# A System for Audio Signalling Based NAT Traversal

Ashish Patro, Yadi Ma, Fatemeh Panahi, Jordan Walker, and Suman Banerjee
Department of Computer Sciences, University of Wisconsin-Madison
{patro, yadi, fatemeh, jwalker, suman}@cs.wisc.edu

***Abstract*—Mobile users often connect through WiFi access points and frequently find themselves behind NATs that are built into common off-the-shelf home access points or enterprise wireless deployments. Punching a hole through the NATs to establish a P2P connection can be a challenging task for lay users. We present our system, ANT, that utilizes Audio signaling for NAT Traversal. With ANT, unlike other NAT traversal approaches, two mobile clients can establish a direct connection with minimal user intervention and without connecting to an intermediate server. ANT uses UPnP to obtain configuration information for NAT traversal which is then encoded using different audio frequencies and converted to audio sounds that are transmitted through the users' phones. Upon receiving the audio samples through the phone, the remote client converts them back into NAT traversal configuration data. Error correction is added to enhance the reliability of ANT and eliminate the need for retransmissions. Experimental results show that a TCP connection can be swiftly established between mobile clients behind NATs with no manual configuration, even in existence of heavy noise. We believe that ANT can be proved to be a simple, yet practical scheme for NAT traversal, which is as simple as dialing a phone number.**

## I. Introduction

Assume two friends want to set up a connection for a peer to peer application (file sharing, instant messaging, etc.). These friends are both using their wireless laptops, and are located behind different NATs. The specific application is not able to setup an appropriate connection because it does not know that the NAT blocks any unknown incoming connections. The individuals may not be immediately aware of their own public IP and port number to be used on the external side of the NAT to set up a connection. Currently, to set up a connection they need to either manipulate their routers' configuration, go through a relay server which will transfer the application's data back and forth between the peers, or perform a handshake through an external server in order to get each other's IP and port pair. In fact, except for very savvy users, few others can quickly figure out this information, and may find this process all too complicated. The goal of this paper is to make this process quite simple for lay persons using tools and technology that we are all familiar with — phones that are always within our reach at any given time. More specifically, we propose a system for Audio signaling based NAT Traversal, or ANT.

Today, majority of people have cell phones that they carry around with them. These friends who want to share data very likely know each other's cell phone numbers and can call each other. ANT proposes leveraging this existing opportunity for setting up a connection when both peers are behind NATs. With ANT, all this is very easy. One friend calls the other. Subsequently a lightweight application installed on their laptops extracts their public IP and port number of the local NAT. This and some other associated information is encoded using an audio form and communicated over the audio connection between the two phones (local laptop speaker to local cell phone, to remote cell phone over the phone network, and to remote laptop audio receiver). Once the configuration data is successfully communicated in both directions, the appropriate configurations are made by each laptop on their local NATs enabling the application to establish a direct connection. The process completes without the intervention of any server or the need for special configuration.

### A. NAT traversal and P2P applications

Network Address Translation (NAT) is a technology by which endpoints' IP address or IP address and port number are translated from private address realm to public address realm and vice versa. NAT techniques hide private hosts, thus hosts in private address realms cannot be reached directly from the public Internet. More over, NAT updates IP addresses and port numbers in IP packets as they are forwarded and thus breaks the common end-to-end semantics.

Figure 1 shows a simple NAT scenario. As shown in the figure, NAT allows a single device, such as a router, to act as an agent between the Internet and a local network. This means that a single IP address represents an entire group of computers. The NAT device is responsible for translating traffic coming into and leaving the private local network.

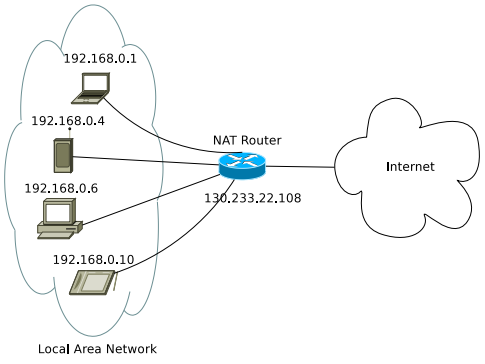


Fig. 1. A NAT Scenario

978-1-4244-8953-4/11/$26.00 

In normal client-server connections, NATed environments do not present a significant problem. In these connections, the client almost always initiates the connection with a server in the public address space, with a dedicated IP address. As services move from servers in static locations to mobile devices that are required to accept connections wherever they may be, NATs present a bigger issue, for example, when one mobile client is having a cup of coffee in a coffee shop and trying to set up a connection with another mobile client who is waiting for his airplane in an airport. In this scenario, these two mobile clients are likely both connected to routers in their respective locations, such that both of them are behind NATs. If they attempt to set up a peer-to-peer (P2P) connection directly, NAT traversal is non-trivial. In NATed environments, general NAT rules do not allow incoming connections to private hosts unless the private hosts initiate the connection or the NAT is specifically configured to forward the connections to the hosts. In P2P connections, where peers may act as both client and server, the burden of accepting connections falls equally to all peers.

An ideal NAT traversal technique should satisfy the following two requirements:

- Require minimal user involvement. Manual configuration requires user-intervention and is error-prone. Moreover, users may not have the technical knowledge nor the privileges to change NAT configurations.
- Does not depend on the availability, security, and the resources of intermediate servers.

Cell phone usage has increased dramatically in the United States and worldwide in the last ten years. According to research from Wireless Intelligence [16], in a single year period (from September 2005 to September 2006) the world's cell phone subscribers increased by 25% (from 2 billion to 2.5 billion). The UN's agency for information and communication technologies estimates that by end of 2008, the number of worldwide mobile users exceeded 4 billion, which is more than half the planet's estimated inhabitants [2].

In this paper, we ask this question: *Given the large availability of cell phones, can we perform NAT traversal for clients behind NATs through audio telephony without the involvement of any intermediate server or requiring the clients to perform router configuration changes?*

We believe that our ANT system will provide a unique and interesting solution for NAT traversal, one that can be easily used by any lay person, and is as simple as dialing a phone number (of a friend). After all, people world wide are very conversant in making phone calls today.

### B. Design overview of ANT

In this section, we present an overview of ANT and how it can be used in the mobile settings to ensure P2P connectivity. Figure 2 shows the ANT framework.

As shown in Figure 2, mobile clients A and B are using laptops which are behind NATs. Both A and B have cell phones and they know each other's phone numbers. The

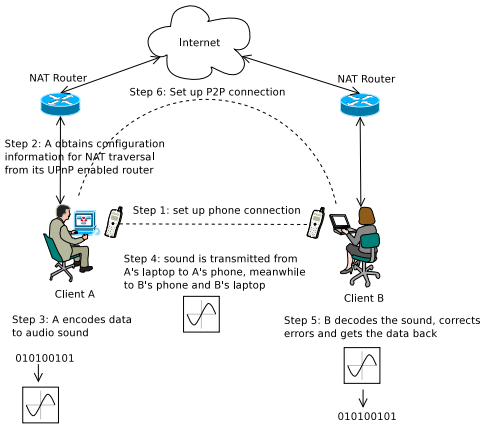


Fig. 2. Framework of ANT

following steps describe ANT's NAT traversal technique so that A and B are able to set up a P2P connection.

1) A dials B's phone number and B picks up her phone.
2) A's laptop automatically gathers the necessary information for NAT traversal. The traversal information includes A's external IP address and a mapped port number. We employ UPnP as a technique to automatically open a port on a gateway router so that A's laptop becomes accessible to the Internet via that port.
3) A's laptop encodes the traversal information and generates sound waves accordingly. The traversal information is encoded using $N$ ($N$ is a power of 2) different frequencies, with each frequency representing $log_2 N$ bits. A sinusoidal audio tone is generated for each frequency.
4) A's cell phone transmits the generated sound to B's cell phone. Simultaneously, B's laptop captures the sound.
5) B's laptop converts the audio sounds into frequencies and decodes the frequencies to get back the traversal information. A Fast Fourier Transform is used to obtain the original frequencies given off by the sound waves. Because of the existence of noise and distortion of the sound, an error correction mechanism is needed. We use Reed-Solomon error correction codes [14], which are able to recover errors by adding redundancy to the original data.
6) Now that B gets the correct traversal information for NAT traversal, B connects directly to A.

### C. Other potential applications of ANT

The idea of ANT can also be applied to other applications such as password exchange, product key exchange, and etc.

**Password exchange.** Suppose a user forgot his password of an online bank account, the common practice nowadays is that the user requests the password and the online banking system either sends his password through email or through phone message (SMS). Emails encounter variable amount of delays and sometime the delay could be long. SMS is considered instant, however the user has to enter the password by hand.

While ANT can provides an alternative that involves minimal user involvement. Once a user requests a password, the online banking system calls the user and sends the encoded password through voice, which is received by the user's phone and decoded by a light-weight software running on the user's computer. The only thing the user needs to do is picking up his phone.

**Product key exchange.** ANT could be extremely useful for those applications such as product key exchange where complicated keys are exchanged between the product company and authorized users. If a user somehow lost the product key for a software (e.g., Microsoft office suite), he needs to obtain a new product key if he need to reinstall the software. The process can be tedious and frustrating. First of all, the user needs to find out the Support telephone number for his area and make a phone call. Once his identity is identified through the phone call, the product key exchange process between the user and a support person usually will be something like: A as in apple, B as in Bob, ..., and etc, which is quite error-prone. While in this situation, ANT can help to alleviate the pain greatly. If the user registered his phone number when he bought the software suites, then his phone provides a good way to verify the user's identity. The user can request a new product key online by providing his phone number. Once the phone number matches the company's database, a product key is encoded and transmitted to the user's phone, which is in turn decoded by the user's computer. Even if the user still needs to call to verify his identity, ANT can be used to exchange the product key between the user and the support person.

We believe there are many other existing or future applications for which ANT provides a good solution or an alternative worth considering.

The rest of the paper is structured as follows. The details of how to convert binary data into audio sounds and how to get the data back from audio sounds in ANT is given in Section II. Section III presents how we perform NAT traversal using ANT. We evaluate the performance of ANT and summarize the experimental results in Section IV. Finally we discuss the extensibility and usage of ANT in Section V.

## II. Design of ANT

In this section, we present the core idea behind ANT. We explain how binary data is converted into audio samples which are sent over actual phones. Such binary data can be IP addresses and port numbers in the case of NAT traversal, or password and keys in the case of applications such as password and key exchange.

Once the sending side obtains data to be transmitted, it first converts the data into audio samples through modulation, then these samples are transmitted to the receiving side over phones. Upon receiving and capturing these audio samples, the receiver converts them back to get the original binary data. The audio samples received can be distorted due to noise, reflection, diffraction and refraction during transmission. To retrieve the original data, an error correction mechanism is built into ANT to correct errors and erasures caused by distortions.

### A. Modulate binary data to audio signals

The first question to ask in designing ANT is how to modulate binary data over an analog audio carrier. Keying is a traditional modulation technique. Keying techniques, such as Amplitude Shift Keying (ASK), Frequency Shift Keying (FSK) and Phase Shift Keying (PSK), modify the amplitude, frequency and phase of audio signals, respectively. In ANT, we explore a modulation technique analogous to Frequency Shift Keying, in which different audio frequencies are used to represent different binary bit sequences. While this scheme may not be particularly bandwidth efficient, it is adequate for our needs of communicating a small amount of data.

*1) Frequency range:* While the range of frequencies that any individual can hear varies for each person, the generally accepted standard range of audible frequencies is 20 to 20,000 Hertz (Hz).

We tested a variety a sound cards and phones to determine the frequency range that works well with all of them. Our experiments showed that 1000-4000 Hz is a good range that works well for all the sound cards we tested, while newer sound cards may have broader frequency range. Attenuation is especially prevalent at the higher frequency end of the spectrum. For most sound cards in normal laptops, the upper frequency limit for which the sound card generated audio without high attenuation was between 7000 Hz and 10000 Hz. The lower bound for the generated frequency was observed to be between 800 Hz and 1000 Hz. When testing on phones to determine their operating frequency range, we observed that it is narrower compared with sound cards of laptops and desktop machines. The phones performed especially poorly at high frequencies (greater than 4000 Hz). This is due to the fact that phones are designed to exchange human voices, whose maximum is below 4000 Hz for most people. To accommodate both phones and sound cards, we used frequencies in the range of 1200-3100 Hz, which worked adequately on all the devices we tested.

*2) Encode binary data:* In ANT, the binary data to be transmitted is encoded using $N$ ($N$ is a power of 2) different frequencies, with each frequency encoding $log_2 N$ bits of the binary data. For example, by using 8 different frequencies, each frequency represents 3 bits of the binary data. The distance between two consecutive frequencies times $N - 1$ gives us the frequency range needed for audio transmissions. Suppose the distance between two consecutive frequencies is 200 Hz and $N = 8$, $200 \times (8 - 1) = 1400$. In this case, the frequency range is 1400 Hz. When starting from 1000 Hz, the frequencies will be 1000, 1200, 1400, 1600, 1800, 2000, 2200 and 2400 Hz. As seen here, the number of frequencies required for the audio transmission is exponential to the number of bits encoded in each symbol (audio signal).

To reduce errors at the receiver, we would like the frequencies to be far away from each other. We observed that the

consecutive frequencies had to be separated by at least 30 Hz for the receiver to decode these frequencies with a manageable number of errors. We considered different schemes to encode binary data to frequencies. For example, if each frequency represents 8 bits, 256 different frequencies are needed. As such, using 256 frequencies requires a large frequency band (with a 30 Hz gap, a frequency band of 7650 Hz is required) which does not work properly for many devices. Phones especially can not handle this broad range. Finally we decided to use 16 different frequencies, with each frequency representing 4 binary bits. Thus, each byte of data required two symbols for transmission. With 16 frequencies we can spread them out over our spectrum better, and it leaves room for more frequencies.

We also considered other algorithms for encoding the binary data into audio signals. One such scheme transmits $N$ frequencies simultaneously, where $N$ different frequencies are used to encode $N$ bits in each symbol. Thus, by using $N$ different frequencies, each symbol could encode $2^N$ bits. Thus, this scheme could send $2^N$ bits within each audio signal. This method is more efficient but it complicates the decoding process by being more prone to errors caused by noise or when the receiver looses one of the frequencies during the capturing/decoding process.

*3) Generate audio signals:* Once the binary data is encoded into a sequence of different frequencies, a sinusoidal audio tone is generated for each frequency. To generate an accurate copy of the original signals on the receiving side, the original signals are sampled at discrete instants and they are usually sampled at uniformly spaced intervals. The sampled version is eventually used by the receiver to generate a copy of the original signal. Each sampled signal has a frequency spectrum. Let us assume that the highest frequency component in the signal is $f_{max}$. To reproduce a signal with $f_{max}$, the sampling frequency (the frequency at which samples are taken) must be at least twice the highest frequency component. In other words, the signal cannot be reproduced accurately unless the sampling frequency is at least $2f_{max}$, which is referred to as the Nyquist frequency for the signal. If the sampling frequency is lower than the Nyquist frequency, that is referred to as under-sampling. Since the highest frequency an individual can hear is 20,000 Hz, a sample frequency of 44K should be enough, and that is what we use as our sampling rate (44k samples per second). A high sampling frequency gives us more information about the original signal but also increases the decoding overhead.

*4) Synchronization:* When the binary data is successfully modulated into audio samples, they are ready to be transmitted. The sending and receiving entities need to know the start and end of the data transmission.

In ANT, we use two distinct frequencies as synchronization frequencies to start a transmission. These synchronization frequencies are used by the sender to inform the receiver that an audio transmission has started. The receiver checks for a known pattern of synchronization frequencies to determine when the actual data begins. This method is similar to the preamble used in wireless transmissions.

The synchronization stage has another use: the receiver uses the synchronizing tones to infer timing information about the audio transmission. It takes the average of the differences between the arrival times of the consecutive synchronization frequencies in the preamble to get an estimate of the rate at which the receiver is receiving the audio signals. For example, we send two synchronization frequencies, say $f_1$ and $f_2$, in a pattern such as $f_1f_2f_1f_2f_1f_2$. Suppose the receiver receives them at times $t_1$, $t_2$, $t_3$, $t_4$, $t_5$ and $t_6$, respectively. The average time difference between two consecutive arrival times, $T$, is calculated as:

$$T = \frac{(t_2 - t_1) + (t_3 - t_2) + (t_4 - t_3) + (t_5 - t_4) + (t_6 - t_5)}{5}$$

From our observations, $T$ is relatively constant for a specific laptop (depending on its sound card, CPU, and etc). We use $T \times C$ as the timing information to infer the missed audio samples in the following data transmissions, where $C$ is a constant. If the time difference between the arrival times of two audio signals is above $T \times C$, we infer that an audio sample is lost. Experimental results show that $C = 1.5$ works well for all the devices we tested. If the location of an error is known, which is called an erasure, the ability of error correction is enhanced by a factor of two, as shown in Section II-D. After the transmission of the synchronization, data transmission can start.

An end tone (using a unique frequency) is used to determine the end of audio transmission.

*5) Separate consecutive audio signals:* Each transmitted audio signal consists of multiple sinusoids of the same frequency so that the receiver can receive correctly. Due to this fact, it is difficult for the receiver to distinguish two or more consecutive audio signals from one if they are of the same frequency. The timing information obtained from the preamble is not enough as the audio signals are usually not evenly spaced out. The timing information is only used as a lower bound to detect missing audio signals.

One way to separate two consecutive audio signals is to send a "separation beep" after every audio signal, where a separation beep is an audio signal of a pre-defined frequency different from other frequencies used for data transmissions. However, the use of a separation beep would halve the data rate as each transmitted audio symbol requires a separation beep.

We decided to use two non-overlapping bands of frequencies, a higher frequency band and a lower frequency band, with each band containing 16 different frequencies. Instead of transmitting a separation beep, we transmit consecutive audio signals using frequencies in alternating frequency bands. For example, to transmit 0110, the first 1 is transmitted using a frequency in the lower frequency band while the second 1 is transmitted using a frequency in the higher frequency band. This makes it much easier for the decoder to recognize and differentiate between consecutive audio signals.

Even after using the two bands of 16 frequencies, ANT requires an overall frequency range of less than 2000 Hz.

This approach provides the same throughput compared to the method where each frequency represents 8 bits and separation beeps are used between consecutive symbols, while using a narrower frequency range.

ANT's current design uses a total of 35 frequencies, two non-overlapping bands, each consisting of 16 frequencies (separated by 50 Hz each), plus two synchronization frequencies for the preamble and one end-tone frequency.

### *B. Convert audio signals into binary data*

At the receiver, the captured audio samples are decoded to obtain the original data. We do it by using Fourier transforms, specifically Fast Fourier Transform (FFT). The main advantage of FFT in our case is that it enables us to efficiently compute the Fourier transfer on the received sound samples in real-time.

ANT repeatedly applies the FFT to the most recent sound samples in the receiver's sound buffer to extract various frequencies out of the sound samples. We scan the frequencies and check for the presence of synchronization frequencies. In case of the presence of synchronization frequencies, the receiver saves the sequence of synchronization frequencies received to determine whether it has received the preamble sequence. Once it receives the preamble, the receiver gets ready to receive the data audio samples. The receiver also uses the arrival times of the synchronization frequencies at the receiver to estimate the rate at which audio signals are received at the receiver.

After synchronization, the receiver looks for the presence of data frequencies in an alternating order in the two frequency bands. Each frequency is decoded to a particular sequence of bits. The end tone signals the end of data transmission.

### *C. Noise filtering*

An important requirement for the proper functioning of ANT is noise filtering. We filter out all the noise outside the frequency band (in our case, 1200 Hz to 3100 Hz) being used for transmission of the audio signals. We also set a lower bound on the intensity of the audio signals to filter out most of the background noise.

When converting audio signals back to frequencies, we observe that each frequency lies in a very narrow frequency band. We set it to $(F-15, F+15)$ Hz , where $F$ is the frequency being observed, and 15 is called the "Frequency Error Bandwidth". This range is used to reduce the overhead for the Fourier transform as the buffer size required for a Fourier transform increases with the increase in precision requirements. Also, the maximum intensity of the received audio signal is not always at the center frequency $F$ due to distortion during transfer. By combining these parameters, ANT becomes robust to noise as it is only sensitive to noise in small frequency bands and only if that noise is above a certain intensity.

### *D. Error correction*

After the receiver converts the sounds to binary data, the binary data may differ from the original data. This is is due to the error introduced by noise in the environment or the errors during audio transmission/reception.

Therefore an error correction mechanism is needed to correct errors observed at the receiver. We decided to use Forward Error Correction (FEC) because FEC avoids using a back-channel. One advantage of FEC is that retransmission of data can often be avoided (at the cost of higher bandwidth requirements during each transmission). FEC is applied in ANT because retransmissions are relatively costly and difficult to do (the users need to tell the application to send/receive the tones again).

There are many types of FEC codes, but amongst the classical ones the most notable is Reed-Solomon coding [14] because of its widespread use on the Compact disc, the DVD, and hard disk drives. ANT employs Reed-Solomon error correction codes. A good property of Reed-Solomon is that the error-correcting ability of any Reed-Solomon code is determined by $n-k$ ($k$ is the number of actual data symbols per $n$ transmitted symbols), the measure of redundancy in the block. If the locations of the erroneous symbols are not known in advance, then a Reed-Solomon code can correct up to $(n-k)/2$ erroneous symbols, i.e., it can correct half as many errors as there are redundant symbols added to the block. Sometimes error locations are known in advance (these are called erasures). A Reed-Solomon code is able to correct twice as many erasures as errors, and any combination of errors and erasures can be corrected as long as the relation $2E+S<n-k$ is satisfied, where $E$ is the number of errors and $S$ is the number of erasures in the block.

We use Reed-Solomon codes to generate a set of redundant samples corresponding to a set of actual samples. As discussed above, it increases the overhead of transmitting data but nevertheless removes the overhead of a back-channel from the receiver to the sender to acknowledge the received data. This greatly simplifies ANT as a single channel from the sender to the receiver is sufficient to send the binary data using audio samples.

Since we encode 4 bits at a time within each audio signal (symbol), we use Reed-Solomon error correction algorithm over a field of GF(16). There is a trade-off between higher throughput and greater robustness to noise. We generated two redundant 4-bit nibbles per four actual 4-bit data nibbles. We found that this level of redundancy was sufficient for most of the cases and could handle 2 erasures or 1 error at a time per six transmitted signals. A majority of the errors in our experiments were caused due to erasures when the decoder could not capture an audio signal at a given time, and errors in captured audio signals were comparatively less frequent. In our experiments, we increased the redundancy in some non-conducive situations but we never required more than 4 redundant symbols per 4 data symbols at a time.

## III. Using ANT for NAT Traversal

Network Address Translation (NAT) maps addresses on a local network segment to the global IP address space. This is done by modifying the IP address and TCP port fields of

the packets as they pass through a NAT device. NAT devices introduce unique problems to the regular Internet connection model as special steps need to be taken to connect with machines behind a NAT.

The primary issue with NATs is that computers attached to a NAT device can only make outbound connections, unless they perform complicated configuration changes on the router. These machines are unable to listen for peers trying to connect to them, and they cannot connect to other computers behind a NAT, as the problem applies to both machines. Because neither of the peers are able to accept inbound connections, it is not possible to establish a connection using normal techniques.

There are four different configurations for NAT port mappings that must be considered when developing a NAT traversal technique [6].

- Full cone mapping. In full cone mapping, there is a well established mapping and anyone from the public Internet that wants to reach a client behind a NAT only needs knowledge of the mapping scheme in order to send packets to it.
- Restricted cone mapping. In a restricted cone NAT, the external address and port pair is only opened up when the internal computer sends out data to a specific destination IP.
- Port restricted cone mapping. A port restricted cone type NAT is almost identical to a restricted cone. It differs in that the NAT blocks all packets unless the client had previously sent out a packet to a matching IP and port.
- Symmetric mapping. The last type of NAT symmetric is different from the first three in that a specific mapping of internal address and port pair to the NAT's public address and port pair is dependent on the destination IP address that the packet is sent to. As in the case of the restricted NAT, the external address and port pair is only opened up once the internal computer sends out data to a specific destination.

STUN [8] and NUTTS [7] address the NAT traversal in peer to peer applications for the first three types of NAT port mappings. By connecting to an external server, peers are able to exchange their IP:port pair and then communicate directly. These approaches do not work for the symmetric NATs as the port mapping is specific to the destination IP address. However, studies over the years show that symmetric NAT is becoming less common since it cannot effectively support online gaming or similar applications [9].

Popular P2P applications today address the NAT traversal problem in different ways. Usually an intermediate machine is used to transfer information between the two peers. In any NAT traversal approach, to get around the limitations added by the NAT mapping scheme some communication is required between endpoints so that they can at least be informed of each others public IP and port pair.

The NAT device must be aware of the port on which to accept the connection, and of the local address to forward the request to. We need a mechanism to automatically open a port on a NAT router so that it becomes accessible from the Internet without the need to change the router configuration manually. This is essential because many users do not have the technical knowledge to modify the router's configuration. One option that we considered for this purpose was SNMP [11]. Unfortunately, SNMP has some pitfalls which makes it unsuitable for this approach. SNMP is not always implemented on entry level routers. Furthermore, SNMP's interface for opening ports on a router is not standardized. Therefore, one would need to develop specific SNMP routines for each vendor's SNMP aware routers.

Universal Plug and Play (UPnP) [4] is an alternative that has a significant advantage over SNMP. There is a defined interface in Internet Gateway Devices (IGD) [4] for the purpose of mapping ports to clients. Therefore, all IGDs can work with the same piece of code. The IGD Protocol is implemented via UPnP. Many routers and firewalls expose themselves as Internet Gateway Devices. This allows any local UPnP controller to perform a variety of actions, including retrieving the external IP address of the device, enumerating existing port mappings, and adding or removing port mappings. By adding a port mapping, an internal client allows an external client to connect to it.

UPnP was established as a standard in 1999 and benefits from wide scale adoption. Most new broadband routers are UPnP capable, and a growing number of devices have it enabled.ANT performs UPnP calls to get the required traversal information and to punch a hole in the NAT device from which it will accept data sent from the other peer. We use Java UPnP libraries from sbbi website [13].

*a) Getting IP address information:* There are two important IP addresses that must be found in the discovery process. First, the private IP of the peer is looked up from the local interface information. This address is used as part of the mapping, in order to set up the forwarding table in the NAT device. Second, the public IP seen by the outside world must be found to be sent to the other peer for a P2P the connection. To get this address, UPnP is leveraged to query the external address of the IGD without requesting it from an external server.

*b) Getting port information:* UPnP is used to create a mapping on the NAT device. A port is chosen in the allowed range, and a mapping is attempted. If the port is already taken by an application, another port will be tried. When this goes through, the same port is used locally to accept new connections. When this port information along with the public IP address is transmitted to the other peer, it will have all the information needed to establish a connection.

*c) Setting up Peer to Peer connection:* Once one peer gets its IP address and creates a port mapping, it converts its IP address and port number into audio sounds as stated in Section II. The other peer captures the sound from his cell phone, decodes it to get back the traversal information and set up a direct connection with the other peer.

## IV. Evaluation

In this section, we evaluate the performance of ANT using different laptop models, cell phones models, and under various noise conditions.

As shown in Figure 2, we set up two different NATed networks using two wireless routers, with one client behind each NAT. This setting prevents the two clients from establishing a connection directly. To establish a direct P2P connection across the NATed networks, client A calls client B using his cell phone, and client B answers the phone. Then client A starts transmitting sound waves when B is ready to capture. The only thing client A needs to do is to click a "Start" button, while client B only needs to click a "Capture" button. By running ANT, client A's laptop automatically opens a port, prepares a socket that listens to the port, converts its external IP address and port number into audio sounds, and transmits the sound waves from his cell phone to client B's cell phone. Upon receiving the audio signals at B's cell phone, B's laptop automatically captures the sound waves, converts them back to get configuration information, and starts a connection to A by sending a hello message. Once the message is received by A, A replies with a hello message. If both A and B get a hello message, this means a P2P connection behind NATs is set up successfully.

### A. Experiment setup

We used the standard router models from Linksys, Belkin and D-Link in our experiments to setup separate NATs for the two clients. The experiments were performed under various noise conditions. We used SPL (sound pressure Levels) to characterize the background noise. The noise level was measured on a scale of 0 to -80 dB where a value closer to 0 dB represented a high amount of noise and a value closer to -80 dB represented a very low level of noise.

In our experiments, we categorized the background noise conditions into three general types:

- No or Light noise. A client was located in a quiet environment without noticeable external noise or very low background noise. To observe this kind of noise, we performed the experiments in quiet offices and conference rooms. The noise was observed to be around -55 dB to -50 dB.
- Medium noise. A client was located in an environment where there is some constant noise. This kind of noise could be caused by people chatting occasionally or constantly, people walking around and occasionally opening or closing doors or some desktop machine generating noise in the background. To observe these kind of noise, we performed experiments at home, in hallways and offices of the department building and in conference rooms when there are people talking to each other. The noise was observed to be around -32 dB.
- Heavy noise. A client was located in an environment where there is heavy noise. To observe this kind of noise, we performed experiments in a computer lab where there are a large number of servers running making continuous background noise, plus students working in the lab talking to each other. The noise was observed to be around -20 dB.

TABLE II
A summary of cell phones used for the experiments

| Name | Brand | Model | Sound condition |
|---|---|---|---|
| C1 | Sony Ericsson | W580i | bad |
| C2 | LG | CF360 | very good |
| C3 | Nokia | N95 | good |
| C4 | HTC | P4600 | fair |

Table I summarizes the laptops we used for the experiments (brands, models, sound card conditions and operating systems). Table II summarizes the cell phones we used, their brands and models. As seen from the tables, we test ANT using a variety of devices with various sound card conditions. In addition, the laptops are running on different operating systems: Windows XP, Windows 7, Windows Vista, etc.

### B. Experimental results

We performed experiments using various combinations of noise conditions, laptops, and cell phones. For the 3 types of noise conditions used, altogether, there were 9 possible noise combinations for two clients. For example, if client A was located in a medium noise environment, B could be present in a no/light noise, medium noise or a heavy noise environment. We did our experiments using a number of representative settings that covered the majority of the common scenarios. We discussed in Section II that phones were the limiting factor in ANT's performance for the chosen range of frequencies, so we concentrated on the environmental conditions and the cell phones used in the experiments. The laptops worked well for all the experiments in this section.

For a given noise combination for clients A and B, we randomly chose a number of laptops and cell phones, and repeated the experiment for each setting for around 4-6 times. For each setting, we reported the percentage of times that data was transmitted correctly on the first attempt and a connection between two clients was set up successfully. In this section, we report the percentage of successes by transmitting configuration data only once. In ANT, we can always increase the number of attempts (transmit data multiple times) if failure occurs until a connection is successfully setup or the number of unsuccessful attempts exceeds a number, since each transmission only takes seconds to finish.

*1) No/Light noise:* Figure 3 shows the percentage of successes when both clients were in a quiet environment. The x-axis represents different phone settings, in the format of A $\rightarrow$ B, where A is the name of the cell phone at the generating side, and B is the name of the cell phone at the receiving side. The y-axis shows the percentage of successes for each setting. This set of experiments were performed when the two clients were located either at home or in conference rooms. The background noise at each side was about -52 dB.

TABLE I
A SUMMARY OF LAPTOPS USED FOR THE EXPERIMENTS

| Name | Brand | Model | Sound card model | Sound condition | OS |
|---|---|---|---|---|---|
| L1 | Lenovo | T60 | SoundMax Integrated Digital HD | fair | Windows XP |
| L2 | Sony | VAIO | Intel HD Audio | very good | Windows Vista |
| L3 | Lenovo | T400 | Conexant 20561 SmartAudio HD | good | Windows 7 |

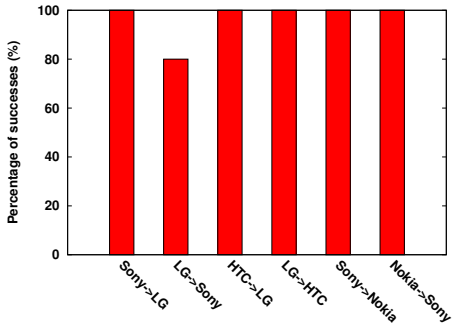


Fig. 3. Percentage of successes when both clients are in quiet environments with no/light noise

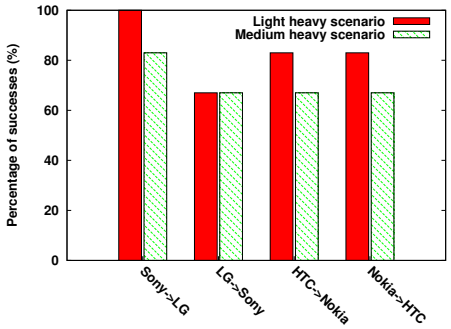


Fig. 4. Percentage of successes in noisy environments with heavy/light or medium noise

We observed that ANT performed really well in the presence of little or no noise. The only pair that failed the test was the LG $\rightarrow$ Sony pair. This occurred because the Sony phone had a bad speaker and was unable to reproduce the sounds properly, which caused more erasures than that could be corrected at the receiving side.

*2) Heavy noise/Light,Medium noise:* We did this experiment in a scenario where one of the peers was present in a server room with constant loud background noise caused by the servers running in the room. For the other peer, we used two different scenarios. In one of them, the second peer was present in a department office shared by 3 people. There was some constant noise caused by the desktop machines in the room, people occasionally talking and a few disturbances from outside the office room. In the other scenario, the second peer was present in a quiet room representing light noise/no noise conditions.

Figure 4 shows the experimental results for the two scenarios:

1) First Peer (with LG, Nokia) in heavy noise environment and the second peer in the light noise environment (with Sony, HTC).
2) First peer (with LG, Nokia) in heavy noise environment and the second peer in the medium noise environment (with Sony, HTC).

As just mentioned, the LG and Nokia phones were used by the first peer in the heavy noise conditions and the Sony and HTC phones were used by the second peer in the light or medium noise conditions.

For each cell phone combination A $\rightarrow$ B (transmission from A to B) , we present two results in Figure 4. Light heavy scenario is used to denote the scenario when the first peer is in the heavy noise condition and the second peer is in the light noise conditions. Medium heavy scenario is used to denote the scenario when the first peer is in the heavy noise condition while the second peer is in the medium noise conditions.

The LG phone had a good Loudspeaker and was generating the received signals at a good volume, so ANT worked great when the LG phone was at the receiving side, even though it was present in a environment having heavy background noise. The other combinations except LG $\rightarrow$ Sony worked well under these conditions. The failures could be caused by either erasures at the receiving side due to the heavy noise, and/or corruption of the signals at the generating side due to the noisy conditions. Also the Sony phone's loudspeaker was not good at reproducing the received tones at a high volume, so ANT was more susceptible to erasures in this case.

*3) Speak test (people talking):* In this experiment, ANT was tested against errors when there were human speakers close to the laptop. We observed that the speakers in the background did not cause problems, so we tested for the situation when users with cell phones themselves were talking while the transmissions were going on. We tested ANT in four scenarios:

1) S1: The generating peer is speaking with a soft voice, the receiving peer is quiet with no/light background noise.
2) S2: The generating peer is speaking with a loud voice, the receiving peer is quiet with no/light background noise.
3) S3: The generating peer is speaking with a soft voice, the receiving peer is quiet with heavy background noise.
4) S4: The generating peer is speaking with a loud voice, the receiving peer is quiet with heavy background noise.

Figure 5 shows the results under these conditions. The loud voice was around -15 to -20 dB while the soft noise was around -30dB. Under these scenarios, due to the speaking person's voice, the transmitted audio signals got corrupted at the generator which resulted in problems while decoding the sound samples at the receiver. ANT eventually succeeded

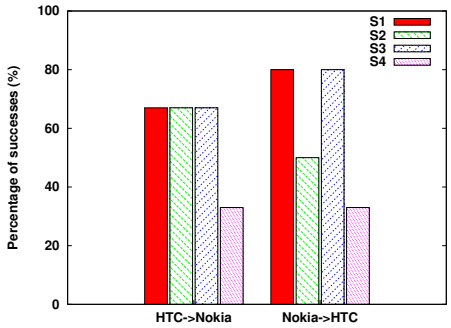


Fig. 5. Percentage of successes in speak tests

in establishing a connection in about 2-3 attempts of the audio transmission. We concluded through this experiment that, ANT's performance gets affected when there is a speaker close to the laptop while the transmissions are going on. ANT is more sensitive to nearby speakers compared to the background noise.

The only situation where ANT encounters major problems is in the presence of background music, which is expected because music generates a vast range of frequencies and this interferes with ANT by inserting spurious symbols into the audio transmissions which causes problems for the decoder.

## V. Discussion

In this section, we discuss the throughput of ANT, its extensibility and usage, and the distinction between ANT and dial-up modems.

*a) Throughput:* The throughput of audio data is dependent on the amount of redundancy used and the rate at which the transmitter is generating the audio signals. We observed that the rate at which the transmitter generates the audio signals is dependent on the sound card used by the transmitter. For the same set of parameters used for audio signal generation, different sound cards reacted differently. The rate ranged from 1 beep per second to 7-8 beeps per second. Due to the use of a unidirectional communication channel and a variety of factors(noise, sound cards), employing the necessary amount of redundancy is essential to successfully transmit the data in a single attempt. The slow data rate should not be an issue as the audio signals are only used to transmit the initial configuration information in order to setup a P2P network connection, which just takes a few seconds.

*b) Extensibility of ANT:* Not only mobile users with cell phones and laptops can benefit from ANT, but also can other users. As long as a user has a computer (no matter its a laptop, desktop or PDA) and a phone (cell phone or landline phone), they can use ANT. Other than automatic NAT traversal, ANT can be extended to other applications. For example, ANT provides an alternative for file transfers when users do not have Internet access. Small files can be converted to audio sounds and transmitted through phones. ANT is especially useful for transmitting a small amount of data. If we require a secure data channel to minimize security risks, ANT can be used as a building block. Public key and password exchange would be examples of this types of smaller data that would be possible to transmit through ANT by providing a secure channel.

*c) Using ANT more effectively:* There are some precautions that can be taken to maximize the performance of ANT. For example, the person at the receiving end of the phone connection should turn on the cell phone speaker and use it at the maximum volume, so that it is easy for the decoder software to process the audio signals. Also, at the receiving end, the phones should be kept close to the laptop microphone for good performance. The sender should generate the audio signals at an intensity greater than the background noise to reduce the number of erasures at the receiver.

*d) UPnP considerations:* UPnP is a standard which should be available on any new Internet gateway router, but it may not be enabled. While it is easy to enable, many users may not want to enter the router configuration to do this. Mobile users have the added problem that they do not have the flexibility of enabling it on the router they are connected to. Since many applications use UPnP and public providers want to provide good service to their users, we believe that a good majority of routers either have UPnP enabled or the service providers can be convinced to enable it.

*e) ANT versus dial-up modems:* The high level approach of exchanging signaling information for NAT configuration in ANT can be considered similar to the high level approach for establishing communication by dial-up modems. In a sense, both systems use the phone network to communicate some data. However, the applications are quite different. ANT provides NAT traversal, while dial-up modems provide basic Internet access services. Dial-up modems use the phone network as the primary data path. Hence, the encoding of data in the case of dial-up modems is quite efficient. In contrast, ANT's requirement and use of the phone network is really minimal, because ANT used it to communicate a minuscule amount of control messages. Hence, the data encoding mechanism we use in ANT is quite simple, but is adequate for the specific needs of our system.

## VI. Related Work

In this section, we discuss some of prior approaches for NAT traversal (including those commonly used by various P2P applications) and Audio based data transmissions.

### A. NAT traversal techniques

Our approach utilizes UPnP for opening a port and getting a mobile client's public IP and port pair without the use of an intermediary server. Instead the client exchanges this pair through sound waves over an audio channel. STUNT [8] and NUTSS [7] enable the clients to get the public IP and port pair that the outermost NAT assigns to them by sending a request to an external server. Peers perform a handshake through this server to get the IP and port pair and subsequently connect directly to each other. These approaches do not require any specific configuration on the router. However, by relying on an external server for negotiating a connection, these approaches

are more prone to attack. Moreover, a successful peer-to-peer connection is dependent upon the availability of the server.

TURN [12] allows a host to select a globally-addressable TCP relay, which can subsequently be used to bridge a TCP connection between two NATed hosts. The advantage of TURN is that it works for all types of NATs, while the above approaches are not effective for symmetric NATs. However, TURN does not allow direct connectivity between NATed hosts, and all the data is transferred through the external server. Aside from security issues present here, this approach requires extensive resources on the server for handling many connections transmissions, so scalability and cost is a concern.

All the above approaches assume that both peers set up a connection to a secure external server, while our method works independent of this intervention.

### B. Peer-to-Peer applications and NAT traversal

Different peer-to-peer applications have chosen different approaches for solving the NAT traversal issue. In Kazaa [10], if only one peer is NATed, it requires the NATed peer to initialize the connection. This approach will not work if both peers are behind NATs. Some applications such as BitTorrent [1] recommend users configure their NAT boxes to forward incoming requests on specific ports to one computer behind the NAT. This approach is not effective as many users do not have the technical knowledge to change the router configuration, and in mobile settings users do not have control over the router to make configuration changes. Groove [5] routes peer-to-peer communication through central relay servers, and Skype [15] routes connections through instances of the application that are running on open non-NATed computers. In these scenarios the burden of the communication will be on the central servers or on other clients. Compared to these approaches, our approach has no scalability issues as it only uses the resources of the two clients. Furthermore, setting up the connection over cell phones works even when both peers are behind NATs.

### C. Audio based data transmissions

In recent concurrent work, Hermes [3] presents a technique for data transmission through audio encoding over voice channels. Our technique is similar to this technique as we encode initial NAT traversal information into audio signals and transmit them over a voice channel. But, we mainly use the voice channel as a control channel for transmitting a small number of audio signals. We focus more on creating a robust and effective mechanism for audio transmissions (through synchronisation, error correction). We test ANT in the wild in a variety of conditions and with a variety of devices to validate the performance of the audio transmissions in these conditions.

## VII. Conclusion

In this paper, we presented a system for NAT traversal by transmitting out-of-band information through phones by converting it into audio signals. Mobile users are relieved from the pain of, if possible, configuring a NAT manually. The only user intervention required in ANT is the process of dialing the phone number of the other peer and keeping the phone connection open for a short duration while the connection is being set up. ANT performs well in most of the scenarios that we tested. These scenarios represent common environmental situations and standard equipment in use today. In environments where the noise is low, ANT is able to successfully transmit the audio information in only one attempt in almost all cases. In cases with medium and heavy noise, ANT is resilient to erroneous conditions and is able to successfully transmit the audio information in two to three attempts, as long as the received audio signals have a higher intensity than background noise. We believe such an overhead is tolerable as the duration of audio transmissions for NAT traversal information is short and the potential benefits are great for mobile users. We believe that our relatively simple approach of audio signaling based NAT traversal is actually quite effective. The system can also be used as a building block for systems where the audio channel is used for short, robust and secure data transmissions.

A video demonstrating the working of ANT is made publicly available at:

http://www.cs.wisc.edu/wings/projects/ant

## VIII. Acknowledgement

Yadi Ma and Suman Banerjee were supported in part by US NSF awards CNS-1040648, CNS-0916955, CNS-0855201, CNS-0747177, and CNS-0627589.

## References

[1] BitTorrent. Bittorrent faq. http://www.dessent.net/btfaq.
[2] U. N. Centre. Number of cell phone subscribers to hit 4 billion this year. http://www.un.org/apps/news/story.asp?NewsID=28251&Cr=Telecommunication&Cr1, 2008.
[3] A. Dhananjay, A. Sharma, M. Paik, J. Chen, T. K. Kuppusamy, J. Li, and L. Subramanian. Hermes: data transmission over unknown voice channels. In *MobiCom '10: Proceedings of the sixteenth annual international conference on Mobile computing and networking*, pages 113–124, New York, NY, USA, 2010. ACM.
[4] U. Forum. Upnp devices. http://www.upnp.org/.
[5] Groove. Get into the groove: Solutions for secure and dynamic collaboration. http://technet.microsoft.com/en-us/Magazine/2006.10.intothegroove.aspx.
[6] S. Guha and P. Francis. Characterization and measurement of tcp traversal through nats and firewalls, 2005.
[7] S. Guha, Y. Takeda, and P. Francis. Nutss: A sip-based approach to udp and tcp network connectivity. In *SIGCOMM'04 Workshops*, 2004.
[8] T.-C. Huang, C.-K. Shieh, and W.-H. L. Y.-B. Miao. Smart tunnel union for nat traversal. In *NCA'05*, 2005.
[9] C. Jennings. Nat classification test results. http://tools.ietf.org/html/draft-jennings-behave-test-results-04.
[10] J. Liang, R. Kumar, and K. W. Ross. The kazaa overlay: A measurement study. *Computer Networks Journal (Elsevier)*, 2005.
[11] Net-SNMP. Snmp protocol. http://www.net-snmp.org/.
[12] J. Rosenberg, R. Mahy, and C. Huitema. Traversal using relay nat (turn). http://www.jdrosen.net/midcom_turn.html, 2005.
[13] SBBI. Upnplib. http://www.sbbi.net/site/index.html.
[14] B. Sklar. Reed-solomon codes. http://ptgmedia.pearsoncmg.com/images/art_sklar7_reed-solomon/elementLinks/art_sklar7_reed-solomon.pdf.
[15] Skype. P2P telephony explained for geeks only. http://www.skype.com/help/guides/p2pexplained.
[16] WIREFLY. Cell phone facts. http://www.wirefly.org/news/cell-phone-facts.php.