UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ
DEPARTAMENTO ACADÊMICO DE INFORMÁTICA
CURSO DE ENGENHARIA DE COMPUTAÇÃO

HAMILTON ROTH FILHO

SISTEMA P2P TRANSPARENTE PARA TRANSFERÊNCIA DE ARQUIVOS UTILIZANDO UDP HOLE PUNCHING

TRABALHO DE CONCLUSÃO DE CURSO

PATO BRANCO
2017

HAMILTON ROTH FILHO

SISTEMA P2P TRANSPARENTE PARA TRANSFERÊNCIA DE ARQUIVOS UTILIZANDO UDP HOLE PUNCHING

Trabalho de Conclusão de Curso como requisito parcial à obtenção do título de Bacharel em Engenharia de Computação, do Departamento Acadêmico de Informática da Universidade Tecnológica Federal do Paraná.

Orientador: Prof. Dr. Bruno César Ribas

PATO BRANCO
2017

Ministério da Educação
Universidade Tecnológica Federal do Paraná
Câmpus Pato Branco
Departamento Acadêmico de Informática
Curso de Engenharia de Computação

UTFPR
UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ

TERMO DE APROVAÇÃO

Às 8 horas e 30 minutos do dia 05 de dezembro de 2017, na sala V007, da Universidade Tecnológica Federal do Paraná, Câmpus Pato Branco, reuniu-se a banca examinadora composta pelos professores Bruno Cesar Ribas (orientador), Fábio Favarim e Luciene de Oliveira Marin para avaliar o trabalho de conclusão de curso com o título **Sistema P2P Transparente para Transferência de Arquivos Utilizando UDP Hole Punching**, do aluno **Hamilton Roth Filho**, matrícula 01171879, do curso de Engenharia de Computação. Após a apresentação o candidato foi arguido pela banca examinadora. Em seguida foi realizada a deliberação pela banca examinadora que considerou o trabalho aprovado.

_____
Bruno Cesar Ribas
Orientador (UTFPR)

_____
Fábio Favarim
(UTFPR)

_____
Luciene de Oliveira Marin
(UTFPR)

_____
Profa. Beatriz Terezinha Borsoi
Coordenador de TCC

_____
Prof. Pablo Gauterio Cavalcanti
Coordenador do Curso de
Engenharia de Computação

A Folha de Aprovação assinada encontra-se na Coordenação do Curso.

RESUMO

Roth Filho, Hamilton. Sistema p2p transparente para transferência de arquivos utilizando UDP Hole Punching. 2017. 41f. Trabalho de Conclusão de Curso de bacharelado em Engenharia de Computação - Universidade Tecnológica Federal do Paraná. Pato Branco, 2017.

Com o grande aumento no número de usuários na Internet, diversas ferramentas passaram a ser utilizadas para garantir a segurança das conexões entre outras adversidades devido o grande número de usuários e dados presentes na rede. Entretanto certas ferramentas e aplicações acabaram diminuindo a transparêcia da Internet na comunicação entre dispositivos. Este trabalho busca contornar a perda de transparência nas comunicações utilizando do método UDP Hole Punching, com o desenvolvimento de uma aplicação com suporte para a transmissão de arquivos sem a objeção de NATs e Firewalls.

Palavras-chaves: UDP Hole Punching. Transmissão. Arquivos. NAT. Firewall. P2P.

ABSTRACT

Roth Filho, Hamilton. Sistema p2p transparente para transferência de arquivos utilizando UDP Hole Punching. 2017. 41f. Trabalho de Conclusão de Curso de bacharelado em Engenharia de Computação - Universidade Tecnológica Federal do Paraná. Pato Branco, 2017.

With the great rise in numbers of users in the Internet, a lot of tools started to be used to guarantee secure connections and deal with other problems caused by the great number of users and data in the network. Although some tools and applications caused a loss of transparency on the Internet regarding the communication peer-to-peer. This works intends to workaround the loss of transparency in communications with UDP Hole Punching, developing an application that supports file transfer without the intervation of NATs and Firewalls.

Key-words: sdi. distribuited execution. docker. virtualization. P2P.

LISTA DE FIGURAS

Figura 1 – Pacote TCP 12
Figura 2 – Aperto de mão em três vias 13
Figura 3 – Pacote UDP 13
Figura 4 – Imagem do NAT 15
Figura 5 – Full Cone 16
Figura 6 – Restricted Cone 17
Figura 7 – Port Restricted Cone 17
Figura 8 – Symmetric 18
Figura 9 – Dispositivos atrás do mesmo NAT 21
Figura 10 – Dispositivos atrás de diferentes NAT 22
Figura 11 – Dispositivos atrás de múltiplos níveis de NAT 23
Figura 12 – Procedimento para se dar inicio a comunicação 26
Figura 13 – Exemplo da transmissão de arquivos com perda de pacote 29
Figura 14 – Cenário 1 31
Figura 15 – Cenário 2 32
Figura 16 – Cenário 3 33
Figura 17 – Cenário 4 36

LISTA DE LISTAGENS

Listagem 4.1 – Struct dos clientes 26
Listagem 4.2 – Log do sistema no Cliente A 27
Listagem 5.1 – traceroute entre Hamilton-PC e Jaguapitanga, Cenário 3 34
Listagem 5.2 – traceroute entre Jaguapitanga e Hamilton-PC, Cenário 3 35
Listagem 5.3 – traceroute entre Hamilton-PC e Jaguapitanga, Cenário 4 36
Listagem 5.4 – traceroute entre Jaguapitanga e Hamilton-PC, Cenário 4 37

LISTA DE TABELAS

Tabela 1 – Transferência entre máquinas no mesmo laboratório 32
Tabela 2 – Primeiro teste de desempenho 35
Tabela 3 – Segundo teste de desempenho 37
Tabela 4 – Terceiro teste de desempenho 37

LISTA DE ABREVIATURAS E SIGLAS

NAT	*Network Address Translator*
TCP	*Transmission Control Protocol*
SYN	*Synchronize*
ACK	*Acknowledgment*
SYN-ACK	*Synchronize-Acknowledgment*
UDP	*User Datagram Protocol*
STUN	*Simple Traversal of UDP Through NATs*
VPN	*Virtual Private Network*
VoIP	*Voice Over Internet Protocol*
MTU	*Maximum Transmission Unit*
VPS	*Virtual Private Server*
UDT	*UDP-based Data Transfer*

SUMÁRIO

1 INTRODUÇÃO **9**
1.1 CONSIDERAÇÕES INICIAIS 10
1.2 OBJETIVOS 10
1.3 OBJETIVOS ESPECíFICOS 10
1.4 JUSTIFICATIVA 10
2 FUNDAMENTAÇÃO TEÓRICA **12**
2.1 TRANSMISSION CONTROL PROTOCOL 12
2.2 USER DATAGRAM PROTOCOL 13
2.3 FIREWALL 14
2.4 NETWORK ADDRESS TRANSLATOR 14
2.4.1 CLASSIFICAÇÃO 16
2.4.2 VANTAGENS E DESVANTAGENS 18
3 UDP HOLE PUNCHING **20**
3.1 SIMPLE TRAVERSAL OF UDP THROUGH NATS 20
3.2 ESTABELECENDO A CONEXÃO FIM-A-FIM 21
3.2.1 DISPOSITIVOS ATRÁS DO MESMO NAT 21
3.2.2 DISPOSITIVOS ATRÁS DE DIFERENTES NAT 22
3.2.3 DISPOSITIVOS ATRÁS DE MÚLTIPLOS NÍVEIS DE NAT 22
3.2.4 APLICAÇÕES EXISTENTES 23
4 DESENVOLVIMENTO **25**
4.1 EXECUÇÃO 25
4.1.1 TRANSMISSÃO DE ARQUIVOS 27
5 EXPERIMENTOS **30**
5.1 CENÁRIO 1 30
5.2 CENÁRIO 2 32
5.3 CENÁRIO 3 33
5.4 CENÁRIO 4 35
5.5 CENÁRIO 5 37
6 CONCLUSÃO **39**

1 INTRODUÇÃO

No início a Internet era uma rede de acesso restrito com poucos usuários, entretanto seu grande crescimento nas últimas décadas, fez com que a Internet alcançasse em torno de 40% da população mundial (PROJECT, 2016), com bilhões de acessos diários. Garantir a segurança nas conexões tornou-se uma necessidade, o que eventualmente causou mudanças na maneira como são feitas as conexões.

A transparência da Internet é um conceito que define a capacidade de um serviço ter acesso a outro através da rede utilizando seu sistema de endereçamento único, tanto quanto sua capacidade de enviar e receber pacotes sem a intervenção de um serviço intermediário (CARPENTER, 2000).

Originalmente, o protocolo da Internet tinha como proposta a transmissão de dados por pacotes entre dispositivos finais por meio de uma rede, onde estes dispositivos eram identificados por endereços únicos de tamanho fixo (POSTEL, 1981). Esses princípios garantiam a transparência da Internet.

Tratando-se de conexões fim-a-fim, ao que se refere à comunicação entre dispositivos finais, o conceito de transparência se refere ao sistema de endereçamento proposto pelo protocolo da Internet em que cada dispositivo tem um endereço de IP único e também garante que um pacote transmitido pela rede não seja alterado no seu canal de comunicação, com exceção de práticas que não comprometem a transparência como a fragmentação e compressão de pacotes, já que essas alterações são reversíveis quando chegam ao destino (BORGES, 2008),(CARPENTER, 2000).

Todavia, para o funcionamento ideal da transparência da Internet, se requer uma conduta extremamente ética e colaborativa por parte de todos os usuários da rede, objetivo este que poderia ser atingido com um número limitado de usuários. Entretanto considerando o grande número de usuários da Internet, tal conduta seria uma utopia. Tal liberdade inexoravelmente possibilitou diversos acessos indesejados em determinadas redes (BELLOVIN, 1992).

Com a crescente necessidade de proteger as redes de acessos indesejados e limitar o fluxo de determinados dados para fora de uma rede, passou-se a utilizar sistemas como o Network Address Translator (NAT) e o firewall que entre suas funções tem como objetivo tornar as redes mais seguras (BELLOVIN, 1992).

Porém, não apenas o conceito de transparência, mas a conectividade fim-a-fim, que se define pela capacidade de um dispositivo enviar livremente pacotes através da rede para outro dispositivo sem bloqueio desde que o endereço do destinatário seja conhecido, vem sendo afetada devido ao emprego dos sistemas de NAT e firewall (CARPENTER, 2000).

Devido ao amplo uso de aplicações, como NAT e *firewall*, restaurar a transparência da Internet como, inicialmente proposto no protocolo da Internet, não é uma opção válida (POSTEL, 1981). Entretanto é possível amenizar as consequências desta perda de transparência uti-

lizando de técnicas como a que é apresentada neste trabalho.

1.1 CONSIDERAÇÕES INICIAIS

Aplicações como *Network Address Translator* (NAT) e *firewall* atuam como intermediários entre a conexão de dois dispositivos conectados a uma rede, este princípio causa não só a perda de transparência na conexão, como também pode impedi-la por completo. Desta forma para permitir que dois ou mais dispositivos que utilizem dessas aplicações por motivos de segurança ou para outros fins, possam se comunicar, certas técnicas podem ser aplicadas.

Neste trabalho daremos foco ao UDP Hole Punching, técnica usada para estabelecer conexões entre dispositivos que utilizam de aplicações intermediárias, através do *User Datagram Protocol* (UDP).

1.2 OBJETIVOS

O objetivo geral deste trabalho é implementar um sistema capaz de realizar a técnica de UDP Hole Punching entre vários clientes. O sistema poderá ser utilizado por qualquer aplicação que suporte o protocolo UDP para sua comunicação.

1.3 OBJETIVOS ESPECÍFICOS

- Implementar o reenvio de pacotes UDP por meio da aplicação.
- Utilizar o sistema para executar um problema de teste.

1.4 JUSTIFICATIVA

Considerando a necessidade intrínseca de diversas aplicações em estabelecer conexões entre dispositivos e o deliberado uso de aplicações intermediárias entre estes dispositivos que podem comprometer sua capacidade de estabelecer tais conexões. Se torna necessário a implementação de técnicas que garantam a capacidade de comunicação de dispositivos como proposto no protocolo da Internet (POSTEL, 1981).

Este trabalho está dividido em capítulos. No Capítulo 2, é mostrado o referencial teórico que serve de base para este trabalho. No Capítulo 3, é apresentado o funcionamento do UDP Hole

Punching. No Capítulo 4 será apresentada o desenvolvimento do trabalho. E no capítulo 5 será apresentada a conclusão.

2 FUNDAMENTAÇÃO TEÓRICA

Neste capítulo são apresentados os conceitos teóricos fundamentais para o entendimento da aplicação proposta nesse trabalho.

2.1 TRANSMISSION CONTROL PROTOCOL

Transmission Control Protocol (TCP) é um protocolo orientado a conexão fim-a-fim projetado para ser utilizado na hierarquia de camadas. Este protocolo permite a transmissão de dados continua em qualquer direção, assim como a confiabilidade do fluxo de dados (POSTEL, 1980a).

Durante a transmissão de dados, devido a falhas na conexão, certos pacotes podem ser danificados, perdidos, duplicados ou até transmitidos fora de ordem. Para garantir a confiabilidade na transmissão de dados, o protocolo TCP atribui números sequenciais para os pacotes e espera do receptor uma mensagem de recebimento, se tal mensagem não for recebida durante um intervalo de tempo, o protocolo faz a requisição para o reenvio dos dados, no receptor utilizando os números sequenciais atribuídos aos pacotes, estes podem ser ordenados e eliminados em caso de duplicação. Para garantir que os pacotes recebidos não estejam danificados é feito a verificação com um *checksum*, o cabeçalho do protocolo TCP pode ser visto na Figura 1.

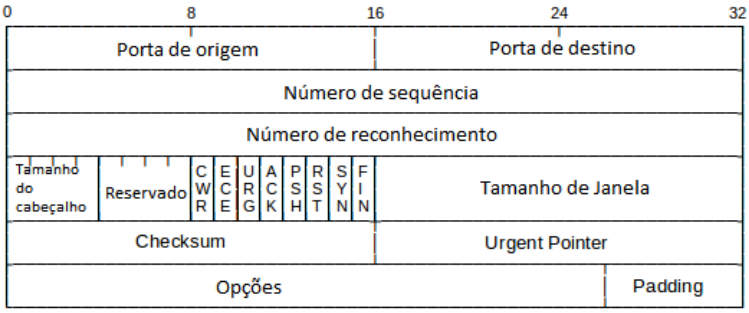

Figura 1 – Pacote TCP
Fonte: Adaptado de(EDDY, 2006)

Para garantir confiabilidade, o protocolo TCP mantém a conectividade entre os dispositivos, o principio básico que mantém a conexão deste protocolo é o aperto de mão em três vias, *three way handshake*. De maneira simplificada um *three way handshake* entre um dispositivo A e B funciona da seguinte maneira: A envia um pacote *synchronize* (SYN) para indicar o início da sequência numérica na conexão com B; B recebe SYN e envia um pacote *synchronize-acknowledgment* (SYN-ACK) para informar que recebeu o pacote SYN; quando A recebe SYN-ACK, ela envia um pacote *acknowledgment* (ACK) para B para informar a B que recebeu o pacote SYN-ACK (POSTEL, 1980a).

O processo do aperto de mão em três vias descrito anteriormente é ilustrado na Figura 2.

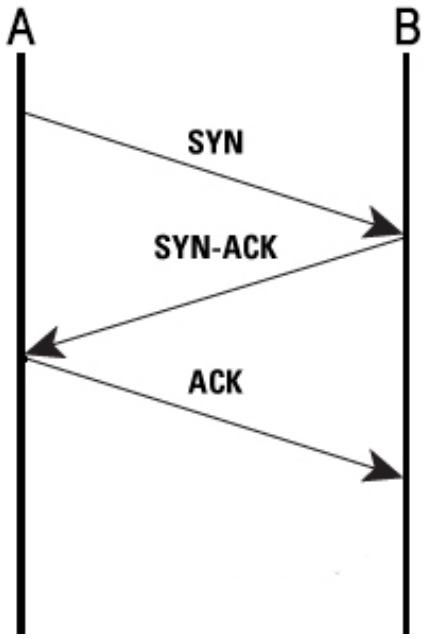

Figura 2 – Aperto de mão em três vias
Fonte: Adaptado de(EDDY, 2006)

2.2 USER DATAGRAM PROTOCOL

O UDP ao contrário do TCP é considerado um protocolo de transmissão de baixa confiabilidade porque ele não tem nenhuma ferramenta que garanta a chegada dos pacotes que foram enviados, todavia este protocolo tem seu uso em outras aplicações.

Aplicações de voz sobre IP, ((VoIP) - *Voice over Internet Protocol*) ou Transmissões em tempo real (*livestreams*), em que a transmissão de dados é contínua e caso ocorra uma falha na conexão, não há necessidade de recuperar os dados perdidos durante a falha por que isso causaria um atraso na transmissão que deixaria de ser em tempo real ou vários pacotes seriam recebidos no mesmo instante e não seriam processados corretamente. A falta de confiabilidade deste protocolo é compensada pelo fato que ao não fazer as verificações, uma transmissão UDP enfrenta menos problemas com latência que seriam os problemas mais críticos em aplicações VoIP ou *livestreams*(POSTEL, 1980b).

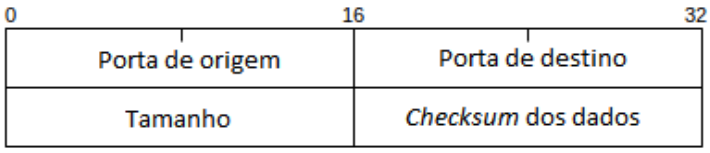

Figura 3 – Pacote UDP
Fonte: Adaptado de(EDDY, 2006)

2.3 FIREWALL

Firewall é um sistema criado com a intenção de interceptar dados que trafegam entre uma intranet e a Internet, autorizando apenas o fluxo de dados selecionados que pertencem a um número limitado de aplicações(CARPENTER, 2000).

Entre as diversas maneiras de proteger uma rede, uma abordagem é a garantia de segurança de cada sistema operacional, de cada dispositivo que pertence a rede, porém apesar desta abordagem supostamente criar uma rede segura, esta tarefa se torna custosa e complexa.

Devido a este fator o emprego do firewall apresenta uma solução mais viável à segurança de uma rede já que ela protege a rede como uma única entidade e não cada dispositivo separadamente, seguindo o princípio que todo tráfego através das redes deve passar pelo firewall e ele através de sua política interna, decidir quais pacotes estão autorizados a entrar ou sair da rede.

Entretanto, devido a atuação do firewall ser na extremidade das redes e sua proteção de todos dispositivos como uma única entidade, a violação de um dos dispositivos compromete toda a segurança da rede.

Os tipos mais comuns de firewall são aqueles baseados em filtragem de pacotes e os que atuam na camada de aplicação. Firewall de filtragem de pacotes atua na camada de transporte e rede analisando o cabeçalho de cada pacote adquirindo informações como o endereço de IP dos dispositivos de origem e destino do pacote e suas respectivas portas, que juntamente com um conjunto de regras previamente definido pela administrador avalia se o pacote irá passar pelo firewall. Já os sistemas que atuam na camada de aplicação examinam os dados em todas as camadas. Estes sistemas tem a capacidade de analisar o conteúdo da mensagem, por isso são considerados mais seguros, entretanto tem complexidade superior em sua implementação devido ao número de protocolos específicos que o firewall deve suportar (BORGES, 2008)(ROMANOFSKI, 2002).

Devido a seu princípio básico de atuação, que se define em bloquear a comunicação entre duas redes, o firewall afeta diretamente a transparência fim-a-fim, já que no caso do firewall de filtragem de pacotes os dispositivos internos a rede, protegidos pelo sistema, podem enviar pacotes a dispositivos externos, porém impedem o contrário. Em um caso em que ambos dispositivos estão protegidos por um firewall, seria impossível estabelecer uma conexão mesmo com o consentimento de ambos os usuários.

2.4 NETWORK ADDRESS TRANSLATOR

Uma intranet é um conceito de uma rede privada doméstica ou corporativa utilizada para o uso de recursos e compartilhamento de dados entre dispositivos sem que as informações

utilizadas sejam acessíveis por outra rede, como a Internet. Com a eminente escassez de endereços IPv4 (HAIN, 2005), passou-se a utilizar os conceitos de endereços públicos, que fazem parte do sistema de endereçamento único global e endereços privados que são únicos apenas dentro de uma determinada rede (REKHTER B. MOSKOWITZ, 1994). A grande maioria de intranets passaram a utilizar endereços privados para seus dispositivos, entretanto há casos em que dispositivos da intranet necessitam de acesso a rede de Internet, nestes casos é empregado um sistema NAT.

NAT é um sistema capaz de mapear inteiramente uma rede intranet com um único endereço público, possibilitando a interligação de redes com domínios diferentes, sendo assim responsável pela troca de pacotes entre os dispositivos internos e dispositivos conectados pela Internet. Seu método de funcionamento se baseia em duas interfaces, uma delas conectada a intranet e outra a Internet, fazendo a tradução de endereços públicos e privados entre as redes (EGEVANG, 1994).

Apesar da eficiência dos sistemas NAT em atrasar de maneira significante a escassez de endereços IPv4, seu emprego causa grandes problemas referentes a transparência, já que os dispositivos internos a rede tem endereços privados não roteáveis, prejudicando também a integridade das comunicações devido ao fato que as verificações utilizadas por sistemas de segurança falham em função da tradução de IPs inerente do NAT (BORGES, 2008).

Este princípio básico de funcionamento acarreta consequências para a transparência fim-a-fim, já que bloqueia a conexão UDP ou TCP/IP entre dispositivos externos a rede com dispositivos internos a rede.

Em contrapartida, a migração para o endereçamento IPv6 deve extinguir o uso de NATs, o que traria necessidade das redes que utilizam o sistema adotarem outro sistema de segurança como o firewall que causaria as mesmas consequências na transparência da Internet (DEERING, 1998).

A Figura 4 ilustra como é feito o mapeamento dos dispositivos da intranet pelo NAT.

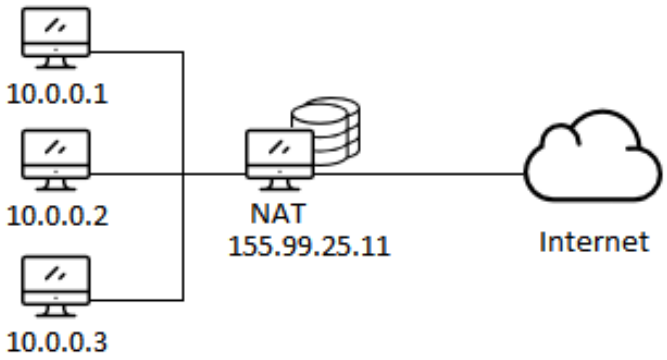

Figura 4 – Imagem do NAT
Fonte: Autoria própria

2.4.1 CLASSIFICAÇÃO

Para o desenvolvimento do trabalho é importante notar que diferentes implementações de NAT tratam o protocolo UDP de diferentes maneiras, como *Full Cone*, *Restricted Cone*, *Port Restricted Cone* e *Symmetric*(ROSENBERG J. WEINBERGER, 2003). Os métodos serão descritos a seguir.

Full Cone

Em um NAT *Full Cone*, todas requisições vindas de um mesmo endereço interno e porta interna, são mapeados para o mesmo endereço externo e porta externa, dessa maneira qualquer aplicação externa que deseja enviar pacotes a este endereço interno, usará o endereço/porta externo equivalente. Este procedimento está ilustrado na Figura 5.

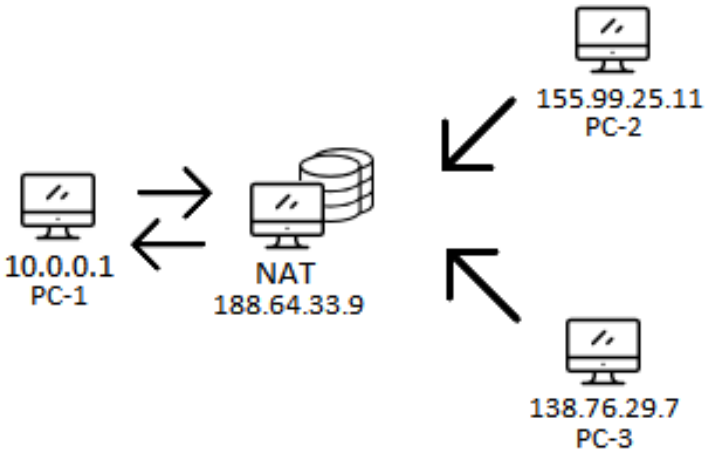

Figura 5 – Full Cone
Fonte: Autoria própria

A Figura 5 representa um ambiente em que o NAT de PC-1 aceita pacotes de PC-2 e PC-3 no mesmo endereço de IP e número de porta, mesmo que PC-1 tenha iniciado a comunicação somente com PC-2.

Restricted Cone

O NAT *Restricted Cone* funciona de maneira semelhante a um NAT *Full Cone*, mapeando as requisições de um mesmo endereço interno e porta interna para o mesmo endereço externo e porta externa, todavia para uma aplicação externa enviar pacotes a este endereço interno, o endereço interno deve previamente enviar um pacote para tal aplicação. Este procedimento está ilustrado na Figura 6.

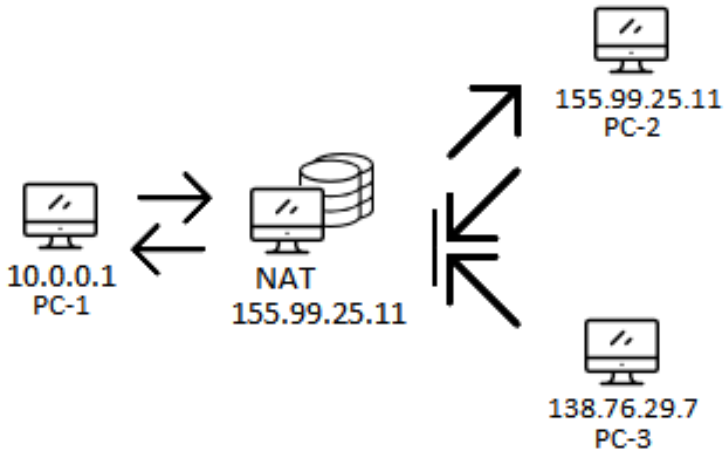

Figura 6 – Restricted Cone
Fonte: Autoria própria

A Figura 6 representa um ambiente em que o NAT de PC-1 aceita pacotes de PC-2 após PC-1 ter iniciado tal conexão, todavia se PC-3 tentar enviar pacotes para o endereço de IP e número de porta mapeados para receber os pacotes de PC-2, estes pacotes serão bloqueados pelo NAT.

Port Restricted Cone

Um NAT *Port Restricted* Cone atua como um NAT *Restricted Cone* com a adição da restrição de porta, ou seja, uma aplicação externa pode enviar pacotes para o endereço interno apenas pelo endereço e porta pelos quais o endereço interno iniciou a troca de pacotes. Este procedimento está ilustrado na Figura 7.

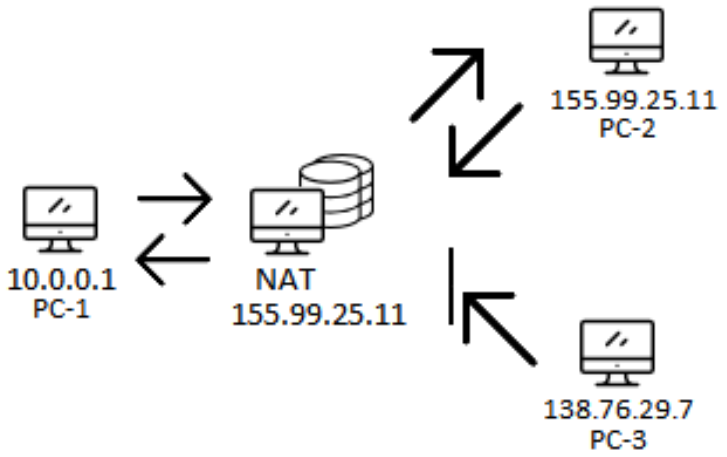

Figura 7 – Port Restricted Cone
Fonte: Autoria própria

A Figura 7 representa um ambiente em que mesmo PC-1 tendo iniciado a comunicação com PC-2, quando PC-2 tenta enviar pacotes para PC-1 em um número de porta diferente, o NAT de PC-1 bloqueia estes pacotes.

Symmetric

Em um NAT *Symmetric*, todas as requisições vindas do mesmo endereço interno e porta interna com destino a um mesmo endereço externo e porta externa são mapeadas para uma mesmo endereço e porta externa, porém se uma requisição é feita do mesmo endereço interno e porta interna com destino a um endereço e porta externos diferentes, outro mapeamento é realizado. Dessa forma apenas o dispositivo que receber o pacote do endereço interno pode enviar pacotes para este endereço interno. É interessante notar que o NAT *symmetric* aplica as mesmas restrições que o *Port Restricted Cone*, todavia sua maneira de mapeamento é diferente. Este procedimento está ilustrado na Figura 8.

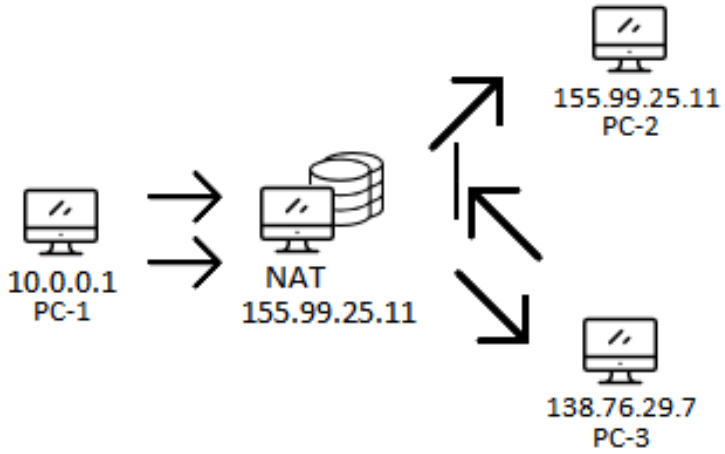

Figura 8 – Symmetric
Fonte: Autoria própria

A Figura 8 representa o mesmo ambiente da Figura 7, já que a restrição dos NATs *Symmetric* e *Port Restricted Cone* são as mesmas, com formas de mapeamento diferente.

2.4.2 VANTAGENS E DESVANTAGENS

Nesta subseção são listadas as vantagens e desvantagens do uso de NATs:

Vantagens:

- Eficaz em reduzir o esgotamento de endereços IPv4, devido sua capacidade de mapear diversos dispositivos com apenas um endereço na Internet;
- Aplicação simples e de menor custo a tentar garantir a segurança de cada dispositivo de maneira individual;
- Transparente para aplicações que não se conectam com dispositivos externos, já que dentro da mesma intranet os dispositivos tem acesso direito pelo IP privado;

Desvantagens:

- Os endereços privados da intranet não são diretamente endereçáveis para dispositivos na Internet, diminui a transparência devido a tradução de endereços do NAT;
- A alteração no cabeçalho dos pacotes afeta a utilização de sistemas de segurança de pacotes;
- O NAT pode ter alto consumo de memória para armazenar os endereços e sessões de cada comunicação.

É possível observar que devido ao crescimento no número de usuários e comunicações, aplicações que buscam garantir a segurança destas comunicações se tornaram um requerimento, todavia a adição destes meios intermediários entre a comunicação de dispositivos vai contra o principio de transparência estabelecido no protocolo da Internet.

3 UDP HOLE PUNCHING

Devido ao crescimento do uso de firewall e NAT a transparência da Internet tem sido afetada e como consequência quando dois dispositivos que estão protegidos por NAT e firewall precisam estabelecer uma conexão ponto a ponto, o que dificulta o funcionamento de aplicações fundamentadas na comunicação entre dispositivos finais nas extremidades da Internet (BORGES, 2008). A técnica estudada e posteriormente aplicada nesse trabalho procura amenizar essas dificuldades na conectividade em aplicações como VoIP Schmitd (2006).

UDP Hole Punching é um método que possibilita que dois dispositivos estabeleçam uma conexão UDP ponto a ponto, mesmo ambos dispositivos estando atrás de firewall ou NATs com o auxílio de um terceiro dispositivo com endereço conhecido (BORGES, 2008). Neste trabalho este terceiro dispositivo será chamado de rendezvous. Para ser dar início ao método do UDP Hole Punching se assume que ambos os dispositivos que desejam estabelecer uma conexão tem conhecimento do endereço de um servidor rendezvous em comum. Um servidor rendezvous é um dispositivo com endereço conhecido pelos outros dispositivos que desejam fazer a conexão (FORD PYDA SRISURESH, 2008). O servidor armazena dois pontos para cada cliente, sendo deles seu endereço privado e público.

O UDP Hole Punching é uma técnica aplicada a NATs do tipo *Cone* (*Full Cone, Restricted Cone* e *Port Restricted Cone*), todavia ele pode também ser aplicado em NATs *symmetric* com o auxílio de técnicas de predição de portas.

3.1 SIMPLE TRAVERSAL OF UDP THROUGH NATS

O Simple Traversal of UDP Through NATs (STUN) é um protocolo que utiliza o princípio de requisição/resposta para determinar a presença de um NAT ou firewall entre um dispositivo e a Internet. Um dispositivo que deseja descobrir a presença de um NAT ou firewall envia um pacote para um servidor STUN que responde com o endereço de IP e porta coletados do pacote de requisição, assim o dispositivo pode comparar o seu endereço com o endereço que o servidor STUN enviou como resposta, se eles forem diferentes, o dispositivo sabe que foi feita uma tradução. O STUN também permite a identificação do tipo de NAT presente.

O STUN é útil para aplicação do UDP Hole Punching devido ao fato que seu uso pode não só descobrir se o uso do UDP Hole Punching é realmente necessário como também se seu uso é possível, já que a técnica do UDP Hole Punching assume NATs do tipo *Cpone*.

3.2 ESTABELECENDO A CONEXÃO FIM-A-FIM

Assumindo que um cliente A e B tenham uma conexão UDP já estabelecida com um rendezvous S, quando um dos clientes deseja se comunicar com o outro:

- Cliente A não sabe o endereço de B, logo ele faz uma requisição do endereço de B para S.
- S informa para A o endereço público e privado de B e ao mesmo tempo informa B do endereço público e privado de A.
- A então tenta estabelecer conexão com ambos endereços de B e assim que a primeira conexão é realizada, B faz o mesmo procedimento com os endereços de A

A seguir são apresentadas possíveis configurações para os clientes A e B e é feita a análise do comportamento da técnica de UDP Hole Punching.

3.2.1 DISPOSITIVOS ATRÁS DO MESMO NAT

Supondo uma configuração de dispositivos em que um cliente A e um cliente B estão atrás do mesmo NAT, portanto, eles estão na mesma intranet. Assim como no procedimento previamente explicado, A informa S que deseja estabelecer uma conexão com B, em seguida S informa A o endereço público e privado de B e também informa B o endereço público e privado de A, assim ambos dispositivos tentam estabelecer a conexão diretamente assim como ilustrado na Figura 9(FORD PYDA SRISURESH, 2005).

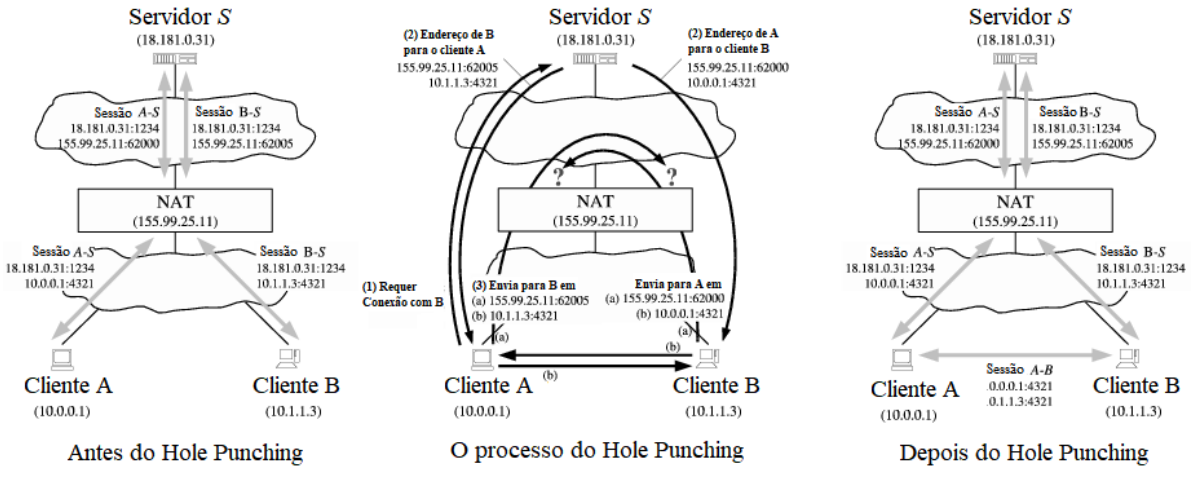

Figura 9 – Dispositivos atrás do mesmo NAT
Fonte: Ford, Srisuresh e Kegel (2005, p.182).

3.2.2 DISPOSITIVOS ATRÁS DE DIFERENTES NAT

Seguindo o procedimento inicial do UDP Hole Punching, o cliente A faz a requisição a S de que deseja se comunicar com B, porém, assim como ilustrado na Figura 10, o cliente A e o cliente B estão localizados em diferentes intranets, logo seus endereços privados não são roteáveis, como consequência a mensagem de A para o endereço privado de B irá chegar a algum dispositivo dentro da intranet de A ou até mesmo a dispositivo algum. Para evitar estabelecer uma conexão com um cliente indesejado é necessário que a aplicação seja identificada por um nome específico ou outra informação.

Quando o cliente A envia uma mensagem ao endereço público de B, o NAT de A verifica que o endereço de origem da mensagem é o mesmo da comunicação com o rendezvous S, porém como seu endereço de destino é diferente, outra sessão UDP é iniciada, dessa forma criando uma via de comunicação através do NAT de A.

Se a mensagem de A chegar ao NAT de B antes que a mensagem de B para A passe pelo NAT de B, a mensagem de A será descartada, todavia após as mensagens de A e B passarem por seus respectivos NATs, se tem vias através de ambos os NATs e a comunicação pode prosseguir como ilustrado na Figura 10(FORD PYDA SRISURESH, 2005).

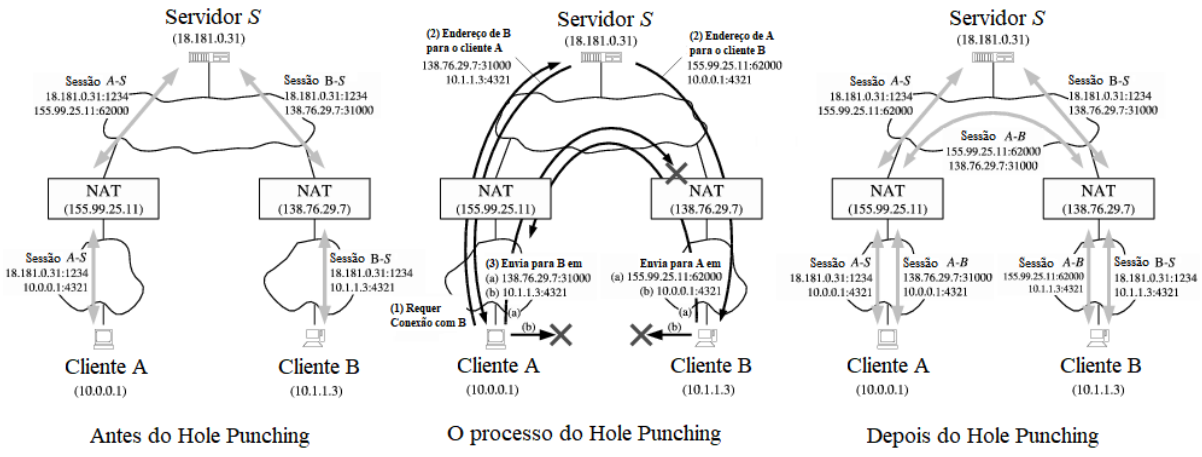

Figura 10 – Dispositivos atrás de diferentes NAT
Fonte: Ford, Srisuresh e Kegel (2005, p.183).

3.2.3 DISPOSITIVOS ATRÁS DE MÚLTIPLOS NÍVEIS DE NAT

Em um caso em que o cliente A e B estão atrás de mais de um NAT como ilustrado na Figura 11, pode-se observar que o endereço público do NAT A e B na verdade é um endereço privado do NAT C, sendo assim apenas o NAT C e o rendezvous S tem endereços publicamente roteáveis.

A melhor maneira de se estabelecer uma conexão entre um cliente A e B seria utilizando

o endereço "intermediário"sem passar pelo nível do NAT C em um procedimento como o de dois dispositivos atrás do mesmo NAT.

Entretanto o servidor rendezvous S apenas tem conhecimento dos endereços públicos dos clientes, logo a conexão mencionada no parágrafo anterior não poderia ser estabelecida já que os clientes não tem conhecimento dos endereços dos dispositivos na qual desejam se comunicar.

Para a conexão ser estabelecida, deve-se considerar a capacidade do NAT C de suportar NAT *reflection* na qual se refere a capacidade de dois dispositivos dentro da mesma intranet se comunicarem utilizando seu endereço público e não seu endereço real. Assumindo esse suporte como verdade, o procedimento segue como na seção 3.2.2(FORD PYDA SRISURESH, 2005). O procedimento completo está ilustrado na Figura 11.

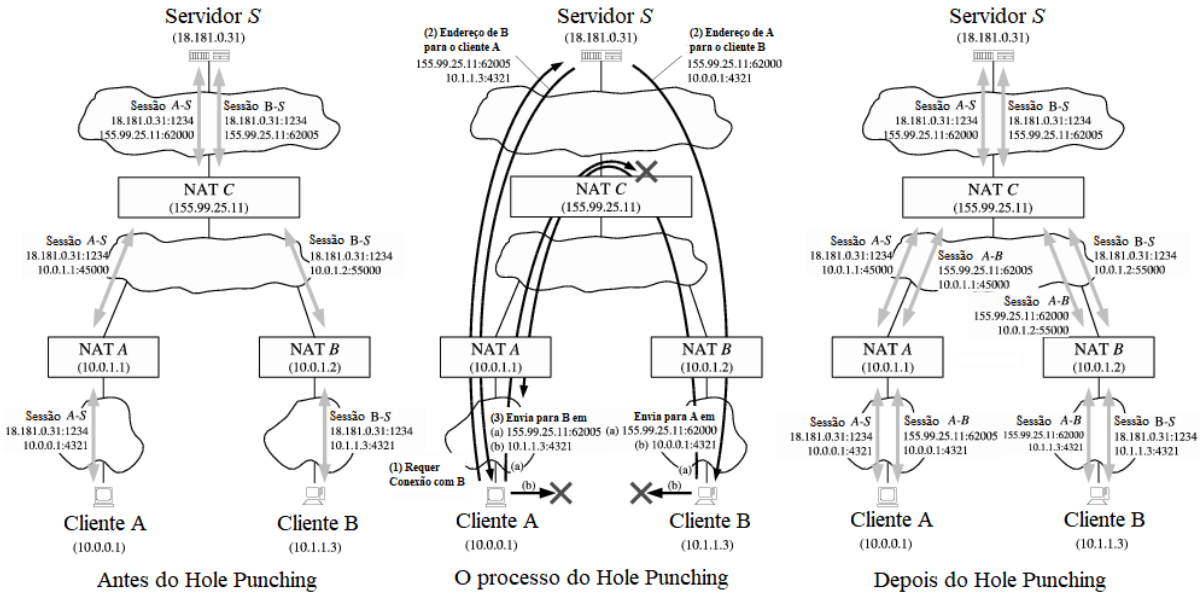

Figura 11 – Dispositivos atrás de múltiplos níveis de NAT
Fonte: Ford, Srisuresh e Kegel (2005, p.184).

3.2.4 APLICAÇÕES EXISTENTES

Diversas aplicações existentes utilizam da técnica de UDP Hole Punching para estabelecer suas conexões, como o Tinc (TIMMERMANS, 2016), Skype (SCHMITD, 2006), OpenVPN (DINHA, 2016) e o UDT (GU, 2012).

O tinc é uma rede virtual privada, *Virtual Private Network*(VPN), que utiliza de técnicas de tunelamento e encriptação para criar uma rede privada entre dispositivos na Internet. Tinc dá suporte ao UDP Hole Punching desde que o usuário da VPN forneça o servidor rendezvous.

O Skype é uma aplicação que permite estabelecer chamadas de áudio e vídeo entre dois ou mais dispositivos, assim como troca de arquivos e mensagem de textos. O Skype utiliza de UDP Hole Punching quando necessário para estabelecer suas conexões, todavia ele não

utiliza um servidor próprio como rendezvous e sim qualquer dispositivo com endereço publicamente roteável conectado ao Skype. Em casos nos quais o UDP Hole Punching não é possível, o Skype utiliza de *relaying*, porém esta técnica tem a desvantagem de utilizar processamento dos servidores do skype.

O OpenVPN é um recurso VPN de código aberto projetado pela OpenVPN Technologies. OpenVPN dá suporte a UDP Hole Punching.

O UDP-based Data Transfer (UDT) é uma aplicação para transferência de dados utilizando UDP com confiabilidade implementada a nível de aplicação, sua proposta é a transmissão rápida e eficiente de dados utilizando UDP de forma a ter menor latência que o TCP.

Neste capítuo foi apresentado que o UDP Hole Punching é uma técnica factível para realizar comunicações previamente bloqueadas para o uso de NATs e *firewalls* e também está presente em diversas aplicações, todavia é importante ressaltar que estas aplicações tem suporte para outras funções além de *VoIP* como transferência de arquivos ou mensagens de texto, funções que não tem a mesma tolerância para perda de pacotes como aplicações de tempo real.

4 DESENVOLVIMENTO

Neste capítulo são apresentados as etapas de desenvolvimento deste trabalho.

Antes de se iniciar o desenvolvimento da aplicação, foi necessário entender o funcionamento dos protocolos de comunicação como UDP e TCP, estudar o funcionamento de NATs e *firewalls*, já que estes são os principais responsáveis pela perda de transparência que nossa aplicação deseja tratar. Por fim o entedimento da técnica a ser implementada, UDP Hole Punching.

Ao planejar as etapas de desenvolvimento desta aplicação, de imediato notou-se uma desvantagem do uso de UDP Hole Punching que é o uso de pacotes UDP. Como mostrado anteriormente neste trabalho, o UDP tem como característica sua falta de confiabilidade, todavia a adaptação desta técnica para o protocolo TCP foi considerada inviável(FORD PYDA SRISURESH, 2005).

Logo, a abordagem escolhida para sanar os problemas inerentes do uso de pacotes UDP, foi a implementação de técnicas que garantem a confiabilidade da conexão, semelhantes aquelas presentes no TCP, mas ao nível de aplicação.

O desenvolvimento foi dividido nas etapas a seguir:

- **Comunicação com o Rendezvous:** Inicia a comunicação entre os clientes e o servidor Rendezvous que será o intermediário para iniciar a comunicação direta entre os clientes.
- **Comunicação entre clientes:** Os clientes se utilizam do Rendezvous para obter o endereço dos outros clientes no qual se deseja iniciar a comunicação.
- **Reenvio de Pacotes:** Implementação para garantir que caso ocorra perda de pacotes entre os clientes, estes pacotes devem ser reenviados
- **Reorganização:** A reorganização se refere a criação de um arquivo por parte de um cliente utilizando os dados recebidos de outro cliente.

Com estas etapas finalizadas o sistema está apto a realizar uma transferência de dados entre dois clientes que utilizam de NATs.

4.1 EXECUÇÃO

A primeira etapa a ser realizada foi iniciar a comunicação entre as partes de um cliente e servidor através de pacotes UDP. Inicialmente foram implementadas duas partes de um sistema: o cliente responsável por enviar uma mensagem de texto para a outra parte e o servidor que posteriormente foi usado como rendezvous, no qual ao receber a mensagem de texto do cliente, modifica os caracteres da mensagem para caixa alta e os reenvia para o cliente.

Foram feitas alterações para que o sistema funcione com mais clientes e o servidor passe a ser o Rendezvous responsável por ser o intermediário que irá auxiliar os clientes a iniciarem sua comunicação. A Listagem 4.1 apresenta os principais elementos da *struct* de cada cliente, sendo a varíavel *Address* e *Port* responsáveis por armazenar o endereço dos clientes e a variável ID para o número de identificação de cada cliente. As váriaveis usadas para a transmissão de dados são *data*, cks e cod, para armazenar respectivamente os dados, o checksum(MD5) e o número de sequência do pacote(cod).

```
typedef struct
{
        unsigned int address, cod;
        unsigned short port;
        unsigned char id[3], cks[50], data[BUFFER];
} client;
```

Listagem 4.1 – Struct dos clientes

Para a comunicação direta entre dois clientes é necessário que estes saibam o endereço de IP para qual enviar os pacotes. Tendo em vista o funcionamento de um NAT, se sabe que o cliente deve sempre iniciar a conexão para que possa obter uma resposta do rendezvous, desta forma a abordagem escolhida para que os pacotes do servidor sejam aceitos pelos NAT dos clientes, foi o envio da lista de clientes por parte servidor para cada cliente da lista após a adição de um novo cliente. Desta maneira, cada vez que um novo cliente se comunicar ao rendezvous, todos clientes terão suas listas atualizadas. Este procedimento está ilustrado na Figura 12.

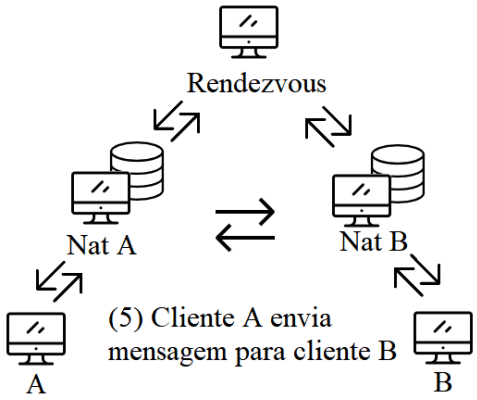

Figura 12 – Procedimento para se dar inicio a comunicação
Fonte: Autoria própria

Após os clientes terem conhecimento dos endereços de IP e portas utilizadas pelos outros clientes para se comunicar com o rendezvous, um cliente neste trabalho chamado de A abre um arquivo já existente e envia seus dados através de pacotes para outro cliente neste trabalho chamado de B. Ao receber os pacotes de A, o cliente B cria um arquivo com os dados

recebidos de A. Desta forma acontece o processo do *UDP Hole Punching*, já que sem o servidor rendezvous como intermediário os pacotes de A para B seriam bloqueados. A Listagem 4.2 ilustra o log do sistema, cliente A após o UDP Hole Punching.

```
Pacote recebido de: 18.181.0.31:1234 (Rendezvous)
Novo cliente[1]: 155.99.25.11:62000 (Cliente A)
Número de clientes conectados: 1
Pacote recebido de: 18.181.0.31:1234 (Rendezvous)
Novo cliente[2]: 138.76.29.7:31000 (Cliente B)
Número de clientes conectados: 2
Pacote recebido de: 138.76.29.7:31000 (Cliente B)
```

Listagem 4.2 – Log do sistema no Cliente A

4.1.1 TRANSMISSÃO DE ARQUIVOS

O grande desafio da transmissão de arquivos utilizando o UDP hole punching é claramente as limitações do próprio protocolo UDP. No capítulo 3 deste trabalho foram apontados diversos aspectos que tornam o protocolo UDP não confiável, todavia em casos onde o TCP não é uma opção ou quando se deseja ter um nível razoável de confiabilidade com uma latência menor, implementar confiabilidade ao UDP se torna uma estratégia interessante.

De maneira a acrescentar funcionalidade ao sistema, foi decidido dar suporte a transferência de arquivos entre os clientes A e B. Para tal funcionalidade, primeiramente foi necessário identificar os possíveis problemas para decidir quais funções deveriam ser implementadas no sistema.

Deve ser notado que os datagramas UDP são encapsulados dentro de pacotes IP, estes pacotes tem teoricamente um limite de 65,535 *bytes* incluindo seu cabeçalho e o datagrama UDP encapsulado que também inclui seu próprio cabeçalho e os dados transmitidos, o que inclui as variáveis de controle como o ID dos clientes e seus endereços. A partir destas informações, deve-se limitar o número máximo de *bytes* de dados do arquivo a ser transmitido por pacote. Este número máximo irá variar com diferentes conexões.

Após se descobrir o número máximo de dados que serão transmitidos por pacotes, o cliente que deseja enviar o arquivo deverá calcular a partir do tamanho do arquivo e do número de *bytes* transmitidos por pacote, qual será o número de pacotes necessários para a transmissão total do arquivo. Este número será usado para alocar memória dinamicamente para um *array* de bytes que será usado como *buffer*.

A transferência de arquivos entre um cliente A e o cliente B se inicia com A enviando o pacote com os primeiros *bytes* de dados, juntamente a um *checksum* calculado com *md5sum* e

o número do pacote para B, ao receber o pacote, B verifica através do número do pacote se este é o pacote esperado, se isso for verdade, B calcula o *checksum* dos dados recebidos e compara com o *checksum* recebido, se isso for verdade, B escreve os dados em seu próprio *buffer* e envia para A o número de sequência do próximo pacote esperado. Desta maneira se dá continuidade ao ciclo de transferência de pacotes.

Como visto no Capítulo 2, falhas na conexão poderão ocasionar problemas como a perda de pacotes, pacotes chegando fora de ordem ou pacotes corrompidos. Como o sistema apresentado neste trabalho não utiliza de janelas deslizantes como o TCP, não há possibilidade de pacotes chegarem fora de ordem já que o cliente enviando o arquivo apenas enviará o próximo pacote quando receber a confirmação que o pacote anterior foi aceito no outro cliente.

O problema de perda de pacotes e pacotes corrompidos foram tratados com a implementação de uma política de reenvio e a implementação de uma função *timeout*, pacotes corrompidos serão identificados graças a verificação do *checksum* realizada pelo cliente receptor, este por sua vez, irá enviar ao cliente que está enviando dados o número de sequência do pacote esperado. O checksum é calculado através do MD5SUM que é uma *hash* de 128 bits composta por 32 caracteres hexadecimais, onde qualquer alteração nos dados transmitidos irá produzir uma nova *hash*. O cliente que envia dados ao receber o número de sequência do cliente receptor irá identificar que este é o número de sequência de um pacote já enviado e fará o reenvio deste pacote até receber a confirmação do cliente receptor.

Caso um pacote dados seja perdido, o cliente que envia pacotes não receberá nenhuma confirmação do cliente receptor o que irá ocasionar no uso da função de *timeout*, o que ocasionará no reenvio do pacote até a recepção de uma confirmação do cliente receptor. É importante notar que um possível Cenário será a perda do pacote de confirmação vindo do receptor e quando o cliente que envia dados, reenviar o mesmo pacote que já foi aceito pelo receptor, este deve identificar através do número de sequência do pacote, que este já foi adicionado ao buffer e reenviar a confirmação para o cliente que está enviando dados.

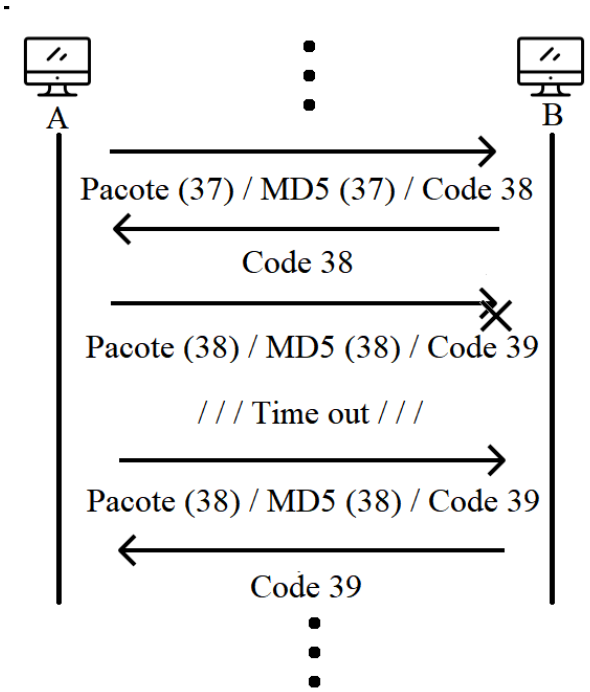

Figura 13 – Exemplo da transmissão de arquivos com perda de pacote
Fonte: Autoria própria

A Figura 13 ilustra um exemplo da transmissão de arquivos, nela é possível observar o uso do número de sequência para o controle síncrono da transmissão, deve ser notado que em ambos os lados o número de sequência será incrementado apenas se o cliente receber a confirmação de outro cliente, no caso da figura, após uma perda de pacote por parte do cliente A, este deve reenviar para B o mesmo pacote com o mesmo número de sequência. Em um caso onde o cliente A não recebe o número de sequência de confirmação de B, ele também irá reenviar o mesmo pacote, porém tendo em vista que B já recebeu tal pacote, este deve ser descartado e B deve reenviar o número de sequência do pacote requisitado para A.

Foi possível perceber a evolução do sistema de maneira a tornar possível a comunicação entre os dispositivos que utilizam de NATs, tal como as verificações e estratégias TCP que foram implementadas a nível de aplicação para o controle dos pacotes UDP para tornar a transmissão de arquivos confiável.

5 EXPERIMENTOS

Este capítulo detalha os experimentos realizados com o sistema em 5 diferentes cenários e os resultados obtidos. No Cenário 1, realizamos o teste do UDP Hole Punching utilizando máquinas virtuais, no Cenário 2, testamos a transferência de arquivos em uma rede sem perda de pacotes, a partir do Cenário 3 começamos os testes reais utilizando dispositivos em diferentes locais e diferentes redes. Para os cenários 3, 4 e 5 foi utilizado como rendezvous um servidor hospedado em São Paulo e um cliente receptor conectado a uma rede ampernet com NAT, no Cenário 3, o cliente que envia dados estava situado na UTFPR, no Cenário 4, este cliente estava em residência no centro de Pato Branco conectado a rede Copel e o Cenário 5 foi um teste de desempenho com uma rede sobrecarregada. As figuras dos casos reais são aproximações das conexões reais baseadas em informações obtidas através do comando *traceroute* para traçar as rotas entre os usuários. Os cenários foram executados diversas vezes durante as fases de desenvolvimento, todavia para a coleta de dados apresentada neste capítulo, os cenários foram executados uma vez.

5.1 CENÁRIO 1

Para a realização de uma prova de conceito do sistema é necessário de pelo menos dois clientes que utilizam de NAT e um computador para atuar como o rendezvous. A dificuldade encontrada para isso foi a simulação do comportamento de um NAT diferente para cliente, devido a isso a solução encontrada foi o uso de máquinas virtuais com o auxílio do software VirtualBox.

O VirtualBox é um software *open source* para virtualização. Este software não só oferece a capacidade de simular sistemas operacionais em suas máquinas virtuais como também configurar suas propriedades de rede, simulando desta forma um NAT através de *iptables* (CORPORATION, 2017).

Entre os modos de rede oferecidos pelo VirtualBox, utilizaremos o NAT, este modo faz com que a máquina virtual tenha um endereço privado e utiliza o endereço público do próprio *Host*, que é a máquina física. Isto faz com que a máquina virtual possa enviar pacotes a Internet, mas apenas os receba pacotes de volta se estes forem respostas aos pacotes previamente enviados (CORPORATION, 2017).

Para o ambiente de testes, foram utilizados de dois computadores com uma máquina virtual utilizando o sistema operacional ubuntu em cada um deles, desta forma, obtivemos nossos clientes A e B simulando IPs públicos e NATs distintos. Para o rendezvous foi o utilizado o servidor orion da UTFPR.

É importante notar que os NATs utilizados pelo VirtualBox são NATs do tipo *Full Cone*, todavia o sistema tem capacidade para suporte dos NATs tipo *Full Cone, Restricted Cone* e *Port*

Restricted Cone. Como explicado no capítulo 3, a atuação do *UDP Hole Punching* é similar entre estes tipos de NAT. Para o NAT *Restricted Cone* A deve enviar um pacote para o servidor, informando o desejo de enviar pacotes a B, para que B também envie pacotes para A de maneira a mapear uma porta por onde os pacotes vindos de A serão aceitos pelos NATs de B. Para o NAT *Port Restricted Cone* uma estrutura de repetição é o suficiente para descobrir o número da porta que o NAT de B mapeou para receber os pacotes de A.

De maneira a verificar o funcionamento dos NATs do primeiro Cenário ilustrado na Figura 14, o primeiro teste a ser realizado foi a tentativa de estabelecer a comunicação direta entre os clientes A e B, em ambas direções, como previsto os pacotes enviados não foram aceitos pelos NATs de nossos clientes. Em seguida tentou-se estabelecer a comunicação por parte do servidor rendezvous com os clientes A e B, os pacotes do rendezvous também foram negados pelos NATs.

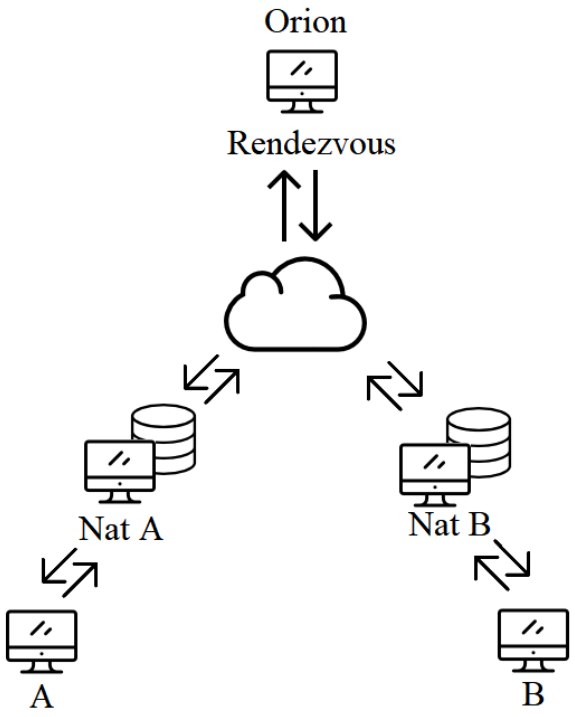

Figura 14 – Cenário 1
Fonte: Autoria própria

Como explicado nos capítulos 2 e 3, os resultados obtidos com os testes iniciais foram coerentes com a teoria, já que para que a comunicação dos clientes para com o servidor rendezvous seja realizada, os clientes devem iniciar tal comunicação e para que os clientes se comuniquem entre si, eles precisam do auxilio do rendezvous.

Para o teste final, se tratando de comunicação, enviamos um pacote de A para o rendezvous com seu número de identificação que foi própriamente adicionado na lista de clientes do servidor assim como o endereço de IP e porta utilizados por A para enviar o pacote. Como resposta o servidor envia para sua lista de clientes que momentaneamente é constituída apenas com o próprio cliente A, porém desta vez o pacote do servidor é aceito pelo NAT de A.

De maneira semelhante o cliente B envia um pacote com seu número de identificação para o rendezvous, que atualiza sua lista de clientes e envia a nova lista para o cliente A e B. Ao receber a lista com os endereços dos clientes, B envia um pacote SYN para o cliente A na tentativa de utilizar o endereço e porta que o NAT de A abriu para se comunicar com o rendezvous e com sucesso o cliente A recebe o pacote de B, mostrando assim o funcionamento do UDP hole Punching.

5.2 CENÁRIO 2

Para o primeiro teste da transferência de arquivos, foram utilizados dois computadores da UTFPR no mesmo laboratório, ambos conectados a Internet via cabo como ilustrado na Figura 15. O número de *bytes* de dados transferido por pacotes foi de 43500, número aproximado do máximo através de testes. Foram realizados testes com três arquivos de tamanhos diferentes e formatos diferentes: .mp3 de 9,5MB, .tif de 184,4MB e .mkv de 535,5MB.

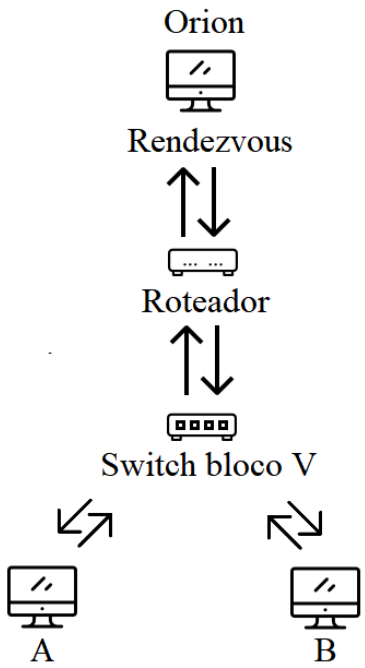

Figura 15 – Cenário 2
Fonte: Autoria própria

Os resultados obtidos nas transferências pode ser encontrado na tabela 1:

Tabela 1 – Transferência entre máquinas no mesmo laboratório

Tamanho do Arquivo	Pacotes Aceitos	Pacotes Reenviados	Taxa de transferência
$9,5$ MB	219 (100%)	0 (0%)	17 MB/s
$184,4$ MB	4240 (100%)	0 (0%)	14 MB/s
$535,5$ MB	12311 (100%)	0 (0%)	13 MB/s

5.3 CENÁRIO 3

Após a realização de testes básicos para verificar o funcionamento do sistema em ambientes controlados, se iniciou a fase de testes em ambientes reais para a avaliação de desempenho do sistema.

Como pode ser visto na Figura 16, para o Rendezvous foi utilizado um servidor *Virtual Private Server* (VPS) de endereço conhecido na web hospedado na cidade de São Paulo, para o cliente receptor foi utilizado um computador localizado em uma residência em Pato Branco chamado de Jaguapitanga em um rede do provedor Ampernet com NAT e para o cliente que deseja enviar o arquivo foi utilizado uma máquina localizada na UTFPR conectada a rede Wifi com o nome de Hamilton-PC. Foram necessárias adaptações para o UDP Hole Punching, devido ao NAT neste caso ser um NAT *Restricted Cone*. O número de *bytes* de dados transferido por pacotes foi de 43500, foram realizados testes com três arquivos de tamanhos diferentes e formatos diferentes: .mp3 de 9,5MB, .tif de 184,4MB e .mkv de 535,5MB.

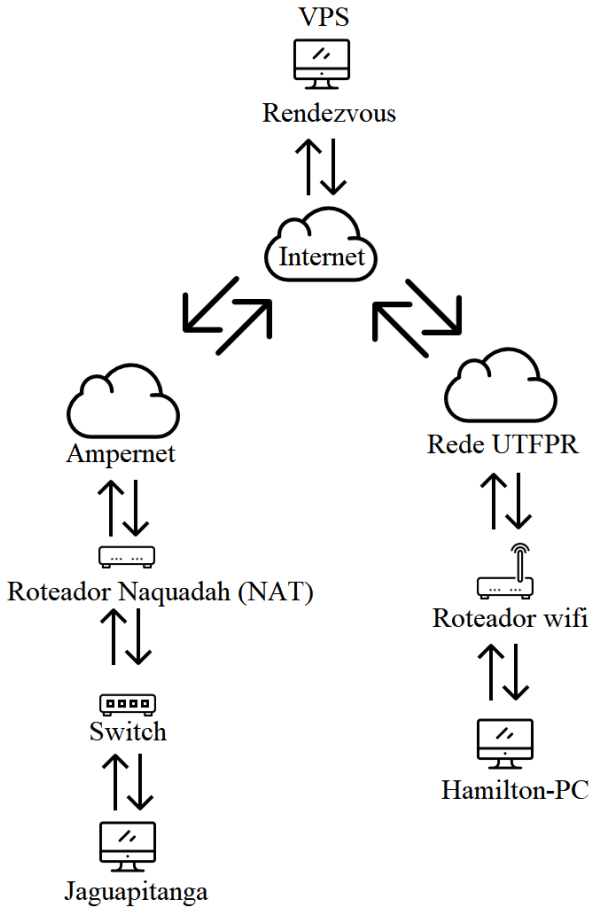

Figura 16 – Cenário 3
Fonte: Autoria própria

Se utilizou do comando *traceroute* para descobrir a rota de tráfego de dados entre os clientes, o resultado pode ser visto na Listagem 5.1 e 5.2.

```
traceroute to jaguapitanga.naquadah.com.br (177.101.140.31), 30 hops max,
60 byte packets
 1 gateway (172.30.0.1) 4.284 ms 4.671 ms 4.660 ms
 2 10.10.10.2 (10.10.10.2) 2.709 ms 3.022 ms 3.021 ms
 3 * * *
 4 187.55.243.109 (187.55.243.109) 14.390 ms 14.364 ms 14.341 ms
 5 BrT-G13-21-cscgo300.brasiltelecom.net.br (201.10.251.142) 13.721 ms
14.236 ms 14.221 ms
 6 100.120.17.161 (100.120.17.161) 28.824 ms 63.226 ms 177.2.219.41
 (177.2.219.41) 32.148 ms
 7 100.120.18.109 (100.120.18.109) 44.347 ms 100.120.18.89 (100.120.18.89)
30.248 ms 27.845 ms
 8 xe-0-1-1-ETPN-SP-ROTB-J01.brasiltelecom.net.br (201.10.242.116)
28.315 ms etpn-sp-rotb-j01-xe-1-0-1.
 brasiltelecom.net.br (201.10.241.9) 28.260 ms 100.120.16.140 (100.120.16.140)
24.555 ms
 9 as12989.saopaulo.sp.ix.br (187.16.216.177) 28.176 ms 28.111 ms 27.101 ms
10 * * *
11 54.111.109.187.dynamic.ampernet.com.br (187.109.111.54) 40.756 ms 40.211 ms
40.642 ms
```

Listagem 5.1 – traceroute entre Hamilton-PC e Jaguapitanga, Cenário 3

Na Listagem 5.1, é possível observar que os pacotes enviados de Hamilton-PC para Jaguapitanga saem da rede da UTFPR e vão até São Paulo como mostrado no item 9 do *traceroute*, onde trocam tráfego para a rede Ampernet para chegarem até Jaguapitanga. Fazendo a análise da Listagem 5.2, não vemos o mesmo nó localizado em SP o que mostra que a comunicação não é simétrica, ou seja os pacotes passam por diferentes rotas dependendo de seu destino.

Considerando que ambos clientes estão situados em Pato Branco, o fato de cada pacote que sai de Hamilton-PC para Jaguapitanga passar por São Paulo causa um atraso na comunicação devido as redes serem diferentes.

```
traceroute to 187.6.138.220 (187.6.138.220), 30 hops max, 60 byte packets
 1 naquadria.naquadah.com.br (10.31.38.254) 0.197 ms 0.236 ms 0.192 ms
 2 1.140.101.177.dynamic.ampernet.com.br (177.101.140.1) 1.360 ms 1.290 ms
1.213 ms
 3 53.111.109.187.dynamic.ampernet.com.br (187.109.111.53) 1.874 ms 2.020 ms
1.948 ms
 4 73.53.186.200.sta.impsat.net.br (200.186.53.73) 16.184 ms 16.325 ms 16.263 ms
 5 64.208.27.222 (64.208.27.222) 26.298 ms 64.208.27.70 (64.208.27.70)
21.533 ms 21.220 ms
 6 BrT-G5-0-0-ctje-pr-rotn-01.brasiltelecom.net.br (201.10.243.39) 31.529 ms
 100.120.17.85 (100.120.17.85) 30.521 ms BrT-G4-0-0-ctme-pr-rotn-01.
 brasiltelecom.net.br (201.10.243.37) 33.785 ms
 7 201.10.243.74 (201.10.243.74) 32.815 ms 100.120.17.160 (100.120.17.160)
57.433 ms 57.633 ms
 8 BrT-G5-1-1-cscgo1010.brasiltelecom.net.br (201.10.251.145) 58.462 ms
58.681 ms 58.586 ms
```

Listagem 5.2 – traceroute entre Jaguapitanga e Hamilton-PC, Cenário 3

Os resultados obtidos no Cenário 3 com transferências de arquivos pode ser encontrado na tabela 2. Para método de comparação, a transferência também foi realizada através do comando UNIX Secure copy (SCP).

Tabela 2 – Primeiro teste de desempenho

Tamanho do Arquivo	Pacotes Aceitos	Pacotes Reenviados	Taxa de transferência	Secure copy
$9,5$ MB	214 $(97,72\%)$	5 $(2,28\%)$	$415,282$ KB/s	$1,2$ MB/s
$184,4$ MB	4188 $(98,78\%)$	52 $(1,22\%)$	$302,168$ KB/s	$1,2$ MB/s
$535,5$ MB	12049 $(98,88\%)$	262 $(2,12\%)$	$282,287$ KB/s	$1,2$ MB/s

5.4 CENÁRIO 4

Para o segundo teste de desempenho, como pode ser visto na Figura 17, se manteve o Rendezvous e o Cliente receptor do teste primeiro teste, porém o cliente que envia pacotes foi trocado para um computador localizado em uma residência conectado a rede Copel. Para este teste o número de *bytes* de dados por pacote foi modificado para 12000, já que utilizando números maiores o conteúdo dos pacotes foi sobrescrito, devido a um *Maximum Transmission Unit* (MTU) menor para os pacotes IP na rede Copel. Deve ser notado que os provedores de Internet Copel e Ampernet tem pontos de troca de tráfego em Curitiba e Foz do Iguaçu.

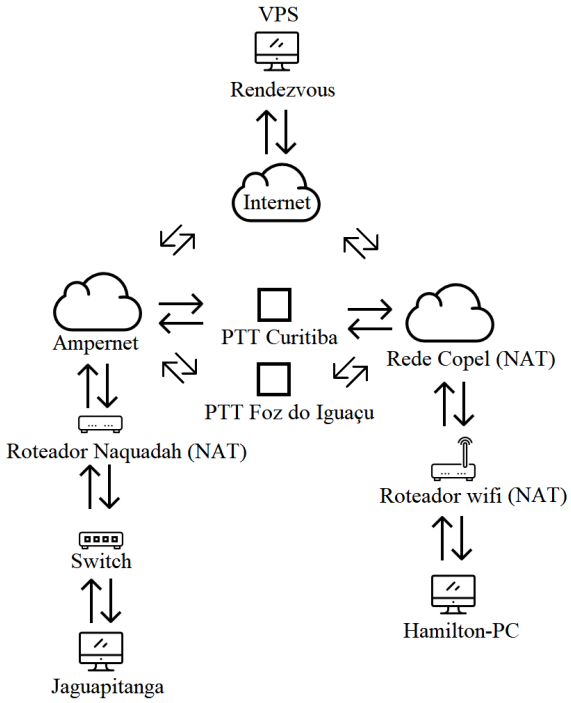

Figura 17 – Cenário 4
Fonte: Autoria própria

Se utilizou do comando *traceroute* para descobrir a rota de tráfego de dados entre os clientes, o resultado pode ser visto na Figura 5.3 e 5.4.

```
traceroute to jaguapitanga.naquadah.com.br (177.101.140.31), 30 hops max,
60 byte packets
 1 gateway (192.168.100.1) 1.242 ms 4.722 ms 5.387 ms
 2 * * *
 3 200.150.94.4 (200.150.94.4) 129.902 ms 13.200 ms 13.934 ms
 4 tengiga400-src1phs-src1cen.copel.net (200.150.93.99) 21.838 ms 22.544 ms
20.991 ms
 5 te1400-brs1km3a-trunk10-998-src1km3a.copel.net (200.150.92.177) 23.852 ms
28.278 ms 24.706 ms
 6 187.16.204.16 (187.16.204.16) 27.580 ms 25.085 ms 25.202 ms
 7 17.215.60.187.dynamic.ampernet.com.br (187.60.215.17) 25.838 ms 28.772 ms
29.993 ms
 8 54.111.109.187.dynamic.ampernet.com.br (187.109.111.54) 28.167 ms 29.391 ms
26.863 ms
```

Listagem 5.3 – traceroute entre Hamilton-PC e Jaguapitanga, Cenário 4

Analisando a Listagem 5.3 é possível ver que o caminho feito pelos pacotes que vão de Hamilton-PC para Jaguapitanga passam por Foz do Iguaçu de acordo com o item 6, que é o ponto de troca de tráfego entre a rede Copel e Ampernet. Diferentemente do Cenário 3, no Cenário 4, o caminho percorrido pelos pacotes de Jaguapitanga para Hamilton-PC aparenta ser

o mesmo caminho fazendo desta uma conexão simétrica e com o a troca de contexto em um ponto fisicamente mais próximo da localização dos clientes.

```
traceroute to 168.181.50.103 (168.181.50.103), 30 hops max, 60 byte packets
 1 naquadria.naquadah.com.br (10.31.38.254) 0.220 ms 0.246 ms 0.278 ms
 2 1.140.101.177.dynamic.ampernet.com.br (177.101.140.1) 1.922 ms 1.930 ms
1.937 ms
 3 53.111.109.187.dynamic.ampernet.com.br (187.109.111.53) 1.997 ms 2.571 ms
2.585 ms
 4 18.215.60.187.dynamic.ampernet.com.br (187.60.215.18) 3.946 ms 3.956 ms
4.102 ms
 5 as14868.fozdoiguacu.pr.ix.br (187.16.204.13) 8.215 ms 8.211 ms 8.679 ms
 6 tengiga310-src1vyo-src1foz.copel.net (200.150.92.174) 7.634 ms 6.513 ms
6.747 ms
 7 tengiga507-src1cos-src1vyo.copel.net (200.150.93.188) 15.627 ms 15.933 ms
16.045 ms
```

Listagem 5.4 – traceroute entre Jaguapitanga e Hamilton-PC, Cenário 4

Os resultados obtidos no Cenário 4 com transferências de arquivos pode ser encontrado na tabela 3.

Tabela 3 – Segundo teste de desempenho

Tamanho do Arquivo	Pacotes Aceitos	Pacotes Reenviados	Taxa de transferência	Secure copy
$9,5$ MB	777 $(98,11\%)$	15 $(1,89\%)$	$171,54$ KB/s	$923,2$ kB/s
$184,4$ MB	15194 $(98,88\%)$	173 $(1,12\%)$	$198,92$ KB/s	$863,6$ kB/s
$535,5$ MB	44089 $(98,80\%)$	536 $(1,20\%)$	$203,12$ KB/s	$878,3$ kB/s

5.5 CENÁRIO 5

Para o terceiro teste de desempenho, foram utilizadas as mesmas configurações do segundo teste e foi transmitido apenas o maior arquivo de 535,5 MB. O intuito deste teste foi avaliar o desempenho do sistema em uma rede sobrecarregada, para isso, simutaneamente a transferência do arquivo, foram executadas no dispositivo do cliente diversas aplicações de uso de rede intensivo de maneira a forçar a perda de pacotes.

Os resultados obtidos no terceiro teste podem ser vistos na tabela 4.

Tabela 4 – Terceiro teste de desempenho

Tamanho do Arquivo	Pacotes Aceitos	Pacotes Reenviados	Taxa de transferência
$535,5$ MB	$40980(91,84\%)$	$3645(8,16\%)$	$175,516$ kB/s

Foi possível perceber com análise dos resultados obtidos, que a taxa de transferência dados não só é afetada pelo tráfego na rede, mas principalmente pelo tamanho do arquivo. Devido ao funcionamento do sistema ser bloqueante, ao que se refere esperar uma resposta de confirmação para o envio do próximo pacote, se para o envio de arquivo for necessário o envio de N pacotes, desconsiderando a perda de pacotes, será necessário a espera pela confirmação do cliente receptor N vezes o que irá atrasar a transferência. Tal problema poderia ser resolvido com a implementação de janelas deslizantes e retransmissão seletiva, o que todavia poderia ocasionar a chega de pacotes fora de ordem, um novo problema a ser tratado.

Outro fato importante é que quanto maior o tamanho do arquivo, mais pacotes serão perdidos na rede e para cada pacote perdido, o sistema irá esperar o tempo necessário para execução da função de *timeout*.

A partir deste capítulo, foi possível afirmar que o sistema é factível, todavia apresenta limitações de desempenho se comparado com outros métodos existentes e já amplamente utilizados. O repositório com o sistema está no repositório *github* [1].

[1] https://github.com/HRNM2/UDPholepunchFiletransfer

6 CONCLUSÃO

Este trabalho teve como objetivo expor o conceito de transparência na Internet, sua importância, avaliar as consequências e problemas referentes a sua perda devido ao uso de aplicações intermediárias entre a comunicação de dois dispositivos e propor uma solução para estes problemas.

Foi possível perceber que o maior obstáculo em manter a transparência na Internet é a mudança na maneira como são feitas as transferências de dados, o crescimento de usuários na Internet e novas necessidades para a segurança ou para integridade dos dados com o passar dos anos.

A ferramenta adotada neste trabalho para estabelecer as conexões entre os dispositivos, o UDP Hole Punching, se tornou extremamente eficiente, sendo capaz de realizar sua função mesmo em ambientes reais com diferentes tipos de NAT presentes.

O maior desafio encontrado neste trabalho foi o sistema de transferência de arquivos, via protocolo UDP, já que para o funcionamento deste diversas funções inerentes do protocolo TCP tiveram que ser implementadas, todavia como mostrado anteriormente devido sua necessidade de confirmações para estabelecer uma conexão não torna o TCP uma opção para os cenários testados neste trabalho. Notou-se que diversas aplicações implementam certo nível de confiabilidade ao UDP que em certos casos pode ter vantagem a uma conexão TCP devido a ter um cabeçalho menor e latência melhor.

Após a realização de diversos testes e análise de desempenho percebeu-se que o sistema sempre atingiu seu objetivo, porém ele possuiu diversas limitações enquanto a seu desempenho, durante o desenvolvimento deste trabalho e a realização de testes, surgiram ideéias que poderiam ser implementadas em trabalhos futuros para a melhoria de seu desempenho, como a implementação de janelas deslizantes e retransmissão seletiva, a implementação com o uso de *multithreading* com uma *thread* para recepção e outra para envio, a implementação de predição de portas para NATs do tipo *symmetric* e acrescentar uma interface TUN/TAP IP ao UDP para evitar a implementação de estratégias TCP/IP, já que a própria interface se tornaria responsável pelo controle dos pacotes.

O sistema final oferece suporte a comunicação de clientes a todos o NATs do tipo cone, a transferência de arquivos via UDP com funcionalidades que garantem a confiabilidade na transmissão. O sistema é livre para o uso.

REFERÊNCIAS

BELLOVIN, S. M. There be dragons. **UNIX Security Symposium**, 1992.

BORGES, J. G. G. **Stunpede: Um sistema p2p para conectividade fim-a-fim transparente na internet usando túneis IPV6-sobre-udp**. 62 p. Dissertação (Mestrado) — Universidade Federal do Paraná, Curitiba, 2008.

CARPENTER, B. **Internet transparency**. [S.l.], 2000. [Acessado em 31 de março de 2016]. Disponível em: <https://www.ietf.org/rfc/rfc2775.txt>.

CORPORATION, O. **User Manual**. [S.l.], 2017. [Acessado em 01 de Novembro de 2017]. Disponível em: <https://www.virtualbox.org/manual/ch06.html>.

DEERING, R. H. S. **Internet Protocol, Version 6 (IPv6)**. [S.l.], 1998. [Acessado em 27 de agosto de 2017]. Disponível em: <https://www.ietf.org/rfc/rfc2460.txt>.

DINHA, J. Y. F. [S.l.], 2016. [Acessado em 15 de junho de 2016]. Disponível em: <https://openvpn.net/index.php/about-menu/about-us.html>.

EDDY, W. M. Defenses against tcp syn flooding attacks. **The Internet Protocol Journal, v.9**, 2006.

EGEVANG, P. F. K. **The IP Network Address Translator (NAT)**. [S.l.], 1994. [Acessado em 31 de março de 2016]. Disponível em: <https://www.ietf.org/rfc/rfc1631.txt>.

FORD PYDA SRISURESH, D. K. B. Peer-to-peer communication across network address translators. **USENIX Annual Technical Conference, p.179-192**, 2005.

____. **State of Peer-to-Peer (P2P) Communication across Network Address Translators (NATs)**. [S.l.], 2008. [Acessado em 27 de agosto de 2017]. Disponível em: <https://tools.ietf.org/html/rfc5128>.

GU, Y. **UDP-based Data Transfer**. [S.l.], 2012. [Acessado em 08 de dezembro de 2017]. Disponível em: <http://udt.sourceforge.net/>.

HAIN, T. A pragmatic report on ipv4 address space consumption. **The Internet Protocol Journal, v.8, p.2-19**, 2005.

POSTEL, J. **Transmission Datagram Protocol**. [S.l.], 1980. [Acessado em 31 de março de 2016]. Disponível em: <https://www.ietf.org/rfc/rfc793.txt>.

____. **User Datagram Protocol**. [S.l.], 1980. [Acessado em 31 de março de 2016]. Disponível em: <https://www.ietf.org/rfc/rfc768.txt>.

____. **Internet Protocol**. [S.l.], 1981. [Acessado em 31 de março de 2016]. Disponível em: <http://www.ietf.org/rfc/rfc0791.txt>.

PROJECT, R. T. S. **Internet Live Stats**. [S.l.], 2016. [Acessado em 31 de março de 2016]. Disponível em: <http://www.internetlivestats.com/internet-users>.

REKHTER B. MOSKOWITZ, D. K. G. d. G. Y. **Address Allocation for Private Internets**. [S.l.], 1994. [Acessado em 31 de março de 2016]. Disponível em: <https://www.ietf.org/rfc/rfc1597.txt>.

ROMANOFSKI, E. **A Comparison of Packet Filtering Vs Application Le vel Fire wall Technology**. [S.l.], 2002. [Acessado em 31 de março de 2016]. Disponível em: <https://www.giac.org/paper/gsec/693/comparison-packet-filtering-vs-application-level-firewall-technology/101569>.

ROSENBERG J. WEINBERGER, C. H. R. M. J. **STUN - Simple Traversal of User Datagram Protocol (UDP) Through Network Address Translators (NATs)**. [S.l.], 2003. [Acessado em 31 de março de 2016]. Disponível em: <https://www.ietf.org/rfc/rfc3489.txt>.

SCHMITD, J. **How Skype & Co. get round firewalls**. [S.l.], 2006. [Acessado em 31 de março de 2016]. Disponível em: <http://infocom.uniroma1.it/alef/enav/skypenat.pdf>.

TIMMERMANS, G. S. I. **tinc Manual**. [S.l.], 2016. [Acessado em 15 de junho de 2016]. Disponível em: <https://www.tinc-vpn.org/documentation/tinc.pdf>.