A Literature Review about Peer to Peer Protocol

Phi H. Nguyen
Earlham College
Richmond, Indiana
phnguyen17@earlham.edu

KEYWORDS

p2p, NAT hole-punching, decentralization

ACM Reference Format:
Phi H. Nguyen. 2020. A Literature Review about Peer to Peer Protocol. In *Proceedings of ACM Conference (Conference'17).* ACM, New York, NY, USA, 3 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

Napster(www.napster.com) was perhaps the most popular application that began the wave of Peer-to-peer applications in the early 2000s.[2] Built by Shawn Fanning, a freshman at Northeastern University, Napster allowed client to publish and download mp3 files directly from any other client. Its failure was inevitable because of the copyright infringement. However, it opens the possibility of communication directly via Peer-to-peer, without any third party regulation, without the worry of their activities being tracked, and their data being sniffed. A very important problem that Peer-to-peer protocol can solve really well is messaging on the internet: we could send a message directly to the recipient without any influence of a centralized server. Using a messaging service that stores our conversations inside a centralized server means that the people inside the conversation are not the only ones who know about it, there is a heavy concern over privacy in this method. Facebook, for example, shared 86 millions of user profiles with Cambridge Analytica without consent, used for political reasons since 2013 [5]. Google lost a lawsuit concerning them tracking their users in incognito mode, and you can actually join the victim team and have a chance to get 5000 dollar if you have used Google Chrome incognito any time since 2016. [8] Peer-to-peer is the ultimate decentralized solution for the issues of data bleaching, security and privacy of users.

2 CHALLENGES

There are numerous challenges in designing a Peer-to-peer application in today world - Our current Internet architecture is just not optimized for it. Although there are many challenges in Peer-to-peer network in dealing with file transfer, this section only focuses on the challenges that a peer-to-peer chat system may encounter.

Conference'17, July 2017, Washington, DC, USA

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

2.1 Missing data

Stutzbach et al. addresses the challenge of transferring data from one client directly to another client.[6] Since the connection rely on the bandwidth of the two users, which could differ significantly, packets can be missing during the transmission. There must be a way to handle missing data. This problem has been solved relatively well in a client-server structure, but not in a Peer-to-peer environment.

2.2 Long session

Stutzbach et al. also brings up the issue of handling long session between two clients in an unreliable network. During the transmission of exchanging packets, one client can have a better bandwidth than the other, which can affect the longevity of the connection.

2.3 Dynamic addressing

One thing that client-server architecture has been able to solve but not Peer-to-peer is dynamic addressing. In client-server architecture, the server will only sends data to client once requested, or a session has been established. Moreover, the address of the server is constant, hence, the client will always know where to request and always expect to get an answer. For Peer-to-peer, once a client address changes, due to dynamic addressing by DHCP and PPP, any peer that does not have this new address will not know where to look for.[6]

2.4 Network Address Translation

Perhaps this is the most discussed challenge in designing a Peer-to-peer application because of the complexity as well as popularity of the problem.

Network Address Translation(NAT) resides in majority of home routers/private networks. Its job is to block unwanted outside request to private network and only allows authorized ones. When we want to make a request to someone else, NAT record our local IP address and port number, then send out the request using the public IP address and a new port number that it assigns to our request. Then when receiving the data, NAT will know what local IP address/port number to send it to, since the public port has been opened when sending the message. Only request to opened port can be forwarded to the recipient.

This is the main reason why Peer-to-peer could not function well in the current Internet architecture. When a peer wants to send a message to another peer to initiate the conversation, it would not know what port is open on the other side, hence the message will be automatically dropped by the recipient's NAT.

Halkes et al claims that 75 percent machines on the internet are not connectable, which means there is a high chance that a peer that you wants to connect is behind a NAT.[3]

Not only bypassing NAT is difficult, there are also a variety of NATs in real world that have different configurations, designed by different brands. Some method can bypass some particular NAT models but not the others. [1]

2.5 Peer reputation

Before connecting two peers, we need to determine the reputation of each peer, whether it is a trustworthy client. Trustworthiness here can involve many things: low bandwidth, low capacity, low uptime, etc. [7]

3 METHODS

While there are various issues need to be solved in a peer-to-peer network, this section will focus on how to bypass Network Address Translation (NAT), since it is one of the first and also most important part to get a connection between two peers.

3.1 Universal Plug and Play

Developed by Microsoft, Universal Plug and Play(UPnP) NAT Traversal is a technique in Windows XP. When a new host needs a connection, a Windows XP machine configure the network addressing and broadcast its presence in a subnet, This machine can act as a NAT Gateway, or a control point, and detect whether it is behind a NAT. This NAT Traversal technique allows peer-to-peer applications to traverse the NAT Gateway by dynamic control of public ports. The main downside of this technique is insecurity, and only a subset of routers allow this technique to be used. [4]

3.2 Simple Traversal UDP Through Network Address Translators(STUN)

STUN is a lightweight protocol that lets devices behind NAT discover the type of NATs between them. A STUN client learns about port assignment of a NAT by sending a STUN request and expects to receive a STUN response. A STUN request is a Binding Request over UDP. This Binding Request would arrive in a STUN server that sits between the clients. Since the request could traverse through multiple NATs, it would carry the address of the last NAT device to the STUN server, which would send a STUN response that contains this address back to the client. By comparing the local address with the content of the response, a client can determine whether they are behind a NAT, and aware of the last NAT before it could reach another peer (assume being on a different subnet). [4] The downside of this is STUN does not work well with Symmetric NAT, which creats a binding based on source and destination IP address and port. This technique creates a new IP address after reaching the server and hence would fail the connection with other peers.[4] Moreover, STUN requires the application to be upgraded to support public STUN server, which makes this technique very hard to deploy. [4]

3.3 NAT hole-punching

Perhaps this is the most popular techniques to help a peer request to bypass a NAT. As mentioned in the NAT section above, before a request can reach the outside world, the private IP/port needs to be translated into a public IP/port. And any response/request to this public port at this public IP address will be forwarded to the client. Hence, this port has been opened and there is a "hole" in the NAT table. Hole punching means to open a port on the NAT so that we requests that want to reach us can bypass the NAT by going through that "hole" (or public port).

Generally, hole punching would need a relay server that sits between the clients to establish a connection. This technique can use the server to maintain the connection between them, or leaving it outside of the equation once the connection has been established.

Hu (2008) studies the architecture of BitTorrent and Skype and claimed that while a Peer-to-peer over TCP is possible, UDP is a more favored transport. The reason for this is very simple.[4] TCP needs a three-way handshake before they can actually send data, while UDP does not require a session to be established for the data to be sent. In an untrusted and unreliable environment of Peer-to-peer network, UDP is hence easier to manage and deploy. The architecture that Hu (2008) suggests also aligns with the architecture of a relay server, though the name was changed to "connection broker". This architecture solves the problem posed by STUN by using an additional trick: they use the same IP/port with the brokers and with the other peers, hence it does not create a new IP/port pair in Symmetric NAT.

Ford 2005 claimed that having a relay server to forward messages is generally a more robust technique.[1] Using a hybrid architecture, the network can still rely on the client-server method while eliminating the fear of their data being stored in the centralized server. The role of the relay server in this scenario is merely to forward message and nothing else. However appealing it may look, it is still a single point of failure architecture - the two clients need the server to be up all time to maintain a session - and the clients do not know if there data is being stored or not and whether or not they should trust this server.

Another method is to leave the relay server once the connection has been established. When A wants to connect with B, it sends a request to the response server. The data includes the public IP/port pair of A. The relay server will record A and its target client B in a table. When B connects to the server and requests to connect with A, the relay server forwards A's IP/public port address to B and B's ones to A. Now A and B knows each other opened port, hence can communicate with each other. One of the issues that might occur is that the port must be opened before the target client sends packets. One possible solution is to confirm the connection between A to server and B to server before connecting the two together.

4 CONCLUSION

In conclusion, data privacy is a significant issue in today's centralized architecture. And Peer-to-peer network is one of the solutions to decentralize the Internet, making each connection become autonomous and anonymous. While there are many challenges to this protocol due to the way our Internet is configured, there are also proposed solutions to solve. One of the biggest challenges is NAT, which blocks unwelcoming requests from outside world to our computer. While this is beneficial in terms of security, it also blocks Peer-to-peer transmission. The paper presented multiple ways to bypass a NAT and the most popular and robust method yet is Hole-punching with a relay server.

REFERENCES

[1] Bryan Ford, Pyda Srisuresh, and Dan Kegel. 2005. Peer-to-Peer Communication Across Network Address Translators.. In *USENIX Annual Technical Conference, General Track*. 179–192.

[2] Geoffrey Fox. 2001. Peer-to-peer networks. *Computing in Science & Engineering* 3, 3 (2001), 75–77.

[3] Gertjan Halkes and Johan Pouwelse. 2011. UDP NAT and Firewall Puncturing in the Wild. In *International Conference on Research in Networking*. Springer, 1–12.

[4] Zhou Hu. 2005. NAT traversal techniques and peer-to-peer applications. In *HUT T-110.551 Seminar on Internetworking*. Citeseer, 04–26.

[5] Jim Isaak and Mina J Hanna. 2018. User data privacy: Facebook, Cambridge Analytica, and privacy protection. *Computer* 51, 8 (2018), 56–59.

[6] Daniel Stutzbach and Reza Rejaie. 2006. Understanding churn in peer-to-peer networks. In *Proceedings of the 6th ACM SIGCOMM conference on Internet measurement*. 189–202.

[7] Yao Wang and Julita Vassileva. 2003. Trust and reputation model in peer-to-peer networks. In *Proceedings Third International Conference on Peer-to-Peer Computing (P2P2003)*. IEEE, 150–157.

[8] Davey Winder. 2020. Google Chrome Privacy Lawsuit: Could You Get A 5,000 Payout? https://www.forbes.com/sites/daveywinder/2020/06/03/google-chrome-privacy-lawsuit-could-you-get-a-5000-payout-incognito-mode-class-action/