Detection of Peer-to-Peer Nodes based on Query Routing

Nobutaka Kawaguchi, Kazuya Okochi and Masato Terada
Systems Development Laboratory, Hitachi, Ltd.
890, Kashimada, Kawasaki, Kanagawa, 212-8567, Japan
E-mail: {nobutaka.kawaguchi.ue, kazuya.okochi.ak, masato.terada.rd}@hitachi.com

***Abstract*— In this paper, we propose a novel P2P node detection method by analyzing network traffic and extracting packets which contain query messages. Most previous methods detect P2P nodes by using signatures of known applications or taking advantages of traffic features in P2P nodes. However, they cannot detect hosts running unknown P2P applications while keeping low false positive rate. To address the problem, we focus on the resource discovery mechanism where query messages are routed and transmitted through several nodes to locate hosts providing certain files. Then, we attempt to detect hosts that appear to receive and transmit queries with other hosts. To do so, our approach monitors the traffic of targets and searches for pairs of inbound/outbound packets which are likely to contain same queries by computing their similarities. Through evaluation experiments with two popular P2P based file sharing software, LimeWire and Winny, we show this approach detects P2P nodes within a few hundreds of seconds with a few false alerts in a week.**

I. INTRODUCTION

With the increases in computational power and storage capacity of personal computers, P2P networks have been popular applications. At present, some major file sharing networks such as LimeWire [1] and Winny [2] have hundreds of thousands to millions of users spreading over the world.

While P2P networks are promising technologies, however, they also pose several problems. First, many illegal copies of digital media contents such as music, movies and software are exchanged via the networks. Second, various confidential files are exposed and distributed in the networks. For example, there are viruses that expose confidential files stored in their infected hosts to P2P networks and the Internet. In addition, since vast amount of packets are transmitted between P2P nodes, many ISPs are suffering from bandwidth saturation caused by the P2P traffic. Thus, many organizations restrict the use of P2P applications for the security and management costs.

Then, such problems have motivated the development of techniques for identifying hosts running P2P applications. Especially, there have been many methods using traffic analysis. They can be classified into either of misuse detection or behavior based detection. Misuse based detections search for data that represent particular known P2P applications. While the approaches can detect known P2P applications with low false negatives, they cannot detect unknown ones. On the other hand, behavior based detections take advantage of traffic features inherent in P2P applications such as the low success rate of connection establishments to other nodes. Although these approaches could detect unknown P2P applications, however, they render to produce many false alarms.

To achieve the detection of unknown P2P applications while keeping low false positive rate, we propose a new behavior based detection approach called *Query Routing based Detection* (*QRD*, for short). QRD focuses on a file discovery mechanism introduced by many pure P2P networks. When a P2P node searches for a certain file from the network, it sends out a query message to the neighbor nodes to gather the location information of the file. Then, the receiving nodes route and forward the query to other nodes repeatedly. We call the query process performed at each intermediate node as *Query Routing*, and attempt to detect hosts that appear to perform query routing by analyzing the traffic of detection target.

QRD searches for pairs of inbound and outbound packets which are likely to contain the same queries. To do so, QRD computes their similarities, and a pair with high similarities is judged to contain the same query. Then, QRD judges hosts that appear to receive queries from a certain number of hosts and transmit to a certain number of others as P2P nodes. Since only P2P applications perform query routing, QRD can clearly distinguish P2P nodes from non-P2P nodes.

Through the evaluation experiments using traffic logs of two popular applications, LimeWire and Winny, we have confirmed that QRD can detect P2P nodes within hundreds of seconds while keeping a few false positives in a week.

To the best of our knowledge, QRD is the first work that focuses on query routing for P2P node detection.

The rest of this paper is organized as follows. Section 2 reviews related works in P2P node detections. In section 3, query routing mechanisms in major P2P networks are introduced. Then, we propose QRD and describe the procedures in section 4. Then, section 5 shows evaluation experiments of QRD. Section 6 discusses about limitation of QRD. Finally, section 7 discusses future works and concludes this paper.

II. RELATED WORKS

There have been many methods proposed to detect the existence of P2P nodes by analyzing network traffic. The

methods are classified into two approaches; misuse based detection and behavior based detection.

In the first category, misuse based detection takes advantage of detailed knowledge about particular known P2P applications. Signature based approaches [3] compare the payloads of monitored packets against signatures of known P2P applications such as the fixed strings. However, these approaches cannot detect unknown P2P applications or applications encrypting their traffic. Port based approaches [4] recognize hosts that use certain listening ports as P2P nodes. However, recent P2P applications are designed to easily change their listening ports to evade detection. Generally, misuse based approaches cannot detect unknown applications, although they produce few false alarms.

In the second category, behavior based detections focus on the inherent features in P2P applications so that they possibly detect nodes using unknown P2P applications. Connection establishment success rate approach [5] is based on the observation that connection establishment attempts from one node to another are failed with high probability since P2P nodes are occasionally offline and change their IP addresses and listening ports frequently, and therefore address lists of known P2P nodes decay quickly. With this approach, however, hosts that run address scanners can be falsely identified. In addition, this approach does not detect applications that use UDP flows instead of TCP connections. Connection direction based detection [5] computes the ratio of number of incoming connection attempts to the total of number of incoming and outgoing connection attempts, and a host with the ratio within 0.1-0.8 is recognized as P2P node. With this method, servers that frequently open connections to other hosts (e.g. mail servers) and hosts that run both the client applications (e.g. web browsers) and server applications (e.g. web servers) are falsely recognized. Chain based detection [6] recognizes a group of hosts which are chained with TCP connections as P2P nodes when the chains become longer than a threshold. However, it requires monitoring large IP address space and may recognize ssh/telnet-chains as P2P nodes. Distribution of packet length can be used for detection [7] since P2P applications tend to send small fixed-length packets for transmitting several commands. This approach is similar to ours in that both are based on the length of command packets. The difference is that we especially focus on the query command and its transmission feature.

As mentioned above, behavior based approaches usually suffer from many false positives since it is not easy to extract traffic features which can clearly distinguish P2P applications from the other applications. Although combination of some features could reduce false alarms, it may have adverse effects on the detection speed.

III. QUERY ROUTING

A. Overview

P2P networks have mechanisms for their nodes to search for vast amount of files provided by other nodes. Early P2P applications such as Napster have directory servers that manage location information of files. When applications are launched, they access to the servers to register their providing files. Then, requester nodes address the location of files by sending queries to the servers. However, the directory server based approach is not scalable against the network size and can be a single point of failure. For the reason, many recent pure P2P networks introduce server-less distributed file search mechanisms in which all or a portion of nodes in the network are involved.

In the distributed mechanism, each node has some neighbor nodes with which TCP connections or UDP flows are established. Then, when a node searches for a certain file, it sends out a command called query message to its neighbor nodes. A query message contains keywords of the searched file. Nodes that receive the query message route and forward it to others according to the contents of the query. Then, the query is repeatedly routed and forwarded through several nodes. If a receiving node knows the addresses of nodes that provide the file, it sends back the file location information to the requester node. The query transmission is terminated when conditions specified in the query message are satisfied. The process of routing and forwarding query message at each intermediate node is referred to as query routing.

Figure.1 shows a typical example where node X and C perform query routing. First, node A, which searches for the “File F”, sends out a query message. The message has two fields; file name field and TTL field, which specifies the number of times the query can be forwarded. In this case, the field is set to 3. On receiving the query from node A, node X decrements the TTL by 1, and forwards it to node B and C, which are X’s neighbor nodes. Then, since node B has File F, the response message including node B’s IP address is returned to node A though node X. On the other hand, node C further forwards the query to node D. In the end, the TTL becomes 0 at node D, and then node D does not forward it further and just discards it.

In practice, there are various implementations of query routing. In early versions of Gnutella, all nodes are recognized as homogenous and randomly connect to others. Every node performs query routing. At each node, a query is forwarded to all its neighbor nodes. Thus, a file search request can generate many copies of a query message.

In later versions of Gnutella and Winny, nodes are recognized as heterogamous, and develop hierarchical structures. Each node has some parent nodes and child nodes. Then, queries are sent up from child nodes to parent nodes. In this approach, hosts equipped with broad bandwidth and high computation powers are positioned at higher layers. This approach achieves more effective search performance than previous one since queries are sent only to the higher part. Query routing is performed at nodes that have both parent and child nodes.

In the following sections, we analyze the features of query routing on LimeWire and Winny.

B. Query routing on LimeWire

LimeWire is a pure P2P application based on Gnutella protocol, which is a popular P2P protocol implemented by

various applications.

Figure.2 shows the topology of LimeWire network, which is a two-tier architecture. Nodes operate on the network in one of two roles; Ultrapeer and Leafpeer. An ultrapeer is connected to several other ultrapeers and leafpeers, and is responsible for forwarding messages received by its leafpeers to other nodes. Leafpeers are connected to a few ultrapeers. Thus, only the ultrapeers perform query routing.

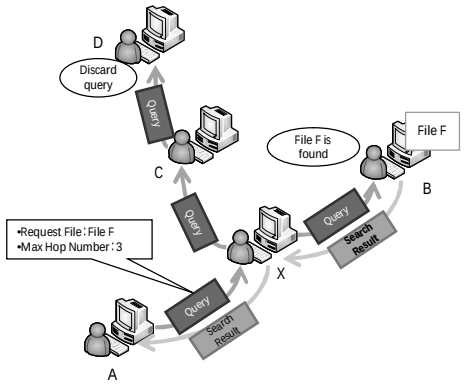


Figure.1 Example of Query Routing

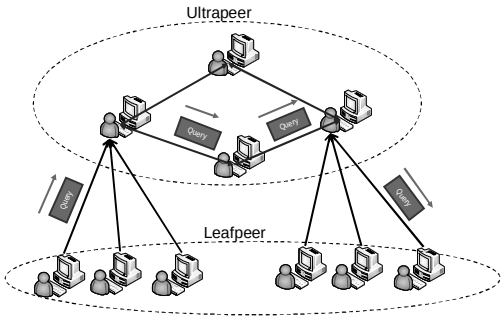


Figure.2 Topology of LimeWire Network

In this network, leafpeers periodically send information about their providing files to the ultrapeers. Ultrapeers also exchange the file location information among other ultrapeers. Then, when a leafpeer issues a query message, the message is firstly forwarded to its ultrapeers. On receiving the message, ultrapeer searches file location information obtained from its leafpeers. Then, if no leafpeers have files matching to the query, it forwards the message to its neighbor peers with TTL=1-3. Then, ultrapeers receiving the message route and forward to another ultrapeers.

Table.1 shows the message formant of Gnutella Protocol. Payload Descriptor of query message is set to 0x80. File names and keywords are put in search criteria. During transition, query contents are not modified except the TTL field decreases. Thus, query message length does not change during transmission.

When forwarding a query, node records the query's Descriptor ID and destination node IP. If a node that receives the query knows the location of the file described in the query, it returns a response message with Payload Descriptor=0x81. The response is transmitted in reverse direction to the requester node. The intermediate nodes use the records to find the destination of the response message.

Next, we measure one hour traffic of an Ultrapeer node while its TLS based encryption and data compression options are turned off. Figure.2 shows the distribution of query message length. The average is 57 byte. Since LimeWire supports advanced search criteria using XML, some messages are relatively long, exceeding 100 byte.

Figure.4 shows the distribution of time spent for routing and forwarding a query. This time is a interval between when a query is received and when the corresponding query is forwarded. 50% of queries are forwarded within 100 msec. Moreover, more than 10% of queries are processed within only 1 msec.

Table.1 Message Format of Gnutella Protocol

Length (byte)	Field
16	Descriptor ID
1	Payload Descriptor (0x80:Query)
1	TTL
1	Hops
4	Payload Length
1	Minimum Speed
Variable	Search criteria

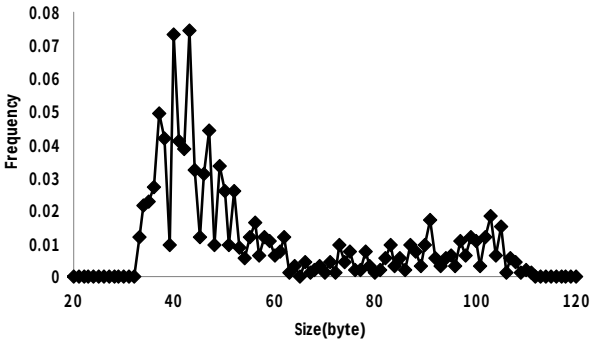


Figure3 Distribution of query message length

C. Query routing on Winny

Winny is another popular pure P2P application, especially in Japan. Different from LimeWire nodes, Winny nodes develop a typical tree-like structure. Every node except root node has a few parent nodes. Queries are sent up from child to parent nodes. Thus, all nodes except leaf and root nodes perform query routing.

Table.2 shows the query message format of Winny protocol. Each time a query is forwarded, the forwarder node adds its IP address and listening port number to the query, and therefore the query length is increased by 6 byte. When the query is forwarded 6 times or more than 30 file location information is found, the corresponding response is returned to the requester via routes recorded in the query. Like LimeWire, Winny encrypts all messages. Since they use

stream encryption algorithms such as RC4, the packet length is not changed by the encryption.

Winny supports a search option called hash based search. This option enables a requester to search a file using the file's 32 bytes hash. This option is frequently used since it is effective in finding the exact file that the requester demands. With this option, the query keyword field length is 33 byte (1 byte flag to indicate the hash based search is added). Thus, a query length at *h*th hop nodes is *61 +6*h* byte. Since the maximum hop number is 6, most queries are no longer than 97 byte unless the queries contain key information.

Figure.5 shows the distribution of interval between when a query is arrived and when it is forwarded to other nodes. From the result, more than 35% of queries appear to be transmitted within 1 msec. Thus, Winny nodes tend to forwards query faster than LimeWire nodes.

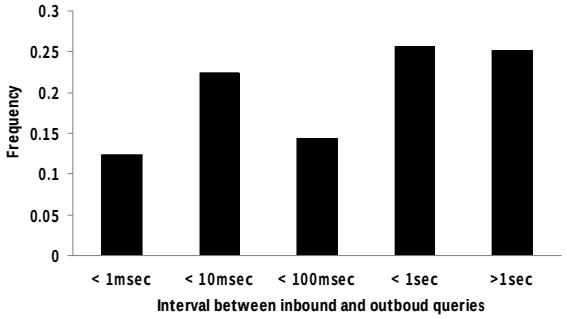


Figure.4 Distribution of time spent for routing and forwarding a query message by Limewire.

Table.2 Query Message Format of Winny Protocol

Length (byte)	Fields
9	Header
4	Query ID
1	Keyword Length
Variable	Query Keyword
11	Query Trip
1	Number of routing nodes (Nr)
6 x Nr	Routing nodes (IP address, service port)
2	Number of file location information
Variable	File location information (added to query when a node knows the file location)

D. Features of Query Messages

The analysis of LimeWire and Winny queries reveals the following features of query messages.

1. Once a query message is arrived, the receiving node immediately routes and forwards it to another node.
2. During the transmission, a few byte length data may be added to the original query.
3. The query message length is relatively short, usually less than a few hundreds byte.

In the next section, we will show a P2P detection method that takes advantage of the features.

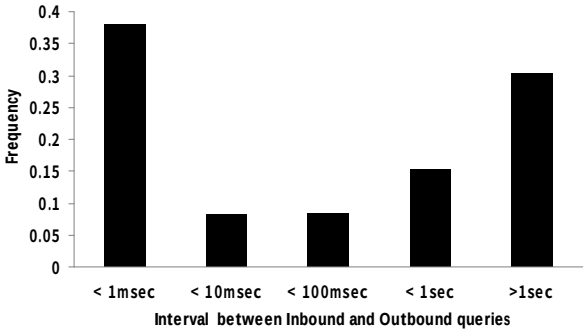


Figure.5 Distribution of time spent for routing and forwarding a query message by Winny

IV. QRD: QUERY ROUTING BASED DETECTION OF P2P NODES

A. Concept

Our P2P nodes detection approach, called QRD, detects P2P nodes by finding hosts that appear to perform query routing.

QRD is supposed to be deployed near the targets so that it can capture all the incoming/outgoing packets and measure when the packets are sent or received precisely. For example, QRD can be installed in target host, L2/L3 switches or gateway routers as security modules.

QRD detection process takes the following three phases.

1. QRD captures packets to/from the target host. Then, Packets in a flow are divided into subgroups called *Chunks* according to the captured times.
2. Inbound chunks in inbound flows and outbound chunks in outbound flows are compared to find pairs of inbound/outbound chunks that contain the same query message. This process is called *pairing check*.
3. Finally, if a flow that appears to forward queries received from more than *TH* nodes is found, the flow is judged as a *routing-flow*. Then, when the number of routing-flows exceeds *Nf* in a period of *Tf*, the target is judged as a routing node.

In the following subsections, we will describe each phase.

B. Phase 1: Packet grouping into packets

As a preprocessing step for detection, QRD discards a captured packet if the packet satisfies any of the following conditions.

1. The packet does not contain any payload.
2. The packet is an outgoing packet and the target host has received any packets from the destination host within a period of T_{recv}.

3. The packet is an incoming packet and the target host has sent any packets to the source host within a period of T_{send}.

Condition 1 filters packets that do not contain query messages. Condition 2 and 3 are introduced to filter packets which are considered to be response from/to foreign hosts.

Next, the remaining packets are grouped into chunks. Figure.6 shows how packets in a flow are grouped into chunks. If the difference of captured times of two successive packets in a flow is within an interval *PI* (e.g. 100msec), they are grouped into same chunk, otherwise, different chunks. Each chunk includes more than one packet. This process is performed for dividing packet into messages. An entire query message will be in a single chunk if the grouping is properly done. In addition, since most query messages are no more than a few hundreds byte as described in Section 3, a chunk of more than 1k byte is judged as not containing a query message, and then discarded.

Although most parts of queries in known P2P applications are contained in single packets, we introduce a notion of chunk to prevent an evasion technique that divides a message into several packets no to be detected by per-packet based sensors. In addition, this approach can distinguish a packet that is a part of successive packets containing non-query data from a same length packet containing an entire query, and therefore reduce false positives.

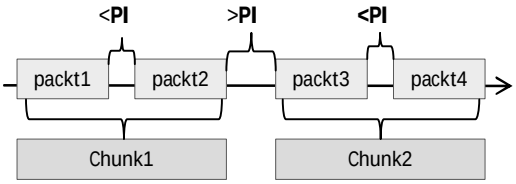


Figure.6 Packets grouping into Chunks

C. Phase 2: Chunk pairing check

In the pairing check, a pair of inbound chunk C_{in} and outbound chunk C_{out} is judged as containing the same query if they satisfy the both of following conditions, which are the features of query message as described in Section 3.

1. The last packet of Cin is captured before the first packet of Cout is captured, and the time difference is less than a time *QI* (e.g. 1msec).
2. The size of Cout is equal or larger than that of Cin, and the difference is less than *QS* (e.g. 20 bytes).

Chunks that satisfy the condition above are called *query-paired* chunks, and chunks are query-paired with each other.

On the operation of QRD, when an outbound chunk is generated or updated by adding a new outbound packet, it is compared with inbound chunks which have not been query-paired with any outbound chunks. An outbound chunk can be query-paired with just one inbound chunk. If there are some inbound chunks satisfying the conditions, one with the minimum difference of captured time is chosen to be query-paired.

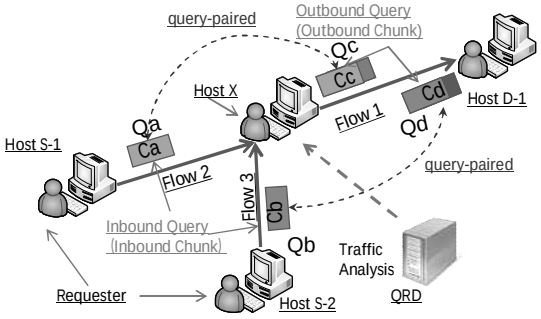
D. Phase 3: Checking paired outbound chunks in each flow

If the target host an outbound flow whose outbound chunks are query-paired with inbound chunks received from more than *TH* unique hosts in total, QRD judges the flow as a routing-flow. Finally, when the number of detected routing-flows within a period of *Tf* exceeds *Nf*, the target is recognized as a query routing node. This approach is based on the observation that a query routing node is usually connected to several nodes, and therefore there will be several outbound flows forwarding queries received from several other nodes.

This feature is essential for distinguishing P2P query routing nodes from intermediate hosts exploited by stepping stone [10]. Stepping stone is an attack where attackers perform malicious activities thorough a chain of compromised intermediate hosts to conceal their identities. Intermediate hosts are usually chained via ssh and telnet connections. When an intermediate host receives a command packet from the precessor host in the chain, it immediately forwards the packet to the successor. The difference between query routing node and intermediate host in stepping stone is that the former receives query packets from more than one node and forwards them to more than one node, while the latter usually receives packets from a single precessor and forwards them to a single successor.

E. Case study

Finally, we give a simple case study with Figure.8 to show how QRD works.



Query-Pairing Table

Inbound Packet Chunk	Outbound Packet Chunk	Pair ?
Ca (Flow 2)	Cc (Flow 1)	○
Ca (Flow 2)	Cd (Flow 1)	×
Cb (Flow 3)	Cc (Flow 1)	×
Cb (Flow 3)	Cd (Flow 1)	○

Outbound Chunks contained in Flow 1
Pc
Pd

Outbound chunks in Flow 1 are query-paired with inbound chunks received from 2 hosts, S-1 and S-2

Figure.7 How QRD works

Assume, host *X* runs P2P application, and is connected to host *S-1*, *S-2* and *D-1*. *S-1* sends a query *Qa* over a chunk *Ca* to *X*, and *X* forwards it as *Qc* over a chunk *Cc* to *D-1* via an outbound flow *Flow-1*. Similarly, a query *Qb* over a chunk *Cb* is forwarded as *Qd* over a chunk *Cd*.

First, QRD captures packets from targets, and groups them into the chunks. From the view point of QRD, however, it is still unknown which chunks contain queries.

Then, as depicted in pairing table in Figure.7, QRD performs paring check over inbound and outbound chunks. If the paring check is properly performed, a pair of *Ca* with *Cc* and pair of *Cb* with *Cd* are judged as query-paired chunks.

Finally, it is found that *Flow-1* contains queries from 2 hosts. Then, if *TH*=2 and *Nf*=1, *X* is judged as a query routing node.

V. EVALUATION EXPERIMENT

In this section, we conduct evaluation experiments to measure the detection performance of QRD.

A. Evaluation Condition

In the experiments, we measure the detection delay of QRD using traffic logs of a LimeWire ultrapeer and Winny node. Detection delay is an interval between when QRD starts monitoring the target and when an alert is produced.

In addition, the false positive rate is measured using traffic logs captured at a gateway of a typical LAN for a week. The logs were provided by DARPA in 1999 [11] and have been frequently used for the performance evaluation of IDS. The LAN contains several clients and servers such as www, ssh and smtp servers.

We assume QRD is deployed near the target host so that all the incoming and outgoing traffic can be captured with a small delay. Table.3 shows the QRD default parameters used in the experiments.

Table.3 Default Parameters

Parameter	Value
TH	3
PI	0.1 sec
QI	0.001 sec
QS	20 byte
Nf	1
Tf	2000 sec
T_{recv}	0.1 sec
T_{send}	0.1 sec

B. Evaluation Results

Figure.8 shows the detection delay as a function of TH. QRD detects LimeWire and Winny nodes within 200 sec. Once connected to a P2P network, a P2P node usually stays the network for more than tens of minutes. Thus this delay is tolerable to detect most nodes before they leave the networks.

Figure.9 shows the detection delay as a function of PI. The delay increases in proportion of PI since query packets are more likely to be grouped with packets irrelevant to queries as PI increases, and as a result the detection is further delayed.

Figure.10 shows the detection delay as a function of QI. More chunks are properly query-paired as QI increases, and which results in the faster detection.

Figure.11 shows the detection delay as a function of Nf. As Nf increases, detection delay with Winny node is steeply increased compared to that with LimeWire node. This is because a Winny node is restricted to connect to only a few parent nodes at once while a LimeWire ultrapeer node can connect to more than ten nodes

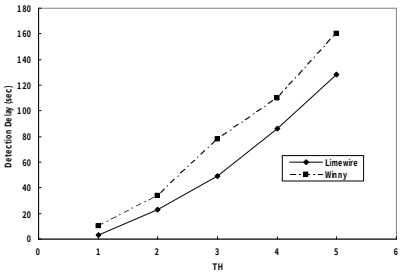


Figure.8 Detection Delay against TH

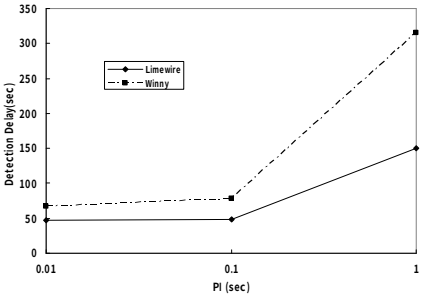


Figure.9 Detection Delay against PI

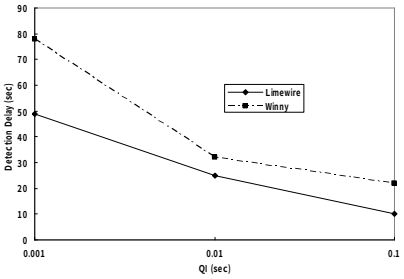


Figure.10 Detection Delay against QI

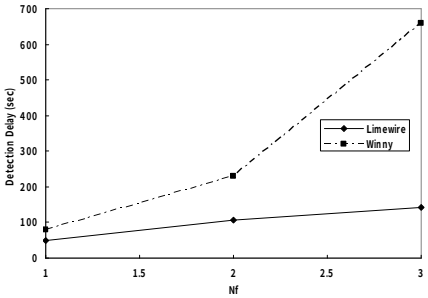


Figure.11 Detection Delay against Nf

As for false positives, no false alert is produced with default parameters. Even with TH=2, QI=0.1 and PI=0.01,

there is only 1 false alert produced in a week. The alert is caused by IRC sessions.

The results show that QRD can detect query routing nodes within a few hundreds of seconds with quite low false positives, while additional experiments with various applications and networks would be required for evaluating the performance more precisely.

VI. LIMITATION

Here, we discuss about limitations of QRD.

First of all, QRD cannot directly detect P2P nodes that do not perform query routing. Thus, QRD is not effective against leafpeers in LimeWire. However, since such nodes are connected to query routing nodes, QRD could indirectly detect the nodes by judging connection peers of detected query routing nodes as P2P nodes.

Second, P2P applications that do not adopt query routing mechanisms for locating files cannot be detected. For example, Bittorrent [8] nodes accesses to the tracker sites to addresses the nodes that have a certain file and thus no nodes perform query routing. For such applications, there are other detection methods taking advantage of the unique features such as [9].

Third, P2P nodes may evade QRD by inserting a few seconds delay before transmitting a query or appending a random size dummy data to the query. We will investigate the effects of such measures on evasion success probability and query routing performance, and then provide solutions in the future works.

VII. CONCLUSION

This paper proposed QRD, a new P2P node detection method based on query routing. Different from misuse based methods which are ineffective against unknown applications and existing behavior based methods suffering from many false positives, QRD focuses on the essential behavior inherent in query routing nodes to achieve the precise detection of unknown application with low false positive rates.

For detecting queries, packets are grouped into chunks. Then, the pairs of inbound and outbound chunks are judged to contain the same queries if the differences of their captured times and lengths are within thresholds. Finally, a host with several outbound flows containing more than a certain number of such query-paired chunks is judged as a query routing node.

Then, evaluation experiments showed that QRD can detect query routing P2P nodes within a few hundreds of seconds with a few false positives in a week.

In the future works, we will evaluate the QRD performance with various network environments and compare to existing detection methods. In addition, the effects of measures for evading QRD will be also investigated.

ACKNOWLEDGEMENT

This research was supported by a consignment research from the Ministry of Internal Affairs and Communications, Japan.

REFERENCES

[1] LimeWire Official Web site & Free Download, *http://www.limewire.com/*, (accessed at 03/15/08).

[2] T.Ohzahara, Y.Hagiwara, M.Terada and K.Kawashima, "A traffic identification method and evaluation for a pure P2P application Passive and Active Network Measurement", *Lecture Notes in Computer Science*, vol.3431, pp.55-68, 2005.

[3] S.Subhabrata, O.Spatscheck and D.Wang, "Accurate, Scalable In-Network Identification of P2P Traffic Using Application Signature, In *Proc. of the 13th International Conference on World Wide Web*, pp.512-521, 2005.

[4] S.Sen and J.Wang, "Analyzing peer-to-peer traffic across large networks", *IEEE/ACM Transactions on Networking*, vol.12, no.2, pp.219-232, 2004.

[5] G.Bartlett, J.Heidemann and C.Papadopoulos, "Inherent Behaviors for On-line Detection of Peer-to-Peer File Sharing", In *Proc. of the 10th IEEE Global Internet Symposium*, pp.55-60, 2007.

[6] F.Constantinou, P.Mavrommatis, "Identifying known and unknown peer-to-peer traffic", In *Proc. of IEEE International Symposium on Network Computing and Applications (NCA)*, pp.93-102, 2006.

[7] M.Collins and M.Reiter, "Finding Peer-to-peer file-sharing using coarse network behaviors, In *Proc. of the European Symposium on Research in Computer Security*, 2006.

[8] BitTorrent Protocol, *http://bitconjurer.org/BittTorrent*, (accessed at 03/15/09).

[9] W.Ngiwlay, C.Intanagonwiwat and Y.Teng-amnuay, "Bittorrent Peer Identification based on Behaviors of a Choke Algorithm", In *Proc. of ACM Asian Internet Engineering Conference'08*, 2008.

[10] Y.Zhang, V.Paxson, "Detecting Stepping Stones", In. *Proc. of 9th USENIX Security Symposium*, pp.171-184, 2000.

[11] MIT Lincoln Laboratory: DARPA Intrusion Detection Evaluation Data Sets, MIT, (online), available from *http://www.ll.mit.edu./IST/ideval/data/dta_index.html*, (accessed at 03/15/09).