UNIVERSIDADE FEDERAL DO PARÁ
CAMPUS UNIVERSITÁRIO DE TUCURUÍ
FACULDADE DE ENGENHARIA DE COMPUTAÇÃO

UM SCRIPT BASEADO NA BIBLIOTECA SCAPY QUE INTERCEPTA E BLOQUEIA O TRÁFEGO BITTORRENT

AMÓS SANTOS BRANDÃO

TUCURUÍ - PA
2019

UNIVERSIDADE FEDERAL DO PARÁ
CAMPUS UNIVERSITÁRIO DE TUCURUÍ
FACULDADE DE ENGENHARIA DE COMPUTAÇÃO

AMÓS SANTOS BRANDÃO

UM SCRIPT BASEADO NA BIBLIOTECA SCAPY QUE INTERCEPTA E BLOQUEIA O TRÁFEGO BITTORRENT

Trabalho de Conclusão de Curso apresentado para obtenção do grau de engenheiro da Computação Faculdade de Engenharia de Computação, do Campus Universitário de Tucuruí.

Tucuruí - PA
2019

UM SCRIPT BASEADO NA BIBLIOTECA SCAPY QUE INTERCEPTA E BLOQUEIA O TRÁFEGO BITTORRENT

Este trabalho foi julgado adequado em _____ / _____ / _____ para obtenção do grau de Engenheiro de Computação, aprovado em sua forma final pela banca examinadora que atribuiu o conceito _____.

_____
Prof. Me. Caio Carvalho Moreira
Orientador

_____
Prof. Dr. Adonney Allan de Oliveira Veras
Membro da Banca Examinadora

_____
Prof. Dr. Otávio Noura Teixeira
Membro da Banca Examinadora

_____
Prof. Otávio Noura Teixeira
Diretor da Faculdade de Engenharia de Computação

DEDICATÓRIA

Dedico este trabalho primeiramente Deus, aos meus pais que sempre me ajudaram nessa jornada da faculdade e aos meus professores.

RESUMO

Aplicações peer-to-peer ou P2P geram alto consumo de tráfego de banda de internet. Este trabalho tem como objetivo analisar e identificar e bloquear o fluxo de rede de internet oriundo de aplicações P2P pertencentes ao protocolo BitTorrent. A Metodologia usada é a pesquisa quantitativa apoiada em técnica de coleta de dados também quantitativas. A análise da carga útil dos pacotes busca identificar características do protocolo BitTorrent. Alguns roteadores atuais não oferecem a opção de bloqueio de pacotes por tipo de protocolo da camada de aplicação.

Palavras-chave: P2P; tráfego; BitTorrent; aplicação

ABSTRACT

Peer-to-peer or P2P applications generate high consumption of internet bandwidth traffic. This work aims to analyze and identify and block the Internet network flow from P2P applications belonging to the BitTorrent protocol. The methodology used is quantitative research based on data collection techniques that are also quantitative. The identification of BitTorrent traffic is done by analyzing the packet payload for BitTorrent protocol characteristics. Some current routers do not offer the option of blocking packets by protocol type of the application layer.

Keywords: P2P; traffic; BitTorrent; application

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE COMANDOS

LISTA DE GRÁFICOS

LISTA DE CÓDIGOS

SUMÁRIO

1. INTRODUÇÃO

A internet é uma rede de computadores mundial, isto é, uma rede que conecta milhares de dispositivos de todos os tamanhos e modelos (KUROSE, 2014). Existem diversos programas que fazem uso da internet para desenvolver suas tarefas básicas, um exemplo disso são programas de EMAIL[1], FTP[2] (Protocolo de Transferência de Arquivos) e aplicações cliente do protocolo BitTorrent[3].

Uma parcela considerável das aplicações em funcionamento em redes locais utiliza a arquitetura Par-a-Par (P2P [4]). Através de protocolo BitTorrent, essa arquitetura objetiva compartilhar arquivos de áudio, de vídeo, documentos, imagens e outros (WONG, 2011). O processo de compartilhamento de dados é dividido em duas partes: Par-a-Servidor e Par-a-Par. A primeira é a comunicação entre programa cliente e servidor, e a segunda ocorre a comunicação e transferência de dados entre parceiros (MEMON, 2014).

O protocolo BitTorrent representa aproximadamente 45 a 78% de todo o tráfego P2P e 27 a 55% de todo o tráfego da Internet a partir de fevereiro de 2009, (KLEMM, A. et al, 2004). Mais recentemente, de acordo com o site Observatory, que oferece estatísticas de tráfego de Internet em tempo real e ilustra quanta largura de banda[5] é consumida por diferentes aplicativos, em setembro de 2011 o tráfego P2P na Europa representava mais de 25% de toda a largura de banda e cerca de 40% de todos os pacotes enviados foram da arquitetura P2P. Em virtude da quantidade de pacotes enviados ser considerável

É notório a crescente popularidade de programas clientes P2P usados para download de arquivos. Por apresentarem interface simples, facilitam o seu manuseio por pessoas inexperientes. Conforme defendido por (BIN, L.; ZHITANG, L.;

[1] Sistema de comunicação baseado no envio e recebimento de mensagens.
[2] Significa Protocolo de Transferência de Arquivos, é uma forma bastante rápido de transferência de arquivos.
[3] É uma maneira de compartilhar arquivos entre usuários em forma de protocolo de rede que permite ao utilizador realizar downloads.
[4] Arquitetura quer torna os nós presentes na rede funcionar como cliente e servidor.
[5] Indica a medida da capacidade de uma transmissão.

ZHANCHUN, L, 2006), o tráfego da rede P2P chegou a 80% no Backbone [6] CARNET. O número de clientes do protocolo BitTorrent tende a crescer com o passar dos anos (DONALD, 2018).

Dados a esses enormes fluxos de tráfego, muitas organizações precisam controlar seu fluxo de rede através da detecção e limitação da largura de banda (MEMON, 2014). De acordo com (BIN, L.; ZHITANG, L.; ZHANCHUN, L, 2006), uma séria consequência das aplicações BitTorrent é o alto tráfego de consumo que ocasiona na lentidão da rede local, visto que a aplicação BitTorrent aumenta seu consumo de forma proporcional a quantidade de banda.

A maioria dos roteadores comerciais da atualidade não possui função de controle de aplicação P2P automatizada por inspeção de pacotes, consequentemente, da taxa de download e upload é limitado. De acordo com o site oficial da Intelbras o roteador 240 WRN possui a função de controle de banda, entretanto, é estática e imprecisa por só limitar a taxa de download e upload de uma aplicação. Semelhantemente, o site da Tp-Link informa também que diversos modelos de roteadores como TL-WR841N, TL-WDR3500, TL-WR743ND, Archer C50(V1), entre outros, apresentam bloqueio de banda estático como nos modelos da Intelbras.

Com base nestas características e desvantagens de tráfego P2P, sugere-se a criação de um Script de bloqueio através da análise do conteúdo do pacote e identificar a utilização do protocolo BitTorrent. Baseado nos problemas gerados pelas aplicações cliente BitTorrent, a criação de um Script de Inspeção Profunda de Pacotes (DPI) para redes LAN[7] (Área de Rede Local) se faz necessária. Uma vez o Script criado, todo o fluxo da rede é analisado e bloqueado à medida que for identificado.

1.2. OBJETIVO GERAL

,

[6] É a rede principal por onde os dados dos clientes trafegam.
[7] Conjunto de computadores conectados.

Desenvolver um Script para a última versão básica do Debian 10 baseado na biblioteca Scapy[8] e Iptables[9], que bloqueia qualquer IP[10] (Protocolo de Internet) externo que originou o pacote BitTorrent.

1.2.1. Objetivos Específicos:

- Buscar na documentação da biblioteca Scapy métodos que filtram pacotes.
- Procurar configuração necessária do Iptables para transformar o Sistema Operacional Debian 10 em roteador Wi-Fi.[11]
- Desenvolver um Script para o Debian 10 que bloqueie a passagem do tráfego BitTorrent através de características intrínsecas dos pacotes.
- Bloquear pacotes que apresentam protocolo BitTorrent usado na comunicação.
- Melhorar a velocidade de internet dos dispositivos conectados à rede local.

1.3. ORGANIZAÇÃO DO TRABALHO

O documento está dividido em 8 capítulos. No primeiro capítulo é mostrado a contextualização do problema, apresentação do projeto e os objetivos.

No segundo capítulo é feito um levantamento da definição análise de redes P2P, mostrando alguns trabalhos relacionados a identificação de pacotes P2P e suas técnicas usadas.

O terceiro capítulo é mostrado o método utilizado para a captura e análise dos dados de fluxo de pacotes e como foi feito os testes.

O quarto capítulo faz uma revisão de como os pacotes são filtrados pela biblioteca Scapy da Linguagem de programação Python e com são bloqueados através da ferramenta Iptables.

---

[8] Biblioteca em Python para captura de pacotes.
[9] É um conjunto de ferramentas e medidas que permite o controle e a definição de regras de firewall.
[10] É responsável por endereça e encaminhar os pacotes.
[11] É uma tecnologia de comunicação que não utiliza cabos.

O quinto capítulo nos mostra o detalhamento do projeto, como foi feito, bem como é bloqueado conexões P2P pertencentes ao protocolo BitTorrent.

O sexto capítulo discute todos os resultados obtidos que o Script em funcionamento proporciona.

O sétimo capítulo mostra melhoramento para futuras versões do Script.

O oitavo capítulo conclui o trabalho.

2. REVISÃO BIBLIOGRÁFICA

2.1. Principais Técnicas de detecção dos Tráfego P2P

Nesse capítulo é introduzido os métodos de detecção de fluxo P2P encontrados na área acadêmica. Em geral existem três métodos principais de detecção que são: Inspeção de Porta TCP [12](Protocolo de Controle e Transmissão) ou UDP (Protocolo de Datagrama do Usuário), DFI[13] (Inspeção Profunda de Fluxo) e DPI[14] (Inspeção Profunda de Pacote).

2.1.1. Inspeção de Porta

Inspeção de porta TCP ou UDP[15], também conhecida como análise de porta ou escaneamento de porta, é o nome dado à técnica usada para identificar o estado de uma porta em uma rede. De acordo com (FINSTERBUSCH, 2014), A análise por porta é uma das técnicas mais antigas usada na classificação do tráfego. Trata-se de uma técnica largamente utilizada para identificar alvos, uma vez que todos os computadores conectados na internet ou com uma rede local necessariamente precisam ter algumas de suas portas abertas para eventuais trocas de dados ou informações.

[12] Protocolo de transporte fim-a-fim que, orientado a conexão, que fornece um serviço de transferência confiável.
[13] Analisa o fluxo de pacotes em busca de características que descrevem um comportamento de um protocolo.
[14] Inspeciona a carga útil dos pacotes.
[15] Protocolo do nível de transporte que implementa um serviço do tipo best-effort (não confiável sem garantia na entrega de pacotes).

Em essência, a varredura de portas consiste em analisar cada porta, de determinado sistema final, referente ao protocolo TCP ou UDP, visto que cada aplicação usa porta fixa (MEMON, 2014). O tipo de resposta pode evidenciar, por exemplo, que uma determinada porta está sujeita a passagem de fluxo P2P.

2.1.2. Inspeção Profunda de Pacotes (DPI)

A DPI (inspeção profunda de pacotes) é a tecnologia usada para capturar pacotes de rede à medida que passam por roteadores e outros dispositivos de rede, além de realizar uma filtragem de pacotes para examinar os dados e localizar informações mais profundas sobre os dados levados pelos pacotes (BIN, L.; ZHITANG, L.; ZHANCHUN, L, 2006).

O artigo proposto por (KERALAPURA, R.; NUCCI, A.; CHUAH, C.-N, 2008) usa dois estágios para detectar o tráfego P2P. O primeiro estágio é baseado em um algoritmo de métrica de correlação de tempo, que identifica o tráfego P2P. Este algoritmo é com base no tempo de chegada dos pacotes e também no fluxo da carga útil dos pacotes além de poder trabalhar com tráfego criptografado. O segundo estágio é baseado em uma extração de assinatura de carga útil do pacote. Com o estágio 1 e o estágio 2, os tráfegos P2P podem ser classificados com precisão.

2.1.3. Inspeção Profunda de Fluxo (DFI)

Na inspeção de fluxo profundo (DFI) é analisado o padrão de nível do fluxo de pacotes, desprezando a carga útil e criptográfica dos mesmos. Não existe uma regra geral para obter os padrões de níveis (MEMON, 2014). No trabalho de (LE, T. M.; BUT, J, 2009) foi usado algoritmo DFI para identificar tráfego P2P. Ele propõe estatísticas de comprimento de pacotes de tráfego como recursos para classificar o tráfego de pacotes BitTorrent.

2.2. Trabalhos Relacionados

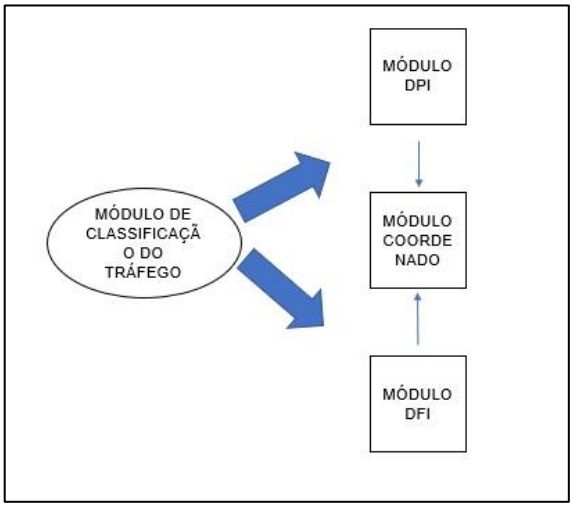
2.2.1. Modelo híbrido de detecção de pacotes P2P

No trabalho de (CHEN, H. et al, 2009), é identificado o tráfego BitTorrent baseado em um modelo de detecção híbrido que integra tanto DFI quanto DPI. A

classificação do tráfego é feita em quatro partes que são: captura do tráfego, módulo DPI, módulo DFI e coordenação do modulo DFI e DPI.

Na captura do tráfego os pacotes da rede LAN são capturados para análise. A Figura 1 mostra o fluxo geral de classificação do tráfego.

Figura 1. Diagrama classificação de pacotes DFI e DPI



Fonte: Do próprio Autor

No módulo DPI são usados os algoritmos Aho-Corasick[16], Wu-Manber[17] e SBOM [18](Set Backward Oracle Matching) para identificar padrões nos pacotes DPI. Todos esses algoritmos usam o conceito de autômatos finitos na busca por Strings características do protocolo P2P. A Figura 2 mostra as Strings identificadas por esses algoritmos.

Tabela 1. Strings características de alguns protocolos P2P

Protocolos P2P	Strings Características
BitTorrent	0x13BitTorrent protocol
Gnuttela	“GNUT”, “GIV”, “GND”
eDonkey2000	0xe319010000
Fastracker	“Get/.hash”
QQLive	0xfe290404

[16] Busca Strings (Palavras) a partir de uma única interação.
[17] Algoritmo que busca um padrão com o uso de autômatos finitos.
[18] Algoritmo de autômato finito.

Fonte: Do próprio Autor

O método DFI identifica padrões no fluxo dos pacotes. Padrões esses que podem ser número total em bytes do fluxo (TBF), média de bytes de pacote de fluxo (APBF), entre outros. Com base na afirmação de (MEMON, 2014) não existe um padrão na análise DFI. Algumas análises dos fluxos DFI podem ter efeitos positivos na classificação, enquanto outros efeitos negativos. A seleção de recursos não só aumenta precisão da classificação, mas também reduz o tempo de treinar e melhorar o desempenho do algoritmo.

Para analisar os recursos são usados os algoritmos SVM[19] (Support Vector Machines), Redes neurais[20], Teorema de Bayes[21] e Árvore de Decisão[22]. Todos esses algoritmos envolvem inteligência artificial e são capazes de analisar o fluxo dos pacotes de acordo com os padrões de fluxo TBF e APBF.

Ao termino da análise do módulo DFI e DPI é feito a classificação coordenada dos pacotes. O método coordenado divide o fluxo dos pacotes em quatro classes que estão representadas na Tabela 2.

Tabela 2. Método Coordenado

Classificação do Tráfego	Descrição
Tráfego P2P predefinido	Aplicativos P2P com códigos Predefinidos na biblioteca DPI.
P2P não predefinido	Aplicativos P2P com códigos não predefinidos na biblioteca DPI.
P2P criptografado	Aplicativos criptografados.
Tráfego não P2P	Aplicativos não P2P.

Fonte: Próprio Autor

A primeira classe representa os pacotes identificados pelo algoritmo de autômatos finitos do Módulo DPI e que possuem a String da aplicação P2P salva na

[19] Técnica de aprendizado de máquina que utiliza reconhecimento de padrão.
[20] São sistemas de computação com nós conectados que funcionam como neurônio.
[21] Fórmula matemática usada para cálculo da probabilidade de um evento dado quer outro evento já ocorreu.
[22] Representa uma tabela de decisão dobre a forma de uma árvore.

biblioteca DPI. Na segunda classe os pacotes capturados não possuem a String da aplicação P2P vinculada a biblioteca DPI. E por último tem a classe dos pacotes criptografados pertencentes ao módulo DFI e os que não pertencem a arquitetura P2P.

Segundo o autor o método híbrido de classificação obteve êxito. Esse método não só aumentou a veracidade de identificação do tráfego P2P como a extensão de identificação das aplicações P2P.

2.2.2. Identificação de pacotes P2P pela String Hexadecimal

O trabalho de (LIU, Y.; YANG, Y, 2013) faz uma análise dos pacotes em busca de Strings características semelhante ao trabalho de (CHEN, 2009). Nessa análise é considerado o nome característico do protocolo da aplicação escrito em Hexadecimal dentro de cada pacote do fluxo.

A String característica da aplicação BitTorrent é dividida em duas partes, que consiste em “0x13”, que representa 1 byte e o nome do protocolo representado em Hexadecimal “426974”. Na Tabela 3 é possível verificar as Strings características em hexadecimal dos principais protocolos P2P.

Tabela 3. Códigos característicos hexadecimal

Aplicação	Protocolo	String Característica	Posição
Oicq[9]	UDP	0x12	0x2a
Bittorrent[6]	TCP	0x13426974	0x36-0x39
QQDOWNLOAD	UDP	0xfe+lenght (payload-3)	0x2a-0x2c
qqlive	UDP	0xfe+lenght (payload-3)	0x2a-0x2c
pplive	UDP	0x010000	0x34-0x36
Thunder[6]	UDP	0x3b000000 /0x3c000000 /0x32000000	0x2a-0x2d
	TCP	0x32000000	0xb7-0xba
PPS	UDP	0x80/0x84	0x2b
	UDP	Lenght(payload-	0x2a-0x2d

		6)+0x4300	
eDonkey/Emule[10]	UDP	E3 ou c5 ou d4	0x2a
	TCP	E3 ou c5 ou d4	0x36

Fonte: Próprio Autor

O autor ainda informa que a representação por String característica do protocolo P2P sofre mudanças no decorrer de atualizações da aplicação cliente, consequentemente, a identificação de pacotes P2P não é aplicado apenas a carga útil dos pacotes e sim a porta em uso pela aplicação P2P.

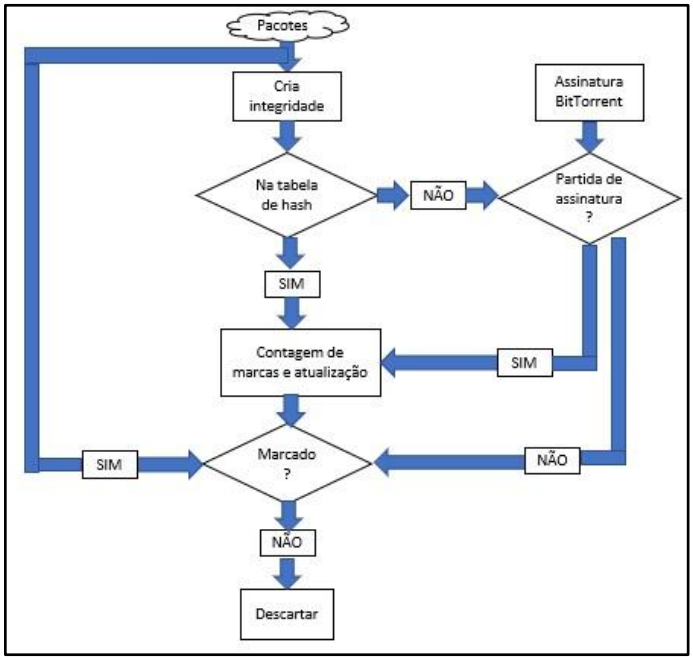
Os resultados do trabalho mostram que o protótipo criado não é capaz de identificar o fluxo P2P corretamente, apresentando índices de 27,19% do tráfego identificado como P2P, mas de acordo com as estatísticas da rede a porcentagem correta do fluxo é entre 40% e 60%. Esse resultado incorreto, de acordo com o autor, ocorreu por conta de o tráfego ser criptografado, tornando a captura de pacotes é limitada.

2.2.3. Assinatura da camada de aplicação

(BIN, L.; ZHITANG, L.; ZHANCHUN, L, 2006) propôs um trabalho de análise de tráfego BitTorrent semelhante ao de (LIU, Y.; YANG, Y, 2013) que considera a assinatura da camada de aplicação que se encontra nos pacotes iniciais de comunicação entre parceiros. A análise é dividida em duas partes que consistem em separar o tráfego em BitTorrent e não BitTorrent e segunda é feito o bloqueio do pacote de acordo com o Netfilter[23]. A figura 2 mostra o fluxo de classificação dos pacotes capturados.

[23] Módulo que fornece ao sistema operacional as funções do Firewall.

Figura 2. Fluxo de funcionamento do algoritmo



Fonte: Próprio Autor

Ao capturar um pacote é analisado o campo hash[24] do pacote em busca de informações importantes como tamanho do pacote e código de verificação de integridade. Essas informações do campo hash são comparadas com uma lista conhecida, caso o sistema encontre semelhança o pacote é marcado como sendo do protocolo BitTorrent ou como não BitTorrent.

De acordo com o autor, essa abordagem de análise de tráfego pela assinatura da camada de aplicação demostrou que seu desempenho é suficiente para identificar o tráfego, embora o trabalho possa está voltado para o protocolo BitTorrent, essa técnica pode ser facilmente modificada para outros protocolos em execução no TCP.

3. METODOLOGIA

A abordagem de pesquisa quantitativa da proposta está apoiada em técnicas de coletas de dados também quantitativas. De acordo com Neves (1996, p.01), a pesquisa quantitativa não busca expressar os sentidos dos fenômenos. Ela serve

[24] Funciona como assinatura ou identificador de uma máquina na rede.

para obter dados descritos que enumeram e medem eventos. O estudo foi desenvolvido a partir de:

1) Pesquisa bibliográfica: Os principais autores são: (WONG, 2011), (MEMON, 2014), (LIU, Y.; YANG, Y, 2013) e (BORGES, 2010);
2) Configuração da rede de teste Wi-Fi: É criada a rede Wi-Fi com a ferramenta Hostpot[25] presente no sistema operacional básico Debian 10;
3) Configuração do roteador usando Iptables: é configurado a passagem de fluxo de uma interface LAN para WAN[26] (Grande Área de Rede) no Debian 10;
4) Implementação da ferramenta no Sistema Operacional Debian 10: Nessa etapa é desenvolvido o Script com o Interpretador Python 2.7.16;
5) Teste do script: nessa etapa é colocado o cliente BitTorrent em execução em algum HOST[27] e executado o script de bloqueio de pacotes;
6) Ajustes de eventuais erros e falhas no sistema de bloqueio;

3.1. Cenários de testes

Para a comprovação de eficácia do sistema, foram feitos testes em três cenários. Estes testes mostraram parâmetros das conexões dos clientes, como velocidade de download e upload, JITTER[28], PING[29] e perda de pacote de cada usuário que não realizaram a transferência de arquivos via BitTorrent. Em cada cenário três dispositivos agiram como cliente e uma máquina como roteador Wi-Fi. O objetivo é identificar e bloquear pacotes BitTorrent que diminui a banda de outros usuários da rede. O site usado para medir a taxa de Download e Upload dos dispositivos é o SIMET. Nesse site é encontrado também dados sobre Latência bidirecional, perda de pacotes e JITTER.

- Cenário 1: Nenhum cliente BitTorrent ativo. Sistema de detecção e bloqueio de pacotes no roteador inativo.

[25] Cria rede sem fio.
[26] É a rede que cobre uma área física extensa.
[27] Qualquer dispositivo conectado na rede local.
[28] Variação da latência (atraso) na transmissão sequencial das mensagens.
[29] Medida de tempo para uma mensagem ir ao destino e voltar.

- Cenário 2: Um cliente de BitTorrent ativo. Sistema de detecção e bloqueio de pacotes no roteador ativo.
- Cenário 3: Um cliente BitTorrent ativo. Sistema de detecção e bloqueio de pacotes no roteador inativo.

4. FIREWALL DE PACOTES

O Firewall[30] é o sistema responsável por filtrar todo o tráfego de entrada e saída da rede local. Ocorre mediante a análise de regras previamente inseridas pelo administrador do mesmo.

O filtro de pacotes possui a capacidade de analisar o cabeçalho de pacotes que trafegam pela rede. Mediante essa análise é proposto o destino de um determinado pacote. A filtragem pode, então, permitir que o pacote trafegue livremente pela rede ou simplesmente é ignora por completo (NETO, 2004).

4.1. Tipos de Firewall

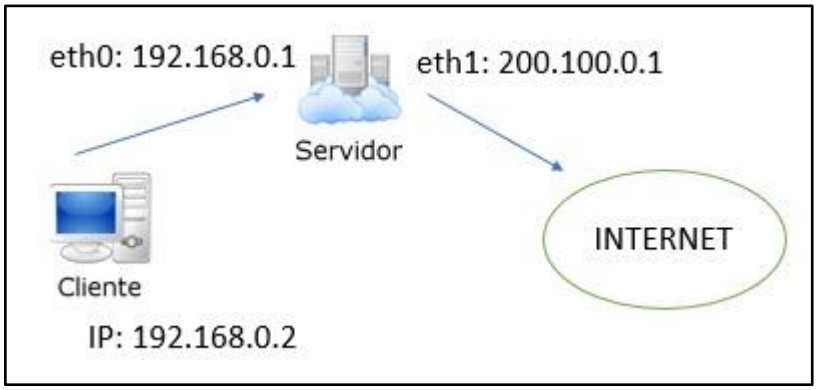
4.1.1. Firewall NAT

Um Firewall aplicado à classe NAT[31] (Tradução de Endereço de Rede), a princípio, possui o objetivo de manipular a rota de pacotes que atravessam o kernel[32] do dispositivo que possui o Firewall e faz a tradução de endereços. Isso lhe agrega ao Firewall funcionalidades como a de manipular o endereço de origem SNAT[33] (Tradução de Endereço de Origem de Rede) e destino DNAT[34] (Tradução de Endereço Destino de Rede) dos pacotes e realizar Mascaramento sobre conexões PPP[35], entre outras potencialidades. Um Firewall NAT pode, por exemplo, realizar o trabalho de um proxy de forma simples e eficiente, independente de IP fixo ou dinâmico.

[30] Aplica uma medida de segurança em um ponto na rede.
[31] Reescreve os endereços IP.
[32] Componente central do sistema operacional.
[33] Tradução de endereços IP de origem.
[34] Tradução de endereços de IP de destino.
[35] Estabelece conexão direta entre dois nós.

A tradução de endereços é o estabelecimento de conexão entre duas redes distintas. Para que isso aconteça o Firewall altera o endereço de origem do pacote que é enviado pela rede local e em seu lugar disponibiliza a rede internet seu próprio endereço, ou seja, o endereço de origem do pacote enviado pela rede local passará a ser o do host Firewall NAT. A figura 3 esquematiza a tradução de endereço pertencente as interfaces eth0 e eth1.

Figura 3. Diagrama de uma Firewall



Fonte: Próprio Autor

4.1.2. Firewall Híbrido

Um Firewall Híbrido agrega a si tanto funções de filtragem de pacotes quanto de NAT. Trata-se, da união de ambas as classes e não tão somente de uma classe isolada com propriedades próprias.

4.2. Ferramentas de Manipulação de Pacotes

4.2.1. Iptables

Iptables é uma ferramenta Front-End[36] que permite manipular as tabelas do Netfilter. É uma versão mais robusta e completa dos seus antecessores IPFWADM[37] e IPCHAINS[38], implementados no kernel 2.0 e 2.2 respectivamente (NETO, 2004).

[36] Trabalha com a aplicação que interage com o usuário.
[37] Antecessor do Iptables.
[38] Antecessor do Iptables.

O Sistema Operacional Linux utiliza-se de um recurso independente em termos de kernel para controlar e monitorar todo o tipo de fluxo de pacotes em seu sistema. Esse controle é feito pela ferramenta Netfilter.

O Netfilter contém em sua estrutura três tabelas padrões: FILTER[39], NAT e MANGLE[40]. Cada uma dessas tabelas contém regras direcionadas e seus objetivos básicos. A tabela FILTER, por exemplo, guarda todas as regras aplicadas a um Firewall filtro de pacotes, a tabela NAT as regras direcionadas a um Firewall NAT e a MANGLE as funções mais complexas de tratamento de pacotes. O comendo 1 é um exemplo de controle de fluxo que aplica prioridade máxima aos pacotes de saída da interface eth0 da porta 22.

Comando 1. Aplica prioridade

```
iptables –t mangle –A OUTPUT –o eth0 –p tcp --dport 22 –j TOS –set-tos 16
```

Fonte: Próprio Autor

A tabela FILTER apresenta três modelos de situação de fluxo, INPUT[41], OUTPUT [42] e FORWARD[43]. A tabela NAT é geralmente usada para estabelecer comunicação entre duas redes distintas. O NAT por sua vez, possui diversas utilidades, dentre elas a principal é a tradução de endereços. Essa tabela apresenta três modelos de situação de fluxo que são: PREROUTING [44] , OUTPUT e POSTROUTING[45].

A tabela MANGLE implementa alterações especiais em pacotes em um nível mais complexo. Essa tabela é capaz de alterar a prioridade de entrada e saída de um pacote baseado no tipo de serviço ao qual o pacote se destinava.

Ao analisar o fluxo de dados do kernel Linux o entendimento sobre entrada e saída de pacotes fica mais claro (NETO, 2004). Para um computador fazer parte da rede ele precisa de situações de fluxo (chains) de entrada (INPUT) e saída

[39] As regras contidas nessa tabela aceitam ou não os pacotes.
[40] Especifica ações especiais que devem ser aplicadas ao tráfego.
[41] Situação de fluxo ou chains da tabela FILTER.
[42] Situação de fluxo ou chains da tabela FILTER ou NAT.
[43] Situação de fluxo ou chains da tabela FILTER.
[44] Situação de fluxo ou chains da tabela NAT.
[45] Situação de fluxo ou chains da tabela NAT.

(OUTPUT). O redirecionamento de pacotes usa a chain FORWARD para reenviar pacotes não tratados para outra rede ou interface.

4.2.2. Função de redirecionamento de pacotes

Antes de utilizar qualquer regra de manipulação de pacotes é necessário ativar a passagem de fluxo de pacotes entre interfaces. Para uma máquina local que possui sistema operacional Linux funcionar como um Firewall que analisa pacotes entre duas interfaces, é preciso ativar o encaminhamento de pacotes através da configuração do arquivo ip_forward[46] da máquina. O comando 2 habilita a passagem de pacotes através das interfaces de rede.

Comando 2. Passagem de fluxos entre interfaces

```
]echo "1" >/proc/sys/net/ipv4/ip_forward
```

Fonte: (NETO, 2004)

4.2.3. Formas de bloqueio de pacotes

O comando 3 é um exemplo de comando Iptables que usa a chain FORWARD para descartar todo o fluxo de pacotes com origem do IP 10.0.80.32 e destino 10.0.30.4. Como a interface não foi especificada a ferramenta usa a padrão da própria máquina.

Comando 3. Bloqueio por FORWARD

```
[root@johann /]# iptables -A FORWARD -s 10.0.80.32 -d
10.0.30.4 -j DROP
```

Fonte: (NETO, 2004)

O comando 4 usa a chain INPUT. O bloqueio ocorre considerando todo o fluxo de entrada na interface de rede com protocolo TCP e direcionado a porta 80, automaticamente esses pacotes são bloqueados.

[46] Permite o redirecionamento de pacotes.

Comando 4. Bloqueio por INPUT

```
[root@johann /]# iptables -A INPUT -p tcp -dport 80 -j DROP
```

Fonte: (NETO, 2004)

4.3. A Biblioteca SCAPY

Scapy é uma Biblioteca em Python que permite ao usuário enviar, identificar, dissecar e forjar pacotes de rede. É capaz de montar pacotes de um grande número de protocolos, enviá-los na rede e capturar.

O Scapy pode lidar facilmente com tarefas mais clássicas, como tracerouting[47], sondagem, unidade testes, ataques ou descoberta de rede. Pode substituir hping[48] e arpspoof[49] e até algumas partes do Nmap[50] (BIOND, 2019).

Para capturar pacotes de qualquer interface de saída ou entrada é necessário usar a função *sniff*[51] da biblioteca Scapy. Essa função implementa o módulo promíscuo que permite capturar todo o tráfego da rede local. O método *sniff* efetua diversos tipos de filtragem de pacotes através do parâmetro FILTER. O comando 5 mostra um exemplo de filtragem de pacotes. Nesse exemplo com o parâmetro *count* é especificado a captura de dois pacotes originados do host 66.35.250.151 e pertencentes ao protocolo ICMP (Protocolo de Mensagens de Controle da Internet).

Comando 5. Função sniffer

```
sniff(filter="icmp and host 66.35.250.151", count=2)
```

Fonte: https://scapy.readthedocs.io/en/latest/usage.html

4.3.1. Instalação da ferramenta no ambiente de desenvolvimento

[47] Rastreia a rota de um pacote.
[48] Gerador e analisador de pacotes.
[49] Traduz endereços IP em endereços MAC.
[50] Software que realiza o escaneamento de portas.
[51] Captura todo o fluxo de pacotes da interface de rede

O sistema de captura e bloqueio de pacotes foi instalado no sistema operacional Debian 10. A instalação pacote de desenvolvimento é feita pelo comando mostrado na Tabela 4. É importante ressaltar que essa instalação é somente para sistemas baseado no Unix[52] como Linux, BSD e Mac OS X.

Tabela 4. Formas de instalação do Scapy

Padrão	Apenas Scapy	pip install scapy

Fonte: (BIOND, 2019)

5. SISTEMA DE BLOQUEIO DE FLUXO BITTORRENT

5.1. Apresentação

O Script escrito em Python tem o objetivo de capturar todo o tráfego da rede LAN e identificar pacotes Handshake[53] de início de comunicação dos parceiros que possuam a String especial com o nome protocolo Bittorrent.

Programas cliente BitTorrent apresentam duas formas de transferência. A primeira delas é por arquivo Torrent[54] que é um pouco mais demorada, por se comunicar com o servidor tracker[55] antes de iniciar a transferência com os parceiros. A segunda forma de transferência é por Link Magnético que consiste em transferir o arquivo entre parceiros sem necessitar do servidor tracker, visto que cada cliente BitTorrent se conecta diretamente ao parceiro através do Link Magnético (FILHO, 2013).

O bloqueio consiste em interceptar e rejeitar os pacotes Handshake da comunicação par a par e os pacotes HTTP pertencente a comunicação par a servidor das requisições GET originados da aplicação cliente BitTorrent. Os pacotes Handshak são a primeira forma de comunicação entre parceiros na rede, ele constitui em um aperto de mão logo depois que a comunicação com o servidor tracker é finalizada já as requisições GET é o pedido pelo cliente BitTorrent pela lista dos parceiros aptos a transferência de arquivo. Os comandos 1, 2 e 3 mostram o código fonte do Script de bloqueio.

[52] Sistema operativo portável de multitarefa.
[53] Aperto de mão da comunicação entre os parceiros.
[54] Extensão de arquivos utilizados pelo protocolo de transferência P2P.
[55] Servidor que contém a lista com os parceiros disponíveis para uma transferência de arquivo.

5.2. Funcionamento do Script de Bloqueio

O código fonte do Script é compacto e composto de 50 linhas escritas em Python 2.7. Na linha 1 é declarado o cabeçalho do código fonte e nas linhas de 3 a 5 as bibliotecas utilizadas. As bibliotecas são Scapy, *re*[56] e *os*[57].

A biblioteca Scapy contém funções de filtro de pacote como *haslayer*[58] que retorna valor booleano[59] caso o pacote analisado for de um protocolo especificado.

A biblioteca *re* é usada para procurar no pacote a String contendo o nome do protocolo P2P. A biblioteca *os* permite executar funções que interagem com o sistema operacional, ela possui a função *os.system*[60] que executa comandos no kernel do sistema operacional. O uso das bibliotecas é feito pelo comando *import* como mostra no código 1.

Código 1. Bibliotecas

```
1  # !/usr/bin/python
2
3  from scapy.all import *
4  import re
5  import os
6
7
```

Fonte: Próprio Autor

Com todas as bibliotecas úteis devidamente importadas no Script o primeiro comando executado é a linha 52 do código fonte 2. Nesse comando a função *sniff* que funciona em modo promíscuo[61] e intercepta todo o tráfego da rede local. Essa função recebe o método *pac_analise* e wlan0[62] como parâmetro. Além disso, a função *sniff* executa automaticamente a cada pacote capturado.

[56] Pesquisa a existência de uma sequência de caractere em uma String.
[57] Possui funções que interagem diretamente com o sistema operacional
[58] Função da biblioteca Scapy que filtra pacotes pelo tipo de Protocolo.
[59] Pode assumir verdadeiro ou falso.
[60] Executa comandos diretamente no terminal.
[61] Configuração que torna a interface de rede receptora de todos os pacotes que trafegam na rede.
[62] Interface de rede Wi-Fi.

Todos os pacotes interceptados pela função *sniff* são enviados para o método de tratamento de pacotes o *pac_analisa* para posterior análise e identificação do tipo de protocolo usado.

Código 2. Função Sniff e blokPort

```
42  # bloqueia portas
43  def blokPort(pacote):
44      ipSrc = pacote[IP].src
45      ipDst = pacote[IP].dst
46
47      # pacotes que saem da interface wlan0
48      if ipSrc == ip_local:
49          os.system('sudo iptables -I FORWARD -o wlan0 -d {} -j REJECT'.format(ipDst))
50
51  #captura pacotes que entram e saem da interface de rede
52  sniff(prn=pac_analise, iface="wlan0")
```

Fonte: Próprio Autor

A análise do método *pac_analise* se constitui em separar os pacotes TCP e UDP dentre os demais pacotes presentes na rede local e obter o cabeçalho Payload da variável *pacote* para posterior análise.

O tipo de dado pacote recebido pela função *pac_analise* na linha 13 do comando 3 implementa funções específicas da biblioteca Scapy que podem retornar valores booleano como a *haslayer* representada na linha 16 e 20. O método *haslayer* recebe como parâmetro o tipo de protocolo a ser verificado e retorna valor booleano verdadeiro caso o protocolo existir no pacote ou falso caso não.

Ainda na função *pac_analise* do código 3, especificamente na linha 17 e 21, a variável *pacote* retorna o cabeçalho Payload para ser identificado o nome do protocolo BitTorrent pela função *analisePacote*.

Código 3. Função pac_analise

```
12      # filtro de pacote
13      def pac_analise(pacote):
14          # verifica se e pacote tcp
15
16          if (pacote.haslayer(TCP)):
17              pay = pacote[TCP].payload
18              analisePacote(pay, pacote)
19          ##verifica se e pacote udp
20          if (pacote.haslayer(UDP)):
21              pay = pacote[UDP].payload
22              analisePacote(pay, pacote)
23
```

Fonte: Próprio Autor

A função *analisePacote* recebe como parâmetro na linha 31 do código 4 a variável *pacotePay* e *pacote*. A variável *pacotePay* representa o Payload retornado nas linhas 17 e 21 do código 3. Na linha 27 do código 4 a variável *pacotePay* é transformada em String pelo método *str* e colocada na variável *pay*. Na linha 30 do código 4 a função *re.search* recebe como parâmetro a String “x13BitTorrent protocol”, a variável *pay* e o inteiro *re.INGNORECASE*.

A função *re.search* na linha 34 do código 4 busca na variável *pay* a String “x13BitTorrent protocol” e é regida pelo inteiro *re.IGNORECASE* para não diferenciar letras maiúsculas de minúsculas. Essa String está presente nos pacotes handshak e nos metadados da transmissão por link magnético entre parceiros da rede P2P. A função *re.search* pode retornar para o comando de seleção *if* o valor booleano verdadeiro, caso a String do protocolo BitTorrent for encontrada, ou falso caso não. Se o valor de retorno for verdadeiro o comando da linha 31 do código 4 é executado e o pacote é enviado para o método de bloqueio *blokPort.*

Na linha 34 do código 4 o método *re.search* é usado para identificar na variável *pay* a String “BitTorrent” e “GET”. Ambas as Strings existem no início da transmissão entre o cliente BitTorrent faz e o servidor Tracker. Na linha 39 do código 4 o pacote identificado é enviado para o método *blokPort* para executar o bloqueio do IP externo que gerou o pacote.

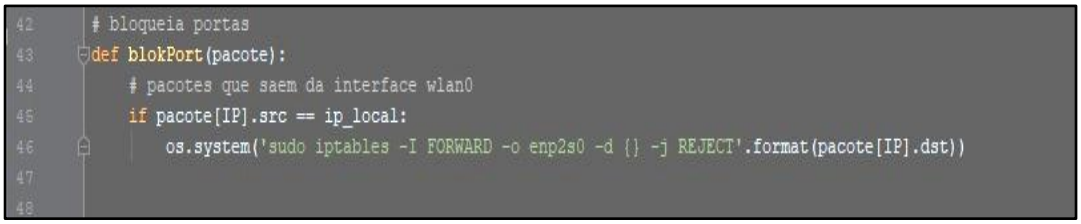
Código 4. Função analisePacote

Fonte: Próprio Autor

Na linha 43 do código 5 a função *blokPort* recebe como parâmetro a variável pacote que contém o IP externo a ser bloqueado. A linha 45 do código 5 faz a condição de comparação entre o campo IP da variável *pacote* e o *ip_local*. Caso essa condição for verdade significa que o dispositivo que gerou o pacote pertence a própria rede local e o bloqueio do pacote é efetuado.

O bloqueio do pacote é executado somente depois que a condição da linha 45 do código 5 é satisfeita. A função *os.system* recebe o comando Iptables como parâmetro para efetuar o bloqueio do pacote logo depois que a condição de comparação do comando *if* da linha 45 do código 5 for estabelecida. O bloqueio é feito na linha 46 do código 5 pelo comando Iptables com o uso da chain FORWARD para redirecionar o pacote para fora da rede local.

Código 5. Função blokPort



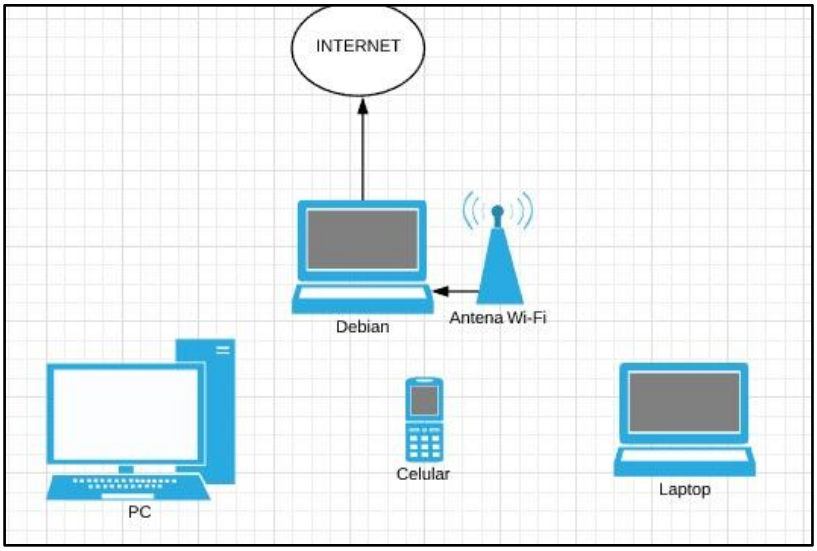
Fonte: Próprio Autor

5.3. Dispositivos Usados

O Sistema Operacional Debian 10 foi escolhido para o roteamento por ser uma distribuição GNU/Linux livre, de fácil instalação e pela compatibilidade com grande parte das arquiteturas de computadores.

A rede LAN é formada por três dispositivos conectados ao computador que compartilha sinal Wi-Fi. O Notebook que funciona como roteador Wi-Fi e o Desktop cliente estão configurados com 4 GB de memória RAM (Memória de Acesso Aleatório), 2.46 GHz de clock interno, sistema operacional Debian 10 e processador Core i3. O Smartfone tem o sistema Android 9, memória RAM de 2 GB e processador Quad-Core 1.4 GHz. O Notebook cliente possui configuração de 4 GB de memória RAM, sistema operacional Windows 10 e processador Intel Celeron Dual Core. A rede está representada na figura 4.

Figura 4. Diagrama da rede de teste



Fonte: Próprio Autor

5.4. Criação da rede Wi-Fi

A rede Wi-Fi formada apresenta a segurança padrão WPA2 (Wi-Fi Acesso Protegido II) ela foi formada usando a ferramenta de criação de redes que o Debian 10 disponibiliza. Essa ferramenta pode criar novas redes Wi-Fi a partir de uma conexão cabeada já estabelecida.

5.5. Testes

Os testes apresentam cinco simulações para cada cenário com o Script de bloqueio ativo e inativo e o cliente BitTorrent ligado e desligado para obter os

resultados taxa de download e upload, PING, JITTER e perda de pacotes. Esses parâmetros são necessários para a análise de cada cenário.

De acordo com o site SIMET, a velocidade de download se refere à rapidez com que um arquivo pode ser baixado de um serviço na Internet. A velocidade de upload se refere à rapidez com que um arquivo pode ser enviado a um serviço na Internet.

Ferraz afirma que:

> "O aplicativo PING é uma ferramenta de diagnóstico para verificar conectividade entre dois hosts em uma rede, ou seja, é um teste importante para o gerenciamento de redes de computadores. Além disso, o PING mede o tempo de atraso entre o pacote ICMP enviado e o recebido, nos dando uma ideia de como a velocidade da rede está entre o computador local e o remoto." (FERRAZ et al, 2002)

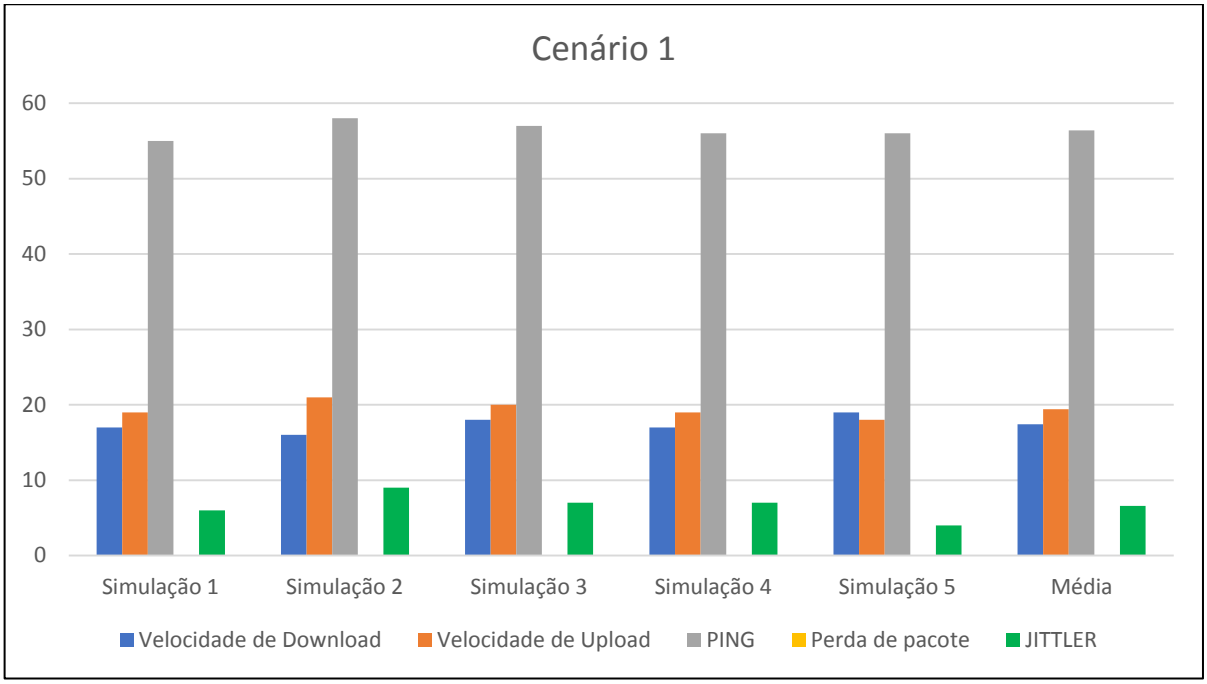
Uma perda de pacote pode ser entendida como um pacote enviado através da rede e não é recebido corretamente pelo receptor. (SANTOS et al, 2019)

Tanenbaum afirma que a variação do tempo na chegada de pacotes é:

> "A variação (isto é, o desvio padrão) nos tempos de chegada de pacotes é chamada flutuação (JITTER). Por exemplo, uma flutuação elevada, na qual alguns pacotes demoram 20 ms e outros demoram 30 ms para chegar ao destino" (TANENBAUM, 2000)

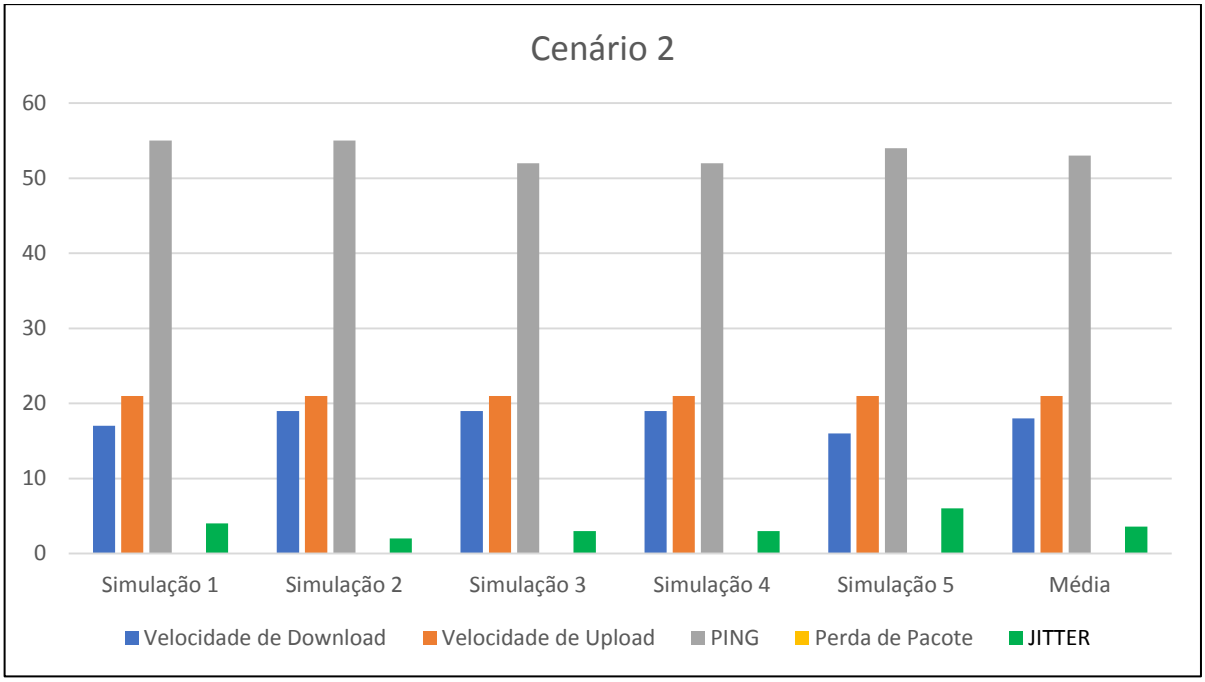
As figuras 14, 15 e 16 representam exemplos de testes com o medidor SIMET para os cenários 1, 2 e 3 respectivamente.

Gráfico 1. Script e cliente BitTorrent inativos



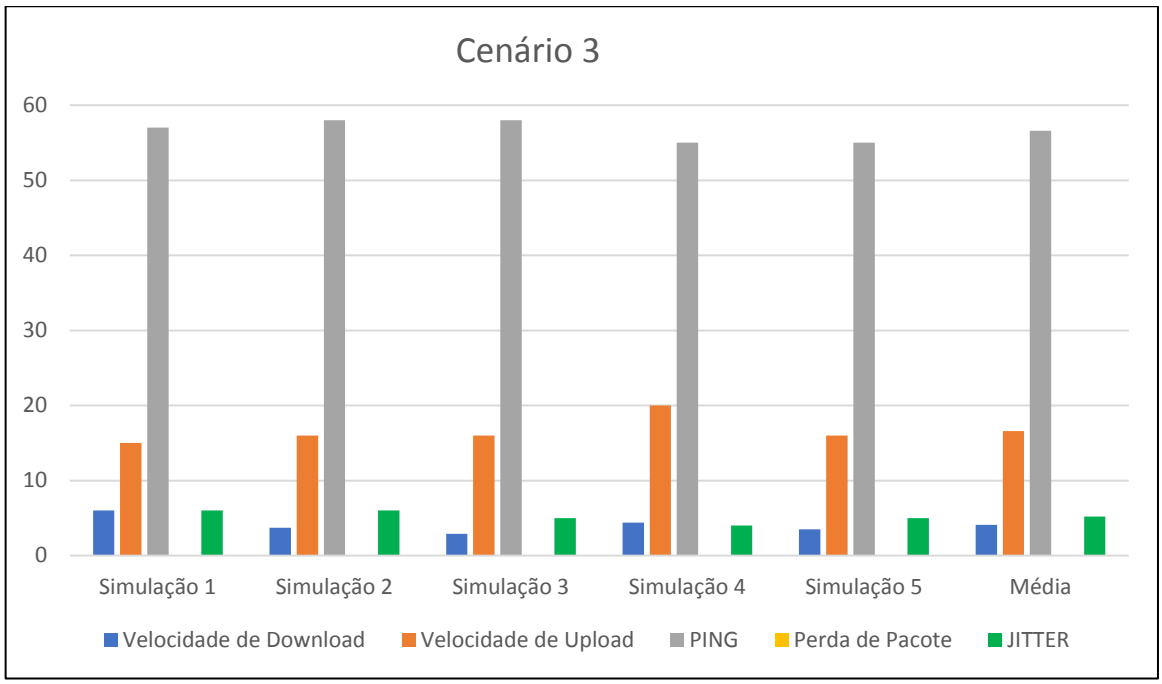
Fonte: Próprio Autor

Gráfico 2. Script e BitTorrent ativos



Fonte: Próprio Autor

Gráfico 3. Script inativo e BiTorrent ativo



Fonte: Próprio Autor

6. DISCUSSÕES E RESULTADOS

No gráfico 1 a rede não possui Script de bloqueio e nem aplicação BitTorrent em execução. As simulações feitas mostram que a rede possui uma estabilidade de download e upload entre 10 e 20 MB/s. Os dados de PING mostram que o tempo média de envio e recebimento de um pacote entre o provedor de internet e a máquina local leva em torno de 56,4 ms. O JITTER, tempo de atraso sequencial no envio de pacotes, teve em média 6,6 ms. Esses dados são aceitáveis levando em conta que a conexão entre a rede Wi-Fi local e o provedor é por fibra ótica e não houve perdas de pacotes durante a transmissão de dados.

O gráfico 2 demonstra o cenário em que o script está ativo e o cliente BitTorrent ativos. Ao efetuar o cálculo da média de todas as simulações é notório que a taxa de download e upload se manteve praticamente igual ao do cenário 1. Isso ressalva que o Script bloqueou o tráfego P2P originado do cliente BitTorrent, visto que tanto download quanto upload se mantiveram semelhantes à média encontrada no cenário 1.

No cenário 3 os três dispositivos clientes estão conectados na rede Wi-Fi do roteador. Um desses clientes está com a aplicação P2P ativada e no roteador não está rodando o Script de bloqueio BitTorrent. No gráfico 3 a média de download e upload diminuiu drasticamente comparado ao cenário 2, logo, a taxa de download e upload dos dispositivos conectados sofre diminuição, visto que o programa bloqueio do BitTorrent não está ativado no roteador.

Na figura 5, 6 e 7 é mostrado que a transferência de arquivo BitTorrent por link magnético apresentado no cenário 2. Nas figuras 5 e 6 a aplicação P2P está em execução no computador cliente e a figura 7 pertence ao dispositivo móvel.

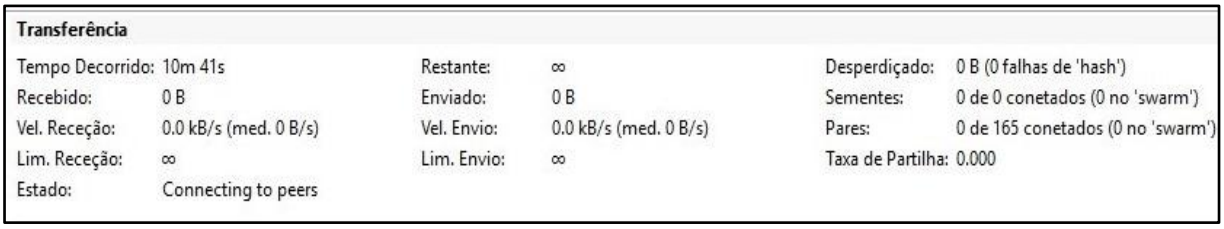
Figura 5. Aplicação uTorrent em Loop

Fonte: Próprio Autor

A figura 6 é a segunda parte da figura 5. Das informações mais relevantes que essa figura possui podemos citar o tempo decorrido do download de aproximadamente 10 minutos, a velocidade de recepção de 0,0 kB/s, o estado da conexão, a velocidade de envio de 0,0 kB/s e a quantidade de pares que fazem parte da transferência do arquivo.

O tempo decorrido de 10 minutos na figura 6 mostra que a aplicação BitTorrent não consegue estabelecer comunicação dos os pares BitTorrent e fica em loop para estabelecer a transferência de arquivos. Na figura 6 o tempo de execução da aplicação cliente, a quantidade zero de parceiros conectados e a velocidade de recepção de envio zerados mostram que não está havendo transferência de arquivo da aplicação P2P e que o Script de bloqueio de pacotes BitTorrent em execução apresenta bom resultado no bloqueio de pacotes.

Figura 6. Tempo de execução



Fonte: Próprio Autor

A figura 7 mostra a aplicação BitTorrent que também está relacionada ao cenário 2.

Figura 7. uTorrent em funcionamento no Android

Fonte: Próprio Autor

O cenário 3 simula a rede sem o Script de bloqueio em execução e a aplicação P2P em funcionamento. No gráfico 3 é mostrado que a taxa de download e upload do computador diminuiu bastante comparado ao cenário 1 e 2.

7. PESPECTIVAS FUTURAS

O sistema precisa ser melhorado em vários aspectos para aumentar a precisão de identificação do fluxo de aplicação BitTorrent. Uma das possíveis melhora para o sistema é a implementação do algoritmo DFI que melhora a identificação do protocolo P2P.

A identificação do fluxo P2P depende da String característica da aplicação BitTorrent. À medida que ocorre atualizações de aplicações P2P a String característica sofre mudanças na forma que é escrita. Para melhor resposta o

sistema de bloqueio precisa de um mecanismo que pesquise e identifique automaticamente a nova String característica à medida que a aplicação BitTorrent é lançada.

8. CONCLUSÃO

Pesquisas recentes nos mostram que aplicações P2P tem aumentado drasticamente nos últimos anos. Muitas redes locais de pequenas empresas e escolas ainda não possuem equipamentos que dão suporte ao controle de banda baseado no tipo de protocolo da camada de aplicação, deixando a desejar no controle de banda de aplicações BitTorrent.

No presente trabalho é proposto uma ferramenta para bloqueio de pacotes BitTorrent através de um programa feiro em Python e comandos Iptables que geram lentidão na rede. Alguns roteadores atuais não oferecem suporte a bloqueio de aplicações por protocolo da camada de aplicação, limitando o bloqueio somente a portas usadas e limitação de banda.

O Script de bloqueio não possibilita a captura de String de todas as versões do protocolo BitTorrent. Semelhante ao trabalho de (LIU, Y.; YANG, Y, 2013), o número de protocolos P2P detectáveis pelo Script é limitado, por isso caso o nome do protocolo contido no Payload mude, o Script fica impossibilitado de realizar a captura dos pacotes. Para que o sistema de bloqueio independa de atualizações das P2P é necessário que o código fonte do Script mude a cada nova versão dos protocolos P2P.

De acordo com (LIU, Y.; YANG, Y, 2013) a análise da String do protocolo BitTorrent, da porta e IP associados a transferência entre parceiros é necessário para detectar tráfego o P2P. O sistema de bloqueio proposto no presente trabalho apresenta a mesma lógica de funcionamento, no entanto, não faz a análise por porta da aplicação BitTorrent, somente pela String característica e IP dos parceiros conectados a aplicação cliente. Mesmo assim, se mostra eficiente na captura e bloqueio dos pacotes P2P e não efetua três verificações e sim somente duas que

são a análise do IP externo que gerou o pacote BitTorrent e a String característica da aplicação P2P.

9. BIBLIOGRAFIA

SIMET. Disponivel em: <https://beta.simet.nic.br/>. Acesso em: 30 jun. 2019.

INTELBRAS. Disponivel em: <https://www.intelbras.com/pt-br/>. Acesso em: 30 jun. 2019.

ARAÚJO, M. A. L. D.; MONTEIRO, C. D. C. Um estudo de QoS para predição da perda de pacotes em redes sem fio a partir, 18 maio 2019. 6.

BIN, L.; ZHITANG, L.; ZHANCHUN, L. Traffic Measurements of BitTorrent System Based on Netfilter, 2006. 5.

BIONDI, P. Starting Scapy, 2019. Disponivel em: <https://scapy.readthedocs.io/en/latest/usage.html>. Acesso em: 01 jun. 2019.

BORGES, L. E. **Python para Desenvolvedores**. 2ª. ed. [S.l.]: [s.n.], 2010.

CHEN, H. et al. A New Model for P2P Traffic Identification Based on DPI and DFI, 2009. 3.

COMO usar o Controle de Banda no seu roteador TP-Link? Disponivel em: <https://www.tp-link.com/br/support/faq/557/>. Acesso em: 30 jun. 2019.

DONALD. µTorrent Web Passes 1 Million Daily Active Users, 2019. Disponivel em: <http://blog.utorrent.com/2018/10/04/%C2%B5torrent-web-passes-1-million-daily-active-users/>. Acesso em: 10 jun. 2019.

FERRAZ, T. L.; ALBUQUERQUE, M. P.; ALBUQUERQUE, M. P. INTRODUÇÃO AO PING E TRACEROUTE, 2019. Disponivel em: <http://www.rederio.br/downloads/pdf/nt01002.pdf>. Acesso em: 18 maio 2019.

FILHO, P. C. H. Anatomia do BitTorrent a Ciência da Computação no Transmission, 2013. 142.

FINSTERBUSCH, M. et al. A Survey of Payload-Based Traffic, 2014. 22.

KERALAPURA, R.; NUCCI, A.; CHUAH, C.-N. Self-Learning Peer-to-Peer Traffic Classifier, 2009. 8.

KLEMM, A. et al. Characterizing the Query Behavior in Peer-to-Peer File Sharing Systems, 2004. 55.

KUROSE, J. F.; ROSS, K. W. **Redes de Computadores e a Internet uma abordagem top-down**. 6ª. ed. [S.l.]: [s.n.], 2014. 658 p.

LE, T. M.; BUT, J. Bittorrent traffic classification, 2009. 8.

LIU, Y.; YANG, Y. Analysis of P2P Traffic Identification Methods, 2013. 4.

MEMON, Q. A. **Distributed Networks intelligence, Security and Applications**. [S.l.]: [s.n.], 2014.

NETO, U. **Dominando Linux Firewall Iptables**. [S.l.]: [s.n.].

SANTOS, E. L. D. et al. Analise de Perda de Pacotes em Sistema de Controle em Rede sem fio com Aplicac¸ao de kalman, 2017. 5.

TANENBAUM, A. S. **Computer Networks**. 4ª. ed. [S.l.]: Editora Campus, 2019.

WONG, R. BitTorrent Traffic Detection with Deep Packet, 2011. 49.