Performance Measurement of Skype under Network Dynamics

Yang Song
Department of Electrical and Computer Engineering
University of Massachusetts, Amherst, MA, 01002
Email: ysong@ecs.umass.edu

Lixin Gao
Department of Electrical and Computer Engineering
University of Massachusetts, Amherst, MA, 01002
Email: lgao@ecs.umass.edu

***Abstract*—Skype is one of the most popular VoIP software. It has more than 100 million users registered so far, and typically has more than 10 million users online at the same time. One of the key innovations of Skype software is that it uses peer-to-peer technology or supernodes to penetrate firewalls or network translation boxes (NATs). Despite the popularity of Skype, it is common for Skype users to experience call drops or performance degradation during a conversation. In this paper, we perform extensive measurement studies of Skype clients in order to understand how Skype performs under various network dynamics. In particular, we show how Skype reacts to end-to-end network path failures under various network topologies and configurations emulated with VINI. We have two major findings. First, while Skype reacts to end-to-end path outages by selecting backup relay nodes when both clients are behind firewall or NAT, it does not do so when one of the clients uses a public IP address. That is, when one of the clients uses a public IP address, the clients will connect directly with each other without using a relay supernode. If the direct path between the two clients experiences an outage, Skype will not select a supernode to provide overlay paths in order to maintain the connectivity. Second, even when both clients are behind NATs or firewalls, under an outage of the path between a client and the relay supernode, Skype might not be able to successfully switch to a backup supernode, and it is still likely that the call will be dropped or there is significant degradation of performance. Our findings pinpoint the deficiency of Skype design and provide guidelines for the design of future peer-to-peer VoIP systems.**

I. Introduction

Skype is one of the most popular VoIP software. It has more than 100 million users registered and more than 10 million users online at the same time. One of the key innovations of Skype software is that it uses peer-to-peer technology or overlay nodes to enhance the robustness of the VoIP communication and penetrate firewalls or network translation boxes (NATs). Despite the popularity of Skype and overlay structure built on top of Skype, Skype users often experience call drops or performance degradation during a conversation.

In this paper, we investigate what may cause Skype communication to experience call drops or performance degradation. In particular, we explore how robust the Skype overlay network structure is by studying how Skype software reacts to overlay node failures or network outages. Skype overlay network takes advantage of online Skype clients to act as overlay nodes to relay VoIP traffic. Therefore, the robustness of the Skype software hinges on whether it can react to overlay node failures or network end-to-end path failures. Both failures scenarios are common. A Skype user can quit the software or shut down its computer. It has also been observed that end-to-end path failures are widespread and can last as long as 20 seconds [16].

We perform extensive measurement studies of Skype clients in order to understand how Skype performs under various network dynamics. In particular, we show how Skype reacts to end-to-end network path failures under various network topologies and configurations. We emulated the controlled network environment with VINI [7]. By analyzing network traces collected from Skype clients, we discover that while Skype software is robust to overlay node failures, it is not sufficiently robust to network end-to-end path failures. In particular, our major findings are as follows.

First, we explore the overlay or supernode network structure of Skype. Each supernode in the Skype network can potentially play two roles. One role is to act as a distributed node for maintaining information or state of Skype clients such as whether it is active. In fact, each Skype client relies a particular supernode (we referred to it as a parent node) for login or discovering other Skype state. Another role that a supernode can play is to relay VoIP traffic between two Skype clients. If two communicating Skype clients are both behind NATs, a relay supernode is used to ensure that Skype clients can penetrate network translation boxes (NATs). In addition, each Skype client might have backup relay supernodes so that it can rely on one of these nodes if its relay supernode is down.

Second, Skype can recover from supernode failures in a timely fashion. When a parent supernode leaves the Skype network, its Skype client can automatically discover an alternative supernode to act as its parent supernode. Further, when a relay supernode is down, its Skype client can automatically fall back to a backup relay supernode.

Third, Skype is not sufficiently resilient to end-to-end path failures. We first consider the scenarios that at least one of the two communicating Skype clients uses public IP address. In this case, no relay supernode is used. When there is end-to-end path (sometime transient) outage between the two communicating Skype clients, Skype clients will not take advantage of the relay supernode network to maintain the connectivity, and therefore can lead to transient performance degradation or even call drops.

Fourth, even when a relay supernode is used between two communicating Skype clients, Skype clients might not be able to successfully switch to a backup supernode or any supernode. Considering the scenario that both clients are behind NATs and there is an end-to-end path outage between a client and the relay supernode, it is likely that the call will be dropped or there is significant degradation of performance. Our findings pinpoint the deficiency of Skype design and provide guidelines for the design of future peer-to-peer VoIP systems.

The rest of the paper is organized as follows: in section II, we presents a brief analysis of related work to Skype; in section III, we show the methodology and testbed of the experiments; in section IV, we introduce parent supernode and relay supernode; in section V, we discuss how Skype reacts to the link failures, and show some suboptimal methods Skype implements. Section VI gives the conclusions.

II. Related work

Skype can work seamlessly across NATs and firewalls, and has superior voice quality among similar applications, so there have been increasing interests in it. First, researchers are interested in the protocols used by Skype. Baset et al. show the Skype protocols during login, NAT and firewall traversal, and call establishment processes [6]. Second, there are measurement studies about Skype mechanisms in the congestion control [8] and bandwidth variations [9]. Extensive Skype traffic measurement is performed in [11]. Xie et al. analyses further on the impacts of access capacity constraint and the AS policy constraint on the performance of Skype [17]. Third, a latency-aware method for Skype clients to select a good relay supernode so as to improve the voice quality is proposed in [14]. Fourth, the detection of Skype-relayed traffic is presented in [15] [**?**]. In this paper, we focus on investigating what may cause Skype call drops or performance degradation, and point out the deficiency of Skype design.

III. Methodology and testbed of the measurements

Since Skype uses encrypted platform to control and transfer data, we are not going to figure out the exact application layer protocols of Skype. Instead, we treat the Skype clients as a partial black box with some well known key components. By gathering and analyzing the traffic, we try to figure out the Skype's methods to deal with all kinds of failures.

There are two major parts of the experiment setting: how to control the network event and how to gather the traffic. We want to control the network event with real routing software and traffic load, and make them happen in the real Internet. So we choose VINI [7]. VINI is a virtual network infrastructure that allows researchers to conduct network experiments in a realistic and controlled environment. The Prototype of VINI is developed on Planet-lab [1]. VINI allows users to build an overlay network on slices, and each node in the slices can be accessed. User can run overlay routers on each slice using User Mode Linux virtual machine [3] and Click forwarding engine [12]. We can route traffic into VINI by OpenVPN clients, and make the traffic access Internet by NAPT gateways in VINI. On this platform, we create different topologies and network events to perform all the experiments.

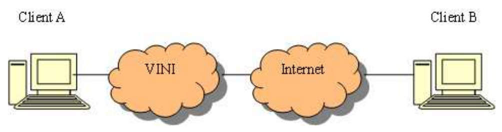

Fig. 1. The experiment settings

The experiment setting is shown in Figure 1. We first bring traffic into VINI overlay network, and then make it go through the Internet before it reaches the destination. In this setting, VINI can be viewed as a part of Internet. During the experiment, we assume the Internet is stable, and use VINI to emulate controllable Internet by making all kinds of network events in the VINI overlay platform.

We use both Linux and Windows systems to do the experiments. In order to gather the traffic, we use sniffers on both clients, tcpdump [2]for Linux and wireshark [4] for Windows. We use Skype version 1.3.0.53 in Linux and 2.5.0.91 in Windows XP. Both machines are connected to the Internet with a campus network.

IV. parent supernode and relay supernode

Skype participants are organized in two categories, the ordinary nodes and the supernodes. Supernodes are promoted from ordinary nodes with sufficient bandwidth and public IP address. Each Skype client holds a list of some supernodes' addresses and port numbers in host cache [6]. By connecting to some of the supernodes, ordinary nodes join the Skype networks. Skype is a peer-to-peer application robust to NAT and firewall limitations. If the clients are behind the NAT or firewall, the supernodes are used to relay the sessions. This behavior potentially improves VoIP performance as well.

Before we investigate the performance of Skype under network events, we first study the structure of Skype overlay networks. This structure improves Skype's robustness in failures. First, Skype separates control and relay functions into different supernodes. These supernodes are called parent supernodes and relay supernodes accordingly. This makes Skype more robust in node failures. Second, Skype sets up more than one path to connect a caller and a callee behind the NATs or firewalls in the overlay network. If the current path is down, other paths will replace it to maintain the connection. So there are always more than one relay supernodes for every call between the clients behind the NATs or firewalls. In the following part of this section, we will show the detailed action of parent and relay supernode in Skype overlay structure.

A. Parent supernode

When Skype client logs in, it will first communicate with the overlay network by connecting with the supernodes in host cache. If no valid entry exists in the cache, Skype client will communicate with the bootstraps to get the available supernode addresses. After communicating with some supernodes by

UDP packets, Skype client will get a certain one, and starts a TCP session with that supernode by SYN. That supernode will keep a TCP connection with the Skype client until it signs out. That supernode is the *parent supernode*. From what is observed in the experiments, we see that after communicating with the parent supernode, Skype client will show online state. Each time, when the buddy logs in or signs out Skype, parent supernode will send TCP message to inform the client. If the client searches an on-line user who is not in the buddy list, the client will also reach the user's parent supernode after communicating with many supernodes in the overlay network by UDP packets. If the searched one is off line, the client will still obtain a few IP addresses which the off-line user used to log in Skype for last few logins, and client will send some packets to those IP addresses before it realizes that searched one is offline. Skype client relies the parent supernode for login or discovering other Skype state.

Normally, parent supernodes are the supernode in the client's host cache. But in some cases, Skype client will search a valid one from the supernode network. Because parent supernode will keep a TCP session with the client until it signs out, we can easily discover its IP address.

B. Relay supernode

Relay supernodes play an important role in Skype application. First, because of the relay supernodes, Skype is robust to the NATs or firewalls limitations. When a caller and a callee are both behind the NATs or firewalls, relay supernodes will be promoted from the supernodes to relay the session. Second, there will be more than one relay supernodes assigned for each call. This action significantly improves the robustness of Skype. Because the relay supernodes may sign out Skype and quit the overlay network at any time, and the links in the Internet are not reliable as well. The backup path will guarantee the connections in case the current one is down. Further, [14] shows the overlay path will improve the communication quality as well. We introduce the relay supernodes in the following part.

If both the clients are behind the NAT, and the caller starts the call, the caller will first communicate with some supernodes in UDP sessions. Then the caller will receive the ICMP packets to report port unreachable from callee's public reachable IP address. That process may test the categories of the NAT or the firewall. After that, it will start a TCP session with several supernodes by SYN packets. These nodes are *relay supernode* and *backup relay supernodes*. One of them will be used for relay the traffic between the two clients, and others are used as the backup supernodes in case the current one is not available. On the callee side, it will receive the ICMP traffics during the establishment process as well. After that the callee will receive a TCP packet from the parent supernode, and then starts the TCP session with the relay supernode and the backup relay supernode by TCP SYN. During the communicating process, the backup relay supernode will keep a TCP session with the two clients. Even when the call ends, the relay and backup supernodes will still keep the TCP session with these clients.

If one of the clients is with public IP address, no relay or backup relay supernodes are needed. The call will be established directly with the public IP address. Here, we notice a main difference between clients with and without public IP address: if at least one of the clients is with public IP address, the call will be established with the public IP address without any relay or backup relay supernode.

V. How Skype client reacts to failures

In this section, we investigate two kinds of failures in the Skype network. One is node failures, and the other is the end-to-end path failures. We mainly focus on the path failures, because Skype is robust to node failures.

A. How Skype reacts to node failures

Node failures must be assumed to occur frequently in Skype. The supernode network consists of many unreliable desktop machines, and the user may sign out of Skype and quit the overlay network at any time. So neither parent nor relay supernodes are reliable. From the experiments, we find that if parent supernode sign out Skype application, they will send TCP FIN packets to inform the client it serves. New supernode will replace the unavailable one. If the relay supernode signs out Skype, both clients will get the TCP FIN packet as well. The backup relay supernode will replace the former one immediately. Skype designs a good method to deal with the node failure. But if there is a path failure, no notice information will be sent. It may cause problem in the time-sensitive communication. In the following two sections, we describe how Skype reacts to the path failures. In section V-B, we describe how clients react to path failures if the call is established without the relay supernode. In section V-C, we show how clients react to path failures if the relay supernodes are needed.

B. How clients react to path failures if the call is established without the relay supernode

From section IV we know that if one of the clients is with public IP address, the call will be established with the public IP address directly. In this part, we test how Skype reacts to path failures when the call is established without the relay supernodes. We set the experiment as shown in figure 1. Client A is behind the NAT, and client B is with a public IP address. If there is a path failure, how do these clients react? Will it use the relay supernode if the clients can't reach each other directly by public IP address? With VINI, we set two links. One link is used for client A to connect to client B's public IP address directly. The other link is used for client A to connect to the Skype overlay networks. When we make the direct path between the two clients down, client A can still communicate with the supernodes in the Skype overlay network. Figure 2 shows the internal structure of VINI. There are three nodes in VINI as shown in the picture. By changing the settings in VINI, we make client A reach client B's public reachable IP

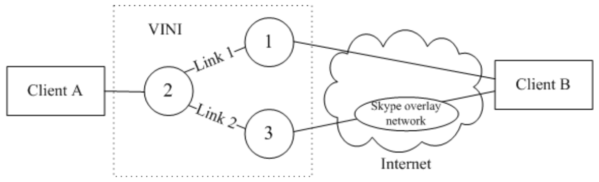

Fig. 2. The experiment topology of the calls established without relay supernodes

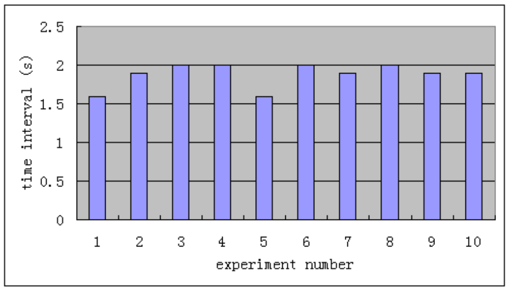

Fig. 3. The time interval from the path failure to client A starts to ping client B's public IP address by TCP packets

address through link 1, and reach other IP addresses by link 2. If link 1 is down, client A will not be able to communicate with client B directly, but it still can connect to other nodes in Skype overlay network through link 2. During the experiment, we create different network events to emulate them appearing in the Internet, and the link failure in VINI emulates the path failure between client A and client B.

In the following parts, we describe how clients with public IP addresses react to path failures. From the experiments, we find the clients reactions are related to the failure duration. There are three typical reactions of Skype to the path failures. We show the experiments in detail below.

1) Bring the link down until the call is dropped: In the first experiment we bring link 1 down, and don't bring it up. When client A hasn't received any packet from client B for around 1.7 seconds, it begins to send a TCP packet to B's public IP address. Figure 3 shows the interval time between the path failure and client A sends TCP packet to B. We can see that if there has been only the outgoing traffic for around 1.7 seconds, the client will start to send the TCP traffic to the other client. If the path has not recovered for around 14 seconds, the call will be dropped. Figure 4 shows the drop time in these experiments. We can see the time intervals are between 14.0s and 14.7s. In the experiment, we find that during the failure time, client B is the only client which client A communicates with. That's a suboptimal part of Skype policy dealing with the link failure. Because in this situation, only link 1 is down, Skype client can recover the call by relaying the traffic by other supernodes with link 2, but Skype doesn't do so.

2) Bring the link down for 2 seconds: In this experiment, we try to bring the path down for only 2 seconds, and then bring it up. The time that the network needs to recover may fluctuate a little for different experiments. For all these experiments, the call continues with UDP traffic from A to

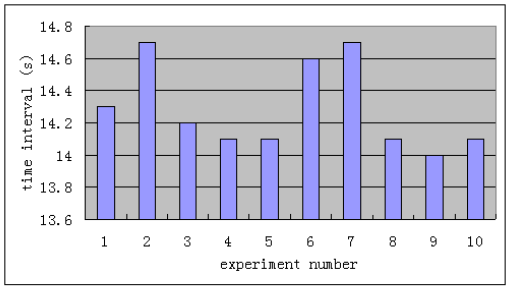

Fig. 4. The time when the calls are dropped

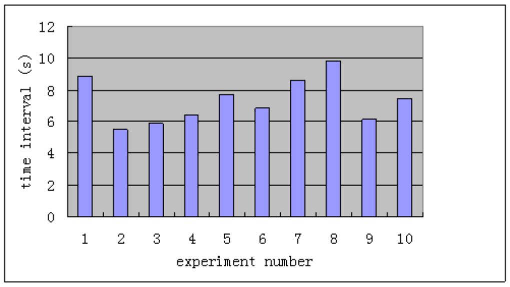

Fig. 5. The time interval between when the link is up and when the voice traffic changes into TCP

B's public IP address. So if the failure duration is short, Skype client tends to keep the current state.

3) Bring the link down for 7 seconds: In this experiment, we bring the path down for around 7 seconds. In that case, the call will recover with UDP packets when the path is back, but after that, it changes into TCP. Figure 5 shows the time interval from the path is up to the traffic changes into TCP. We can see that the time change to TCP is distributed from 5.5s to 9.8s. So if the link is down for 7 seconds, Skype client tends to use TCP instead of UDP, whic means Skype will use a more reliable protocol if the link is down for a longer time.

In this section, we introduce how Skype reacts to different path failure durations. For the failure duration mentioned above, the action of Skype is almost stable. But for the failure duration between these time points, Skype reaction is not fixed. From the experiment, we also find that if Skype client establishes the call without the relay supernode, no relay supernodes will be used for recovering from the path failure. Even though there are many available supernodes which can relay the call for A and B, Skype doesn't use them.

C. How clients react to path failures if the relay supernodes in needed

If caller and callee are both behind NATs or firewalls, a relay supernode will be needed for the communication [1]. If there is a path failure, how Skype client reacts to this? Can the backup relay supernode guarantee the connection during the failure? Does Skype switch to the backup relay supernode smoothly when the failure is detected? We set different scenarios to test Skype's reaction under different path failures. We find that Skype designs a strategy to recover from path failures for the

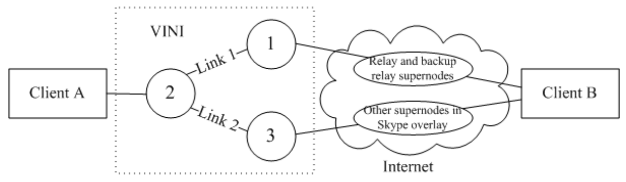
VINI
Client A
2
Link 1
Link 2
1
3
Relay and backup relay supernodes
Other supernodes in Skype overlay
Client B
Internet

Fig. 6. The experiment topology of the calls established with relay supernode

clients are behind the NATs or the firewalls. That strategy improves the Skype performance in network events. But the method is not robust to for all the path failures, especially, some common failures which will be shown below. In these cases, the network possesses the conditions for recovering the path, but because of the disadvantage of the strategy design, the call drops. We will show the recovering strategy in two scenarios. The first one is the paths to relay and backup relay supernodes are all down. In the case, we try to figure out how Skype reacts if all the paths are unavailable. The second scenario is only failing the current path between the clients and the relay supernode, and let other potential backup relay paths on. In this case,we try to figure out how Skype switch relay supernode to the backup ones. We test this situation under two routing protocol: OSPF and BGP. In both cases, we find Skype can't recover successfully because two Skype clients don't cooperate with each other correctly. We show the disadvantage of Skype design in both symmetric and asymmetric failure. The details are shown in the following parts.

1) Bring the paths to all the relay and backup relay supernodes down: We first try to fail all the paths from client A to all the relay and backup relay supernode. The topology is shown in Figure 6. Client A will connect to relay and backup relay supernodes through link 1, and other supernodes in the overlay network through link 2. When we fail link 1, Skype client will realize there is only out going voice traffic, so it starts to ping all the relay supernode and backup relay supernodes by TCP packets to test the paths. We only cut the paths from client A to all the relay and backup relay supernodes but don't cut the paths of client B, so the two clients will react differently. On client B side, when Skype pings all the relay and backup relay supernodes, it will get ACK immediately. On client A side, when it realizes the path failure and starts the ping, it can't receive ACK immediately. Skype designs different strategies for the two reactions. If the ACK is received immediately, the client tends to use the current relay supernode. But if the ACK can't be received, it will try to change the relay supernode. How to change the relay supernode is related to the failure time. In the following parts, we introduce three typical reactions.

a) Bring the paths to all the relay and backup relay supernodes down until the call is dropped: In the experiment, we bring link 1 down, and don't bring it up. That means all the relay and backup relay paths from A to B are down. When these paths have been down for around 1.7 seconds, the client will ping all the relay supernode and backup relay supernodes. If the client doesn't get replies, it will still keep sending the

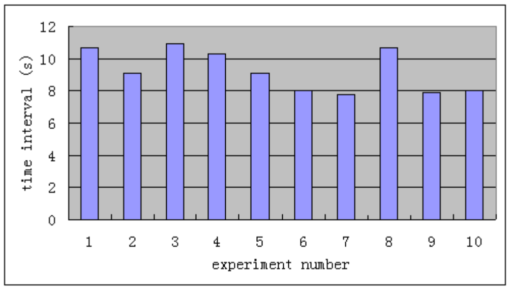

Fig. 7. The times from the path is down to the client changes the relay supernode

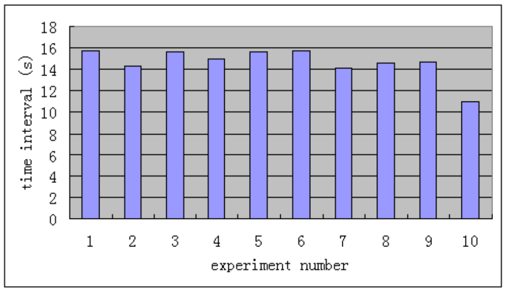

Fig. 8. The time from the paths are down to the call is dropped

UDP voice traffic to the current relay supernode. This state will keep for a while, then it will choose a new relay supernode, even though it gets no information whether the new one is available or not. The client will keep sending the UDP traffic to the new one until the call is dropped. We repeat the experiment for 10 times, and show statistics in several figures. Figure 7 shows the time from the paths are down to when client changes to the new relay supernode. From this picture, we can see that the changing time fluctuates in different experiments. The range is between 7.8s and 11s. Figure 8 shows the time from the paths are down to the call is dropped. The range of the time is between 11s and 15.8s.

During the path failure time, except the TCP packet to ping the relay and backup relay supernodes, there are also some TCP SYN packets to several new relay supernodes, and they don't show up at the establishment process of the call. That shows when all the paths are in bad condition, Skype will add some new backup relay supernodes to that session. The client can communicate with new relay supernodes successfully, but in all of the experiments, the client doesn't use them, and the call is dropped. Figure 9 shows the frequency of adding new relay nodes and the number of the new backup relay supernodes added in each time. The number of the bars shows how many times the new supernodes are added in each experiment, and the value of the bar shows how many new supernodes are added in each time. We can see that in each experiment, 2 new relay supernodes are added at least, but the call is still dropped. So Skype is not robust to the failures that all the relay and backup relay supernodes are unavailable.

Client B acts different from client A. When client B finds there is only out-going traffic, it starts to ping all the relay

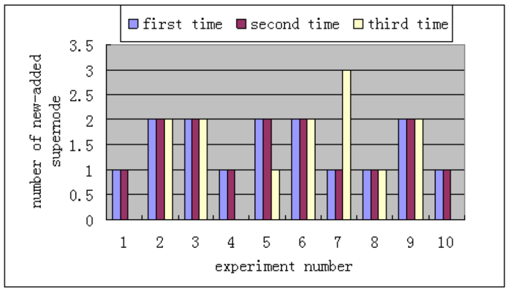

Fig. 9. The frequency of adding new relay nodes and the number of the new relay supernodes added

and backup relay supernodes. Because we only bring down the path between client A and the relay supernode, client B will get all the replies from these supernodes successfully. Client B will ping the relay and backup relay supernodes every second after the path has been down for around 1.7 seconds. After pinging for 8 times, it stops. When client B has not received any packet from client A for around 14 seconds, the call will be dropped.

From this experiment, we know that when all the paths are down, clients will first try to ping all the relay and backup relay supernodes, if it can get all the replies successfully, it will keep using the current relay node; if it can't get reply from all the relay and backup relay supernodes, it will try to establish the connection with some new relay supernodes.

b) Bring the paths to the relay and backup relay supernodes down for 3 to 7 seconds: If the path is down roughly between 3 seconds and 5 seconds, the call will recover by sending UDP voice traffic to the same relay supernode. That means if the link failure can be recovered in a short time interval, Skype client will use the same relay supernode with UDP. If the path down roughly between 5 seconds and 7 seconds, the call continues with the same relay supernode, but the protocol is changed from UDP to TCP.

c) Bring the paths to the relay and backup relay supernodes down for 10 seconds: We do this experiment in this setting for 5 times. The path failure duration is from 9.3 seconds to 10.6 seconds. From Figure 7, we know that the client will try to send UDP voice traffic to a new relay supernode if the path is down for around 9 seconds. That's what happens here. Around 9 seconds after the path is down, client A will try to send UDP voice traffic to a new relay supernode chosen from the backup relay supernodes. When the path is up, client A will use TCP protocol to communicate with the new relay supernode.

From this section, we know that, if the failure duration is short, Skype client will try to use the same relay supernode, and the protocol may change from UDP to TCP. If the failure duration reaches around 9 second, Skype client will try to switch to another relay supernode. As mentioned before, for the failure duration between these time points, Skype reaction is not fixed. It will act either the same as the shorter failure duration or the longer failure duration.

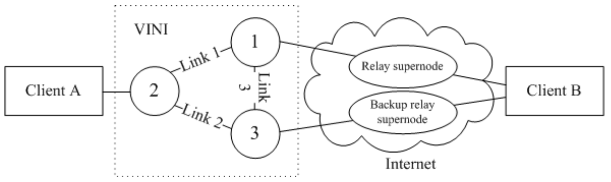

Fig. 10. The topology of bringing the path to the current relay supernode down

2) Bring the path to the current relay supernode down.: In this experiment, we try to only bring down the path between client A and the current relay supernodes, and leave the path to other backup relay supernodes up. In that case, we only affect the current relay supernode. So we can observe how Skype client reacts to the path failures using the backup relay supernodes and whether Skype could switch to backup relay supernode smoothly or not. We finish the experiments in two scenarios. One is with OSPF, and the other is with BGP. Using OSPF, we build symmetrical links, which means that the links can transfer traffic in two directions. Using BGP, we build asymmetrical links during the BGP update processes, and the asymmetrical links can only transfer traffic with one direction. We find shortcomings of Skype design in both scenarios which make the call dropped.

a) Experiments with OSPF in VINI: We first describe the scenario with OSPF setting. In this case, we try to emulate the topology in real networks. We make two links in VINI which client A can use to connect the relay supernode. The second one is a backup path. If the current link is down, it will run OSPF protocol to discover another best path. As shown in Figure 10, we make 3 nodes in VINI, and connect them. We set OSPF value of each link to be 350. Using VINI, we set the traffic to current relay supernode go through link 1, and traffic to backup relay supernode go through link 2. By connecting node 1 and node 3, we make link 3 be the backup path for client A to reach the relay supernode. If link 1 is down, it only affects the current path to the current relay supernode. Client A can still communicate with all the backup relay supernodes through link 2. It will take a while for client A to find the other path to the current relay supernode through link 3. We can control the recovering time by setting the hello message time and the dead-time interval in OSPF.

If we bring the current path down less than 6 seconds, the reaction of Skype is the same as we bring all the paths down in last section. In this section, we only describe the longer failure duration which brings some different reactions.

- OSPF hello message is set to 5 seconds
  We set the hello message of OSPF to be 5 seconds, and dead-time interval to be 10 seconds. The same experiment is done for 5 times. From Figure 3, we know that when the link has been down for around 1.7 seconds, the client will start to ping the relay supernode and the backup relay supernodes. In this case, the client will get the ACK packets from the backup relay supernodes. So unlike the scenario that all the paths are down, in this setting, the

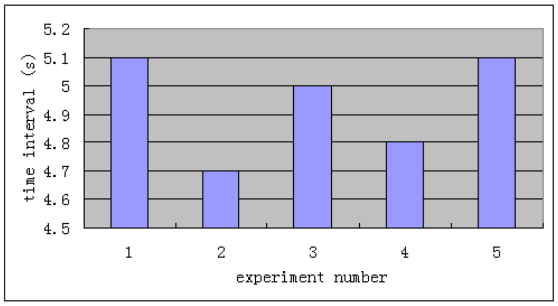

Fig. 11. The time when the client changes to send TCP traffic to the current relay supernode

client A will change to sending TCP traffic to the current relay supernode at around 5 seconds (The time is shown in Figure 11). The time is much smaller than the former case. That shows if the client can get ACK from the backup relay supernodes, it will take a shorter time for Skype to use the more reliable protocol. But client A still can't receive any ACK packet after changing into TCP. So client A stops sending traffic to the current relay supernode, and begins to send TCP voice traffic to one of the backup relay supernode. On client B side, it will still use the former relay supernode. So after the route table updates, the call continues with two relay supernodes. One is the former relay supernode, relaying traffic from client B to client A, and the other one is the new chosen relay supernode relaying traffic from client A to client B. We are also interested in which backup relay supernode the client prefers. Whether it tries to choose the one which is in the good condition path or not? So we do the following experiment. Let the traffic of current relay supernode and one of the backup relay supernode go through node 1, and other backup relay supernodes' traffic go through node 3. We only bring down link 1, and test if the new chosen relay supernode is in the good path or not. We do the experiment for 5 times. In 3 out of 5 experiments, client A chooses the backup relay supernode which is in the good path. In 2 out of 5 experiments, client A chooses the backup relay supernode in the down path. It's more reliable to choose the backup relay supernode in the good path. But Skype doesn't test the path quality of the backup relay supernode before it chooses one. That shows Skype doesn't use the optimal method in choosing the new relay supernode.

- OSPF hello message is set to 8 seconds
  In this case, we set the hello message of OSPF to be 8 seconds, and the dead-time interval to be 16 seconds. We repeat the experiment for 5 times. In this case the path failure duration shows to be longer than 14 seconds. Client A will start ping all the relay and backup relay supernodes at around 1.7 seconds after the path is down, and then send voice traffic in TCP to the current relay supernode. Because no ACK is received, client A will try to send packets in TCP to a new relay supernode chosen from the backup relay supernode. But the call is still dropped. To explain this, we need recall client B's action which is mentioned before. When pinging the relay and backup relay supernode, Client B will get all replies. So it will still use the same current relay supernode. That causes the problem. The path between current relay supernode and client A will be down for more than 14 seconds. If client B keeps sending traffic to the current one, during that time, client A will not receive any coming traffic from B. Even though client A can communicate successfully with the new relay supernodes, the call will still be dropped for not receiving traffic in around 14 seconds. This case is shown in Figure 12 and Figure 13. This scenario shows another disadvantage of supernode chosen during the path failure. Clients will not be informed that the callee can't receive the packets from its supernode. That means each client will only consider a part of path between itself and the supernode, and the rest of the path will be ignored.

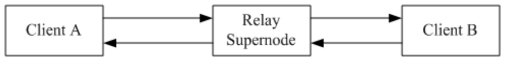

Fig. 12. The traffic flow before the path failure

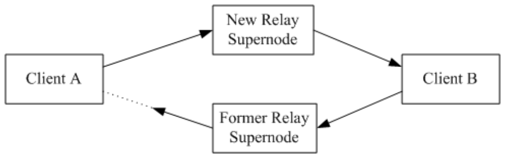

Fig. 13. The traffic flow after client A switches to a new relay supernode

b) Experiment with BGP in VINI: The dynamics of BGP brings certain impact on the traffic [5]. Except failures similar to OSPF which is discussed before, the update process of BGP will cause the asymmetric path in which only one direction traffic can be transferred successfully. When this happens, the update process will significantly affects the performance of the path, and it's common in the Internet [16]. Here, we discuss this scenario, and find out whether Skype is robust to this kind of failure.

We set both clients are behind NAT, and the VINI structure is shown in Figure 14. There are 6 ASes in VINI, and each AS has a node in it. The arrow lines show the provider and customer relationships [10], where swords point to the customers. The solid line shows the peer-to-peer relationship.

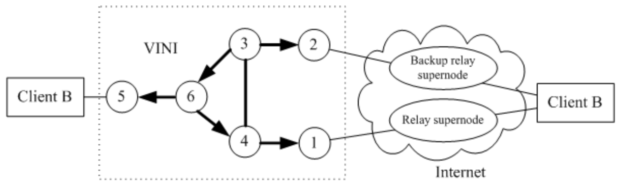

Fig. 14. The topology of building an asymmetric path during the path failure with BGP. The arrow lines show the provider-customer relationships, where swords point to the customers. The solid line shows the peer-peer relationship.

We let the current relay supernode connect to node 1, and backup relay supernode connect to node 2. According to the BGP protocol [13],client A will communicate with client B with the path node5-node6-node4-node1. Because of the loop avoidance property of BGP, node 3 won't broadcast its reachability of node 1 to node 6. So node 6 only knows one entry to node 1, and doesn't know there is a backup link to node 1 through node 3. However, in node 4's route table, there will be two entries to node 5. One is through node 3-node6, and the other is through node 6 directly. According to the BGP protocol, the path through node 6 is the best path. So in this topology, these nodes will know there are two routes from client B to client A, but only one route from client A to client B. The other path from client A to client B will only be noticed when the link between node 4 and node 6 is down. But it will take tens of seconds for BGP to finish the routing update.

If the link between node 4 and node 6 is down, node 4 will find another path quickly, because there are two entries in the routing table. But node 6 won't notice the backup path through node 3 until it receives node 3's broadcast. It will take even longer time for node 5 to notice the backup path because it is farther than node 6. So after the link between node 4 and node 6 is down, client B will reach client A immediately after the link failure(through node1-node4-node3-node6-node5), but it will take a while for client A to find the same path to client B. This forms an asymmetric path during the link failure process.

We do the experiments in this scenario, and we only fail the path of the current relay supernode, and the path to the backup relay supernodes are still on through node 2. When the link between node 4 and node 6 are down, client A will keep receiving the packets from B, because node 4 will find the backup path immediately. But client B won't receive any packet from client A, since it will take a while for node 5 to recover the backup path. For about 1.7 seconds client B hasn't received any packet from A, it will start to ping all the relay and backup relay supernodes. At the same time, client A will also receive the ping packet from relay and backup relay supernodes. But client A can't send the ACK packet to the current relay supernode successfully because the path from client A to client B is down. From the former experiment, we find out that if the client can't receive any packet from the current relay supernode, it will switch the relay supernode. But here, because client A can receive the voice packets from the current relay supernode, it doesn't switch to any backup relay supernode. It keeps sending traffic to the current relay supernode, and only changes the protocol from UDP to TCP. At last, because client B can't receive any packet from client A for around 14 seconds, the call is dropped. We can see that, during that 14 seconds, client A can communicate to every backup relay supernodes by TCP packet, but client A doesn't switch to any of them. That shows another disadvantage of Skype design.

From the experiments, we show that if the client can receive ping packet or voice packet from the relay supernode, it will not switch to other backup relay supernodes, even though

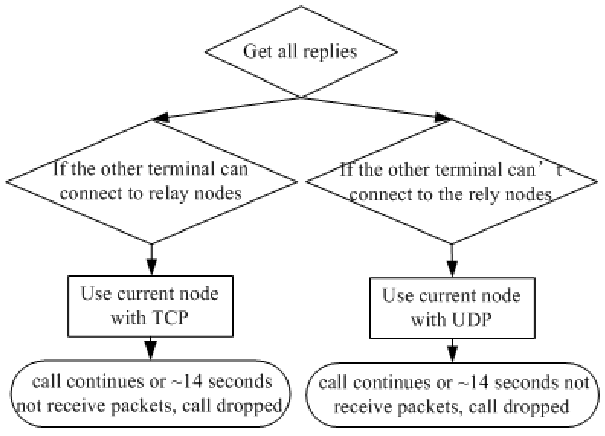
Get all replies
If the other terminal can connect to relay nodes
If the other terminal can't connect to the rely nodes
Use current node with TCP
Use current node with UDP
call continues or ~14 seconds not receive packets, call dropped
call continues or ~14 seconds not receive packets, call dropped

Fig. 15. Action A:the flow chart of Skype client actions if the client can get replies from all relay and backup relay supernodes

the current relay supernode can't send packet successfully to the other clients(as shown in experiments with OSPF) or the current relay supernode can't receive any packet from that client(as shown in experiment with BGP).

3) summary of Skype's reactions to path failures if both of the clients are behind the NATs or firewalls: Here, we give some flow charts(Figure 15 16 17) of how Skype clients behind the NATs or firewalls react to link failures. The process is as following: When the clients realize not receiving voice packets for around 1.7 seconds, it starts to ping all the relay and backup relay supernodes. With different replies, Skype client gives different actions.

If the clients can get replies from all relay and backup relay supernodes, it will still use the current relay supernode(Figure 15). If the other client can communicate with the backup relay supernodes, the client will change the protocol from UDP to TCP. If the other client can't communicate with any relay and backup relay supernode, the client will try to still use UDP protocol to send the packets to the current relay supernode. If the client doesn't receive any packet for around 14 seconds, it tends to drop the call.

After pinging all the relay and backup relay supernodes, if the client gets no reply from all of them, Skype will keep sending UDP traffic with the same relay supernode, and waiting the path to recover(Figure 16). If the path recovers in less than 9 seconds, Skype client will tend to use the current relay supernode, and change the protocol from UDP to TCP. If the path doesn't recover in 9 seconds, Skype will send UDP packet to a new backup relay supernode. If the path is back within around 14 seconds, it will send TCP voice packets to the new relay supernode, if not, the call will be dropped.

Figure 17 shows that after ping, the client only gets replies from backup relay supernodes, but not the current relay supernodes. There are two cases. First, if the client can't receive voice packet from the current relay supernode(as shown in experiments with OSPF), Skype client will try to send UDP traffic with the same relay supernode for around 5 seconds. If the link recovers in less than 5 seconds, it will try to use

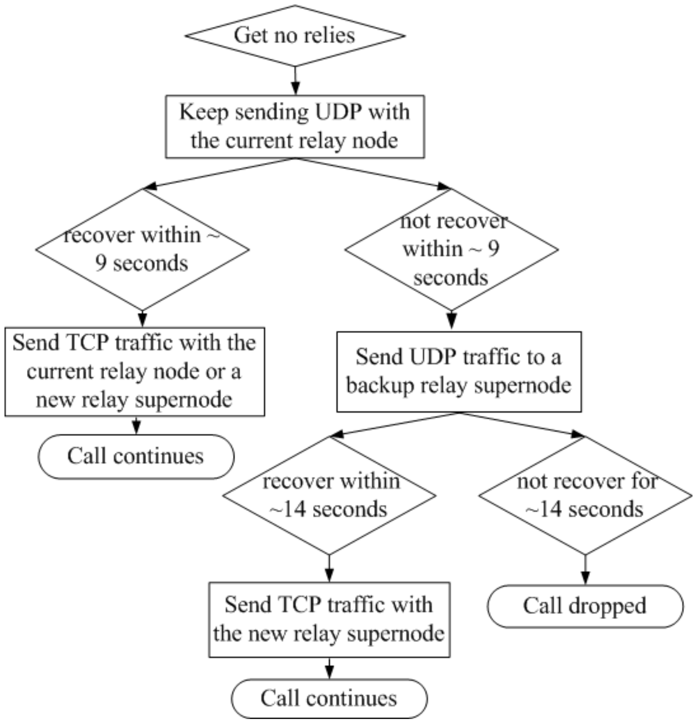
Get no relies
Keep sending UDP with the current relay node
recover within ~ 9 seconds
not recover within ~ 9 seconds
Send TCP traffic with the current relay node or a new relay supernode
Call continues
Send UDP traffic to a backup relay supernode
recover within ~14 seconds
not recover for ~14 seconds
Send TCP traffic with the new relay supernode
Call dropped
Call continues

Fig. 16. Action B: the flow chart of Skype client actions if the client gets no reply from the relay and backup relay supernodes

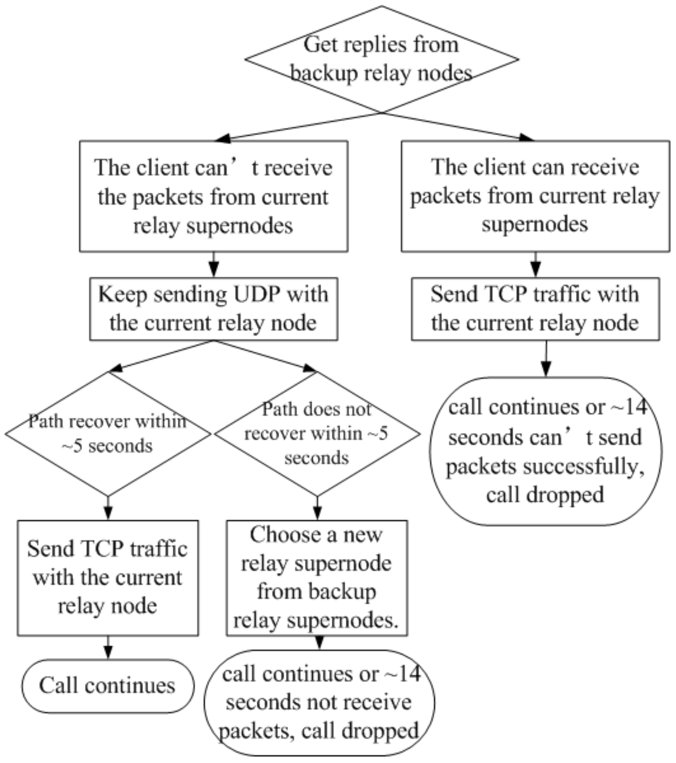
Get replies from backup relay nodes
The client can't receive the packets from current relay supernodes
The client can receive packets from current relay supernodes
Keep sending UDP with the current relay node
Send TCP traffic with the current relay node
call continues or ~14 seconds can't send packets successfully, call dropped
Path recover within ~5 seconds
Path does not recover within ~5 seconds
Send TCP traffic with the current relay node
Choose a new relay supernode from backup relay supernodes.
Call continues
call continues or ~14 seconds not receive packets, call dropped

Fig. 17. Action C:the flow chart of Skype client actions if the client only gets replies from backup relay supernodes

the same relay node with TCP traffic. If the link doesn't recover in 5 seconds, it switches to a new backup relay supernode. But the other client will still use the former relay supernode. That will cause the call to be dropped as shown in experiments with OSPF. Second, if the client can receive voice traffic from the current relay supernode, and can't send packets to it(as shown in experiments with BGP), it will not switch to any other backup relay supernode. If after around 14 seconds the path is still down, the call will be dropped. That's another disadvantage of switching the relay supernode. These situations give the common failure in the networks. In both cases, because of the disadvantage of the strategy, the performance degrades or the call is dropped.

VI. Conclusion

In this paper, we use VINI to do measurement study on Skype. We show how Skype client works with the supernode network in login, user searching, and call establishment processes. Further, we show how Skype client reacts to path failures in different scenarios. In the end, we point out some disadvantage in the strategy that Skype client uses to deal with the failures. By showing this, we try to give a hint for future design of peer-to-peer VoIP systems.

References

[1] Planet-lab. *http://www.planet-lab.org/*.
[2] tcpdump. *http://linux.die.net/man/8/tcpdump*.
[3] User mode linux. *http://user-mode-linux.sourceforge.net/*.
[4] wireshark. *http://www.wireshark.org/*.
[5] S. Agarwal, C.-N. Chuah, S. Bhattacharyya, and christophe Diot. The impact of bgp dynamics on intra domain traffic. *In Proceedings of ACM SIGMETRICS*, June 2004.
[6] S. A. Baset and H. Schulzrinne. An analysis of the skype peer to peer internet telephony protocol. *Proceedings of the INFOCOM'06*, Apr 2006.
[7] A. Bavier, N. Feamster, M. Huang, L. Peterson, and J. Rexford. In vini veritas: Realistic and controlled network experimentation. *Proc. ACM SIGCOMM*, August 2006.
[8] L. D. Cicco, S. Mascolo, and V. Palmisano. An experimental investigation of the congestion control used by skype voip. *WWIC '07*, 4517(153), May 2007.
[9] L. D. Cicco, S. Mascolo, and V. Palmisano. Skype video responsiveness to bandwidth variations. *NOSSDAV'08*, May 2008.
[10] L. Gao and J. Rexford. A stable internet routing without global coordination. *IEEE/ACM Transactions On Networking*, pages 681–692, December 2001.
[11] S. Guha, N. Daswani, and R. Jain. An experimental study of the skype peer-to-peer voip system. *Proc of IPTPS*, Feb 2006.
[12] E. Kohler, R. Morris, B. Chen, J. Jannotti, and M. Kaashoek. The click modular router. *In proceedings of SOSP*, December 1999.
[13] Y. Rekhter and T. Li. A border gateway protocol 4. *RFC 1771*, 1995.
[14] S. Ren, L. Guo, and X. Zhang. Asap: an as-aware peer-relay protocol for high quality voip. *Proceedings of IEEE ICDCS'06*, July 2006.
[15] K. Suh, D. R. Figueiredo, J. Kurose, and D. Towsley. Characterizing and detecting skype-relayed traffic. *Proceedings of the INFOCOM'06*, Apr 2006.
[16] F. Wang, Z. M. Mao, J. Wang, L. Gao, and R. Bush. A measurement study on the impact of routing events on end-to-end internet path performance. *SIGCOMM Comput. Commun. Rev.*, 36(4):375–386, 2006.
[17] H. Xie and Y. R. Yang. A measurement-based study of the skype peer-to-peer voip performance. *Proc of IPTPS*, Feb 2007.