Identify P2P Traffic by Inspecting Data Transfer Behaviour*

Mingjiang Ye[1], Jianping Wu[1], Ke Xu[1], and Dah Ming Chiu[2]

[1] Tsinghua National Laboratory for Information Science and Technology, Department of Computer Science, Tsinghua University, Beijing, 100084, P.R. China
yemingjiang@csnet1.cs.tsinghua.edu.cn, jianping@cernet.edu.cn, xuke@csnet1.cs.tsinghua.edu.cn
[2] Dept. of Information Engineering
The Chinese University of Hong Kong Hong Kong, P.R. China
dmchiu@ie.cuhk.edu.hk

Abstract. Classifying network traffic according to its applications is important to a broad range of network areas. Since new applications, especially P2P applications, no longer use well-known fixed port numbers, the native port based traffic classification technique has become much less effective. In this paper, we propose a novel approach to identify P2P traffic by leveraging on the data transfer behaviour of P2P applications. The behaviour investigated in the paper is that downloaded data from a P2P host will be uploaded to other hosts later. To find the shared data of downloading flows and uploading flows online, the content based partitioning schema is used to partition the flows into data blocks. Flows sharing the same data blocks are identified as P2P flows. The effectiveness of this method is demonstrated by experiments on various P2P applications. The results show that the algorithm can identify P2P applications very accurately while only keeping a small set of data blocks. The method is generic and can be applied to most P2P applications.

Keywords: traffic management, P2P traffic identification, data transfer behaviour, content based partitioning, Rabin fingerprint.

1 Introduction

Classifying network traffic according to its applications is important to a broad range of network areas including network monitoring, network management and optimization, network security, traffic accounting etc. Different from the traditional applications (http, email, ftp), new applications, especially P2P applications, usually use dynamic port numbers. The traffic classification technique based on native port has become less effective. However, the payload signature

* This research is supported by NSFC-RGC Joint Research Project (20731160014), 973 Project of China (2009CB320501), 863 Project of China (2008AA01A326, 2006AA01Z205, 2006AA01Z209), and Program for New Century Excellent Talents in University.

L. Fratta et al. (Eds.): NETWORKING 2009, LNCS 5550, pp. 131–144, 2009.

based traffic classification technique [1,2] can achieve a high accuracy. But the technique also has its limitations. It is ineffective in classifying encrypted traffic. Besides, a lot of P2P applications use proprietary protocols. Lacking open protocol specifications makes analysing signatures and maintaining up-to-date signatures very difficult.

Recently, machine learning algorithms which classify network traffic using flow statistical information [5,6,7,8,9] have been proposed. There are several challenges in classifying network traffic by using flow properties. First, the statistical characteristics used in classification are unstable since the delays and/or the packet loss ratios of the networks are dynamic. Second, flows belonging to different applications can have similar per-flow statistical characteristics. It is hard to distinguish these similar flows by using flow properties.

This paper proposes a novel approach to identify P2P traffic based on its data transfer behavior. The idea of the approach is based on the observation that a P2P peer uploads data to other peers after downloading it. The idea is first proposed by Xing Lu [10]. In their method, the first k bytes of each packet in downloading flows are stored for each host. When the same content are found in uploading flows of the host, the flows associated with the content are classified as P2P flows. The partitioning schema in their method is named the head packet partitioning schema in this paper. As shown in our experiments, the performance of the head packet partitioning schema is very poor in identifying some P2P applications.

The content based partitioning schema is proposed in the paper to solve the problem. The schema divides payloads of flows into data blocks (a data block is a contiguous content block of payload). The shared files and videos in P2P applications are usually divided into small data pieces during exchanges. For flows sharing the same data piece, the schema can synchronize the boundary of the data piece, and extract the same part of the data piece as a data block. The schema is generic and can be applied to most P2P applications.

Our contributions are as follows. First, we proposed the content based partitioning schema in identifying data transfer behaviour. As shown in experiments, The schema performs much better than the head packet partitioning schema. Besides, Head tail partitioning schema, a simple enhancement schema of the head packet partitioning schema, also can improve performance greatly, though not as good as the content based partitioning schema. Second, we have studied the performance impactions of some important issues. They are the size of the data block, the number of the data blocks to be stored, the data block replacement algorithms and the ratio of unobservable communications.

From the studies, we have drawn several conclusions. First, 256 bytes is a suitable size for data block. Second, the random replacement algorithm is suggested in replacing the old data blocks. Third, with the random replacement algorithm, the method performs quit well while only keeping a small data block set (3 minutes). Last, even though the communications of a large fraction of peers (about 30%) are not observed, the performance degradation is rather small.

2 Related Work

Payload signatures are useful in classifying network traffic [1,2]. Application signatures are the common strings in the P2P protocols to identify P2P traffic whereas our method focuses on the data being shared in P2P applications.

In addition to signature based methods, other payload based methods are also proposed. ACAS [3] uses the first N bytes payload as the input to train a machine learning model and uses it to classify flows. Levchenko et al. build several probabilistic models on payload, including the statistical model treating each n-byte flow distribution as a product of N independent byte distributions and the Markov process model which relies on introducing independence between bytes [4]. The models still employ frequently appearing bytes in application protocols to classify traffic since the random bytes in application data are meaningless in statistics.

The machine learning algorithms classify network traffic by using traffic characteristics [5,6,7,8,9], such as average length of packets, arrival interval of packets, etc. The algorithms can be further summarized into supervised and unsupervised methods. Zander et al. compare several supervised algorithms, including Naive Bayes, C4.5 decision tree, Bayesian Network and Naive Bayes Tree [9]. They find that the classification precision of the algorithms is similar, but computational performance can be significantly different. McGregor et al. use the unsupervised expectation maximum algorithm to cluster the flows [6]. The experiment finds that the average precision of classification is very high, but some applications are very difficult to distinguish.

The special communication patterns of applications are used in traffic classification. Karagiannis et al. study the communication pattern of P2P traffic in transport layer to identify P2P traffic [11]. BLINC attempts to capture the communication pattern of a host at three levels: the social level, the functional level and the application level [12]. Graphlet is used to describe the communication pattern of each application and classify network traffic.

It is hard to find a general communication pattern to fit all P2P applications. Y Hu et al. propose a method to build behavioural profiles of the target application which then describes the dominant communication patterns of the application [13]. The profiles of each application are built by data mining algorithms in the training phase. The data transfer behaviour used in our method is general for all the P2P applications, so our method does not need any training phases.

Using the data transfer behaviour to identify P2P traffic was first proposed by Xing Lu et al. [10]. But they only use the first k bytes of packet to find the shared data. As shown in the experiments, our schemas performances much better. Our method is general for different P2P applications while their schema has some limitation to identify some P2P applications. Besides, we have studies some important issues in identifying the data transfer behaviour which are not studied before.

3 Method

3.1 Payload Partitioning Schemas

The basic idea of the method is simple. The method inspects the P2P data transfer behaviour on the host level. From the view of a host, flows can be classified as downloading flows and uploading flows. If a host downloads a data piece in a downloading flow, and the uploads it to other hosts in some uploading flows. These flows are classified as P2P flows.

The biggest obstacle in the method is how to find the same data pieces in different flows. One way is comparing the payloads of flows directly. But it is time consuming. Besides, complete payloads of flows have to be saved in the memory before comparing. It is quit difficult - and therefore, unrealistic - as an online process.

In our method, payloads of flows are divided into data blocks and then the signatures of the data blocks are compared. A payload partitioning schema decides how to divide the payload into data blocks. The ideal result is that each data block is an exact data piece in a P2P application. For example, for two distinct flows in Fig. 1, the ideal case is that the generated data blocks in the first flow equal to data pieces 1, 5 and 7 and the generated data blocks in the second flow equal to data pieces 9, 4 and 1. Thus, the shared data (pieces 1) of the two flows can be found.

However, a prerequisite to generating such ideal results is the detailed protocol information. As shown in Fig. 1, there are several challenges in locating the boundaries of data pieces. First, a flow can contain protocol fields with variable sizes. Second, the sizes of data pieces are variable in some P2P live streaming applications. In these applications, a data piece contains 1 second of video content. The size of the data piece is variable in most video coding schemas.

Several partitioning schemas are considered. The first one is the method used in [10]. We called it as the head packet partitioning schema. If the payload size of a packet exceeds the threshold S, the first S bytes of the packet is extracted as a data block.

The second one is the head tail packet partitioning schema. If the payload size of a packet exceeds the threshold value S, the first S bytes of payload and the last S bytes of payload are extracted as two data blocks. This schema is an

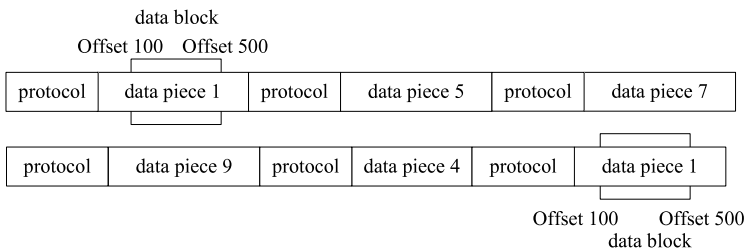

Fig. 1. Shared data piece

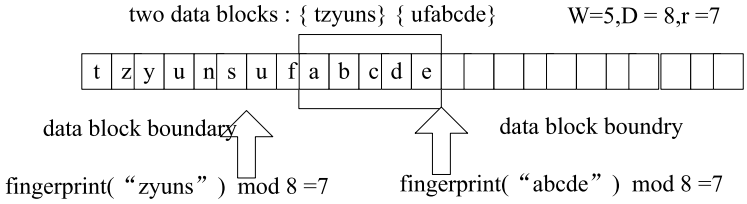

Fig. 2. Content based partitioning schema

enhancement of the previous one. If the target packets contain some protocol fields at the beginning or at the end, the schema can skip them.

The last one is the content based partitioning schema. It has been used in saving bandwidth in network file systems [14] and automatically generating signatures of worms in security fields [10]. The schema divides a flow into variable-size, non-overlapping data blocks. The schema works as follows. First, a pair of pre-determined integers D and r ($r < D$) are set. Then a sliding window of fixed width W moves across the byte sequence. The window begins from the first W bytes in the sequence, and slides one byte at a time toward the end. At every position of the byte sequences, a fingerprint F is calculated according to the content in the current window. If F mod D equals r, the end of the window is a data block boundary.

For example, in Fig. 2, the windows size is 5, D is 8 and r is 7. A window of width 5 moves across the byte sequence and fingerprints are calculated in every position. When the window reaches abcde, two positions have satisfied the condition. Two data blocks tzynuns and ufabcde are extracted. The fingerprint is calculated by using Rabin fingerprint [16] which has a low collision rate. Using the pre-computed table, it is vey efficient to calculate the Rabin fingerprint [17].

The principle behind the content based partitioning schema is that, because the schema determines the boundaries of data blocks based on the local content of payload in a small window, the boundaries can synchronize in the same data pieces. For example, in Fig. 3, there are two flows containing the same data piece (the black part) in different positions. In the first flow, there are two positions satisfying the condition. They locate at offset 100 and 500 from the beginning of the data piece. In the second flow, the positions at the same offsets from the beginning of the data piece should also satisfy the condition, as the contents in the windows are the same. The same data block is extracted in both flows.

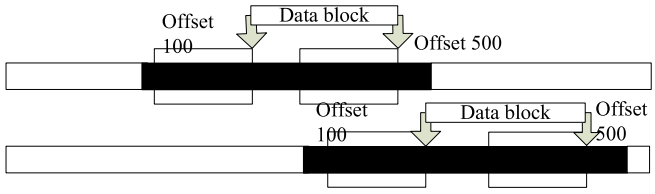

Fig. 3. Principle of content based partitioning

The content based partitioning schema can create blocks with various sizes. Although the average size of blocks is D, the data blocks can be as short as several bytes. Short data blocks introduce many false positives, so the results are further filtered to only keep data blocks whose sizes exceed threshold S.

The value of D decides the average size of data blocks. If D is much smaller than S, many data blocks which are smaller than S will be generated and filtered. So D is equal to S in the algorithm. Since the performance is not sensitive to the value of r, we set r to $D - 1$ in the algorithm.

The boundaries of data blocks are determined by the local content in the window. The window size should be much smaller than the size of data pieces to generate data blocks inside a data piece. On the other side, a small window is sensitive for random content in flows. In the experiments, the window size is 32 bytes.

3.2 Algorithm

The details of the algorithm are as follows. For each host, the algorithm keeps a hash set, which is called the download set, to save data blocks. Payloads of flows are divided into data blocks. Data blocks of each downloading flow and flow identifiers (a flow identifier is the 5-tuple: source IP address, source port, destination IP address, destination port, and protocol) are saved in the corresponding download set. To save the memory capacity, signatures of data blocks (Ranbin fingerprint) are calculated and saved instead of the original data blocks. If the size of the download set exceeds the limitation, then data block replacement is applied. Data blocks of each uploading flow are checked whether they have been saved in the corresponding download set already. If a data block of the uploading flow has been saved in the download set, the uploading flow, the downloading flow and their reverse flows are identified as P2P flows.

A flow has two roles. One is the downloading flow of the target host. The other is the uploading flow of the source host. The data blocks of the flow are saved in the download set of the target host and checked in the download set of the source host.

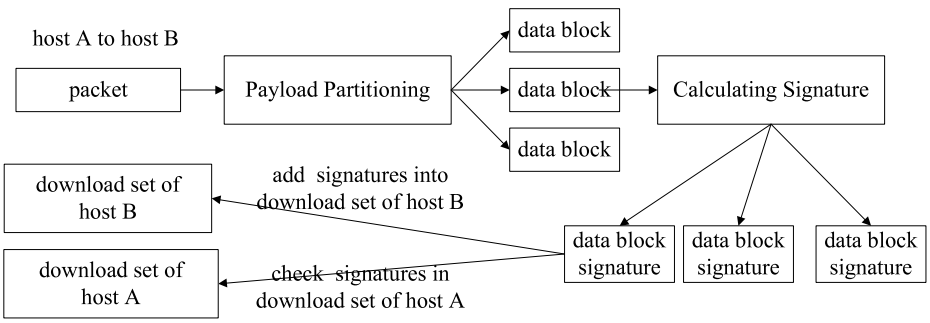

Fig. 4. Online process of the algorithm

The online process of the algorithm is illustrated in Fig. 4. When a packet arrives, it is first partitioned into several data blocks. And then, the signatures of the data blocks are calculated. Finally the signatures are saved into the download set of the destination host and checked in the download set of the source host.

3.3 Practical Problems

Our method is payload based, which focuses on the same data being shared in P2P applications, so it is ineffective in classifying encrypted traffic. Besides, the P2P applications using network coding [18] are also undetectable under our method. We argue that encryption and network coding pose a burden on the P2P application developers, so the mainstreaming P2P applications are still transferring data without any transformation. To identify encrypted traffic, non-payload based methods are more efficient. There are some studies using machine learning algorithms to identify encrypted traffic [19].

The computation and memory overhead are important in identifying P2P traffic online by using our method. The head packet partitioning and head tail packet partitioning have little computation overhead, while the content based partitioning required calculating the Rabin fingerprint which is also very fast. The hash set can be used to check whether a data block have been saved in the download set. In average, there are only several look-up operations for each packet. So the computation overhead is affordable.

Memory is used mainly for two purposes. First, in the payload partitioning, the byte level partitioning schemas have to keep some intermediate information for each flow. For example, to update the fingerprint in the content based partitioning schema, the fingerprint and the content of the last window are saved for each flow. The intermediate information is quite small and 40 bytes are enough to keep the intermediate status. Suppose there are 1M concurrent flows, only 40MB memory is needed.

Second, the signatures and flow identifiers of the data blocks are saved. Suppose the size of a signature is 16 bytes and the size of the flow identifier is 4 bytes (an index in the flow table is used instead of original 5-tuple), 20 bytes are needed to record a data block. If a data block is as large as 512 bytes, a bidirectional 1Gbps link with 50% link utilization can generate at most $1Gb/8/512*20 = 5MB$ records per second. It costs about 300MB of memory for saving 1 minute records and 1.46GB of memory for saving 5 minutes records.

The experiments show that keeping data blocks for several minutes is enough. So the memory consumption is affordable in current hardware capabilities. Besides, because a lot of small data blocks are not saved, the simple estimation causes the results to be upper bounds. Other methods can further reduce memory consumption. For example, flows which are too small or too short are unnecessary to be saved since they are unlikely to be P2P downloading flows.

4 Evaluation

4.1 Experiments Setup

Two metrics are used in the evaluation [20]. The first one is the True Positive (TP). It is used to measure the traffic fraction that can be recognized by the algorithm out of the traffic belonging to the given application.

$$TP = \frac{\text{application traffic classified as the applciation}}{\text{Total applciation traffic}} \tag{1}$$

The second one is the False Positive (FP). It is used to measure the traffic fraction which is not really produced by a given application out of the traffic classified as the application.

$$FP = \frac{\text{Non-applicationtraffic classified as the applciation}}{\text{Total traffic classified as the applciation}} \tag{2}$$

They are very important metrics especially when flow type identification is used in traffic management. For example, if network operators differentiate the service qualities according to the flow types, high false positive or low true positive rates can make high priority traffic suffer from performance degradation. A good algorithm should have low false positive and high true positive rates.

As shown in table 1, a lot of popular P2P applications are evaluated in the experiments. They are classified into three types: P2P file sharing, P2P live streaming and P2P video on demand (VOD).

Table 1. P2P applications

type	applications
P2P file sharing	BitTorrent, Emule
P2P live streaming	PPLive,PPStream,TVAnt,FeiDian,PPMate,SinaLive,TVULive
P2P video on demand	PPLive VOD, XL VOD, PPStream VOD

To evaluate the method, traffic traces are captured and replayed. Two kinds of traffic traces are used in the experiments. The first ones are the pure application traces. The traces are captured from a host which only has the given application running on it. Each P2P file sharing applications trace contains the traffic generated by downloading several files. Each P2P live streaming or P2P VOD applications trace contains the traffic generated by watching a video. The time slot of each trace file varies from half an hour to 1 hour. The traces are only used to evaluate the True Positive.

The second ones are the mixed traces, which are captured from a host running various applications. Each trace contains the traffic generated by a P2P application and some non-P2P applications, such as HTTP, FTP, and POP3 etc. P2P applications in them are labelled by their payload signatures. In each mixed trace, the P2P traffic accounts for 30%-40% of the total traffic. The traces are used to evaluate both the True Positive and the False Positive. we have generated mixed traces with two P2P applications. They are BitTorrent and PPLive.

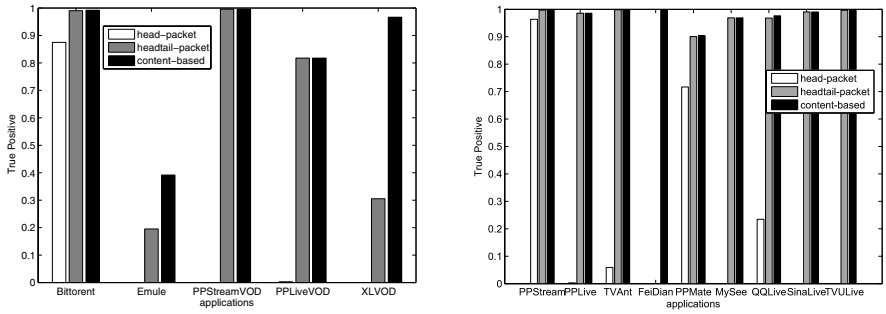

Fig. 5. True Positives

4.2 Comparing Partitioning Schema

First, the True Positive of different partitioning schemas are studied by applying them on the pure application traces. In the experiments, the threshold S is 256 bytes. To study the best True Positive that each schema can achieve, the size of the download set is unlimited.

True Positives are shown in Fig. 5. Among the three partitioning schemas, performance of the head packet partitioning schema is the worst. Quit a few P2P applications can't be recognized by the schema. Performance of the head tail packet partitioning schema is almost as effective as the content based partitioning schema, except for the FeiDian live streaming application. If the applications have protocol fields in both the head and the tail of the packets, the packet level based partitioning schemas don't work. The content based partitioning schema is more generic.

For most applications, the True positives in bytes are more than 90%, but it is only 40% for the Emule application. There are two reasons. First, Emule will upload the files which have been downloaded are older while data uploading in BitTorrent are fresher. Second, Emule transfers a part of a file with 1300 bytes in a separate packet each time to avoid fragmentation. The start position of the part in the file is specified in a request message. The data pieces exchanged in BitTorrent protocol are globally divided, but the data pieces exchanged in the Emule protocol are not. It is more difficult to find the shared data pieces in flows of the Emule application.

4.3 Data Block Size

The threshold S in the content based partitioning schema is important.The effect of threshold S is evaluated in Fig. 6. The size of the download set is unlimited.

The consequences of a smaller threshold S are: the generation of more data blocks; higher memory consumption and increased False Positives. On the other side, the probability of finding the shared data pieces decreases as the threshold S increases.

True Positives of most applications do not decrease significantly when S is smaller than 1024 bytes, except the XLVOD application. It implies that the data

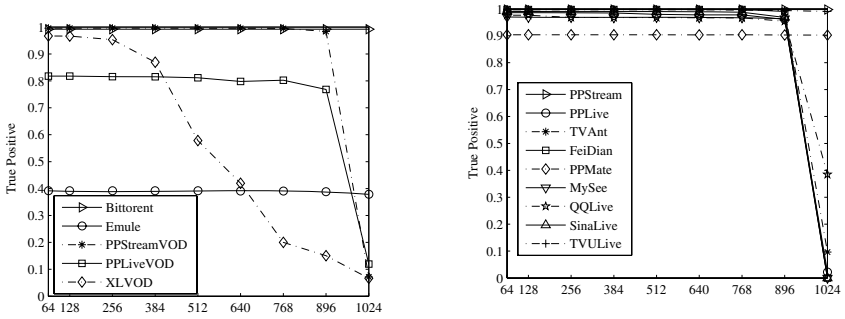

Fig. 6. True Positives of different threshold S

pieces of most P2P applications are not less than 1024 bytes. The threshold S of 256 bytes is suggested in the algorithm and used in the following experiments.

4.4 Download Set Size

The algorithm keeps recent data blocks in a download set for each host. Saving data blocks for an extended period can improve the True Positive, but it requires a lot of memory.

The effect of the download set size is shown in Fig. 7. The threshold S is 256. The size of the download set is measured in time windows. The size of 1 minute means that the download set will keep data blocks in last 1 minute.

Because peers exchange the video content in a small time window in p2p live streaming applications, the True positives of P2P live streaming applications are much better than P2P file sharing and P2P VOD applications for small download sets. For P2P file sharing and P2P VOD applications, downloaded data pieces can be uploaded after a long time. Keeping the most recent data blocks requires a large time window for the applications.

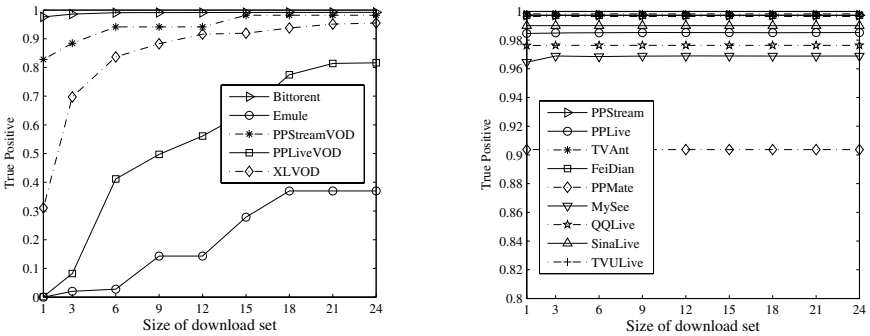

Fig. 7. True Positives of different download set sizes

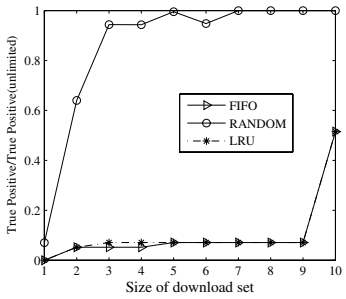

Fig. 8. True Positives for Emule

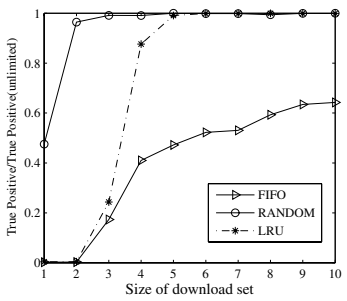

Fig. 9. True Positives for PPLive VOD

The performance can be further improved using other replacement policies instead of the current first in and first out(FIFO) policy. There are two other data block replacement policies to consider. They are least recently used (LRU) and random replacement (RANDOM).

We have evaluate the performance of all the P2P applications. Limited by the space, only the result of Emule and PPLive VOD are shown in Fig. 8 and Fig. 9. The x-axis is the size of the download set and the y-axis is the normalized True Positive. It is calculated by dividing the current True Positive by the True Positive of the unlimited download set. If the value is 1, it implies that the performance of a small download set is as good as the one using the unlimited download set.

The experiment results indicate that 1 minute time window is large enough for P2P live streaming applications, each of the three policies work almost the same for them. But for P2P file sharing and P2P VOD applications, the random replacement algorithm works much better than the other algorithms. A download set of a 3 minutes time window is enough for all P2P applications using random replacement policy.

The idea behind the random replacement policy is that, old data blocks are more likely to be kept by the algorithm than other algorithms. Keeping the old data blocks can improve True Positives for P2P file sharing and P2P VOD applications. Besides, the algorithm has a positive bias to big flows, which can also improve True Positives in bytes. A flow transferring more traffic has a higher probability to be recorded and identified in random replacement.

4.5 False Positive

The mixed traces are used to evaluate False Positives for BitTorrent and PPLive. The size of the download set is unlimited in the experiment. The results are shown in Fig. 10. The x-axis is the threshold S while the left y-axis is the False Positive and the right y-axis is the True Positive.

As the result, a threshold of 256 bytes can help to guarantee a low False Positive while non-P2P applications are transferring random data. But some behaviors of non-P2P applications may lead to false positives. For example,

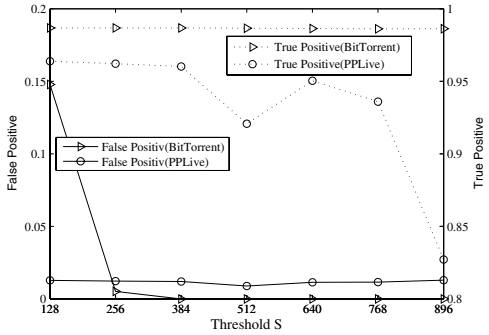

Fig. 10. False Positives and True Positives for BitTorrent and PPLive

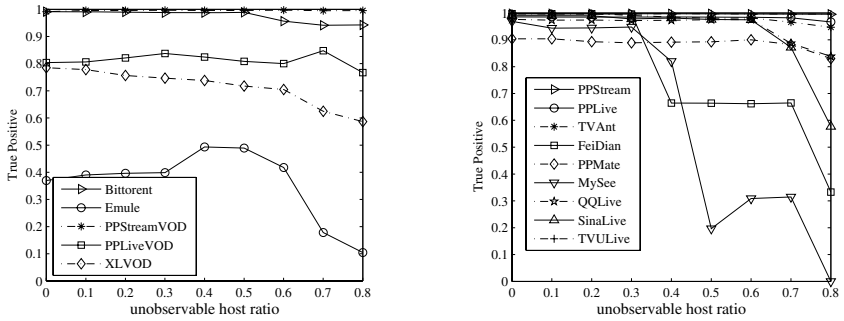

Fig. 11. True Positives of different unobservable host ratios

people forward email they received. Some methods can be used to eliminate these false positives [11]. Since a P2P host always have a service port, we can further identify the P2P {IP, port} pairs which associates with many identified flows. The flows which are not associated with a P2P {IP, port} pairs are filtered to eliminate false positives.

4.6 Deploying Place

In the previous experiments, all the communications of the host can be observed and analysed. When deploying the algorithm in the gateway of an institute or an edge router, the communications inside the intra network are unobservable. The performance of absent communications is studied in the experiment. The threshold S is 256 byte, the size of the download set is 3 minutes and the random replacement policy is used in the experiment.

The results are shown in Fig. 11. The x-axis is the fraction of hosts which are unobservable. For example, 0.2 means that flows between the monitored host and 20% of the other hosts are filtered. The missed hosts are selected randomly and the flows are removed from the original trace. The y-axis is the True Positive on the filtered trace.

The results indicate that even though a large fraction of hosts (about 30%) are not observed, the algorithm can still achieve a high True Positive. It also implies that the algorithm can work well even when the deploying place is not close to the hosts being inspected, such as the gateways of large institutes and edge routers.

5 Conclusion

The paper proposed the content based partitioning schema to identify the P2P data transfer behavior. The schema is generic in identifying P2P applications. Besides, some important issues are studied by experiments. The experiments show that the method can achieve a high True positive and a low False Positive while only keeping a rather small data block set with random replacement policy.

Our future work is to extend our approach to distinguish different P2P application flows by using their relationships in data exchange and flow properties.

References

1. Moore, A., Papagiannaki, K.: Toward the Accurate Identification of Network Applications. In: Passive and Active Measurements Workshop, Boston, MA, USA, March 31- April 1 (2005)
2. Sen, S., Spatscheck, O., Wang, D.: Accurate, Scalable In-Network Identification of P2P Traffic Using Application Signatures. In: Proc. of ACM WWW 2004 (2004)
3. Haffner, P., Sen, S., Spatscheck, O., Wang, D.: ACAS: automated construction of application signatures. In: Proceedings of the 2005 ACM SIGCOMM Workshop on Mining Network Data 2005, pp. 197–202. ACM, New York (2005)
4. Ma, J., Levchenko, K., Kreibich, C., Savage, S., Voelker, G.M.: Unexpected means of protocol inference. In: Proceedings of the 6th ACM SIGCOMM Conference on internet Measurement 2006, pp. 313–326. ACM, New York (2006)
5. Erman, J., Mahanti, A., Arlitt, M., Cohen, I., Williamson, C.: Semi-supervised network traffic classification. In: Proceedings of the 2007 ACM SIGMETRICS (2007)
6. McGregor, A., Hall, M., Lorier, P., Brunskill, J.: Flow Clustering Using Machine Learning Techniques. In: Barakat, C., Pratt, I. (eds.) PAM 2004. LNCS, vol. 3015, pp. 205–214. Springer, Heidelberg (2004)
7. Zander, S., Nguyen, T., Armitage, G.: Self-Learning IP Traffic Classification Based on Statistical Flow Characteristics. In: Dovrolis, C. (ed.) PAM 2005. LNCS, vol. 3431, pp. 325–328. Springer, Heidelberg (2005)
8. Moore, A.W., Zuev, D.: Internet Traffic Classification Using Bayesian Analysis Techniques. In: ACM SIGMETRICS (2005)
9. Williams, N., Zander, S., Armitages, G.: A Preliminary Performance Comparison of Five Machine Learning Algorithms for Practical IP Traffic Flow Classification. SIGCOMM Computer Communications Review, 36(5), 5–16 (2006)
10. Lu, X., Duan, H., Li, X.: Identification of P2P traffic based on the content redistribution characteristic. In: Communications and Information Technologies, 2007. ISCIT 2007, pp. 596–601 (2007)
11. Karagiannis, T., Broido, A., Faloutsos, M.: Transport Layer Identification of P2P Traffic IMC (2004)

12. Karagiannis, T., Papagiannaki, K., Faloutsos, M.: BLINC: Multilevel Traffic Classification in the Dark. In: ACM SIGCOMM, Philadelphia, PA (August 2005)
13. Hu, Y., Chiu, D.M., Lui, J.C.S.: Application Identification based on Network Behavioral Profiles. In: IEEE IWQoS (2008)
14. Muthitacharoen, A., Chen, B., Mazieres, D.: A low-bandwidth network file system. In: Proceedings ofthe 18th ACM Symposium on Operating SystemsPrinciples (SOSP 2001), Banff, Canada, pp. 174–187 (2001)
15. Kim, H., Karp, B.: Autograph: Toward automatic distributed worm signature detection. In: Proc. of the USENIX Security Symp. Diego, pp. 271–286 (2004)
16. Rabin, M.O.: Fingerprinting by Random Polynomials. Tech. Rep.TR-15-81, Center for Research in Computing Technology, Harvard University (1981)
17. Broder, A.Z.: Some applications of Rabin's fingerprinting method. In: Capocelli, R., De Santis, A., Vaccaro, U. (eds.) Sequences II: Methods in Communications, Security, and Computer Science, pp. 143–152. Springer, New York (1993)
18. Gkantsidis, C., Miller, J., Rodriguez, P.: Comprehensive view of a live network coding P2P system. In: Proceedings of the 6th ACM SIGCOMM Conference on internet Measurement, IMC 2006, pp. 177–188. ACM, New York (2006)
19. Bonfiglio, D., Mellia, M., Meo, M., Rossi, D., Tofanelli, P.: Revealing skype traffic: when randomness plays with you. SIGCOMM Comput. Commun. Rev. 37(4), 37–48 (2007)
20. Salgarelli, L., Gringoli, F., Karagiannis, T.: Comparing traffic classifiers. SIGCOMM Comput. Commun. Rev. 37(3), 65–68 (2007)