Hindawi Publishing Corporation
Advances in Multimedia
Volume 2007, Article ID 27958, 9 pages
doi:10.1155/2007/27958

Research Article

A Measurement Study of the Structured Overlay Network in P2P File-Sharing Systems

Mo Zhou, Yafei Dai, and Xiaoming Li

CNDS Lab, School of Electrical Engineering and Computer Science, Peking University, Beijing 100871, China

Received 30 January 2007; Revised 20 May 2007; Accepted 4 July 2007

Recommended by Yi Cui

The architecture of P2P file-sharing applications has been developing to meet the needs of large scale demands. The structured overlay network, also known as DHT, has been used in these applications to improve the scalability, and robustness of the system, and to make it free from single-point failure. We believe that the measurement study of the overlay network used in the real file-sharing P2P systems can provide guidance for the designing of such systems, and improve the performance of the system. In this paper, we perform the measurement in two different aspects. First, a modified client is designed to provide view to the overlay network from a single-user vision. Second, the instances of crawler programs deployed in many nodes managed to crawl the user information of the overlay network as much as possible. We also find a vulnerability in the overlay network, combined with the character of the DNS service, a more serious DDoS attack can be launched.

1. INTRODUCTION

In a P2P system, each peer participated in the system contributed some resource, such as computation ability, memory, and storage to complete some huge tasks together, and each peer can get the benefits of the whole systems by providing service for each other. Some typical P2P applications include massive content-delivering [1], file-sharing [2–5], P2P streaming [6], and cooperative computation [7]. This paper mainly concerns about the file-sharing P2P systems.

There are several kinds of architecture in the P2P file-sharing systems. The central index server architecture, the unstructured overlay network, and the structured overlay network, also known as DHT (distributed hash table) [8–11]. There are also some systems constructed by more than one architecture. Recently, DHT has shown much superiority to the other architecture, because of its good scalability and low search cost.

Measurement study to the P2P systems will give helpful insights to the improvement of such systems. In the case of file-sharing P2P systems, we usually concern about these aspects of the system.

(i) Resources or Files. How many files are shared in this system? What files are shared? Reference [12] has shown some study in this aspect in central index server architecture and unstructured overlay network.

(ii) Users. How many users are connected to this system? How about their experience in the system? Reference [13] has shown the user experience in structured and unstructured overlay networks.

(iii) Network. How much network bandwidth cost is needed to maintain the function of the system? What kind of message consumes the most bandwidth in the system? If we find some rules in the messages transferred in the P2P network, we can redesign the P2P protocol to reduce the network bandwidth cost.

Many measurement studies have been carried out to the P2P file sharing system itself, while our motive is to learn how the structured overlay network makes the P2P file-sharing systems more usable. The resource meta-information distribution and locating is easier when the structured overlay network is available in a P2P file-sharing system, hence we focus on how this happens by measuring the DHT. We do this by finding a proper application and gathering data from it. Overnet [14] is the first massively used DHT in the practical file-sharing application, eDonkey [15]; but Overnet is not open sourced as research upon it needs some reverse engineering. eMule [4] is an open-source alternative for the eDonkey program, it support the ed2K protocol used in eDonkey. eMule also begins to support structured overlay network called kad after version 0.42. Both the overnet used in eDonkey and the kad network used in eMule is

based on kademlia [11], but the two structured overlay networks are not compatible. We can modify the source code of eMule, and let it record all the messages transfered from and to the kad network, and perform various analysis on the data.

2. RELATED WORKS

Reference [12] has performed some study on the central index server P2P systems and the unstructured overlay network P2P systems (napster and gnutella). It studies the character of the hosts in both kinds of the systems, which includes the connecting and disconnecting time profiles of the participating peers and the resource they shared.

References [16, 17] proposed some approach to decrease the search cost in the unstructured overlay network. Reference [16] used a query algorithm based on multiple random walks to provide enough searching ability while reducing the searching cost. Reference [17] used the fact that peers that have the same interests in some category of resources may be more possible to solve each other's demands, and reduced the searching cost by grouping the peers with the common interests.

Reference [18] is the first study to the structured overlay network used in a practical file-sharing P2P system. It used some reverse engineering to perform the measurement because the eDonkey is not an open source program. Reference [18] claims that no open-source system can meet the requirements of the measurement. After eMule comes out, the solid source code of this P2P application can be used to make more sophisticated study. Reference [18] has measured some date in the overnet to estimate the availability of the hosts in the system. From [18], we can make an evaluation to the scale of the overnet, and we can compare it with the scale of the kad network in eMule.

References [19, 20] focus on crawling the resource information of the traditional ed2K network (network with central index servers) in the eDonkey/eMule. eMule uses both ed2k network and kademlia to distribute and exchange the information of the sharing resources. We are focusing on the part of the kademlia network. Reference [21] analyzed some parameters of the kad network in the eMule, and tried to adjust parameters of eMule client to get better performance.

Reference [13] makes some measurement study toward both structured and unstructured overlay network. It mainly concerns about the user experience to the two kind of systems, such as the bandwidth cost, searching performance, and load balance. Reference [13] also discovers that the results did not change too much if the data was measured from different geographical position in the world. So although this paper emphasizes particularly on other aspect of the structured overlay network, we can use this conclusion to reduce the unnecessary work with single user vision in different network position.

Reference [22] showed some basic measurement results of the structured overlay network in eMule. This paper showed more analysis toward those results, and some changes have been made to the programs used in [22], for example, the crawlers are redesigned to get better performance with less resource cost. Some new information is also updated after the [22] was written.

3. METHODOLOGY

The measurement of our study has two aspects. A modified eMule client is used for single user measurement, which has the same function in all the other aspects compared to the normal eMule client, but modified clients will log all the messages of both directions between the kad network and the host, and when the routing table changed, the modified clients will also record it. The other aspect of our measurement is a crawler program deployed at multiple nodes in the PlanetLab [23]. The crawler program is modified from amule [24], a linux port for eMule, this is because we need to run many instances of the crawler on the machines of the PlanetLab, which are running linux. We removed most of the code in amule, only remained the code handling kademlia protocol, and the crawler can use it to interact with other clients in the network.

3.1. eMule and its kad network

eMule is an open-source file-sharing P2P application, and its architecture is hybrid, both central index server and DHT are used in eMule. Peers can search files in servers, and when the servers encounter a failure, the peers can still search in DHT, thus the system become more reliable. The kad network used in eMule is based on kademlia [11], which use the XOR result of two peers' IDs to compute the logical distance of two peers. eMule uses kad network to store and retrieve both key word information for metadata search and file ID information for precise file locating.

3.2. The recorded data of the single client measurement

We focus on the structured overlay network. To perform the analysis, we need the following data: all the message transferred from and to the single client, the change in the routing table, and the login and logout record of the client. The data files we generated for analysis include six types of records.

(i) Startup record. This record indicates the eMule client started its connection to the kad network. It includes record time, host ip address, host tcp port, host udp port, and host ID used in kad network.

(ii) Ending record. This record indicate the client closed its connection to the kad network. This record only includes the record time.

(iii) Incoming packet record. This record contains the information of a kad network packet received by the host. It includes record time, source ip, source udp port, packet length, and the content of the packet.

(vi) Outgoing packet record. It contains the information of a kad network packet sent by the host. Its structure is similar to the incoming packet record, only the destination ip and destination udp port replace the source ip and source udp port.

Table 1: Number of records.

Record type	Quantities of the record
Startup records	26
Ending records	26
Adding contact records	46279
Removing contact records	16244
Incoming packet records	443203
Outgoing packet records	608515

(v) Adding contact record. This record contains the contact information of a new peer added to the routing table, it includes record time, new peer's ip, tcp port, udp port and ID. We noticed that not all the contact information received from the kad network will be added to the routing table, thus we only record the contact information actually added.

(iv) Removing contact record. When a peer's information is removed from the routing table due to time out or some other reasons, its information will be recorded. This record has the same structure as the adding contact record.

The data for our measurements is collected continuously from a typical eMule user for more than two weeks. He shared some files in his computer, and also used eMule for file searching and downloading. As in [13], the result will not change too much when we measure the data from different physical locations. Thus, we believe that we do not need to collect data from more users. Table 1 shows some statistical information of the data in general. It includes the number of the six types of record we mentioned above.

3.3. The measurement from the crawler program

The crawler program is focused on the user information in the kademlia network of the eMule application. The purpose of the program is to exploit the user information of the eMule kademlia network as much as possible. In a structured overlay network, this can be done by dividing the ID spaces into many parts, and every instance of the program exploits only one part of the spaces. The crawler program acts like normal kademlia node in the view of other eMule clients, but the difference between them is that the crawler has no interests in searching specific content in the kademlia network, it sends node searching requests packets to the nodes in its routing table continuously, hopes that can help it retrieving more user information. The rate of the node searching requests packets sent by the crawler program is controlled in a set level, and the crawler program will not send two node searching requests packets to the same eMule client in a short time. The crawler program will also handle the searching or publishing requests for specific keyword or file from other eMule client correctly, all of these will minimize the impact to the kademlia network.

In the kademlia network, the length of the ID of a node is 160 bits. This provides an explicit division method for dividing the ID space; we can group all the kademlia nodes whose ID has the same highest n bits; we can adjust the granularity of the division by selecting a good value for n. In our measurement, we set the n as 8, this means all the nodes in the same part of the ID spaces have at least 8 same highest bits, and the whole ID space is divided into 256 parts. The reason we divide the space into 256 part is related to the searching tolerance in eMule. We will mention the searching tolerance in the later of this paper, here what we need to know is that any information of a specified file ID or keyword will only appear in nodes whose IDs are in the same part of the ID space. Every instance of the crawler program sends node searching requests packets only to the nodes who are in the same part of the ID space with it. The information retrieved from other eMule clients by the node searching requests will be used recursively for retrieving more user information. Because every instance of the crawler program knows contact information of other instances, they will send the information of the nodes who are not in the same part of the ID space with itself to the corresponding instance of the crawler program. The crawler program who received the user information from other instance of the crawler will treat the information the same as received from normal eMule clients, and use it in a recursive way.

Every node in the PlanetLab has only some restricted resource for every program instance running on it. So the previous design of our crawler system did not completely achieve our goal. This is because with the increasing of the obtained user information, the memory cost of every node running the instance also increases. In order to make the crawlers obtain more user information, and control the resource cost in a certain level, we adjusted the design of the crawler system. Every instance of the crawlers running in the nodes of the PlanetLab will not do anything without receiving orders, and the orders they will receive is also designed rather simple. They will receive only two types of orders. The first is orders to set a home address, the crawlers will transfer all the packets they received to the home address. The home address is the computers we can control, and the usage of the resource is not restricted. The second type of orders is to perform a set of scanning operations on a small set of contacts. After receiving this type of order, the crawlers will begin to crawl user information from the received contacts. If any result is retrieved, the crawlers will send them back to the home address. After scanning all the contacts, the crawlers will stop any action until new orders are received. All the analysis of the results from the crawlers are performed in the home computers. We can see this redesign of our crawler system can make all the crawler run on the PlanetLab nodes with only a limited resource cost. The home computers combine all the results they received, and send another set of contacts to the crawlers and they will continue to scan them. The Figure 1 shows the structure of our distributed crawler system.

4. MEASUREMENT RESULTS FROM THE SINGLE CLIENT

We can learn about the status of the kad network from several aspects. First, we can learn the routing table size changing

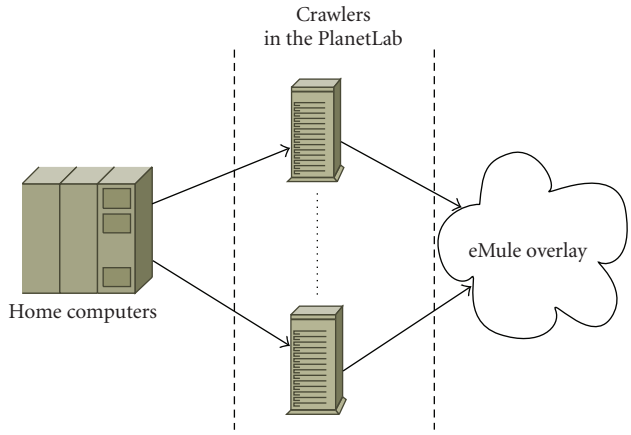

Figure 1: Routing table size.

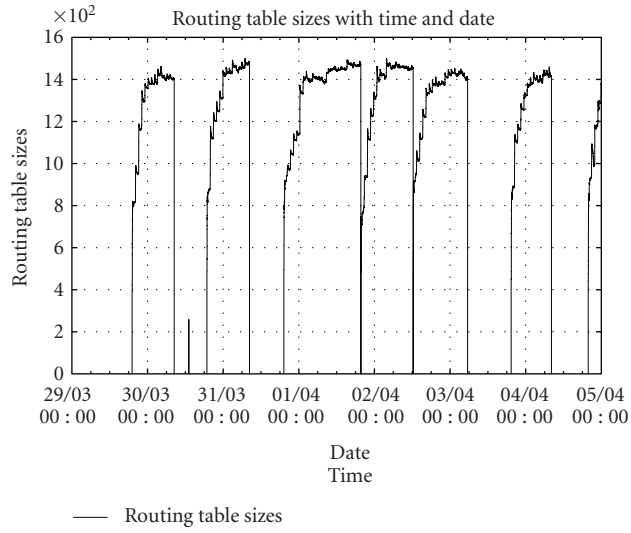

Figure 2: Routing table size.

with the time and the speed of the changing of the contacts in the routing table. When the peer received a request searching for a special node, we can check the message and learn that whether a searching request or publishing request will come at the next step. Among the searching node request, the distribution of the distance between the host and the target node is another aspect we would like to observe. The same observation can also be applied to the resource searching request and publishing request.

4.1. Routing table size

The data contains every record of adding or removing contacts, so we can learn how many contacts are in the routing table at a specific moment. We can also learn the trends of the changing of the routing table. Figure 2 shows the routing table size change with the time. Axis x is the time, and Axis y is the routing table size.

From Figure 2, we can see that in every online session of a user, the routing table size will come to a rather stable state

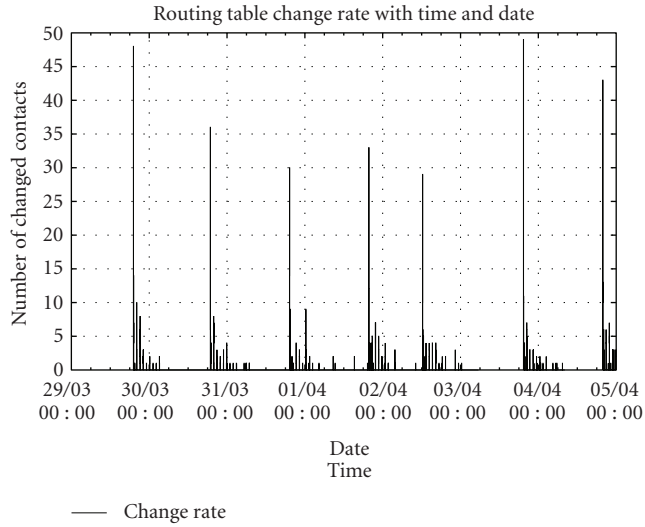

Figure 3: The changing rate of routing table.

at about 1400 contacts or so. The time a peer need to come to this stage is about six hours. Although users will connect to and disconnect from the network according their using style, most users' online session time is long enough to get their routing table into this stable state, and this forms the basis that the kademlia network needs to function normally.

4.2. Contacts change rate

Another question we would like to ask about the routing table is, at a specific moment, how many contacts are new in the routing table that they did not appear in the routing table five minutes ago? Or we can say how fast the contacts change in the routing table? Figure 3 shows the answer to this question. Every two adjacent sampling point in the figure has a distance of one minute.

Figure 3 tells us that the changing rate of the routing table is not very high. That means a contact in a peer's routing table is very likely to appear in the peer's routing table one minute later. If we observe this figure and Figure 2 together, we can also find that when a peer is just connecting to the kad network the changing rate is higher than stable state. This is because new contacts information are more likely to be filled into the routing table when it is rather empty, but when in the stable state, new contacts information cannot be written into the routing table unless some of the old contacts are detected offline and deleted from the routing table. This also tells us that when a client comes to a stable state, the possibility that many of the contacts in its routing table suddenly go offline is rather small, and thus the kademlia network can keep available in most of the time.

4.3. Nodes request distribution

The main purpose of the kademlia network is searching or publishing resource information. So when a peer received a message searching for a particular node, there are three cases

Table 2: The purpose of node searching.

Purpose	Quantities of node searching
Just searching for nodes	1504
Searching files or keywords	10758
Publishing files or keywords	61292

about the real intention of the peer at the other side of the network:

(i) The other peer is trying to search some peer whose ID is near a special ID, and use the searched contact information to fill its routing table. This happens when a peer finds its routing table does not contain enough contact information.

(ii) The other peer is trying to search some special file or keyword. The node searching process is used to find out the peers who are possible to have the information it wants.

(iii) The other peer is trying to publish information of a file or keyword.

Table 2 shows the distribution of the three cases. It illustrates that when a node searching request appears, most of the case the request is preparing for a publishing request. It is nearly about five times as many as the requests for file and keyword searching, this is because a peer needs to periodically publish all the information of his sharing files, but not all the resource it shared will be searched. Finally, we learn that the kademlia network can be rather resilient only through the resource publishing and searching, and a peer does not need many pure node searching requests to fill the routing table.

4.4. Nodes request distance distribution

To find out the character of the routing process, we can check distance between the host and the target. We use the number of same bits from the higher order to measure the distance. If the whole ID space is divided into 2^n subspaces, two IDs with the same n high-order bits will be in one of the same subspaces, but they will not be in the same subspaces if the whole ID space is divided into 2^{n+1} subspaces. So when this number is zero the distance between the two peers is very far in the logical space, and 128 means the two peers have the same ID, that is most likely they are actually the same peer. In Figure 4, we observed that most node searching requests ask this peer to search nodes which have about 10 high-order bits same as it. After the distance number comes greater than 20, the number of searching requests nearly drops to zero. We also notice that there are a few requests ask the peer to search for itself, that is, with a distance number of 128.

The node search process in the kad network usually has several attempts. The most searching requests indicate the situation of the first node search attempt. In such case, a peer begins this search rely on its local routing table only, and in most case it will find a contact with about 10 same high-order bits with any target. In the implementation of the kademlia

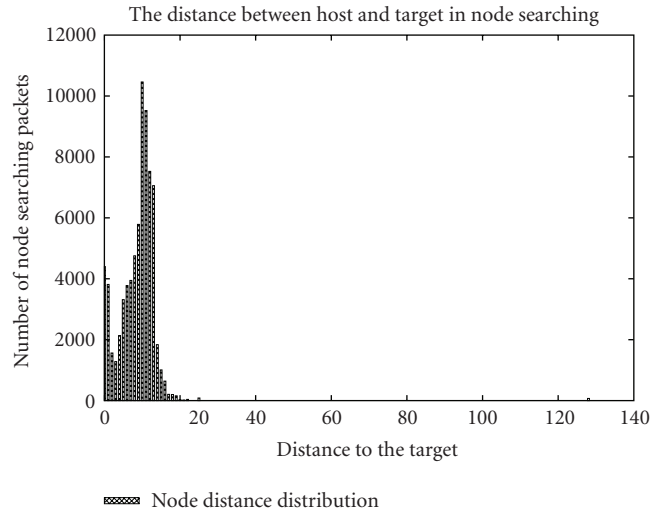

Figure 4: The distance distribution of nodes requests.

protocol, some parameters can be adjusted to balance the performance and the price. One of the parameters is the split depth, this means when a bucket depth is not enough, new contact will cause the splitting of the bucket in spite of its location, this implementation is some different from original kademlia [11]. Another is the size of a bucket. The above two parameters in the official eMule client implementation is 5 and 20. This means that in a normal eMule client has about 2^5 buckets full of contact information after it connect to the kad network for some time (several hours). So about 640 contacts are distributed in the ID space equably, when a peer begin a new search task, its very likely that it can find a contact with about 10 high bits same with the target. The requests number quickly drops to zero after the distance number greater than 20, this indicates most search attempts end here. This means when finding a node with a special ID, in the most case, we can find an ID with about 20 high-order bits same with it after searching the whole ID space. This reveals the density of the peers distributed in the whole kad network. We believe that the few requests with a distance number of 128 can be concluded as some rogue requests by other modified clients, and should be ignored.

4.5. Search request and publish request distance distribution

The method used here is quite the same as the measurement for the nodes request distance distribution; but the searching and publishing requests can only be sent to the peers whose ID are close enough to the target. This is called search tolerance in eMule, and normal eMule client will reject the requests of publishing or searching for some resource if the client's ID and the computed ID of the resource do not have 8 same high bits. So in Figure 5, we can see that it is obviously different from Figure 4 that when the number is smaller than 8, the number of requests can almost be ignored; and we have sufficient reasons to believe that the peers who send the requests are not normal eMule clients. The number of requests

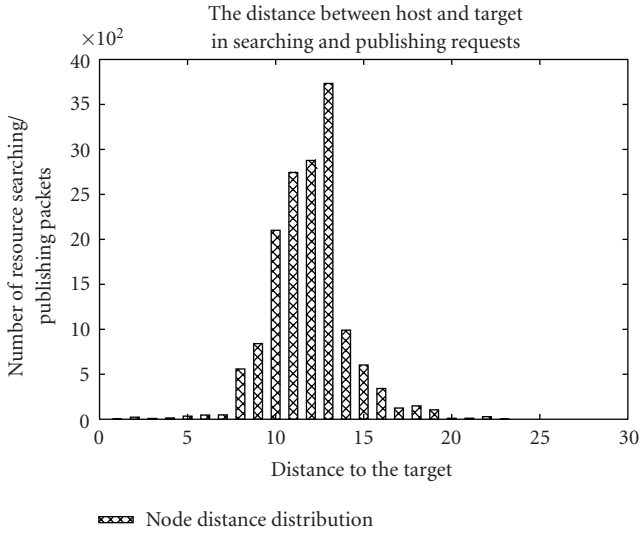

Figure 5: The distance distribution of searching and publishing requests.

drops zero after the number is larger than 25, this means it is very unlikely that a random-generated ID can have too many high-order bits same as some file or keyword ID.

4.6. Summary of the data from the single client

From the above data from single client, we can see that the kademlia network functions well in most of the time. Because the peers need to periodically publish the information of the resource they shared, the kademlia network kept well connected by the daily publishing and searching messages. We observed that a few clients are beyond normal behavior, but their impact to the whole kademlia network is rather small.

5. USER INFORMATION FROM THE CRAWLER

The instances of the crawler periodically return the user information they retrieved. The results they returned are separated by the part of the ID space. We observed the returned user information, and found that pollution in the user information continuously exists. Most of the false user contact information has a udp port number 53, which is the standard service port of DNS. Because all of the contact information from the crawlers are gathered from other peers' routing table, this means some of the peers in the structured overlay network have already got confused between normal eMule clients and DNS servers before our crawlers got those information from them. The crawlers can simply use a filter to remove all the incorrect results, but the confusion still happened in the routing tables of the other eMule clients without such kind of filter. We believe that this is a very important phenomenon and after studying the related RFC documents of the DNS [25, 26], we also believe that we have found the reason.

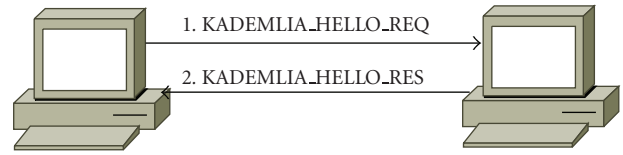

Figure 6: Correct heart-beat checking between two normal eMule clients.

5.1. Heart-beat checking mechanism in eMule

In a normal eMule client, after it received new contacts information and added them to its routing table, the new contacts will not be treated as alive until they are proved to be alive. This means the new contacts will only be used by the client (or peer) itself, such as searching and publishing resource. When other peers request for the information of users near to some ID, a normal eMule client will only tell them the information of living contacts. The way it proves whether other peers are alive is sending heart-beat packets to them, and recognizing the heart-beat response. In eMule kademlia protocol, every packet is composed by one byte of protocol code, one byte of operation code(opcode), and the rest are the content of the packet. Every kademlia protocol packet has a protocol code of OP_KADEMLIAHEADER(0xE4), and the heart-beat request and heart-beat response packet have opcodes of KADEMLIA_HELLO_REQ(0x10) and KADEMLIA_HELLO_RES(0x18), separately. This shows us that an eMule client can recognize a packet type by checking the first two bytes of the packet. Figure 6 shows how two normal eMule clients finish a heart-beat checking.

5.2. Confusion between eMule clients and DNS servers

We found many incorrect user contact information in the results from our crawler, this means that many eMule clients have the wrong information in their routing table, and they incorrectly think the information is from valid and alive eMule clients. Now, we can examine Figure 7 and see what happened when they use heart-beat packets to check the information. First, a heart-beat request packet is sent out using the ip and port information, but as we see, the target is a DNS server, not an eMule client, then the DNS server will try to explain the packet in the DNS protocol, and of course this will not succeed. But the way a DNS server treating an invalid DNS packet is not simply discarding it, it sends some packet back. The DNS server treats any packet sent to it as a DNS protocol packet, which has a message header, and the first two bytes of the header formed a query ID, this identifier is copied to the corresponding reply. The DNS server cannot interpret the incoming packet as a DNS packet, it then sends a reply whose first two bytes are the same as the incoming packet. Now that the eMule client receives the packet, it find that it is a valid eMule heart-beat request packet because the first two bytes remain unchanged, and the normal behavior of a eMule client after receiving a valid eMule heart-beat request packet is sending out a standard eMule heart-beating response packet. The DNS server will then try

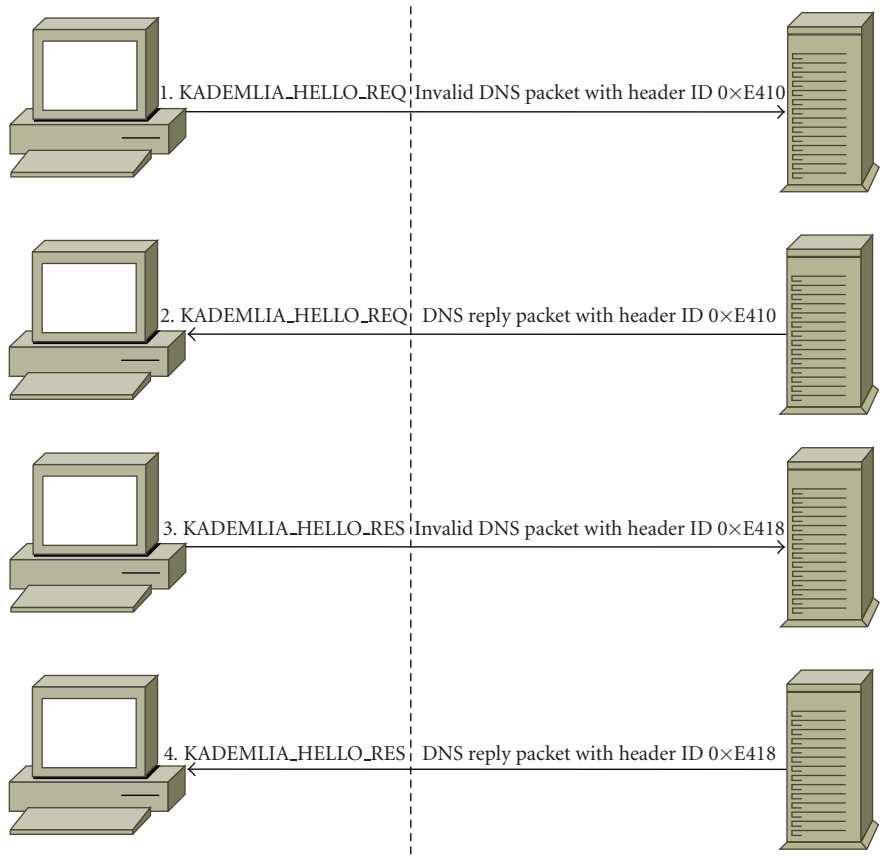

Figure 7: The eMule client is misguided when getting echo from dns servers.

to treat the packet as a DNS protocol packet only to finds that it fails again. In Figure 7, after the eMule client receives the second packet from the DNS server, that is, the message number four in the figure, the confusion begins. The eMule client will treat message number four in Figure 7 as a response for message number one, and it will believe that there is a valid and alive eMule client in the other side of the network.

Reference [27] has found that P2P systems can be used to launch a DDos attack, and it found two kinds of DDos threats to the structured overlay network in the Overnet. One of the ways is to publish false resource information to the peers, this is called index poisoning [28]. Index poisoning can be used to start a DDos attack by sending information about many popular files and telling other peers that the victim has the files, this will cause many downloading requests for the popular files to the victim, thus caused a DDos attack. The other way is directly polluting the routing tables of the peers, by sending lots of false IDs' information connected with the target host, many peers will try to contact with the target host. The contacting messages themselves can be a hazard to the target host if too many peers want to contact with it. Although the studies of [27, 28] are based on the Overnet, we can believe that it also works on the kademlia network in eMule; both of the attacks will stop after some time if the attacker does not fill false information to the kademlia network any more. But we can see the attack will be more serious if the target hosts provide services like DNS or other services which listen on udp ports and echo unrecognized packets, because in this situation the wrong information will be spread more wildly when more and more P2P clients believe that they are not wrong. The spreading of the wrong information will not stop even after the original attacker stopped polluting the nodes' routing tables. This is the reason why we continuously found DNS server contact information in the results returned by the crawlers. Other P2P systems can also be used to attack the above hosts if their heart-beat checking process is not carefully designed.

At the time of the above experiments, the most of the eMule clients in the internet have a version of 0.47a. In September 2006, eMule 0.47c was released, some of the code handling the kademlia network messages changed. Now it has a mechanism of recording all the requests it sent out in the last three minutes, the requests include the heartbeat requests, search requests, and other type of requests. If any type of "response" messages are received from the kademlia network, the new clients will check if it had corresponding request messages sent out, if not, the "response" is very suspicious, and the clients should discard it. We can see in this version that the clients have some resistance to the direct routing

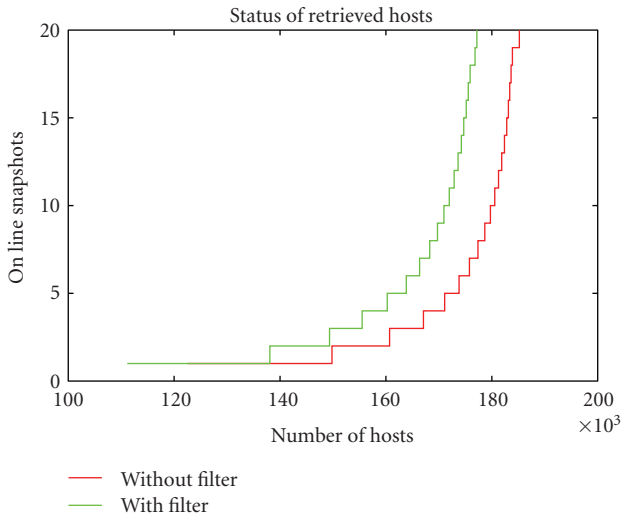

Figure 8: The results from crawlers with and without the filter.

table pollution made by malicious peers who continuously send "search node response" messages to other peers; but the threats are not completely solved. The index poisoning attack can still happen on the new clients, because it used "publish resource request" messages to perform the attack, not "response" message type. The routing table pollution can still be carried out by a passive way, this means that if any peers send search node request messages first, and malicious peers can return false contacts information. If the false contacts information look more believable to the normal peers, for example, the normal peers will believe that the false contacts as real contacts after they attempted a successful heart-beat checking. The DDos attack to the DNS servers we mentioned above will cause this type of confusion. This type of threat is still there, not solved.

5.3. *Improved validity checking of the retrieved information*

We improved the AI of the crawlers, and they will filter the wrong information. This filter includes checking the port of the contact information, contacts with port number 53 will be discarded. We have also filtered other suspicious ports. To have more confidence about the validity of the nodes, we used some test program to them. The test program will send a random-generated file ID to the node, and then search the same ID, a normal eMule client will return the file ID. Figure 8 shows the peers online distribution with and without the filter. Each time we finished crawling user information from a set of contacts, we get a snapshot of the status of peers, and we will use the snapshot to prepare for the next crawling. We selected 20 continuous snapshots for both crawlers with the filter and without the filter. We check the times that contacts appears in the snapshots. We can find crawlers using the filter will have a number of living peers about 5% less than that of crawlers without the filter in corresponding snapshots number. The false contact information is removed.

6. CONCLUSIONS

This paper makes some measurement study of the eMule overlay network from several aspects. We find that the routing table of a peer will go into a stationary state after a period of time, and the size of the routing table in stationary state will not fluctuate too much. We also find that the number of publishing is in a dominative position, so if we want to make some improvement to decrease the network cost, we can try to decrease the number of the publishing messages a peer issued. The structured overlay network keeps well connected with just the searching and publishing of resource. The distance between the target and host in node searching is rather further than that in file searching and publishing, this is mainly because the search tolerance in eMule makes peers do not like to search files or publish information at peers too far away from them, but the node searching is not restricted by the search tolerance. We retrieved many users' information and find a vulnerability in the overlay network, which is more dangerous than the normal routing table pollution. Although the eMule clients of the new version increased some resistance to the attacks, but the threats to the overlay network have not disappeared completely.

ACKNOWLEDGMENT

This work is supported by National Grand Fundamental Research 973 program of China under Grant number 2004CB318204.

REFERENCES

[1] B. Cohen, "Incentives build robustness in bitTorrent," in *Proceedings of the 1st Workshop on Economics of Peer-to-Peer Systems*, Berkeley, Calif, USA, June 2003.
[2] napster, http://www.napster.com/.
[3] gnutella, http://www.gnutella.com/.
[4] eMule, http://www.emule-project.net/.
[5] Y. Zhao, X. Hou, M. Yang, and Y. Dai, "Measurement study and application of social network in the maze P2P file-sharing system," in *Proceedings of the 1st International Conference on Scalable Information Systems (InfoScale '06)*, p. 57, ACM Press, Hong Kong, May 2006.
[6] Pplive, http://www.pplive.com/.
[7] D. P. Anderson, J. Cobb, E. Korpela, M. Lebofsky, and D. Werthimer, "SETI@home: an experiment in public-resource computing," *Communications of the ACM*, vol. 45, no. 11, pp. 56–61, 2002.
[8] I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, and H. Balakrishnan, "Chord: a scalable peer-to-peer lookup service for internet applications," in *Proceedings of the International Conference on Applications, Technologies, Architectures, and Protocols for Computers Communications (SIGCOMM '01)*, pp. 149–160, ACM Press, San Diego, Calif, USA, August 2001.
[9] A. I. T. Rowstron and P. Druschel, "Pastry: scalable, decentralized object location, and routing for large-scale peer-to-peer systems," in *Proceedings of the IFIP/ACM International Conference on Distributed Systems Platforms Heidelberg (Middleware '01)*, pp. 329–350, Springer, Heidelberg, Germany, November 2001.

[10] B. Zhao, L. Huang, J. Stribling, S. Rhea, A. Joseph, and J. Kubiatowicz, "Tapestry: a resilient global-scale overlay for service deployment," *IEEE Journal on Selected Areas in Communications*, vol. 22, no. 1, pp. 41–53, 2004.

[11] P. Maymounkov and D. Mazieres, "Kademlia: a peer-to-peer information system based on the XOR metric," in *Proceedings of the 1st International Workshop on Peer-to-Peer Systems (IPTPS '02)*, pp. 53–62, Cambridge, Mass, USA, March 2002.

[12] S. Saroiu, P. K. Gummadi, and S. D. Gribble, "A measurement study of peer-to-peer file sharing systems," in *Multimedia Computing and Networking (MMCN '02)*, vol. 4673 of *Proceedings of SPIE*, pp. 156–170, San Jose, Calif, USA, January 2002.

[13] Y. Qiao and F. E. Bustamante, "Structured and unstructured overlays under the microscope: a measurement-based view of two P2P systems that people use," in *Proceedings of USENIX Annual Technical Conference*, pp. 341–355, Boston, Mass, USA, May-June 2006.

[14] Overnet, http://en.wikipedia.org/wiki/Overnet/.

[15] eDonkey, http://en.wikipedia.org/wiki/Edonkey2000.

[16] Q. Lv, P. Cao, E. Cohen, K. Li, and S. Shenker, "Search and replication in unstructured peer-to-peer networks," in *Proceedings of the 16th International Conference on Supercomputing (ICS '02)*, pp. 84–95, ACM Press, New York, NY, USA, June 2002.

[17] K. Sripanidkulchai, B. Maggs, and H. Zhang, "Efficient content location using interest-based locality in peer-to-peer systems," in *Proceedings of the 22nd Annual Joint Conference on the IEEE Computer and Communications Societies (INFOCOM '03)*, vol. 3, pp. 2166–2176, San Francisco, Calif, USA, March-April 2003.

[18] R. Bhagwan, S. Savage, and G. Voelker, "Understanding availability," in *Proceedings of the 2nd International Workshop on Peer-to-Peer Systems (IPTPS '03)*, pp. 256–267, Berkeley, Calif, USA, February 2003.

[19] J. Yang, H. Ma, W. Song, J. Cui, and C. Zhou, "Crawling the eDonkey network," in *Proceedings of the 5th International Conference on Grid and Cooperative Computing Workshops (GCCW '06)*, pp. 133–136, Hunan, China, October 2006.

[20] K. Tutschku, "A measurement-based traffic profile of the eDonkey filesharing service," in *Proceedings of the 5th anuual Passive and Active Measurement Workshop (PAM '04)*, pp. 12–21, Antibes Juanles-Pins, France, April 2004.

[21] D. Stutzbach and R. Rejaie, "Improving lookup performance over a widely-deployed DHT," in *Proceedings of the 25th IEEE Conference on Computer Communications (INFOCOM '06)*, pp. 1–12, Barcelona, Spain, April 2006.

[22] M. Zhou, Y. Dai, and X. Li, "A measurement study of the structured overlay network in P2P file-sharing applications," in *Proceedings of the 8th IEEE International Symposium on Multimedia (ISM '06)*, pp. 621–628, San Diego, Calif, USA, December 2006.

[23] B. Chun, D. Culler, T. Roscoe, et al., "PlanetLab: an overlay testbed for broad-coverage services," *ACM SIGCOMM Computer Communication Review*, vol. 33, no. 3, pp. 3–12, 2003.

[24] amule, http://www.amule.org/.

[25] Rfc1034, http://www.ietf.org/rfc/rfc1034.txt.

[26] Rfc1035, http://www.ietf.org/rfc/rfc1035.txt.

[27] N. Naoumov and K. Ross, "Exploiting P2P systems for DDoS attacks," in *Proceedings of the 1st International Conference on Scalable Information Systems (InfoScale '06)*, vol. 152, p. 47, ACM Press, Hong Kong, May-June 2006.

[28] J. Liang, N. Naoumov, and K. W. Ross, "The index poisoning attack in P2P file sharing systems," in *Proceedings of the 25th IEEE Conference on Computer Communications (INFOCOM '06)*, pp. 1–12, Barcelona, Spain, April 2006.