We aim to identify Bittorrent subflows with accuracy more than 95%.

Our report has seven main parts: Introduction, problem overview, method, data analysis, experimental results, limitations and conclusion. Problem overview section introduces Bittorrent and machine learning. This is followed by the description of our method. The next section is the deep analyses all of my attributes. Experimental result section contains the classifier performance in both whole-of-flow and sub-flow. Some limitations are also mentioned in the following section.

[1] This author is currently engineer student at Swinburne University of Technology. This report was written during the author's winter internship at CAIA in 2009

II. Problem Overview

In this section we aim to introduce some background on Bittorrent and machine learning techniques (the main method to classify traffic). It also covers some basic issues and the solutions.

A. *Bittorrent Overview*

BitTorrent is one of the most popular protocols used for transferring large files, contributing to over one third of all Internet traffic [2]. BitTorrent is an open source peer-to-peer (p2p) protocol. There are several publically available BitTorrent programs, however all of them implement the same network protocol and share similar characteristics:

- True peer-2-peer communications
- Each peer both uploads and downloads content at the same time
- BitTorrent typically uses TCP as the carrier Transport Layer protocol

BitTorrent is different to other file transfer protocols, particularly in comparison to other file transfer protocols like FTP. Firstly, consider an FTP-based download. When you "download" a file, a point-to-point TCP connection is established. The client host sends a file request and the server responds with the requested data. When the file transfer is complete, the connection is released and that bandwidth can be used to service other clients. This causes problems as the number of concurrent clients increases, as bandwidth must be shared.

Instead, BitTorrent uses a different distribution method. Here, clients establish a peer-2-peer mesh network between each other. Downloading clients make available parts of the file they have already downloaded to other users. At this point the client is called a leech. When the download is complete, the client may continue to provide pieces of the file to other peers, at this point they assume the role of a seeder. This is managed by dividing the file into many small pieces which are individually requested and downloaded.

The download process begins by connecting to a tracker which provides the client with a list of available peers. P2p connections are then established. Clients exchange information about which pieces they currently have and request pieces from each other until the entire file is downloaded [3].

The main advantage of this approach over a client-server model is the distribution of required server power and bandwidth which also avoids the problem of a single point of failure. It also improves performance for popular downloads – as the number of concurrent downloads increases, the download speed increases rather than decreases.

B. *Why classify Bittorrent*

Although Bittorrent is very popular, we do not have any effective method to recognise it from other type of traffic. Correctly classifying Bittorrent may be useful for network administrator to manage their traffic and optimise bandwidth. In addition, there are a lot of Bittorrent downloads have issues with copyright law.

C. *Machine Learning*

Machine Learning is a method that uses algorithms to allow a computer to recognise complex patterns within a data set and to use these patterns to recognise different classes of data [4]. In this work, we use Machine Learning to distinguish between BitTorrent and other traffic types using statistical properties of the network traffic flows as the data sets.

D. *Other Classification Methods*

Typically, traffic classifiers are based on either port number matching or deep packet inspection. Port number matching is becoming increasingly inaccurate due to the use of random port numbers by applications in general and BitTorrent in particular [5]. While deep packet inspection can yield good results, it requires more processing and does not scale well to large traffic flows [6]. Further, there are also potentially legal issues when probing the contents of packets transmitted by network users.

III. Method

In order to classify individual BitTorrent traffic flows, we need to first group any captured network traffic into transport layer flows. This is done using the IP Address/Port Number tuples. We can then use different sets of "flows" to both train and test our classifier. This allows for post-flow classification, and does not extend to real-time classification. To do this we employ the sub-flow algorithm first developed by Nguyen et al. [7]. Using this technique we create our training data set in two steps:

1. Group packets into flows based on IP Address/Port Number tuples
2. Split each flow into a series of sub-flows of a fixed duration

Once the sub-flows are created, we can train a classifier to detect and classify captured sub-flows. By classifying a sub-flow it is possible to classify a flow before it has terminated, classify a flow where no all packets of the flow are captured, and update a flow classification as further sub-flows are captured and processed.

A. *Data sets*

Three types of traffic traces were used to train and test our classifier. These include known BitTorrent traffic, known FTP traffic, and Other traffic.

The Bittorrent traffic was captured under laboratory conditions. Several virtual computers on a single host were configured to participate in a Bittorrent download. DummyNet pipes simulated different types of connections. So the Bittorrent traffic may vary in delay, bandwidth or packet loss.

FTP traffic was captured from both local and remote FTP servers. More than 10GB of data transfer has been captured from local transfers, while a few hundred MB of data has been captured from remote transfers.

The Other traffic consists of traffic from publicly available traffic captured at the University of Twente [8]. This traffic is anonymised traffic captured at the University network and consists of generic Internet access traffic from the network users. The actual type of this traffic is unknown but we assume it not to contain any BitTorrent traffic.

B. *Software*

The main software packages used to perform the data analysis are the Weka toolset [9] and NetMate [10].

Netmate reads input from tcpdump captured files, divides the packets into flows, and then calculates a nominated set of statistics (eg. minimum, mean, maximum packet length) for each flow. Source code is available, it is possible to extend NetMate to calculate new attributes as required. Flow statistics are output in the arff file format [11]. NetMate can also be used to create sub-flows prior to calculating and outputting statistics.

Weka uses Machine Learning algorithms to process the generated arff files to construct a classifier. We typically use one set of arff files to train the classifier and a separate set of arff files (generated from a different set of traffic traces) to evaluate the performance of the classifier.

C. *Process*

We assume that all BitTorrent traffic is transported using the TCP protocol, as such all UDP traffic is removed from traffic traces prior to analysis.

We examined the traffic statistics for potentially suitable attributes. We built and tested the classifier using two approaches. Whole flow classification was performed to determine classification accuracy where the maximum amount of data was available to the classifier. These tests were run with different combinations of our selected classification attributes.

Next, the process was repeated for different sub-flow sizes with different combinations of the classification attributes. This allows for selection of the optimal set of attributes and sub-flow size.

IV. Data analysis

We initially explored the theoretical characteristics that might be found in the BitTorrent protocol. BitTorrent is typically used to transfer large amounts of data, as such it is likely to contain large (MTU) sized TCP segments.

We can surmise that if a TCP flow consists of many large sized segments, it is probably a file distribution protocol of some type.

We also note that while a BitTorrent client is a leech, data transfer is typically bi-directional, whereas a client-server based file transfer application is typically uni-directional.

We surmise that if a TCP flow consists of many large sized segments in both directions, it is probably a p2p file distribution protocol

Finally, examination of the BitTorrent protocol indicates that some messages transmitted between peers are of unique length. BitTorrent interest and unchoke packets are frequently transmitted throughout a BitTorrent session and have a fixed payload length of 5 bytes. Further BitTorrent request piece and cancel piece messages have a fixed payload length of 17 bytes.

We surmise that flows that exhibit significant proportions of segments with a payload size of 5 or 17 are probably BitTorrent flows

BitTorrent uses TCP as a transport layer protocol, packet delivery times are dictated by the TCP protocol and not the application layer protocol. Since these are often dependent on available bandwidth, congestion, packet loss, and current link utilisations, packet interarrival time statistics would not be useful to perform classification. Instead we concentrate on size-based statistics. We first discard all packets with a TCP payload size of 0 bytes (TCP Ack Packets) as they carry no communication value other than to acknowledge receipt of a packet and can distort any statistics calculated on sizes. Further, we also base our statistics on TCP payload size rather than packet size to remove any distorting affects from different hosts which may deploy different options within the IP/TCP header.

Below we nominate our selected attributes along with data to show why they may be useful for classification purposes, in both whole-of-flow and sub-flow classification.

A. *Whole flow attributes analysis*

1. Minimum payload

We define the minimum payload ($pload_{min}$) for a flow as the minimum TCP payload of all TCP packets which have a TCP payload greater than 0.

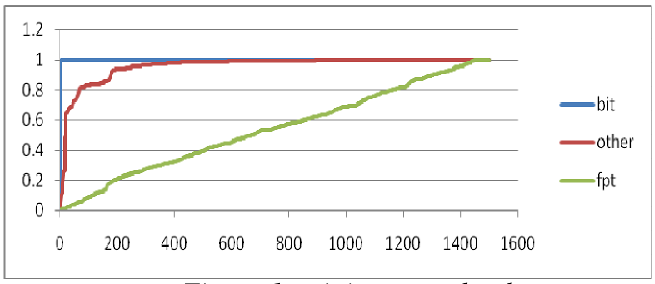

Figure 1: minimum payload

Figure 1 plots the CDF of $pload_{min}$ for our captured flows segmented into BitTorrent, FTP and Other. For BitTorrent flows, most flows have a $pload_{min}$ of 5 bytes (either Interest or Unchoke messages). In contrast, FTP flows have no specific $pload_{min}$. This is because during file transfer, almost all packets are the maximum MTU size, except for the final packet. The size of the final packet is based on the number of bytes remaining to finish the transfer and so we expect this packet to not only have the minimum payload size, but also for this size to be evenly distributed. $pload_{min}$ for other traffic flows have a range of values, but the majority fall into the range of 18-40 bytes.

2. Ratio of small packets

We define a small packet as one with a TCP payload in the range 1-40 bytes. We calculate the Small Packet Ratio ($ratio_{small}$) as the ratio of small packets to total packets within a flow.

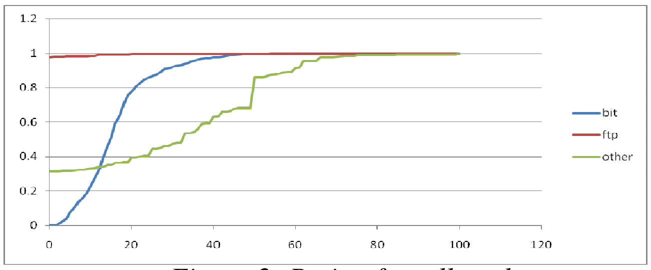

Figure 2: Ratio of small packets

Figure 2 plots the CDF of $ratio_{small}$ for the three flow types. FTP flows have a minimal number of small packets and thus typically have a $ratio_{small}$ value of ~0%. A large proportion of BitTorrent flows have $ratio_{small}$ ranging from 10% to 30% while very few Other traffic flows have a $ratio_{small}$ within this same range. Most Other flows have a value in the range of 30% to 60% while a significant proportion have a ratio_small of 50%. These are probably web flows which consist of a small HTTP request coupled with a single – not small – HTTP reply.

3. Ratio of big packets

We define a large packet as one with a TCP payload greater than 1350 bytes. We calculate the Large Packet Ratio ($ratio_{large}$) as the ratio of large packets to total packets within a flow.

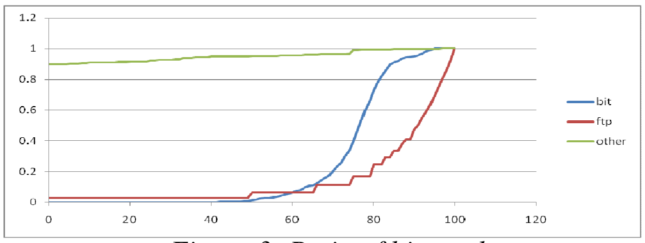

Figure 3: Ratio of big packets

Figure 3 plots the CDF of $ratio_{large}$ for the three flow types. The majority of Other traffic flows do not have many large packets and probably consist of applications such as web, email, chat, etc. However, as file transfer protocols, both BitTorrent and FTP have a significant number of large packets and similar values for $ratio_{large}$. Closer inspection reveals that FTP has a slightly higher $ratio_{large}$ (typically 85%-95%) than BitTorrent (typically 70%-80%)

4. Smaller Payload Standard Deviation

Each flow consists of data flowing in two directions. For each direction we calculate the standard deviation of the TCP payload size. We define the smaller of these two values as the Smaller Payload Standard Deviation ($stddev_{small}$)

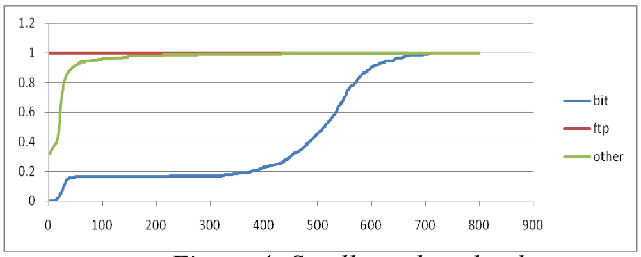

Figure 4: Smaller std payload

Figure 4 plots the CDF of $stddev_{small}$ for the three flow types. For FTP, data transfer is uni-directional so the standard deviation for the reverse direction is always 0. In contrast, BitTorrent flows consist of a bi-directional flow of both large and small packets. As such, the standard deviation in either direction is expected to be large. The $stddev_{small}$ for Other traffic is typically less than 100.

B.*Subflow size selection*

Selection of the sub-flow size is important. A smaller sub-flow size would result in more timely classification at the expense of using less accurate statistics. We would like to choose as small a sub-flow size as possible without compromising the performance of the classifier.

In the previous section, we showed that the most obvious feature for classifying BitTorrent was $pload_{min}$. This is due to the presence of unique BitTorrent Interest and Unchoke message packets which have a fixed payload size of 5 bytes. Almost all BitTorrent flows (see Figure 1) have a $pload_{min}$ of 5 bytes. We should determine a subflow size to try to take advantage of this particular feature, each subflow should have at least one Interest or Unchoke packet.

Figure 5 plots the CDF for the number of packets between two packets with a payload size of 5 bytes within the same BitTorrent flow. This demonstrates the frequency of these packets. Most of the "distances" between 5-byte packets are less than 200 packets while 50% are less than 50 packets. A sub-flow size of 50 should ensure that half of the sub-flows contain a 5-byte packet. For our experiments, we use sub-flow sizes of 50, 70, 100, 150, 200, 250 and 300 packets.

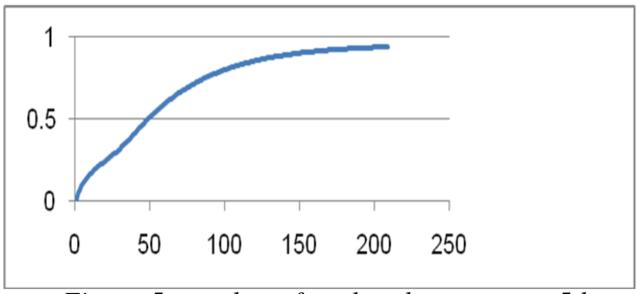

Figure 5: number of packets between two 5 byte payload packets

C.*Subflow attributes analysis*

We repeat the above analysis of the four attributes using a sub-flow size of 300 packets. Figures 6-10 plot the CDF for the nominated attribute, Figure 7 is a magnified plot of Figure 6.

1. Minimum payload

This attribute remains useful for classification purposes, the curves are clearly separated. As expected, most BitTorrent sub-flows have a $pload_{min}$ of either 5 or 17 bytes. About 60% of FTP sub-flows have a $pload_{min}$ of 1460 bytes (all packets have the maximum MTU) while the remainder typically have a $pload_{min} < 500$ bytes. Other traffic has the full range of $pload_{min}$, but 60% of these are less than 100 bytes. Consequently, a classifier may confuse FTP and Other sub-flows, however our main aim is to classify BitTorrent from all other flows.

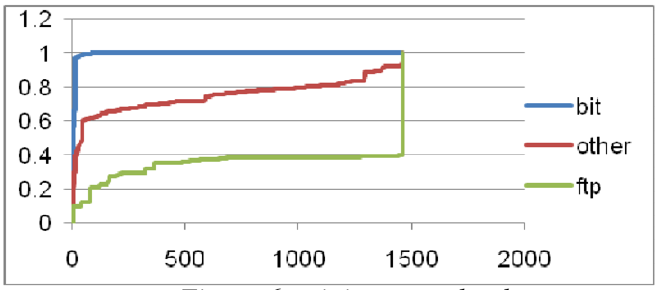

Figure 6: minimum payload

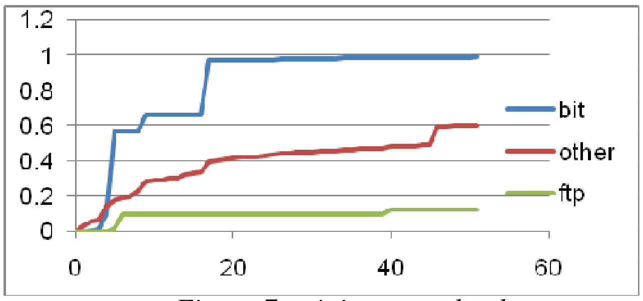

Figure 7: minimum payload

2. Ratio of small packets

Figure 8 plots the CDF of $ratio_{small}$. The shape is significantly different than for whole-of-flow analysis. 90% of FTP sub-flows and 60% of Other traffic sub-flows have no small packets. We can surmise that if a sub-flow has a $ratio_{small}$ of 0, it is likely to not be BitTorrent.

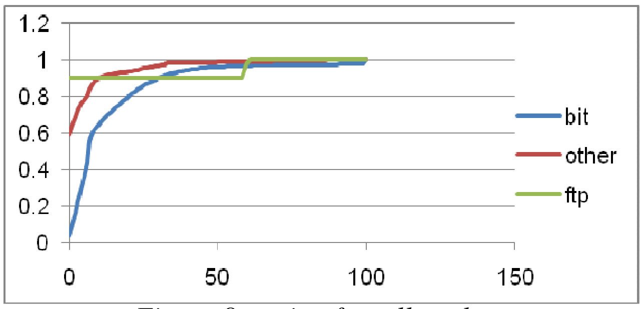

Figure 8: ratio of small packets

3. Ratio of big packets

Figure 9 plots the CDF of ratio_{large}. It is difficult to differentiate BitTorrent based on this attribute alone due to the similar shape of the curves. The major differentiation is that ratio_{large} for FTP sub-flows is typically very large (~100%).

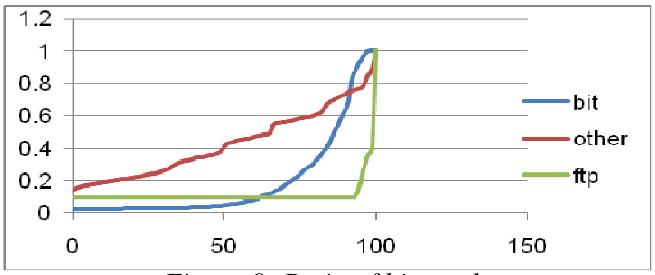

Figure 9: Ratio of big packets

4. Smaller Payload Standard Deviation

Figure 10 plots the CDF of stddev_{small}. About 60% of BitTorrent sub-flows have stddev_{small} in the range 200-700. Complicating the process is that a significant number of sub-flows for all traffic types have a stddev_{small} of 0.

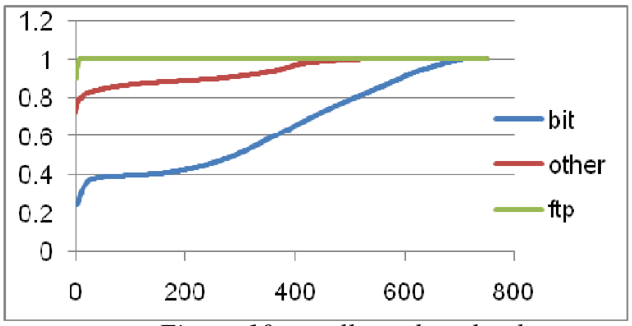

Figure 10: smaller std payload

V.EXPERIMENTAL RESULTS

We have two sets of results, classifier performance for whole-of-flow classification and for sub-flow classification. As expected, performance is better for whole-of-flow classification due to having more data to work with. This is at the expense of not being able to perform real-time classification. In all cases the classifier was trained to classify FTP traffic as Other traffic.

A. *Whole-of-flow Classification*

1. Single attribute result

The classifier was built and tested using a single attribute only. Each of the four nominated attributes was tested in turn.

Table 1 shows classification performance for each traffic type using the nominated attribute. The numbers indicate the percentage of flows that are correctly classified, technically these are referred to as recall. Pload_{min} and ratio_{small} produce good results in classifying BitTorrent while the other attributes are not as good. These results can be improved by combining attributes.

Attribute name	other	Bittorrent	ftp
pload_{min}	96.6	100	99.5
ratio_{small}	95.4	94.8	98.2
ratio_{large}	96.9	58.9	98.2
stddev_{small}	98.1	68.5	100

Table 1: single attribute result in whole flows

Attribute combination	other	Bittorrent	ftp
pload_{min}, ratio_{small}	99.1	100	99.7
pload_{min}, ratio_{large}	99.6	100	99.7
pload_{min}, stddev_{small}	99.7	95.2	100
ratio_{small}, ratio_{large}	99.2	98.4	98.7
ratio_{small}, stddev_{small}	98.5	87.9	100
ratio_{large}, stddev_{small}	98.6	99.6	100
pload_{min}, ratio_{small}, ratio_{large}	99.6	100	99.7
pload_{min}, ratio_{small}, stddev_{small}	99.5	96.8	100
pload_{min}, ratio_{large}, stddev_{small}	99.7	100	100
ratio_{small}, ratio_{large}, stddev_{small}	99.5	98.4	100
pload_{min}, ratio_{small}, ratio_{large}, stddev_{small}	99.7	100	100

Table 2: attribute combination in whole flows

2. Attributes combination result

Table 2 shows classification performance when more than one attribute is used for training and testing the classifier. Classification performance increases significantly. When combined with other attributes, the ratio_{large} and stddev_{small} attributes show improved outcomes. The best combination uses all four attributes where all BitTorrent and FTP flows were correctly classified and a very small percentage of other flows were incorrectly classified. Considering that the actual flow type for the Other flows was unknown and could contain p2p traffic, these results are very good.

B. *Sub-flow Classification*

The previous experiments are repeated for different sub-flow sizes (50, 70, 100, 150, 200, 250 and 300) for all possible attribute combinations. Figures 11-18 plot the classifier performance (vertical axis) for the nominated attribute combinations against the sub-flow size (horizontal axis). Results are plotted for all single attributes and some attribute combinations (to save space).

1. Single attribute performance

Results are shown in Figures 11-14. As expected, the performance is not as good as for whole-of-flow classification. In all cases, classification performance for BitTorrent sub-flows is poor for a sub-flow size of 50 packets but improves as the sub-flow size increases. However, overall performance is still poor, while some traffic types are accurately classified, others aren't. For example, in Figure 11 BitTorrent traffic is classified with accuracy ~95% but Other traffic is only classified with accuracy ~85%. Or consider Figure 14 where Other traffic classification is better at ~90% but BitTorrent performance is poor at ~80%.

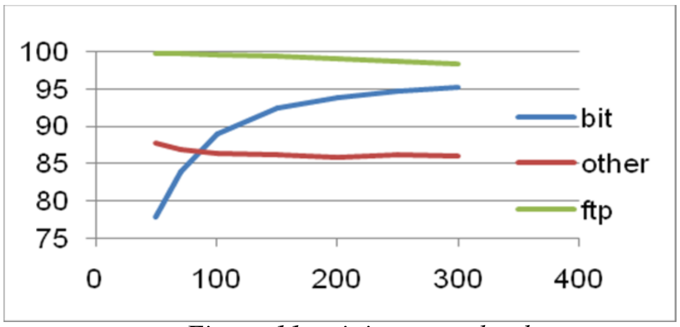

Figure 11: minimum payload

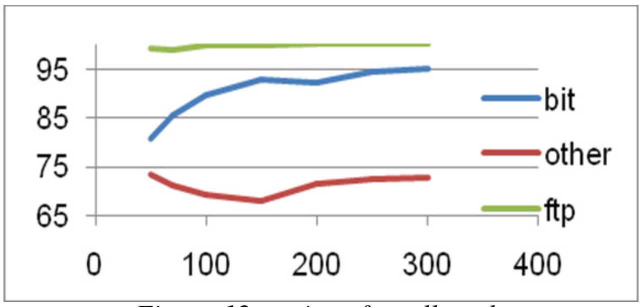

Figure 12: ratios of small packets

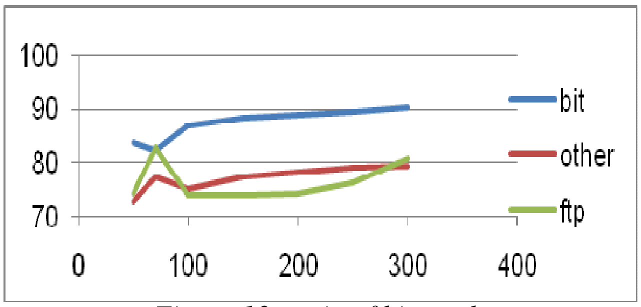

Figure 13: ratio of big packets

2. Multiple attribute performance

As for whole-of-flow classification, performance improves when we use multiple attributes. Some combinations are plotted in Figures 15-18. For these examples we still see BitTorrent performance increase as the sub-flow size increases, classification performance of Other and FTP flows in relatively stable at all sub-flow sizes.

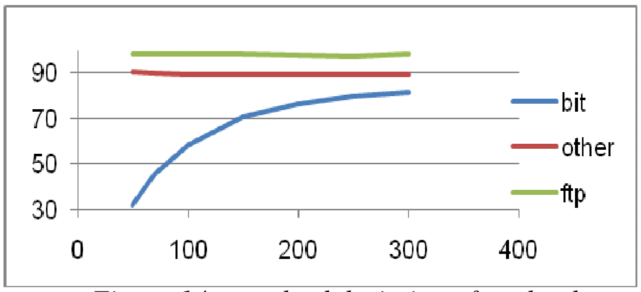

Figure 14: standard deviation of payload

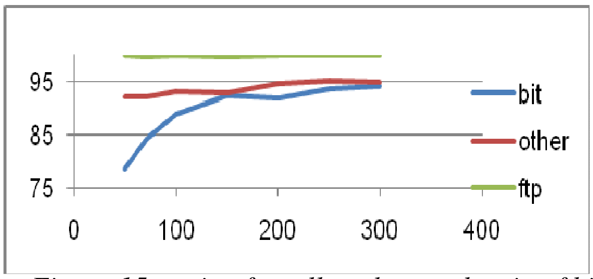

Figure 15: ratio of small packets and ratio of big packets

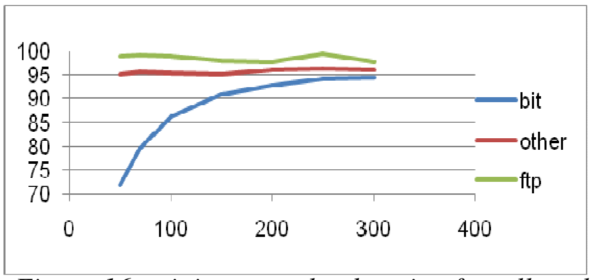

Figure 16: minimum payload, ratio of small packets and ratio of big packets

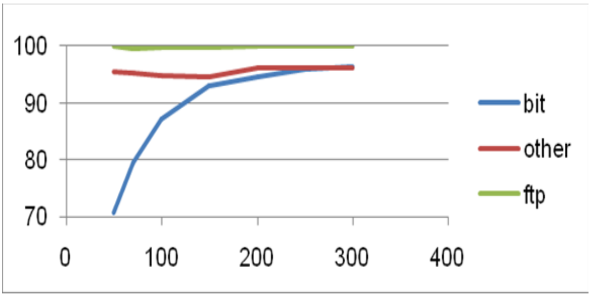

Figure 17: ratio of small packets, ratio of big packets and std payload

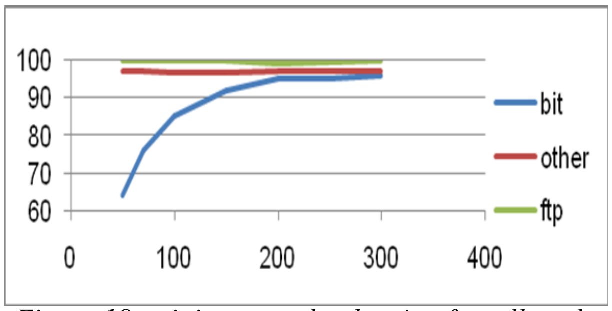

Figure 18: minimum payload,ratio of small packets, ratio of big packets and std payload

As we increase the sub-flow size, BitTorrent classification performance increases rapidly until the sub-flow size reaches a point of 200 packets. For larger sub-flow sizes, improvement in performance is minimal – for most cases the optimal sub-flow size is 200 packets.

The classifier performance is different with different attribute combinations. Normally performance improves as more attributes are used but this is not always the case. Even so, best performance was achieved when all four attributes were used (see Figure 18). For all four attributes and sub-flow size > 200 packets, accuracy for BitTorrent and other traffic approached 95% while FTP traffic was correctly classified nearly 100% of the time.

C. *Delay for Real-Time Classification*

We have nominated an optimum sub-flow size of 200 packets. How long will it take to capture 200 packets in a typical BitTorrent flow so that we have enough packets to generate a sub-flow for classification? The actual answer depends on a range of factors including network conditions, congestion and available bandwidth.

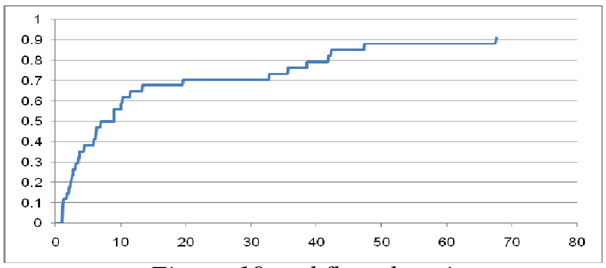

Figure 19: subflow duration

To calculate an approximate time duration for a 200 packet sub-flow with our data set, we plot the CDF of the time duration for each 200 packet sub-flow in our data set (see Figure 19). This time varies from 1 second to hundreds of seconds. However, 90% of our BitTorrent sub-flows has a duration of less than one minute. This indicates that it would typically take less than one minute from the start of traffic capture to classify a BitTorrent flow in real-time.

VI. LIMITATIONS

There are some issues that may affect classifier performance:

- The classifier was only tested with TCP Bittorrent transmission. Some applications actually use UDP to transfer Bittorrent data.
- Most of the Bittorrent data are captured under laboratory environment. It means there is only Bittorrent traffic in the connection link. The real Bittorrent traffic may be more complicated when Bittorrent shares link with other traffic.
- All captured BiTorrent traffic was generated using the transmission[12] BitTorrent client. The statistics may not be relevant for other BitTorrent clients or communications between different client applications.
- The classifier needs a reasonable amount of packets in each flow for analysing data. If a flow has fewer packets than the subflow size, it is assumed not to be a Bittorrent flow. However, for flows with few packets, it is not contributing to network congestion and not relevant in terms of traffic management.

VII. CONCLUSION

Bittorrent is a popular protocol but currently there are no efficient methods to classify it from other Internet traffic. This report introduces a new approach to classify Bittorrent traffic based only on TCP payload statistics.

The captured packets are grouped into flows. A flow is divided into smaller subflows and the TCP payload statistics of each subflow are calculated. The classifier uses those numbers to decide whether it is a Bittorrent subflow or not.

Four main attributes are used in the classification process. They are minimum payload, ratio of small packets, ratio of big packets and smaller std payload.

The classifier performance depends on subflow size and attribute combination. Larger subflow sizes result in more accurate classification. The optimal subflow size for both accuracy and timeliness is around 200 packets. The best attribute combination is using all four attribute.

The result when using all four attributes at a subflow size of 200 packets are: 95.3% BitTorrent subflows correctly classified, 99.1% FTP subflows correctly classified, and 97% other subflows correctly classified.

There are also some limitations to our work and results. Further research is required to develop this method and aim to improve classifier performance.

ACKNOWLEDGMENTS

Tung would like to thank Jason But and Philip Branch for their supervision, Grenville Armitage for providing the opportunity to undertake this project and CAIA for hospitality throughout my internship.

REFERENCES

[1] Thuy T. T. Nguyen and Grenville Armitage, “A Survey of Techniques for Internet Traffic Classification using Machine Learning”, IEEE Communications Surveys & Tutorials, vol. 10 no. 4 pp. 56-76, 2008

[2] Ipoque,http://www.ipoque.com/resources/internet-studies/internet-study-2008_2009, accessed September 2009

[3] WiseGreek, http://www.wisegeek.com/what-is-bittorrent.htm, accessed August 2009

[4] Nils J. Nilsson, “Introduction to Machine Learning”[online], http://robotics.stanford.edu/people/nilsson/MLDraftBook/MLBOOK.pdf, accessed September 2009

[5] Yu Wang, Shun-Zheng Yu, "Move Statistics-Based Traffic Classifiers Online," csse, vol. 4, pp.721-725, 2008 International Conference on Computer Science and Software Engineering, 2008

[6] Young H. Cho, William H. Mangione-Smith , “Deep network packet filter design for reconfigurable devices”, Volume 7 , Issue 2 (February 2008)

[7] Thuy T. T. Nguyen, “A novel approach for practical real-time, machine learning based IP traffic classification”, PhD thesis, Swinburne University of Technology, 2009

[8] University of Twente, http://www.universiteittwente.nl/en,

accessed July 2009

[9] University of Waikato, http://www.cs.waikato.ac.nz/ml/weka/, accessed July 2009

[10] Netmate, http://www.ip-measurement.org/tools/netmate/, accessed July 2009

[11] Attribute relation file format, http://www.cs.waikato.ac.nz/~ml/weka/arff.html, accessed July 2009

[12] Transmission Bittorrent client, http://www.transmissionbt.com/, access September 2009