

Transport Layer Identification of P2P Super nodes

DJ Oneil

Hun Jeong Kang

Jinoh Kim

Donghyong Kwon

Abstract

Since P2P applications emerged, the amount of P2P traffic has continually grown. This poses potential costs as it entails extra workload on the network and host machines. First-generation P2P protocols used well-defined port numbers, allowing network administrators to control user access to these networks. Some of the earliest P2P networks, such as Napster, relied on central servers to connect peers and took on the processing and traffic costs of doing so. Many current P2P protocols allow users to define their listening port number and use random ports to connect to other peers. Consequently, the only reliable way to identify P2P traffic is by using user payload. However, there are legal and technical issues associated with this. Also, emerging P2P technologies are beginning to encrypt the payload, rendering payload validation impossible. These new P2P networks use super nodes to connect groups of peers. Super nodes are similar to the central servers of earlier networks, but are responsible for a small subset of peers. Any node on a P2P network can be promoted to a super node and will consequently be responsible for networking connected peers and forwarding on their searches. This can add a significant computational burden and increase in incoming and outgoing network traffic to the super node. This is often done without the user's knowledge and explicit permission. Our goal is to identify hosts running P2P applications and classify each P2P node as being a super node or regular peer node.

1. Introduction

P2P activity has been a significant and constantly growing component of Internet traffic. P2P networks allow users around the world to share files, chat and, with the recent development of Skype, make free phone calls. This technology is also completely free, which makes it an especially attractive alternative to purchasing proprietary software or establishing dedicated links. Despite recent popular media claims that P2P traffic has declined due to efforts by the Recording Industry Association of America (RIAA), more sophisticated P2P traffic identification methods have shown that P2P traffic has been growing [6]. In fact, recent studies indicate that over 50% of all consumer traffic is P2P with roughly 6 million simultaneous users [18]. This growth has the potential to perpetuate itself as more users mean more instances of shared files which leads to faster download times and an even more diverse and extensive resource of content.

In its purest sense, P2P networks consist of equal nodes connected adhocly. Each node is responsible for checking all incoming queries against its own shared file set and then forwarding the query on to neighboring nodes. This controlled flooding scheme faces scaling problems due to bandwidth consumption and the uncontrolled number of possible hops to any peer on the network. On the other end of the P2P networking spectrum is a model that is very similar to the

client-server paradigm where a small set of central servers are used to connect all peers on the network. This method solves some of the issues associated with controlled flooding, but still has scalability issues as the entire burden is shifted to a very small set of servers and network connections.

To cope with the increasing network size, a hybrid of pure P2P and the client-server models has emerged. Using the new model, most nodes continue to function as peers. However, nodes with adequate resources and a fast and reliable connection can automatically assume the role of a super node. Super nodes can be used to index files and act as a central database for finding hosts connected to the network. Furthermore, these super nodes form a P2P network amongst themselves, allowing for better load balancing and survivability in case of node failure.

In spite of all the benefits of P2P technology, the widespread use has caused several issues. One is the existence of super nodes that function as servers as shown Figure 1. The concept of super nodes was created to cope with the increasing network size and has allowed higher bandwidth and connectivity in P2P networks. Supernodes act as the gateways between smaller P2P networks, and consequently, tend to have significantly more connections as peers communicate. However, being a super node is not always optional and users may have no way to determine if their computer is being used as a super node. Users of super nodes will experience performance loss as their computing resources are consumed by other users without the owner's

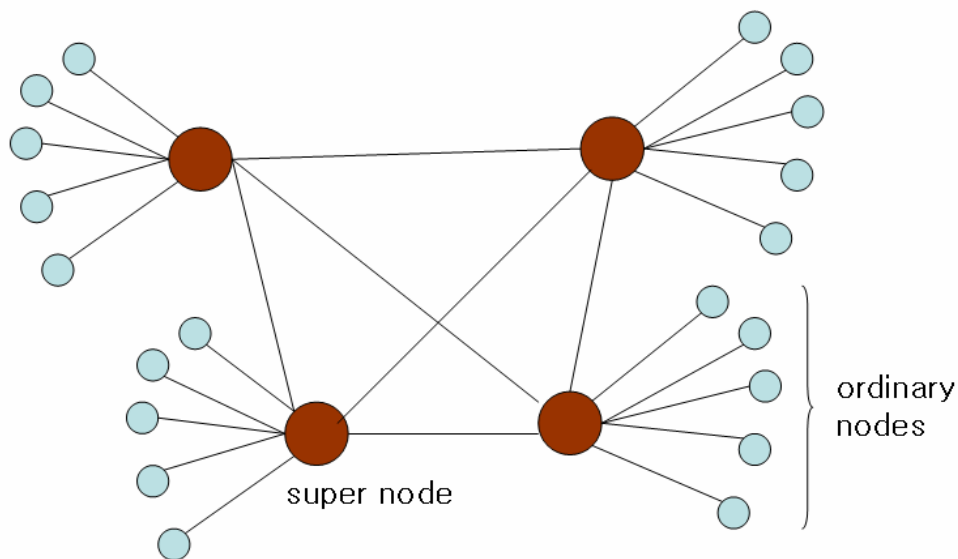


Figure 1: Hybrid Topology in P2P networks.

consent. From the perspective of network administrators, some of the most valuable computing and network resources of an organization can be squandered by external users.

To the best of our knowledge, there hasn't been research on identifying super nodes; though there is a plethora of research in classifying P2P traffic. However, classifying traffic only determines the relative amounts of each traffic type utilizing the network. It does not indicate which nodes - ordinary and super - are causing the traffic and to what degree. Understanding of nodes' types and their activities can help network operators determine the severity of each case and provide a means of effectively tracking any machines running P2P software.

Our goal is to find hosts running P2P applications within a network and classify each P2P node as being an ordinary node or super node. Finding the super node is not an easy task because it is dynamically assigned to nodes which have superior specification including CPU power, access bandwidth, and storage. Even though there has been a lot of research in classifying peer-to-peer traffic, our work would be the first contribution with respect to characterizing ordinary nodes and super nodes.

The rest of the paper is organized as follows: Section 2 describes related work. Section 3 discusses important background information relevant to the problem. This includes a description of the characteristics of super nodes and our experiments. Section 4 describes our algorithm for detecting leaf nodes and super nodes using only transport level data. Section 5 contains an evaluation of our method. Here we discuss our evaluation approach, some important notes on verification, and the results. Finally, Section 6 concludes the paper and discusses future work.

2. Related Work

There have been several methods to identify P2P applications. The traditional methods use corresponding default port numbers for P2P applications, such as TCP port 6346/6347 (Gnutella), 1214 (FastTrack), and 411/412 (DirectConnect) [1]. However, new versions of the KaZaA system do not use default port number (1214) any longer. They use dynamically assigned port numbers to transfer data between peers. Recent research [2] finally gave up identifying the traffic according to application.

Statistical methods use relationships between P2P application and their observed statistical properties. Traffic can be grouped in predetermined applications according to a set of criteria that usually includes the average packet size of a flow, the average flow duration, the inter-arrival times between packets, or the variability thereof. Some techniques such as machine learning [3] and statistical clustering [4] can be used to group traffic.

Payload-based methods make use of signatures appearing in the application layer of packets [5]. By using application-specific information, relatively accurate identification results can be obtained, and this approach is used to validate other methods. However, signatures are changed whenever new applications or new versions of existing applications are released. Consequently, the signature must be updated frequently. Also, when applications, such as Skype, encrypt their payloads, this approach cannot be applied.

Transport layer identification identifies P2P traffic at the transport layer by using connection patterns in P2P networks [6]. It uses knowledge of network dynamics, rather than depending on payload information. Also, profiling of end-host communications can be used to find hosts running P2P applications [7].

3. Background Information

All hosts on a P2P network have different capability levels in computation, network bandwidth, storage, etc... To shift the burden evenly according to host abilities, P2P network designers have devised the concept of a supernode. In this section we will take a glance at the basic characteristics of super nodes as they function in KaZaA and the new version of Gnutella. We will also discuss the results of our own observations of P2P traffic.

3.1 Characteristics of Supernodes

FastTrack is the second generation P2P protocol using super node functionality and is also the base protocol of KaZaA. With the user's permission, P2P hosts with superior capabilities can assume the role as a super node. Ordinary host nodes upload a list of files they want to share to the super nodes and send queries to the super nodes to find files. In KaZaA, the incoming port number for super nodes is always port 1214. However, it is likely that next generation P2P protocols will not use fixed port numbers in order to better disguise the application.

Though the original Gnutella network was a pure P2P network, it has evolved using the supernode concept to overcome scalability problems. Gnutella 0.6 extends the Gnutella 0.4 protocol by adding ultrapeers [26]. Like Kazza, superior nodes are classified as ultrapeers that provide index caching and query processing. In this protocol, the transport layer port numbers are dynamically assigned making it impossible to identify ultrapeers using port numbers. Though the lack of predefined port numbers adds additional complications to validating our method, we have devised an approach that will be discussed below.

One of the major characteristics of a super node is that it acts as a server to the ordinary node. When an ordinary node wants to search for a file, it sends a query to the supernode. The super node then forwards the query on to neighboring supernodes. This forwarding characteristic is one of the key distinguishing characteristics of our method. As P2P applications discover multiple nodes with matching content, these hosts make direct connections independent of the super node backbone [27]. As a result, the super node backbone is almost exclusively reserved for control traffic and querying, making it simpler to observe query forwarding.

3.2 Experiments

In the following two experiments, a query for a specific file is forwarded by super nodes. We ensured the uniqueness of the search by creating files on each machine with unique names. When a user queries for the files, nodes send query messages to their super nodes. Then super nodes send those queries to other P2P nodes including our ordinary nodes or to other super nodes. When node A sends a query to its super nodes, the query eventually arrives at node B and node A directly gets the query result from node B. The first experiment is supposed to demonstrate that ordinary nodes do not forward on queries, while the second one demonstrates the forwarding characteristic of supernodes.

Experiment 1:

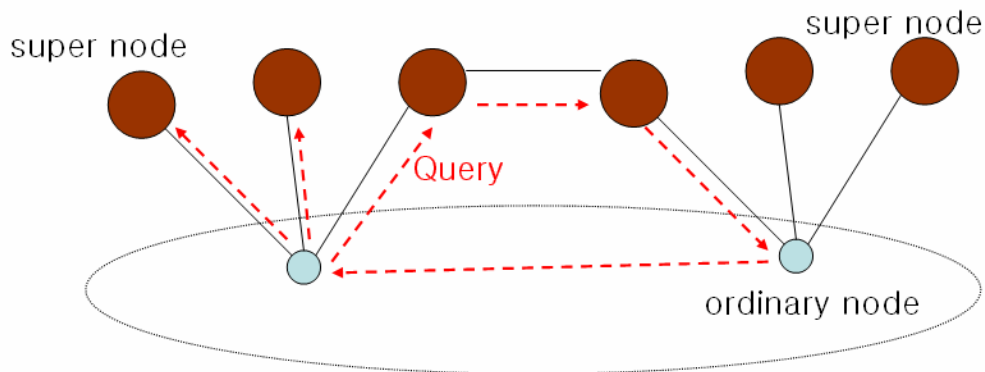


Figure 2. Experiment 1.

As the simplest case, we ran two P2P applications on the same network. When a node joins a P2P overlay network, it sends packets (mainly UDP packets) to several hosts to find super nodes to discover which nodes already exist in the network. The hosts that respond are supernodes. Even though two new ordinary nodes are in the same network, their super nodes are likely to be different. From our observation, each node connects to more than one super node for redundancy so that they can continue to participate in P2P network even when a super node disappears from the network. Generally, these connections are constructed on the TCP transport protocol, and nodes periodically exchange packets to maintain connections (i.e. to check whether nodes are still alive). This assumption is based off our observation that ordinary nodes sometimes change their supernodes.

Experiment 2:

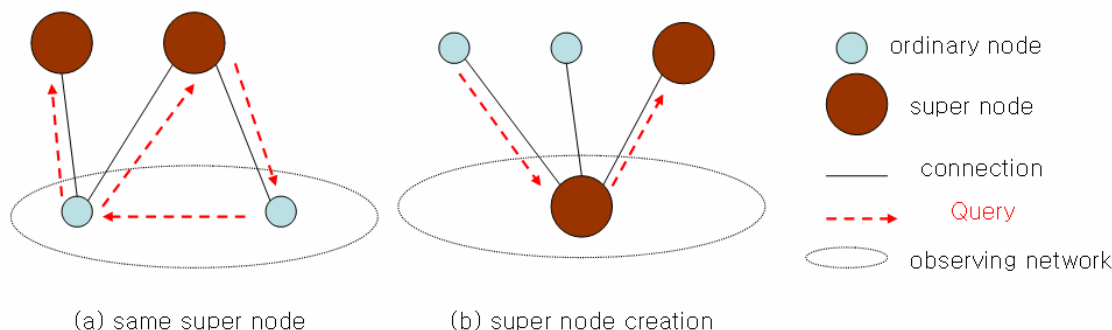


Figure 3. Experiment 2.

In the second experiment, we tried to get traffic to and from a super node. We ran the P2P application on a powerful host with a fast internet connection for a long time to promote it to a supernode. We repeated experiments so two ordinary nodes connect to the same super node by making two nodes send connecting request messages to the same host.

4. Transport Layer Identification of Peer nodes and Supernodes

In this section we will describe our Transport Layer methodology for identifying and distinguishing ordinary peer nodes and super nodes. Our method uses only packet header information and is based on P2P node and super node characteristics rather than protocol specific information, making it protocol independent. It should be noted that we take advantage of cases where protocols use specific port numbers in order to increase the accuracy of our method.

Our method works in two stages as shown in Figure 4. In the first stage we use the transport level identification of P2P traffic method described in [6]. The input of this first stage is the unfiltered transport layer traffic data. The prefilter produces the P2P traffic data (flows). This filtered data set is then fed into the second stage where we apply our heuristics to delineate between P2P super nodes and regular nodes.

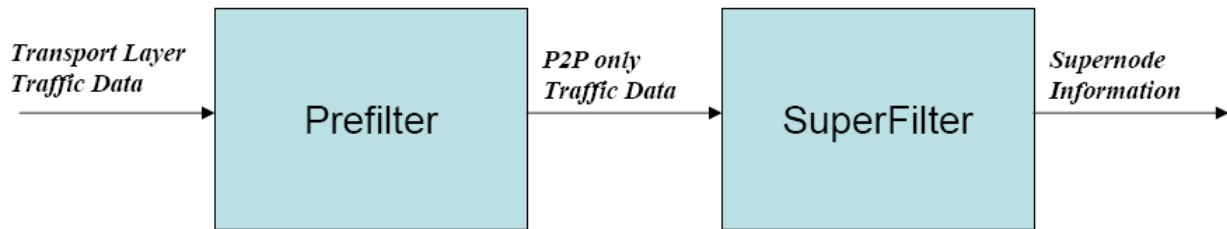


Figure 4: Two stage method for identifying super nodes.

There are several advantages to this two staged method. First, the P2P traffic identifying method described in [6] has been proven to be an effective way to identify P2P flows with 95% accuracy (and there's no reason to reinvent the wheel). The Prefilter method also uses only transport level data and behavioral characteristics to identify P2P traffic. Second, because all super nodes are also P2P nodes, it makes sense to apply the distinguishing heuristics to a data set that only contains candidates for either node type. Finally, it allows for a modularized approach. The Prefilter has a broader scope and the best P2P traffic identifying method will change as P2P protocols continue to evolve. Similarly, the behavior of super nodes may change as ways to exploit them are implemented, such as with relay nodes in Skype.

4.1 Prefilter

The Prefilter takes as its input the initial traffic data and outputs only P2P traffic. From this data we can extract the IP addresses of all P2P nodes. Using the output data, we can apply further heuristics to find super nodes within the IP set. In this section we'll present a brief summary of the transport level identification of P2P traffic presented in [6].

The method works on a flow level using the standard 5-tuple (\langle src IP, dest IP, protocol, src port, dest port \rangle) and the 64 second flow timeout. A high-level description of the method is as follows:

- The initial data set is processed to build flow tables. Various characteristics of the {IP, port} pairs are also collected including sets of distinct IPs and ports that the {IP, port} pair is connect to, packet sizes used, and transferred flow sizes.
- Potential P2P pairs are identified based on the TCP/UDP usage and {IP, port} connection heuristics.
- False positives are eliminated using specific information about similar traffic.

TCP/UDP Heuristic:

The authors observed that 6 of the 9 major P2P protocols studied used both TCP and UDP (eDonkey, FastTrack, WinMx, Gnutella, MP2P and Direct Connect). In most cases UDP is used for control traffic and queries and TCP is used for the actual data transfer. There are a number of applications that also use both TCP and UDP such as DNS, NETBIOS, IRC, and gaming and malware. However, the amount of similar traffic caught by this heuristic is small (98.5% in their data sets). In general, if a source-destination IP pair concurrently uses both TCP and UDP as transport protocols, the flows between the pair are considered P2P so long as the source or destination ports are not in a list of well-known port numbers of similar traffic.

{IP, port} Heuristic:

The {IP, port} pairs heuristic is based off the observation that multiple connections between P2P hosts are rare. Current P2P protocols have a consistent listening port for other hosts to connect to. However, they use a random port when communicating with other hosts. Thus, for an advertised destination {IP, port} pair, if the number of distinct IPs connected is roughly equal to the number of distinct ports used to connect to it, then the related flows are flagged as P2P. This is very different from web traffic where hosts usually have several connections to the webserver to speed up the data retrieval.

The Algorithm:

The algorithm is applied over each time interval, t . For each interval a flow table is built using the 5 tuple and 64 second flow timeout. The algorithm searches for source-destination pairs that use both TCP and UDP. If the ports are not in the table of well-known ports of similar traffic, then the related flows are flagged as P2P. All pairs during t for an {IP, port} pair for which the number of connected IPs is roughly equal to the number of ports are flagged as P2P. In all cases, only one pair of the flow needs to be classified in order for the entire flow to be classified.

False positives occurred with pairs with few connections and similar protocols. To mitigate the problem of similar protocols, several protocol specific heuristics for mail, DNS, and gaming and malware were applied. There were also a few simple filters applied to eliminate traffic from port scans, one-packet pairs, MSN messenger service, and hosts that used all well-known ports.

The method was evaluated against the results from a payload method they developed. The results showed that the method is successful in identifying 95% of P2P flows and a false positive range between 8% and 12%. Furthermore, the method was successful in identifying three additional P2P protocols that hadn't been used in the payload method.

4.2 Supernode Filter

In this section we will describe our methodology for delineating super nodes from regular peer nodes. Our methodology is based off research and our own observations of super node behavior and the interaction of regular peer nodes with super nodes. These sources led to the following conditions:

Condition 1: Using well-known super node ports

Some protocols, such as KaZaA, reserve a set of port numbers exclusively for super nodes. Because only super nodes can use these port numbers on a given P2P network, we flag these nodes as P2P. Other protocols, such as FastTrack, reserve ports for peers to use to communicate with super nodes (for queries, control traffic, etc...). Using this information, we can confidently identify the destination IP of a packet with one of these source ports as a super node. Table 2 contains a list of such protocols and port numbers.

Protocol	Super node listening port(s) (set X)	Peer-to-super node port(s) (set Y)
KaZaA	1214	
Fasttrack		1213

Table 2: super node related well-known port numbers

To summarize, given a set X of well-known super node listening port numbers and a set Y of well-known peer-to-super node port numbers, a node is flagged as a super node if any of the following conditions hold:

- ingress packet destination port number is in X or Y
- egress packet source port number is in X or Y .

Condition 2: Forwarding queries from peer nodes.

When a peer joins the network, it establishes connections with one or more super nodes. The connected peer informs the super nodes what files it wishes to share and also sends all queries through its super nodes. When a peer sends a query to its super nodes, the super nodes check their database to determine if any of the other connected peer nodes are sharing the file. They then forward the query on to other super nodes in the network. To summarize, a node is flagged as a super node if all the following hold:

- $(\text{egress packet timestamp}) - (\text{ingress packet timestamp}) < t$, where t is some predefined constant time for the max expected time it takes for a query to be forwarded.
- payload size of egress packet equals the payload size of the ingress packet.
- IP protocol is TCP or UDP.
- TCP flag is not SYN, FIN, ACK

Condition 3: Super nodes forwarding queries to super nodes.

As stated in *Condition 2*, super nodes forward received queries to other super nodes. Therefore, once a super node has been identified, we can identify all nodes that it forwards the query on to as super nodes. To summarize, a node is flagged as a super node if the following holds:

- ingress packet is forwarded from a super node.

The Algorithm:

```

1: Super node set, S = {}
2: Regular peer node set, P = {input set}
3: Well-known super node port set, X
4: Well-known peer-to-super node port set, Y
5: for all packets Pin, Pout
6: if Pin.dstPort ∈ X, then S ← Pin.dstIP //C1
7: if Pout.srcPort ∈ X, then S ← Pout.srcIP //C1
8: if Pin.srcPort ∈ Y, then S ← Pin.srcIP //C1
9: if Pout.dstPort ∈ Y, then S ← Pout.dstIP //C1
10: if Pin.dstIP == Pout.srcIP AND //C2
11: Pin.time - Pout.time < t AND
12: sizeof(Pin.Payload) == sizeof(Pout.Payload) AND
13: Pin.IpProto == TCP or UDP AND
14: Pout.IpProto == TCP or UDP AND
15: Pin.TcpFlag != SYN, FIN, ACK AND
16: Pout.TcpFlag != SYN, FIN, ACK
17: then S ← Pin.dstIP
18: if Pin(i).srcIP ∈ S, then S ← Pin(i).dstIP //C3

```

To increase the accuracy of our method, each of the conditions is assigned a *confidence* weight.

5. Evaluation

5.1 Approach

In this section we describe the procedures for evaluating and verifying our method. First, we collect traffic data in order to verify the algorithm results. There are several constraints on getting network data:

- It is impossible to construct the isolated P2P test bed without contacting the outside P2P overlay network.
- There is no way to verify the distinction between super nodes and regular peer nodes for the UMASS data.
- There are only a few networks from which we can collect traffic data.

We observed behaviors of P2P nodes by capturing and analyzing traffic generated by P2P nodes. In order to monitor a super node and ordinary node, we performed experiments in following networks:

Location	Medium	Address Allocation	IP address	Speed
University	Wireless	Dynamic	Public	10 Mbps
Netlab	Wired	Dynamic or Static	Public	100 Mbps
Home	Wireless	Dynamic	Private	5 Mbps

Table 1. Traffic source environments.

To verify the output, we label the collected data sets. Because many P2P applications encrypt the payload, it is impossible in such circumstances to accurately label the data. In cases where the payload is not encrypted, there is not always a convenient way to distinguish regular peer traffic from super node traffic. We use the following methods to accurately distinguish nodes:

- Gnutella (Limewire): there exists a menu that shows the relation between hosts and super nodes. We are able to get the super node list in the menu [24]
- KaZaA: we can confirm the super node by looking up the registry [25]

To identify super nodes and ordinary nodes, we apply the *Supernode Filter* algorithm to the P2P only traffic output from the *Prefilter* algorithm. Then, we compare the labeled data with our results. To evaluate the correctness of our method, we will consider two statistics:

- true positive (= 1- false negative) =
$$\frac{\text{Num. of true P2P hosts classified as super nod}}{\text{Num. of true super node}}$$
- false positive (= 1- true negative) =
$$\frac{\text{Num. of false P2P hosts classified as super nod}}{\text{Num. of false super node}}$$

5.2 Verification

In order to verify our results we needed one of our machines to be promoted to a supernode. We have tried to make a supernode for several days under the following conditions:

Location	Address	IP	Speed	Resource	F/W	Period	Results
----------	---------	----	-------	----------	-----	--------	---------

	Allocation	Address				(days)	
Netlab	Static	Public	100Mbps	Host1,2,3	○	10	Fail
Netlab	Static	Public	100Mbps	Host2,3	X	10	Fail
University	Dynamic	Public	10Mbps	Host 4-7	X	8	Fail
Home	Dynamic	Private	5Mbps	Host 4-7	X	8	Fail

Table 2. Environment Configuration.

		CPU	Memory(Mega)	OS
Host1,2	Desktop	P4 2Ghz	512	Linux
Host3	Desktop	P4 2Ghz	512	Windows
Host4-7	Laptop	P4 1.8-2.4	512-1G	Windows

Table 3. Resource Specification.

Specifications for minimum performance requirements for a supernode are not documented. Because of this, we have no way to determine whether our configuration should work. Table 2 indicates that all of our configurations failed. There are several possible reasons for this. In environments 1 and 2 the host machine was behind a firewall. We suspect that this disqualified our machine from becoming a supernode. In environments 3 and 4 we suspect that the host machine did not meet the performance requirements necessary to become a candidate supernode.

We were unable to make a supernode within our network, and, consequently, are unable to evaluate our algorithm at this point. We intend on doing more work to get supernode traffic. However, even though we cannot effectively evaluate our method, we can still proceed to apply it to a subset of the UMASS data.

5.3 Experimental Results

Based on our methodology, we have generated statistics from the UMASS data. Our algorithm identified the following characteristics:

We have performed experiments using 1,250,000 packets in UMASS data. The statistics for P2P are derived by the Prefilter function and the statistics for SN (supernode) are derived by the Superfilter. We use four time window values of 1.0, 0.1, 0.01, and 0.001 to observe how it affects the results. The value 1.0 means that the difference between timestamp for supernode filtering is less than 1.0 in the dataset (i.e., the forwarding time bound).

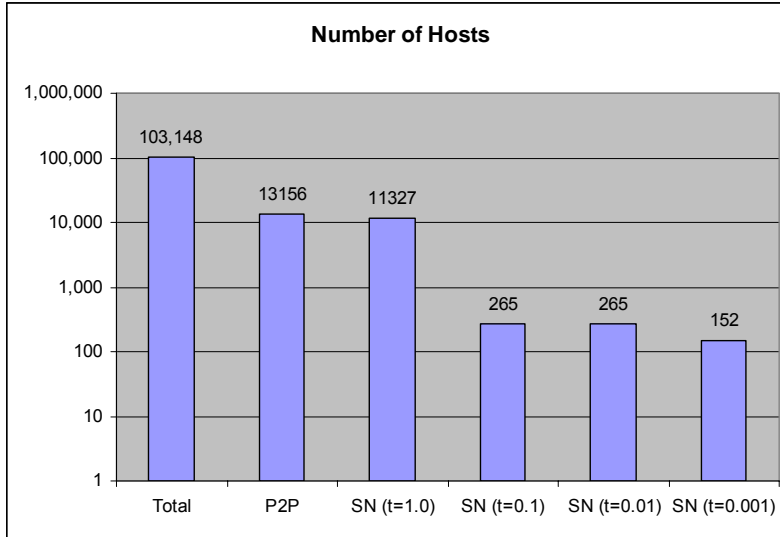


Figure 5: Number of Hosts of each group.

This figure shows the number of hosts. The P2P nodes are about 13% of total nodes. When the time window is set to 1.0, the number of supernode is almost same as the number of P2P nodes. When the time window is set to 0.1, however, the number of supernode is sharply decreased to 2 percent of the total P2P nodes. There is no difference between time windows 0.1 and 0.01, but the number of supernode decreases when the time window is set to 0.001.

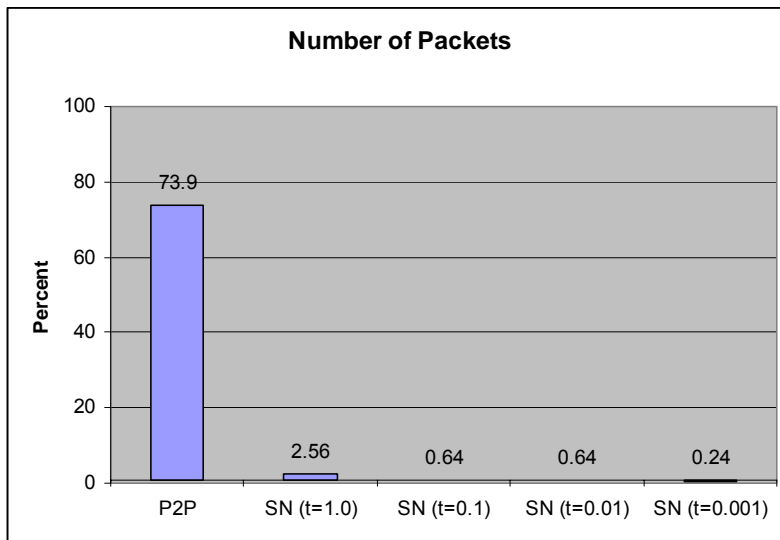


Figure 6: Number of Packets of each group.

This figure shows the rate of P2P packets. It is surprising the P2P packets are over 70% of the total packets. We presume that this is because we could not apply methods to reduce false positives. In spite of some loose constraints, we believe that most of the traffic must be P2P packets. The supernode traffic is relatively low compared to P2P traffic. When the time window is 1.0, the percentage of supernode traffic is only 2.56%.

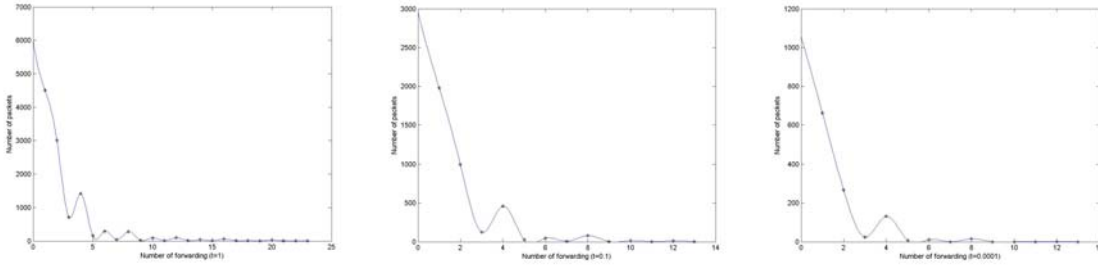


Figure 7: Number of forwarding ($t=1, 0.1, 0.001$)

This figure shows the number of forwarding multiplication versus total number of packets with variance of the time window. Regardless of the change in time window, the figures show almost the same pattern. Supernodes usually multiply a packet to 1 ~ 4 packets when forwarding.

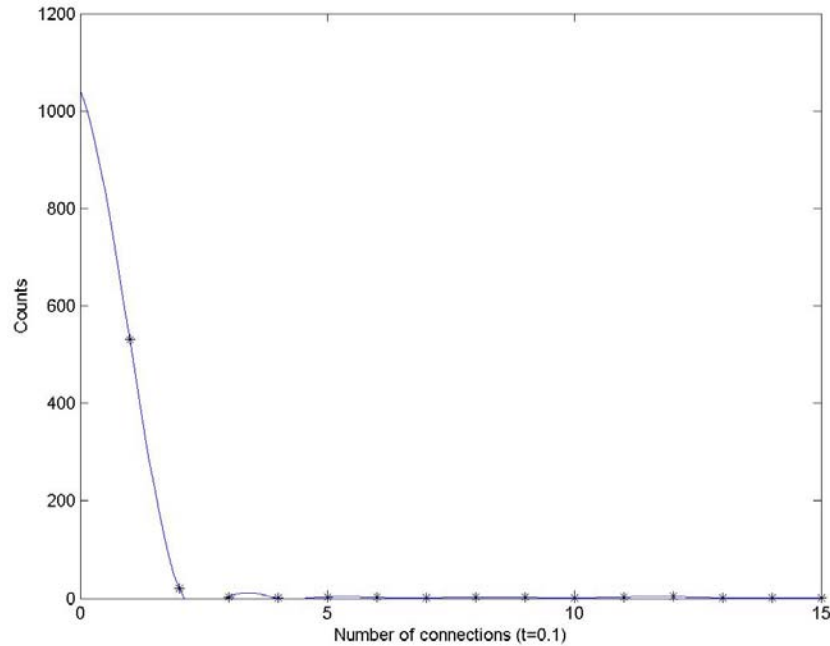


Figure 8: Number of Connections ($t=1, 0.1, 0.001$)

This figure shows the number of connections for supernodes. Most supernodes have less than 2 connections with others. It is difficult to say what the reasons are. However, we believe that there are more connections between UMASS nodes. Consequently, the number of connections is underestimated rather than real, given that supernodes maintain more than three connections in general.

6. Conclusion

In this paper, we proposed the algorithm to identify supernodes and analyze results related to supernodes from the data set. The algorithm is based on observations that supernodes forward

ordinary nodes' queries and advertisements. The experiment was performed by using network trace data from UMASS. The results implicate some parameters such as tolerant time delay of forwarding time significantly affect to decide supernodes. The difference between the number of supernodes of the delay time was a factor of 10 in our experiments. The number of super nodes was 265 when the delay time was 0.1, which is much higher than what we expected. One reason might be that we set the parameters a little loosely.

However, we failed to validate our algorithm because we could not run a supernode on our machines. This might be caused by reasons such as low machine specifications or network environments (eg. firewall setup). In addition, the characteristics of supernodes were not clearly defined or discovered.

For the future work, we intend on getting supernode traffic for validation. After validation, we are planning to enhance the accuracy of algorithm to find supernodes by leveraging characteristics of leaf-super node communication to identify super nodes. When leaf nodes log onto the network, they usually begin with a boot-strapping process to locate live supernodes. Some of the possible supernodes respond and the peer application uses a subset of those as their link into the network. Once a leaf is connected to a peer, the connection is maintained by keep-alive messages. It is part of our on going research (to be continued in the summer) to study these characteristics more and add corresponding new conditions to the algorithm based on our results.

7. References

- [1] S. Sen and J. Wang, "Analyzing Peer-to-Peer Traffic across Large Networks", in Proc. of the second ACM SIGCOMM Workshop on Internet Measurement Workshop, Nov. 2002.
- [2] A. Gerber, J. Houle, H. Nguyen, M. Roughan, and S. Sen, "P2P The Gorilla in the Cable," in National Cable & Telecommunications Association (NCTA) 2003 National Show, Chicago, IL, June 8-11, 2003.
- [3] A. McGregor, M. Hall, P. Lorier, and J. Brunskill, "Flow Clustering Using Machine Learning Techniques", in PAM, 2004.
- [4] A. W. Moore and D. Zuev, "Internet Traffic Classification Using Bayesian Analysis Techniques", in ACM SIGMETRICS, 2005.
- [5] S. Sen, O. Spatscheck, and D. Wang. Accurate, "Scalable In-Network Identification of P2P Traffic Using Application Signatures", in WWW, 2004.
- [6] T. Karagiannis, A. Broido, M. Faloutsos, and K. Claffy, "Transport layer identification of P2P traffic", in ACM/SIGCOMM IMC, 2004.
- [7] K. Xu, Z. Zhang, and S. Bhattacharya, "Profiling Internet Backbone Traffic: Behavior Models and Applications", in SIGCOMM, 2005.
- [8] T. Karagiannis, A. Broido, N. Brownlee, K. Claffy, and M. Faloutsos, "File-sharing in the Internet: A characterization of P2P traffic in the backbone" Technical report, 2004.
- [9] B. Yang and H. Garcia-Molina, "Comparing hybrid peer-to-peer systems", in The VLDB Journal, pages 561–570, 2001.
- [10] D. Ilie, D. Erman, A. Popescu, and A.A. Nilsson, "Measurement and analysis of Gnutella signaling traffic", in IPSI 2004.
- [11] J. Liang, R. Kumar, and K.W. Ross, "The KaZaA Overlay: A Measurement Study", in Proceedings of the 19th IEEE Annual Computer Communications Workshop, 2004.

- [12] S.A. Baset and H. Shultzrinne, “An Analysis of the Skype Peer-to-Peer Internet Telephony Protocol”, in Seminar on Internetworking 2005-04-26/27, Department of Computer Science, Columbia University, NY, 2004.
- [13] Fasttrack, <http://www.fasttrack.nu>.
- [14] A. Crespo and H. Garcia-Molina, “Routing indices for peer-to-peer systems”, in Proceedings International Conference on Distributed Computing Systems, July 2002.
- [15] Karagiannis, et al, “BLINC: Multilevel Traffic Classification in the Dark”, ACM SIGCOMM 2005.
- [16] Kyoungwon Suh, et al, “Characterizing and detecting related traffic: A case study using Skype,” INFOCOM 2006.
- [17] “Peer-to-Peer”, <http://en.wikipedia.org/wiki/Peer-to-peer>
- [18] “CacheSwitch 310”,
http://www.cachelogic.com/products/CacheLogic_Datasheet_Cacheswitch310_v2.0.pdf
- [19] Borland, “Network monitoring reveals hidden P2P traffic”, CNET News.com, August 28, 2003
- [20] “Ultrapeer Election Principles”,
http://www.the-gdf.org/wiki/index.php?title=Ultrapeer_Election_Principle
- [21] “Peer to Peer and SPAM in the Internet”,
<http://www.netlab.tkk.fi/opetus/s38030/F03/Report-p2p-spam-2003.pdf>
- [22] “Super nodes”, http://www.KaZaA.com/us/help/faq/super_nodes.htm
- [23] “giFT isn’t FastTrack”, <http://cvs.sourceforge.net/viewcvs.py/gift/giFT/README?rev=1.3>
- [24] LimeWire, <http://www.limewire.com>
- [25] KaZaA Super node Tool
http://projects.resnet.purdue.edu/docs/ResNet%20KaZaA%20Super_node.pdf#search='super_node'
- [26] The Gnutella 0.4 protocol specification, 2002, <http://www.limewire.org>.
- [27] M. Castro, et al., “Debunking some myths about structured and unstructured,” Proc. of NSDI, 2005