UNIVERSIDADE REGIONAL DE BLUMENAU

CENTRO DE CIÊNCIAS EXATAS E NATURAIS

CURSO DE CIÊNCIA DA COMPUTAÇÃO – BACHARELADO

COMPORTAMENTO E DESEMPENHO DE REDES: UMA ANÁLISE BASEADA EM FLUXOS

GUILHERME HENRIQUE RAMOS

BLUMENAU
2018

GUILHERME HENRIQUE RAMOS

COMPORTAMENTO E DESEMPENHO DE REDES: UMA ANÁLISE BASEADA EM FLUXOS

Trabalho de Conclusão de Curso apresentado ao curso de graduação em Ciência da Computação do Centro de Ciências Exatas e Naturais da Universidade Regional de Blumenau como requisito parcial para a obtenção do grau de Bacharel em Ciência da Computação.

Prof. Francisco Adell Péricas, Mestrado - Orientador

BLUMENAU
2018

COMPORTAMENTO E DESEMPENHO DE REDES: UMA ANÁLISE BASEADA EM FLUXOS

Por

GUILHERME HENRIQUE RAMOS

Trabalho de Conclusão de Curso aprovado para obtenção dos créditos na disciplina de Trabalho de Conclusão de Curso II pela banca examinadora formada por:

Presidente:	Prof. Francisco Adell Péricas, Mestrado – Orientador, FURB
Membro:	Prof. Gilvan Justino, Mestrado – FURB
Membro:	Prof. Marcos Rodrigo Momo, Especialização – FURB

Blumenau, 09 de julho de 2018

Dedico este trabalho à minha família e amigos e principalmente aos meus pais e namorada pela cooperação e apoio durante o desenvolvimento deste.

AGRADECIMENTOS

Aos meus pais, por estarem sempre ao meu lado me apoiando.

A minha namorada, pelo incentivo e cooperação nos momentos difíceis.

Ao meu orientador, Francisco Adell Péricas pela atenção e colaboração no decorrer do trabalho.

Se quisermos um mundo de paz e justiça, devemos resolutamente pôr a inteligência a serviço do amor.

Antoine de Saint-Exupéry

RESUMO

Este trabalho apresenta o desenvolvimento de uma ferramenta capaz de exibir estatísticas de uso da rede com base na análise do seu fluxo de dados. Além disso, aborda as soluções e estratégias utilizadas para observar e exportar o fluxo de dados, e coletar e analisar os dados. A exportação do fluxo de dados é feita de um ponto estratégico da rede. A ferramenta foi desenvolvida na plataforma Linux Debian, utilizando o conjunto de ferramentas nfdump para coletar, persistir e analisar o fluxo de dados. Os dados são exibidos por meio de uma plataforma Web, desenvolvida utilizando as últimas tecnologias disponíveis no mercado. É utilizado a linguagem de scripts Shell script para automatizar as tarefas de análise do fluxo de dados e persistência das estatísticas extraídas em arquivo JSON. Foi realizado um monitoramento com a ferramenta em um ambiente de rede real, onde foi possível validar com sucesso os objetivos estabelecidos e os benefícios da ferramenta no dia-a-dia do administrador de rede.

Palavras-chave: Fluxo de dados. Rede. Monitoramento de redes. Gerência de desempenho.

ABSTRACT

This work presents the development of a tool capable of displaying network usage statistics based on the analysis of its data flow. In addition, it addresses the solutions and strategies used to observe and export the data flow and collect and analyze the data. The export of the data flow is made from a strategic point of the network. The tool was developed on the Linux Debian platform, using the nfdump toolkit to collect, persist and analyze the data flow. The data is displayed through a Web platform, developed using the latest technologies available in the market. The Shell script scripting language is used to automate data flow analysis tasks and the persistence of extracted statistics in the JSON file. A monitoring with the tool was carried out in a real network environment, where it was possible to successfully validate the established objectives and the benefits of the tool in the day-to-day of the network administrator.

Key-words: Flow. Network. Network monitoring. Performance Management.

LISTA DE FIGURAS

LISTA DE QUADROS

LISTA DE ABREVIATURAS E SIGLAS

API - Application Programming Interface

CSS - Cascading Style Sheets

CSV - Comma-separated values

DMZ - Demilitarized Zone

DoS - Denial-of-Service

FCAPS - Fault, Configuration, Accounting, Performance and Security

HTTP - Hypertext Transfer Protocol

IA - Internet Accounting

ICMP - Internet Control Message Protocol

ID – Identifier

IEs - Information Elements

IETF - Internet Engineering Task Force

IP - Internet Protocol

IPFIX - IP Flow Information Export

ISO - International Organization for Standardization

JSON - JavaScript Object Notation

LAN - Local Area Network

NIC - Network Interface Card

NMS - Network Management Station

PMTU - Path Maximum Transmission Unit

RTFM - Realtime Traffic Flow Measurement

SCTP - Stream Control Transmission Protocol

SNMP - Simple Network Management Protocol

SSH - Secure Shell

TCP - Transmission Control Protocol

UDP - User Datagram Protocol

URL - Uniform Resource Locator

SUMÁRIO

1 INTRODUÇÃO

Atualmente, a Internet é possivelmente o maior sistema de engenharia já inventado pela humanidade, com centenas de milhões de dispositivos conectados (KUROSE, JIM F. ; ROSS, 2013). Não obstante, segundo Stallings (2005), a tendência é em direção a redes ainda mais complexas, que aceitam mais aplicações e mais usuários.

"O importante papel das redes de telecomunicações nos diversos segmentos da sociedade faz o gerenciamento de redes uma tarefa vital, tanto para os provedores de serviço de telecomunicações quanto para seus usuários" (AMARAL, 2011, p. 31). Para Stallings (2005), o administrador de rede precisa de estatísticas de desempenho para ajudá-lo a planejar, administrar e manter grandes redes em operação. Estas estatísticas podem ser usadas para reconhecer possíveis "gargalos" antes mesmo que ocasionem transtornos aos usuários. Assim, ações corretivas apropriadas podem ser tomadas.

Segundo Hofstede et al. (2014), entre as técnicas de monitoramento disponíveis nos dias de hoje, a mais viável é o monitoramento do fluxo de dados utilizando o método passivo. Nele os dados são capturados e analisados posteriormente. A abordagem passiva mais utilizada é a coleta e exportação do fluxo de dados, pois suporta o aumento do tráfego de forma uniforme. Em contraste com o método passivo, o método ativo requer muito processamento e armazenamento uma vez que captura o pacote.

Com base nesta contextualização, propõe-se o desenvolvimento de uma ferramenta, que através do monitoramento do fluxo de dados com a abordagem passiva, forneça estatísticas detalhadas do comportamento e desempenho da rede.

1.1 OBJETIVOS

O objetivo deste trabalho é desenvolver uma ferramenta para prover informações do comportamento e desempenho da rede, analisando seu fluxo de dados.

Os objetivos específicos são:

a) exportar o fluxo de dados de um equipamento central;
b) coletar e armazenar o fluxo de dados;
c) analisar o fluxo de dados;
d) exibir o resultado da análise na forma de gráficos e tabelas.

1.2 ESTRUTURA

A monografia está dividida em quatro capítulos. O primeiro capítulo apresenta ao leitor a introdução e seus objetivos. O segundo capítulo apresenta os conceitos e técnicas mais

relevantes para a criação deste e três trabalhos correlatos. No terceiro capítulo é descrito o processo de desenvolvimento, mais especificamente os requisitos, especificação, técnicas e ferramentas utilizadas na implementação e análise dos resultados. Por fim, o quarto capítulo apresenta as conclusões e sugestões de extensão para futuros trabalhos.

2 FUNDAMENTAÇÃO TEÓRICA

Este capítulo está dividido em três seções. A seção 2.1 apresenta o que é gerenciamento de redes e o modelo criado pela ISO, que divide o gerenciamento em 5 áreas. A seção 2.2 aborda o que é fluxo ou fluxo de tráfego, sua história e os processos desde a coleta até a análise dos dados. Por último, a seção 2.3 descreve os trabalhos correlatos.

2.1 GERENCIAMENTO DE REDES

"O gerenciamento de redes é um serviço que utiliza uma variedade de ferramentas, aplicativos e dispositivos para auxiliar os administradores de redes no monitoramento e manutenção de redes." (GUPTA, 2006, p. 483, nossa tradução). Segundo Raman (1998), a International Organization for Standardization (ISO), com o intuito de permitir que duas redes distintas se comuniquem, criou um padrão de comunicação de dados dividido em 5 categorias: configuração, falha, desempenho, segurança e contabilização. Raman (1998) e Kurose, Jim F. ; Ross (2013) definem as cinco categorias como:

a) configuração: o gerenciamento de configuração permite que o administrador de rede estime recursos e serviços, e monitore e controle o estado e status das informações de hardware e software;
b) falha: o gerenciamento de falha trata da vigilância de alarmes, de reportar alarmes, teste e isolamento de falhas;
c) desempenho: o gerenciamento de desempenho monitora parâmetros de performance, como por exemplo: segundos errados, número de mensagens mal formatadas, estatísticas de tráfego e ainda permite controlar o tráfego para que não haja congestionamento. O gerenciamento de falha e gerenciamento de desempenho são parecidos, a diferença é que o gerenciamento de falhas faz o tratamento imediato e o gerenciamento de desempenho faz o tratamento a longo prazo;
d) segurança: o gerenciamento de segurança controla o acesso aos recursos da rede, como por exemplo o controle de acesso, autenticação, criptografia dos campos de dados e ainda a criptografia da conexão;
e) contabilização: o gerenciamento de contabilização coleta o uso de dados dos usuários para serem usados pelo provedor de serviço. Assim é possível que eles cobrem por utilização.

Este padrão, também conhecido como modelo de gerenciamento de redes OSI FCAPS, é utilizado tanto no gerenciamento de redes do modelo OSI, como na gerencia de rede de telecomunicações (RAMAN, 1998).

2.1.1 Gerência de Desempenho

Segundo Stallings (2005), o administrador tem a obrigação de acompanhar os diferentes níveis de indicadores de desempenho, afim de estar pronto para responder as indagações do usuário e porque permite que uma ação precisa seja tomada de imediato e a longo prazo. São diversos os dispositivos que podem ser monitorados.

> O gerenciamento de desempenho; portanto, precisa monitorar muitos recursos para fornecer informações sobre o nível operacional da rede. Coletando essas informações, analisando-as e, depois, usando a análise resultante como feedback para o conjunto de valores prescrito, o gerente de rede pode se tornar cada vez mais apto a reconhecer situações indicativas de queda de desempenho atual ou iminente (STALLINGS, 2005, p. 412).

Segundo Stallings (2005), o gerenciamento de desempenho pode ser dividido em duas categorias: o monitoramento que observa as atividades na rede e o controle que intervém de modo a interferir no desempenho da rede.

2.1.2 Categorias de Monitoramento

O monitoramento da rede pode ser dividido em dois grupos: métodos passivos e ativos. Os métodos passivos monitoram o tráfego por observação enquanto os métodos ativos injetam pacotes na rede, como por exemplo o comando *ping*, que envia um pacote ICMP para o *host* de destino e assim é possível determinar o status da conexão de ponta a ponta e calcular o tempo de ida e volta entre a origem e o destino (DOERR; KUIPERS; VAN ADRICHEM, 2014) (HOFSTEDE et al., 2014) (GREŽO; NAGY, 2017).

Segundo Fernandez et al. (2017) e Hofstede et al. (2014), entre os métodos disponíveis nos dias de hoje, o tradicionalmente escolhido é o passivo, pois permite uma profunda análise da rede e das conexões. No método passivo é possível, por exemplo, capturar os pacotes por completo, isso significa capturar não somente os cabeçalhos, mas também os dados (*payload*). Em redes de alta velocidade onde o tráfego passa de 100 Gbps, a captura de pacotes exige servidores de alta performance e infraestrutura substancial para arquivamento e análise. Um método passivo que é muito utilizado por ser mais escalável é a exportação do fluxo, onde os pacotes são agrupados em fluxos e exportados para arquivamento e posterior análise.

A exportação do fluxo possui diversas outras vantagens em comparação à captura de pacotes. Segundo Hofstede et al. (2014), os principais pontos positivos são:

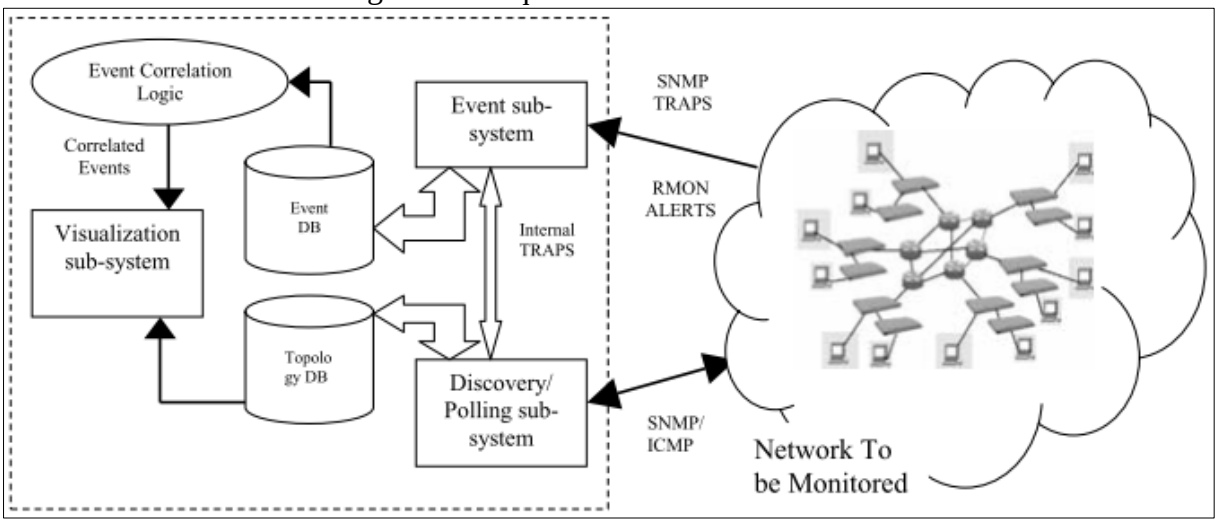
a) grande quantidade dos ativos de rede, como por exemplo *switches*, roteadores e *firewalls* serem capazes de exportar fluxo de dados;
b) ser amplamente utilizado pela comunidade para análise de segurança, planejamento de capacidade, contabilidade, entre outros;
c) redução do volume de dados, na ordem de 1/2000 do volume original;
d) privacidade do usuário, uma vez que apenas os cabeçalhos dos pacotes são analisados.

A exportação do fluxo reduz o volume de dados, mas isso não significa que os dados não possam ultrapassar a casa dos terabytes. Essa grande massa de dados, considerando que hoje em dia é possível adicionar mais campos no fluxo de dados, ou seja, mais informações sobre o usuário, pode ser considerada um “Big Data” (HOFSTEDE et al., 2014).

2.1.3 Arquitetura de uma NMS

A arquitetura padrão de uma Network Management Station (NMS), ou seja, uma Estação de Gerenciamento de Rede é exibida na figura 1.

Figura 1 – Arquitetura de uma rede NMS



Fonte: Gupta (2006, p. 2).

Normalmente redes NMS focam no gerenciamento de falhas e posteriormente na solução da falha. O primeiro passo geralmente é a identificação da topologia da rede e depois a pesquisa de todos os hosts disponíveis na rede. Essa pesquisa ocorre em intervalos de tempo fixo por meio do ICMP e o Simple Network Management Protocol (SNMP). O ICMP vem da sigla em inglês Internet Control Message Protocol que significa Protocolo de Controle de Mensagem da Internet. Este protocolo faz parte do protocolo IP. Qualquer host que utilize o protocolo IP aceita necessariamente o protocolo ICMP. O SNMP é um protocolo da camada de aplicação utilizado para troca de informações de gerenciamento entre os hosts da rede. Esse

tipo de pesquisa traz algumas desvantagens, como por exemplo, o fato da falha de rede não ser identificada de imediato (GUPTA, 2006).

Redes NMS não contemplam necessariamente as cinco áreas de gerenciamento de redes que são especificadas pela ISO, mas se concentram somente em gerenciamento de falhas e de certa forma no gerenciamento de desempenho. Outra desvantagem é que ela é centralizada, isso quer dizer que na maioria das vezes possui uma única estação de gerenciamento, o que representa um único ponto de falha. Caso algum agente SNMP mal-intencionado resolvesse enviar uma mensagem mal formatada, isso poderia derrubar a estação de gerenciamento (GUPTA, 2006).

2.2 FLUXO

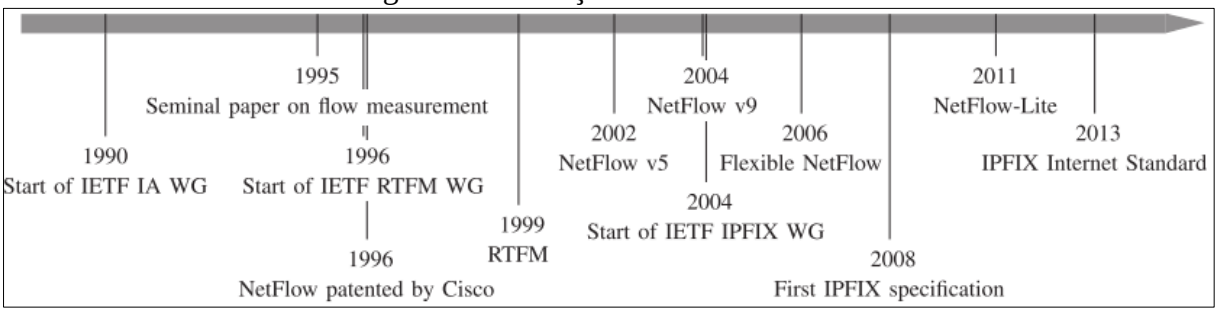
Fluxo ou fluxo de tráfego é definido como "um conjunto de pacotes ou quadros passando por um ponto de observação na rede durante um determinado intervalo de tempo. Todos os pacotes pertencentes a um fluxo têm um conjunto de propriedade comuns." (TRAMMELL; WAGNER; CLAISE, 2013, p. 6, nossa tradução). Ainda segundo Trammell, Wagner, Claise (2013), um pacote pertence a um fluxo somente se ele satisfaz completamente todas as propriedades definidas do fluxo. Cada propriedade é resultado da aplicação de uma função aos valores de um ou mais campos do cabeçalho do pacote (por exemplo, endereço IP de destino), campos do cabeçalho do transporte (por exemplo, número de porta de destino) ou campos do cabeçalho de aplicação (por exemplo, campos do cabeçalho Real-Time Transport Protocol (RTP), dentre outros.

2.2.1 História

Segundo Hofstede et al. (2014), o primeiro registro sobre exportação de fluxo de dados foi feito em 1991 pelo grupo de trabalho da Internet Accounting (IA) da Internet Engineering Task Force (IETF). Entretanto o grupo finalizou o trabalho dois anos depois devido à falta de interesse do fornecedor e o forte entendimento que a internet deveria ser livre, ou seja, sem captura de tráfego, contabilidade, monitoramento e etc. O assunto voltou a ser estudado em 1995 e resultou em uma metodologia para concepção de fluxo de dados com base na agregação de pacotes. No ano seguinte, a IETF criou um grupo de trabalho denominado Realtime Traffic Flow Measurement (RTFM) com a finalidade de averiguar problemas com a medição do tráfego, resultando assim, em 1999, na criação de uma arquitetura universal para medição do fluxo de dados que foi chamada de RTFM Traffic Measurement System. Contudo novamente a falta de interesse dos fornecedores não permitiu a criação de um padrão para exportação de

fluxo. Simultaneamente ao RTFM, foi criado pela Cisco, em 1996, uma tecnologia proprietária chamada NetFlow, porém a tecnologia teve ampla adoção somente em 2002, com o NetFlow versão 5. Dois anos depois a Cisco disponibiliza o NetFlow versão 9, onde é possível adicionar mais informações ao fluxo de dados. A figura 2 mostra os principais acontecimentos nesta área na linha do tempo.

Figura 2 – Evolução do fluxo de dados



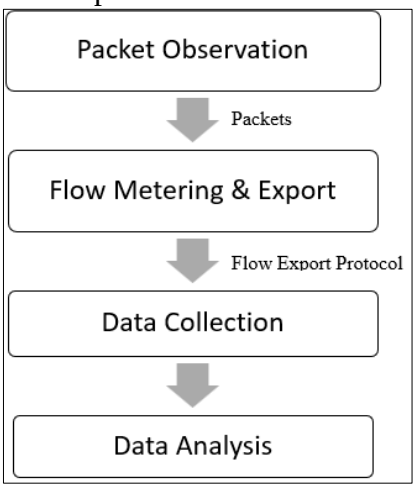
Fonte: Hofstede et al (2014, p. 2).

Além do NetFlow da Cisco, existem outras tecnologias semelhantes, como por exemplo o J-Flow desenvolvido pela empresa Juniper e o sFlow (Sampling Flow) desenvolvido pela InMon Corporation. Segundo Hofstede et al. (2014), como ainda não existia uma tecnologia padrão, a IETF criou um grupo de trabalho, em 2004, chamado IP Flow Information Export (IPFIX) e este grupo elegeu o NetFlow versão 9 como base para o novo protocolo IPFIX. Vale ressaltar que o IPFIX não é somente a versão universal do NetFlow versão 9, ele suporte vários outros novos recursos.

2.2.2 Etapas do monitoramento do fluxo de dados

O monitoramento do fluxo de dados é dividido em várias etapas. (FERNANDEZ et al., 2017) (HOFSTEDE et al., 2014). Conforme mostra a figura 3 a primeira etapa é a observação dos pacotes. Nesta fase os pacotes são capturados e pré-processados. A segunda etapa é a medição e exportação do fluxo de dados. Dentro do processo de medição os pacotes são agregados em fluxos, e depois disso é criado um registro que por sua vez é colocado em um datagrama do protocolo IPFIX e é exportado através do processo de exportação. As etapas seguintes são responsáveis pela coleta, armazenamento, pré-processamento e por último análise dos dados (HOFSTEDE et al., 2014). Segundo Hofstede et al. (2014), o processo de observação, medição e exportação são normalmente feitos em um único dispositivo.

Figura 3 – Etapas do monitoramento do Fluxo

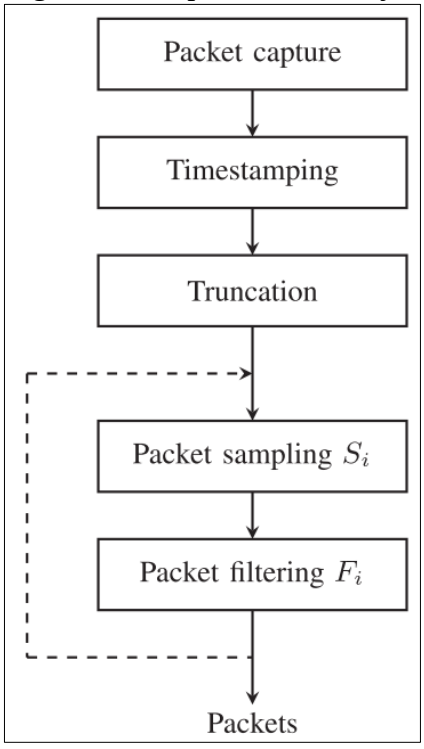


Fonte: Hofstede et al (2014, p. 4).

2.2.2.1 Observação dos Pacotes

Nesta etapa, conforme dito anteriormente, é feita a captura e pré-processamento dos dados. O processo de captura ocorre a partir de um Ponto de Observação (HOFSTEDE et al., 2014). Um Ponto de observação é “um local da rede onde os pacotes podem ser observados. Como por exemplo uma ou mais portas de um roteador.” (TRAMMELL; WAGNER; CLAISE, 2013, p. 6, nossa tradução). A figura 4 mostra exatamente quais os passos que ocorrem na observação dos dados.

Figura 4 – Etapas da observação

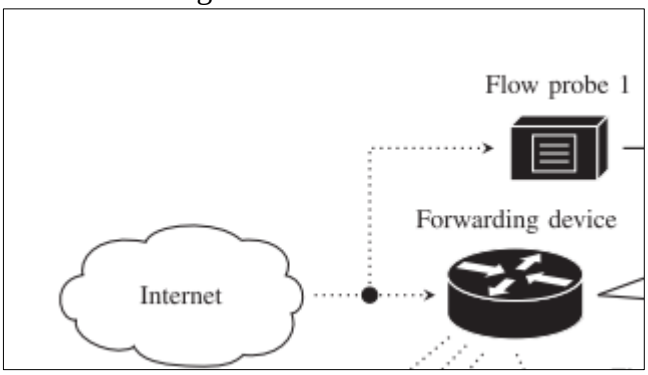


Fonte: Hofstede et al (2014, p. 5).

A captura dos pacotes ocorre na placa de rede, também conhecida como Network Interface Card (NIC), depois passa por algumas verificações, como por exemplo *checksum error*, é armazenada em *buffers* e por último é enviado para memória da máquina. O *timestamping* é o registro de data e hora dos pacotes e exerce uma função vital, pois é utilizado como parâmetro para várias funções, como por exemplo, a fusão dos pacotes de diferentes pontos de observação. Os próximos passos mostrados na figura 4 não são necessariamente executados. O *truncation* seleciona apenas os dados do pacote que se encaixam em um layout pré-definido, ignorando os dados restantes. Por fim a etapa de *sampling* e *filtering* de pacotes, que significa amostragem e filtragem respectivamente, permitem que somente alguns pacotes sejam selecionados para medição, diminuindo a quantidade de processos e consequentemente o consumo da banda. (HOFSTEDE et al., 2014).

A captura de pacotes normalmente acontece em redes cabeadas, porém não necessariamente. Isso ocorre pelo fato de a rede cabeada ter baixa interferência externa e alta velocidade. Em redes cabeadas, a captura de pacotes pode ser no modo *in-line* ou espelhamento. No modo *in-line* o dispositivo de captura está diretamente conectado entre o link monitorado (HOFSTEDE et al., 2014). Conforme mostra a figura 5, onde o ponto preto representa o dispositivo de captura.

Figura 5 – Mono *in-line*



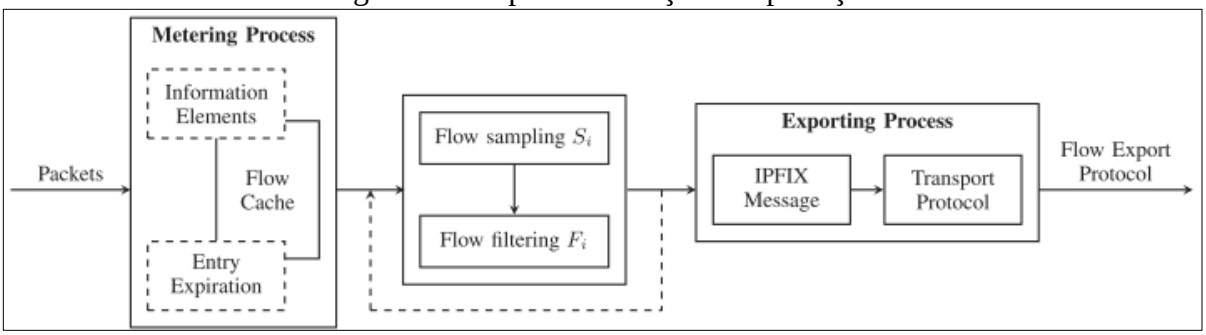
Fonte: Hofstede et al (2014, p. 4).

No modo de espelhamento, mais conhecido como *port mirroring*, o dispositivo espelha uma porta, ou seja, os pacotes que trafegam por uma determinada porta são copiados para outra porta, onde os pacotes são capturados (HOFSTEDE et al., 2014).

2.2.2.2 Medição e Exportação do Fluxo de Dados

Nesta etapa os pacotes são reunidos dentro de um fluxo e o registro do fluxo é exportado. A figura 6 mostra quais as etapas desde a medição até a exportação.

Figura 6 – Etapas de medição e exportação



Fonte: Hofstede et al (2014, p. 7).

O procedimento de reunir os pacotes acontece no processo de medição, utilizando para isso os elementos definidos no layout do fluxo. Estes elementos são denominados Information Elements (IEs). Os IEs possuem nome, ID numérico, descrição, tipo, tamanho e status. Um subconjunto de IEs frequentemente utilizado é mostrado no quadro 1. Neste subconjunto os IEs são retirados somente da camada de transporte e camada de rede, quando é possível utilizar os IEs da camada de enlace até a camada de aplicação (HOFSTEDE et al., 2014).

Quadro 1 – Lista de IEs

ID	Name	Description
152	flowStartMilliseconds	Timestamp of the flow's first packet.
153	flowEndMilliseconds	Timestamp of the flow's last packet.
8	sourceIPv4Address	IPv4 source address in the packet header.
12	destinationIPv4Address	IPv4 destination address in the packet header.
7	sourceTransportPort	Source port in the transport header.
11	destinationTransportPort	Destination port in the transport header.
4	protocolIdentifier	IP protocol number in the packet header.
2	packetDeltaCount	Number of packets for the flow.
1	octetDeltaCount	Number of octets for the flow.

Fonte: Hofstede et al (2014, p. 8).

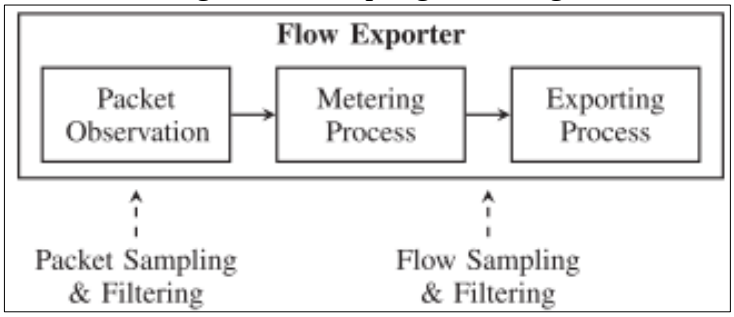
Conforme mostra a figura 6, depois do processo de reunir os pacotes, vem o processo de *sampling* e *filtering*. Porém, antes de chegar no processo de *sampling* e *filtering* acontece algo muito importante: o cache do fluxo. O cache do fluxo consiste em manter "tabelas dentro do processo de medição que armazenam informações relativas a todos os fluxos de tráfego de rede ativos" (HOFSTEDE et al., 2014, p. 2045, nossa tradução). Os dados da tabela nada mais são do que IEs que podem ser um campo chave ou não do layout de exportação. Utilizando esses dados é que os fluxos podem ser agrupados corretamente. Os campos chaves que normalmente são usados são o endereço IP e o número da porta de origem e de destino. Os campos que não são chaves são usados para coletar características do fluxo, como contadores de pacotes e bytes (HOFSTEDE et al., 2014).

As informações que são adicionadas no cache do fluxo permanecem lá até que o fluxo seja considerado como terminado, após isso as informações são consideradas expiradas. Para que as entradas do cache sejam consideradas como expiradas, o IPFIX fornece algumas regras:

a) tempo limite ativo – o fluxo está ativo por um determinado tempo;
b) tempo limite ocioso – nenhum pacote foi agregado ao fluxo em um período determinado;
c) restrições de recursos – heurísticas especiais.

O processo de *sampling* e *filtering* de registros é uma forma de selecionar um conjunto de *flow records* e assim diminuir o processamento nos próximos estágios. Diferente do processo de *sampling* e *filtering* de pacotes, o processo de *sampling* e *filtering* de registros acontece depois do processo de medição, mas as técnicas de *sampling* e *filtering* são semelhantes (HOFSTEDE et al., 2014). A figura 7 mostra onde ocorre o processo de *sampling* e *filtering* de pacotes e registros.

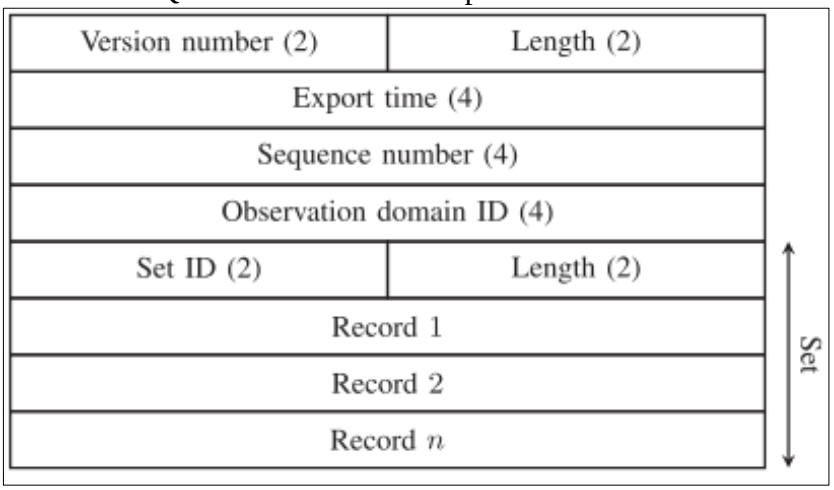
Figura 7 – Sampling e filtering



Fonte: Hofstede et al (2014, p. 10).

Depois do processo de *sampling* e *filtering* é feita a exportação, ocasião em que entra em campo o protocolo IPFIX. No quadro 2 é possível visualizar uma versão simplificada do formato da mensagem do protocolo IPFIX.

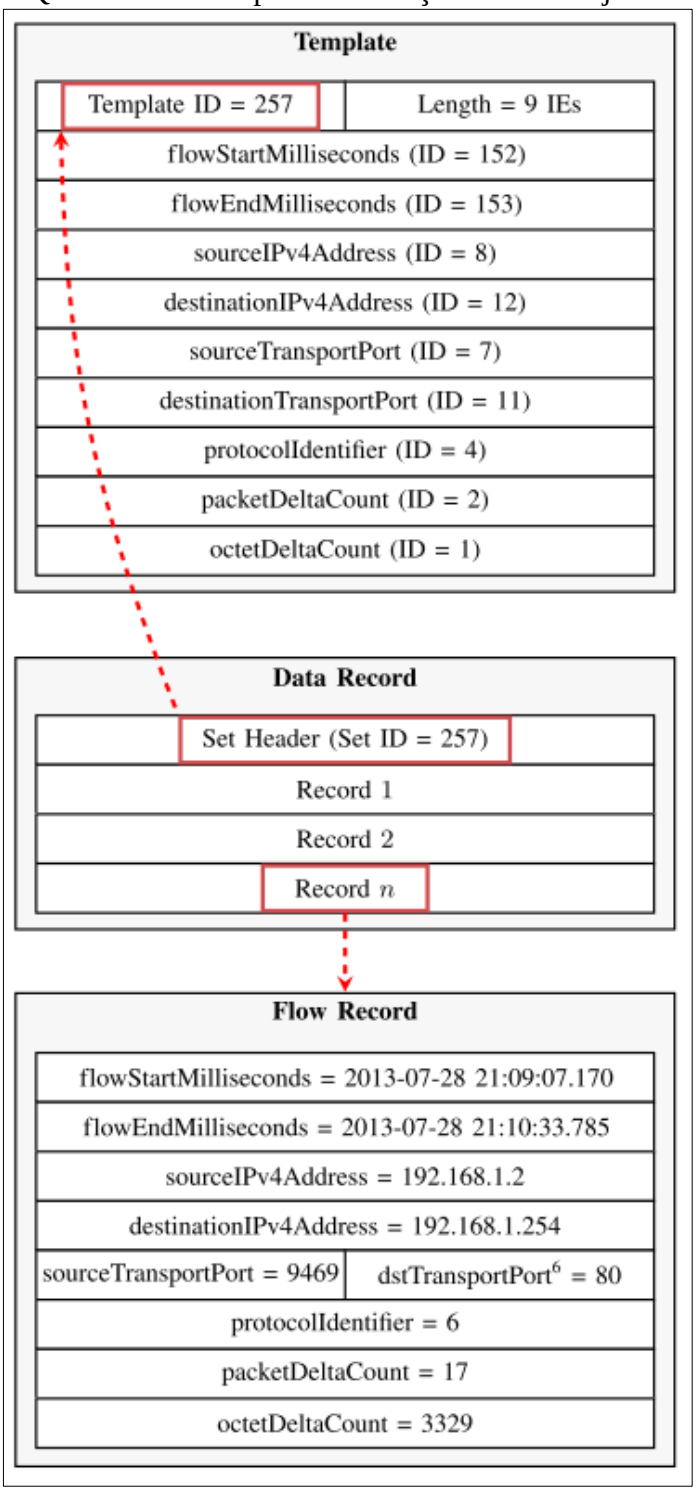
Quadro 2 – Formato do protocolo IPFIX



Fonte: Hofstede et al (2014, p. 11).

No quadro são mostrados o nome e o tamanho dos campos em bytes. Os primeiros 16 bytes da mensagem formam o cabeçalho da mensagem e possuem o número da versão do protocolo, tamanho da mensagem, tempo de exportação e o ID no domínio de observação. Depois do cabeçalho vem um ou mais conjuntos (sets), cada conjunto possui um ID e um comprimento (HOFSTEDE et al., 2014). Um conjunto é "uma coleção de registros que possuem uma estrutura semelhante, prefixada por um cabeçalho. Existem três tipos diferentes de conjuntos: Template Sets, Data Sets e Options Template Sets" (HOFSTEDE et al., 2014, p. 2047, nossa tradução). No quadro 3 é exibido um exemplo de como é formado um conjunto. O Template é o cabeçalho do conjunto, onde estão definidos o ID 257 e 9 IEs. No Data Record está definido o ID do template e uma lista de registros. Por último o Flow Record que possui os 9 IEs definidos no template.

Quadro 3 – Exemplo da formação de um conjunto



Fonte: Hofstede et al (2014, p. 11).

A quantidade de registros é normalmente limitada para evitar a fragmentação do IP, que ocorre no processo de exportação (HOFSTEDE et al., 2014). “O tamanho máximo de mensagens exportadas deve ser configurado de tal forma que o tamanho total do pacote não exceda o Path Maximum Transmission Unit (PMTU). Se o PMTU é desconhecido, um tamanho máximo de pacote de 512 octetos deve ser usado.” (TRAMMELL; WAGNER; CLAISE, 2013).

Depois que todos os campos da mensagem IPFIX estiverem completos, um protocolo de transporte deve ser selecionado. O IPFIX permite a utilização do SCTP, TCP e UDP. O SCTP e o TCP possuem várias vantagens, mas não são amplamente usados devido a sua performance. O protocolo mais utilizado para exportação do fluxo é o UDP, pois é de fácil implementação, mas deve ser usado com cuidado porque não garante a entrega dos pacotes e a sequência que os pacotes são entregues. Durante ataques como Denial-of-Service (DoS) o protocolo UDP pode se tornar inútil, pois o volume de registros aumenta drasticamente. O IPFIX possui ainda um formato próprio chamado IPFIX File Format e neste formato podem ser utilizados os protocolos SSH, HTTP, FTP e NFS (HOFSTEDE et al., 2014).

2.2.2.3 Coleta dos fluxos

Esta etapa é responsável pela coleta, armazenamento e pré-processamento dos fluxos. A coleta dos dados pode ser realizada por um ou mais processos de coleta. No processo de armazenamento os dados podem ser armazenados em vários formatos. O desempenho e a funcionalidade da coleta dos fluxos são alterados de acordo com o formato de armazenamento que é utilizado. Existem dois formatos de armazenamento, o volátil e o persistente. O volátil é o mais rápido, pois é executado na memória. O persistente é utilizado para armazenar dados por mais tempo, por isso é mais lento, o que pode acarretar em um gargalo no armazenamento devido essa diferença de velocidade. Normalmente os dados têm de ser armazenados por um longo período, como por exemplo, quando tem a necessidade de cumprir leis de retenção de dados (HOFSTEDE et al., 2014). Segundo Hofstede et al. (2014), o armazenamento persistente pode salvar os dados em:

a) arquivo simples – muito veloz ao ler e gravar arquivos, no entanto tem recursos de consulta limitado. Um arquivo simples pode ser um arquivo binário e de texto;
b) banco de dados orientado a linhas – é utilizado em sistemas de gerenciamento de banco de dados. Na consulta é feita a leitura completa da linha mesmo se somente parte dos dados é necessária;
c) banco de dados orientado a colunas – na consulta, somente o dado requisitado é lido.

Hofstede et al. (2014) faz uma análise desses três formatos. Na questão de espaço em disco, o arquivo simples supera os demais. Na inserção, o arquivo simples também lidera, porém na consulta, o FastBit, que é um banco de dados orientado a colunas, supera o arquivo simples. É possível melhorar o desempenho dos três formatos utilizando o armazenamento em RAID, contudo é necessário a utilização de um sistema de gerenciamento que decida como os dados vão ser distribuídos.

2.2.2.4 Análise dos Dados

Esta é a etapa final do monitoramento de fluxo de dados e segundo Hofstede et al. (2014), ela pode ser dividida em três áreas: análise e relatório do fluxo, detecção de ameaças e monitoramento do desempenho.

Análise e relatório do fluxo são funções mais básicas e são as que normalmente vários aplicativos fornecem. Elas geralmente permitem navegar e filtrar os dados do fluxo, oferecem uma visão geral das estatísticas, como por exemplo os dispositivos que mais enviam pacotes e emite relatórios e alertas, como por exemplo quando o limite de tráfego foi excedido. Com a análise dos dados é possível identificar os momentos em que o gráfico se comporta de maneira incomum e a partir daí verificar mais afundo os dados daquele período, podendo assim constatar o que acontecia naquele momento, utilizando os métodos da detecção de ameaças (HOFSTEDE et al., 2014).

A detecção de ameaças pode ser dividida em dois tipos de uso, primeiro identificar com quais hosts a máquina investigada se conectou e segundo analisar a massa de dados para identificar certos tipos de ameaças, como ataques Distributed-Denial-of-Service (DDoS), uma varredura na rede, disseminação de *worms* e *botnets*. O segundo tipo de uso utiliza a própria sequência do fluxo de dados para identificar os tipos de ataques, como por exemplo em ataques de força bruta no serviço SSH, onde é possível identificar um padrão, pois muitos usuário e senhas são testados e na sequência o serviço SSH interrompe a conexão depois de um número de tentativas de login. Isso é facilmente identificado no fluxo de dados, pois resulta em muitas conexões TCP com pacotes de tamanhos parecidos (HOFSTEDE et al., 2014).

O monitoramento do desempenho consegue mostrar o status do serviço que está sendo executado na rede. Algumas das métricas que são possíveis monitorar são Round-Trip-Time (RTT), *delay*, *jitter*, tempo de resposta, perda de pacotes e uso da rede. As aplicações que fazem o monitoramento do desempenho podem ser dividias em dois grupos. O primeiro grupo utiliza os IEs disponíveis no próprio fluxo para estimar o desempenho, porém somente eles não são suficientes em determinadas circunstâncias. O segundo grupo utiliza um exportador de fluxos que extrai métricas de desempenho, como por exemplo o nProbe. nProbe e outras aplicações avançadas permitem, por exemplo exportar a latência de um servidor Web (HOFSTEDE et al., 2014).

2.3 TRABALHOS CORRELATOS

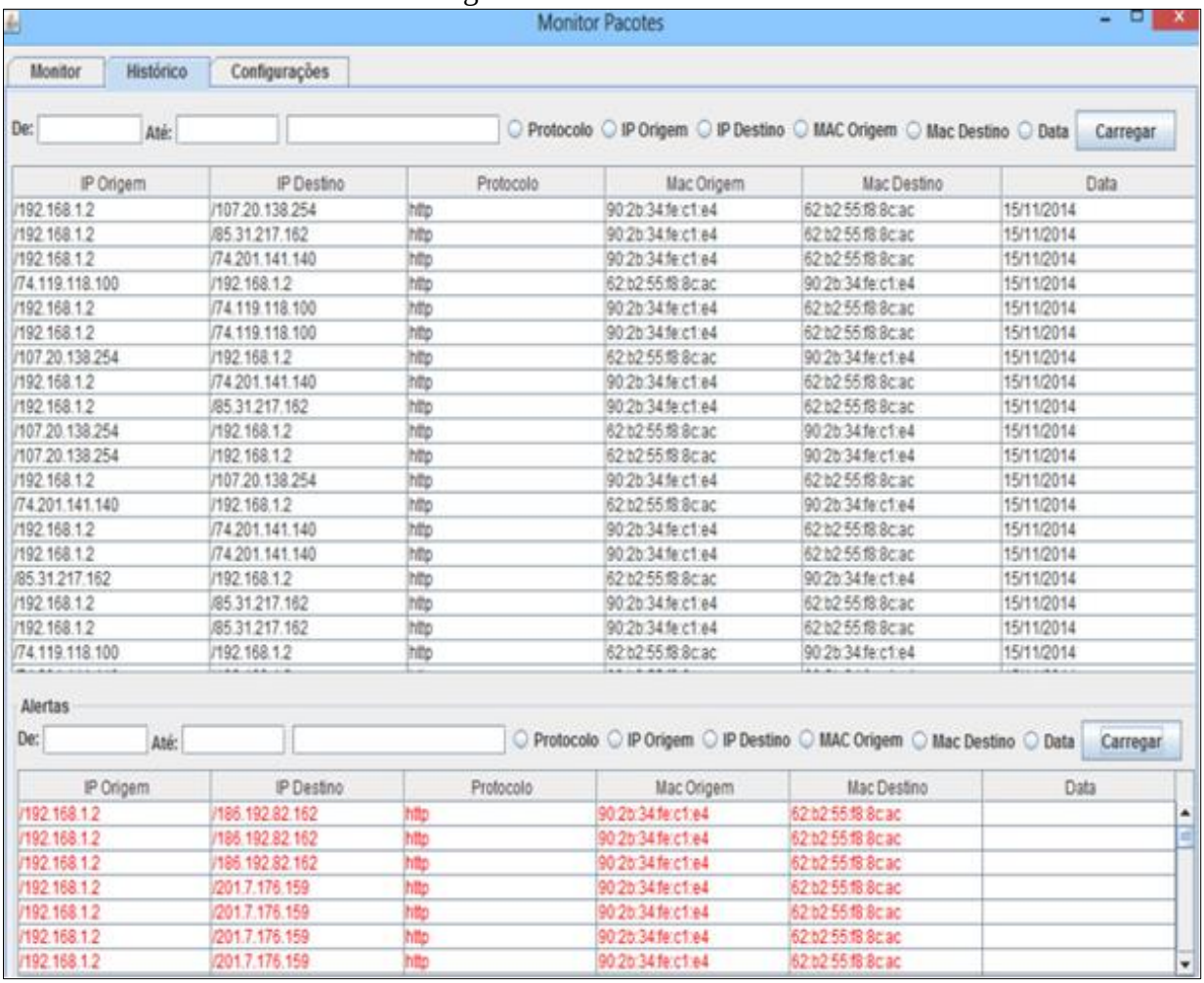
Serão apresentados três trabalhos correlatos, sendo que todos são trabalhos acadêmicos que atuam na área de monitoramento de redes. Bennertz (2014) desenvolveu uma ferramenta

para monitoração e gerenciamento do tráfego da interface local. Karing (2002) desenvolveu um protótipo de um sistema de monitoramento de desempenho para dispositivos de uma LAN, utilizando o protocolo SNMP V3 e o trabalho de Lingnau (2012) é um sistema de monitoramento de servidores e dispositivos de rede, capaz de coletar dados de desempenho de dispositivos de rede utilizando o protocolo SNMP.

2.3.1 Ferramenta para Monitoramento e Gerenciamento de Tráfego em uma Rede Local

Bennertz (2014) desenvolveu uma ferramenta para monitoração e gerenciamento do tráfego da interface local, ou seja, é possível monitorar e gerenciar o tráfego de apenas um desktop. As técnicas e ferramentas utilizadas no desenvolvimento foram a linguagem de programação Java na versão 7.0, o ambiente de desenvolvimento Eclipse na versão Juno, a biblioteca JPCAP, a biblioteca Winpcap e o banco de dados relacional Oracle na versão 10g. O processo consiste na coleta e análise dos pacotes capturados. A biblioteca Winpcap é uma biblioteca para ambientes Windows e é responsável por conceder acesso a interface de rede em modo promíscuo, permitindo que todos os pacotes sejam capturados. Já a biblioteca JPCAP é uma biblioteca Java *open source* responsável pela captura dos pacotes. A ferramenta desenvolvida permite que o usuário selecione a interface de rede responsável pela captura, visualize os pacotes capturados, consulte os pacotes capturados, filtre informações com base no IP de origem e destino dentre outras e defina alertas com base no IP de origem e destino dentre outros. Na figura 8 é possível ver a tela de busca da ferramenta.

Figura 8 – Tela de busca



Fonte: Bennertz (2014, p. 45).

Segundo Bennertz (2014), o protótipo demonstrou ser uma ferramenta de monitoração eficaz para redes Ethernet, pois a captura e visualização das informações são realizadas de forma simples, flexível e rápida. O conteúdo monitorado é visualizado em tempo real, e ao mesmo tempo que visualiza as informações, o administrador pode salvar as informações na base de dados. Porém, salvar uma grande quantidade de informações ao mesmo tempo na base de dados, poderá resultar em uma pequena lentidão no processo. Segundo o autor, as informações foram armazenadas com sucesso e as funções de consulta e exclusão das informações foram feitas de forma clara e específica. Com isso, a ferramenta pode extrair informações úteis para serem analisadas posteriormente, possibilitando tomar decisões rápidas e eficazes.

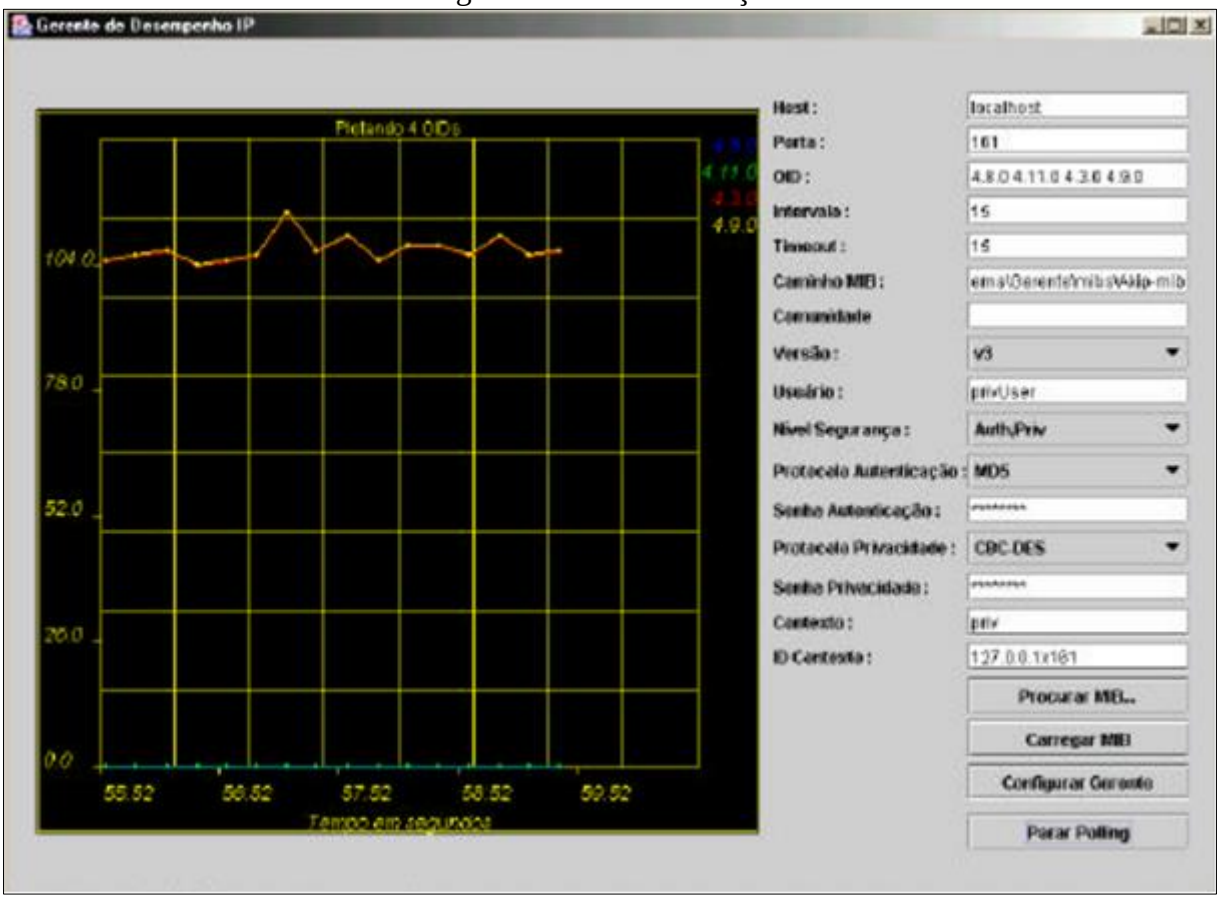
2.3.2 Protótipo de um Sistema de Monitoramento de Desempenho de Redes de Computadores Baseado no Protocolo SNMP V3

Karing (2002) desenvolveu um protótipo de um sistema de monitoramento de desempenho para dispositivos de uma LAN utilizando o protocolo SNMP V3. O protótipo foi desenvolvido em Java, utilizando API SNMP V3 da empresa AdventNet e o sistema operacional Windows 2000 Server nos testes. O monitor possui dois elementos:

a) os agentes, que são responsáveis por fornecer informações das máquinas hospedeiras, tais como endereço IP, informações do sistema operacional, uso da interface de rede, dentre outras informações;
b) o gerente, que solicita os dados dos agentes e expõe em tempo real o gráfico de desempenho.

O agente permite que o usuário indique em que porta ele deve aguardar as requisições, a versão do SNMP que o agente irá suportar, o nível de depuração a ser utilizado pelo agente, entre outras informações. Já o gerente possibilita que o usuário configure todas as opções utilizadas pelo SNMP V3 conforme mostra a figura 9, onde pode-se ver o gráfico e as opções de configuração do SNMP V3.

Figura 9 – Tela de medição



Fonte: Karing (2002, p. 112).

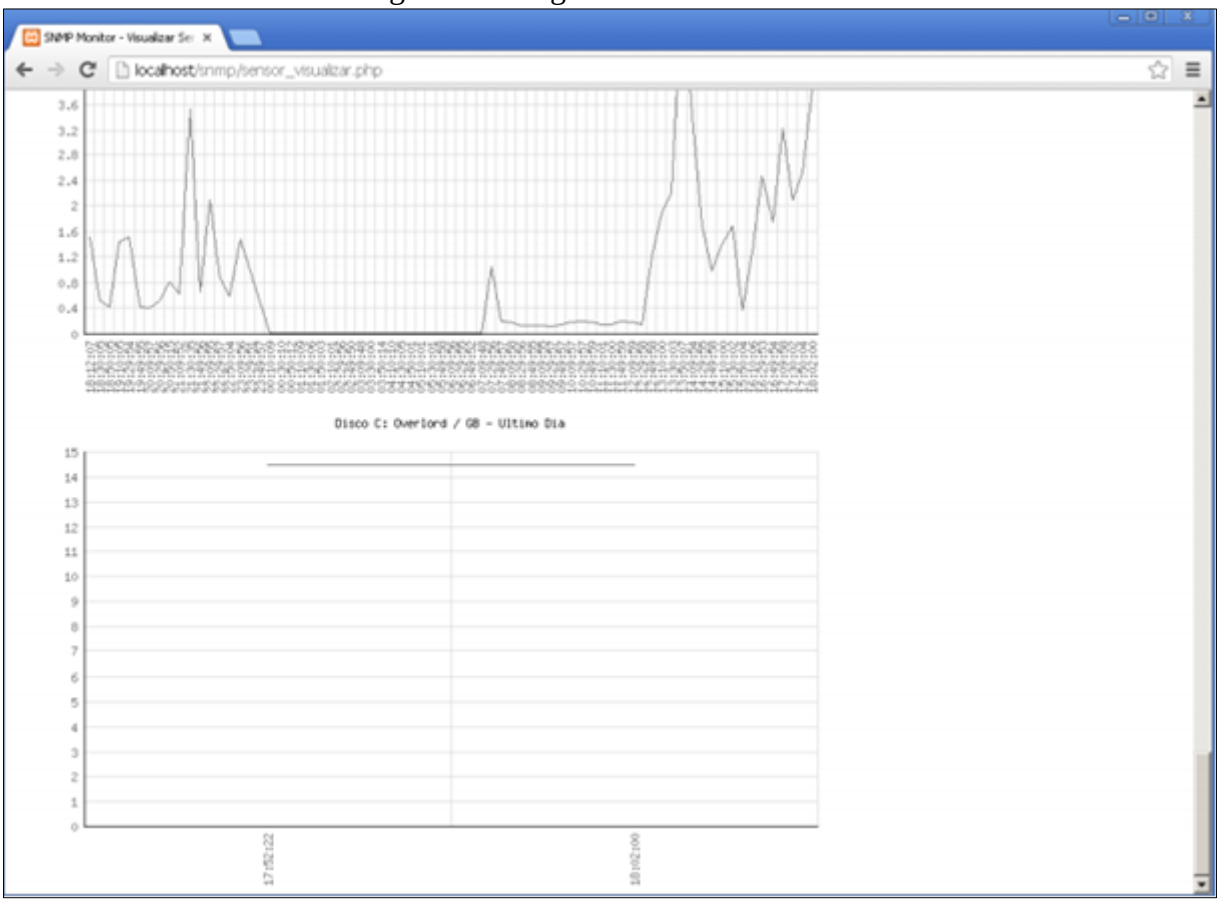
Karing (2002) ressalta que a utilização do SNMP como plataforma de gerenciamento de redes é prática e fácil, tornando possível realizar o gerenciamento de forma segura e eficaz, porém o SNMP V3 possui controle de acesso e privacidade, o que o torna complexo. Uma das limitações do protótipo é o fato de o gerente não realizar nenhum processamento ou tomada de decisão sobre os dados coletados. Além disso, apesar do protótipo ter sido desenvolvido em Java, parte da portabilidade foi perdida, devido ao uso de um método nativo. Ele menciona ainda que o objetivo de gerenciar o desempenho real de uma rede de computadores não foi atingido completamente, pois havia pouca literatura existente e não havia tempo hábil para realizar um estudo mais detalhado das funções de gerenciamento de desempenho existentes.

2.3.3 Monitoramento de Servidores e Dispositivos de Rede Utilizando SNMP

Lingnau (2012) desenvolveu um sistema de monitoramento de servidores e dispositivos de rede capaz de coletar dados utilizando o protocolo SNMP V2. Além de coletar esses dados, o sistema gera gráficos dentro de um website com acesso restrito por usuário e senha. O sistema também permite enviar alertas por e-mail quando houver alguma leitura incomum. As técnicas e ferramentas utilizadas no desenvolvimento foram a linguagem PHP, bibliotecas do PHP, Java

script para permitir funcionalidades do sistema, Apache como servidor web e banco de dados MySQL. Na figura 10 é possível visualizar a página de monitoramento que apresenta dois gráficos com estatísticas de uso do disco C.

Figura 10 – Pagina de monitoramento



Fonte: Lingnau (2012, p. 42).

Lingnau (2012) diz que os resultados obtidos foram satisfatórios. Além disso, também foram desenvolvidos mecanismos para facilitar a utilização por usuários que não estejam habituados ou que não conheçam o SNMP, tornando o sistema mais fácil e acessível. O sistema possui algumas limitações como por exemplo, o fato de que suporta apenas o protocolo SNMP V2 e o fato de que permite criar apenas um limiar por sensor.

3 DESENVOLVIMENTO DA FERRAMENTA

Neste capítulo serão apresentadas as etapas do desenvolvimento da ferramenta. Na seção 3.1 são apresentados os requisitos funcionais e não funcionais estabelecidos. A seção 3.2 apresenta a especificação da ferramenta com diagramas. A seção 3.3 aborda os principais pontos da implementação e por fim na seção 3.4 são apresentados os casos de testes e os respectivos resultados.

3.1 REQUISITOS

Abaixo estão listados os Requisitos Funcionais (RF) e os Não Funcionais (RNF) previstos para ferramenta. A ferramenta deve:

a) exibir um gráfico em linha e barra com o total de fluxos (RF);
b) exibir um gráfico em linha e barra do consumo diário em GB e dos últimos 5 minutos em MB (RF);
c) exibir um gráfico em linha dos protocolos TCP, UDP e ICMP (RF);
d) exibir um gráfico em pizza dos protocolos utilizados nos últimos 5 min (RF);
e) exibir um gráfico em pizza dos top 10 IPs que mais transferiram dados (RF);
f) exibir um gráfico em pizza das top 10 origens que mais transferiram dados (RF);
g) exibir um gráfico em pizza dos top 10 destinos que mais transferiram dados (RF);
h) exibir um gráfico em pizza das top 10 portas mais utilizadas (RF);
i) possibilitar que o usuário escolha visualizar os dados do gráfico das últimas 24 horas ou 7 dias (RF);
j) rodar na plataforma Linux (RNF);
k) ser desenvolvido para plataforma WEB (RNF);
l) utilizar HTML 5, CSS, Bootstrap, JQuery e Java Script no front end (RNF);
m) utilizar a biblioteca Chart.js para criar gráficos (RNF);
n) ser acessível pelo navegador Google Chrome (RNF);
o) coletar o fluxo de dados exportado com a ferramenta Nfcap (RNF);
p) utilizar a ferramenta nfdump para leitura e análise dos registros (RNF);
q) armazenar os resultados obtidos da leitura e análise em arquivos JSON (RNF).

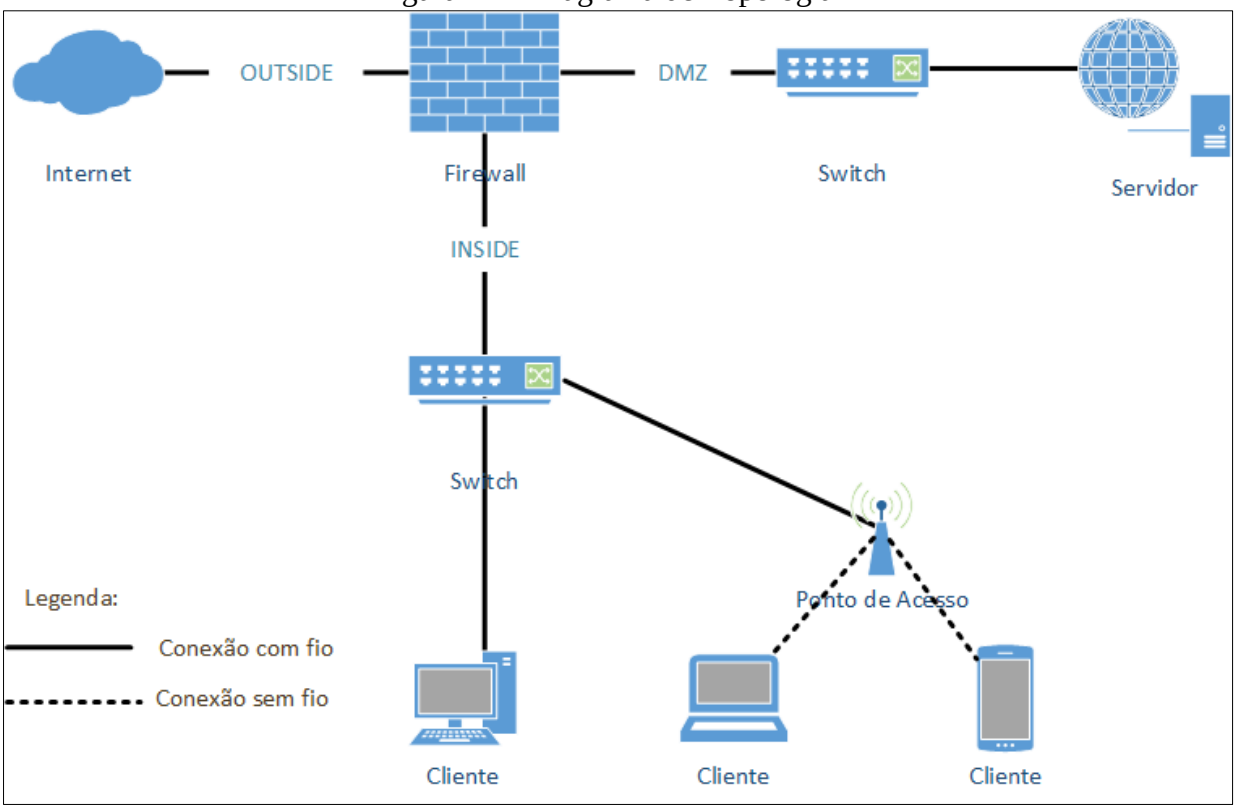
3.2 ESPECIFICAÇÃO

Nesta seção é apresentada a especificação do trabalho através do diagrama de topologia, diagrama de sequência e do diagrama de casos de uso. Para desenvolver os diagramas foi utilizado o programa Microsoft Visio Professional 2016.

3.2.1 Diagrama de topologia

O diagrama de topologia exemplificado na figura 11 demonstra a estrutura de rede onde o trabalho foi implantado e o tipo de conexão dos dispositivos.

Figura 11 – Diagrama de Topologia



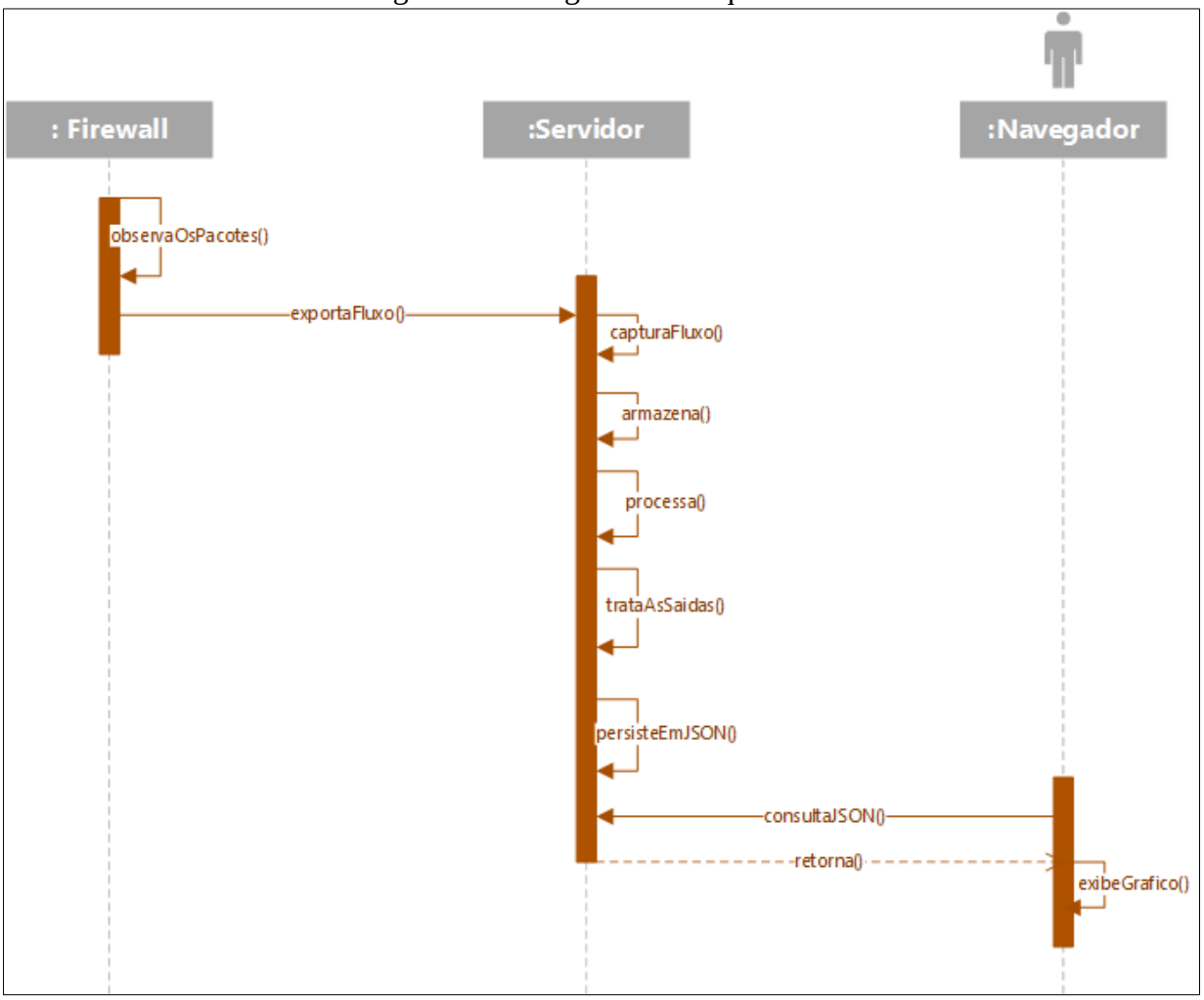
Fonte: elaborado pelo autor.

A rede está dividida em três zonas. A primeira zona está à esquerda do firewall e é denominada *outside*. A segunda zona está à direita do firewall e é chamada de DMZ. Na DMZ estão hospedados os serviços disponíveis na internet. O servidor que captura os fluxos de dados está conectado a esta zona. A terceira e última zona está na região abaixo do firewall e é chamada de *inside*. Nesta região encontra-se toda estrutura de rede que conecta os computadores dos clientes. Este é somente um exemplo simplificado, uma rede real possui dezenas de milhares de dispositivos.

3.2.2 Diagrama de sequência

O diagrama de sequência é demonstrado da figura 12. Nele pode-se ver os principais processos e atores da ferramenta.

Figura 12 – Diagrama de Sequência



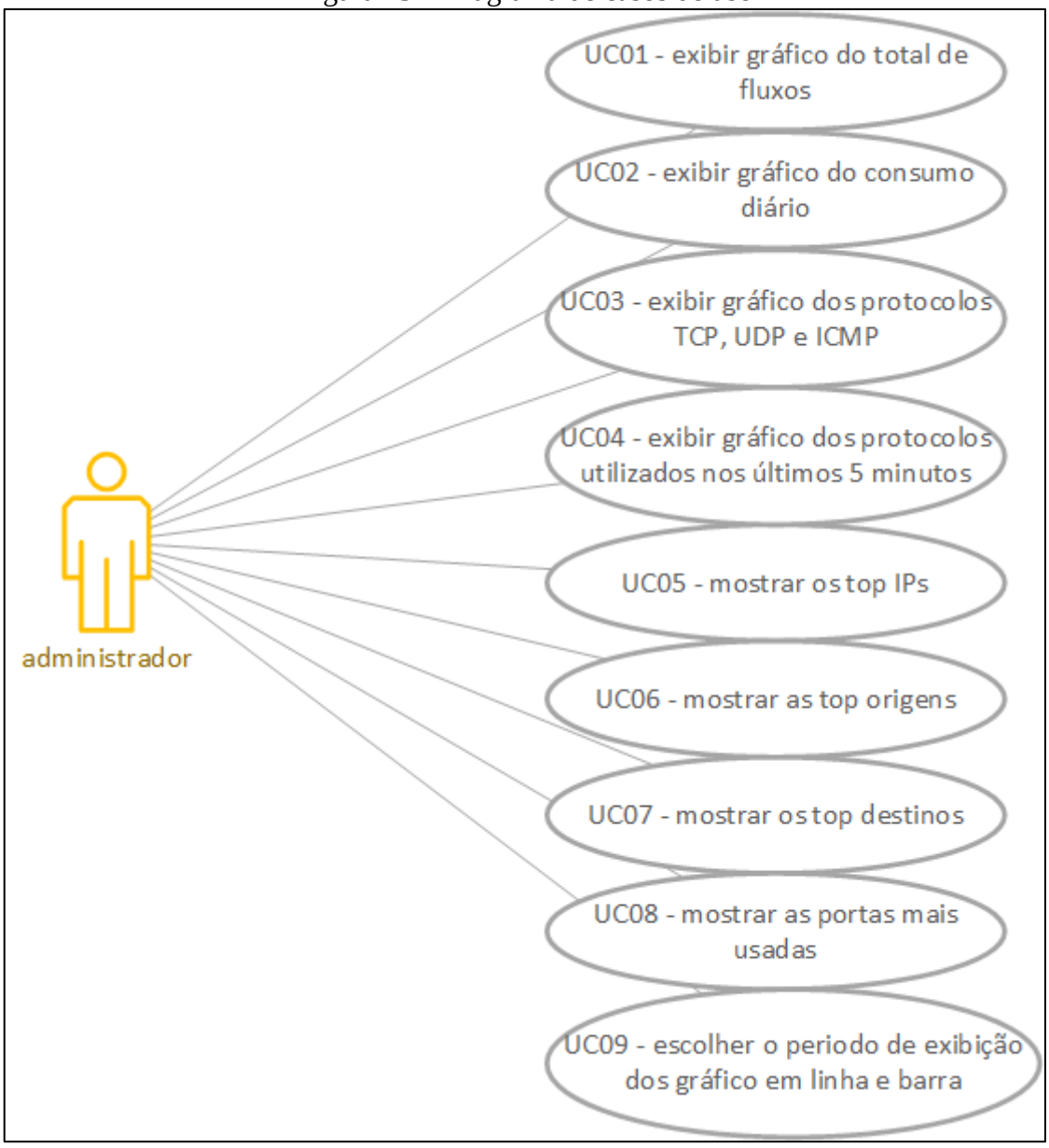
Fonte: elaborado pelo autor.

A ferramenta possui três atores. O primeiro ator é o firewall e nele ocorre os processos de observação dos pacotes e exportação do fluxo de dados. O próximo ator, responsável por grande parte dos processos captura o fluxo de dados, armazena, processa, ajusta os dados processados e persiste as saídas em JSON. Por fim, o navegador busca os arquivos JSON e exibe os dados em forma de gráficos para o usuário.

3.2.3 Diagrama de Casos de Uso

O diagrama de casos de uso apresenta as ações realizadas pelo administrador na página web. Todos os 9 requisitos funcionais são implementados, respectivamente nos 9 casos de uso da figura 13.

Figura 13 – Diagrama de casos de uso



Fonte: elaborado pelo autor.

O administrador da rede é o ator responsável por acessar a ferramenta web e acompanhar o comportamento da rede. Ao acessar a ferramenta web, o administrador pode visualizar os gráficos: histórico do total de fluxos trafegados, histórico do consumo de dados (valor acumulado e valor atual), histórico do uso dos protocolos TCP,UDP e ICMP, os protocolos mais utilizados nos últimos 5 minutos, os endereços IPs que mais trafegaram dados, as origens que mais trafegaram dados, os destinos que mais trafegaram dados, as portas mais utilizadas e escolher o período de exibição dos gráficos em linha e barra.

3.3 IMPLEMENTAÇÃO

Nesta seção será detalhada toda implementação da ferramenta. Na Seção 3.3.1 são apresentadas as técnicas e ferramentas utilizadas para que fosse possível atender os requisitos funcionais e não funcionais, bem como os trechos de códigos relevantes quando necessário. Por fim na seção 3.3.2 é apresentado a operacionalidade da implementação acompanhado da interface do sistema.

3.3.1 Técnicas e ferramentas utilizadas

Esta seção foi dividida em 4 subseções para melhor representar o funcionamento da solução e proporcionar uma melhor compreensão da ferramenta como um todo. Na seção 3.3.1.1 é demonstrado o processo de exportação do fluxo de dados. A seção 3.3.1.2 apresenta como ocorre a coleta do fluxo de dados. A seção 3.3.1.3 explica como é feita a análise, tratamento e persistência dos dados. Por fim a seção 3.3.1.4 demonstra o desenvolvimento da página Web.

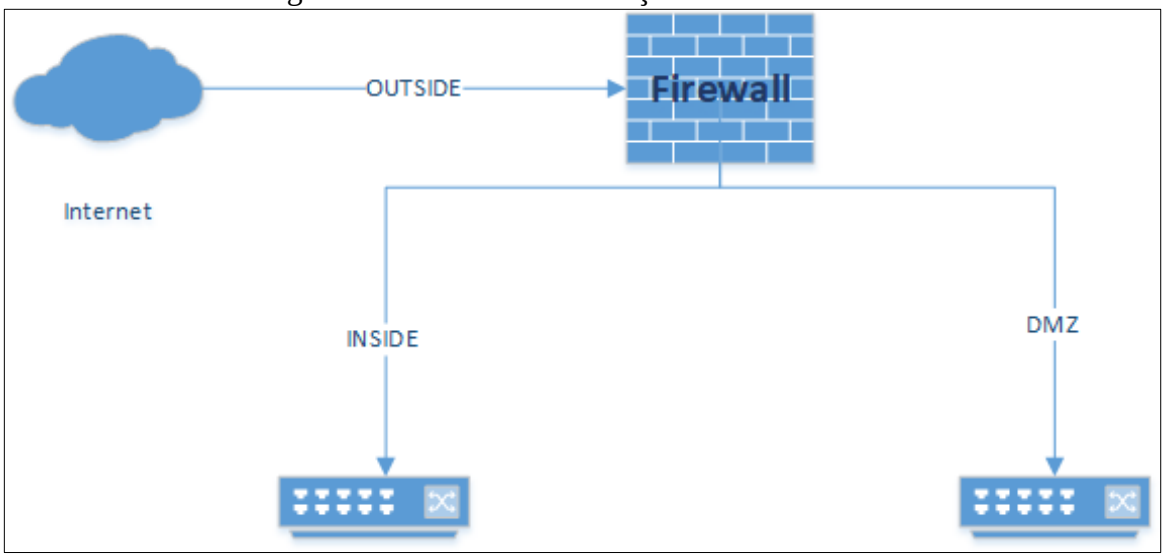
3.3.1.1 Exportação

A observação e a exportação podem ocorrer em dispositivos distintos, porém neste trabalho elas ocorrem em um único dispositivo. A motivação de exportar os fluxos de dados de um ponto central da rede se dá, pois:

> A chave para monitorar o tráfego é identificar pontos de observação significativos, permitindo que os dispositivos de captura coletem a maior parte das informações sobre o tráfego que passa pelo ponto de observação. Esses pontos de observação devem preferencialmente estar em redes com fio. (HOFSTEDE et al., 2014, p. 6, nossa tradução).

Por isso, o dispositivo escolhido para observar e exportar o fluxo de dados foi um Firewall, pois conforme mostra a figura 14, na rede onde os testes são realizados ele é o primeiro dispositivo depois do link de internet e todos os pacotes passam necessariamente por ele.

Figura 14 – Ponto de observação na rede: Firewall



Fonte: elaborado pelo autor.

Desta forma a análise é feita na rede como um todo, não somente as conexões com a internet, mas também as conexões internas da rede.

Após a escolha do dispositivo, é necessário configurar o Firewall e definir quais *interfaces* devem ser monitoradas, para qual endereço IP e porta o fluxo deve ser exportado e outras especificações conforme mostra a figura 15.

Figura 15 – Configuração Firewall

```
flow-export destination server 10.60.0.100 9995
flow-export template timeout-rate 1
flow-export delay flow-create 20
flow-export active refresh-interval 60

policy-map global_policy
        class global-class
                flow-export event-type all destination 10.60.0.100
```

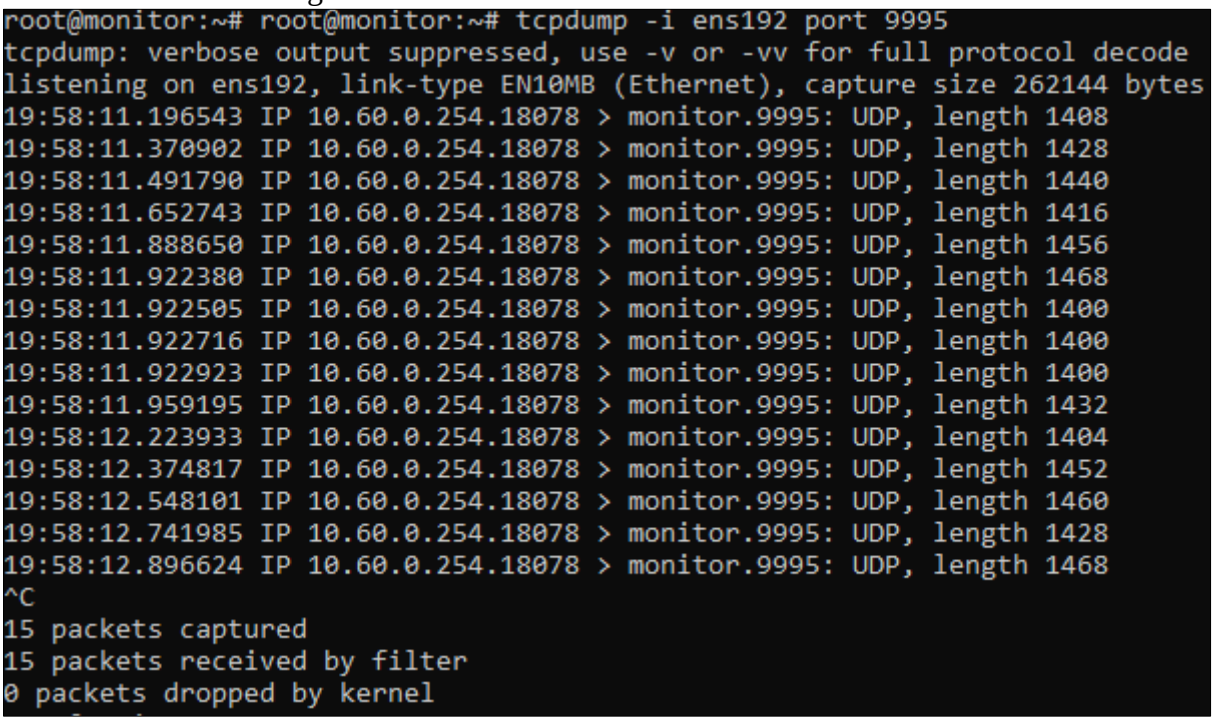
Fonte: elaborado pelo autor.

A sintaxe dos comandos pertence a equipamentos da marca CISCO. A primeira linha define para onde o fluxo de dados deve ser exportado. Na segunda, é definido o *template* de configuração. Na linha seguinte é configurado o *delay* de exportação em segundos. Depois é configurado o intervalo de atualização para conexões ativas e por fim nas linhas seguintes são definidos os eventos que serão exportados para o coletor.

3.3.1.2 Coleta

Assim que o processo de exportação estiver ativo, é necessário verificar se os fluxos de dados estão chegando no coletor, e para isso pode ser utilizado o programa Tcpdump. A figura 16 demonstra um exemplo de uso do Tcpdump.

Figura 16 – Verificando o envio do fluxo de dados



Fonte: elaborado pelo autor.

Conforme pode-se ver na figura 16, o parâmetro "-i" é utilizado para definir o nome da interface que será monitorada e em seguida é definido a porta que deve ser analisada. Abaixo do comando estão os pacotes que foram capturados, o que confirma que o exportador está encaminhando os dados para o coletor.

Uma vez que os dados estão chegando no coletor, utiliza-se um serviço de captura de fluxo de dados chamado nfcapd. O serviço nfcapd faz parte do conjunto de ferramentas nfdump, sua função é capturar o fluxo de dados e armazenar os dados em arquivos. Para iniciar o nfcapd deve-se definir alguns parâmetros, como por exemplo:

a) parâmetro -w responsável por definir o intervalo de rotação do arquivo que por padrão é 5 minutos;
b) parâmetro `-D` define que o serviço deve ser executado em *background*;
c) parâmetro `-p` define em que porta o serviço será executado. A porta deve ser a mesma configurada no exportador;
d) parâmetro `-u` define o *userid* dos arquivos;
e) parâmetro `-g` define o *groupid* dos arquivos;
f) parâmetro `-S` define a estrutura de pastas em que os dados serão armazenados;
g) parâmetro `-I` define o nome do fluxo de dados que está sendo coletado;
h) parâmetro `-l` define o diretório base em que os dados serão armazenados.

Depois de ter definido todos os parâmetros, o comando para iniciar o serviço ficou assim: `nfcapd -w -D -p 9995 -u netflow -g www-data -S 1 -I ASA5515 -l /data/nfcap/ASA5515/`. A figura 17 demonstra a execução do comando nfcapd.

Figura 17 – Execução do comando nfcapd

```
root@monitor:~# nfcapd -w -p 9995 -u netflow -g www-data -S 1 -I ASA5515 -l /data/nfcap/ASA5515
Add extension: 2 byte input/output interface index
Add extension: 4 byte input/output interface index
Add extension: 2 byte src/dst AS number
Add extension: 4 byte src/dst AS number
Bound to IPv4 host/IP: any, Port: 9995
Startup.
Init IPFIX: Max number of IPFIX tags: 65
Process_v9: New exporter: SysID: 1, Domain: 0, IP: 10.60.0.254
```

Fonte: elaborado pelo autor.

Após iniciar o serviço, o diretório definido no parâmetro `-l` deve ser verificado, pois há cada 5 minutos um novo arquivo é salvo contendo os dados dos últimos 5 minutos. O arquivo salvo é nomeado, como por exemplo: nfcapd.201107110845, onde os caracteres numéricos significam respectivamente o ano, mês, dia, hora e minuto.

3.3.1.3 Análise, tratamento e persistência dos dados

A análise do fluxo de dados é feita utilizando a ferramenta nfdump. O nfdump lê os dados de um ou mais arquivos armazenados pelo nfcapd e permite extrair diversas informações do uso da rede. O nfdump, assim como o nfcapd e o tcpdump, é uma ferramenta acessível somente via console. A sintaxe do comando é `nfdump [options] [filter]`, onde `options` quer dizer os parâmetros que podem ser definidos e `filter` os filtros. Em `options` podem ser utilizados:

a) `-r` define o arquivo de entrada que deve ser analisado;
b) `-R` é utilizado quando a entrada é uma sequência de arquivos em um diretório;
c) `-s` gera estatísticas de uso da rede. Aqui é definido quais dados devem ser analisados e a ordem que serão exibidos;
d) `-o` define o formato de saída do arquivo.

Na figura 18 pode-se ver um exemplo do uso da ferramenta.

Figura 18 – Comando nfdump

```
nfdump -r /data/nfcap/ASA5515/2018/06/15/nfcapd.201806151115 -s port/flows
Top 10 Port ordered by flows:
Date first seen          Duration     Proto     Port    Flows(%)     Packets(%)     Bytes(%)    pps  bps   bpp
1969-12-31 21:00:00.000 1529072379.932 any       53    44235(45.1)     0( 0.0)    2.3 M( 0.6)   0    0     0
1969-12-31 21:00:00.000 1529072398.191 any      443    28026(28.6)     0( 0.0)  301.9 M(87.0)   0    1     0
1969-12-31 21:00:00.000 1529072398.481 any       80     5782( 5.9)     0( 0.0)    7.5 M( 2.2)   0    0     0
1969-12-31 21:00:00.000 1529072374.102 any     7547     2163( 2.2)     0( 0.0)        0( 0.0)   0    0     0
1969-12-31 21:00:00.000 1529072279.164 any       23     2071( 2.1)     0( 0.0)     1426( 0.0)   0    0     0
2018-06-15 11:15:00.478   295.483       any        0     1844( 1.9)     0( 0.0)    1.2 M( 0.4)   0    33144 0
1969-12-31 21:00:00.000 1529072362.142 any     8291     1084( 1.1)     0( 0.0)        0( 0.0)   0    0     0
1969-12-31 21:00:00.000 1529072358.022 any      137      974( 1.0)     0( 0.0)    24612( 0.0)   0    0     0
1969-12-31 21:00:00.000 1529072370.902 any       22      965( 1.0)     0( 0.0)     2334( 0.0)   0    0     0
1969-12-31 21:00:00.000     0.000      any    16403      735( 0.7)     0( 0.0)        0( 0.0)   0    0     0

Summary: total flows: 98145, total bytes: 347155725, total packets: 0, avg bps: 1, avg pps: 0, avg bpp: 0
Time window: Time Window unknown
Total flows processed: 98145, Blocks skipped: 0, Bytes read: 5889304
Sys: 0.024s flows/second: 4089375.0  Wall: 0.024s flows/second: 3957300.1
```

Fonte: elaborado pelo autor.

Conforme pode-se ver, o comando nfdump recebeu como parâmetro um arquivo de entrada chamado “nfcapd.201806151115”. Após o arquivo de entrada, foi definida qual estatística será exibida. A sintaxe para definir a estatística é a seguinte: primeiro o parâmetro `-s` e depois qual informação quer-se analisar, como por exemplo porta, endereço de destino, endereço de origem, qualquer endereço IP, porta de destino, porta de origem, dentre outros. Em seguida, após a barra define-se qual dado será considerado para ordenar o resultado, podendo ser o número de fluxos, número de bytes, dentre outros. No exemplo da figura 18 foram exibidos as top 10 portas com maior número de fluxos.

A saída do comando, conforme pode-se ver na figura 18, é um texto com todos os dados organizados como se fosse uma tabela. No entanto, os dados precisam ser persistidos de forma legível e de fácil manipulação para posterior análise. Por isso é utilizado o parâmetro `-o`, que permite selecionar diversos formatos de saída, como por exemplo CSV e JSON. O formato JSON, embora esteja descrito no manual da ferramenta, não funciona. Por isso, seleciona-se o formato CSV. A figura 19 mostra a saída do comando nfdump no formato CSV.

Figura 19 – Saída do comando nfdump no formato CSV

```
nfdump -r /data/nfcap/ASA5515/2018/06/15/nfcapd.201806151115 -s port/flows -o csv
ts,te,td,pr,val,fl,flP,ipkt,ipktP,ibyt,ibytP,ipps,ibps,ibpp
1969-12-31 21:00:00,2018-06-15 11:19:39,1529072379.932,any,53,44235,45.1,0,0.0,2256126,0.6,0,0,0
1969-12-31 21:00:00,2018-06-15 11:19:58,1529072398.191,any,443,28026,28.6,0,0.0,301852778,87.0,0,1,0
1969-12-31 21:00:00,2018-06-15 11:19:58,1529072398.481,any,80,5782,5.9,0,0.0,7478393,2.2,0,0,0
1969-12-31 21:00:00,2018-06-15 11:19:34,1529072374.102,any,7547,2163,2.2,0,0.0,0,0.0,0,0,0
1969-12-31 21:00:00,2018-06-15 11:17:59,1529072279.164,any,23,2071,2.1,0,0.0,1426,0.0,0,0,0
2018-06-15 11:15:00,2018-06-15 11:19:55,295.483,any,0,1844,1.9,0,0.0,1224200,0.4,0,33144,0
1969-12-31 21:00:00,2018-06-15 11:19:22,1529072362.142,any,8291,1084,1.1,0,0.0,0,0.0,0,0,0
1969-12-31 21:00:00,2018-06-15 11:19:18,1529072358.022,any,137,974,1.0,0,0.0,24612,0.0,0,0,0
1969-12-31 21:00:00,2018-06-15 11:19:30,1529072370.902,any,22,965,1.0,0,0.0,2334,0.0,0,0,0
1969-12-31 21:00:00,1969-12-31 21:00:00,0.000,any,16403,735,0.7,0,0.0,0,0.0,0,0,0

Summary
flows,bytes,packets,avg_bps,avg_pps,avg_bpp
98145,347155725,0,1,0,0
```

Fonte: elaborado pelo autor.

Como foi definido nos requisitos, que os dados deveriam ser persistidos em arquivos no formato JSON, utiliza-se a ferramenta CSVtoJSON, que converte a saída do comando nfdump em CSV para o formato JSON. A figura 20 demonstra a saída no formato JSON.

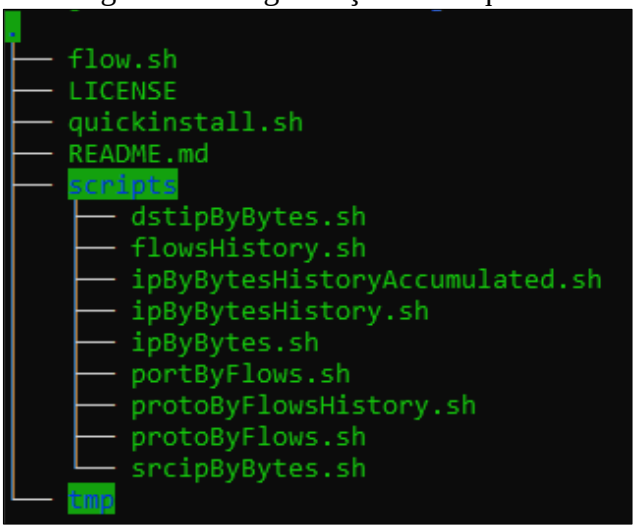
Figura 20 – Conversão do formato CSV para JSON

```
nfdump -r /data/nfcap/ASA5515/2018/06/15/nfcapd.201806151115 -s port/flows -o csv | csv2json
[
{"ts":"1969-12-31 21:00:00","te":"2018-06-15 11:19:39","td":"1529072379.932","pr":"any","val":"53","fl":"44235",
"flP":"45.1","ipkt":"0","ipktP":"0.0","ibyt":"2256126","ibytP":"0.6","ipps":"0","ibps":"0","ibpp":"0"},
{"ts":"1969-12-31 21:00:00","te":"2018-06-15 11:19:58","td":"1529072398.191","pr":"any","val":"443","fl":"28026",
"flP":"28.6","ipkt":"0","ipktP":"0.0","ibyt":"301852778","ibytP":"87.0","ipps":"0","ibps":"1","ibpp":"0"},
{"ts":"1969-12-31 21:00:00","te":"2018-06-15 11:19:58","td":"1529072398.481","pr":"any","val":"80","fl":"5782",
"flP":"5.9","ipkt":"0","ipktP":"0.0","ibyt":"7478393","ibytP":"2.2","ipps":"0","ibps":"0","ibpp":"0"},
{"ts":"1969-12-31 21:00:00","te":"2018-06-15 11:19:34","td":"1529072374.102","pr":"any","val":"7547","fl":"2163",
"flP":"2.2","ipkt":"0","ipktP":"0.0","ibyt":"0","ibytP":"0.0","ipps":"0","ibps":"0","ibpp":"0"},
{"ts":"1969-12-31 21:00:00","te":"2018-06-15 11:17:59","td":"1529072279.164","pr":"any","val":"23","fl":"2071",
"flP":"2.1","ipkt":"0","ipktP":"0.0","ibyt":"1426","ibytP":"0.0","ipps":"0","ibps":"0","ibpp":"0"},
{"ts":"2018-06-15 11:15:00","te":"2018-06-15 11:19:55","td":"295.483","pr":"any","val":"0","fl":"1844","flP":"1.9",
"ipkt":"0","ipktP":"0.0","ibyt":"1224200","ibytP":"0.4","ipps":"0","ibps":"33144","ibpp":"0"},
{"ts":"1969-12-31 21:00:00","te":"2018-06-15 11:19:22","td":"1529072362.142","pr":"any","val":"8291","fl":"1084",
"flP":"1.1","ipkt":"0","ipktP":"0.0","ibyt":"0","ibytP":"0.0","ipps":"0","ibps":"0","ibpp":"0"},
{"ts":"1969-12-31 21:00:00","te":"2018-06-15 11:19:18","td":"1529072358.022","pr":"any","val":"137","fl":"974",
"flP":"1.0","ipkt":"0","ipktP":"0.0","ibyt":"24612","ibytP":"0.0","ipps":"0","ibps":"0","ibpp":"0"},
{"ts":"1969-12-31 21:00:00","te":"2018-06-15 11:19:30","td":"1529072370.902","pr":"any","val":"22","fl":"965",
"flP":"1.0","ipkt":"0","ipktP":"0.0","ibyt":"2334","ibytP":"0.0","ipps":"0","ibps":"0","ibpp":"0"},
{"ts":"1969-12-31 21:00:00","te":"1969-12-31 21:00:00","td":"0.000","pr":"any","val":"16403","fl":"735","flP":"0.7",
"ipkt":"0","ipktP":"0.0","ibyt":"0","ibytP":"0.0","ipps":"0","ibps":"0","ibpp":"0"},
{},
{"ts":"Summary"},
{"ts":"flows","te":"bytes","td":"packets","pr":"avg_bps","val":"avg_pps","fl":"avg_bpp"},
{"ts":"98145","te":"347155725","td":"0","pr":"1","val":"0","fl":"0"}
]
```

Fonte: elaborado pelo autor.

Os processos de análise e conversão dos dados foram automatizados com uma série de Shell Scripts. Shell Script é uma linguagem de script. “As linguagens de script pressupõem que uma coleção de componentes úteis já existe em outras linguagens. Elas não são destinadas a desenvolver aplicações do zero, mas sim para combinar componentes.” (OUSTERHOUT, 1998, p. 24, nossa tradução). Na figura 21 pode-se ver como os scripts estão organizados.

Figura 21 – Organização dos arquivos



Fonte: elaborado pelo autor.

No diretório raiz tem-se dois scripts, o `flow.sh` e o `quickinstall.sh`. O `quickinstall.sh` instala o nfdump e todas as dependências necessárias na distribuição Linux Debian de forma automatizada. No quadro 4 é exibido o trecho principal do código.

Quadro 4 – Script instalador

```
install() {
        logging "Installing " + "${1}"
        apt-get -y install "${1}"
        local ret=$?

        if [ ${ret} -ne 0 ];
        then
            run_failed
        else
            run_ok
        fi
}

curl -sL https://deb.nodesource.com/setup_8.x | bash -

install gcc
install build-essential
install make
install tcpdump
install net-tools
install linux-headers-$(uname -r)
install git
install libtool
install pkg-config
install libbz2-dev
install autoconf
install bison
install flex
install librrd-dev
install nodejs

npm install --global csv2json

git clone https://github.com/phaag/nfdump.git

perl -MCPAN -e 'install Socket6'

cd nfdump/
chmod +x autogen.sh
mkdir m4

ACLOCAL_FLAGS="-I /usr/share/aclocal-1.15" ./autogen.sh

./configure --enable-nfprofile --enable-nftrack

make && make install
```

Fonte: elaborado pelo autor.

O método `install` recebe como parâmetro o nome do pacote, instala-o e exibe uma mensagem de sucesso ou falha caso ocorra algum erro. Em seguida os pacotes são instalados, inclusive a ferramenta CSVtoJSON. Depois realiza-se a instalação e configuração do nfdump.

O `flow.sh` é responsável por verificar a cada 5 segundos se existe um novo arquivo gerado pelo nfcapd, e caso exista, ele executa todos os scripts de análise passando como parâmetro o arquivo nfcapd atual. No quadro 5 é exposto o código fonte do script.

Quadro 5 – Script gerenciador

```
#!/bin/sh

#ls -lt         - List by newest first
#sed -n '2p'    - Show only line number 2
#cut -d ' ' -f9 - Get only the name of the archive

run_refresh() {
        year=$(date +%Y)
        month=$(date +%m)
        day=$(date +%d)
        current_nfcap=$(ls -lt /data/nfcap/ASA5515/$year/$month/$day/ | sed -n '2p' | tail -c 20)
}

run_refresh
old_nfcap=""
log='/var/www/html/log/flow.log'

while true
do

        if [ "$current_nfcap" != "$old_nfcap" ] && [ -n "$current_nfcap" ];
        then
                echo $(date +"%b %d %H:%M:%S") mainProcess'['$$']:' Start main script... >> $log
                echo $(date +"%b %d %H:%M:%S") mainProcess'['$$']:' This file is being analized: $current_nfcap >> $log

                for file in scripts/*; do
                        [ -f "$file" ] && [ -x "$file" ] && "$file" /data/nfcap/ASA5515/$year/$month/$day/$current_nfcap
                done

                echo $(date +"%b %d %H:%M:%S") mainProcess'['$$']:' End main script... >> $log

        fi

        sleep 5

        old_nfcap=$current_nfcap
        run_refresh
done
```

Fonte: elaborado pelo autor.

No diretório “scripts” estão todos os scripts responsáveis pela análise, tratamento e persistência dos dados. O diretório “tmp” serve de auxílio para armazenar os arquivos temporários, que são gerados pelos scripts. No quadro 6 é exibido o script `ipByBytes.sh`, cujo objetivo é exibir os IPs com maior número de bytes.

Quadro 6 – Script ipByBytes

```
#!/bin/sh

scriptName='ipByBytes'
data_path='/var/www/html/data/'
tmp_path='/home/netflow/git/flow-script/tmp/'
log='/var/www/html/log/flow.log'

output="${tmp_path}${scriptName}.csv"
data="${data_path}${scriptName}.json"

echo $(date +"%b %d %H:%M:%S") $scriptName'['$$']:' Start >> $log

nfdump -r $1 -s ip/bytes -o csv > $output

top_10=$(sed -n '1,11p' $output | csv2json)
summary=$(sed -n '14,15p' $output | csv2json | sed -n '2p')

echo '{ "top_10": '  $top_10  ', "summary":'  $summary  ' }' > $data

echo $(date +"%b %d %H:%M:%S") $scriptName'['$$']:' End >> $log
```

Fonte: elaborado pelo autor.

Nas primeiras linhas são definidos alguns parâmetros como por exemplo, o nome do script, o local onde os arquivos JSON devem ser salvos, o local onde os arquivos temporários

devem ser salvos, dentre outros. Em seguida, na linha 13, o nfdump analisa os dados que foram passados por parâmetro pelo script principal e envia as estatísticas no formato CSV para um arquivo temporário no diretório “tmp”. Após a execução do comando nfdump, é feito o tratamento dos dados com auxílio do comando sed. O sed é um editor de *streams*, capaz de filtrar e transformar textos. Na linha 15 ele é utilizado para selecionar os dados da linha 1 até a linha 11. A saída do comando é redirecionada para a ferramenta CSVtoJSON e o resultado é armazenado em uma variável. Esse processo ocorre novamente na linha seguinte, porém seleciona linhas diferentes. Esse procedimento é necessário, pois a ferramenta CSVtoJSON não considera os dois índices do arquivo CSV, resultando assim em uma conversão errada. Após selecionar os dados, de forma que eles sejam convertidos para JSON corretamente, na linha 18, eles são armazenados em um arquivo com a extensão ponto JSON. Conforme pode-se ver no código demonstrado no quadro 6, o script possui a seguinte estrutura:

a) definição dos parâmetros;
b) registro de início de execução;
c) análise dos dados;
d) tratamento dos dados e conversão para JSON;
e) persistência dos dados;
f) registro de termino da execução.

Os outros 8 scripts seguem a mesma lógica, a mudança principal ocorre nos passos c, d e e (análise, tratamento e persistência). No quadro 7 é exibido o script “protoByFlowsHistory.sh”.

Quadro 7 – Script protoByFlowsHistory

```
13 nfdump -r $1 -s proto -o csv > $output
14
15 tcpNum=$(grep -n 'TCP' $output | cut -c1)
16 udpNum=$(grep -n 'UDP' $output | cut -c1)
17 icmpNum=$(grep -n 'ICMP,' $output| cut -c1)
18
19 sed -i "1s/^/hora,/" $output
20 sed -i "2s/^/${hora},/" $output
21 sed -i "3s/^/${hora},/" $output
22 sed -i "4s/^/${hora},/" $output
23 sed -i "5s/^/${hora},/" $output
24
25 tcp=$(sed -n "1,${tcpNum}p" $output |  csv2json | sed -n "${tcpNum}p")
26 udp=$(sed -n "1,${udpNum}p" $output |  csv2json | sed -n "${udpNum}p")
27 icmp=$(sed -n "1,${icmpNum}p" $output | csv2json | sed -n "${icmpNum}p")
28
29 sed -i "3s/$/,${tcp}/" $data
30 sed -i "6s/$/,${udp}/" $data
31 sed -i "9s/$/,${icmp}/" $data
```

Fonte: elaborado pelo autor.

Neste script é utilizado a estatística `-s proto` que exibe os protocolos com maior número de fluxos. Em seguida é armazenado, com auxílio da ferramenta grep e cut, o número da linha em que os protocolos TCP, UDP e ICMP se encontram. O grep busca no arquivo, o padrão que foi passado por parâmetro e retorna as linhas que correspondem ao padrão dado. O cut remove as seções indesejadas do arquivo. Após a seleção dos protocolos, é adicionada a hora no arquivo CSV, para que seja possível identificar o período em que os dados apresentados no gráfico pertencem. Nos processos, da linha 25 até a linha 27, é feito a conversão dos dados para o formato JSON e é selecionado o objeto JSON que corresponde aos dados dos protocolos TCP, UDP e ICMP.

Os outros scripts utilizam estatísticas diferentes das usadas até agora, as estatísticas usadas são:

a) `-s srcip/bytes` exibe estatística dos top 10 endereços de origem com mais bytes trafegados;
b) `-s dstip/bytes` exibe estatística dos top 10 endereços de destino com mais bytes trafegados;
c) `-s port/flows` exibe estatística das top 10 portas com maior número de fluxos;
d) `-s record` exibe estatística do total de fluxos.

Após a análise, realiza-se o tratamento e conversão dos dados utilizando as ferramentas mencionadas até agora.

3.3.1.4 Página Web

A página Web foi desenvolvida utilizando as tecnologias HTML 5, CSS, Bootstrap, JQuery, Java Script e Chart.js. O HTML 5, CSS e Bootstrap são responsáveis por customizar e estruturar a página, além disso, tornam o site responsivo, podendo ser acessado por diversos dispositivos. O JQuery é utilizado para realizar interações na página, como por exemplo, ocultar e mostrar a barra de menu e definir se o gráfico diário ou semanal está ativo. O quadro 8 demonstra o código que altera a barra de menu.

Quadro 8 – Código JQuery

```
186         $(document).ready(function () {
187             $('#sidebarCollapse').on('click', function () {
188                 $('#sidebar').toggleClass('active');
189                 $(this).toggleClass('active');
190             });
191         });
```

Fonte: elaborado pelo autor.

O bloco de código exibido é acionado quando o botão em forma de xis é pressionado. Assim que pressionado, é adicionado a classe “active” no elemento. Se o elemento já possuir a classe “active”, ela é retirada. Com isso, a barra de menu lateral é exibida ou ocultada e aplica-se uma transformação no botão.

O Java Script é responsável por buscar e manipular os arquivos JSON e exibi-los em forma de gráfico utilizando a biblioteca Chart.js. O quadro 9 demonstra como estes processos são realizados.

Quadro 9 – Método Java Script: loadLineGraph

```
const loadLineGraph = (label, titulo, url, doc, tipo, dias = 1, ...funcao) => {
  fetch(url)
    .then(resp => resp.json())
    .then(arquivo => {
      const total = arquivo.top_10.length;
      const inicio = total - dias * 288;
      const horas = arquivo.top_10.map(funcao[0]).slice(inicio, total);
      let dados;

      if (funcao.length > 2) {
        dados = arquivo.top_10
          .map(funcao[1])
          .map(funcao[2])
          .slice(inicio, total);
      } else {
        dados = arquivo.top_10.map(funcao[1]).slice(inicio, total);
      }

      //Line chart
      new Chart(doc, {
        type: tipo,
        data: {
          labels: horas,
          datasets: [
            {
              data: dados,
              label: label,
              borderColor: "#3e95cd",
              backgroundColor: "#ebf4fa",
              fill: true
            }
          ]
        },
        options: {
          title: {
            display: true,
            text: titulo
          }
        }
      });
    });
};
```

Fonte: elaborado pelo autor.

O código do quadro 9 pertence ao método loadLineGraph, que foi criado para facilitar a criação do gráfico em linha. O método funciona da seguinte forma: na linha 13, é utilizado o método `fetch` para buscar o arquivo JSON. Este método, que busca recursos locais ou da rede, é fornecido pela API Fetch. O método recebe como parâmetro a URL do arquivo JSON. A resposta é processada pelo método json(), que retorna um objeto JSON. Esse objeto é nomeado de “arquivo” na linha 15. Em seguida é feito o cálculo para selecionar a quantidade de dados que serão exibidos no gráfico. Logo depois são definidas duas variáveis, horas e dados. Cada variável recebe um *Array*, onde cada posição dos *Arrays* corresponde às horas e aos dados que serão exibidos no gráfico. Para atribuir os *Arrays* às variáveis, é utilizada a função `map`. A função map mapeia o *Array* do objeto JSON para outro *Array* de mesmo tamanho, porém somente com os dados que precisamos. A seleção é feita com as funções demonstradas no quadro 10.

Quadro 10 – Método Java Script: loadLineGraph

```
1   const apenasPr = flow => flow.pr;
2   const apenasFl = flow => flow.fl;
3   const apenasVal = flow => flow.val;
4   const apenasHora = flow => flow.hora;
5   const apenasFlows = flow => flow.flows;
6   const apenasByt = flow => flow.byt;
7   const apenasIbyt = flow => flow.ibyt;
8   const apenasGBytes = flow => flow.bytes / 1024 / 1024 / 1024;
9   const apenasMBytes = flow => flow.bytes / 1024 / 1024;
10  const format = flow => flow.toFixed(2);
```

Fonte: elaborado pelo autor.

As funções da linha 1 até a linha 10 retornam um valor específico do objeto JSON que recebem. Nas seguintes converte-se Byte para Megabyte e Gigabyte e formata-se o número de casas após a virgula. Depois de selecionar os dados, o gráfico é montado a partir da linha 31 do quadro 9, onde são definidos o elemento da página no qual o gráfico será exibido, o tipo do gráfico, os dados que serão exibidos e a exibição do título. No quadro 9, é construído um gráfico com apenas uma fonte de dados, e quando o gráfico possui mais de uma fonte de dados, ele é construído conforme o quadro 11.

Quadro 11 – Gráfico em linha com mais de uma fonte de dados

```
const horas = arquivo.tcp.map(funcao[0]).slice(inicio, total);
const tcp = arquivo.tcp.map(funcao[1]).slice(inicio, total);
const udp = arquivo.udp.map(funcao[1]).slice(inicio, total);
const icmp = arquivo.icmp.map(funcao[1]).slice(inicio, total);

//Line chart
new Chart(doc, {
  type: tipo,
  data: {
    labels: horas,
    datasets: [
      {
        data: tcp,
        label: "TCP",
        borderColor: "#3e95cd",
        fill: false
      },
      {
        data: udp,
        label: "UDP",
        borderColor: "#8e5ea2",
        fill: false
      },
      {
        data: icmp,
        label: "ICMP",
        borderColor: "#3cba9f",
        fill: false
      }
    ]
  },
  options: {
    title: {
      display: true,
      text: titulo
    }
  }
});
```

Fonte: elaborado pelo autor.

A mudança ocorre na adição dos dados no *datasets*. Para cara nova fonte deve-se definir um nome e uma cor. Além do método `loadLineGraph`, existe também o método `loadPieGraph` e `loadBarGraph`. Não há modificação significativa na construção de gráficos em barra e pizza. A principal alteração é a definição do estilo do gráfico e a apropriada fonte de dados.

3.3.2 Operacionalidade da implementação

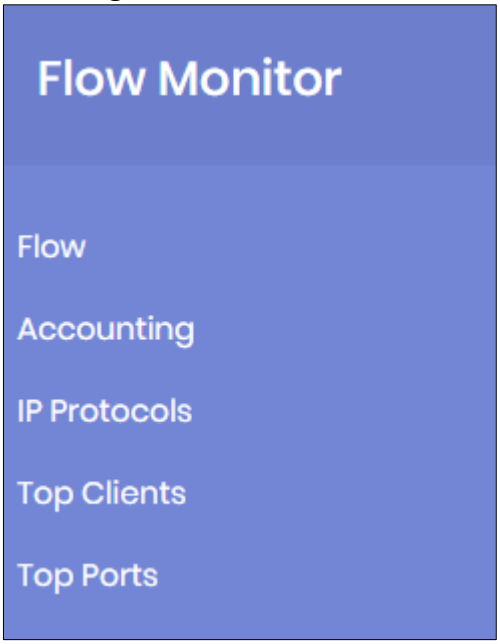
Esta seção apresenta a ferramenta sob a perspectiva do administrador. Para acessar a ferramenta o administrador deve utilizar o navegador Google Chrome e acessar a URL https://ghramos.github.io/flow-web/. Na figura 22 é exibido o primeiro acesso à página web.

Figura 22 – Primeiro acesso página web

Fonte: elaborado pelo autor.

Na lateral esquerda da página está localizado o menu que dá acesso a todos os gráficos. Na parte superior encontra-se um botão que tem formato de xis, ele é responsável por mostrar e ocultar o menu. Ainda na parte superior, porém no canto direito há um link que dá acesso à página inicial. A região central da página é onde todos os gráfico e tabelas são exibidos. A figura 23 mostra os itens do menu.

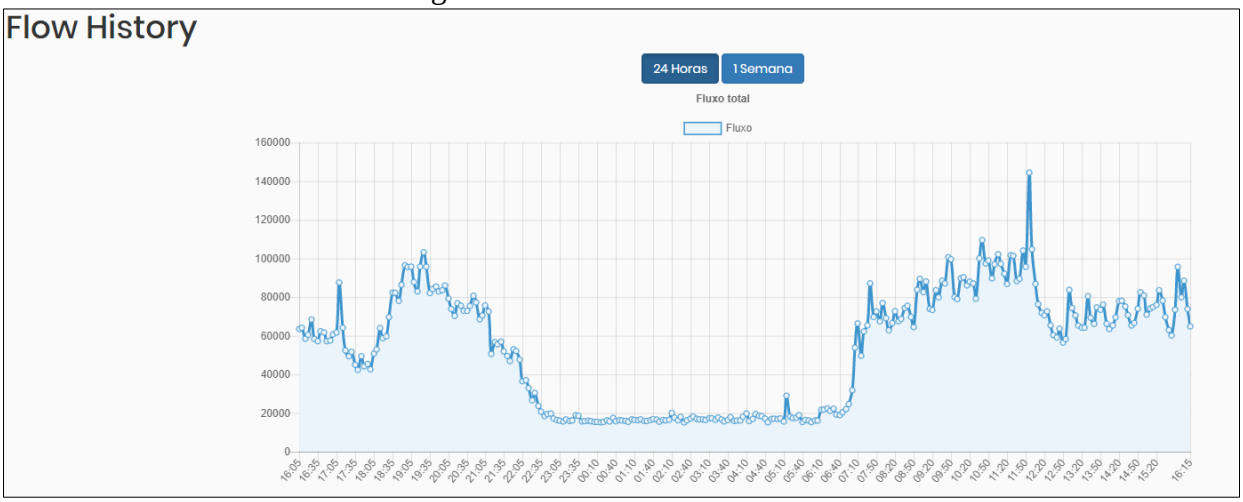
Figura 23 – Itens do menu



Fonte: elaborado pelo autor.

O primeiro item no menu se chama Flow, que exibe o total de fluxos que trafegaram nas últimas 24 horas (abscissa), podendo ainda alterar o modo de exibição para mostrar os dados da última semana. A figura 24 demonstra o gráfico em linha do total de fluxos.

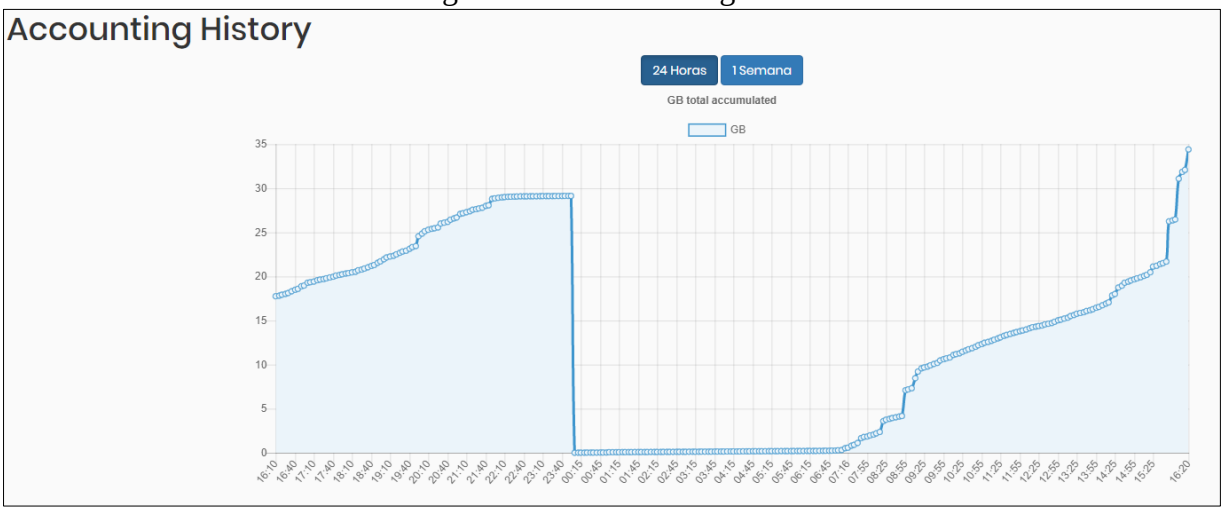
Figura 24 – Histórico de fluxos



Fonte: elaborado pelo autor.

O segundo item tem o nome de Accounting. Nesta página é exibida a quantidade de dados trafegados de forma acumulativa, ou seja, a soma dos dados trafegados no dia e a quantidade de dados trafegados nos últimos 5 minutos. A figura 25 mostra o gráfico em linha do total de dados trafegados (abscissa).

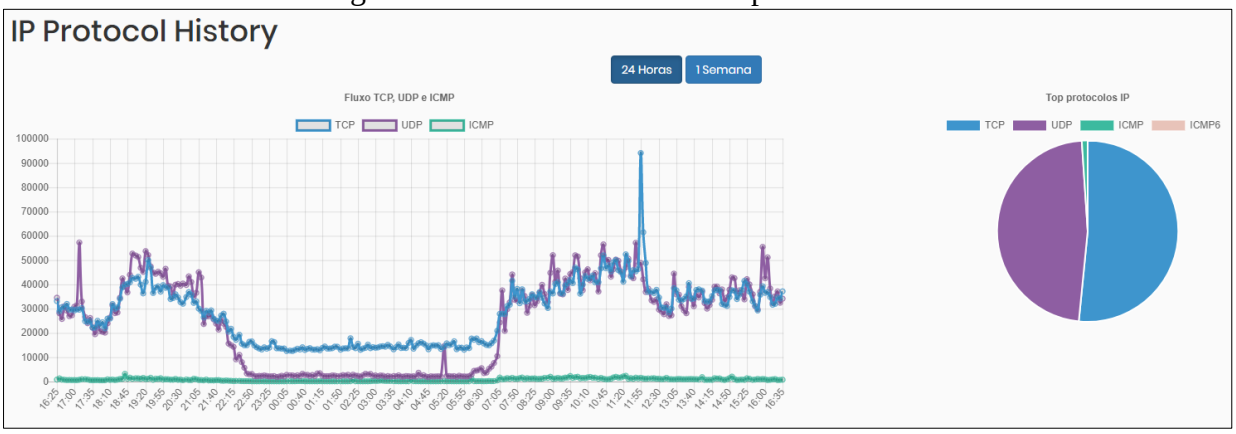
Figura 25 – Dados trafegados



Fonte: elaborado pelo autor.

O terceiro item do menu é chamado de IP Protocols. Nesta página é mostrado em um gráfico em linha, o uso dos protocolos TCP, UDP e ICMP (abscissa) nas últimas 24 horas. Além do gráfico em linha é exibido um gráfico em pizza do uso de todos os protocolos analisados pelo nfdump. O gráfico em pizza apresenta os dados dos últimos 5 minutos. Estes gráficos são demonstrados na figura 26.

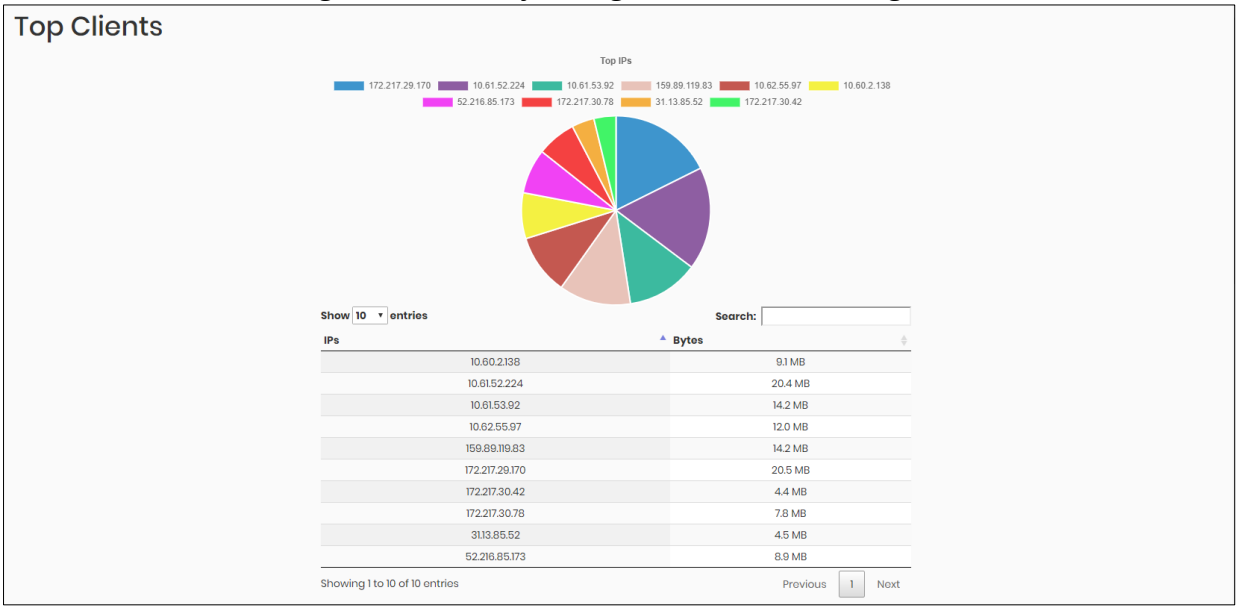
Figura 26 – Histórico do uso dos protocolos



Fonte: elaborado pelo autor.

O próximo item é o Top Clients. Nesta página são exibidos três gráficos em pizza e abaixo de cada gráfico uma tabela com os dados exibidos no gráfico. O primeiro gráfico mostra os top IPs que mais trafegaram dados. O segundo gráfico mostra os destinos que mais trafegaram dados e o terceiro as origens que mais trafegaram dados. A figura 27 demonstra como o gráfico e a tabela são exibidos.

Figura 27 – Exibição do gráfico de maior tráfego



Fonte: elaborado pelo autor.

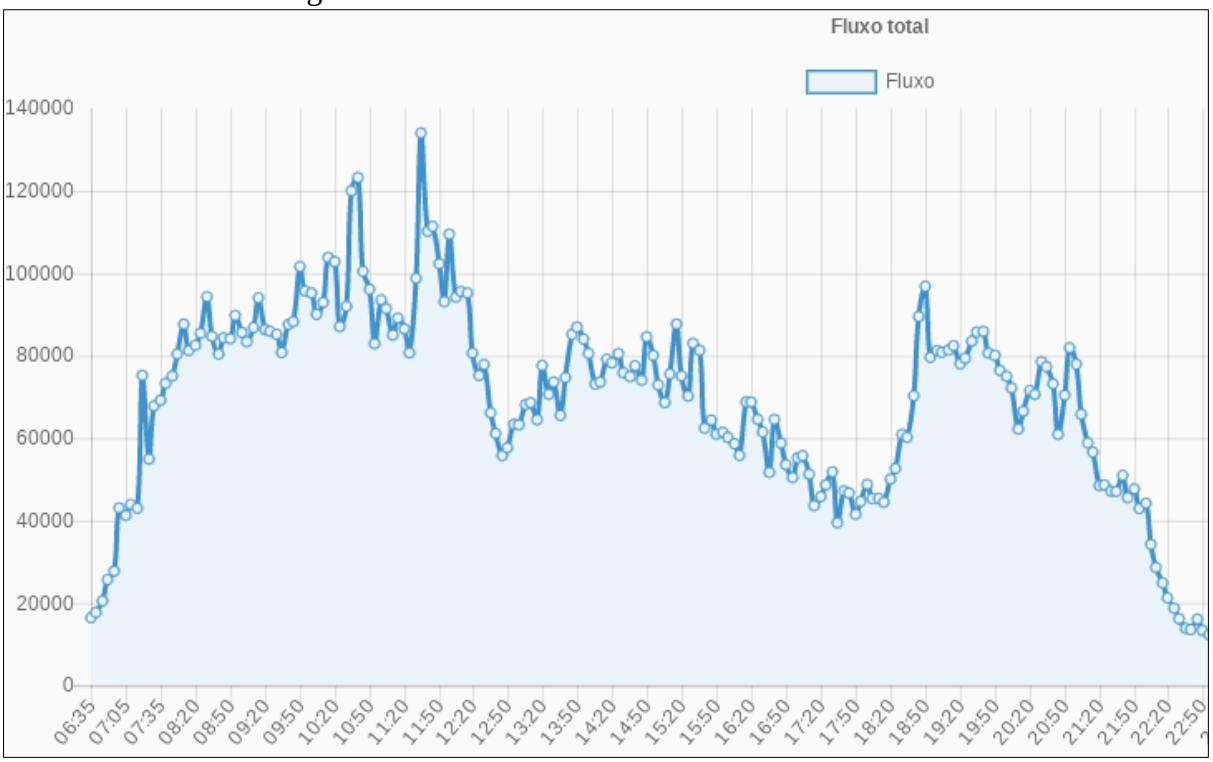
Por fim, o quinto item do menu é o Top Ports, nele são exibidas as portas mais utilizadas nos últimos 5 minutos. O gráfico e a tabela exibidos nesta página seguem o mesmo leiaute do gráfico exibido na figura 27.

3.4 ANÁLISE DOS RESULTADOS

Para analisar os resultados foi realizado o acompanhamento dos gráficos com dados de uma rede real. A rede tem aproximadamente 50 dispositivos de rede, 200 computadores e uma

média de 250 dispositivos móveis. Este acompanhamento permitiu que várias conclusões do uso da rede fossem tiradas. Veja a figura 28, que exibe o histórico do fluxo de dados.

Figura 28 – Análise do histórico do fluxo de dados

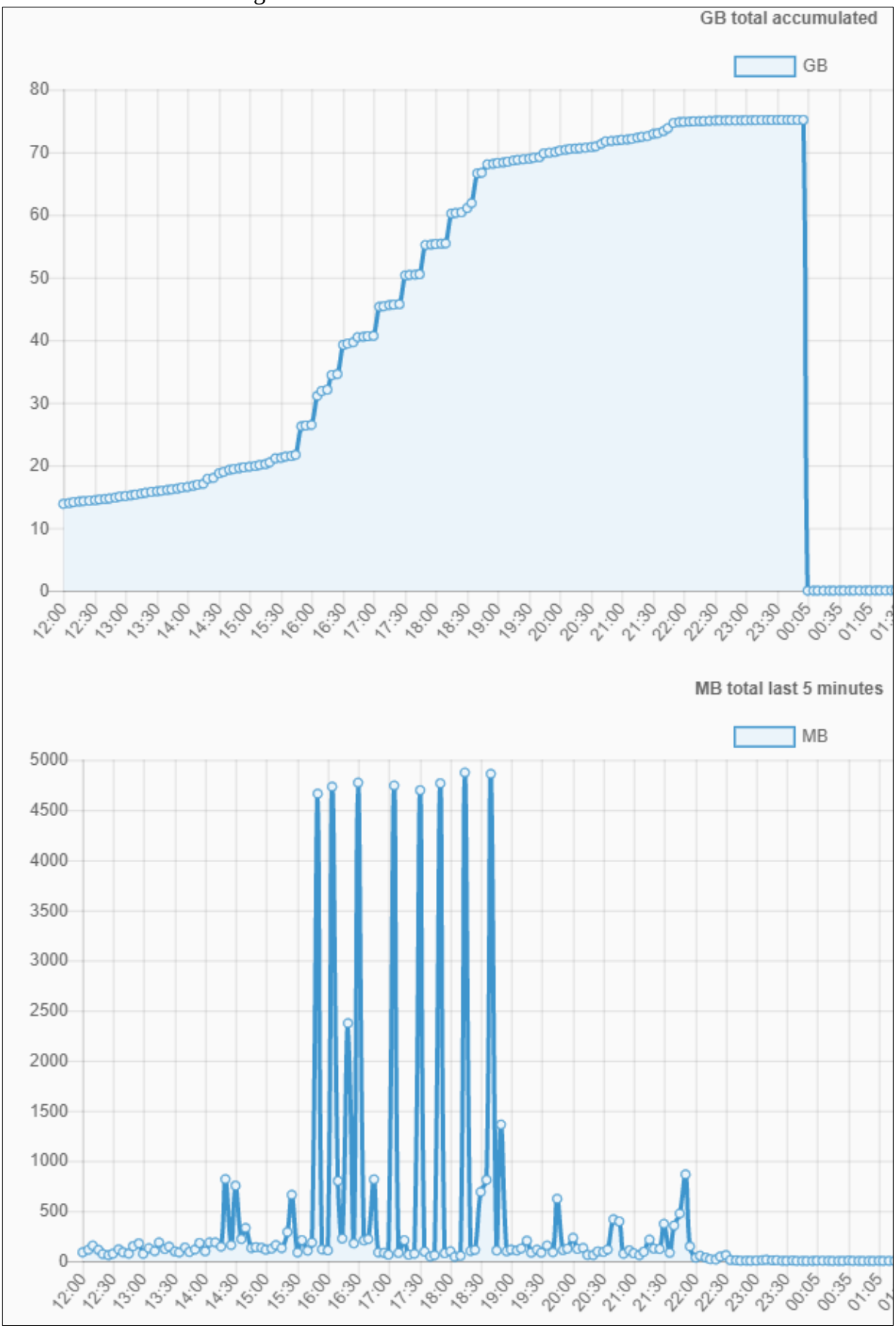


Fonte: elaborado pelo autor.

Neste gráfico pode-se notar que a utilização da rede tem maior concentração entre às 7 horas e às 22 horas, sendo que o pico se concentra no período da manhã. Se alterar o modo de exibição para mostrar os últimos 7 dias, pode-se constatar qual foi o dia da semana em que houve o maior pico.

Com o gráfico demonstrado na figura 26, que demostra os dados de um intervalo de tempo diferente, pode-se afirmar que os usuários estavam acessando algum *streaming* de vídeo, pois esse tipo de acesso utiliza o protocolo UDP. Ao analisar os gráficos demonstrados na figura 29, pode-se descobrir em que momento foi trocado uma grande quantidade de dados. No gráfico que exibe a estatística dos últimos 5 minutos pode-se notar que em oito ocasiões foram transferidos mais de 4 GB de dados. A partir disso é possível rastrear quais foram os top IPs responsáveis por trafegar esse volume de dados acessando o gráfico demonstrado na figura 27.

Figura 29 – Histórico dos dados transferidos



Fonte: elaborado pelo autor.

4 CONCLUSÕES

O objetivo deste trabalho foi desenvolver uma ferramenta para prover informações do comportamento e desempenho da rede analisando seu fluxo de dados. Para que isso fosse possível, esperava-se observar e exportar o fluxo de dados de um equipamento central, coletar e armazenar os dados exportados e por fim, analisar e exibir os dados em forma de gráfico, em uma página Web. Todos objetivos foram alcançados. O desenvolvimento do presente trabalho permitiu uma análise de como o monitoramento pode realmente auxiliar o administrador a conhecer melhor a rede que gerencia. Além disso, possibilitou que o monitoramento fosse ágil, devido ao fato de os dados serem exibidos em forma de gráfico. Os gráficos permitem que o administrador tenha em mãos informação atualizada e com isso tome as decisões necessárias, caso alguma anomalia seja detectada.

As ferramentas utilizadas no desenvolvimento deste trabalho foram eficazes e importantes, pois permitiram que os objetivos fossem alcançados. Destaca-se o uso do conjunto de ferramentas nfdump, tendo em vista que simplificou a captura e análise do fluxo de dados, possibilitando inclusive a extração de diversas estatísticas.

Com relação aos trabalhos correlatos, o presente trabalho colaborou para oferecer mais uma opção de fonte de dados de uso da rede, que é o fluxo de dados. Os dados são exibidos ao administrador por meio de gráficos amigáveis e uma página Web responsiva.

No que diz respeito ao meio acadêmico, notou-se que o assunto é pouco explorado pela academia, sendo as grandes empresas detentoras das maiores soluções, e deste modo, este trabalho auxilia na disseminação do conhecimento nesta área.

A ferramenta desenvolvida possui algumas limitações, dentre as quais: exibir um grande volume de dados, causa leve lentidão no navegador e resulta na demora em exibir o gráfico. Isto está diretamente ligado ao desempenho do computador. Outra limitação é o fato de que os gráficos em pizza mostram apenas a estatística dos últimos 5 minutos.

4.1 EXTENSÕES

Além da correção das limitações citadas na conclusão, sugere-se para trabalhos futuros:

a) persistir os dados em banco de dados;
b) implementar controle de acesso;
c) emitir alerta no navegador;
d) enviar alerta por e-mail;
e) enviar relatório diário, semanal e mensal.

REFERÊNCIAS

AMARAL, A. DE A. **Inferindo a fonte e o destino do tráfego anômalo em redes de computadores usando correlação espaço - temporal**. [s.l.] Universidade Estadual de Campinas, 2011.

BENNERTZ, E. **Ferramenta para monitoração e gerenciamento de tráfego em uma rede local**. [s.l.] UNIVERSIDADE REGIONAL DE BLUMENAU, 2014.

FERNANDEZ, D. et al. Tools for Managing Network Traffic Flows : a Comparative Analysis. **2017 IEEE 16th International Symposium on Network Computing and Applications (NCA)**, p. 1–5, 2017.

GREŽO, R.; NAGY, M. Network Traffic and Infrastructure Analysis in Software Defined Networks. **2017 3rd IEEE International Conference on Computer and Communications Network**, p. 541–546, 2017.

GUPTA, A. Network management: Current trends and future perspectives. **Journal of Network and Systems Management**, v. 14, n. 4, p. 483–491, 2006.

HOFSTEDE, R. et al. Flow monitoring explained: From packet capture to data analysis with NetFlow and IPFIX. **IEEE Communications Surveys and Tutorials**, v. 16, n. 4, p. 2037–2064, 2014.

KARING, A. **Protótipo de um sistema de monitoramento de desempenho de redes de computadores baseado no protocolo SNMPv3**. [s.l.] UNIVERSIDADE REGIONAL DE BLUMENAU CENTRO, 2002.

KUROSE, JIM F. ; ROSS, K. W. **Redes de Computadores e a Internet - Uma Abordagem Top-Down**. São Paulo: Pearson, 2013.

LINGNAU, L. **Monitoramento de servidores e dispositivos de rede utilizando snmp**. [s.l.] UNIVERSIDADE REGIONAL DE BLUMENAU, 2012.

OUSTERHOUT, J. K. Scripting: higher level programming for the 21st Century. **Computer**, v. 31, n. 3, p. 23–30, 1998.

RAMAN, L. OSI systems and network management. **IEEE Communications Magazine**, v. 36, n. 3, p. 46–53, 1998.

STALLINGS, W. **Redes e Sistemas de Comunicação de Dados: Teoria e Aplicações corporativas**. Rio de Janeiro: Elsevier Ltd, 2005.

TRAMMELL, B.; WAGNER, A.; CLAISE, B. Specification of the IP Flow Information Export (IPFIX) Protocol for the Exchange of Flow Information. **IETF RFC**, p. 1–76, 2013.

VAN ADRICHEM, N. L. M.; DOERR, C.; KUIPERS, F. A. OpenNetMon: Network monitoring in OpenFlow software-defined networks. **2014 IEEE Network Operations and Management Symposium (NOMS)**, p. 1–8, 2014.