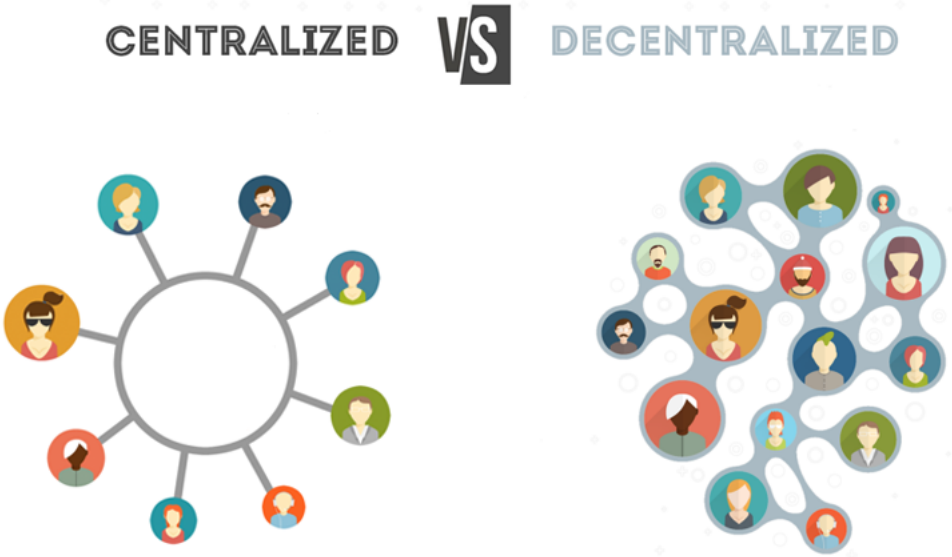
Exploration des solutions de contournement des dispositifs de traduction d'adresses réseau

Étude des solutions de communication pair à pair dans le web



Auteurs

Benjamin NIDDAM

Encadrants

Adrien LUXEY-BITRI

// page vide après la page de garde

Table des matières

Chapitre 1

Introduction

1.1 Contexte

Dans le cadre du PJI (Projet Individuel) en fin de master 1, j'ai choisi de travailler sous la tutelle de monsieur LUXEY-BITRI sur le sujet des communications réseaux dites pair à pair. Avec peu de connaissances en réseaux, j'y ai vu l'occasion de découvrir un domaine qui m'était inconnu et de me confronter à un sujet de recherche pour l'instant peu exploré mais avec un potentiel d'application très important. En effet, si aujourd'hui les communications réseaux sont majoritairement centralisées, on voit émerger de plus en plus de réseaux dits décentralisés.

Les réseaux pair-à-pair (P2P), offrent une approche alternative à la communication et à l'échange de données. Contrairement aux réseaux centralisés, où les données transitent par un point central, les réseaux P2P permettent aux utilisateurs de communiquer directement entre eux, en utilisant leurs propres ressources (telles que la puissance de calcul et la bande passante).

Ce modèle de communication présente plusieurs avantages, notamment une meilleure résilience face aux pannes, une répartition de la charge plus équilibrée et une réduction de la dépendance à l'égard d'entités centrales. Les réseaux P2P sont largement utilisés dans divers domaines, tels que le partage de fichiers (comme BitTorrent), la messagerie instantanée (comme Skype) et les systèmes de paiement (comme Bitcoin).

J'ai choisi ce sujet car la majorité de mon travail personnel consiste à développer des applications web basées sur des architectures centralisées. Je souhaitais donc découvrir un nouveau domaine et me confronter à de nouvelles problématiques. De plus, les réseaux pair à pair sont un sujet de recherche très actuel et qui a un fort potentiel d'application. En effet, les réseaux pair à pair sont de plus en plus utilisés dans le domaine des communications réseaux. On peut citer par exemple le protocole WebRTC qui permet la communication audio et vidéo entre deux clients sans passer par un serveur.

1.2 Problématique

Quelles sont les solutions possibles pour mettre en place une communication textuelle temps réel entre deux clients dans un réseau pair à pair ?

1.3 Objectifs

L'objectif de ce projet est l'étude des mécanismes permettant le "pair à pair" dans le web moderne grace à l'implémentation de prototypes permettant la communication textuelle entre deux clients.

Chapitre 2

Contexte et Motivation

Dans le domaine des communications réseaux pair à pair, il est nécessaire d'établir une communication directe entre deux clients. Cependant, certains défis se posent lors de la mise en place d'une telle communication, notamment en présence de dispositifs de traduction d'adresses réseau (NAT [1]) et de pare-feu. Ces dispositifs qui limitent la connectivité directe entre les clients. Dans ce rapport, nous allons examiner les défis liés à la mise en place d'une communication directe entre les clients,

2.1 Les dispositifs de traduction d'adresses réseau (NAT)

Le NAT est souvent utilisé dans les réseaux domestiques et les petites entreprises pour partager une seule adresse IP publique fournie par le fournisseur d'accès à Internet (FAI) entre plusieurs dispositifs du réseau local. Plutôt que d'attribuer une adresse IP publique unique à chaque dispositif, le NAT attribue des adresses IP privées à chaque dispositif du réseau local. Lorsque ces dispositifs se connectent à Internet, le NAT traduit automatiquement les adresses IP privées en adresse IP publique, permettant ainsi à ces dispositifs de communiquer avec des serveurs sur Internet.

Outre la conservation des adresses IP publiques, le NAT offre également une certaine forme de sécurité en masquant les adresses IP privées des dispositifs internes. Cela rend plus difficile pour les personnes extérieures au réseau local de cibler directement ces dispositifs.

1. Network address translation

Cependant, le NAT introduit une limitation en ce qui concerne la connectivité entre les dispositifs du réseau local et les dispositifs externes. Étant donné que les adresses IP privées ne sont pas routables sur Internet, les dispositifs internes d'un réseau NAT ne peuvent pas recevoir de connexions entrantes directement. Ils sont uniquement capables d'initier des connexions sortantes vers des serveurs externes. Cela signifie que les dispositifs internes ne peuvent pas être facilement accessibles depuis Internet, sauf si des règles spécifiques de translation de port (port forwarding) sont configurées pour rediriger les connexions entrantes vers des dispositifs spécifiques.

En résumé, le NAT offre une solution efficace pour partager une adresse IP publique et protéger les dispositifs internes, mais il peut limiter la connectivité directe entre les clients du réseau local et les dispositifs externes.

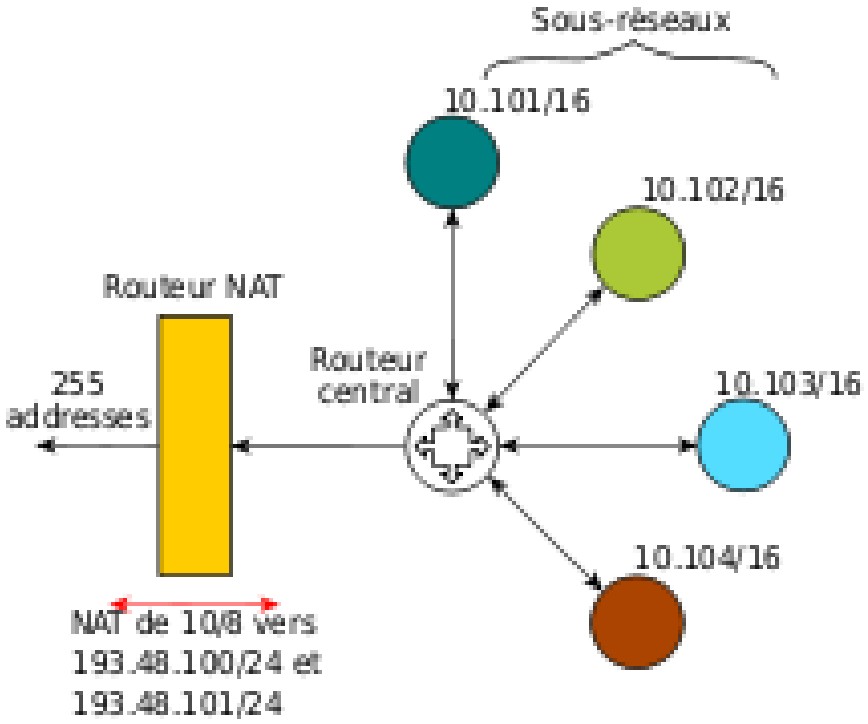


FIGURE 2.1 – Exemple de NAT

2.2 Problèmes liés aux dispositifs de traduction d'adresses réseau

Lorsqu'un client se trouve derrière un NAT, il reçoit une adresse IP privée non routable, qui n'est pas accessible depuis Internet. Cela rend difficile l'établissement d'une communication directe avec d'autres clients qui se trouvent également derrière des NAT.

2.3 La nécessité d'un tiers pour faciliter l'établissement de la connexion

Dans le contexte des communications pair à pair, il devient nécessaire d'avoir un tiers (un serveur intermédiaire) pour faciliter l'établissement de la connexion entre les clients. Ce tiers agit comme un relai en permettant aux clients de s'échanger les informations nécessaires pour établir la connexion, même s'ils ne peuvent pas le faire directement en raison des NAT.

Le relai joue un rôle crucial en tant qu'entremetteur dans le processus d'établissement de la connexion entre les clients. Lorsqu'un client souhaite établir une communication avec un autre client, il s'enregistre auprès du relai et fournit des informations telles que son adresse IP publique et son port. Le relai enregistre ces informations et les rend disponibles aux autres clients qui souhaitent se connecter à ce client spécifique.

Lorsqu'un client souhaite établir une connexion avec un autre client, il contacte le relai pour obtenir les informations de connexion de sa cible. Le relai transmet alors ces informations au client demandeur, lui permettant ainsi d'établir une connexion directe avec le client cible.

Il est important de noter que le rôle du relai se limite à faciliter l'établissement de la connexion en transmettant les informations nécessaires entre les clients. Une fois que la connexion directe est établie entre les clients, les données échangées ne passent pas par le relai, mais sont directement transmises entre les clients.

2.4 Solutions alternatives ou complémentaires pour contourner les limitations des dispositifs NAT

Bien que les dispositifs de traduction d'adresses réseau puissent poser des défis en matière de connectivité directe entre les clients, il existe plusieurs solutions alternatives ou complémentaires pour contourner ces limitations. En voici quelques-unes :

Les protocoles TURN et STUN

Le protocole STUN [2] permet à un client d'obtenir des informations sur son adresse IP publique et le type de NAT qu'il traverse. Cela permet au client de comprendre comment il est connecté au réseau et d'adapter ses stratégies de connexion en conséquence. STUN est généralement utilisé pour faciliter l'établissement de connexions pair à pair lorsque les clients sont derrière des NAT symétriques ou des pare-feu restrictifs.

Le protocole TURN [3] est utilisé lorsque les connexions directes ne sont pas possibles en raison de pare-feu ou de NAT restrictifs. TURN utilise un serveur intermédiaire (relais) auquel les clients se connectent pour acheminer leurs données. Ce serveur intermédiaire agit comme un proxy et transfère les données entre les clients. Cela permet aux clients de communiquer même s'ils ne peuvent pas établir une connexion directe.

Lorsqu'un client rencontre des difficultés pour établir une connexion directe en raison des restrictions du réseau, il peut utiliser STUN pour obtenir son adresse IP publique, puis essayer d'établir une connexion directe. Si cela échoue, le client peut passer à l'utilisation de TURN pour acheminer les données via un serveur relais.

Il est important de noter que l'utilisation de TURN peut entraîner une latence supplémentaire car les données doivent être acheminées via le serveur relais. De plus, l'utilisation d'un serveur relais peut entraîner des coûts supplémentaires, notamment si vous utilisez un service cloud ou un fournisseur tiers pour héberger le serveur.

2. Session Traversal Utilities for NAT
3. Traversal Using Relays around NAT

Chapitre 3

Établir une connexion pair à pair

3.1 Avec WebRTC

L'utilisation de WebRTC [1] est l'une des solutions possibles pour mettre en place une communication textuelle en temps réel entre deux clients dans un réseau pair à pair. Pour établir cette connexion, les étapes suivantes peuvent être suivies :

Initialisation d'une pair et récupération de son UUID Chaque client doit être initialisé en tant que pair et se voit attribuer un UUID [2]. Cet identifiant est utilisé pour l'identifier et permettre à d'autres de s'y'connecter.

Partage des UUID entre les deux pairs Une fois que chaque noeuds a obtenu son UUID, ces identifiants doivent être partagés entre les deux clients afin qu'ils puissent s'identifier mutuellement lors de l'établissement de la connexion. Cela peut être réalisé par l'intermédiaire d'un serveur de signalisation ou d'un canal de communication préalablement établi, tel qu'un serveur centralisé ou un autre moyen de communication sécurisé.

Création d'une connexion entre les deux pairs Une fois que chaque pair dispose de l'UUID de l'autre, ils peuvent utiliser ce dernier pour établir une connexion directe entre eux en utilisant le protocole WebRTC. Ce dernier permet aux clients d'établir des connexions directes, contournant ainsi les NAT et réduisant la latence. Le tou grace à l'utilisation de protocoles de traversée NAT (NAT traversal protocol) ou les techniques de trous de poinçonnage (hole punching) pour contourner les limitations des NAT.

1. Web Real-Time Communication
2. Universally Unique Identifier

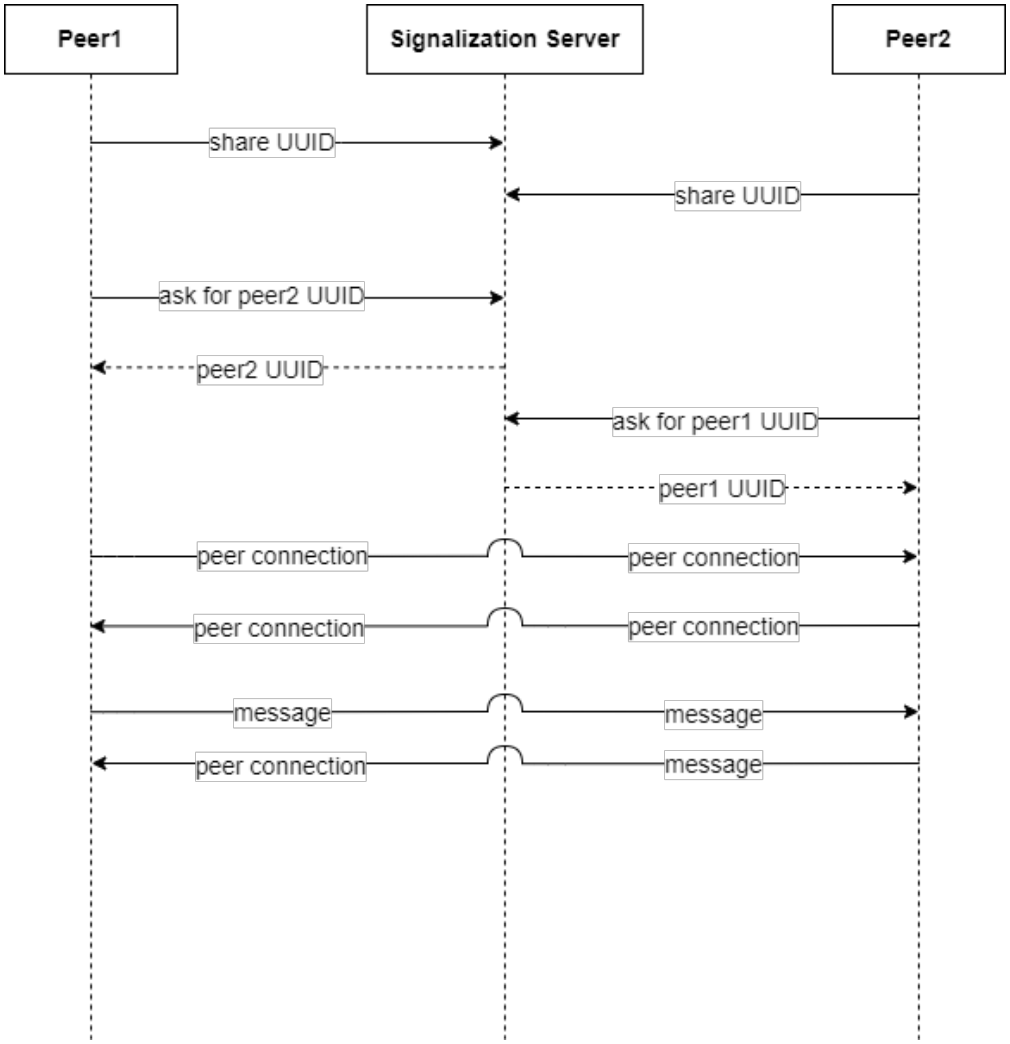


FIGURE 3.1 – Schéma de connection avec WebRTC

Pour pouvoir fonctionner, WebRTC utilise le protocole ICE (Interactive Connectivity Establishment) qui permet de contourner les limitations des NAT et de trouver le chemin le plus efficace pour établir une connexion entre les deux pairs. Ce protocole utilise les techniques de "trou de poinçonnage" (hole punching) pour contourner les limitations des NAT. Et ce grace grace à un serveur STUN qui permet de récupérer l'adresse IP publique et le port du client. Une fois que les deux pairs ont récupéré leurs adresses publiques, ils peuvent s'échanger leurs adresses et ports privés. Cela permet aux deux pairs de s'envoyer des paquets directement sans passer par le serveur STUN.

Par défaut, la librairie WebRTC du navigateur utilise un serveur STUN de Google [3] pour récupérer l'adresse IP publique et le port du client. Cependant, il est possible de configurer un serveur STUN personnalisé pour effectuer cette tâche. Chose que j'ai faite en utilisant le serveur STUN fourni par la librairie coturn [4]. Coturn est un serveur TURN et STUN open source qui permet de configurer un serveur STUN personnalisé. Une fois le serveur configuré, il suffit de le spécifier dans la configuration de la librairie WebRTC du navigateur.

Afin de vérifier le bon fonctionnement du serveur STUN, j'ai utilisé l'outil Trickle ICE [5] qui permet de récupérer les adresses IP et ports utilisés par la librairie WebRTC du navigateur. En utilisant cet outil, j'ai pu vérifier que les adresses IP et ports récupérés sont bien ceux du serveur STUN configuré.

3. stun.l.google.com :19302
4. coturn.net
5. https ://webrtc.github.io/samples/src/content/peerconnection/trickle-ice/

3.2 Avec libp2p

Libp2p [6], (pour "bibliothèque peer-to-peer") est un cadre de mise en réseau peer-to-peer (P2P) qui permet le développement des applications P2P. Il se compose d'une collection de protocoles, de spécifications et bibliothèques qui facilitent la communication P2P entre les participants au réseau, appelées "pairs". Libp2p fournit une interface de programmation d'application (API) pour les applications P2P, ainsi que des implémentations de ces protocoles.

Dans un premier temps développé pour le projet IPFS [7], libp2p, a au fil du temps été mit à jour et amélioré pour être utilisé dans d'autres projets. Il est aujourd'hui réputé pour sa modularité, sa sécurité et sa résilience. C'est un outil puissant pour effectuer des tâches comme le "trou de poinçonnage" (hole punching), la distribution et la diffusion de contenu, la découverte de pairs, la communication sécurisée, etc.

Libp2p est un framework qui permet de créer des applications pair à pair. Il est composé de plusieurs modules qui peuvent être utilisés indépendamment les uns des autres. Il est également possible d'ajouter des modules personnalisés pour répondre à des besoins spécifiques.

Pour établir une connexion pair à pair avec libp2p, les étapes suivantes peuvent être suivies :

Initialisation des nœuds Chaque nœud initialise sa propre instance de Libp2p en configurant les fonctionnalités et les protocoles souhaités.

Découverte des pairs Les nœuds utilisent des mécanismes de découverte pour trouver d'autres pairs disponibles dans le réseau. Cela peut se faire en utilisant des serveurs de rendez-vous, des protocoles de découverte basés sur DHT (Distributed Hash Table), etc.

Échange des adresses multiadresses Une fois que les nœuds ont découvert d'autres pairs, ils s'échangent mutuellement leurs adresses multiadresses. Une adresse multiadresse est une adresse qui encapsule différentes informations, telles que l'adresse IP, le port, les protocoles pris en charge, etc.

6. https ://libp2p.io/
7. InterPlanetary File System

Négociation des protocoles Les nœuds négocient les protocoles qu'ils souhaitent utiliser pour la communication. Cela peut inclure des protocoles spécifiques à une application ou des protocoles de base fournis par Libp2p, tels que TCP [8], WebSockets, etc.

Établissement de la connexion Les nœuds utilisent les adresses multiadresses échangées précédemment pour établir une connexion directe. Cela peut impliquer l'établissement d'une connexion TCP, l'établissement d'une connexion WebSockets, ou d'autres mécanismes de transport pris en charge.

Handshake sécurisé Une fois la connexion établie, les nœuds peuvent effectuer un handshake sécurisé pour s'authentifier mutuellement et échanger des clés de chiffrement. Cela peut se faire en utilisant des protocoles de chiffrement tels que TLS [9].

Communication Une fois que la connexion est établie et sécurisée, les nœuds peuvent commencer à échanger des données selon les protocoles applicatifs spécifiques. Cela peut impliquer l'utilisation de protocoles de haut niveau pour la transmission des messages, le partage de fichiers, etc.

3.3 Le cas de WebSockets

Les WebSockets sont un protocole de communication bidirectionnel qui permet d'établir une connexion persistante entre un client et un serveur. Contrairement au protocole HTTP [10], qui est un protocole de communication unidirectionnel, les WebSockets permettent aux clients et aux serveurs de s'envoyer des données à tout moment, sans avoir à attendre une requête du client.

Les WebSockets sont souvent utilisés pour les applications qui nécessitent une communication bidirectionnelle en temps réel, telles que les applications de chat, les jeux en ligne, etc. Ils sont également utilisés pour les applications qui nécessitent une communication persistante entre le client et le serveur, telles que les applications de suivi en temps réel, les applications de surveillance, etc.

8. Transmission Control Protocol
9. Transport Layer Security
10. Hypertext Transfer Protocol

Les WebSockets sont un protocole de communication de haut niveau qui s'appuie sur le protocole TCP pour établir une connexion. Ils sont souvent utilisés avec des protocoles de haut niveau tels que HTTP, TLS, etc. pour établir une connexion sécurisée.

Nous avons ici pensé à utiliser les WebSockets pour établir une connexion entre les deux pairs. En effet, les sockets se divisent en deux parties, un serveur et un client. Le serveur est le programme qui attend les connexions des clients et le client est le programme qui se connecte au serveur. Dans notre cas, chaque pair peut être considéré comme un serveur et un client. En effet, chaque pair doit être capable d'accepter les connexions des autres pairs et de se connecter aux autres pairs. Nous avons donc tenté de créer un serveur WebSocket dans chaque pair et de connecter les deux serveurs entre eux.

Cependant, nous avons rencontré un problème lors de la mise en place de cette solution. En effet, comme évoqué plus tôt, le NAT empêche toute communication entrante vers un client. Cela signifie que le serveur WebSocket ne peut pas recevoir de connexion entrante d'un autre pair. Pour contourner ce problème, nous avons pensé à mettre ne place un tiers serveur qui permettrait de faire le lien entre les deux pairs. Chaque client via un serveur STUN pourrait récupérer son adresse IP publique et son port et les envoyer au serveur afin que par la suite, chaque client qui souhaite établir une connexion n'ai qu'à récupérer l'adresse IP publique et le port de l'autre client et se connecter à ce dernier.

Chapitre 4

Conclusion

En conclusion, ce document a souligné la problématique de l'établissement d'une connexion pair à pair en présence de dispositifs de traduction d'adresses réseau (NAT). Les NAT, bien qu'utiles pour partager une adresse IP publique et protéger les dispositifs internes, limitent la connectivité directe entre les clients.

Pour contourner ces limitations, il est nécessaire de recourir à des solutions telles que WebRTC et libp2p. WebRTC, conçu pour les communications en temps réel, et libp2p, une bibliothèque modulaire pour la communication pair à pair, offrent des mécanismes pour établir des connexions directes malgré la présence de dispositifs NAT.

En utilisant ces approches, les clients peuvent s'échanger les adresses multiadresses nécessaires pour établir la connexion et communiquer directement, sans dépendre exclusivement d'un tiers relai. Cependant, il est important de prendre en compte les besoins spécifiques de chaque application pour choisir la solution la mieux adaptée.

En explorant et en implémentant ces solutions, il devient possible de surmonter les limitations des dispositifs NAT et de faciliter l'établissement de connexions pair à pair. Cela ouvre la voie à une communication directe et efficace entre les clients, améliorant ainsi les possibilités de collaboration et d'échange de données dans divers scénarios d'application.