INSTITUTO FEDERAL DE SANTA CATARINA

GIOVANNI SALVATORE DE ALMEIDA CURCURUTO

Plataforma para automatização de testes com QoS em Switches Ethernet

São José - SC

Julho/2019

PLATAFORMA PARA AUTOMATIZAÇÃO DE TESTES COM QOS EM SWITCHES ETHERNET

Monografia apresentada à Coordenação do Curso Superior em Sistemas de Telecomunicações do Instituto Federal de Santa Catarina para a obtenção do diploma Tecnólogo em Sistemas de Telecomunicações.

Orientador: Prof. Juliana Camilo Inácio, Dra

São José - SC

Julho/2019

Giovanni Salvatore de Almeida Curcuruto
Plataforma para automatização de testes com QoS em Switches Ethernet/ Giovanni Salvatore de Almeida Curcuruto. – São José - SC, Julho/2019-
60 p. : il. (algumas color.) ; 30 cm.

Orientador: Prof. Juliana Camilo Inácio, Dra

Monografia (Graduação) – Instituto Federal de Santa Catarina – IFSC
Campus São José
Sistemas de Telecomunicações, Julho/2019.

1. Palavra-chave1. 2. Palavra-chave2. 2. Palavra-chave3. I. Orientador. II. Instituto Federal de Santa Catarina. III. Campus São José. IV. Título

GIOVANNI SALVATORE DE ALMEIDA CURCURUTO

PLATAFORMA PARA AUTOMATIZAÇÃO DE TESTES COM QOS EM SWITCHES ETHERNET

Este trabalho foi julgado adequado para obtenção do título de Tecnólogo em Sistemas de Telecomunicações, pelo Instituto Federal de Educação, Ciência e Tecnologia de Santa Catarina, e aprovado na sua forma final pela comissão avaliadora abaixo indicada.

São José - SC, 01 de julho de 2019:

Prof. Juliana Camilo Inácio, Dra.
Orientador
Instituto Federal de Santa Catarina

Prof. Marcelo Maia Sobral, Dr.
Instituto Federal de Santa Catarina

Prof. Ederson Torresini, Me.
Instituto Federal de Santa Catarina

A todos aqueles que de alguma forma estiveram e estão próximos de mim, fazendo esta vida valer cada vez mais a pena.

AGRADECIMENTOS

Os agradecimentos principalmente irão para minha família como um todo que me deu forças e ânimo para chegar neste ponto. Também agradeço a muitos dos meus amigos que também me ajudaram me apoiando em diversos momentos desta trajetória.

Gostaria de agradecer também na área acadêmia a todos os professores do Instituto Federal de Santa Catarina pelo conhecimento a mim fornecido. Em especial gostaria de agradecer imensamente ao Professor Sandro Carlos Lima, pelo conhecimentos na área de telefonia, aos Professores Ederson Torresini, Tiago Semprebom e Eraldo Silveira e Silva, na área de redes, aos Professores Roberto Wanderley da Nóbrega, Saul Silva Caetano e Deise Monquelate Arndt, na área de Sinais e Sistemas e a professora e orientadora Juliana Camilo Inácio pela insistência, paciência e direcionamentos que tornaram isso possível.

"Para se ter sucesso, é necessário amar de verdade o que se faz.
Caso contrário, levando em conta apenas
o lado racional, você simplesmente desiste.
É o que acontece com a maioria das pessoas."
(Steve Jobs)

RESUMO

O objetivo do trabalho é desenvolver uma ferramenta capaz de realizar testes em *switches* de maneira semi automática para verificar seu comportamento de acordo com a norma IEEE 802.1p. Neste trabalho realizamos um estudo sobre a camada de enlace, focando em qualidade de serviço (QoS, do inglês *quality of service*), política de filas e parâmetro de prioridade. Após o estudo, foram realizados diversos ensaios e a partir destes resultados foram estudadas formas de automatização deste processo, visando uma ferramenta de baixo custo e com o mínimo de software externo necessário. Existem equipamentos completos que realizam esses testes, porém os existentes possuem custos muito altos. Mediante as alternativas gratuitas, com as quais não seria possível realizar uma automatização eficiente, verificamos que seria mais viável o desenvolvimento de uma ferramenta utilizando *Python* e os recursos do terminal do Linux. Após o desenvolvimento da ferramenta, foram realizados ensaios com a ferramenta proposta em *switches* distintos, e comprovando suas características de filas e prioridades também distintas através de seus resultados. Com a ferramenta proposta e operando uma "semi automatização", com poucas informações fornecidas pelo usuário, são mostradas informações sobre o tráfego recebido pelo *switch*.

Palavras-chave: Switch. Automatização. Qualidade de Serviço. Camada de enlace. Filas. Prioridades

ABSTRACT

The objective of this work is to develop a tool capable of semi-automatic testing of textit switches to verify its behavior according to IEEE 802.1p standard. In this paper we carry out a study about the link layer, focusing on Quality of Service (QoS), queuing policy and priority parameter. After the study, several tests were performed and from these results ways of automating this process were studied, aiming at a low cost tool and with the minimum external software needed. There are complete equipment that perform these tests, but existing ones have very high costs. By the free alternatives, with which it would not be possible to perform an efficient automation, we found that it would be more feasible to develop a tool using textit Python and the features of the Linux terminal. After the development of the tool, tests were performed with the proposed tool in different textit switches, and proving its queuing characteristics and also distinct priorities through its results. With the proposed tool and operating a "semi automation", with little information provided by the user, information about the traffic received by the textit switch is shown.

Keywords: Switch. Automation. Quality of Service, Link Layer. Queue. Priorities

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

ARP Adress Resolution Protocol

CRC Cyclic Redundancy Check

CFI Canonical Format Indicator

CLI Command-line interface

diffserv Differentiated Services

HDLC High Level Data Link Control

intserv Integrated Services

ICMP Internet Control Message Protocol

LAN Local Area Network

MAC Media Access Control

MS MAC Service

OSI Open System Interconnection

PPP Point-to-Point

QoS Quality of Service

TCP Transmission Control Protocol

TCI Tag Control Information

TPID Tag Protocol Identifier

VID VLAN Identifier

VoIP Voice over Internet Protocol

VLAN Virtual LAN

WFQ Weighted Fair Queuing

SUMÁRIO

1 INTRODUÇÃO

Existem diversos tipos de *switches* disponíveis no mercado com suas características específicas. Dentre as diversas características do *switch*, existem normas onde nela é descrito o funcionamento detalhado. O equipamento *switch* é usado por diversos tipos de empresas de grande, médio e pequeno porte. Em muitos casos o produto é utilizado de maneira *plug and play*, não tornando obrigatório o uso de configurações. Existem dezenas de modelos no mercado, cada um com suas características para cada tipo de aplicação.

Em redes multimídia o uso de aplicações de voz sobre IP (VoIP, do inglês *Voice over Internet Protocol*) é muito comum e em diversas situações é preciso uma priorização ou reserva de recurso na rede. Para isso é necessário que possua um equipamento que consiga provar essa priorização ou reserva de recurso com a maior qualidade possível.

Neste trabalho está sendo proposta uma ferramenta capaz de entregar informações sobre o equipamento *switch*. Deste modo é possível avaliar se o equipamento será útil para determinada aplicação escolhida pelo gestor da rede, evitando possíveis gastos desnecessários ocasionados por uma aquisição equivocada.

O padrão utilizado na aplicação deste trabalho foi o IEEE 802.1p (do inglês *Institute of Electrical and Electronics Engineers*). Essa norma apresenta alguns requisitos para que haja qualidade de serviço (QoS, do inglês *quality of service*) em camada de enlace, com diversos parâmetros. Porém, na aplicação desenvolvida foi utilizado o parâmetro *priority* do quadro Ethernet (a norma IEEE 802.1p está incorporada na norma IEEE 802.1Q).

Dentro do quadro Ethernet existe um campo chamado de *TAG*, o qual é composto por diversos parâmetros, e seu uso mais comum é o uso de rede local virtual (VLAN, do inglês *virtual local area network*). No quadro Ethernet foi realizado uma modificação para inserir alguns octetos a mais, deste modo podendo controlar a questão de prioridade e a questão de VLANs. Utilizando o parâmetro de prioridade na *TAG* dentro da norma IEEE 802.1Q-2018, é possível definir as prioridades no quadro. No *switch*, se compatível com a norma, é encaminhado o quadro para uma fila na qual a prioridade estiver decretada.

As filas possuem um papel importante, até mais importante do que a prioridade, pois sem as filas do *switch*, todos os quadros possuem o mesmo peso com relação aos demais (se o algoritmo de enfileiramento não estiver configurado). Dependendo do algoritmo utilizado, podemos realizar uma priorização por filas e desta maneira é possível perceber com clareza a variação proporcionada por filas diferentes e prioridades diferentes.

Com a ferramenta proposta neste trabalho é possível entender melhor o funcionamento das filas e prioridades injetando um grande tráfego no equipamento e provocando a necessidade de ocorrer uma priorização na saída do *switch* para o seu receptor.

1.1 Objetivos

Este trabalho tem como objetivo principal implementar uma solução semi automatizada para que se consiga verificar se um *switch* está em conformidade com a norma IEEE 802.1Q/p, com relação ao seu uso de filas e suas prioridades.

2 FUNDAMENTAÇÃO TEÓRICA

Neste capítulo será apresentada uma descrição sobre *switches*, focando em suas características relacionadas ao QoS. Abaixo serão descritos os padrões IEEE 802.1p, IEEE 802.1Q, e outros itens relacionados ao funcionamento da camada de enlace.

2.1 O *switch*

O *switch* é um equipamento que interliga uma rede de computadores, e é amplamente utilizado para diversos tipos de aplicações em todo o mundo. Generalizando, podemos dizer que em grande parte dos casos, o uso mais comum de rede local (LAN, do inglês *local area network*) seria utilizando um *switch*. As interfaces de controle do *switch* permite que o processador consiga controlar o *hardware* da melhor maneira possível. Esse tipo de interface é vital para que seja feito um controle eficiente de portas utilizando algum tipo de protocolo (como o STP).

Segundo Kurose e Ross (2010), o modelo de interconexão de sistemas abertos (OSI, do inglês Open System Interconnection) é um dos métodos de divisão que conceitualmente consegue separar as atribuições de cada camada, fazendo com que de maneira didática, cada camada consiga-se comunicar entre si, de uma maneira bem homogênea. As camadas do modelo OSI são: aplicação, apresentação, sessão, transporte, rede, enlace e físico. Outro modelo de divisão onde é possível separar as atribuições por camadas, que é considerado o padrão, é chamado de modelo de protocolo de controle de transmissão (TCP, do inglês *Transmission Control Protocol*), onde é divido em camadas de aplicação, transporte, rede, enlace e física.

De acordo com o Kurose e Ross (2010) cada camada depende uma da outra para que seja entregue a informação. Um exemplo disso seria a maneira que a camada de rede necessita da camada de enlace, pois necessita das informações de MAC e outras que são necessárias para compor o pacote. Entre as três camadas iniciais (camada física, enlace e rede) estão contidos os principais protocolos de controle e gerenciamento do *switch* tal como STP.

Conforme Seifert e Edwards (2008) se um *switch* estiver enquadrado na camada de rede (dependendo do fabricante e modelo), esse equipamento pode estar sendo utilizado primordialmente em camada de enlace com algumas funções extras (como roteamento). Esse tipo de equipamento não será o foco do trabalho, mas sim equipamentos puramente camada de enlace.

A camada de enlace, de acordo com Kurose e Ross (2010) é uma das sete camadas do modelo OSI, que tem a responsabilidade principal de controlar a transmissão e a recepção de cada quadro. Além das funções que são respectivas há transmissão e recepção, outros protocolos/padrões importantes a serem citados, que necessitam desta camada em especifico, são: ponto a ponto (PPP, do inglês *point-to-point*) e o protocolo de controle de enlaces de alto nível (HDLC, do inglês *high-level data link Control*).

Em conformidade com Kurose e Ross (2010), o quadro Ethernet pode ser dividido em diversas partes conforme mostrado na Figura 1, no qual é divido em preâmbulo, endereço de destino, endereço de origem, tipo, dados e verificação cíclica de redundância (CRC, do inglês *cyclic redundancy check*).

Conforme citado na Figura 1, de acordo com Kurose e Ross (2010) o quadro Ethernet possui os endereços de controle de acesso de mídia (MAC, do inglês *Media Acess Control*) de destino e de origem. Além dos campos de **endereço de destino** e **endereço de origem** , no quadro Ethernet são armazenados

Figura 1 – Campos do quadro Ethernet (KUROSE; ROSS, 2010)

outras diversas informações ou parâmetros (como Preâmbulo, TAG, e o *payload*). No Endereço MAC de origem, é armazenado o MAC da interface de rede que realiza a transmissão e o endereço MAC de destino, é armazenado o MAC da interface de destino. Se o quadro for encaminhado para uma interface na rede de maneira ponto a ponto, e essa interface possuir um endereço MAC diferente, ele poderá ser descartado.

Segundo Kurose e Ross (2010), o MAC também conhecido como Endereço de LAN ou Endereço físico, é na realidade o endereço que a camada de enlace utiliza, sendo que na camada de rede o endereçamento respectivo seria o IP. O endereço MAC possui 6 bytes de comprimento e na literatura citada, Kurose e Ross (2010) relata que não existem dois (ou mais) endereços MAC's iguais, mesmo entre empresas fabricantes diferentes. Ele relata que como o endereçamento MAC possui um comprimento de 2^{48} as empresas compram uma faixa de 2^{24} de endereços fazendo com que o restante consiga variar naquela faixa dada pelo órgão IEEE.

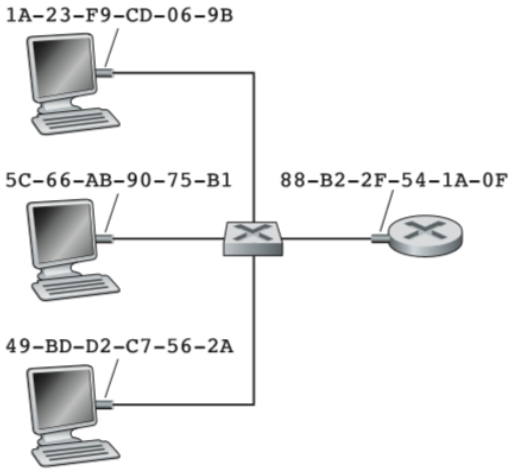

Figura 2 – Topologia com diversas as interfaces de rede e seus MAC's.(KUROSE; ROSS, 2010)

Cada equipamento que é conectado a uma rede, possui um endereço MAC associado, conforme exemplo mostrado na Figura 2, e no caso das interfaces do *switch* não é necessário. Como o *switch* é um equipamento que faz a união de diversos equipamentos de rede, considerando que o equipamento não faz roteamento, ele não necessita de um endereço MAC. O equipamento possui meios de ler o quadro Ethernet, e encaminhar localmente o quadro para sua respectiva interface, sendo quase uma comunicação ponto a ponto.

De acordo com Seifert e Edwards (2008) o MAC e as portas compreendem as funções tradicionais da interface de rede. No lado da recepção, esse dispositivo lê os sinais elétricos ou ópticos do meio e decodifica-os em um bit ou fluxo de bytes conforme apropriado para a tecnologia específica em uso. Quando recebido o quadro Ethernet (contendo todas as informações já citadas) ele irá verificar o CRC, e se verificado que está inválido ou fora de sequência, será descartado. Em momentos que não é possível ler todos os endereços de destino dos quadros recebidos, o dispositivo estará recebendo um quadro (contendo os MACs) e precisará guardar os quadros em um *buffer*, antes de terminar o encaminhamento.

É necessário que seja feito um *buffer* local para armazenar os quadros recebidos, e conforme Seifert e Edwards (2008), existem duas maneiras de realizar esse armazenamento utilizando uma memória compartilhada e outra uma memória dedicada. Quando é comentado sobre memória dedicada, ele refere-se a uma memória dedicada para cada porta ou grupo de portas, desde modo é armazenado localmente, porém já repassado para a estrutura responsável que irá validar o quadro. Quando é comentado sobre uma memória compartilhada, é apenas um único *buffer*, porém comum para todos, com o objetivo de evitar duplicatas de quadro. Neste caso é feito uma única estrutura de fila, no qual vai se ajustando e encaminhando porta a porta (tanto para o recebimento quanto envio).

2.2 *Switches* não-gerenciáveis e gerenciáveis

Existem diversos *switches* no mercado atualmente e todos eles realizam as mesmas funções básicas de comutação de quadros Ethernet, porém alguns deles conseguem desempenhar e possuir algumas outras funções. Todos os *switches* estão relacionados diretamente com a camada de enlace, existindo equipamentos específicos que implementam funções de outras camadas.

De acordo com Kurose e Ross (2010) grande maioria dos equipamentos (cabeados) que utilizam o protocolo IP, conseguem operar na camada de enlace utilizando o padrão Ethernet através do MAC. Deste modo, conforme mostrado na Figura 3, é possível confirmar que vários tipos de equipamentos de camadas acima (rede, transporte, etc), em conexão com o *switch*, necessitam compreender o padrão Ethernet da camada de enlace.

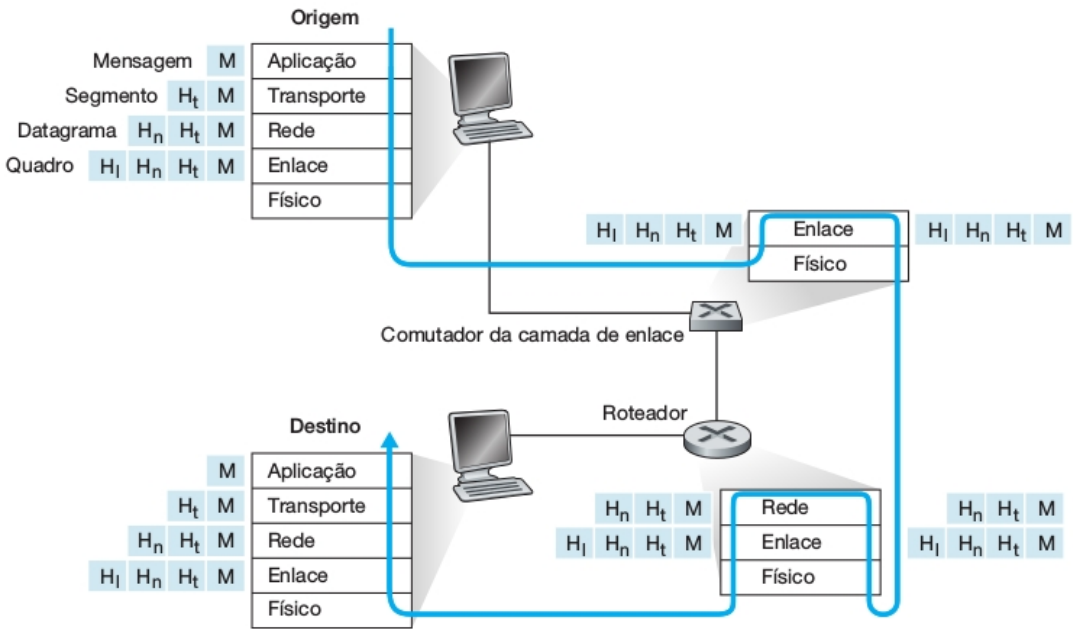

Figura 3 – Modelo OSI exemplificado em cada segmento da rede. (KUROSE; ROSS, 2010)

Na Figura 3 o *switch* é citado como **comutador da camada de enlace**, sendo que ele está operando somente nas duas primeiras camadas. Podemos ver os terminais de destino e origem que implementam quase todas as camadas e existe um terceiro tipo de equipamento, onde ele implementa a camada de rede chamado de roteador. Internamente no roteador, existem diversas facilidades de gerenciamento e principalmente de repasse de dados. De maneira simplificada podemos dizer que o roteador irá receber um pacote, verificará o seu destinatário (se possuir o destinatário dentro de suas tabelas de roteamento), e estará encaminhando para o dispositivo. Existem casos em que um *switch*

pode implementar algumas funções da camada de rede, fazendo também assim as funções similares a um roteador.

Conforme (CISCO, 2018), simplificando os *switches* criam redes (interligação redes), porém os roteadores fazem a conexão das redes (encaminhamento dos quadros/pacotes entre redes. Podemos dizer que *switches* não cuidam de nada relacionado ao roteamento de dados, buscando os melhores caminhos, pois quem fica com essa responsabilidade, seria o roteador. O Roteador possui diversos protocolos de encaminhamento e roteamento fazendo com que os dados consigam chegar entre os pontos da maneira mais eficiente.

2.3 Qualidade de Serviço

Qualidade de serviço ou QoS, é conceitualmente um ou mais parâmetros que visam a melhor entrega de determinados serviços em uma rede, oferecendo uma boa garantia de desempenho. Muito utilizado com aplicações multimídia, conforme Kurose e Ross (2010) com as garantias do QoS por conexão, é possível reservar uma largura de banda fim a fim, e assim conseguir o desempenho esperado. Usando o exemplo de um sistema VoIP, que necessita de uma determinada qualidade e priorização perante os demais serviços da rede.

Podem existir diversas maneiras de QoS dentro de uma rede, dentre eles podemos citar os mais conhecidos: serviços diferenciados (diffserv, do inglês *Differentiated Services*), serviços integrados (intserv, do inglês *Integrated Services*), IEEE 802.1p. Dentre os citados, alguns são utilizados na camada de rede e os outros utilizados na camada de enlace, manipulando os pacotes IP e os quadros Ethernet, respectivamente.

Indiferentemente do protocolo utilizado, segundo Tanenbaum et al. (2003), existem alguns requisitos/parâmetros básicos para realizar a função de QoS, dos quais são citados a confiabilidade, retardo, flutuação e largura de banda, parâmetros esses que são considerados na aplicação. De acordo com Tanenbaum et al. (2003) fluxo seria uma sequência de pacotes desde uma origem até um destino. Esses parâmetros, comentados a seguir, definem o QoS exigido pelo fluxo:

- **Confiabilidade**: De acordo com Tanenbaum et al. (2003) a confiabilidade será alcançada calculando-se o total de verificações em cada pacote e verificando isso em seu destino:

- **Retardo**: Aplicações restritas a retardo, são aplicações dos tipos interativas e mais próximas de ser em tempo real. Conforme Tanenbaum et al. (2003), são exemplos disso videoconferência e telefonia, porque são aplicações em tempo real, diferentemente de aplicações de mídia armazenada:

- **Flutuação**: Aplicações restritas a flutuação, podem ser definidas como aplicações que não podem sofrer variações entre pacotes recebidos. Diferentemente das aplicações com restrições a retardo, e segundo Tanenbaum et al. (2003) esse tipo de aplicação pode ocorrer uma demora mais considerável entre o recebimento entre pacotes, porém não é admitido pacotes com tempos de recebimento muito diferentes:

- **Largura de banda**: As aplicações restritas a largura de banda é possível comparar diversas aplicações, porém em especial vídeo e e-mail. Enquanto um vídeo é necessário uma largura de banda muito grande (pois é necessário muita informação) em contrapartida os e-mails são aplicações muito mais leves exigindo pouca banda(TANENBAUM et al., 2003).

2.3.1 Padrao IEEE 802.1p

O IEEE 802.1p em conjunto com outras normas e padrões conhecidos formam a IEEE 802.1Q. Estes padrões fazem parte das varias normas do IEEE e no caso do padrão IEEE 802.1Q é fornecido soluções para VLANs e *Bridges* MAC. Dentro da norma IEEE 802.1Q e IEEE 802.1aq existem citações da norma IEEE 802.1p, porém foi incorporado inteiramente na norma IEEE 802.1Q-2018.

De acordo com (DUARTE; BICUDO, 2002), o padrão IEEE 802.1p tem como principal objetivo permitir e definir maneiras de realizar o encaminhamento de tráfego de maneira expressa, por meio de níveis de prioridades dentro do quadro. De acordo com Duarte e Bicudo (2002) também está dentro da norma a questão de definição de filtros para utilização dinâmica de grupos de endereços MAC.

Dentre os principais parâmetros para se prover QoS, de acordo com Duarte e Bicudo (2002), estão a disponibilidade de serviço MAC, perda de quadro, desordenamento dos quadros, atraso de transmissão, tempo de vida do quadro, e taxa de erros não detectados, tamanho máximo de unidade de serviços, priorização e vazão.

De acordo com Duarte e Bicudo (2002), a norma IEEE 802.1p consegue realizar a priorização de diversos quadros, porém como já foi visto, o campo MAC é um parâmetro da camada de enlace, dentro do quadro Ethernet (podendo ser observado na Figura 1) e o quadro Ethernet não possui um campo específico para priorização. Dentro do quadro Ethernet foi necessário uma extensão do quadro para 4 octetos, porém com isso ocasionando uma redução de 4 octetos do campo *payload*. Essa alteração foi efetuada para a implementação da norma IEEE 802.1Q (utilizando o campo TAG, também referentes a VLAN), para que não se alterasse o tamanho total do quadro, sendo de tamanho máximo de aproximadamente 1500 bytes de dados, mais 18 octetos de cabeçalho e o CRC. Existem casos onde o *payload* tem aproximadamente 9000 bytes, porem de maneira geral, o equipamento precisa possuir essa função, sendo um padrão mais recente em comparação aos 1500 bytes de *payload*.

De acordo Tanenbaum et al. (2003) em meados de 1995, foi realizado uma discussão para definir as alterações do quadro Ethernet e o formato novo foi publicado na norma IEEE 802.1Q. Padrão no qual é utilizado até os dias de hoje. O quadro não precisou ser alterado a ponto do equipamento existente ser descartado, mas sim adaptado.

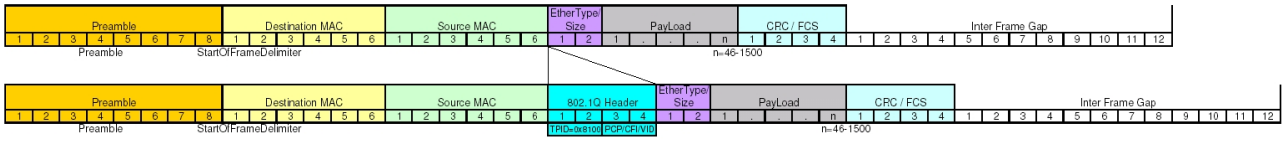

Figura 4 – Quadro Ethernet antes e após a alteração da estrutura (STAFFORD, 2019)

Chamado de TAG, é o campo onde é configurado diversas informações sobre a VLAN. Conforme mostra a Figura 4, além do identificador de protocolo TAG (TPID, do inglês *Tag Protocol Identifier*), existem outros 3 campos: Identificador de VLAN (VID, do inglês *VLAN Identifier*), o Indicador de formato canônico (CFI, do inglês *Canonical Format Indicator*) e o campo de prioridade. Conforme comentado acima, o campo *TAG* possui 4 bytes, onde 2 bytes ficam exclusivamente para o *VLAN Protocol ID*, enquanto os outros 2 bytes são distribuídos para o restante.

De acordo com Duarte e Bicudo (2002), o campo TPID serve para sinalizar que os 2 bytes seguintes possuem informações que devem ser consideradas. O campo dos 2 bytes restantes (contendo o restante dos campos), é chamado de controle de informação do TAG (TCI, do inglês *Tag Control Informatio*).

Dentro do TCI possuímos o campo CFI, e de acordo com Tanenbaum et al. (2003), esse campo

indica que está contido dentro do quadro Ethernet, um quadro congelado. Esse quadro fica no aguardo esperando encontrar a LAN de destino, enquanto está sendo transportado. Segundo Duarte e Bicudo (2002) é utilizado no método de acesso ao meio roteados por FDDI/Token-Ring (outro protocolo, sendo reconhecido como camada física) para sinalizar a ordem da informação de endereço encapsulado no quadro.

Também dentro do TCI possuímos o VID, e de acordo com Duarte e Bicudo (2002) é o campo onde é identificado, de qual VLAN o pacote pertence. Muito importante o campo para utilização do protocolo VLAN IEEE 802.1Q, porém não é obrigatório o uso deste campo se não estiver utilizando VLAN, mas sim outras propriedades da TAG.

O campo de prioridade dentro do TCI, de acordo com Duarte e Bicudo (2002), é utilizado para carregar a informação de prioridade através da rede. O campo de prioridade, possui 3 bits, deste modo conseguindo representar 8 níveis de prioridades diferentes (de 0 até 7). Este campo será citado mais baixo quando comentado sobre o campo prioridade.

Apesar da norma IEEE 802.1p depender do campo TAG para sua utilização e implementação, e o campo TAG é normatizado pela norma IEEE 802.1Q, conforme já citado anteriormente, não é necessário a implementação da TAG (tendo em vista a não utilização da VLAN por exemplo). De acordo com Duarte e Bicudo (2002), a norma IEEE 802.1p consegue realizar a manipulação dos três bits do campo de prioridade da norma IEEE 802.1Q.

Os quadros marcados com a TAG de prioridade tem a priorização em relação aos demais. De acordo com Duarte e Bicudo (2002), a única informação que é considerada para a priorização é o campo de prioridade, sendo que o restante das informações do quadro não são relevantes para o protocolo. É necessário que o equipamento *switch* consiga compreender sobre o QoS, de outro modo ele não conseguirá fazer a diferenciação dos pacotes (o equipamento também necessitará compreender a norma IEEE 802.1Q).

A norma IEEE 802.1p pontua alguns itens importantes para se prover o QoS na camada de enlace (nível MAC):

1. **Disponibilidade de Serviço**: Disponibilidade do serviço (do inglês *service availability* de acordo com Duarte e Bicudo (2002), é a razão do tempo total em que o Serviço MAC (do inglês MAC Service - MS) ficou disponível no *switch*. De acordo com Society (2014) dependendo da operação na *bridge* o desempenho da rede pode aumentar, porém também pode ocasionar em efeitos colaterais como a diminuição do desempenho. De acordo com Duarte e Bicudo (2002), a disponibilidade do serviço está relacionada também ao equipamento a infraestruturada utilizada, sendo que casos de falhas nos equipamentos, mal contato nos conectores, ou falhas nos *switches* resultaria em perdas de quadros, corrompimentos ou filtragem indevida. Ainda de acordo com Duarte e Bicudo (2002), é possível dizer que uma auto-configuração do *switch*, conseguiria fazer com que muitas destas situações citadas acima fossem resolvidas.

2. **Perda de quadro**: Na camada de enlace não é possível garantia de entregas, tendo em vista que podem ocorrer diversos problemas no momento da entrega (ou na tentativa de entrega). De acordo com Duarte e Bicudo (2002), existe uma alta probabilidade entrega do quadro e conforme Society (2014), existe uma perda porém pode ser miníma. Dentre os problemas que podem ser citados, de acordo com Society (2014) um deles é o problema físico. O quadro pode se corromper no meio físico fazendo com que não seja possível realizar a entrega. Um outro problema que pode ocorrer, de acordo com Society (2014), é com o *buffer*, onde é excedido a capacidade máxima de armazenamento, fazendo que com os quadros sejam descartados. Uma outra causa possível de perda de quadros, seria referente ao *lifetime*, que deverá ser explicado mais abaixo.

3. **Duplicação de quadro** - Em casos onde existe duplicação de quadro, estamos falando de situações onde existem múltiplas origens e múltiplos destinos, onde não é possível essa situação. Não seria possível essa situação porque precisaria existir retransmissão de quadros, porem não existe a retransmissão na camada de enlace (SOCIETY, 2014).

4. **Desordenamento de quadro** - O desordenamento de quadro pode ocorrer em momento que ocorra falhas de hardware, atraso considerável na rede onde o quadro fica congestionado na rede e entregue fora de ordem (SOCIETY, 2014).

5. **Atraso de transmissão**: De acordo com Society (2014), é o tempo previsto para o envio e recepção do quadro.

6. **Tempo de vida do quadro**: Neste caso refere-se ao tempo máximo em que se precisará das informações do quadro, sendo utilizado por outras camadas superiores e armazenados no *buffer*. De acordo com Society (2014) para garantir que o quadro consiga essa entrega e esse tempo, será necessário descartar pacotes no *switch*. De acordo com Duarte e Bicudo (2002), após esse tempo, o *switch* pode realizar o descarte do quadro.

7. **Tamanho máximo de unidade de serviços**: Segundo Society (2014), deve ser menor que a taxa na qual a porta do *switch* suporta, pois pode variar muito entre fabricantes de equipamentos.

8. **Taxa de Erros Não Detectados**: Esse campo permite, de acordo com Duarte e Bicudo (2002) que em seu destinatário consiga-se verificar o CRC e se estiver incorreto, deverá descarta o quadro.

9. **Prioridade**: A priorização é algo de extrema importância. Como já foi citado anteriormente, o campo prioridade é o local onde é realizado a validação do serviço utilizado e nele são inseridas as informações sobre o tipo de serviço que irá ser utilizado. O *switch* pode mapear o campo prioridade em uma ou mais classes de serviço (CoS), e os *switches* por sua vez podem suportar o encaminhamento expresso, de acordo com Duarte e Bicudo (2002).

2.4 Algoritmos de enfileiramento

De acordo com Seifert e Edwards (2008) existem diversos tipos de algoritmos/políticas de enfileiramento de quadros. Ainda de acordo com Seifert e Edwards (2008), esses enfileiramentos podem ser relacionado ao comportamento do *switch* e a prioridade do quadro nele inserido. Os mais usados em nossos experimentos são: *Strict Priority* e *Weighted Fair Queuing*. Ambos podem ser utilizados individualmente acessando as informações de prioridade encaminhadas no quadro Ethernet.

Quando um quadro é encaminhado em um *switch*, de acordo com Seifert e Edwards (2008), ele passa por um controle de tráfego com o objetivo de realizar uma priorização, quadro perante quadro. Essas políticas citadas acima, fazem com que exista esse controle e logo exista QoS.

Os algoritmos de enfileiramento de acordo com Cisco (2018), estão presente em *switches* que implementem QoS, mais especificamente o IEEE 802.1p. Deste modo fazendo a distribuição mais eficaz dos quadros. De acordo com Seifert e Edwards (2008), o enfileiramento ocorre por porta, e com isso conseguindo gerenciar exatamente os quadros de acordo com seu algoritmo específico.

2.4.1 Prioridade estrita

Prioridade estrita (do inglês *Strict Priority*) é um tipo de algoritmo que faz que o *switch* analise cada pacote recebido com sua prioridade estabelecida, e sempre encaminha primeiro o pacote com a prioridade maior. De acordo com Seifert e Edwards (2008), se receber N pacotes com uma prioridade alta,

e X pacotes com uma prioridade baixa, ele irá encaminhar todos os N pacotes, para depois encaminhar os X pacotes. Deve se considerar que exista uma única fila por porta.

2.4.2 Enfileiramento justo ponderado

De acordo com Seifert e Edwards (2008) o enfileiramento justo ponderado (WFQ, do inglês *Weighted Fair Queuing*) consegue fornecer pesos para filas diferentes. Diferentemente da prioridade estrita, o WFQ não faz a fila com maior prioridade encaminhar todos quadros primeiro, mas sim realiza um ponderamento entre todas filas conforme seus pesos.

Em casos de congestionamento do *switch* em todas filas, o WFQ consegue gerenciar a entrega dividindo a banda total por pesos conforme cada fila. Esse gerenciamento é feito de maneira ponderada, utilizando a técnica *Round Robin*. Conforme Seifert e Edwards (2008) a fila com o maior peso, deve encaminhar a maior quantidade de quadros, enquanto as demais filas com menores pesos, respectivamente, devem encaminhar gradativamente menos quadros.

Existe uma relação entre pesos e prioridades. Em cada equipamento que implementa o WFQ, deve existir um número determinado de filas (podendo variar entre modelos de equipamentos) porém sempre terá 8 prioridades. Dentre essas 8 prioridades devem estar separadas dentro das filas, e consequentemente as filas com seus pesos específicos.

Tabela 1 – Exemplo de algoritmo sendo usado com apenas 3 filas

Fila	Peso	Prioridade
1	1	Baixa
2	4	Média
3	10	Alta

Por exemplo, considerando a Tabela 1, cada 15 pacotes recebidos, serão separados 10 pacotes com a prioridade alta/máxima, 4 pacotes serão separados com prioridade média, e 1 pacote será separado com a prioridade mais baixa.

3 CENÁRIOS E FERRAMENTAS

Neste capítulo será apresentado o cenário juntamente com os equipamentos utilizados. Também será explicado a ferramenta PacNew, utilizada e desenvolvida para o nosso trabalho.

Com o cenário explicado neste capítulo, deseja-se medir o desempenho do *switch* através de uma ferramenta semi-automática. Com esse software é possível medir a quantidade de quadros entregues no receptor e medir quantos quadros possuem em cada prioridade.

3.1 Descrição de cenário e metodologia de testes

Foi proposto um cenário com 3 computadores e um *switch* e destes 3 computadores, 2 deles devem gerar fluxos, enquanto a outra máquina apenas fica com a função de armazenamento de dados recebidos. Na Figura 5 foram nomeadas as máquinas **transmissor 1, transmissor 2 e receptora**, os fluxos com suas prioridades saindo de cada máquina e um fluxo sainte do *switch* chamado de **Fluxo Tratado**. Em ambos os fluxos o *switch* deverá verificar a TAG com o campo de prioridade e dependendo do algoritmos da fila de saída, irá tratar de maneira diferente.

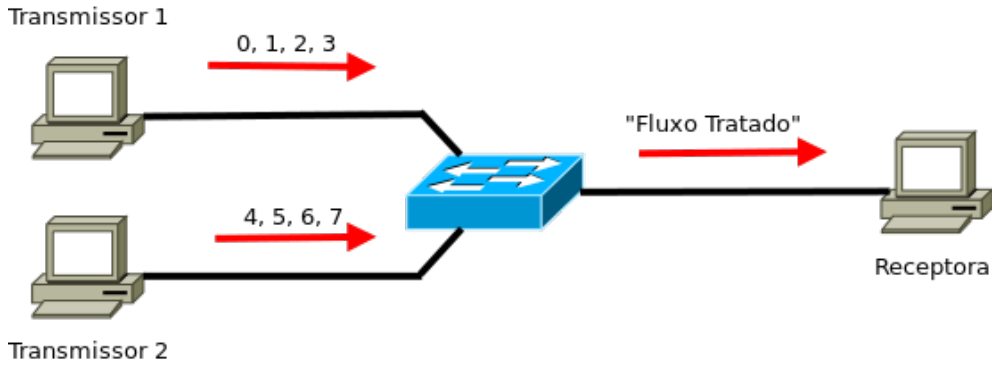

Figura 5 – Cenário utilizado para todos os testes

Nas máquinas ditas como **transmissor 1** e **transmissor 2**, deve-se executar o *script PacNew-Forward* (Apêndice B), que fará o encaminhamento das prioridades 0, 1, 2 e 3 ou 4, 5, 6, 7. Em cada máquina será gerado um fluxo constante por alguns segundos (aproximadamente 20 segundos, podendo variar de acordo com a máquina utilizada). Em cada máquina deverá ser configurado o endereço IP na interface utilizada (interface utilizada para comunicação com *switch*).

Na máquina dita como **receptora**, ela apenas irá receber o trafego e irá demonstrar algumas informações coletadas. Nesta máquina deve-se estar executando o *script PacNewReceiving* (Apêndice C). Nesta máquina também deve-se configurar o endereço IP na interface de comunicação direta com o *switch*.

No *switch* foi utilizado uma facilidade do equipamento onde o mesmo consegue delimitar a velocidade da porta, fazendo com que uma interface que nativamente consiga trafegar em *Fast* Ethernet passe a trafegar somente em Ethernet (10Mbps) . Essa configuração foi utilizada somente na interface que comunica com a máquina **receptora**, somente para provocar o *switch* a realizar a priorização, já que há um gargalo entre os enlaces fim a fim.

Um dos principais motivos que nos fez descartar outros cenário, foi devido ao *software* desenvolvido. Devido as limitações da aplicação (*PacNew*), ao encaminhar todos os fluxos pela mesma interface, diversas

vezes ocorreu um grande número de erros. Esses erros foram ocasionados devido ao estouro de *buffer*, sendo encaminhado 8 fluxos simultaneamente com a mesma taxa pela mesma interface. Com outras ferramentas esse tipo de erro é tratado de outras maneiras.

Deve existir uma porcentagem de erro pequena, devido a cargas computacionais nos quais o sistema operacional, pode estar tratando ou descartando os quadros. Possivelmente devido a alta taxa de quadros enviados não estava sendo possível gerenciar corretamente esse fluxo encaminhado (nos testes realizados). Devido aos problemas ocorridos, decidiu-se pela não adoção de uma única máquina, já que não é possível detectar a fonte de possíveis problemas, seja do sistema operacional, interface de rede ou até mesmo do *switch*.

3.1.1 Equipamentos utilizados

Conforme informado anteriormente, foi necessário utilizar 3 computadores, e um *switch*. Nos experimentos foram utilizados os *switch* TP-Link TL-SG3210, Intelbras SF 800 VLAN, Intelbras SF 800 Q+. Abaixo seguem as informações e especificações dos computadores:

- Computador 1:
  Processador: Intel(R) Pentium(R) D CPU 3.00GHz
  Memória RAM: 4Gb
  Sistema Operacional: Linux Ubuntu 11 (2.6.38)
  Velocidade placa de rede: 10/100/1000 Mbps/s

- Computador 2 e 3:
  Processador: Intel(R) Core(TM)2 Duo CPU E6750 @ 2.66GHz
  Memória RAM: 3Gb
  Sistema Operacional: Linux Ubuntu 11 (2.6.38)
  Velocidade placa de rede: 10/100/1000 Mbps/s

De acordo com Intelbras (2018a) e Intelbras (2018b), em ambos os casos os *switches* operam com QoS em 802.1p com duas filas de prioridades por portas. Já no caso do Intelbras (2018c), é informado que consegue operar com o padrão 802.1p com 4 filas de prioridades. No manual do usuário do Intelbras (2018c) é informado que é utilizado 4 algoritmos de filas, sendo eles SP, WFQ, uma junção entre eles (SP+WFQ) e um algoritmo que prioriza o quadro com pesos iguais.

3.2 Ferramentas utilizadas

Nesta seção iremos apresentar a ferramenta *PacNew*, como utilizar e em qual situação utilizar, caso alguém precise utilizar essa ferramenta no futuro. A ferramenta *PacNew* é divida em duas partes, a ferramenta que irá injetar tráfego na rede, e a ferramenta que irá capturar o tráfego na rede, respectivamente *PacNewForward* e *PacNewReceving*.

A ferramenta *PacNewForward* que deve estar sendo executada nas duas máquinas transmissoras, deverá injetar um grande tráfego na rede a ponto de fazer com que o *switch* seja forçado a realizar alguma priorização nos quadros. A ferramenta *PacNewReceving* estará recebendo o tráfego, e auxiliando na analise com os fluxos gerados pelas máquinas transmissoras, de maneira mostrar se existe uma priorização entre cada fluxo recebido.

3.2.1 *PacNew*

Pensando no objetivo da "automatização" e a manipulação de dados (coletados), foi desenvolvida uma ferramenta para este fim. Com isso também foi possível prevenir problemas de configurações erradas, erros em instalações de programa, ou outros possíveis erros durante a instalação ou execução. Na proposta de automatização, procuramos fazer com que exista o mínimo de interação do usuário, objetivando um sistema de utilização e informações limpas.

Verificando novas alternativas, foi reaproveitado um código feito em Python (código citado no Apêndice A), no qual é utilizado um *socket* para fazer o envio do quadro para ser reconhecido em seu receptor. Esse código originalmente, de forma manual (passando parâmetros), necessitava do MAC de origem e destino, e algum dado para ser enviado, o que foi melhorado para tentar fazer com que o usuário tenha que fornecer menos informações.

Após analisar e modificar esse arquivo, foram projetados dois *scripts* que iriam manipular diversas informações, dentre elas a tabela protocolo de resolução de endereços (ARP, do inglês *address resolution protocol*), o MAC de origem e destino. O *script PacNewForward* faz o encaminhamento enquanto o *script PacNewReceiving* recebe a faz uma pequena leitura das informações.

3.2.1.1 *PacNewFoward*

O *script PacNewFoward* tem como objetivo realizar o encaminhamento de 8 fluxos distintos, utilizando duas máquinas para encaminhar fluxos e uma máquina para receber estes fluxos. Com o objetivo de fazer com que o *switch* seja forçado a realizar o enfileiramento destes quadros e consequentemente realizando a priorização dos mesmos, sendo que cada máquina irá encaminhar uma ordem de fluxos pré-determinados.

Conforme mostrado no item 3.1, uma máquina irá encaminhar os fluxos com prioridades de 0 a 3 e a outra máquina deve encaminhar os fluxos com as prioridades restantes (4 a 7).

É necessário que já esteja configurado um IP na interface de rede que comunica com o *switch*. Uma das poucas informações que o *script* pede é o IP da máquina de destino e é desta maneira que o *script* atualiza a tabela ARP, e consegue cruzar as informações de MAC de destino, interface de encaminhamento.

Para que o *script* consiga receber o MAC de destino, sem que seja fornecido manualmente, foi pensado em utilizar a tabela ARP. No Código 3.1 são mostrados diversos parâmetros dentre eles alguns IPs, MACs e interfaces. Para que essa tabela seja atualizada, é necessário que o equipamento esteja visível na rede e consiga ser acessado.

Código 3.1 – Informações fornecidas a partir da Tabela ARP.

```
1 Endereço TipoHW EndereçoHW Flags Máscara Iface
2 10.0.0.100        ether   00:1A:3F:D6:50:20 C        wlp2s0
3 10.0.0.115        ether   60:02:B4:A1:17:DE C        wlp2s0
4 10.0.0.118        ether   00:E0:4C:68:6A:75 C        wlp2s0
```

No Código 3.2 é possível ver que é solicitado pelo *script* o IP da máquina **receptora**. Com essa informação, o *script* consegue encaminhar uma mensagem ICMP (do inglês *Internet Control Message Protocol*), e com isso é atualizado a tabela ARP. Caso a máquina não seja acessível por algum motivo as informações de MAC do Receptor, MAC do Transmissor, IP do Receptor e Interface de saída Se todas as informações (MAC do Receptor, MAC do Transmissor, IP do Receptor e Interface de saída) estiverem aparecendo devidamente, pode-se prosseguir com o experimento.

Código 3.2 – Informações da tela do script PacNewForward

```
Informe o endereço IP da maquina receptora:
 10.0.0.118
PING 10.0.0.118 (10.0.0.118) 56(84) bytes of data.
64 bytes from 10.0.0.118: icmp_seq=1 ttl=64 time=66.2 ms

--- 10.0.0.118 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 66.210/66.210/66.210/0.000 ms


Adquirindo o MAC da maquina transmissora:
IP do receptor:
 10.0.0.118
IP do transmissor:
 10.0.0.117
MAC do receptor:
ac:1f:74:83:7f:3f
MAC do transmissor:
e0:06:e6:d0:32:cb
 Interface de saída:
wlp2s0


Será necessário colocar o Switch em um porta Taggeada, antes de continuar
Pressione "Enter" após alterar a porta do Switch

Nesta maquina deseja enviar os pacotes com prioridade [0, 1, 2, 3] ou [4, 5, 6, 7]?
Para [0, 1, 2, 3] digite "1"
Para [0, 1, 2, 3] digite "2"
```

Um dos pré-requisitos para que se funcione é que o as portas do *switch* utilizadas estejam em modo *access* ou híbrida. O usuário deverá verificar as especificações com o fabricante.

Por questões de processamento e *buffer* do *socket*, não é possível encaminhar 7 fluxos distintos (cada um com sua prioridade). Quando gerado uma taxa de quadros por segundos, muito superior a taxa suportada pela porta do dispositivo esses quadros entram em um *buffer* aguardando o encaminhamento. Em casos onde os quadros entram em um *buffer* não é limitado a capacidade porque o objetivo é mandar realmente a maior quantidade de quadros possível em um curto espaço de tempo e pela limitação da aplicação que repete uma rotina diversas vezes para encaminhar diversos quadros . Nos testes foi observado que ao tentar enviar acima de 4 fluxos ou mais, com uma alta taxa de quadros, ocorrem alguns erros em seu envio, respectivos ao *buffer*. Então o método adotado neste caso seria dividir 4 fluxos em uma máquina e outros quatro fluxos em outra máquina. Logo após as informações aparecerem na tela (conforme mostrado na Código 3.2), aparece um questionamento de quais quadros com suas prioridades deseja encaminhar. Após escolher a opção e pressionar a tecla **enter**, automaticamente começará a ser encaminhado o fluxo por um período maior que 20 segundos, dando tempo de iniciar o *script PacNewReceiving*.

De maneira resumida, o Código 3.2 explica o que acontece no *script PacNewForward*. Ao iniciar o

usuário precisará informar o endereço da máquina receptora, para que atualize a tabela ARP, porém caso esteja fora do normal, ele irá mostrar com as informações nulas. Caso positivo, ele informa as informações atualizadas da tabela ARP, com o MAC de destino e MAC de origem. Após isso o *script* irá solicitar quais são as prioridades que aquela máquina irá encaminhar e se for válido a opção, ele irá encaminhar e depois finalizar. Se não for válido, ele irá encerrar o aplicativo.

No *Script PacNewForward* após a triagem de dados inicial, diversos códigos são iniciados. Esses programas foram baseados em um código já existente, com o objetivo de encaminhar apenas um quadro, e modificados para encaminhar uma grande quantidade. Além das funções de leitura de arquivo (criados anteriormente pela triagem do *script*), pode-se dividir em duas funções.

3.2.1.2 *PacNewReceiving*

O *script PacNewReceiving* tem como objetivo fazer a coleta das informações e dados da máquina **transmissora**, além de receber todo o fluxo com todas as prioridades. É necessário que já esteja configurado um IP na interface de rede que comunica com o *switch*.

Diferentemente do *script PacNewFoward*, o *PacNewReceiving* somente necessita da tabela ARP para saber por qual interface que ele irá receber os quadros. Mostrado na Código 3.3, é bem mais simples no seu recebimento e na interação com o usuário. Da mesma forma, ele também pede que antes de prosseguir, até o momento o usuário deve estar conectado em uma porta do tipo *trunk*, porém no momento que o experimento for executado, é necessário que esteja em uma porta *access* ou híbrida.

Na Código 3.3 mostra somente o que foi coletado na interface receptora e alguns recados para utilização do *script* da maneira correta.

Código 3.3 – Informações da tela do *script PacNewReceving*

```
34 Informe o endereço IP da maquina transmissora.
35 10.0.0.118
36 PING 10.0.0.118 (10.0.0.118) 56(84) bytes of data.
37 64 bytes from 10.0.0.118: icmp_seq=1 ttl=64 time=28.2 ms
38
39 --- 10.0.0.118 ping statistics ---
40 1 packets transmitted, received, 0% packet loss, time 0ms
41 rtt min/avg/max/mdev = 28.236/28.236/28.236/0.000 ms
42
43 Adquirindo as informações da interface deste equipamento [Maquina Receptora]
44
45 Interface utilizada: wlp2s0
46
47 Antes de começar a captura de pacotes, certifique-se que a interface utilizada apareceu
       corretamente.
48 Tambem deverá ser alterado a porta no Switch, para uma porta que esteja Taggeada.
49 Caso apareça e tenha alterado para aporta Taggeada, pressione enter para iniciar as capturas...
```

Após o recebimento de todos os dados, mostra algumas informações para o usuário segmentadas em 3 blocos demonstrado no Código 3.4. Na tela são mostradas 3 blocos de informações, onde em cada bloco possui sua característica para ser analisada.

No primeiro bloco é mostrado, se o fluxo respectivo a prioridade, chegou na máquina receptora. Caso não tenha chegado nada, ele irá acusar como **Não**, e conforme mostrado, se possuir o fluxo correspondente aparecera **Sim**.

No segundo bloco, é mostrado o tamanho do arquivo dividido por prioridade. Com isso é possível ver de maneira perceptível que as maiores prioridades possuem arquivos maiores e consequentemente, maiores quantidade de quadros recebidos na máquina Receptora. No terceiro bloco, é mostrado de maneira dinâmica os arquivos em comparação ao arquivo total. Se por algum motivo não tiver recebido o fluxo correspondente a prioridade, não irá aparecer o tamanho daquele arquivo, somente deixando as prioridades detectadas. Foi pensado desta maneira para que se possa fazer uma breve conferência com o tamanho total do arquivo.

Código 3.4 – Informações da tela do script PacNewReceving

```
50 No teste até o momento, possuímos as prioridades:
51 Prioridade 0: Sim
52 Prioridade 1: Sim
53 Prioridade 2: Sim
54 Prioridade 3: Sim
55 Prioridade 4: Sim
56 Prioridade 5: Sim
57 Prioridade 6: Sim
58 Prioridade 7: Sim
59
60 Tamanho do arquivo/prioridade:
61 Prioridade 0: 4646460 bytes
62 Prioridade 1: 827392 bytes
63 Prioridade 2: 1439250 bytes
64 Prioridade 3: 3328758 bytes
65 Prioridade 4: 5476624 bytes
66 Prioridade 5: 5419862 bytes
67 Prioridade 6: 13583692 bytes
68 Prioridade 7: 13682470 bytes
69
70 O fluxo total correspondente a: 48407076 bytes
71 Das capturas feitas de cada fluxo, podemos dizer que equipamento consegue capturar (caso não
       apareça nada, não foi feito com sucesso):
72 Da prioridade 0, gerou um arquivo no tamanho de: 4646460 bytes
73 Da prioridade 1, gerou um arquivo no tamanho de: 827392 bytes
74 Da prioridade 2, gerou um arquivo no tamanho de: 1439250 bytes
75 Da prioridade 3, gerou um arquivo no tamanho de: 3328758 bytes
76 Da prioridade 4, gerou um arquivo no tamanho de: 5476624 bytes
77 Da prioridade 5, gerou um arquivo no tamanho de: 5419862 bytes
78 Da prioridade 6, gerou um arquivo no tamanho de: 13583692 bytes
79 Da prioridade 7, gerou um arquivo no tamanho de: 13682470 bytes
```

De maneira geral, a imagem Código 3.4 mostra o que faz o *script*. Ao iniciar é solicitado o IP (de uma das máquinas transmissoras), para saber qual a interface monitorada. Após isso ele mostra as informações de MAC de destino e de origem. Após isso ele fica aguardando uma ordem do usuário para

que seja capturado os pacotes. Após o início da transmissão dos outros dois, apresenta os resultados tratados na tela (conforme os blocos citados) e encerra o *script*.

Todo o *script PacNewReceving* é baseado em leitura de parâmetros se comparado ao *PacNew-Forward*, que fazia a chamada de aplicações externas. No *script* é realizado uma coleta geral, e depois a separação por prioridades dos arquivos. Com o arquivo geral, é possível verificar se a prioridade está contida ou não, e nos arquivos individuais é possível verificar o tamanho da mesma e com isso disponibilizar para o usuário.

3.3 Utilizando a ferramenta *PacNew*

Para utilizar a ferramenta, você deverá garantir que o cenário de utilização com as 3 máquinas e o *switch*, estejam conforme a Figura 6. No Apêndice D, existe um tutorial de como instalar a ferramenta para sua utilização.

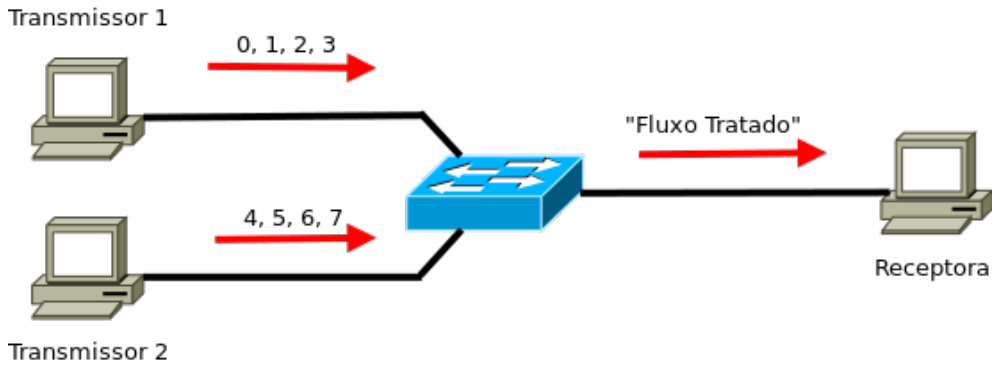

Figura 6 – Cenário utilizado

Para que seja possível que cada máquina consiga coletar as informações da outra máquina, será necessário que todas as máquinas possuam um IP atribuído na interface de rede utilizada. Também será necessário que a máquina responda a uma requisição de mensagem ICMP. Recomendamos que todas as máquinas estejam na mesma faixa de rede, caso contrario, não irá funcionar.

Nas máquinas **transmissoras 1 e 2** será necessário a instalação da ferramenta *PacNewForward*. Como é uma ferramenta semi-automatizada, será necessário que o usuário saiba exatamente qual fluxo será utilizado em cada máquina. Recomenda-se que utilize o fluxo 0, 1, 2, 3 na máquina **transmissora 1** e o fluxo 4, 5, 6, 7 na máquina **transmissora 2**.

Na máquina **receptora**, será necessário a instalação da ferramenta *PacNewReceving*. Essa ferramenta somente será executada, quando os fluxos das outras máquinas já estiverem rodando. Recomendamos que seja lido o tutorial contido no Apêndice D.

Não existe um requisito formal, relacionado a processamento ou memória, para as máquinas utilizadas para reprodução do experimento, porém no item 3.1.1 possuímos 2 exemplos de máquinas que foram utilizadas. O único requisito obrigatório é que o sistema operacional utilizado deverá ser um sistema Linux.

4 RESULTADOS OBTIDOS

Para testar a ferramenta e comprovar o funcionamento dos *switches*, foram efetuados alguns ensaios com alguns equipamentos diferentes. Neste capítulo são apresentados os resultados obtidos durante os ensaios e algumas considerações.

4.1 Switch Gerenciável - Cisco Catalyst 2960-X

Iniciado o teste no *switch*, conforme o cenário mostrado no capítulo anterior, é considerado que esse equipamento seja o nosso referencial perante os demais por possuir 8 filas. Todos os outros equipamentos testados mais a frente possuem 4 filas ou menos.

Foram realizados os primeiros ensaios com o *switch* configurado com o algoritmo de enfileiramento WFQ. De acordo com as informações contidas no *switch*, as filas estavam configuradas com os seguintes pesos: "1:2:3:4:5:6:7:8". Na Figura 7 é possível perceber o resultado da priorização passada pelo *switch*, sendo que de maneira ponderada, é possível perceber uma diferenciação da quantidade fluxos distintos. O algoritmo utilizado permite esse tipo de resultado, realizando um eficiente balanceamento de carga.

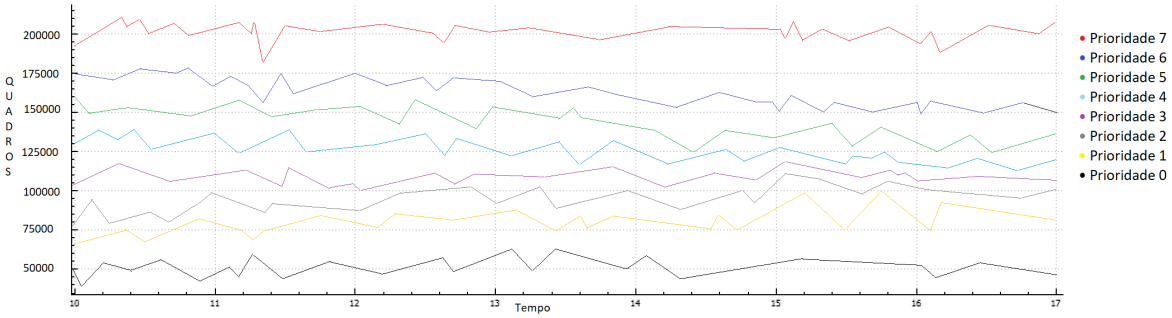

Figura 7 – Resultado gráfico da priorização feita no *switch* usando o algoritmo WFQ no equipamento Catalyst 2960-X

Na Tabela 2 é possível ver a quantidade de quadros recebidas por prioridade em relação ao que foi recebido. Esses dados coletados estão coerentes, pois percebe-se que existe um padrão de crescimento entre as quantidades e suas porcentagens.

Utilizando o algoritmo SP + WFQ o resultado altera-se um pouco, porém continua na mesma lógica, como pode ser visto Tabela 3. Esse algoritmo funciona unificando o SP com o WFQ, sendo que pela característica do algoritmo de SP, a prioridade mais alta, fica com a maior parte (até que se tenha transmitido tudo) e o restante realiza a disputa com as demais prioridades, de maneira ponderada utilizando o algoritmo WFQ.

Os dois ensaios tinham com objetivo comprovar se o equipamento de fato possuía 8 filas. Com as informações contidas nas tabelas 2 e 3, é possível dizer que esse *switch* possui 8 filas. Na Tabela 2 é possível notar que a cada porcentagem entre prioridades, não há uma aproximação em nenhum dos casos. Na Tabela 3, fica mais nítido essa situação por consequência do algoritmo utilizado neste ensaio.

Deste modo então é possível concluir que esse equipamento possui 8 filas de priorização e será utilizado como referencial do trabalho. Mesmo ocorrendo a diferenciação entre algoritmos de fila, é possível observar claramente que existem diferenças entre prioridades recebidas.

Tabela 2 – Resultado a priorização feita no *switch* usando o algoritmo WFQ utilizando o *switch* Catalyst 2960-X

Quantidade de quadros	Porcentagem	Prioridade
305461	21%	7
282330	19,4%	6
247890	17%	5
207212	14,2%	4
165674	11,4%	3
124334	8,5%	2
82890	5,7%	1
41446	2,8%	0

Tabela 3 – Resultado da priorização feita no *switch* usando o algoritmo SP+WFQ utilizando o *switch* Catalyst 2960-X

Quantidade de quadros	Porcentagem	Prioridade
661522	50%	7
330761	25,1%	6
165380	12,5%	5
82029	6,2%	4
43660	3,3%	3
20904	1,58%	2
10717	0,8%	1
5424	0,4%	0

4.2 Switch Gerenciável - TP-Link TL-SG3210

Configurado o cenário conforme detalhado no capítulo anterior e iniciado os *scripts PacNewReceving e PacNewForward*, será gerado um arquivo na máquina **receptora**. Através desse arquivo é possível realizar a leitura dos dados respectivos dos *scripts* e neste arquivo será possível realizar a análise dos fluxos. Esse equipamento possui 4 filas e pode-se escolher entre 4 métodos de escalonamento de filas: SP, WRR/WFQ, SP+WRR/SP+WFQ e sem prioridades (prioridades).

Para analisar o comportamento do *switch*, foram gerados 3 ensaios utilizando os mesmos fluxos (8 fluxos distintos com diferentes prioridades), utilizando o algoritmo SP+WFQ. Na Figura 8 é possível verificar que o fluxo com maior prioridade mantém a vazão durante todo o período acima dos demais.

Na Figura 8 é percebido que existe diferenciação entre os fluxos. Os pares de prioridades (7,6), (5,4), (3,1), (2,0), pertencem as mesmas filas. Durante os testes não foram alterados quaisquer tipos de prioridades dos equipamentos e foi efetuado o reset de fábrica e a ordem se manteve.

Percebemos que na Figura 8 é possível confirmar que todos os pontos que a fila de prioridade superior das demais é a demonstrada mais acima das restantes (vermelha). Na Tabela 5 é possível confirmar que a porcentagem com maior vazão de quadros entregues seriam os quadros com as prioridades 6 e 7.

Individualmente ao observar os resultados obtidos na Tabela 4 por prioridades, não é possível tirar uma conclusão definitiva sobre o funcionamento real do *switch* e sua priorização. Olhando os resultados obtidos com os ensaios no *switch* com relação as filas, observar o fato que está correto como o *switch* estava trabalhando com os pacotes, sendo que no algoritmo SP+WFQ os pacotes com maior prioridade/fila devem ser entregues primeiro. O restante deverá ser entregue disputando com as demais prioridades de menor custo de maneira ponderada. Na Tabela 5 é possível ver essa relação por fila.

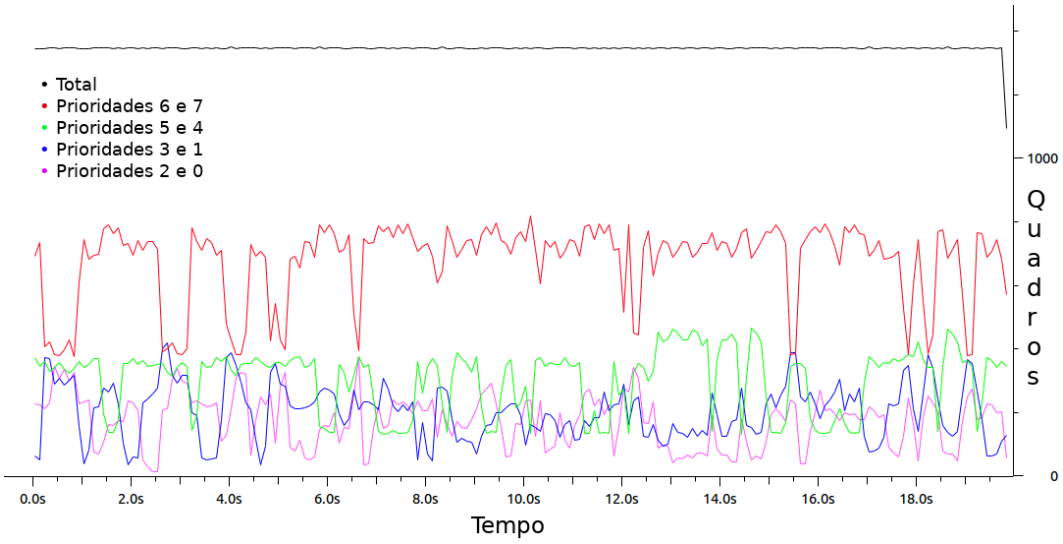

Figura 8 – Gráfico gerado em resposta do *switch* com a fila SP+WFQ com o equipamento TL-SG3210

Tabela 4 – Resultado da priorização feita no *switch* usando o algoritmo SP+WFQ utilizando o *switch* TL-SG3210

Fila	Porcentagem	Prioridade
3	21%	7
3	19,4%	6
2	17%	5
2	14,2%	4
1	11,4%	3
0	8,5%	2
1	5,7%	1
0	2,8%	0

Tabela 5 – Resultado da priorização feita no *switch* usando o algoritmo SP + WFQ utilizando o *switch* TL-SG3210

Fila	Porcentagem
3	50%
2	22,4%
1	14,8%
0	12,7%

4.3 Switch Não-Gerenciável - Intelbras SF 800 Q+

Diferentemente do TP-Link, este modelo é um *switch* extremamente simples, não possuindo interface gráfica configurável ou via interface de linha de comando (CLI, do inglês *command-line inteface*). É o equipamento barato e simples, completamente diferente do TP-Link com relação a funções. Este experimento foi executado exatamente da mesma forma que foi apresentado na seção 4.1.

No arquivo "bruto" gerado após a coleta, basicamente percebe-se que existe uma priorização porém simples. Esse tipo de *switch*, como é simples, funciona com o mínimo de priorização possível. De acordo com Intelbras (2018b), esse equipamento possui apenas duas filas de saída, porém com 8 níveis de prioridade.

Na Figura 9, conseguimos perceber que existe uma priorização entre os 8 fluxos distintos. Todos

os 4 primeiros fluxos são similares, recebendo a priorização alta enquanto os outros fluxos ficam com pouquíssimo fluxo restante, recebendo a priorização baixa.

Sendo o equipamento mais simples testado, ele recebeu a priorização mais simples possível e conforme o esperado, como mínimo de priorização possível.

A diferença entre os resultados, pode se dar com relação a quantidade de filas do *switch*. Neste caso podemos ver claramente que o *switch* conseguiu realizar a priorização, ainda que miníma e em comparação ao equipamento da TP-LINK que possui uma quantidade filas superior.

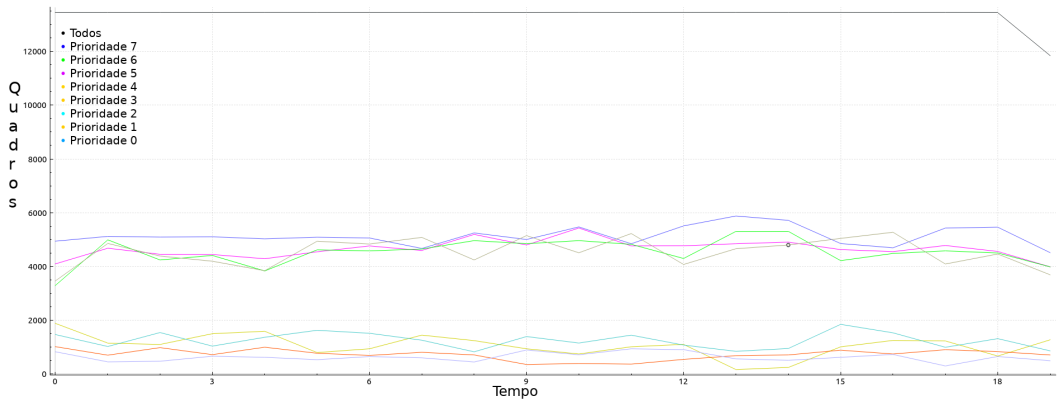

Figura 9 – Gráfico gerado em resposta do *switch* com o equipamento SF 800 Q+

4.4 Switch Não-Gerenciável - Intelbras SF 800 VLAN

Esse *switch* apresentou um resultado diferente com relação ao equipamento TP-Link TL-SG3210, porém muito similar ao SF 800 Q+. Esse equipamento possui apenas duas filas por portas, utilizando o algoritmo de filas WFQ. Neste caso como não é gerenciável, é dito pelo fabricante em suas especificações técnicas que ele opera desta maneira.

Os mesmos problemas ocorridos no SF 800 Q+, também ocorreram neste devido a ser um equipamento não-gerenciável. Porém o método para chegar na mesma conclusão teve de ser diferente.

De acordo com Intelbras (2018b), esse *switch* possui 7 VLANs distintas, cada uma associada a uma única porta. Por ser um *switch* de 8 portas, e possuir 7 VLANs, a porta restante recebe todo o tráfego das demais portas. Os testes foram feitos conforme orientação e características de uso do fabricante.

De acordo com os resultados, podemos analisar que de todo o tráfego recebido, nas prioridades mais baixas (0 até 3), em média 20% chegou na máquina que capturava o tráfego. Enquanto as demais prioridades (4 até 7), em nossos testes apresentou em média 80% do tráfego capturado. Na Figura 10 é mostrado um fluxo com prioridade alta e um fluxo com prioridade baixa e seu resultado com relação ao tipo de algoritmo de fila WFQ.

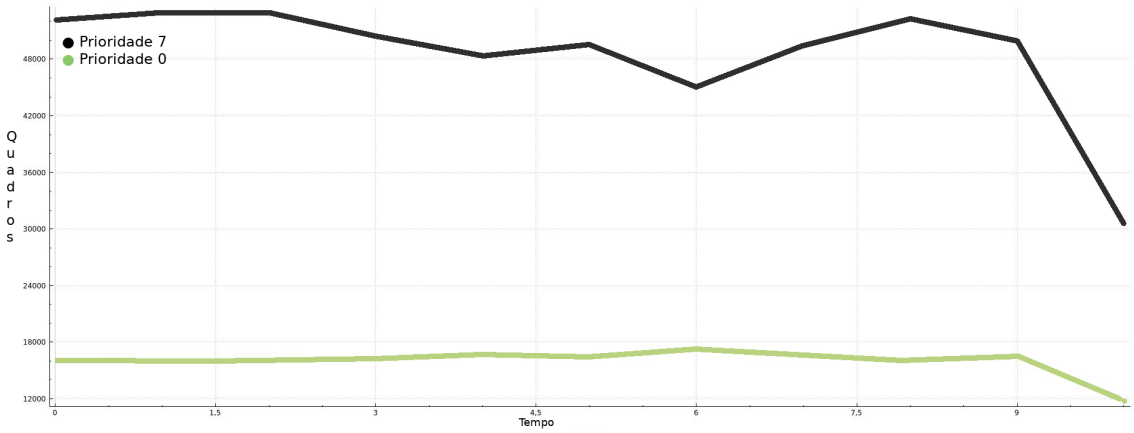

Figura 10 – Gráfico gerado em resposta do *switch* com o equipamento SF 800 VLAN

5 CONCLUSÕES

Neste trabalho foi possível gerar um processo semi automatizado através ensaios com equipamentos para verificar a compatibilidade com a norma IEEE 802.1p. Para essa tarefa, foi desenvolvida uma ferramenta para o usuário visando o mínimo esforço com validações diretas e simples. Foi possível desenvolver uma aplicação na qual consegue verificar se o *switch* está tratando corretamente os quadros conforme suas filas e prioridades. Com a metodologia proposta, foi possível realizar uma análise de alguns *switches* e conseguimos comprovar a diversidade de equipamentos hoje no mercado com suas características.

Cada *switch* tem suas características de funcionamento, e elas são fornecidas pelos fabricantes e nenhum dos equipamentos testados forneceram informações divergentes do que o fabricante informa. Variando o equipamento podemos concluir que os seus resultados podem ser bem diferentes, respectivamente sobre o funcionamento de filas e prioridades. Com relação a avaliação de *switch* podemos concluir que o equipamento realiza a priorização, dependendo da quantidade de filas e isso poderá variar da maneira que é escalonada as filas e seu enfileiramento.

Em experimentos conseguimos comprovar as diferenças entre cada prioridade e como o *switch* trata perante as filas. Dentre os resultado obtidos podemos concluir que não existe uma grande variação, quando são encaminhados muitos fluxos com pacotes de prioridades muito próximas (exemplo: prioridade 6 e 7), tendo em vista as filas de prioridade. Os resultados analisados mostram que o método de fila SP + WFQ, consegue realizar uma priorização visivelmente (Figura 7) ponderada, entre todas as filas com as prioridades. Dentre os ensaios realizados, conseguimos testar também com somente os métodos SP e WFQ individualmente, onde se mostraram métodos bem diferentes, sendo respectivamente método que priorização a mais alta prioridade perante as demais (até que se tenha concluído a transmissão daqueles quadros) e uma priorização ponderada entre a quantidade de quadros recebidos.

Tendo resultados particulares entre equipamentos diferentes, podemos obter resultados e concluir que se configurado filas do tipo SP+WFQ e WFQ essa afirmação é verdadeira. Já se feito uma comparação com métodos com filas próximas (com pesos iguais e não ponderados) e o método SP esse resultado fica bem diferente. Desta forma que os critérios para a conclusão pelas propriedades existentes utilizando esse método, é a sua característica em que a fila com maior valor (maior prioridade) utiliza o método SP, enquanto as filas restantes (com as prioridades restantes) utilizam de maneira ponderada o método WFQ. Quando um quadro encaminhado com a prioridade que está com a maior fila, esse quadro naquele instante ganha a largura total da banda. Quando um quadro encaminhado com as demais filas, ponderadamente entre pesos por fila, dividem a largura total da banda.

Com relação ao método utilizado para neste trabalho (resultado obtido através do SP+WRR), um dos critérios para nossa conclusão é através das propriedades e a sua característica em que a fila com maior valor (maior prioridade) utiliza o método SP, enquanto as filas restantes (com as prioridades restantes) utilizam de maneira ponderada o método WFQ. É necessário observar que para cada tipo de aplicação existe seu tipo de escalonamento de filas para melhor uso. É possível concluir com o trabalho desenvolvido, que a aplicação desenvolvida seja um facilitador para que seja possível identificar as filas e o comportamento do escalonamento dos quadros.

Como trabalho futuro, recomendaria uma possível integração do que já foi feito, com outros possíveis testes automatizados (STP, VLANs, entre outros) e que se consiga chegar a relatórios que indiquem se o equipamento é ou não compatível com as respectivas normas.

REFERÊNCIAS

CISCO. *Qual a diferença entre um Switch de Rede e um roteador?* [S.l.], 2018. Disponível em: <https://www.cisco.com/c/pt_br/solutions/small-business/resource-center/what-is-a-network-switch.html>. Acesso em: 21 Mar. 2018. Citado 2 vezes nas páginas 28 e 31.

DUARTE, O. C. M. B.; BICUDO, M. D. D. *IEEE 802.1p - QoS na camada MAC*. 2002. Citado 3 vezes nas páginas 29, 30 e 31.

INTELBRAS. *Datasheet SF 800 Q+*. [S.l.], 2018. Disponível em: <http://www.intelbras.com.br/sites/default/files/downloads/datasheet_sf_800_q_01-18.pdf>. Acesso em: 21 dez. 2018. Citado na página 34.

INTELBRAS. *Datasheet SF 800 VLAN*. [S.l.], 2018. Disponível em: <http://www.intelbras.com.br/sites/default/files/downloads/datasheet_sf_800_vlan_01-18.pdf>. Acesso em: 21 dez. 2018. Citado 3 vezes nas páginas 34, 43 e 45.

INTELBRAS. *Datasheet SF 800 VLAN*. [S.l.], 2018. Disponível em: <https://www.tp-link.com/br/business-networking/managed-switch/tl-sg3210/>. Acesso em: 21 dez. 2018. Citado na página 34.

KUROSE, J. F.; ROSS, K. W. *Redes de Computadores e a Internet 5ª edição*. [S.l.: s.n.], 2010. Citado 5 vezes nas páginas 15, 25, 26, 27 e 28.

SEIFERT, R.; EDWARDS, J. *The All-New Switch Book: The Complete Guide to LAN Switching Technology*. [S.l.]: John Wiley & Sons, 2008. Citado 5 vezes nas páginas 25, 26, 27, 31 e 32.

SOCIETY, I. C. *Bridges and Bridged Networks*. [S.l.: s.n.], 2014. Citado 2 vezes nas páginas 30 e 31.

STAFFORD, B. *IEEE 802.1Q*. [S.l.], 2019. Disponível em: <https://pt.wikipedia.org/wiki/IEEE_802.1Q>. Acesso em: 10 mai. 2019. Citado 2 vezes nas páginas 15 e 29.

TANENBAUM, A. S. et al. *Computer networks, 4-th edition*. [S.l.: s.n.], 2003. Citado 2 vezes nas páginas 28 e 29.

APÊNDICE A – CÓDIGOS EM PYTHON

Código A.1 – Código responsável pela prioridade 0 em Python (existem 8 códigos idênticos somente alterando a prioridade). Código necessário para utilização da ferramenta *PacNew* juntamente com os *scripts PacNewForward* e *PacNewReceving*. Esse arquivo deve ser adicionado na pasta raiz dos *scripts* já citados.

```
#Código baseado em outro código fornecido pelo professor Eraldo Silveira e Silva, que foi baseado no c
    ódigo: https://gist.github.com/cslarsen/11339448

# -*- coding: utf-8 -*-
import socket
import re, uuid
import sys, getopt

def build_ethernet_hdr(dest, src, proto):
 dest = "0x" + dest.replace(":", ",0x").upper()
 src = "0x" + src.replace(":", ",0x").upper()
 proto = "0x" + proto.replace(":",",0x").upper()
 temp = dest + "," + src + "," + proto
 temp = temp.split(",")
 temp = [int(i,16) for i in temp]
 temp = "".join(map(chr,temp))
 return temp

def main(argv):

#Função para ler os arquivos gerados pelo script PacNew

   y = open('./config/MAC_T', 'r')
   y_1 = y.readline()
   src = y_1.replace('\n','')
   y.close()

   z = open('./config/MAC_R', 'r')
   z_1 = z.readline()
   dest = z_1.replace('\n','')
   z.close()

#Dado padrão em todos os arquivos.

   ethernet_data_str = "Em um alfabeto de mudos, o fofoqueiro é depressivo Prioridade 0"

  #############
  # Prioridades:
  # 0 - 00:01 X
  # 1 - 20:01
  # 2 - 40:01
  # 3 - 60:01
  # 4 - 80:01
  # 5 - A0:01
  # 6 - C0:01
  # 7 - E0:01

#Parametros setados para configurar o pacote em 802.1Q/p
```

```
127     prio = "00:01"
128     proto = "81:00" + ":" + prio
129
130
131     try:
132        opts, args = getopt.getopt(argv,"hs:d:i:",["src=","dst=", "info="])
133     except getopt.GetoptError:
134        print 'sendraw.py -s <mac_src> -d <mac_dst> -c <data>'
135        sys.exit(2)
136     for opt, arg in opts:
137        if opt == '-h':
138           print 'sendraw.py -s <mac_src> -d <mac_dst> -i <data>'
139           sys.exit()
140        elif opt in ("-s", "--src"):
141             src = arg
142        elif opt in ("-d", "--dst"):
143             dest = arg
144        elif opt in ("-i", "--info"):
145             ethernet_data_str = arg
146        elif opt in ("-p", "--proto"):
147             proto = arg
148
149 #Função para ler os arquivos gerados pelo script PacNew
150
151     x = open('./config/INTERFACE', 'r')
152     x_1 = x.readline()
153     interface = x_1.replace('\n','')
154     x.close()
155
156     protocol = 0
157
158     print 'MAC FONTE', src
159     print 'MAC DESTINO', dest
160     print 'INTERFACE', interface
161
162     s = socket.socket(socket.AF_PACKET, socket.SOCK_RAW)
163     s.bind((interface,protocol))
164
165     ethernet_hdr_str = build_ethernet_hdr(dest, src, proto)
166
167
168 #Função "While" feito para durar aproximadamente 10 segundos, em certas condições
169
170     contx = 0
171     while (contx <= 99999):
172      s.send(ethernet_hdr_str + ethernet_data_str)
173          contx = contx + 1
174
175
176 if __name__ == "__main__":
177     main(sys.argv[1:])
```

APÊNDICE B – CÓDIGOS REFERENTE AO *PACNEW-FORWARD*

Código B.1 – *Script* referente ao código *PacNewForward*

```
#!/bin/bash
#Código efetuado com o intuito ciêntifico e didatico
#Objetivo: Código encaminhador de frames.
#Espera-se do úsuario alguns dados para que o script finalize (IP de destino e a opção respectiva ao frames enviados naquela maquina.
#Autor: Giovanni Salvatore de Almeida Curcuruto


ajustaParametros(){
  sudo arp -a > ./config/arqBruto.txt
  sed -i '/incompleto/d' ./config/arqBruto.txt
  sed 's/?\|em/\t/g' ./config/arqBruto.txt > ./config/lixoArp.txt
        grep -i -n -w $ip_recep ./config/lixoArp.txt > ./config/utilArp.txt
  tr '\t' "\n" < ./config/utilArp.txt > ./config/1Arp.txt
  sed 's/\[ether]//g' ./config/1Arp.txt > ./config/2Arp.txt
  sed 's/)//g' ./config/2Arp.txt > ./config/3Arp.txt
  sed 's/(//g' ./config/3Arp.txt > ./config/4Arp.txt
  sed -n '2p' ./config/4Arp.txt > ./config/ip1.txt
  sed -n '3p' ./config/4Arp.txt > ./config/mac1.txt
  sed -n '4p' ./config/4Arp.txt > ./config/interface1.txt
  sed 's/ //g' ./config/ip1.txt > ./config/IP
  sed 's/ //g' ./config/mac1.txt > ./config/MAC_R
  sed 's/ //g' ./config/interface1.txt > ./config/INTERFACE
}

enviaFrame1(){
  echo enviando 4 prioridades
  sudo python gop0.py | sudo python gop1.py | sudo python gop2.py | sudo python gop3.py
}

enviaFrame2(){
  echo enviando outras 4 prioridades
  sudo python gop4.py | sudo python gop5.py | sudo python gop6.py | sudo python gop7.py
}

enviaFrameAll(){
  echo envia prio 0
  sudo python gop0.py
  echo envia prio 1
  sudo python gop1.py
  echo envia prio 2
  sudo python gop2.py
  echo envia prio 3
  sudo python gop3.py
  echo envia prio 4
  sudo python gop4.py
  echo envia prio 5
  sudo python gop5.py
  echo envia prio 6
  sudo python gop6.py
  echo envia prio 7
  sudo python gop7.py

}

#Inicio do Script

echo Antes de iniciar, vamos limpar a tela...
```

```
235 sleep 3
236 clear
237
238 echo Informe o endereço IP da maquina receptora.
239 read ip_recep
240 ping -c 1 $ip_recep
241 echo
242 ajustaParametros
243 echo
244 echo Adquirindo o MAC da maquina transmissora.
245 var=$(cat ./config/INTERFACE)
246 sudo ifconfig  $var | grep -i hw | awk '{print $7}' > ./config/MAC_T
247
248 #Dados demonstrados no terminal respectivos a cada vez que é executado o experimento
249
250 echo IP do receptor: $ip_recep
251 echo
252 echo MAC do receptor:
253 cat ./config/MAC_R
254 echo
255 echo MAC do transmissor:
256 cat ./config/MAC_T
257 echo
258 echo Interface de saída:
259 cat ./config/INTERFACE
260 echo
261 echo Nesta maquina deseja enviar os pacotes com prioridade [0, 1, 2, 3] ou [4, 5, 6, 7]?
262 echo Para [0, 1, 2, 3] digite \"1\"
263 echo Para [4, 5, 6, 7] digite \"2\"
264 read OP
265
266 #Opções:
267 #Opção 1 - enviar os frames com as prioridades 0, 1, 2 e 3
268 #Opção 2 - enviar os frames com as prioridades 4, 5, 6 e 7
269
270 if [ "$OP" = "1" ]
271 then
272   enviaFrame1
273 elif [ "$OP" = "2" ]
274 then
275   enviaFrame2
276 else
277   echo Opção incorreta. Encerrando...
278 fi
```

APÊNDICE C – CÓDIGOS RESPECTIVO AO *PACNEW-RECEIVING*

Código C.1 – *Script* referente ao código *PacNewReceiving*

```
#!/bin/bash
#Código efetuado com o intuito ciêntifico e didatico
#Objetivo: Código encaminhador de Receptor e quantitativo.
#Espera-se somente a informação do IP da maquina emissora.
#Autor: Giovanni Salvatore de Almeida Curcuruto


##########################################################################################################3

prioridades(){

  v0=$(ls -l ./Receiving/p0 | awk '{print $5}')
  if [ $v0 = "0" ]
  then
    echo Prioridade 0: Não
    pv0="0"
  else
    echo Prioridade 0: Sim
    pv0="1"

  fi
  v1=$(ls -l ./Receiving/p1 | awk '{print $5}')
  if [ $v1 = "0" ]
  then
    echo Prioridade 1: Não
    pv1="0"
  else
    echo Prioridade 1: Sim
    pv1="1"
  fi
  v2=$(ls -l ./Receiving/p2 | awk '{print $5}')
  if [ $v2 = "0" ]
  then
    echo Prioridade 2: Não
    pv2="0"
  else
    echo Prioridade 2: Sim
    pv2="1"
  fi
  v3=$(ls -l ./Receiving/p3 | awk '{print $5}')
  if [ $v3 = "0" ]
  then
    echo Prioridade 3: Não
    pv3="0"
  else
    echo Prioridade 3: Sim
    pv3="1"
  fi
  v4=$(ls -l ./Receiving/p4 | awk '{print $5}')
  if [ $v4 = "0" ]
  then
    echo Prioridade 4: Não
    pv4="0"
  else
    echo Prioridade 4: Sim
    pv4="1"
  fi
  v5=$(ls -l ./Receiving/p5 | awk '{print $5}')
```

```
341   if [ $v5 = "0" ]
342   then
343     echo Prioridade 5: Não
344     pv5="0"
345   else
346     echo Prioridade 5: Sim
347     pv5="1"
348   fi
349   v6=$(ls -l ./Receiving/p6 | awk '{print $5}')
350   if [ $v6 = "0" ]
351   then
352     echo Prioridade 6: Não
353     pv6="0"
354   else
355     echo Prioridade 6: Sim
356     pv6="1"
357   fi
358   v7=$(ls -l ./Receiving/p7 | awk '{print $5}')
359   if [ $v7 = "0" ]
360   then
361     echo Prioridade 7: Não
362     pv7="0"
363   else
364     echo Prioridade 7: Sim
365     pv7="1"
366   fi
367
368 }
369
370 ##################################################################################################3
371
372 capBruto(){
373   sudo tcpdump -i $INTER -w ./Receiving/capBruto.cap &
374   sudo sleep 20
375   sudo killall tcpdump
376   sudo tcpdump -n -e -r ./Receiving/capBruto.cap > ./Receiving/capBrutoVisual.cap
377   echo Capitura concluida.
378 }
379
380 ##################################################################################################3
381
382 capDiv(){
383   sudo grep "p 0" ./Receiving/capBrutoVisual.cap > ./Receiving/p0
384   sudo grep "p 1" ./Receiving/capBrutoVisual.cap > ./Receiving/p1
385   sudo grep "p 2" ./Receiving/capBrutoVisual.cap > ./Receiving/p2
386   sudo grep "p 3" ./Receiving/capBrutoVisual.cap > ./Receiving/p3
387   sudo grep "p 4" ./Receiving/capBrutoVisual.cap > ./Receiving/p4
388   sudo grep "p 5" ./Receiving/capBrutoVisual.cap > ./Receiving/p5
389   sudo grep "p 6" ./Receiving/capBrutoVisual.cap > ./Receiving/p6
390   sudo grep "p 7" ./Receiving/capBrutoVisual.cap > ./Receiving/p7
391 }
392
393 ##################################################################################################3
394
395 ajustaParametros(){
396   sudo arp -a > ./Receiving/arqBruto.txt
397   sed -i '/incompleto/d' ./Receiving/arqBruto.txt
398   sed 's/?\|em/\t/g' ./Receiving/arqBruto.txt > ./Receiving/lixoArp.txt
399         grep -i -n -w $ip_recep ./Receiving/lixoArp.txt > ./Receiving/utilArp.txt
400   tr '\t' "\n" < ./Receiving/utilArp.txt > ./Receiving/1Arp.txt
401   sed 's/\[ether]//g' ./Receiving/1Arp.txt > ./Receiving/2Arp.txt
402   sed 's/)//g' ./Receiving/2Arp.txt > ./Receiving/3Arp.txt
403   sed 's/(//g' ./Receiving/3Arp.txt > ./Receiving/4Arp.txt
404   sed -n '2p' ./Receiving/4Arp.txt > ./Receiving/ip1.txt
405   sed -n '3p' ./Receiving/4Arp.txt > ./Receiving/mac1.txt
406   sed -n '4p' ./Receiving/4Arp.txt > ./Receiving/interface1.txt
407   sed 's/ //g' ./Receiving/ip1.txt > ./Receiving/IP
408   sed 's/ //g' ./Receiving/mac1.txt > ./Receiving/MAC_R
409   sed 's/ //g' ./Receiving/interface1.txt > ./Receiving/INTERFACE
410 }
```

```
411
412 ##########################################################################################################3
413
414 pesoPorP(){
415   echo Tamanho do arquivo/prioridade:
416   echo Prioridade 0: $v0 bytes
417   echo Prioridade 1: $v1 bytes
418   echo Prioridade 2: $v2 bytes
419   echo Prioridade 3: $v3 bytes
420   echo Prioridade 4: $v4 bytes
421   echo Prioridade 5: $v5 bytes
422   echo Prioridade 6: $v6 bytes
423   echo Prioridade 7: $v7 bytes
424
425 }
426
427 ##########################################################################################################3
428
429 porcentagem(){
430   total=$(ls -l ./Receiving/capBrutoVisual.cap | awk '{print $5}')
431
432   echo O fluxo total corresponde a: $total bytes
433   echo Das capturas feitas de cada Fluxo, podemos dizer que o equipamento consegue capturar \(caso não apareça nada,
        não foi feito com sucesso\):
434   if [ $pv0 != "0" ]
435   then
436     echo Da prioridade 0, gerou um arquivo no tamanho de: $v0 bytes
437   fi
438   if [ $pv1 != "0" ]
439   then
440     echo Da prioridade 1, gerou um arquivo no tamanho de: $v1 bytes
441   fi
442   if [ $pv2 != "0" ]
443   then
444     echo Da prioridade 2, gerou um arquivo no tamanho de: $v2 bytes
445   fi
446   if [ $pv3 != "0" ]
447   then
448     echo Da prioridade 3, gerou um arquivo no tamanho de: $v3 bytes
449   fi
450   if [ $pv4 != "0" ]
451   then
452     echo Da prioridade 4, gerou um arquivo no tamanho de: $v4 bytes
453   fi
454   if [ $pv5 != "0" ]
455   then
456     echo Da prioridade 5, gerou um arquivo no tamanho de: $v5 bytes
457   fi
458   if [ $pv6 != "0" ]
459   then
460     echo Da prioridade 6, gerou um arquivo no tamanho de: $v6 bytes
461   fi
462   if [ $pv7 != "0" ]
463   then
464     echo Da prioridade 7, gerou um arquivo no tamanho de: $v7 bytes
465   fi
466 }
467
468 ##########################################################################################################3
469
470 echo Antes de iniciar, vamos limpar a tela...
471 sleep 3
472 clear
473 #Coletando a interface
474
475 echo Informe o endereço IP da maquina transmissora.
476 read ip_recep
477 ping -c 1 $ip_recep
478 echo
479 ajustaParametros
```

```
480 echo
481 echo Adquirindo as informações da interface deste equipamento. [Maquina Receptora]
482 INTER=$(cat ./Receiving/INTERFACE)
483 echo
484 echo Interface utilizada: $INTER
485 echo
486 echo Antes de começar a captura de pacotes, certifique-se que a interface utilizada apareceu corretamente.
487 echo Tambem deverá ser alterado a porta no Switch, para uma porta que esteja Taggeada.
488 echo Caso apareça e tenha alterado para a porta Taggeada, pressione enter para iniciar as capturas...
489 read nothing
490 capBruto
491 capDiv
492 echo
493 echo No teste até o momento, possuimos as prioridades:
494 prioridades
495 echo
496 pesoPorP
497 echo
498 porcentagem
```

APÊNDICE D – TUTORIAL DE UTILIZAÇÃO DA FERRAMENTA *PACNEW*

Código D.1 – Tutorial Passo-a-Passo de como utilizar a ferramenta.

```
Tutorial de utilização – PacNew

O PacNew é uma ferramenta que gera e coleta quadros ethernet. Ele forma um pequeno relatório
    referente aos quadros recebidos e suas respectivas prioridades.

Esse tutorial tem como objetivo auxiliar na instalação e execução desse da ferramenta PacNew

1) Download:

Antes de executar será necessário fazer o download dos arquivos necessários. Você poderá ter
    acesso aos arquivos através do diretório no GitHub a baixo:
      https://github.com/giovannicurcuruto/PacNew.git

Esses arquivos

2) Preparando os arquivos

Será necessário dar as permissões de execução nos arquivos. Recomendamos o uso do comando a
    baixo:

      chmod 777 "arquivo"

Lembrando que esse comando deverá ser feito nos arquivos baixados do Github.
Esse comando altera as permissões de leitura e escrita, então deverá ser feito com cautela e
    atenção.

3) Iniciando o aplicativo PacNewReceving

Antes de iniciar verifique se as três maquinas estão visíveis. Recomendamos encaminhar um
    ping nos endereços IP das maquinas que serão utilizadas.

Através do terminal inicie o script PacNewReceving:

      ./PacNewReceving

Ele irá iniciar, limpando a tela e solicitando o endereço IP de uma das maquinas transmissoras
     (maquinas que serão as geradoras de tráfego).
Após informar o endereço IP, na tela deverá aparecer algumas informações sobre a interface na
    qual irá utilizar (informação útil para maquinas que possuam mais de uma interface de rede
```

```
)

Como o principal objetivo de script é apenas capturar e demonstrar os dados capturados, ele irá aguardar o momento que o usuario informar que poderá coletar.
Quando for o momento correto, deverá pressionar a teclar "ENTER".

Após isso, ele irá mostrar as informações na tela e depois irá encerrar.

4.1) Iniciando o PacNewForward 1

Antes de iniciar verifique se as três maquinas estão visíveis. Recomendamos encaminhar um ping nos endereços IP das maquinas que serão utilizadas.

Através do terminal inicie o script PacNewForward:

    ./PacNewForward

Ele irá iniciar, limpando a tela e solicitando o endereço IP da maquina receptora
Após informar o endereço IP, na tela deverá aparecer algumas informações sobre a interface na qual irá utilizar (informação útil para maquinas que possuam mais de uma interface de rede)

Logo depois irá um menu com duas opções, que nela deverá ser escolhido 1 ou 2. Você precisará lembrar a opção, pois precisará escolher na outra maquina transmissora.
Após escolhido, será encaminhado o tráfego e será encerrado.

4.2) Iniciando o PacNewForward 2

Antes de iniciar verifique se as três maquinas estão visíveis. Recomendamos encaminhar um ping nos endereços IP das maquinas que serão utilizadas.

Através do terminal inicie o script PacNewForward:

    ./PacNewForward

Ele irá iniciar, limpando a tela e solicitando o endereço IP da maquina receptora
Após informar o endereço IP, na tela deverá aparecer algumas informações sobre a interface na qual irá utilizar (informação útil para maquinas que possuam mais de uma interface de rede)

Logo depois irá um menu com duas opções, que nela deverá ser escolhido 1 ou 2. Você precisará lembrar a opção, pois precisará escolher na outra maquina transmissora.
Após escolhido, será encaminhado o tráfego e será encerrado.
```