

# Simple Music Streaming with MediaPipes

## *Zero-Configuration NAT Traversal*

Noah Richards  
Rochester Institute of Technology  
Rochester, NY  
nbr5959@rit.edu

Matthew Campbell  
Rochester Institute of Technology  
Rochester, NY  
mjc4648@rit.edu

## I. Introduction

One of the popular trends in software and general computer services is allowing the user to access his data and resources from anywhere. This primarily comes in two ways: portable devices that can physically follow you around, and programs that are designed to be accessed from anywhere in the world. The first category has brought us the boom of portable music/media players, and the latter brings us web sites and services that allow us to access e-mail, transfer files and data, and gain access to our computers remotely.

As the innovations in communication and the Internet push further towards the edges of available networks, software engineers of networked systems strive to find ways to balance large amounts of content over the relatively limited bandwidth of the Internet. This is the somewhat stalled trend towards Peer-2-Peer (P2P) applications, which design communication to go directly between nodes of the network. However, the gain in reduction of overhead is offset by the difficulty of negotiating connections through disparate network topologies.

With MediaPipes, we intend to continue the vein of “data anywhere” on the software/services side by allowing users to access their music from anywhere on the Internet. In addition, we will utilize a matching server and a P2P network infrastructure to both make the music easily locatable and keep the actual bandwidth considerations limited to the actual client and server machines.

## II. MediaPipes Architecture

The MediaPipes architecture can be understood in three portions: the home computer, the remote client, and the MediaPipes webhost/relay.

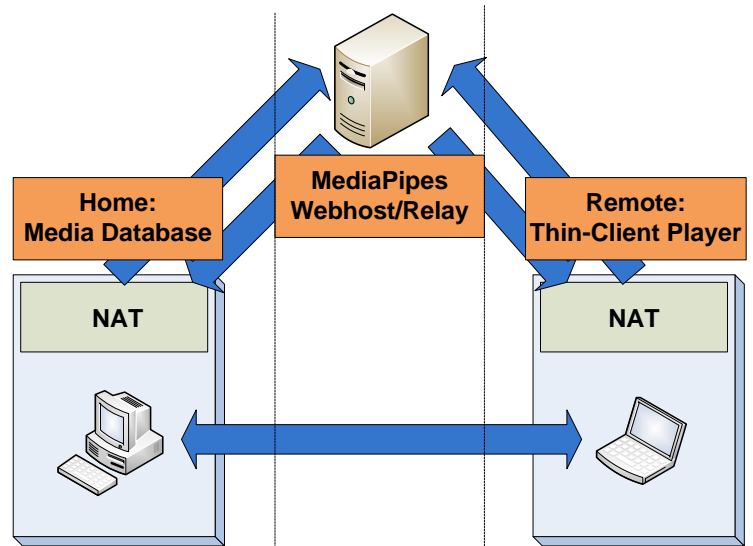


Figure 1 - MediaPipes Three-Tiered Architecture.

The “Home” tier is the client you would install on your home machine. The visible portion to the user allows him to configure what music to share (*e.g.* what directories to watch or file-types to share) and a few options for the MediaPipes Webhost/Relay, if the user decides to use this feature.

The “Remote” tier is the remote computer you are working on. In order to have as little footprint and requirements as possible, the media player will be some type of web control, downloaded from the webhost running on the MediaPipes server. To actually stream music, the thin-client player will directly connect to the Home machine (in most cases; in some, this is not possible, and the relay must be used).

The center tier’s primary duty is to act as a service for negotiating connections for the actual streaming of media. In a worst case scenario, the central tier can be a relay for the connection. Being able to traverse various NAT setups is the most difficult of the networking considerations, and is where we spent the majority of our time this quarter. The MediaPipes

server also hosts a website for Remote clients to download the thin-client player and find music.

In order to be effective, the MediaPipes Home and Remote machines need to be able to connect in the vast majority of cases directly, as the relay connection is ineffective and places resource requirements on the owner of the MediaPipes server. As with technologies such as BitTorrent, by spreading the cost out to the nodes of the network, an inexpensive solution can be deployed for the central server to do only the most necessary of tasks, such as handshaking and discovery. Our research for this quarter was spent almost entirely on what types of situations we need to deal with, and how to effectively overcome the barriers to direct connections.

### III. NAT Types

Because of the complexities inherent to the give-and-take between security and restricting the needs of well-meaning programs, Network Address Translation (NAT) devices and software/hardware firewalls place varying levels and layers of constraints which tend to hinder P2P protocols. There are primarily two approaches to dynamically allocating publicly accessible ports to machines inside the network: cone and symmetric NATs.

Cone NATs can be visualized as the name suggests: when a computer attempts to connect to an outside node, the NAT creates a “cone” with the point on the NAT and the open end towards the destination. These NATs are further classified by the size of the open end of the cone. The external port of the cone created by the NAT device is based solely upon the source (internal computer’s) address and port.

“Full cone” is the least restrictive type, which allows any external to communicate back through the cone, regardless of whether or not it was the original intended destination.

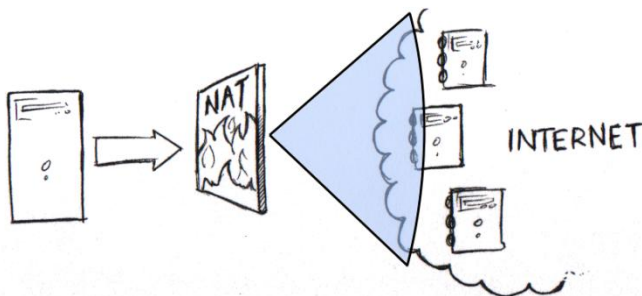


Figure 2 - Full Cone NAT

“Restricted cone” lets only the original destination connect back through the cone, although that computer may communicate back on any port (including the original destination port).

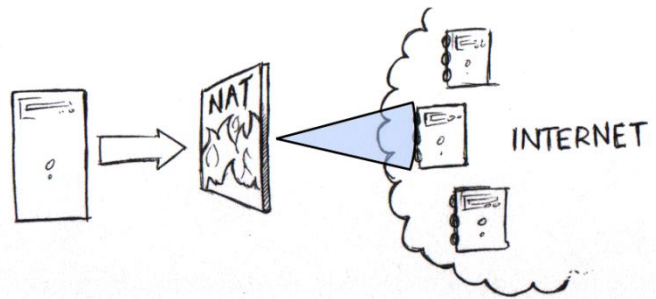


Figure 3 - Restricted Cone NAT

“Port-Restricted cone” is the most restricted type, which limits the communication the destination computer *and* port (*i.e.* more of a “pipe” than a “cone”). (1)

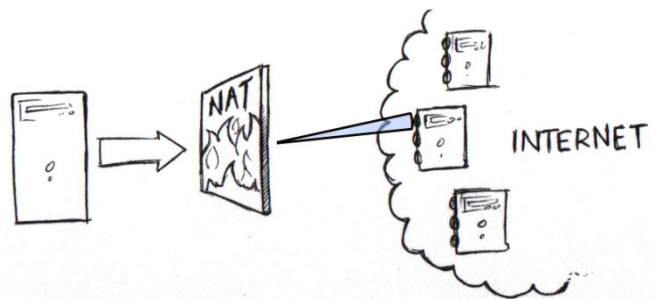


Figure 4 - Port-Restricted Cone NAT

Note that, in all of these cases, the cone is open regardless of the success or failure of the originally delivered packet. This is due in part to the “best effort” delivery mechanism of one of the primary types of packets being sent, UDP, and gives rise to many of the hole punching techniques used to circumvent NAT and firewall devices. Because the outgoing port has no dependency on the destination address, once the source machine has determined its external port (by various techniques, to be mentioned later), it can open a hole to any destination by sending a packet to that destination. This is a large part of the manner in which the Skype Voice-over-IP (VOIP) application conducts its network traversal. (2)

The other type of NAT is the only real problem area when it comes to traversing network topologies, and that is Symmetric NAT. With this type, the router opens up an outgoing port based both the source and destination address. Even though both cone and

symmetric NATs have generally unpredictable port-mapping rules, the cone NATs are deterministic from the point of view of the destination address. With symmetric NATs, the only determinism is that most routers have some consistent rules for how they map, but there is no definite way of knowing, guessing, or transferring this information in real-life circumstances.

There are a few important notes to make about these NAT types. First, these are all dynamic NATs, and there are NATs (sometimes called Demilitarized Zones (DMZs) on common NAT devices) that act as a simple passthrough to the internal device, but these can be considered a simplified case of Full Cone NATs. Secondly, Cone NATs are not entirely deterministic due mostly to collisions. If two machines are vying for the same port, the NAT device may assign special outgoing ports based upon the collision, and that mapping may change later if one of the machines stops sending over that port. However, in general, the window of stability is large enough to consider the mappings “deterministic.”

#### IV. NAT TRAVERSAL

There are many common ways of creating direct and indirect connections regardless of NAT and firewall setup.

**Universal Plug and Play (UPnP)** is a general solution by which NAT devices broadcast availability of service that allows internal devices to request certain port mapping rules. UPnP allows for the most direct communication, as it requires no go-between to either setup or handle communication, and allows for the greatest freedom and flexibility for applications that wish to control their outside signature on the NAT device. Unfortunately, NAT has not yet gained wide-enough implementation to be universal enough to be depended upon. In general, it is a good mechanism to try, but applications should not rely upon the existence of UPnP.

An **Application Layer Gateway (ALG)** refers to a portion of a NAT device or firewall which has some awareness of the application layer of the network protocol. This layer can inspect packets for certain information; for example, an ALG could sniff VOIP packets for clues on how to map the private outgoing port to the expected port on the received side, or even modify the packet itself to reflect the actual public port

mapping. Either way, the communication is direct and requires no mediation service, but ALGs have many drawbacks. By requiring the NAT device to inspect application layers of packets, it adds theoretically large latencies. Also, by placing the logic in the NAT instead of in the application itself (as with UPnP), the ALG logic is more difficult to extend or update. Finally, ALGs are uncommon in all but specialized NAT devices designed specifically for applications like VOIP. (3)

**Simple Traversal of UDP (User Datagram Protocol) through NATs (or STUN)** is a two-fold service. First, a STUN server will allow a machine behind a NAT to discover what type of NAT/firewall it is behind and its public IP address and port. Secondly, a STUN server can mediate a connection between two machines, by passing off one machine’s public information to another, since the STUN server is openly accessible (*i.e.* not itself behind a NAT). STUN is the next best thing to directly negotiating with the NAT device for open ports, although STUN begins the class of solutions that require an external server to play some role in the communication. With STUN, once the initial connection has been setup, the STUN server is no longer involved in the communication process. There are a few publicly usable STUN servers, but they only implement the first portion of the STUN protocol (determining NAT type and external IP/port) and not the information handoff.

**TURN, Traversal Using Relay NAT**, is a protocol where a server acts as a communication pass-through, where machines send all of their information to be indirectly delivered to the destination machine. TURN effectively eliminates any P2P aspect to the application, as all communication must pass through the server, but it has the benefit of working in any situation. TURN is a generalized solution to the common client/server pass-through paradigm, which means that it could theoretically replace that portion of applications that communicate in this way. However, since most applications only need a specific portion of TURN and supplying a public TURN server begs for security problems, most applications implement something akin to TURN but not TURN itself. Of all the NAT traversal methods, the only guaranteed success is to use TURN, which will only fail in situations where communication would be impossible in any

circumstance, such as a NAT or router that blocks *all* UDP traffic.

Best Effort Hole Punching is a non-deterministic method by which two machines behind Symmetric NATs attempt a brute-force “attack” on each other, hoping that by attempting to guess many common port-mappings of popular NATs, they can eventually find a point of overlap to communicate. The obvious drawback to this is the unpredictability, which means that you cannot guarantee both the allocated port and the connection itself.

## V. Traversal Strategy

For an application, the best possible traversal strategy will attempt a number of the different traversal methods, settling on the method with the least latency and bandwidth overhead. Within MediaPipes, we have adopted the following traversal strategy:

### TRAVERSAL STRATEGY

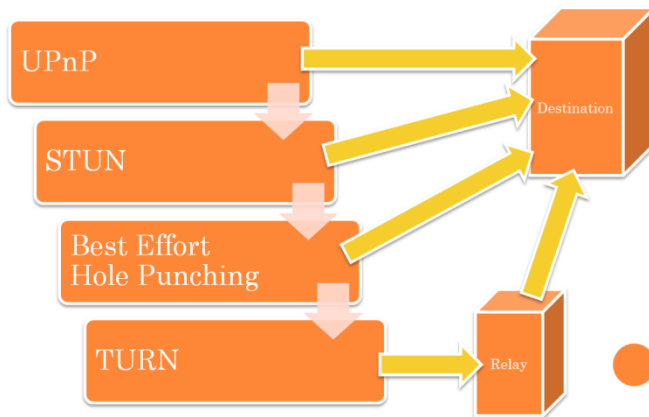


Figure 5 - MediaPipes Traversal Strategy

We begin with UPnP, which will allow us to attempt to map either a pre-defined public port or just any available port (which falls back to a sort of STUN situation, where a mediator hands off the information to the other machine). Trying to open a pre-defined public port gives the advantage that if the user wants to open a port on his NAT/firewall, he could open the well-defined port and thus skip all of the overhead of negotiating NAT traversal. This is not a requirement, but doing so could possibly remove the necessity for the middle-man relay at all, and would allow a remote client to connect to the home server much in the same way that Windows Remote Desktop does now. Either

way, if we have an already open port or UPnP succeeds, the two machines are connected directly.

If this fails, we try to use the relay as a STUN server. For a bit of speedup, we determine and cache the results of NAT type and public IP/port, so that each attempt at a connection does not require the entire STUN process. If at least one of the parties is behind anything but a Symmetric NAT, then a connection can be made. If both machines are behind Symmetric NATs, we fail and move on to the next step. Fortunately, according to a relatively popular study, somewhere around 82% of common NAT devices are non-Symmetric, and will never need to continue past this step. (4)

In the Best Effort Hole Punching phase, MediaPipes attempts to send out from a range of ports to a range of ports that are commonly mapped by popular NATs/firewalls. This goes on for a period of time in the hope that there will eventually be an overlap with both sides attempting to connect to each other. Since we are still working out testing across disparate NAT manufacturers and setups, we do not have any meaningful statistics for the success rate of this portion.

Finally, we drop back to a TURN-like protocol, which uses the central server as a relay. We did not actually implement TURN, as we do not need to be a general relay of UDP data (just relaying our own packets).

## VI. Current Deliverables and Future Work

Our implementation of this traversal strategy is a framework in Java, using a few existing projects released under the GPL. These are UPNPLib<sup>1</sup> for UPnP and JSTUN<sup>2</sup> for the discovery portion of the STUN specification. The Best Effort Hole Punching is a work in progress, and the TURN-like relay is in place.

In the upcoming quarter, we will be implementing the MediaPipes architecture. Our completed work acts as a framework for creating applications of this sort, and so MediaPipes will be specific implementation of this more general application type.

<sup>1</sup> <http://www.sbbi.net/site/upnp/>

<sup>2</sup> <http://jstun.javawi.de/>

## References

- [1] *NUTSS: a SIP-based approach to UDP and TCP network connectivity*. **Guha, S., Takeda, Y., and Francis, P.** FDNA'04, Portland, Oregon : ACM Press, August 30, 2004, Proceedings of the ACM SIGCOMM Workshop on Future Directions in Network Architecture, pp. 43-48. Available: <http://www.cs.cornell.edu/People/francis/nutss-fdna.pdf>.
- [2] *An Experimental Study of the Skype Peer-to-Peer VoIP System*. **S Guha, N Daswani, R Jain.** 2006, Proceedings of IPTPS. Available: <http://iptps06.cs.ucsb.edu/papers/Guha-skype06.pdf>.
- [3] **Newport Networks, Ltd.** Solving the Firewall and NAT Issues for Multimedia over IP. *Newport Networks*. [Online] 2006. [Cited: February 27, 2007.] <http://www.newport-networks.com/cust-docs/33-NAT-Traversal.pdf>.
- [4] *Peer-to-Peer Communication Across Network Address Translators*. **B Ford, P Srisuresh, D Kegel.** Anaheim : USENIX, 2005. Available: [http://www.usenix.org/events/usenix05/tech/general/full\\_papers/ford/ford.pdf](http://www.usenix.org/events/usenix05/tech/general/full_papers/ford/ford.pdf).
- [5] *NAT Traversal in SIP*. **B Sterman, D Schwartz.** IEC Annual Review of Communications. Available: <http://ag-projects.com/docs/PressArticles/NATtraversal-BestPractices.pdf>.
- [6] **Hu, Z.** *NAT Traversal Techniques and Peer-to-Peer Applications*. Helsinki University of Technology. Helsinki : Publications in Telecommunications Software and Multimedia, 2005. ISBN: 951-22-7738-7.