HMC: A Novel Mechanism for Identifying Encrypted P2P Thunder Traffic

Chenglong Li* and Yibo Xue
Department of Computer Science & Technology,
Research Institute of Information Technology (RIIT),
Tsinghua University, Beijing, China.
{li-cl07}@mails.tsinghua.edu.cn
{yiboxue}@tsinghua.edu.cn

Yingfei Dong
Dept of Electrical Engineering,
University of Hawaii,
Honolulu, HI 96822.
{yingfei}@hawaii.edu

***Abstract*—Thunder (also called Xunlei) is the most popular P2P file sharing application in China and probably the most popular P2P software in term of traffic volume and number of users. Precisely identifying Thunder traffic can help network administrators to efficiently manage their networks. Traditional methods of identifying P2P traffic such as port-based or content-based approaches are ineffective to Thunder traffic, because of its dynamic packet format, flexible port numbers, and payload encryption. In this paper, we developed a novel Heuristic Message Clustering approach (HMC) to identify Thunder traffic, and obtain its state machine and key transaction cycles , thus identifying Thunder traffic fast and accurately. We first evaluate our method in a controlled environment and then with real campus traces. The results show that HMC is able to identify Thunder flows with high precision and low error rate. We will further investigate how to extend the proposed method extended to other applications with unknown protocols and dynamic formats as well.**

Keywords: P2P; Thunder; Traffic Identification;HCM

I. INTRODUCTION

P2P applications have consumed the largest portion of global network bandwidth in the recent years. As reported in [1], the percentage of P2P traffic on the Internet keeping over 50% during 2006 to 2008. Thunder (Xunlei) is one of the most popular P2P file sharing tools in China. It is claimed that the total number of Thunder users is about 3,290 million and active users per month are over 1,660 million now [4]. Thunder is growing rapidly due to its extraordinary fast download speed. Furthermore, Thunder supports BT and eMule formats for file sharing. Thunder describes itself as a Peer to Server and Peer (P2SP) system, which integrates isolated server resources and extensive user resources together to provide stable and fast file sharing services. Although such a P2SP system is fundamentally similar to other P2P file sharing systems, the wide acceptance of Thunder shows the effectiveness of peer-assistance in distributed file sharing.

Identifying Thunder traffic has significant impacts because Thunder causes a few severe issues in spite of its spectacular popularity. (1) *Network Management* — ISPs are highly concerned about Thunder traffic, especially in China. Related issues include traffic and performance monitoring and control, traffic classifying strategies, and billing. In particular, Thunder greedily consumes a large amount of bandwidth, hurting the performance of other applications. (2) Increasing Research Interest. Due to the rapid deployment of Thunder, it attracts more and more interests from both academic and industry. (3) *Spread of pirated copies*. Thunder prompts the spread of pirated software, videos, games, etc by P2P file sharing.

Unfortunately, we still do not have an effective way to identify Thunder traffic. In practice, the most common approach to prohibit the use of Thunder is to block common Thunder ports and main Thunder servers. However, as we discussed in the following, simply blocking ports and main servers is proved ineffective. In this paper, we propose a practical and effective framework to recognize Thunder traffic based on sessions. Through a great amount of experiments, analysis and observations, we have derived a rough structure of Thunder protocol. To deal with the dynamic format of Thunder structure, we have designed a Heuristic Message Clustering (HMC) technique to obtain its state machine and key transaction cycles. We have conducted both experiments and theoretical analysis to evaluate the proposed method.

The main contributions of this paper are: (1) The properties and structure of Thunder protocol are extracted and summarized for practical use. The key cycles of Thunder interactive process are obtained to indicate the internal fundamental actions of Thunder. (2) We have developed a novel and scalable HMC to identify Thunder traffic effectively. (3) The proposed framework is applicable to other proprietary protocol with dynamic ports, dynamic formats, and encrypted payloads.

The paper is organized as follows. We discuss related work in Section 2, and introduce the Thunder work flow and protocol in Section 3. We present our solution in Section 4, combining message clustering technique and heuristic conditions. We report our experiment evaluation results in Section 5 and conclude the paper and present future work in Section 6.

* Mr Chenglong Li is a Ph.D student at Tsinghua Univ.

II. Related Work

P2P traffic identification has been addressed in [2, 7, 9-15]. For the first-generation P2P applications, it is easily classify their traffic based on well-known port numbers. However, the use of arbitrary ports and encrypted payloads make port-based approaches infeasible as reported [12]. Other methods [2, 15] are based on the behaviors of P2P nodes and connections, such as nodes behavior analysis, network diameter measuring, and flow connection pattern detecting, without checking port number and packet payload. These methods are not accurate enough to tell the targeted P2P traffic from other traffic that has similar patterns and behaviors. Another kind of methods [9, 13] is to differentiate P2P traffic through application signatures. Such methods work well on P2P applications that have plaintext format or fixed procedures. Unfortunately, Thunder's payloads are encrypted and its packet format is dynamic. By maintaining network connectivity and clients' state, the signaling behavior of P2P systems [10] can be used to identify their traffic. However, further research is desired to verify whether the signaling behavior can be influenced by the circumstances, such as network conditions or server actions. Flow properties and statistical information are also used to detect P2P traffic in [11, 14, 7]. These approaches help but still are not able to deal with protocol dynamics. A recent work on Thunder [8] discusses its working procedure and service policies. But the protocol structure summarized in [8] is not accurate in all cases. It did not address the issue of identifying Thunder traffic as well.

Our analysis of Thunder protocol exploits the *message clustering* idea in [3]. While in [3], they primarily focus on host-level analysis, we mainly use network-level analysis with some reverse engineering help. Clearly, combining both approaches is a more powerful solution. In this paper, we emphasize gaining understanding at the network level to complement the host-level method, e.g., a host-level method is inapplicable when the code is inaccessible.

III. Thunder Protocol Structure

A. Thunder Work Flow

Thunder supports many types of download and P2P protocols, including HTTP, FTP, BT, Emule and Kad. In [6, 8], the authors have analyzed a portion of Thunder working process. However, the internal structures of Thunder protocol are ignored. To better understand its complex properties, we need to first check its work flow. Our analysis is based on our observations on a large amount of Thunder traffic and expands existing results [6, 8].

A typical working process of Thunder involves three steps: login, idling and file sharing which are shown in Fig. 1.

(1) *Login*. Both TCP and UDP are used in this process. For registered users, a *login* process includes establishing connections to Thunder main servers, and receiving individual options and history information. The main servers include resource servers, advertisement servers, registering servers, news server, multimedia servers, etc.

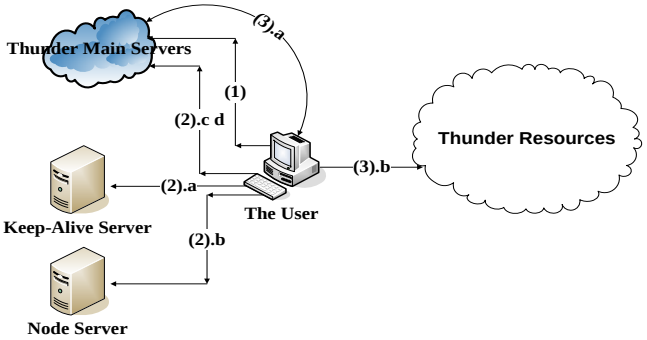


Fig. 1. Thunder Work Flow

(2) *Idling*. A client has four types of interactions with Thunder servers when idling: (a) an ICMP interaction to a keep-alive server, (b) an UDP connection to a node server, (c) an UDP connection to a main server, and (d) an UDP connection to another main server. If there are other Thunder nodes in the same LAN, UDP interactions will be conducted between these nodes. Each interaction has its respective interval.

(3) *File sharing*. When sharing files, a client first establishes TCP connections to resource servers. After receiving replies, the client will share files with multiple Thunder peers via mainly UDP connections and a few TCP connections. So, the Thunder file sharing protocol is only shown in this process and the majority of Thunder traffic belongs to this process. This paper mainly focuses on file sharing traffic.

B. The Protocol Structure

Because Thunder is a proprietary application and its payload is encrypted, the details of Thunder protocol are still unknown up to now. We have observed a great amount of Thunder traffic, and find that a Thunder packet can be divided into two parts: *Thunder Header* and *Thunder Body*, as shown in Fig.2.

- *Thunder Header*. The Header part is mandatory, including *Command* and *Connection*(s). The first 4 bytes of Header is the command part that defines operations. Following the command is the connection part, which indicates node and connection information. By reverse engineering, we find that command part includes more than 300 types of different commands. The fact that the Header part is not encrypted is deduced not only by observation but also by reverse engineering and some native characteristics of P2P application. If it was encrypted, a well-known third party must involve in setting up the shared key between two peers who first meet each other. It will create a single failure point, not scalable to millions of peers, and also hurt performance. Attacking the third party may degrade or disable all sharing processes.

- *Thunder Body*. The Body part is optional, including the payload of sharing data in encryption. This means that a Thunder session may include only headers without data exchange, e.g., because of failures to fetch resources.

Header (Mandatory)		Body (Optional)
Command	Connection	Data
4B	Indeterminate	Indeterminate

Fig. 2. The Thunder Protocol Structure

C. *Properties of Thunder Protocol*

Based on our observation and analysis of Thunder protocol, there are two main challenges to identify Thunder traffic:

- *The dynamic format of Thunder protocol*. Commands and Thunder headers are difficult to determine. A command has a fixed size of 4 bytes; however, the length of the Connection part is variable, ranging from dozens of bytes to over one hundred bytes, making the Thunder Header fairly dynamic. Fig.3 shows most frequently header sizes from a real traffic trace. Moreover, an individual Thunder packet may include header only, or data only, or both, it would still in demand to verify which packet includes Thunder header.
- *The protection of traffic*. The body part is encrypted, so payload-based methods are ineffective to identify Thunder traffic.

These difficulties make existed solutions ineffective when facing Thunder traffic. However, despite its dynamic structure, there are still several useful heuristic conditions that can recognize Thunder protocol and differentiate header and body part, described in the following.

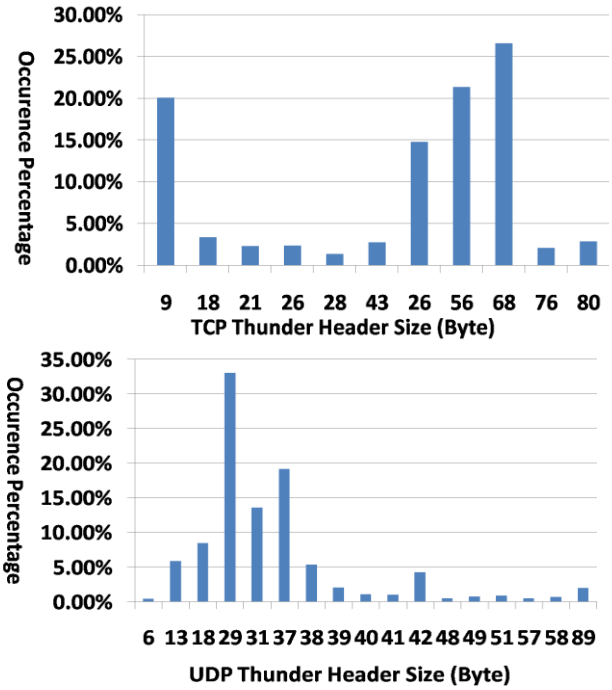


Fig. 3. Most Common Header Sizes.

1. The Headers are not encrypted, but all data parts are encrypted.
2. There are relatively more 0x00s in the Connection part, especially two or three continuous 0x00s, based on our collected data.
3. Many headers end with three continuous 0x00s, or a string with a certain length after two or three continuous 0x00s.

We can determine the Header part and the Data part in a packet based on these heuristics. Feature 1 tells the main difference between the header and the body, such that we can apply a randomness test to determine Thunder headers and bodies. Feature 2 and 3 are special distribution characteristics of headers, which are used for further confirming Thunder headers.

Here, we introduce a randomness test on feature 1. The chi-square test [5] is proposed to judge randomness of data based on the idea that encrypted data is always stochastic. If a packet contains both random and nonrandom data, the rough boundary between its header and body can be obtained through such randomness test. In addition, feature 2 and 3 are utilized to separate a header and a body.

IV. Proposed Solution

In this section, we describe a practical method to recognize Thunder traffic. To handle the challenges of identifying Thunder traffic as discussed in Section III.C, we use message clustering [3] to obtain key interaction cycles of Thunder traffic, then integrate the state machine and heuristic conditions to deal with the dynamics of Thunder protocol.

A. *Message Clustering*

To deal with the difficulties mentioned in Section III.B, we try to find out internal and essential characteristics of Thunder traffic. Inspired by the ideas in [3], message clustering technique is used to handle target traffic.

We analyze Thunder traffic and protocol based on sessions instead of packets. In a Thunder session, several properties can be derived and used for message analysis. First, we need to define the message which split Thunder sessions to smaller steps. A Thunder message is the practical embodiment of Thunder structure presented in the above: a message starts with a Thunder Header and ends before the next Thunder Header. Several other features of Thunder messages including command, header size, and data condition, etc., can also be extracted to cluster similar messages together to discover

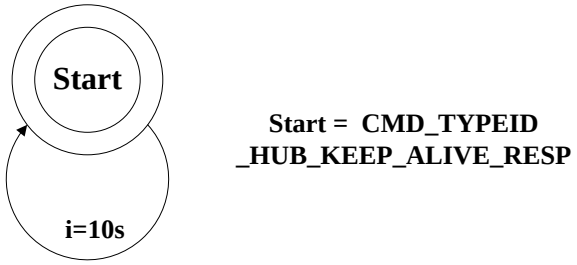
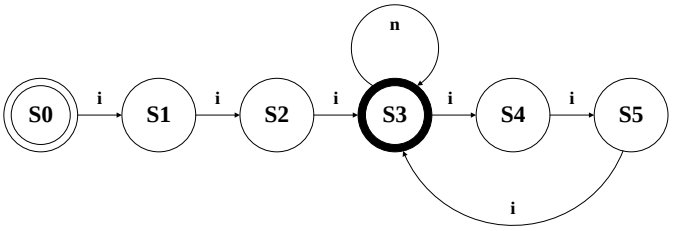


Fig. 4. The SMI diagram



S0 = cmd_query_p2phub, S1= cmd_query_p2phub_resp,
S2= cmd_request, S3= cmd_request_resp,
S4= cmd_query_tracker, S5= cmd_query_tracker_resp.

Fig. 5. The SMS diagram

internal properties of Thunder protocol.

Through a great deal of analysis to Thunder traffic by message clustering technique, two key interaction cycles are clustered: one is for idling process and the other is for file sharing process, which represent two main working steps of Thunder. In these two key cycles, there are key commands indicating interactive operations. Though there are quite a lot of command strings, the number of key command strings is relatively easy to verify and collect. All Thunder traffic of current versions during idling and file sharing observe these two key cycles.

To express key cycles with key commands, we employ two simple state machines SMI (state machine for idling, SMI) and SMS (state machine for sharing, SMS). In these machines, states are embodied by key command names, which are obtained from reverse engineering to Thunder applications. We will introduce these two state machines in detail in the following.

B. SMI for Idling

Firstly, the key cycle of UDP interactions between nodes in idling process SMI is described as shown in Fig.4. In SMI, there is only one state whose name is CMD_TYPEID _KEEP_ALIVE_RESP, which is both the start state and the accept state. From the observation to Thunder traffic, we collect dozens of strings for this command including 0x32000000, 0xbc010000, 0x2c0b0000, and etc. The length of Thunder headers containing these command strings ranges from 29 to 57. The only element of transition condition *i* is used to represent the time interval. As shown in the figure, the time interval is fixed as 10 seconds. Moreover, all messages in this cycle have only Thunder headers without body parts.

C. SMS for File Sharing

We apply the same idea of message clustering technique to analyze the structure for other complicated transactions. We obtain , the state machine SMS of Thunder file sharing process as shown in Fig.5. In SMS, possible command strings are too much to list, so we only point out its key commands. The start state is S_0 and the accept state is S_3. States $\{S_0, S_1, S_2\}$ are requests to the resource provider or another Thunder peer which has the requesting resource. State S_3 is downloading state. States $\{S_4, S_5\}$ aim to find more resources from provider peers. The transition condition *i* represents indeterminate time interval, and *n* is the number of loops which is not fixed. The

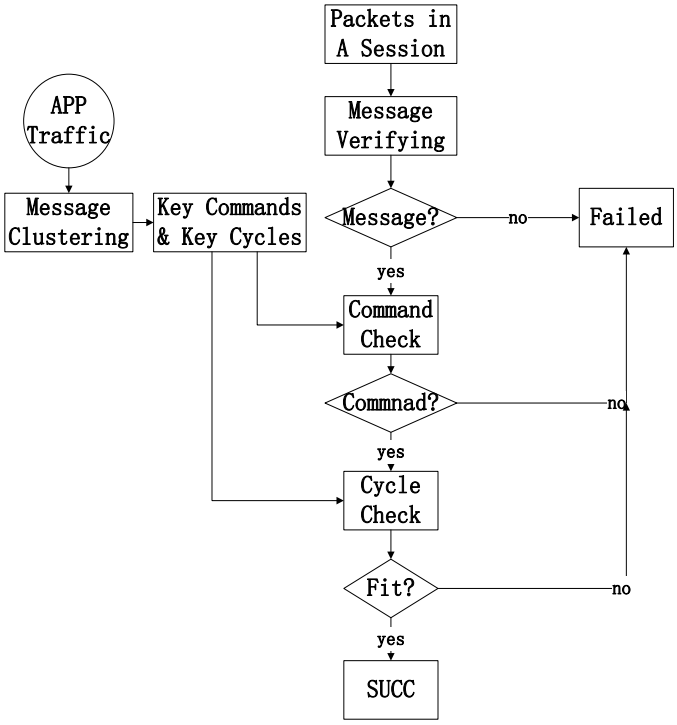


Fig. 6. The HMC Framework

reason why S_4 *and* S_5 exist is that Thunder intends to sharing file with providers as much as possible.

Note that command strings may be changed quickly due to new versions of the software. However, key cycles are rarely altered, at least in several existing versions. So when a new version comes out, the method proposed in this paper is still effective by adding new key command strings to the framework described in IV.D.

Message clustering is not only useful for protocol analysis, but also simplifies the traffic identification. When a key Thunder message is verified, the whole session that may include hundreds of messages is verified. The inspected packets only occupies a small part of the session. Thus to identify Thunder traffic becomes to verify key Thunder messages, in spite of dynamic Header sizes and formats.

D. The HMC Solution

In this section, we present the HMC framework to solve not only Thunder classification but also other protocols with dynamic format, dynamic ports, and encrypted payloads. The HMC framework uses the same techniques mentioned above, including message clustering, heuristic conditions, etc, as shown in Fig. 6.

This framework is obtained from Thunder related work but designed for more flexible circumstances. In our small-scale tests, HCM is effective to many different types of protocols, including standard protocols, P2P protocols and some unpublished protocols whose traffic can be divided into similar messages.

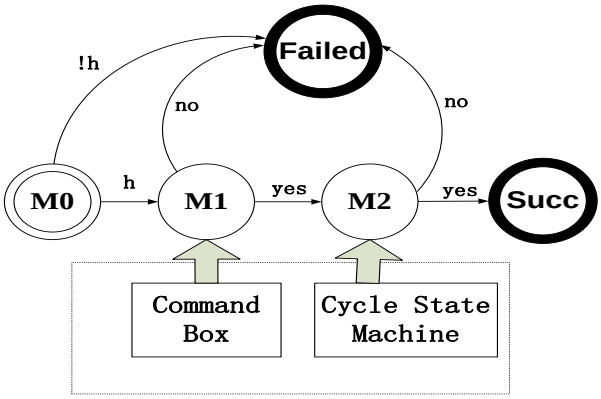


Fig. 7. The state diagram for M

With the HMC framework, we introduce a concrete state machine system to recognize Thunder traffic specifically. We define the state machine M as shown in Fig. 7.

Because M is used in practical classification, the objective of M (except Key Cycles) is consecutive messages composed by packets in a session. In the dotted line box, there are command box and cycle state machine mentioned in IV.B and IV.C. The start state M_0 checks several starting messages in a session. Due to the structure of Thunder's protocol and the definition of Thunder message, the first packet of a message must contain a Thunder header. So in M, the first option is based on header check which mainly depends on heuristic conditions introduced in III.C. The element *h* in alphabet indicates header check passed, and *!h* means failed. After these messages passing through header check, then key commands and key cycles (e.g. SMI and SMS) will be picked up here to check if these messages follow Thunder interactive processes. In the Command box, there are command strings collected from existing Thunder versions. With the previous header check, the maximal accuracy can be guaranteed.

At the same time, the work flow is concise as well. After a round of key cycle examination, the final result can be known. If passed key cycles in M, the session can be classified as Thunder traffic, otherwise it failed. In general, only first 10 to 20 packets in a session need to be checked to get the result, therefore the whole solution is fast as well as precise which will be proved by practical experiments in the next section.

The performance of the HMC framework is also essential in practice. The algorithm complexity of the framework (M) will be calculated here. Assume there are *n* sessions in the traffic. And each session need check first *A* messages where *A* can be set as a constant number (for example *A=10*). Each message need to check first *B* bytes (in M *B=4*) as command. Then HCM need to compare real messages with given key cycles, and in the worst case, needing A times comparisons. So the algorithmic complexity of M (HCM as well) is $O(n)= ABn$ because *A* and *B* are constant numbers. The space complexity of HCM depends on the numbers of commands and cycles. In M there are only two cycles so the majority part is the size of the command box.

The framework proposed has three main advantages: effective, fast and flexible.

TABLE 1 Results of LAN Dataset

	Number	FP	FP%	FN	FN%
Accurate	51746	—	—	—	—
HMC	51123	98	0.19	721	1.39
Total	603173	—	—	—	—

V. EXPERIMENTS AND VALUATIONS

In this section, we present our experimental results in order to test the proposed method on different traffic traces. In particular, we choose two different datasets:

- LAN: refers to a 48-hour long trace collected from a LAN connecting to Unicom , a major ISP in China, in May 2009. This dataset is relative small but under our complete control. We recorded the use of Thunder on each computer in the LAN. During the capturing period, different users connected to Internet through the LAN and varies kinds of applications were kept online and in use. Thus we can get accurate records for reference.
- EDU: refers to a 1-hour long trace collected from our campus access link in May 2009. This dataset is more complex. We choose this dataset for the purpose of not only verifying our algorithms but also providing real usage of Thunder in China.

A. Measurement Results of LAN

The measurement results of dataset LAN are shown in Table 1, the actual result acting as the contrast to the results from our method introduced in the previous section. The False positive (FP) and false negative (FN) are counted in the number of flows. The FP rate is defined as the number of false positive flows over the actual number of Thunder flows we started. The FN rate is defined as the number of false negative flows over the actual number of Thunder flows we started.

The result is varied with different degrees of heuristic conditions and randomness test. When choosing a wide range of heuristic conditions, we have a low FN rate but the FP rate increases because other protocols such as IM and anonymous communication applications may have similar structure and encrypted payloads as well. However, when we set strict rules the FP rate is deduced with an increasing FN rate.

We find an effective setting with many tests. That is, the condition differentiating a Thunder header and a Thunder body should be most stringent; the condition classifying a Thunder header from other protocols should be the second strict condition; the randomness test should be the least rigid one. With this setting, we obtain the desired results as shown in Table 1.

Because there is no standard method to perfectly identify Thunder traffic, we design such a dataset with actual known Thunder sessions to confirm the effectiveness of our method. The results are promising with both FP and FN rates are below 1.5%.

B. Measurement Results of EDU

In order to prove the accuracy of HMC framework further, we design a contrast test called IP-Command whose

results can be a counterpart to the framework. The contrast test uses another classifier to Thunder traffic and obtained as follows: (a) first IP addresses of all Thunder active servers are collected by related domain name lookups; (b) then all nodes that has communication with Thunder servers are gathered and their traffic is recognized; (c) then hot commands used in the framework are matched in the selected traffic and sessions in which hot commands occur multiple times are set as Thunder sessions and contrast results.

The IP-Command test is based on the work flow and traffic analysis of Thunder. Thunder clients, whether registered or not, need login to Thunder servers, including main server, advertisement server, resource server, etc. following the program settings during use. So nodes communicating with these servers cover all nodes that may use Thunder and generate Thunder traffic. Then hot commands are used to identify real Thunder traffic based on the analysis mentioned in Section IV. If commands appear several times at the beginning of packets in a session from selected traffic, it may be a Thunder session in a very strong possibility. The classifier is reasonable to be contrast test to the proposed framework by the behavior check and payload inspection. However it can only be a temporary and limited classification method because servers may change quickly and secretly, and it depends on login step which cannot extend freely to other similar protocols.

To compare results between the framework and the contrast classifier, we choose EDU dataset because real traffic can perform more reliable and believable results. Table 2 shows the results of the experiments.

The results are obtained by treating IP-Command method as accurate, so FP and FN are false positive and false negative number of HMC method and FP rate and FN rate are FP and FN flow numbers over the IP-Command flow number. In Table 2, the FP and FN rates are quite high compared with the results of LAN dataset. The FP rate is near 10 percent which mainly because that IP-Command neglects the traffic of nodes whose Thunder login processes are not captured in EDU dataset due to time factor or packet loss. The FN rate is near 5 percent because IP-Command uses simple command inspection and may misjudge some other traffic as Thunder's. Though IP-Command has certain defects, the HMC framework show a relatively high accuracy whose FP rate is over 90% and FN rate is over 95%.

Based on the results of the HMC test in Table 2, over one fifth of EDU TCP/UDP traffic and about 5% of TCP/UDP flows belong to Thunder. Because on our campus P2P applications are not prohibited, the usage of Thunder on our campus and in the nation is partly exposed by such experiments. The Thunder traffic occupies quite a large proportion of traffic on campus. However, the use of Thunder is even more popular off campus due to that other main ISPs in China provide more resources and have more end users than education network. Based on reports from [1], about 50% of consumer Internet traffic is P2P traffic and assuming China also has the similar Internet conditions, Thunder may take about half of all P2P traffic in China. Therefore identifying Thunder traffic is even more significant than we think before.

TABLE 2 RESULTS OF EDU DATASET

	Flows	FP	FP%	FN	FN%	Traffic (GB)
HMC	13721	1273	9.79	550	4.23	6.65
IP-Command	12998	-	-	-	-	5.79
Total TCP/UDP	283460	-	-	-	-	30
Proportion (HMC)	4.8%	-	-	-	-	22.2%

VI. CONCLUSION AND FUTURE WORK

The main contribution of this paper is to propose a framework to solve Thunder like protocols with dynamic format and encrypted payloads. The framework can be extended easily to identify many emerging protocols. The proposed framework is one solid step towards this direction.

ACKNOWLEDGMENT

This work was supported by National High-Tech R&D 863 Program of China under grant No. 2007AA01Z468.

REFERENCE

[1] Cisco Systems. "White Paper: Cisco Visual Networking Index: Forecast and Methodology, 2008-2013," June 9, 2009.

[2] T. Karagiannis, K. Papagiannaki, and M. Faloutsos, "Blinc: multilevel traffic classification in the dark," in Procs. of SIGCOMM, 2005.

[3] P. Comparetti, G. Wondracek, C. Kruegel and E. Kirda, "Prospex: Protocol Specification Extraction," In Proc. Of 30th IEEE Symposium on Security and Privacy, 2009.

[4] http://hr.xunlei.com/introduce.html/.

[5] Knuth, D, "The Art of Computer Programming, Volume 2: Seminumerical Algorithms," Addison-Wesley, 1981 (1st ed. 1969).

[6] M. Zhang, C. Chen and N. Brownlee, "A Measurement-Based Study of Xunlei", PAM2009, April 1-3, 2009, Seoul, Korea.

[7] W. Moore , and D. Zuev, "Internet Traffic Classification Using Bayesian Analysis Techniques," in Prof. of ACM SIGMETRICS 2005.

[8] Shuying Chen, "Analysis of Thunder's P2SP Structure and Service Policy," master thesis, BJTU, 2008.

[9] S. Sen, O. Spatscheck, and D. Wang, "Accurate, Scalable In-Network Identification of P2P Traffic Using Application Signatures," *WWW2004*, New York, USA, May 17–22, 2004.

[10] C. Wu, K. Chen, Y. Chang, and C. Lei, "Detecting Peer-to-Peer Activity by Signaling Packet Counting," *ACM SIGCOMM'08*, Seattle, Washington, USA, August 17–22, 2008.

[11] D. Bonfiglio, M. Mellia, D. Rossi, and P. Tofanelli, "Revealing Skype Traffic: when randomness plays with you," *ACM SIGCOMM '07*, Kyoto, Japan, August, 2007.

[12] M. Roughan, S. Sen, O. Spatscheck and N. Duffield, "Class-of-Service Mapping for QoS: A Statistical Signature-based Approach to IP Traffic Classification," Proceedings of the 4th ACM SIGCOMM conference on Internet measurement. New York: ACM Press, 2004: 135-148.

[13] R. Wang, Y. Liu, Y. Yang, and X. Zhou, "Solving the App-Level Classification Problem of P2P Traffic Via Optimized Support Vector Machines," Proc. of the 6th International Conference on Intelligent Systems Design and Applications Jinan, China 2006: 534-539.

[14] M. Crotti, M. Dusi, F. Gringoli, and L. Salgarelli, "Traffic Classification through Simple Statistical Fingerprinting," ACM SIGCOMM Computer Communication Review. New York: ACM Press, 2007: 5-16.

[15] F. Constantinou, and P. Mavrommatis, "Identifying known and unknown peer-to-peer traffic," 5th IEEE International Symposium On Network Computing And Applications, Cambridge, MA USA: IEEE Xplore, 2006: 93-102.