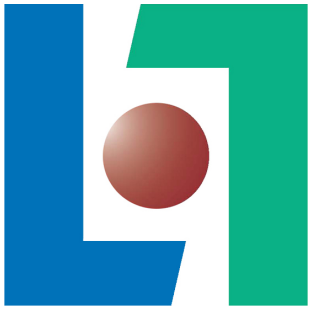
LaBRI
Laboratoire Bordelais de
Recherche en Informatique



University Bordeaux 1
Sciences and Technology

NAT Traversal and DHT for transmission of H264 stream on P2P networks

Master thesis

Tutor
Toufik AHMED

Student
Pierre OBAME MEYE

Academic year 2010-2011

Contents

List of Figures

Introduction

Nowadays, the widespread of P2P technologies is more and more important in the Internet and become an important part of the Internet traffic but Peer to Peer systems have to face some difficulties to work well across networks with the presence of routers performing a Network Address Translation (NAT). This is the well known problem of NAT traversal. Several solutions to this problem exist. In this report, we make an overview of the existing solutions and we proposed and implemented a specific solution considering the existing platform and the requirements of the project in which our work is a part of. It a European Research and Development Project named Envision that has for purpose the transmission of H264 SVC stream on P2P[1] systems. Our internship has been done in the research group COMET of LaBRI (Laboratoire Bordelais de la Recherche en Informatique).

We worked on an existing platform organised as a centralized P2P network. This platform provides services to stream H264 SVC resources between peers but these peers must have public IP addresses. In fact, the platform didn't take into account the NAT traversal issue and it was difficult or impossible to peers located behind some NAT[2] to participate to the activity of the network. Our first purpose was to take this issue into account and enhance the platform so that it can work across NAT in an automatic, simple and efficient manner and be transparent to the user. The second purpose of our internship was to study the Distributed Hash Table (DHT) based system in P2P systems for their benefits in P2P networks and make some proposals to use it improve the scalability of the platform considering the fact that the purpose is to transmit H264 SVC stream.

In the first part our this document, we make an overview of P2P systems, then we will talk about the NAT traversal issue and the work we done to design and implement to achieve the first purpose and in the last part we make a study and proposal about DHT-based P2P systems.

[1] P2P, in this document refers to Peer to Peer

[2] NAT, in this document refers to routers that perform Network Address Translation

1 Overview of P2P systems

The origins of P2P networks come from the limitations of the client/server systems. In fact in those systems, all the peers are connected to server and communicate only with the server. So, the server become a single point of failure. Furthermore, the scalability of this system is not very important and server can be bottleneck with a growing number of clients. We can take as example a FTP or HTTP server on Internet. So in the purpose of providing systems which work in large scale that researchers focus on distributing the workload among all the nodes in the distributed system [58]. Thus, P2P systems are more scalable and more powerful as every node brings its processing power in the system.

A P2P system refers to a distributed system where a node or peer can play the role of a server and the role of a client. P2P systems improve scalability by avoiding dependency on centralized points [49]. P2P become popular since the late of 1990s with many P2P networks applications. Most of them are file-sharing applications, Napster, Freenet, Gnutella, eDonkey, BitTorrent... P2P is often assimilate to the word *illegal* because file-sharing is the most popular use of P2P and the use of P2P file-sharing applications became the most used method for illegally share movies, softwares on Internet(Gnutella, Napster, Freenet, Kazaa...). But the application field of P2P is larger than that, it can be applied in many applications field:

- Distributed computing : This field can benefit a lot of the advantages of P2P system. In fact, in a P2P, we dispose of the capacities (processing power, disk storage...) of all the peer which can be very important. Seti@home is an example of P2P distributed computing
- Communication : Instant messaging (MSN, ICQ, Yahoo Messenger...), VoIP (Skype, Webphone...), video conference (CUseeME...)
- Media transport : streaming, P2P live TV, Video-On-Demand (VOD)

1.1 P2P network structures

There is many ways to design a P2P network architecture depending on how to manage the resources index, how to establish connection to others peers, how to discover resources or peers. So, there is several P2P architectures [39, 49, 58]. In the following points we present the different P2P architectures, centralized, decentralized or unstructured, hybrid and structured.

1.1.1 Centralized P2P networks

This is the most basic approach to design a P2P network. In contrast with a client-server model where the index of the resource and the resources are locate on the

server, in a P2P centralized network, the index is centralized on the server but the resources are decentralized among all the peers of the network. When joining the network, each peer sends to the server the list of resources it want to share and keeps a connection with the server, so the server knows all the resources that can be shared in the network and the resources possessed by each peers. When a peer searches a resource, it sends a query to the server, if the resource is in the network, the server sends to the client the addresses of peers that own the resource and the client will contact these node directly for the file transfer process. The figure 1 illustrates a centralized P2P system.

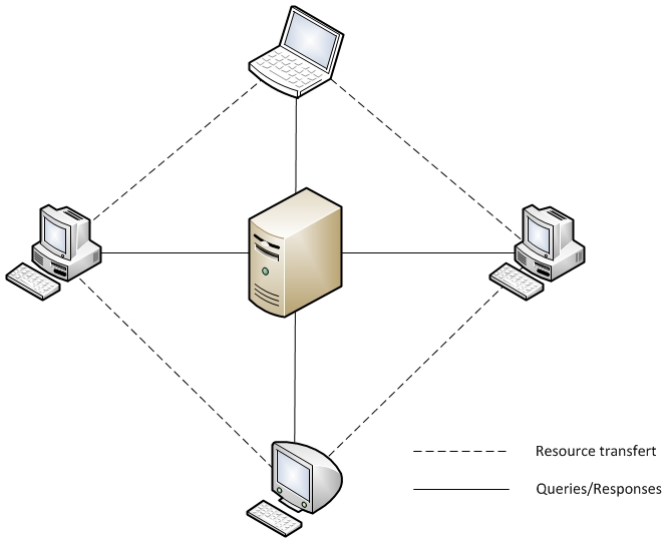


Figure 1: Centralized P2P system

1.1.2 Unstructured P2P networks

In an Unstructured or decentralized P2P network, there is not a central node (server for centralized P2P network). All the nodes are equal and have the same role, each node is a server and in the same time a client. There is no index. Each peer is responsible of the resources it possessed. There is no essential node so it makes this kind of networks more robust and scalable than centralized P2P networks. *Gnutella v0.4* is an example of unstructured network.

To join the network, the node has to find a node already in the network. It uses some bootstrap techniques to achieve it. There is several approaches to achieve it like the use of static overlay nodes (bootstrapping servers), web caches, random address probing, employing network layer mechanism [25, 71]. After finding a bootstrap server, a list of several nodes already in the network is given by the server to the node so it will establish connections to them in order to become a node participating in the overlay network. The basic approaches of the lookup mechanism in unstructured

P2P networks are flooding and random walk. In a flooding technique, the node send query to all of its neighbour, in a random walk technique, it sends query to a randomly number of peers. When a node receives a query, if it has the response, it sends it back, else it forwards the query with the same techniques. In both techniques, it uses a Time-To-Live (TTL) parameter decremented at each hop to limit the scope of queries [58]. The drawbacks of the TTL-based methods is that flooding and random walk are the basic methods but many works has been done to design other techniques and compare them [68, 62, 48]. the figure 2 illustrates an unstructured network. An example of unstructured P2P network is Gnutella v0.4.

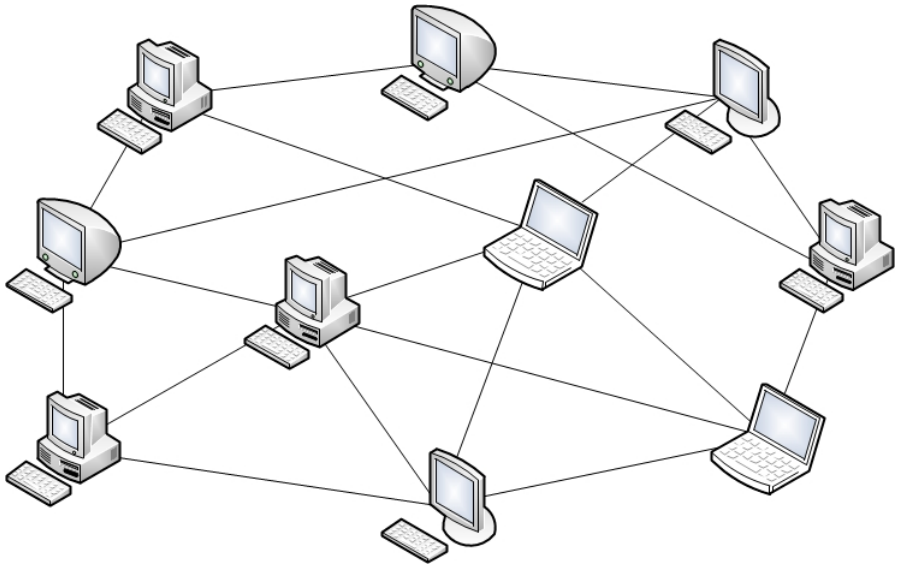


Figure 2: Unstructured P2P system

1.1.3 Hybrid (centralized + unstructured) P2P networks

This kind of architecture is a mix of the centralized and the unstructured P2P network. In this kind of structure, some nodes have more responsibilities. These nodes are called *super-node* or *super-peers*. A super-peer is a peer with higher capacities than others peers (bandwidth, memory, reliability, processing power) which acts as a centralized server to other *normal* peers and equal *server/client* to other super-peers. The overlay network of super-peers is organised in an unstructured P2P network and a super-peer and other peers connected to it act as in a centralized P2P network.

When a peer joins the network, a connection is established with a super-peer and exchanges informations about what the *normal* peer want to share. A super-peer knows all the resources owned by peers connected to him. If a peer searches a resource, it sends query to the super-peer to which it's connected to. If a super-peer receives a query and doesn't know where is the resource, it retransmits the query

to others super-peers it's connected to. When a super-peer leaves the network, an other super-peer is chosen among the normal peers and if no peer has the capacity to be a super-peer, they will be connected to an other super-peer.

This kind of architecture is less fault-tolerant than an unstructured system because of the inclusion of super-peers which are some single points of failure And hybrid avoids bottlenecks on peers with low capacity, and it is a fast system because most of all the communications is made by peers with high capacities. In example of network using this architecture we can notice : Gnutella v0.6, FasTrack (Kazaa, IMesh, Grokster, Morpheus...). The figure 3 illustrates an hybrid P2P network.

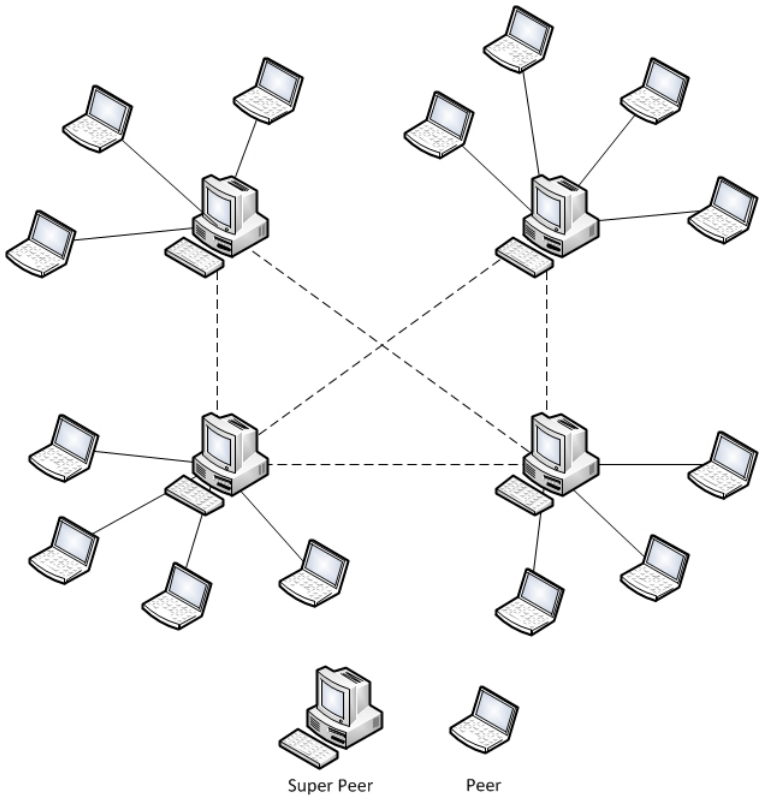


Figure 3: Hybrid (centralized + unstructured) P2P system

1.1.4 Structured P2P networks

In a structured P2P network, the index of resources is placed at specified locations which makes the lookup process efficient. Usually, the index is shared by all the peers. Each peer is responsible of a part the index. Structured P2P networks commonly use Distributed Hash Table (DHT) to organise their entities and to set up an efficient resources search mechanism. In a DHT, we assign an unique identifier to nodes and resources. Those identifiers are called *keys* and are chosen from the same identifier space. Each peer is responsible for a certain key range and keeps a routing

table (typically a hash table) containing the keys and the addresses of a certain number of peers. These peers are its *neighbours*. Structured systems are very scalable and their search mechanism are efficient. In contrast with unstructured systems, we sure to find a resource if it exists in the network but where structured system are efficient for rare resource, they have a higher overhead for higher replicated item than in a unstructured P2P networks. To join the structured network, a node has to find a node already in the network. In this bootstrapping step, we assign to this node, a list of neighbours. This new node will be connected to them and they will share with the new node a part of their index so that it can't participate the network activities. When a node leaves the network, its routing table is shared among its neighbours.

2 The NAT traversal issue

Since the emergence of the Internet, we have to face a problem, we don't have enough IP addresses to support this emergence. So, this is where the Network Address Translation (NAT) comes.

The NAT is one of the solution of the depletion of the IPv4 address space among the *classful addressing*, *subnetting*, *Classless Inter Domain Routing* (CIDR). The economy of IPv4 addresses is not the only advantage of the NAT, it is also a security factor. In fact, it helps also to limit the access to a private network from the public network (Internet or other private networks). Thus, there is only one access point to the private network which is less difficult to control than all the hosts of the private network. NAT doesn't only have advantages, NAT is a very big problem for P2P connections establishment. In fact, in P2P, a peer can be a client or server depending of the purpose of the communication but peers[3] located behind a NAT router can be unreachable by those not located in the same network. Several NAT traversal techniques exist but sometimes implementations are rare or these solutions are commercial and we don't have access of the techniques used to perform NAT traversal.

In this part, we explain how the NAT works, we also present some of the NAT traversal solutions. In the second part we present the solution that we designed and implemented.

2.1 NAT principes

The NAT is performed by a router NAT which separates a private network from an other network, private or not, the figure 4 illustrates it.

[3]Peer, in this document refer to node or client

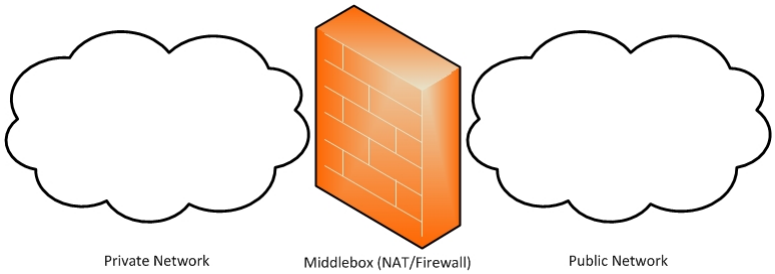


Figure 4: Router NAT

The NAT consists of changing the IP address in the header of inbound and/or outbound data passing through the router. If it changes the IP source, it's called a *SNAT*, if it's the IP destination which is changed, it's a *DNAT*.

When a peer A behind the NAT sends data to a peer B outside the network, the NAT performs the translation and saves the mapping created. When B replies, the NAT compares the inbound packet with mappings created earlier so it can forward the packet to the right peer, A.

2.2 Types of NAT

We commonly name two types of NAT, the *basic NAT* and the *Network Address Translation-Port Translator (NAPT)*.

2.2.1 Basic NAT

In the basic NAT, the router has a set of public IP addresses and assigns them to peers in the private network that communicate with peers outside the network. If the number of local hosts is less than or equal to the number of public IP addresses, to each private address is guaranteed a public address to map to. Otherwise, peers allowed to have simultaneous access to external network are limited by the number of addresses in the set of public IP addresses. Individual private addresses may be statically mapped to specific public addresses to ensure guaranteed access to the outside or to allow access to the local host from external hosts via a fixed public address[2].

The figure 5 taken from [2] illustrates a basic NAT. When stub A host 10.33.96.5 wishes to send a packet to stub B host 10.81.13.22, it uses the globally unique address 198.76.28.4 as destination, and sends the packet to its primary router. The stub router has a static route for net 198.76.0.0 so the packet is forwarded to the WAN-link. However, NAT translates the source address 10.33.96.5 of the IP header to the globally unique 198.76.29.7 before the packet is forwarded. Likewise, IP packets on the return path go through similar address translations.

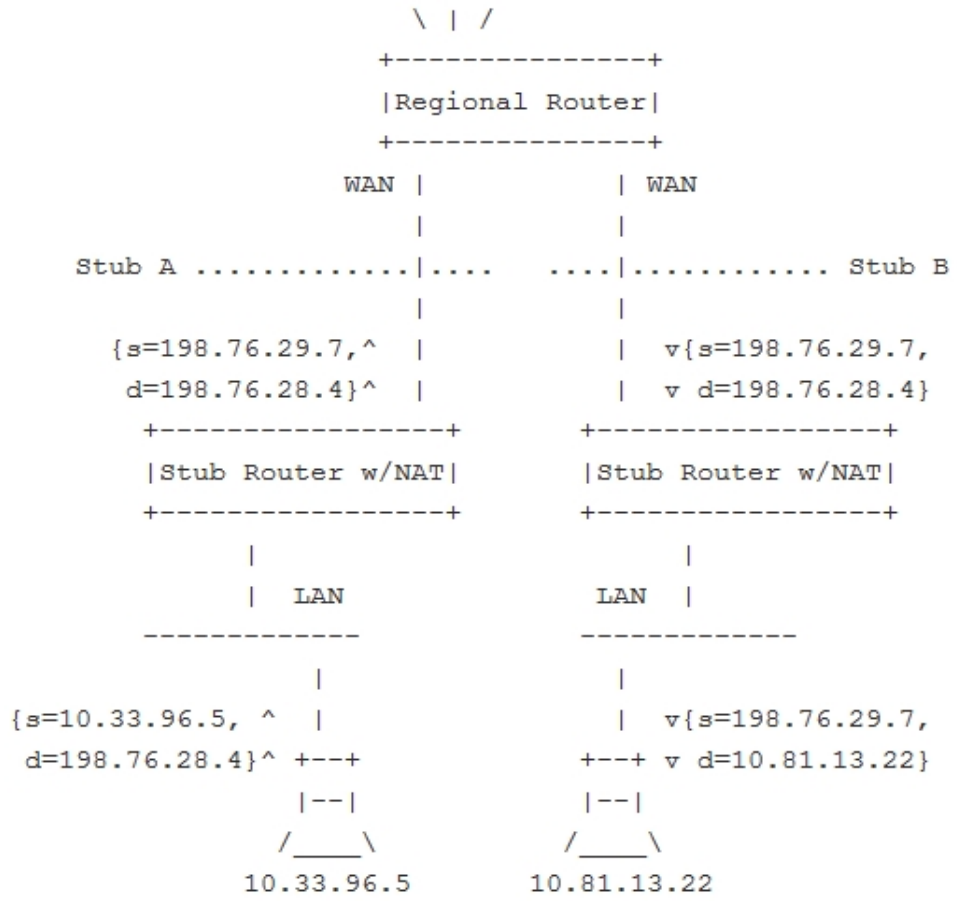


Figure 5: Basic NAT

2.2.2 Network Address Translation-Port Translator (NAPT)

Network Address Translation-Port Translator (NAPT) is the most deployed NAT type [7]. It allows a number M of internal host to share a single public address or a number N of public addresses with $M > N$. To do so, the NAPT doesn't only change dynamically its IP address, but also the port. The translation of a NAPT consist of changing the tuple (private IP address, private port number) to (public IP address, public port number) and vice versa.

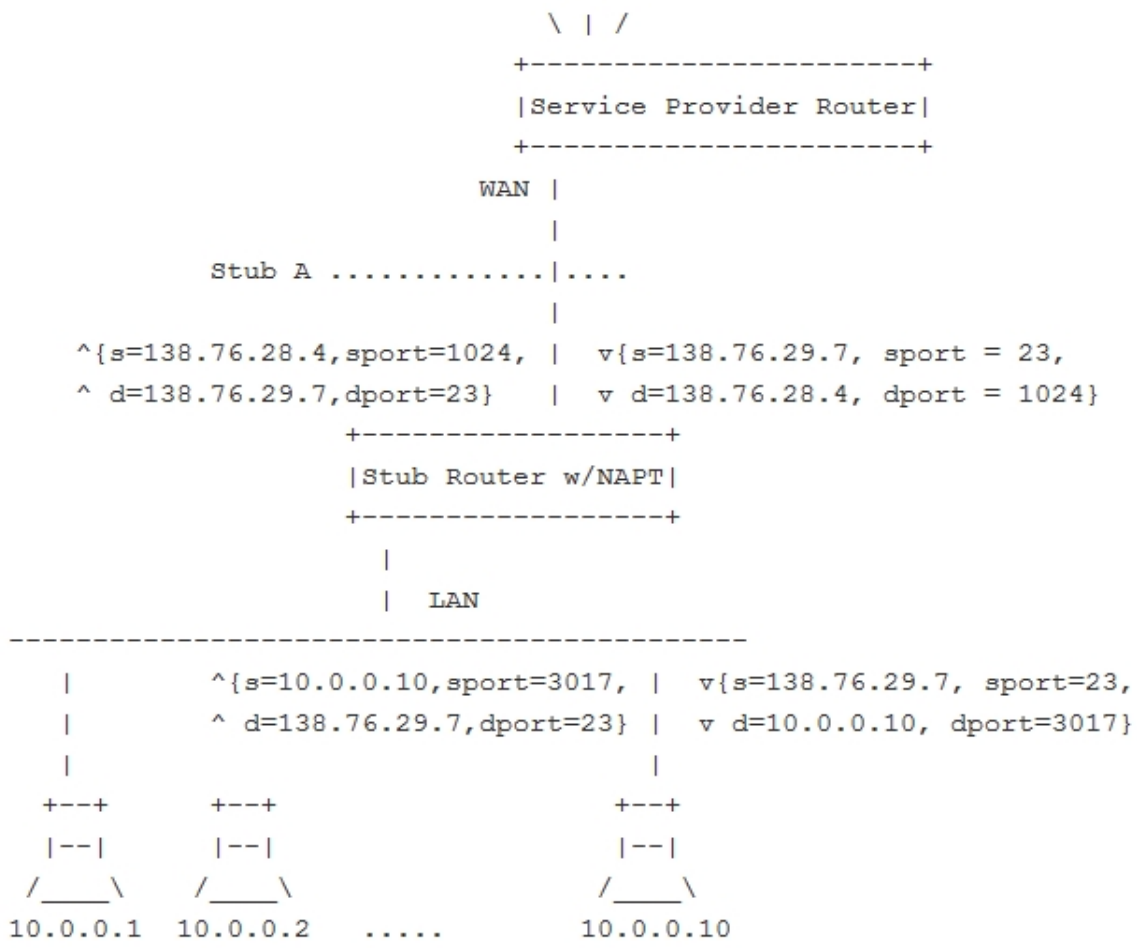


Figure 6: Network Address Translation-Port Translator (NAPT

2.3 Different behaviours of NAT

Commonly, we classify the different types of NAT in four categories [11], this is the classification made by a working group of the Internet Engineering Task Force (IETF). Some researchers go further [35] in classification and add three other categories *asymmetric full cone*, *asymmetric restricted cone* and the *asymmetric port restricted cone*. In the next points we consider the classification made by the IETF as the three others can be include in them.

2.3.1 Full Cone NAT

A full cone NAT is one where all requests from the same internal IP address and port are mapped to the same external IP address and port. Furthermore, any external host can send a packet to the internal host, by sending a packet to the mapped external address.

2.3.2 Restricted Cone NAT

A restricted cone NAT is one where all requests from the same internal IP address and port are mapped to the same external IP address and port. Unlike a full cone NAT, an external host (with IP address X) can send a packet to the internal host only if the internal host had previously sent a packet to IP address X.

2.3.3 Port Restricted Cone NAT

A port restricted cone NAT is like a restricted cone NAT, but the restriction includes port numbers. Specifically, an external host can send a packet, with source IP address X and source port P, to the internal host only if the internal host had previously sent a packet to IP address X and port P.

2.3.4 Symmetric NAT

A symmetric NAT is one where all requests from the same internal IP address and port, to a specific destination IP address and port, are mapped to the same external IP address and port. If the same host sends a packet with the same source address and port, but to a different destination, a different mapping is used. Furthermore, only the external host that receives a packet can send a packet back to the internal host.

3 Existing solutions for NAT Traversal

To solve the NAT traversal, several methods are known. In this section, we discuss about the main techniques used over the Internet. We can classify these techniques in two groups:

- **Techniques using relay :** these techniques focuses on the use of a third-party server or modifying the NAT device so that this entity can relaying communication. Even if, technique using relays can solve the NAT traversal for all the kind of NAT, in P2P network, those kind of techniques are used in the last case because of their high latency due to the relay. Example: Application Level Gateway (ALG), application server, application server with agent, TURN.
- **Direct techniques :** these techniques focuses on modifying communication protocols or NAT device so that we can establish direct communication through them. Example : tunnel, NAT-PMP, UPnP, MidCom, STUN hole punching, ICE, Teredo.

With theses techniques we resolve NAT traversal but only technique with relay are efficient for symmetric NAT. Some researcher focus their work on port prediction

to resolve the traversal of symmetric NAT but those techniques are not very efficient and reliable. In the following point we will present the main concept of these techniques.

3.1 Applications servers

3.1.1 Application Server

An Application Server for NAT traversal acts like a proxy. We set a server in the private network or in a DMZ and all the communications will go through its. It's a simple method, we don't have to modify network devices, applications, security policies of the network or protocols and its solves NAT traversal for all its kind of behaviour. The drawbacks of this method is the deployment of the server and its maintenance, the high latency because communications are not direct between clients and it's specific to a protocol.

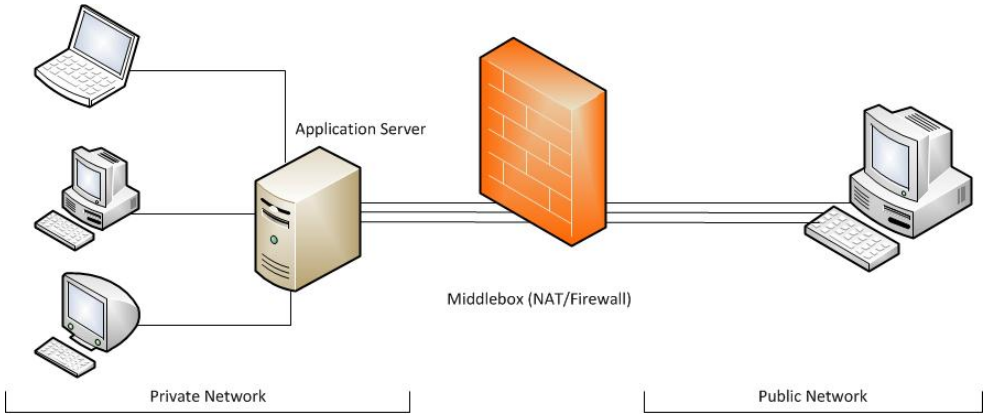


Figure 7: Application Server

3.1.2 Application Server with Agent

In this method, the application server can be placed in DMZ or public zone, in the private network we place the agent. The client communicates with the agent and the agent communicates with the application server. The agent creates a tunnel with the application and the communication is made on ports allowed through firewall and routers. This method solves the NAT traversal issue, it can be used by more users than in the previous method because the application server is in a public zone. This method also shares the same drawbacks with the previous method, maintenance of the application server and the agent, specific to a protocol and high latency.

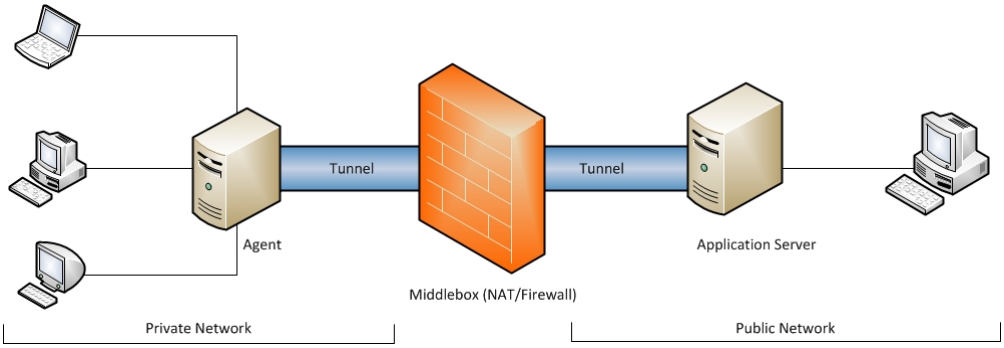


Figure 8: Application Server with agent

3.2 Application Level Gateway (ALG)

Not all applications lend themselves easily to translation by NAT devices; especially those that include IP addresses and TCP/UDP ports in the payload. Application Level Gateways (ALGs) are application specific translation agents that allow an application on a host in one address realm to connect to its counterpart running on a host in different realm transparently. An ALG may interact with NAT to set up state, use NAT state information, modify application specific payload and perform whatever else is necessary to get the application running across disparate address realms. ALGs may not always utilize NAT state information. They may glean application payload and simply notify NAT to add additional state information in some cases. ALGs are similar to Proxies, in that, both ALGs and proxies facilitate Application specific communication between clients and servers. Proxies use a special protocol to communicate with proxy clients and relay client data to servers and vice versa. Unlike Proxies, ALGs do not use a special protocol to communicate with application clients and do not require changes to application clients.[1]

In contrast with the previous methods, we don't add an entity to help in the communication. The ALG solution focuses on modifying and upgrading the NAT device [57] so that it can also take into account the content carrying by the communication, interpret a specific protocol and it can modify data so that it can be correctly route in the both side of the network. In comparison with the previous methods, using ALG requires more advanced configuration and management skills.

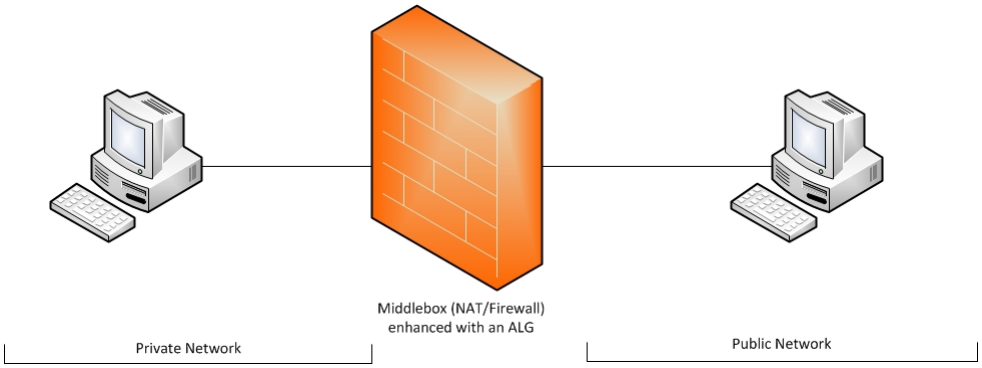


Figure 9: Application Level Gateway (ALG)

3.3 TURN (Traversal Using Relay NAT)

Traversal Using Relay NAT (TURN) is a solution proposed by the Internet Engineering Task Force (IETF) to solve the problem of symmetric NAT [13, 15, 17, 19, 38]. This technique uses a third-party server in public zone or DMZ that plays the role of relay. Each client keeps a connection open with the TURN server. So, when a client communicates with an other client, all the traffic goes through the TURN server. It solves the NAT traversal because for the NAT routers, it's always the TURN server which communicates with the clients behind the NAT. It also solves the symmetric NAT because no other connection is opened. This technique is a quite different technique from the previous we presented because it is not a specific to a protocol of communication. We don't have to modify our network or router but applications have to be compatible with the TURN protocol. Like all the relay solutions, the drawbacks are the high latency, the maintenance of the server.

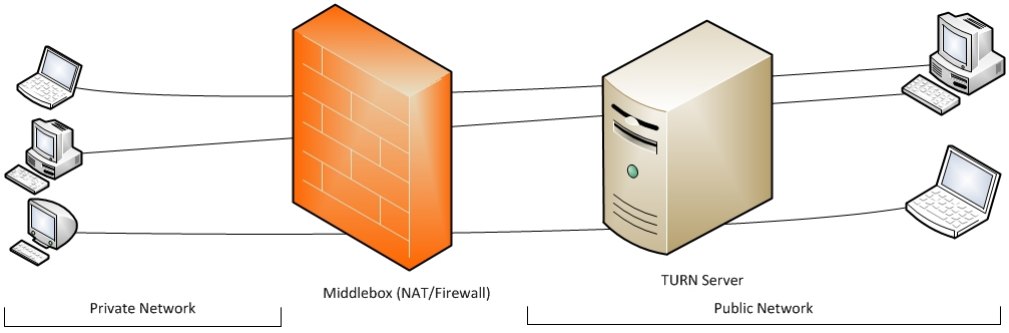


Figure 10: Traversal Using Relay NAT (TURN)

3.4 Tunneling

The tunneling method uses the existing standard protocols of communication (PPTP, L2TP, IPSec...) to create Virtual Private Network (VPN) between clients to solve

the NAT traversal. Some works has been done by the IETF to help VPN protocol to go through NAT [5].

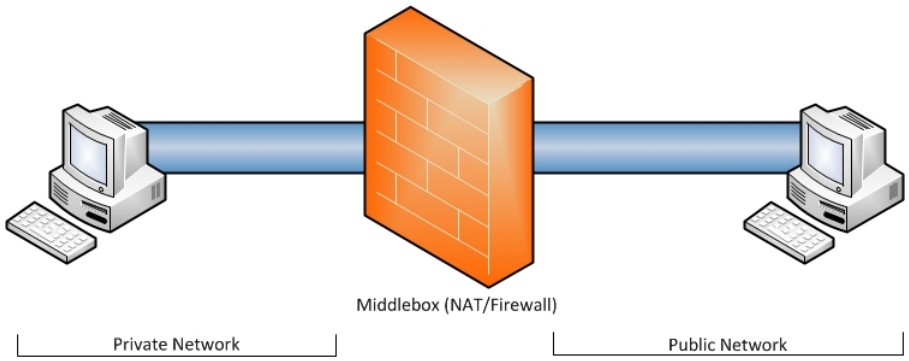


Figure 11: Tunneling

3.5 MidCom (Middlebox Communication)

MidCom is a similar solution to ALG in the sense that they focus on modifying and upgrading the middleboxes. It's a work in progress of the IETF, framework [3], requirements [4], and protocol semantics [8] for communication between applications and middleboxes acting as firewalls, NATs, or a combination of both. Instead of ALG, it doesn't examine each packet, it just open hole to communication and it's not specific to a protocol, it's protocol independent. MidCom also introduces an authentication mechanism.

3.6 Teredo : Tunneling IPv6 over UDP through NAT

Teredo is a protocol introduced by Microsoft that enables hosts located behind one or more IPv4 NATs to obtain IPv6 connectivity by tunneling packets over UDP [6]. IPv6 packets are sent as IPv4-based (UDP) messages to go through NATs. It uses the 6to4 protocol which is a transition mechanism IPv4 to IPv6 that encapsulates IPv6 packets over IPv4 packets. In Teredo we use two third-party servers, *Teredo servers* and *Teredo relays*.

- **Teredo server :** a node that has access to the IPv4 Internet through a public address, and is used to provide IPv6 connectivity to Teredo clients [4].
- **Teredo relay** : an IPv6 router that can receive traffic destined to Teredo clients and forward it using the Teredo protocol.

In the first specifications, Teredo was able to solve NAT traversal for all kind of NATs except symmetric NAT. To use Teredo, clients located behind a symmetric NAT had to open a Teredo service port for each client on the NAT. Some works have been done in Teredo protocol by adding extensions to solve symmetric NAT.[16, 18].

[4]We call Teredo client, a node that want to access to the IPv6 Internet

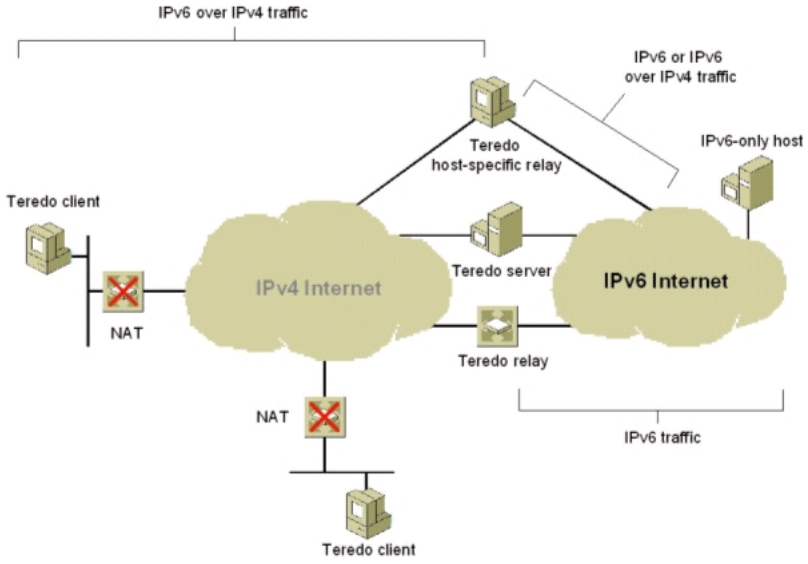


Figure 12: Teredo : Tunneling IPv6 over UDP through NAT

3.7 UPnP (Universal Plug and Play)

UPnP technology defines an architecture for pervasive peer-to-peer network connectivity of intelligent appliances, wireless devices, and PCs of all form factors. It is designed to bring easy-to-use, flexible, standards-based connectivity to ad-hoc or unmanaged networks whether in the home, in a small business, public spaces, or attached to the Internet. UPnP technology provides a distributed, open networking architecture that leverages TCP/IP and Web technologies to enable seamless proximity networking in addition to control and data transfer among networked devices. [33]

UPnP provides services to discover and control devices that support UPnP. In the case of the NAT routers, if the NAT router is UPnP compatible, a client behind this NAT can discover the router NAT, get its public address and create port mapping on the NAT so that it (the client) could be reachable by other peers out the network and all of these actions are made *automatically*. A way to use UPnP, is to get its external address, to open a port and share it so that clients that know this endpoint could contact it through NAT without problem.

UPnP allows to open hole in NAT without authentication so it often considered as a risk of security. Thus, by default it's disable in most of the NAT router but more and more router supports UPnP because is a easy way to solve the NAT traversal. A drawback in NAT traversal with UPnP is that it's doesn't work with cascaded NAT.

3.8 NAT-PMP (NAT Port Mapping Protocol)

NAT Port Mapping Protocol (NAT-PMP) is a work in progress to a standardization by IETF introduced by Apple [36]. NAT-PMP is an alternative to UPnP as they provide the same functionality for NAT traversal but it less used because of the less number of routers implementing the NAT-PMP.

3.9 Hole Punching and STUN

The hole punching is the most practical and robust method for the NAT traversal as it on the most efficient technique. The hole punching needs a third-party server or a rendez-vous server to work so it most of the time used with the help of a STUN server. Thus, STUN is often considered as a NAT traversal solution.

the **Session Traversal Utilities for NAT (STUN)** [11] is not a NAT traversal solution but a protocol used as a tool by other protocols to solve the NAT traversal. It can be used by a client to discover if they are behind a NAT, determine the behaviour of this NAT [14] and its external address (IP and port). STUN is a protocol based on UDP and provide a keep-alive mechanism to maintain NAT bindings but the mechanism can be extend to TCP.

The **Hole Punching** technique enables two clients to set up a direct peer-to-peer session with the help of a well-known rendezvous server, even if the clients are both behind NATs but it's doesn't work with the symmetric NAT [30]. The rendezvous server let the clients to know each other endpoints (address IP and port) so they can communicate directly.

Figure 13 is an example of hole punching process:

1. **Step 1–4 :** peer A and B open connection with the rendezvous server S
2. **Step 5 :** B ask to S informations to reach A.
3. **Step 6 :** S sends to B the informations about A.
4. **Step 7 :** B sends a message to A which will blocked by the NAT of A, the interest is to create a mapping in the mapping the B
5. **Step 8 :** B sends a message to S in which it ask to S to ask to A to establish a connection to to B.
6. **Step 9 :** S asks A to open a connection with B
7. **Step 10 :** A opens a connection to B, because of the step 7, this operation will traverse the NAT of B
8. **Step 11 :** B can communicate with A

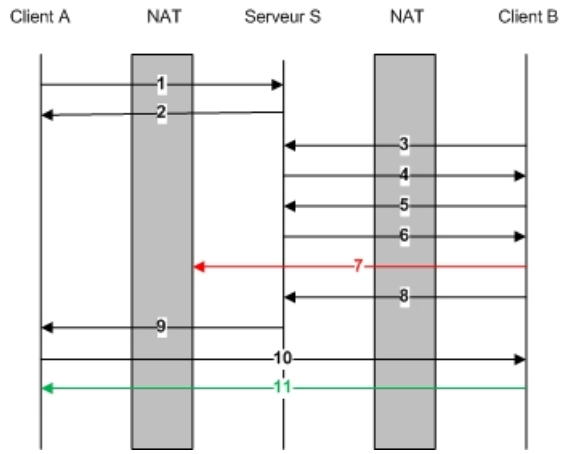


Figure 13: The hole punching technique

3.10 ICE (Interactive Connectivity Establishment)

ICE is a standard made by the IETF, it a mix of TURN and STUN [9]. The purpose of ICE is to know the behaviour of the NAT with STUN to determine the best method for NAT traversal, relay or direct connection via STUN hole punching. ICE works with UDP but an Internet Draft proposes to extends it to TCP [37]. It's still a work in progress.

3.11 Other solutions

To Solve the NAT traversal, many other works exists. Some are better in theory but we think that they are not recommended in practice like **pwnat** [54] that uses fake ICMP messages to open hole in NAT but ICMP is often blocked by firewall. An other technique is the **prediction port** [74] to solve the symmetric NAT traversal which is not recommended to use because the behaviour of the allocation of the port of the NAT can be different depending on the manufacturers of the NAT, some companies use a random allocation port mapping [42]. The **connection reversal** is used when a peer A that is not behind a NAT (direct connection can be established to it) want to set up a connection with a peer B located behind a NAT. A can't established the connection directly so it request B via a third-party server to initiate the connection. In the case of **hairpin connection**, the peers use their private address to established connection. The hairpin connection is when two peers behind the same NAT want to be connected, some NAT doesn't support that two peers of the private network established connection with their external addresses.

4 Design and Implementation of a NAT traversal solution

4.1 Proposal of a NAT traversal solution

The only solution which works with the all kind of NAT is the solution using a relay server, in our case we think about TURN. However, this method doesn't provide good performance because the all the traffic has to pass by a third-party node. So, this solution is used only when all the others solutions don't work. Usually, we use relay server in the case of a NAT symmetric.

So, considering all the issue of NAT traversal in P2P networks, what we propose to use is a mix of several solutions. In the solution we propose, we plan to use Universal Plug and Play (UPnP), a STUN-like solution, a hole punching technique and a TURN-like solution. Our motivations to suggest such an architecture is the benefit of efficiently that can result if we combine the advantages of these different techniques.

The first step of our NAT traversal process, is to find if we are behind a NAT, if not, we use direct connection, if we are behind a NAT, we find the kind of the NAT to choose the optimal technique for its traversal. First, we try to use UPnP, if we don't find a router supporting UPnP, we use the other techniques depending on which kind of NAT we are behind.

In our work, we study and develop prototype to test their feasibility in UDP and TCP before integrating it in the platform of the project Envision. In the next sections we describe the implementation of the solutions.

4.2 Implementation the NAT traversal solution

In the beginning of our internship, the platform was based on MonoTorrent [50] is a cross platform and open source implementation in C of the BitTorrent protocol [70]. During the internship, our team decide to change MonoTorrent by a platform on which we will have better possibilities of modifications in order to have a very well control on what we plan to develop in future. We choose to use Sharp P2P. Sharp P2P is a project fully coded in C on the .Net platform which will allow the user to download file through various protocols [66]. It provides to us the basic functionalities to include a P2P layer in a project. We added more functionalities in Sharp P2P like message framing, UDP and TCP NAT traversal support, more message types for peers communication (bootstrapping, peer discovery, peer informations exchange...). In the following points, we present the main points of the implementation.

4.2.1 Platform architecture

Our P2P overlay network is in a centralized structure and we have two kind of nodes : *tracker* and *peer*. A peer is simple client and a tracker is a central node. In the current architecture we use only one tracker but it could easily be extends. All the peer have to be connected to its for integrating the overlay network. When a node join the network, it opens two connections with the tracker, a connection for control and a connection for data. We implements a certain number of messages to make the nodes communicate.

1. **Connection :** When a peer want to join the overlay network, its opens two connections with the tracker, a control connection and a data connection. It's the method to get connected between any peer. A control connection is for signaling messages and data connection is for data transfer.

2. **Share a resource :** When a peer want to share a resource in the overlay network, it sends a message to the tracker, so it can be find by peers interested by the file. In the message it specifies the percent of the file it possesses. This percent is updated periodically, when change appears.

3. **Resource discovery :** When a peer is interested by a resource, it sends a message to the tracker. The tracker sends in response a list of peers which share or are downloading this resource. The peer can send this message with some parameters to filtrate the list the peers the tracker will send in response. After receiving this list, the peer contacts the peers of the list and they exchange informations about the part of the file they possess in order to build requests.

4. **Disconnection :** There is two ways a peer can leave the network, willingly or accidentally (an error appears). If it's willingly, the peer sends a message to tracker and the tracker sends back a message to notify and then the connection can be closed. If a peer leave the network without sending a message of disconnection, our keep-alive mechanism (4.2.2 p.23) will detect it and it will not be considered alive any longer.

5. **Peer informations storage :** The tracker keeps informations about peers like IP addresses, geographic coordinates, bandwidth (download or upload). Those informations can be use in peers selection.

6. **Communication in an asynchronous mode :** To manage connections and data transmission, we develop our communication module dealing with socket programming to have a full control about connection establishment. There is many ways to program socket in distributed applications, using *threads*, using the *Select function for Multiplexed I/O* or *Asynchronous I/O*. We choose the asynchronous mode that alleviates the need to create and manage threads. This, leads to much simpler code and also is a more efficient I/O model [44].

4.2.2 Keep-alive mechanism (TCP/UDP)

We implemented a keep-alive mechanism [20] to keep connection alive on every node of our overlay network. When there is no traffic (download or upload...) between 2 peers (peer and peer or peer and tracker), they send to each other keep-alive message to notify that their still alive. This mechanism is crucial in UDP because UDP is a connectionless protocol and a keep-alive mechanism assure that a node is still in activity even if the connection is idle and also keeps the NAT bindings open [12]. Although TCP is a connection-oriented protocol, we also implemented a keep-alive mechanism because in both cases, it is useful for the NAT traversal solution. In fact, a NAT keeps a port mapping only if there is traffic in both directions [56]. If a peer A behind a router NAT communicates with a peer B, and A only receives data and doesn't sends any data, the router NAT may delete the port mapping and then close the connection.

4.2.3 IP Address Geolocation

We develop an IP geolocation module in the project. Informations collected by this module will help in peer selection in resource discovery and will also help in the case of implementation of service like multicast to peers with the same Internet Service Provider (ISP). We use the non commercial service provided by the web site trace [69]. This service provides an XML interface for the integration of results in applications. the non-commercial service is limited to 100 free queries per day. There are many WebSites like *Utrace* which provide APIs to geo-locate IP addresses (MaxMind [46], Hostip.info [34]...) but for most of them the non commercial service is limited in term of query number. We didn't search further because the platform was not stable yet and for simple test. The module was quite satisfying.

4.2.4 TCP Message Framing

During the development of the TCP of the platform, we face the message framing problem. We notice that, sometimes we send message and the message is not received complete. The platform was built in the idea of sending a message in one time and receiving a message in one time but TCP works on streams and not on packets or messages. We found an article of Stephen Cleary which explains very well the concept.

One of the most common beginner mistakes for people designing protocols for TCP/IP is that they assume that message boundaries are preserved. For example, they assume a single "Send" will result in a single "Receive" [...] There is no way to send a packet of data over TCP; that function call does not exist. Rather, there are two streams in a TCP connection: an incoming stream and an outgoing stream. One may read from the incoming stream by calling a "receive" method, and one may write to the outgoing stream by calling a "send" method. If one side calls "send" to send 5 bytes, and then calls "send" to send 5 more bytes, then there are 10 bytes that are

placed in the outgoing stream. The receiving side may decide to read them one at a time from its receiving stream if it so wishes (calling "receive" 10 times), or it may wait for all 10 bytes to arrive and then read them all at once with a single call to "receive". [...] Since TCP operates on streams, one must design a "message framing" protocol that will wrap the messages sent back and forth.[59]

There is commonly two methods to solve the message framing :

- **Length prefixing :** In this method, if we want to send a message, we prepend it with a header containing the size of the message we want to send. In the reception side, we first have to receive the *size* of the data and then loop until *size* data is received. This is the method which is mostly suggested.
- **Delimiters :** In this method we use delimiters to differentiate the start and the end of a message. The reception is more complex than in the length prefixing method.

We use the length prefixing method to implement the message framing as it the most efficient and reliable method. The figure 14 illustrates the format of our messages and illustrate the way messages are written in a socket in TCP.

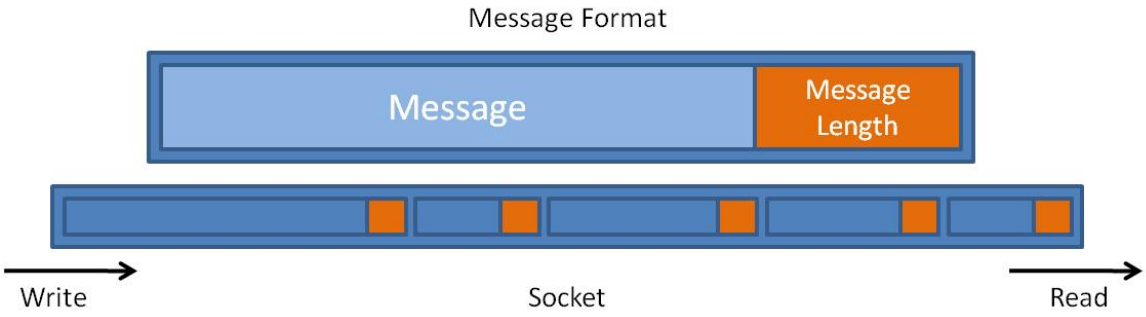


Figure 14: Message format : length prefixing

4.2.5 Universal Plug and Play (UPnP)

Since UPnP [32] is more and more supported by routers and it's an easy and simple technique for the NAT traversal, we introduce a UPnP module in the NAT traversal solution. In fact, a peer can be located behind any kind of NAT, if it uses UPnP to create a mapping on a router, the mapping will works as a mapping in a full cone NAT. So the peer will be reachable by all the peers in the P2P overlay network.

In our works, we implemented two libraries, one using the NAT Traversal API of Microsoft [51] and an other using the UPnP APIs of Microsoft [52].

- **Using the MSDN UPnP NAT API :** We implemented a library using this API. This API is quite easy and simple to use but we didn't introduce it in the platform because of a drawback which can be a source of bug. In fact, with this API, when you search UPnP routers, if there is many UPnP router (considering the case when our computer has many network interface card enabled), it's the first to response that will be considered but with this PAI we have no information about this entity. So, when we want to add a port mapping, it can result in some strange situations like mapping connections from a router to an network interface that is not in the same network.

- **Using the MSDN UPnP APIs :** There is actually two APIs in the UPnP APIs of MSDN. A *Device Host API* which consists of a set of COM interfaces used to implement devices that are hosted by a computer and a *Control Point API*, the one we use, which consists of a set of COM interfaces used to find and control devices. This API is not specific to an UPnP device so it's more complex to use than the NAT API but we have more control and possibilities, to control devices, we have to refer to the Device Control Protocol documents to know the actions supported by a device and how to invoke action. In our case, we referred to the document about the device *WANIPConnection* [31] which is responsible of the mapping port service.

We integrate the second library because it is more flexible and we have more control. The UPnP solution is the first method we try to use when a peer is locate behind a NAT and want to join the overlay network. So with or library, it discovers the NAT router, its public IP address, it creates a port mapping and send all this informations to the tracker in the connection process. The drawback of the use of UPnP in a NAT traversal solution it that it doesn't works if the peer is located behind a cascaded NAT the peer can only discover and controls the routers that are located in the same network. We implement also a mechanism to detect if a peer using UPnP is located behind a chain of NAT. In this case we cancel the use of the UPnP for our STUN-like solution.

4.2.6 TCP and UDP Hole punching

We implemented a TCP/UDP Hole punching for NAT traversal when direct connection (UPnP include) can't be used. We implements a certain number of messages to perform the UDP and TCP hole punching using the tracker. The TCP hole punching is more complex to implement, we use a TCP simultaneous-open to perform it [10]. We use the socket option *SO_REUSEADDR* to keep the same endpoint to be reachable while opening connection with tracker or other peers [30].

4.3 Overview and future works

In the previous points we present our NAT traversal solution, the development of the solution is not complete but for now, it can work through all kind of NAT except symmetric NAT although we didn't implement yet the module to discover the kind of NAT a peer is behind of. In fact, we perform a full hole punching if direct connection is not possible, this is not optimal even if it works. In some case, we send more data than we should and the overhead is higher than it should. With a *NAT type finder* our solution will be more efficient and we will be able to go through symmetric NAT. In the future we also need to work on the security, in fact sockets that allow other sockets to bind on a same port via the option *SO_REUSEADDR* can impact and affect network application security [53].

5 Distributed Hash Table in P2P systems

From this section, we will talk about the second part of our intern-ship. Our work was to make a study of the different Distributed Hash Table (DHT) systems and propose a P2P infrastructure for the platform Envision using DHT that can be adapted and efficient to the transmission of H264 SVC stream. In the following points, we present some of the existing DHT systems, a proposal of an P2P architecture using DHT and we will talk about some points to improve those systems with features like NAT traversal or locality awareness.

A Distributed Hash Table (DHT) is the concept of hash table but in a distributed environment. All the entities participating of in the activity of the distributed environment own a part of the hash table. It's a technology used to locate and identify resources in distributed systems like P2P networks.

DHT improves several properties in P2P networks like *decentralization*, in fact we don't need a central node to index resources because the index is shared among all the nodes of the network in the DHT. They have also a very good *scalability*, DHT has proved to be a useful substrate for large distributed systems [61]. *Fault tolerance*, in fact, there is not an essential node, each node can join or leave arbitrarily the network, DHT systems are self-organizing. they automatically adapt to the arrival, departure and failure of nodes. when a peer leaves the network, its part of the hash table is shared among certain peers in the network. When a node joins the network, a part of the DHT depending on its identifier or key is attributed to this node so that it can participate to the activity of the network.

There is many ways to design a DHT depending on how they organise the network, the way they build their identifiers space, their routing strategies. In the following we present some of the main DHT systems.

5.1 DHT and existing DHT-Based P2P Systems

5.1.1 Pastry

Pastry is a scalable, distributed object location and routing substrate for wide-area peer-to-peer applications[63]. In pastry algorithm, a unique numeric identifier of 128-bit is randomly assigned to each node and resource. Each node is responsible of keys that are closest numerically. The topology of a pastry overlay is a circle, nodes are placed in the circular space depending of the value of their identifier. In Pastry, the neighbours of a node are nodes that have identifiers numerically closed. To provide a more routing, each node can have another set of neighbours spread out in the key space. A node in Pastry maintains a routing table of $\theta(log\ n)$ neighbours. The lookup up mechanism of Pastry is based on the longest shared prefix. In fact, queries are forwarded to node that has the longest shared prefix with the key resource. Pastry routes queries within $\theta(log\ n)$ steps in a space of n nodes.

5.1.2 Chord

Chord, a Scalable Peer-to-Peer Lookup Protocol for Internet Applications is one of the most popular DHT algorithm [67]. It also organise peers in a circular key space. In Chord, neighbours are called *successors*. Each node keeps a routing table of its successors, its *successor list*. A node is responsible of a resource if its identifier most closely follows the resource key numerically. Performances in Chord are pretty much the same as in Pastry, a successor list of $\theta(log\ n)$, and a lookup perform in $\theta(log\ n)$ hops in a space of n nodes.

5.1.3 Viceroy

Viceroy, a Scalable and Dynamic Emulation of the Butterfly [45] is designed to handle the discovery and location of data and resources in a dynamic butterfly fashion [43]. It functions as a distributed hash table (DHT), managing the disribution of data among a changing set of servers and allowing clients to contact any participating server in the network to find any stored resource by name. It uses consistent hashing like Chord [40]. Viceroy routes queries in $\theta(log\ n)$ with n the number of peers in the network.

5.1.4 Kademlia

Kademlia, a Peer-to-Peer Information System Based on the XOR Metric [47] is problably the DHT algorithm the most implemented in P2P applications. Kademlia's benefits result from its use of a novel XOR metric for distance between points in the key space. XOR is symmetric, allowing Kademlia participants to receive lookup queries from precisely the same distribution of nodes contained in their routing tables. its routing algorithm is dynamic, in contrast with previous algorithm, Kademlia can send a query to any node within an interval, allowing it to select

routes based on latency or even send parallel asynchronous queries. The neighbour list in kademlia is called the k-bucket $\theta(k * log_2 bN)$ of updated by information about nodes like *time last seen*, *mostly seen node*... Kademlia routes queries in $\theta(log_2 bN)$

5.1.5 CAN

In CAN, a Scalable Content Addressable Network [60], the overlay is organised in a virtual d-dimensional Cartesien coordinate space on a d-torus. in a CAN system we assign to nodes and resources as identifier, coordinates created by a hash function of m bits for each coordinate. The coordinate space is dynamically partitioned among all the nodes of the system such as every nodes owns its individual distinct zone, these zones are hypercubal regions. Nodes are responsible of resources that keys are in the zone they own. The neighbours of a node are the nodes that own the contiguous hypercubes [61]. In a d-dimensional torus, two nodes (or zones) are neighbours if their edges overlap in exactly (d-1) dimensions and abut in exactly 1 dimension. Performances in CAN are different from the previous algorithms, a node has $\theta(log\ d)$ with d the number of dimensions of the key space. The lookup is performed in $\theta(dn^{1/n})$ with n the number of nodes.

5.2 Proposal of a Hybrid P2P network

In this section, we present a view of the P2P architecture using DHT that we propose. The architecture we propose is a hybrid architecture *structured + centralized* using super-peers. Our choice is motivate by the factor of heterogeneity which have been found very important in P2P networks [64].

5.2.1 Heterogeneity in P2P networks

We notice, that many of proposal of structured P2P network, consider the role of each node as equal because they don't take into account the heterogeneity of peers [61, 64]. All the peers don't have the same capacity in term of memory, bandwidth, processing power. So if the workload is the same for all the peers, it can result in some bottleneck or deny of service at some points of the network. A way to deal with heterogeneity is to use architecture based on super peer. This kind of architecture has the advantages to have a good load balancing so its deals very well with heterogeneity.

[72] suggests a technique to reduce the workload on a super-peer. They design a super-peer as a group of k nodes, a virtual super-peer *k–redundant*, and queries are sent to the k nodes in a round–robin manner.

Usually, P2P systems using super-peers are unstructured networks like Gnutella and a drawback of those systems is that their search mechanism is flooding based or random walk based so it's not efficient especially for rare resources. What we propose

to improve the search mechanism is to organise the super-peers in a structured network using a DHT and the cluster around super-peer will be organised in a centralized way. The picture 15 shows the network we propose. *SN* are super-peers or super-nodes and *N* are normal peers or nodes.

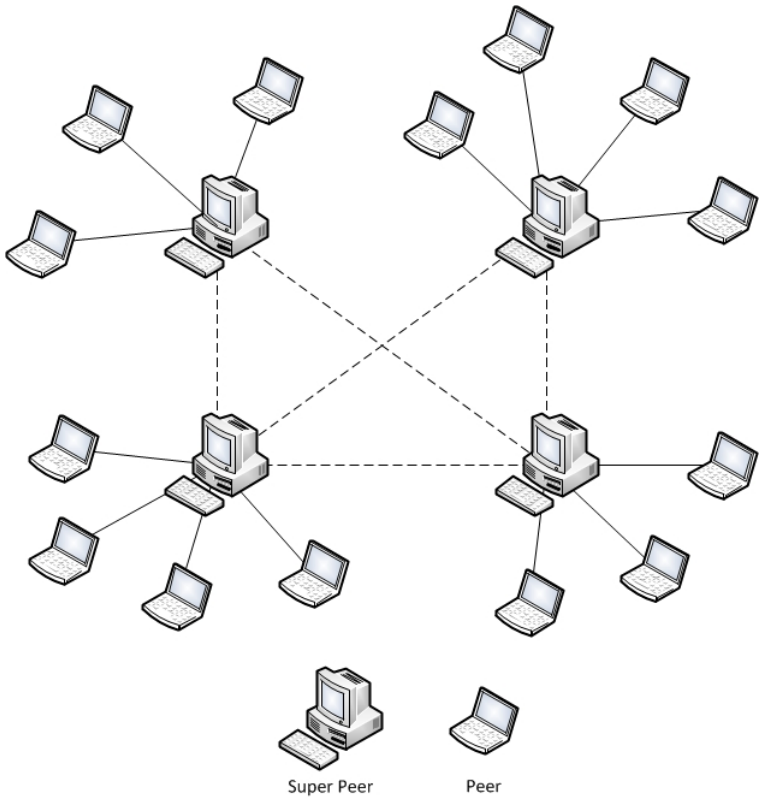


Figure 15: Hybrid Network : structured P2P (SN) + centralized (N + SN)

6 DHT and NAT traversal

We already saw the issue of NAT traversal in P2P networks. This issue, is more present in structured P2P networks because problems doesn't appear in connection establishment for a resource download like in centralized P2P network but also when a peer joins the network. In a structured network, the new peer has to be inserted in the network so it can be a problem of NAT traversal but also in the resources search mechanism because the index is no more centralized on a server on a public zone but decentralized and shared among all the peers of the network.

We propose to use a third-party server on which peers will be connected in the purpose to help in the connections establishment. What we plan, it's that it will be the same node which will insert nodes in the P2P network, a node known and accessible

by all the others peers.

To improve the overhead of the overlay network, we have to choose an efficient routing technique for the messages of the resources search mechanism for a environment where peers located behind NAT routers could participate to the activity of the network. We can notice three routing techniques : iterative routing, symmetric recursive routing, symmetric semi-recursive routing.

6.1 Iterative routing

In this kind of routing, a node N sends a request to a neighbour, if the neighbour doesn't have the response to the request, it answers with the address of a node (its neighbour) to which send the request. Then, the node N sends the request to that node and so on, until the resource location is found if it exists.

This kind of routing doesn't use a lot of resource (processing power) of the neighbours and allow to have a better knowledge of what happened in case of a dysfunction in a lookup because we can build the tree of the lookup and locate a failing node.

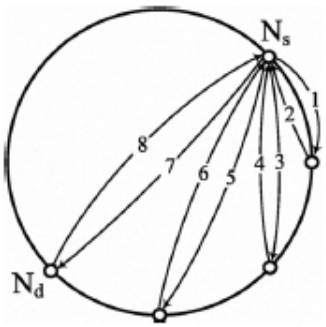


Figure 16: Iterative routing

6.2 Symmetric recursive routing

In this routing technique, each node forwards requests to the next nodes. No informations is sent to the originator during the lookup. So the overhead of this technique is less than in an iterative routing. The response of the request is routed back by the same path back to the originator of the request.

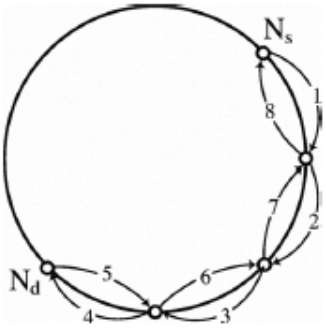


Figure 17: Symmetric recursive routing

6.3 Symmetric semi-recursive routing

This method works like the recursive routing but responses don't take the path back to the originator of the request. The response is sent directly to originator. This method is faster than the recursive routing.

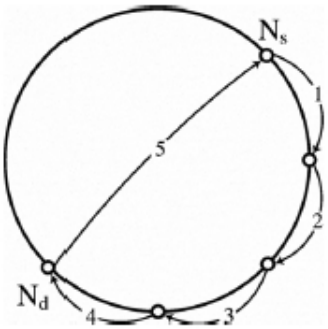


Figure 18: Symmetric semi-recursive routing

6.4 Proposal

Previous research shows that symmetric recursive technique is more efficient than iterative routing [73] (which gives more explanations about the three methods). We have less connections establishment because messages are routed among neighbours, and thus, less use of NAT traversal solution. What we suggest, is to use this method of routing but we plan to modify the algorithm in the manner that we will improve the delay of response and a good knowledge of what happen in a lookup process.

We suggest that when a node forwards a request, it sends a notifications directly back to the originator and the node responsible of the request will send the response also directly back to the originator.

In this method, we have not problem of NAT traversal if we use the architecture of P2P network we propose (cf. 5.2 page 29) because the overlay network using DHT is formed only by super-nodes which are accessible by all the peers. thus, if a lookup process fails, the originator know where a problem happened and it doesn't have to restart the lookup process to the beginning.

To resume the proposal, we suggest a network with a node in public zone which will insert new nodes in the overlay network and will help in connection establishment between peers located behind NAT routers and the overlay network using DHT will use the routing method we presented.

7 DHT and Locality awareness

One of the main characteristic of DHT-based is to reduce the numbers of hops in the lookup process based on nodes and resources identifiers. However those identifiers are created in a randomly manner, it doesn't take into account their geographic position, bandwidth (upload or download), ISP...

So, in this kind of system we measure the performance of the system considering the path size of hops in the lookup process. Considering the fact that identifiers are created without taking into account physical network, it can result with path with high latency if its includes hops across different continents or nodes that are very distant.

7.1 Coordinate based systems

To improve the above points, several systems are based on coordinate system on Cartesian system. In fact, previous works, shown it possible to give coordinate to peers so that the euclidean distance between peers can determine the latency between them [55]. Several others coordinate-based system has been created, vivaldi [26], GNP [55], PIC [24], ICS [41], NPS [28]... and certains has been integrated in some existing DHT systems, for example vivaldi in Chord [67].

To improve performance in structured P2P network with locality awareness, some approaches are suggested, routing proximity, proximity neighbour selection, geography layout, ISP-friendly P2P system [61, 23].

7.2 Proximity Neighbour Selection (PNR)

When a peer join the P2P network, we have to build its routing table. In this operation we take network proximity into account so peers that are closed in the

P2P network, are also closed in the Internet network. The proximity can be in term of latency or bandwidth (upload or download).

7.3 Proximity Routing (PR)

The algorithm in a proximity Routing send request to the node which is not only the one which make the best progress toward the resource but also the closest node in the routing table in term of latency.

7.4 Geography Layout

The routing tables are built in a manner that makes nodes that are closed in the P2P Network are also closed geographically.

7.5 ISP-friendly P2P system

Nowadays, P2P became an important part in the Internet traffic and because hops in a routing process can involve trips across different continents which is costly for ISPs, researches are made to find a way to make P2P users and ISPs cooperate to improve P2P service quality [22, 21, 29]. It can be helpful to develop new services like multicast.

7.6 Proposal

Considering these points, we propose to use CAN as DHT system because it is a coordinate based system which make it more compatible to an evolution taking locality awareness into account by introducing Vivaldi like coordinate based system to help in locality awareness to improve performance in term of latency. We choose Vivaldi like [27] because vivaldi algorithm is lightweight and decentralized.

8 DHT and Scalable Video Coding (SVC)

The purpose of the project we are working on is to provide a platform of SVC video streaming. So, we are searching an optimal way to index these kind of files in a P2P network using DHT.

8.1 Scalable Video Coding (SVC)

The Scalable Video Coding (SVC) is a standardized extension of the H.264/AVC video coding standard [65]. The SVC enables the encoding of a high-quality video bit stream that contains one or more subset bit streams. Those sub-streams can themselves be decoded with a complexity and reconstruction quality similar to that

achieved using the existing H.264/AVC design with the same quantity of data as in the subset bit stream. Hence, it enables the transmission and decoding of partial bit streams to provide video services with lower temporal or spatial resolutions or reduced fidelity. Hence, SVC provides functionalities such as graceful degradation in lossy transmission environments as well as bit rate, format, and power adaptation.

A video bit stream is called scalable when parts of the stream can be removed in a way that the resulting sub-stream forms another valid bit stream for some target decoder. Moreover, the sub-stream represents the source content with a reconstruction quality that is less than the one of the complete original bit stream. Bit streams that do not provide this property are referred to as single-layer bit streams. The usual modes of scalability of SVC are temporal, spatial, and quality scalability.

- **Temporal scalability :** this scalability concerns the different frame bit-rates of the video provided by the SVC file.
- **Spatial scalability :** this scalability concerns the different resolution of the video.
- **Quality scalability :** this scalability concerns the different qualities provide, is often informally referred to as signal-to-noise ratio (SNR). Quality scalability is also commonly referred to as fidelity or *SNR scalability*

The different types of scalability can also be combined, so that a multitude of representations with different spatio-temporal resolutions and bit rates can be supported within a single scalable bit stream. The figure

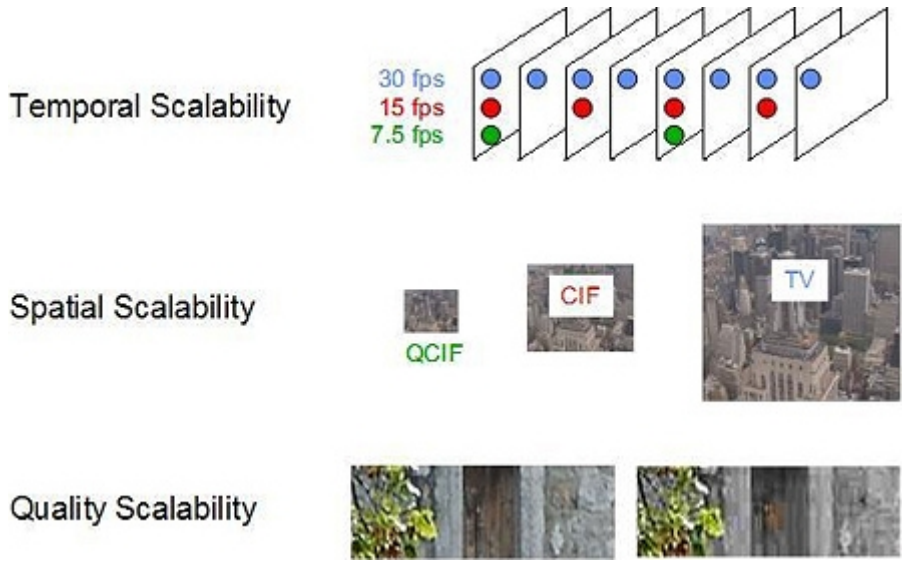


Figure 19: Types of scalabilities of SVC

8.2 Indexing by scalability keys of SVC resource

Considering the fact that a SVC file can contains many scalabilities and a peer may be interested to a certain kind of scalability and not an another. We suggest to index SVC files by their scalabilities. We propose two architectures of ours hash table based on the nature of the key we use for indexing resource.

The first method to index resource in our hash table is to use keys that represent scalability and the value of the hash table will be a structure-like when we will store informations about peers which possesses this resource (IP address, percent of the resource owned, bandwidth, geographic position...). The picture 20 shows the structure of this kind of hash table.

KEY	VALUE
Resource1_scalability0	...
Resource1_scalability3	...
...	...
...	...
Resource1_scalability4	...
Resource2_scalability7	...
Resource3_scalability0	...

Figure 20: Hash table indexed by resource scalability keys

8.3 Indexing by SVC resource keys

In this method we suggest to use the resource key in our hash table and the value will be a hash table in the format presented in the previous point.

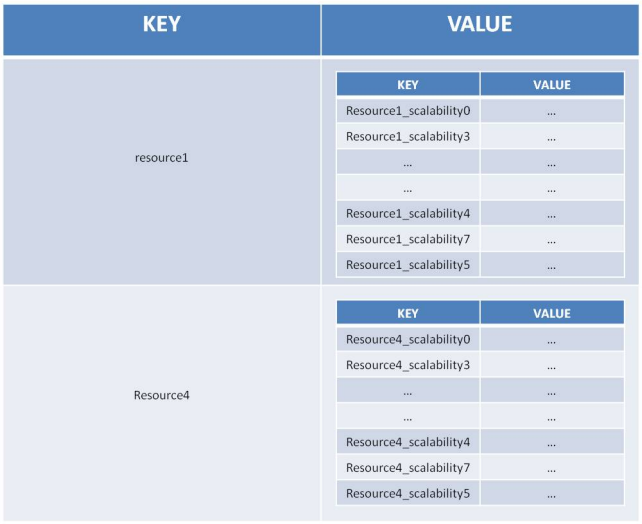


Figure 21: Hash table indexed by resource keys

We propose to use the second proposal because it can be more scalable to other kind of resource.

Conlusion

In this document, we seen the different method to solve the NAT traversal and the architecture we proposed and implemented. The implementation is not finished, some works has to be done about the TURN-like solution or the security of the application. For, now the capacity to our platform to work across NAT has been tested in the team, with different routers we possess (Linksys, Asus...). Our solution is able to work across all the type of NAT except the symmetric NAT but with the TURN-like solution, we will be able to work across symmetric NAT.

We can't not say that the work about DHT systems, coordinate-based systems and locality awareness is done. We have to study more about it, their implementation and how could we introduce it in the platform we develop. We think that more studies has to be done before implementing a solution.

References

[1] RFC 2663. **IP Network Address Translator (NAT) Terminology and Considerations**, August 1999. 15

[2] RFC 3022. **Traditional IP Network Address Translator (Traditional NAT)**, January 2001. 10

[3] RFC 3303. **Middlebox communication architecture and framework**, August 2002. 17

[4] RFC 3304. **Middlebox Communications (MidCom) Protocol Requirements**, August 2002. 17

[5] RFC 3947. **Negotiation of NAT-Traversal in the IKE**, January 2005. 17

[6] RFC 4380. **Teredo: Tunneling IPv6 over UDP through Network Address Translations (NATs)**, February 2006. 17

[7] RFC 4787. **Network Address Translation (NAT) Behavioral Requirements for Unicast UDP**, January 2007. 11

[8] RFC 5189. **Middlebox Communication (MIDCOM) Protocol Semantics**, March 2008. 17

[9] RFC 5245. **Interactive Connectivity Establishment (ICE): A Protocol for Network Address Translator (NAT) Traversal for Offer/Answer Protocols**, April 2010. 20

[10] RFC 5382. **NAT Behavioral Requirements for TCP**, October 2008. 25

[11] RFC 5389. **Session Traversal Utilities for NAT (STUN)**, October 2008. 12, 19

[12] RFC 5626. **Managing Client-Initiated Connections in the Session Initiation Protocol (SIP)**. page 18, October 2009. 23

[13] RFC 5766. **Traversal Using Relays around NAT (TURN): Relay Extensions to Session Traversal Utilities for NAT (STUN)**, April 2010. 16

[14] RFC 5780. **NAT Behavior Discovery Using Session Traversal Utilities for NAT (STUN)**, May 2010. 19

[15] RFC 5928. **Traversal Using Relays around NAT (TURN) Resolution Mechanism**, August 2010. 16

[16] RFC 5991. **Teredo Security Updates**, Septembre 2010. 17

[17] RFC 6062. **Traversal Using Relays around NAT (TURN) Extensions for TCP Allocations**, November 2010. 16

[18] RFC 6081. **Teredo Extensions**, January 2011. 17

[19] RFC 6156. **Traversal Using Relays around NAT (TURN) Extension for IPv6**, April 2011. 16

[20] RFC 6223. **Indication of Support for Keep-Alive**. April 2011. 23

[21] V. Aggarwal, O. Akonjang, and A. Feldmann. **Improving user and ISP experience through ISP-aided P2P locality**. In *INFOCOM Workshops 2008, IEEE*, pages 1–6, April 2008. 34

[22] Vinay Aggarwal, Anja Feldmann, and Christian Scheideler. **Can ISPS and P2P users cooperate for improved performance?** *SIGCOMM Comput. Commun. Rev.*, 37:29–40, July 2007. 34

[23] Miguel Castro, Peter Druschel, Y. Charlie Hu, and Antony Rowstron. **Exploiting network proximity in distributed hash tables**. In *in International Workshop on Future Directions in Distributed Computing (FuDiCo*, pages 52–55, 2002. 33

[24] Manuel Costa, Miguel Castro, Antony Rowstron, and Peter Key. **PIC: Practical Internet Coordinates for Distance Estimation**. In *In International Conference on Distributed Systems*, pages 178–187, 2003. 33

[25] C. Cramer, K. Kutzner, and T. Fuhrmann. **Bootstrapping locality-aware P2P networks**. In *Networks, 2004. (ICON 2004). Proceedings. 12th IEEE International Conference on*, volume 1, pages 357–361 vol.1, November 2004. 6

[26] Frank Dabek, Russ Cox, Frans Kaashoek, and Robert Morris. **Vivaldi: A Decentralized Network Coordinate System**. In *In SIGCOMM*, pages 15–26, 2004. 33

[27] Frank Dabek, Jinyang Li, Emil Sit, James Robertson, M. Frans Kaashoek, and Robert Morris. **Designing a DHT for low latency and high throughput**. In *IN PROCEEDINGS OF THE 1ST NSDI*, pages 85–98, 2004. 34

[28] T. S. Eugene and Ng Hui Zhang. **A Network Positioning System for the Internet**. In *Proceedings of the 4th Symposium on Internet Technologies and Systems; USENIX*, 2004. 33

[29] A. Feldmann. **A possibility for ISP and P2P collaboration**. In *Broadband Communications, Networks and Systems, 2008. BROADNETS 2008. 5th International Conference on*, page 239, September 2008. 34

[30] Bryan Ford and et al. **Peer-to-Peer Communication Across Network Address Translators**. In *IN USENIX ANNUAL TECHNICAL CONFERENCE*, pages 179–192, 2005. 19, 25

[31] UPnP Forum. **Internet Gateway Device (IGD) V 1.0, WANIPConnection:1** http://upnp.org/specs/gw/UPnP-gw-WANIPConnection-v1-Service.pdf. 25

[32] UPnP Forum. **Universal Plug and Play.** http://www.upnp.org. 24

[33] UPnP Forum. **UPnP Device Architecture 1.1.** http://upnp.org/specs/arch/UPnP-arch-DeviceArchitecture-v1.1.pdf, October 2008. 18

[34] Hostip.info. http://www.hostip.info. 23

[35] Chi-Long Huang and Shaw-Hwa Hwang. **The asymmetric NAT and its traversal method**. In *Proceedings of the Sixth international conference on Wireless and Optical Communications Networks*, WOCN'09, pages 177–180, Piscataway, NJ, USA, 2009. IEEE Press. 12

[36] Internet-Draft. **Port Control Protocol (PCP) NAT-PMP Interworking Function**, March 2011. 19

[37] Internet-Draft. **TCP Candidates with Interactive Connectivity Establishment (ICE)**, April 2011. 20

[38] Internet-Draft. **Traversal Using Relays around NAT (TURN) Uniform Resource Identifiers**, January 2011. 16

[39] J. R. Diaz Santos M. Esteve Domingo. J. Lloret Mauri, G. Fuster. **Analysis and characterization of Peer-to-Peer Filesharing Networks**. In *WSEAS TRANSACTIONS on SYSTEMS. Issue 7*, volume 3, pages 2574–2579, sept. 2004. 5

[40] David Karger, Eric Lehman, Tom Leighton, Matthew Levine, Daniel Lewin, and Rina Panigrahy Abstract. **Consistent hashing and random trees: Distributed caching protocols for relieving hot spots on the World Wide Web**. In *In Proc. 29th ACM Symposium on Theory of Computing (STOC*, pages 654–663, 1997. 28

[41] Hyuk Lim, Jennifer C. Hou, Chong-Ho Choi, and Seoul Korea. **Constructing Internet Coordinate System Based on Delay Measurement**, 2003. 33

[42] Zhewen Lin and Tiantong You. **TT-STUN protocol design for effective TCP NAT traversal**. In *Broadband Network and Multimedia Technology (IC-BNMT), 2010 3rd IEEE International Conference on*, pages 970–974, oct. 2010. 20

[43] Eng Keong Lua, Jon Crowcroft, Marcelo Pias, Ravi Sharma, and Steven Lim. **A Survey and Comparison of Peer-to-Peer Overlay Network Schemes**. *IEEE Communications Surveys and Tutorials*, 7:72–93, 2005. 28

[44] MSDN Magazine. **Get Closer to the Wire with High-Performance Sockets in .NET** http://msdn.microsoft.com/en-us/magazine/cc300760.aspx, Last visite 7 june 2011. 22

[45] Dahlia Malkhi, Moni Naor, and David Ratajczak. **Viceroy: A Scalable and Dynamic Emulation of the Butterfly**. pages 183–192, 2002. 28

[46] MaxMind. http://www.maxmind.com/app/api. 23

[47] Petar Maymounkov and David Mazières. **Kademlia: A Peer-to-Peer Information System Based on the XOR Metric**. In *Revised Papers from the First International Workshop on Peer-to-Peer Systems*, IPTPS '01, pages 53–65, London, UK, 2002. Springer-Verlag. 28

[48] Elena Meshkova, Janne Riihijärvi, Marina Petrova, and Petri Mähönen. **A survey on resource discovery mechanisms peer-to-peer and service discovery frameworks**. *Comput. Netw.*, 52:2097–2128, August 2008. 7

[49] Dejan S. Milojicic, Vana Kalogeraki, Rajan Lukose, Kiran Nagaraja, Jim Pruyne, Bruno Richard, Sami Rollins, and Zhichen Xu. **Peer-to-Peer Computing**. Technical report, 2002. 5

[50] MonoTorrent. http://www.monotorrent.com. 21

[51] MSDN. **Network Address Translation (NAT) Traversal API.** http://msdn.microsoft.com/en-us/library/aa366187%28v=VS.85%29.aspx. 24

[52] MSDN. **UPnP APIs.** http://msdn.microsoft.com/en-us/library/aa382303%28v=vs.85%29.aspx. 24

[53] MSDN. **Using SO_REUSEADDR and SO_EXCLUSIVEADDRUSE** http://msdn.microsoft.com/en-us/library/ms740621%28v=vs.85%29.aspx, Last visite 05 June 2011. 26

[54] A. Muller, N. Evans, C. Grothoff, and S. Kamkar. **Autonomous NAT Traversal**. In *Peer-to-Peer Computing (P2P), 2010 IEEE Tenth International Conference on*, pages 1–4, aug. 2010. 20

[55] T. S. Eugene Ng and Hui Zhang. **Towards Global Network Positioning**. In *Proceedings of the First ACM SIGCOMM Workshop on Internet Measurement*, pages 25–29, 2001. 33

[56] Eyeball AnyFirewall Technology White Paper. **NAT Traversal for VoIP and Internet Communications using STUN, TURN and ICE.** http://www.nattraversal.com/EyeballAnyfirewallWhitePaper.pdf. January 2011. 23

[57] Xuena Peng, Jinshan Sun, Hong Zhao, Dapeng Li, and Yingyou Wen. **An ALG-Based NAT Traversal Solution for SIP-Based VoIP**. In *Proceedings of the 2009 Second International Workshop on Computer Science and Engineering - Volume 02*, IWCSE '09, pages 101–105, Washington, DC, USA, 2009. IEEE Computer Society. 15

[58] B. Pourebrahimi, K. Bertels, and S. Vassiliadis. **A survey of peer-to-peer networks**. In *Proceedings of the 16th Annual Workshop on Circuits, Systems and Signal Proessing*, 2005. 5, 7

[59] Nito Programming. **Message Framing.** http://nitoprograms.blogspot.com/2009/04/message-framing.html. 24

[60] Sylvia Ratnasamy, Paul Francis, Mark Handley, Richard Karp, and Scott Shenker. **A Scalable Content-Addressable Network**. In *IN PROC. ACM SIGCOMM 2001*, pages 161–172, 2001. 28

[61] Sylvia Ratnasamy, Scott Shenker, and Ion Stoica. **Routing Algorithms for DHTs: Some Open Questions**. pages 45–52, 2002. 27, 28, 29, 33

[62] John Risson and Tim Moors. **Survey of research towards robust peer-to-peer networks : search methods**. *Comput. Netw.*, 50:3485–3521, December 2006. 7

[63] Antony I. T. Rowstron and Peter Druschel. **Pastry: Scalable, Decentralized Object Location, and Routing for Large-Scale Peer-to-Peer Systems**. In *Proceedings of the IFIP/ACM International Conference on Distributed Systems Platforms Heidelberg*, Middleware '01, pages 329–350, London, UK, 2001. Springer-Verlag. 27

[64] Stefan Saroiu, P. Krishna Gummadi, and Steven D. Gribble. **A Measurement Study of Peer-to-Peer File Sharing Systems**. 2002. 29

[65] H. Schwarz, D. Marpe, and T. Wiegand. **Overview of the Scalable Video Coding Extension of the H.264/AVC Standard**. *Circuits and Systems for Video Technology, IEEE Transactions on*, 17(9):1103 –1120, sept. 2007. 34

[66] Sharp2p. http://sourceforge.net/projects/sharpp2p. 21

[67] Ion Stoica, Robert Morris, David Karger, M. Frans Kaashoek, and Hari Balakrishnan. **Chord: A Scalable Peer-to-Peer Lookup Service for Internet Applications**. pages 149–160, 2001. 28, 33

[68] Dimitrios Tsoumakos and Nick Roussopoulos. **A Comparison of Peer-to-Peer Search Methods**. pages 61–66, 2003. 7

[69] Utrace. **Locate IP addresses and domain names.** http://en.utrace.de/api.php. 23

[70] BitTorrent Specification wiki. http://wiki.theory.org/BitTorrentSpecification. 21

[71] D.I. Wolinsky, P. St. Juste, P.O. Boykin, and R. Figueiredo. **Addressing the P2P Bootstrap Problem for Small Overlay Networks**. In *Peer-to-Peer Computing (P2P), 2010 IEEE Tenth International Conference on*, pages 1–10, aug. 2010. 6

[72] Beverly Yang and Hector Garcia-m. **Designing a Super-peer Network**. 2003. 29

[73] Cheng Yin-bo and Wen Xiang-ming. **Performance analysis of look strategies of DHT in NATed environment**. In *Computational Intelligence and Industrial Applications, 2009. PACIIA 2009. Asia-Pacific Conference on*, volume 2, pages 65–68, November 2009. 32

[74] Lizhuo Zhang, Weijia Jia, Xun Xiao, Bin Dai, and Huan Li. **Research of TCP NAT Traversal Solution Based on Port Correlation Analysis & Prediction Algorithm**. In *Wireless Communications Networking and Mobile Computing (WiCOM), 2010 6th International Conference on*, pages 1–4, sept. 2010. 20