Uma Metodologia para Caracterização de Tráfego e Medidas de Desempenho em Backbones IP

Claus Rugani Töpke

29 de junho de 2001

UMA METODOLOGIA PARA CARACTERIZAÇÃO DE TRÁFEGO E MEDIDAS DE DESEMPENHO EM BACKBONES IP

Claus Rugani Töpke

TESE SUBMETIDA AO CORPO DOCENTE DA COORDENAÇÃO DOS PROGRAMAS DE PÓS-GRADUAÇÃO DE ENGENHARIA DA UNIVERSIDADE FEDERAL DO RIO DE JANEIRO COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU EM MESTRE EM CIÊNCIAS EM ENGENHARIA DE SISTEMAS E COMPUTAÇÃO

Aprovada por:

Prof. Luis Felipe Magalhães de Moraes, Ph.D.

Prof. Mauricio Ferreira Magalhães, Dsc.

Prof. Jorge Lopes de Souza Leão, Dsc.

Prof. Marcio Portes de Albuquerque, Dsc.

RIO DE JANEIRO,RJ - BRASIL
JUNHO DE 2001

TÖPKE, CLAUS RUGANI

Uma Metodologia para Caracterização de Tráfego e Medidas de Desempenho em Backbones IP. [Rio de Janeiro] 2001

VIII, 64p. 29,7cm (COPPE/UFRJ, M.Sc., Engenharia de Sistemas e Computação, 2001)

Tese - Universidade Federal do Rio de Janeiro, COPPE

1. Redes de computadores

I. COPPE/UFRJ II. Título (série)

Mensagem

.
.
.
Doce é saber que não estou sozinho,
sou uma parte de uma imensa vida.
Uma pequena parte de uma imensa vida.
Que generosa reluz em torno a mim
imenso dom do teu amor sem fim.
O céu nos destes e as estrelas claras,
nosso irmão sol, nossa irmã lua.
Nossa mãe terra, com frutos, campos, flores,
o fogo e o vento, o ar e a água pura,
fonte de vida de tua criatura.
.
.
.
São Fransisco de Assis

Resumo da Tese apresentada à COPPE/UFRJ como parte dos requisitos necessários para a obtenção do grau de Mestre em Ciências (M.Sc.)

UMA METODOLOGIA PARA CARACTERIZAÇÃO DE TRÁFEGO E MEDIDAS DE DESEMPENHO EM BACKBONES IP

Claus Rugani Töpke

Junho/2001

Orientador: Luis Felipe Magalhães de Moraes

Programa: Engenharia de Sistemas e Computação

Identificar o tráfego que passa na rede e como passa na rede é atualmente um dos maiores desafios para a maioria dos Backbones IP. Este desafio tem se acentuado pelo simples fato de que hoje não existem ferramentas nem produtos no mercado que façam tal avaliação e, principalmente, não existe uma metodologia definindo a construção de uma infraestrutura para tal análise.

A identificação dos parâmetros críticos do Backbone IP e a caracterização do tráfego são processos importantes para as medidas do seu desempenho.

Esta tese procura ampliar a visão sobre o assunto, coletando informações de trabalhos da comunidade, criando definições, organizando os conceitos existentes e introduzindo exemplos de coleta e análise. Como resultado este trabalho representa uma poderosa fonte de consulta, para a comunidade acadêmica/científica e para as empresas que utilizam uma infraestrutura baseada em redes IP.

Abstract of Thesis presented to COPPE/UFRJ as a partial fulfillment of the requirements for the degree of Master of Science (M.Sc.)

METODOLOGY FOR TRAFFIC CARACTERIZATION AND PERFORMANCE MEASUREMENTS ON IP BACKBONES

Claus Rugani Töpke

June/2001

Advisor: Luis Felipe Magalhães de Moraes

Department: System and Computer Engineering

The identification of which traffic passes the network and how it does so is currently one of the major challenges for most IP Backbones. This challenge has been stressed by the simple fact that there are no tools, or market products, today which are capable to do such evaluation and, mainly, there is not methodology defining the construction of an infrastructure for such analisys.

Identification of the critical parameters of the IP Backbone and the traffic caracterization are important processes to measure its performance.

This thesis aims at extending the view about the subject, collecting information from previous works, creating definitions, organizing existing concepts and introducting examples for data collect and analysis. As a result, this work presents a powerful reference tool, for academic/scientific community and for the companies which utilize an infrastructure based on IP networks.

Sumário

Lista de Figuras

Lista de Tabelas

Capítulo 1

Introdução

Observando o crescimento notável que a Internet vem apresentando ao longo dos anos e a recente "onda internauta" que a mídia tem feito, pode-se concluir que esta rede será o mais popular meio de comunicação com abrangência mundial. Um dos maiores motivos da Internet ter se popularizado é por ser uma rede sem restrições, onde o acesso à informação é fácil quando comparada aos meios tradicionais.

A rede Internet é constituida pela interconexão de uma série de pequenas redes, tais como: empresas, universidades, instituições do governo, centros de pesquisa e servidores de acesso remoto à Internet. O presente trabalho é voltado para as redes de grande porte , chamadas de *Backbones IP*, que funcionam como meio de acesso e transporte para as pequenas redes.

Tendo em vista o crescimento que a Internet terá nos próximos anos, torna-se de fundamental importância para o gerenciamento e planejamento futuro de sua infraestrutura, um trabalho que forneça meios de avaliação do tráfego e de desempenho voltado para os Backbones IP.

Identificar qual tráfego que passa na rede e como passa na rede é atualmente um dos maiores desafios para a maioria dos Backbones IP. Este desafio tem se agravado pelo simples fato de que hoje não existem ferramentas nem produtos no mercado que façam tal avaliação, e principalmente não exite uma metodologia definindo a construção de uma infraestrutura para tal análise.

Este capítulo irá apresentar questão de como caracterizar o tráfego e como realizar medições de desempenho dentro de um Backbone IP. Apresentará um pequeno resumo de todo o trabalho e suas contribuições para a comunidade.

Por ser muito extensa a quantidade de informações necessárias para um entendimento maior das tecnologias e das definições utilizadas neste trabalho,

foi criado um pequeno resumo dos pontos mais importantes, que pode ser encontrado no apêndice D.

Para facilitar o entendimento de alguns termos e palavras em inglês foi criado um pequeno glossário contido no final deste trabalho (apêndice D.5).

1.1 O Problema

Hoje a Internet é tecnologicamente muito diversa, tanto em infraestrutura de rede como em protocolos e aplicações que a utilizam. Em termos de infraestrutura de rede a Internet pode ter as seguintes características:

- Enlaces com capacidades de transmissão variando de dezenas de Kbps até altas capacidades, da ordem de Gbps.
- Grandes variações de latência dos pacotes em função das distâncias percorridas e das velocidades dos enlaces.
- Variações na taxa de perda de pacotes, passando por enlaces congestionados com altas taxas de descartes e espera em filas e algumas vezes por enlaces livres e sem espera em filas.
- Utilização de tecnologias diferentes como infraestrutura para o encaminhamento dos seus pacotes, passando por exemplo por: FDDI, Ethernet, ATM, Frame Relay , X25 e até mesmo por redes de telefonia celular e convencional.

Além da infraestrutura ter uma característica bem vasta e diversificada, as aplicações que utilizam esta infraestrutura também têm características muito diversas. Alguns exemplos de aplicações utilizadas na Internet: comércio eletrônico, troca de mensagens, conversação, filmes sob demanda, jogos, apresentações e shows ao vivo, divulgação de notícias, pesquisas de opinião, propaganda, transmissão de música, etc.

Devido a esta diversidade de aplicações e componentes de rede, os parâmetros que definem o desempenho de um Backbone IP e que caracterizam o seu tráfego são bastante complexos.

Para que um Backbone IP possa ter uma avaliação do seu funcionamento e de como deve prosseguir no futuro ele deve, fundamentalmente, possuir medidas que avaliem o seu tráfego de forma qualitativa e quantitativa.

1.2 Resumo deste Trabalho

Este trabalho está dividio em 4 partes, como descrito a seguir:

Primeira Parte:

O objetivo desta parte é apresentar uma introdução do trabalho, mostrar a sua importância, apresentar um pequeno resumo, definir um Backbone IP e finalmente mostrar uma metodologia para a construção de uma infraestrura de coleta e análise do tráfego de um Backbone IP. Os capítulos 1, 2 e 3 estão dentro da primeira parte.

Segunda Parte:

O objetivo da segunda parte do trabalho é apresentar os parâmetros de medida, a metodologia da coleta dos dados e, finalmente, a descrição dos tipos de análise e consolidação dos dados obtidos. Os capítulos 4, 5 e 6 estão dentro da segunda parte.

Terceira Parte:

O objetivo da terceira parte é mostrar algumas ferramentas utilizadas pela comunidade para coleta de dados e análise, apresentar alguns resultados utilizando tais ferramentas e por fim apresentar as conclusões de todo o trabalho. Os capítulos 7, 8 e 9 estão dentro da terceira parte.

Parte Adicional:

A parte adicional tem como objetivo apresentar algumas instituições que realizam trabalhos voltados ao tema desta tese, mostrar detalhes de alguns protocolos , listar alguns programas utilizados para coleta e análise, apresentar algumas definições utilizadas durante o trabalho e por fim um pequeno glossário. A última parte é composta por todos os Apêndices (A, B, C e D).

Ao final de todo o trabalho estão as referências bibliográficas que foram utilizadas durante o desenvolvimento deste trabalho de tese.

1.3 Contribuição deste Trabalho

As definições feitas pela comunidade sobre medidas de desempenho e caracterização de tráfego IP na Internet, são muitas vezes precárias e

"limitadas". Este trabalho procura "ampliar" a visão sobre o assunto, abrindo uma série de novas definições.

Com isso este trabalho representa uma poderosa fonte de consulta para a comunidade acadêmica/científica e para as empresas prestadoras de serviço de acesso a Backbones IP.

A contribuição foi feita tanto na teoria, com a criação de novas definições, como na prática, pela criação de alguns programas e utilização de ferramentas.

A contribuição teórica envolve:

- Pesquisa prévia realizada sobre o assunto na comunidade (apêndice A);
- Descrição de uma metodologia de medida (capítulo 3);
- Definições das variáveis de medida (capítulo 4);
- Definições dos métodos de coleta dos dados (capítulo 5);
- Definições dos métodos de análise dos dados (capítulo 6);

A contribuição prática envolve:

- Estudo das principais ferramentas existentes (capítulo 7);
- Aplicação de coleta e análise em 4 exemplos (capítulo 8);
- Criação de uma série de programas para atender as coletas e análises feitas no capítulo 8 (a listagem de alguns pode ser encontrado no apêndice C).

1.4 Considerações sobre os *Softwares* Utilizados

O movimento científico de maior importância demostrado na Internet na década de 90 é o conjunto de instituições que promovem a disseminação cultural de sistemas, aplicativos e ferramentas. Existem muitos nomes para estes tipos de aplicativos e sistemas, porém o termo mais utilizado é *opensource*. Ou seja, são sistemas que possuem seu código fonte aberto, possibilitando qualquer pessoa ter acesso ao conjunto de programas, alterando, melhorando ou simplesmente estudando. A contribuição para estes movimentos é fundamental para o aprimoramento cultural e científico.

Muitas dessas ferramentas tem maior utilização em relação aos produtos comerciais, e principalmente maior desempenho, escalabilidade e flexibilidade.

Este trabalho foi todo desenvolvido com softwares *opensource*.

Capítulo 2

O Backbone IP

Backbone IP é um termo que será utilizado neste trabalho para designar uma rede de computadores de grande porte a qual utiliza o protocolo IP para realizar a interconexão de outras pequenas redes. O Backbone IP geralmente é administrado por alguma instituição que funciona como um sistema autônomo ou *autonomous system* (AS). O AS possui sua própria política de roteamento, tanto interno como para interconexão com outros Backbones IP. No Brasil podemos citar como exemplos de Backbones IP as redes das seguintes instituições: RNP (Rede Nacional de Pesquisa [81]), RedeRio [82], ANSP [84] e a Internet da Embratel [85].

O encaminhamento dos pacotes IP na Internet dependerá unicamente de dois fatores: Do roteamento interno dentro de um AS e do roteamento externo entre dois ASs. O roteamento interno é implementado através de um protocolo conhecido como IGP (*Interior Gateway Protocol*) e o roteamento externo é implementado através de um protocolo conhecido como EGP (*Exterior Gateway Protocol*). O roteamento externo dependerá das políticas de roteamento estabelecidas na ligação de cada AS (vizinho ou *peer*).

Atualmente, o protocolo de roteamento externo largamente utilizado é o BGP versão 4. Os protocolos de roteamento interno mais utilizados são o OSPF, IS-IS e o IBGP. O roteamento estático também é utilizado e geralmente é colocado na rede para implementar a interconexão com redes finais ou para corrigir possíveis flutuações no protocolo interno e/ou externo.

Para implementar o protocolo BGP é necessário que o Backbone IP tenha um número associado à sua rede, este número é único na Internet, chamado de ASN (*autonomous system number* veja como obter este número em [62]). Para as instituições citadas no exemplo acima os números associados são respectivamente: 1916 (RNP), 2715 (RedeRio), 1715 (ANSP) e 4230

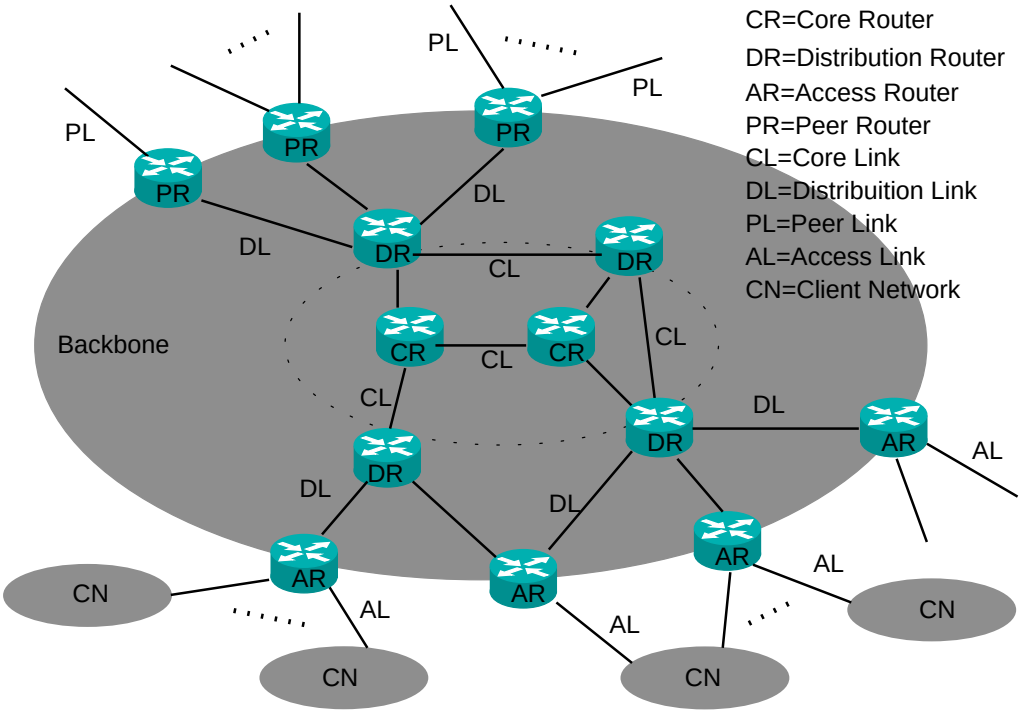


Figura 2.1: Topologia interna de um Backbone IP

(Embratel).

Neste capítulo iremos descrever a arquitetura interna e externa de um Backbone IP, os protocolos e as políticas de roteamento.

2.1 Topologia Interna

Existem muitas maneiras de se criar um Backbone IP, utilizando uma arquitetura simples ou uma arquitetura mais completa. A diferença entre uma arquitetura mais completa e uma mais simples está na especialização das funcionalidades de cada equipamento da rede. A figura 2.1 mostra uma arquitetura mais completa com 4 especialidades de roteadores com as seguintes funções:

Roteadores de Núcleo (*Core Routers* CR): São os roteadores que carregam mais tráfego da rede pois agregam o tráfego de todos os outros roteadores. São equipamentos com alta capacidade de roteamento e com interfaces de alta velocidade. Apresentam-se em pequenas quantidades comparados com os roteadores de outras funcionalidades;

Roteadores de Distribuição (*Distribution Routers* DR): São roteadores que têm como função agregar o tráfego dos roteadores de acesso e dos

roteadores de *peering*. Muitas vezes são utilizados como redundância de acesso aos roteadores do núcleo. Possuem interfaces de velocidade intermediária. Em muitas arquiteturas esta funcionalidade de roteador não é utilizada, interligando o núcleo diretamente aos roteadores de *peering* e de acesso.

Roteadores de Acesso (*Access Routers* AR): São roteadores especializados em acesso a redes que não possuem outra ligação a não ser com este Backbone IP. São equipamentos com alta concentração de interfaces. A quantidade de roteadores com esta função é sempre muito grande. Possuem interfaces de velocidades intermediárias a baixas.

Roteadores de *Peering* (*Peer Routers* PR): São roteadores especializados em acesso a outros Backbones IP. São equipamentos que implementam as políticas de roteamento externo estabelecidas na conexão com um outro AS. Possuem interfaces de velocidades variando de intermediárias a altas.

Uma boa prática é separar interfaces de velocidades compatíveis em roteadores distintos, ou seja não colocar interfaces de 64Kbps com interfaces de 100Mbps no mesmo roteador. Esta prática deve ser tomada quando um dos roteadores utiliza um *buffer* de saída compartilhado, causando danos a interfaces de alta velocidade quando as de baixa velocidade se apresentam saturadas.

O roteamento interno é feito através de um protocolo IGP. Os protocolos mais utilizados são o IS-IS, o OSPF e o IBGP. A política de roteamento interna define como será o encaminhamento dos pacotes para os enlaces do tipo DL, CL e AL (figura 2.1).

Existem duas maneiras de fazer o roteamento até as redes dos clientes (CN): A primeira é colocar a rede CN como rede integrante do protocolo de roteamento interno; a segunda é colocar a rede CN como uma rede externa ao protocolo de roteamento interno, redistribuindo esta rede no protocolo de roteamento interno quando necessário.

Quanto ao acesso das redes CN, a redistribuição é sempre mais segura por não interferir no algoritmo que calcula o melhor caminho. A implementação é feita criando-se rotas estáticas nos roteadores de acesso (AR) para as redes CN, posteriormente estas rotas estáticas são redistribuidas no protocolo de roteamento interno.

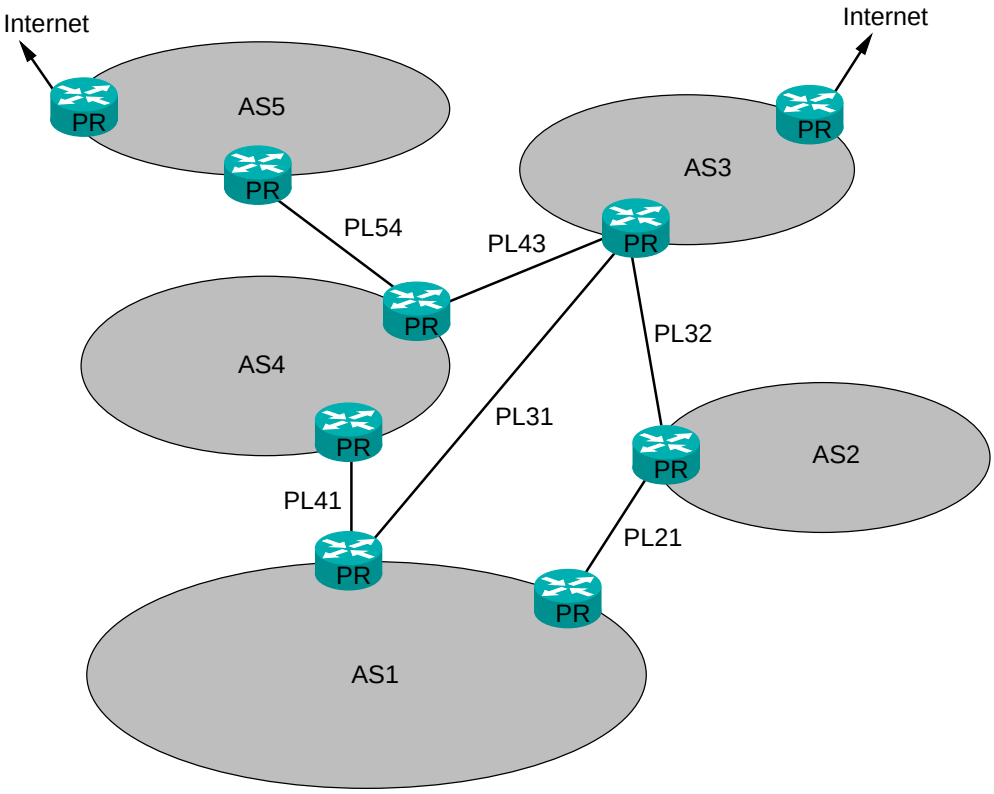


Figura 2.2: Topologia de conexão externa de Backbones IP

A monitoração do comportamento das políticas internas de roteamento é um parâmetro importante para a avaliação do desempenho do Backbone IP.

2.2 Topologia Externa

A figura 2.2 apresenta um exemplo de uma topologia de interligação entre 5 sistemas autônomos chamados de: AS1, AS2, AS3, AS4 e AS5. A interligação entre estes ASs só é feita através dos roteadores e enlaces de *peering* chamados de: PL21, PL31, PL32, PL41, PL43 e PL54.

Cada sistema autônomo possui a sua própria topologia interna conforme mostrado na figura 2.1. As redes internas de um ASx são enxergadas pelos outros ASs como um aglomerado de redes pertencentes a este ASx.

O tráfego entre os ASs é feito através do anúncio das rotas pertencentes a cada AS para o seu vizinho, utilizando o protocolo BGP. Políticas de roteamento devem ser feitas para evitar tráfego de pacotes indesejado entrante e/ou saínte a um Backbone IP.

São consideradas redes internas a um AS todas as redes pertencentes aos roteadores de núcleo (CR) , aos roteadores de distribuição (DR), e aos

roteadores de acesso (AR). Para os roteadores de *peering* (PR) apenas os enlaces DL são considerados internos ao AS (figura 2.2). As redes dos clientes (CN) ligadas ao roteadores de acesso não participam do Backbone IP, porém são consideradas redes internas ao AS quando visualizadas pelos outros ASs. Isto ocorre pelo simples fato de que tais redes apenas possuem ligação com um Backbone IP.

É importante não confundir a rede de um cliente como rede pertencente a um AS, por exemplo: Na figura 2.2 o AS2 pode ser cliente do AS1, porém não é parte integrante do AS1 pois possui seu próprio ASN (*Autonomous System Number*)

A política de roteamento externa consiste em regras de roteamento estabelecidas na ligação entre dois ASs quaisquer. Na figura 2.2 o enlace PL41 que interliga o AS1 e o AS4 pode possuir uma política de roteamento diferente da política estabelecida em PL21. Ou seja, a política de roteamento é individual para cada *peering*.

Como exemplo, imagine que o AS1 possui as seguintes redes: 200.12.20.0/24, 200.12.21.0/24 e 200.12.22.0/24. Para que estas redes sejam enxergadas pelo AS4 é necessário primeiramente que AS1 anuncie (via protocolo BGP) estas redes para o AS4 e que o AS4 não filtre os anúncios recebidos do AS1.

Três políticas básicas podem ser feitas entre dois ASs:

1-**Trânsito**: Esta política é utilizada para que o AS ligado funcione como trânsito para os pacotes saíntes/entrantes. Na figura 2.2 o AS1 deve utilizar AS4 ou AS2 como trânsito para acesso aos outros ASs e ao resto da Internet.

2-**Redundância de trânsito**: Esta política é utilizada como redundância da política anterior. Na figura 2.2 o AS1 pode utilizar AS4 como redundância de trânsito ao AS2. Esta redundância é implementada anunciando as redes diferentemente pelo AS4 e pelo AS2.

3-**Troca de tráfego das redes internas**: Esta política é utilizada quando um AS só deseja acessar as redes internas de um outro AS. Geralmente os custos para este tipo de ligação são menores do que os de trânsito.

A monitoração de tais políticas é muito importante como parâmetro de avaliação de desempenho de um Backbone IP.

Capítulo 3

Metodologia

Neste capítulo será apresentada uma metodologia para a construção de uma infraestrutura de medida de desempenho e análise de tráfego de um Backbone IP.

A metodologia descrita neste capítulo é geral, podendo apenas parte dela ser utilizada para a construção de uma versão simplificada.

A figura 3.1 ilustra resumidamente o fluxo de procedimentos necessários para a contrução da infraestrutura. Cada processo da figura será detalhado nas sessões seguintes deste capítulo.

3.1 Os Objetos

Esta é a etapa inicial (número 1 do fluxograma da figura 3.1, a qual consiste na escolha dos objetos do Backbone que irão participar direta ou indiretamente dos alvos das medidas. Geralmente estes objetos são os próprios recursos do Backbone.

Uma boa prática é descrever todos os objetos existentes e não apenas os que serão utilizados para as medidas. Esta descrição geral tem o objetivo de criar um relacionamento entre os objetos, identificando quando o funcionamento de um é dependente ou independente dos demais.

A identificação da interdependência entre os objetos irá mostrar qual o objeto que tem maior influência no resultado de uma medida. Exemplo: o comportamento das interfaces de um roteador está relacionado à quantidade consumida de memória compartilhada (*buffer*). Logo, os objetos "interface" e "memória" são interdependentes. Dependendo do caso, apenas a coleta dos dados da interface é insuficiente para indicar o seu desempenho. Muitas vezes esta medida torna-se mais completa quando se adiciona a coleta da

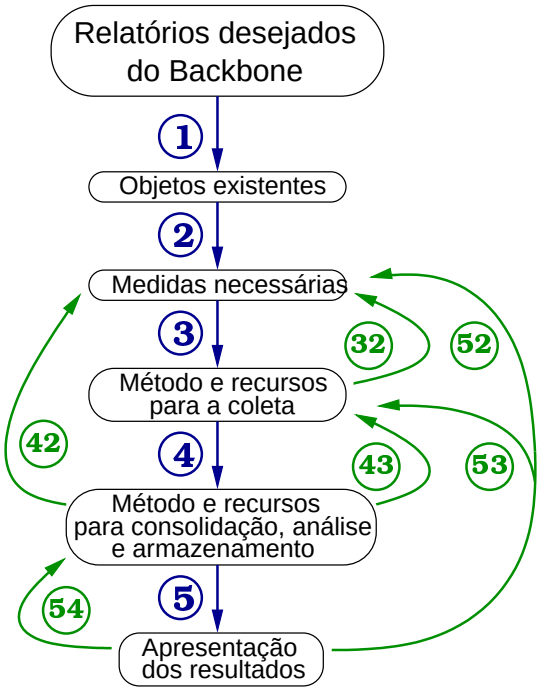


Figura 3.1: Fluxograma para medida de desempenho

quantidade de *buffer* consumida.

Os objetos podem ser: processamento, memória, tabelas de roteamento, filas, roteadores, interfaces, enlaces, comutadores, modems, multiplexadores, etc...

3.2 As Variáveis de Medida

Uma vez escolhido o conjunto de objetos e a dependência entre eles, torna-se necessário definir quais as variáveis de medida que serão obtidas (número 2 da figura 3.1).

Cada tipo de medida está relacionado a um certo conjunto de objetos. Logo a escolha dos objetos irá definir o universo de medidas possíveis a serem realizadas. Exemplo: uma medida como tamanho de tabela de roteamento deve ser obtida em objetos que possuam a característica de roteamento. Objetos como enlace, comutadores ou modems, não podem ser utilizados para tal medida.

Uma boa prática é descrever todas as variáveis de medida possíveis de serem feitas, identificando quais são suas interdependências e enumerando as que têm mais importância para o desempenho do Backbone. Exemplo: no Backbone IP descrito na figura 2.1 a medida da tabela de roteamento nos

roteadores de acesso (AR) não é tão importante quanto nos roteadores de *peering* (PR). Possíveis instabilidades nos anúncios de rotas que chegam nos PR causam instabilidade também nos roteadores de distribuição (DR) e de núcleo (CR), principalmente pela quantidade de rotas externas (*full routing table* tem mais de 90.000 linhas).

O resultado obtido irá depender de uma série de fatores como: os objetos escolhidos, medidas escolhidas, o processo de coleta e o método de análise. O resultado obtido deve ser avaliado para verificar se reflete o desempenho do Backbone.

As medidas de maior interesse são as relacionadas aos parâmentros de desempenho do Backbone. Estes parâmetros são pontos fundamentais para o comportamento eficiente do Backbone, refletindo assim a satisfação dos seus usuários.

Exemplos de medidas: comportamento da tabela de roteamento, atraso de um enlace, latência do envio de pacotes entre dois roteadores, caminho de envio de pacotes, utilização de CPU, descarte de pacotes na fila de saída de interface, retransmissão de pacotes TCP, etc...

As principais medidas possíveis a serem realizadas em um Backbone IP serão melhor abordadas no capítulo 4.

3.3 Método e Recursos da Coleta

Para a obteção das medidas escolhidas é necessário definir o método de coleta (3 da figura 3.1). O método de coleta envolve a descrição dos recursos utilizados e o relacionamento destes com os objetos da medida escolhida.

Na maioria das empresas prestadoras de Backbone IP os recursos destinados à coleta de dados para a avaliação de tráfego e de desempenho é sempre muito pequeno, principalmente pelo crescimento desproporcional da Internet nos últimos anos. Logo, a escolha dos resursos geralmente é limitada a um pequeno conjunto de *hardware* e *software*.

Além dos resursos de *hardware* e *software*, o método de coleta deve prover descrições sobre: o tempo de amostragem, o modo de coleta (ex: ativo, passivo, ...), o sincronismo necessário do coletor (ex: NTP, GPS, local, ...), os protocolos necessários para obtenção dos dados (ex: SNMP, TELNET, BGP, ...) e se for o caso as senhas de acesso.

Se o modo de coleta for ativo é necessário informar o quanto dos recursos da rede do Backbone estão sendo utilizados. Se estes recursos são

concorrentes com o que se está medindo é necessário que este "consumo" fique explícito para que a análise seja feita de forma "correta" (coleta destrutiva).

A descrição dos recursos de *hardware* deve conter detalhes dos equipamentos necessários (ex: memória, CPU, ...) assim como o espaço físico, a localização e o *layout* das ligações físicas e lógicas (ex: endereçamento IP). A descrição dos recursos de *software* deve conter detalhes do sistema operacional, do banco de dados e dos programas e processos necessários.

É necessário redefinir os parâmentros de medida (número 32 da figura 3.1) se uma determinada medida descrita no processo número 2 não puder ser realizada por alguma limitação de recursos encontrada no processo número 3.

Os processos de coleta serão melhor abordados no capítulo 5.

3.4 Análise, Consolidação e Armazenamento

Depois de definir o processo de coleta dos dados é necessário então definir qual o tratamento que será feitos sobre estes dados e como eles serão armazenados. Este processo será chamado de método de análise, consolidação e armazenamento. (número 4 da figura 3.1).

A análise e consolidação dos dados é basicamente um tratamento matemático dos dados coletados, que envolve cálculos tanto no tempo como no espaço. Uma correlação entre medidas diferentes também pode ser feita neste processo. Os dados antes de serem consolidados são chamados de dados "crus".

Muitas vezes a análise e a consolidação são feitas imediatamente após a coleta dos dados. A principal vantagem de se consolidar os dados logo após a sua coleta é que a quantidade de dados para armazenamento será menor que a dos dados crus. A principal desvantagem é a perda de detalhes da coleta dos dados, que eventualmente possam ser necessários para uma análise e consolidação diferentes sobre os dados "crus".

A definição do método de análise e consolidação deverá conter os cálculos no tempo e no espaço que deverão ser feitos sobre os dados. Exemplos de consolidação: média do tamanho das filas de um roteador em 1 hora; variância em um dia do atraso (latência) em todos os enlaces ligados aos roteadores de *peering* (figura 2.1).

O método deve também contemplar o tratamento necessário que se deve fazer nos dados caso ocorra alguma falha temporária no processo de coleta.

O método pode utilizar artifícios estatísticos para estimar valores coletados.

A descrição do método de armazenamento deverá ilustrar a quantidade de dados a serem armazenados no tempo, assim como o tempo de acesso aos dados (é necessário para se estimar o tempo de criação de relatórios).

A descrição dos recursos deverá conter detalhes do *hardware* e *software* necessários para o armazenamento, análise e consolidação. Muitas vezes os recursos são compartilhados entre a coleta, análise, consolidação e armazenamento.

Se não houver método de análise e/ou consolidação capazes de obter os relatórios desejados, é necessário rever os métodos de coleta e as medidas escolhidas (números 42 e 43 da figura 3.1). A revisão dos métodos anteriores pode também ser feita se no futuro for desejado algum resultado adicional ou diferente.

O capítulo 6 irá detalhar melhor o processo de análise dos dados coletados.

3.5 Apresentação dos Resultados

A etapa final é a descrição do método de visualização e apresentação dos dados analisados e consolidados (número 5 da figura 3.1).

Esta é a etapa final e consiste na definição de como implementar a visualização dos resultados. Parâmetros podem ser descritos como entrada para melhorar a visualização. Exemplo: cores, escalas, quantidade e tamanho de gráficos, escolha entre gráficos ou tabelas e escolha do tipo dos gráficos.

Muitas vezes a interface gráfica que apresenta os resultados possui entradas que definem o processo de análise nos dados coletados. Neste caso a consolidação deve ser feita conforme esta solicitação. Isto não é possível de ser implementado se a quantidade de dados coletados for muito grande, necessitando uma consolidação imediatamente após a coleta.

Também deve-se definir a plataforma, a interface, o protocolo e se necessário um controle de acesso para a visualização. Exemplos de plataformas e protocolos: visualização através de um *browser* utilizando-se do protocolo HTTP (*Hiper Text Transfer Protocol*); a programação visual pode ser feita em HTML (*HyperText Markup Language*), Java ou XML. A entrada dos dados pode ser feita através de Java ou CGI (*Common Gateway Interface*).

Os recursos utilizados de *software* e *hardware* também devem ser

descritos. Caso a visualização dos resultados não seja satisfatória, devido à velocidade de sua apresentação ou à falta de informações, é possível a revisão dos processos anteriores (números 52, 53 e 54 da figura 3.1).

Capítulo 4

Definições das Variáveis a Serem Medidas

Neste capítulo serão definidos os parâmetros de medida possíveis de serem retirados em um Backbone IP. Os parâmetros de medida definidos neste capítulo dizem respeito apenas à grandeza e à natureza das variáveis e não ao método de como serão obtidas. O método de obtenção destas medidas será descrito no próximo capítulo, o qual fala especificamente sobre o modo de coleta dos dados.

Os valores e resultados obtidos com tais medidas dependerão de dois principais fatores: dos objetos escolhidos, do método de coleta dos dados e do método de análise e consolidação. Uma medida pode apresentar resultados diferentes conforme a escolha do objeto. Ex: o valor da latência no transporte de pacotes em um enlace é diferente do valor de latência no processo de encaminhamento de um roteador. O modo como os dados da medida serão coletados é um ponto importante para avaliar os valores obtidos. Ex: A medida de latência entre dois pontos pode ser feita de duas maneiras, uma ativa e outra passiva. Os valores obtidos para cada um destes métodos será diferente. Conforme o fluxograma da figura 3.1 a interpretação dos resultados (4) deverá considerar os objetos escolhidos (1), as variáveis de medida (2) e o método de coleta dos dados (3).

Como as medidas podem ser o resultado de uma composição de objetos e/ou variáveis, um trabalho matemático nos dados coletados se faz necessário na maioria das medidas. Este tratamento será chamado de análise e consolidação, que será abordado no capítulo 6.

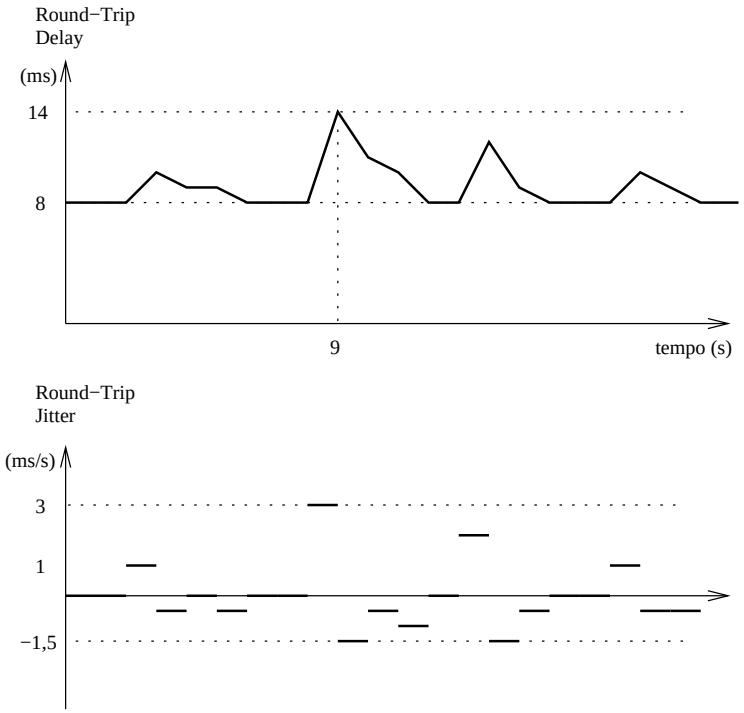


Figura 4.1: Round-trip Delay e Jitter

4.1 Latência e *Jitter*

Existem basicamente duas maneiras de se medir a latência, ou atraso (*delay*) entre dois pontos em uma rede TCP/IP: (i) o atraso de ida (ou *one-way delay*); (ii) o atraso de ida-e-volta (ou *round-trip delay*). As definições adotadas neste trabalho estão a seguir:

(i) *One-Way Delay*: Definido como sendo o tempo decorrido desde a inserção do primeiro *bit* de um pacote IP de tamanho D até a chegada do último *bit* do mesmo pacote IP ao destino. O caminho pelo qual este pacote percorre até chegar ao destino, assim como sua priorização nas filas e CPUs dos roteadores são fatores importantes a serem considerados na avaliação do atraso.

(ii) *Round-trip Delay*: Definido como sendo o tempo decorrido desde a inserção do primeiro *bit* de um pacote IP_1 de tamanho D_1 até a chegada do último *bit* de um outro pacote IP_2 de tamanho D_2 ao emissor. Nesta medida é necessário a utilização de 2 pacotes IP diferentes, o primeiro pacote (IP_1) é gerado pelo emissor, e o segundo pacote (IP_2) é enviado pelo receptor como resposta à requisição do primeiro pacote (IP_1) recebido. Muitas medidas de atraso utilizam o tamanho dos pacotes

de envio e de retorno iguais, ou seja $D_1 = D_2$, porém esta prática pode ser desvantajosa para medidas de atraso simulando aplicações assimétricas, como o HTTP. O caminho de ida do primeiro pacote, o caminho de volta do pacote de resposta, o tratamento de prioridade de filas, a compressão no caminho e os recursos de CPU utilizados são fatores importantes a serem considerados na avaliação do atraso.

Outro parâmetro a ser medido é a variação do *delay* ao longo do tempo chamado de *jitter*. O *jitter* é a primeira derivada no tempo da medida do *delay*, veja figura 4.1.

Para o ponto de vista de um usuário a latência (atraso) é entendido como sendo o tempo decorrido entre o envio de um pacote e o recebimento de um pacote de confirmação retornado pelo receptor. Para o usuário quanto maior a latência menor será a sua capacidade de perceber a interatividade entre ele e o destino.

Motivações para a medida de *delay* e *jitter*:

- Muitas aplicações em tempo real (*real-time*) necessitam que as medidas de *jitter* estejam o mais próximo da realidade. Um erro nesta medida pode comprometer o desempenho ou a qualidade da aplicação.

- Aplicações de Voz sobre IP (VoIP) necessitam que os parâmetros de *delay* da rede estejam abaixo de um valor recomendável para uma conversação (menor que 250ms, detalhes em [20]).

- Quanto maior o valor medido do atraso, pior será a manutenção de altas taxas de transmissão nas camadas de transporte (camada TCP). Logo a medida de atraso tem fundamental importância para o acompanhamento do desempenho dos protocolos de transporte.

- Tendo estabelecido um *delay* mínimo entre dois pontos, variações de *delay* sobre este valor podem indicar o congestionamento no caminho onde passam os pacotes da medida. Este valor mínimo de *delay* é entendido como sendo a soma do tempo de propagação, tempo de processamento e de codificação dos dados do pacote.

A seguir estão algumas comparações entre a medida *one-way delay* e a medida *round-trip delay*:

- O caminho do emissor até o destino muitas vezes é diferente do caminho entre o destino e o emissor. Esta característica assimétrica

pode ser verificada tanto pelo número de *hops* como pela diferença do comportamento de filas (muitos enlaces apresentam saturações em apenas um sentido). Assim a medida *one-way delay* pode refletir melhor as características de cada caminho em relação a medida *round-trip packet loss*.

- Muitos protocolos de aplicações como o FTP e o HTTP possuem características assimétricas, logo seu desempenho pode ser melhor avaliado por medidas de *one-way delay*.
- A medida *one-way delay* é mais difícil de ser feita, uma vez que requer dois pontos de medida e um bom sincronismo de relógio entre eles.
- Para o usuário da rede a medida *round-trip delay* é mais fácil de ser realizada e interpretada.

Maiores detalhes podem ser obtidos em [107], [109] e [23].

4.2 Perda de Pacotes

A perda de pacotes é calculada sobre o número de pacotes perdidos dentro de um universo de pacotes enviados. O pacote é considerado perdido quando o destino não o recebe por um tempo limite pré-determinado, conhecido como *time-out*. O *time-out* irá depender da distância, do caminho por onde passa, da capacidade de canalização e o tamanho das filas entre os dois pontos de medida.

Assim como o atraso, a perda de pacotes pode ser feita de duas maneiras:

(i) *One-Way Packet Loss*: É a relação dos pacotes que são considerados perdidos em relação aos pacotes recebidos com sucesso apenas no caminho de ida entre o emissor e o destino. Esta medida geralmente é apresentada em porcentagem de um universo de pacotes enviados em um certo período de tempo. Na atual tecnologia e constituição da Internet o tempo de *time-out* para esta medida está no máximo em torno dos 2 segundos.

(ii) *Round-trip Packet Loss*: É a relação dos pacotes que são considerados perdidos em relação aos pacotes que são recebidos com sucesso pelo emissor até o destino e novamente do destino até o emissor. Esta medida, geralmente, é feita em percentagem de um universo de pacotes

enviados em um certo período de tempo. O *time-out* para esta medida geralmente não ultrapassa os 4 segundos, ou seja, se um pacote demorar mais de 4 segundos entre chegar até o destino e retornar, ele é considerado perdido.

Motivações para a medida de perda de pacotes:

- As aplicações de Voz sobre IP (VoIP) necessitam que os parâmetros de perda de pacotes da rede estejam abaixo de um valor para uma conversação aceitável (menor que 10%, detalhes em [20]).
- O aumento no índice de perda de pacotes pode indicar que a rede por onde a medida foi feita apresenta algum tipo de problema. Este problema pode ter sido causado por congestionamento ou erros de transmissão.
- A medida de perda de pacotes pode ser útil na avaliação dos algoritmos de retransmissão no que tange a eficiência dos protocolos de transporte.
- Em Backbones Internet este parâmetro pode ser utilizado para indicar desempenho.

A seguir estão algumas comparações entre a medida *one-way packet loss* e a medida *round-trip packet loss*:

- O caminho do emissor até o destino muitas vezes é diferente do caminho entre o destino e o emissor. Esta característica assimétrica pode ser verificada tanto pelo número de *hops* como pela diferença do comportamento de filas (muitos enlaces apresentam saturações em apenas um sentido). Assim a medida *one-way packet loss* pode refletir melhor as características de cada caminho em relação a medida *round-trip packet loss*.
- Muitos protocolos de aplicações como o FTP e o HTTP possuem características assimétricas, logo seu desempenho pode ser melhor avaliado por medidas de *one-way packet loss*.
- A medida *one-way packet loss* é mais difícil de ser feita, uma vez que requer dois pontos de medida e um bom sincronismo de relógio entre estes.
- Para o usuário da rede a medida *round-trip packet loss* é mais fácil de ser realizada e interpretada.

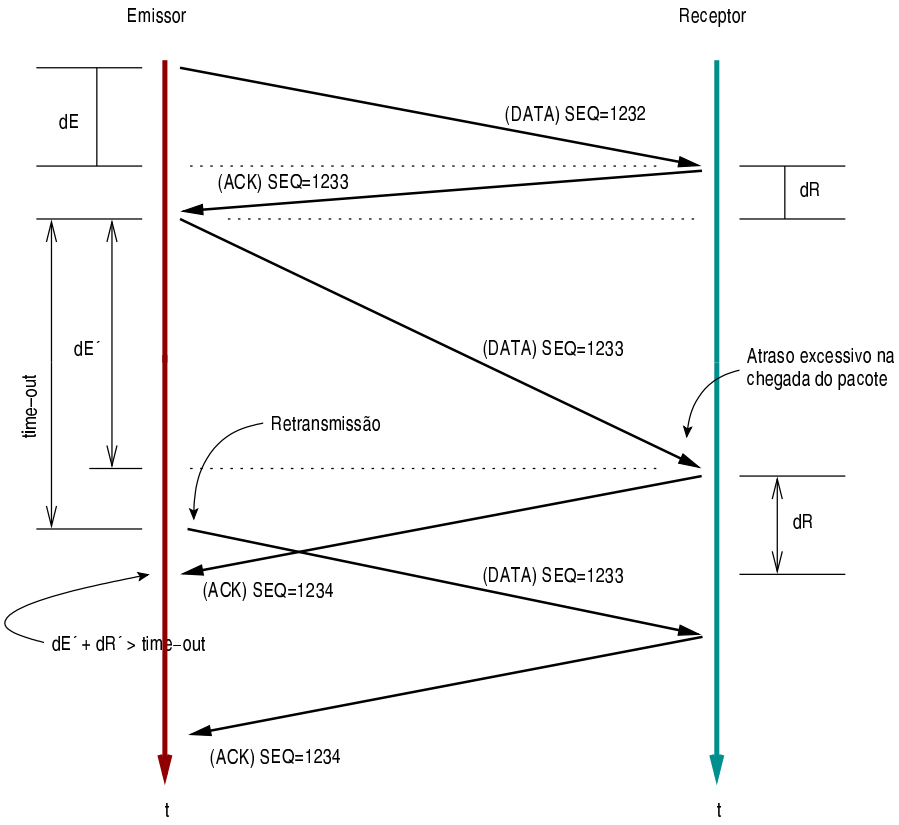


Figura 4.2: Retransmissão por *time-out*

Maiores detalhes podem ser obtidos em [108] e [23].

4.3 Retransmissão

Em termos quantitativos o tráfego do protocolo TCP correponde hoje a mais de 80% do tráfego total de bytes transferidos pela Internet, o que torna a avaliação do seu tráfego e do seu desempenho pontos muito importantes em um Backbone IP.

Uma das funções do TCP é a de realizar um transporte de dados em modo confiável ou conhecido também como "orientado a conexão". O protocolo TCP utiliza a confirmação da sequência de envio para implementar o transporte "orientado a conexão". Esta confirmação é feita através do receptor que ao receber os dados de sequência X envia um pacote de reconhecimento com o número de sequência $X + 1$ (veja figura 4.2). Para a identificação do pacote de reconhecimento o bit "`ACK`" (ACK*nowledgement*) é colocado em 1.

Uma outra função do TCP é a de realizar o controle de fluxo de dados.

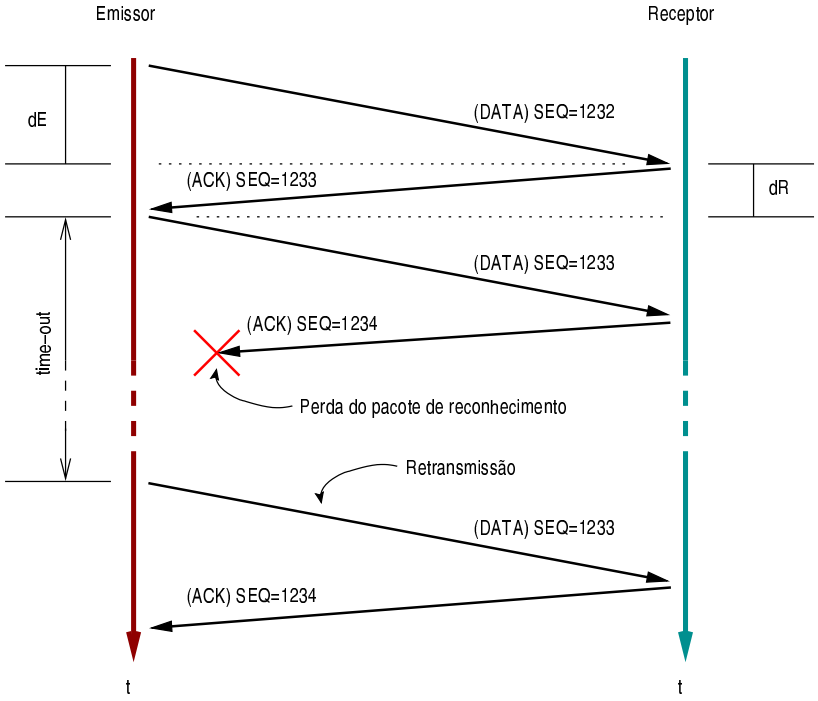


Figura 4.3: Retransmissão por perda do pacote de reconhecimento

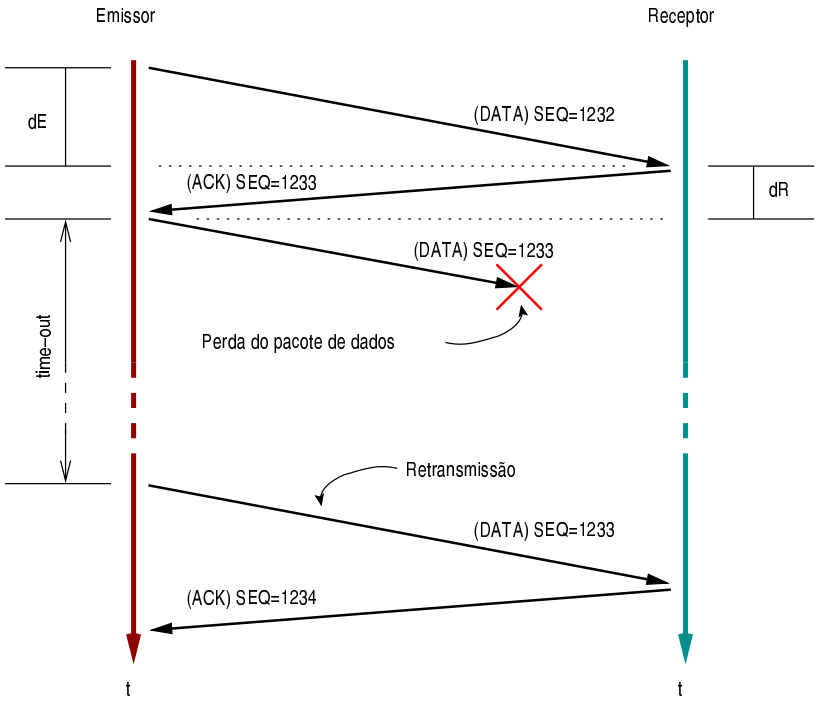


Figura 4.4: Retransmissão por perda do pacote de dados

Existem vários algoritmos que implementam o controle de fluxo de dados no TCP, eles estão descritos na referência [106]. O funcionamento básico destes algoritmos é "tentar" detectar um possível congestionamento no caminho, para então reduzir a taxa de envio de pacotes. Esta detecção é feita basicamente através da composição de três variáveis:

- Tempo de chegada de um pacote de confirmação;
- Quantidade de pacotes que ainda não receberam confirmação;
- Quantidade de pacotes que foram considerados perdidos por não receberem pacotes de confirmação (perda por time-out figura 4.2).

A restransmissão ocorrerá sempre que o emissor não receber um pacote de confirmação da sequência de dados enviada anteriormente. (Veja as figuras 4.2, 4.3, 4.4).

A restransmissão pode ocorrer por três motivos:

- O tempo é muito grande entre o envio e a chegada da confirmação (figura 4.2). Geralmente não ocorre pois o *time-out* é muito maior que o tempo gasto para o trânsito dos pacotes na maioria das redes na Internet ($dE + dR$).
- Perda do pacote de confirmação enviado pelo receptor ao emissor (figura 4.3). Ocorre quando existiu um problema no caminho no sentido do receptor para o emissor.
- Perda do pacote de dados enviado pelo emissor ao receptor (figura 4.4). Ocorre quando existiu um problema no caminho do sentido do emissor para o receptor.

A avaliação da taxa de pacotes retransmitidos pode indicar uma deficiência nos algoritmos de controle de fluxo do TCP ou uma subcapacidade de algum enlace/conexão (veja figura 4.5).

Para um backbone Internet a retransmissão é muito importante de ser medida. Quanto maior for a quantidade de pacotes retransmitidos maior será a quantidade de dados que estarão trafegando para serem descartados, ou seja, a carga útil diminui.

Criando-se uma matriz de tráfego com a quantidade de retransmissões, pode-se identificar qual é a rede para a qual o backbone deve melhorar a sua conexão. Muitos backbones contratam serviços de conexão com outros

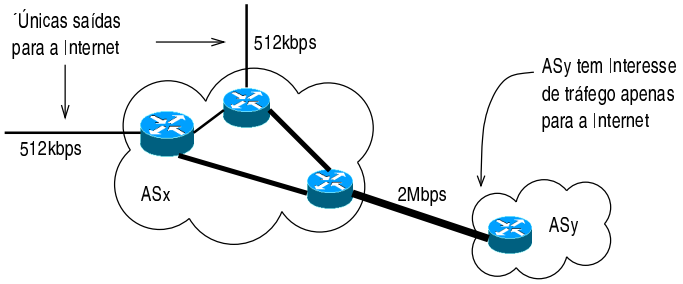


Figura 4.5: Conexão sobrevendida de 2Mbps provida pelo ASx para o ASy

backbones que funcionam como trânsito. Possíveis gargalos nos enlaces do backbone contratado não podem ser visualizados a não ser pela taxa de retransmissão. Muitos prestadores de serviço vendem velocidades de conexão que não correspondem às suas saídas, por exemplo, vendendo conexões de 2Mbps tendo apenas dois circuitos de 512kbps para a sua saída. Com a matriz de tráfego das retransmissões é possível detectar esta "sobrevenda". Na figura 4.5 podemos ver a sobrevenda que o ASx fez para o ASy, considerando que o ASy só tem interesse de tráfego para a Internet (para fora do ASx).

Outro ponto importante é onde coletar os dados. Nas figuras 4.3 e 4.4 podemos ver a retransmissão ocorrendo tanto por causa de uma perda do pacote de reconhecimento como pela perda do pacote de dados.

Colocando-se o coletor de dados perto do receptor a retransmissão não será percebida se os dados forem descartados durante o envio, porém se o coletor for colocado perto de onde "saem" os dados a retransmissão será detectada nos dois casos. Uma boa prática é colocar o coletor o mais próximo de onde os dados estão saindo, ou perto das fontes de informação as quais os clientes irão acessar.

4.4 Vazão (*Throughput*)

A vazão (ou *throughput*) é definida como sendo a quantidade de dados que passam dentro de um sistema durante um certo intervalo de tempo. Este sistema pode ter um ou vários elementos geradores de tráfego.

A medida de vazão pode se feita em âmbito global (ex: vários roteadores) ou local (ex: apenas uma interface). Geralmente ela é feita em bits por segundo (bps) ou em pacotes por segundo (pps).

Para avaliar o desempenho dos roteadores e equipamentos de rede, a medida da vazão em pacotes por segundo é melhor do que a de bits por

segundo, pois o processamento nestes equipamentos é feito baseado no *header* de cada pacote IP que chega pela sua interface. Por outro lado a vazão em bits por segundo é mais interessante de ser utilizada quando se quer avaliar a capacidade dos enlaces, uma vez que eles são especificados em bits por segundo (a utilização de octetos ou bytes por segundo é opcional e apenas reflete o valor em bits por segundo dividido por 8).

Existem basicamente duas maneiras de se medir a vazão dentro de uma rede. Uma é feita passivamente e a outra ativamente. A medida passiva é realizada através da utilização de apenas uma máquina, a medida ativa é feita com uma ou duas máquinas estrategicamente localizadas.

A medida passiva é feita através da coleta dos pacotes passantes por um enlace ou dispositivo de rede. A vazão entre dois pontos é medida filtrando-se todos os outros pacotes que não pertencem ao fluxo deste tráfego fim-a-fim a ser analisado. A medida da vazão de forma passiva utilizando apenas uma máquina é a mais simples, mas com ela não é possível a medida da capacidade máxima entre dois pontos.

A medida ativa é feita quando se deseja medir a capacidade máxima ou vazão máxima entre dois pontos. Existem dois métodos de se medir a vazão máxima: utilizando uma máquina ou utilizando duas máquinas.

A medida da vazão máxima utilizando apenas uma máquina é mais simples, porém só disponibiliza a medida da vazão em apenas um sentido e em alguns casos pode apresentar erros. Este método é feito através do envio de uma série de pacotes UDP de tamanho variável. O roteador ou dispositivo de rede ao receber o pacote responde com uma mensagem `ICMP Destination Port Unreachable`. Variando-se o tamanho dos pacotes e medindo os tempos de recepção é possível estimar a capacidade máxima do canal até o destino (apenas no sentido de ida). A principal vantagem de utilizar esta medida é na análise de vazões entre pontos remotos os quais a princípio não se tem acesso.

A medida da vazão máxima utilizando duas máquinas é mais complexa principalmente por necessitar de duas máquinas estrategicamente localizadas e bem sincronizadas , porém os resultados são mais corretos além de possibilitar a medida de vazão em ambas as direções. A medida de vazão é feita através de uma série de pacotes enviados para a máquina destino, que apenas retorna a quantidade de pacotes que não chegaram. As duas máquinas funcionam como uma conexão TCP, sem a necessidade de controlar o fluxo corretamente mas sim a de "estressar" o canal ao máximo. Após a medida

é possível retirar dados sobre a capacidade do canal em relação ao tamanho dos pacotes, além de obter a taxa de pacotes descartados.

É interessante realizar a medida de vazão máxima para avaliar a capacidade dos enlaces do Backbone e a capacidade de processamento dos roteadores. As especificações dos roteadores são descritas em pacotes por segundo, então para se calcular a vazão máxima de uma porta é necessário que se tenha o valor médio do tamanho do pacote utilizado na rede. Como exemplo deste cálculo imagine um roteador especificado para comutar até 10Kpks/s sobre uma interface FastEthernet (100Mbps). Se a média do tamanho do pacote IP for de 10Kbits (1250bytes) a capacidade máxima que o roteador conseguirá enviar para esta porta será de 100Mbps, porém se a média do tamanho do pacote IP for 1Kbits (125bytes) a capacidade máxima que o roteador consiguirá enviar será 10Mbps !

O principal ponto está em entender que se uma interface de um roteador está especificada como sendo de 622Mbps não significa que sua vazão máxima seja esta, pois a "taxa" descrita como 622Mbps só indica a velocidade de transmissão e não a capacidade máxima de transmissão. São conceitos realmente confusos, e só percebe-se a diferença quando o tráfego aumenta e a interface não consegue entregar o que "promete".

4.5 Capacidade de Rajadas (*Bulk Transfer*)

A medida de rajadas pode ser entendida como uma medida de vazão de forma ativa, porém focada para as grandes variações da vazão. Esta medida é interessante de ser feita para avaliar a capacidade do caminho de suportar variações de tráfego e também para avaliar as técnicas de descarte de pacotes nas filas, como por exemplo o RED (*Random Early Discard* [36]), WFQ (*Weighted Fair Queuing*) e outros (veja [12] e [21]).

A medida de rajadas irá ajudar a validar serviços IPs que garantem taxa média e taxa de pico, como por exemplo transporte de IP sobre *Frame-Relay* ou serviços do tipo *Web Farm* com portas Ethernet compartilhadas.

Como a medida de vazão de forma ativa, o método para a medida de rajadas pode ser feito com duas ou apenas uma máquina. O método de medida utilizando apenas uma máquina é feito através do envio de pacotes UDP com variações na taxa de envio e no tamanho dos pacotes. Este método é mais simples porém pode apresentar erros e apenas reflete a medida de rajada em um sentido (o de ida). O método utilizando duas máquinas é mais

completo e apresenta resultados melhores.

Os parâmetros que devem ser utilizados para definir a capacidade de rajadas ainda não estão padronizados e constituem um dos assuntos discutidos no grupo de trabalho IPPM [6].

4.6 Disponibilidade

Um parâmetro muito importante a ser medido é a disponibilidade dos recursos utilizados pelo prestador de serviços Internet. Atualmente são oferecidos inúmeros serviços Internet, porém os que são voltados para manter o núcleo da rede dos Backbones Internet são os mais críticos pois envolvem basicamente o transporte de todos os outros serviços. Nesta sessão serão abordados apenas os recursos de rede.

Para pessoas leigas pode parecer pouco importante a medida de disponibilidade dos recursos de rede do Backbone Internet, porém a verdade é que atualmente estes Backbones têm crescido com uma velocidade muito grande, tornando a estrutura da rede cada vez mais complexa. Neste crescimento não é só a complexidade que aumenta, mas também os equipamentos que mudam de uma escala para outra, ou seja, um Backbone Internet para atender 100 acessos dedicados tem características técnicas diferentes de um Backbone para atender 5000 conexões. Com esse crescimento as modificações na topologia, no roteamento, nos equipamentos e nos recursos de softwares são constantes. Muitas empresas utilizam períodos de janelas de manutenção para realizar tais alterações no Backbone, como por exemplo o Backbone Internet da Embratel [85] que tem um período de 4:00 as 7:00 da manhã para realizar manutenções todas as terças e quintas da semana.

Muitas empresas fazem acordos contratuais com clientes para garantir a disponibilidade, logo a sua medida é muito importante tanto para o prestador de serviços como para o cliente que deseja avaliar a medida por sí só (conhecidos como acordos SLA - *Service Level Agreement*).

Ainda não foi criada e nem descrita uma maneira correta de realizar a medida de disponibilidade, causando uma série de discussões entre prestadores de serviço e clientes. Este assunto vem ganhando ainda mais força pelos recentes acordos SLA firmados entre o prestador e o cliente. Atualmente, mesmo com esse crescimento, não existe nenhum movimento específico para a padronização da medida de disponibilidade, talvez o mais

indicado seria o grupo de trabalho do IETF chamado IPPM [6].

Os parâmetros utilizados para calcular a disponibilidade variam de empresa para empresa, e para se fazer um boa comparação de serviços é necessário uma pequena análise nestes cálculos. O método de medida de disponibilidade é outro fator importante para a verificação dos serviços prestados.

Os métodos de medida de disponibilidade dos recursos de rede são basicamente feitos sobre os aplicativos e protocolos que tem a característica de *request* e *reply*, podemos citar:

- `ping` implementado com o `ICMP`;
- `echo` implementado sobre `UDP` ou `TCP`;
- `daytime` implementado sobre `UDP` ou `TCP`;
- `snmp` implementado sobre o `UDP`;
- Pacotes `UDP` acessando portas sem servidores com recebimento de mensagens `ICMP Destination Port Unreachable` como resposta.

As medidas são basicamente feitas sobre os roteadores, podendo também serem feitas sobre dispositivos intermediários como modems, *switches*, *hubs*, rádios e conversores. A medida geralmente é feita em um intervalo que não excede 10 minutos, porém os cálculos de disponibilidades são geralmente feitos em relatórios diários, mensais e anuais.

4.7 Tamanho dos Pacotes IP

O tamanho dos pacotes é uma variável importante de ser medida, ela reflete indiretamente o comportamento dos fluxos de comunicação da rede. Abaixo estão alguns motivos para se medir o tamanho dos pacotes IP da rede:

- Para avaliação da capacidade de comutação dos dispositivos da rede é importante ter a distribuição do tamanho dos pacotes IP (veja na sessão 4.4).
- Geralmente em uma comunicação entre cliente e servidor, os pedidos feitos pelo cliente têm a característica de serem de tamanho pequeno, ao contrário do servidor que envia pacotes geralmente grandes carregando a informação pedida pelo cliente. Com a avaliação do tamanho dos

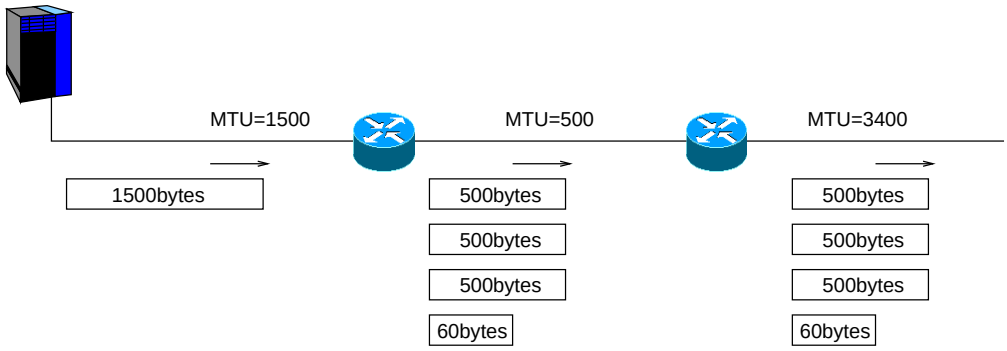


Figura 4.6: Fragmentação de um pacote IP

pacotes por fluxos de comunicação IP é possível detectar servidores e pontos de maior interesse para futuras trocas de tráfego (*peering* veja no capítulo 2.1).

- O tamanho médio dos pacotes juntamente com o índice de fragmentação dos pacotes IP que trafegam na rede podem ajudar a ajustar melhor os MTUs;

A medida do tamanho dos pacotes IP é obtida lendo-se o campo *Total Length* (detalhes em B.1).

4.8 Fragmentação dos Pacotes IP

A fragmentação é entendida como sendo a divisão de um pacote IP em outros de tamanho menor. A fragmentação ocorre quando em alguns enlaces o tamanho máximo de transmissão (MTU) é menor que o tamanho do pacote IP. Ela geralmente ocorre durante o percurso do pacote IP pela rede e é realizada por um roteador. Uma vez que um pacote IP é fragmentado ele só será remontado no seu destino mesmo que o MTU durante o percurso seja maior. Veja figura 4.6.

A fragmentação não é desejável basicamente por dois motivos:

- A cada fragmento é adicionado um novo cabeçalho IP de 20 bytes;
- A fragmentação causa um aumento no consumo de CPU e de memória do roteador, causando um aumento de *delay*;

Os campos *Fragment Offset* e *Identification* do cabeçalho IP são utilizados para fragmentar e remontar os pacotes (veja B.1).

4.9 Utilização de Recursos

Outra variável importante a ser medida é a taxa de utilização dos recursos da rede no Backbone Internet.

Abaixo estão listados alguns dos recursos mais importântes a serem monitorados;

- A quantidade de utilização do processador (CPU) é considerada um parâmetro importante, pois quanto maior for a média de utilização da CPU menor será a capacidade do roteador de tratar o processamento de excessões (ex: *flaps* de roteamento);
- A utilização das portas deve ser monitorada para evitar que seu tráfego chegue perto da saturação. O estado de saturação é ruim pois além de descartar pacotes, ocupa os *buffers* de forma irregular e aumenta o tempo de envio dos pacote.
- A taxa de erros, descarte e colisões nas interfaces podem indicar alguma anomalia no roteador e em seus enlaces.
- A memória deve ser acompanhada principalmente em roteadores com utilização de tabelas de roteamento muito grande.

A medida geralmente é feita pelo próprio dispositivo da rede que guardam os dados em variáveis MIB. O protocolo SNMP então é utilizado para disponibilizar os dados para as máquinas que irão fazer efetivamente a gerência.

4.10 Carga Útil e Eficiência (*Payload*)

A medida de carga útil (*payload*) é utilizada para avaliar a relação entre a quantidade de dados utilizados, apenas pela rede, no transporte e a quantidade de dados que foram "encapsuladas" durante o transporte. Esta medida irá indicar a eficiência da rede, indicando o quanto é gasto apenas transportando informações. Redes com baixa eficiência (baixo *payload*) são redes que necessitam de muitos dados de controle durante o transporte em relação à quantidade de informações a serem enviadas efetivamente para o destino.

Esta medida se torna ainda mais importante quando o pacote IP passa por diversas tecnologias de redes e por diferentes MTUs. A medida do tamanho médio do pacote IP também é um fator importante para se medir o *payload*.

Como exemplo podemos citar a avaliação das soluções de IP sobre redes ATM, como: LANE, MPOA, CIP, MPLS, etc. A medida de *payload* pode ser muito importante para a escolha da melhor solução. Na solução em que se associa um VC/VP a uma porta em conexões ponto-a-ponto em redes IP sobre ATM, a medida de payload pode também ser útil para a escolha do melhor tipo de encapsulamento, evitando assim possíveis *overheads* (ex: AAL5-snap ou AAL5-mux).

No trabalho de análise do tráfego na MCI foram feitas medidas de eficiência do transporte de pacotes IP sobre ATM, e concluiu-se que conforme a técnica de encapsulamento escolhida era possível melhorar a eficiência de 80% para 84,5% (detalhes em [24]).

4.11 *Hopcount* e Caminho (*Route Path*)

Na Internet a distância em que uma máquina fica em relação a uma outra é medida pelo número de roteadores, ou seja pelo número de *hops*.

Motivações para a medida de *hop count*:

- Como a rede Internet é "não orientada a conexão", quanto maior o número de hops maior a chance de um pacote ser descartado. A probabilidade de descarte aumenta quando um roteador ou enlace está saturado ou próximo da saturação.

- Outra característica importante é o *jitter* que tende a aumentar com o aumento do número de *hops*, isto se deve ao fato de que em cada roteador o pacote IP entra em uma fila de pacotes diferente.

- Aplicações que necessitam de taxas constantes de *delay* e baixa perda de pacotes devem se comunicar em distâncias pequenas, ou seja com um valor de *hopcount* pequeno.

- A medida de *hopcount* pode indicar também uma falha no roteamento, tanto externo como interno. Geralmente dentro de um Backbone Internet o número de *hops* não deve ultrapassar 4 ou 5. Pontos extremos na Internet são geralmente alcançados com menos de 40 *hops*, portanto, valores acima de 40 são considerados "infinitos".

Além da medida de *hopcount* a medida do caminho (*route path*) do envio dos pacotes é muito importante. O *route path* irá identificar como as políticas de roteamento externo e interno estão implementadas na prática.

A medida do *route path* deve ser feita com uma regularidade média, não sendo necessário fazê-la a todo instante. Uma medida completa de um caminho a um destino uma vez por dia é suficiente na maioria dos casos.

A medida do *route path* e do *hop count* podem ser feitas no mesmo processo de coleta dos dados, um apenas armazena o número de roteadores e o outro armazena a ordem e os nomes dos roteadores no caminho.

O método de medida para o *route path* e para o *hop count* pode ser feito de duas maneiras:

- O de Ida, onde apenas o caminho de ida até um destino é medido;
- O de Ida e Volta, onde o caminho de ida e de volta são coletados.

A coleta de dados de Ida e Volta é mais interessante pois mostra em muitos casos o caminho diferente que um pacote de retorno percorre em relação ao pacote de envio.

Quando uma máquina servidora de informações está ligada à Internet, geralmente o consumo de tráfego de sua saída é maior que o consumo de tráfego de entrada (ex: requisições de páginas WWW). Isto pode ser um fator importante para a escolha de um caminho diferente de retorno, por exemplo utilizando enlaces de maior capacidade. A medida do caminho dos pacotes irá policiar tais implementações.

4.12 Tabela de Roteamento e AS *Path*

A tabela de roteamento é importante de ser obtida para a avaliação do seu conteúdo, tamanho, variação e profundidade. Duas tabelas de roteamento podem ser avaliadas, a interna e a externa. A tabela externa geralmente é mais importante por apresentar um número muito grande de rotas em relação à tabela de roteamento interno. Esta tabela é geralmente obtida nos roteadores de *peering*, de distribuição e de núcleo (PR, DR e CR figura 2.1).

O conteúdo da tabela pode ser analisado para verificar as políticas de roteamento e também para verificar possíveis sumarizações de rotas. A avaliação do tamanho da tabela de roteamento é obtida através do número total de entradas ou linhas. A variação da tabela é conseguida com a taxa de entrada e de retirada de rotas na tabela de roteamento (ex: *route flaps*). A profundidade é medida através do número de ASNs contidos em cada linha da tabela, conhecido como AS *path*.

O tamanho da tabela de roteamento é um parâmetro importante para a avaliação do desempenho do processo de roteamento do roteador. Quanto maior a tabela, mais tempo ele perde "procurando" o melhor caminho para um determinado destino. O cálculo do tamanho da tabela de roteamento é feito contando-se o número de prefixos. Os prefixos são representados pelo endereço da rede com o seu respectivo tamanho em notação decimal. A notação decimal representa o número de *bits* em "1" da máscara da rede.

O AS *path* (ou profundidade da tabela) é um parâmetro importante para a avaliar o quanto os ASs "trânsito" (veja na sessão 2.2) estão próximos das redes destino desejadas.

A criação de uma metodologia para avaliar o comportamento das tabelas de roteamento será muito útil para a verificação do desempenho de tecnologias de *Tag* dos pacotes, como o MPLS [13].

Exemplo de uma tabela de roteamento externo (protocolo BGP):

```
   Network          Next Hop            Metric LocPrf Weight Path
*> 24.108.64.0/19   157.130.30.65      300      0 701 6327 10482 i
*> 24.108.96.0/20   157.130.30.65      300      0 701 1239 3602 14562 i
*> 61.4.0.0/24      157.130.30.65      300      0 701 703 10098 10132 i
*> 61.4.0.0/18      157.130.30.65      300      0 701 703 10098 10132 i
*> 194.77.218.0     157.130.30.65      300      0 701 1 8750 6705 i
*> 195.214.128.0/19 157.130.30.65      300      0 701 6453 8297 5377 8705 i
```

Na tabela de roteamento acima a profundidade do destino 195.214.128.12 é 5 e para o destino 24.108.66.25 é 3.

Abaixo segue um exemplo de possível sumarização na tabela de roteamento recebida de um AS:

```
   Network          Next Hop            Metric LocPrf Weight Path
*> 63.56.16.0/24    157.130.30.65      300      0 701 705 i
*> 63.56.17.0/24    157.130.30.65      300      0 701 705 i
*> 63.56.18.0/24    157.130.30.65      300      0 701 705 i
*> 63.56.19.0/24    157.130.30.65      300      0 701 705 i
*> 63.56.20.0/24    157.130.30.65      300      0 701 705 i
*> 63.56.21.0/24    157.130.30.65      300      0 701 705 i
*> 63.56.22.0/24    157.130.30.65      300      0 701 705 i
*> 63.56.23.0/24    157.130.30.65      300      0 701 705 i
*> 63.56.24.0/24    157.130.30.65      300      0 701 705 i
*> 63.56.25.0/24    157.130.30.65      300      0 701 705 i
*> 63.56.26.0/24    157.130.30.65      300      0 701 705 i
*> 63.56.27.0/24    157.130.30.65      300      0 701 705 i
*> 63.56.28.0/24    157.130.30.65      300      0 701 705 i
*> 63.56.29.0/24    157.130.30.65      300      0 701 705 i
*> 63.56.30.0/24    157.130.30.65      300      0 701 705 i
*> 63.56.31.0/24    157.130.30.65      300      0 701 705 i
*> 63.56.32.0/24    157.130.30.65      300      0 701 705 i
*> 63.56.33.0/24    157.130.30.65      300      0 701 705 i
*> 63.56.34.0/24    157.130.30.65      300      0 701 705 i
*> 63.56.35.0/24    157.130.30.65      300      0 701 705 i
*> 63.56.36.0/24    157.130.30.65      300      0 701 705 i
*> 63.56.37.0/24    157.130.30.65      300      0 701 705 i
*> 63.56.38.0/24    157.130.30.65      300      0 701 705 i
*> 63.56.39.0/24    157.130.30.65      300      0 701 705 i
*> 63.56.40.0/24    157.130.30.65      300      0 701 705 i
*> 63.56.41.0/24    157.130.30.65      300      0 701 705 i
*> 63.56.42.0/24    157.130.30.65      300      0 701 705 i
```

```
*> 63.56.43.0/24    157.130.30.65     300      0 701 705 i
*> 63.56.44.0/24    157.130.30.65     300      0 701 705 i
*> 63.56.45.0/24    157.130.30.65     300      0 701 705 i
*> 63.56.46.0/24    157.130.30.65     300      0 701 705 i
*> 63.56.47.0/24    157.130.30.65     300      0 701 705 i
```

A tabela de roteamento acima contém 32 linhas que poderíam ser sumarizadas em apenas duas linhas mostradas abaixo:

```
   Network          Next Hop            Metric LocPrf Weight Path
*> 63.56.16.0/20    157.130.30.65     300      0 701 705 i
*> 63.56.32.0/20    157.130.30.65     300      0 701 705 i
```

A verificação da sumarização de rotas realizadas pelo AS é um ponto importante para a redução da tabela de roteamento, reduzindo assim *flap* de rotas pequenas e o tempo gasto para *lookup* na tabela de roteamento.

4.13 Localidade e *Cache Hit*

A medida de localidade reflete a probabilidade de um certo tráfego a um destino se repetir no futuro. A localidade é proporcional a probabilidade de um pacote ir para um certo destino. Quanto maior o tráfego para um destino em relação ao tráfego total, maior será a sua probabilidade e sua localidade.

A localidade é medida em relação a um destino físico ou lógico, que pode ser: um IP destino, uma rede IP destino, uma interface ou um roteador.

Para acelerar o processo de roteamento muitos roteadores implementam uma pequena "tabela" de roteamento que tem o seu acesso mais rápido. Essas tabelas são chamadas de tabela *cache* (*route cache*). Geralmente uma rota vai para a tabela *cache* quando está sendo acessada, e recebe uma variável associada a ela que indica a sua "idade" no *cache*. As rotas mais procuradas ficam mais tempo na tabela *cache* pois sua idade é pequena, porém as redes com pouco acesso acabam sendo retiradas da tabela *cache* por apresentarem idades maiores.

Um *cache hit* é o momento em que um objeto (ex: rede IP) é achado na tabela *cache*, indicando assim que este objeto já foi acessado anteriormente em um período de tempo muito curto (em tabelas de roteamento é na ordem de milisegundos).

A medida de localidade é importante, não só para o roteamento, mas também para algumas aplicações como HTTP, transmissões de audio/vídeo e outra aplicações onde existe grande acesso a um **mesmo** conteúdo. Por exemplo, suponha que na topologia interna mostrada na figura 2.1 o roteador de acesso (AR) possua 300 usuários acessando um mesmo conteúdo localizado

em outro Backbone. Neste caso seira adequado colocar uma máquina com este mesmo conteúdo ao lado do roteador de acesso.

Hoje muitos Backbones estão implementando o *cache* para protocolos como o HTTP. Estes cache estão sendo colocados de forma transparente (*transparent proxy*) através de protocolos como o WCCP (*Web Cache Control Protocol* [37])

Uma medida que avalie o quanto é necessário um *cache* de um certo serviço, tem grande importâcia para a melhoria do desempenho do Backbone.

4.14 Fluxos de Comunicação

Através da medida do fluxo de comunicação dos pacotes IP é possível obter uma série de informações úteis, tais como:

- Uma vez com o fluxo de dados é possível escolher um melhor ponto de conexão. Por exemplo, no caso da maioria dos pacotes estarem vindo de uma rede ASx, e a conexão existente estar em ASy, é mais adequado trocar o ponto de conexão para ASx.

- Através da quantidade de dados e de pacotes dos fluxos é possível ter uma estimativa da aplicação utilizada.

- Ataques do tipo que se utiliza IPs de origem diferentes do original (IP *spoofing* [18]) podem ser detectados através da análise dos fluxos medidos.

- Analizando as *flags* do cabeçalho do protocolo TCP é possível detectar ataques do tipo DDoS (*Distributed Denied of Services* [17]).

- É possível detectar falhas nos filtros de roteamento.

Os seguintes campos podem ser utilizados para identificar um fluxo: TOS (*Type Of Service*), IP *flags*, TCP *flags*, AS origem, AS destino, IP origem, IP destino, Porta origem, Porta destino, Rede IP origem (com a máscara), Rede IP destino (com a máscara), Protocolo de transporte e Protocolo de rede.

Os campos acima descritos podem ser relacionados de forma a definir um fluxo de dados, inúmeras combinações podem ser feitas. Na figura 4.7 estão os principais tipos de fluxos utilizados em Backbones Internet.

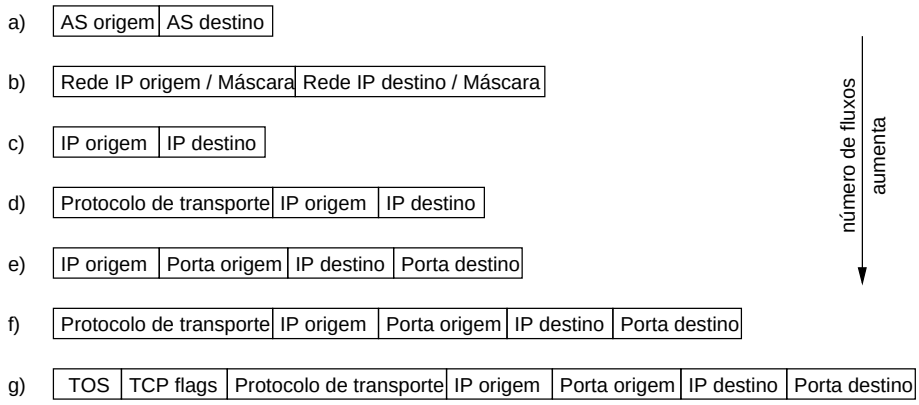


Figura 4.7: Fluxos de comunição mais comuns

O fluxo de dados pode ser do tipo mais genérico até o mais detalhado. Quanto mais detalhado o fluxo, maior será o número de fluxos encontrados na rede e consequentemente maior será os recursos de memória e CPU para analisá-los. Na figura 4.7 estão exemplos de tipos de fluxos, do mais genérico (a) até o mais específico (g).

4.15 Segurança

Atualmente um ponto importante que deve ser verificado pelo operador de Backbone é a segurança. O ponto mais importante na segurança é o policiamento da utilização dos recursos da rede. Uma rede que estiver utilizando recursos para prejudicar o desempenho do Backbone ou de uma outra rede deve ser reprimida para evitar a queda do desempenho do Backbone.

Abaixo estão descritas as 3 técnicas de ataque mais importantes que devem ser monitoradas:

- **IP *spoofing***: É uma técnica conhecida entre *hackers* para atacar ou invadir redes. Esta técnica consiste no envio de pacotes ao IP destino utilizando um IP origem diferente, geralmente um IP associado a própria rede atacada ([18]). Para se tentar acabar com tais ataques foram criadas técnicas de filtragem dos pacotes baseado no endereço IP origem. Esta técnica deve ser aplicada nas redes origem ligadas aos roteadores de acesso do Backbone (AR figura 2.1, detalhes em [100] e [19]).

- **IP *Flood***: O *Flood* de pacotes IP é um ataque utilizado para degradar

ou saturar os recursos de uma rede. Geralmente o atacante tem capacidade de recursos muito maior do que a vítima, principalmente recursos de enlaces e conexões com a Internet. A detecção de um ataque IP *Flood* não é trivial para o provedor de Backbone, pois depende de uma série de fatores, principalmente das velocidades dos enlaces finais.

- **DDoS** *Distributed Denied of Services*: Técnica utilizada de forma distribuída. O IP vítima é único e as redes origem são diversas. O DDoS é como um IP *Flood* distribuido, utilizando IP *spoofing* nas origens. Este ataque geralmente é feito utilizando o pacote TCP com a *flag* SYN ativada, ou seja, simula-se um pedido de conexão, porém com uma origem "falsificada" (detalhes em [17]).

Boas referências sobre estes 3 tipos de ataque podem ser conseguidos na referência [16]. Estes ataques podem ser detectados através da avaliação dos fluxos de comunicação nos pontos de conexão com o cliente e com os outros ASs (AR e PR da figura 2.1). Exemplo: Avaliando-se os fluxos exportados pelo Netflow [51] é possível detectar a maioria dos ataques.

Capítulo 5

Metodologia para Coleta dos Dados

Este capítulo descreve os principais métodos de coleta possíveis de serem realizados.

A importância dos parâmetros de medida e dos seus resultados dependerão essencialmente de como é feita a coleta dos dados na rede. A análise dos dados pode ficar comprometida se a coleta sugerida for incompatível.

Dependendo da topologia da rede, a localização da coleta pode influenciar nos dados obtidos e consequentemente nos resultados , exemplo: para se fazer a medida de latência entre dois roteadores de um backbone, a máquina que realizará as medidas deverá preferencialmente estar conectada diretamente a um deles. Se o medidor de latência não estiver diretamente ligado a um dos roteadores, o dado coletado pode apresentar erros. Os possíveis erros tendem a crescer se a ordem de grandeza do valor da latência entre os roteadores for da ordem de grandeza da latência entre o roteador e a máquina de coleta.

Durante a análise dos dados, técnicas estatísticas podem ser utilizadas para diminuir o erro apresentado nos dados coletados. Os erros da coleta devem estar explícitos na descrição dos resultados.

5.1 Métodos de Operação e Coleta

Nesta sessão serão definidos os métodos e modos de operação dos coletores.

5.1.1 Método de Operação Ativo

O coletor ativo é aquele que altera de algum modo a constituição inicial da rede, utilizando recursos ou introduzindo alguma perturbação. O coletor é considerado ativo mesmo que a perturbação introduzida ou os recursos utilizados sejam insignificantes para o funcionamento normal da rede.

São considerados recursos da rede aqueles que são utilizados para o transporte dos dados na operação normal da rede. O espaço físico ou a porta onde este coletor está ligado não são considerados. Em uma rede ATM os recursos podem ser CPU dos *switches*, células, agentes SNMP (ou RMON) ou atendimento a filas (priorização). Em uma rede Ethernet os recursos podem ser CPU (de placas, *hubs* ou *switches*), quadros (broadcast, multicast ou unicast), agentes SNMP (ou RMON) ou atendimento a filas (priorização). A aplicação `ping` utiliza o protocolo `ICMP` (`echo request`) para o envio de pacotes IP pela rede, logo o coletor de dados que utiliza o `ping` é ativo e utiliza os recursos de comunicação da rede (roteamento, filas, CPU, etc.).

Também podem ser considerados coletores ativos aqueles que interligam dois ou mais segmentos de rede, mesmo que eles sejam transparentes para ambos os segmentos. Um exemplo de um coletor de dados transparente é um analisador ligado entre dois roteadores, onde o analisador lê todo o quadro ou parte dele antes de enviar para o outro roteador. Este método introduz uma perturbação em forma de latência na comunicação entre os dois roteadores (*delay*). Mesmo que esta latência seja mínima, o coletor é considerado ativo.

5.1.2 Método de Operação Passivo

O coletor passivo é aquele que não utiliza nenhum recurso da rede e não introduz nenhuma perturbação no funcionamento normal da rede. Os coletores passivos ,obrigatoriamente, deverão derivar o sinal físico gerado nas interfaces a que estão interligados. Em interfaces ponto-a-ponto, sempre será necessário derivar o sinal físico para evitar a introdução de latência na transmissão de dados.

Em uma interface ótica STM1 (155Mbps) de uma rede ATM, o coletor deve utilizar um cabo em "Y" para derivar o sinal ótico gerado pelas interfaces. Este derivador ótico também é chamado de *optical splitter* ou *splitter ótico* (veja figura 5.1). Um coletor de dados ligado a uma rede Ethernet 10Base5 (coaxial) através de uma placa de rede pode ser considerado um coletor passivo se sua placa de rede não transmitir nenhum

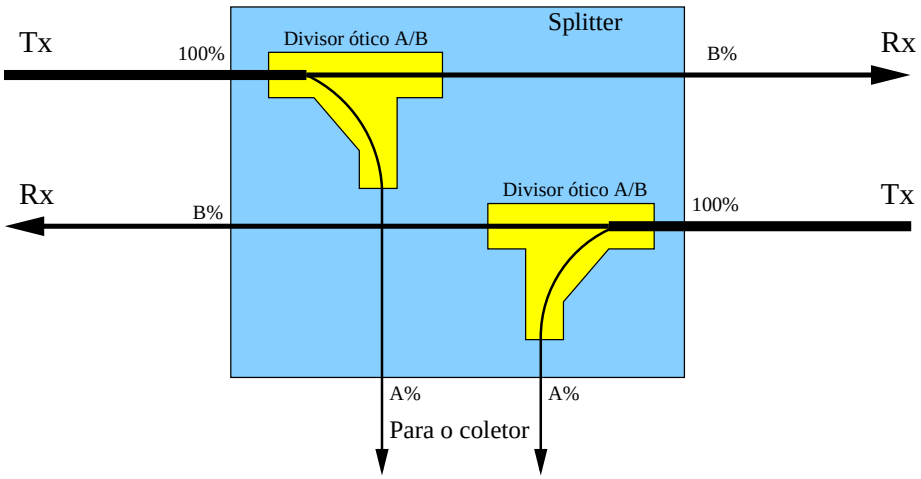


Figura 5.1: *Splitter* ótico com relação A/B

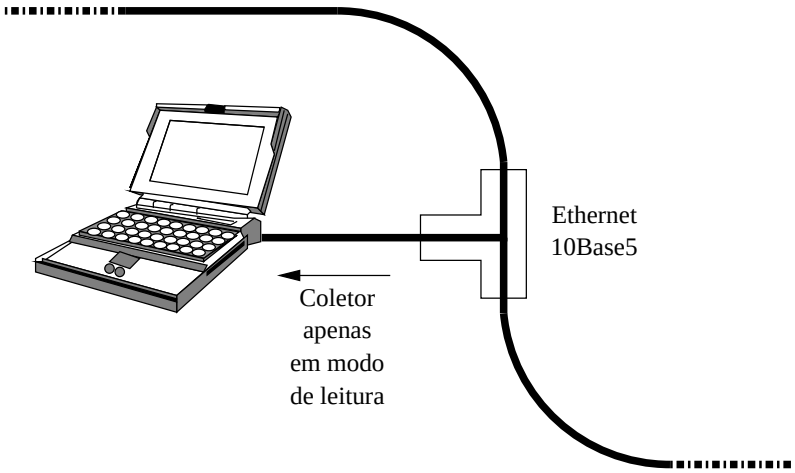


Figura 5.2: Coletor em Ethernet 10Base5

quadro (figura 5.2).

Em uma rede local com tecnologia de *hub* ou *switch* (10BaseT ou 100BaseTX), o coletor só é considerado passivo se o mesmo estiver conectado a alguma porta do concentrador (*hub* ou *switch*) e este não alterar o seu funcionamento.

Em um *hub*, o coletor sempre será passivo, pois o funcionamento do *hub* utiliza o compartilhamento do meio. O *hub* Ethernet ou FastEthernet repete em todas as portas o sinal recebido em apenas uma. Se o coletor estiver em uma destas portas apenas "escutando", ele não interferirá no funcionamento.

O *switch* não utiliza o compartilhamento de meio para o seu funcionamento. Logo, se a porta A necessitar se comunicar com a porta B, sua comunicação será independente das outras (figura 5.3). Normalmente

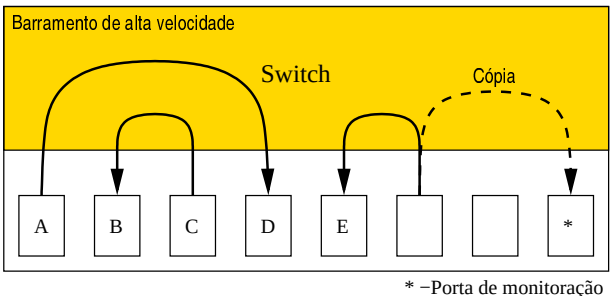


Figura 5.3: O coletor de dados no *switch*

o barramento interno do *switch* tem a capacidade muito maior do que a velocidade de cada porta, de forma a possibilitar a comunicação entre um par de portas separadamente. Portanto não é possível coletar os dados de todas as portas simultaneamente. Alguns *switches* podem ser habilitados com um comando que "espelha" o tráfego enviado por uma porta para uma outra porta específica, chamada porta espelho ou porta de monitoração (veja figura 5.3). Dependendo de como for feita a configuração da porta de monitoramento, o coletor pode alterar o seu funcionamento. É importante verificar se esta habilitação altera o modo de chaveamento do *switch* (*cut-through* ou *store-and-forward*) ou carrega a CPU de algum modo.

5.1.3 Método de Operação Destrutivo

O coletor é considerado destrutivo quando o método de coleta de dados utilizado influencia no funcionamento da rede e/ou nos dados coletados. O coletor destrutivo sempre é um coletor ativo, porém nem sempre o ativo é considerado destrutivo. A referência [102] comenta sobre a medida destrutiva.

Para que um coletor ativo seja considerado destrutivo ele deve utilizar os recursos da rede de tal forma, que altere a coleta dos dados ou o funcionamento da rede. Na prática, o tráfego gerado pelo coletor deve estar no máximo entre 0.5% a 5% da menor velocidade encontrada no caminho dos pacotes gerados pelo coletor durante a coleta. O tempo decorrido durante a geração de tráfego do coletor também deve ser considerado, principalmente quando o coletor gera muito tráfego em pequenos intervalos de tempo (rajadas ou *burst*).

Mesmo que o coletor seja considerado destrutivo, é possível estimar resultados próximos da realidade, sempre observando os parâmetros

escolhidos e a perturbação introduzida.

5.1.4 Método de Operação Não-Destrutivo

Os coletores não-destrutivos são aqueles que não introduzem grandes influências na medida e nem na rede onde estão ligados. Os coletores passivos são obrigatoriamente não-destrutivos, porém a recíproca não é verdadeira, ou seja, os não-destrutivos não são obrigatoriamente passivos.

Como mostrado anteriormente, os coletores ativos podem ser considerados não-destrutivos quando utilizam poucos recursos da rede ou introduzem baixa perturbação.

Os parâmetros que são utilizados para controlar o comportamento do coletor para que ele seja considerado não-destrutivo variam de rede para rede. Conforme a tecnologia dos componentes da rede, estes parâmetros podem ser maiores ou menores.

Exemplo 1: Os parâmetros utilizados em um Backbone IP formado por enlaces de 622Mbps são provavelmente maiores do que os de um Backbone que utiliza enlaces de 128Kbps.

Exemplo 2: Se o Backbone utilizar priorização de tráfego por controle de filas para os pacotes de medida, o parâmetro também deve considerar estes fatores.

5.1.5 Método de Coleta *One-way*

Este método de coleta é utilizado para medir características de um caminho entre dois pontos, ou seja, para tirar medidas entre um ponto e outro, unidirecionalmente. Para implementar este método de coleta é necessário que exista pelo menos dois coletores, um funcionando na emissão e outro na recepção. Por exemplo: a latência one-way (*One-way Delay* [107]) é definida como sendo o tempo em que o primeiro bit do pacote a ser enviado para o destino sai pela interface do coletor até o término da leitura do pacote pela interface do coletor de recepção.

Questões sobre o sincronismo dos relógios entre os dois coletores devem ser cuidadosamente analisadas, pois uma falta de sincronismo pode introduzir erros nos dados coletados. A próxima sessão relata estas questões.

Os Coletores ativos e passivos podem ser utilizados para implementar a coleta *one-way*, porém a implementação através de coletores ativos é mais simples. O tipo do pacote a ser enviado, o seu tamanho e o seu intervalo

de envio, podem ser determinados pelo coletor ativo. Porém em um coletor passivo, estes dados não podem ser definidos. Para implementar a medida em coletores passivos, a medida deve analisar a conversação dos protocolos acima do IP e dentro desta análise tirar resultados.

5.1.6 Método de Coleta *Round-trip*

A coleta de dados *round-trip* é basicamente a coleta de dados que apresenta características de ida de um ponto da rede e volta para o mesmo ponto. Por exemplo: as características interativas entre cliente e servidor. Como a Internet é "orientada a datagrama", o caminho de ida pode ser diferente do caminho de volta, não só nas tabelas de roteamento, mas também nas características das filas do sentido inverso.

Como exemplo, a medida de latência de *round-trip* é definida como sendo o tempo em que o primeiro bit de um tipo de pacote sai da interface de um emissor até o término da leitura de um pacote de resposta na interface de entrada do mesmo emissor (*Round-trip Delay* [109]).

Este método de coleta é utilizado para medir bidirecionalmente, características entre dois pontos. Este método de coleta pode ser implementado utilizando um ou dois coletores.

Abaixo estão algumas vantagens e desvantagens na utilização de dois coletores:

- Com dois coletores é possível controlar a priorização do processamento de medida no receptor. Exemplo: suponha que foi utilizado o protocolo ICMP *echo request* e *echo reply* como método de medida do *round-trip delay* entre os pontos A e B. O equipamento B que recebe as requisições de ICMP pode demorar a responder por motivos de priorização de CPU. Isto acontece quando o processo de resposta a pacotes ICMP trabalha no nível de usuário e o sistema operacional está sobrecarregado. Comentários sobre este problema podem ser vistos em [102] e [109];

- Com a utilização de dois coletores é necessário que estes estejam em sincronismo de relógio para que marquem os *timestamps* corretos;

- O custo pode ser maior que o dobro por questões administrativas e operacionais;

Abaixo estão algumas vantagens e desvantagens na utilização de apenas um coletor:

- Como o processamento da resposta no receptor não é controlado, os dados coletados podem apresentar erros causados pela latência de processamento do receptor;
- Os custos são menores.

5.1.7 Método de Coleta Pontual

A coleta de dados pontual é geralmente feita por um coletor de dados passivo. Este apenas escuta os dados que passam pelo meio onde está conectado. Com este método é possível criar inúmeros resultados estatísticos, porém relacionados apenas ao tráfego passante pela interface monitorada.

Dados do tipo fim-a-fim como eram medidos pelas coletas *one-way* e *roud-trip* podem perder o sentido para os coletores do tipo pontual. O foco nos coletores pontuais é no tráfego passante, os dados de coleta podem ser: redes de origem e destino, protocolos, retransmissões, utilização, fragmetação e tamanho dos pacotes.

5.1.8 Método de Coleta Agente-gerente

O método de coleta de dados pelo protocolo SNMP (*Simple Network Management Protocol*) é chamado de agente-gerente. O agente é o dispositivo que disponibiliza variáveis MIB (*Management Information Base*) através da diretiva SNMP `get response`. O gerente é a máquina que coletará os dados do agente através da diretiva SNMP `get request`. Veja no apêndice **??**.

Praticamente, a maioria dos agentes trabalha de forma passiva na atualização dos dados na MIB, ou seja ele não envia pacotes, mas apenas observa os pacotes que passam pelo seu dispositivo e assim atualiza as variáveis MIB. A Cisco implementou um conjunto de variáveis MIB que tornará o agente um coletor ativo, esta nova técnica é chamada de RTT (*Round Trip Time*).

Muitos dados de performance e caracterização de um Backbone IP podem ser conseguidos com as variáveis MIB. Existem inúmeras RFCs que descrevem novos conjuntos de variáveis MIB que podem ser incorporados aos equipamentos de rede, porém muitas delas não são possíveis de serem implementadas por motivos de consumo excessivo de CPU, como exemplo o RMON2 (*Remote Monitoring Version 2*). Existe um grupo de trabalho no IETF dedicado para as definições de MIB para RMON, veja em [10].

5.2 Sincronismo dos Coletores

O sincronismo do relógio e a exatidão nos instantes de medida são fundamentais para alguns tipos de análise. Abaixo estão as definições de alguns parâmetros do relógio do coletor que devem ser considerados na análise:

- **Offset** : É a diferença entre a hora verdadeira e a hora indicada no coletor.
- **Precisão** (*Accuracy*) : É o quanto o valor do *offset* está perto do zero.
- **Skew** : É definido como sendo a primeira derivada no tempo da variação do relógio do coletor em relação a hora verdadeira.
- **Resolução** (*Resolution*) : É definida como sendo o menor intervalo de tempo disponibilizado pelo relógio. A resolução não está relacionada com o incremento interno do clock, mas com a menor unidade fornecida pelo relógio. Para o coletor medir o tempo de passagem de um quadro em uma rede Ethernet, o relógio deve ter uma resolução compatível.

Para medidas com baixa precisão, na ordem de milesegundos, a utilização de servidores NTP (*Network Time Protocol* [96]) pode ser uma boa solução. Para maiores precisões é necessário a utilização de sincronismo por GPS (*Global Position System*).

Uma discussão mais detalhada sobre as questões do sincronismo do relógio pode ser verificada na RFC 2330 [102].

5.3 Amostragem da Coleta

Para se obter um resultado estatístico mais próximo da realidade é necessário que o conjunto de dados coletados na rede, dentro de um certo intervalo de tempo, seja previamente estudado e avaliado.

O conjunto de dados coletados em um intervalo de tempo é chamado de *sample* ou amostra, e o método é chamado de *sampling* ou amostragem. A duração do conjunto de amostras é obtido pelo somatório de intervalos de tempo entre as amostras. Considera-se o intervalo de tempo entre duas amostras quaisquer como sendo independente.

Quanto maior for a quantidade de dados obtidos na amotragem melhor será a avaliação estatística da realidade.

Quanto maior o conjunto de dados coletados, menor deverá ser o intervalo entre as medidas. O intervalo entre as medidas é um fator importante na confiabilidade da medida dos coletores ativos. Se um coletor ativo injetar na rede um conjunto muito grande de medidas, este pode estar causando interferência no funcionamento normal da rede e consequentemente alterações nas medidas de modo a torná-las não reais ou mesmo inválidas (coletor destrutivo).

Para gerar estatísticas mais próximas da realidade é necessário que o conjunto de dados coletados na rede em um certo intervalo de tempo seja previamente estudado e avaliado. O estudo prévio é fundamental para não causar interferências no funcionamento normal da rede, como por exemplo oscilações [102].

O conjunto de dados coletados em um intervalo de tempo é chamado de *sample* ou amostra, e o método é chamado de *sampling* ou amostragem. A duração do conjunto de amostras é dada pelo somatório de intervalos de tempo entre as amostras. Considera-se os intervalos de tempo entre duas amostras consecutivas quaisquer como sendo independentes entre si.

5.3.1 Intervalos Constantes

A amostragem constante é definida como tendo o seu período constante. Logo, a probabilidade do intervalo entre amostras ser igual a uma constante pré definida é igual a 1. Na equação de probabilidade 5.1 o valor K é constante e representa o intervalo de tempo entre amostragens.

$$P(T_n = K) = 1 \text{ ; onde } K > 0 \text{ e } n \geqslant 1 \tag{5.1}$$

A principal vantagem em se realizar medidas com intervalos constantes é a facilidade de implementação e análise no domínio de freqüência (ex: Transformada de *Fourier*). Porém este modo de medida pode trazer alguns problemas.

O maior problema em utilizar intervalos constantes é a interferência que esta medida pode causar no comportamento da rede. Como exemplo, a utilização de pacotes de tamanho muito grande pode causar interferências no comportamento das filas. Esta interferência constante pode causar oscilações e levar a rede a um estado de sincronização [102]. Este estado de sincronização pode alterar as medidas e o funcionamento normal da rede.

5.3.2 Intervalos Aleatórios

Intervalos Exponecialmente Distribuídos

Como o intervalo de medidas constante pode causar problemas, a técnica de gerar intervalos aleatórios pode ser utilizada.

A amostragem pode ser feita utilizando uma soma de amostras de intervalos randômicos. As amostras são separadas por intervalos independentes e com uma distribuição comum a $P(t)$. Por exemplo: na equação 5.1, a função $P(t)$ gera intervalos iguais a K com probabilidade igual a 1, este método de amostragem é chamado constante.

Uma técnica para se tentar gerar intervalos não previstos é a da utilização da distribuição exponencial. A equação 5.2 representa a probabilidade de um intervalo de amostragem ser menor que T. A constante λ representa a taxa média de amostragem, ou seja λ^{-1} representa o intervalo médio entre amostras.

$$P(T \leqslant t) = 1 - e^{-\lambda t} \text{ ; onde } t \geqslant 0 \tag{5.2}$$

Como existe a probabilidade de um intervalo ser muito grande a equação 5.3 pode ser utilizada para ajustar a saída impedindo que valores muito grandes sejam utilizados no intervalo entre medidas. A constante T_{max} expressa o tempo máximo entre amostragens.

$$G(t) = Unif(0, T_{max}) \tag{5.3}$$

Para implementar o método de amostragem de Poisson é necessário primeiro definir a taxa média de amostragem λ. Por exemplo: para um intervalo de tempo médio entre amostragens de 30 segundos o $\lambda = \frac{1}{30}$. Após a definição de λ, é necessário gerar uma série de números pseudo aleatórios com distribuição exponencial. Chamaremos estes números de E_1, E_2, ..., E_n. Então, a primeira medida será realizada no instante E_1, a segunda no instante $E_1 + E_2$, e assim por diante.

Uma técnica para gerar números aleatórios distribuídos exponencialmente é utilizar a função `rand()` para gerar valores aleatórios uniformemente distribuídos no intervalo entre 0 e 1. E então depois de gerar os valores U_1, U_2, ..., U_n, utiliza-se a equação 5.4 para calcular o seu valor "exponencial" correspondente.

$$E_i = -\frac{ln(U_i)}{\lambda} \tag{5.4}$$

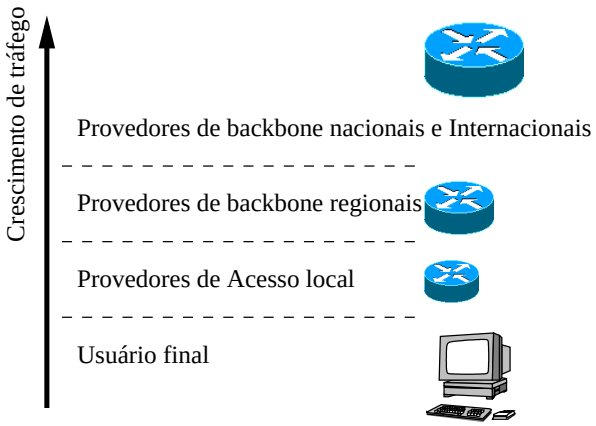


Figura 5.4: Pontos de coleta possíveis

Intervalos Geometricamente Distribuídos

Similarmente ao método de geração de intervalos exponencialmente distribuidos, a amostragem geométrica é feita através de uma probabilidade fixa. A função que define o tempo de intervalo entre medidas é uniforme. Um exemplo seria gerar valores aleatórios uniformemente distribuídos entre 0 e 1, se o resultado for menor que uma constante p previamente definida o coletor realiza a amostra.

5.4 Localização do Coletor

Para alguns tipos de medida a localização do coletor tem fundamental importância. Na figura 5.4 é possível visualizar alguns níveis hierárquicos encontrados na Internet.

Se um dos objetivos da análise for comparar o tráfego entre ASN nacionais, a localização do coletor deverá ser obrigatoriamente no provedor de backbone nacional. Porém, se o objetivo da análise for comparar o desempenho de um usuário em uma comunicação fim-a-fim, o ponto de análise deve ser no provedor local ou no usuário.

5.5 Entrega dos Dados Coletados

Outra característica importante a ser definida na escolha do coletor de dados é a forma e a velocidade com que ele deverá apresentar os dados coletados.

Após a coleta dos dados o coletor deverá disponibilizar estes dados para o analisador, que irá construir os resultados necessários, veja a figura 5.5. A

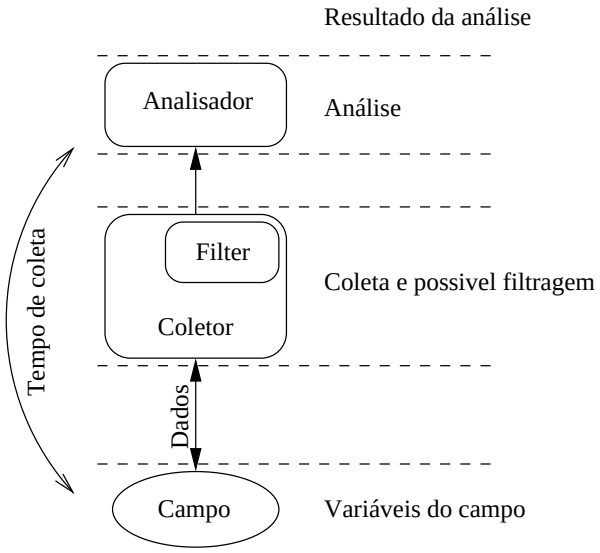


Figura 5.5: Processo de coleta e análise dos dados

velocidade de entrega dos dados coletados é definida como sendo o tempo decorrido entre a coleta dos dados e a sua posterior disponibilização.

Os dados que o analisador pode pegar com o coletor basicamente se apresentam de duas formas, a forma "crua" (*raw*) e a forma filtrada ou pré-consolidada (*filtered*).

Existe uma relação entre o formato dos dados e a velocidade de entrega deles para o analisador. Esta relação é basicamente devida às limitações de recursos encontradas no coletor.

Suponha que o coletor esteja ligado de forma passiva em uma interface ATM de 155Mbps (STM1). Se o tráfego médio desta interface for de 100 mil células por segundo significa que o coletor deve coletar em uma taxa média de 42.4Mbps. Se o coletor estiver armazenando todas as células recebidas de forma crua, isto significa uma taxa média de armazenamento de 5Mbytes a cada segundo, logo para armazenar todas as células que passam em uma hora o disco deve ter a capacidade de aproximadamente 18Gbytes.

Para taxas altas de tráfego o coletor deve ter capacidade de realizar uma filtragem, ou mesmo uma pré-consolidação. Imagine que no exemplo do parágrafo anterior o coletor fosse pegar apenas os pacotes IP encapsulados nas células transmitidas. Isso reduziria consideravelmente o tamanho de disco necessário para o armazenamento dos dados. A filtragem dos dados geralmente é feita por software e é feita na memória do coletor durante o processo de coleta para evitar latências de acesso a disco.

A velocidade em que o coletor disponibiliza os dados para o analisador

dependerá do modo de armazenamento e de como é a comunicação entre o analisador e o coletor. Dependendo da implementação alguns coletores só podem disponibilizar os dados após o término do processo de coleta, que em alguns casos pode demorar horas ou dias. Logo, se a intenção do analisador é apresentar resultados *on-line*, é necessário que o coletor seja capaz de continuamente enviar dados para o analisador sem parar o processo de coleta.

Capítulo 6

Metodologia e Parâmetros utilizados na Análise dos Dados Coletados

6.1 Propriedade dos Dados

6.1.1 A Média Aritmética

A média aritmética (ou apenas média) representa o valor da tendência central do conjunto de dados amostrados.

A representação da média não reflete claramente as características do conjunto de dados, mas apenas indica sua tendência central. Muitas vezes outras propriedades são utilizadas para representar melhor o conjunto de dados, como: mínimo, máximo, desvio padrão e outras que serão apresentadas ainda neste capítulo.

A média de um conjunto de dados amostrados, representados pela variável X, é obtida somando-se todos os valores e dividindo pelo número total de dados da amostra (n), como ilustrado na equação 6.1.

$$\overline{X} = \frac{\sum X_i}{n} \tag{6.1}$$

Onde i varia em $i = 1, 2, ..., n$.

6.1.2 A Média Ponderada

A média ponderada $\overline{X}_w$ difere da média aritmética $\overline{X}$ apenas pelo fato de existir um peso associado a cada valor X_i da variável X. O cálculo da média

ponderada está ilustrado na equação 6.2.

$$\overline{X}_w = \frac{\sum w_i X_i}{\sum w_i} \tag{6.2}$$

Onde i varia em $i = 1, 2, ..., n$, n é o total de amostras e w_i representa o peso do valor X_i da variável X.

A média ponderada pode ser utilizada para atribuição de maior peso para valores provenientes de objetos de maior importância para a rede. Exemplo: a média da latência dos roteadores pode levar em conta a topologia da rede e a importância do roteador para o desempenho.

Utilizando-se a média ponderada é possível atribuir pesos maiores aos valores obtidos mais recentemente (média deslizante no tempo). Exemplo: a *Cisco* utiliza a média ponderada para algumas estatísticas de interface. A média da taxa de bps nos últimos 5 minutos é calculada através de uma tabela qua atribui maior valor para os instantes mais recentes.

6.1.3 A Mediana

A mediana M é também uma medida da tendência central dos dados, porém o seu método de cálculo é diferente da media aritmética. A mediana é calculada ordenando os dados de forma crescente e posteriormente separando-os em duas metades iguais. Se o conjunto de dados for par, o cálculo da mediana será a média dos dois valores da intercessão das duas metades. Se o conjunto de dados for impar, a mediana será o valor que estiver exatamente no meio das duas metades.

Exemplo: a mediana dos dados apresentados na tabela 6.1 é 1040ms. Se na tabela 6.1 retirarmos o valor 1010, a mediana passará a ser 1050ms ($\frac{1040+1060}{2}$).

Tabela 6.1: Latência entre dois pontos da Internet (em ms)

1010	1020	1020	1040	1040
1040	1040	1040	1060	1060
1070	1070	1080	1100	1220

A mediana pode muitas vezes ser mais importante que a média, uma vez que não apresenta distorções quando existe um valor muito grande ou muito pequeno na amostra.

6.1.4 A Moda

A moda m é outra medida da tendência central dos dados da amostra. Ela representa o valor de maior frequência dentro da amostra.

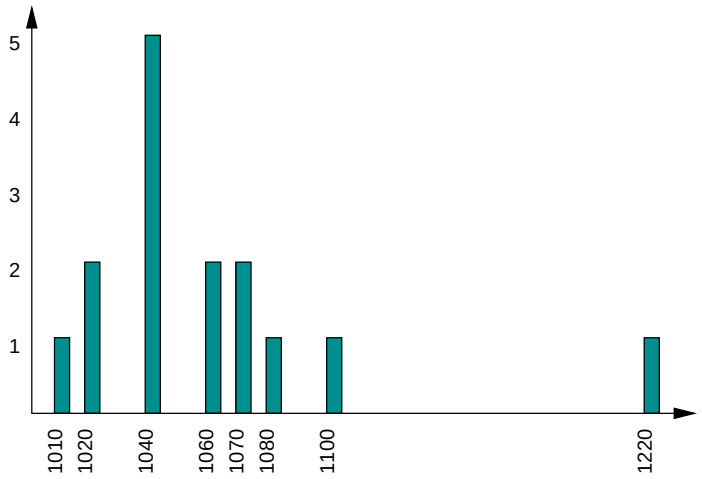


Figura 6.1: Gráfico da tabela 6.1

Na tabela 6.1 o valor da moda m é 1040. A figura 6.1 ilustra um histograma de frequências dos valores retirados da tabela 6.1. O valor da moda pode ser facilmente visualizada como sendo a coluna de maior altura.

6.1.5 A Variância e o Desvio Padrão

A variância s^2 é utilizada para representar a dispersão dos valores da amostra em relação ao valor médio $\overline{X}$. A equação 6.3 mostra como obter o valor da variância.

$$s^2 = \frac{\sum (X_i - \overline{X})^2}{n-1} \tag{6.3}$$

A dimensão obtida pelo cálculo de s^2 não tem interpretação direta, uma vez que representa o quadrado da dimensão da variável X. Para se obter uma melhor interpretação do índice de dispersão utiliza-se o desvio padrão (s), que é exatamente a raiz quadrada da variância, veja equação 6.4.

$$s = \left[\frac{\sum (X_i - \overline{X})^2}{n-1}\right]^{\frac{1}{2}} \tag{6.4}$$

Quando a quantidade de dados da amostra for muito grande, as equações 6.3 e 6.4 podem ser aproximadas para as equações 6.5 e 6.6 respectivamente.

$$s^2 = \frac{\sum {X_i}^2}{n} - \overline{X}^2 \tag{6.5}$$

$$s = \left[\frac{\sum {X_i}^2}{n} - \overline{X}^2\right]^{\frac{1}{2}} \tag{6.6}$$

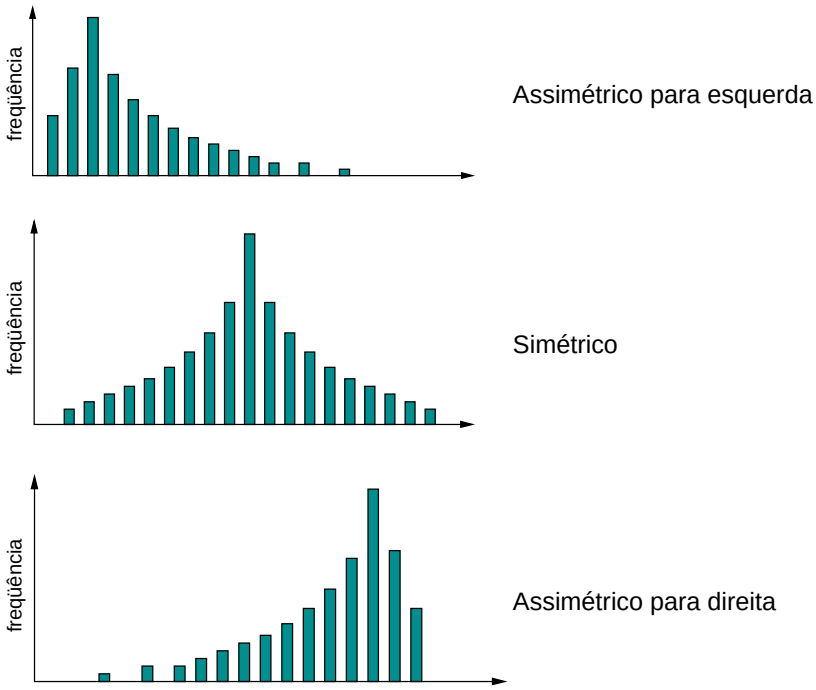


Figura 6.2: Formatos de gráficos de freqüência

6.1.6 A Simetria

É possível avaliar o histograma de freqüências de uma amostra em relação a sua simetria. A figura 6.2 ilustra 3 tipos de gráficos em relação a sua simetria: assimétrico para esquerda, simétrico e assimétrico para direita.

Um gráfico assimétrico possui os valores da média ($\overline{X}$), mediana (M) e da moda (m), diferentes. Quando o gráfico for assimétrico para direita, a seguinte propriedade pode ser observada: $\overline{X} < M < m$. E quando assimétrico para esquerda: $m < M < \overline{X}$.

A

avaliação da assimetricidade do gráfico pode ser obtida matematicamente utilizando o terceiro momento central, como ilustrado na equação 6.7.

$$a_3 = \frac{\sum (X_i - \overline{X})^3 / n}{s^3} \tag{6.7}$$

Quando o valor de $a_3 > 0$ o gráfico será assimétrico para direita, quando $a_3 < 0$ o gráfico será assimétrico para esquerda e quando $a_3 = 0$ o gráfico será simétrico.

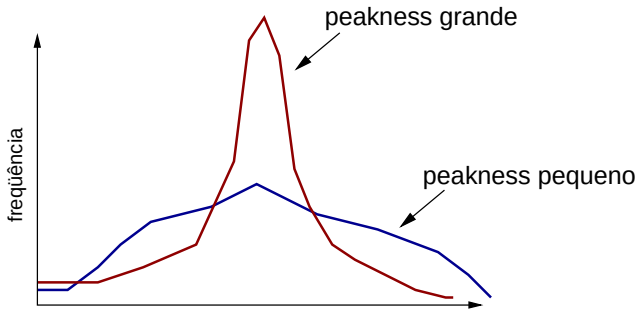


Figura 6.3: Avaliação de pico entre duas distribuições

6.1.7 *Peakness*

Outro parâmetro importante que pode ser calculado nos dados da amotra é a avaliação de pico (*peakness*). A figura 6.3 ilustra a comparação de duas ditribuições em relação a avaliação de pico.

A avaliação de pico pode ser obtida com o momento central de quarta ordem (m_4), conforme a equação 6.8.

$$a_4 = \frac{\sum (X_i - \overline{X})^4 / n}{s^4} \tag{6.8}$$

O resultado obtido pela equação 6.8 não tem dimensão e nem interpretação direta. O valor a_4 é apenas utilizado para comparação de duas distribuições.

Capítulo 7

Ferramentas Utilizadas para Coleta e Análise

Este capítulo tem o objetivo de descrever o funcionamento de algumas ferramentas utilizadas pela comunidade na medições de desempenho e na caracterização de tráfego em Backbones IP.

As ferramentas escolhidas neste capítulo são de fácil acesso e possuem o código fonte aberto (*opensource*).

Uma documentação mais aprofundada das ferramentas pode ser encontrada na Internet, através de listas de discussão, arquivos de perguntas mais freqüentes (FAQ *Frequently Asked Questions*), páginas WEB e documentos diversos.

7.1 ping

O `ping` talvez seja uma das primeiras implementações para o protocolo ICMP [93]. Atualmente é o aplicativo mais utilizado para diagnosticar problemas de conectividade na Internet.

O `ping` pode ser encontrado na maioria dos sistemas operacionais que utilizam o protocolo TCP/IP, porém o modo como ele funciona e seus parâmetros diferem de sistema para sistema.

O seu funcionamento é de maneira ativa e utiliza basicamente dois tipos de mensagem do protocolo ICMP, estas mensagens são chamadas na RFC792 [93] (apêndice **??**) como `Echo` e `Echo Reply` . Estas mensagens são respectivamente a requisição e a resposta. Muitos autores, para evitar confusões, chamam a mensagem ICMP `Echo` de ICMP `Echo Request`.

Para fazer o teste de conectividade o aplicativo `ping` envia uma mensagem

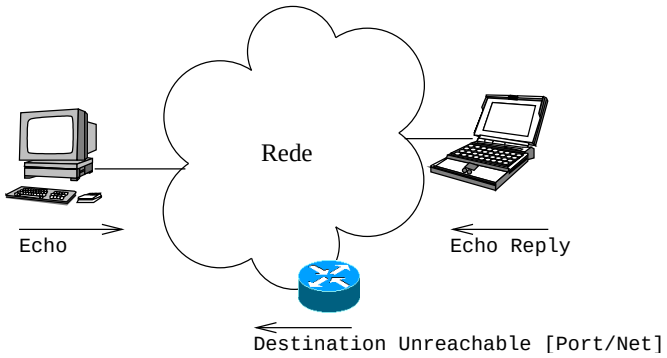


Figura 7.1: Mensagens ICMP recebidas e enviadas pelo aplicativo `ping`

ICMP `Echo Request` (encapsulada em um pacote IP) para o destino. A máquina destino então irá responder com uma mensagem ICMP `Echo Reply` para o emissor.

Caso um roteador não consega encaminhar um pacote IP, por motivos de falta de localização em sua tabela de roteamento, este roteador envia uma mensagem ICMP `Destination Unreachable [Host]`. Logo o aplicativo `ping` além de interpretar mensagens ICMP `Echo Request` recebidas, deve interpretar também mensagens ICMP de erro, como por exemplo `Destination Unreachable [Host]` e `Destination Unreachable [Net]`. Veja figura 7.1.

Como mostrado na figura 7.2 o ICMP encontra-se na camada de rede, sendo encapsulado diretamente no pacote IP.

Em operação normal o aplicativo `ping` retorna o tempo de *round-trip* que equivale ao tempo decorrido do envio de um pacote ICMP `Echo` até o recebimento de um pacote ICMP `Echo Reply`. O parâmetro *time-out* pode ser utilizado para evitar que o aplicativo espere muito pela resposta, ultrapassando este *time-out* o pacote é considerado perdido.

Utilizar o aplicativo `ping` para medida de latência pode não ser o ideal. Atenção especial deve ser tomada em relação a como a rede irá tratar os pacotes ICMP e como o sistema remoto irá responder a estas requisições. Muitos sistemas operacionais (como o IOS da Cisco) respondem a pacotes ICMP com menos prioridade que outros processos que rodam na mesma máquina. Algumas vezes uma instabilidade no protocolo de roteamento pode causar aumento de CPU e consequentemente o sistema operacional aumenta o tempo de atendimento a processos não prioritários (veja página 7 em [109]).

É importante também que os pacotes ICMP carreguem em seu campo de dados um conjunto de dados não redundantes, ou seja, não facilmente

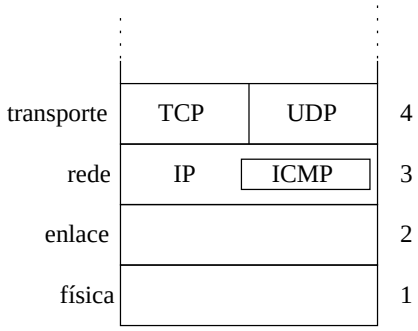


Figura 7.2: Posicionamento do protocolo ICMP na pilha TCP/IP

comprimíveis, como por exemplo a utilização de dados apenas com 0s ou 1s. Esta atenção deve ser levada em conta para a medida de latência entre dois pontos, pois no caminho podem existir enlaces que utilizam compressão. Utilizando um formato de dados o menos redundante possível pode-se conseguir uma resposta de tempo de *roud-trip* mais próximo do tempo em que as aplicações de alto nível irão utilizar.

O TTL (*Time To Live*) [92] do pacote IP representa o número de roteadores em que um pacote IP pode passar sem ser jogado fora. Na prática quando um pacote IP passa por um roteador na rede o valor do TTL é subtraído de uma unidade.

Atualmente um pacote IP atravessa um máximo de 40 roteadores na Internet, porém é possível que por alguma falha de roteamento este valor seja maior. Na prática se diz que para o transporte de pacotes TCP encapsulados no IP o valor do TTL deve ser igual ou maior que 60 (No linux 2.2.12 é 64, BSD 4.3 é 30 , Solaris 7 (SunOS 5.7) é 255).

O maior valor possível deste campo é 255, e a maioria dos sistemas Unix utilizam TTL igual a 255 para o envio de pacotes ICMP `echo request`. Devido a esse fato pode ser que aconteça de "pingar" uma máquina, sem conseguir estabelecer uma conexão TCP.

Alguns aplicativos `ping` mostram o TTL recebido na chegada, porém este valor não deve ser utilizado como contagem de roteadores (*hop-count*). Este fato está relacionado ao modo como será tratado o campo TTL na máquina remota que receberá os pacotes de requisição de echo. A máquina pode compor o campo TTL das seguintes formas:

- Não altera: o valor do TTL recebido na origem será igual a 255 menos o número de roteadores na ida e na volta do pacote.

- Coloca em 255: este modo é como os Unix BSD fazem hoje. Neste caso o TTL recebido na origem será igual a 255 menos o número de roteadores no caminho de volta.

- Coloca em um valor não determinado: em algumas máquinas o valor do TTL pode ser colocado como os dos pacotes TCP, ou UDP.

O aplicativo `ping` utiliza os campos `sequence number` e `identifier` para distinguir os pacotes enviados. Utilizando estes campos é possível o funcionamento de mais de um aplicativo `ping` ao mesmo tempo, para isso o campo `identifier` é utilizado. O campo `sequence number` é utilizado para evitar erros causados por chegadas de pacotes fora de ordem. O aplicativo `ping` incrementa este campo de uma unidade a cada envio de um pacote ICMP `echo request`. Estes campos são mostrados no apêndice **??**.

7.2 fping

O `fping` funciona de forma parecida com o `ping`, utilizando as mensagens `Echo` e `Echo Reply` do protocolo ICMP. A principal vantagem do `fping` sobre o `ping` está na possibilidade de fornecer uma lista de endereços a serem testados (alvos), através de um arquivo ou via `STDIN`.

O funcionamento do `fping` é feito de forma paralela entre os alvos, ou seja, ele não espera a resposta de um alvo para enviar um pacote para outro alvo.

O `fping` além das opções comuns ao `ping`, possue algumas opções adicionais interessantes, como:

-i n : n representa o intervalo de tempo (em ms) entre os envios de um pacotes. Isto é importante para espalhar o tráfego de envio de pacotes no tempo, evitando possíveis saturações.

-r n : n representa o número de tentativas que serão feitas para cada alvo.

-t n : n representa o "time-out" inicial, em ms.

-e : Mostra o tempo de "round-trip" de cada resposta.

-c n : n representa o número de pacotes a serem enviados para cada alvo, e também mostra os valores mínimo, máximo e médio nos resultados.

-B n : n representa um fator multiplicativo do "time-out". Para um valor de 1.5, o valor inicial do "time-out" é multiplicado por 1.5 a cada "tentativa".

A seguir estão alguns exemplos de sua saída:

```
fear# fping -f lista.fping.txt -e -i 50 -B 2 -t 2000 -r 5 -c 5 -q
www.mci.com         : xmt/rcv/%loss = 5/5/0%, min/avg/max = 362/364/366
www.msn.com         : xmt/rcv/%loss = 5/0/100%
www.waikato.ac.nz   : xmt/rcv/%loss = 5/4/20%, min/avg/max = 579/581/586
www.ufrj.br         : xmt/rcv/%loss = 5/5/0%, min/avg/max = 1.76/2.00/2.42
www.fiocruz.br      : xmt/rcv/%loss = 5/5/0%, min/avg/max = 1.57/1.69/1.91
www.ufmg.br         : xmt/rcv/%loss = 5/3/40%, min/avg/max = 946/962/980
www.fapesp.br       : xmt/rcv/%loss = 5/5/0%, min/avg/max = 7.21/7.52/8.24
www.rnp.br          : xmt/rcv/%loss = 5/5/0%, min/avg/max = 2.51/2.67/2.75
www.embratel.net.br : xmt/rcv/%loss = 5/5/0%, min/avg/max = 3.70/4.74/7.93

fear# fping -f lista.fping.txt -e -i 50 -B 2 -t 2000 -r 5 -C 5 -q
www.mci.com         : 3630.6 3631.7 3630.2 3640.5 3630.3
www.msn.com         : - - - - -
www.waikato.ac.nz   : 5810.2 5810.8 5815.2 5797.4 -
www.ufrj.br         : 19.5 38.7 18.9 31.0 18.0
www.fiocruz.br      : 16.2 17.8 18.4 16.7 17.6
www.ufmg.br         : - - - - -
www.fapesp.br       : 72.2 72.8 71.8 76.0 73.7
www.rnp.br          : 26.2 25.6 86.5 28.1 27.1
www.embratel.net.br : 39.7 64.5 38.9 36.7 90.8

fear# fping -f lista.fping.txt -e -i 50 -B 2 -t 2000 -r 5 -C 5 -q -s
www.mci.com         : 3629.6 3639.5 3663.4 3694.8 3649.5
www.msn.com         : - - - - -
www.waikato.ac.nz   : 5826.2 5806.0 5829.2 5806.6 -
www.ufrj.br         : 97.4 21.8 28.1 19.2 17.9
www.fiocruz.br      : 16.1 18.2 15.5 17.2 17.5
www.ufmg.br         : - - - - -
www.fapesp.br       : 72.3 74.9 74.4 74.0 73.7
www.rnp.br          : 27.9 26.8 27.9 28.6 27.1
www.embratel.net.br : 48.0 39.9 63.5 72.3 45.1

       9 targets
       7 alive
       2 unreachable
       0 unknown addresses

       0 timeouts (waiting for response)
      45 ICMP Echos sent
      34 ICMP Echo Replies received
       0 other ICMP received

 1.55 ms (min round trip time)
 125 ms (avg round trip time)
 582 ms (max round trip time)
        4.539 sec (elapsed real time)
```

7.3 traceroute

A principal função da ferramenta `traceroute` é descobrir o caminho entre a origem e o destino, mostrando cada roteador no seu caminho (*hop-by-hop*).

Sua primeira implementação foi feita por Van Jacobson em 1988 por sugestão de Steve Deering. Atualmente é uma ferramenta poderosa e largamente utilizada. Existem muitas variações do `traceroute` que acrecentam outras funcionalidades.

O funcionamento do `traceroute` se baseia no modo de como algumas máquinas na rede Internet, durante o percurso, irão se comportar ao receber um tipo de pacote. O primeiro é de como os roteadores interpretam pacotes com TTL igual a 1, e a segunda é como a máquina final irá interpretar um pacote UDP enviado para uma porta inexistente. Os dois tipos de resposta que o `traceroute` recebe ao enviar um pacote UDP estão mostrados a seguir:

- Se o roteador receber um pacote IP endereçado para ele mesmo, ele apenas passa o conteúdo do pacote para o sistema operacional que irá tratá-lo. Caso ele receba um pacote endereçado para outro destino, o roteador subtrai o campo TTL de uma unidade e o roteia pela interface correspondente. Quando um roteador recebe um pacote IP com o campo TTL igual a 1, o roteador gera uma mensagem ICMP `Time Exceeded` para o endereço IP origem descrito no pacote IP que chegou ao roteador;

- Se uma máquina receber um pacote UDP endereçado a ela, ela verifica se existe algum processo que está em modo *listen* na porta destino especificada pelo pacote UDP. Se existir um processo/servidor escutando nesta porta, a máquina redireciona o pacote para o processo correspondente. Caso não exista tal porta UDP "aberta", a máquina gera uma mensagem ICMP `Destination Unreachable [Port]` para o endereço IP que originou o pacote UDP. O `traceroute` utiliza como *default* a porta 33434.

O funcionamento do `traceroute` pode ser visto na figura 7.3

Para descobrir o caminho entre origem e destino o `traceroute` gera um pacote IP com o endereço IP de origem/destino e coloca o campo TTL igual a 1. Dentro deste pacote IP é então encapsulado o protocolo UDP com uma porta origem "qualquer" (utilizado para identificar a resposta no caso de vários processos `traceroute`) e uma porta destino "inexistente" (geralmente maior que 30000). Assim que o primeiro roteador receber este pacote UDP com o TTL igual a 1 ele enviará uma mensagem ICMP de `Time Exceeded` para o IP origem. Para descobrir o caminho até o destino o aplicativo `traceroute` incrementa de 1 em 1 o valor do campo TTL. Quando o pacote UDP finalmente chegar ao destino especificado no IP destino deste pacote, a máquina destino irá verificar se existe tal "serviço" UDP na porta destino especificada. Se não existir, a máquina destino envia uma mensagem ICMP de `Destination Unreachable [Port]` para o IP origem, ao receber esta

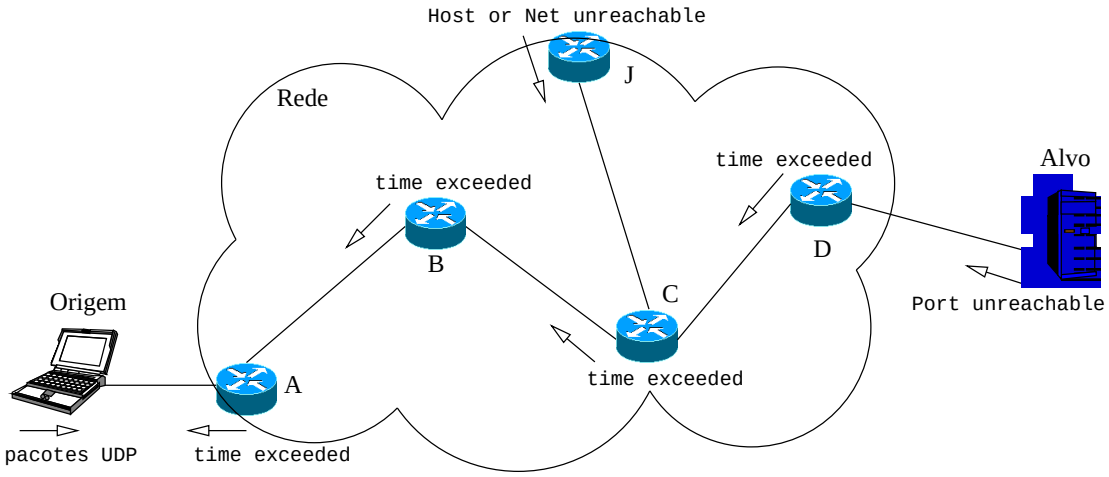


Figura 7.3: Funcionamento do aplicativo `traceroute`

mensagem o `traceroute` sabe que chegou ao destino e pára de enviar os pacotes UDP.

Abaixo estão relacionados algumas respostas possíveis de serem recebidas durante a utilização do `traceroute`. O estudo de todas as possíveis respostas recebidas pelo aplicativo `traceroute` deve ser feito de forma criteriosa para evitar que os resultados recebidos pela ferramenta não sejam mal interpretados.

- A resposta normal para o `traceroute` se apresenta abaixo:

```
[rio2]% traceroute www.fapesp.br
traceroute to www.fapesp.br (143.108.25.108), 30 hops max, 40 byte packets
 1  prov-E1-2-acc04.rjo.embratel.net.br (200.255.251.5)  0.949 ms  0.872 ms  0.814 ms
 2  ebt-F5-0-0-dist02.rjo.embratel.net.br (200.255.197.40)  2.983 ms  2.761 ms  1.282 ms
 3  ebt-A5-0-1-gsr01.rjo.embratel.net.br (200.255.197.110)  1.085 ms  1.137 ms  1.641 ms
 4  ebt-A5-2-1-gsr01.spo.embratel.net.br (200.230.2.46)  10.972 ms  10.463 ms  9.639 ms
 5  ebt-A5-0-0-1-dist03.spo.embratel.net.br (200.230.0.129)  7.798 ms  7.227 ms  8.173 ms
 6  ebt-F11-0-0-acc04.spo.embratel.net.br (200.230.162.39)  11.181 ms  12.157 ms  12.068 ms
 7  fapesp-S3-3-acc04.spo.embratel.net.br (200.246.81.22)  43.107 ms  26.618 ms  15.593 ms
 8  border5.spo.ANSP.BR (143.108.13.66)  11.425 ms  14.770 ms  19.186 ms
 9  www.fapesp.br (143.108.25.108)  8.518 ms  12.848 ms  8.577 ms
```

- Caso um roteador durante o caminho não saiba como enviar o pacote para o destino, este roteador envia uma mensagem de ICMP `Destination Unreachable [Net]` ou ICMP `Destination Unreachable [Host]` . Se o aplicativo receber um destes pacotes de ICMP como destino `[Net]` ou `[Host]` desconhecidos, o `traceroute` imprime na tela os caracteres '`!N`' e '`!H`' respectivamente. O comportamento abaixo é particular dos roteadores Cisco: Na linha 7 mesmo não tendo em sua tabela de roteamento o destino 200.244.43.35, o roteador primeiro avalia o campo TTL. Como ele é igual a 1 ele

responde com pacotes ICMP `Time Exceded`. Na linha 8 o mesmo roteador envia algumas respostas ICMP `Destination Unreachable [Port]`, assim como respostas ICMP `Destination Unreachable [Host]` '`!H`' :

```
[rio2]% traceroute 200.244.43.35
traceroute to 200.244.43.35 (200.244.43.35), 30 hops max, 40 byte packets
 1  200.255.125.30 (200.255.125.30)  13 ms  14 ms  15 ms
 2  200.255.125.229 (200.255.125.229)  15 ms  15 ms  15 ms
 3  ebt-P2-0-gsr01.rjo.embratel.net.br (200.255.197.102)  15 ms  14 ms  15 ms
 4  ebt-A12-0-0-1-dist02.rjo.embratel.net.br (200.255.197.109)  15 ms  15 ms  15 ms
 5  ebt-F5-1-acc03.rjo.embratel.net.br (200.255.197.75)  13 ms  14 ms  15 ms
 6  200.244.175.106 (200.244.175.106)  14 ms  14 ms  15 ms
 7  200.244.43.6 (200.244.43.6)  15 ms  15 ms  15 ms
 8  200.244.43.6 (200.244.43.6)  15 ms !H *  16 ms !H
```

- Se a máquina destino possuir duas interfaces de rede, com um endereço IP diferente para cada interface, e a chegada dos pacotes for feita por uma interface, e a saída por outra, a resposta ao `traceroute` irá apresentar o IP da interface que enviou o pacote de ICMP `Destination Unreachable [Port]` e não a que recebeu o pacote UDP, veja abaixo:

```
[rio2]% traceroute 200.255.151.70
traceroute to 200.255.151.70 (200.255.151.70), 30 hops max, 40 byte packets
 1  200.255.225.30 (200.255.225.30)  20 ms  19 ms  20 ms
 2  200.255.225.229 (200.255.225.229)  20 ms  20 ms  20 ms
 3  ebt-P9-5-core01.embratel.net.br (200.253.198.102)  20 ms  20 ms  20 ms
 4  200.255.151.41 (200.255.151.41)  30 ms 45ms  68 ms
```

- Se não houver resposta durante várias linhas '`*`', isto pode indicar basicamente três comportamentos: o primeiro é que o último roteador enviou o pacote adiante, porém não existe nenhuma máquina com este IP na subrede que foi roteada; o segundo é que apartir da última resposta existe um filtro que impede os pacotes UDP de serem enviados, ou os pacotes ICMP de serem retornados; o terceiro é que a máquina destino possui algum filtro ou não consegue responder a pacotes UDP destinados a uma porta inexistente. Veja um exemplo abaixo:

```
[rio2]% traceroute 200.255.124.211
traceroute to 200.255.125.211 (200.255.125.211), 30 hops max, 40 byte packets
 1  prov-E1-2-acc04.rjo.embratel.net.br (200.255.251.5)  0.818 ms  0.787 ms  0.736 ms
 2  ebt-F5-0-0-dist01.rjo.embratel.net.br (200.255.197.37)  1.755 ms  1.165 ms  1.010 ms
 3  ebt-A5-3-1-gsr01.rjo.embratel.net.br (200.255.197.114)  1.113 ms  1.103 ms  0.881 ms
 4  ebt-P4-0-0-core02.rjo.embratel.net.br (200.255.197.101)  1.822 ms  1.999 ms  1.975 ms
 5  * * *
 6  * * *
 7  * * *
.
.
.
29  * * *
30  * * *
```

- A falta de resposta pode ocorrer quando outros dispositivos pelo caminho não reponderem corretamente a pacotes UDP com TTL=1. Alguns roteadores simplesmente não implementam o ICMP `Time Excede` e consequentemente não enviam resposta para o `traceroute`. Outros roteadores respondem com ICMP `Time Exceded` utilizando um valor para o campo TTL muito pequeno, muitas vezes insuficiente para retornar. A saída para estes tipos de ocorrência pode ser vista abaixo:

```
[rio2]% traceroute www.teste.br
traceroute to www.teste.br (203.211.1.11), 30 hops max
 1  solar.jud.embratel.net.br (200.243.112.254)  1 ms  2 ms  1 ms
 8  S0-1.dist.embratel.net.br (200.243.113.2)  60 ms  69 ms  77 ms
 9  A1-1.acc.teste.br (200.243.112.17)  129 ms  137 ms  154 ms
 10  200.253.200.49 (200.253.200.49)  199 ms  180 ms  300 ms
 11  * * *
 12  * * *
 13  www.teste.br (203.211.1.11)  639 ms  579 ms  579 ms
```

- Outro problema pode ocorrer se a máquina destino não responder corretamente aos pacotes UDP. No exemplo abaixo a máquina destino responde com ICMP `Destination Unreachable [Port]` porém utiliza o valor do campo TTL igual ao TTL recebido pelo pacote UDP. Então o pacote de resposta só irá chegar ao emissor quando o TTL na máquina destino for suficientemente grande para retornar. Quando o aplicativo `traceroute` recebe pacotes com TTL igual a 1 ele indica esta ocorrência com o caractér '!', veja exemplo abaixo:

```
[rio2]% traceroute mailhost.teste.br
traceroute to mailhost.teste.br (203.211.101.19), 30 hops max
 1  solar.jud.embratel.net.br (200.243.112.254)  2 ms  2 ms  1 ms
 8  S0-1.dist.embratel.net.br (200.243.113.2)  69 ms  70 ms  77 ms
 9  A1-2.acc.teste.br (200.243.112.77)  120 ms  120 ms  126 ms
 9  Fast0-2.acc2.teste.br (200.243.112.77)  120 ms  120 ms  126 ms
 11  200.243.211.47 (200.243.211.47)  180 ms  180 ms  178 ms
 12  * * *
 13  * * *
 14  * * *
 15  * * *
 16  * * *
 17  mailhost.teste.br (203.211.101.19)  181 ms ! 179 ms ! 180 ms !
```

7.4 pchar

A ferramenta pchar é utilizada para medir características de rede dentro de um caminho entre duas máquinas na Internet. O nome pchar vem da expressão "*path characteristics*". O pchar é uma reimplementação da ferramenta pathchar escrita por Van Jacobson. A principal função destas duas ferramentas é medir parâmetros de *throughput*, *delay* e perda de pacotes através dos enlaces de um caminho na Internet.

Similarmente ao traceroute o pchar envia pacotes UDP com o campo TTL incrementado a cada *hop*. O método para a medida de *delay* e *throughput* é feito através do envio de pacotes UDP de tamanho variável. A ferramenta então calcula a perda, *delay* e *throughput* através do recebimento dos pacotes ICMP retornados pelos roteadores no percurso.

Exemplo da saída:

```
pchar to cs01.cs.embratel.net.br 200.255.125.241) using IPv4 save file
Packet size increments by 32 to 1500
46 test(s) per repetition
32 repetition(s) per hop
0:
    Partial loss:      0 / 1472 (0%)
    Partial char:      rtt = 11.452788 ms, (b = 0.008391 ms/B), r2 = 0.851766
                       stddev rtt = 0.396826, stddev b = 0.000528
    Partial queueing:  avg = 0.005124 ms (610 bytes)
    Hop char:          rtt = 11.452788 ms, bw = 953.441014 Kbps
    Hop queueing:      avg = 0.005124 ms (610 bytes)
 1: 200.255.125.30 (200.255.125.30)
    Partial loss:      287 / 1472 (19%)
    Partial char:      rtt = 12.070545 ms, (b = 0.007401 ms/B), r2 = 0.838762
                       stddev rtt = 0.367872, stddev b = 0.000489
    Partial queueing:  avg = 0.006142 ms (610 bytes)
    Hop char:          rtt = 0.617757 ms, bw = 8083.706746 Kbps
    Hop queueing:      avg = 0.001018 ms (0 bytes)
 2: 200.255.125.241 (200.255.125.241)
    Path length:       2 hops
    Path char:         rtt = 12.070545 ms, r2 = 0.838762
    Path bottleneck:   953.441014 Kbps
    Path pipe:         1438 bytes
    Path queueing:     average = 0.006142 ms (610 bytes)
```

7.5 tcpdump

O `tcpdump` foi originalmente escrito por Van Jacobson que tinha como objetivo utilizá-lo em pesquisas para melhorar a perfomance do TCP e de roteadores, dai o nome "tcpdump". Muitas melhoras foram introduzidas no seu código original e atualmente esta ferramenta trabalha em conjunto com a biblioteca `libpcap` [54]. Hoje o `tcpdump` se constitui em uma poderosa ferramenta de coleta de tráfego (detalhes [25]). Tanto o `tcpdump` como a biblioteca `libpcap` são mantidos pelo LBL [74].

7.6 tcptrace

A ferramenta `tcptrace` foi originalmente escrita por Shawn Ostermann, professor do departamento de ciência da computação da universidade de Ohio (USA) [56]. O `tcptrace` é uma poderosa ferramenta de análise de conexões TCP, avaliando até mesmo a quantidade de retransmissões por fluxo de comunicação.

A utilização desta ferramenta é feita de forma off-line, ou seja, é necessário que se faça a coleta do tráfego previamente. O `tcptrace` pode ler os dados coletados por: `tcpdump`, `snoop` (SUN), NetMetrix (HP) e Etherpeek (Macintosh). Para ler os dados armazenados pelo `tcpdump` é necessário que se tenha instalado previamente a biblioteca `libpcap`.

7.7 Cflowd

O Cflowd [53] é um conjunto de programas e bibliotecas que possibilitam a coleta de dados de fluxo (*flows*) provenientes dos roteadores Cisco que utilizam o Netflow. O conjunto Arts++ [52] foi desenvolvido para auxiliar o Cflowd. A principal função do Arts++ é estabelecer um formato para o armazenamento de fluxos coletados pelo Cflowd. Além de um formato de armazenamento o Arts++ provê também um conjunto de programas de análise. O conjunto de ferramentas do Cflowd e do Arts++ são também mantidos pelo CAIDA [65]. Nas proximas subsessões serão descritos com mais detalhes o NetFlow, o Cflowd e o Arts++.

7.7.1 O NetFlow da Cisco

Com o crescimento acelerado de redes de backbone, a colocação de medidores passivos em todos os enlaces da rede fica muitas vezes inviável. Para evitar tal necessidade a Cisco desenvolveu o NetFlow. As principais vantagens da utilização do Netflow são as seguintes:

- Sua utilização funciona como cache para acelerar os *lookups* nas tabelas de roteamento.
- Com o NetFlow não é necessário verificar as tabelas de *access-list* (apenas de entrada) toda vez que um pacote chega, ficando mais eficiente o processo de roteamento.
- É possível a exportação das informações de fluxo utilizadas pelo cache do NetFlow. Isto facilita a coleta de dados para futuras análises sem a necessidade de colocar um analisador em cada enlace.

Para que o NetFlow funcione em um roteador é necessário que sua versão do sistema operacional (chamada de IOS) seja compatível [51]. Nos roteadores Cisco não existe um comando geral que habilite o NetFlow para todas as interfaces, logo para sua utilização é necessário a habilitação

individual por interface através do comando `route-cache flow`. O processo de NetFlow funciona apenas para os pacotes de entrada, logo os dados exportados pelo NetFlow dizem respeito apenas ao tráfego entrante na interface habilitada.

Para o NetFlow o fluxo é definido como sendo um conjunto de 5 variáveis: O campo `Protocol Type`, IP origem, IP destino, Porta origem e Porta destino. Além destas 5 variáveis as tabelas do NetFlow guardam a interface destino para e a interface origem relativo ao trânsito do pacote IP.

A cada pacote IP entrante na interface, o NetFlow identifica o seu fluxo e verifica se já existe uma entrada deste fluxo na tabela de cache. Se existir, ele comuta diretamente para a interface destino especificada. Se não existir, ele então realiza um *lookup* nas tabelas de roteamento e nas tabelas de *access-list*. Se este pacote possuir alguma restrição nas tabelas de *access-list* ou se o seu IP destino não for achado nas tabelas de roteamento o pacote então será enviado para a interface `NULL` (um pacote com o destino para interface `NULL` identifica que este foi descartado). O Netflow tabém cria uma entrada na sua tabela de cache para o destino `NULL`.

Outro processo importante no NetFlow é o processo de exportação dos dados, conhecido como *flow-export*. O *flow-export* é feito através do envio de dados encapsulados em pacotes UDP. Seu destino é o IP do coletor configurado previamente no roteador. O conteúdo do pacote UDP dependerá da versão em que o NetFlow estiver funcionando. Atualmente o roteador pode exportar os fluxos criados pelo NetFlow nas versões de 1 a 8 (A descrição dos pacotes exportados das versões 1, 5, 6 e 8 podem ser vistas em 7.7.3). O momento pelo qual o roteador começa a exportar os dados de fluxo dependerá da configuração, geralmente o roteador envia uma informação de fluxo assim que ela é expirada. A informação de fluxo se expira assim que sua entrada na tabela do NetFlow é removida. A remoção da tabela do NetFlow pode ocorrer por vários fatores:

- O fluxo criado pelo NetFlow na tabela está ativo por muito tempo (até 30minutos *default*).
- O tamanho da tabela do NetFlow chegou ao limite previamente configurado.
- Não existe mais nenhum tráfego de pacotes no fluxo criado na tabela do Netflow por um certo período de tempo, conhecido como *flow time-out*.

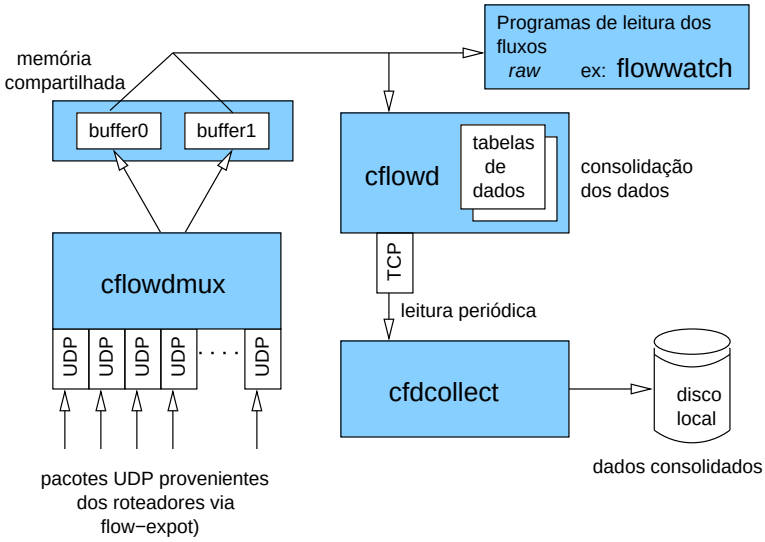


Figura 7.4: A relação entre `cflowdmux`, `cflowd` e `cfdcollect`

- Conexões TCP que tenham enviado uma mensagem de finalização (FIN) ou tenham enviado uma mensagem de *reset* (RST).

Para maiores detalhes veja [51].

7.7.2 O Conjunto de bibliotecas do Arts++

O conjunto Arts++ é formado por uma série de classes e aplicações escritas em C++ para a manipulação de arquivos gravados no formato ARTS. A estrutura de armazenamento do ARTS foi criada pelo CAIDA para facilitar o armazenamento e a manipulação de dados coletados pelo Cflowd e pelo Skitter [55]. O Arts++ pode manipular os seguintes tipos de estruturas: AS matrix, net matrix, port matrix, selected port table, protocol table, TOS table, interface matrix, nexthop table, forward IP path and RTT, BGP4 route table e RTT time series table. Para maiores detalhes veja [52].

7.7.3 O Cflowd

Basicamente o Cflowd é composto de 3 programas além de uma série de aplicativos. Três principais programas compõe o "coração" do processo de coleta:

O primeiro programa é responsável pelo tratamento inicial dos dados enviados pelos roteadores chamado de dados *raw*. Este programa é chamado de `cflowdmux`.

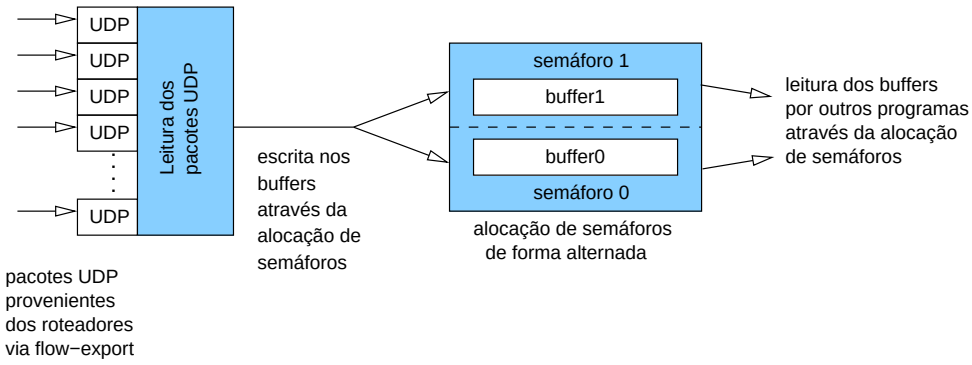


Figura 7.5: O funcionamento básico do `cflowdmux`

O segundo programa que é responsável em manter uma tabela de dados por interface de entrada para cada roteador Cisco, também é responsável para enviar os dados para o coletor central. Este programa é chamado de `cflowd`.

O terceiro programa é responsável por coletar os dados de todas as instâncias do `cflowd` e gravar em disco no formato ARTS. Este programa é chamado de `cfdcollect`.

O diagrama mostrando o relacionamento entre estes programas pode ser visto na figura 7.4.

O programa `cflowdmux`

O `cflowdmux` é responsável pela recepção dos pacotes UDP de *flow-export* enviados pelos roteadores Cisco e o seu armazenamento em uma memória compartilhada (*shared memory*). O `cflowdmux` trabalha com dois *buffers* de pacotes, para que os programas clientes possam ler os dados de um *buffer* enquanto o `cflowdmux` grava os dados em outro *buffer*. Veja a figura 7.5. O `cflowdmux` utiliza semáforos para a alocação dos *buffers*. Assim que os pacotes UDP de *flow-export* chegam, o `cflowdmux` aloca um semáforo para evitar que um processo de leitura esteja lendo um *buffer* de escrita.

O `cflowdmux` não interpreta os dados de fluxo recebidos, logo é necessário que algum programa cliente esteja rodando para interpretar, agrupar e armazenar os dados. A interpretação e agrupamento é feito pelo `cflowd` e o armazenamento pelo `cfdcollect`.

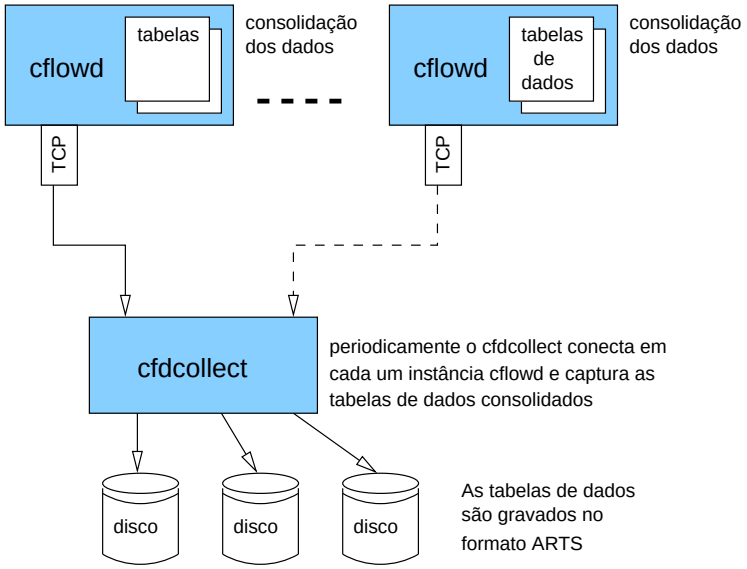


Figura 7.6: O modo de funcionamento do `cfdcollect`

O programa `cflowd`

O programa `cflowd` fica constantemente olhando os semáforos dos *buffers* de pacote criados pelo `cflowdmux`. Se algum semáforo se torna disponível o `cflowd` lê o *buffer* de pacotes, interpretando os dados e os agrupando em tabelas na memória. O processo de agrupamento é feito através de tabelas na memória e deve ser configurado previamente, quanto maior os tipos de agrupamento maior a quantidade de memória alocada. As tabelas para agrupamento podem ser nos seguintes formatos:

- **asmatrix** : matriz indicando a quantidade de pacotes e bytes por AS origem e AS destino.
- **netmatrix** : matriz indicando a quantidade de pacotes e bytes por rede de origem e rede de destino (a máscara da rede origem e destino é obtida na tabela de roteamento do roteador que exportou os dados de fluxo , logo deve-se tomar cuidado para os Backbones que utilizam sumarização).
- **portmatrix** : Bytes e pacotes por porta origem e porta destino.
- **ifmatrix** : Bytes e pacotes por interface origem para interface destino (A interface designada é o número utilizado como `ifIndex` na MIB).
- **protocol** : Tabela mostrando pacotes e bytes por protocolo IP.

- **nexthop** : Tabela mostrando pacotes e bytes por *nexthop*.
- **tos** : Tabela mostrando pacotes e bytes por *Type of Service* (TOS).
- **flows** : Fluxos sem tratamento, ou seja em formato *raw* (consomem muita memória pois não realizam agrupamento).

O tamanho das tabelas com os dados agrupados aumenta a cada vez que o programa `cflowd` lê os dados dos *buffers* e os reconhece como informação pertencente a algum agrupamento. Se por exemplo, o `cflowd` for configurado apenas com o agrupamento **portmatrix** e os dados lidos dos *buffers* representarem apenas pacotes ICMP estes dados não serão agrupados e serão descartados.

Existem duas maneiras de ler os dados agrupados nas tabelas do `cflowd`: localmente ou remotamente. A leitura local é feita através de um socket local e permite que programas rodando localmente acessem as tabelas de agrupamento (como os programas `cfdases` e `cfdnets`). A leitura remota é feita através de uma porta TCP previamente especificada (como o programa `cfdcollect`), veja figura 7.6.

Talvez o maior problema da ferramenta está no processo de limpeza das tabelas de agrupamento na memória. Esta limpeza só é feita quando existe um programa acessando remotamente o `cflowd`, ou seja com a utilização do `cdfcollect`. Se não existir coleta remota, o programa `cflowd` aloca memória indefinitivamente conforme os novos fluxos vão sendo agrupados. Se não existir informações novas de fluxo a memória não cresce. Porém devido à característica dos ataques DDoS [17] a quantidade de fluxos tem taxa de crescimento muito grande, tornando a tabela na memória infinitamente grande e causando muitas vezes o "crash" do programa `cflowd`.

O programa `cfdcollect`

Este programa é responsável pela conexão TCP até a máquina onde o `cflowd` estiver rodando, e também responsável pela leitura das tabelas de agrupamento e o armazenamento no formato ARTS em disco. Com ele é possível a centralização e o armazenamento das informações coletados individualmente pelos programas `cflowd` localizados remotamente. Veja as figuras 7.6 e 7.7.

Um importante parâmetro a ser configurado no `cfdcollect` é o período de coleta. Este período de coleta definirá de quanto em quanto tempo o `cfdcollect` irá se conectar ao `cflowd` remoto para pegar as tabelas de

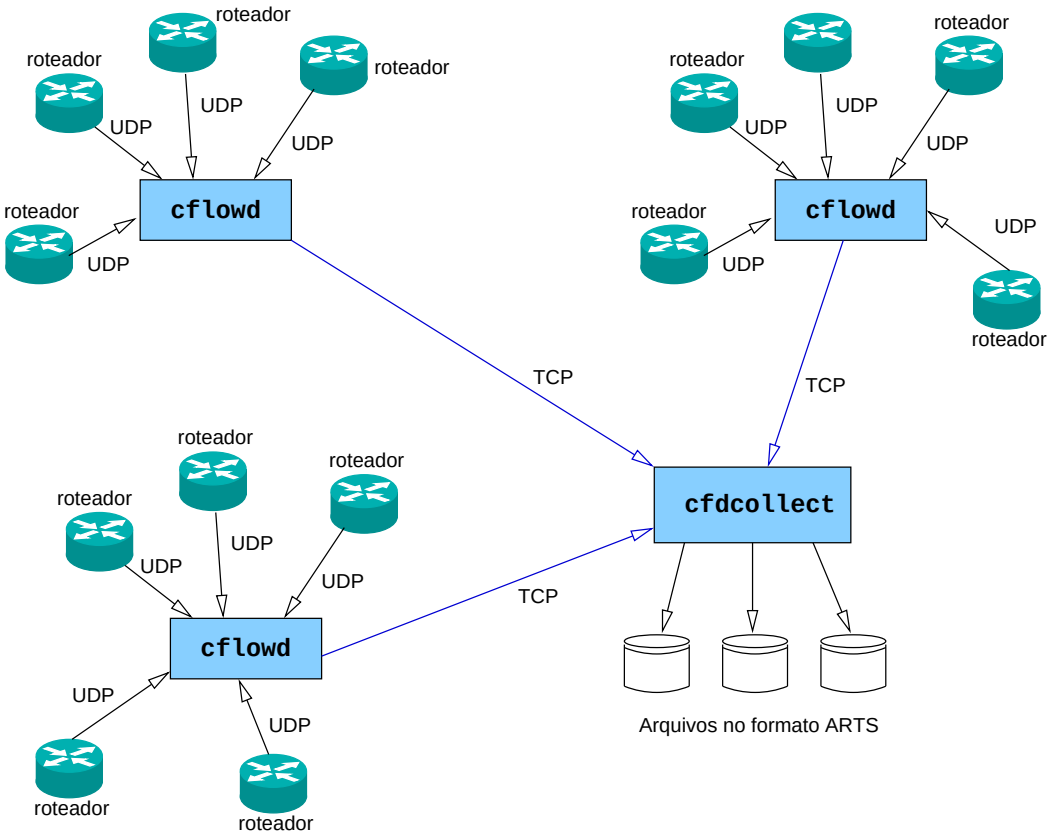


Figura 7.7: A centralização das coletas feita pelo `cfdcollect`

agrupamento de fluxo. Este período pode ser crítico quando a taxa de crescimento de fluxos for muito grande e incapaz de ser suportada pela memória da máquina onde o `cflowd` está rodando, isto ocorre durante ataques [17]. Veja a sessão 4.15.

Tipos `export-flow` interpretados pelo Cflowd

O Cflowd pode interpretar as versões 1, 5, 6 e 8 do NetFlow. As diferenças entre as versões estão basicamente no formato dos pacotes exportados. O processo de exportação dos pacotes contendo informações de fluxo pelo roteador é praticamente o mesmo, apenas a quantidade de entradas de fluxo é que varia (*flow entry*). Abaixo estão as descrições dos campos de dados dos pacotes UDP exportados pelas versões interpretadas pelo Cflowd. Estas informações foram retiradas do arquivo `CflowdFlowPdu.h`:

- Cabeçalho versão 1:

```
16bits    version        //  flow-export version number
16bits    count          //  number of flow entries
32bits    sysUptime
32bits    unix_secs
32bits    unix_nsecs
```

  Descrição de um fluxo da versão 1 (máximo de 24 entradas de fluxo por cada pacote UDP exportado (máximo 24 *flow entries*)):

```
32bits     srcaddr     // source IP address
32bits     dstaddr     // destination IP address
32bits     nexthop     // next hop router's IP address
16bits     input       // input interface index
16bits     output      // output interface index
32bits     pkts        // packets sent in duration
32bits     bytes       // octets sent in duration
32bits     first       // SysUptime at start of flow
32bits     last        // and of last packet of flow
16bits     srcport     // TCP/UDP source port number or equivalent
16bits     dstport     // TCP/UDP destination port number or equivalent
16bits     pad1
8bits      prot        // IP protocol, e.g., 6=TCP, 17=UDP, ...
8bits      tos         // IP Type-of-Service
32bits     pad2
32bits     pad3
```

- Cabeçalho versão 5:

```
16bits    version        //  flow-export version number
16bits    count          //  number of flow entries
32bits    sysUptime
32bits    unix_secs
32bits    unix_nsecs
32bits    flow_sequence  //  sequence number
8bits     engine_type    //  no VIP = 0, VIP2 = 1
8bits     engine_id      //  VIP2 slot number
16bits    reserved       //  unused
```

Descrição de um fluxo da versão 5 (máximo de 30 entradas de fluxo por cada pacote UDP exportado (máximo 30 *flow entries*)):

```
32bits     srcaddr     // source IP address
32bits     dstaddr     // destination IP address
32bits     nexthop     // next hop router's IP address
16bits     input       // input interface index
16bits     output      // output interface index
32bits     pkts        // packets sent in duration
32bits     bytes       // octets sent in duration
32bits     first       // SysUptime at start of flow
32bits     last        // and of last packet of flow
16bits     srcport     // TCP/UDP source port number or equivalent
16bits     dstport     // TCP/UDP destination port number or equivalent
8bits      pad
8bits      tcp_flags   // bitwise OR of all TCP flags in flow; 0x10
                          //  for non-TCP flows
8bits      prot        // IP protocol, e.g., 6=TCP, 17=UDP, ...
8bits      tos         // IP Type-of-Service
16bits     src_as      // originating AS of source address
16bits     dst_as      // originating AS of destination address
8bits      src_mask    // source address prefix mask bits
8bits      dst_mask    // destination address prefix mask bits
16bits     reserved
```

- Cabeçalho versão 6:

```
16bits  version         // version
16bits  count           // the number of records in PDU
32bits  sysUptime       // current time in msecs since router booted
32bits  unix_secs       // current seconds since 0000 UTC 1970
32bits  unix_nsecs      // residual nanoseconds since 0000 UTC 1970
32bits  flow_sequence   // seq counter of total flows seen
8bits   engine_type     // type of flow switching engine
8bits   engine_id       // ID number of the flow switching engine
16bits  reserved
```

Descrição de um fluxo da versão 6 (máximo de 27 entradas de fluxo por cada pacote UDP exportado (máximo 27 *flow entries*)):

```
32bits    srcaddr       // source IP address
32bits    dstaddr       // destination IP address
32bits    nexthop       // next hop router's IP address
16bits    input         // input interface index
16bits    output        // output interface index
32bits    pkts          // packets sent in duration
32bits    bytes         // octets sent in duration
32bits    first         // SysUptime at start of flow
32bits    last          // and of last packet of flow
16bits    srcport       // TCP/UDP source port number or equivalent
16bits    dstport       // TCP/UDP destination port number or equivalent
8bits     rsvd
8bits     tcp_flags     // bitwise OR of all TCP flags seen in flow
8bits     prot          // IP protocol, e.g., 6=TCP, 17=UDP, ...
8bits     tos           // IP Type-of-Service
16bits    src_as        // originating AS of source address
16bits    dst_as        // originating AS of destination address
8bits     src_mask      // source address prefix mask bits
8bits     dst_mask      // destination address prefix mask bits
8bits     in_encaps     // size in bytes of the input encapsulation
8bits     out_encaps    // size in bytes of the output encapsulation
32bits    peer_nexthop  // IP address of the nexthop w/in the peer (FIB)
```

- A versão 8 trabalha com 5 tipos de agregados, abaixo está o descrito o cabeçalho:

```
16bits    version         // flow-export version number
16bits    count           // number of flow entries
32bits    sysUptime       // current time in msecs since router booted
32bits    unix_secs       // current seconds since 0000 UTC 1970
32bits    unix_nsecs      // residual nanoseconds since 0000 UTC 1970
32bits    flow_sequence   // sequence number
8bits     engine_type     // type of flow switching engine
8bits     engine_id       // ID number of the flow switching engine
8bits     agg_method      // aggregation method
8bits     agg_version     // aggregation version
32bits    reserved        // unused
```

Descrição de um fluxo agregado por AS da versão 8 (máximo de 51 entradas de fluxo por cada pacote UDP exportado (máximo 51 *flow entries*)):

```
32bits  flows    // number of flows
32bits  pkts     // number of packets
32bits  bytes    // number of bytes
32bits  first    // sysUptime at start of flow
32bits  last     // sysUptime at end of flow
16bits  src_as   // source AS
16bits  dst_as   // destination AS
16bits  input    // input interface index
16bits  output   // output interface index
```

Descrição de um fluxo agregado por Protocolo e Porta da versão 8 (máximo de 51 entradas de fluxo por cada pacote UDP exportado (máximo 51 *flow entries*)):

```
32bits  flows     // number of flows
32bits  pkts      // number of packets
32bits  bytes     // number of bytes
32bits  first     // sysUptime at start of flow
32bits  last      // sysUptime at end of flow
8bits   prot      // IP protocol (TCP=6, UDP=17, etc.)
8bits   pad
16bits  reserved
16bits  srcport   // source port
16bits  dstport   // destination port
```

Descrição de um fluxo agregado por Rede da versão 8 (máximo de 44 entradas de fluxo por cada pacote UDP exportado (máximo 44 *flow entries*)):

```
32bits    flows      // number of flows
32bits    pkts       // number of packets
32bits    bytes      // number of bytes
32bits    first      // sysUptime at start of flow
32bits    last       // sysUptime at end of flow
32bits    srcnet     // source network
32bits    dstnet     // destination network
8bits     dst_mask   // destination netmask length (bits)
8bits     src_mask   // source netmask length (bits)
16bits    reserved
16bits    src_as     // source AS
16bits    dst_as     // destination AS
16bits    input      // input interface index
16bits    output     // output interface index
```

Descrição de um fluxo agregado por Rede de origem da versão 8 (máximo de 44 entradas de fluxo por cada pacote UDP exportado (máximo 44 *flow entries*)):

```
32bits    flows       // number of flows
32bits    pkts        // number of packets
32bits    bytes       // number of bytes
32bits    first       // sysUptime at start of flow
32bits    last        // sysUptime at end of flow
32bits    srcnet      // source network
8bits     src_mask    // source network mask length (bits)
8bits     pad
16bits    src_as      // source AS
16bits    input       // input interface index
16bits    reserved
```

Descrição de um fluxo agregado por Rede de destino da versão 8 (máximo de 35 entradas de fluxo por cada pacote UDP exportado (máximo 35 *flow entries*)):

```
32bits    flows      // number of flows
32bits    pkts       // number of packets
32bits    bytes      // number of bytes
32bits    first      // sysUptime at start of flow
32bits    last       // sysUptime at end of flow
32bits    dst_net    // destination network
8bits     dst_mask   // destination network mask length (bits)
8bits     pad
16bits    dst_as     // destination AS
16bits    output     // output interface index
16bits    reserved
```

7.8 CoralReef

O CoralReef não é apenas um programa, mas sim um conjunto de ferramentas. Este conjunto de ferramentas possui alta performance para a análise e coleta de dados de tráfego Internet. Atualmente o CoralReef é mantido e desenvolvido pelo CAIDA [65]. Este conjunto de ferramentas incluem desde programas para coleta de dados a nível físico, até programas de análise e apresentação. O conjunto de programas é de domínio público, porém a sua documentação é precária e algumas versões só estão disponíveis para os membros do CAIDA.

A coleta de dados no CoralReef é feita no modo passivo, para tal é necessário a utilização de *splitters* ópticos nos enlaces a serem monitorados. Atualmente é possível utilizar a ferramenta para coletar dados em enlaces IP sobre ATM de 155Mbps e 622Mbps (OC3 e OC12 ou STM-1 e STM-4), É possível utilizar o software para a obtenção de pacotes IPs diretamente em enlaces POS, porém é necessário utilizar as placas Dag3 desenvolvidas pela universidade de Waikato [64].

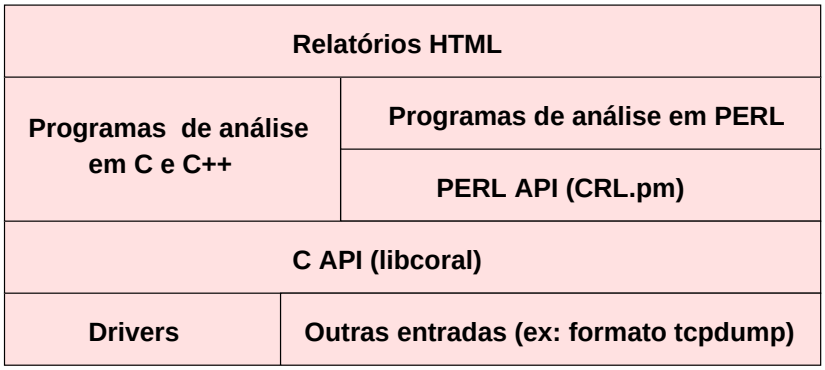


Figura 7.8: Camadas da Ferrameta CoralReef

O CoralReef é composto por um conjunto de programas de coleta, bibliotecas, APIs, programas de análise e programas para geração de gráficos e páginas HTML.

Como mostrado na figura 7.8 o CoralReef é dividido em camadas. A divisão em camadas facilita a entrada de dados provenientes de outro tipo de coletor, além de simplificar o desenvolvimento de novos aplicativos. A seguir serão descritos cada bloco mostrado na figura 7.8.

Drivers

Este bloco representa o conjunto de *drivers* que irão ser incorporados ao sistema operacional para permitir a coleta de dados de forma passiva. A instalação destes *drivers* é feita aplicando-se alguns *patches* ao sistema operacional e recompilando o *kernel* do sistema. Estes *drivers* estão disponíveis apenas para o FreeBSD UNIX e para as placas FORE, POINT e DAG.

Input Traces

É possível a utilização de dados provenientes do resultado da coleta feita em outros tipos de coletores. Por exemplo, os dados coletados com o `tcpdump` podem ser utilizados da mesma forma pelas aplicações do CoralReef de mais alto nível.

A biblioteca `libcoral`

Esta camada intermediária é composta de um conjunto de bibliotecas escritas em linguagem C. Este conjunto de bibliotecas pode ser utilizado

por aplicativos de mais alto nível para a inicialização, leitura e escrita de diversas fontes. Estas fontes são basicamente de 4 tipos:

- Placas FORE: Apenas inicialização e leitura de dados;
- Placas POINT: Inicialização e Leitura de dados;
- Arquivos de dados não compactados: Leitura e escrita;
- Arquivos de dados compactados: Leitura e escrita (utilizando a biblioteca `libz`);
- Arquivos de dados no formato "`libcap`": leitura apenas.

As fontes descritas acima como arquivos de dados também são conhecidas como *file traces*.

O módulo CRL.pm

Com o módulo CRL.pm é possível acessar as funcionalidades da biblioteca `libcoral` atráves de programas escritos em PERL.

Programas de análise em C/C++ e em PERL

Existe um conjunto de programas escritos em C/C++ e em PERL [89] que realizam algumas análises nos dados coletados. A diferença entre as implementações em C e em PERL está na velocidade das respostas. O PERL algumas vezes não é tão eficiente para apresentar respostas no tempo necessário da análise requerida. Os programas distribuidos em conjunto com o CoralReef estão em constante mudança e sua principal falha é não apresentar uma boa documentação. Muitos dos programas só podem ser utilizados como exemplos. Exemplos de programas:

- `crl_ipmatrix`: Programa escrito em C que cria uma matriz de tráfego entre IP origem e destino, mostrando o número de pacotes e a quantidade de bytes.
- `crl_tos.pl`: Programa escrito em PERL que mostra os tipos de serviços encontrados nos pacotes IP e a sua porcentagem em relação ao total.

Ferramentas para geração de páginas HTML

Na última camada estão os programas capazes de gerar relatórios gráficos e páginas HTML. Muitos dos programas nesta camada não estão disponíveis para o público "comum" e para obtê-los é necessário ser menbro do CAIDA.

Capítulo 8

Aplicações de Coleta e Análise

Este capítulo tem o objetivo de aplicar a coleta e análise de dados utilizando para tal alguns exemplos. Ao final de cada exemplo serão apresentados pequenos cometários e conclusões dos resultados.

Os quatro exemplos apresentados neste capítulo foram realizados em ambiente de produção nos Backbones IP da RedeRio [82] e da Embratel [85].

8.1 Exemplo 1

8.1.1 Metodologia

Objetivos

Deseja-se obter um relatório que avalie o comportamento da tabela BGP e as conexões BGP dos vizinhos ao Backbone da Embratel (Outros ASs conectados diretamente na Embratel).

Objetos existentes

Os obtejos do Backbone envolvidos nesta implementação são basicamente os Roteadores de *Peering* (PR figura 2.1) e as conexões BGP destes para os outros ASs.

Variáveis de medida necessárias

Número de prefixos, a profundidade da tabela, o tamanho de cada prefixo, o espaço de endereçamento e o número de ASs.

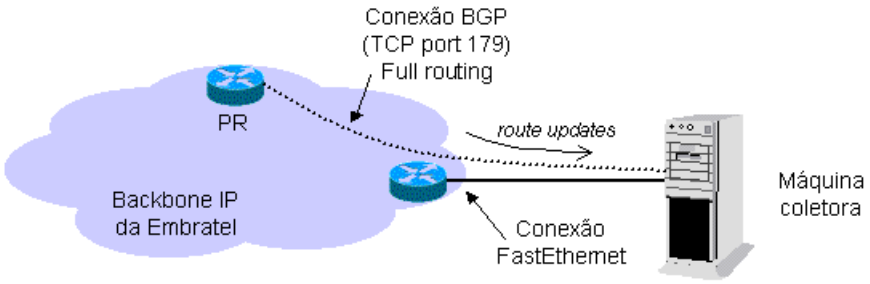


Figura 8.1: Topologia de coleta de rotas BGP na Embratel

Método e recursos para coleta

Os recursos para a coleta dos dados são:

- Um Computador PC com 1 processador PentiumIII de 450MHz;
- 256Mbytes de memória RAM;
- Uma placa de rede 10/100Tx Intel EEPro;
- Um disco de 9GBytes Ultra-SCSI;
- Sistema operacional FreeBSD 4.0 [88];
- Pacote GNU Zebra [59] (`bgpd`), ou MRT [61] da Merit;
- Porta Ethernet em algum roteador do Backbone;
- Conexão IBGP entre o roteador PR e o coletor;

O coletor deve ter uma conexão BGP permanentemente ativa com o roteador. A conexão permanente é feita para que após o recebimento de toda a tabela de roteamento, apenas os *updates* de rede sejam trocados, evitando assim um tráfego desnecessário na rede. O funcionamento é feito da seguinte maneira: o processo `bgpd` estabelece uma conexão BGP (TCP port=179) com o roteador, recebe a inserção de todas as redes e depois permanece recebendo *updates* de inserção ou remoção de redes. A figura 8.1 ilustra a topologia da coleta.

O método de coleta é feito através do *dump* da tabela de roteamento localizada dentro do coletor (processo `bgpd`), que é realizado a cada 2 horas. Os arquivos *dump* são gravados no formato MRT binário, descrito no apêndice .

O espaco necessário para o armazenamento é de aproximadamente 60 bytes por linha da tabela de rotemanto. Uma tabela de roteamento completa

(*full routing*) tem aproximadamente 97000 linhas, o que dá um espaço em disco 5,5Mbytes por *dump* realizado. O espaço consumido em disco é de aproximadamente 2Gbytes por mês.

A coleta foi realizada dentro do Backbone da Embratel, no período de 15/01/2001 até 14/02/2001.

Método e recursos para consolidação e análise

Os recursos para análise são os mesmos utilizados na coleta, porém adicionando os seguintes ítens:

- Pacote PERL 5.005 [89] (desenvolvimento dos consolidadores);
- Pacote libpng versão 1.0.9 (para criação de imagens PNG);
- Pacote GD versão 1.8.4 (auxiliar na criação de gráficos);
- Pacote RRDTool versão 1.0.28 (para armazenamento e criação de gráficos temporais);
- Programa desenvolvido para criação de gráficos 2D e 3D;

Para a análise final é necessário retirar os ASs repetidos, eles são utilizados na técnica *as-prepend* dentro de um *as-path*. A retirada de ASs repetidos é fundamental para se ter a profundidade correta das redes na tabela de roteamento.

Apresentação dos resultados

A apresentação dos dados será feita apartir de tabelas em texto, histogramas de distribuição de frequência e gráficos no tempo.

8.1.2 Análise e apresentação dos resultados

Número de vizinhos BGP da tabela

A avaliação do número de vizinhos ao longo do tempo, pode ser muito útil para avaliar indiretamente o comportamento das conexões BGP aos outros ASs ligados ao Backbone. A instabilidade do número de vizinhos é prejudicial a rede, uma queda de um vizinho causa aumento na CPU do roteador, que precisa enviar/receber uma nova tabela, além de novamente calcular os "melhores caminhos" da tabela de roteamento. O desempenho de tecnologias como MPLS [13] pode também ser prejudicado com tais "variações".

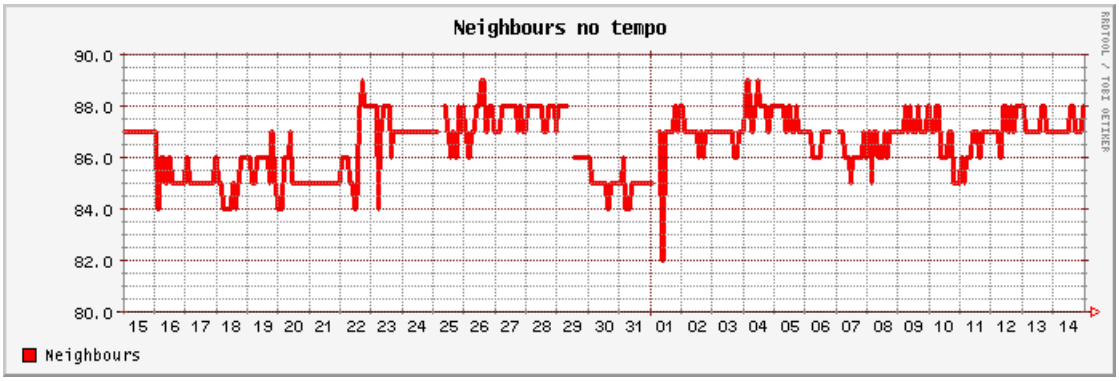


Figura 8.2: Número de *neighbours* da tabela

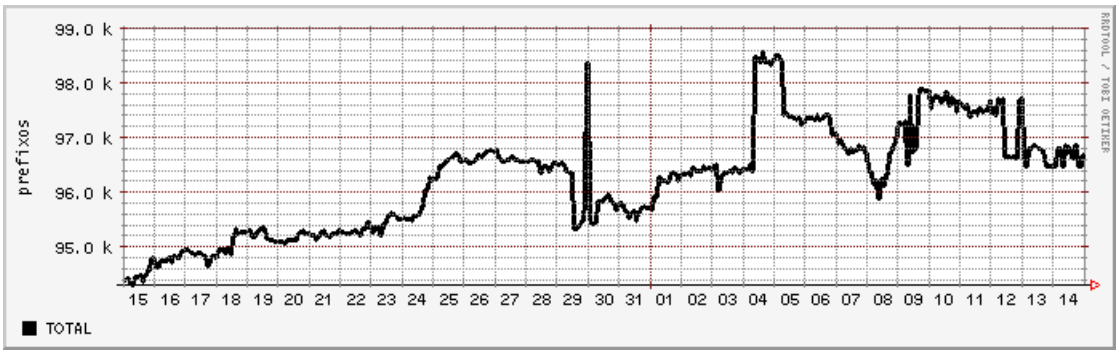


Figura 8.3: Número total de prefixos da tabela

A figura 8.2 mostra o número de vizinhos BGP (*neighbours*) ao longo do período da coleta. Na tabela 8.1 estão as propriedades dos dados coletados durante o período.

Tabela 8.1: Valores estatíticos do número de *neighbours*

Amostras = 368
Valor máximo = 89 *neighbours*
Valor mínimo = 82 *neighbours*
Media = 86.535 *neighbours*
Variância = 1.180 *neighbours* (1.364% da média)
Mediana = 87 *neighbours*
Simetria= -0.470
Peakness= 2.693

Número de prefixos na tabela

O tamanho da tabela é calculado contando o número total de prefixos existentes na tabela de roteamento. Sua análise é util na avaliação da instabilidades de anúncios, assim como a sua agregação.

Na figura 8.3 é possível visualizar o número de prefixos da tabela de roteamento durante o período de coleta. Na tabela 8.2 estão as propriedades

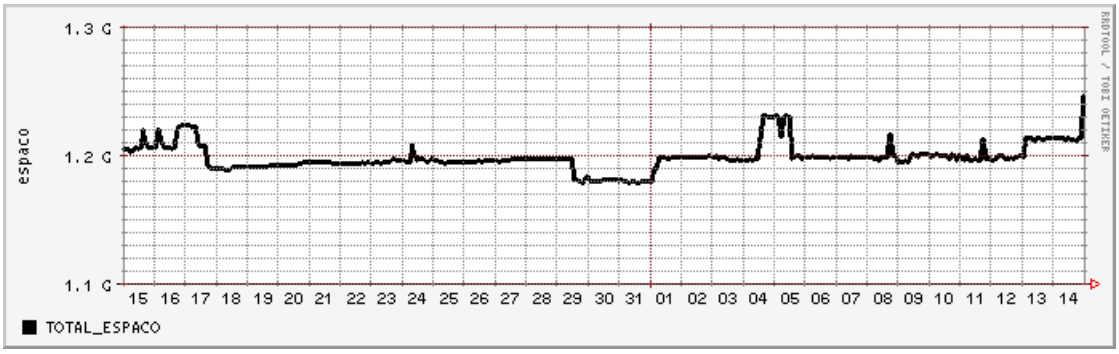


Figura 8.4: Espaço de endereçamento da tabela

dos dados coletados durante o período.

Tabela 8.2: Valores estatíticos do número de prefixos

Amostras = 368
Valor máximo = 98576 prefixos
Valor mínimo = 94297 prefixos
Média = 96223.076 prefixos
Variância = 987.914 prefixos (1.027% da média)
Mediana = 96372 prefixos
Simetria = 0.164
Peakness = 2.292
Taxa de crescimento no mes + ou - 2000 prefixos/mes

Espaço de endereçamento da tabela

O espaço de endereçamento total da tabela é calculado através do somatório do espaço de endereçamento individual de cada prefixo.

O espaço de endereçamento de um prefixo é calculado a partir de sua máscara. Exemplo: um prefixo com máscara 255.255.0.0 (/16) tem o seu espaço igual a $2^{16} = 65536$, um prefixo com máscara 255.255.248.0 (/21) tem o seu espaço igual a $2^{11} = 2048$. O espaço total de endereçamento destes dois prefixos é igual a $65536 + 2048 = 67584$

Na figura 8.4 é possível visualizar o espaço de endereçamento da tabela de roteamento durante o período de coleta. Na tabela estão as propriedades dos dados coletados durante o período.

Tabela 8.3: Valores estatíticos do espaço de endereçamento

Amostras = 368
Valor mínimo = 1175132159
Valor máximo = 1230801270
Média = 1198511854.2527
Variância = 10114734.1158 (0.844% da média)
Mediana = 1197306818.5
Simetria= 0.9412
Peakness= 5.0462

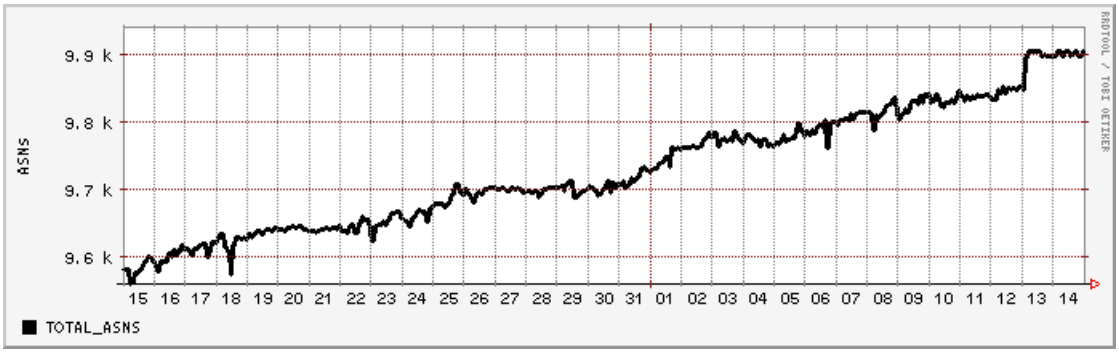


Figura 8.5: Número de ASNs durante o mês

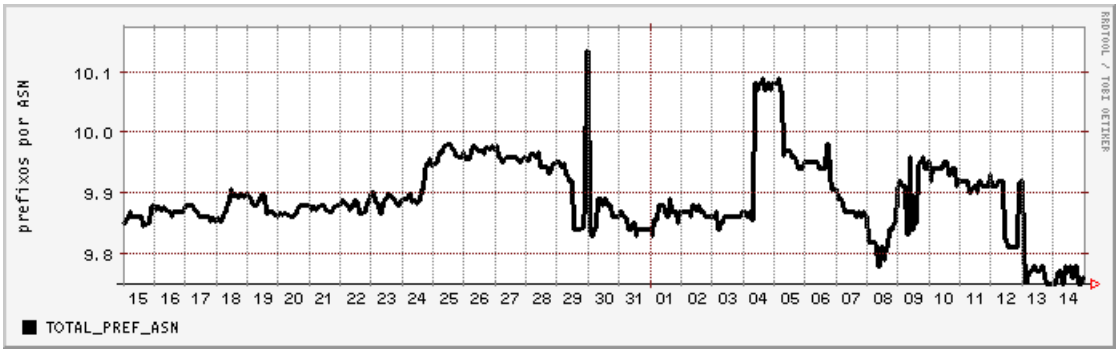


Figura 8.6: Número de prefixos por ASN durante o mês

Número de ASNs

A figura 8.5 ilustra o total de ASNs encontrados na tabela de roteamento durante o período de coleta.

Número de prefixos por ASN

A figura 8.6 ilustra a média de prefixos por ASN durante o período de coleta.

A tabela 8.1.3 estão as propriedades dos dados coletados durante o período.

Tabela 8.4: Estatística da quantidade de prefixos por ASN

Amostras = 368
Valor mínimo = 9.75
Valor máximo = 10.15
Média = 9.892
Variância = 0.063 (0.637% da média)
Mediana = 9.88
Simetria= 0.583
Peakness= 4.640

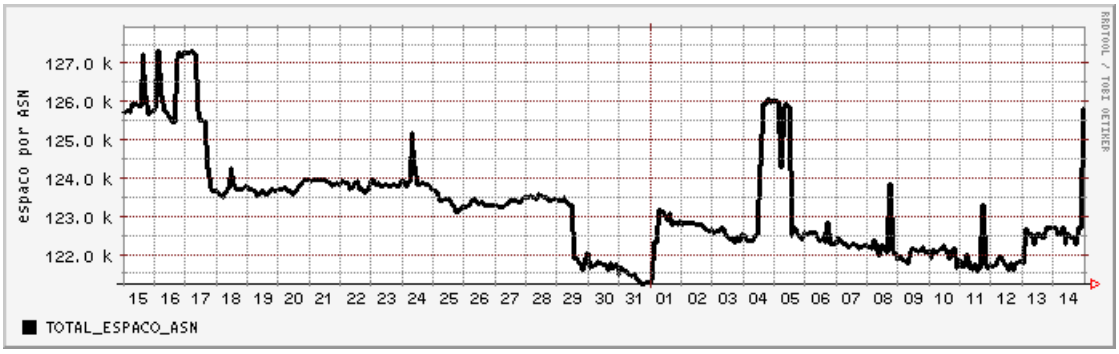


Figura 8.7: Espaço de endereçamento por ASN durante o mês

Espaço de endereçamento por ASN

A figura 8.7 ilustra a média de espaço de endereçamento por ASN durante o período de coleta.

Na tabela 8.5 estão as propriedades dos dados coletados durante o período.

Tabela 8.5: Estatística do espaço de endereçamento por ASN

Amostras = 368
Valor mínimo = 121224.98
Valor máximo = 127692.67
Média = 123211.121
Variância = 1342.826 (1.09% da média)
Mediana = 123176.505
Simetria= 1.109
Peakness= 4.116

Distribuição do tamanho dos prefixos na tabela

A figura 8.8 ilustra o histograma das médias do número de prefixos de tamanho (/x) econtrados na tabela de roteamento durante o período de coleta.

A distribuição relativa à média do total de prefixos está na figura 8.9. Na tabela 8.6 estão listados os valores obtidos.

A figura 8.10 ilustra o número de prefixos no tempo para os tamanhos /24 e o somatório do restante.

Profundidade dos prefixos na tabela

A figura 8.11 ilustra a porcentagem média de prefixos por profundidade no período de coleta entre 15/01/2001 e 14/02/2001. Na tabela 8.7 estão listados os detalhes dos valores obtidos.

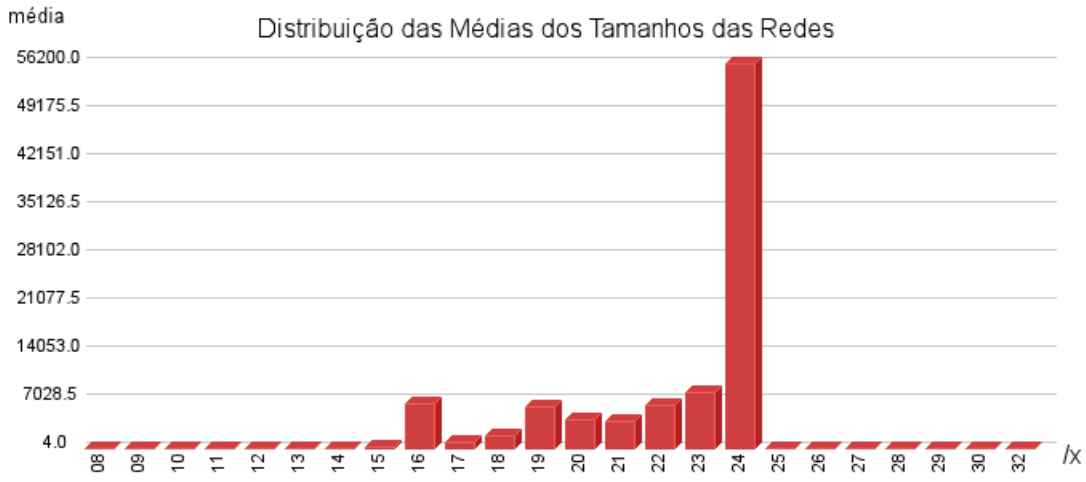


Figura 8.8: Quantidade de prefixos por tamanho no mês

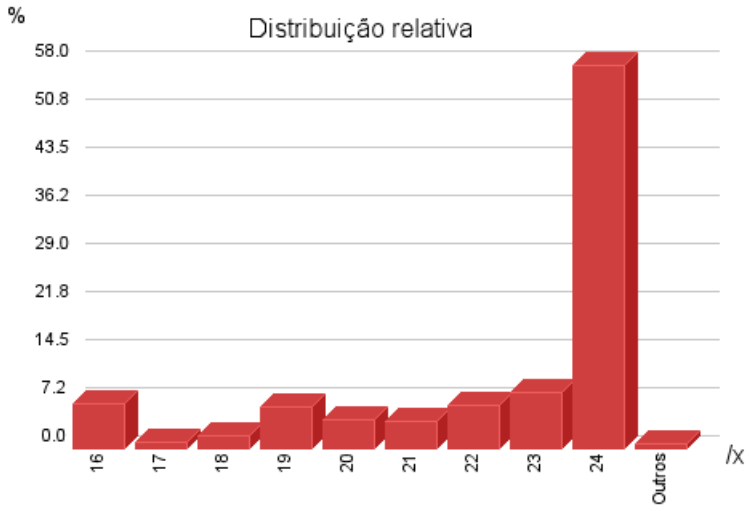


Figura 8.9: Porcentagem de prefixos por tamanho no mês

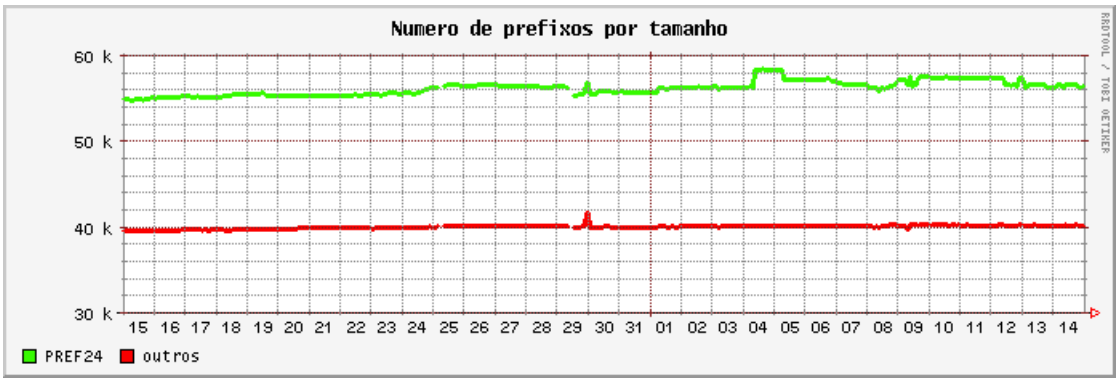


Figura 8.10: Número de prefixos por tamanho de 15/01/2001 até 14/02/2001

Tabela 8.6: Distribuição do tamanho dos prefixos

Tamanho	prefixos	% do total
08	21.158	0.022
09	4.000	0.004
10	4.962	0.005
11	10.929	0.011
12	31.842	0.033
13	60.682	0.063
14	193.712	0.200
15	303.639	0.313
16	6745.133	6.959
17	1006.266	1.038
18	1979.764	2.043
19	6239.636	6.437
20	4440.927	4.582
21	4203.054	4.336
22	6334.587	6.535
23	8249.280	8.511
24	56214.348	57.996
25	3.429	0.004
26	13.446	0.014
27	26.796	0.028
28	44.340	0.046
29	27.861	0.029
30	38.185	0.039
32	25.169	0.026

Tabela 8.7: Distribuição da profundidade dos prefixos

Profundidade	prefixos	% do total
01	31.000	0.032
02	3425.921	3.535
03	29156.861	30.081
04	42238.133	43.577
05	16796.201	17.329
06	3894.389	4.018
07	558.065	0.576
08	98.682	0.102
09	22.283	0.023
10	1.516	0.002
11	1.000	0.001
12	1.000	0.001

Profundidade por espaço de endereçamento

A figura 8.12 ilustra um histograma do espaço médio total de endereçamento por profundidade.

Número de prefixos por vizinho

A tabela 8.8 estão listados a média de prefixos na tabela de roteamento durante o período de coleta dos vizinhos: 1251 (Rede Ansp), 1916 (RNP), 2715 (RedeRio), 7303 (Telecom Argentina), 8657 (Marconi Portugal) e 5511 (France Telecom):

A figura 8.13 ilustra os prefixos dos vizinhos AS 1916 (RNP), AS 2715 (Rede Rio) e AS 1251 (ANSP) no tempo.

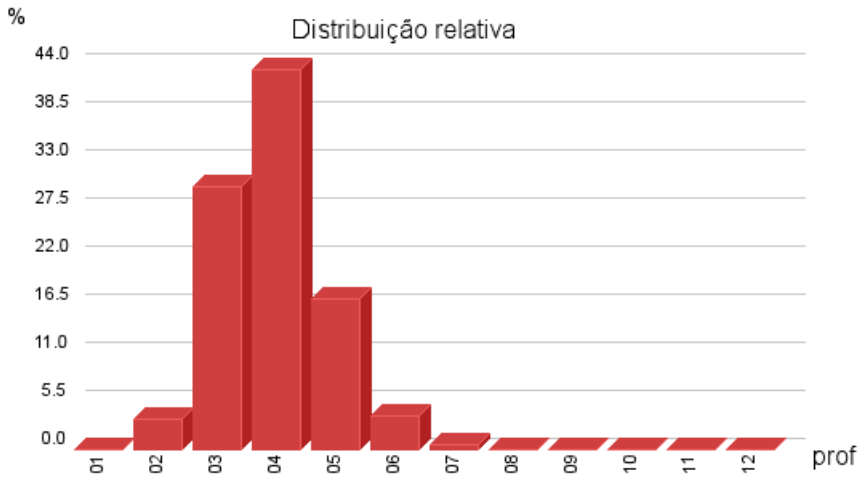


Figura 8.11: Porcentagem do total de prefixos por profundidade no mês

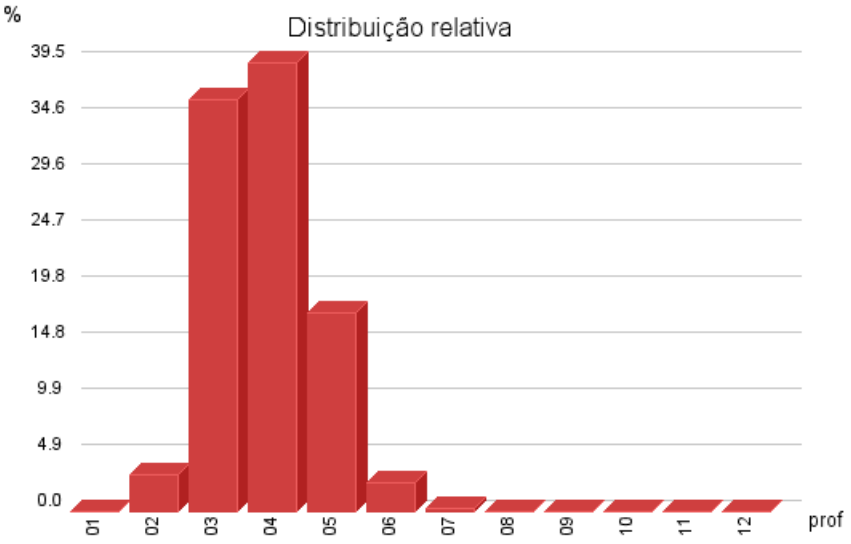


Figura 8.12: Porcentagem do espaço total de endereçamento por profundidade no mês

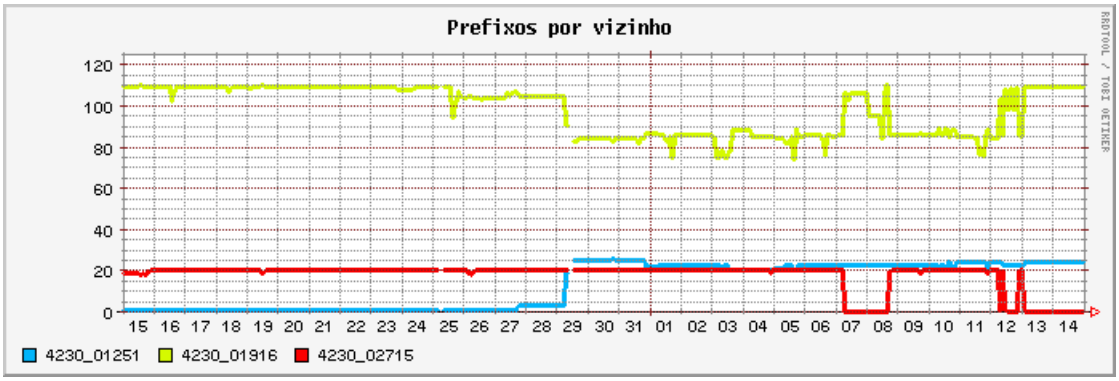


Figura 8.13: Número de prefixos para 1916, 2715 e 1251 de 15/01/2001 até 14/02/2001

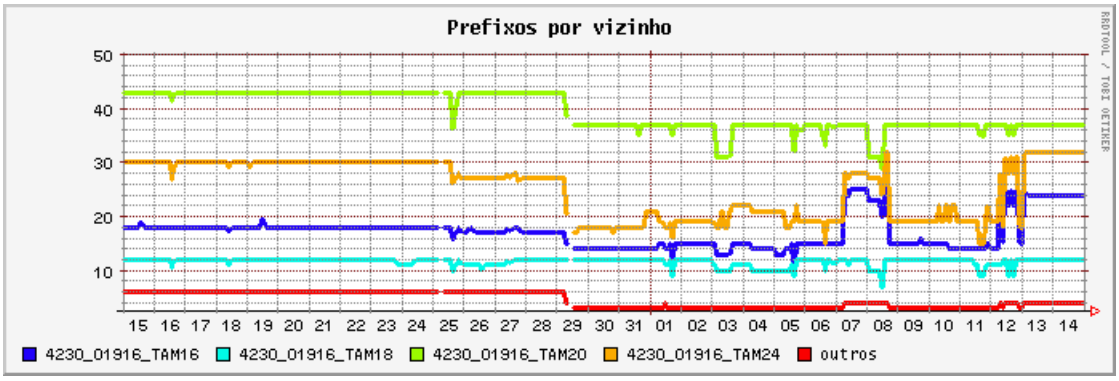


Figura 8.14: Número de prefixos de tamanho x para 1916 de 15/01/2001 até 14/02/2001

Tabela 8.8: Prefixos por vizinho

Route Path	prefixos	% do total
4230 1251	12.910	0.013
4230 1916	98.008	0.101
4230 2715	19.935	0.021
4230 5511	114.008	0.118
4230 7303	181.344	0.187
4230 8657	82.353	0.085

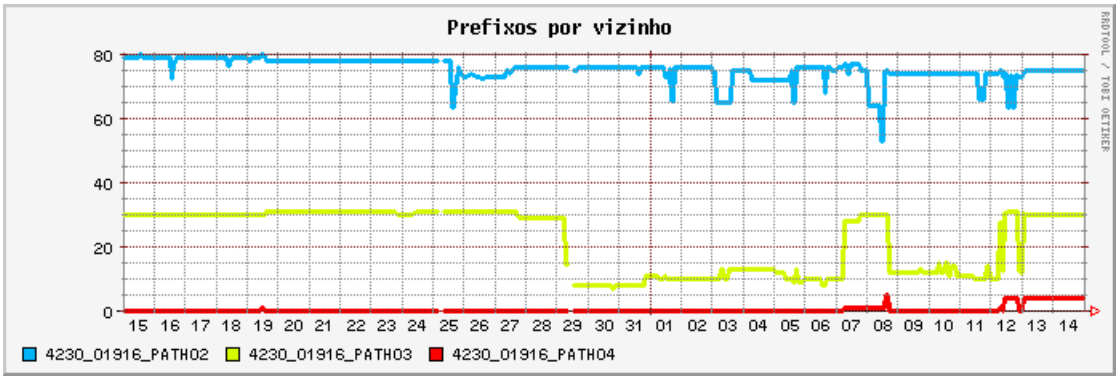


Figura 8.15: Número de prefixos de profundidade x para 1916 de 15/01/2001 até 14/02/2001

Tamanho dos prefixos por vizinho

A figura 8.14 mostra a evolução da quantidade de prefixos dos tamanhos de rede para o vizinho 1916 durante o período de coleta.

Profundidade dos prefixos por vizinho

A figura 8.15 mostra a evolução da quantidade de prefixos por profundidade para o vizinho 1916 durante o período de coleta.

8.1.3 Comentários e conclusões

Os dados coletados neste exemplo foram referentes a apenas 1 mês. Para avaliação do crescimento das redes o ideal é que o período de coleta seja de pelo menos 1 ano. Porém a coleta de 1 mês foi suficiente para esclarecer as necessidades de se utilizar este tipo de coleta e análise.

Número de vizinhos BGP da tabela

A avaliação do número de vizinhos (*neighbours*) ao longo do tempo, pode ser muito útil para avaliar indiretamente o comportamento das conexões BGP aos outros ASs ligados ao Backbone. A instabilidade do número de vizinhos

é prejudicial a rede, uma queda de um vizinho causa aumento na CPU do roteador, que precisa enviar/receber uma nova tabela, além de novamente calcular os "melhores caminhos" da tabela de roteamento. O desempenho de tecnologias como MPLS [13] podem também serem prejudicadas com tais "variações".

A figura 8.2 mostra um gráfico da evolução do número de vizinhos ao longo do mês. O ideal é que este gráfico fosse praticamente reto sem grandes variações de subida e descida. Para um Backbone em crescimento este número tende a aumentar, e muitas vezes existem alguns que são instáveis por motivos de inexperiência. A monitoração individual pode ser utilizada como parâmetro de qualidade.

Número de prefixos na tabela

O tamanho da tabela é calculado contando o número total de prefixos existentes. O gráfico da figura 8.3 ilustra a quantidade de prefixos ao longo do mês, o ideal é que este gráfico fosse na horizontal e sem variações. No gráfico é possível verificar o surgimento de dois indesejáveis aumentos do número de prefixos, no final do dia 29 e no dia 04.

O aumento do número de prefixos não é muito bom para rede, principalmente quando for causada por uma falha em sumarizações de endereços. O aumento repentino do número de prefixos causa instabilidade carga nos roteadores, que precisam recalcular muitos dos caminhos, e algumas vezes precisam limpar os *caches* de roteamento.

Muitos dos prefixos que estão na tabela de roteamento podem ser sumarizados, a criação de um procedimento que faça esta sumarização na tabela de roteamento pode ser útil.

Espaço de endereçamento da tabela

O espaço de endereçamento de um prefixo é calculado a partir de sua máscara. Exemplo: um prefixo com máscara 255.255.0.0 (/16) tem o seu espaço igual a $2^{16} = 65536$, um prefixo com máscara 255.255.248.0 (/21) tem o seu espaço igual a $2^{11} = 2048$. O espaço total de endereçamento destes dois prefixos é igual a $65536 + 2048 = 67584$

O espaço de endereçamento total da tabela é calculado através do somatório do espaço de endereçamento individual de cada prefixo.

O cálculo deste espaço de endereçamento pode conter erros quando existem anúncios de pedaços de uma rede mais específicos por outro

caminho. Para resolver tais discrepâncias o processo de cálculo do espaço de endereçamento deve evitar sobreposições de redes, o que não foi feito nos calculos deste exemplo. Estas discrepâncias podem causar o aumento do espaço de endereçamento, mas em termos comparativos é um aumento muito pequeno, uma vez que o anuncio mais específico por outro caminho não é muito encontrado na tabela de roteamento. A criação de um procedimento que calcule a quantidade de prefixos com sobreposição de espaço de endereçamento pode ser interessante.

A figura 8.4 mostra a evolução do espaço de endereçamento ao longo do mês. A média do espaço de endereçamento é de aproximadamente 1.2G endereços. O espaço de endereçamento disponível no endereçamento IP versão 4 (retirando as redes reservadas, multicast e *loopback*) é de em torno de 3.74G endereços. O que mostra que estamos dentro de 1/3 de toda a alocação de IPs. Este é um dado importante para verificar a real necessidade de uma implementação do protocolo IP versão 6 apenas para resolver questões de escassez de endereços IP.

Sabendo que o valor calculado de espaço de endereçamento foi calculado com uma margem de erro somada, com a utilização cada vez maior de NATs (*Network Address Translator*), com uma campanha de reorganização de IPs na Internet, com a utilização racional de endereços IPs e outros fatores, podemos dizer com certeza que não ocorrerá escassez de endereços para os serviços hoje utilizados pelo menos durante alguns anos. Talvez uma previsão teórica possa ser conseguida dos orgãos de alocação IP como o ARIN [62], APNIC [63] e o RIPE [77].

Número de ASNs

O gráfico da figura 8.5 mostra a quantidade de ASNs encontrados na tabela de roteamento ao longo do mês. Esta tabela mostra um crescimento de em torno de 300 ASNs por mês. Se este crescimento for linear podemos prever que os 16bits designados para um ASN no protocolo BGP seriam suficientes pelo menos até os próximos 15 anos. Podemos associar de modo aproximado o aumento de cada ASN como sendo o surgimento de um novo Backbone IP.

Número de prefixos por ASN

O gráfico da figura 8.6 mostra a quantidade média de prefixos por ASN ao longo do mês. Podemos ver que durante a coleta dos dados existiam em torno de 10 prefixos por ASN, e que ao longo do mês este número de

prefixos reduziu. Esta redução pode ter sido causada por alguns fatores como: aumento de Backbones IP com um menor número de redes IP válida (utilizando endereços reservados) ou o crescimento do número de Backbones que passaram a utilizar um ASN.

A tabela mostra as estatísticas do número de prefixos por ASN em cada amostra. A variância é pequena porque cada amostra utilizada neste cálculo é a média de prefixos por ASN do instante da amostra.

Espaço de endereçamento por ASN

Melhor que a avaliação da quantidade de prefixos por ASN é o espaço médio de endereçamento IP por ASN, pois a quantidade de prefixos contém pouca informação em relação ao tamanho das redes.

Pela tabela 8.5 podemos ver que em média existe 123K endereços por ASN, o que é aproximadamente em torno de dois prefixos de tamanho /16.

O gráfico da figura 8.7 mostra o espaço de endereçamento médio por ASN ao longo do mês. Podemos ver que no gráfico houve uma redução de aproximadamente 4K endereços ao longo do mês.

Com o gráfico do número de ASN (figura 8.5) e com a redução do espaço de endereçamento por ASN é possível verificar que cada vez mais os Backbones estão se ligando a Internet e cada vez mais eles tem menos endereços IP, mostrando uma tendência do aumento da capilaridade.

Distribuição do tamanho dos prefixos na tabela

O histograma da figura 8.8 mostra a quantidade de prefixos pelo seu tamanho, e o histograma da figura 8.9 mostra a mesma distribuição de forma percentual. Pela tabela 8.6 podemos ver que o tamanho /24 é responsável por aproximadamente 58% de todos os prefixos encontrados na tabela. A média do tamanho dos prefixos calculado pela tabela 8.6 é de /22.32, ou seja, na média um prefixo tem o tamanho de /22 na tabela de roteamento.

Isso mostra que na maioria os anúncios nas tabelas de roteamento são feitos com pequenas, utilizando pouca agregação, uma vez que no exemplo anterior o espaço de endereçamento médio por ASN é de em torno um prefixo /15.

Profundidade da tabela

A profundidade da tabela é um parâmetro importante para verificação das políticas de roteamento. Com uma monitoração mensal é possível acompanhar o crescimento dos vizinhos e suas redes interconectadas, verificando o quanto os vizinhos estão se afastando do núcleo da Internet ou se aproximando.

Um prefixo (ou rede) com profundidade 02 significa que é um prefixo anunciado diretamente pelo vizinho. Uma profundidade de 01 significa que os prefixos são originados internamente ao Backbone. Uma profundidade de 03 indica que o prefixo é vizinho de um vizinho do Backbone, ou seja, está a dois *route-paths*.

Foram calculados duas profundidades da tabela de roteamento, a profundidade por prefixos e a profundidade por espaço de endereçamento.

A profundidade por prefixos da tabela de roteamento mostra o quanto os prefixos estão espalhados na tabela de roteamento em relação a profundidade. No histograma da figura 8.11 verifica-se que a maioria nos prefixos (73%) estão a 3 ou a 4 de profundidade.

O histograma da figura 8.12 mostra a distribuição relativa do espaço de endereçamento por cada profundidade. Este cálculo é feito fazendo o somatório dos espaços de endereçamento de cada prefixo em cada profundidade. Este histograma mostra o Backbone está perto do núcleo da Internet, quanto maior for as barras para a esquerda melhor será esta proximidade.

Comparando os histogramas das figuras 8.12 e 8.11 verifica-se que a porcentagem de prefixos de profundidade 3 é de 30%, mas estes prefixos são responsáveis por 35% do espaço de endereçamento. O que mostra que a distribuição dos tamanhos dos prefixos nas profundidades não acompanham a distribuição dos espaços de endereçamento. Uma futura análise no tempo relacionando a profundidade, a quantidade de prefixos e o espaço de endereçamento pode ser importante.

Análise de prefixos por vizinho

A avaliação do número de prefixos anunciados por cada vizinho tem muitas aplicações como: verificar as políticas de roteamento, avaliar o vizinho que "injeta" muitos prefixos na tabela e verificar agregação de rotas.

No gráfico da figura 8.13 é possível ver o relacionamento de anúncios

entre a RNP (1916), a RedeRio (2715) e a ANSP (1251). As políticas de roteamento podem ser verificadas acompanhando os anúncios BGP entre estes vizinhos. No dia 29 do gráfico os anúncios de alguns prefixos que estavam sendo anunciados pela RNP, começaram a ser anunciados pela ANSP. O mesmo comportamento de trânsito pode ser visto entre a RedeRio e a RNP no período dos dias entre 07 e 14.

Outras análise podem ser feitas apenas para os prefixos anunciados por um vizinho, no gráfico da figura 8.14 mostra o impacto no tamanho dos prefixos ao longo mês, o anúncio das redes que passou a ser feito pela ANS pode ser visto no dia 29.

O gráfico da figura 8.15 a quantidade de prefixos de profundidade 2, 3 e 4 ao longo do mês. Os prefixos do Backbone da RNP devem aparecer apenas na linha azul, os prefixos verificados na linha verde são todos os prefixos anunciados pelos vizinhos da RNP, como a ANSP e a RedeRio. A troca de anúncios mostrada na figura 8.13 entre a RNP e a ANSP do dia 29 pode ser confirmada aqui, e também as trocas de rotas feitas pela RedeRio e a RNP nos dias 07, 08, 12 e 13.

8.2 Exemplo 2

8.2.1 Metodologia

Objetivos

Deseja-se obter um relatório que avalie o comportamento dos enlaces de clientes (AL figura 2.1).

Objetos existentes

Os objetos do Backbone envolvidos nesta implementação são basicamente os roteadores de acesso , os enlaces às redes dos clientes e os roteadores dos clientes.

Variáveis de medida necessárias

Latência, *jitter*, descarte de pacotes na fila de saída e vazão de entrada e saída.

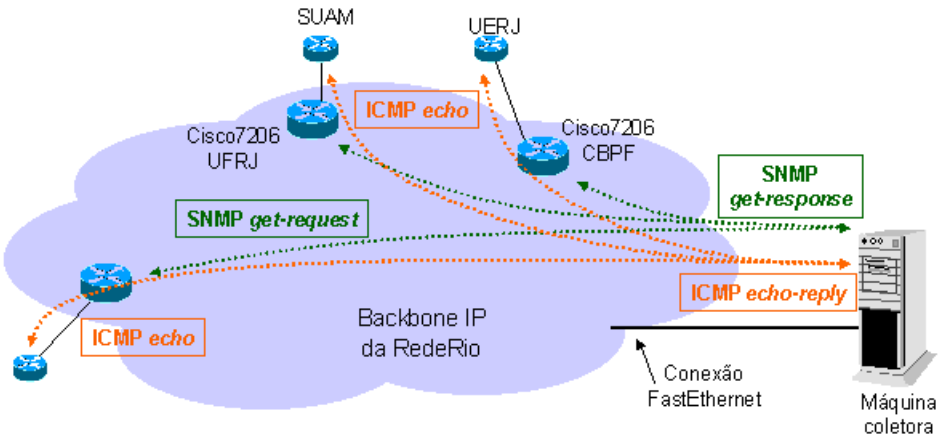


Figura 8.16: Topologia de coleta via SNMP/ICMP na RedeRio

Método e recursos para coleta

Os recursos para a coleta dos dados são:

- Um Computador PC com 1 processador PentiumII de 300MHz;
- 256Mbytes de memória RAM;
- Uma placa de rede 10/100Tx Intel Pro;
- Uma placa de rede 10BASET NE2000 Compatível;
- Um disco IDE de 4GBytes;
- Um disco IDE de 20GBytes;
- Sistema operacional FreeBSD 4.2 [88];
- Pacote PERL 5.005 [89];
- Pacote PERL Net::SNMP;
- Programa `fping`;
- Porta Ethernet em algum roteador do Backbone;

A coleta é iniciada através do `cron`, a cada minuto é enviado uma *query* SNMP e um pacote ICMP para cada enlace monitorado. Como ilustrado na figura 8.16.

A *query* SNMP é feita no roteador de acesso do Backbone (AR figura 2.1), através da diretiva `get request`. Os objetos monitorados estão da tabela 8.9.

Tabela 8.9: Objetos monitorados via SNMP

ifSpeed = "1.3.6.1.2.1.2.2.1.5"
ifAdminStatus = "1.3.6.1.2.1.2.2.1.7"
ifOperStatus = "1.3.6.1.2.1.2.2.1.8"
ifInOctets = "1.3.6.1.2.1.2.2.1.10"
ifInUcastPkts = "1.3.6.1.2.1.2.2.1.11"
ifOutOctets = "1.3.6.1.2.1.2.2.1.16"
ifOutUcastPkts = "1.3.6.1.2.1.2.2.1.17"
ifOutDiscards = "1.3.6.1.2.1.2.2.1.19"
ipAdEntIfIndex = "1.3.6.1.2.1.4.20.1.2"
ipAdEntNetMask = "1.3.6.1.2.1.4.20.1.3"
ifAlias = "1.3.6.1.2.1.31.1.1.1.18"

A latência é medida utilizando o IP do roteador do cliente. Este IP é obtido através do objeto `ipAdEntIfIndex`. Apenas enlaces com a máscara igual a 255.255.255.252 são monitorados (obtido em `ipAdEntNetMask`). O valor da latência é obtido medindo-se o tempo decorrido apartir do envio de um pacote ICMP `echo` (para o IP do roteador do cliente) até a chegada de um pacote ICMP `echo reply`.

O espaço necessário para o armazenamento dos dados é de aproximadamente 200Kbytes por enlace por dia. O monitor armazenou dados num total de 60 enlaces de 3 roteadores da RedeRio, relacionados a seguir:

```
cisco7206-atm-puc.rederio.br (200.20.94.81)
cisco7206-atm-cbpf.rederio.br (200.20.94.41)
cisco7206-atm-ufrj.rederio.br (200.20.94.1)
```

A coleta foi realizada dentro do Backbone da RedeRio, no período de 00:00:00 11/02/2001 até 00:00:00 07/04/2001.

Método e recursos para consolidação e análise

Os recursos para análise são os mesmos utilizados na coleta, porém adicionando os seguintes ítens:

- Pacote libpng versão 1.0.9 (para criação de imagens PNG);
- Pacote GD versão 1.8.4 (auxiliar na criação de gráficos);
- Pacote RRDTool versão 1.0.28 (para armazenamento e criação de gráficos temporais);
- Programa desenvolvido para criação de gráficos;

A análise final foi feita sobre gráficos temporais no período de uma semana, iniciando no dia 15/03/2001 e terminando no dia 21/03/2001.

Apresentação dos resultados

A apresentação dos dados foi feita apartir de gráficos no tempo.

8.2.2 Análise e apresentação dos resultados

Foram escolhidos 5 enlaces a serem analisados, a saber:

- Enlace com a SUAM: Velocidade de 64Kbps, roteador da UFRJ (200.20.94.1), interface Serial1/1.
- Enlace com a Faculdade Carioca: Velocidade de 64Kbps, roteador da UFRJ (200.20.94.1), interface Serial1/4.
- Enlace com o IME: Velocidade de 2Mbps, roteador do CBPF (200.20.94.41), interface Serial3/2.
- Enlace com a LLS Consultoria: Velocidade de 64Kbps, roteador da UFRJ (200.20.94.1), interface 3/6.
- Enlace com a UERJ: Velocidade de 2Mbps, roteador do CBPF (200.20.94.41), interface Serial5/3.

Os gráficos mostrados nesta sessão são relativos aos dias 15/02/2001 até 21/02/2001.

Vazão em bits por segundo

A figuras 8.17, 8.18, 8.19, 8.20 e 8.21 ilustram a vazão em bits por segundo dos enlaces da SUAM, Faculdade Carioca, IME, LLS e UERJ respectivamente.

A cor azul representa a quantidade de bits por segundo recebidos na entrada da interface do roteador de acesso em média de 30 minutos, a cor amarela representa o valor máximo no mesmo intervalo de 30 minutos.

A cor vermelha clara representa a quantidade de bits por segundo enviados na saída da interface do roteador de acesso em média de 30 minutos, a cor vermelha escura representa o valor máximo no mesmo intervalo de 30 minutos.

Percentual de Vazão de entrada

A figuras 8.22, 8.18, 8.24, 8.20 e 8.26 ilustram a vazão em bits por segundo dos enlaces da SUAM, Faculdade Carioca, IME, LLS e UERJ respectivamente.

A cor vermelha clara representa a percentagem da quantidade de bytes na entrada da interface em relação a quantidade de bytes na saída da interface. Os gráficos foram limitados em 100%.

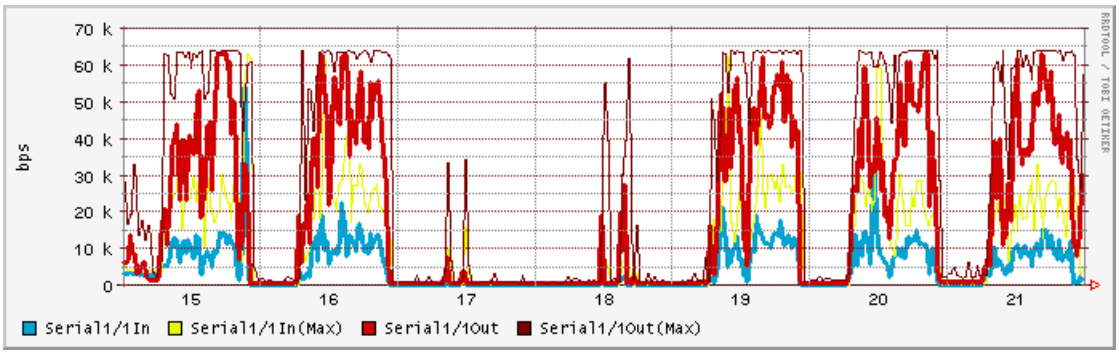


Figura 8.17: Vazão em bps no enlace com a SUAM

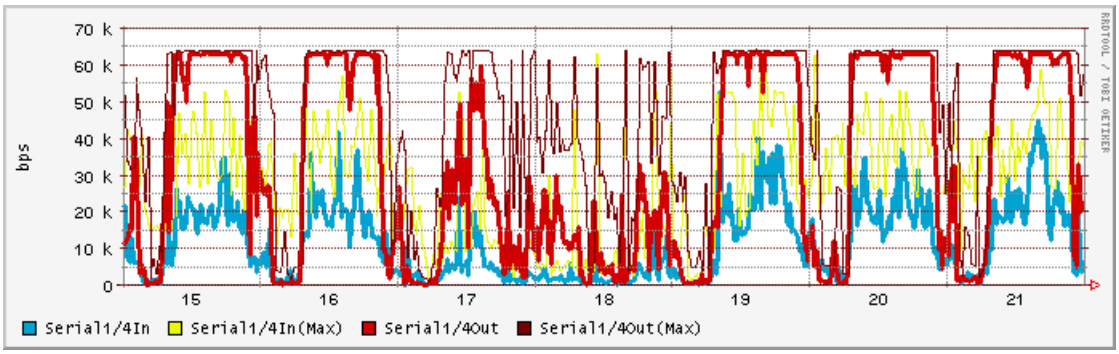


Figura 8.18: Vazão em bps no enlace com a Faculdade Carioca

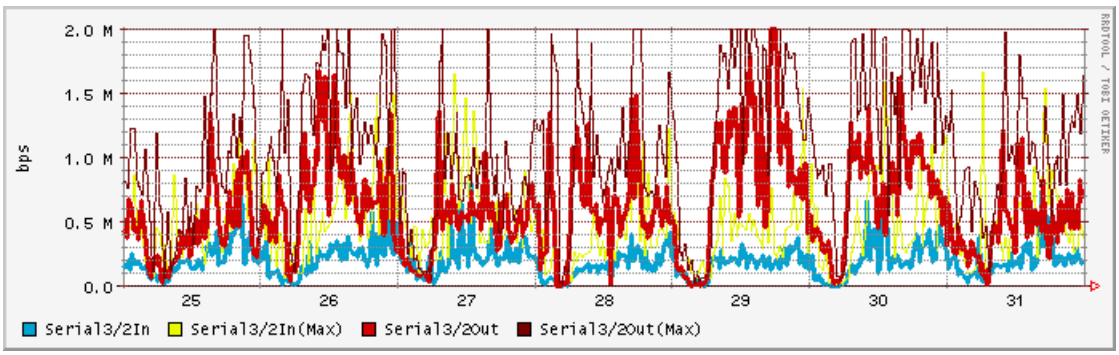


Figura 8.19: Vazão em bps no enlace com o IME

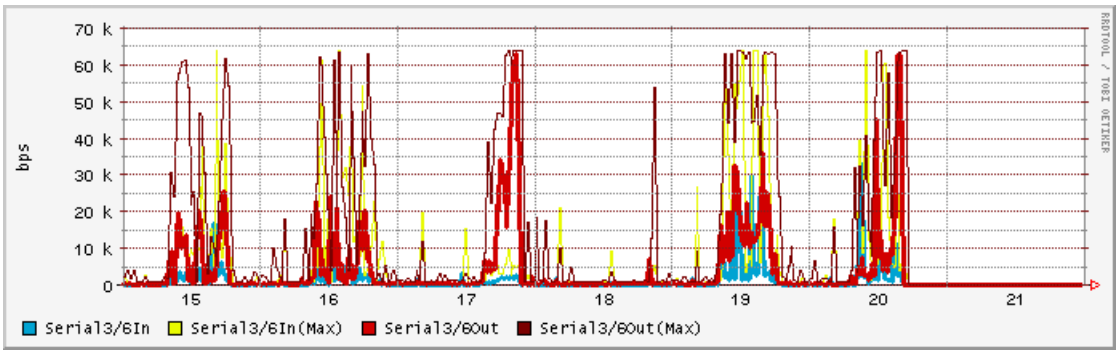


Figura 8.20: Vazão em bps no enlace com a LLS

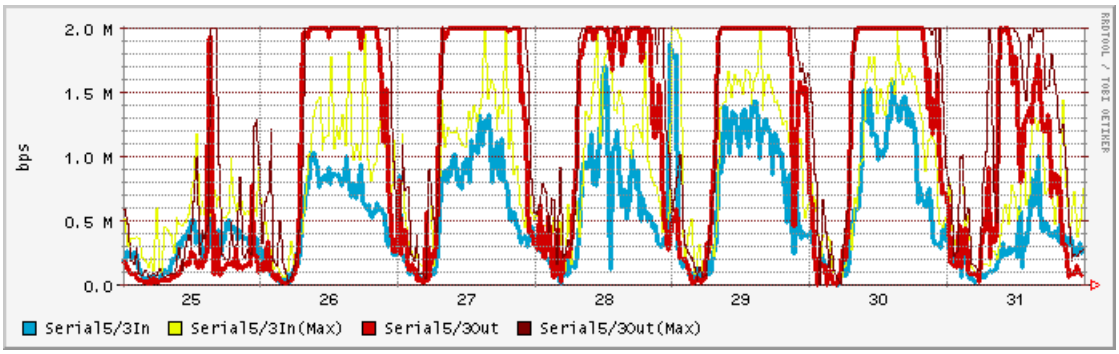


Figura 8.21: Vazão em bps no enlace com a UERJ

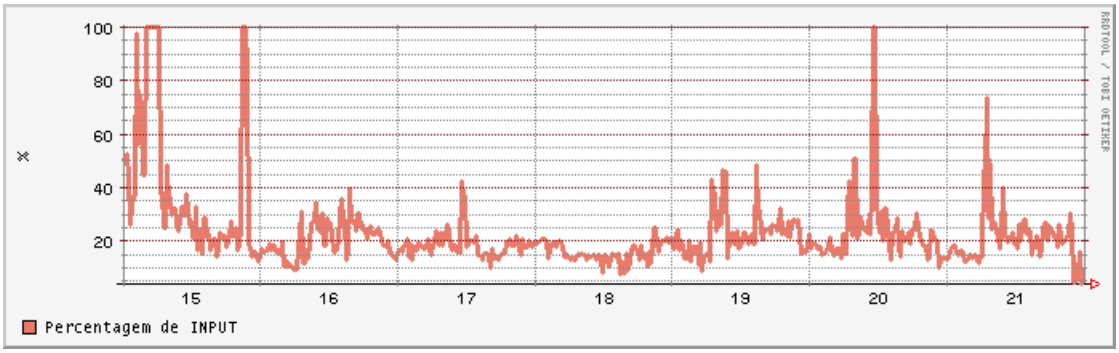


Figura 8.22: Percentual da vazão de entrada no enlace com a SUAM

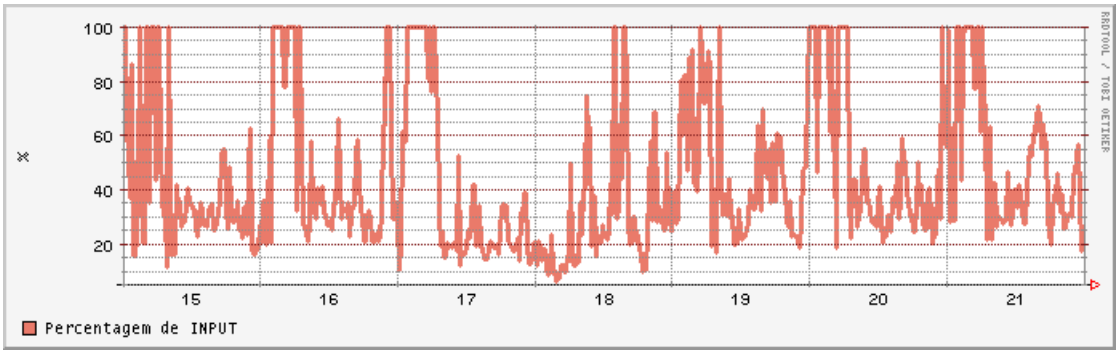


Figura 8.23: Percentual da vazão de entrada no enlace com a Faculdade Carioca

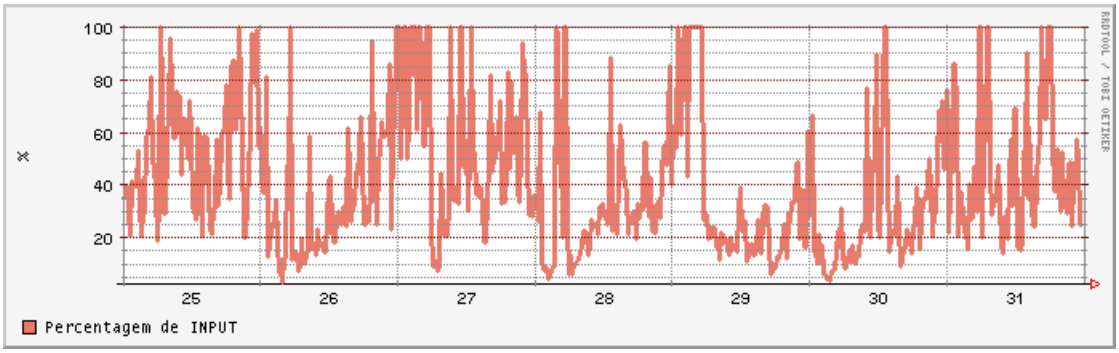


Figura 8.24: Percentual da vazão de entrada no enlace com o IME

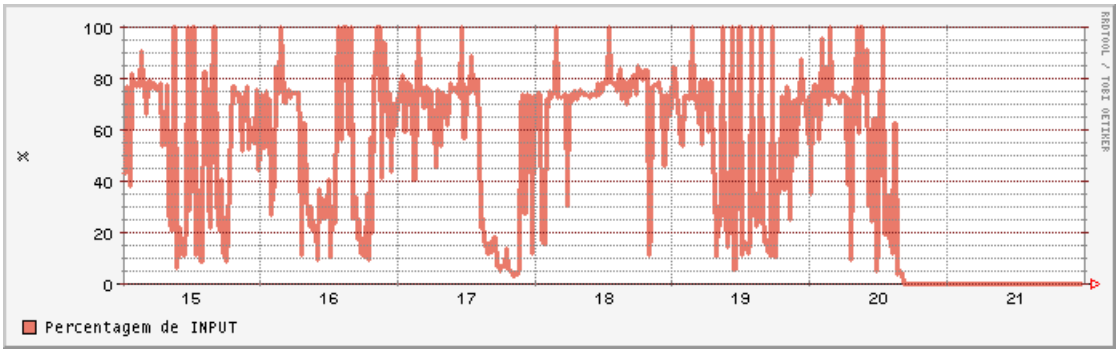


Figura 8.25: Percentual da vazão de entrada no enlace com a LLS

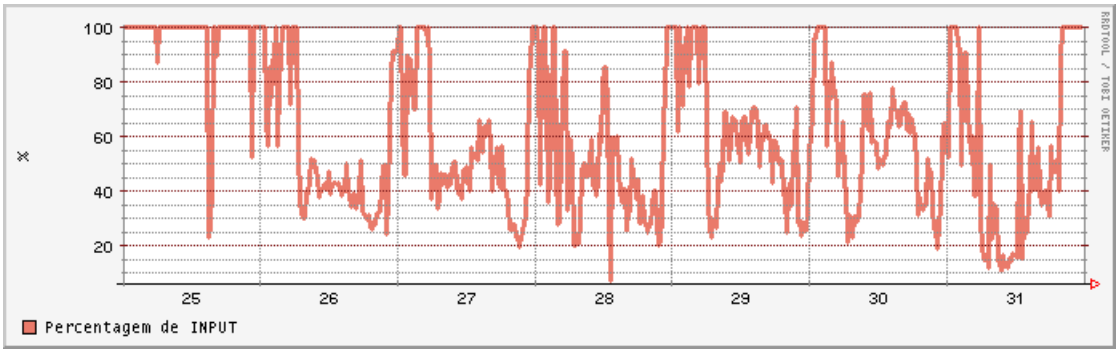


Figura 8.26: Percentual da vazão de entrada no enlace com a UERJ

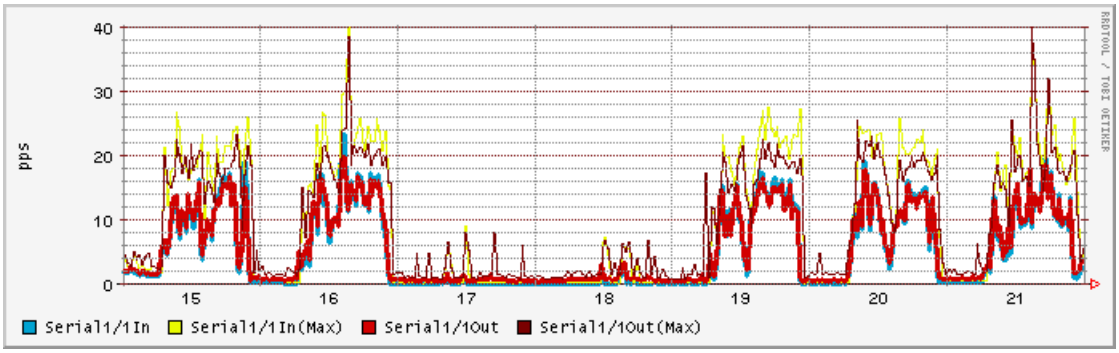


Figura 8.27: Vazão em pps no enlace com a SUAM

Vazão em pacotes por segundo

As figuras 8.27, 8.28, 8.29, 8.30 e 8.31 ilustram a vazão em pacotes por segundo dos enlaces da SUAM, Faculdade Carioca, IME, LLS e UERJ respectivamente.

A cor azul representa a quantidade de pacotes por segundo recebidos na entrada da interface do roteador de acesso em média de 30 minutos, a cor amarela representa o valor máximo no mesmo intervalo de 30 minutos.

A cor vermelha clara representa a quantidade de pacotes por segundo enviados na saída da interface do roteador de acesso em média de 30 minutos, a cor vermelha escura representa o valor máximo no mesmo intervalo de 30 minutos.

Pacotes descartados na fila de saída

As figuras 8.32, 8.33, 8.34, 8.35 e 8.36 ilustram a quantidade de pacotes descartados por segundo dos enlaces da SUAM, Faculdade Carioca, IME, LLS e UERJ respectivamente.

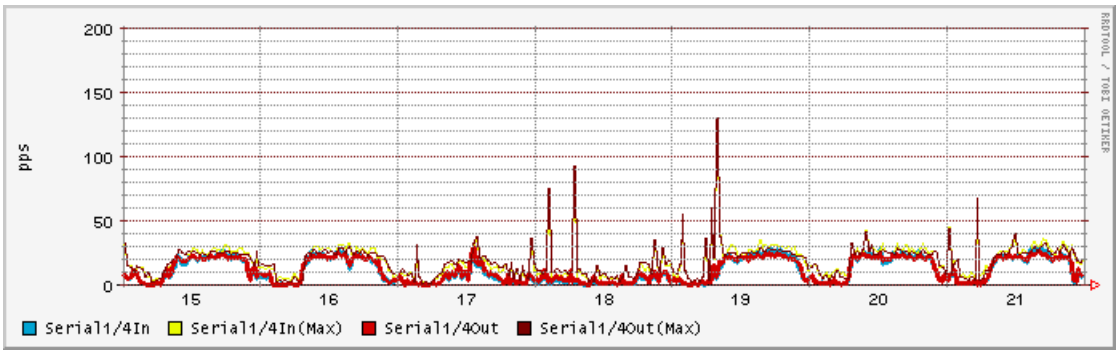


Figura 8.28: Vazão em pps no enlace com a Faculdade Carioca

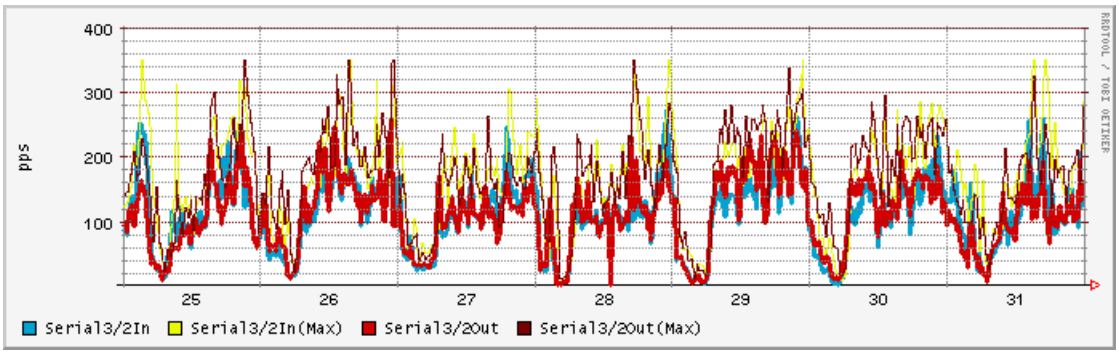


Figura 8.29: Vazão em pps no enlace com o IME

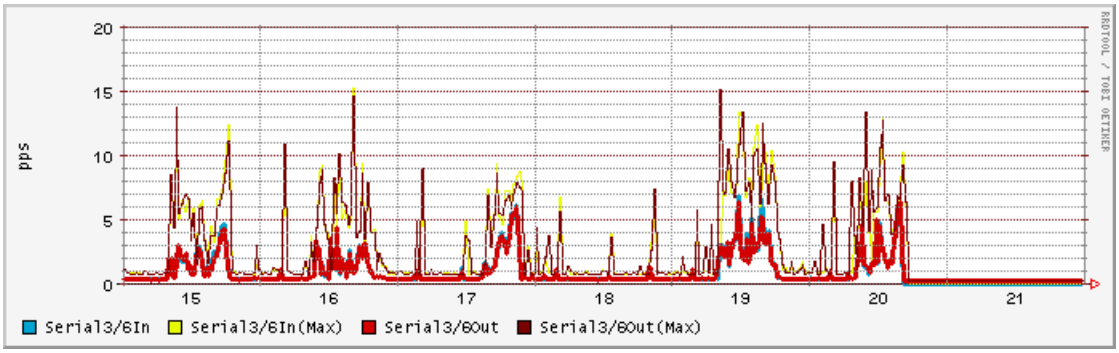


Figura 8.30: Vazão em pps no enlace com a LLS

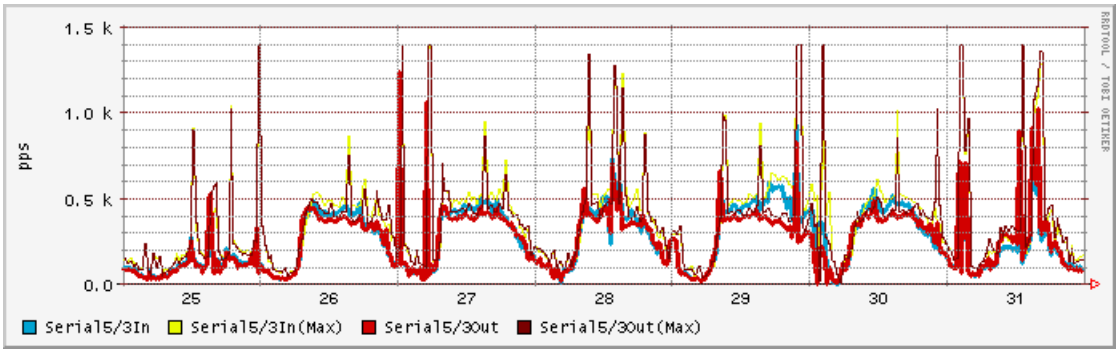


Figura 8.31: Vazão em pps no enlace com a UERJ

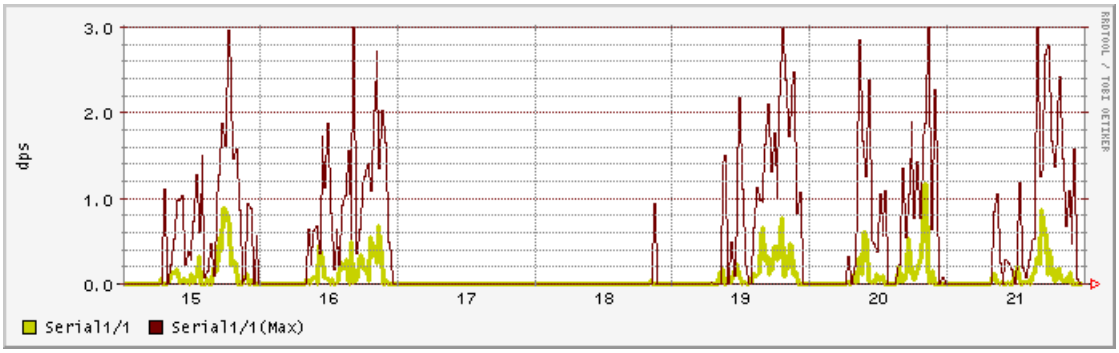


Figura 8.32: Pacotes descartados na fila de saída por segundo no enlace com a SUAM

A cor amarela representa a quantidade de pacotes descartados por segundo na fila de saída da interface do roteador de acesso em média de 30 minutos, a cor vermelha escura representa o valor máximo no mesmo intervalo de 30 minutos.

Latência para o roteador do cliente

As figuras 8.37, 8.38, 8.39, 8.40 e 8.41 ilustram a latência em milesegundos (ida e volta) da máquina coletora até o roteador do outro lado, nos enlaces da SUAM, Faculdade Carioca, IME, LLS e UERJ respectivamente.

A cor amarela representa a latência entre o roteador do cliente e a máquina coletora em uma média de 30 minutos, a cor vermelha escura representa o valor máximo no mesmo intervalo de 30 minutos.

Jitter para o roteador do cliente

As figuras 8.42, 8.43, 8.44, 8.45 e 8.46 ilustram o *jitter* em milesegundos por segundo (ida e volta) da máquina coletora até o roteador do outro lado, nos

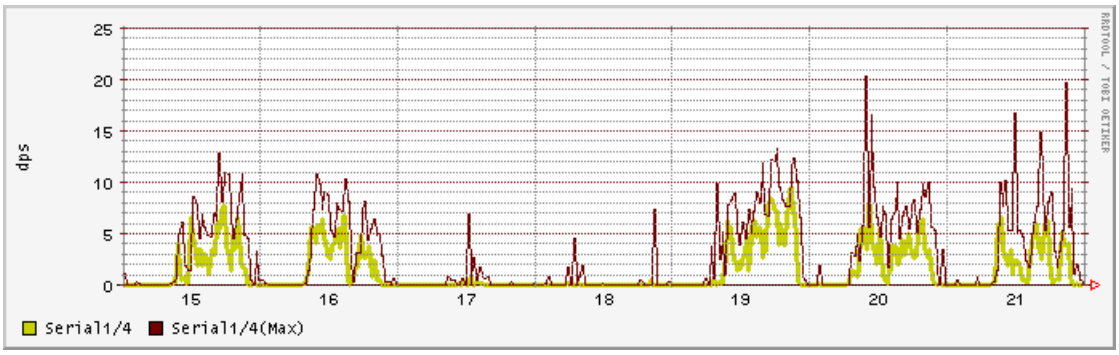


Figura 8.33: Pacotes descartados na fila de saída por segundo no enlace com a Faculdade Carioca

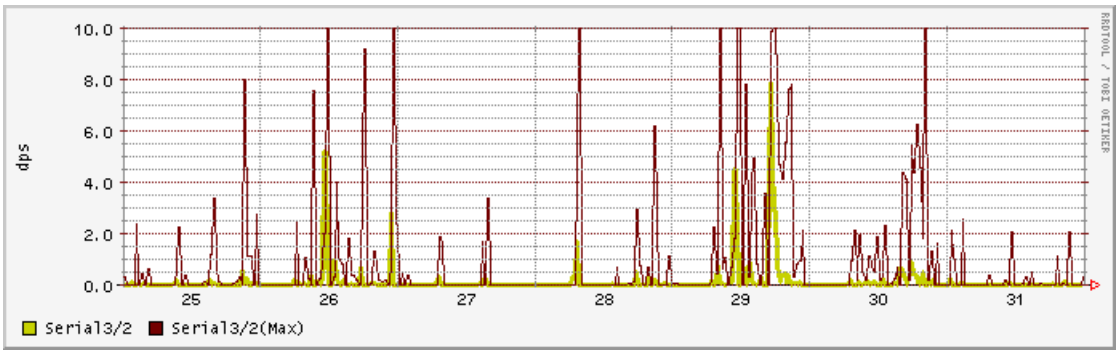


Figura 8.34: Pacotes descartados na fila de saída por segundo no enlace com o IME

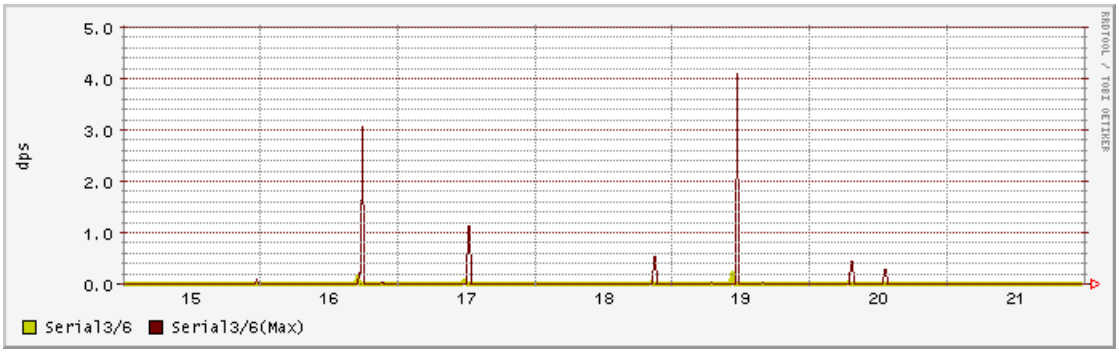


Figura 8.35: Pacotes descartados na fila de saída por segundo no enlace com a LLS

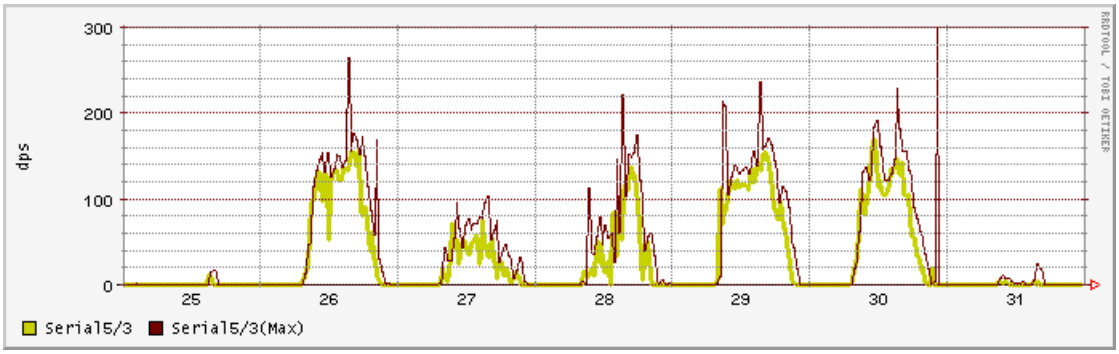


Figura 8.36: Pacotes descartados na fila de saída por segundo no enlace com a UERJ

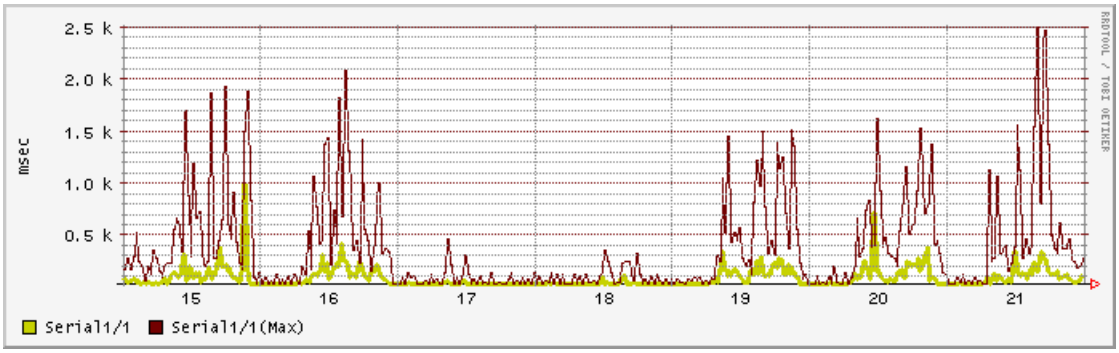


Figura 8.37: Latência em ms para o roteador da SUAM

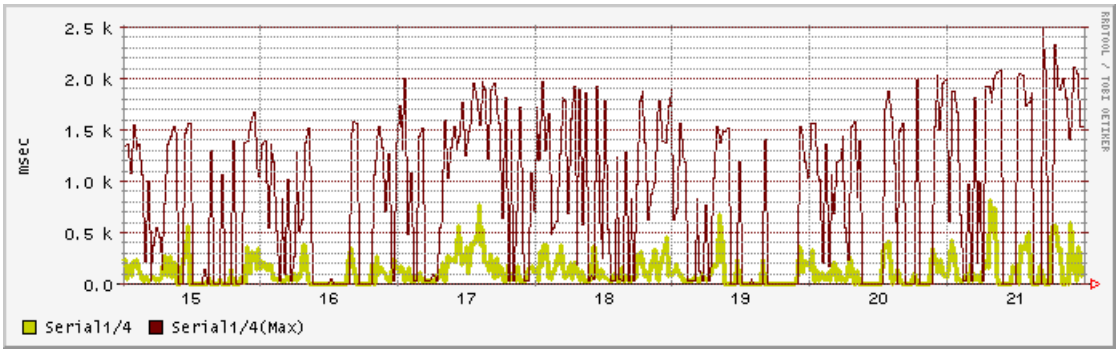


Figura 8.38: Latência em ms para o roteador da Faculdade Carioca

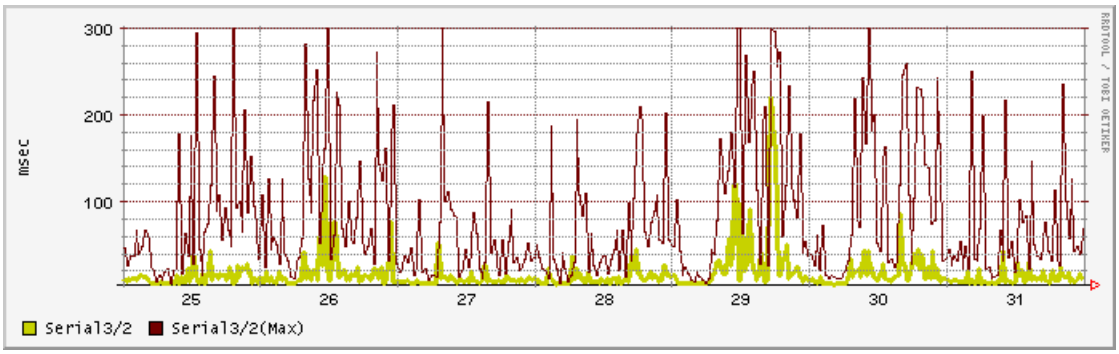


Figura 8.39: Latência em ms para o roteador do IME

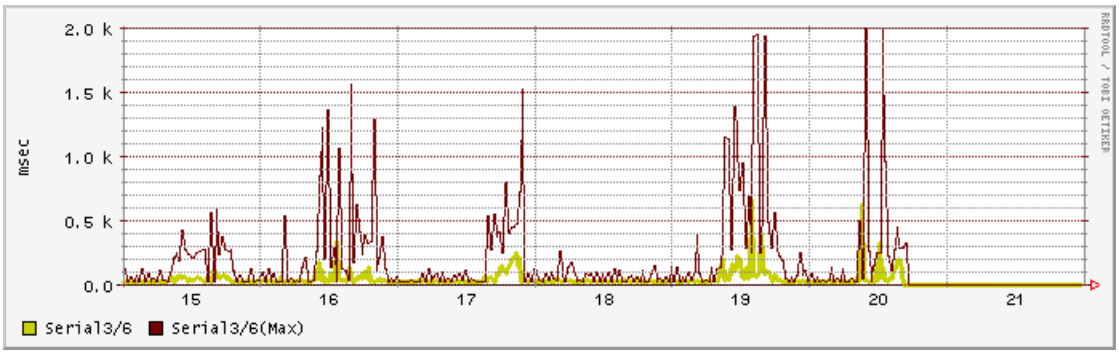


Figura 8.40: Latência em ms para o roteador da LLS

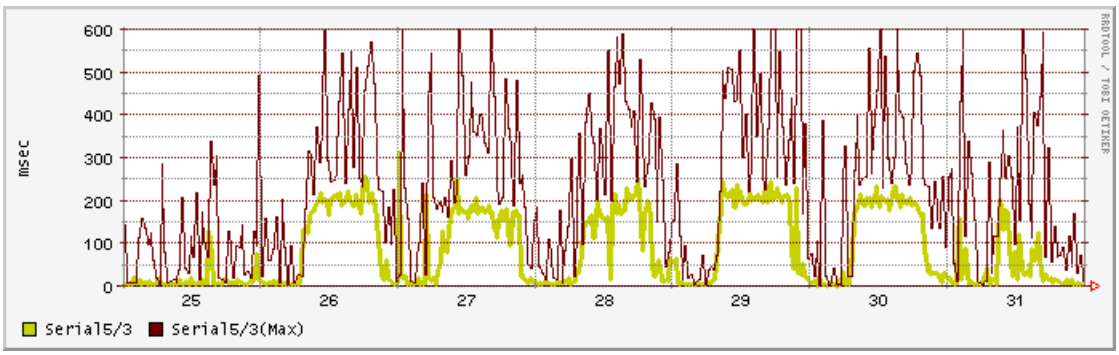


Figura 8.41: Latência em ms para o roteador da UERJ

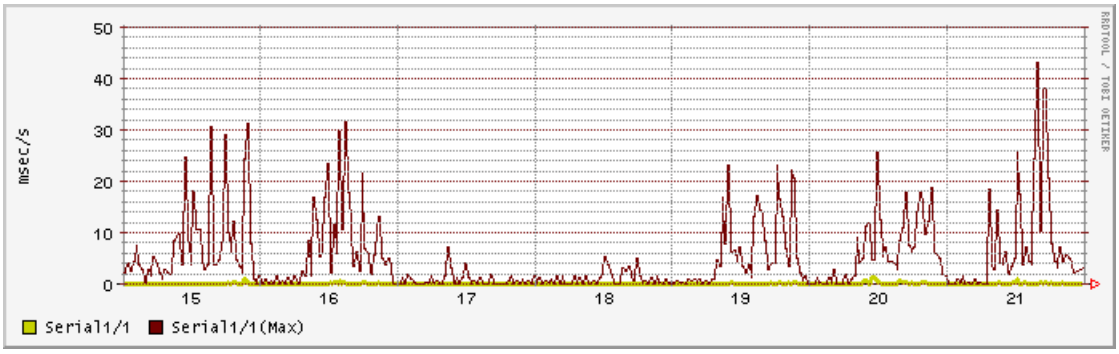


Figura 8.42: Jitter em ms/s para o roteador da SUAM

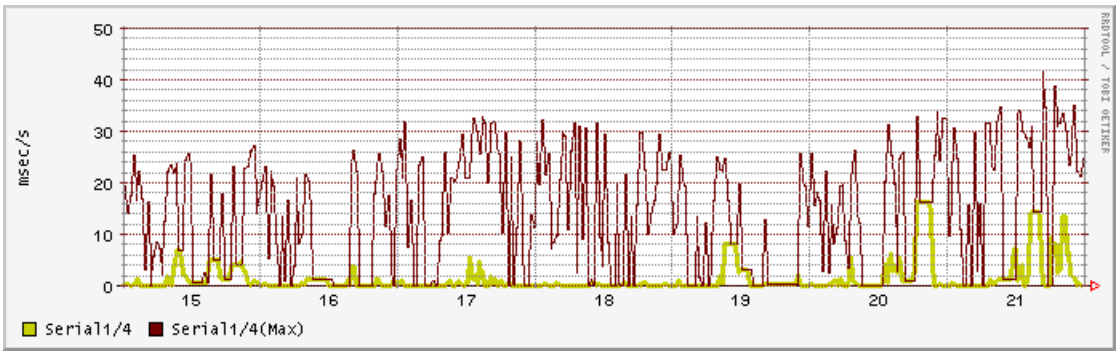


Figura 8.43: Jitter em ms/s para o roteador da Faculdade Carioca

enlaces da SUAM, Faculdade Carioca, IME, LLS e UERJ respectivamente.

A cor amarela representa o *jitter* entre o roteador do cliente e a máquina coletora em uma média de 30 minutos, a cor vermelha escura representa o valor máximo no mesmo intervalo de 30 minutos.

8.2.3 Comentários e conclusões

A proposta deste exemplo é criar um meio de caracterizar os enlaces de acesso através de sua monitoração.

Latência e *jitter*

Para se coletar os dados de latência e de *jitter* o ideal era colocar duas maquinas coletoras, uma ligada no Backbone da RedeRio e uma outra ligada do outro lado de cada enlace que se quer medir. O problema é que os custos, o acesso a rede e o tempo inviabilizariam a implementação.

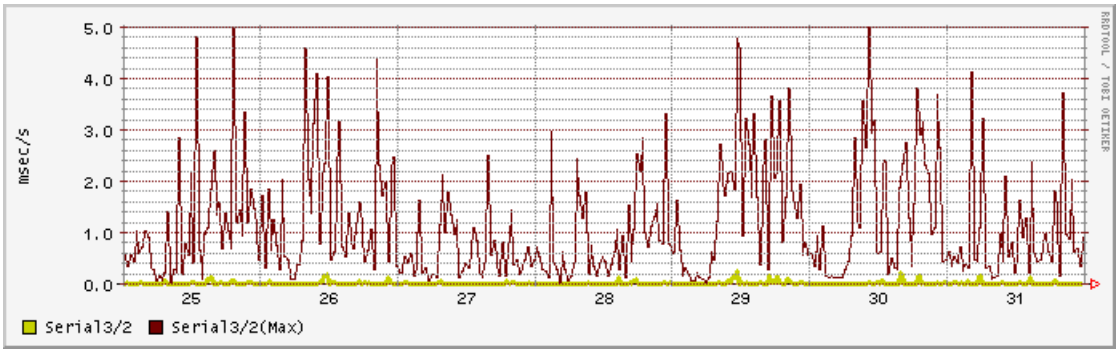


Figura 8.44: Jitter em ms/s para o roteador do IME

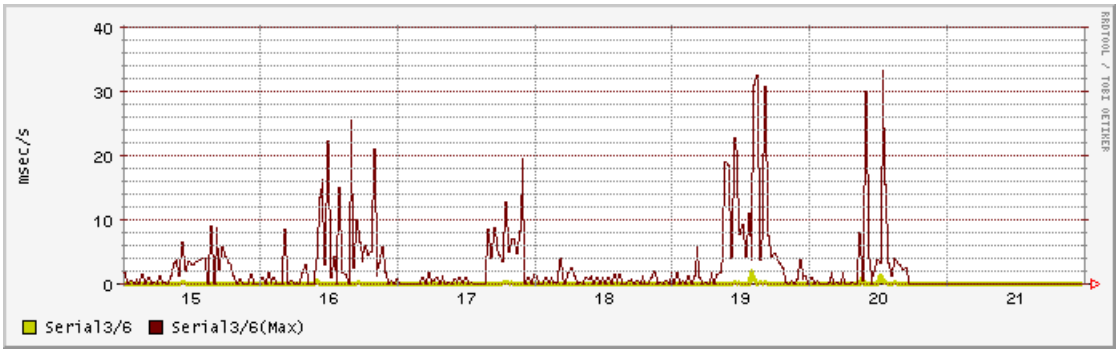


Figura 8.45: Jitter em ms/s para o roteador da LLS

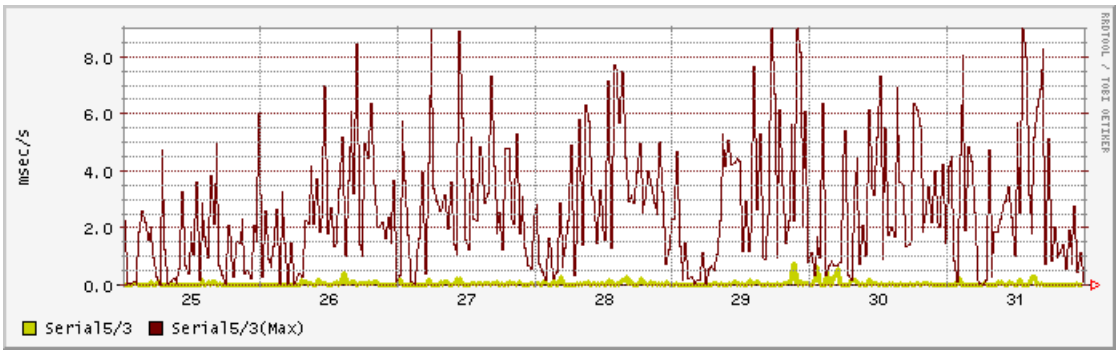


Figura 8.46: Jitter em ms/s para o roteador da UERJ

Saturação e tratamento de filas

Dois dos 5 enlaces escolhidos são de 2Mbps, um da UERJ e outro do IME. A diferença entre o enlace da UERJ em relação ao do IME é que a utilização do enlace durante os dias da semana é muito superior a sua capacidade. Os outros 3 enlaces são de 64Kbps, sendo o enlace com a Faculdade Carioca com saturação e o da LLS sem saturação.

A informação de vazão apresentada na figura 8.21 é insuficiente para avaliar o quanto saturado está o enlace. Com a utilização adicional dos gráficos de vazão de pacotes e de descarte (figuras 8.31 e 8.36) é possível verificar que a quantidade de pacotes descartados na interface chega a 40% em média. No exemplo da UERJ, estes 40% refletem uma vazão de aproximadamente 800Kbps. Ou seja, o enlace da UERJ na verdade consome dentro do Backbone da RedeRio uma vazão de 2800Kbps, porém apenas 2Mbps são efetivamente enviados.

No exemplo da Faculdade Carioca, a percentagem de pacotes descartados chega a um valor próximo de 40% (figuras 8.18 e 8.33) , o que reflete uma vazão de aproximadamente 25Kbps. A vazão causada por dercarte de pacotes na interface da Faculdade Carioca não representa muito em relação a uma interfaces de maior capacidade como a da UERJ.

Pela figura 8.41 é possível estimar o tamanho da fila de saída. Conforme a figura 8.41 a latência do enlace quando saturado (utilização plena da fila) é de aproximadamente 200ms em média. Considerando os tempos de propagação e de transmissão despresíveis, o tempo gasto na fila de saída será de 200ms, logo o tamanho da fila pode ser estimado em aproximadamente 50Kbytes (2048Kbps / 8bytes x 0.2s = 51200bytes).

Este resultado pode ser aproximado para este valor pois a política de tratamento da fila de saída é FIFO. Políticas de tratamento de fila como *Weight Fair Queue* podem melhorar o tempo de resposta para pacotes pequenos, porém são efeitos ilusórios que apenas pioram o desempenho das outras aplicações como WWW. Uma política de aumento de fila de saída ou de utilização de RED para o enlace da UERJ é praticamente inútil, uma vez que a taxa de chegada é muito maior que a taxa de serviço.

Ao contrário do enlace da UERJ, o aumento do tamanho fila pode ser utilizado no enlace do IME para evitar os pequenos descartes de pacotes verificados no gráfico 8.34. Uma política melhor no tratamento das filas de saída também pode ser utilizada para evitar grandes variações na latência (figuras 8.39 e 8.44).

Em algumas arquiteturas de roteadores a saturação em uma interface pode causar degradação em outras interfaces que utilizam de forma compartilhada o acesso ao *buffer* (Cisco 7500).

Uma solução para evitar que este tráfego inútil circule no Backbone, é controlar o fluxo de entrada de dados da interface, limitando assim o número de solicitações. Porém esta limitação só pode ser feita em enlaces que tem a característica de cliente, ou seja, que tem a maioria do tráfego de entrada no Backbone como sendo de requisição a informações.

Característica de cliente ou servidor

A maior aplicação na Internet hoje é o HTTP (maior que 80%) e o seu comportamento é do tipo cliente servidor. Esta característica torna o tráfego de dados, em termos de quantidade de bytes, assimétrico. Se o enlace possuir uma vazão em bps assimétrica ele tem predominantemente uma característica de cliente ou de servidor. Se a vazão em bps de entrada no Backbone for maior que a saída a sua característica predominante é a de servidor, se a vazão maior for a de saída do Backbone a sua característica predominante é a de cliente.

Na vazão em pps (figuras 8.27, 8.29 e 8.30) a característica de assimitricidade não existe como verificado nos gráficos de bps (figuras 8.17, 8.19 e 8.20).

Nos enlaces com alta taxa de descarte de pacotes a avaliação de assimetricidade do tráfego pode conter erros.

As figura 8.22, , 8.24 e 8.25 ilustram a porcentagem de tráfego de entrada do backbone em relação a saída. O tráfego da SUAM tem maior característica de cliente do que o da LLS, pois o tráfego de entrada da SUAM é de em torno de 20% do tráfego de saída ao contrário da LLS que tem 75% do tráfego de saída na entrada (figuras 8.22 e 8.25).

Estas avaliações de tráfego ajudam no provisionamento de enlaces compartilhados (Frame-Relay ou ATM) e na colocação de *caches*.

Como a vazão de pacotes é praticamente simétrica em termos de entrada e saída (veja figuras 8.27, , 8.29 e 8.30) o controle de entrada de pacotes ao Backbone irá limitar a quantidade de pacotes de saída. Assim é possível criar um procedimento de controle que evite a saturação em enlaces com características de cliente, como nos enlaces da UERJ e da Faculdade Carioca (figuras 8.21, 8.36, 8.18 e 8.33)

Talvez a criação de um protocolo que avaliasse o estado destas interfaces e

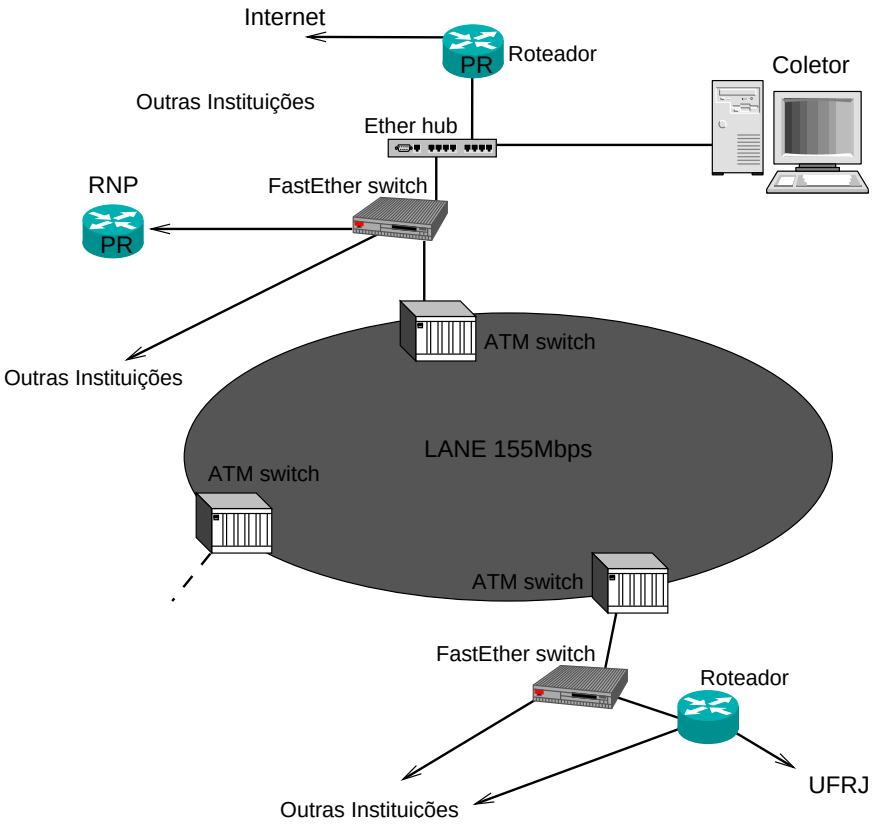


Figura 8.47: Topologia de coleta via libpcap na RedeRio

distribuisse uma penalidade no Backbone fosse uma boa proposta de solução.

8.3 Exemplo 3

8.3.1 Metodologia

Objetivos

Deseja-se obter relatórios que caracterizem o tráfego IP da UFRJ [86] passante no roteador do Backbone da RedeRio [82], localizado no CBPF [83].

Objetos existentes

Os objetos envolvidos nesta implementação estão na figura 8.47. O principal objeto de onde os dados são coletados é um Ethernet HUB, que apenas repete para a máquina coletora todos os pacotes passantes entre o roteador e o switch.

Medidas necessárias

As medidas realizadas nesta coleta serão: Vazão, Análise de fluxos TCP, Retransmissão TCP, Tamanho dos pacotes IP, Fragmentação dos pacotes IP, Fluxos de Comunicação IP e Segurança.

Método e recursos para coleta

Os recursos para a coleta dos dados são:

- Um Computador PC com 1 processador PentiumII de 300MHz;
- 256Mbytes de memória RAM;
- Uma placa de rede 10/100BASETX Intel Pro;
- Uma placa de rede 10BASET NE2000 Compatível;
- Um disco IDE de 4GBytes;
- Um disco IDE de 20Gbytes;
- Sistema operacional FreeBSD 4.2 [88];
- Pacote libpcap versão 0.6.2;
- Programa tcpdump versão 3.6.2;

O coletor funciona de modo passivo, apenas coletando o tráfego de pacotes IP passantes no Ethernet HUB da figura 8.47 A coleta de dados foi realizada no domingo e na segunda-feira dos dias 11/02/2001 e 12/02/2001. O processo de coleta foi feito através do tcpdump com *snap length* de 100bytes e com um filtro de MAC Ethernet ("`ether host 0:0:c:36:1e:ca`") para permitir apenas o tráfego da UFRJ (entrante e sainte) ser coletado.

Durante o dia 11/02/2001 (domingo) o espaço em disco utilizado para armazenamento foi de 7691658 Kbytes e no dia 12/02/2001 (segunda-feira) o espaço em disco foi de 10782981 Kbytes.

O processo de coleta para um período maior pode ser feito bastando que a os processos de análise e consolidação sejam bem definidos.

Método e recursos para consolidação e análise

Os resursos para análise são os mesmos utilizados na coleta, porém adicionando os seguintes ítens:

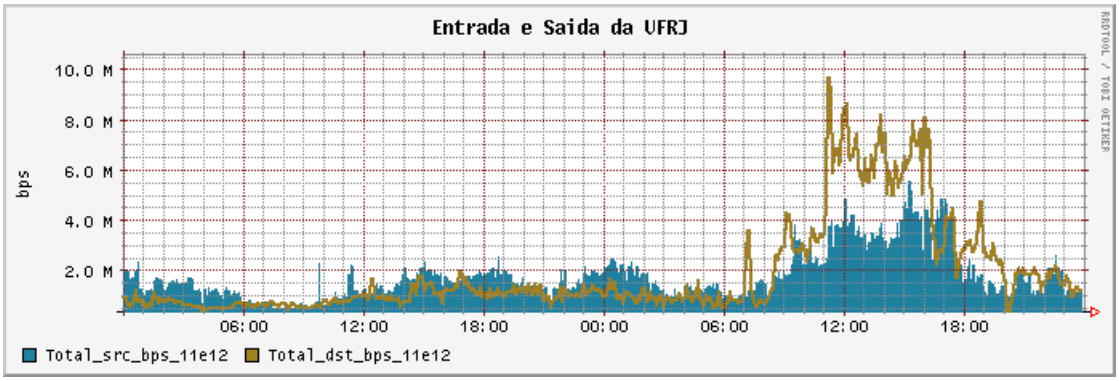


Figura 8.48: Vazão em bits por segundo nos dias 11/02 e 12/02

- Pacote PERL 5.005 [89] (desenvolvimento dos consolidadores);
- Pacote libpcap versão 0.6.2;
- Pacote libpng versão 1.0.9 (para criação de imagens PNG);
- Pacote GD versão 1.8.4 (auxiliar na criação de gráficos);
- Pacote RRDTool versão 1.0.28 (para armazenamento e criação de gráficos temporais);
- Programa desenvolvido para a consolidação dos dados;
- Programa desenvolvido para criação de gráficos 2D e 3D;

A consolidação dos dados foi feita dentro de períodos de 120 segundos. O armazenamento total dos dados consolidados para os 2 dias de coleta não passou de 5Mbytes.

Apresentação dos resultados

A apresentação dos dados será feita apartir de tabelas em texto, histogramas de distribuição de frequência e gráficos no tempo.

8.3.2 Análise e apresentação dos resultados

Vazão dos pacotes IP

A figura 8.48 mostra a vazão em bps (bits por segundo) de entrada e saída da UFRJ dos pacotes IP durante o período de coleta.

A figura 8.49 mostra a vazão em pps (pacotes por segundo) de entrada e saída da UFRJ dos pacotes IP durante o período de coleta.

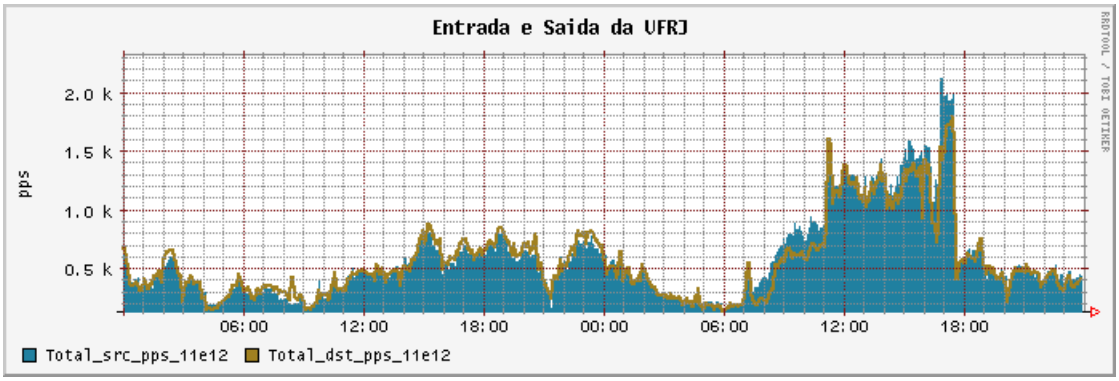


Figura 8.49: Vazão em pacotes por segundo nos dias 11/02 e 12/02

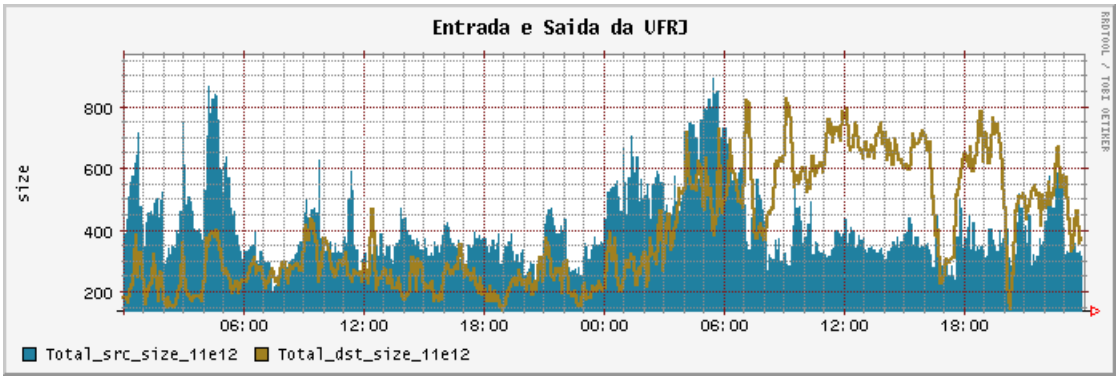


Figura 8.50: Tamanho dos pacotes IP nos dias 11/02 e 12/02

Tamanho dos pacotes IP

A figura 8.50 ilustra o tamanho dos pacotes IP de entrada e saída da UFRJ durante o período de coleta.

A tabela 8.10 ilustrada as propriedades da amostra coletada durante o intervalo.

Tabela 8.10: Valores estatíticos dos tamanhos do pacote IP

Amostras = 1433 (média de 2 minutos)
Mínimo médio = 120.916 bytes
Máximo médio = 940.272 bytes
Média das médias = 386.754 bytes
Variância = 187.919 (49% da média)
Simetria= 0.598
Peakness= 2.051

Fragmentação dos pacotes IP

A figura 8.51 ilustra a quantidade de pacotes IP fragmentados dentro dos intervalos de 2 minutos na entrada e saída da UFRJ durante o período de

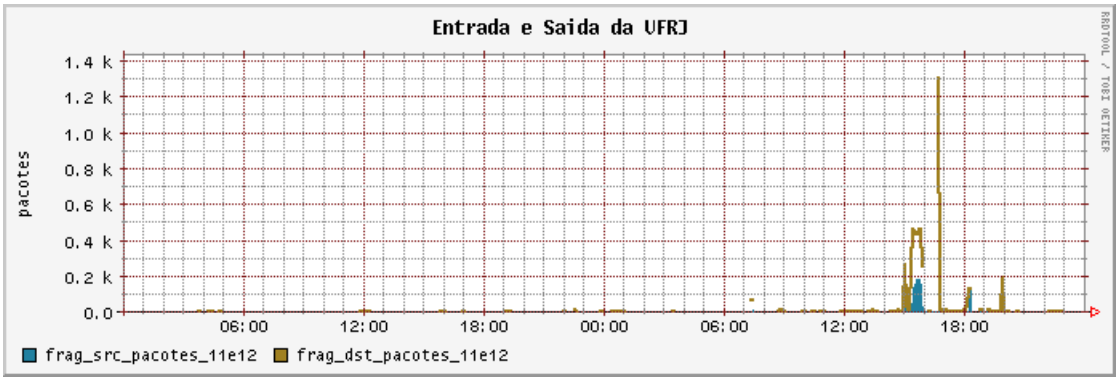


Figura 8.51: Pacotes IP fragmentados nos dias 11/02 e 12/02

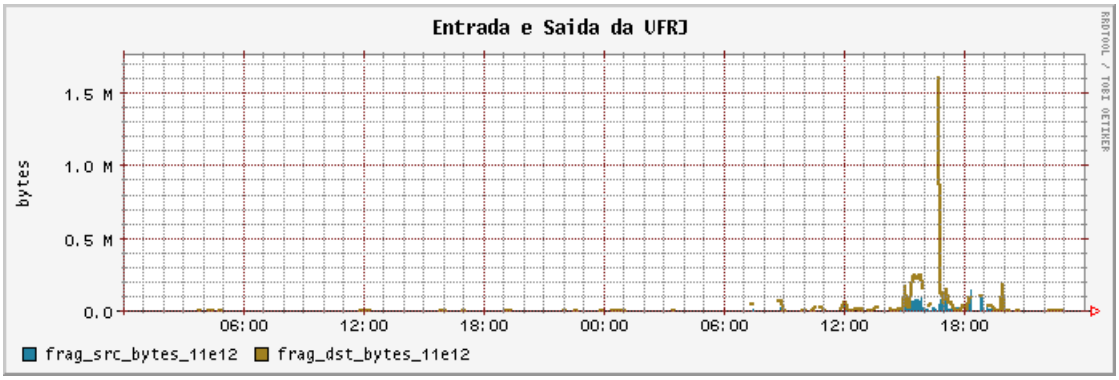


Figura 8.52: Quantidade de bytes dos fragmentos nos dias 11/02 e 12/02

coleta.

A figura 8.52 ilustra a quantidade em bytes dos pacotes IP fragmentados dentro dos intervalos de 2 minutos na entrada e saída da UFRJ durante o período de coleta.

A figura 8.53 ilustra a porcentagem da quantidade de pacotes que eram fragmentados com destino na UFRJ.

A figura 8.54 ilustra a porcentagem da quantidade de bytes que eram fragmentados com destino na UFRJ.

Protocolos

A tabela 8.11 mostra a distribuição da quantidade de pacotes e de bytes para os protocolos encontrados durante a coleta.

A figura 8.55 ilustra a porcentagem dos protocolos ICMP, UDP, TCP e IP-IP em relação ao total de tráfego de destino para UFRJ ao longo do tempo.

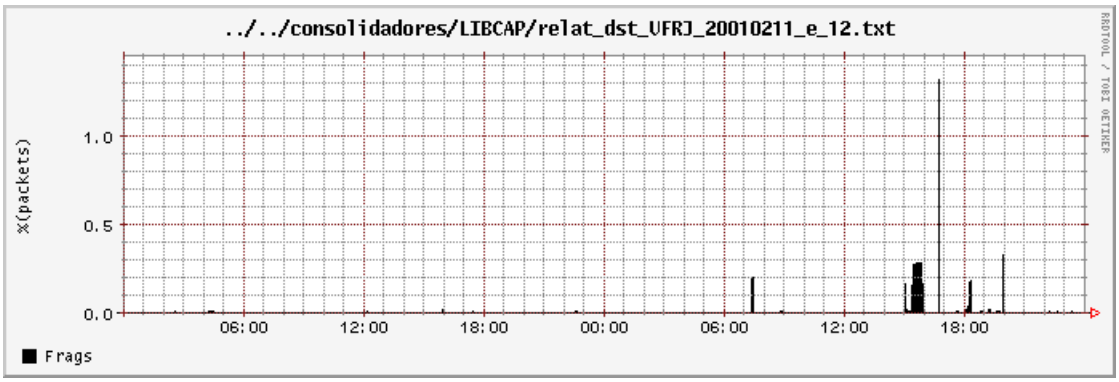


Figura 8.53: Porcentagem de pacotes IP fragmentados nos dias 11/02 e 12/02

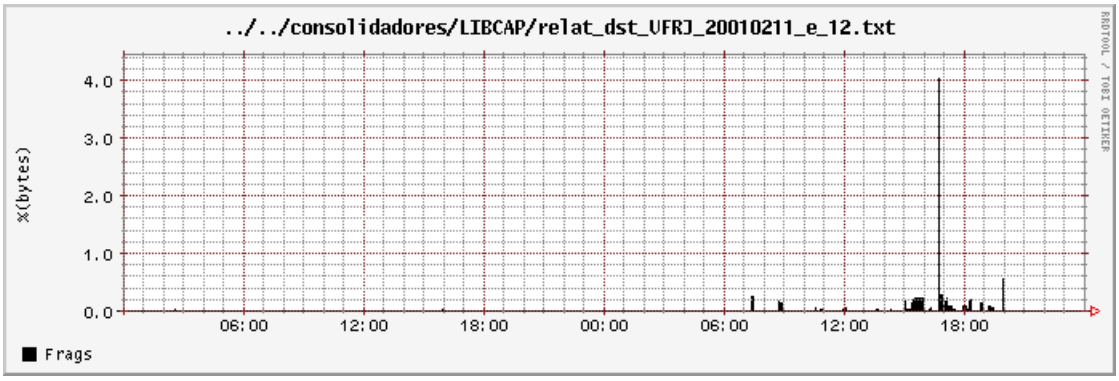


Figura 8.54: Porcentagem de bytes dos fragmentos nos dias 11/02 e 12/02

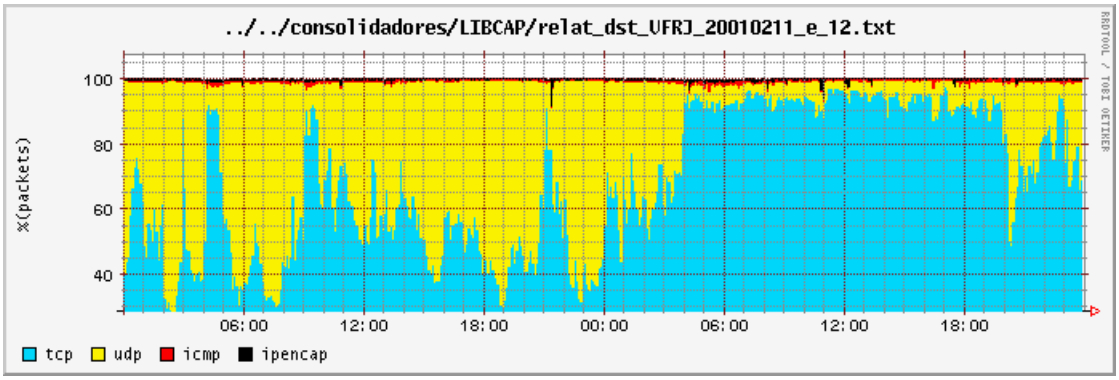


Figura 8.55: Porcentagem de protocolos em relação ao total de pacotes de destino

Tabela 8.11: Distribuição dos protocolos sobre o IP

Protocolo	pkts	bytes	% de pkts	% de bytes
tcp	143453025	73538456850	73.305	93.872
udp	50224336	4532676356	25.665	5.786
icmp	1609181	131560890	0.822	0.168
ipencap	217472	103564436	0.111	0.132
igmp	141805	18440338	0.072	0.024
ospf	19316	1947504	0.010	0.002
ax.25	27439	6961571	0.014	0.009
nsfnet-igp	392	5450086	0.000	0.007
narp	700	47600	0.000	0.000
unknown	8	12000	0.000	0.000
TOTAL	195693674	78339117631	100	100

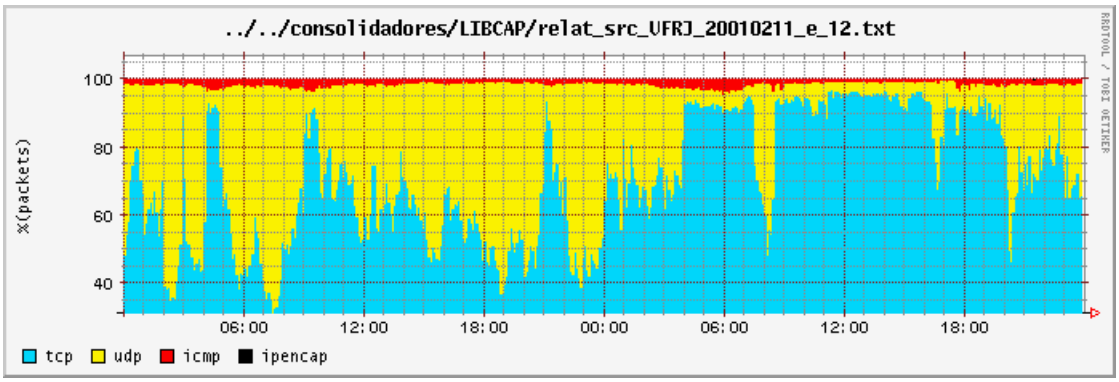


Figura 8.56: Porcentagem de protocolos em relação ao total de pacotes de origem

A figura 8.56 ilustra a porcentagem dos protocolos ICMP, UDP, TCP e IP-IP em relação ao total de tráfego de origem da UFRJ.

Flags TCP

A tabela 8.12 mostra a quantidade de pacotes e de bytes de todos os pacotes TCP encontrados durante a coleta. Os pacotes TCP estão classificados segundo as suas *flags*. A porcentagem mostrada está em relação a quantidade total de pacotes e bytes TCP.

A figura 8.57 ilustra a porcentagem das 8 maiores classes de flags TCP em relação ao total de pacotes TCP de destino para a UFRJ. A figura 8.58 ilustra o tráfego de origem da UFRJ.

A figura 8.59 ilustra a porcentagem das 8 maiores classes de flags TCP em relação ao total de bytes dos pacotes TCP de destino para a UFRJ.

A figura 8.60 ilustra a porcentagem das 8 maiores classes de flags TCP em relação ao total de bytes dos pacotes TCP de origem na UFRJ.

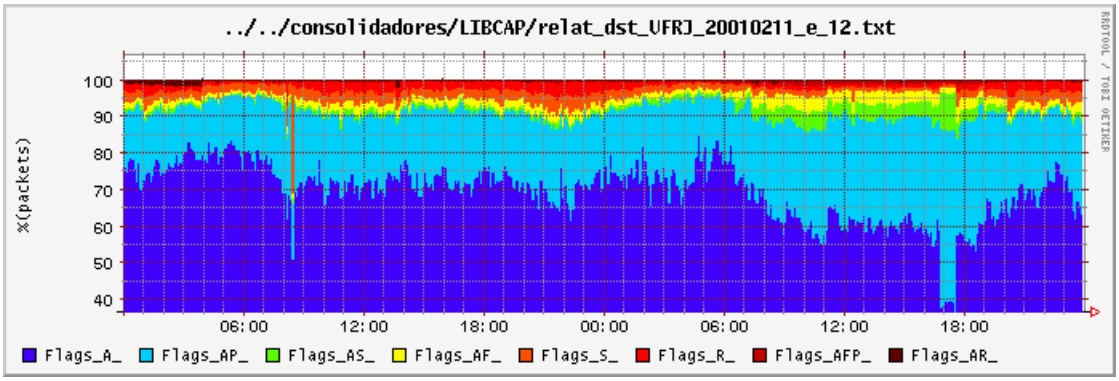


Figura 8.57: Porcentagem das classes de flags TCP relação ao total de pacotes destino

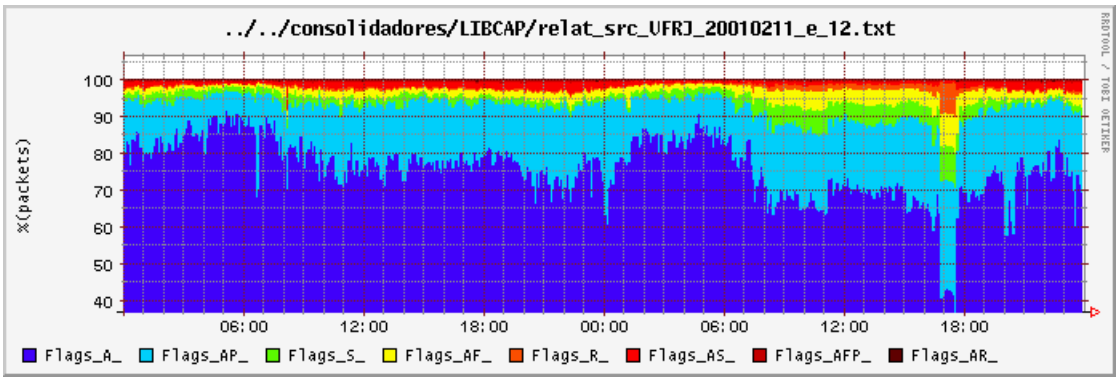


Figura 8.58: Porcentagem das classes de flags TCP relação ao total de pacotes origem

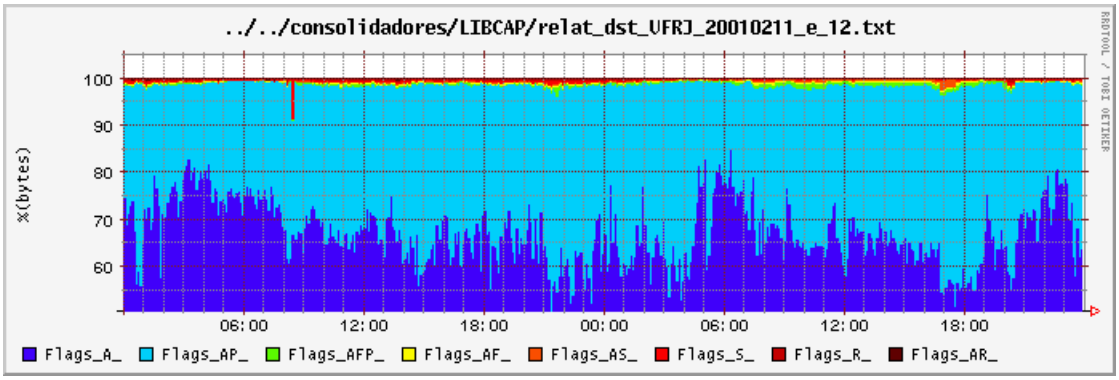


Figura 8.59: Porcentagem das classes de flags TCP relação ao total de bytes destino

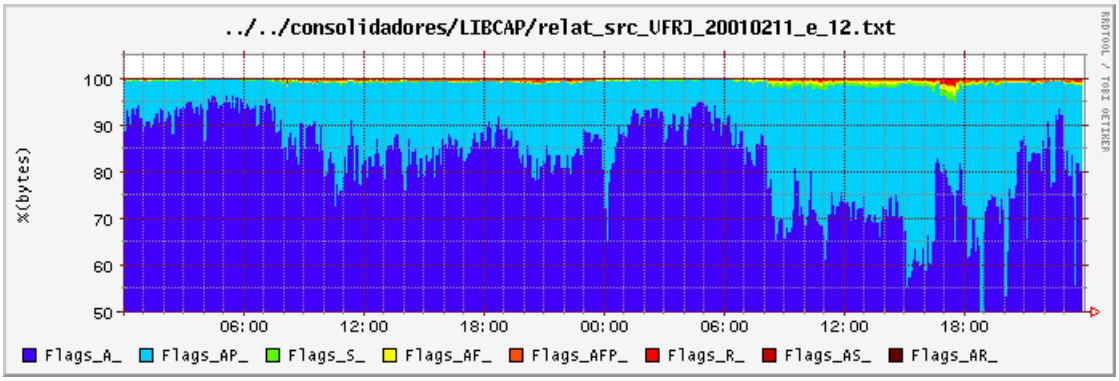


Figura 8.60: Porcentagem das classes de flags TCP relação ao total de bytes origem

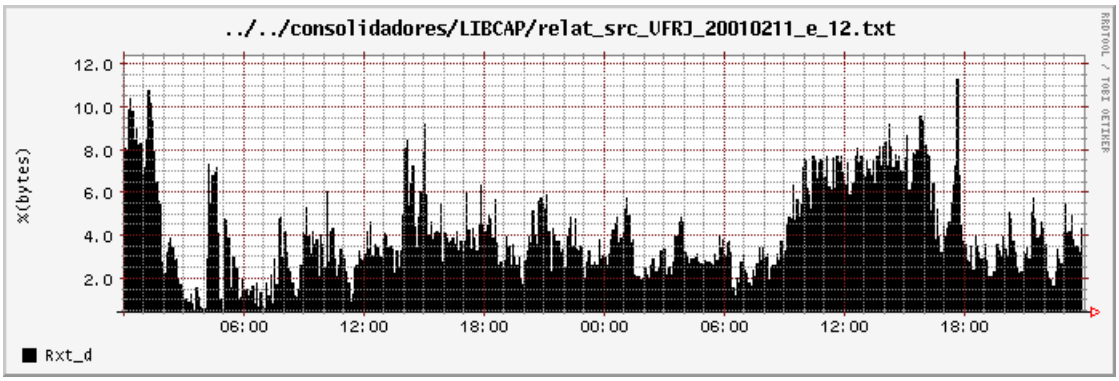


Figura 8.61: Porcentagem de bytes retransmitidos de origem na UFRJ

Retransmissão de dados TCP

A figura 8.61 ilustra a porcentagem dos bytes retransmitidos em relação ao total de bytes originados na UFRJ.

A figura 8.62 ilustra a porcentagem dos bytes retransmitidos em relação ao total de bytes destinados para UFRJ.

A figura 8.63 ilustra a porcentagem dos pacotes retransmitidos em relação ao total de bytes originados na UFRJ.

A figura 8.64 ilustra a porcentagem dos pacotes retransmitidos em relação ao total de bytes destinados para UFRJ.

Repetições de pacotes TCP

A figura 8.65 ilustra a porcentagem dos bytes repetidos em relação ao total de bytes de dados TCP originados na UFRJ.

A figura 8.66 ilustra a porcentagem dos bytes repetidos em relação ao

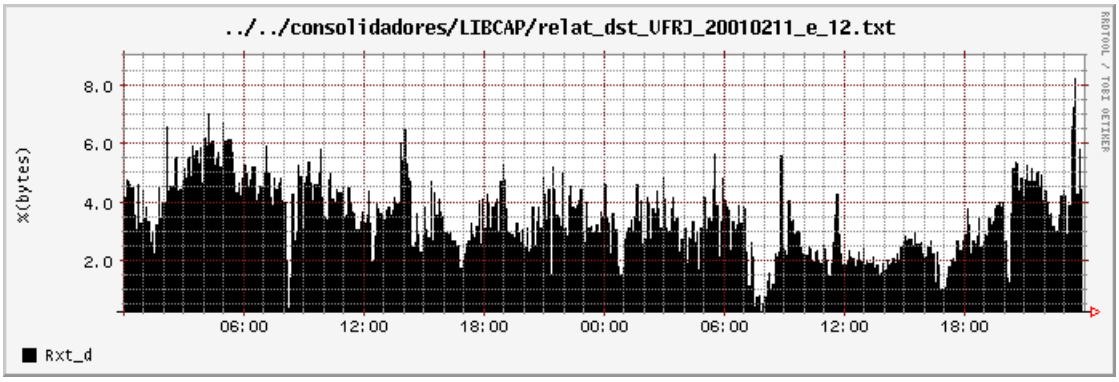


Figura 8.62: Porcentagem de bytes retransmitidos para a UFRJ

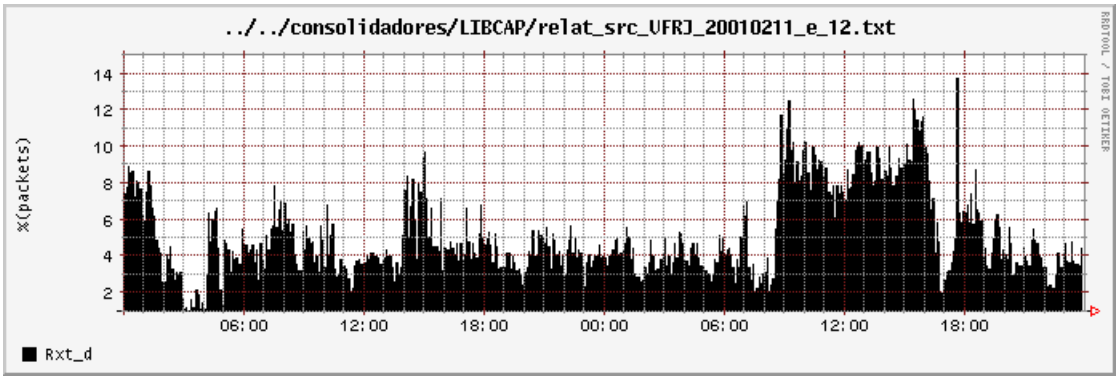


Figura 8.63: Porcentagem de pacotes retransmitidos de origem na UFRJ

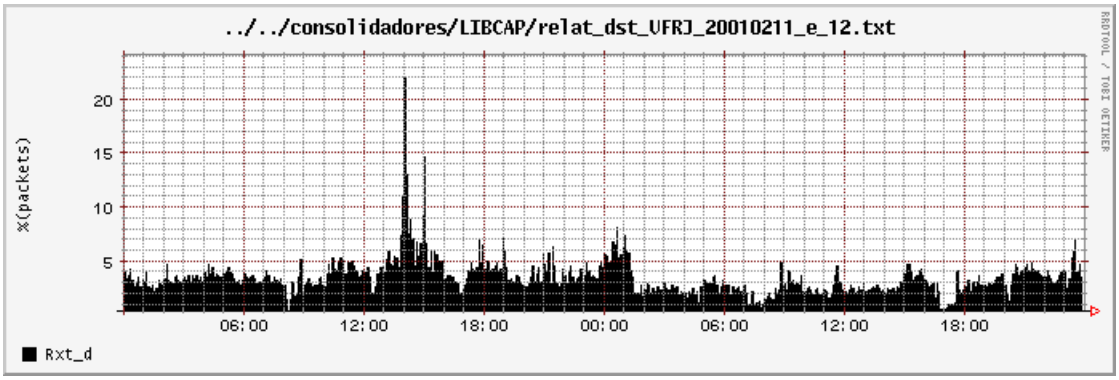


Figura 8.64: Porcentagem de pacotes retransmitidos para a UFRJ

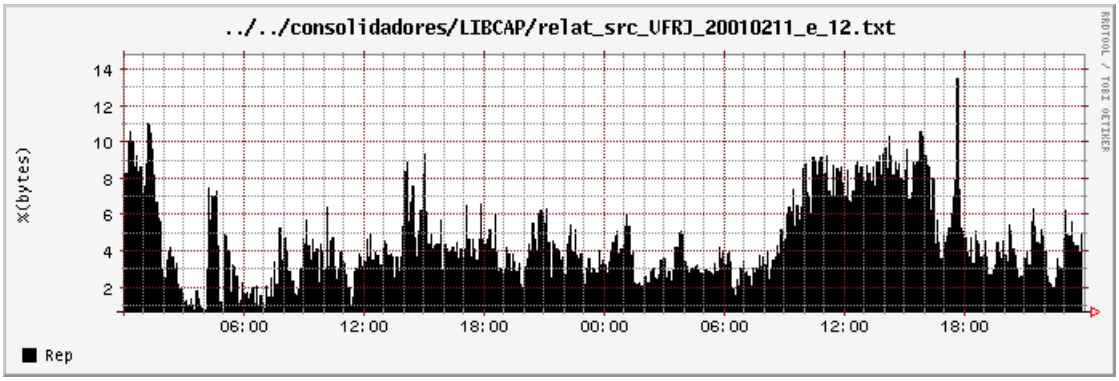


Figura 8.65: Porcentagem de bytes repetidos de origem na UFRJ

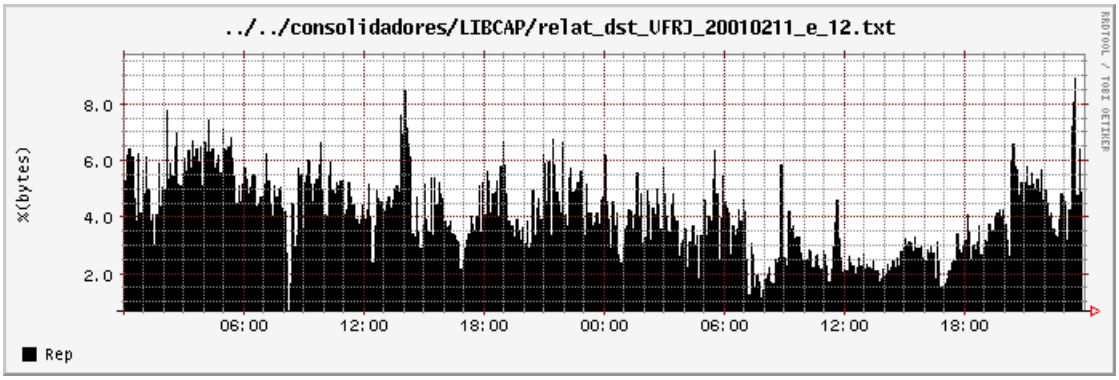


Figura 8.66: Porcentagem de bytes repetidos para a UFRJ

total de bytes de dados TCP destinados para UFRJ.

A figura 8.67 ilustra a porcentagem dos pacotes repetidos em relação ao total de pacotes de dados originados na UFRJ.

A figura 8.68 ilustra a porcentagem dos pacotes repetidos em relação ao total de pacotes de dados destinados para UFRJ.

8.3.3 Comentários e conclusões

Este exemplo é o mais completo em termos de coleta de dados, pois armazena o cabeçalho de todos os pacotes IPs passantes. Com ele é possível avaliar uma série de variáveis.

Vazão de entrada e saída

A figura 8.48 mostra a vazão em bps registrada nos pacotes IP passantes de entrada e saída para a UFRJ. Pelo gráfico é possível verificar que a característica do tráfego da UFRJ é de cliente apartir das 11hs até as 16hs no

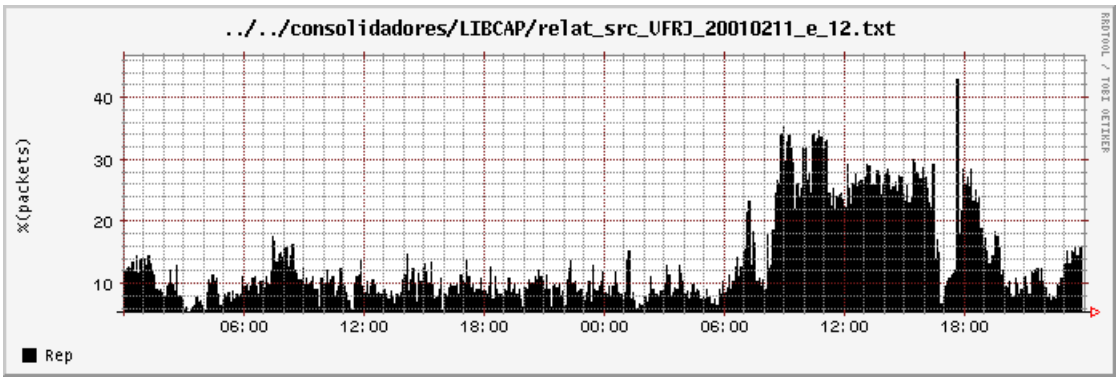


Figura 8.67: Porcentagem de pacotes repetidos de origem na UFRJ

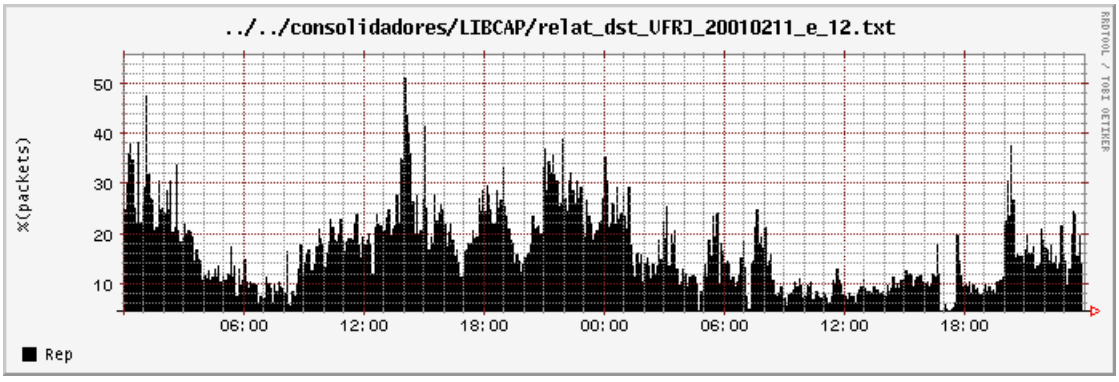


Figura 8.68: Porcentagem de pacotes repetidos para a UFRJ

dia 12/02 (segunda-feira), porém o tráfego tem uma característica de servidor no dia 11/02 (domingo) e durante os horários do dia 12/02 dia 11/02 (as definições de tráfego cliente e servidor estão no exemplo anterior)

Tamanho dos pacotes IP

A figura 8.50 mostra a média do tamanho do pacote no tráfego de entrada e de saída durante a coleta dos dados. A característica de cliente pode ser observada durante o tráfego no horário das 11hs até as 16hs de segunda-feira, onde a média do tamanho do pacote de ida para a UFRJ foi bem maior que a média do tamanho do pacote de subida da UFRJ.

Fragmentação dos pacotes IP

As figuras 8.51 e 8.52 mostram a quantidade de pacotes fragmentados na rede. Apenas foi detectado um aumento de tráfego de fragmentos das 15hs às 17hs do dia 12/02. Muitas vezes o aumento de pacotes fragmentados indica uma possível falha na escolha do MTU de algum caminho, porém pacotes IP fragmentados são utilizados para ataques a redes. Neste exemplo existe grande chance de ser um tipo de ataque. Esta monitoração é importante para verificar possíveis melhorias no MTU das redes ou verificar a incidência de ataques. Porém neste exemplo este tráfego é praticamente desprezível durante os outros períodos.

Analisando detalhadamente os pacotes IP fragmentados, foi verificado que o pico no gráfico da figura 8.51 foi causado por um rajada de pacotes UDP fragmentados de origem no IP `200.246.248.120` e destino `146.164.33.33`. A rajada iniciou as 16:50:36 e terminou as 16:57:33, sendo enviado um total de 4873 pacotes.

Início do bloco:

```
15:50:36.874553 200.246.248.120.3971 > 146.164.33.33.1201:  udp 6146 (frag 52566:1480@0+)
15:50:36.875173 200.246.248.120 > 146.164.33.33: (frag 52566:1480@1480+)
15:50:36.875824 200.246.248.120 > 146.164.33.33: (frag 52566:1480@2960+)
15:50:36.876560 200.246.248.120 > 146.164.33.33: (frag 52566:1480@4440+)
15:50:36.876587 200.246.248.120 > 146.164.33.33: (frag 52566:234@5920)
```

Término do bloco:

```
15:57:33.028713 200.246.248.120.3971 > 146.164.33.33.1201:  udp 6147 (frag 63873:1480@0+)
15:57:33.029205 200.246.248.120 > 146.164.33.33: (frag 63873:1480@1480+)
15:57:33.029858 200.246.248.120 > 146.164.33.33: (frag 63873:1480@2960+)
15:57:33.029885 200.246.248.120 > 146.164.33.33: (frag 63873:235@5920)
15:57:33.030843 200.246.248.120 > 146.164.33.33: (frag 63873:1480@4440+)
```

Porcentagem de protocolos

Dentro de todos os protocolos encontrados durante o período de coleta os protocolos TCP, UDP, ICMP e IP-IP são responsáveis por mais de 99% de todo tráfego. As figuras 8.55 e 8.56 mostram a porcentagem de utilização destes 4 protocolos durante o tempo, de destino para UFRJ e de origem da UFRJ respectivamente. Pelos gráficos é possível verificar que o tráfego TCP chega em 95% durante o dia 12/02 das 04hs até 20hs. Conforme o gráfico de vazão da figura 8.48, o aumento de tráfego se iniciou por volta das 09hs, porém o aumento no tráfego TCP iniciou apartir das 04hs. Com estes resultados podemos concluir que o crescimento relativo da vazão do tráfego TCP não acompanha o crescimento da vazão total. Essa diferença deve ser melhor verificada em uma análise mais detalhada dos fluxos TCP neste período.

Flags TCP

Dentro de um pacote TCP existem 6 *flags* responsáveis pela identificação do seu propósito. A tabela 8.12 ilustra a distribuição das *flags* TCP dentro do conjunto de pacotes TCP coletados. Em um fluxo normal de comunicação TCP os pacotes podem assumir as seguintes 11 combinações de *flags*:

S : Responsável pelo início de uma conexão;

AS : Reconhecimento de uma conexão;

A ou AP : Reconhecimento (pode conter dados);

AR ou R : Reset de uma conexão ou de uma tentativa de conexão;

AF ou AFP : Finalização de uma conexão;

R, AR ou ARP : Finalização de uma conexão ou recusa de.

As flags `A` e `AP` são responsáveis por 90% de todos os pacotes TCP e carregam 98.7% de todos os bytes. O que é normal, uma vez que são responsáveis por carregar os dados. As outras combinações de *flags* dos pacotes TCP encontrados na coleta podem ter sido causadas por uma falha na implementação ou um possível ataque,

Uma medida que poderia ter sido feita ao longo do tempo é a relação entre os pacotes TCP com *flag* **S** e os com *flag* **AS**. Esta relação irá mostrar

o quanto as solicitações de conexão TCP obtiveram sucesso. Neste exemplo esta relação está em 0.7, o ideal é que seja 1.

Nos gráficos 8.57, 8.58 é possível verificar que durante o período iniciando em torno das 17:00hs e terminando um pouco antes das 18:00hsd do dia 12/02 houve um aumento de tráfego TCP de controle em relação ao tráfego de dados (A e AP). Isto pode ter sido causado por uma falha de roteamento ou no encaminhamento de muitos fluxos de comunicação. Esta falha pode ser confirmada também no gráfico do tamanho dos pacotes da figura 8.50, durante o mesmo período pode-se ver uma redução muito grande na média do tamanho de pacotes o que caracteriza que os pacotes de destino para a UFRJ deixaram de carregar informação, e o tráfego deixou de ser cliente. O aumento do número de pacotes no mesmo período tentanto conexão pode ser verificado na figura 8.49 A falha demostrada não pode ser verificada facilmente no gráfico de vazão em bps, figura 8.48.

Retransmissão de dados e repetição

As figuras 8.61, 8.62 representam a porcentagem de bytes retransmitidos nos fluxos de comunicação TCP. Na figura 8.61 é possível verificar que o aumento da porcentagem de pacotes retrasmitidos de origem na UFRJ acompanha o aumento de vazão de bytes mostrado na figura 8.48, chegando a 9% de todo o tráfego de dados TCP. Isso mostra um possível congestionamento de saída da UFRJ.

A monitoração desta variável é um fator importante na avaliação da qualidade das conexões TCP, e consequentemente da satifação dos usuários.

As retransmissões foram calculadas utilizando contadores indexados pelos campos: IP origem, Porta origem, IP destino, Porta destino e sequência TCP. Em intervalos de 120 segundos são computados a quantidade de pacotes TCP retransmitidos de forma não acumulativa, ou seja de 120 em 120 segundos todos os contadores de retramissão dos fluxos TCP são resetados. O ideal para os cálculos de retransmissão é utilizar contadores com tempo de reset independete.

As figuras 8.67 e 8.68 mostram a quantidade de pacotes repetidos de origem e destino da UFRJ. Os pacotes repetidos são todos os pacotes TCP que foram repetidos, incluindo pacotes com *flags* **S**, **R** e **F**. Quanto maior a porcentagem de tráfego repetido de origem maior é a característica do tráfego ser cliente, esta característica pode ser vista nas mesmas figuras e comparadas as figuras de vazão (figura 8.48) e de tamanho dos pacotes (figura 8.50).

A porcentagem de bytes das figuras 8.65 e 8.66 mostra o a eficiência do protocolo TCP.

O processo de cálculo de pacotes repetidos é similar ao utilizado no cálculo de retransmissão. Assim como o cálculo das restransmissões o ideal é utilizar contadores com tempo de reset independete.

8.4 Exemplo 4

8.4.1 Metodologia

Objetivos

Deseja-se obter relatórios que avaliem os fluxos de dados passantes em um roteador Internacional da Embratel.

Objetos existentes

Um roteador de saída internacional (conhecido como PR da figura 2.1), as interfaces de entrada, a interface que exporta os fluxos e a interface onde o coletor está ligado.

Variáveis de medida necessárias

Matrizes de tráfego por: ASN, rede e porta TCP. Distribuição dos protocolos IP e do tipo do serviço.

Método e recursos para coleta

Os recursos para a coleta dos dados são:

- Um Computador PC com 1 processador PentiumIII de 450MHz;
- 256Mbytes de memória RAM;
- Uma placa de rede 10/100Tx Intel EEPro;
- Um disco de 9GBytes Ultra-SCSI;
- Sistema operacional FreeBSD 4.0 [88];
- Pacote Cflowd Versão 2.1.a6 **??**;
- Porta FastEthernet em algum roteador do Backbone;

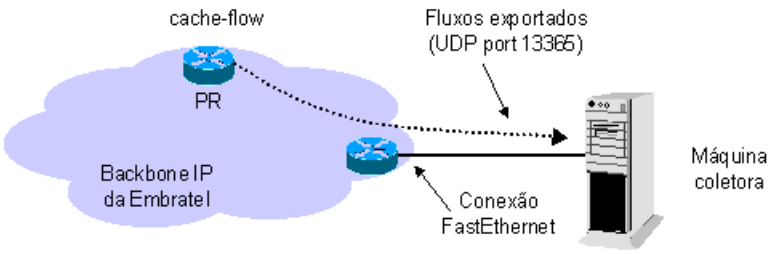


Figura 8.69: Topologia de coleta via NetFlow na Embratel

- Configuração de `cache flow` e habilitação de `export-flow` no roteador de interesse;

Foi escolhido um roteador internacional (PR figura 2.1) para a configuração de *cache* de roteamento baseado em fluxos (`route cache flow`) e do comando de exportação dos fluxos `export-flow`. Outro parâmetro passado na configuração da exportação dos fluxos é o IP do coletor e a porta de serviço que irá receber os fluxos exportados via UDP.

A coleta é feita de forma assíncrona através do envio de fluxos do roteador para o cliente (figura 8.69). A frequência do envio de fluxos depende de quando o fluxo é considerado terminado, ou quando a tabela de *cache* de fluxos fica cheia. A expiração do fluxo pode ocorrer por 2 motivos: por *time-out* ou porque foi detectado um final do fluxo (ex: RST ou FIN nos pacotes TCP).

Foi utilizada a versão 5 da exportação de fluxos (*NetFlow export version 5*). Dois métodos de exportação na versão 5 do NetFlow são suportados, o `origin-as` e o `peer-as`. A diferença entre eles é no ASN que será exportado, no `origin-as` o ASN exportado (tanto origem como destino) é o ASN que está no final do `route-path` da tabela de roteamento BGP e no `peer-as` o ASN exportado é o ASN que está imediatamente ligado ao Backbone, ou seja o ASN subsequente ao ASN 4230 no `route-path`.

Os prefixos de rede utilizados foram retirados da tabela de roteamento pelo processo `export-flow`.

A coleta foi realizada durante o mês de junho e julho de 2000.

Método e recursos para consolidação e análise

Os resursos para análise e consolidação são:

- Pacote Arts++ Versão 1.1.a5 [52];
- Pacote PERL 5.005 [89];

- Pacote GD versão 1 (auxiliar na criação de gráficos);
- Programa desenvolvido para exibição de matrizes em 3D;

A análise final foi feita sobre os dados armazenados no formato ARTS [52]. A análise dos dados foi feita sobre períodos de tempo variados.

Apresentação dos resultados

A apresentação dos dados foi feita apartir de tabelas em texto e gráficos em 2D e 3D.

8.4.2 Análise e apresentação dos resultados

Matriz por ASN

Para auxiliar no entendimento das matrizes de tráfego por ASN, na tabela 8.13 estão listados todos os ASNs mostrados neste exemplo e a sua instituição responsável. Estas informações foram retiradas através do `whois` do RIPE [77] e do ARIN [62].

A tabela 8.14 mostra a matriz de tráfego entre ASN origem e destino no período de 00:58:16 até 01:58:17 do dia 26/06/2000. Os ASNs mostrados na tabela se referem aos ASNs dos vizinhos da Embratel, e não aos ASNs que geraram ou destinarão o tráfego (configuração de `peer-as` do Netflow). Apenas os tráfegos entre ASNs que ultrapassaram a quantidade transferida de 5Mbytes foram mostrados, pois a tabela completa seria muito grande.

O gráfico da figura 8.70 ilustra a matriz de tráfego entre ASN origem e destino no período de 04:59:16 até 05:59:17 do dia 27/06/2000. Os ASNs mostrados na figura se referem aos ASNs dos vizinhos da Embratel (configuração de `peer-as` do Netflow). Na figura apenas foram representados os 7 ASNs destino que apresentaram maior tráfego em bps de destino. Na esquerda do gráfico estão os ASNs de origem e a direita os ASNs destino.

O gráfico da figura 8.71 ilustra a matriz de tráfego entre ASN origem e destino durante o dia 11/06/2000. Os ASNs mostrados na figura se referem aos ASNs que originaram ou destinaram o tráfego (configuração de `origin-as` do Netflow). Na figura apenas foram representados os 7 ASNs que apresentaram maior quantidade de pacotes recebidos. Na esquerda do gráfico estão os ASNs de origem e a direita os ASNs destino.

A tabela 8.15 mostra a matriz de tráfego entre ASN origem e destino no período de 08:58:18 até 08:59:16 do dia 27/06/2000. Os ASNs mostrados

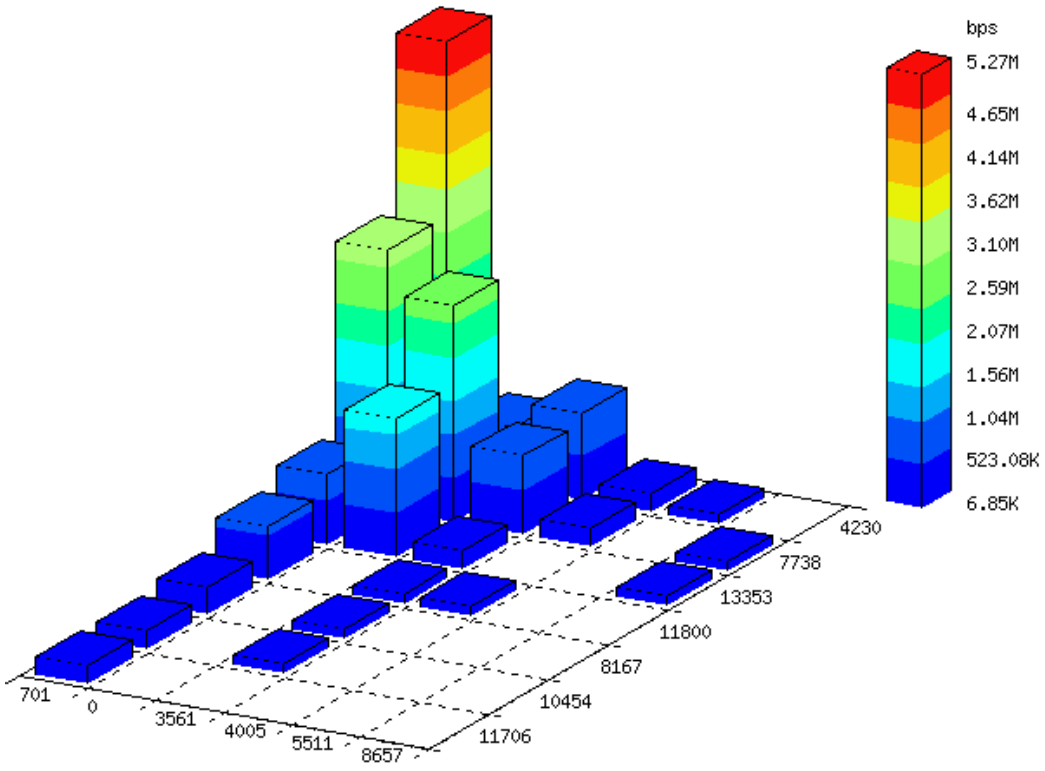


Figura 8.70: Matriz de tráfego por ASN no período de 1 hora

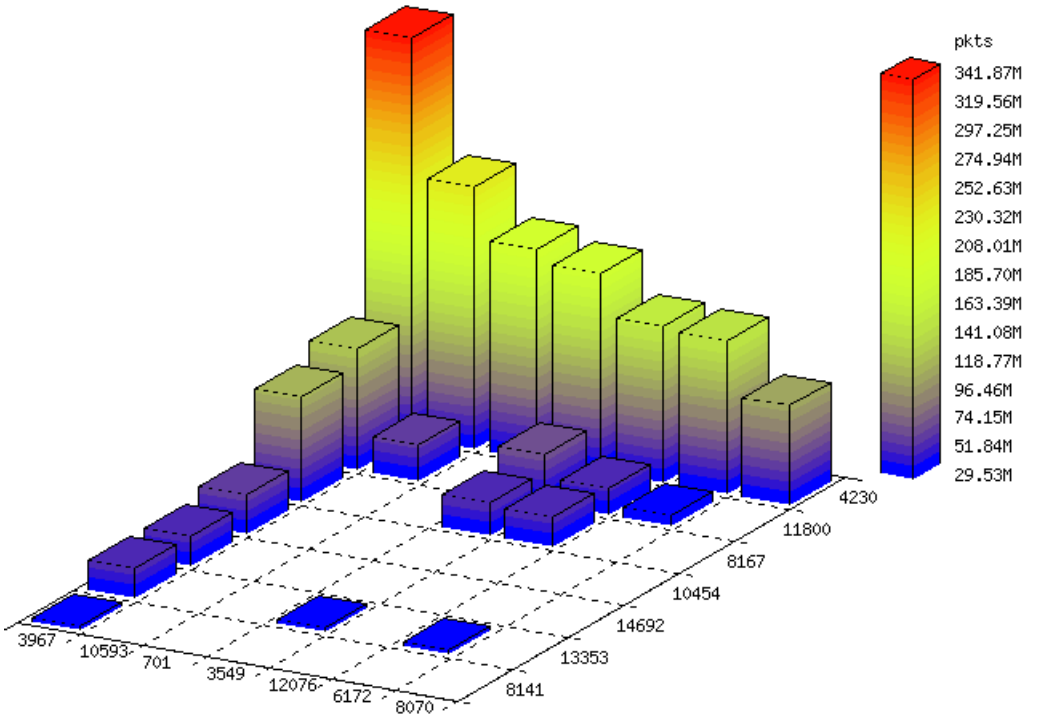


Figura 8.71: Matriz de tráfego por ASN no período de 1 dia

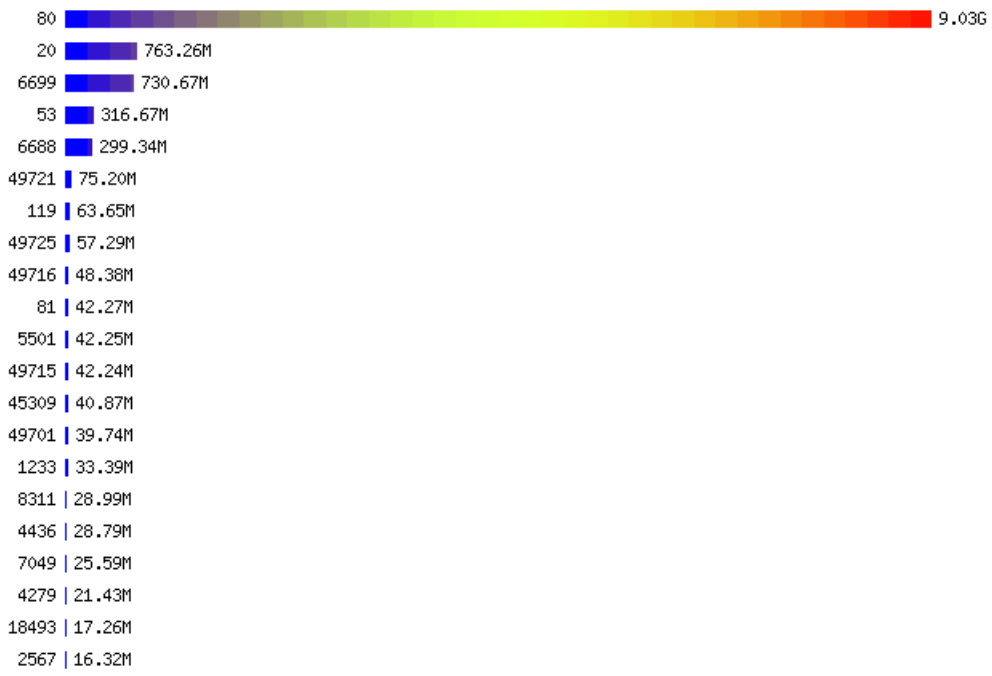


Figura 8.72: Distribuição do tráfego por porta de origem em bytes

na tabela se referem aos ASNs dos vizinhos da Embratel (configuração de `peer-as` do Netflow). Apenas os tráfegos entre ASNs que ultrapassaram a quantidade transferida de 1Mbytes foram mostrados.

Matriz por prefixo de rede

A tabela 8.16 mostra a matriz de tráfego entre prefixo de rede origem e destino no período de 15:52:27 até 15:59:27 do dia 06/07/2000. Apenas os tráfegos entre prefixos de rede que ultrapassaram a quantidade transferida de 6Mbytes foram mostrados.

Matriz por portas TCP

A tabela 8.17 mostra a matriz de tráfego entre porta TCP origem e destino no período de 18:25:13 até 18:25:25 do dia 03/06/2000. Apenas os tráfegos que ultrapassaram a quantidade transferida de 10Kbytes foram mostrados.

O gráfico da figura 8.72 ilustra a distribuição da quantidade de bytes transferidos por porta TCP origem durante o dia 13/07/2000.

O gráfico da figura 8.73 ilustra a distribuição da quantidade de bytes transferidos por porta TCP destino durante o dia 13/07/2000.

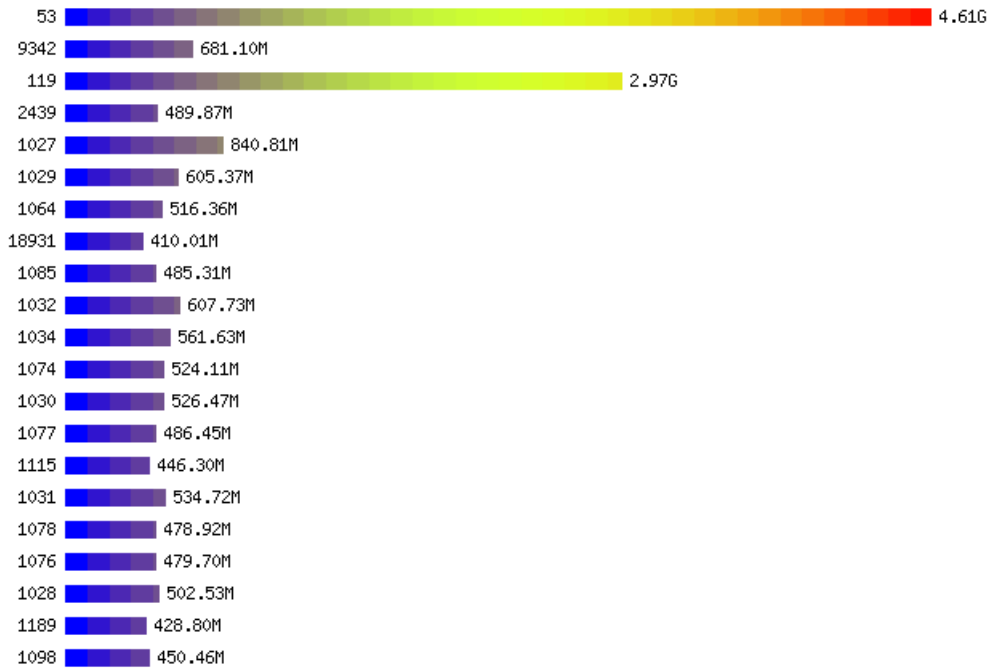


Figura 8.73: Distribuição do tráfego por porta de destino em bytes

Distribuição do tráfego por protocolo

A tabela 8.18 mostra a ditribuição de tráfego dos protocolos IP observados no período de 18:25:25 até 18:30:29 do dia 05/06/2000.

Tipo de serviço do pacote IP

A tabela 8.19 mostra a distribuição do valor do campo ToS do pacote IP encontrado no período de 18:25:25 até 18:30:29 do dia 03/03/2000.

8.4.3 Comentários e conclusões

Matrizes de tráfego por ASN

O processo de escolha do número do AS em relação ao fluxo que será exportado pelo Netflow é baseado na tabela de roteamento BGP (versão 4). Por esses motivos alguns erros podem exitir, sendo assim muito importante o entendimento de como é feito a escolha do ASN pelo processo Netflow, principalmente para evitar interpretações erradas.

Na tabela 8.14 os campos de ASN que estão com "zero" indicam que o processo Netflow não conseguiu achar na tabela de roteamento o ASN do

IP origem do fluxo. Para que não ocorram estes "zeros" na exportação de fluxos é importante que o roteador onde o Netflow está configurado possua uma tabela de roteamento BGP completa (*full routing table*).

Como o roteamento IP na Internet é baseado no IP destino, a coleta de tráfego por ASN origem pelo método do Netflow pode estar errada principalmente quando existe roteamento assimétrico. Como exemplo a tabela 8.14 mostra em sua segunda linha o tráfego proveniente da UUNET (AS701) com destino a TeleBahia (AS7738). Neste caso temos certeza que o tráfego de destino está realmente indo para a TeleBahia pois o processo Netflow procurou na tabela de roteamento a entrada do fluxo IP e econtrou o AS7738 como destino (o mesmo procedimento é feito pelo processo de roteamento para escolha do destino). Porém a origem do tráfego não obrigatóriamente pode ter vindo pela UUNET, ou por motivos de IP *spoofing* ou por configuração de roteamento assimétrico na descida do tráfego.

A questão da assimetricidade de tráfego é apenas relevante nos casos onde a exportação dos fluxos foi feita no modo `peer-as`. Nos casos de configuração com *origin-as* apenas o IP *spoofing* causaria um erro.

O ideal para evitar erros causados por IP *spoofing* é que a coleta dos fluxos de informação possam ser confirmados com um retorno do destino, ou seja, que exista basicamente uma conversação, não apenas um fluxo de mão única.

As tabelas 8.14 e 8.15 mostram a matriz de tráfego por ASN origem e destino na configuração do Netflow como `peer-as`, nos períodos de 1 hora e 1 minuto respectivamente. Estes fluxos foram retirados de enlaces de entrada da UUNET com a Embratel. A quantidade de dados provenientes de outros ASNs além da UUNET, como Cable and Wireless e Sprint, é causado pela característica de tráfego assimétrico feito através de políticas de roteamento com os vizinhos na Embratel.

O gráfico da figura 8.70 ilustra o tráfego de dados no mesmo enlace da UUNET em outro período de 1 hora.

O gráfico da figura 8.71 possue duas características diferentes em relação ao da figura 8.70. A primeira está em relação aos valores das alturas do gráfico, que na figura 8.71 representa a quantidade de pacotes transmitida em 1 dia. A segunda é que os ASNs mostrados a esquerda são os ASNs de origem do tráfego que são obtidos na configuração de `origin-as` do Netflow.

Outra característica importante é que a figura 8.71 tem menor característica de tráfego de trânsito em relação ao da figura 8.70. Para se

observar issa característica basta olhar para as barras que estão na linha do AS4230. Quanto menor for a soma destas barras em relação a soma as outras linhas, maior é a caracterísica de trânsito do tráfego pelo Backbone.

Com as informações de matrizes de tráfego de ASNs é possível a criação de uma política de roteamento que atenda melhor aos interesses de trânsito.

Muitas vezes a distribuição do tráfego de AS destino é mais interessante que as matrizes de tráfego de ASs origem e destino. Esta distribuição pode ser útil para avaliar a porcentagem de tráfego que está como trânsito no Backbone.

Matrizes de tráfego por prefixo de rede

Um detalhamento melhor nos fluxos de dados pode ser conseguido através das matrizes de tráfego por prefixos de rede.

Assim como no processo de escolha do ASN, o Netflow utiliza a tabela de roteamento para identificar as máscaras de rede dos prefixos a serem exportados. Logo podem existir erros em relação ao tamanho do prefixo original, que algumas vezes pode ser alterado nos anúncios BGP por motivos de sumarização de rotas.

Com a matriz de tráfego por prefixo de rede é possível avaliar o quanto de tráfego está indo para cada "região" do Backbone. Uma composição entre ASN de origem com "região" de destino do Backbone pode ser interessante para avaliar o interesse de tráfego de cada região.

A matriz de tráfego por prefixo pode ser interessante também para o mapeamento das redes por países. É possível relacionar os prefixos de forma aproximada para cada país, isto pode ser conseguido utilizando as tabelas de registro de prefixos de rede no ARIN, RIPE e APNIC.

Como exemplo das possibilidade de descrição do tráfego por região podemos comparar a segunda linha com a vigésima linha da tabela 8.16. A segunda linha da tabela mostra o tráfego da UUNET (`198.6.0.0/16` AS701) para o centro de roteamento da Embratel do Rio de Janeiro (`200.255.253.224/27`). A vigésima linha mostra o tráfego da UUNET (`212.249.0.0/16` AS702) para o centro de roteamento da Embratel de Belo Horizonte (`200.251.134.0/24`).

Distribuição do tráfego por porta TCP

A matriz de tráfego por porta TCP origem e destino não é muito interessante, uma vez que um dos lados funciona como cliente. A porta cliente é escolhida

"aleatoriamente" no processo inicial da conexão TCP apartir do número 1024. Com isso na matriz de tráfego por porta muito fluxos de dados apenas serão de uma porta "servidora" para uma porta cliente, como a maioria das linhas na tabela 8.17.

A distribuição do tráfego por porta origem pode ser mais interessante conforme ilustrado no gráfico da figura 8.72. A figura 8.73 mostra a distribuição das portas destino encontradas nos fluxos.

Na figura 8.72 podemos ver que o maior serviço acessado no exterior em termos de bytes é o WWW (porta 80) em segundo o Napster (portas 6699 e 6688) e em terceiro o FTP (porta 20). Ainda podemos ver o tráfego de tranferência de DNS (porta 53) e de NEWS (porta 119).

Distribuição do tráfego por protocolo

A tabela 8.18 mostra a quantidade de pacotes e de bytes por protocolo. Um acompanhamento destes protocolos ao longo do tempo pode ser interessante para avaliar o comportamento de outras aplicações. Como o do IP encapsulado sobre o IP no caso do MBone.

Tipo de serviço do pacote IP

A distribuição do tráfego por tipo de serviço pode ajudar em na implementação de uma política de diferenciação do tráfego como no DiffServ [12]. Um exemplo de distribuição do campo ToS nos pacotes IP pode ser visto na tabela 8.19 .

Tabela 8.12: Distribuição das Flags TCP

Flags	pkts	bytes	% de pkts	% de bytes
A	97484016	53289942892	67.955	72.465
AP	31720825	19322126463	22.112	26.275
S	4229046	205530932	2.948	0.279
AF	4213291	201419307	2.937	0.274
AS	2975206	137293806	2.074	0.187
R	2105965	84506649	1.468	0.115
AFP	517338	286276036	0.361	0.389
AR	203712	10430623	0.142	0.014
ARP	2589	805163	0.002	0.001
	123	10491	0.000	0.000
AUP	117	7546	0.000	0.000
ARF	56	2240	0.000	0.000
ARFP	50	5527	0.000	0.000
AU	27	7505	0.000	0.000
F	24	988	0.000	0.000
AFU	17	680	0.000	0.000
SRU	16	640	0.000	0.000
FUP	15	2716	0.000	0.000
RF	15	2225	0.000	0.000
ARU	14	3480	0.000	0.000
RFUP	14	560	0.000	0.000
ASUP	13	520	0.000	0.000
SF	12	4860	0.000	0.000
ASRU	12	1461	0.000	0.000
SRP	12	776	0.000	0.000
ASF	12	488	0.000	0.000
ASRUP	11	780	0.000	0.000
U	10	1993	0.000	0.000
SUP	10	1082	0.000	0.000
ARFUP	10	1022	0.000	0.000
SRFU	10	724	0.000	0.000
SFUP	10	675	0.000	0.000
ARFU	10	404	0.000	0.000
RFU	10	400	0.000	0.000
SP	10	400	0.000	0.000
SRF	9	1050	0.000	0.000
SFP	9	960	0.000	0.000
ASU	9	952	0.000	0.000
ASRFU	9	548	0.000	0.000
SRFP	9	380	0.000	0.000
ASRFP	9	360	0.000	0.000
ASRFUP	9	360	0.000	0.000
ASR	9	360	0.000	0.000
RFP	9	360	0.000	0.000
ASP	8	2081	0.000	0.000
SFU	8	1808	0.000	0.000
ASFUP	8	1215	0.000	0.000
RU	8	753	0.000	0.000
SRUP	8	673	0.000	0.000
P	8	320	0.000	0.000
AFUP	7	1257	0.000	0.000
SRFUP	7	775	0.000	0.000
ARUP	7	280	0.000	0.000
ASRF	7	280	0.000	0.000
FP	7	280	0.000	0.000
ASRP	6	252	0.000	0.000
RP	6	240	0.000	0.000
ASFU	5	200	0.000	0.000
FU	5	200	0.000	0.000
RUP	5	200	0.000	0.000
ASFP	4	640	0.000	0.000
SR	8	328	0.000	0.000
SU	7	870	0.000	0.000
UP	4	870	0.000	0.000
TOTAL	143453025	73538456850	100	100

Tabela 8.13: Lista com ASNs e as instituições responsáveis (retirado do ARIN/RIPE)

ASN	Instituição responsável
701	UUNET Technologies, Inc. (ASN-ALTERNET)
2715	Fundacao de Amparo a Pesquisa do Estado de Sao Paulo (ASN-REDERIO)
3549	Global Crossing (ASN-GBLX)
3561	Cable & Wireless USA (ASN-CWUSA)
3967	Exodus Communication (ASN-EXODUS)
4005	Sprint International (ASN-GLOBAL-SPLK-ASNBLOCK)
4230	Embratel (ASN-EMBRATEL-BR)
5511	France Telecom (OPENTRANSIT)
5772	Unisys Brasil Ltda. (ASN-UNINET-BR-RJ)
6125	RNP / Centro Regional do Distrito Federal (ASN-RNP-POP-DF)
6172	@Home Network (ASN-HOME-NET-1)
7365	CentroIn Internet Provider (ASN-CENTROIN)
7414	PontoCom Consultoria e Tecnologia Internet (ASN-PONTOCOM)
7738	Telecomunicacoes da Bahia S.A. (ASN-TELEBAHIA)
8070	Microsoft Corporation (ASNBLK-MICROSOFT-AS-BLOCK)
8141	TBA INFORMATICA LTDA. (ASN-TBAINFORMATICA)
8167	TELESC - Telecomunicacoes de Santa Catarina SA (ASN-TELESC)
8657	Companhia Portuguesa Radio Marconi (CPRM)
10454	Horizontes Internet (ASN-HORIZONTES)
10593	America Online (ASN-AOL-DTC2)
10670	PRODEMGE Cia. Processamento Dados MG (ASN-PRODEMGE-AS)
10688	ISM Automacao Ltda. (ASN-ISMNET)
11156	Ponto de Presenca da RNPesquisa no Ceara (ASN-POP-CE)
11706	Mago Informacoes e Dados Ltda. (ASN-BHNET)
11800	Domain Acesso e Servicos Internet Ltda. (ASN-DOMAIN-ASN)
11921	Secrelnet Informatica LTDA (ASN-SECREL)
10938	ITEP (ASN-POP-PE)
10954	SERPRO - Empresa do Ministério da Fazenda (ASN-SERPRO)
12076	Hotmail Corporation (ASN-HOTMAIL-AS)
12135	PROINTERNET DO BRASIL (ASN-PROINTERNET-DO-BRASIL)
12136	e-net Teleinformatica Ltda (ASN-E-NET-BR)
13353	Netstream Telecom Ltda. (ASN-NETSTREAM)
14692	Sao OnLine SC LTDA (ASN-PSINETSTI)

Tabela 8.14: Matriz de tráfego por ASN no período de 1 hora

ASN Origem	ASN Destino	Pacotes	pps	Bytes	bps
701	4230	5169207	1435.49	2922726266	6.49314e+06
701	7738	3045511	845.74	506463719	1.12516e+06
3561	4230	964787	267.922	455480440	1.0119e+06
0	7738	10530062	2924.2	421229673	935806
701	13353	812479	225.626	323799306	719354
701	11800	546725	151.826	302658709	672388
701	8167	311971	86.6345	194777762	432719
701	10454	203322	56.4626	113601249	252377
4005	4230	192150	53.3602	84903959	188623
701	11706	147641	41	69561920	154539
3561	7738	906358	251.696	67805099	150636
0	13353	1625548	451.416	65022144	144454
701	14692	96960	26.9259	55916299	124224
701	8141	97598	27.103	54598910	121297
701	11921	60791	16.8817	42867447	95234.5
701	12136	65182	18.1011	34455139	76545.7
3561	11800	73551	20.4252	32073264	71254.1
5511	4230	69674	19.3485	30379772	67491.9
0	4230	710535	197.316	28608306	63556.4
3561	8167	39917	11.085	25262535	56123.4
701	2715	30239	8.39739	20953337	46550
3561	13353	126026	34.9975	19938720	44296
4005	7738	157307	43.6843	15278946	33943.8
701	5772	25401	7.05387	13350612	29659.8
3561	10454	24624	6.8381	11993985	26645.9
701	10938	18623	5.17162	9591354	21308.2
701	7365	16987	4.7173	9205680	20451.4
701	12135	12724	3.53346	8789225	19526.2
8657	7738	13416	3.72563	8641872	19198.8
701	10670	13611	3.77978	8250040	18328.3
701	11156	14809	4.11247	8023450	17824.9
3561	8141	15185	4.21688	7999749	17772.3
8657	13353	15642	4.34379	7993637	17758.7
701	10688	13660	3.79339	7553246	16780.3
3561	14692	12159	3.37656	6726670	14944
701	10954	6847	1.90142	6688284	14858.7
701	7414	12810	3.55735	6576363	14610.1
701	6125	11439	3.17662	6209322	13794.7
4005	11800	10764	2.98917	5706695	12678

Tabela 8.15: Matriz de tráfego por ASN no período de 1 minuto

ASN Origem	ASN Destino	Pacotes	pps	Bytes	bps
701	4230	149062	2570.03	98142999	1.3537e+07
3561	4230	22596	389.586	12574492	1.73441e+06
701	7738	48103	829.362	8380090	1.15587e+06
0	7738	162945	2809.4	6517800	899007
701	11800	7798	134.448	4283241	590792
701	8167	5416	93.3793	3039760	419277
701	13353	14268	246	3009164	415057
701	2715	3081	53.1207	2417467	333444
4005	4230	4312	74.3448	2071978	285790
0	13353	43275	746.121	1731076	238769
701	8141	2389	41.1897	1666363	229843
701	11921	1725	29.7414	1542990	212826
5511	4230	2349	40.5	1415866	195292
701	10954	1714	29.5517	1332877	183845
701	14692	2102	36.2414	1186790	163695
701	10670	1815	31.2931	1177229	162376
701	10454	2083	35.9138	1170212	161409
3561	7738	13098	225.828	1058838	146047
701	6125	1272	21.931	1055189	145543

Tabela 8.16: Matriz de tráfego por prefixo de rede no período de 5 minutos

Prefixo de origem	Prefixo de destino	Pacotes	Bytes
198.112.0.0/14	200.244.42.0/24	45650	51886402
198.6.0.0/16	200.255.253.224/27	28556	39069648
206.161.224.0/23	200.244.192.0/18	19119	26856721
208.128.0.0/11	200.193.192.0/18	16634	24514954
64.70.0.0/19	200.244.60.0/25	14307	20093890
216.81.0.0/19	200.188.32.0/19	15407	17439836
216.35.128.0/19	129.222.64.0/19	11098	16616666
128.227.0.0/16	200.188.16.0/20	9452	13966849
140.192.0.0/16	200.252.233.128/26	9284	10558092
216.2.8.0/23	200.255.210.128/25	7789	10221319
24.0.0.0/12	200.255.111.0/24	16178	9076184
210.172.32.0/19	200.214.96.0/24	15478	8911552
192.233.80.0/24	200.244.74.0/25	7798	8512265
205.188.128.0/17	200.188.32.0/19	21996	8462419
24.229.0.0/18	200.255.242.0/24	7265	8355366
192.233.80.0/24	200.255.202.0/23	5562	7841456
193.224.0.0/15	200.214.7.128/25	5148	7542344
207.211.0.0/16	200.214.6.0/24	4973	7427767
198.5.128.0/19	200.253.128.0/17	6280	7360886
212.249.0.0/16	200.251.134.0/24	12875	7318144
194.133.128.0/22	200.255.150.0/24	14777	7248881
216.35.0.0/19	200.214.68.128/25	6845	7193397
128.11.0.0/16	200.188.16.0/20	12271	7091619
148.184.0.0/16	200.255.202.0/23	4974	7046088
4.0.0.0/8	200.255.75.0/25	12316	6793885
146.186.0.0/16	200.244.44.0/23	4520	6777100
206.204.0.0/16	200.251.192.0/20	5072	6591348
216.246.0.0/18	200.255.210.128/25	4372	6528726
193.12.0.0/14	200.244.74.0/25	4449	6231275
205.188.128.0/17	200.193.192.0/18	10340	6139061
216.240.128.0/20	200.188.32.0/19	4365	6121106
203.0.0.0/10	200.255.116.0/24	4054	6076484
200.224.192.0/18	200.188.16.0/20	25618	6026000

Tabela 8.17: Matriz de tráfego por porta TCP

Porta origem	Porta destino	Pacotes	pps	Bytes	bps
80	61021	167	13.9167	241464	160976
80	1064	169	14.0833	95709	63806
80	1394	31	2.58333	45044	30029.3
80	1455	78	6.5	43233	28822
80	1249	29	2.41667	40584	27056
443	1136	48	4	35717	23811.3
80	1337	28	2.33333	35696	23797.3
80	1077	61	5.08333	32640	21760
80	1166	48	4	24809	16539.3
80	1078	47	3.91667	23661	15774
20	1036	41	3.41667	23616	15744
80	1397	13	1.08333	16999	11332.7
80	1072	34	2.83333	16290	10860
80	1225	35	2.91667	16125	10750
80	64941	11	0.916667	15514	10342.7
80	64938	14	1.16667	15072	10048
80	8800	23	1.91667	14657	9771.33
80	2315	19	1.58333	13481	8987.33
80	1202	27	2.25	12487	8324.67
80	1062	27	2.25	12227	8151.33
80	1274	10	0.833333	11631	7754
80	1330	22	1.83333	10919	7279.33
80	1151	15	1.25	10874	7249.33
80	1974	10	0.833333	10592	7061.33

Tabela 8.18: Distribuição do tráfego por protocolo em 5 minutos

Protocolo	Pacotes	pps	Bytes	bps
tcp	60821	200	32869973	864999
udp	1518	4	198021	5211
icmp	105	0	8447	222
ipencap	12	0	1695	44

Tabela 8.19: Distribuição do tráfego por ToS (*Type of Service*)

ToS	Pacotes	pps	Bytes	bps
0	55288	181	32139259	845769
32	165	0	90172	2372
200	72	0	63916	1682
40	30	0	20996	552
16	45	0	16501	434
192	71	0	8996	236
8	21	0	6256	164
232	29	0	4648	122
104	10	0	2430	63
136	10	0	2257	59
168	4	0	503	13
160	1	0	40	1
208	1	0	40	1

Capítulo 9

Conclusões e Continuações deste Trabalho

Os objetivos principais deste trabalho foram alcançados, que eram coletar informações a respeito do assunto, procurar trabalhos já existentes, organizar idéias, criar definições e aplicar exemplos de coleta e análise.

A coleta de dados foi o ponto crítico do trabalho, pois além de demandar a aquisição de equipamentos implica algumas vezes na alteração da topologia ou na configuração da rede. Os tipos de coletor de dados que necessitam a inclusão de *splitters* (passivos) não são adequados pois interrompem a operação da rede, o que torna o coletor uma peça não versátil, muitas vezes dificultando a sua ligação em um enlace específico da rede. As coletas de dados utilizando SNMP, ICMP, espelhamento de switches e exportação de fluxos são as mais versáteis, pois podem ser configuradas sem a interrupção da rede. No caso das coletas feitas via exportação de fluxo é necessária uma configuração planejada nos roteadores da rede para evitar que duas exportações distintas enviem as "mesmas" informações.

O procedimento de análise dos dados muitas vezes é feito através da visualização de gráficos, porém o ideal é modelar matematicamente alguns parâmetros dos dados no tempo para que seja possível fazer comparações sem a necessidade de "grafar" os dados. Os intervalos de consolidação dos dados nos gráficos apresentados nos exemplos muitas vezes foram escolhidos por tentativa e erro, para serem melhor visualizados.

Outra conclusão a respeito deste trabalho é que não existe um regra geral que possa ser utilizada em um Backbone IP. São muitas as variáveis de medida (15 apresentadas aqui), sendo algumas delas mais abrangentes do que se parece (ex: tabelas de rotemanto do exemplo 1 do capítulo 8). As

necessidades devem ser levantadas, e algumas delas só aparecem quando a coleta de dados começa a apresentar resultados. Logo o universo de coleta de dados e variáveis de medida pode crescer quando os resultados de desempenho e análise do tráfego forem não apenas incompletos mas também motivantes.

Um dos problemas encontrados é que muitas vezes é mais prático e rápido escolher o universo das variáveis de medida a partir de uma coleta, pois muitas vezes a variável de medida pode ser coletada de várias maneiras, e algumas vezes esse coletor pode fornecer outras variáveis de medida além daquelas para as quais ele foi primeiramente direcionado.

A utilização de *softwares* de domínio público (*opensource*) possibilitou o estudo mais aprofundado das técnicas de coleta, da análise e da apresentação dos resultados. Muito se pode ainda fazer para contribuir para o desenvolvimento de novas ferramentas de coleta e análise, esta tese é apenas uma motivação e uma referência para este desenvolvimento.

A continuação deste trabalho seria a criação de uma infra-estrutura de coleta de dados, análise e apresentação em um Backbone IP acadêmico. Assim como no movimento *opensource* este futuro trabalho tem a intenção de deixar públicas informações de tráfego e de desempenho, assim como divulgar as ferramentas utilizadas e os *softwares* utilizados.

Uma sugestão seria a criação de uma organização que mantenha esta infraestrutura, para que as instituições que contribuem financeiramente possam usufruir de ferramentas e técnicas desenvolvidas na organização, aproximando assim o desenvolvimento tecnológico da aplicação direta. Abrindo-se uma porta para a comunidade contribuir para o seu desenvolvimento, assim como o exemplo de iniciativas como o SourceForge [60]. A continuação seria criar um "portal" a respeito do assunto para possíveis referências para comunidade Internet.

Apêndice A

Contribuições da Comunidade

O estudo prévio dos trabalhos realizados pela comunidade tem fundamental importância para a atualização e para a avaliação da metodologia proposta neste trabalho.

Existem centenas de trabalhos relacionados com medida de desempenho e caracterização de tráfego. Neste apêndice serão ilustrados os principais trabalhos e organizações que contribuem de alguma forma para o desenvolvimento do assunto abordado nos próximos capítulos.

A.1 O IETF

Ligados ao IETF (*Internet Engineering Task Force* [14]) existem uma série de grupos de trabalho que mais se aproximam nas questões de performance e avaliação de desempenho do tráfego na Internet: IPPM, RTFM, BMWG, TEWG, RMONMIB [10] e INTSERV [11]. Dentro destes grupos descreveremos os 4 mais importantes:

O IPPM

A principal função do grupo IPPM (*Internet Protocol Performance Metrics Working Group* [6]) é desenvolver uma série de padrões, métricas e recomendações que possam ser utilizadas para avaliação de qualidade, performance e confiabilidade dos métodos de entrega de dados da Internet. Estas métricas e serão desenvolvidas de tal forma que possam ser utilizadas tanto pelos operadores de Backbones Internet como pelos usuários finais. A importância da definição destas métricas é evitar que medidas de performance sejam feitas de forma diferente.

O IPPM tem as seguintes *Internet-Drafts* em discussão:

- `draft-ietf-ippm-ipdv-05.txt` (*Instantaneous Packet Delay Variation Metric for IPPM*);
- `draft-ietf-ippm-loss-pattern-03.txt` (*One-way Loss Pattern Sample Metrics*);
- `draft-ietf-ippm-npmps-02.txt` (*Network performance measurement for periodic streams*).

O IPPM tem as seguintes RFCs publicadas:

- `RFC2330`: *Framework for IP Performance Metrics* [102];
- `RFC2678`: *IPPM Metrics for Measuring Connectivity* [**?**];
- `RFC2679`: *A One-way Delay Metric for IPPM* [107];
- `RFC2680`: *A One-way Packet Loss Metric for IPPM* [108];
- `RFC2681`: *A Round-trip Delay Metric for IPPM* [109];

O RTFM

O grupo de trabalho RTFM (*Realtime Traffic Flow Measurement* [7]) tem como objetivo estabelecer padrões e recomendações para medidas de tráfego do fluxo de dados na Internet. Este grupo trabalha, por exemplo, com os seguintes assuntos: Segurança dos equipamentos de medida e dos dados produzidos por estes equipamentos, procura meios de simplificar a definição de fluxo para medições de tráfego com uma grande quantidade de fluxos, melhorar o controle de acesso ao medidor e implementar um *hardware* mais eficiente.

O RTFM tem as seguintes RFCs publicadas:

- `RFC2123`: *Traffic Flow Measurement: Experiences with NeTraMet* [99];
- `RFC2720`: *Traffic Flow Measurement: Meter MIB* [110];
- `RFC2721`: *RTFM: Applicability Statement* [111];
- `RFC2722`: *Traffic Flow Measurement: Architecture* [112];
- `RFC2723`: *SRL: A Language for Describing Traffic Flows and Specifying Actions for Flow Groups* [113];
- `RFC2724`: *RTFM Working Group - New Attributes for Traffic Flow Measurement* [114].

O BMWG

O grupo de trabalho BMWG (*Benchmarking Methodology* [9]) tem o objetivo gerar padrões e recomendações para medidas de performance de dispositivos em várias tecnologias ligadas a Internet, como rede locais, redes de longa distância, rotadores, FireWalls, etc...

O BMWG tem as seguintes *Internet-Drafts* em discussão:

- `draft-ietf-bmwg-fr-term-04.txt` (*Terminology for Frame Relay Benchmarking*);
- `draft-ietf-bmwg-mcastm-04.txt` (*Methodology for IP Multicast Benchmarking*);
- `draft-ietf-bmwg-atm-method-02.txt` (*Methodology for ATM Benchmarking*);
- `draft-ietf-bmwg-atm-term-abr-02.txt` (*Terminology for ATM ABR Benchmarking*);
- `draft-ietf-bmwg-fib-term-00.txt` (*Terminology for Forwarding Information Base (FIB) based Router Performance Benchmarking*);
- `draft-ietf-bmwg-rtr-framework-00.txt` (*Framework for Router Benchmarking*);
- `draft-ietf-bmwg-firewall-00.txt` (*Benchmarking Methodology for Firewalls*).

O BMWG tem as seguintes RFCs publicadas:

- `RFC1242`: *Benchmarking Terminology for Network Interconnection Devices* [95];
- `RFC2285`: *Benchmarking Terminology for LAN Switching Devices* [101];
- `RFC2432`: *Terminology for IP Multicast Benchmarking* [103];
- `RFC2544`: *Benchmarking Methodology for Network Interconnect Devices* [104];
- `RFC2647`: *Benchmarking Terminology for Firewall Performance* [105];
- `RFC2761`: *Terminology for ATM Benchmarking* [115];
- `RFC2889`: *Benchmarking Methodology for LAN Switching Devices* [116];

O TEWG

O grupo de trabalho TEWG (*Internet Traffic Engineering* [8]) tem como objetivo definir, desenvolver, especificar, recomendar técnicas e mecanismos para a engenharia de tráfego na Internet. Este grupo participa serve também para se discutir possíveis melhoras nos protocolos da Internet para melhorar a engenharia de tráfego.

Engenharia de tráfego Internet tem o seu escopo relacionado com a engenharia de rede voltada para a melhora da performance dos aspectos voltados para a operação da rede Internet. Isso envolve como aplicar a tecnologia existente, princípios de medida, modelagem, caracterização e controle do tráfego Internet, melhorando a utilização dos recursos, o planejamento e escalabilidade da rede.

Este grupo interage com as seguintes áreas do IETF: *Routing Area* (ex: MPLS, OSPF, etc), *Transport Area* (ex: IPPM, RTFM, etc), *Management Area* (ex: RMOMMIB, POLICY, etc).

Este grupo ainda não tem uma RFC publicada, apenas uma *Iternet-Draft* para discussão: `draft-ietf-tewg-framework-02.txt` (*A Framework for Internet Traffic Engineering*).

A.2 O RIPE

O RIPE (*Réseaux IP Européens*) [77] é uma organização européia contruida para disponibilizar coordenação técnica e administrativa das redes IP operadas na europa. O RIPE funciona como um órgão regulador gerador de documentos de padronização e recomendação.

Seu maior esforço para medidas de performance de redes IP é o grupo de trabalho Test-Traffic [39] (RIPE *Working Group*) O projeto Test-Traffic tem o objetivo de criar uma estrutura de coleta de dados através de uma série de máquinas localizadas em pontos estratégicos. As medidas que se tem interesse neste projeto são: Atraso, variações do atraso, Vazão (Throughput), caminho de roteamento e tabelas de roteamento.

Irá organizar o PAM2001 (*Passive & Active Measurement Workshop* [50])

A.3 O TERENA

O TERENA (*Trans-European Research and Education Networking Association*) [78] foi criado em 1994 para promover difundir a pesquisa, a

informação e a educação no que tange as tecnologias de telecomunicações. Mais de 40 países da Europa participam do TERENA como membros, e algumas instituições também como a IBM, a Cisco, a Teleglobe e o CERN.

A contribuição do TERENA para medidas de performance e análise de tráfego estão na última força de trabalho TF-ETM (*European Traffic Measurements Task Force*) [40] criado em 1997 e substituido pela TF-TANT (*Testing Advanced Networking Technologies Task Force*) [41] em 1999. A TF-TANT é um trabalho conjunto entre o TERENA e o DANTE [79]. O DANTE tem como objetivo disponibilizar infraestrutura de conexão entre os órgãos de pesquisa na Europa.

Dentro da força de trabalho TF-TANT podemos citar os seguintes trabalhos voltados para o foco de medida de performance e análise de tráfego:

- *Differentiated Services* [42];
- *QoS Monitoring* [43];
- *Flow-based Monitoring Analysis* [44];
- *Route Monitoring* [45];

A.4 O MIDS

O MIDS (*Matrix Information and Directory Services*) [80] foi criado em 1990 pela empresa Matrix Internet Quality. Esta empresa apresenta um bom exemplo de uma organização comercial que realiza medidas de performance na Internet, como atraso, perda de pacotes e disponibilidade. Um bom exemplo de resultados pode ser visto no endereço: `http://ratings.miq.net`

A.5 O NLANR

O NLANR (*National Laboratory for Applied Network Research*) [71] foi criada em 1995 pela NSF (*National Sciense Foundation* [69]) como um laboratório de pesquisa em redes de computadores. Seu principal objetivo é prover suporte técnico para engenharia, análise de tráfego e desenvolvimento de ferramentas para as redes da NSF como a vBNS (*very high performance Backbone Network Service* [70]).

O NLANR é dividido em três grupos:

DAST

O grupo DAST (*Distributed Applications Support Team* [72]) é gerenciado pela universidade de Illinois (UIUC EUA) e está focado no suporte de usuários da rede e também a aplicações de alta performance para a redes de alta velocidade da NSF. Algumas ferramentas e projetos suportados por este grupo são: *Iperf* [58], *Myproxy*, *Netlog*, *Terabyte Challenge*, *Access Grid* e *TCP Socket Buffer Tuning*.

NCNE

O grupo NCNE (*National Center for Network Engineering*) é gerenciado pela Universidade Carnegie Mellon [67] e pelo Centro de Supercomputação de Pittsburgh [68], sua função é fornecer suporte e informação a tudo que se refere a interconexão das redes dos campus ao Backbone de alta velocidade, assim como serviços de engenharia. Alguns projetos relacionados a este grupo: *NCNE gigaPoP*, NIMI (*National Internet Measurement Infrastructure* [30]), PITAC (*Performance Measurement Project* [31]) e *NLANR On-Site* (voltado para treinamento).

MOAT

O grupo MOAT (*Measurement and Operations Analysis Team*) é gerenciado pela Universidade de San Diego [66] e está focado nos procedimentos de medida e análise dos fluxos de dados que passam nas redes de alta velocidade da NSF. Alguns projetos relacionados a este grupo: PMA (*Passive Measurement and Analysis* [32]), NAI (*Network Analysis Infrastructure* [33]) e a publicação do *Network Analisys Times* [34].

A.6 A Universidade de Berkeley

Ligados a Universidade de Berkeley (UCB EUA) podemos citar 2 principais grupos:

ACIRI

O ACIRI (*AT&T Center for Internet Research* [73]) está localizado no Instituto de Ciência da Computação na universidade de Berkeley. Foi criado em 1998 em conjunto com a AT&T. Os objetivos do ACIRI são: Participar ativamente dos grupos de pesquisa e padronização como o IRTF,

o SIGCOMM e o IETF; Funcionar como ponte entre os interesses da comunidade de pesquisa e os interesses comerciais. Além de uma série de publicações como *Internet-Drafts* e RFCs o ACIRI se destaca pelos seguintes projetos: Bro (*A System for Detecting Network Intruders in Real-Time*), TBIT (*TCP Behavior Indentification Tool* [35]), NS (*Network Simulator* e RED (*Random Early Detection* [36]).

LBNL

O grupo de pesquisa em redes de computadores NRG (*Network Research Group* [74]) localizado no LBNL (*Lawrence Berkeley National Laboratory*) teve como parte integrante pesquisadores como Vern Paxson e Van Jacobson. Atualmente hospeda uma série de trabalhos e projetos: libpcap, tcpdump, traceroute e ITA (*The Internet Traffic Archive* [38]).

A.7 A Merit

A rede Merit [75] é uma instituição sem fins lucrativos criada para promover interconeão de computadores no estado de Michigan (EUA). Seus principais esforços no que tange caracterização de tráfego e análise de performance são: Hospedar e organizar o NANOG (*North American Network Operators' Group* [76]) e IPMA (*Internet Performance Measurement and Analysis* [48]).

A.8 O CAIDA

O CAIDA (*Cooperative Association for Internet Data Analysis* [65]) é uma instituição que tem como objetivo pesquisar e promover uma melhor cooperação na engenharia e na manutenção de uma infraestrutura Internet escalável e robusta.

Originalmente o CAIDA era um projeto ligado ao NLANR (*National Laboratory for Applied Network Research* [71]) dentro da universidade de San Diego na Califórnia [66]. Sendo apoiada pelo governo e pela NSF (*National Science Foudation* [69]) a idéia era criar uma instituição que pudesse desenvolver soluções para as diversas comunidades da Internet (privadas, de pesquisa, do governo e usuários), disponibilizando ferramentas e tecnologia para as medidas mais críticas da Internet.

Dentro das principais ferramentas suportadas pelo CAIDA, estão: `Cflowd`, `CoralReef`, `Skitter`, `RRDtool`, `NetGeo` dentre outros.

Alguns dos projetos ligados ao CAIDA: NGI (*Next Generation Internet* [26]), IEC (*Internet Engineering Curriculum Repository* [27] , *Internet Atlas Project* [28] e ISMA (*Internet Statistic and Metrics Analysis*).

A.9 A Universidade de Waikato

A universidade de Waikato [64] está localidada na Nova Zelândia e tem como principal grupo de pesquisa o WAND [46].

Seu principal trabalho é desenvolver um *hardware* especial capaz de coletar dados em redes de alta velocidade, capazes de coletar células ATM ou quadros SDH em velocidades de 155Mbps, 622Mbps e 2.5Gbps [47]. Também foi responsável pela organização do PAM2000 (*Passive & Active Measurement Workshop* [49]).

Apêndice B

Cabeçalho dos principais protocolos da Internet

B.1 IP (*Internet Protocol*)

Abaixo está a descrição dos campos do cabeçalho IP, detalhes podem ser obtidos na RFC791 [92]:

```
\begin{center}
    0                   1                   2                   3
    0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |Version|  IHL  |Type of Service|          Total Length         |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |         Identification        |Flags|      Fragment Offset    |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |  Time to Live |    Protocol   |         Header Checksum       |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                       Source Address                          |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                    Destination Address                        |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                    Options                    |    Padding    |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
\end{center}
```

Na representação acima cada "`-`" representa um bit.

- `Version`: 4 bits, indica a versão do protocolo IP utilizado, atualmente existem a versão 4 e a 6. A versão atualmente mais utilizada é a versão 4;
- `IHL`: 4 bits *Internet Header Length*, indica o tamanho do cabeçalho IP em palavras de 32 bits, o valor mínimo é 5;
- `Type of Service`: 8 bits (conhecido também como TOS), indica o tipo de qualidade de serviço desejado. Atualmente este campo é pouco utilizado;

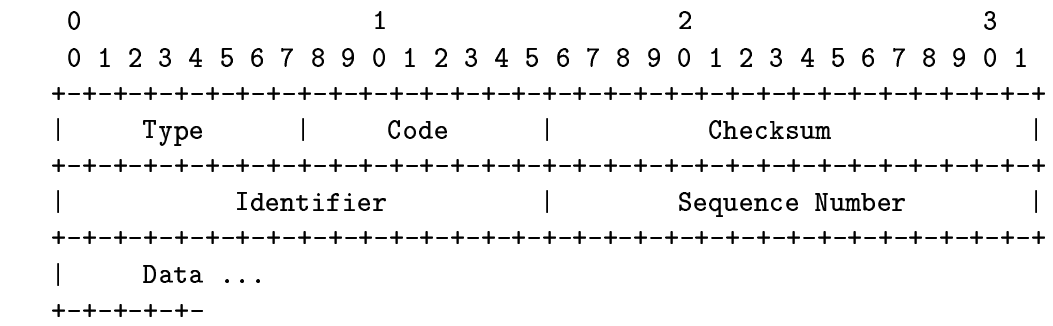
- `Total Length`: 16 bits. Indica o tamanho total do pacote IP em bytes (datagrama IP). O tamanho máximo será de 65535bytes;
- `Identification`: 16 bits. Campo utilizado para identificar um pacote IP. Este campo ajuda na fragmentação e na montagem dos pacotes IP;
- `Flags`: 3 bits. São bits utilizados para controle

```
Bit 0: reserved, must be zero
Bit 1: (DF) 0 = May Fragment,  1 = Don't Fragment.
Bit 2: (MF) 0 = Last Fragment, 1 = More Fragments.
    0   1   2
  +---+---+---+
  |   | D | M |
  | 0 | F | F |
  +---+---+---+
```

- `Fragment Offset`: 13 bits. Este campo indica qual a parte do pacote original que este fragmento pertence. O valor é medido em unidades de 8 bytes, o primeiro fragmento tem *offset* igual a zero;
- `Time to Live`: 8 bits (conhecido também como TTL). Campo utilizado para evitar que um pacote IP fique circulando eternamente na Internet por um problema de roteamento. Este campo é decrementado de 1 a cada *hop.* quando chega a 0 o pacote é descatado;
- `Protocol`: 8 bits. Campo utilizado para indicar qual é o protocolo utilizado no próximo nível. A completa lista pode ser obtida na RFC1700 [97];
- `Header Checksum`: 16 bits. Campo que indica o *checksum* do cabeçalho. Toda vez que o cabeçalho é alterado este campo é recalculado, o que ocorre pelo menos a cada *hop* (TTL diminue de 1);
- `Source Address`: 32 bits. Campo designado para conter o endereço IP de origem do pacote;
- `Destination Address`: 32 bits. Campo designado para conter o endereço IP de destino do pacote;
- `Options`: Tamanho variável. Este campo é utilizado para opções adicionais, que podem ou não estar presentes. Maiores estão descritos na RFC791 [92];
- `Padding`: Tamanho variável. Este campo é utilizado para garantir que o cabeçalho do pacote IP sempre tenha um tamanho multiplo de 32bits.

B.2 ICMP (*Internet Control Message Protocol*)

Echo or Echo Reply Message



IP Fields:

Addresses

The address of the source in an echo message will be the destination of the echo reply message. To form an echo reply message, the source and destination addresses are simply reversed, the type code changed to 0, and the checksum recomputed.

ICMP Fields:

Type

8 for echo message;

0 for echo reply message.

Code

0

Checksum

The checksum is the 16-bit ones's complement of the one's complement sum of the ICMP message starting with the ICMP Type. For computing the checksum , the checksum field should be zero. If the total length is odd, the received data is padded with one octet of zeros for computing the checksum. This checksum may be replaced in the future.

Identifier

If code = 0, an identifier to aid in matching echos and replies, may be zero.

Sequence Number

If code = 0, a sequence number to aid in matching echos and replies, may be zero.

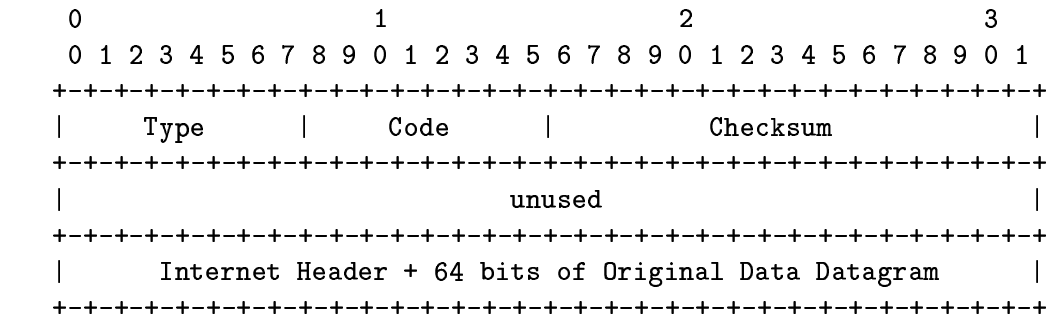
Description

The data received in the echo message must be returned in the echo reply message.

The identifier and sequence number may be used by the echo sender to aid in matching the replies with the echo requests. For example, the identifier might be used like a port in TCP or UDP to identify a session, and the sequence number might be incremented on each echo request sent. The echoer returns these same values in the echo reply.

Code 0 may be received from a gateway or a host.

Time Exceeded Message



IP Fields:

Destination Address

The source network and address from the original datagram's data.

ICMP Fields:

Type

11

Code

0 = time to live exceeded in transit;

1 = fragment reassembly time exceeded.

Checksum

The checksum is the 16-bit ones's complement of the one's complement sum of the ICMP message starting with the ICMP Type. For computing the checksum , the checksum field should be zero. This checksum may be replaced in the future.

Internet Header + 64 bits of Data Datagram

The internet header plus the first 64 bits of the original datagram's data. This data is used by the host to match the message to the appropriate process. If a higher level protocol uses port numbers, they are assumed to be in the first 64 data bits of the original datagram's data.

Description

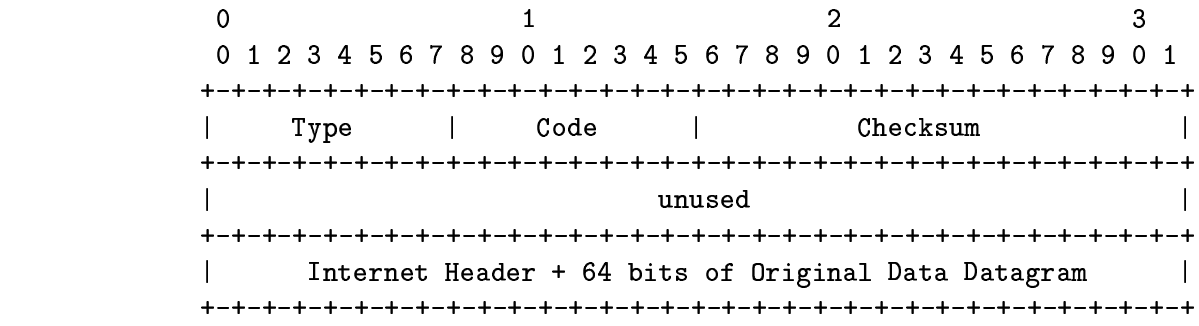
If the gateway processing a datagram finds the time to live field is zero it must discard the datagram. The gateway may also notify the source host via the time exceeded message.

If a host reassembling a fragmented datagram cannot complete the reassembly due to missing fragments within its time limit it discards the datagram, and it may send a time exceeded message.

If fragment zero is not available then no time exceeded need be sent at all.

Code 0 may be received from a gateway. Code 1 may be received from a host.

Destination Unreachable Message



IP Fields:

Destination Address

The source network and address from the original datagram's data.

ICMP Fields:

Type

3

Code

0 = net unreachable;

1 = host unreachable;

2 = protocol unreachable;

3 = port unreachable;

4 = fragmentation needed and DF set;

5 = source route failed.

Checksum

The checksum is the 16-bit ones's complement of the one's complement sum of the ICMP message starting with the ICMP Type. For computing the checksum , the checksum field should be zero. This checksum may be replaced in the future.

Internet Header + 64 bits of Data Datagram

The internet header plus the first 64 bits of the original datagram's data. This data is used by the host to match the message to the appropriate process. If a higher level protocol uses port numbers, they are assumed to be in the first 64 data bits of the original datagram's data.

Description

If, according to the information in the gateway's routing tables, the network specified in the internet destination field of a datagram is unreachable, e.g., the distance to the network is infinity, the gateway may send a destination unreachable message to the internet source host of the datagram. In addition, in some networks, the gateway may be able to determine if the internet destination host is unreachable. Gateways in these networks may send destination unreachable messages to the source host when the destination host is unreachable.

If, in the destination host, the IP module cannot deliver the datagram because the indicated protocol module or process port is not active, the destination host may send a destination

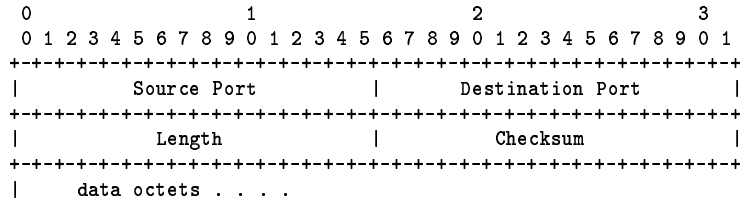
```
unreachable message to the source host.

Another case is when a datagram must be fragmented to be forwarded
by a gateway yet the Don't Fragment flag is on.  In this case the
gateway must discard the datagram and may return a destination
unreachable message.

Codes 0, 1, 4, and 5 may be received from a gateway.  Codes 2 and
3 may be received from a host.
```

B.3 UDP (*User Datagram Protocol*)

Abaixo está a descrição dos campos do cabeçalho do protocolo UDP, detalhes podem ser obtidos na RFC768 [91]:

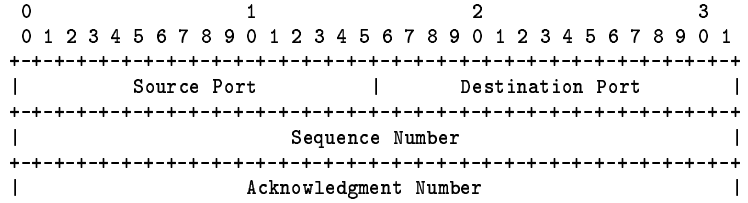


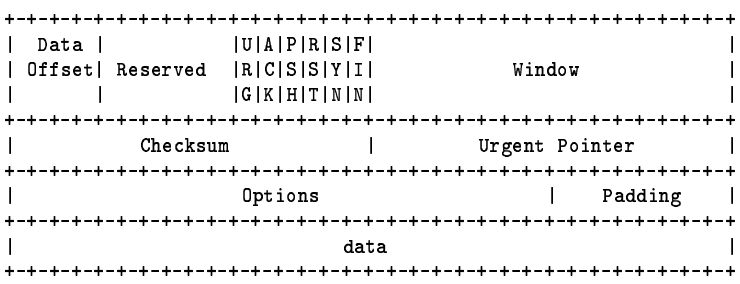
Na representação acima cada "-" representa um bit.

- `Source Port`: 16 bits. Campo que carrega o número da porta origem;
- `Destination Port`: 16 bits. Campo que carrega o número da porta destino do serviço UDP, ex: 161 para SNMP;
- `Length`: 16bits. Indica o tamanho em bytes do cabeçalho mais os dados do pacote UDP;
- `Checksum`: 16 bits. Campo que indica o *checksum* do cabeçalho.

B.4 TCP (*Transmission Control Protocol*)

Abaixo está a descrição dos campos do cabeçalho do protocolo TCP, detalhes podem ser obtidos na RFC793 [94]:





Na representação acima cada "-" representa um bit.

- `Source Port`: 16 bits. Campo que carrega o número da porta origem da conexão;
- `Destination Port`: 16 bits. Campo que carrega o número da porta destino da conexão, ex: 80 para HTTP;
- `Sequence Number`: 32 bits. Número de sequência do envio de bytes de dados;
- `Acknowledgment Number`: 32 bits. Se o bit ACK estiver ativo este campo indica o próximo número de sequência que o emissor da mensagem está esperando receber;
- `Data Offset`: 4 bits. Campo que indica o tamanho do cabeçalho do TCP em palavras de 32bits;
- `Reserved`: 6 bits. Reservado para uso futuro, deve ser igual a zero;
- `Control Bits`: 6 bits. Da esquerda para direita: URG: Urgent Pointer field significant, ACK: Acknowledgment field significant, PSH: Push Function, RST: Reset the connection, SYN: Synchronize sequence numbers, FIN: No more data from sender;
- `Window`: 16 bits. Este campo indica a quantidade de bytes que podem ser recebidos pelo emissor sem que seja necessário enviar uma confirmação de recepção (ACK);
- `Checksum`: 16 bits. Campo que indica o *checksum* do cabeçalho;
- `Urgent Pointer`: 16 bits. Campo utilizado em conjunto com o bit URG para apontar o número de sequência do segmento que contém os dados "urgentes";
- `Options`: Tamanho variável. Campo reservado para opções adicionais;

- `Padding`: Tamanho variável. Campo utilizado para garantir que o cabeçalho do TCP tenha um tamanho multiplo de 32bits;

Apêndice C

Listagem dos programas utilizados nos exemplos

C.1 Coleta de ICMP e SNMP

```
#!/usr/bin/perl

use SNMP_util;
use IPC::Open3;
use POSIX;

die <<USAGE unless $#ARGV == 2 ;

USAGE: $0 <hostname> <community> <drift>
where 0 =< drift < 60

USAGE


# FORMATO community@host:port:timeout:retries:backoff
$host = $ARGV[0];
$community = $ARGV[1];
$port = "161";
$drift = $ARGV[2];

die if ($drift < 0 || $drift > 59 );

$host_q = sprintf("%s@%s:%s",$community,$host,$port);

$ifDescr =         "1.3.6.1.2.1.2.2.1.2";
$ifType  =         "1.3.6.1.2.1.2.2.1.3";
$ifSpeed =         "1.3.6.1.2.1.2.2.1.5";
$ifAdminStatus =   "1.3.6.1.2.1.2.2.1.7";
$ifOperStatus =     "1.3.6.1.2.1.2.2.1.8";
$ifInOctets =      "1.3.6.1.2.1.2.2.1.10";
$ifInUcastPkts =   "1.3.6.1.2.1.2.2.1.11";
$ifInNUcastPkts =  "1.3.6.1.2.1.2.2.1.12";
$ifInDiscards =    "1.3.6.1.2.1.2.2.1.13";
$ifInErrors =      "1.3.6.1.2.1.2.2.1.14";
$ifOutOctets =     "1.3.6.1.2.1.2.2.1.16";
$ifOutUcastPkts =  "1.3.6.1.2.1.2.2.1.17";
$ifOutNUcastPkts = "1.3.6.1.2.1.2.2.1.18";
$ifOutDiscards =   "1.3.6.1.2.1.2.2.1.19";
$ifOutQLen =       "1.3.6.1.2.1.2.2.1.21";
$ipAdEntIfIndex =  "1.3.6.1.2.1.4.20.1.2";
$ipAdEntNetMask =  "1.3.6.1.2.1.4.20.1.3";
$ifAlias =         "1.3.6.1.2.1.31.1.1.1.18";

my ( %hash_ip, %hash_discard_valid);
```

```
my ( %hash_proto, %hash_if, %hash_mask);



#############################################################
# SUBROTINA de Leitura de configuracao #
#############################################################
sub LeConfiguracao {

  # Apaga todas as variaveis
  %hash_ip=();
  %hash_ip2oid = ();
  %hash_discard_valid=();
  %hash_proto=();
  %hash_if=();
  %hash_mask=();

  #
  # Le e coloca no %hash_ip
  # os IPs das interfaces, para serem "pingados"
  #
  @ret = &snmpwalk( $host_q, $ipAdEntIfIndex);
  foreach $x ( @ret )
  {
   @a = split /:/, $x;
if ( $hash_ip2oid{$a[0]} )
{
print "IP $a[0] duplicado\n";
next;
}
   $hash_ip{$a[1]} = $a[0];
$hash_ip2oid{$a[0]} = $a[1];
  }

  #
  #  Le e coloca no %hash_discard_valid
  #  quais as interfaces que tem o contador Discard
  #
  @ret = &snmpwalk( $host_q, $ifOutDiscards);
  foreach $x ( @ret )
  {
            @a = split /:/, $x;
      $hash_discard_valid{$a[0]} = 1 ;
  }

  #
  #  Le quais os protocolos das Interfaces
  #  FrameRelay = 32
  #
  @ret = &snmpwalk( $host_q, $ifType);
  foreach $x ( @ret )
  {
          @a = split /:/, $x;
   $hash_proto{$a[0]} = $a[1];
  }

  #
  #  Coloca no %hash_mask apenas os ids com mascara /30
  #
  foreach $x ( keys %hash_ip )
  {
     $oid = sprintf("%s.%s",$ipAdEntNetMask, $hash_ip{$x});
    @ret = &snmpget($host_q, $oid);

#  Retira os "Frame Relays" por causa dos filtros da EBT
if ( $hash_proto{$x} eq "32" )
{
delete $hash_ip{$x};
next;
}

   if ( $ret[0] =~/255.255.255.252/ )
   {
```

```
   $hash_mask{$x} = $ret[0];
# Calcula o IP do Alvo (outro lado do enlace)
         ($ip1,$ip2,$ip3,$ip4)=split (/\./,$hash_ip{$x});
         $a=($ip4 & 252);
         if (($a+1) == $ip4) { $a+=2; }
         else { $a+=1; }
$hash_ip{$x} ="$ip1.$ip2.$ip3.$a";
   }
else
{
delete $hash_ip{$x};
}
  }

  #
  #  Coloca no %hash_if os nomes das interfaces
  #
  foreach $x ( keys %hash_ip )
  {
   $oid = sprintf("%s.%s",$ifDescr, $x);
   @ret = &snmpget($host_q, "$ifDescr.$x", "$ifAdminStatus.$x");
        if ( $ret[0] =~ /Loopback/ || $ret[1] != "1" )
{
delete $hash_ip{$x};
next;
}
   $hash_if{$x} = $ret[0];
  }

}
#  Fim da Rotina LeConfiguracao


############################################################
#   Rotina "ObtemDados()":
#   faz SNMPGET e grava no MySQL(NAO IMPLEMENTADO)
############################################################
sub ObtemDados {

  # Envia ICMPs via fping
  #
  $cmd = "/usr/local/sbin/fping -e -r 2 -i25 -t 2000";
  # Retirado por problemas no BUFFER
  #  $pid = open3(\*FPING_IN, \*FPING_OUT, \*FPING_ERR, $cmd );

  my $agora = time;
  my $date = localtime;

  $fpingtmpfile="/tmp/fping_tmp-$host-$agora.txt";
  open (ARQTMP, ">$fpingtmpfile")
      or die "$date:ERROR: Opening for write temporary EBTfping file\n";

  $fpingoutfile="/tmp/fping_out-$host-$agora.txt";
  open (ARQOUT, ">$fpingoutfile")
      or die "$date:ERROR: Opening for write output EBTfping file\n";

  $fpingerrfile="/tmp/fping_err-$host-$agora.txt";
  open (ARQERR, ">$fpingerrfile")
      or die "$date:ERROR: Opening for write error EBTfping file\n";


  foreach $x ( keys %hash_ip )
  {
        print ARQTMP "$hash_ip{$x}\n";
  }
  close (ARQTMP);

  my %hash_fping=();

  $a=system("cat $fpingtmpfile | $cmd > $fpingoutfile 2> $fpingerrfile");

  if ( $a eq "32512" )
  {
```

```
      print "$date:ERROR:$a: Executing $cmd \n";
      close (ARQERR);
      close (ARQOUT);
      `rm $fpingtmpfile`;
      `rm $fpingoutfile`;
      `rm $fpingerrfile`;
      exit;
  }
  close(ARQERR);
  close(ARQOUT);

  open (ARQOUT, "<$fpingoutfile")
      or die "$date:ERROR: Opening for read output EBTfping file\n";

  open (ARQERR, "<$fpingerrfile")
      or die "$date:ERROR: Opening for read error EBTfping file\n";

  while ( <ARQOUT> )
  {
        chomp;
        if ( /(.+) is alive \((.+) ms\)/ )
        {
                $hash_fping{$1}=$2;
        }
        if ( /(.+) is unreachable/ )
        {
if ( ! ( $hash_fping{$1} < 0 && $hash_fping{$1} > -1000 ) ){
                 $hash_fping{$1}= -10000 ;
}
        }
  }
  close (ARQOUT);

  while ( <ARQERR> )
  {
chomp;
if ( /(.+) for ICMP Echo sent to (.+)/ )
{
$alvo = $2;
if ( /Time Exceeded/ )
{
if ( $hash_fping{$alvo} < -200 &&
                           $hash_fping{$alvo} > -299 )
                        {
                                $hash_fping{$alvo}--;
}
                        else
                        {
                                $hash_fping{$alvo}= -201;
                        }
}
if ( / Host Unreachable /)
{
if ( $hash_fping{$alvo} < -100 &&
                           $hash_fping{$alvo} > -199 )
                        {
                                $hash_fping{$alvo}--;
}
                        else
                        {
                                $hash_fping{$alvo}= -101;
                        }
}
if ( /Unreachable \(Host Precedence Violation\)/ )
{
if ( $hash_fping{$alvo} < -300 &&
     $hash_fping{$alvo} > -399 )
{
$hash_fping{$alvo}--;
}
else
{
$hash_fping{$alvo}= -301;
```

```
}
}
  }
  }
  close (ARQERR);

  `rm $fpingtmpfile`;
  `rm $fpingoutfile`;
  `rm $fpingerrfile`;

  $need_read_conf = 0;
  # Loop para pegar os dados na MIB
  foreach $x ( keys %hash_ip )
  {
# Infelizmente a Cisco nao implementa Discards em toda MIB
   if ( $hash_discard_valid{$x} )
   {
   $n = @ret = &snmpget($host_q, "$ifDescr.$x",
"$ifOperStatus.$x", "$ifSpeed.$x", "$ifInOctets.$x",
"$ifInUcastPkts.$x", "$ifOutOctets.$x", "ifOutUcastPkts.$x",
"$ifOutDiscards.$x", "$ifAlias.$x");
   }
# Se n nao for maior que 5, significa alguma troca de oids,
# entao e' necessario reler as configuracoes.
        if ( $n > 5 )
{
$need_read_conf = GravaBancoDados(
 $hash_fping{$hash_ip{$x}}, @ret );
}
else
{
$need_read_conf = 1;
}
  }

  return ($need_read_conf);

}
#
#  Fim da rotina "ObtemDados()"
############################################################

############################################################
# Rotina: GravaBancoDados
# Ainda nao implementada
############################################################

sub GravaBancoDados
{
$retorno = join ("\#",@_);
$agora = time;
print "$agora#$host#$retorno\n";
return (0);
}



MAIN:
{
  sleep ($drift);

  LeConfiguracao();
  ObtemDados();

}
```

C.2 Coleta e consolidação via `libcap`

```
extern "C" {
```

```
#include <stdlib.h>
#include <signal.h>
#include <stdio.h>

#include <unistd.h>
#include <ctype.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <netinet/in_systm.h>
#include <net/if.h>
#include <arpa/inet.h>

#include <netdb.h>

#include <netinet/ip.h>
#include <netinet/tcp.h>
#include <netinet/udp.h>
#include <netinet/ip_icmp.h>

#include <pcap.h>
#include <net/bpf.h>
#include <math.h>

#include "analise.h"
}

#include <string>
#include <fstream>
#include <iostream>
#include <iomanip>
#include <map>

typedef struct {
    u_int    Pkts;
    u_int    Bytes;
}  SimpleType;

typedef map< string, SimpleType> CounterMapSimple;
CounterMapSimple ToPrint;

typedef struct {
  string      CountryCode;
  uint16_t    Age;
} CrtTblType, *CrtTblTypePtr;

typedef map<ipv4addr_t, CrtTblType> CrtTblMap;
CrtTblMap  CrtTblMapArr[33];

typedef map< u_int, string > ProtoNameMap;
ProtoNameMap ProtoName;

typedef map< string, u_int> RxtMap;
RxtMap Rxts, IPFlow, ConFlow;

SimpleType Tot_Traf;

time_t  anterior = 0;

u_int rxt_analysis = 0;



int main ( int argc, char **argv )
{

    char *read_file = NULL;
    char pc_err[PCAP_ERRBUF_SIZE];
    pcap_t *pd = NULL;
    int link_offset;
    struct bpf_program pcapfilter;
    struct in_addr mask;
```

```
    int c;

    while ((c = getopt(argc, argv, "rI:")) != EOF)
    {
        switch (c)
{
            case 'I':
       read_file = optarg;
       break;
    case 'r':
rxt_analysis = 1;
break;
            default:
usage(-1,argv);
}
    }

    if (read_file)
    {
        if (!(pd = pcap_open_offline(read_file, pc_err)))
{
            perror(pc_err);
            exit_program(-1);
        }
cerr << "Input: " << read_file << endl;
    }
    else usage(-1,argv);

    if (argv[optind])
    {
        string filter;
        for ( c = optind; c < argc; c++)
{
    filter.append(" ");
    filter.append(argv[c]);
}
        if (pcap_compile(pd, &pcapfilter,(char *)filter.c_str(),0,mask.s_addr))
{
            pcap_perror(pd, "pcap compile");
            exit_program(-1);
        }
cerr << "Filter: " << filter << endl;
        if (pcap_setfilter(pd, &pcapfilter))
{
            pcap_perror(pd, "pcap set");
            exit_program(-1);
        }
    }

    switch( pcap_datalink(pd) )
    {
        case DLT_IEEE802:
case DLT_EN10MB:
    break;
        default:
    cerr << "Error: Only EN10MB or IEEE802 are supported\n";
    exit_program(-1);
    }

    if ( LeTabelaRegistros() ){
        cout << "ERRO na leitura da tabela do ARIN\n";
        exit(1);
    }

    while (pcap_loop(pd, 0, (pcap_handler)process, 0));

    //print_results();

}


//--------------------------------------------------------------------------
```

```
//                    int LeTabelaRegistros (void)
//.....................................................................
//
//---------------------------------------------------------------------
int LeTabelaRegistros ( void )
{
  fstream       fin;
  string        linha;
  string::size_type     begIdx, endIdx;
  // Definicoes para tabela de rotas
  ipv4addr_t    CidrTmp, NetTmp;


  fin.open("registro_redes.txt", std::ios::in);

  if ( ! fin ) {
    cerr << "Error: opening input table registro_redes.txt" << endl;
    return(EXIT_FAILURE);
  }

  while ( getline(fin,linha) ) {
    // Linha exemplo:
    // ARIN|US|155.189.0.0|24

    #ifdef DEBUG2
    cout << linha << endl;
    #endif

    begIdx=0;

    endIdx = linha.find('|',begIdx);
    begIdx = endIdx+1;

    endIdx = linha.find('|',begIdx);
    string pais_code(linha,begIdx,endIdx-begIdx);
    begIdx = endIdx+1;

    endIdx = linha.find('|',begIdx);
    string rede_ip(linha,begIdx,endIdx-begIdx);
    begIdx = endIdx+1;

    endIdx = linha.find('|',begIdx);
    string rede_cidr(linha,begIdx,endIdx-begIdx);
    begIdx = endIdx+1;

    NetTmp = inet_addr(rede_ip.c_str());
    CidrTmp = atoi(rede_cidr.c_str());

    CrtTblMapArr[CidrTmp][NetTmp].CountryCode=pais_code;
    CrtTblMapArr[CidrTmp][NetTmp].Age=0;

  }
  fin.close();
  return (0);
}


//---------------------------------------------------------------------
//   CrtTblMap::iterator ProcuraTabela ( ipv4addr_t InAddr )
//.....................................................................
//
//---------------------------------------------------------------------
CrtTblMap::iterator ProcuraTabela ( ipv4addr_t InAddr )
{
  CrtTblMap::iterator      MapPos;
  CrtTblType       Tbl;
  int                   Cidr, Int;
  ipv4addr_t            Mask, Net;
  struct in_addr        sAddr;

  for (int Cidr = 32; Cidr > 0; Cidr-- )
  {
    // Acha a rede IP correspondente a CidrLocal
```

```
    Mask = 0xffffffff;
    for (Int=32; Int > Cidr ; Mask=Mask<<1, Int--);
    Net = InAddr & htonl(Mask);
    MapPos = CrtTblMapArr[Cidr].find(Net);
    // Para sair do loop caso tenha achado a rede
    if (MapPos != CrtTblMapArr[Cidr].end()) { Cidr = 0; }
  }

  return (MapPos);
}



//---------------------------------------------------------------------
//                    void print_results (time_t norm120)
//.....................................................................
//
//---------------------------------------------------------------------
void print_results (time_t norm120)
{
    CounterMapSimple::iterator   Pos;

    cout << norm120 << ":";
    for (Pos = ToPrint.begin(); Pos != ToPrint.end(); ++Pos) {
cout << ":" << Pos->first << "(" << Pos->second.Pkts;
cout << "," << Pos->second.Bytes << ")";
    }
    cout << endl;
}


//---------------------------------------------------------------------
//      void process(u_char *data1, struct pcap_pkthdr* h, u_char *p)
//.....................................................................
//
//---------------------------------------------------------------------
void process(u_char *data1, struct pcap_pkthdr* h, u_char *p)
{

    struct ether_header* ep = (struct ether_header *)p;
    struct ip* ip_packet = (struct ip *)(p + ETHER_HDRLEN);
    unsigned ip_hl = ip_packet->ip_hl*4;
    unsigned ip_off = ntohs(ip_packet->ip_off);
    unsigned fragmented = ip_off & (IP_MF | IP_OFFMASK);
    unsigned frag_offset = fragmented?(ip_off & IP_OFFMASK) * 8:0;

    char *data;
    int len;
    struct protoent *ppn;
    char nome[20];

    ipv4addr_t SrcIP, DstIP;

    CounterMapSimple::iterator Pos;
    ProtoNameMap::iterator Pos0;

    //printf("%d.%d:",h->ts.tv_sec, h->ts.tv_usec);

    // Testa para ver se e' um pacote IP dentro da Ethernet
    if ( ntohs(ep->ether_type) != 0x0800 ) {
        cerr << "Non IP" << endl;
        return;
    }

    // Verifica de o pacote IP nao tem options
    int options = 0;
    if ( ip_packet->ip_hl != 5 ) {
        // cerr << "IP with options NOT supported" << endl;
options = 1;
    }

    string protoname;
    Pos0 = ProtoName.find(ntohs(ip_packet->ip_p));
```

```
    if ( Pos0 == ProtoName.end() ) {
        setprotoent(1);
        ppn = getprotobynumber(ip_packet->ip_p);
        if ( ppn ) {
    protoname = ppn->p_name;
    ProtoName[ntohs(ip_packet->ip_p)] = ppn->p_name;
} else {
    protoname = "unknown";
    ProtoName[ntohs(ip_packet->ip_p)] = "unknown";
}
        endprotoent();
    }
    else  protoname = Pos0->second;

    CrtTblMap::iterator MapPos;
    string tmp;

    DstIP = ip_packet->ip_dst.s_addr;
    MapPos=ProcuraTabela(DstIP);
    if ( MapPos != CrtTblMapArr[1].end() ) {
tmp = "Dst," + MapPos->second.CountryCode;
        ToPrint[tmp].Bytes+=ntohs(ip_packet->ip_len);
        ToPrint[tmp].Pkts++;
    } else {
ToPrint["Dst,null"].Bytes+=ntohs(ip_packet->ip_len);
        ToPrint["Dst,null"].Pkts++;
    }

    SrcIP = ip_packet->ip_src.s_addr;
    MapPos=ProcuraTabela(SrcIP);
    if ( MapPos != CrtTblMapArr[1].end() ) {
tmp = "Src," + MapPos->second.CountryCode;
        ToPrint[tmp].Bytes+=ntohs(ip_packet->ip_len);
        ToPrint[tmp].Pkts++;
    } else {
        ToPrint["Src,null"].Bytes+=ntohs(ip_packet->ip_len);
        ToPrint["Src,null"].Pkts++;
    }

    time_t norm60 = (time_t)(60*floor(h->ts.tv_sec/60));

    if ( norm60 > anterior ) {
if ( anterior ) print_results(norm60);
ToPrint.clear();
    }

    anterior = h->ts.tv_sec;
}



void usage ( int e, char **argv )
{
      cerr << "usage: " << argv[0] << " [-r] -I <tcpdump_bin_dump_file>"
           << " [filter expression]" << endl;
      exit(e);
}



void exit_program ( int signal )
{
    if ( signal >= 0 ) printf("exit\n");

    exit ( signal );
}
```

C.3 Consolidação para tabelas BGP

```
#!/usr/bin/perl -w

use diagnostics;
use Time::Local;

die <<USAGE unless $#ARGV == 2 ;

USAGE: $0 <dump_text_dir> <IBGP|EBGP> <ASN>

USAGE


my $dumpdir   = $ARGV[0];
my $bgptype   = $ARGV[1];
my $bgpasn    = $ARGV[2];


MAIN:
{

    @lista = `ls $dumpdir | egrep "^2001(.+).txt"`;

    foreach $file ( sort @lista )
    {
chomp($file);
     open (IN, "<$dumpdir/$file") or die "ERROR opennig $file\n";
print STDERR "Working on $file\n";


my %hash_obj = ();
my %hash_nei = ();
my %hash_path = ();
my %hash_pref = ();

$datao=$file;
$datao=~s/\.txt//;
        $YY = substr($datao,0,4); $mm = substr($datao,4,2);
$dd = substr($datao,6,2); $HH = substr($datao,8,2);
$MM = substr($datao,10,2); $SS = substr($datao,12,2);

        $epoch = timelocal($SS, $MM, $HH, $dd, $mm-1, $YY-1900);

        $totalr=0;
while (<IN>)
{
    chomp;
    @a = split /\|/;
    if ( $#a > 0 && $bgptype eq "IBGP" ) { $a[1] = $bgpasn." ".$a[1]; }
    if ( $#a == 0 && $bgptype eq "IBGP" ) { $a[1] = $bgpasn; }
    @b = split (/\//, $a[0]);
    @c = split (/\s/, $a[1]);

       # Retira os as-prepends ...
    my %achou = ();
    my @unico = ();
    foreach my $tmp (@c)
    {
push(@unico, $tmp) unless $achou{$tmp}++;
    }

    $pa = sprintf("PATH%02d",$#unico+1);
    $pr = sprintf("PREF%02d",$b[1]);

    if ( $unico[1] )
       {
$nei = sprintf("%d_%05d",$unico[0],$unico[1]);
    }
    else
    {
$nei = "$unico[0]";
    }
```

```
    if ( $hash_obj{"${nei}|${pa}"} ) { $hash_obj{"${nei}|${pa}"}++; }
    else  { $hash_obj{"${nei}|${pa}"}=1; }

    if ( $hash_obj{"${nei}|${pr}"} ) { $hash_obj{"${nei}|${pr}"}++; }
            else  { $hash_obj{"${nei}|$pr"}=1; }

    if ( $hash_nei{$nei} ) { $hash_nei{$nei}++; }
            else  { $hash_nei{${nei}}=1; }

    if ( $hash_path{$pa} ) { $hash_path{$pa}++; }
            else  { $hash_path{${pa}}=1; }

    if ( $hash_pref{$pr} ) { $hash_pref{$pr}++; }
            else  { $hash_pref{$pr}=1; }

    $totalr++;
}
        close (IN);

#if ( $totalr < 50000 ) { next; }

foreach $x ( sort keys %hash_obj )
{
    print "$epoch:$x:$hash_obj{$x}\n";
}
        foreach $x ( sort keys %hash_nei )
{
      print "$epoch:$x:$hash_nei{$x}\n";
        }
foreach $x ( sort keys %hash_path )
{
    print "$epoch:$x:$hash_path{$x}\n";
}
foreach $x ( sort keys %hash_pref )
        {
            print "$epoch:$x:$hash_pref{$x}\n";
        }
print "$epoch:TOTAL:$totalr\n";
    }

}
```

C.4 Biblioteca para criação de matrizes em 3D

```
#include <gd.h>
#include <stdio.h>
#include <stdlib.h>
#include <strings.h>
#include "crt3D.h"
#include <math.h>
#include <gdfonts.h>

/* Remove comment to DEBUG
#define DEBUG
*/

#define PI 3.1415927

/***********************************************/
/* Parameters used to convert 3D space to 2D plane*/
/***********************************************/
#define XOFFSET 250 /* offset for (0,0,0) */
#define YOFFSET 260 /* offset for (0,0,0) */
#define ZANGLE 60 /* angle from Z axis to vertical axis*/
#define ZRAD ZANGLE*PI/180 /* or radians */
```

```
#define XANGLE  10 /* angle from X axis to horizontal axis*/
#define XRAD XANGLE*PI/180 /* or radians */

/*************************************************************/
/* Main function used to convert the 3D space into 2D plane */
/*************************************************************/
void crt3Dconvert2D(crt3DPointPtr p3, gdPointPtr p2, int c)
{
int a;
int dx, dy;

for (a=0; a<c; a++)
{
         dx=(int)(p3[a].x*cos(XRAD)-p3[a].y*sin(ZRAD));
         dy=(int)(-p3[a].x*sin(XRAD)-p3[a].y*cos(ZRAD)+p3[a].z);
         p2[a].x=XOFFSET+ dx;
         p2[a].y=YOFFSET- dy;
#ifdef DEBUG
  printf("Convert 3D->2D: [%d,%d,%d] => [%d,%d]\n",
p3[a].x,p3[a].y,p3[a].z,p2[a].x,p2[a].y);
#endif
}
}

void crtDrawPie2D(gdImagePtr im, int *co, int l,
         int bg, int fg, int black, int white,
         gdFontPtr f, crt2DObjectTypePtr o, int omax,
         double vmin, double vmax, double vtotal, char *unit)
{
    div_t m;
    unsigned int a, b, c;
    char s[20];

#ifdef DEBUG
    printf("crtDrawHisto2D: Input: im=%d, *co=%d, l=%d, bg=%d, fg=%d, black=%d, white=%d, f=%d, o=%d, omax=%d
, vmin=%.4f, vmax=%.4f, *unit=%s",
         im, *co, l, bg, fg, black, white, f, o, omax, vmin, vmax, *unit);
#endif

    printf("DrawPie:l=%d, omax=%d, vmin=%.2f, vmax=%.2f, vtotal=%.2f, unit=%s\n",
l,omax,vmin,vmax,vtotal,unit);

    for (a=0; a<(unsigned int)omax; a++)
    {
        printf("DrawHisto:a=%d, o[a].o=%s\n",a,o[a].o);
        crt2DImageStringRight(im, f, 35, a*20+40 , o[a].o , fg);
        printf("Entrando no loop:(%.2f-%.2f)/(%.2f-%.2f)*510=%.2f\n",
        o[a].v,vmin,vmax,vmin,(o[a].v-vmin)/(vmax-vmin)*510);
        for (b=0; b<= (o[a].v-vmin)/(vmax-vmin)*510; b++)
        {
            gdImageLine(im, b+40, a*20+42, b+40, a*20+52, co[b*l/511]);
        }
        CrtConvUnidade(s,o[a].v);
        printf("DrawHisto:a=%d, s=%s\n",a,s);
        gdImageString(im, f, b+45, a*20+40 , s, fg);
sprintf(s,"(%.1f%%)",100*o[a].v/vtotal);
        gdImageString(im, f, b+45+40, a*20+40 , s, fg);
    }

}

void crtDrawHisto2D(gdImagePtr im, int *co, int l,
         int bg, int fg, int black, int white,
         gdFontPtr f, crt2DObjectTypePtr o, int omax,
         double vmin, double vmax, char *unit)
{
    div_t m;
    unsigned int a, b, c;
    char s[20];

#ifdef DEBUG
    printf("crtDrawHisto2D: Input: im=%d, *co=%d, l=%d, bg=%d, fg=%d, black=%d, white=%d, f=%d, o=%d, omax=%d, vmin=
 im, *co, l, bg, fg, black, white, f, o, omax, vmin, vmax, *unit);
```

```
#endif

    printf("DrawHisto:l=%d, omax=%d, vmin=%.2f, vmax=%.2f, unit=%s\n",l,omax,vmin,vmax,unit);

    for (a=0; a<(unsigned int)omax; a++)
    {
     printf("DrawHisto:a=%d, o[a].o=%s\n",a,o[a].o);
        crt2DImageStringRight(im, f, 35, a*20+40 , o[a].o , fg);
printf("Entrando no loop:(%.2f-%.2f)/(%.2f-%.2f)*540=%.2f\n",
o[a].v,vmin,vmax,vmin,(o[a].v-vmin)/(vmax-vmin)*540);
        for (b=0; b<= (o[a].v-vmin)/(vmax-vmin)*540; b++)
        {
            gdImageLine(im, b+40, a*20+42, b+40, a*20+52, co[b*l/541]);
        }
CrtConvUnidade(s,o[a].v);
     printf("DrawHisto:a=%d, s=%s\n",a,s);
        gdImageString(im, f, b+45, a*20+40 , s, fg);
    }

}


void crtDrawHisto3D(gdImagePtr im, int *co, int l,
         int bg, int fg, int black, int white,
         gdFontPtr f, crt2DObjectTypePtr o, int omax,
         double vmin, double vmax, char *unit)
{
    crt3DPoint p[1];
    gdPoint pn[1];
    div_t m;
    unsigned int a, b;
    char s[20];

    for (a=0; a<(unsigned int)omax; a++)
    {
p[0].x=5+30*a-170;   p[0].y=70;   p[0].z=-40;
crt3DImageStringXYLeft(im, f, p, o[a].o , fg);
        for (b=0; b<= (o[a].v-vmin)/(vmax-vmin)*250; b+=2)
        {
    p[0].x=5+30*a-170;   p[0].y=50;   p[0].z=b-40;
    crt3DImageCube(im, p, 20, 2, co[b*l/251], black);
        }
p[0].y-=15;
p[0].x-=8;
CrtConvUnidade(s,o[a].v);
crt3DImageStringXYRight(im, f, p, s, fg);
    }

}

void CrtConvUnidade(char *o, double i)
{
if (i/1000 > 1)
{
if (i/1000000 > 1)
{
if (i/1000000000 > 1)
{
sprintf(o,"%.2fG",i/1000000000);
}
else
{
sprintf(o,"%.2fM",i/1000000);
}
}
else
{
sprintf(o,"%.2fK",i/1000);
}

}
else
{
```

```
sprintf(o,"%.2f",i);
}
}

void crt2DImageStringRight(gdImagePtr im, gdFontPtr f, int x, int y,
       char *s, int color)
{
    unsigned int a;

    for (a=0 ; a < strlen(s) ; a++)
    {
        x-=6;
        gdImageChar(im, f, x, y, s[strlen(s)-a-1], color);
    }
}

void crt3DImageStringUp(gdImagePtr im, gdFontPtr f, crt3DPointPtr p, char *s, int c)
{
gdPoint pn[1];

crt3Dconvert2D(p, pn, 1);
/* printf("String: %s\n",s);*/
gdImageStringUp(im, f, pn[0].x, pn[0].y, s, c);
}

void crt3DImageString(gdImagePtr im, gdFontPtr f, crt3DPointPtr p, char *s, int c)
{
gdPoint pn[1];

crt3Dconvert2D(p, pn, 1);
/* printf("String: %s\n",s);*/
gdImageString(im, f, pn[0].x, pn[0].y, s, c);
}

void crt3DImageStringRight(gdImagePtr im, gdFontPtr f, crt3DPointPtr p, char *s, int c)
{
gdPoint pn[1];

crt3Dconvert2D(p, pn, 1);
crt2DImageStringRight(im, f, pn[0].x, pn[0].y, s, c);
}


void crt3DImageStringXYRight(gdImagePtr im, gdFontPtr f, crt3DPointPtr p,
       char *s, int color)
{
    unsigned int a;
    gdPoint pn[1];

    for (a=0; a<strlen(s); a++)
    {
p[0].y-=7;
crt3Dconvert2D(p, pn, 1);
        gdImageChar(im, f, pn[0].x, pn[0].y, s[a], color);
    }
}

void crt3DImageStringXYLeft(gdImagePtr im, gdFontPtr f, crt3DPointPtr p,
       char *s, int color)
{
    unsigned int a;
    gdPoint pn[1];

    p[0].y+=7*strlen(s);
    for (a=0; a<strlen(s); a++)
    {
        p[0].y-=7;
        crt3Dconvert2D(p, pn, 1);
        gdImageChar(im, f, pn[0].x, pn[0].y, s[a], color);
    }
}
```

```
void crtAllocateColors(gdImagePtr im, int *c, int l)
{
/*
        c[0] = gdImageColorAllocate(im, 0, 0, 248);
        c[1] = gdImageColorAllocate(im, 0, 80, 248);
        c[2] = gdImageColorAllocate(im, 0, 172, 248);
        c[3] = gdImageColorAllocate(im, 0, 252, 248);
        c[4] = gdImageColorAllocate(im, 0, 255, 149);
        c[5] = gdImageColorAllocate(im, 112, 255, 89);
        c[6] = gdImageColorAllocate(im, 169, 255, 113);
        c[7] = gdImageColorAllocate(im, 232, 242, 8);
        c[8] = gdImageColorAllocate(im, 248, 188, 8);
        c[9] = gdImageColorAllocate(im, 251, 121, 8);
        c[10] = gdImageColorAllocate(im, 248, 12, 8);
*/
  int a, x, r, g, b;
  double rad;
  int *cl;

  cl = (int *) *c = calloc(l,sizeof(int));
  for (a=0;a<l;a++)
  {

    rad = a*(PI/2)/l;
    g=rint(255*sin(rad*2));

    x=a+1;
    r=rint(255*(log(x)/log(l)));
    b=255-r;

/*    printf("Alloca cor[%d]=%d,%d,%d\n",a,r,g,b);*/
    cl[a] = gdImageColorAllocate(im, r, g, b);
  }

}


void crtDrawMatrix(gdImagePtr im, int *co, int l,
 int bg, int fg, int black, int white,
 gdFontPtr f,
 crt3DMatrixTypePtr inm, int n,
 double vmin, double vmax,
 crt3DLegendTypePtr xleg, crt3DLegendTypePtr yleg,
 int xlegmax, int ylegmax, char *unit)
{
crt3DPoint p[1];
double z;
div_t m;
int a, b, c, zg;
char s[20];

z=vmin;
        for (zg=0; zg<=250; zg+=2)
        {
            for (a=n-1; a>=0; a--)
            {
                if (inm[a].v >= z )
                {
                        for (b=0; b<xlegmax; b++)
                        {
                                for (c=0; c<ylegmax; c++)
                                {
                                        if ( (0 == strcmp(xleg[b].n,inm[a].o1)) &&
                                             (0 == strcmp(yleg[c].n,inm[a].o2)) )
                                        {
                                                p[0].x = 5+(b*40);
p[0].y = 5+(c*40);
p[0].z = zg;

                                                crt3DImageCube(im, p, 30, 2,
                                                         co[zg*l/251], black);
                                        }
                                }
                        }
```

```
     p[0].x = 40+(xlegmax*40); p[0].y = 0; p[0].z = zg+20;
          crt3DImageCube(im, p, 20, 2, co[zg*1/251], black);
m = div(zg,18);
if ( 0 == m.rem )
{
             p[0].x = 70+(xlegmax*40);
p[0].y = 0; p[0].z = zg+20;
CrtConvUnidade(s,z);
             crt3DImageString(im, f, p, s , fg);
}
                }
            }
    z+=(vmax-vmin)/125;
        }

p[0].x = 70+(xlegmax*40);
        p[0].y = 0; p[0].z = zg+20;
        CrtConvUnidade(s,z);
        crt3DImageString(im, f, p, s , fg);
        p[0].z = zg+40;
        crt3DImageString(im, f, p, unit , fg);


for (a=0; a<xlegmax; a++){
                p[0].x = 5+(a*40);
p[0].y = 5+(ylegmax*40);
p[0].z = 0;
crt3DImageString(im, f, p, xleg[a].n, fg);
}
for (a=0; a<ylegmax; a++){
p[0].x = 5+(xlegmax*40);
                p[0].y = 10+(a*40);
p[0].z = 0;
crt3DImageString(im, f, p, yleg[a].n, fg);
}

}

void crtDraw2ValueMatrix(gdImagePtr im, int *co, int l,
 int bg, int fg, int black, int white,
 gdFontPtr f,
 crt3DMatrix2ValueTypePtr inm, int n,
 double v1min, double v1max, double v2min, double v2max,
 crt3DLegendTypePtr xleg, crt3DLegendTypePtr yleg,
 int xlegmax, int ylegmax, char *unit)
{
crt3DPoint p[1];
double z1, z2;
div_t m;
int a, b, c, zg;
char s[20];

z1=v1min;
z2=v2min;
        for (zg=0; zg<=250; zg+=2)
        {
            for (a=n-1; a>=0; a--)
            {
                if (inm[a].v1 >= z1 )
                {
                        for (b=0; b<xlegmax; b++)
                        {
                                for (c=0; c<ylegmax; c++)
                                {
                                        if ( (0 == strcmp(xleg[b].n,inm[a].o1)) &&
                                             (0 == strcmp(yleg[c].n,inm[a].o2)) )
                                        {
                                                p[0].x = 5+(b*40);
p[0].y = 5+(c*40);
p[0].z = zg;
                                                crt3DImageTriangleRight(im, p, 30, 2,
                                                         co[zg*1/251], black);
                                        }
```

```
                                }
                        }
     p[0].x = 40+(xlegmax*40); p[0].y = 0; p[0].z = zg+20;
          crt3DImageTriangleRight(im, p, 20, 2, co[zg*1/251], black);
m = div(zg,18);
if ( 0 == m.rem )
{
             p[0].x = 70+(xlegmax*40);
p[0].y = 0; p[0].z = zg+20;
CrtConvUnidade(s,z1);
             crt3DImageString(im, f, p, s , fg);
}
                }

                if (inm[a].v2 >= z2 )
                {
                        for (b=0; b<xlegmax; b++)
                        {
                                for (c=0; c<ylegmax; c++)
                                {
                                        if ( (0 == strcmp(xleg[b].n,inm[a].o1)) &&
                                             (0 == strcmp(yleg[c].n,inm[a].o2)) )
                                        {
                                                p[0].x = 5+(b*40);
p[0].y = 5+(c*40);
p[0].z = zg;

                                                crt3DImageTriangleLeft(im, p, 30, 2,
                                                         co[zg*1/251], black);
                                        }
                                }
                        }
     p[0].x = 40+(xlegmax*40); p[0].y = 0; p[0].z = zg+20;
          crt3DImageTriangleLeft(im, p, 20, 2, co[zg*1/251], black);
m = div(zg,18);
if ( 0 == m.rem )
{
             p[0].x = 70+(xlegmax*40);
p[0].y = 0; p[0].z = zg-20;
CrtConvUnidade(s,z2);
             crt3DImageStringRight(im, f, p, s , fg);
}
                }
            }
    z1+=(v1max-v1min)/125;
    z2+=(v2max-v2min)/125;
        }

p[0].x = 70+(xlegmax*40);
        p[0].y = 0; p[0].z = zg+20;
        CrtConvUnidade(s,z1);
        crt3DImageString(im, f, p, s , fg);
        p[0].z = zg+40;
        crt3DImageString(im, f, p, unit , fg);


for (a=0; a<xlegmax; a++){
                p[0].x = 5+(a*40);
p[0].y = 5+(ylegmax*40);
p[0].z = 0;
crt3DImageString(im, f, p, xleg[a].n, fg);
}
for (a=0; a<ylegmax; a++){
p[0].x = 5+(xlegmax*40);
                p[0].y = 10+(a*40);
p[0].z = 0;
crt3DImageString(im, f, p, yleg[a].n, fg);
}

}

void crt2DAxis(gdImagePtr im, int xmax, int c)
{
```

```
    crt3DPoint p[2];
    int a;

/*    p[0].x = -180; p[0].y = 75; p[0].z=-40;
    p[1].x = 10+30*xmax-180; p[1].y = 75; p[1].z=-40;
    crt3DImageLine(im, p, c);

    p[0].x = -180; p[0].y = 45; p[0].z=-40;
    p[1].x = 10+30*xmax-180; p[1].y = 45; p[1].z=-40;
    crt3DImageLine(im, p, c);
*/

    for (a=1; a<=xmax; a++)
    {
     p[0].x = 30*a-180; p[0].y = 80; p[0].z=-40;
     p[1].x = 30*a-180; p[1].y = 45; p[1].z=-40;
     crt3DImageDashedLine(im, p, c);
    }
}



void crt3DAxis(gdImagePtr im, int xmax, int ymax, int c)
{
crt3DPoint p[2];
int a;

p[0].x = 0; p[0].y = 0; p[0].z = 0;
p[1].x = 10+xmax*40; p[1].y = 0; p[1].z = 0;
crt3DImageLine(im, p, c);

p[0].x = 0; p[0].y = 0; p[0].z = 0;
p[1].x = 0; p[1].y = 10+ymax*40; p[1].z = 0;
crt3DImageLine(im, p, c);

p[0].x = 0; p[0].y = ymax*40; p[0].z = 0;
p[1].x = xmax*40; p[1].y = ymax*40; p[1].z = 0;
crt3DImageLine(im, p, c);

p[0].x = xmax*40; p[0].y = 0; p[0].z = 0;
p[1].x = xmax*40; p[1].y = ymax*40; p[1].z = 0;
crt3DImageLine(im, p, c);

for (a=1; a<=xmax; a++)
{
p[0].x = a*40; p[0].y = 0; p[0].z = 0;
p[1].x = a*40; p[1].y = 10+ymax*40; p[1].z = 0;
crt3DImageDashedLine(im, p, c);
}

for (a=1; a<=ymax; a++)
{
p[0].x = 0; p[0].y = a*40; p[0].z = 0;
p[1].x = 10+xmax*40; p[1].y = a*40; p[1].z = 0;
crt3DImageDashedLine(im, p, c);
}

}


void crt3DImageLine(gdImagePtr im, crt3DPointPtr p, int c)
{
gdPoint pn[2];


crt3Dconvert2D(p, pn, 2);
gdImageLine(im, pn[0].x, pn[0].y, pn[1].x, pn[1].y, c);
}

void crt3DImageDashedLine(gdImagePtr im, crt3DPointPtr p, int c)
{
gdPoint pn[2];
```

```
crt3Dconvert2D(p, pn, 2);
gdImageDashedLine(im, pn[0].x, pn[0].y, pn[1].x, pn[1].y, c);
}

void crt3DImageTraceLines(gdImagePtr im, crt3DPointPtr p, int n, int c)
{
        gdPoint pn[n];
        int a;

        crt3Dconvert2D(p, pn, n);
for (a=0; a<(n-1); a++){
         gdImageLine(im, pn[a].x, pn[a].y, pn[a+1].x, pn[a+1].y, c);
}
}

void crt3DImageFilledPolygon(gdImagePtr im, crt3DPointPtr p, int n, int c)
{
gdPoint pn[n];

crt3Dconvert2D(p, pn, n);
gdImageFilledPolygon(im, pn, n, c);
}

void crt3DImagePolygon(gdImagePtr im, crt3DPointPtr p, int n, int c)
{
gdPoint pn[n];
int a, dx, dy;

crt3Dconvert2D(p, pn, n);
gdImagePolygon(im, pn, n, c);
}

void crt3DImageCube(gdImagePtr im, crt3DPointPtr p, int b, int h, int f, int l)
{

crt3DPoint f0[4], f1[4], f2[4];

f0[0].x= b+p[0].x; f0[0].y= 0+p[0].y; f0[0].z= h+p[0].z;
f0[1].x= b+p[0].x; f0[1].y= b+p[0].y; f0[1].z= h+p[0].z;
f0[2].x= 0+p[0].x; f0[2].y= b+p[0].y; f0[2].z= h+p[0].z;
f0[3].x= 0+p[0].x; f0[3].y= 0+p[0].y; f0[3].z= h+p[0].z;
crt3DImageFilledPolygon(im, f0, 4, f);
        crt3DImagePolygon(im, f0, 4, l);

f1[0].x= 0+p[0].x; f1[0].y= b+p[0].y; f1[0].z= 0+p[0].z;
f1[1].x= 0+p[0].x; f1[1].y= b+p[0].y; f1[1].z= h+p[0].z;
f1[2].x= b+p[0].x; f1[2].y= b+p[0].y; f1[2].z= h+p[0].z;
f1[3].x= b+p[0].x; f1[3].y= b+p[0].y; f1[3].z= 0+p[0].z;
crt3DImageFilledPolygon(im, f1, 4, f);
        crt3DImageTraceLines(im, f1, 2, l);

f2[0].x= b+p[0].x; f2[0].y= 0+p[0].y; f2[0].z= 0+p[0].z;
f2[1].x= b+p[0].x; f2[1].y= 0+p[0].y; f2[1].z= h+p[0].z;
f2[2].x= b+p[0].x; f2[2].y= b+p[0].y; f2[2].z= h+p[0].z;
f2[3].x= b+p[0].x; f2[3].y= b+p[0].y; f2[3].z= 0+p[0].z;

crt3DImageFilledPolygon(im, f2, 4, f);
        crt3DImageTraceLines(im, f2, 4, l);

}

void crt3DImageTriangleRight(gdImagePtr im, crt3DPointPtr p, int b,
 int h, int f, int l)
{

crt3DPoint f0[4], f1[4], f2[4];

f0[0].x= b+p[0].x; f0[0].y= 0+p[0].y; f0[0].z= h+p[0].z;
f0[1].x= b+p[0].x; f0[1].y= b+p[0].y; f0[1].z= h+p[0].z;
f0[2].x= 0+p[0].x; f0[2].y= 0+p[0].y; f0[2].z= h+p[0].z;
crt3DImageFilledPolygon(im, f0, 3, f);
        crt3DImagePolygon(im, f0, 3, l);
```

```
f2[0].x= b+p[0].x; f2[0].y= 0+p[0].y; f2[0].z= 0+p[0].z;
f2[1].x= b+p[0].x; f2[1].y= 0+p[0].y; f2[1].z= h+p[0].z;
f2[2].x= b+p[0].x; f2[2].y= b+p[0].y; f2[2].z= h+p[0].z;
f2[3].x= b+p[0].x; f2[3].y= b+p[0].y; f2[3].z= 0+p[0].z;
crt3DImageFilledPolygon(im, f2, 4, f);
        crt3DImageTraceLines(im, f2, 4, l);

}

void crt3DImageTriangleLeft(gdImagePtr im, crt3DPointPtr p, int b,
 int h, int f, int l)
{

crt3DPoint f0[4], f1[4], f2[4];

f0[0].x= 0+p[0].x; f0[0].y= b+p[0].y; f0[0].z= h+p[0].z;
f0[1].x= b+p[0].x; f0[1].y= b+p[0].y; f0[1].z= h+p[0].z;
f0[2].x= 0+p[0].x; f0[2].y= 0+p[0].y; f0[2].z= h+p[0].z;
crt3DImageFilledPolygon(im, f0, 3, f);
        crt3DImagePolygon(im, f0, 3, l);

f1[0].x= 0+p[0].x; f1[0].y= b+p[0].y; f1[0].z= 0+p[0].z;
f1[1].x= 0+p[0].x; f1[1].y= b+p[0].y; f1[1].z= h+p[0].z;
f1[2].x= b+p[0].x; f1[2].y= b+p[0].y; f1[2].z= h+p[0].z;
f1[3].x= b+p[0].x; f1[3].y= b+p[0].y; f1[3].z= 0+p[0].z;
crt3DImageFilledPolygon(im, f1, 4, f);
        crt3DImageTraceLines(im, f1, 2, l);

}
```

C.5 Criador de gráficos em 3D

```
#include <stdio.h>
#include <stdlib.h>
#include <strings.h>
#include <gd.h>
#include <gdfonts.h>
#include "crt3D.h"

#define DEBUG
#define UNIT "bps"

/*********************************************/
/* Defining foreground and background colors    */
/* INVERT 1 => fg=white, bg=black  */
/* INVERT 0 => fg=black, bg=white  */
/*********************************************/
#define INV 0

/* Maximum number of objects for X axis */
#define MAXX 19


void CreateHisto(crt3DMatrixTypePtr inm, int n, int mx, int l,
crt2DObjectTypePtr *o, int on, int *omax, double *vmin, double *vmax)
{
    int a, b;
    int achou = 0;
    crt2DObjectTypePtr ol;
    char tmp[20];

    ol=*o=calloc(mx,sizeof(crt2DObjectType));

    *omax=0;
    *vmin=*vmax=0;
```

```
    for (a=n-1; a>=0; a--)
    {
if ( on==1 ) strcpy(tmp,inm[a].o1);
if ( on==2 ) strcpy(tmp,inm[a].o2);
achou=0;
for (b=0; b<*omax; b++)
{
    printf("Histo: strcmp(%s,%s) *omax=%d\n",ol[b].o,tmp,*omax);
    if ( 0 == strcmp(ol[b].o,tmp) )
    {
ol[b].v=ol[b].v+inm[a].v;
achou=1;
    }
}
printf("Histo:if 0==%d AND %d<%d add ol\n",achou,*omax,mx);
if ( (! achou) && (*omax < mx ) )
{
    strcpy(ol[*omax].o,tmp);
    ol[*omax].v=inm[a].v;
    printf("Histo: *omax=%d, ol[%d].o=%s, ol[%d].v=%.2f\n",
*omax, *omax, ol[*omax].o, *omax, ol[*omax].v);
    *omax=*omax+1;
}
    }
    *vmin=ol[*omax].v;
    *vmax=ol[0].v;
    printf("Histo:max=%d\n",*omax);
    printf("Histo:vmin=%.2f,vmax=%.2f\n",*vmin,*vmax);
}



int LoadInputFile(char *s, crt3DMatrixTypePtr *matr)
{
FILE *in;
int count, a, sorted, uniq;
char line[256];
char *temp, *temp2, *temp3;
double v;
crt3DMatrixType ora;
crt3DMatrixTypePtr inm;

        in = fopen(s, "r");
        if (!in)
        {
           fprintf(stderr, "ERROR: File %s does not exist !\n",s);
           exit(1);
        }
count=0;
while ( fgets(line,256,in)) { count++;}
*matr=inm=calloc(count,sizeof(crt3DMatrixType));
rewind(in);

count=0;
while ( fgets(line,256,in))
        {
         temp = strtok(line, "\t ");
                if ( temp == NULL ){
                        fprintf(stderr, "ERROR:%s:%d:Without obj1\n",s,count);
                        exit(1);
                }
#ifdef DEBUG
                 printf("OBJ1=%s,",temp);
#endif

                temp2 = strtok(NULL, "\t ");
                if ( temp2 == NULL ){
                        fprintf(stderr, "ERROR:%s:%d:Without obj2\n",s,count);
                        exit(0);
                }
#ifdef DEBUG
                 printf("OBJ2=%s,",temp2);
```

```
#endif

                temp3 = strtok(NULL, "\t ");
                if ( temp == NULL ){
                        fprintf(stderr, "ERROR:%s:%d:Without value\n",s,count);
                        exit(0);
                }
#ifdef DEBUG
                 printf("VALUE=%f\n",inm[count].v);
#endif

uniq=1;
for (a=0; a<count; a++)
{
if ( (0 == strcmp(inm[a].o1,temp)) &&
     (0 == strcmp(inm[a].o2,temp2)) )
{
inm[a].v=inm[a].v+atof(temp3);
uniq=0;
}
}
if (uniq)
{
strcpy(inm[count].o1,temp);
strcpy(inm[count].o2,temp2);
inm[count].v=atof(temp3);
count++;
}
}
/* Sorting values */
sorted=0;
while (! sorted)
{
sorted=1;
for (a=0; a<(count-1); a++)
{
if (inm[a].v > inm[a+1].v){
ora.v=inm[a].v;
inm[a].v=inm[a+1].v;
inm[a+1].v=ora.v;
strcpy(ora.o1,inm[a].o1);
strcpy(inm[a].o1,inm[a+1].o1);
strcpy(inm[a+1].o1,ora.o1);
strcpy(ora.o2,inm[a].o2);
strcpy(inm[a].o2,inm[a+1].o2);
strcpy(inm[a+1].o2,ora.o2);
sorted=0;
}
}
}

#ifdef DEBUG
printf ("VERIFICAR REPETICOES Sorted \n");
for (a=0; a<count; a++){
printf("OBJ1=%s,",inm[a].o1);
                printf("OBJ2=%s,",inm[a].o2);
                printf("VALUE=%f\n",inm[a].v);
}
#endif

return(count);
close(in);
}


int main(int argc, char **argv)
{
        FILE *pngout;
int ilines, objsmax;
double vmin, vmax;
crt3DMatrixTypePtr inmatrix = NULL;
crt2DObjectTypePtr objs;
char unit[20];
```

```
        gdImagePtr im;
/* Allocate 3D point */
crt3DPoint points[4];
/* Colors */
int Color[LEVELS];
        int red, green, blue, white, black, fg, bg;
/* Temp variables */
int a, b, x, y;

        if (argc != 3)
{
                fprintf(stderr, "Usage: %s <filein> <fileout>\n",argv[0]);
                fprintf(stderr, "  filein  = Input Text File \n");
                fprintf(stderr, " fileout = Output PNG Image file\n");
                exit(1);
        }

ilines=LoadInputFile(argv[1], &inmatrix);
CreateHisto(inmatrix, ilines, MAXX, LEVELS, &objs, 2, &objsmax,
&vmin, &vmax);

        /* Allocate the image: 640 pixels across by 480 pixels tall */
        im = gdImageCreate(640, 480);

        /* Allocate Colors */
bg = gdImageColorAllocate(im, 255*(1-INV), 255*(1-INV), 255*(1-INV));
fg = gdImageColorAllocate(im, 255*INV, 255*INV, 255*INV);
white = gdImageColorAllocate(im, 255, 255, 255);
black = gdImageColorAllocate(im, 0, 0, 0);

crtAllocateColors(im, Color, LEVELS);

crt2DAxis(im, objsmax, fg);

sprintf(unit,UNIT);
crtDrawHisto3D(im, Color, LEVELS, bg, fg, black, white, gdFontSmall,
  objs, objsmax , vmin, vmax, unit);

pngout = fopen("test.png", "wb");
/*gdImagePng(im, pngout);*/
gdImagePng(im, pngout);
fclose(pngout);
gdImageDestroy(im);

}
```

Apêndice D

Termos Utilizados e Definições

Este apêndice tem o objetivo de ilustrar alguns termos e definições utilizados neste trabalho.

D.1 Velocidades dos circuitos

Uma definição importante é quanto as velocidades dos enlaces. Muita confusão é feita em relação a quantidade que representa esta velocidade em termos de *bits* por segundo. As representações em K (Kilo), M (Mega), G (Giga) e T (Tera) representam respectivamente: 1000, $1 * 10^6$, $1 * 10^9$ e $1 * 10^{12}$. Logo um enlace com capacidade de transmissão de 2Mbps é o equivalente a 2000Kbps ou 2x10^6bps. A confusão é feita com as unidades de armazenamento de memória que são multiplos de 10^2=1024 ([102]).

D.2 Características da rede Internet

A rede Internet tem as seguintes características:

- Não existe alocação de recursos fim-a-fim;
- A entrega de um pacote ao destino não é garantida pela rede;
- O tempo em que o pacote irá chegar ao destino não é garantido;
- Não pode garantir a velocidade de transmissão dos pacotes;
- Não provê controle de fluxo dos pacotes transmitidos;
- O percurso em que o pacote é enviado depende de cada ponto por onde ele passa, muitas vezes este percurso é alterado durante o envio dos pacotes ao destino;

- Não existe autenticação dos pacotes;
- Não provê encriptação dos dados;

D.3 Sistema Autonomo e ASN

Quando uma rede de uma determinada instituição cresce, geralmente ela necessita de própria política de roteamento. Esta política definirá as tabelas de roteamento, as quais contém as redes IP destino e suas portas de saída (*gateways*). Geralmente esta política é implementada por algum tipo de protocolo interno (ex: OSPF, IGRP, EIGRP, ...). A instituição que possui seu próprio conjunto de redes e sua própria política de roteamento interna é geralmente chamada de Sistema Autônomo ou *Autonomous System* (*AS*).

Para que a rede desta instituição possa estabelecer comunicação com outras redes, criando assim políticas de troca de tabelas de roteamento entre elas, é necessário a criação de um **ASN** (*Autonomous System Number*). A criação de um ASN só tem sentido para instituições com grande número de redes IP e que possuam comunicação com pelo menos duas instituições diferentes. A política de troca de tabelas entre dois *AS* é geralmente implementada pelo protocolo de roteamento *BGP*(*Border Gateway Protocol*).

Ex: A RedeRio tem o seu ASN igual a 2715, a Embratel tem o seu ASN igual a 4230, a RNP tem o ASN 1916.

D.4 Endereçamento Classless

Como o *classfull* não era uma forma "economica" de utilizar as tabelas de roteamento, não permitindo sumarização, o *classless* foi consebido. Antes dos roteadores utilizarem *classless* eles só suportavam as máscaras de rede correspondentes a sua classe. Logo para uma rede local Ethernet ligada a uma porta do roteador, utilizando uma rede classe A o roteador apenas poderia associar a esta interface a máscara 255.0.0.0. Com a criação do *classless* é possível por exemplo dividir o domínio de uma rede em menores pedaços (*subnets* ou subredes) ou agrupar em pedaços maiores (*supernets* ou superredes). Este trabalho utiliza classless, logo o tamanho das redes A, B e C na forma clássica não são válidos, o seu tamanho vai ser dado pela máscara. A representação da máscara pode ser feita também da forma abreviada, na qual representa o número de bits em 1 da máscara na forma binária.

Ex1: A máscara 255.255.0.0 pode ser representada por /16;

Ex2: A rede 146.164.0.0 antes da criação do *classless* não precisava do acompanhamento da máscara para a descrição do seu tamanho pois ela pertencia a classe B, logo teria a sua máscara igual a 255.255.0.0 . Com o *classless* é preciso que todas as redes acompanhem a sua máscara para determinar o seu tamanho;

Ex3: Rede 201.10.0.0 originalmente classe C teria o tamanho de até 256 números, logo sua máscara seria 255.255.255.0 (ou /24). Com a utilização do *classless* é possível subdividir esta rede por exemplo em pedaços de 64 números, para isso é necessário utilizar a máscara 255.255.255.192 (ou /26). Os quatro pedaços da rede original ficariam representados por 201.10.0.0/26, 201.10.0.64/26, 201.10.0.128/26 e 201.10.0.192/26.

Também é possível agregar redes menores, ou seja, utilizar uma máscara que acomode maior quantidade de números.

Ex: Para o conjunto de redes 201.10.4.0/24, 201.10.5.0/24, 201.10.6.0/24, 201.10.7.0/24 é possível representá-las por 201.10.4.0/22 apenas. Estes agregados de rede também são chamados de sumários ou blocos CIDR (*Classless Inter Domain Routing*).

Uma lista completa das máscaras possíveis e suas abreviações estão na tabela D.1.

Tabela D.1: Máscaras disponíveis

255.255.255.255	/32 *	255.254.0.0	/15
255.255.255.254	/31	255.252.0.0	/14
255.255.255.252	/30	255.248.0.0	/13
255.255.255.248	/29	255.240.0.0	/12
255.255.255.240	/28	255.224.0.0	/11
255.255.255.224	/27	255.192.0.0	/10
255.255.255.192	/26	255.128.0.0	/9
255.255.255.128	/25	255.0.0.0	/8
255.255.255.0	/24	254.0.0.0	/7
255.255.254.0	/23	252.0.0.0	/6
255.255.252.0	/22	248.0.0.0	/5
255.255.248.0	/21	240.0.0.0	/4
255.255.240.0	/20	224.0.0.0	/3
255.255.224.0	/19	192.0.0.0	/2
255.255.192.0	/18	128.0.0.0	/1
255.255.128.0	/17	0.0.0.0	/0 **
255.255.0.0	/16		

*Geralmente utilizado para identificar interfaces lógicas;
**Máscara utilizada para englobar todas as redes (default);

Para maiores detalhes consulte a RFC???? e [3]

D.5 Glossário

ASN ou AS : *Autonomous System Number*, termo utilizado para designar um número de 16 bits utilizado em tabelas de roteamento para designar uma rede que tem o seu próprio sistema de roteamento;

Backbone : Termo utilizado para designar uma infraestrutura de rede, onde o principal foco é prover conexão a todos os outros pontos da rede. Geralmente o Backbone tem grande abrangência e é constituido de uma série de roteadores e equipamento de rede. Um Backbone IP é uma infraestrutura de rede que provê conexões através do protocolo IP;

IETF : *Internet Engineering Task Force* é um órgão ligado ao ISOC (*Internet Society*) que tem como objetivo estabelecer grupos de trabalho em assuntos diversos da Internet, estes grupos de trabalho criam recomendações e padrões publicadas através das RFCs;

Internet-Drafts : São documentos que precedem uma RFC (Request For Comments) e que são levados a votação e discussão pelos grupos de trabalho do IETF. As Internet-Drafts são documentos que não duram muito tempo, ou virarão RFCs, ou simplesmente serão removidas;

IOS : *Internet Operational System*, termo utilizado pela Cisco para designar o sistema operacional que roda nos roteadores Cisco. Geralmenete este a sigla IOS acompanha um número que identifica a sua versão (exemplo IOS12.0(3)S) ;

IP spoofing : Técnica utilizada para ataques a redes e máquinas, o IP origem é alterado para um endereço que não pertence a origem. Funciona como uma carta onde o remetente é falso;

MTU : *Maximum Transfer Unit* é designado em bytes para indicar qual é a máxima quantidade de dados possíveis de serem envidos por vez pelo meio de transporte;

NAT : *Network Address Translator* é utilizado para designar um processo de trocas de IPs utilizados em redes com endereçamento diferente do utilizado no encaminhamento normal (ex: redes privadas prefixo 10.0.0.0/8);

POS : *Packet Over Sonet*, utilizado para designar enlaces e interfaces que utilizam os protocolos do TCP/IP diretamente nos enlaces SONET ou SDH;

RFC : *Request For Comments*, utilizado para descrever uma recomendação, um protocolo ou uma funcionalidade utilizada(o) na Internet;

SLA : *Service Level Agreement* é o acordo em que a operadora de de algum serviço Internet realiza com o seu cliente. Geralmente o acordo *SLA* que o cliente faz com a operadora define uma série de parâmetros no qual indicam a qualidade do serviço. Por exemplo estes parâmetros podem ser: Atraso < 200ms e Disponibilidade > 99

access-list : Este termo é utilizado pelos roteadores Cisco para designar uma lista de acesso. Esta lista pode permitir ou filtrar acessos, identificados por IP ou por Porta (Serviço a nível de transporte);

buffers : Espaço em memória de dados para armazenamento temporário ou memória dedicada para armazenamento temporário;

cache : Termo utilizado para designar uma memória mais rápida dedicada para o acesso a informações que são frequêntemente consultadas na memória principal. Durante o funcionamento a memória cache fica com uma cópia do espaço em memória principal que é mais acessado;

checksum : Termo utilizado para indicar uma verificação de erro através do somatório dos valores;

default : Termo designado para identificar uma configuração ou procedimento padrão, ou para designar uma configuração original;

ethernet : Tecnologia utilizada para comunicação entre computadores em redes locais;

flags : Indicam um conjunto de bits, exemplo: TCP *flags* significa um conjunto de bits do cabeçalho do protocolo TCP;

flaps de roteamento : Termo utilizado para indicar que a tabela de roteamento deve ou está tendo uma variação na sua quantidade de entradas, ou linhas. Este *Flap* é causado geralmente quando uma rede IP destino cai e propaga a sua "retirada" para as tabelas de roteamento;

flow entry : Campo destinado para a descrição de um fluxo, este termo geralmente é utilizado para delimitar uma entrada de fluxo dentro de um pacote exportado pelo NetFlow;

flow-export : Processo realizado nos roteadores Cisco configurados com a funcionalidade NetFlow o qual envia informações de fluxo para um coletor de dados de fluxo previamente configurado;

Frame-Relay : Rede de transporte de dados que foi inventada para substituir a antiga tecnologia X.25. Sua principal utilização é em redes de grande distância. Esta rede é baseada em multiplexação estatística através dos *frames* e também permite negociar taxas de tranferência média e de pico;

hardware : Termo designado para identificar a parte física de um equipamento, ou seja, as peças que o constituem, ex: processador, memória, componentes eletrônicos, etc;

header : Cabeçalho. Geralmente para designar o formato de dados iniciais em um pacote ou arquivo;

hop : Termo designado para um "salto" em um caminho de roteadores, ou seja, cada hop identifica uma passagem por um roteador. Utilizado para designar uma passagem por um dispositivo que roteia pacotes IP, geralmente um roteador;

hub : Concentrador, em redes locais identifica um equipamento que concentra portas com tecnologia ethernet, com a característica de apenas repetir o sinal para as demais portas;

lookups : Em redes de roteadores significa o processo de procura em uma tabela. Geralmente tabelas de roteamento ou listas de acesso;

nexthop : Utilizado para designar o próximo roteador a ser entregue algum pacote IP, geralmente o nexthop identifica o "próximo destino IP" de acesso a uma rede em uma tabela de roteamento;

patches : São um conjunto de alterações no intuito de corrigir ou modificar o programa (conjunto de programas ou Sistema Operacional) original;

peering : Termo utilizado para designar uma comunicação entre dois AS;

raw : Termo utilizado para designar um dado ou uma informação não tratado, ou seja um dado ou informação que não sofreu alterações;

reset : Significa o término de um processo ou o seu desligamento para depois reinicia-lo do estado inicial;

shared memory : Técnica utilizada para o compartilhamento de memória principal, também utilizado para comunicação entre processos;

socket : Ponto de conexão, geralmente designado a pontos de comunicação;

software : Tudo no equipamento que não pertence a constituição física do equipamento, geralmente se refere a um programa;

standalone : Termo utilizado geralmente para designar um processo ou um programa que roda em um servidor sem a necessidade de nenhum outro;

switches : Termo em inglês para designar equipamentos comutadores, em redes locais é utilizado para designar um equipamento que realiza comutação entre as suas portas ethernet, não apenas um repetidor como um *hub*;

time-out : Expressão utilizada para indicar o tempo em que um certo processo ou procedimento irá esperar até receber algum resultado ou resposta. Caso não exista uma resposta até um certo *time-out* este é considerado expirado;

web farm : Serviço Internet que alguns provedores de serviço internet fornecem. A principal idéia é disponibilizar um espaço físico no qual é possível se conectar uma máquina. Para acesso a rede é disponibilizado uma porta com tecnologia Ethernet que pode ser especificada para garantir certos parâmetros de taxas de transferência como taxa média e de pico;

working group : Termo utilizado para definir grupos de trabalho em instituições, geralmente utilizado nas organizações que padronizam e recomendam na Internet como o IETF e o RIPE;

Referências Bibliográficas

[1] Stallings W., *High-Speed Networks TCP/IP and ATM Design Principles*, Prentice Hall, 1999.

[2] K. Claffy, *Internet Traffic Caracterization*, Ph.D. thesis, University of California, San Diego, 1994.

[3] Christian Huitema, *Routing in the Internet*, Prentice Hall, 1995.

[4] RFCs, Request For Comments
`http://www.rfc-editor.org/overview.html`

[5] Soares, L. F. G.,Lemos G., Colcher S., *Redes de computadores: das LANs MANs e WANs às redes ATM*, Campus, 1995.

[6] IPPM, *Internet Protocol Performance Metrics (IETF Working Group)*, `http://www.ietf.org/html.charters/ippm-charter.html`.

[7] RTFM, *Realtime Traffic Flow Measurement (IETF Working Group)*, `http://www.ietf.org/html.charters/rtfm-charter.html` `http://www.auckland.ac.nz/net/Internet/rtfm`.

[8] TEWG, *Internet Traffic Engineering (IETF Working Group)*, `http://www.ietf.org/html.charters/tewg-charter.html`

[9] BMWG, *Benchmarking Methodology (IETF Working Group)*, `http://www.ietf.org/html.charters/bmwg-charter.html`

[10] RMONMIB, *Remote Network Monitoring (IETF Working Group)*, `http://www.ietf.org/html.charters/rmonmib-charter.html`

[11] INTSERV, *Integrated Services (IETF Working Group)*, `http://www.ietf.org/html.charters/intserv-charter.html`

[12] DIFFSERV, *Differentiated Services (IETF Working Group)*, `http://www.ietf.org/html.charters/diffserv-charter.html`

[13] MPLS, *Multiprotocol Label Switching (IETF Working Group)*, http://www.ietf.org/html.charters/mpls-charter.html

[14] IETF, *Internet Engineering Task Force*, http://www.ietf.org/overview.html

[15] IAB, *Internet Architecture Board*, http://www.iab.org/overview.html

[16] SecurityFocus homepage, http://www.securityfocus.com

[17] DDoS, *Distributed Denied Of Services*, http://www.opensourcefirewall.com/ddos_whitepaper_copy.html

[18] *IP spoofing white paper*, http://staff.washington.edu/dittrich/papers/IP-spoof-1.txt

[19] *Egress Filtering*, SANS, http://www.sans.org/y2k/egress.htm

[20] *Voice Technologies For IP and Frame Relay Networks*, http://www.motorola.com/MIMS/ISG/mnd/papers/ voice_technologies_for_ip_and_frame_relay_networks.html

[21] *Differentiated Services for the Internet*, http://www.hut.fi/ msisomak/diffserv.html

[22] S. Floyd and V. Jacobson, *The Synchronization of Periodic Routing Messages*, IEEE/ACM Transactions on Networking, pp. 122-136, Abril de 1994.

[23] Vern Paxson, *Measurements and Analysis of End-to-End Internet Dynamics*, Ph.D. thesis, University of California, Berkeley, 1997.

[24] K. Thompson, G. Miller, and R. Wilder, *Wide Area Internet Traffic Patterns and Characteristics* IEEE Network, Nov 1997. http://www.vbns.net/presentations/papers/MCItraffic.ps

[25] Richard Stevens, *TCP/IP Illustrated, Volume 1*

[26] NGI, *Next Generation Internet*, CAIDA, http://www.caida.org/projects/ngi

[27] IEC, *Internet Engineering Curriculum Repository*, CAIDA,
http://www.caida.org/projects/iec

[28] *Internet Atlas Project*, CAIDA,
http://www.caida.org/projects/internetatlas

[29] ISMA, *Internet Statistic and Metrics Analysis*, CAIDA,
http://www.caida.org/outreach/isma

[30] NIMI, *National Internet Measurement Infrastructure*, NLANR,
http://www.ncne.nlanr.net/nimi

[31] PITAC, *Performance Measurement Project*, NLANR,
http://www.ncne.nlanr.net/research/pitac

[32] PMA, *Passive Measurement and Analysis*, NLANR,
http://moat.nlanr.net/PMA

[33] NAI, *Network Analysis Infrastructure*, NLANR,
http://moat.nlanr.net/NAI

[34] NAT, *Network Analysis Times*, NLANR,
http://moat.nlanr.net/NATimes

[35] TBIT, *the TCP Behavior Indentification Tool*,
http://www.aciri.org/tbit

[36] RED , *Random Early Detection - Queue Management*, ACIRI,
http://www.aciri.org/floyd/red.html

[37] WCCP, *Web Cache Control Protocol*, CISCO,
http://www.cisco.com/univercd/cc/td/doc/product
/software/ios111/ca111/wccp.htm

[38] ITA , *The Internet Traffic Archive*, LBNL,
http://ita.ee.lbl.gov/index.html

[39] *Test-Traffic Working Group*, RIPE,
http://www.ripe.net/ripe/wg/tt

[40] TF-ETM, *European Traffic Measurements Task Force*, TERENA,
http://www.terena.nl/task-forces/tf-etm

[41] TF-TANT, *Testing Advanced Networking Technologies Task Force*, TERENA & DANTE,
http://www.terena.nl/task-forces/tf-tant

[42] *Differentiated Services*, TF-TANT *Work Item*,
http://www.cnaf.infn.it/ ferrari/tfng/ds

[43] *QoS Monitoring*, TF-TANT *Work Item*,
http://www.cnaf.infn.it/ ferrari/tfng/qosmon

[44] *Flow-based Monitoring Analysis*, TF-TANT *Work Item*,
http://www.switch.ch/tf-tant/floma

[45] *Route Monitoring*, TF-TANT *Work Item*,
http://www.switch.ch/tf-tant/rm

[46] WAND, *Waikato Applied Network Dynamics*, Universidade de Waikato, NZ,
http://atm.cs.waikato.ac.nz/atm

[47] DAG *cards documentation*, Universidade de Waikato, NZ,
http://atm.cs.waikato.ac.nz/atm/docs

[48] IPMA, *Internet Performance Measurement and Analysis*,
http://www.merit.net/ipma

[49] PAM2000, *Passive & Active Measurement Workshop*, Hamilton, NZ, 3 e 4 abril de 2000
http:/pam2000.cs.waikato.ac.nz

[50] PAM2001, *Passive & Active Measurement Workshop*, Amsterdam, ???, 23 e 24 abril de 2001
http://www.ripe.net/pam2001

[51] Cisco White Paper, *NetFlow Services and Applications*,
http://www.cisco.com/warp/public/cc/cisco/mkt/ios/
netflow/tech/napps_wp.pdf;

[52] Arts++ software - supported by CAIDA,
http://www.caida.org/tools/utilities/arts;

[53] Cflowd software - supported by CAIDA,
http://www.caida.org/tools/measurement/cflowd.

[54] `libpcap` - a packet capture library, `http://www.tcpdump.org`

[55] Skitter software - supported by CAIDA,
`http://www.caida.org/tools/measurement/skitter`.

[56] Tcptrace homepage,
`http://www.tcptrace.org`

[57] `tcpdump libpcap` homepage, `http://www.tcpdump.org`

[58] Iperf homepage,
`http://dast.nlanr.net/Projects/Iperf`

[59] GNU Zebra homepage
`http://www.zebra.org`

[60] SourceForge Homepage
`http://www.sourceforge.net`

[61] MRT *Multi-Threaded Routing Toolkit* homepage
`http://www.mrtd.net`

[62] ARIN *American Registry for Internet Numbers* homepage
`http://www.arin.net`

[63] APNIC *Asia Pacific Network Information Centre* homepage
`http://www.apnic.net`

[64] University of Waikato, Hamilton, New Zealand,
`http://www.waikato.ac.nz`

[65] CAIDA, *Cooperative Association for Internet Data Analysis*,
`http://www.caida.org`

[66] UCSD, *University of California San Diego* , San Diego, California, EUA,
`http://www.ucsd.edu`

[67] CMU, *Carnegie Mellon University* , Pittsburgh, Pennsylvania, EUA
`http://www.cmu.edu`

[68] PSC, *Pittsburgh Supercomputing Center* , Pittsburgh, Pennsylvania, EUA
`http://www.psc.edu`

[69] NSF, *National Science Foundation*,
`http://www.nsf.gov`

[70] vBNS, *very high performance Backbone Network Service*,
`http://www.vbns.net`

[71] NLANR, *National Laboratory for Applied Network Research*,
`http://www.nlanr.net`

[72] DAST, *Distributed Applications Support Team - NLANR*, homepage e projetos,
`http://dast.nlanr.net` e `http://dast.nlanr.net/Projects`

[73] ACIRI, *AT&T Center for Internet Research*,
`http://www.aciri.org`

[74] NRG, *Network Research Group*, LBNL *Lawrence Berkeley National Laboratory*,
`http://www-nrg.ee.lbl.gov`

[75] Merit Network, `http://www.merit.net`

[76] NANOG, *North American Network Operators' Group*,
`http://www.nanog.org`

[77] RIPE NCC, *Réseaux IP Européens - Network Coordination Centre*,
`http://www.ripe.net/ripencc`

[78] TERENA, *Trans-European Research and Education Networking Association*,
`http://www.terena.nl`

[79] DANTE, *Connecting European Research*,
`http://www.dante.net`

[80] MIDS, *Matrix Information and Directory Services*,
`http://www.mids.org`

[81] RNP, Rede Nacional de Pesquisa,
`http://www.rnp.br`

[82] RedeRio, `http://www.rederio.br`

[83] CBPF-CAT, Centro Brasileiro de Pesquisas Físicas - Coordenação de Atividades Técnicas, `http://www.cbpf.br/cat`

[84] Rede ANSP, an Academic Network at São Paulo, `http://www.ansp.br`

[85] Internet via Embratel, `http://www.embratel.net.br/internet`

[86] UFRJ - Universidade Federal do Rio de Janeiro, `http://www.ufrj.br`

[87] The Linux Home Page, `http://www.linux.org`

[88] The FreeBSD Home Page, `http://www.freebsd.org`

[89] Perl language, `http://www.perl.org`

[90] GNU homepage, `http://www.gnu.org`

[91] J. Postel, *User Datagram Protocol*, RFC 768, Agosto de 1980.

[92] J. Postel, *Internet Protocol*, RFC 791, Setembro de 1981.

[93] J. Postel, *Internet Control Message Protocol*, RFC 792, Setembro de 1981.

[94] Defense Advanced Research Projects Agency, *Tranmission Control Protocol*, RFC 793, Setembro de 1981.

[95] S. Bradner, *Benchmarking Terminology for Network Interconnection Devices*, RFC 1242, Julho 1991.

[96] Mill, D., *Network Time Protocol* (version 3) *Specification, Implementation and Analisys*, RFC 1305, Março de 1992.

[97] J. Reynolds, J. Postel, *Assigned Numbers*, RFC 1700, Outubro de 1994.

[98] Bradner, S., *The Internet Standards Process – Revision 3*, BCP 9, RFC 2026, Outubro de 1996.

[99] N. Brownlee, *Traffic Flow Measurement: Experiences with NeTraMet*, *Informational*, RFC 2123, Março de 1997.

[100] P. Ferguson, D. Senie, *Network Ingress Filtering: Defeating Denial of Service Attacks which employ IP Source Address Spoofing*, *Informational*, RFC 2267, Janeiro de 1998.

[101] R. Mandeville, *Benchmarking Terminology for LAN Switching Devices*, Informational, RFC 2285, Fevereiro 1998.

[102] V. Paxson, G. Almes, J. Mahdavi, and M. Mathis, *Framework for IP Performance Metrics*, Informational, RFC 2330, Maio de 1998.

[103] K. Dubray, *Terminology for IP Multicast Benchmarking*, Informational, RFC 2432, Outubro de 1998.

[104] S. Bradner, J. McQuaid *Benchmarking Methodology for Network Interconnect Devices*, Informational, RFC 2544, Março de 1999.

[105] D. Newman, *Benchmarking Terminology for Firewall Performance*, Informational, RFC 2647, Agosto de 1999.

[106] M. Allman, V. Paxson, W. Stevens, *TCP Congestion Control*, Standards Track, RFC 2581, Abril de 1999.

[107] G. Almes, S. Kalidindi, M. Zekauskas, *A One-way Delay Metric for IPPM*, Standards Track, RFC 2679, Setembro de 1999.

[108] G. Almes, S. Kalidindi, M. Zekauskas, *A One-way Packet Loss Metric for IPPM*, Standards Track, RFC 2680, Setembro de 1999.

[109] G. Almes, S. Kalidindi, M. Zekauskas, *A Round-trip Delay Metric for IPPM*, Standards Track, RFC 2681, Setembro de 1999.

[110] N. Brownlee, *Traffic Flow Measurement: Meter MIB*, Standards Track, RFC 2720, Outubro de 1999.

[111] N. Brownlee, *RTFM: Applicability Statement*, Informational, RFC 2721, Outubro de 1999.

[112] N. Brownlee, C. Mills, G. Ruth, *Traffic Flow Measurement: Architecture*, Informational, RFC 2722, Outubro de 1999.

[113] N. Brownlee, *SRL: A Language for Describing Traffic Flows and Specifying Actions for Flow Groups*, Informational, RFC 2723, Outubro de 1999.

[114] S. Handelman, S. Stibler, N. Brownlee, G. Ruth *RTFM: New Attributes for Traffic Flow Measurement*, Experimental, RFC 2724, Outubro de 1999.

[115] J. Dunn, C. Martin, *Terminology for ATM Benchmarking*, Informational, RFC 2761, Fevereiro de 2000.

[116] R. Mandeville, J. Perser, *Benchmarking Methodology for LAN Switching Devices*, Informational, RFC 2889, Agosto de 2000.