PONTIFÍCIA UNIVERSIDADE CATÓLICA DO RIO GRANDE DO SUL
FACULDADE DE ENGENHARIA ELÉTRICA

GLÉDERSON LESSA DOS SANTOS

SISTEMA BASEADO NAS RECOMENDAÇÕES IPFIX PARA EXPORTAÇÃO E ANÁLISE DE INFORMAÇÕES DE FLUXOS EM REDES CONVERGENTES

Porto Alegre
2007

GLÉDERSON LESSA DOS SANTOS

SISTEMA BASEADO NAS RECOMENDAÇÕES IPFIX PARA EXPORTAÇÃO E ANÁLISE DE INFORMAÇÕES DE FLUXOS EM REDES CONVERGENTES

Dissertação apresentada como requisito para obtenção do grau de Mestre, pelo Programa de Pós-graduação da Faculdade de Engenharia Elétrica da Pontifícia Universidade Católica do Rio Grande do Sul.

Orientador: Dr. Fabian Vargas
Co-orientador: Dr. Jorge Guedes Silveira

Porto Alegre
2007

“*Para meus pais, irmãos e Adriane*”

AGRADECIMENTOS

A CAPES (Coordenação de Aperfeiçoamento de Pessoal de Nível Superior), pelo financiamento e incentivo para a realização do meu mestrado.

A VIVO, pelo incentivo e apoio prestados durante o desenvolvimento desse trabalho.

Ao professor Dr. Fabian Vargas, por ter aceito o convite para me orientar na execução do presente trabalho bem como pelo apoio prestado durante todo este processo.

Ao professor Dr. Jorge Guedes Silveira, pela confiança e apoio prestados no decorrer do meu mestrado.

Aos amigos Roberto Costa, Vinícius Guimarães, Mateus Caruccio, Ricardo Balbinot e Eloísio Bergamaschi pelo auxilio prestado durante todo processo de elaboração deste trabalho.

Aos grandes amigos com quem tive o privilégio de trabalhar durante o período de mestrado, em especial aos componentes do GPARC&TI.

A minha família, com os quais eu sei que, independentemente da distância, poderei sempre contar.

A minha namorada Adriane, por toda paciência, amor e auxílio durante o desenvolvimento deste trabalho.

A Deus, pela luz nos momentos mais difíceis.

“Caminante no hay camino
se hace camino al andar.”
Antonio Machado

RESUMO

A tendência no uso de redes convergentes baseadas na tecnologia IP leva a uma necessidade cada vez maior de se manter a qualidade, a gerência controlada e a disponibilidade dos diferentes serviços oferecidos. Nesse contexto, a obtenção de indicadores de desempenho (KPI) que permitam o gerenciamento e planejamento da rede cresce em importância, fazendo com que diversas alternativas com esse objetivo sejam pesquisadas. Conseqüentemente surgem, por parte de diferentes órgãos normativos, iniciativas buscando a padronização de sistemas capazes de extrair esses indicadores da rede. Entre esses órgãos, o *Internet Engineering Task Force* (IETF) buscou formar diversos grupos de trabalho com objetivos bastante específicos dentro desse contexto, sendo que a especificação da exportação das informações de fluxo ficou a cargo do *IP Flow Information Transport* (IPFIX). Estas informações vêm sendo largamente utilizadas em um grande número de aplicações, incluindo sistemas de engenharia de tráfego, planejamento de redes, gerência de serviços e detecção de ataques. Seguindo as recomendações do IPFIX, o presente trabalho foca no desenvolvimento de um sistema para aquisição, exportação e análise de informações de fluxo. Para tanto, são implementados e avaliados cada um dos elementos responsáveis pelo transporte das informações de fluxo, bem como uma aplicação *case* de geração de matriz de tráfego. A avaliação do protótipo implementado foi realizada em um ambiente real e controlado, visando verificar a sua estabilidade e precisão. A partir dos resultados obtidos, foi realizada uma adaptação ao sistema, melhorando assim o seu desempenho em aplicações que necessitam de uma base histórica de medições.

Palavras-chave: IPFIX, análise de fluxos, medição passiva, medição e monitoramento de redes IP, matriz de tráfego.

ABSTRACT

The tendency of IP based convergent networks technology makes necessary the quality of service maintenance, controlled management and offer of different services availability. In this way, the importance of performance indicators (KPI – Key Performance Indicators) acquirement that allows network planning and management grows, leading several alternatives research. In order to obtain the performance indicators from the network, different institutions established standardization systems. Among them, Internet Engineering Task Force (IETF) formed specific task work groups and the information flow export specification was under IP Flow Information Transport (IPFIX) responsibility. This kind of information has been used in an expressive number of application including traffic engineering tasks, network planning, service management and attack detection. Following the IPFIX recommendations, present work focuses in a development of a system to acquire, store, export and analyze flow statistics. Therefore, each one of flow information transport elements are implemented and evaluated as well as a Traffic Matrix application. The prototype was evaluated in a real and controlled environment to verify stability and accuracy. Based on obtained results, an adaptation was accomplished in order to improve the performance of system in applications that need a historical base of measurements.

Keywords: IPFIX, flows analysis, passive measurement, IP networks measurement and monitoring, traffic matrix.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

ACRÔNIMOS

AMP	*Active Measurement Project*
BGP	*Border Gateway Protocol*
CBR	*Constant Bit Rate*
DARPA	*U.S. Defense Advanced Research Projects*
DoS	*Denial of Service*
DSCP	*Diffserv Codepoint*
DTD	*Document Type Definition*
DTLS	*Datagram Transport Layer Security*
ER	Entidade-Relacionamento
GPARC&TI	Grupo de Pesquisas Avançadas em Redes de Comunicação e Tecnologia da Informação
GPS	*Global Positioning System*
HTTP	*Hypertext Transfer Protocol*
IANA	*Internet Assigned Numbers Authority*
ICMP	*Internet Control Message Protocol*
IETF	*Internet Engineering Task Force*
IP	*Internet Protocol*
IPFIX	*IP Flow Information Transport*
IPPM	*IP Performance Metrics*
IPSec	*Internet Protocol Security*

ISP	*Internet Service Provider*
ITU	*International Telecommunication Union*
KPI	*Key Performance Indicator*
MIB	*Management Information Base*
MPLS	*Multiprotocol Label Switching*
MVCC	*Multi-Version Concurrency Control*
NeTraMet/NeMac	*Network Traffic Meter & Network Manager Collector*
NMS	*Network Management System*
NTP	*Network Time Protocol*
OWD	*One-Way Delay*
PMA	*Passive Measurement and Analysis*
PSAMP	*Packet Sampling*
PUCRS	Pontifícia Universidade Católica do Rio Grande do Sul
QoS	*Quality of Service*
RFC	*Request For Comments*
RSVP	*Reservation Protocol*
RTFM	*Real-Time Traffic Flow Measurement*
RTT	*Round-Trip-Time*
SCTP	*Stream Control Transmission Protocol*
SGBD	Sistema Gerenciador de Banco de Dados
SGML	*Standard Generalized Markup Language*

SLA	*Service Level Agreement*
SLM	*Service Level Management*
SLS	*Service Level Specification*
SMI	*Structure of Management Information*
SNMP	*Simple Network Management Protocol*
SSL	*Secure Sockets Layer*
TCP	*Transmission Control Protocol*
TEWG	*Traffic Engineering Working Group*
TI	Tecnologia da Informação
TLS	*Transport Layer Security*
TMN	*Telecommunications Management Network*
ToS	*Type of Service*
TTM	*Test Traffic Measurement Service*
UDP	*User Datagram Protocol*
VoIP	*Voice Over IP*
VPN	*Virtual Private Network*
W3C	*World Wide Web Consortium*
XML	*Extensible Markup Language*

SUMÁRIO

1 INTRODUÇÃO

A tendência no uso de redes convergentes como meio de comunicação, unindo serviços em uma única infra-estrutura e fazendo com que recursos sejam compartilhados por diferentes aplicações, cada qual com requisitos distintos, leva a uma necessidade cada vez maior de se manter a qualidade, a gerência controlada e a disponibilidade dos diferentes serviços oferecidos.

Nesse contexto, soluções baseadas em redes que utilizam o *Internet Protocol* (IP) para transporte de dados, voz, serviços multimídia e serviços com requisitos específicos como, por exemplo, *Virtual Private Networks* (VPN), ou redes privadas virtuais, e redes *peer-to-peer*, têm obtido cada vez mais espaço no mercado. Entretanto, a natureza da rede IP, baseada no modelo *Best Effort*, onde os pacotes são enviados a rede de forma indiscriminada, torna-se um obstáculo no seu uso em aplicações que exijam parâmetros de qualidade de comunicação mais rígidos.

Assim, para que a rede IP possa suportar requisitos distintos, variáveis de acordo com os serviços associados e o grau de utilização dos mesmos, faz-se necessário o desenvolvimento de novas alternativas capazes de identificar, monitorar e controlar dinamicamente os recursos da rede, evitando, dessa forma, que o comportamento de um tipo de tráfego influa em outro tráfego.

A identificação e o monitoramento dos recursos da rede passam, de forma geral, pelo uso de técnicas de medição ativa ou passiva. Enquanto que a medição ativa propõe a injeção de tráfego de teste através da rede que se deseja monitorar, a medição passiva utiliza o monitoramento do próprio tráfego passante para a aferição da rede, através da adição de pontos de medição, sem que haja a necessidade de adição de tráfego concorrente na mesma.

No contexto da medição passiva, o incremento da quantidade de tráfego passante através dos pontos de observação, ocasionado pelo aumento na capacidade de transporte nas redes de alta velocidade atuais, tende a gerar uma grande quantidade de dados de medição, ocasionando a exaustão de recursos deste sistema de medição.

Dada a limitação de recursos desses sistemas como, por exemplo, a exaustão dos *buffers* de memória dos medidores ligados aos pontos de observação ou a capacidade de transmissão destes, técnicas de medição passiva baseadas na análise dos fluxos ganham destaque.

As diversas alternativas apresentadas nos últimos anos, que utilizam medição baseada em fluxos, acabaram por gerar uma grande dificuldade no que diz respeito à capacidade de obtenção de níveis aceitáveis de interoperabilidade entre cada uma propostas e no desenvolvimento de novas ferramentas que analisem essas informações. Tudo isso motivou o *Internet Engineering Task Force* (IETF) a criar grupos de trabalho com objetivos bastante específicos dentro desse contexto. Entre os grupos formados, o *IP Flow Information Transport* (IPFIX) recebeu a atribuição de, com base nas principais demandas para a análise de fluxo e no estudo das principais propostas existentes no mercado, desenvolver uma especificação que padronize a exportação das informações de fluxo.

Esse processo culminou com a criação de um protocolo próprio para o transporte dessas informações, suportado por toda uma arquitetura de medição especificada pelo IPFIX.

Seguindo essa tendência e as atuais demandas no que diz respeito à medição de tráfego baseada em fluxos, juntamente com as necessidades de pesquisa e desenvolvimento de sistemas de medição e monitoramento de redes a serem supridas pelo Grupo de Pesquisas Avançadas em Redes de Comunicação e Tecnologia da Informação (GPARC&TI), o presente trabalho busca desenvolver todo um sistema de aquisição, exportação e armazenamento das informações de fluxo. Para tanto, faz-se necessária a realização de todo um levantamento sobre as atuais demandas nesse sentido, bem como das ferramentas existentes no mercado com essa função e das especificações definidas pelo IPFIX. Da mesma forma, o trabalho descreve também a prototipação de uma ferramenta *case* de análise de dados de fluxo, que objetiva o desenvolvimento de uma aplicação de aquisição de informações e desenvolvimento de uma matriz de tráfego.

1.1 OBJETIVOS

De acordo com o que foi anteriormente exposto, o presente trabalho tem como objetivo geral o desenvolvimento de um sistema de medição de fluxos baseado nas especificações IPFIX e a análise quanto a precisão de seus resultados quando aferindo um sistema real controlado.

Mais precisamente, essa tarefa passa pela implementação de três sistemas:

- o sistema observador de fluxos, que realiza a aquisição dos dados de fluxo;
- o sistema coletor, que se comunica com observadores de fluxo por meio do protocolo IPFIX, armazenando as informações de fluxo em uma base de dados desenvolvida especificamente para esse fim;
- o sistema de análise de fluxos, que, a partir das informações de fluxo na base de dados, gera uma matriz de tráfego, método de análise de tráfego amplamente utilizado em sistemas de monitoramento de redes.

1.2 ORGANIZAÇÃO DA DISSERTAÇÃO

Este trabalho foi organizado de forma a, inicialmente, proporcionar uma contextualização acerca do processo de medição de tráfego baseado em fluxos, aprofundando o assunto no decorrer do texto, sendo escrito de acordo com o modelo de elaboração de dissertação elaborado e disponibilizado pela Pontifícia Universidade Católica do Rio Grande do Sul (PUCRS).

Após este capitulo introdutório, o capítulo 2 apresenta a revisão da literatura relacionada ao assunto aqui tratado. Para isto, inicialmente o capítulo relaciona os principais tópicos tratados no gerenciamento de redes e na engenharia de tráfego com a necessidade de obtenção de parâmetros de desempenho, destacando as principais técnicas de obtenção destes. Após esta etapa, é realizado um aprofundamento na medição de tráfego baseado em fluxos,

finalizando a análise com as principais propostas de sistemas que realizam esse método de medição.

A partir da contextualização apresentada no capítulo 2, o capítulo 3 realiza o levantamento sobre as especificações IPFIX, destacando em especial os conceitos que envolvem a arquitetura base para a medição de fluxos e a forma de interação entre o dispositivo observador e o coletor IPFIX, através do protocolo de comunicação elaborado para troca de mensagens entre estes elementos.

O capítulo 4 destaca a prototipação realizada com base no levantamento de requisitos e especificações propostas pelo IPFIX, analisando de forma detalhada a implementação de cada sistema que compõe a arquitetura de medição e análise de tráfego.

O capítulo 5 foca nos resultados obtidos com o sistema implementado, destacando seus aspectos positivos e suas limitações, bem como eventuais adequações e modificações realizadas em seus módulos visando a melhora no desempenho do sistema como um todo.

Finalmente, o capítulo 6 apresenta as conclusões obtidas a partir do trabalho elaborado. Também são destacados neste capítulo propostas para trabalhos futuros acerca do tema aqui proposto.

2 REVISÃO BIBLIOGRÁFICA

O presente capítulo visa apresentar os conceitos e as tecnologias que fundamentam este trabalho, assim como definir o estado da arte das tecnologias correlacionadas. Cada um dos itens aqui abordados justifica e forma a base para o desenvolvimento do presente trabalho.

2.1 *QUALITY OF SERVICE* (QoS)

A internet tem suas raízes no arpanet, uma rede de dados experimental desenvolvida pelo DARPA (*U.S. Defense Advanced Research Projects*) no início dos anos 60. De acordo com Wang [WAN 01], uma importante meta desse projeto foi construir uma rede robusta, capaz de permanecer ativa mesmo após ataques militares, como bombardeios. Devido a isso, a rede arpanet foi construída num modelo de datagramas, onde cada pacote é enviado de forma independente ao seu destino. As grandes vantagens do modelo em datagramas são a simplicidade e a habilidade de se adaptar rapidamente a mudanças na topologia da rede.

Quando se tratando de aplicações como transferência de arquivos, troca de e-mails, *Hypertext Transfer Protocol* (HTTP), esse modelo funciona bem. Entretanto, com o surgimento de novas aplicações como vídeo conferência, *voice over IP* (VoIP) e transmissão de mídia eletrônica (*video streaming* e *audio streaming*) os usuários da rede passaram a ter requisitos diferentes daqueles com os quais a internet foi inicialmente projetada.

O modelo em datagramas possui pouca capacidade de gerenciamento de recursos e não pode prover qualquer garantia de recursos aos usuários de forma natural. Além disso, como o protocolo IP teve sua criação associada ao modelo *best effort* o qual tem como premissa que, em caso de congestionamento em um *link*, os pacotes passem a ser descartados indiscriminadamente. Assim, a internet trata todos os pacotes da mesma forma, oferecendo apenas um nível de serviço. Entretanto as aplicações em geral necessitam da diferenciação de serviços, devido à já comentada variação dos requerimentos de acordo com a aplicação utilizada. Um serviço de VoIP, por exemplo, é sensível a atrasos na transmissão ou perda dos

pacotes. Em contraste, uma transferência de arquivos pode tolerar sem maiores problemas perdas de pacotes e atrasos.

Essa variedade de serviços faz com que o perfil dos usuários da rede também varie. Por exemplo, organizações que usam a internet para transações financeiras ou serviços de telemetria necessitam de um serviço onde seu trafego receba tratamento preferencial. De acordo com [WAN 01] a capacidade de prover garantia de recursos e diferenciação de serviços em uma rede é chamado de *Quality of Service* (QoS).

Já [CIS 03], apresenta como definição:

> Qualidade de Serviços refere-se à capacidade de uma rede em prover da melhor forma possível um determinado serviço, para um determinado tráfego selecionado, tendo como objetivo primordial o provisionamento de prioridade, incluindo largura de banda dedicada, atraso e *jitter* controlados e melhoramento nas características de perda.

A *International Telecommunication Union* (ITU) define QoS como sendo o efeito coletivo sob o desempenho do serviço, o qual determina o grau de satisfação de um usuário em relação a esse serviço [ITU 01].

Para verificar os níveis de QoS adequados, o ITU definiu, através da recomendação Y.1541 [ITU 02], a análise das seguintes métricas de desempenho:

- *one-way delay* (OWD), ou atraso-fim-a-fim;
- variação no atraso (*jitter*);
- taxa de perda de pacotes;
- taxa de erros em pacotes IP.

Paralelamente, o IETF, através de seu grupo de trabalho *IP Performance Metrics* (IPPM), estabelece também o uso de outros parâmetros, como a métrica de *throughput*, que avalia a largura de banda disponível para uma aplicação, a verificação da ordem de chegada dos pacotes e o atraso em ida-e-volta (*round-trip delay*).

2.2 *SERVICE LEVEL MANAGEMENT* (SLM)

A partir da idéia de que cada aplicação exige níveis de QoS adequados, surge o conceito de *Service Level Management* (SLM), ou Gerenciamento de Níveis de Serviço. De acordo com Morris *et al*. [MOR 01], SLM são procedimentos aplicados de modo a garantir que níveis adequados de serviços sejam prestados a todos os usuários, levando em consideração a prioridade relativa e a importância comercial de cada um.

Para o melhor entendimento do SLM, faz-se necessário primeiramente o conhecimento de dois conceitos complementares: o *Service Level Agreement* (SLA), ou Acordo de Níveis de Serviço e o *Service Level Specification* (SLS), ou Especificação de Níveis de Serviço.

Os SLA são essenciais para o gerenciamento da qualidade dos serviços prestados ou contratados por uma organização de tecnologia da informação (TI) [MOR 01]. O SLA pode ser definido como um contrato negociado entre as partes envolvidas na contratação de um serviço de telecomunicações ou TI.

Associado ao SLA, o SLS caracteriza os parâmetros técnicos de desempenho firmados entre o cliente e o provedor de um serviço.

Esses contratos firmados compreendem as camadas de maior abstração no *Network Management System* (NMS), ou Sistema de Gerenciamento de Rede. Segundo Räisänen [RÄI 03], o SLM pode, conceitualmente, ser separado em quatro camadas. Em ordem decrescente de abstração, essas camadas são as seguintes:

- camada de Gerenciamento de Negócios: responsável pelos aspectos de negócios dos acordos entre as diferentes partes envolvidas com a prestação do serviço. Nesta camada estão as questões acordadas através do SLA;
- camada de Gerenciamento de Serviço: lida com a parte técnica dos contratos realizados entre os provedores de serviços e os clientes, definidas no SLS, através do controle sobre a camada de gerenciamento de rede;
- camada de Gerenciamento de Rede: provê ao usuário do NMS vislumbrar, em nível de rede, o comportamento de serviços e elementos de rede, atuando, a partir de sua

ligação com a camada de gerenciamento de elemento, no controle individual de cada elemento de rede;

- camada de Gerenciamento de Elemento: interface entre o NMS e cada elemento de rede.

O ITU introduziu o termo *Telecommunications Management Network* (TMN) em sua recomendação M.3010 [ITU 00]. O TMN define cinco áreas funcionais que são abrangidas pelas quatro camadas do SLM, descritas a seguir:

- gerenciamento de falhas, que compreende as tarefas relacionadas com as condições de erro da rede, incluindo a identificação da falha, a informação de sua ocorrência e sua correção;
- gerenciamento de configurações, que se relaciona com o provisionamento de serviços nos elementos de rede, incluindo a configuração destes elementos;
- gerenciamento de utilização, que se concentra nas informações acerca do uso da rede e dos serviços oferecidos através desta;
- gerenciamento de desempenho, cujo foco está centrado na obtenção de indicadores de desempenho dos elementos de rede e serviços;
- gerenciamento de segurança, que inclui os mecanismos responsáveis pelo provimento de segurança.

2.3 GERENCIAMENTO DE REDES

Conforme comentado na seção anterior, a gerência de redes permite o monitoramento e o controle dos recursos desta. Segundo Gaspary *et al* [GAS 01], a utilização das redes de computadores como suporte a um crescente número de negócios e aplicações críticas tem estimulado a busca por soluções de gerenciamento que permitam manter em funcionamento não apenas a infra-estrutura física da rede, mas também os protocolos e serviços que a compõem.

Falhas em elementos ou em partes das redes podem causar impacto negativo nos negócios dos provedores de serviços e levar a quebra de acordos entre estes e seus clientes. Isso torna necessário o investimento em soluções integradas de gerenciamento.

Nesse contexto, em [CIS 06a] é apresentada uma listagem das funcionalidades básicas que devem ser disponibilizadas em um sistema de gerenciamento padrão, destacadas a seguir:

- *discovery* de rede, funcionalidade capaz de descobrir quais os elementos que compõem a rede;
- um mapa topológico mostrando a interconexão entre os elementos de rede;
- um módulo para tratamento de eventos;
- um coletor de dados de desempenho;
- um navegador que permita a verificação dos dados de gerenciamento.

A ampla gama de soluções desenvolvidas para o gerenciamento de redes trouxe a necessidade de criar padrões para o desenvolvimento de NMS. É destacada neste trabalho a família de especificações do *Simple Network Management Protocol* (SNMP), definida pelo IETF para especificar uma arquitetura de gerenciamento de redes. OSNMP é descrito na subseção seguinte.

2.3.1 SNMP

O SNMP refere-se a um conjunto de padrões para gerenciamento que inclui:

- um conjunto de especificações de estrutura e identificação para as informações gerenciais, chamado *Structure of Management Information* (SMI), [ROS 90];
- um protocolo de comunicação entre o gerente e o agente, que recebe o nome de SNMP. A especificação deste protocolo e apresentada no RFC 1157 [CAS 90];
- uma base de informações gerenciais que especifica quais variáveis são mantidas pelos elementos de rede. Esta base de informações é denominada MIB (*Management Information Base*) [PRE 02].

As especificações do SNMP foram desenvolvidas visando ser utilizadas para que os dispositivos de rede troquem suas informações de gerenciamento, através do acesso a suas respectivas MIBs. O SNMP possibilita aos administradores de rede gerenciar o desempenho da mesma, identificando e solucionando falhas, e planejando o crescimento dela.

2.4 ENGENHARIA DE TRÁFEGO EM REDES IP

A partir das seções anteriores, percebe-se a necessidade do gerenciamento dos parâmetros de QoS. Associada a esse contexto, a engenharia de tráfego surge como alternativa para a proposição de técnicas e ferramentas que objetivam a melhora no desempenho da rede e, conseqüentemente, a melhora nesses parâmetros.

Um dos problemas básicos, comum a qualquer técnica de endereçamento em redes é, dada uma rede com suas respectivas demandas de tráfego, como organizar esse tráfego de modo a otimizar o seu funcionamento. Esse é o objetivo primordial da engenharia de tráfego, a partir do qual são traçadas suas principais definições.

Seguindo essa idéia, Wang [WAN 01] cita que o objetivo da engenharia de tráfego é organizar o tráfego na rede de modo a otimizá-la, seja para maximizar a utilização dos recursos na rede ou para minimizar o congestionamento da mesma.

Esse aprimoramento do desempenho de redes operacionais, tanto em termos de uso de recursos quanto do ponto de vista das exigências do tráfego passa pela implementação de um sistema de suporte de QoS que utilize a menor quantidade de recursos possível. Exemplos de recursos envolvidos incluem largura de banda utilizada em um *link*, processamento dos equipamentos da rede e a capacidade de memória dos mesmos.

Por sua vez, [RÄI 03] define engenharia de tráfego como um conjunto de técnicas e ferramentas que podem ser utilizadas para otimizar o desempenho de uma Rede IP operacional.

Outras áreas de concentração da engenharia de tráfego estão focadas no melhoramento e na manutenção e confiabilidade das operações da rede. Em casos atípicos como a falha de equipamentos de rede, é necessária a habilidade de detecção do erro, assim como a habilidade de gerar uma resposta a essa situação, baseada nesta detecção.

De forma geral, essas definições convergem para o que foi apresentado pelo grupo de trabalho do IETF *Traffic Engineering Working Group* (TEWG) em um amplo estudo sobre engenharia de tráfego. Esse estudo, publicado na RFC 3272 [AWD 02], define engenharia de tráfego como a aplicação de princípios tecnológicos e científicos visando a medição, caracterização, modelagem e controle do tráfego da internet.

Segundo Wang [WAN 01], esses objetivos não podem ser alcançados por meio do roteamento tradicional das redes IP, baseado apenas no endereço de destino. Na engenharia de tráfego, técnicas de roteamento mais elaboradas, normalmente denominadas *constraint-based routing*, para distinguir do roteamento clássico do IP, são usadas, buscando a otimização das rotas. Para permitir essa otimização, o sistema de engenharia de tráfego precisa conhecer informações sobre a topologia e a demanda de tráfego de toda a rede.

De acordo com Räisänen [RÄI 03] a Engenharia de Tráfego é também uma base para a administração de uma rede IP multi-serviço de uma maneira eficiente em termos de custos (*cost-efficient*). Suportar serviços na rede em um ambiente comercial requer que o status da rede seja monitorado, otimizado e controlado.

Especialmente com o crescente volume de tráfego, a efetividade da engenharia de tráfego também tem implicação no planejamento de rede e, quando utilizado, em decisões de controle de admissão. Finalmente, a engenharia de tráfego tipicamente faz uso de mecanismos de suporte de QoS.

2.4.1 O Processo de engenharia de tráfego.

Segundo [AWD 02], o processo de engenharia de tráfego é definido como consistindo das seguintes partes:

- definição de políticas de controle relevantes. Este pode ser considerado como estando fora do ciclo de engenharia de tráfego atual, controlando seu progresso. Naturalmente políticas de controle são ajustadas baseadas no desempenho da rede observada;
- mecanismos de realimentação para aquisição dos dados de desempenho da rede de produção (observação da rede);
- análise do estado de rede e caracterização da carga de trabalho de tráfego;
- otimização do desempenho da rede.

O processo de engenharia de tráfego é definido na Figura 1 .

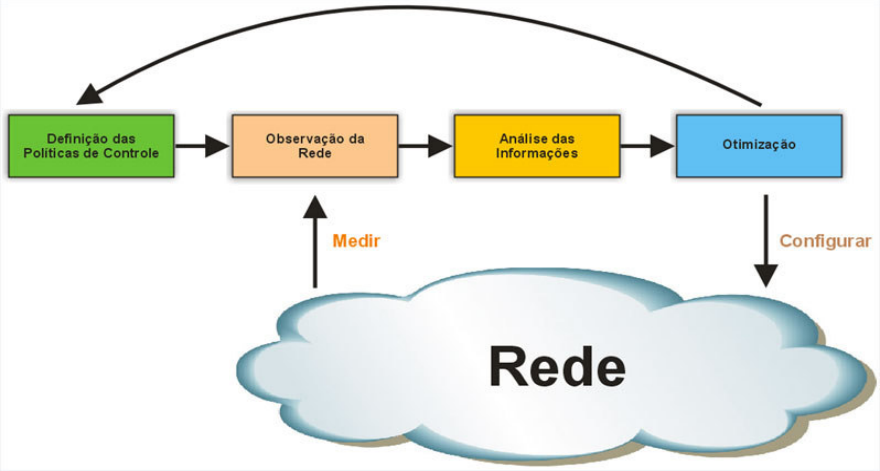


Figura 1 O processo de engenharia de tráfego[AWD 02].

2.5 OBTENÇÃO DE INDICADORES ESTATÍSTICOS DA REDE

A manutenção da qualidade, da capacidade de gerenciamento, da conformidade com características de desempenho aceitáveis e da disponibilidade dos diferentes serviços oferecidos aos usuários de uma rede ganha cada vez mais importância, apesar da complexidade associada a esse processo, conforme pôde ser visto nas seções anteriores deste capítulo. A aquisição de indicadores estatísticos da rede permite vislumbrar cada um desses fatores, permitindo desde o simples controle das condições da rede até o uso de técnicas mais elaboradas de planejamento e gerenciamento proativo da mesma. De acordo com Räisänen

[RÄI 03], essa aquisição dos indicadores da rede pode ser feita usando várias fontes em diferentes níveis de abstração. Cada uma dessas fontes pode ser desde um elemento de rede distinto, consultado individualmente para a aquisição de informações, até um NMS, elemento para aquisição de dados de desempenho de maior nível de abstração. A Tabela 1 mostra os diferentes níveis de monitoramento possíveis.

Tabela 1 Níveis de monitoramento da rede.

Nível de Monitoramento	Características Tipicamente Mensuradas
Elemento de Rede	Carga completa e estatística para cada tráfego agregado
Sistemas de Gerenciamento IP	Dados da rede brutos, médias e análises de tendências
Desempenho agregado	Atraso, *jitter*, perda de pacotes e largura de banda disponível em um domínio de rede
Nível de serviço	Especificação de Níveis de Serviço

Fonte: *Implementing Service Quality in IP Networks* [RÄI 03].

Basicamente, dentro desses níveis de monitoramento, existem três métodos de aquisição de indicadores de desempenho da rede. São eles:

- medição passiva;
- medição ativa;
- *piggybacking.*

Cada um destes métodos possui propósitos específicos, cada qual com métricas alcançáveis definidas. Conforme já comentado, o IETF formou o grupo de trabalho IPPM com o objetivo de produzir especificações para cada uma dessas métricas, bem como as metodologias para o cálculo das mesmas. Entretanto, as propostas não abrangem o método de coleta de dados a ser utilizado, apenas o que cada métrica deve aferir. As características de cada um desses métodos são comentadas nas subseções seguintes.

2.5.1 Medição passiva

A medição passiva é caracterizada por não interferir no tráfego da rede, sendo por esse motivo classificada como uma técnica de medição não-intrusiva. Para isso, baseia-se na utilização de um dispositivo físico que observa todo o tráfego passante e armazena as informações de desempenho a partir dessa observação. Esse dispositivo pode ser um elemento que já compõe a rede, como um roteador, desde que ele possua a capacidade de realizar tal tarefa.

Segundo Costa [COS 05], a medição passiva é ideal para a caracterização do tráfego passante, porém implica na necessidade dos pontos a serem aferidos disporem da tecnologia necessária para utilização desse método. Nesta área, destacam-se os trabalhos desenvolvidos pelo projeto PMA (*Passive Measurement and Analysis*) [GRE 00].

A técnica de medição passiva tende a requerer uma capacidade de processamento maior, principalmente quando o sistema de medição está inserido em redes de alta capacidade, dada a grande quantidade de dados a serem tratados pelo dispositivo coletor. Além disso, o volume de informação gerado no processo de coleta nesses casos poderia tornar o uso efetivo dessa técnica proibitivo. Para resolver esse problema, diversos trabalhos baseados na aplicação de métodos estatísticos de amostragem vêm sendo propostos, o que é detalhado em maior profundidade na seção 2.8.

As técnicas de observação de tráfego definem como os coletores analisam as informações dos pacotes que passam pela rede. A localização desses coletores e a própria informação que pode ser estimada a partir deles podem variar, como é visto na subseção 2.5.4. Assim, o número de pontos de medição associados a esses coletores é fator determinante no tipo de informação que pode ser extraída da rede. Por exemplo, com um único ponto de medição é possível realizar o monitoramento das características do tráfego passante e do protocolo utilizado [RÄI 03]. Já para aferir métricas como o OWD, são necessários ao menos dois pontos de observação entre os pontos em que se deseja verificar o atraso.

2.5.2 Medição ativa

Ao contrário da medição passiva, a medição ativa se baseia na inserção de tráfego de teste para obtenção de indicadores de desempenho da rede, o que a caracteriza como sendo uma técnica de medição intrusiva. Segundo Costa [COS 05], a medição ativa provê as seguintes informações relacionadas à rede: topologia, largura de banda disponível no gargalo, OWD, *Round-Trip-Time* (RTT), perda, variação no atraso (*jitter*), e grau de desordem dos pacotes. Um exemplo comum de uso do método de medição ativa é o *ping* que, utilizando o *Internet Control Message Protocol* (ICMP) [POS 81], permite, além de obter as estatísticas de RTT, gerar uma série de métricas de desempenho derivadas, como latência e perda de pacotes.

Um problema da medição ativa reside no seu próprio princípio de funcionamento: seu uso ocasiona um consumo de recursos da rede intrínseco e a inserção de competitividade com o tráfego corrente [RÄI 03]. Isso ocorre porque, ao injetar pacotes de testes na rede, chamados *probes*, inevitavelmente está se influenciando no tráfego e, por conseqüência, nas próprias métricas de desempenho. Para minimizar esse problema, o tráfego de teste deve ser o menos agressivo possível. Ao mesmo tempo, os *probes* devem ser injetados na rede seguindo critérios rígidos, definidos a partir da escolha do modelo de tráfego adotado, de modo a tornar a medição o mais próxima possível do tráfego real. Mandrawa [MAN 04] e Mao [MAO 03] propõem em seus respectivos trabalhos ferramentas de medição ativa desenvolvidas para emular o tráfego de aplicações específicas, determinando assim o grau de confiabilidade sob a perspectiva da aplicação.

O método de geração do tráfego de teste define quais métricas podem ser estimadas a partir dos dados coletados. Dessa forma, a escolha do método é determinada, muitas vezes, por restrições impostas pela própria necessidade de determinadas métricas a serem obtidas. Nesse contexto, as três principais formas de envio dos *probes* são descritas a seguir:

- unidirecional: nesse formato de envio de tráfego os *probes* são enviados a partir de uma origem, passam pelo caminho a ser avaliado, e são coletadas por um receptor. Por ser avaliado um único sentido do caminho, determinadas métricas têm seus

cálculos inviabilizados a partir da escolha desse método sem o uso de ferramentas especiais. Um exemplo disso é o atraso;

- bidirecional: esse método consiste na geração simultânea de *probes* nos dois sentidos do caminho a ser avaliado. Os dois pontos iniciam a geração em um único sentido, paralelamente. Esse método, além de permitir a coleta de métricas de ida e volta, ao utilizado juntamente com uma variação no tamanho dos pacotes, resolve alguns problemas relacionados à falta de sincronismo na rede ([PAX 98][SUE 99][LOU 00][ZHA 02][TSU 02]);
- ida e volta: o *ping* funciona utilizando esse método. Os *probes* são gerados a partir de um ponto de origem em direção a um ponto destino, de forma semelhante ao método unidirecional. Entretanto, chegando ao destino, esses pacotes são replicados e reenviados para a origem. Assim, é possível estimar as características de ida e volta dessa comunicação a partir dos dados recebidos na origem, após todo o processo de medição.

Na caracterização do tráfego de teste, outro parâmetro importante de ser tratado é o intervalo entre o envio de *probes*. Os intervalos de geração mais comuns são descritos a seguir.

- *Constant Bit Rate* (CBR): os *probes* são gerados seguindo um modelo determinístico, onde os intervalos entre cada envio são constantes;
- Modulado por Markov: o tráfego gerado segue o modelo de Markov de tempo contínuo, com número finito de estados. Uma taxa de envio de tráfego associada a cada estado da cadeia deve ser definida. Informações sobre a aplicação do modelo de Markov e suas variantes podem ser obtidas em [CHU 95] e [AND 06];
- Pares de Pacotes: neste caso são gerados pares de *probes* consecutivos com espaço entre os mesmos tendendo a zero. Esses pares são então enviados com uma periodicidade constante ou variável, normalmente com o intervalo de tempo entre os pares seguindo o modelo determinístico CBR ou o modelo modulado por Markov. Uma variação dessa técnica, chamada trens de pacotes, utiliza mais de dois *probes* consecutivos formando os agrupamentos. Diversos trabalhos, como [DOV 01] e [ROE 03], propõem utilizar essa técnica no cálculo da métrica de largura de banda no gargalo. Entretanto, de acordo com Paxton [PAX 97], ainda existem dificuldades

no uso de pares de pacotes, como o reordenamento de pacotes, limitações de relógio, mudanças de velocidade máxima e enlaces multi-canais.

- *Trace*: o envio dos *probes* depende de uma configuração prévia, determinada por um arquivo de entrada. Dessa forma, é possível tanto simular um dos modelos anteriormente citados, quanto criar um modelo próprio, que não segue nenhum modelo matemático pré-definido. Assim, é possível realizar a representação de situações reais de uso da rede como, por exemplo, simular o comportamento de uma transmissão de vídeo em tempo real.

Outro problema decorrente do uso dessa técnica de medição é que, no caso de aquisição de métricas unidirecionais dependentes do tempo como, por exemplo, o OWD, as soluções propostas invariavelmente recaem no sincronismo entre os relógios dos pontos que realizam os testes. As soluções mais confiáveis até o momento fazem uso do *Global Positioning System* (GPS) ou outros sistemas específicos para esse fim, o que implica em um investimento extra para aquisição de um sistema desse tipo. No caso de impossibilidade de sistemas específicos de sincronização como o GPS, propostas alternativas aconselham o uso do *Network Time Protocol* (NTP) [MIL 92] para minimizar os problemas de sincronismo entre os pontos de medição.

Destacam-se no desenvolvimento de sistemas de medição ativa as iniciativas do *Active Measurement Project* (AMP) [GRE 00], *Surveyor* [KAL 99] e o *Test Traffic Measurement Service* (TTM) [GEO 01]. Também pode ser destacado o trabalho desenvolvido pelo grupo GPARC&TI na proposição de uma arquitetura de medição ativa (*Unified Active Measurement Architecture* - UAMA) [SAN 07]. Essa arquitetura visa unificar o processo de obtenção de parâmetros de desempenho em uma única ferramenta, o que causa a diminuição no *overhead* imposto à rede quando se utilizando de diferentes ferramentas, cada qual visando à obtenção de um parâmetro de desempenho específico.

Apesar das dificuldades e limitações, o método de medição ativa torna-se bastante atrativo, pois, de acordo com Paxton [PAX 98], aplicações emergentes necessitam de uma averiguação cada vez mais condizente com a perspectiva de seus usuários finais.

2.5.3 *Piggybacking*

De acordo com Jormakka [JOR 00], o *piggybacking* é, até o momento, uma aproximação orientada principalmente à pesquisa, onde o *timestamp* e um número de seqüência são inseridos no *payload* dos pacotes que formam o tráfego.

Räisänen [RÄI 03] comenta a necessidade de se considerar o *overhead* adicionado devido ao uso desse processo de medição. Além disso, o consumo extra de processamento gerado tanto pela inserção quanto pela remoção dessas informações ao *payload*, deve ser avaliado durante a extração dos resultados da medição.

2.5.4 Comparação entre as técnicas de medição

Analisando os aspectos expostos nessa seção, percebe-se que qualquer uma das técnicas de medição apresenta inconvenientes. Enquanto que técnicas de medição passiva tendem a consumir mais recursos do dispositivo de medição, tornando sua implementação mais complexa, técnicas de medição ativa possuem o grave inconveniente de gerar tráfego concorrente na rede.

Da mesma forma, nenhuma das técnicas satisfaz completamente às necessidades associadas ao monitoramento de parâmetros de QoS. A medição ativa tende a ser mais eficiente na obtenção de indicadores de desempenho em um contexto fim-a-fim, principalmente em cenários intra-domínios. Além disso, é também considerada a melhor aproximação para a percepção do desempenho sob a ótica de usuários finais.

Por outro lado, através da medição passiva é possível a obtenção de indicadores do tráfego em trânsito e, por conseqüência, uma caracterização mais precisa de seu comportamento.

Percebe-se, deste modo, que as técnicas tornam-se complementares num cenário de aquisição de indicadores de desempenho da rede e de seus serviços.

2.6 MEDIÇÃO DE TRÁFEGO BASEADA EM FLUXOS

Existem diversas ferramentas disponíveis para aferir o tráfego IP utilizando a medição passiva. Contudo, estas geralmente se baseiam em duas técnicas principais. A primeira técnica é fazer a captura individual dos pacotes (ou até mesmo apenas de seus cabeçalhos) direcionando-os para um arquivo de coleta (*trace file*) e analisando-o posteriormente. Isso é feito por ferramentas como o *tcpdump* [TCP 07]. A grande desvantagem decorrente deste tipo de abordagem é que os arquivos de coleta dos pacotes podem se tornar rapidamente muito grandes, impossibilitando seu tratamento e manutenção.

Uma segunda forma de medição do tráfego é através do armazenamento dos fluxos da rede ao invés dos pacotes. O termo fluxo já é bastante tradicional no contexto de redes, porém possui diferentes definições. Quando tratado exclusivamente no estudo do roteamento, por exemplo, o fluxo é definido como uma seqüência de pacotes partindo de uma origem para um destino, identificado apenas pelos endereços IP de origem e destino, sendo, portanto, unidirecional. No contexto de reserva de recursos, o fluxo é utilizado para descrever o tráfego de dados das aplicações em protocolos como o *Reservation Protocol* (RSVP) [BRA 97]. E, finalmente, segundo Quittek [QUI 04], um fluxo pode ser definido como um conjunto de pacotes IP passando por um ponto de observação da rede durante um certo intervalo de tempo, onde todos os pacotes ditos pertencentes a esse fluxo possuem um grupo de propriedades em comum.

Ainda segundo essa definição, essas propriedades que definem o escopo de pacotes pertencentes a um determinado fluxo são definidas como resultado da aplicação de uma função para os valores de:

- um ou mais campos que formam o cabeçalho de um pacote, o seu cabeçalho relativo à camada de transporte ou o seu cabeçalho relativo à camada de aplicação;

- uma ou mais características do próprio pacote, como, por exemplo, o número de *labels* MPLS (*Multiprotocol Label Switching*) [ROS 01] que formam esse pacote;
- um ou mais campos derivados do tratamento do pacote, como, por exemplo, a verificação da interface de saída à qual esse pacote deve ser direcionado.

Um pacote é definido como pertencente a um fluxo se ele satisfaz completamente todas as propriedades definidas para esse fluxo. A partir dessa definição, visivelmente mais ampla, abre-se espaço para distintas definições de fluxos, de acordo com as necessidades da aplicação que monitora o tráfego, desde um escopo no qual o fluxo contém todos os pacotes observados em uma interface de rede até a definição de um fluxo constituído por um único pacote transmitido entre duas aplicações, com o fluxo sendo definido pelo número de seqüência do pacote IP. Em um exemplo prático, é possível dizer que um fluxo é formado por um grupo de pacotes, que durante um determinado intervalo de tempo, passaram por um determinado ponto de observação e possuíam mesmo protocolo, mesmas portas de origem e destino e mesmos endereços de origem e destino. Essas cinco propriedades (protocolo, porta de origem, porta de destino, endereço de origem e endereço de destino) definirão de forma única cada fluxo para este ponto de observação.

A análise baseada em fluxos facilita a quantificação e a qualificação do tráfego, sem que haja a necessidade de analisar cada interface de rede do segmento e cada pacote que passa por esse segmento individualmente.

Visando tornar seu processamento mais distribuído, sistemas de monitoramento de fluxo envolvem tipicamente duas entidades distintas: um medidor e um coletor. Este primeiro pode ser um *probe* de medição conectado a uma linha ou a uma porta a ser monitorada, um *middlebox* [CAR 02] ou mesmo um roteador, que realiza a medição enquanto encaminha o tráfego. Seja qual for o medidor utilizado, ele deve montar um registro de informações sobre as propriedades aferidas durante a medição (número de pacotes que compõe o fluxo, tamanho desse fluxo em bytes) e propriedades características ao fluxo (IP de origem, IP de destino, porta de origem, porta de destino).

O processo de medição realizado pelo coletor consiste de um grupo de funções que incluem a captura do cabeçalho do pacote, aquisição do tempo em que o mesmo foi observado (*timestamping*), amostragem, classificação e manutenção dos registros de fluxo.

Adicionalmente, a manutenção desses registros de fluxo pode incluir a criação de novos registros, a atualização dos mesmos, o cálculo de dados estatísticos para cada um dos fluxos, a derivação de propriedades desses fluxos, detecção de expiração, passagem dos registros de fluxo para o processo de exportação e remoção desses registros.

Outra tarefa do medidor é exportar as informações para um ou mais processos coletores, os quais possuem as regras de interpretação e disponibilizam essas informações para análise.

Atualmente existe uma série de formatos de exportações utilizados, dificultando a interoperabilidade e o desenvolvimento de ferramentas de análise [MOL 03]. Por esse motivo o IETF criou grupos de trabalhos que objetivam o desenvolvimento de especificações que padronizem o processo de análise de tráfego baseado em fluxos. Entre esses grupos, destacam-se:

- *Real-Time Traffic Flow Measurement* (RTFM);
- *IP Flow Information Export* (IPFIX).

O RTFM é apresentado na seção seguinte, enquanto que o IPFIX é detalhado no capítulo 3 deste trabalho.

2.7 *REAL-TIME TRAFFIC FLOW MEASUREMENT* (RTFM)

O grupo de trabalho *Real-Time Traffic Flow Measurement* (RTFM) foi criado em 1995 pelo IETF para construir uma arquitetura de medição de tráfego baseada em fluxos, com seu trabalho resultando na elaboração de oito *Requests for Comments* (RFCs). O grupo concluiu seus trabalhos em outubro de 2000. A lista de RFCs produzidas compreende:

- RFC 2063: *Traffic Flow Measurement Architecture (Experimental)* [BRO 97a]*;*
- RFC 2064: *Traffic Flow Measurement Meter MIB (Experimental)* [BRO 97b]*;*

- RFC 2123: *Traffic Flow Measurement Experiences with NeTraMet (Informational)* [BRO 97c]*;*
- RFC 2720: *Traffic Flow Measurement Meter MIB (Proposed Standard)* [BRO 99a]*;*
- RFC 2721: *RTFM Applicability Statement (Informational)* [BRO 99b]*;*
- RFC 2722: *Traffic Flow Measurement Architecture (Informational)* [BRO 99c]*;*
- RFC 2723: *SRL: A Language for Describing Traffic Flows and Specifying Actions for Flow Groups (Informational)* [BRO 99d]*;*
- RFC 2724: *New Attributes for Traffic Flow Measurement (Experimental)* [HAN 99].

Seguindo as especificações da arquitetura propostas pela RFC 2722 [BRO 99c], existem quatro componentes básicos no processo de medição, de acordo com a RTFM:

- medidores: colocados nos pontos de observação, registram de forma seletiva a atividade da rede a partir da forma como foram configurados. Eles podem também agregar e realizar o tratamento dos dados antes de armazená-los;
- leitor de medição: a tarefa do leitor é transportar os dados dos medidores, também denominados de tabela de fluxo, para as aplicações de análise;
- aplicações de análise: processam os dados coletados de forma a prover informações úteis para o gerenciamento de redes e engenharia de tráfego. Exemplos de aplicações incluem: matrizes de tráfego, que mostram as taxas de fluxo que trafegam no ponto de observação; distribuição dos fluxos em função do tempo; e volume de tráfego enviado e recebido por cada *host*;
- gerente: sua função é configurar o medidor e controlar o leitor de medição. Para isso, ele envia comandos de configuração aos medidores, e supervisiona a operação de cada medidor e de cada leitor. Em caso de conveniência, é possível combinar as funções do leitor de medição e do gerente em uma única entidade de rede.

Um coletor pode recuperar os dados de fluxo de vários agentes, e cada agente pode ter seus dados recuperados por vários coletores. As informações de interesse a serem coletadas são definidas na configuração enviada pelo gerente aos medidores, na forma de regras (*Rule Sets*), como mostra a Figura 2 .

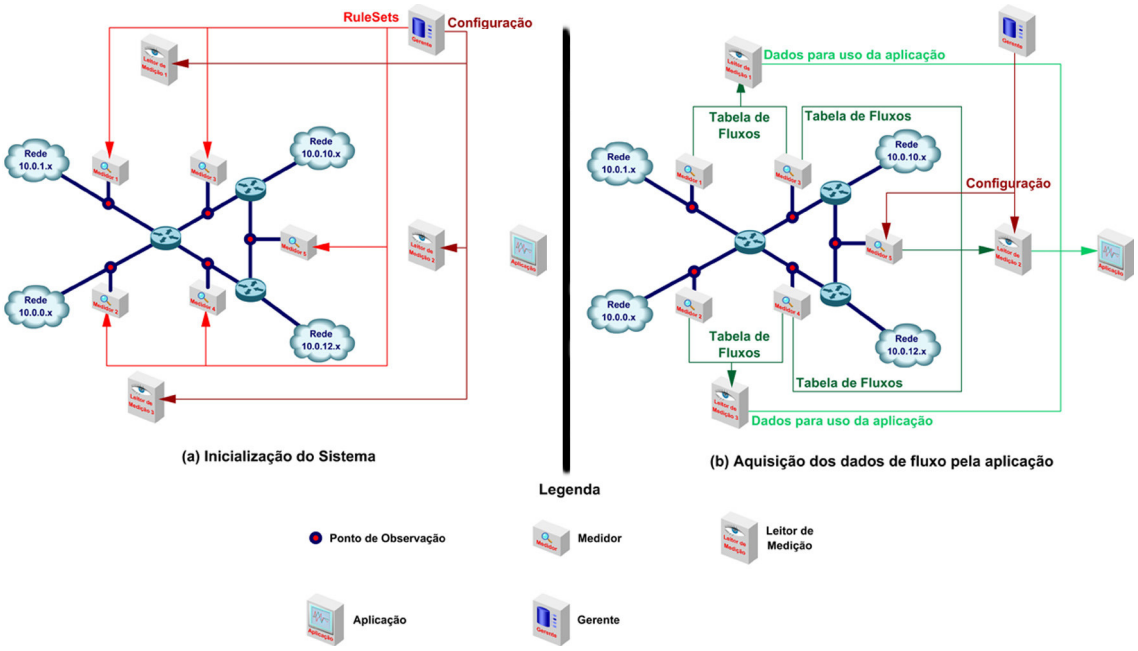


Figura 2 Arquitetura proposta pelo RTFM.

De acordo com Brownlee [BRO 97c], as *Rule Sets* têm por objetivo otimizar o processo executado pelo medidor. Através delas são determinados quais os fluxos de interesse, ignorando, assim, os pacotes que não pertencem a estes fluxos e, consequentemente, diminuindo o consumo de memória do medidor e a quantidade de informações que ele deve transmitir ao leitor de medição.

Na Figura 2 , é visto ainda o processo de comunicação do RTFM. Primeiramente, como pode ser visto em (a), o gerente envia a configuração para os medidores e os leitores de medição, sendo que nesse momento os medidores recebem as *Rule Sets*. A partir dessa configuração o sistema está apto a iniciar o processo de medição propriamente dito (b), onde os leitores de medição recebem as tabelas de fluxo dos medidores e repassam esses dados para a aplicação, que trata as informações coletadas e as provê ao usuário do sistema. Eventualmente, o gerente pode ainda modificar a configuração de medidores ou leitores de medição como mostrado em (b).

A RFC 2720 [BRO 99a] define uma MIB específica para o controle dos medidores de tráfego, obtendo-se desta forma um mecanismo eficiente e padronizado para coleta de informações dos fluxos, permitindo o uso do SNMP no transporte das informações de fluxo coletadas.

Apesar de todos os esforços do RTFM, as especificações propostas pelo grupo não foram efetivamente incorporadas pelo mercado, fazendo com que sua arquitetura de medição, ao invés de ser utilizada para padronizar o mercado, se tornasse apenas mais uma alternativa para análise de fluxos, sendo utilizada especialmente no contexto científico-acadêmico. Dentre as propostas em concordância com as especificações do RTFM destaca-se o NeTraMet/NeMaC, tratado com maiores detalhes na subseção 2.9.2.

Segundo Molina [MOL 03], essa pouca aceitação ao RTFM foi causada pela complexidade na implementação dessa arquitetura. Comparando-a com a proposta do IPFIX, que será detalhada no capítulo 3 , realmente percebe-se uma maior complexidade e rigidez nos elementos formadores da arquitetura. O RTFM, por necessitar de um elemento gerente que controla a operação dos medidores e dos leitores de medição, acaba por aumentar o tráfego de controle concorrente na rede. Além disso, esse gerenciamento mais elaborado acaba adicionando complexidade aos próprios elementos que formam a arquitetura, incluindo os medidores. Esse aspecto é fundamental no caso de incorporar a aquisição de dados de fluxo às funcionalidades de um roteador. Nesse contexto, seria necessário que a complexidade adicional incorporada às tarefas do roteador fosse mínima.

Outro fator a ser analisado é que, na época do desenvolvimento do RTFM, já existiam uma série de sistemas de análise de fluxo com padronização própria no mercado, sendo efetivamente utilizados por diversas empresas. Para essas empresas, a adequação as especificações propostas pelo RTFM foi inviabilizada pelo alto custo envolvido no processo. Neste contexto, destaca-se o grande uso de equipamentos de rede que já possuíam nativamente o protocolo *NetFlow* [CLA 04], desenvolvido pela Cisco para a exportação dos dados.

2.8 USO DE TÉCNICAS DE AMOSTRAGEM NO MONITORAMENTO DE TRÁFEGO

Conforme apresentado anteriormente, o uso da coleta de informações de fluxo ao invés da análise de pacotes, ou mesmo dos cabeçalhos destes, permitiu uma grande diminuição no tamanho dos arquivos com as informações coletadas, simplificando sua manutenção,

transporte e processamento. Entretanto, o grande aumento na capacidade de tráfego das redes, dado a partir do desenvolvimento de redes de alta velocidade (redes Gigabit), criou a necessidade de desenvolver outros métodos que auxiliem no processo de medição de tráfego.

Esse processo invariavelmente levou a uma investigação sobre o uso de técnicas de amostragem como meio de aferir as condições de redes de alta capacidade. Um dos primeiros trabalhos nessa linha de pesquisa foi realizado por Claffy [CLA 93]. Mais precisamente, esse trabalho propõe uma investigação sobre os efeitos da implementação de mecanismos de amostragem na análise da rede em ambientes operacionais. Guimarães [GUI 07] faz uma análise aprofundada sobre as técnicas de amostragem na aquisição de informações de fluxo, focando seu trabalho especialmente na implementação e avaliação da técnica de amostragem aleatória estratificada, proposta por Choi *et al*.([CHO 04] [CHO 06]).

A maturação da pesquisa nesse sentido levou o IETF a criar o grupo de trabalho *Packet Sampling* (PSAMP), que visa padronizar o processo de coleta de pacotes amostrais a partir de métodos estatísticos. O PSAMP é detalhado a seguir.

2.8.1 *Packet Sampling* (PSAMP)

O IETF criou o grupo de trabalho PSAMP visando definir um conjunto padrão de recursos para capacitar elementos da rede a realizar a amostragem, a partir de métodos pré-definidos. Especificamente, é objetivo do grupo a padronização dos seguintes mecanismos:

- seletores para amostragem de pacotes, que definem o conjunto de primitivas para a operação de seleção de pacotes em elementos de rede;
- informações sobre os pacotes, que especificam a extensão do pacote que deverá ser disponibilizada para relatório;
- relatórios dos pacotes amostrados, que definem o formato dos relatórios que deverão ser construídos para cada pacote amostrado;
- *report streams*, que definem o formato para relatórios de uma rajada de pacotes;

- múltiplos *report streams*, que especificam as exigências para múltiplos seletores paralelos de pacotes em um único elemento de rede;
- configuração e gerenciamento, que especifica uma MIB para o seletor de pacotes, sendo esta alocada no elemento de rede;
- apresentação, exportação e transporte dos relatórios de pacotes, que caracteriza a interface para apresentação dos relatórios para as aplicações.

No atual estágio de maturação das propostas do PSAMP, estas ainda se encontram em nível de *draft*. Não existindo, até o momento, nenhuma RFC desenvolvida pelo grupo.

2.9 PRINCIPAIS FERRAMENTAS DE MONITORAMENTO DE TRÁFEGO BASEADO EM FLUXOS

Esta seção apresenta as principais ferramentas desenvolvidas para efetuar o monitoramento de tráfego baseado em fluxos.

2.9.1 nTop

Originado do utilitário de linha de comando *top*, presente em vários ambientes *Unix*, cuja função é mostrar os processos em execução no sistema, além do consumo de memória e CPU de cada um. O nTop [DER 00] segue o mesmo conceito, porém, ao invés de processos, identifica conexões de rede e o uso de recursos do barramento de rede. Com interface baseada em *web*, além da tradicional pelo console, permite ao usuário visualizar:

- indicadores de tráfego;
- monitoramento do tráfego;
- planejamento e otimização da rede;
- detecção de violações na segurança.

O nTop foi construído usando a biblioteca libpcap [TCP 07], a mesma usada no tcpdump. Isto permitiu que houvesse versões para quaisquer plataformas que possuíssem compatibilidade com a libpcap. Basicamente ele age como um *sniffer*, recolhendo todos os dados presentes no barramento da rede, classificando-os em fluxos, tratando suas informações e gerando gráficos e tabelas. Os princípios de funcionamento da libpcap são aprofundados na subseção 4.1.1.

Além desta interface para captura de pacotes, o nTop possui uma base de dados que armazena características da rede. Se por exemplo, um *host* A mandar pacotes para o *host* C, através do *host* B, ele automaticamente entende que B é um roteador ou *gateway*.

Segundo Vieira [VIE 04], entre suas deficiências estão a falta de, nativamente, usar uma base de dados para armazenar as informações coletadas, visto que os dados são perdidos após o fechamento da aplicação e a falta de uma arquitetura baseada em gerenciamento SNMP, pois os papéis de agente e gerente são exercidos conjuntamente pelo único módulo do programa, impossibilitando o uso de agentes distribuídos em diferentes segmentos da rede.

O nTop é uma ferramenta *open-source*. A Figura 3 mostra a sua interface.

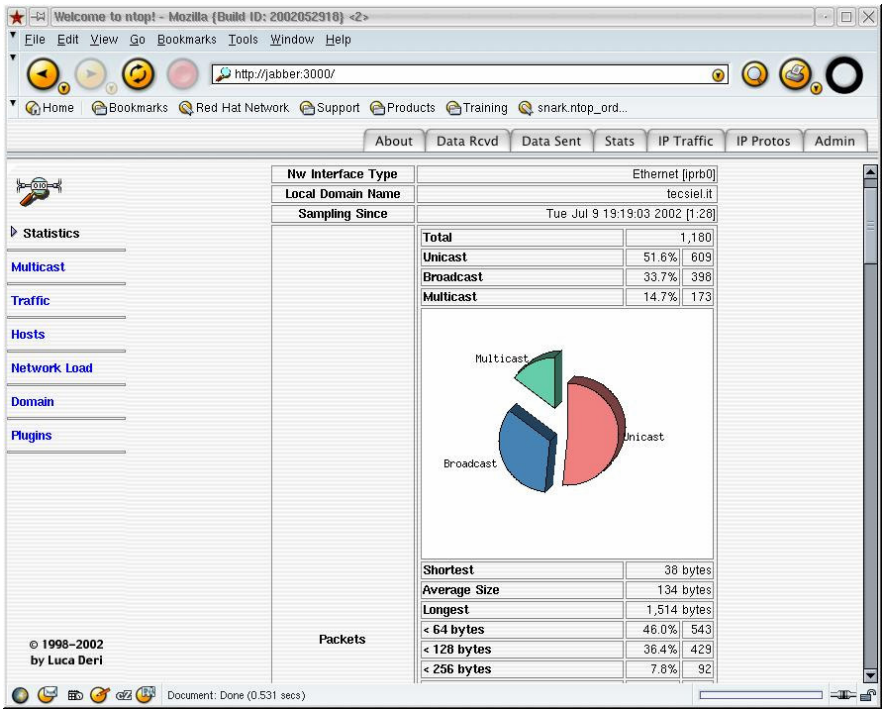


Figura 3 Interface do nTop

2.9.2 NeTraMet/NeMaC

O NeTraMet/NeMaC (*Network Traffic Meter & Network Manager Collector*) [BRO 97c] é uma implementação baseada no modelo RTFM, proposto pelo IETF para medições por fluxo de tráfego, detalhado na seção 2.7. Trata-se de uma implementação *open-source*, que tem a vantagem de permitir ao usuário a possibilidade de definir regras que identifiquem fluxos de dados através da programação de *scripts*, permitindo uma maior flexibilização e dinamismo no trabalho de quantificar e qualificar os tipos de fluxos. O NetraMet/NeMaC pode ser implantado para exercer as funções de medidor, coletor e gerente.

Esta arquitetura foi baseada inicialmente na RFC 1272 [MIL 91], e padronizada na RFC 2722 [BRO 99c] tendo três componentes básicos:

- medidores, *hosts* colocados em um segmento de rede para medição do fluxo de tráfego nesse segmento;
- leitores de medição (coletores), responsáveis pela recuperação das informações de medidores em um segmento de rede;
- gerentes, que instruem os medidores sobre qual tráfego medir e os coletores de quais medidores devem coletar, e a que intervalos de tempo.

A Figura 4 mostra a arquitetura do NeTraMet/NeMaC. Nela, o agente medidor, implementado no subsistema NeTraMet (*Network Traffic Meter*), armazena os dados de fluxo mensurados em memória, utilizando o protocolo SNMP para compartilhar as medidas com o coletor, implementado no subsistema NeMaC (*Network Manager Collector*).

O subsistema NeMaC possui também a funcionalidade de gerente. Assim, através da combinação das funcionalidades de gerente e coletor em um único dispositivo, é possível ao NeMaC realizar a etapa de configuração e gerenciamento de um número arbitrário de medidores NeTraMet, onde cada medidor usa suas regras pré-estabelecidas e a etapa de coleta dos dados de fluxo especificados. Essa simplificação do sistema é permitida na RFC 2722 [BRO 99c] do RTFM.

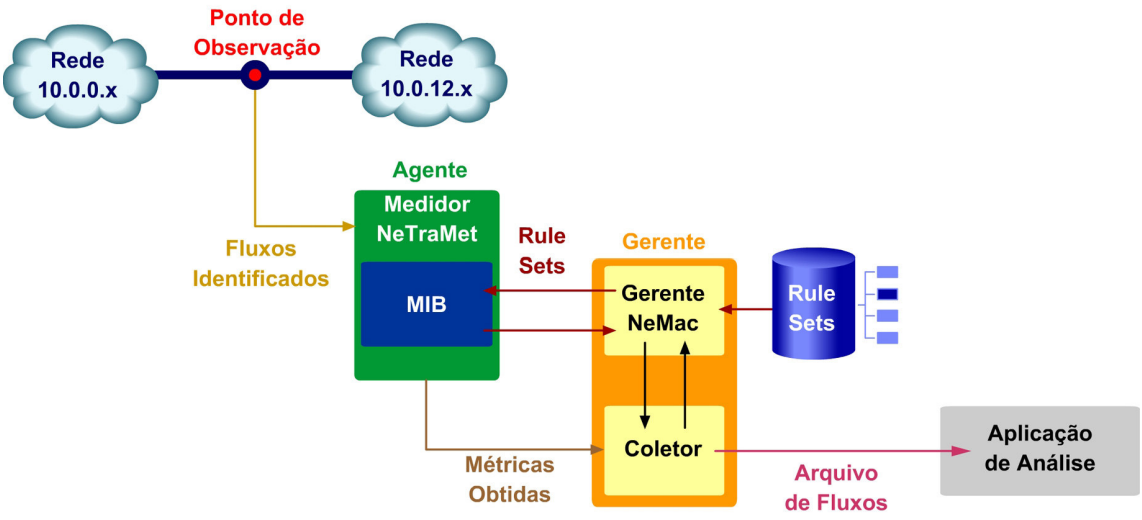


Figura 4 Arquitetura NeTraMet/NeMaC.

Através da especificação de regras MIB do gerente NeMaC ao agente NeTraMet, é possível ainda reduzir o volume de dados armazenados em memória nesse agente e, posteriormente, repassados ao próprio NeMaC, aumentando a eficiência de todo sistema.

O NeTraMet pode ser utilizado em diferentes plataformas, suportando sistemas operacionais *Unix* (*Solaris*, *SunOS*, *Irix* ou *Linux*), usando a biblioteca *libpcap* [TCP 07] para observar os cabeçalhos dos pacotes *Ethernet*, ou sistemas *Microsoft Windows* (WIN32) e *Microsoft* MSDOS usando o *Packet Driver* CRYNWYR (*driver* de rede para sistemas MSDOS). O NeMaC possui suporte apenas para sistemas operacionais *Unix*.

2.9.3 sFlow

O sFlow é um sistema de monitoramento de tráfego em redes de dados capaz de utilizar a tecnologia de amostragem para capturar as estatísticas de tráfego amostrado, sendo definido em [PHA 01]. Seu desenvolvimento visa o uso em redes de alta velocidade (redes Gigabit) e redes com grande número de agentes de medição necessitando, em ambos os casos, da amostragem para a sumarização dos dados a serem tratados pelo sistema.

Sua arquitetura consiste em agentes sFlow, que podem estar embarcados em *switches* ou roteadores e um coletor de dados central, chamado de Analisador sFlow. A Figura 5 mostra esses elementos.

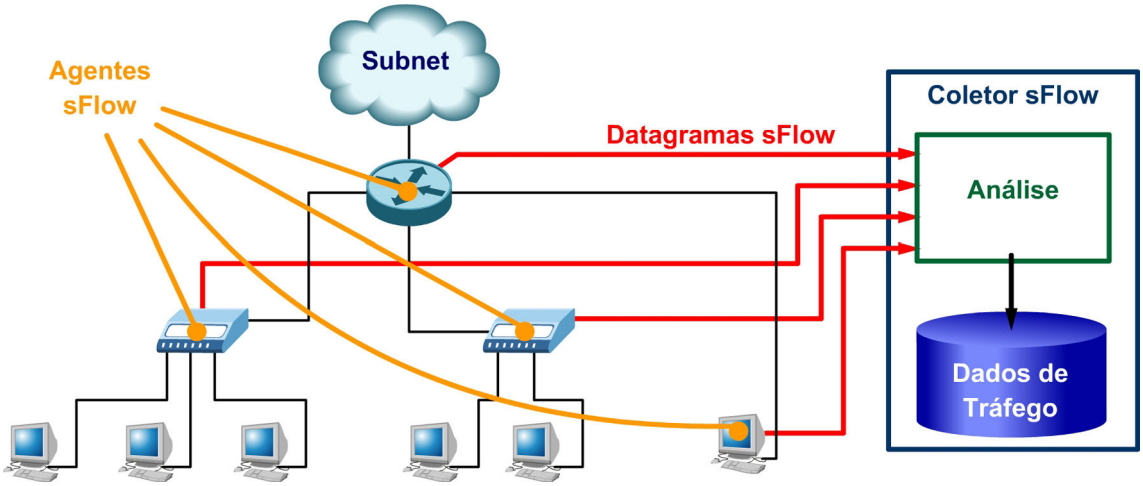


Figura 5 Arquitetura sFlow.

É função do agente sFlow o uso da amostragem na captura dos pacotes e o envio imediato dos dados estatísticos amostrados para um analisador sFlow. O agente é capaz de realizar duas formas de amostragem: amostragem baseada na contabilização dos pacotes, para verificação dos fluxos, e amostragem baseada no tempo, para análise das interfaces de rede. No *sFlow* um fluxo é definido como todos pacotes que são recebidos em uma interface do *switch* ou roteador e são enviados para outra interface.

Para permitir o controle e a configuração remota do agente sFlow foi definida uma MIB (*sFlow* MIB), podendo assim utilizar o protocolo SNMP no transporte dessa configuração. Já para o envio dos datagramas com os dados amostrados do agente para o coletor é utilizado o protocolo *User Datagram Protocol* (UDP) [POS 80], com a configuração da porta sendo especificada na MIB sFlow, de modo a reduzir o uso da memória do agente.

2.9.4 NetFlow

O NetFlow é um protocolo aberto, desenvolvido pela Cisco para coletar a informação de tráfego IP que passa pelos equipamentos desenvolvidos por esta empresa. Roteadores e *switches* que possuem este protocolo implementado podem enviar os registros de fluxo gerados até um coletor NetFlow, responsável pelo armazenamento e análise dos pacotes de informações de fluxo.

O NetFlow define um fluxo IP como uma seqüência de pacotes com um grupo de atributos em comum. De acordo com [CIS 06b], esses atributos são definidos pela combinação dos seguintes campos, denominados campos-chave:

- endereço IP da origem;
- endereço IP do destino;
- número da porta na origem;
- número da porta no destino;
- tipo de protocolo na camada 3;
- *type of service* (ToS), ou Tipo de Serviço;
- entrada na interface lógica.

Todos os pacotes com mesmos campos-chave são agrupados em um único fluxo, contabilizando para este fluxo o número de pacotes e o total de bytes. Todas essas informações são então agrupadas e armazenadas em uma base de dados chamada *NetFlow Cache*. Adicionalmente, uma série de informações pode ser obtida com o NetFlow, incluindo:

- marcações de tempo para o fluxo, permitindo visualizar seu comportamento no decorrer do tempo;
- *Border Gateway Protocol* (BGP) *next hop*;
- máscaras da sub-rede tanto para origem quanto para o destino;
- *flags* TCP para examinar a sessão TCP.

A partir da aquisição dessas informações, o dispositivo de rede usa um mecanismo, chamado *NetFlow Exporter*, para enviar todas as informações ao coletor NetFlow. Para isso, o

NetFlow Exporter faz uso de *templates*, elementos que especificam a estrutura e semântica do grupo de informações que precisam ser enviadas ao coletor.

De acordo com Sadasivan [SAD 06], o uso de *templates* na modelagem do protocolo tem as seguintes vantagens:

- novos campos podem ser adicionados ao registro de fluxo sem que se modifique a estrutura dos registros, fazendo com que novos atributos possam ser incorporados ao protocolo mais rapidamente;
- os *templates* carregam toda a informação estrutural sobre os registros de fluxo exportado. Assim, se o coletor não entende a semântica de novos campos ele pode simplesmente ignorá-los e manter a capacidade de interpretar o registro de fluxo;
- dada a flexibilidade do mecanismo, é possível exportar para o coletor apenas as informações de interesse sobre o fluxo e o processo de medição, auxiliando na redução do volume de dados exportados e possibilitando a diminuição no consumo de memória tanto no dispositivo de aquisição de dados quanto no coletor.

Adicionalmente, [CIS 04] destaca que, a partir do uso de *templates*, torna-se possível o desenvolvimento de protocolos com os mais diversos fins, onde a fácil expansibilidade e adaptabilidade sejam desejáveis.

Efetivamente, por suas qualidades e por ter sido desenvolvido pela Cisco, o NetFlow possui boa aceitação no mercado. Entretanto, o fato de ser exclusivamente adotada por equipamentos Cisco dificulta o seu uso para o gerenciamento total de uma rede.

3 *IP FLOW INFORMATION EXPORT* (IPFIX)

Numa nova tentativa de padronizar a análise de tráfegos através do fluxo, o IETF formou o grupo de trabalho *IP Flow Information Export* (IPFIX), que possui como meta a padronização dos resultados obtidos no medidor, visando a exportação destes indicadores para o coletor. O protocolo foi desenvolvido para o transporte de dados em geral, buscando ser facilmente extensível de modo a adaptar-se às necessidades de diferentes aplicações. Cabe aqui ressaltar que, no atual estágio de maturação das propostas definidas pelo grupo de trabalho IPFIX, a maior parte de sua documentação ainda está em estágio de análise, na forma de *drafts* do IETF. Desse modo, o grupo possui, até o presente momento, apenas duas RFCs já efetivamente elaboradas:

- RFC 3917: *Requirements for IP Flow Information Export* [QUI 04]*;*
- RFC 3955: *Evaluation of Candidate Protocols for IP Flow Information Export (IPFIX)* [LEI 04].

Para definição do IPFIX, primeiramente foram analisados os requisitos básicos que o protocolo de comunicação deveria suprir, a partir da verificação das principais aplicações que necessitam da análise de tráfego através de fluxo. Esse processo é descrito na RFC 3917 [QUI 04], sendo que esses requisitos são usados como base para o desenvolvimento do protocolo IPFIX.

3.1 ANÁLISE DE REQUISITOS DO IPFIX

Cada uma das seguintes subseções apresenta os requisitos das principais aplicações que utilizam as informações oriundas da análise dos dados de fluxo. Esses requisitos baseiam o desenvolvimento do IPFIX.

3.1.1 Contabilização dos dados que formam o fluxo visando sua tarifação

O IPFIX deve prover medição de baixa granularidade, resultando em uma contabilização flexível e detalhada do uso de recursos. Diversos modelos de negócios para venda de serviços baseados na rede IP estão, atualmente, em processo de análise. Esses serviços podem ter sua tarifação baseada no seu uso em função do tempo ou do volume de dados utilizado. A contabilização do fluxo de dados por meio do IPFIX pode servir como entrada para sistemas de bilhetagem, contabilizando os fluxos para cada usuário ou para grupos de usuário, tanto para bilhetagem de acordo com os serviços prestados quanto para contabilização baseada no tipo de conteúdo que é trafegado.

Essa metodologia de tarifação permite também variações na bilhetagem como, por exemplo, a variação na contabilização de acordo com a aplicação utilizada, o período do dia em que o serviço foi prestado, os caminhos utilizados na rede, etc.

A tarifação pode ser baseada, por exemplo, em fluxos fim-a-fim individuais, sendo que, nesse caso, a contabilização é determinada por uma quíntupla consistindo do endereço de origem, endereço de destino, protocolo, e portas origem e destino. Outro exemplo é a tarifação dependente de classe de serviço, existente em redes *DiffServ* [BLA 98]. Nesse caso, os serviços podem ser distinguidos de acordo com o *DiffServ codepoint* (DSCP) e endereços IP de origem e de destino. Os elementos essenciais de informação para a contabilização do fluxo geralmente são o número de pacotes transferidos e o número de *bytes* por fluxo. O IPFIX foi definido com flexibilidade capaz de permitir a escolha da definição de fluxo de acordo com o modelo de tarifação desejado [QUI 06].

3.1.2 Definição do perfil de tráfego

A definição do perfil de tráfego é o processo de caracterização dos fluxos IP pelo uso de um modelo que represente os parâmetros chave dos fluxos como, por exemplo, sua duração, seu volume e o período de suas rajadas [QUI 04]. Trata-se de um pré-requisito no

planejamento e dimensionamento de redes, análise de riscos, desenvolvimento de modelos de negócios, entre outros. Informações típicas obtidas com a aquisição por esse processo são: a distribuição de uso de serviços e protocolos na rede, o montante de pacotes de um tipo específico e a identificação de perfis de tráfego típicos.

Resultados de medições exportados através do IPFIX podem ser usados para definição do perfil de tráfego. Medições capturadas em um longo período de tempo podem ser usadas para traçar e predizer o crescimento do uso de uma rede, informações necessárias no planejamento de mesma.

Os parâmetros de interesse são determinados pelos objetivos definidos no perfil de tráfego. A distribuição dos serviços e protocolos em uso pode ser analisada pela configuração dos parâmetros dos fluxos apropriados para sua caracterização.

A duração do fluxo pode ser calculada conforme definido em [QUI 06]. Além disso, o número de pacotes e o número de bytes de um fluxo devem ser representados em um contador para cada fluxo.

3.1.3 Engenharia de tráfego

Conforme visto na seção 2.4, a engenharia de tráfego busca a otimização da utilização dos recursos da rede e do desempenho do tráfego passante por essa rede. Parâmetros típicos usados nesse estudo são: o grau de utilização de um link, a carga existente em um local específico da rede, o número de nodos em um determinado caminho, o número de pontos de entrada e saída de fluxos ativos e informações de roteamento [QUI 04]. Todas essas estimativas podem ser obtidas através de processos de medição e exportadas através do IPFIX.

3.1.4 Segurança da rede

O IPFIX deve exportar informações de fluxo a partir de definições de fluxo arbitrárias. Assim, informações tanto sobre o desempenho do fluxo, como, por exemplo, o número de bytes que formam esse fluxo, quanto informações sobre os próprios campos dos pacotes que formam esse fluxo, como, por exemplo, os endereços de origem e destino desse fluxo, podem ser exportadas através do IPFIX. Com isso, diversas informações sobre o tráfego na rede tanto na camada IP quanto nas camadas superiores são, teoricamente, acessíveis. Estes dados podem ser definidos de modo a permitir a detecção de anomalias provocadas por ataques ou prover as bases para um processamento que gere métricas de detecção de ataque mais complexas.

Dependendo do tipo de ataque, diferentes técnicas podem ser úteis. Um repentino incremento no tráfego pode ser o sinal de que um ataque está sendo iniciado. Um incremento repentino no tráfego oriundo de diferentes origens para um único destino pode ser causado por um ataque a um *host* ou a um elemento da rede específico, através do uso de endereços IP falsos. Muitos fluxos de uma mesma máquina para diferentes portas ou muitos fluxos para diferentes máquinas, porém em uma mesma porta, podem indicar a atividade de *scanning* de portas horizontal ou vertical. Uma taxa atípica de TCP-SYN para TCP-FIN pode se referir a um ataque de *SYN-flooding*. Até mesmo *worms* podem ser identificados, se for identificado o seu padrão de tráfego [ZSE 06].

Incidentes de segurança podem se tornar uma ameaça até mesmo para o próprio processo IPFIX. Se um ataque gera uma grande quantidade de fluxos, por exemplo pelo envio de pacotes com endereços falsos, ou através da simulação de terminações no fluxo, o processo de coleta e exportação pode ser sobrecarregado pela grande quantidade de dados a serem exportados. Para prevenir isso, se faz necessário o desenvolvimento de métodos de amostragem para prevenir a exaustão de recursos, conforme tratado em [ZSE 06].

3.1.5 Monitoramento de QoS

Conforme visto na seção 2.1, monitoramento de QoS é a observação da qualidade de transmissão para fluxos individuais ou para o tráfego agregado a rede. Um exemplo de seu uso é a validação do QoS negociado em um SLS. A seção 2.5 mostrou que parâmetros típicos de QoS como a perda de pacotes, OWD , *round-trip delay* e *jitter* não são obtidos de forma trivial através do uso da medição passiva. O cálculo dessas métricas exige um processamento analisando cada pacote individualmente e o uso de, ao menos, dois pontos de observação na rede. Essas informações podem ser obtidas através do IPFIX, desde que cada pacote seja considerado como um fluxo individual. Além disso, essa necessidade de correlacionar as informações obtidas entre os diversos pontos de observação, torna necessária a sincronização dos relógios de cada um dos processos envolvidos na aquisição dos dados.

3.1.6 Compartilhamento de dados IPFIX entre domínios

Dados IPFIX podem ser trocados entre domínios autoritativos distintos. Diversas recomendações precisam ser consideradas se dados IPFIX atravessam um domínio publico, quando comparado com o uso do mesmo em um único domínio. Nesses casos, de acordo com [ZSE 06] é necessário o uso do *IPSec* [KEN 98] ou do *Transport Layer Security* (TLS) [DIE 99].

3.1.7 Exportação de métricas derivadas

O IPFIX normalmente é utilizado para transportar informações sobre fluxos e pacotes, provendo a entrada para o cálculo de uma variedade de métricas. Pode, entretanto, também ser usado para transferir essas métricas diretamente.

A partir das necessidades levantadas, a RFC 3955 [LEI 04] apresenta a análise de cinco protocolos candidatos a servirem como base no desenvolvimento do IPFIX:

- *Crane* [ZAN 02];
- *Diameter* [CAL 03];
- LFAP [AMS 97];
- *NetFlow* v9 [CLA 04];
- *Streaming* IPDR [MEY 02].

Destes protocolos, o *NetFlow*, descrito na subseção 2.9.4, foi definido como o protocolo base para o desenvolvimento do IPFIX. Apesar de todo esse avanço, ainda existem muitas adequações que devem ser realizadas de modo a tornar o *NetFlow* v9 totalmente em conformidade com as especificações definidas em [QUI 04]. Entre as principais diferenças, Boschi [BOS 06] destaca:

- o *NetFlow* 9 foi projetado para utilizar o protocolo UDP na camada de transporte, enquanto que o IPFIX necessita garantir a entrega de seus dados. Dessa forma, é especificado o *Stream Control Transmission Protocol* (SCTP) [STE 00] como seu protocolo padrão, enquanto que o *Transmission Control Protocol* (TCP) [DAR 81] e o UDP são opcionais;
- existem diferenças entre os elementos que transportam as informações sobre os fluxos (elementos de informação) padronizados pelo IETF e os elementos não definidos pelo IETF, como os do *NetFlow*. Estes precisam ser enviados juntamente com um identificador que defina o protocolo do qual ele é originado;
- existem campos diferentes entre os cabeçalhos do *NetFlow* e do IPFIX e, em alguns casos, campos que são usados em ambos os protocolos mas que podem assumir valores distintos;
- *templates* de opções de controle, detalhados posteriormente neste trabalho, possuem especificações distintas entre os protocolos;
- o campo tamanho do pacote é tratado de forma distinta entre os dois protocolos;
- o *NetFlow* possui um indicador de *timestamp* extra em seu cabeçalho, inexistente no IPFIX, que indica qual foi a última reinicialização que o processo de exportação dos dados sofreu.

Para identificar as diferenças entre os protocolos, mantendo uma compatibilidade entre os mesmos, o número de versão existente no cabeçalho dos pacotes de ambos é distinto: enquanto que o *NetFlow* usa número de versão 9, o IPFIX recebe número de versão 10.

Apesar das diferenças existentes, devido a solução adotada pelo IPFIX de manter a compatibilidade entre o seu protocolo e o *NetFlow*, é possível o desenvolvimento de ferramentas de coleta capazes de receber os pacotes IPFIX e manter a interoperabilidade com uma série de dispositivos já existentes no mercado, como roteadores e *switches*, que já utilizam o *NetFlow* para exportação de dados. Sadasivan [SAD 06] descreve a arquitetura do IPFIX, detalhada na seção seguinte.

3.2 ARQUITETURA DO IPFIX

A arquitetura IPFIX é formada por dois elementos básicos: o dispositivo IPFIX e o coletor. É considerado um dispositivo IPFIX todo aquele elemento que realiza pelo menos um processo de exportação de dados através do protocolo IPFIX. Esse processo se baseia no envio, em direção a um ou mais coletores, de um registro de fluxo, gerado a partir de um ou mais processos de medição. O processo de medição consiste na captura do cabeçalho de cada pacote que passam pelo seu domínio de observação, a marcação do seu tempo (*timestamping*), a amostragem e a classificação desse elemento, visando a manutenção dos registros de fluxo.

O domínio de observação é um conjunto de pontos de observação (locais da rede onde são observados os pacotes IP para determinação dos fluxos) para os quais as informações de fluxo podem ser agregadas por um processo de medição. Cada domínio de observação possui um identificador exclusivo ao processo de coleta, usado para identificar as mensagens IPFIX geradas. Por exemplo, um dispositivo IPFIX pode ser um roteador capaz de enviar suas informações de fluxo através do protocolo IPFIX, onde seu domínio de observação é sua placa controladora e os pontos de observação são cada uma das suas interfaces de rede. Todo ponto de observação é associado a um domínio de observação.

Por sua vez, coletor é todo dispositivo que recebe registros de fluxo de um ou mais dispositivos IPFIX. Esse elemento pode processar os registros de fluxo ou armazená-los, entretanto essas ações não fazem parte do escopo do IPFIX. A Figura 6 mostra cada um desses elementos, detalhados nas seções seguintes.

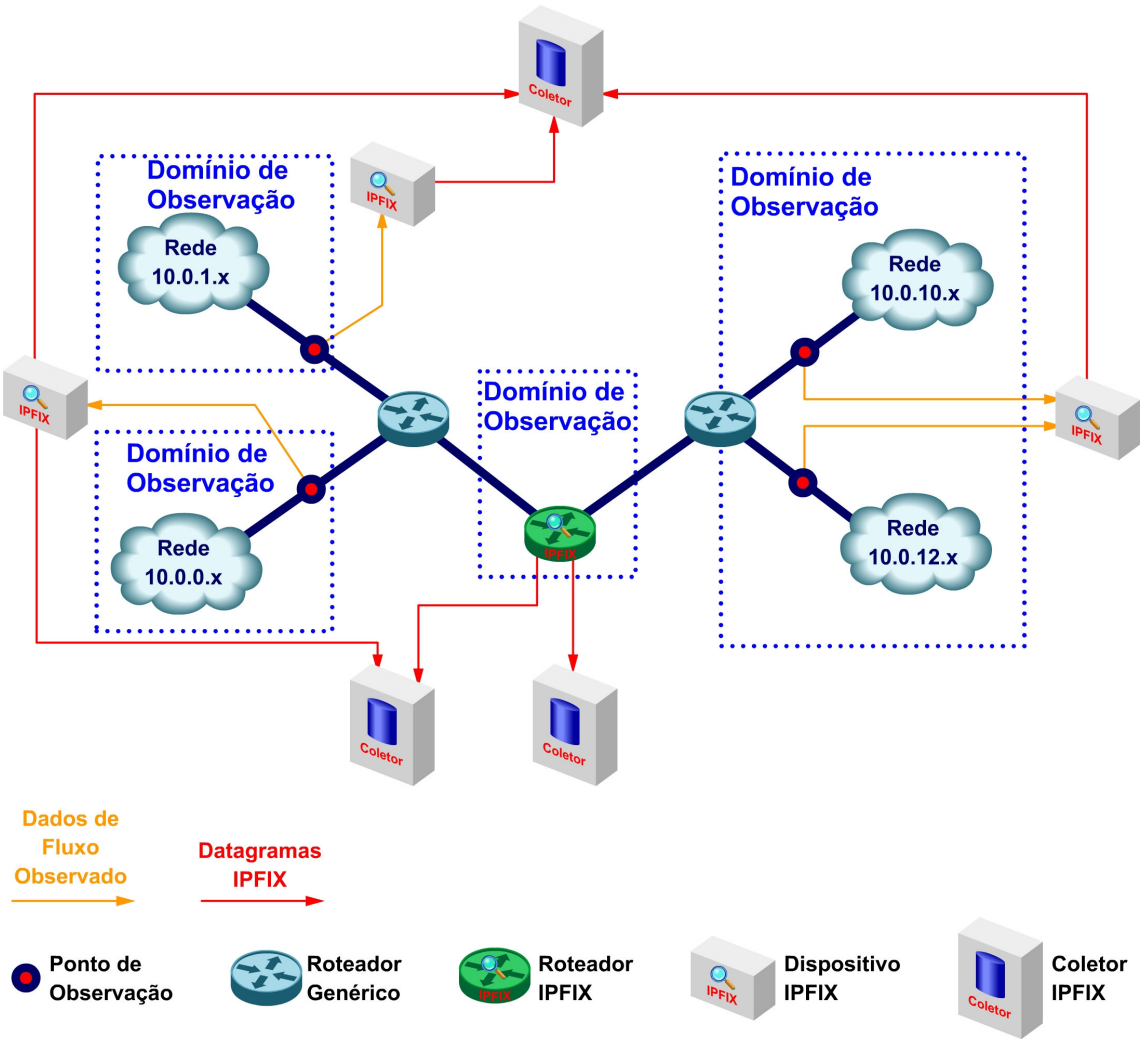


Figura 6 Arquitetura IPFIX.

3.3 DISPOSITIVO IPFIX

O dispositivo IPFIX possui as seguintes atribuições:

- codificar as informações de controle em *templates*;
- extrair as informações de fluxo a partir dos pacotes observados nos pontos de observação;
- empacotar os *templates* selecionados e os registros de fluxo em mensagens IPFIX;
- enviar mensagens IPFIX ao coletor.

Informações de controle incluem a definição do fluxo e critérios de seleção dos pacotes que formam esse fluxo, estatísticas sobre o processo de medição e detalhes sobre os campos que formam os registros de fluxo. As informações de controle transportam toda a informação necessária para os coletores interpretarem os pacotes IPFIX enviados. Conforme visto, o dispositivo IPFIX precisa codificar essas informações em *templates*.

A Figura 7 mostra o diagrama funcional do dispositivo IPFIX. Cada um dos blocos lógicos que forma este dispositivo é detalhado em seguida.

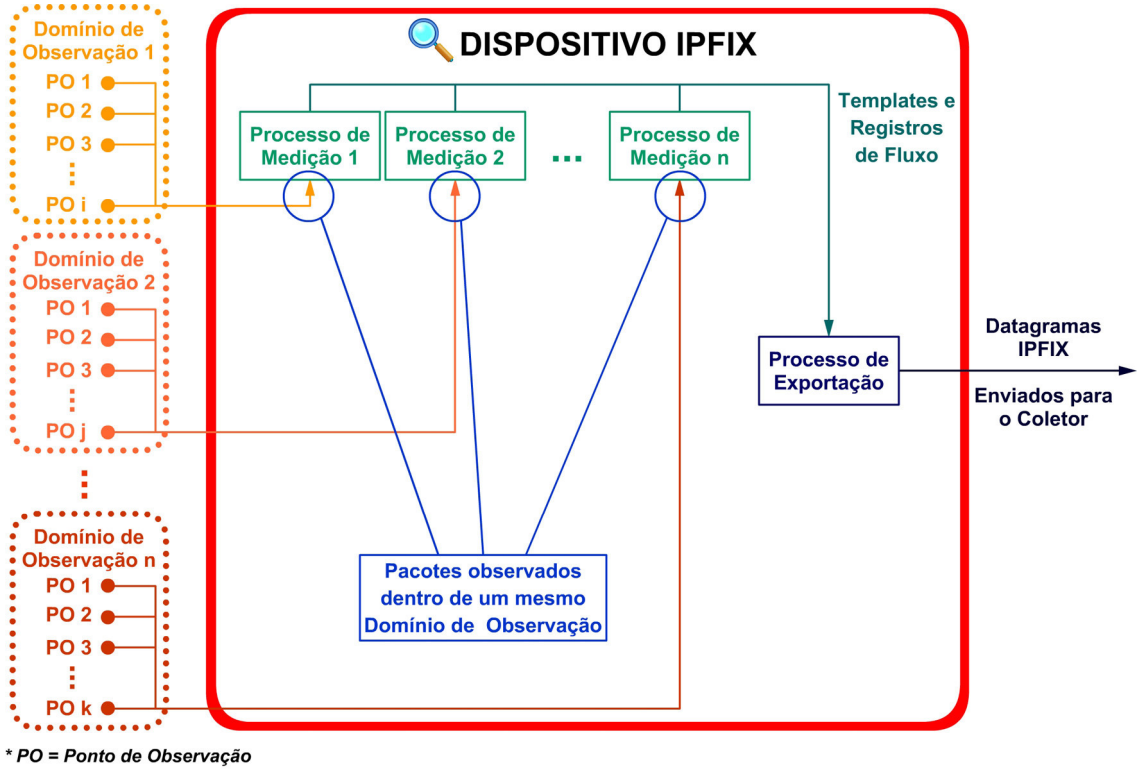


Figura 7 Diagrama em blocos do dispositivo IPFIX.

3.3.1 Processo de medição

Cada pacote que chega a um domínio de observação tem seu cabeçalho replicado e enviado para cada um dos processos de medição associados àquele domínio.

A função do processo de medição é identificar o tempo de chegada do pacote observado e classificá-lo em um fluxo, baseado nos critérios de seleção configurados. O IPFIX não estabelece o meio como o dispositivo IPFIX é configurado, deixando isso a cargo da forma de implementação do dispositivo.

Por ser o bloco funcional que gerencia todo o fluxo gerado a partir de um domínio de observação, são funções do processo de medição:

- a manutenção de todos os registros de fluxo oriundos do domínio de observação associado, incluindo aí a criação de novos registros de fluxo, atualização dos registros já existentes, armazenamento das estatísticas dos fluxos, derivação de propriedades dos fluxos e adição de informações não específicas a estes, como por exemplo, o domínio autoritativo onde o fluxo foi identificado;
- a manutenção dos dados estatísticos acerca do processo de medição em si, como o total de registros de fluxo gerados, total de pacotes observados, etc.

Além dessas tarefas básicas, o processo de medição pode, adicionalmente, utilizar critérios de seleção de pacotes, visando otimizar todo o processo de aquisição de dados de fluxo. Nesse caso, o processo de medição deve definir determinadas regras, permitindo que apenas pacotes de interesse que passam em um ponto de observação sejam escolhidos para medição. Isto pode ser feito utilizando-se os seguintes métodos:

- uso de uma função de amostragem, de modo a determinar quais pacotes serão selecionados para uma medição;
- uso de filtragem, selecionando somente aqueles pacotes que satisfazem um determinado critério. Esse critério é definido como função dos campos que formam o cabeçalho do pacote, campos obtidos durante o processamento dele, ou mesmo, propriedades do próprio pacote.

A seleção dos pacotes é realizada por qualquer um dos dois métodos definidos acima ou por uma combinação deles em qualquer ordem.

Dentre as tarefas relacionadas ao processo de medição, é importante que esse módulo possua a capacidade de identificar a expiração de um fluxo. De acordo com Sadasivan [SAD 06] o fluxo é considerado expirado nas seguintes condições:

- se nenhum pacote pertencente ao fluxo for observado por um determinado período de tempo. Este período de tempo deve ser configurável no processo de medição, com um valor mínimo zero para expiração imediata. Esse caso é uma alternativa para indicar a análise considerando cada pacote como um fluxo único;
- se o dispositivo IPFIX passa por falta de recursos como, por exemplo, falta de memória para armazenamento dos registros de fluxo, um fluxo pode ser prematuramente expirado;
- para fluxos de longa duração, o processo de medição deve expirar o fluxo em um período regular de tempo ou baseado em alguma política de expiração. Esta periodicidade ou política de expiração deve ser configurável no processo de medição. Quando um fluxo desse tipo é expirado, seu registro de fluxo deve ser mantido pelo processo de medição, de modo a dinamizar o processo de manutenção de fluxos, visto que esse fluxo tem grandes chances de ser observado posteriormente.

3.3.2 Processo de exportação dos fluxos

O processo de exportação é o bloco funcional que envia as informações observadas a um ou mais coletores usando o protocolo IPFIX, que comunica com os processos de medição para receber os registros de fluxo.

O processo de exportação que define se existe a necessidade de exportar um fluxo expirado e, se for o caso, realizar essa operação. No caso de fluxos de longa duração, o processo de exportação deve enviar os registros de fluxo em um intervalo regular de tempo,

ou baseado em alguma política de exportação. Esta periodicidade ou política de exportação deve ser configurável no próprio processo de exportação.

A identificação do ponto de observação ao qual o registro de fluxo exportado está associado pode auxiliar na análise das informações coletadas. Para tanto, é recomendado que o processo de exportação dos dispositivos IPFIX envie essa informação aos coletores. Obviamente, em casos em que há um único ponto de observação ou onde a indicação deste ponto é sabidamente dispensável pela aplicação de análise das informações de fluxo, o processo de medição não precisa apresentar esta informação ao processo exportador, eliminando, desta forma, a necessidade de seu envio.

3.4 DETALHAMENTO DO PROTOCOLO IPFIX

O principio básico do protocolo IPFIX, comum ao *NetFlow*, é o uso de *templates* na definição dos registros de fluxo. Conforme visto na seção anterior, o dispositivo IPFIX envia ao coletor não apenas as informações sobre os fluxos (registros de fluxo), mas também informação sobre o processo de medição como um todo, referenciadas como informações de controle. Todas essas informações irão compor os pacotes do protocolo IPFIX. A lista detalhada com cada informação que pode compor o pacote IPFIX, denominadas elementos de informação, está definida em [QUI 06], enquanto que [CLA 06] define de que forma essas informações são transmitidas.

As informações de controle podem ser de dois tipos:

- informação de controle para os fluxos;
- informação de controle para configuração.

As informações de controle para os fluxos são enviadas para que o coletor possa capturar a estrutura e a semântica dos dados exportados nos registros de fluxo. Já as informações de controle para configuração carregam o restante das informações acerca do

processo de medição, como os dados estatísticos do dispositivo IPFIX e os critérios de seleção dos pacotes que formam esse fluxo.

Visando seu encaminhamento, todas essas informações são codificadas em *templates*, formando uma seqüência ordenada de pares <tipo de dado, tamanho>, especificando todos os elementos enviados. Cada um desses *templates* é identificado de forma única por meio de um identificador de *template*. Esse processo gera dois tipos de *templates*:

- *templates* para registros de fluxo, gerados a partir das informações de controle para os fluxos, que informam como os registros de fluxo serão interpretados;
- *templates* de opções de controle, gerados a partir das informações de controle para configuração, que analisam os dados sobre o processo de medição.

A partir do agrupamento de *templates*, dados de controle e registros de fluxo, o datagrama IPFIX é formado. Nesse processo, é adicionado também o cabeçalho IPFIX. Dessa forma, a mensagem IPFIX será formada pelo seu cabeçalho e por um ou mais agrupamentos de informações. Esses agrupamentos podem ser de registros de fluxo, *templates* de registros de fluxo, dados de controle ou *templates* de opções de controle.

3.4.1 Formato das mensagens IPFIX

A Figura 8 mostra alguns exemplos de mensagens IPFIX. Em (a) são transmitidas mensagens onde *templates*, de registro de fluxo ou de opções de controle, são transmitidos de forma intercalada com os dados aos quais eles fazem referência, sejam eles registros de fluxo ou dados de opções. Outra alternativa é primeiramente realizar o envio dos *templates* (b) para que o coletor tenha conhecimento de todos os registros com os quais estará lidando. Logo após, podem ser transmitidas mensagens consistindo somente do agrupamento de dados (c). Dessa forma, a maioria das mensagens IPFIX consistirá apenas de agrupamentos de dados.

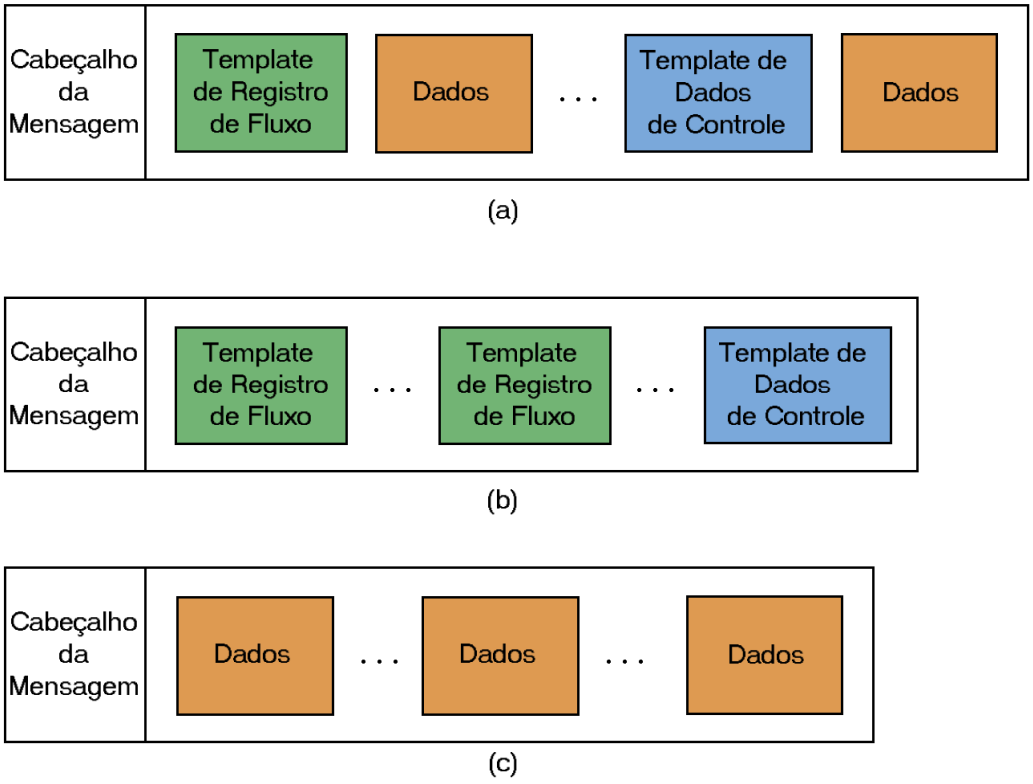


Figura 8 Exemplos de mensagens IPFIX.

As subseções seguintes detalham o formato de cada elemento que pode compor as mensagens IPFIX.

3.4.2 Cabeçalho IPFIX

O cabeçalho das mensagens IPFIX é mostrado na Figura 9 . Cada um dos seus campos é comentado em seguida.

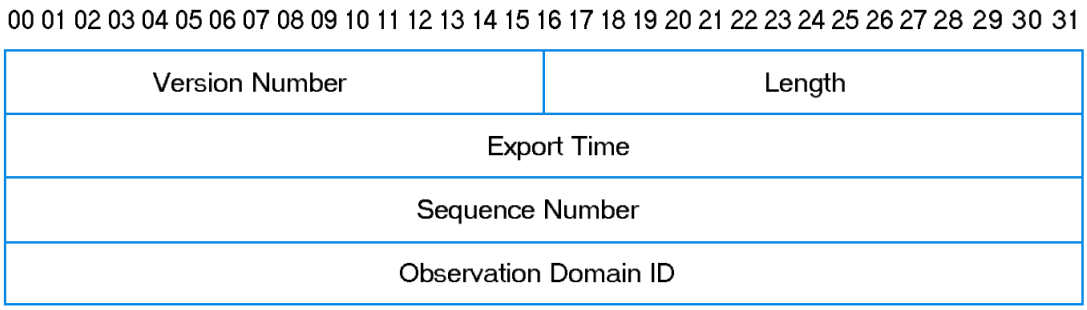


Figura 9 Formato do cabeçalho IPFIX.

- *Version Number*: versão do formato da mensagem. O valor do campo para essa versão é 0x000a, identificado pelo incremento em 1 do NetFlow v9.
- *Length*: tamanho da mensagem IPFIX, expresso em octetos, incluindo o cabeçalho.
- *Export Time*: marcação de tempo em que a mensagem deixou o processo de exportação do dispositivo IPFIX.
- *Sequence Number*: marcação do contador incremental de cada registro de dado enviado pelo processo de exportação, podendo ser usado pelo coletor para identificar registros perdidos. *Templates* não incrementam esse número de seqüência.
- *Observation Domain ID*: identificador do domínio de observação que é localmente único ao processo de exportação de dados.

3.4.3 Formato dos Agrupamentos

Cada agrupamento é formado por um cabeçalho próprio e um conjunto de elementos, sejam eles *templates*, dados de opções de controle ou registros de fluxo. Além disso, pode ser necessário preencher o agrupamento com um enchimento. O formato do agrupamento, incluindo seu cabeçalho, é mostrado na Figura 10 .

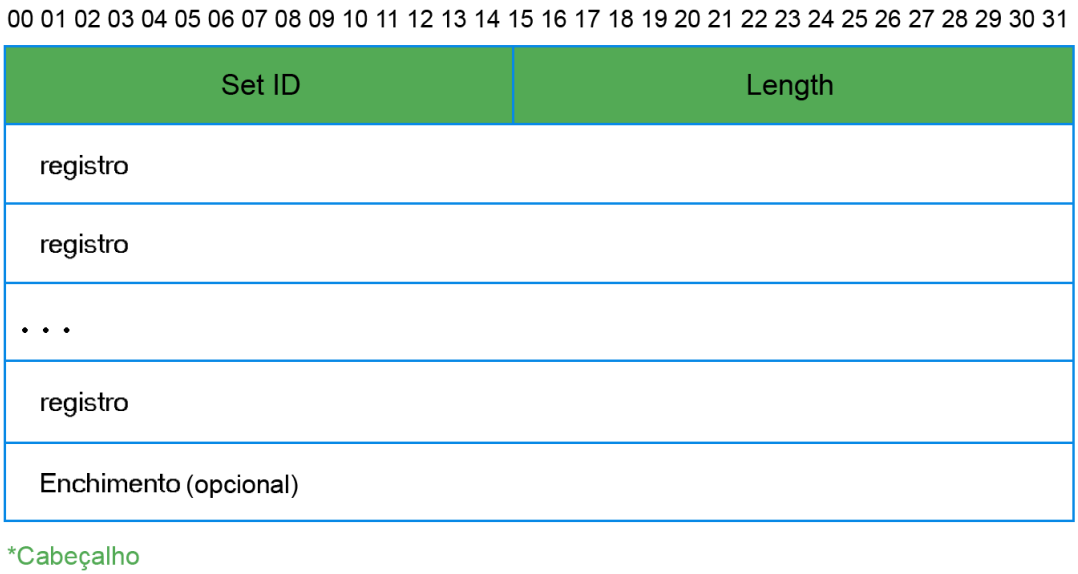


Figura 10 Formato dos agrupamentos.

Como pode ser visto na Figura 10 , o cabeçalho da mensagem é formado apenas por dois campos, o set ID, que identifica o tipo de agrupamento, e o tamanho do agrupamento, que é indicado em octetos (daí a eventual necessidade do uso de bytes de enchimento). O conteúdo do agrupamento varia de acordo com seu tipo. Cada uma das variações é detalhada a seguir.

3.4.4 Agrupamento de *templates* de registros de fluxo

A partir desses agrupamentos o coletor é capaz de analisar todas as informações enviadas para ele acerca dos fluxos encontrados. Seu conteúdo consiste de um cabeçalho seguido de diversos especificadores de campo, cada um descrevendo um elemento de informação a ser analisado.

O cabeçalho do *template* consiste de um identificador de *template* e um contador que armazena o número de especificadores de campo que compõe o mesmo.

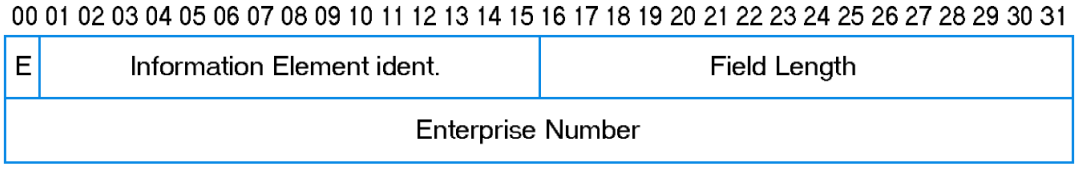


Figura 11 Formato dos especificadores de campo.

O formato dos especificadores de campo é mostrado na Figura 11 . Como pode ser visto nesta figura, o especificador de campo possui as seguintes informações:

- *Enterprise Number* (E): indica se o elemento de informação é especificado pelo IETF, quando setado em 0, ou é definido por outra autoridade (a partir de um padrão proprietário, por exemplo), quando setado em 1;
- *Information Element Identifier*: campo que identifica o tipo de elemento de informação. Os elementos definidos pelo IETF constam em [QUI 06];
- *Field Length*: tamanho do elemento de informação, também consta na listagem apresentada em [QUI 06];
- *Enterprise Number*: campo opcional, que é inserido caso o elemento de informação não seja definida pelo IETF, possui o *IANA Enterprise Number*, da autoridade que padronizou esse elemento. A lista com os números registrados está disponível em http://www.iana.org/assignments/enterprise-numbers.

Cada especificador de campo representa um par <tipo de informação, tamanho da informação> usado para descrever cada informação relativa aos registros de fluxo repassada do dispositivo IPFIX para o coletor.

3.4.5 Agrupamento de *templates* de opções de controle

Conforme descrito anteriormente, esses *templates* definem as informações adicionais que são enviadas ao coletor, as quais não seriam inferidas diretamente apenas com os registros de fluxo. Essa informação deve ser enviada juntamente com a indicação do escopo ao qual ela

se refere, como, por exemplo, uma determinada interface, um processo de medição ou uma etapa desse processo. Cada opção de controle pode estar associada a múltiplos escopos dentro de um mesmo agrupamento. Um exemplo disso é indicar dois escopos, um para descrever um processo de medição ao qual a opção de controle está associada, e outro para identificar uma etapa desse processo. Dessa forma estaria se indicando uma determinada etapa deste processo de medição em específico.

A informação de escopo é inserida no cabeçalho do *template*. O restante de suas informações é idêntico ao cabeçalho dos *templates* para registros de fluxo. O formato especificado para o cabeçalho dos *templates* de opções de controle é mostrado na Figura 12 .

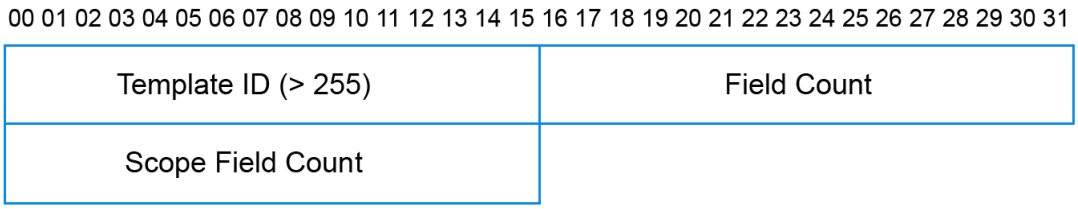


Figura 12 Cabeçalho dos *templates* de opções de controle.

O conteúdo deste *template* é idêntico ao *template* de registros de fluxos, porém contando com os especificadores de campo, definidos em [QUI 06], relativos às informações de controle.

3.4.6 Agrupamentos de dados (registros de fluxo e dados de controle)

Os agrupamentos de dados contêm apenas os campos de valores. Para a associação entre o *template* enviado e seu dado, o agrupamento de dados deve possuir número de identificação (*Set ID*) igual ao *Template ID* ao qual ele se refere.

3.4.7 Protocolo de transporte

O protocolo IPFIX foi especificado para ser transportado independentemente do protocolo da camada de transporte escolhido. Entretanto, [CLA 06] sugere o uso do SCTP para o transporte das informações IPFIX, devido a capacidade desse protocolo de, além de manter a comunicação em *links* susceptíveis a congestionamento, limitar a taxa dessa comunicação. Apesar disso, é possível realizar o transporte usando TCP ou até mesmo UDP, sendo que para o uso do UDP é necessário utilizar algum método que garanta o provisionamento das mensagens IPFIX, como, por exemplo, um *link* dedicado.

Por questões de segurança, é indicado também o uso do TLS [DIE 06] ou do DTLS [RES 06] junto ao protocolo IPFIX, de modo a prover garantias de confidencialidade, integridade e autenticação à comunicação.

3.5 COLETOR IPFIX

O coletor IPFIX usa as definições de *templates* para interpretar cada elemento de informação desse registro de fluxo. Para isso, o processo exportador do dispositivo IPFIX precisa garantir que o coletor conheça o *template* para cada registro de fluxo utilizado.

As funções do processo coletor devem incluir:

- identificação, aceitação e decodificação das mensagens IPFIX de diferentes dispositivos IPFIX. Esse processo deve incluir a identificação dos domínios de observação associados a cada registro de fluxo, caso essa informação seja relevante, conforme já comentado;
- armazenamento das informações de controle e dos registros de fluxo recebidos de cada dispositivo IPFIX.

De forma geral, podemos dizer que o coletor é, basicamente, formado por dois processos principais, relacionados diretamente com cada uma das tarefas acima citadas. Esses processos são:

- processo de coleta dos pacotes IPFIX;
- processo de armazenamento das informações de fluxo.

A forma como as informações de controle e os registros de fluxo são armazenados e acessados pela aplicação de análise dos dados não são padronizadas pelo IPFIX.

4 PROCEDIMENTO EXPERIMENTAL

O presente capítulo destaca o desenvolvimento do sistema de análise de fluxos baseado nas especificações do grupo de trabalho IPFIX. Para tanto, foram seguidas as definições acerca de cada um dos elementos que formam o sistema, destacadas no capítulo anterior. A implementação de cada um desses dispositivos é mostrada nas seções subseqüentes.

Também é destacado neste capítulo o desenvolvimento do coletor, em sua etapa de armazenamento de dados, e uma ferramenta *case* de análise de dados de fluxo para a apresentação da matriz de tráfego da rede. Embora esses módulos não façam parte do escopo de trabalho do grupo IPFIX, são essenciais para a análise do bom funcionamento do sistema como um todo, além de servir como base para a futura implementação de outras aplicações que utilizem as informações de fluxo.

Inicialmente, é importante destacar a definição do sistema operacional utilizado no sistema, onde optou-se pela adoção do sistema operacional *Linux*. Isso foi feito por diversas razões, dentre as quais podem ser destacadas:

- o amplo uso desse sistema operacional pelo grupo de pesquisas GPARC&TI, que possui a maioria dos sistemas, ferramentas e bibliotecas implementados para esse sistema operacional;
- é *open-source*, o que proporciona grande flexibilidade no desenvolvimento de sistemas, pois possibilita a alteração e adequação de suas características de funcionamento de acordo com as necessidades;
- soluções que utilizam essa tecnologia tendem ter um custo de produção mais baixo, devido à existência de grande número de bibliotecas e sistemas de boa qualidade e de livre uso;
- existe uma grande quantidade de bibliotecas específicas para análise de redes largamente utilizadas, tanto em âmbito acadêmico quanto em âmbito comercial.

Outra solução comum aos módulos é o uso da linguagem de programação C++ no desenvolvimento dos sistemas de mais baixo nível. Essa escolha se deve a capacidade do C++ em valer-se de recursos de baixo nível com certa facilidade, aliada as potencialidades do uso

do paradigma de programação orientada a objetos, o que permite o melhor desenvolvimento de sistemas complexos alinhados aos principais conceitos de Engenharia de Software [SCH 98].

Outras vantagens no uso do C++ são a portabilidade dos códigos implementados nessa linguagem, devido a sua padronização, e a diversidade de bibliotecas disponíveis para essa linguagem, além de ser uma linguagem que possui grande interoperabilidade com o *Linux*, visto que as origens deste sistema operacional estão fundamentadas no C.

As demais tecnologias utilizadas são específicas para cada um dos módulos, sendo detalhadas conforme seu uso. A Figura 13 detalha cada um desses recursos. As seções seguintes tratam de cada um dos componentes do sistema, descrevendo, respectivamente:

- o dispositivo IPFIX;
- o coletor;
- a aplicação de análise de fluxos.

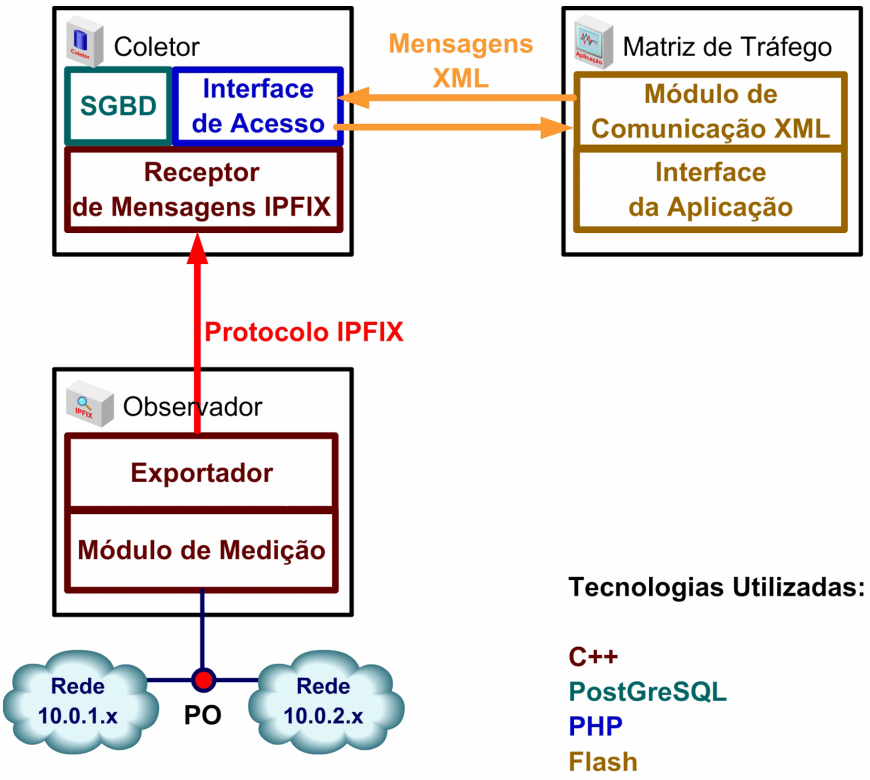


Figura 13 Tecnologias utilizadas no protótipo desenvolvido.

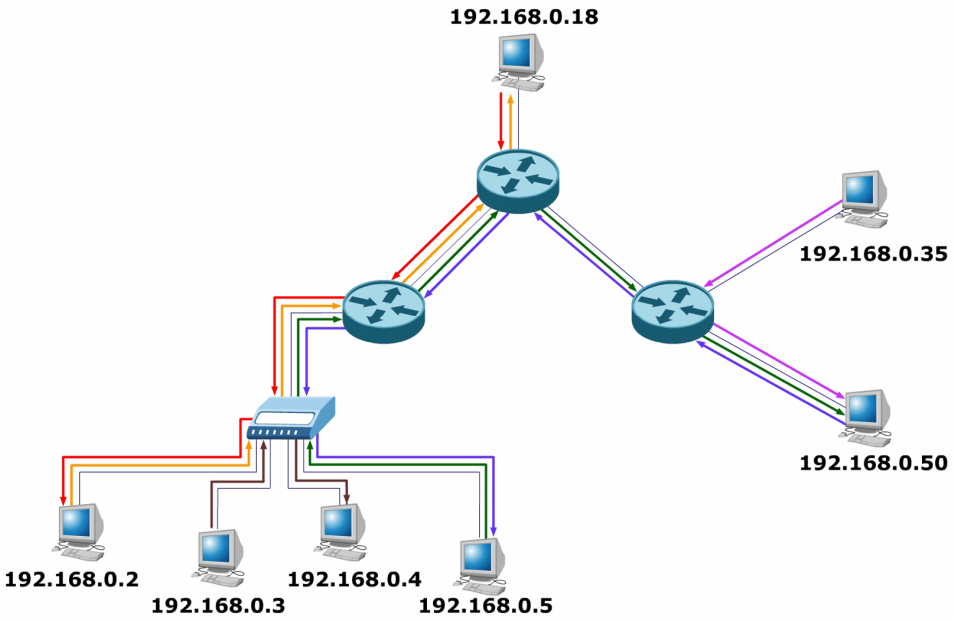
4.1 O SISTEMA OBSERVADOR DE FLUXOS (DISPOSITIVO IPFIX)

É importante destacar que esse trabalho foca o desenvolvimento de um dispositivo IPFIX, cuja função específica é a aquisição de dados de fluxo, sendo aqui denominado também de observador de fluxos, pois não possui nenhuma tarefa adicional, diferente de outros dispositivos como, por exemplo, roteadores IPFIX. Esse dispositivo foi implementado visando possuir apenas um ponto de observação, aumentando desta forma a capacidade de análise dos pacotes que trafegam nesse ponto de observação ao mesmo tempo em que simplifica o processo de aquisição das informações de fluxo.

O sistema foi implementado utilizando uma abordagem usual de definição de fluxos IP [BRO 01], onde estes são definidos a partir dos campos:

- endereço IP de origem;
- endereço IP de destino;
- protocolo da camada de transporte;
- porta da camada de transporte de origem;
- porta da camada de transporte de destino.

A partir da análise desses campos, fica claro que os fluxos encontrados podem ser bidirecionais. A Figura 14 mostra exemplos de fluxos encontrados a partir dessa definição.



Fluxos Verificados

IP Origem	IP Destino	Porta Origem	Porta Destino	Protocolo
192.168.0.2	192.168.0.18	1600	80	TCP
192.168.0.18	192.168.0.2	80	1600	TCP
192.168.0.5	192.168.0.50	34562	53	UDP
192.168.0.35	192.168.0.50	44009	53	UDP
192.168.0.50	192.168.0.5	53	34562	UDP
192.168.0.3	192.168.0.4	123	32768	UDP

Figura 14 Identificação de fluxos a partir das chaves definidas.

Conforme visto na seção 3.3, são funções do dispositivo IPFIX:

- a coleta dos pacotes que trafegam nos pontos de observação;
- a análise dos mesmos visando a identificação de seus respectivos fluxos;
- a marcação de tempo (*timestamping*);
- a amostragem desses pacotes e classificação dos seus respectivos fluxos (critérios de seleção);
- a criação dos registros de fluxos e *templates*; a montagem e envio do datagrama IPFIX.

Para realizar todas essas funções, o dispositivo possui dois módulos funcionais básicos, o módulo de medição dos fluxos e o módulo de exportação. O módulo de medição de fluxos é responsável pelas etapas desde a aquisição dos dados e seu tratamento até a entrega dos

templates e dos registros de fluxos ao módulo de exportação. Este, por sua vez, monta os datagramas e realiza o envio dos mesmos. Além disso, é necessária a especificação de diversos parâmetros de configuração para o funcionamento do observador IPFIX. Esses parâmetros são descritos na subseção 4.1.1, sendo posteriormente detalhados os módulos de medição e exportação, respectivamente nas subseções 4.1.2 e 4.1.3.

4.1.1 Parâmetros de configuração

Toda a configuração do sistema observador IPFIX é passada por parâmetros, quando na inicialização do mesmo. Nessa configuração, são passados os parâmetros básicos de configuração do sistema, seguido de parâmetros opcionais, que permitem flexibilizar esse processo de medição, de acordo com as características do ponto de observação que se deseja medir. Cada um desses parâmetros é descrito a seguir:

- *Observer Identifier* (-I): parâmetro obrigatório que estabelece o identificador único associado ao observador IPFIX;
- *Collector* (-c): parâmetro obrigatório que identifica o endereço IP do coletor;
- *Exporting Period* (-t): identifica o período de exportação do observador IPFIX, em segundos, possuindo valor padrão de 5 segundos;
- *Interface* (-i): interface do dispositivo IPFIX que se deseja observar, sendo que todas as interfaces são observadas por padrão;
- *Configuration File* (-x): arquivo de configuração XML, com a especificação dos *templates* a serem utilizados, com caminho padrão /usr/local/etc/obs-template.xml;
- *Filter* (-F): filtro da *libpcap* utilizado.

Cada um desses parâmetros tem suas funções detalhadas nas subseções seguintes.

4.1.2 Módulo de medição dos fluxos

Conforme destacado anteriormente, a primeira tarefa do observador IPFIX é realizar a coleta dos pacotes a serem analisados no ponto de observação. Para isso, foi utilizada a biblioteca *libpcap* [TCP 07], uma versão para *linux* da biblioteca *pcap*. Essa biblioteca já possui seu uso bastante sedimentado por diversas ferramentas de análise passiva do tráfego, sendo utilizada por sistemas como o *tcpdump*, o *wireshark* (*ethereal*), *snort*, entre outros. A *pcap* foi desenvolvida pelo *Berkeley National Laboratory*, da universidade de Berkeley, sendo interessante destacar que ela possui também uma versão para o sistema operacional *Windows*, denominada *winpcap*.

A *libpcap* permite capturar os pacotes tanto em modo promíscuo quanto em modo não promíscuo, sendo que pelas necessidades desse trabalho ela foi utilizada operando em modo promíscuo, de forma a permitir que todos os pacotes que passem pelo ponto de observação sejam capturados.

Durante a captura dos pacotes, a *libpcap* observa a interface de rede e, quando um pacote é recebido, ela o coleta e faz sua análise. Para isso, ela utiliza três conjuntos de *structs* desenvolvidos, cada um relativo a uma camada IP:

- enlace, que utiliza a *struct ethernet*;
- rede, que utiliza as *structs* IP, ARP e ICMP;
- transporte, que utiliza as *structs* UDP e TCP.

Primeiramente, o *stream* de bits recebido é analisado, identificando o seu protocolo da camada de enlace. Caso o sistema operacional identifique-o como sendo *ethernet*, ele aplica a *struct ethernet*, de modo a recolher os dados de interesse relativos a este protocolo como, por exemplo, o endereço de interface MAC. Se o protocolo utilizado for diferente do *ethernet*, como o *token ring*, por exemplo, o conteúdo do frame é analisado, porém não é possível extrair nenhuma informação do cabeçalho deste frame.

Após essa etapa, os dados extraídos são analisados verificando seu protocolo da camada de rede. Caso seja identificado o uso de um dos três protocolos definidos a partir das *structs*

implementadas, o pacote é analisado. Por exemplo, se for um pacote IP, é possível extrair com o auxílio da *struct* IP os campos de endereço origem e destino e o tipo de protocolo da camada superior. Para um pacote ser contabilizado como fluxo este deve passar ao menos pela camada de rede e possuir algum tipo de endereçamento (*unicast*, *multicast* ou *broadcast*).

Em seguida, com o auxílio do campo IP relativo ao protocolo utilizado na camada superior, é possível identificar a *struct* utilizada na camada de transporte, caso seja utilizado o protocolo UDP ou TCP, recolhendo os dados de interesse presentes no cabeçalho deste pacote como, por exemplo, suas portas de origem e destino.

Além dessa análise, é verificado o número de bytes que compõe esse pacote, visando sua posterior contabilização.

Desse processo resulta a extração de todos os elementos para a criação da chave *hash*, que identifica de forma única esse fluxo. Essa chave é, então, gerada a partir desses elementos e enviada para uma lista estruturada a partir de um elemento *map*, o qual possui a relação de todos os fluxos identificados até o momento e um ponteiro para o local em que seus dados são armazenados. Aqui podem ocorrer duas situações:

- o chave não é encontrada no *map*, o que sinaliza que este fluxo ainda não havia sido identificado pelo observador. O processo gera então uma nova entrada no *map* e armazena nele os dados relativos a esse fluxo. Esses dados são os contadores, que armazenam o total de pacotes em cada um dos sentidos da comunicação e o total de bytes que compõe esse fluxo, tanto no sentido de *upstream* quanto no sentido de *downstream*;
- a chave é encontrada no *map*, fazendo com que o processo identifique os contadores responsáveis pelo armazenamento dos dados desse fluxo e os incremente de acordo com o que foi verificado no pacote.

Visando a liberação de espaço na memória, um fluxo sempre deve ser removido do registro de fluxos quando ocorre a expiração do mesmo. Um fluxo é considerado expirado se nenhum pacote pertencente a ele foi identificado por um determinado período de tempo. Esse período de tempo é definido pelo usuário através do parâmetro *flow_timeout*. Portanto, uma tarefa adicional existente no processo de medição é consultar periodicamente as entradas de

fluxo, identificando os fluxos expirados. Estes são então marcados e, após terem suas informações exportadas, são removidos da lista.

Conforme visto no capítulo anterior, nessa etapa é possível se realizar processos de seleção dos pacotes, a partir do uso de filtragem e técnicas de amostragem. O sistema observador desenvolvido possui essas duas possibilidades disponíveis.

A filtragem pode ser realizada através dos filtros disponíveis na própria *libpcap*, acessados a partir do parâmetro de configuração "*-F <parâmetros do filtro libpcap>*". Assim, ela será responsável por aplicar essas regras definidas durante a captura dos pacotes.

Por sua vez, a amostragem realizada pelo dispositivo IPFIX é implementada no trabalho [GUI 07]. Maiores detalhes sobre a técnica de amostragem utilizada, bem como seus resultados, podem ser obtidos neste trabalho.

4.1.3 Módulo de exportação

Paralelamente ao processo de medição de fluxos, o dispositivo IPFIX inicia o processo de exportação dos dados IPFIX. Para isso, primeiramente, o dispositivo deve verificar a disponibilidade do coletor indicado durante a sua configuração, através do parâmetro "*–c <endereço IP do coletor>*". Caso o coletor não esteja acessível um arquivo com a informação de erro deve ser criado informando o momento em que se tentou efetuar a conexão.

O controle da conexão com o coletor é realizado com o auxílio da biblioteca *libipfix* [MAR 07]. Esta biblioteca está atualmente sendo desenvolvida, já possuindo, entretanto, grande compatibilidade com as especificações do IETF. Sua implementação, inclusive, originou o trabalho [BOS 06] que possui indicações para o desenvolvimento e uso de sistemas que utilizam o protocolo IPFIX. Entretanto, embora a implementação em concordância com as definições do IPFIX esteja adiantada, a biblioteca ainda carece de uma maior documentação. A versão da biblioteca *libipfix* utilizada no presente trabalho foi a 0.7. Basicamente, para estabelecer a conexão com o coletor através desta biblioteca, primeiramente é necessário iniciá-la, através da função *ipfix_init()*, e logo em seguida abrir

seu módulo exportador, utilizando a função *ipfix_open()*. Após esse processo já é possível estabelecer a conexão com o coletor, com a função *ipfix_add_collector()*. A *libipfix* já possui suporte para o envio das informações de fluxo do dispositivo IPFIX para diversos coletores.

Logo após o estabelecimento da conexão, o módulo responsável pela exportação precisa receber o modelo de dados a serem transportados. Conforme visto no capitulo anterior, esse modelo é obtido a partir do uso *templates*.

Conforme visto anteriormente, no dispositivo IPFIX prototipado, o *template* a ser formado é definido pelo usuário, através do uso do parâmetro de configuração "*-x <caminho do arquivo de configuração XML>*". A partir da montagem do arquivo *Extensible Markup Language* (XML) são descritos cada um dos elementos que forma esse *template*, utilizando tanto os elementos de informação definidos em [QUI 06] quanto elementos de informação proprietários.

O XML [BIR 01] é um subtipo de SGML (*Standard Generalized Markup Language*, ou Linguagem Padronizada de Marcação Genérica) capaz de descrever diversos tipos de dados. Seu propósito principal é a auxiliar o compartilhamento de informações através da Internet. Ele foi escolhido na definição dos *templates* por diversos motivos, entre os quais podem ser citados:

- a existência de um *parser* nativo na maioria das linguagens de programação;
- qualquer aplicação que possua um *parser* XML é capaz de extrair as informações do arquivo, desde que receba o *Document Type Definition* (DTD) correspondente;
- o XML simplifica a mudança nos campos de configuração, ou seja, através de seu uso a inserção ou remoção de elementos que compõe a configuração fica simplificada;
- finalmente, o XML é a ferramenta recomendada pelo W3C (*World Wide Web Consortium*) para gerar linguagens de marcação para necessidades especiais.

O XML definido para configuração dos *templates* possui o seguinte DTD:

```
<!ELEMENT ipfix (template)>
<!ELEMENT template (field)>
<!ELEMENT field EMPTY>
<!ATTLIST template
   type (data|option) "data"
>
<!ATTLIST field
  enterprise CDATA '0'
  name       CDATA #REQUIRED
>
```

Como pode ser visto no DTD, o XML apresentará um ou mais elementos *template* formado por vários elementos *field*. Cada um dos campos *field* definem um elemento de informação, especificado pelo atributo *name*. Além de *name*, o campo *field* possui o atributo *enterprise*, que define o *Enterprise Number* da autoridade que define o campo. Conforme comentado na subseção 3.4.4, esse valor é 0 caso a informação seja definida pelo IETF.

Para aplicação a desenvolvida foi definida a seguinte configuração de *template*:

```
<?xml version="1.0" encoding="ISO8859-1"?>
<!DOCTYPE ipfix SYSTEM "obs-template.dtd" >
<ipfix>
      <template type="data">
            <field enterprise="0" name="sourceTransportPort"/>
            <field enterprise="0" name="sourceIPv4Address"/>
            <field enterprise="0" name="destinationTransportPort"/>
            <field enterprise="0" name="destinationIPv4Address"/>
            <field enterprise="0" name="protocolIdentifier"/>
            <field enterprise="99999" name="flowIdentifier"/>
            <field enterprise="99999" name="octetDeltaCountIn"/>
            <field enterprise="99999" name="octetDeltaCountOut"/>
            <field enterprise="99999" name="octetTotalCountIn"/>
            <field enterprise="99999" name="octetTotalCountOut"/>
            <field enterprise="99999" name="packetDeltaCountIn"/>
            <field enterprise="99999" name="packetDeltaCountOut"/>
            <field enterprise="99999" name="packetTotalCountIn"/>
            <field enterprise="99999" name="packetTotalCountOut"/>
      </template>
</ipfix>
```

A Tabela 2 descreve cada um dos elementos de informação que compõe o *template*, de acordo com a configuração descrita acima.

Tabela 2 Informações exportadas pelo sistema de medição

Campo	Descrição
sourceTransportPort	Porta da camada de transporte do transmissor.
sourceIPv4Address	Endereço IP do transmissor.
destinationTransportPort	Número da porta no receptor.
destinationIPv4Address	Endereço IP do receptor.
protocolIdentifier	Número identificador do protocolo da camada de transporte.
flowIdentifier	Chave de identificação do fluxo, formada a partir do processo descrito na subseção 4.1.1.
octetDeltaCountIn	Número de *bytes* observados no sentido *downstream*, desde a última mensagem IPFIX exportada.
octetDeltaCountOut	Número de *bytes* observados no sentido *upstream*, desde a última mensagem IPFIX exportada.
octetTotalCountIn	Total de *bytes* contabilizados no sentido *downstream*.
octetTotalCountOut	Total de *bytes* contabilizados no sentido *upstream*.
packetDeltaCountIn	Número de pacotes observados no sentido *downstream*, desde a mensagem IPFIX exportada.
packetDeltaCountOut	Número de pacotes observados no sentido *upstream*, desde a última mensagem IPFIX exportada.
packetTotalCountIn	Total de pacotes contabilizados no sentido *downstream*.
packetTotalCountOut	Total de pacotes contabilizados no sentido *upstream*.

O tratamento do XML no observador é realizado pela biblioteca *libxerces*. Ela fornece ao C++ todo suporte no tratamento de mensagens XML. A partir da extração da informação sobre os *templates*, utilizou-se novamente a biblioteca *libipfix* para montar os *templates*. Nesta biblioteca, a função *ipfix_new_data_template()* é utilizada para a criação do *template*, enquanto que a função *ipfix_add_field()* insere cada um dos campos nesse *template*.

Para realizar o envio de mensagens, o dispositivo IPFIX utiliza um temporizador que, seguindo um intervalo de tempo pré-configurado através do parâmetro "*-t <intervalo de tempo em segundos>*", envia um sinal ao módulo exportador. Este por sua vez, ao receber o sinal, acessa os dados que estão armazenados no *map* de registros de fluxo e monta a mensagem IPFIX, inserindo nesta o seu identificador único, configurado através do parâmetro "*-I <identificador do observador IPFIX>*". Essa mensagem é então enviada ao dispositivo coletor, através do uso da função *ipfix_export()*.

4.2 O COLETOR

Embora não faça parte de forma direta das especificações IPFIX, a biblioteca *libipfix* provê funções para construção do coletor IPFIX. Para isso, ela fornece dois métodos para armazenar os dados coletados: armazenar os dados coletados em texto plano ou armazená-los em uma base de dados *MySQL*.

Para realizar o armazenamento de informações em um arquivo, utiliza-se a função *ipfix_col_init_fileexport()*, informando, nos parâmetros da mesma, o caminho onde os dados serão armazenados.

Por outro lado, no caso de utilizar uma base de dados *MySQL*, é necessário simplesmente substituir a chamada *ipfix_col_init_fileexport()* pela função *ipfix_col_init_myslexport()*.

Após a escolha do método de armazenamento dos dados de fluxo, o sistema coletor está apto a receber as informações de fluxo dos dispositivos IPFIX. Para isso, é utilizado um *loop* que fica aguardando a chegada de mensagens IPFIX, através da função *ipfix_col_listen()*, para seu posterior armazenamento.

Nesse ponto foi necessário realizar uma alteração no funcionamento da *libipfix*. As ferramentas de análise de rede implementadas pelo GPARC utilizam tradicionalmente o Sistema Gerenciador de Banco de Dados (SGBD) *PostGreSQL*.

A escolha do *PostGreSQL* é motivada pelas vantagens desse sistema, destacadas em [HEU 00]. Primeiramente, trata-se de um sistema objeto-relacional com código aberto, que possui compatibilidade com os padrões SQL92/SQL99. Além disso, provê *Multi-Version Concurrency Control* (MVCC), ou Controle de Concorrência Multi-versão, integridade referencial, possibilidade de desenvolver funções armazenadas (*Stored Procedures*) em várias linguagens de programação, como por exemplo, PL/PgSQL, PHP, Perl, Python e Ruby, gatilhos (*triggers*), possibilidade do usuário criar tipos de dados próprios e o uso de *Secure Sockets Layer* (SSL) no estabelecimento de conexões. Além disso, outro beneficio associado a escolha do *PostGreSQL* é a sua fácil portabilidade ao sistema operacional *Windows*, dada a

existência de uma distribuição nativa para este. As alterações realizadas e a estrutura da base de dados são discutidas na próxima seção.

4.2.1 Base de dados

Conforme comentado anteriormente, optou-se por utilizar a base de dados *PostGreSQL* para o armazenamento das informações de fluxo. Para isso a *libipfix* foi modificada e recompilada, alterando sua estrutura de modo a substituir o uso do *MySQL* pelo *PostGreSQL*, sem a necessidade de se modificar as chamadas para a função da libipfix responsáveis pela manutenção da base de dados.

A Figura 15 apresenta o diagrama entidade-relacionamento (ER) da base de dados implementada no coletor.

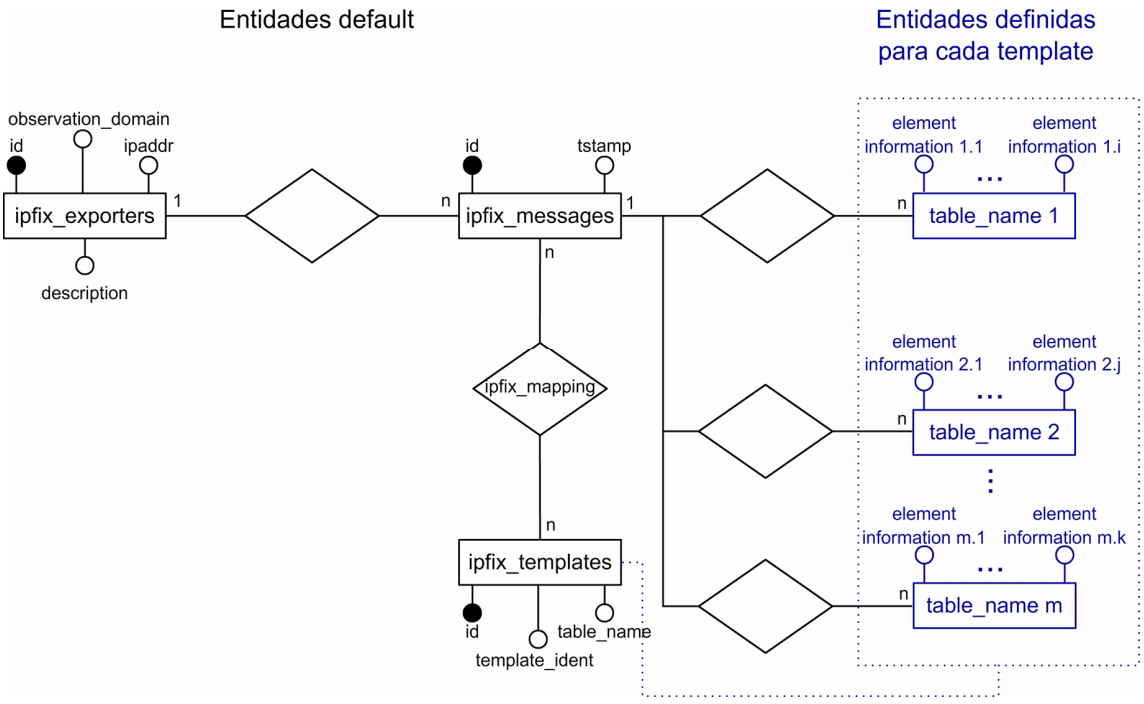


Figura 15 Diagrama ER proposto pela *libipfix*.

Como pode ser visto nesta figura, o coletor monta quatro tabelas padrão, mais uma tabela para cada *template* coletado. Essas quatro tabelas são denominadas *ipfix_exporters*,

ipfix_messages, *ipfix_templates* e *ipfix_mapping*, sendo que cada uma delas é detalhada a seguir.

A tabela *ipfix_exporters* armazena informação sobre cada observador IPFIX (basicamente seu endereço IP) que envia seus dados para o coletor. Cada mensagem IPFIX recebida gera uma entrada na tabela *ipfix_messages* com a marcação de seu *timestamp* e a indicação do domínio de observação, verificado a partir do cabeçalho da mesma.

Para cada *template* diferente recebido no coletor uma nova tabela é criada dinamicamente, com o objetivo de armazenar o conteúdo dos registros de dados relacionados a esse *template*. Esses *templates* recebem um identificador único na tabela *ipfix_templates*. Esta é usada para mapear o identificador de *template* e a nova tabela criada. Finalmente, a tabela *ipfix_mapping* armazena a informação sobre quais *templates* estão sendo utilizados pelos registros de dados de cada mensagem IPFIX.

Cada tabela criada a partir de um novo *template* possui uma coluna para cada elemento de informação que compõe este *template*. Além disso, possui também a coluna *id_ipfix_messages*, que faz o mapeamento entre a mensagem IPFIX recebida e os dados que ela armazenava. Toda essa estrutura é criada de modo a permitir que a base de dados armazene todas as informações coletadas, independentemente da chegada de novos *templates* que possam vir a descrever novos elementos de informação.

4.3 A APLICAÇÃO DE ANÁLISE

Visando a verificação das informações de fluxo, foi desenvolvido um sistema de análise de matriz de tráfegos. A matriz de tráfegos é uma abstração das medições dos fluxos encontrados entre os nodos da rede. Seu uso é de grande importância em *Internet Service Providers* (ISPs), ou Provedores de Serviços, para realizar tarefas de engenharia de tráfego como, por exemplo, encontrar gargalos na rede ou identificar ataques por negação de serviço (*Denial of Service* - DoS) [KOC 03].

A matriz de tráfegos desenvolvida utiliza as informações obtidas pelo dispositivo IPFIX e armazenadas em uma base de dados pelo coletor. Seu desenvolvimento é fundamentado no uso de interfaces *web* no desenvolvimento de aplicações. Esse tipo de interface objetiva viabilizar, de maneira mais amigável e intuitiva, a interação de diferentes perfis de usuários (gerentes, técnicos, especialistas) com o sistema. Esta funcionalidade introduz uma série de benefícios ao protótipo final, como por exemplo, a facilidade de integrar este módulo a outros sistemas legados e demais NMS que possuam sua interface *web*, caso da maior parte das aplicações de monitoramento e gerenciamento de redes desenvolvidas pelo GPARC.

Para o acesso aos dados armazenados na base *PostGreSQL*, foi utilizado o PHP (*PHP Hypertext Preprocessor*) [CAS 00], linguagem de programação orientada a objetos interpretada, amplamente utilizada para geração de conteúdo dinâmico na *web*.

Complementando a ferramenta de análise, sua interface foi desenvolvida com a ferramenta gráfica vetorial *Flash* [REI 06] que, entre outras aplicações, é utilizada para o desenvolvimento de animações interativas. Os arquivos gerados a partir desta ferramenta podem ser visualizados em páginas *web* e seu sistema conta, adicionalmente, com a linguagem de programação *ActionScript* [MOO 04]. Através desta linguagem, o *Flash* tem conquistado espaço no desenvolvimento de aplicações *web* com os mais diversos propósitos.

Basicamente, o funcionamento do sistema ocorre da seguinte forma: o usuário acessa o endereço *web* da aplicação e, após se conectar no sistema, é redirecionado para uma página desenvolvida em PHP. Esta página encaminha a aplicação *flash* a ser aberta pelo navegador e esta, após carregada, solicita automaticamente ao servidor o envio das informações atualizadas acerca dos fluxos medidos. A partir dos registros desses fluxos, a aplicação monta um diagrama com todas as máquinas que estão se comunicando no momento e seus respectivos destinatários.

De acordo com um intervalo de tempo pré-definido pelo sistema, a matriz de tráfego é atualizada, inserindo novas comunicações e removendo *hosts* que não possuem mais tráfego ativo. Para isso, a aplicação *flash* continuamente vai recebendo do servidor mensagens com o status da rede. Essa comunicação é realizada por meio de mensagens XML, sendo que um exemplo de mensagem enviada do servidor para a aplicação pode ser visualizado abaixo.

```
<?xml version="1.0" encoding="ISO8859-1"?>
<!DOCTYPE ipfix SYSTEM "traffic_matrix.dtd" >
<ipfix init_timestamp =" 1166198400" final_timestamp =" 1166198415">
      <devices>
            <device ip="143.54.1.7"></device>
            <device ip="143.54.29.38"></device>
            <device ip="143.54.29.40"></device>
            <device ip="143.54.29.41"></device>
            <device ip="143.54.29.143"></device>
            <device ip="200.144.121.33"></device>
      </devices>
      <flows>
            <flow device_a="143.54.1.7" device_b="143.54.29.40">
                  <bytes total="2083" delta="2083">
                  </bytes>
                  <packages total="19" delta="19">
                  </packages>
            </flow>
            <flow device_a="143.54.29.40" device_b="143.54.29.38">
                  <bytes total="137040" delta="480">
                  </bytes>
                  <packages total="179" delta="8">
                  </packages>
            </flow>
            <flow device_a="143.54.29.41" device_b="143.54.29.143">
                  <bytes total="3974150" delta="4090">
                  </bytes>
                  <packages total="13722" delta="12">
                  </packages>
            </flow>
            <flow device_a="143.54.29.41" device_b="143.54.29.38">
                  <bytes total="2446" delta="2446">
                  </bytes>
                  <packages total="24" delta="24">
                  </packages>
            </flow>
            <flow device_a="200.144.121.33" device_b="143.54.29.40">
                  <bytes total="2236590" delta="2880">
                  </bytes>
                  <packages total="24851" delta="32">
                  </packages>
            </flow>
      </flows>
</ipfix>
```

A matriz de tráfego prototipada detalha ao usuário a quantidade de dados em bytes trocados em cada sentido da comunicação, tanto em seu valor parcial (quantidade de bytes identificada durante a última avaliação realizada pelo dispositivo IPFIX), quanto o total de dados em bytes já transmitidos por aquele fluxo, em cada sentido da comunicação. Também é contabilizado o número de pacotes que formam essa comunicação.

Adicionalmente a essas informações, o sistema permite fazer uma avaliação do histórico detalhado de fluxos identificados pelo dispositivo IPFIX. Para isso, simplesmente o usuário deve clicar no *host* a ter seus fluxos avaliados e a aplicação solicita um relatório com diversos

dados estatísticos mostrados por meio de gráficos sobre o histórico de fluxos efetuados por este *host*. Embora isso não faça parte do conjunto de funcionalidades desempenhadas por uma matriz de tráfego, a análise de relatórios históricos traz grandes benefícios num sistema de monitoramento de redes.

Os gráficos que compõe o relatório histórico são enumerados a seguir:

- volume de tráfego total em *bytes*;
- total de pacotes contabilizados;
- distribuição do volume de tráfego por intervalo de tempo;
- *throughput*;
- distribuição percentual do tráfego durante o dia;
- percentual de uso sobre o tráfego total;
- distribuição de tráfego de acordo com os protocolos utilizados;
- distribuição dos pacotes de acordo com os protocolos utilizados;
- distribuição de tráfego de acordo com os serviços executados;
- distribuição dos pacotes de acordo com os serviços executados.

Esses gráficos podem ser visualizados segundo escalas de tempo distintas, como um dia, semana, mês ou ano específico. O APÊNDICE A mostra exemplos de cada um desses gráficos.

5 TESTES E ANÁLISE DOS RESULTADOS

Após a descrição de como foram implementados cada um dos elementos que compõe o protótipo da ferramenta de análise e fluxo, este capítulo destaca os testes de funcionamento realizados para cada um desses elementos. Eventuais alterações, com o objetivo de otimizar o funcionamento do protótipo desenvolvido, também são aqui detalhadas.

5.1 AMBIENTE DE TESTE

Os testes utilizaram dois cenários básicos, cada qual com propostas de avaliação distintas. Primeiramente, foram realizados testes em ambiente controlado, onde havia o controle de todos os fluxos que trafegavam através do ponto de observação, de forma a avaliar a precisão e exatidão das aferições executadas pelo observador IPFIX e a transmissão correta dos registros desses fluxos para o dispositivo coletor.

Em um segundo teste, foi verificado o desempenho do protótipo em ambiente real, objetivando principalmente avaliar seu comportamento a longo prazo e com um volume de dados relevante. As subseções seguintes detalham cada um desses cenários de testes.

5.1.1 Ambiente de teste em cenário controlado

Primeiramente, visando validar o processo de coleta e exportação dos dados de fluxo, optou-se pelo uso de um ambiente real controlado, com menor complexidade física e lógica, de modo a maximizar o controle acerca das variáveis envolvidas no processo de avaliação do sistema e a confiabilidade dos resultados obtidos.

O cenário de rede adotado para esta etapa de testes é mostrado na Figura 16 . Além disso, os equipamentos utilizados são detalhados nas Tabelas 3, 4, 5, 6 e 7.

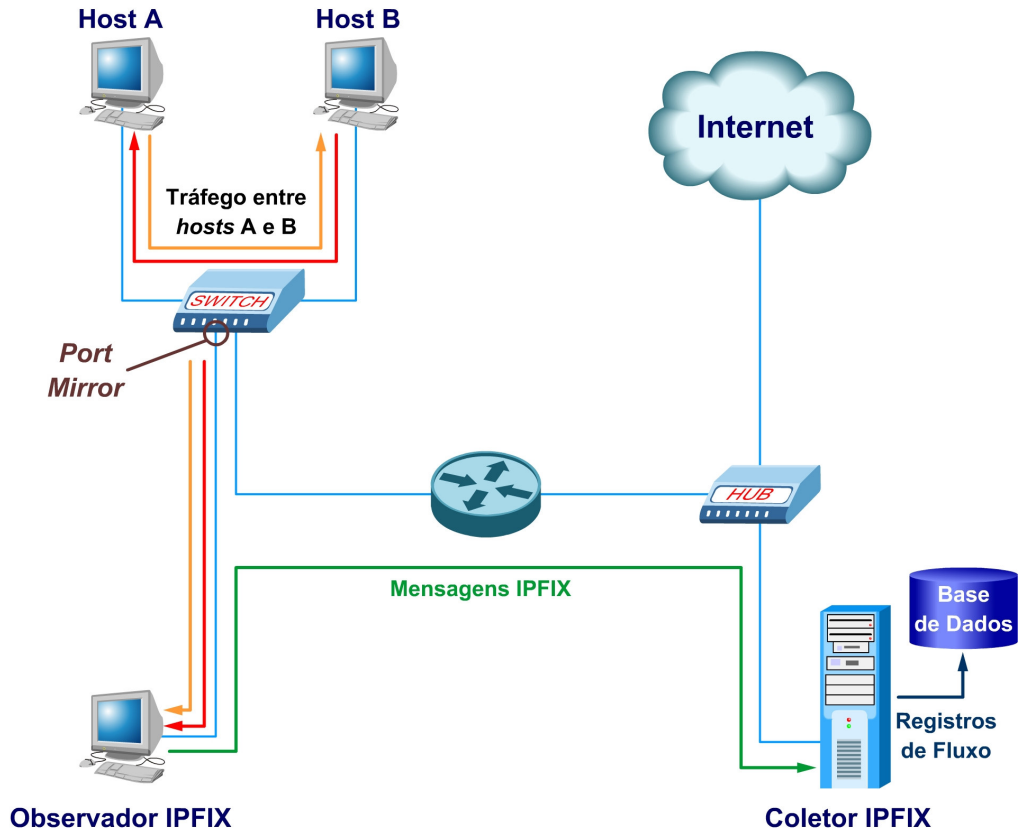


Figura 16 Cenário de testes em ambiente controlado.

Tabela 3 *Host´s* utilizados na comunicação e observador IPFIX.

Processador	*Pentium* 4 HT – 3.00 GHz
Memória	1 GBytes
Interface de Rede	*Fast Ethernet* 100 Mbps
Sistema Operacional	*Debian Linux – Kernel* 2.6.12.1

Tabela 4 Coletor IPFIX.

Processador	*Intel(R) Xeon(TM)* - 3.00 GHz
Memória	2 GBytes
Interface de Rede	*Gigabit Ethernet* 1 Gbps
Sistema Operacional	*Debian Linux - Kernel* 2.6.12.1 i686

Tabela 5 Características do *hub.*

Modelo	ESH – 709 / 10 Mbps
Fabricante	*Encore*
Número de portas	8 portas operando a 10Mbps (*Ethernet*)
Modo de operação	*half-duplex*

Tabela 6 Características do *switch.*

Modelo	*Switch* 3250 *SuperStack* 3
Fabricante	*3Com*
Número de portas	48 portas de 100 Mbps e 2 portas de 1 Gbps (*Ethernet*)
Modo de operação	*full-duplex*

Tabela 7 Características do roteador.

Modelo	*Cisco 3600*
Fabricante	*Cisco Systems*
Modo de operação	*full-duplex*

Do modo como foi implementado, os testes se baseiam na análise do tráfego gerado entre os *hosts* A e B. Para isso, conforme mostrado na Figura 16 , é utilizada a capacidade de espelhamento de porta (*port mirror*) do *switch*, de modo que todo tráfego executado entre as duas máquinas seja replicado para o ponto de observação ao qual o dispositivo IPFIX está associado. Este, por sua vez, envia, a cada intervalo de tempo pré-determinado, as mensagens IPFIX para o coletor, que armazena os respectivos resultados na base de dados.

Para simulação dos fluxos entre os *hosts*, utilizou-se o aplicativo para transferência de arquivos *wget*, que funciona através do protocolo HTTP, de modo a simular comunicações executadas através do TCP. Cada comunicação utilizava arquivos de teste com tamanhos que variam de 500 KBytes a 800MBytes. Além desse método, utilizou-se também o aplicativo *udpflood*, desenvolvido no GPARC&TI, que gera um tráfego de teste UDP com características pré-definidas.

A partir do cenário de rede e das aplicações utilizadas para simulação de tráfego, diferentes rajadas de pacotes foram disparadas, de modo a verificar a acuracidade do sistema de medição de fluxos. Os testes realizados utilizando esta metodologia e os resultados obtidos compõem a subseção 5.2.1.

5.1.2 Testes em ambiente real

Para os testes em ambiente real, optou-se por inserir um ponto de observação de modo a observar o tráfego da própria rede utilizada pelo GPARC&TI. A Figura 17 mostra a disposição do cenário de testes em ambiente real. Os equipamentos utilizados nesse cenário são os mesmos descritos nas Tabelas 4 até 8.

Como pode ser observado na Figura 17 , o segmento de rede ao qual está associado o ponto de observação foi escolhido de tal forma que qualquer tráfego existente entre os servidores do GPARC&TI, as estações de trabalho e a rede externa é captado pelo observador e armazenado pelo coletor. Além disso, para tornar o envio de mensagens IPFIX e o armazenamento dos registros de fluxo oriundos dessas mensagens tarefas mais críticas, utilizou-se um sistema com período de amostragem de baixa granularidade, especificando uma periodicidade de apenas 5 segundos no envio das mensagens IPFIX.

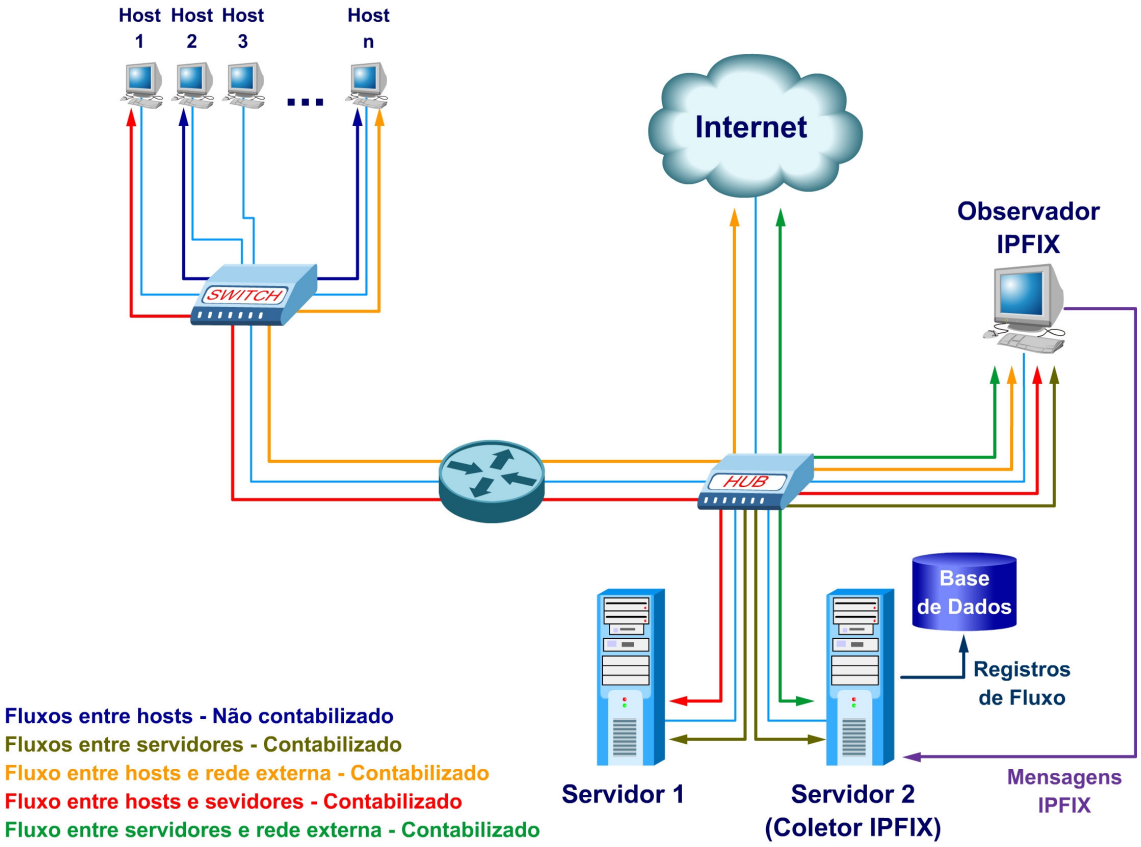


Figura 17 Cenário de testes em ambiente real.

Finalmente, para verificar o comportamento do protótipo de medição de informações de fluxo a longo prazo, a coleta desses dados foi realizada em um período ininterrupto de 1 mês. Nesse período, além do tráfego normal executado na rede, foram realizados também testes de envio controlado de tráfego de teste através do ponto de observação, de modo a verificar a consistência dessas medições no decorrer do tempo. Os testes adotados a partir desse cenário, bem como os seus resultados, são mostrados na subseção 5.1.2.

5.2 TESTES REALIZADOS

A presente seção descreve os resultados obtidos para os dois cenários de teste descritos na seção anterior.

5.2.1 Testes em ambiente controlado

A partir do cenário de rede em ambiente real controlado apresentado, foram gerados diferentes modelos de fluxo, de modo a verificar o funcionamento do módulo de medição do observador IPFIX e da comunicação deste com o coletor, no que diz respeito à concordância dos valores apresentados nos registros de fluxo armazenados e os valores dos fluxos efetivamente transmitidos.

Para isso, primeiramente, foram realizados testes utilizando a transmissão de um único fluxo por vez, de modo a simplificar a avaliação do comportamento do sistema. Em seguida, foi realizada a transmissão de fluxos concorrentes, verificando se o sistema conseguia aferir cada um dos fluxos corretamente.

Utilizando o protocolo TCP, através do aplicativo *wget*, foram enviados os arquivos apresentados na Tabela 8 , com o teste sendo repetido oito vezes para cada um destes arquivos.

Tabela 8 Arquivos utilizados no primeiro conjunto de testes.

Arquivos	Tamanho (MBytes)	Número de Octetos
Arquivo 1	0,5	512000
Arquivo 2	1	1048576
Arquivo 3	10	10485760
Arquivo 4	50	52428800
Arquivo 5	100	104857600
Arquivo 6	500	524288000
Arquivo 7	800	838860800

Pelo fato de tratar-se de uma comunicação TCP, é importante destacar que cada um dos arquivos enviados gera um fluxo bidirecional. Entretanto, foram analisadas apenas as informações no sentido de *downstream*. Portanto, os dados gerados pelo controle de transmissão do TCP são desconsiderados nos testes realizados.

Os resultados médios para cada fluxo são apresentados na Tabela 9 , sendo que a sumarização completa de cada teste executado pode ser verificada no Apêndice B.

Tabela 9 Resultados obtidos para cada um dos arquivos enviados utilizando o *wget.*

Arquivos	Tamanho do Arquivo (octetos)	Octetos contabilizados*	Pacotes contabilizados*
Arquivo 1	512000	536049	361
Arquivo 2	1048576	1097155	732
Arquivo 3	10485760	10958103	7281
Arquivo 4	52428800	54816287	36383
Arquivo 5	104857600	109548899	72856
Arquivo 6	524288000	547245480	364750
Arquivo 7	838860800	876413645	587099

*Valores Médios (média aritmética).

Conforme pode ser visto nesta tabela, o número de bytes total que compõe o fluxo (volume total de tráfego transmitido) excede o tamanho efetivo de cada arquivo transmitido. Entretanto, este comportamento era esperado por ser realizada a contabilização do tamanho total de cada pacote que compõe o fluxo, incluindo todo o *overhead* imposto pelas informações de controle. Nesse contexto, são contabilizados os cabeçalhos inseridos desde a camada de aplicação até a camada de enlace. Desta forma, os resultados obtidos incluem os cabeçalhos oriundos do *Ethernet*, IP, TCP e HTTP (*wget*). O *overhead* imposto por cada um destes protocolos é mostrado na Tabela 10 .

Tabela 10 Tamanho dos cabeçalhos de cada protocolo utilizado.

Protocolo	*Overhead* inserido
Ethernet	14 bytes por pacote
IP	20 bytes por pacote
TCP	20~32 bytes por pacote
HTTP (*wget*)	≈512 bytes por comunicação

Como pode ser observado nessa tabela, o sistema contabiliza o cabeçalho *ethernet*, descontando o *checksum* desse protocolo. Por sua vez, o cabeçalho TCP varia de acordo com o uso dos seus campos opcionais, possuindo até 32 bytes. Finalizando, o cabeçalho HTTP é

composto pelas informações geradas pelo aplicativo *wget*, variando de acordo com os parâmetros utilizados. Assim, são gerados durante a transmissão do arquivo, aproximadamente 512 bytes com informações padrão do cabeçalho do protocolo HTTP. Adicionalmente, também durante a transmissão, o aplicativo *wget* gera informações para controle e estatísticas da transmissão, incrementando desta forma o *overhead* total do fluxo.

De posse dos resultados aferidos durante as medições, apontados na Tabela 9 , é possível inferir o *overhead* adicionado em cada pacote, o quanto este *overhead* representa em termos percentuais na transmissão do fluxo e o tamanho médio de cada pacote, em número de octetos. Esses valores são destacados na Tabela 11 .

Tabela 11 Dados estatísticos obtidos com base na Tabela 9 .

Arquivos	*Overhead* médio por pacote (bytes)	*Overhead* médio (%)	Tamanho médio dos pacotes (octetos)
Arquivo 1	66,71	4,49	1485
Arquivo 2	66,35	4,43	1499
Arquivo 3	64,87	4,31	1505
Arquivo 4	65,62	4,36	1507
Arquivo 5	64,39	4,28	1504
Arquivo 6	62,94	4,19	1500
Arquivo 7	63,95	4,28	1493

De posse das informações expostas na Tabela 10 , podemos inferir o *overhead* mínimo por pacote, para cada fluxo. Para isso, basta realizar o somatório de todas as informações de controle esperadas em cada pacote. No caso do cabeçalho HTTP, basta que seu valor seja dividido pelo número de pacotes. A equação (1) mostra esse cálculo.

$$overhead = (l_{Ethernet} + l_{IP} + l_{TCP}) + \frac{l_{HTTP}}{n} \qquad (1)$$

onde: l = tamanho do cabeçalho
n = número de pacotes

A partir da equação (1), pode-se deduzir número mínimo de bytes de *overhead*. Aplicando esse cálculo, chega-se a um *overhead* mínimo variando de 55 bytes para os menores arquivos e 54 bytes para os arquivos de maior tamanho. Entretanto, os valores mais altos encontrados nos testes realizados são explicados por diversos fatores, como a existência de pacotes transmitidos com cabeçalhos TCP maiores, chegando a 32 bytes e, principalmente, pela transmissão de pacotes de controle efetuada pelo *wget*, incrementando dessa forma o *overhead* total do fluxo.

Por sua vez, o valor do MTU (*Maximum Transfer Unit*), que nos testes era de 1500 bytes serve de base para o cálculo do *overhead* mínimo percentual que cada pacote pode apresentar. Para isso basta calcular o caso ótimo, onde todos os pacotes possuem carga útil igual ao MTU, incluindo aí o espaço reservado para os cabeçalhos IP, TCP e HTTP. Esse cálculo é apresentado na Equação (2).

$$overhead_{\min}(\%) = 100 - \left(\frac{1500 - \left(l_{IP} + l_{TCP} + \frac{l_{HTTP}}{n} \right)}{1500 + l_{Ethernet}} \cdot 100 \right) \qquad (2)$$

onde: l = tamanho do cabeçalho
n = número de pacotes

Com base nesse cálculo é possível encontrar um *overhead* mínimo variando entre 3,57% para o maior arquivo até 3,66% ara o arquivo de menor tamanho. Da mesma forma que anteriormente, o overhead mínimo encontrado deverá possuir um valor maior que este, devido aos pacotes de controle enviados pelo *wget* e também porque diversos pacotes enviados não possuem 1500 bytes de carga útil.

Esses testes comprovaram que os valores encontrados até o momento eram aceitáveis, entretanto não verificavam o funcionamento do dispositivo observador. Embora este estivesse transmitindo os dados de forma correta, seu processo de medição poderia ser falho, dado o grau de incerteza existente nas medições realizadas, gerado tanto pelo tráfego de controle de conexão gerado pelo TCP quanto pelo tráfego de controle gerado pelo *wget*.

Para aferir o sistema com maior precisão, foi necessário utilizar a ferramenta *udpflood*. Desenvolvida pelo GPARC&TI, essa ferramenta possibilita simular um tráfego UDP através da determinação de seus parâmetros básicos. Esse tráfego é determinado pelo número de pacotes, o tamanho de cada um destes e o intervalo entre eles, o endereço de destino, e as portas de origem e destino. A partir deste aplicativo, torna-se possível modelar diferentes padrões de tráfego UDP.

O teste inicial foi usado com o *udpflood* trabalhando com a seguinte configuração:

- número de pacotes: 1000;
- intervalo entre os pacotes: 100μs;
- tamanho dos pacotes: 100000 bytes.

Realizados os testes, o fluxo armazenado apresentava os seguintes valores:

- total de pacotes: 1000;
- volume de tráfego do fluxo: 104200000 bytes.

A Figura 18 e a Figura 19 mostram o comportamento da transmissão, aferido durante o teste. Para melhor entendimento destas figuras deve-se destacar que o observador foi configurado de forma a enviar suas mensagens a cada 10 segundos. Esse teste foi repetido apresentando volume de tráfego e total de pacotes idênticos, indicando a precisão do sistema. Levando em conta o *overhead* imposto pela comunicação, devido aos cabeçalhos dos protocolos *Ethernet*, IP e UDP, percebe-se que o resultado obtido na medição foi correto em todas as repetições realizadas para este teste. Esse cálculo pode ser obtido através de equação semelhante a equação (1), com a diferença de que o protocolo UDP insere um cabeçalho de apenas 8 bytes em lugar do cabeçalho TCP e a aplicação *udpflood* não insere qualquer informação de controle na camada de aplicação. Essa equação e o resultado do cálculo são mostrados abaixo.

$$overhead_{total} = (l_{Ethernet} + l_{IP} + l_{UDP}) \cdot n \qquad (3)$$

onde: l = tamanho do cabeçalho

n = número de pacotes

$$overhead_{total} = (14 + 20 + 8) \cdot 100000$$
$$overhead_{total} = 4200000\,\text{bytes} \quad (4)$$

Somando o valor de *overhead* encontrado com o numero total de bytes a ser transmitido, comprovou-se a exatidão dos valores encontrados. A repetição desse teste com outros modelos de tráfego UDP, sempre mantendo a exatidão das medidas indica um grau de confiabilidade da ferramenta.

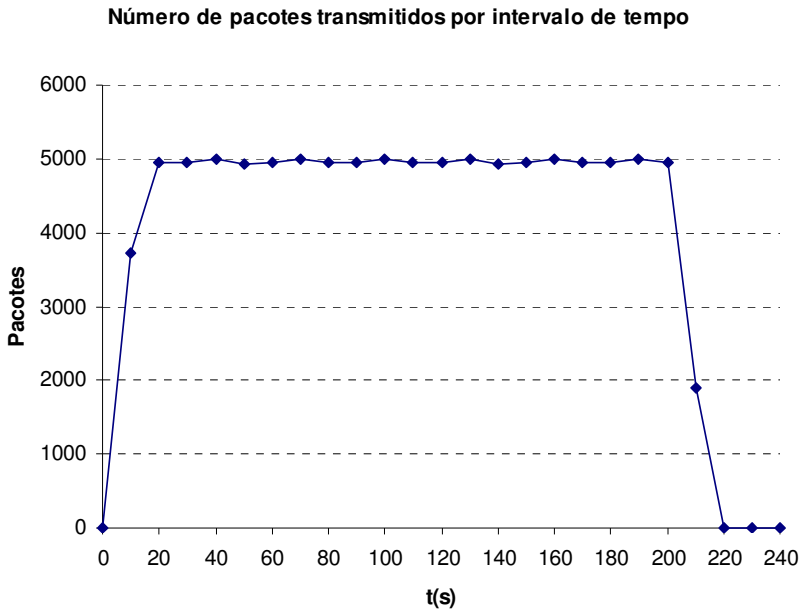


Figura 18 Variação de pacotes por intervalo de tempo.

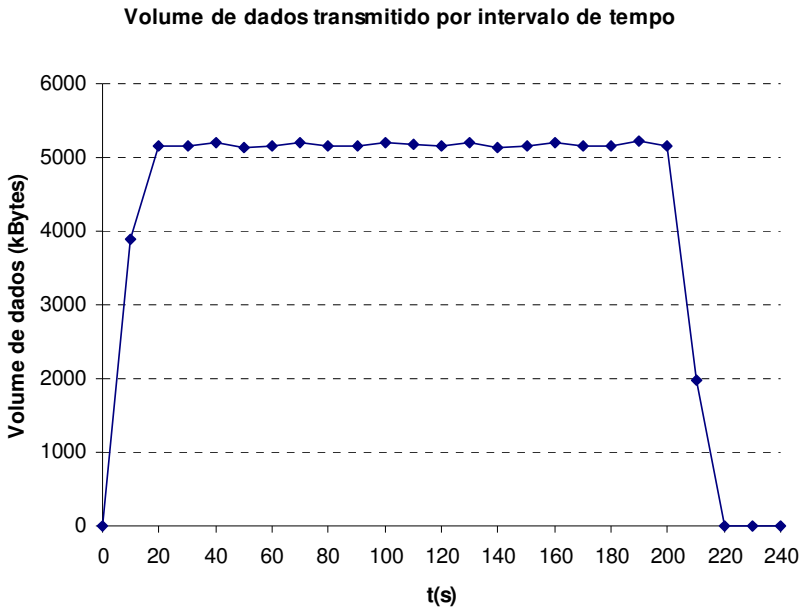


Figura 19 Volume de dados aferido por intervalo de tempo.

Os testes até aqui descritos demonstram a capacidade do sistema em aferir um fluxo com valores coerentes em relação aos valores esperados. Entretanto, esses testes foram realizados sem a presença de tráfego concorrente. O teste seguinte foi executado para verificar a eficiência do sistema quando medindo fluxos paralelos.

Para isso, foram enviados dois fluxos distintos do *host* A até o *host* B. Os parâmetros dos fluxos são destacados na Tabela 12 .

Tabela 12 Parâmetros dos fluxos concorrentes.

Parâmetros	Fluxo 1	Fluxo 2
Número de pacotes	30000	10000
Tamanho de cada Pacote (bytes)	1000	1000
Intervalo entre cada Pacote (µs)	100	100

Novamente, o sistema manteve coerência nas medições para ambos os fluxos, com os resultados sendo apresentados na Tabela 13 .

Tabela 13 Resultados obtidos com o teste de fluxos concorrentes.

Fluxos Identificados	Total de Pacotes	Volume de Dados (kbytes)
Fluxo 1	30000	31260
Fluxo 2	10000	10420

Após os testes efetuados em ambiente real controlado, o protótipo foi instalado na rede do GPARC&TI, visando medir o tráfego desta a longo prazo.

5.2.2 Testes em ambiente real

Conforme destacado na subseção 5.1.2, o ponto de observador do dispositivo IPFIX foi alocado no segmento da rede com maior fluxo de dados. Além disso, o dispositivo enviava mensagens IPFIX com uma baixa granularidade, em intervalos de 5 segundos, gerando um grande número de mensagens IPFIX, visando aferir a capacidade de processamento dessas mensagens pelos módulos do protótipo.

Além disso, de forma periódica, foram realizadas coletas enviando um tráfego de teste, de modo a avaliar o bom funcionamento do sistema. Nesses experimentos, era verificada a conformidade dos valores medidos para o fluxo gerado com os valores reais conhecidos.

Nesse ponto, passou a ser utilizada também a aplicação de matriz de tráfego, de modo a obter a visualização de todos os fluxos concorrentes em um intervalo de tempo. A Figura 20 mostra um exemplo de resultado obtido a partir do uso desta aplicação.

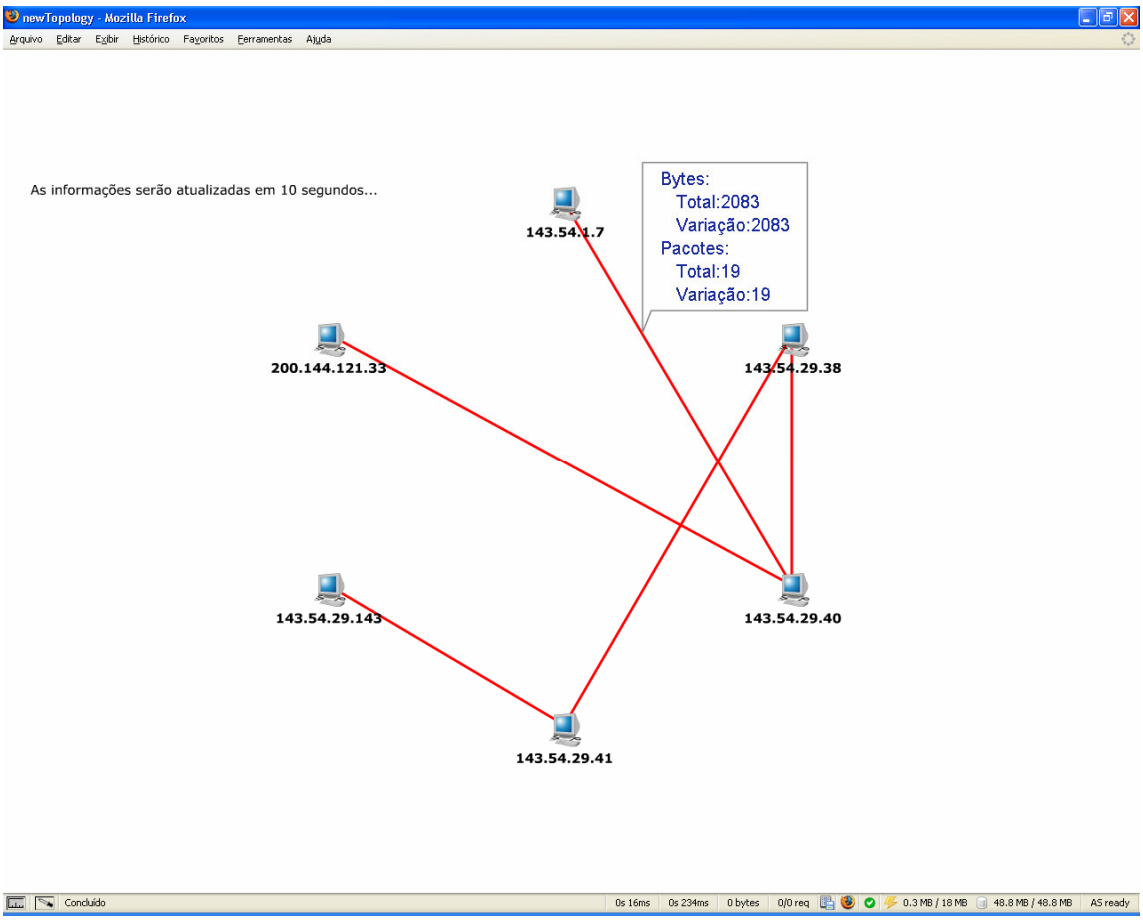


Figura 20 Exemplo de matriz de tráfego visualizada através do sistema.

Conforme comentado na seção 4.3, adicionalmente a matriz de tráfego foi implementado um sistema de relatórios baseados nos dados históricos de fluxos. Esse sistema também foi bastante útil por permitir a rápida visualização de dados armazenados na base do coletor.

No que diz respeito às medições de fluxo realizadas, o sistema se manteve funcionando de forma satisfatória, com os valores aferidos para os fluxos de teste permanecendo concordantes com os valores esperados. Por outro lado, com o passar dos dias, percebeu-se que a quantidade de informações continuamente armazenada na base de dados causou um grande povoamento nesta, fazendo com que o sistema tivesse uma queda acentuada no desempenho, quando na consulta desses dados.

Essa queda no desempenho é inevitável utilizando o sistema com uma granularidade tão baixa sem o uso de nenhuma otimização na base ou no método de coleta dos dados de fluxo. Um fluxo que dure 1 minuto gera a inserção de, pelo menos, 12 entradas na tabela de dados e na tabela *ipfix_messages*, muitas delas com informações redundantes. Ao final de apenas 15 dias, o ambiente de teste gerou mais de 1550000 entradas, em cada uma dessas tabelas, comprovando que, em se tratando de sistemas responsáveis pelo monitoramento de redes de tráfego mais intenso ou que necessitem do envio de informações de tráfego em uma menor granularidade, faz-se necessário a realização de alterações na estrutura funcional do sistema. Além disso, verificou-se o alto grau de redundância de informações nos registros armazenados. Da forma como foi implementada, a base de dados armazena repetidamente uma série de informações. Um exemplo claro deste processo está no armazenamento dos dados que compõe a chave de um fluxo. Apesar de, sabidamente, um fluxo possuir sempre esses valores-chave idênticos, da forma como a base foi proposta, a cada nova inserção de dados de um mesmo fluxo, este terá seus campos-chave novamente armazenados na tabela de dados.

Outra desvantagem da modelagem da base de dados utilizada esta relacionada com as consultas efetuadas. Como não existe um identificador comum para as diferentes entradas de um mesmo fluxo, faz-se necessário o uso de consultas aninhadas na identificação de entradas de um fluxo. Primeiramente, essas consultas resultam num vetor com os identificadores das mensagens correspondentes a um determinado fluxo, obtido na tabela *ipfix_messages*. Esses identificadores são, então, utilizados para encontrar as informações aferidas sobre o fluxo em questão. Além disso, torna-se complexo o tratamento de fluxos com mesmos campos-chave, porém espaçados no tempo. Por exemplo, supondo a existência de um fluxo no TCP no inicio da manhã entre uma máquina origem de IP 192.168.0.5 e uma maquina destino com IP 192.168.0.9, através das portas de origem 32100 e destino 80, este fluxo será inserido normalmente na tabela de dados correspondente. Da mesma forma, um fluxo idêntico a esse ocorrendo ao final da noite será registrado corretamente. Entretanto, uma consulta normal a essa base traria esses dois conjuntos de informações como pertencendo a um fluxo único, já que, do modo como o sistema foi definido, não são armazenadas informações que indiquem o término do primeiro fluxo e o inicio do segundo.

Toda essa complexidade associada ao sistema de armazenamento foi proposta pelos desenvolvedores da *libipfix* com o intuito de tornar o coletor capaz de armazenar todos os

elementos de informação de qualquer modelo de *template* que possa ser recebido por ele. Trata-se de uma grande vantagem do sistema, visto que o coletor torna-se autônomo em relação a eventuais mudanças no conjunto de elementos de informação de interesse. Entretanto, em aplicações como a proposta neste trabalho, onde não existe a pretensão de se fazer alterações no conjunto de elementos de informação armazenados, esse cuidado perde o sentido. Em um caso como esse, uma mudança na modelagem da base traria grande otimização ao sistema. Essa mudança é sugerida na seção seguinte.

5.3 ALTERAÇÕES IMPLEMENTADAS A PARTIR DOS RESULTADOS OBTIDOS NOS TESTES

Conforme visto, a principal alteração no protótipo implementado, cuja necessidade foi percebida a partir dos testes realizados, é a otimização na base de dados do sistema, de modo a dinamizar as consultas nela realizadas. Para isso, se fez necessária a criação de um mecanismo que simplifique esse processo. Além disso, esse mecanismo deve ser capaz de identificar cada fluxo utilizando uma identificação relacionada com o tempo, de modo a permitir que dois fluxos com mesmas chaves, porém espaçados temporalmente, sejam identificados como fluxos distintos. Para isso, uma modificação nos elementos de informação enviados ao coletor teve de ser realizada e, consequentemente, isso resultou em mudanças no observador IPFIX.

A alternativa adotada para melhorar o método de identificação dos fluxos na base de dados foi a adição de um novo elemento de informação aos registros de fluxo enviados. Esse elemento, denominado de *flowIdentifier*, é uma chave *hash* formada a partir das informações-chave do fluxo (IP origem e destino, porta de origem e destino e protocolo da camada de transporte) e do *timestamp* do primeiro pacote identificado, pertencente ao fluxo em questão.

Assim, o processo de identificação dos pacotes no observador IPFIX permanece o mesmo. Entretanto, quando um novo registro de fluxo é criado pelo módulo de medição, automaticamente ele recebe um novo valor para o *flowIdentifier*.

Esse novo elemento de informação é fundamental para os processos realizados no SGBD. Ele torna-se o a chave utilizada na identificação dos dados aferidos para cada fluxo. A partir do *flowIdentifier*, a identificação dos fluxos na base de dados foi bastante simplificada.

Além dessa modificação, levando em conta os elementos de informação necessários a aplicação já estão definidos e a inserção de novos elementos não será uma tarefa rotineira, é possível trabalhar com uma tabela de dados fixa, ao invés da criação dinâmica de tabelas de dados a partir da chegada de novos *templates*. Essa foi a principal simplificação realizada no sistema, o que diminuiu consideravelmente sua complexidade. A Figura 21 mostra o diagrama ER resultante da aplicação dessas modificações.

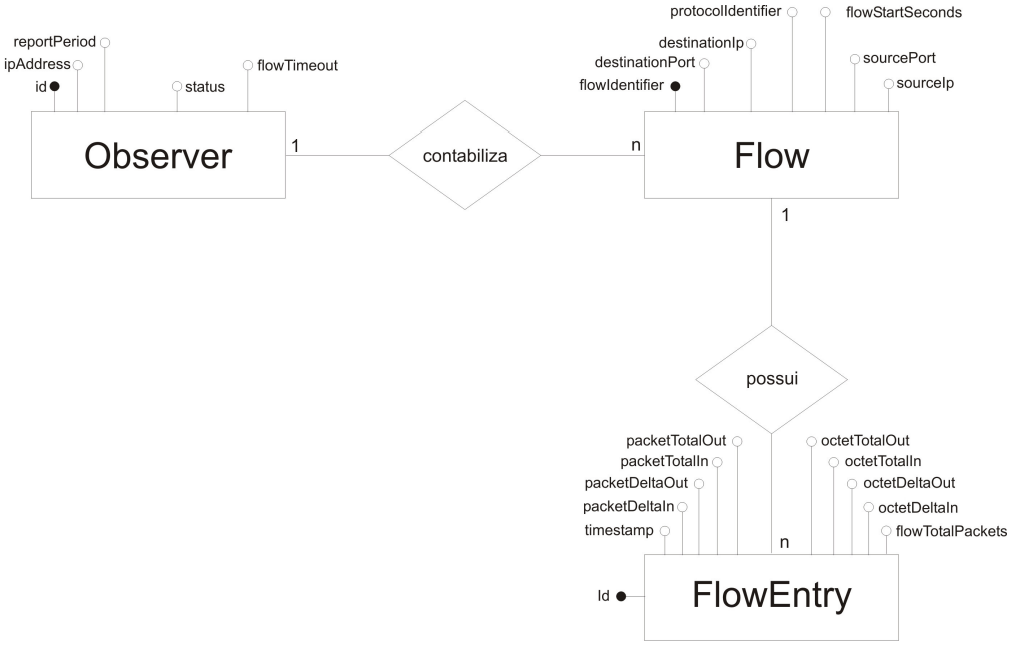


Figura 21 Diagrama ER do protótipo otimizado.

Todas essas modificações não influem no processo de medição e no sistema de exportação dos dados de fluxo, fazendo com que a eficiência destes não seja comprometida. Em compensação, o sistema de aquisição de consulta a base de dados obteve sensível melhora de desempenho.

6 CONCLUSÕES

Este capítulo tem por objetivo apresentar as conclusões relacionadas ao trabalho desenvolvido, bem como algumas sugestões para trabalhos futuros.

O presente trabalho destacou a importância da medição baseada em fluxos e os esforços do grupo de trabalho IPFIX no desenvolvimento de uma arquitetura de medição padronizada, que sirva como base para o desenvolvimento de novas ferramentas de aquisição e tratamento das informações de fluxo.

A arquitetura, ainda em estágio de pesquisa, demonstra diversos aspectos interessantes, como, por exemplo, a capacidade de usar tanto múltiplos observadores de fluxo quanto múltiplos coletores. Um grande benefício do IPFIX a ser destacado é a habilidade do protocolo de comunicação de, através do uso de *templates*, tornar a informação transportada autocontida, ou seja, a partir da interpretação dos *templates* transportados qualquer informação enviada pode ser identificada. A definição de novos elementos de informação de acordo com as necessidades do sistema de análise de dados fica simplificada e, mesmo os coletores que não possuem a capacidade de tratar esse novo elemento de informação, podem receber as mensagens IPFIX e extrair somente as informações que estejam aptos a tratar.

A implementação do dispositivo IPFIX mostrou resultados satisfatórios, identificando e aferindo os fluxos de forma precisa e exata e apresentando durante o período de testes um funcionamento estável. Merece destaque neste aspecto o sistema de exportação baseado na libipfix, capaz de enviar grande tráfego de informações de fluxo em intervalos de tempo de relativamente pequenos sem apresentar problemas, indicando que, apesar de ser uma biblioteca em estágio de implementação, já indica um grau de confiabilidade satisfatório.

Por outro lado, comprovou-se a necessidade de avaliar cuidadosamente a implementação do coletor, de modo a tornar este em conformidade com as necessidades da aplicação utilizada.

O coletor IPFIX, da forma como foi implementado primeiramente, consegue armazenar qualquer tipo de *template*, entretanto isso torna sua base de dados complexa e, por

conseqüência, lenta. Além disso, esse armazenamento possui alto grau de redundância nos campos, especialmente quando se tratando de dados de um mesmo fluxo. Em contrapartida, a implementação otimizada do coletor aumenta a eficiência da base de dados, porém esta se torna mais rígida, impedindo que novos elementos de informação sejam armazenados sem que sejam realizadas alterações prévias em sua modelagem.

Dessa forma, é indicado que o módulo de armazenamento de dados do coletor seja definido somente após uma avaliação detalhada em relação aos requerimentos necessários para o funcionamento da aplicação de interesse. Para sistemas que não necessitam de uma base histórica de informações de fluxo extensa, a primeira implementação é mais indicada visto que, em caso de alterações nos elementos de informação utilizados pelo sistema, os dispositivos IPFIX e o coletor não precisariam sofrer qualquer modificação além da reconfiguração dos *templates* utilizados. Além disso, essa implementação também é indicada no caso do coletor trabalhar com *templates* distintos.

Por outro lado, aplicações que exijam o armazenamento de informações de aferições passadas necessitam de bases de dados mais robustas e eficientes. Nesses casos, faz-se necessária a implementação de um coletor com uma base de dados semelhante a segunda implementação apresentada, onde um modelo de armazenamento de dados mais rígido implica em um aumento na eficiência do sistema.

6.1 TRABALHOS FUTUROS

A partir das considerações feitas acerca do trabalho, propõe-se como possibilidade de trabalhos futuros a implementação de métodos que visem à otimização do sistema de aquisição e armazenamento de dados de fluxo, principalmente para sua utilização em redes de maior capacidade. Entre esses mecanismos, podem ser citados métodos de otimização de base de dados, uso de visões nas mesmas, índices e mecanismos de tabela temporária e de rotatividade.

Outra possibilidade amplamente estudada atualmente é o uso de métodos de amostragem na coleta de dados, tanto para diminuir a carga de processamento no dispositivo IPFIX quanto para minimizar o tráfego entre este e o dispositivo coletor. Dependendo do método de amostragem utilizado, é possível inclusive diminuir a carga na base de dados.

Outro foco para o desenvolvimento de novos trabalhos está na própria análise das informações de fluxo coletadas. A partir do sistema desenvolvido, torna-se viável uma grande quantidade de pesquisa nas mais diversas áreas, desde métodos de detecção de ataques até técnicas de inteligência artificial que visam predizer o comportamento da rede, utilizadas de modo a compor um sistema de gerenciamento proativo.

REFERÊNCIAS

[AMS 97] AMSDEN, P. et al. "**Cabletron's Light-weight Flow Admission Protocol Specification Version 1.0**". RFC 2124, Março de 1997.

[AND 06] ANDO, T.; OKAMURA, H.; DOHI, T. "**Estimating Markov Modulated Software Reliability Models via EM Algorithm**". Dependable, Autonomic and Secure Computing, 2nd IEEE International Symposium, Setembro de 2006.

[AWD 02] AWDUCHE, D.; CHIU, A.; ELWALID, A.; WIDJAJA, I.; XIAO, X. "**Overview and Principles of Internet Traffic Engineering**". IETF - RFC 3272, Maio de 2002.

[BIR 01] BIRBECK, M. et al. "**Professional XML**". Ciência Moderna, 1ª Edição. 2001. 1266 p.

[BLA 98] BLAKE, S. "**An Architecture for Differentiated Services**". RFC 2475, Dezembro de 1998.

[BOS 06] BOSCHI, E. et al. "**IPFIX Implementations Guidelines**". Draft IEEE, Outubro de 2006. Disponível em <http://www.ietf.org/internet-drafts/draft-ietf-ipfix-implementation-guidelines-01.txt> em Janeiro de 2007.

[BRA 97] BRADEN, R. et al. "**Resource ReSerVation Protocol (RSVP)**". RFC 2205, Setembro de 1997.

[BRO 01] BROWNLEE, N. "**Using NeTraMet for Production Traffic Measurement**". *Integrated Network Management Proceedings*, *IEEE/IFIP International Symposium*. Maio de 2001.

[BRO 97a] BROWNLEE, N.; MILLS, C.; RUTH G. **"Traffic Flow Measurement: Architecture"**. RFC 2063, Janeiro de 1997.

[BRO 97b] BROWNLEE, N. **"Traffic Flow Measurement: Meter MIB"**. RFC 2064, Janeiro de 1997.

[BRO 97c] BROWNLEE, N. **"Experiences with NeTraMet"**. RFC 2123, Março de 1997.

[BRO 99a] BROWNLEE, N. **"Traffic Flow Measurement: Meter MIB"**. RFC 2720, Outubro de 1999.

[BRO 99b] BROWNLEE, N. **"RTFM: Applicability Statement"**. RFC 2721, Outubro de 1999.

[BRO 99c] BROWNLEE, N.; MILLS, C.; RUTH G. **"Traffic Flow Measurement: Architecture"**. RFC 2722, Outubro de 1999.

[BRO 99d] BROWNLEE, N. **"SRL: A Language for Describing Traffic Flows and Specifying Actions for Flow Groups"**. RFC 2723, Outubro de 1999.

[CAL 03] CALHOUN, P. et al. "**Diameter Base Protocol**". RFC 3588, Setembro de 2003.

[CAL 05] CALYAM, P.; LEE, C.; ARAVA, P. K.; KRYMSKIY D.; et al. "**OnTimeMeasure: A Scalable Framework for Scheduling Active Measurements".** End-to-End Monitoring Techniques and Services. IEEE Workshop, Maio de 2005.

[CAR 02] CARPENTER B. et al. "**Middleboxes: Taxonomy and Issues**". RFC 3234,

Fevereiro de 2002.

[CAS 00] CASTAGNETTO, J. et al. "**Professional PHP Programando**". Makron Books, 1ª Edição. 2000. 770 p.

[CAS 90] CASE, J.; FEDOR M.; SCHOFFSTALL M. et al. "**A Simple Network Management Protocol (SNMP)**". RFC 1157, Maio de 1990.

[CHO 04] CHOI, B.; PARK, J.; ZHANG, Z. "**Adaptive Packet Sampling for Accurate and Scalable Flow Measurement**". GLOBECOM'04, IEEE Communications Society, Dezembro de 2004.

[CHO 06] CHOI, B.; ZHANG, Z. "**Adaptive random sampling for traffic volume measurement**". Telecommun Syst, Springer Science. Dezembro de 2006.

[CHU 95] CHU, P.P. "**ATM Burst Traffic Generator**". VLSI Proceedings. 5º Great Lakes Symposium, Março de 1995.

[CIS 03] CISCO. "**Internetworking Technologies Handbook**". Cisco Systems, 2003. 1079p.

[CIS 04] CISCO. "**Cisco IOS NetFlow Version 9 Flow-Record Format**". Cisco Systems, 2004. Disponível em <http://www.cisco.com/application/pdf/en/us/guest/tech/tk362/c1550/ccmigration_09186a00800a3db9.pdf> em janeiro de 2007.

[CIS 06a] CISCO. "**Network Management System: Best Practises White Paper**". Cisco System, 2006. Disponível em <http://www.cisco.com/warp/public/126/NMS_bestpractice.pdf> em janeiro de 2007.

[CIS 06b] CISCO. "**Introduction to Cisco IOS® NetFlow—A Technical Overview**". Cisco Systems, 2006. Disponível em <http://www.cisco.com/application/pdf/en/us/guest/products/ps6601/c1244/cdccont_0900aecd80406232.pdf> em janeiro de 2007.

[CLA 04] CLAISE, B. "**Cisco Systems NetFlow Services Export Version 9**". RFC 3954, Outubro de 2004.

[CLA 06] CLAISE, B. "**Specification of the IPFIX Protocol for the Exchange of IP Traffic Flow Information**". Draft IETF, Novembro de 2006. Disponível em <http://www.ietf.org/internet-drafts/draft-ietf-ipfix-protocol-24.txt> em janeiro de 2007.

[CLA 93] CLAFFY, K. C.; POLYZOS, G. C.; BRAUN H. "**Application of Sampling Methodologies to Network Traffic Characterization**". Proceedings ACM SIGCOMM'93, San Francisco, CA, Setembro de 1993.

[COS 05] COSTA, R. I. T. "**Uma Arquitetura Para Apoio À Engenharia de Tráfego De Dados Nos Serviços De Comutação De Pacotes Dos Sistemas Celulares**". 2005. 176 f. Dissertação (Mestrado em Engenharia Elétrica) – PPGEE, PUCRS, Porto Alegre, 2005.

[DAR 81] DARPA."**Transmission Control Protocol Darpa Internet Program Protocol Specification**". RFC 793, Setembro de 1981.

[DER 00] DERI, L.; SUIN, E.; "**Effective Traffic Measurement Using ntop**". IEEE Communications Magazine, Maio de 2000.

[DIE 06] DIERKS, T; ALLEN, C. "**The TLS Protocol Version 1.1**", RFC 4346, Abril de 2006.

[DIE 99] DIERKS, T.; ALLEN, C. "**The TLS Protocol Version 1.0**". RFC 2246, Janeiro de 1999.

[DOV 01] DOVROLIS, P. R.; MOORE, D. "**What do Packet Dispersion Techniques Measures?**". IEEE Infocom, 2001.

[GAS 01] GASPARY, L.; BALBINOT, L. F.; STORCH, R.; et al. "**Uma Arquitetura para Gerenciamento Distribuído e Flexível de Protocolos de Alto Nível e Serviços de Rede**". Simpósio Brasileiro de Redes de Computadores - Florianópolis, Brasil. 2001.

[GEO 01] GEORGATOS, F.; GRUBER, F.; KARRENBERG, D; et al. "**Providing Active Measurements as a Regular Service for ISP´s**". Proceedings of PAM, 2001.

[GRE 00] McGREGOR T.; BRAUN, H. W.; BROWN, J.; "**The NLAR Network Analysis Infrastructure**," IEEE Communication Magazine, vol. 38, no. 5, Maio de 2000.

[GUI 07] GUIMARÃES, V. T. "**Amostragem Aleatória Estratificada Adaptativa para Identificação de Fluxos "Elefantes" em Redes Convergentes**". 2007. Dissertação (Mestrado em Engenharia Elétrica) - PPGEE, PUCRS, Porto Alegre, 2007.

[HAN 99] HANDELMAN, S.; STIBLER, S.; BROWNLEE, N. et al. **"RTFM: New Attributes for Traffic Flow Measurement"**. RFC 2724, Outubro de 1999.

[HEU 00] HEUSER, C. A. "**Projeto de Banco de Dados**". Porto Alegre: Sagra Luzzatto, 2000. 204p.

[ITU 00] ITU. "**Principles for a Telecommunications Management Network**". ITU-T

Recommendation M.3010. 2000.

[ITU 01] ITU. "**Communications Quality of Service: A framework Definitions**". ITU-T Recommendation G.1000. 2001.

[ITU 02] ITU. "**Series Y: Global Information Infrastructure And Internet Protocol Aspects**". *ITU-T Recommendation Y.1541*, Maio de 2002.

[JOR 00] JORMAKKA, J.; HEIKKINEN, K. "**QoS/GOS Parameter Definitions and Measurements in IP/ATM Networks**". Quality of future Internet Services. Lecture Notes in Computer Science, 2000.

[KAL 99] KALIDINDI, S.; ZEKAUSKAS, M. "**Surveyor: An infrastructure for internet performance measurements**". INET'99, San Jose, Junho de 1999.

[KEN 98] KENT, S.; ATKINSON, R. "**Security Architeture for the Internet Protocol**". RFC 2401, Novembro de 1998.

[KOC 03] KOCK, A. "**Flexible Traffic Matrix Analyser for Inter-domain Network Operation and Planning**". Inter-domain Performance and Simulation Workshop, IPS 2003. Fevereiro de 2003.

[LEI 04] LEINEN, S. "**Evaluation of Candidate Protocols for IP Flow Information Export (IPFIX)**". RFC 3955, Outubro de 2004.

[LOU 00] LUONG, D. D.; BIRD, J. "**Needed Services for Network Performance Evaluation**". IFIP Workshop on Performance Modeling and Evaluation of ATM Networks, 2000.

[MAN 04] MANDRAWA, W.; CALYAM, P.; SRIDHARAN, M.; KHAN, A.; SCHOPIS, P.; "**H.323 Beacon: An H.323 Application Related End-to-End Performance Troubleshooting Tool**". ACM SIGCOMM Network

Troubleshooting Workshop, 2004.

[MAO 03] MAO, Z.; BUSH, R.; GRIFFIN, T.; ROUGHAN, M. "**BGP Beacons**", Internet Measurement Conference, 2003.

[MAR 07] MARK, L. "**libIPFIX: Internet Measurement Project**". Disponível por www em: <http://ants.fokus.fraunhofer.de/libipfix/>, último acesso em Janeiro de 2007.

[MEY 02] MEYER, J. "**Reliable Streaming Internet Protocol Detail Records**", Agosto de 2002. Apud [LEI 04].

[MIL 91] MILLS, C; HIRSH, D; RUTH, G. "**Internet Accounting: Background**". RFC 1272, Novembro de 1991.

[MIL 92] MILLS, D. "**Network Time Protocol (version 3) – Specification, Implementation and Analysis**". RFC 1305, Março de 1992.

[MOL 03] MOLINA, M. et al. "**Implementation of an IPFIX Compliant Flow Traffic Meter: Challenges and Performance Assessment**". IEEE Workshop on IP Operations and Management (IPOM2003), Outubro de 2003.

[MOO 04] MOOCK, C. "**Essential ActionScript 2.0**". O'Relly. 2004. 502 p.

[MOR 01] MORRIS, R.; STURM, W.; JANDER, M. "**Service Level Management: Fundamentos do Gerenciamento de Níveis de Serviço**". Rio de Janeiro. Editora Campus. 2001.

[PAX 97] PAXSON, V. "**End-to-End Internet Packet Dynamics**". ACM SIGCOMM, Setembro de 1997.

[PAX 98] PAXSON, V. “**On Calibrating Measurements of Packet Transit Times**”. ACM Sigmetrics, Junho de1998.

[PHA 01] PHAAL, P.; PANCHEN S.; McKEE; N. “**InMon Corporation's sFlow: A Method for Monitoring Traffic in Switched and Routed Networks**”. RFC 3176, Setembro de 2001.

[POS 81] POSTEL, J. "**Internet Control Message Protocol**" RFC 792, Setembro de 1981.

[POS 80] POSTEL, J. "**User Datagram Protocol**" RFC 768, Agosto de 1980.

[PRE 02] PRESUHN, R. “**Management Information Base (MIB) for the Simple Network Management Protocol (SNMP)**”. RFC 3418, Dezembro de 2002.

[QUI 04] QUITTEK, J. et al. “**Requiriments for IP Flow Information Export (IPFIX)**”. RFC 3917, Outubro de 2004.

[QUI 06] QUITTEK, J. et al. “**Information Model for IP Flow Information Export**”. Draft IETF, Outubro de 2006. Disponível em <http://www.ietf.org/internet-drafts/draft-ietf-ipfix-info-14.txt> em janeiro de 2007.

[RÄI 03] RÄISÄNEN, V. “**Implementing Service Quality in IP Networks**”. Editora Wiley, 2003. 305 p.

[REI 06] REINHARDT, R.; DOWD S. “**Macromedia Flash 8 Bible**”. Willey. 2006. 1320 p.

[RES 06] RESCOLA, E.; MODADUGY, N. "**Datagram Transport Layer Security**",

RFC 4347, Abril de 2006.

[ROE 03] ROESLER, V.; FINZSCH, P.; ANDRADE, M.; et al. "**Análise do Mecanismo de Pares de Pacotes Visando Estimar a Banda da Rede Via UDP**". SBRC 2003, Maio de 2003.

[ROS 01] ROSEN, E.; VISWANATHAN, A.; CALLON R. "**Multiprotocol Label Switching Architecture**". RFC 3031, Janeiro de 2001.

[ROS 90] ROSE, M.; McCLOGHRIE, K. "**Structure and Identification of Management Information for TCP/IP-based Internets**". RFC 1155, Maio de 1990.

[SAD 06] SADASIVAM, G. et al. "**IPFIX Architecture**". Draft IEEE, disponível em <http://www.ietf.org/internet-drafts/draft-ietf-ipfix-architecture-12.txt> em Janeiro de 2007.

[SAN 07] SANTOS, G.; GUIMARÃES, V.; SILVEIRA, J.; VIEIRA, A.; NETO, J. A.; COSTA, R.; BALBINOT, R. "**UAMA: a Unified Architecture for Active Measurements in IP Networks**". Proceedings of the IFIP/IEEE International Symposium on Integrated Network Management, Maio de 2007.

[SCH 98] SCHILDT, H. "**C++: The Complete Reference Third Edition**". Osborne McGraw-Hill.1998.1041p.

[SHA 04] SHALUNOV, S.; TEITTELBAUM, B. "**One-Way Active Measurement Protocol (OWAMP)**". RFC 3763, Abril de 2004.

[STE 00] STEWART, R. et al. "**Stream Control Transmission Protocol**". RFC 2960, Outubro de 2000.

[SUE 99] SUE B. M.; TOWSLEY, D. "**Estimation and Removal of clock Skew for Network Delay Measurements**". IEEE Infocom, 1999.

[TCP 07] TCPDUMP/LIBPCAP. "**Dump traffic on a network and Library Packet Capture**". Disponível em <http://www.tcpdump.org/> em Janeiro de 2007.

[TSU 02] TSURU, T. T. M.; OIE, Y. "**Estimation of Clock Offset from One-Way Delay Measurement on Asymmetric Paths**". SAINT International Symposium on Applications and the Internet, 2002.

[VIE 04] VIEIRA, A. T. "**Pesquisa, Desenvolvimento e Construção de uma Ferramenta para Gerência de Desempenho em Redes Convergentes Baseada na Medida em Tempo Real do Tráfego Classificado por Fluxos**". Porto Alegre: PPGEE/PUCRS, 2004. 185p. (Dissertação de Mestrado)

[WAN 01] WANG, Z. "**Internet QoS: Architectures and Mechanisms for Quality Service**". Morgan Kaufmann, 2001. 256 p.

[ZAN 02] ZANG, K.; ELKIN, E. "**XACCT's Common Reliable Accounting for Network Element (CRANE) Protocol Specification Version 1.0**". RFC 3423, Novembro de 2002.

[ZHA 02] ZHANG, Z. L.; XIA, C. H. "**Clock Synchronization Algorithms for Network Measurements**". IEEE Infocom, 2002.

[ZSE 06] ZSEBY, T. "**IPFIX Aplicability**". Draft IETF, Agosto de 2006. Disponível em <http://www.ietf.org/internet-drafts/draft-ietf-ipfix-as-10.txt> em janeiro de 2007.

APÊNDICE A – EXEMPLOS DE RELATÓRIOS HISTÓRICOS APRESENTADOS PELO SISTEMA, PARA UM DETERMINADO IP

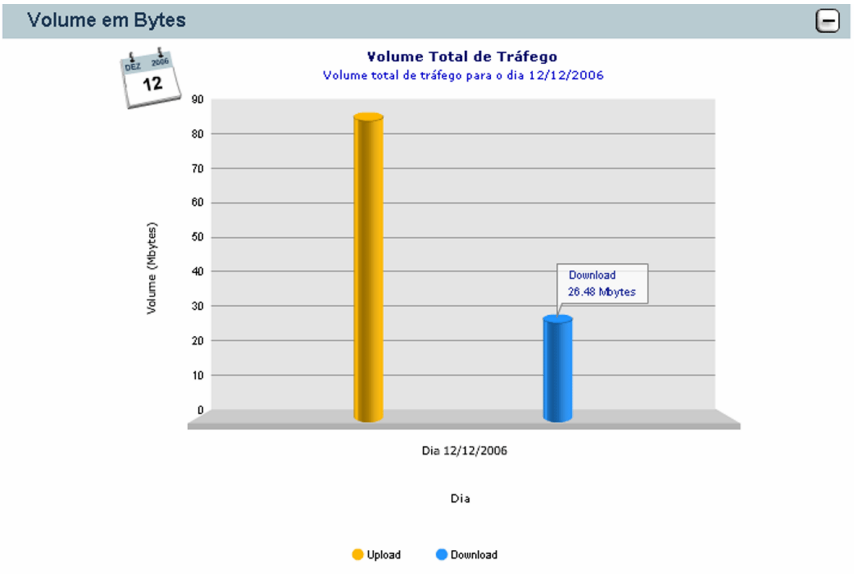


Figura 22 Volume total de tráfego diário.

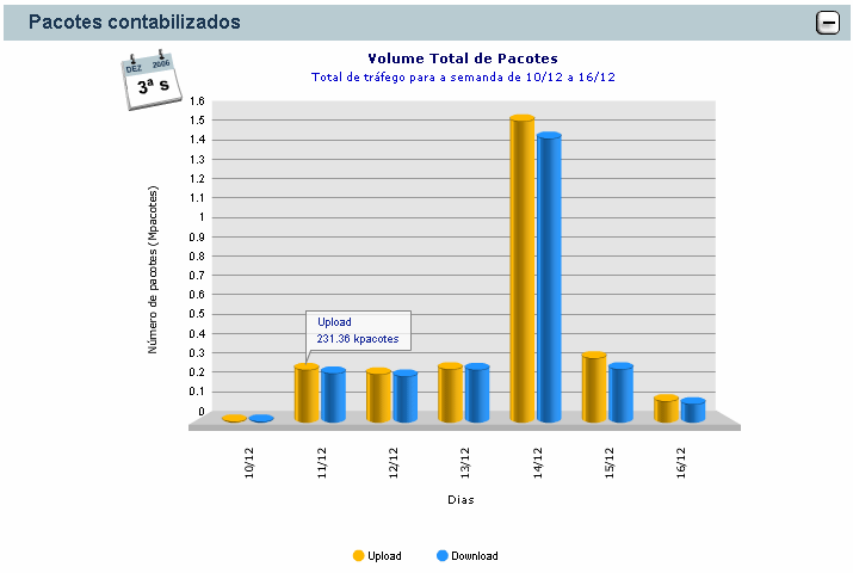


Figura 23 Volume total de pacotes na semana.

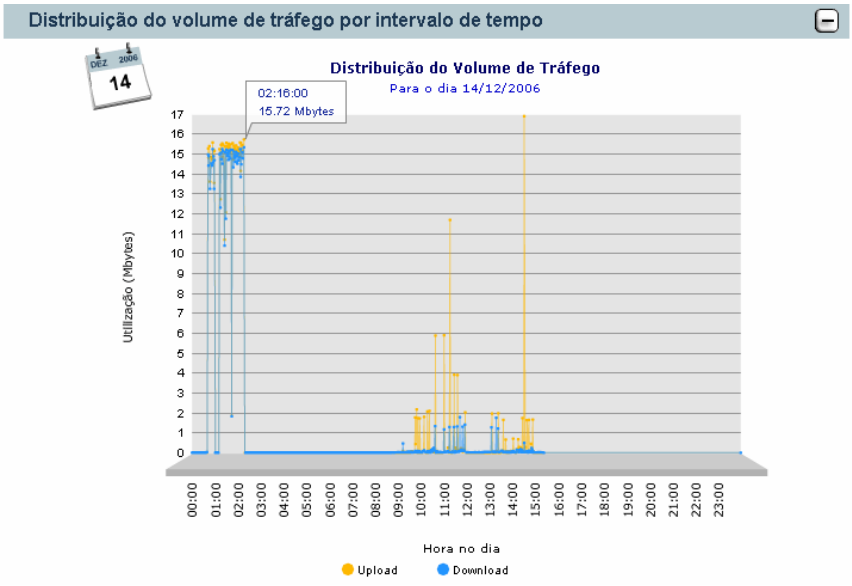


Figura 24 Distribuição diária do volume de tráfego.

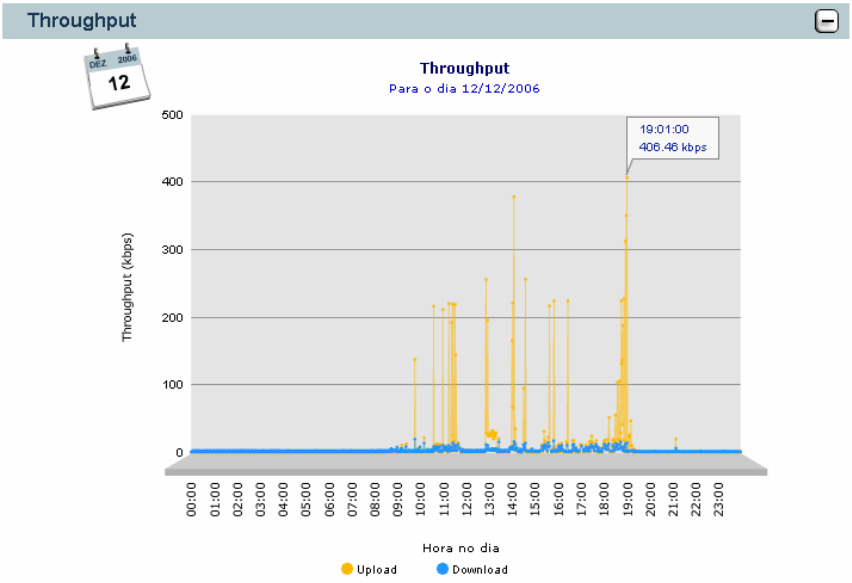


Figura 25 *Throughput.*

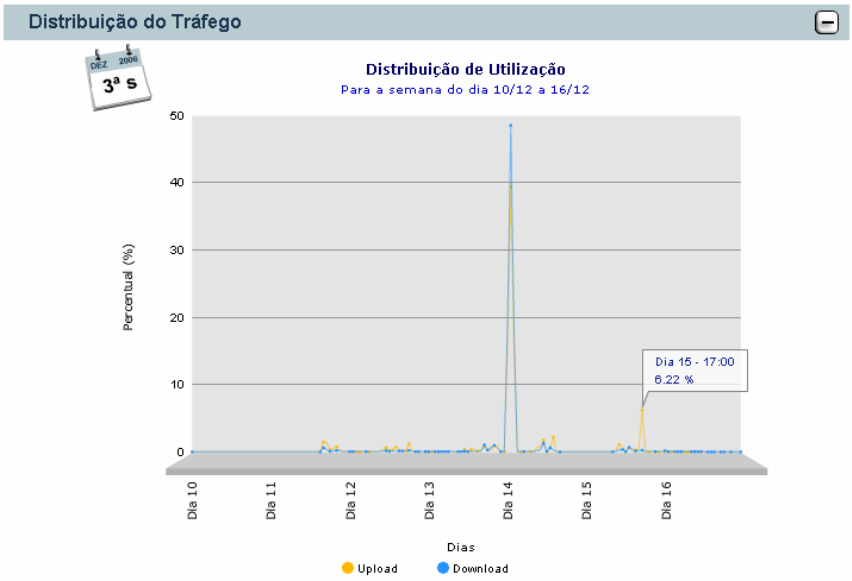


Figura 26 Distribuição percentual do volume de tráfego no decorrer do dia.

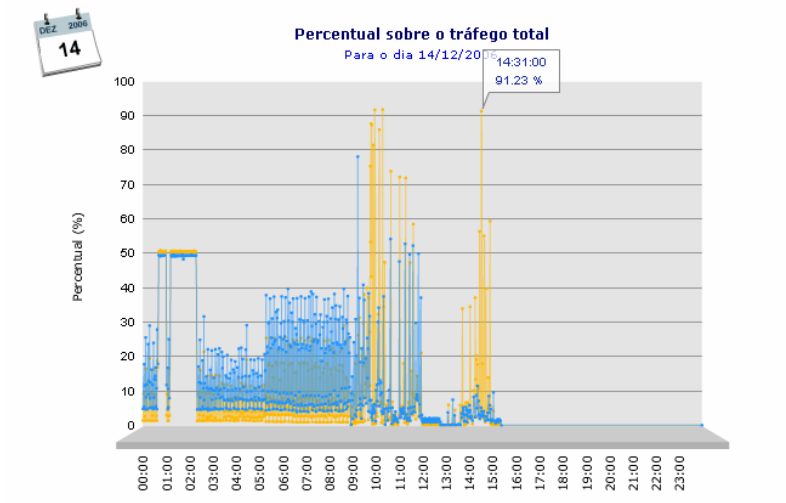


Figura 27 Percentual do fluxo sobre o total, por intervalo de tempo.

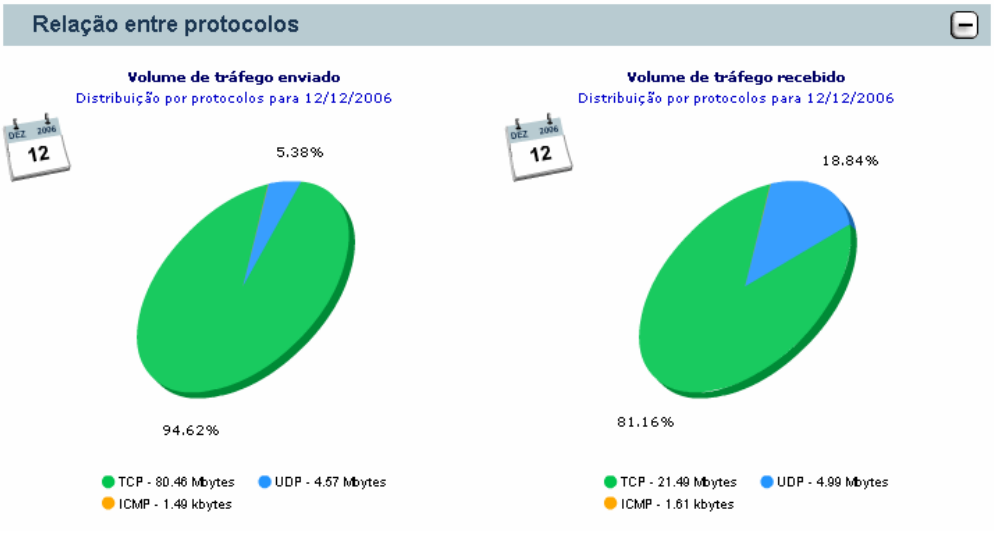


Figura 28 Distribuição do tráfego de acordo com os protocolos utilizados.

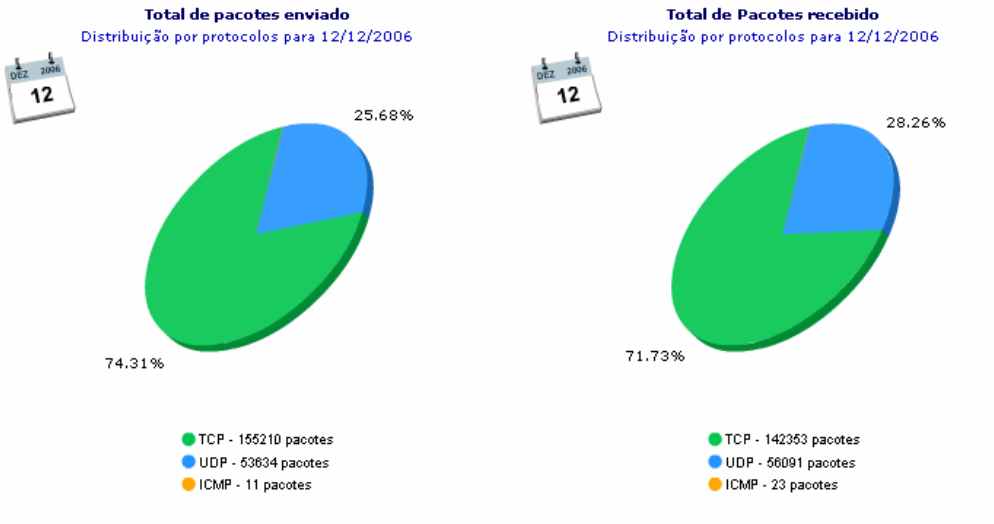


Figura 29 Distribuição dos pacotes de acordo com os protocolos utilizados.

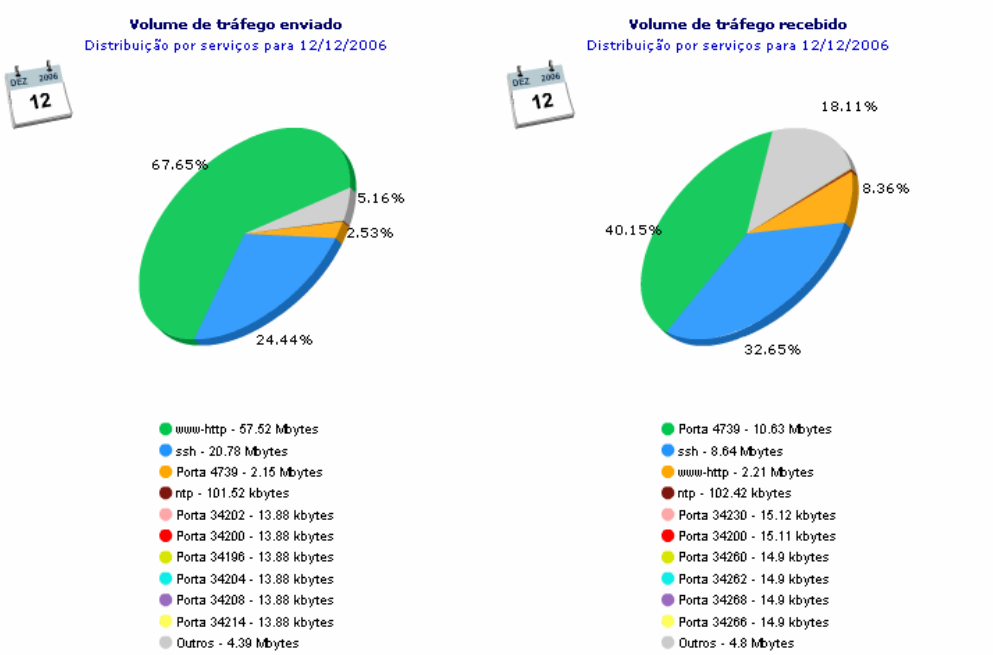


Figura 30 Distribuição do tráfego de acordo com os serviços utilizados.

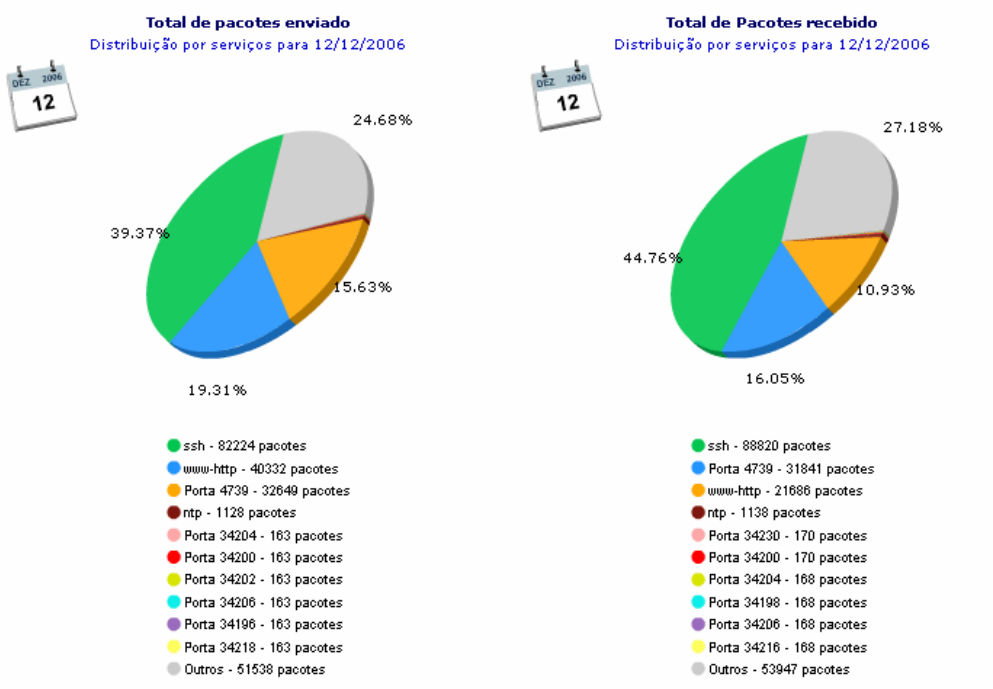


Figura 31 Distribuição de pacotes de acordo com os serviços utilizados.

APÊNDICE B – RESULTADOS DOS TESTES EM AMBIENTE REAL CONTROLADO, UTILIZANDO FLUXOS TCP

Tabela 14 Resultados apresentados para o Arquivo 1 (500 Kbytes).

Processo de Medição	Bytes contabilizados	Pacotes contabilizados	Overhead médio por pacote (bytes)	Overhead médio por pacote (%)
Medição 1	536074	361	66,69	4,49
Medição 2	536074	361	66,69	4,49
Medição 3	536008	360	66,69	4,48
Medição 4	536074	361	66,69	4,49
Medição 5	536074	361	66,69	4,49
Medição 6	536008	360	66,69	4,48
Medição 7	536008	360	66,69	4,48
Medição 8	536074	360	66,87	4,49

Tabela 15 Resultados apresentados para o Arquivo 2 (1 Mbytes).

Processo de Medição	Bytes contabilizados	Pacotes contabilizados	Overhead médio por pacote (bytes)	Overhead médio por pacote (%)
Medição 1	1097072	731	66,34	4,42
Medição 2	1097204	733	66,34	4,43
Medição 3	1097204	733	66,34	4,43
Medição 4	1097204	733	66,34	4,43
Medição 5	1097072	731	66,34	4,42
Medição 6	1097138	732	66,34	4,43
Medição 7	1097138	732	66,34	4,43
Medição 8	1097204	732	66,43	4,43

Tabela 16 Resultados apresentados para o Arquivo 3 (10 Mbytes).

Processo de Medição	Bytes contabilizados	Pacotes contabilizados	Overhead médio por pacote (bytes)	Overhead médio por pacote (%)
Medição 1	10951355	7292	63,85	4,25
Medição 2	10969395	7324	66,03	4,41
Medição 3	10965501	7265	66,03	4,38
Medição 4	10965633	7267	66,03	4,38
Medição 5	10966491	7280	66,03	4,38
Medição 6	10936569	7283	61,9	4,12
Medição 7	10942531	7268	62,85	4,17
Medição 8	10967349	7268	66,26	4,39

Tabela 17 Resultados apresentados para o Arquivo 4 (50 Mbytes).

Processo de Medição	Bytes contabilizados	Pacotes contabilizados	Overhead médio por pacote (bytes)	Overhead médio por pacote (%)
Medição 1	54830000	36378	66,01	4,38
Medição 2	54828350	36353	66,01	4,38
Medição 3	54831716	36404	66,01	4,38
Medição 4	54788630	36405	64,82	4,31
Medição 5	54830594	36387	66,01	4,38
Medição 6	54790850	36391	64,91	4,31
Medição 7	54829736	36374	66,01	4,38
Medição 8	54800420	36374	65,2	4,33

Tabela 18 Resultados apresentados para o Arquivo 5 (100 Mbytes).

Processo de Medição	Bytes contabilizados	Pacotes contabilizados	Overhead médio por pacote (bytes)	Overhead médio por pacote (%)
Medição 1	109590669	72916	64,91	4,32
Medição 2	109568929	72806	64,71	4,3
Medição 3	109564861	73108	64,39	4,3
Medição 4	109389255	72773	62,27	4,14
Medição 5	109659683	72755	66	4,38
Medição 6	109483613	72764	63,58	4,23
Medição 7	109665363	72863	65,98	4,38
Medição 8	109468815	72863	63,29	4,21

Tabela 19 Resultados apresentados para o Arquivo 6 (500 Mbytes).

Processo de Medição	Bytes contabilizados	Pacotes contabilizados	Overhead médio por pacote (bytes)	Overhead médio por pacote (%)
Medição 1	547859878	363852	64,78	4,3
Medição 2	546907886	363952	62,15	4,14
Medição 3	547148504	363897	62,82	4,18
Medição 4	547522134	364020	63,83	4,24
Medição 5	546933626	364478	62,13	4,14
Medição 6	548102412	369531	64,44	4,34
Medição 7	546955064	364133	62,25	4,14
Medição 8	546534338	364133	61,09	4,07

Tabela 20 Resultados apresentados para o Arquivo 7 (800 Mbytes).

Processo de Medição	Bytes contabilizados	Pacotes contabilizados	Overhead médio por pacote (bytes)	Overhead médio por pacote (%)
Medição 1	876839300	582247	65,23	4,33
Medição 2	877841730	591818	65,87	4,44
Medição 3	877410120	591193	65,21	4,39
Medição 4	875067852	581731	62,24	4,14
Medição 5	874740150	582720	61,57	4,1
Medição 6	877420016	591541	65,18	4,39
Medição 7	874694988	583787	61,38	4,1
Medição 8	877295000	591758	64,95	4,38