Rodrigo Farias

Observador didático do funcionamento de algoritmos e protocolos de roteamento

São José – SC
setembro / 2010

Rodrigo Farias

Observador didático do funcionamento de algoritmos e protocolos de roteamento

Monografia apresentada à Coordenação do Curso Superior de Tecnologia em Sistemas de Telecomunicações do Instituto Federal de Educação, Ciência e Tecnologia de Santa Catarina (IF-SC) para a obtenção do diploma e, reconhecimento do grau de Tecnólogo em Sistemas de Telecomunicações.

Orientador:
Prof. Dr. Evandro Cantú

Co-orientador:
Prof. M. Eng. Eraldo Silveira e Silva

CURSO SUPERIOR DE TECNOLOGIA EM SISTEMAS DE TELECOMUNICAÇÕES
INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DE SANTA CATARINA

São José – SC
setembro / 2010

Monografia sob o título “Métodos e Ferramentas para facilitar a Aprendizagem de Algoritmos e Protocolos de Roteamento”, defendida por Rodrigo Farias e aprovada em 3 de setembro de 2010, em São José, Santa Catarina, pela banca examinadora assim constituída:

_____

Prof. Evandro Cantú, Dr.
Orientador

_____

Prof. Odilson Tadeu Vale, M. Eng.
IF-SC

_____

Prof. Tiago Semprebom, M. Eng.
IF-SC

"Não importa o tamanho da montanha,
ela não pode tapar o sol."

Provérbio chinês

Agradecimentos

Aos meus pais, Abrão Farias e Leonete Steinbach Farias, por viabilizarem a melhor herança que um filho pode receber: O conhecimento.

À minha noiva, Simara Sonaglio, pelo incentivo constante ao longo deste projeto e do curso.

Ao professor e amigo Evandro Cantú pelo acompanhamento e auxílio prestado ao longo deste projeto.

Aos professores Eraldo Silveira e Silva e Ederson Torresini pelo auxílio prestado neste projeto.

Aos professores do CST em Sistemas de Telecomunicações pelo aprendizado ao longo do curso.

Ao Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq) pelo fornecimento de bolsa de pesquisa PIBIC durante a execução deste projeto.

Ao IFSC por fornecer a estrutura necessária para a realização deste projeto.

Aos amigos e colegas do CST em Sistemas de Telecomunicações.

Resumo

Este projeto descreve um Observador Didático sobre o funcionamento de algoritmos e protocolos de roteamento utilizados em redes TCP/IP, visando facilitar a aprendizagem dos mesmos. A ferramenta apresentada permite mostrar, de maneira didática, como as tabelas de roteamento são construídas nos roteadores, assim como, observar como ocorrem modificações nestas tabelas quando acontecem mudanças físicas nos enlaces entre os roteadores.

Palavras-Chave: Algoritmos e protocolos de roteamento, arquitetura TCP/IP.

Abstract

This project describes a Didatic Observer for routing algorithms operations and routing protocols used in TCP / IP networks, to facilitate the learning of them. The presented tool allows to show, in a didactic way, such as routing tables are built in routers, as well as observe how changes occur in these tables when changes occur in the physical links between routers.

Keywords: Algorithms and routing protocols, TCP/IP architecture.

Sumário

Lista de figuras

Lista de tabelas

1 Introdução

O processo de aprendizagem dos algoritmos e protocolos de roteamento pode se tornar difícil sem a interpretação prática. Desta forma, para facilitar o aprendizado dos mesmos foi desenvolvido um Observador Didático do funcionamento de algoritmos e protocolos de roteamento utilizados em redes TCP/IP, visando facilitar a aprendizagem dos mesmos. A ferramenta apresentada permite mostrar, de maneira didática, como as tabelas de roteamento são construídas nos roteadores, assim como, observar como ocorrem modificações nestas tabelas quando acontecem mudanças físicas nos enlaces entre os roteadores.

A construção do Observador Didático faz uso de máquinas virtuais UML (*User-Mode Linux*). Uma máquina virtual é semelhante a um computador implementado a partir de *software*, que executa programas como um computador real, e é implementado a partir de um simulador de *hardware*.

Com máquinas virtuais é possível construir um cenário de redes, com vários roteadores interligados, como em um caso real. É importante ressaltar que toda a estrutura montada é executada em um único computador, entretanto, o comportamento da rede e as trocas de mensagens entre os roteadores são idênticas àquelas observadas em uma rede real.

1.1 Motivação

Os protocolos de roteamento são muito importantes para o encaminhamento correto de pacotes na internet. No entanto a compreensão dos mesmos é uma tarefa difícil. O principal motivo para o Observador Didático ser criado foi facilitar o estudo e compreensão dos protocolos de roteamento na prática.

Em cursos onde são ministradas disciplinas voltadas às redes de computadores geralmente são apresentados os algoritmos e protocolos de roteamento. Desta forma, com o Observador Didático os alunos destes cursos terão uma opção a mais para aprofundar os conceitos estudados.

1.2 Objetivos gerais

Este projeto tem por objetivo desenvolver uma ferramenta didática para o estudo do funcionamento de algoritmos e protocolos de roteamento utilizados em redes TCP/IP. A ferramenta desenvolvida Observador Didático visa facilitar a aprendizagem destes algoritmos e protocolos, mostrando de forma didática como as tabelas de roteamento são montadas na prática e como as mesmas são alteradas quando ocorrem mudanças nos enlaces que interligam os roteadores.

1.3 Organização do texto

Para entender melhor a utilidade deste projeto é preciso uma boa fundamentação teórica, sendo assim, no Capítulo 2 são apresentados conceitos sobre a camada rede, presente na arquitetura TCP/IP, e seus principais protocolos. Além disso, neste mesmo capítulo também são descritos os principais algoritmos de roteamento e os protocolos que os implementam.

No Capítulo 3 é apresentado rapidamente os resultados de simulações feitas com a ferramenta OPNET. No Capítulo 4 são descritas como funcionam as máquinas virtuais, a utilidade do protocolo SNMP para este projeto, as ferramentas necessárias para a execução do Observador Didático e como este foi construído. Também é descrito no Capítulo 4 as possíveis aplicações deste projeto. No Capítulo 5 são apresentadas as conclusões obtidas.

2 Fundamentação teórica

Atualmente, a rede de computadores mais abrangente no mundo é a Internet, cuja estrutura está baseada na arquitetura TCP/IP. A arquitetura TCP/IP é composta de cinco camadas de protocolos, sendo elas, do nível mais alto ao mais inferior: a camada de aplicação, a camada de transporte, a camada de rede, a camada enlace a camada física (CANTÚ, 2009). As camadas que compõe a pilha de protocolos TCP/IP são mostradas na Figura 1, as camadas enlace e física por atuarem muito próximas são comumente apresentadas juntas. Qualquer hospedeiro que venha a usar uma aplicação Internet deve obrigatoriamente fazer uso destas cinco camadas de protocolos, mesmo que indiretamente.

Figura 1 – Pilha de protocolos TCP/IP

A camada de Aplicação é responsável por interagir diretamente com o usuário. Tem a função de processar, gerar e receber a informação a ser transmitida/recebida pela rede.

A camada de Transporte tem por objetivo prover um canal lógico fim-a-fim de comunicação entre os processos de uma aplicação que rodam no cliente e os processos que rodam no servidor. Esta camada oferece dois tipos de serviços: um orientado a conexão, com transmissão garantida; e outro não orientado a conexão, com transmissão tipo “melhor esforço” (CANTÚ, 2009).

A camada de Rede tem como papel principal prover a comutação de pacotes entre os computadores de uma rede, assim envolvendo todos os computadores e roteadores entre o cliente e o servidor de uma aplicação. Esta camada organiza toda a informação vinda da camada superior em um formato legível para que todos os equipamentos de rede possam fazer o encaminhamento de forma correta pela rede. A camada de rede do TCP/IP faz uso de comutação de pacotes tipo datagrama, na qual cada pacote é encaminhado ao destino em

função do endereço do destinatário (endereço IP) e provê um serviço tipo “melhor esforço”, não orientado a conexão (CANTÚ, 2009) (KUROSE; ROSS, 2004).

A camada Enlace é responsável pelo encaminhamento da informação a ser transmitida de um ponto ao outro no mesmo enlace. Finalmente, a camada Física é a camada que é responsável diretamente pela transmissão/recepção dos bits pelo meio físico.

Neste capítulo estão descritos os principais conceitos para que se possa entender, de forma teórica, o funcionamento dos protocolos de roteamento. Na Seção 2.1 está brevemente descrito alguns conceitos e algumas das principais tarefas da camada rede, na Seção 2.2 tem-se a descrição dos principais algoritmos de roteamento e na Seção 2.3 são apresentados alguns dos principais protocolos de roteamento.

2.1 Camada Rede

Neste nível da estrutura TCP/IP tem-se importantes tarefas a serem executadas para que todas as aplicações de rede funcionem sem empecilhos. Para isso faz-se uso de alguns protocolos que por sua vez fazem uso de seus atributos, variáveis ou campos de cabeçalho para determinar para onde uma determinada informação deve ser encaminhada. A grosso modo, como anteriormente exemplificado no início deste capítulo, a camada rede provê um serviço do tipo “melhor esforço” onde os pacotes são encaminhados até seu destino sem garantia.

O principal protocolo da camada rede é o protocolo IP o qual define, entre outras funções, o formato do datagrama e as convenções de endereçamento na Internet. Nas subseções a seguir tem-se uma descrição detalhada de suas funcionalidades.

O Protocolo de Internet (IP), presente na camada Rede, é fundamental para o funcionamento dos demais protocolos da mesma camada e das camadas acima, Transporte e Aplicação. Dentre suas principais responsabilidades destacam-se o endereçamento, o formato dos datagramas definido como Unidade de Informação de Protocolo (PDU) e as relações com a camada superior através do encaminhamento das PDU’s da camada superior, chamadas de segmentos (KUROSE; ROSS, 2004). Neste projeto apenas o protocolo IP versão 4 será utilizado, que é a versão com maior uso na Internet.

O datagrama IP é conhecido por ser a unidade básica de transferência da Internet TCP/IP, ou seja, a PDU da camada de rede (KUROSE; ROSS, 2004). Este datagrama contém áreas de cabeçalho e de dados, como pode ser visto na Figura 2.

Na primeira linha do cabeçalho IP encontra-se a versão do protocolo IP, campos que identificam o comprimento do cabeçalho e do datagrama e o campo ToS (*Type of Service*), utilizado para implementar mecanismos de Qualidade de Serviço (QoS) na Internet. Na segunda linha encontram-se campos destinados à fragmentação do datagrama. Na terceira linha o TTL (*Time to Live*), o qual é usado, entre outras funções, para evitar que o datagrama fique circulando para sempre na rede, este campo é decrementado a cada roteador e quando chega a zero o mesmo é descartado. Na mesma linha encontra-se o campo protocolo, que identifica o protocolo da camada superior ao qual são destinados os dados deste datagrama, e o *checksum* do cabeçalho que é verificado a cada roteador. Nas duas seguintes linhas estão presentes os principais campos do datagrama IP, os endereços de origem e destino de 32 bits cada. O campo Opções serve para ampliar o cabeçalho caso seja necessário outras variáveis, raramente é utilizado(KUROSE; ROSS, 2004) (COMER, 2006).

Na área de dados pode-se encontrar conteúdo dirigido a algum protocolo ou ao uso de alguma aplicação de rede (KUROSE; ROSS, 2004) (COMER, 2006).

32 bits

0 a 3	4 a 7	8 a 15	16 a 18	19 a 31
Versão	Compr. Cabeç.	Tipo de Serviço (ToS)	Comprimento total do datagrama	
Identificador			*Flags*	Deslocamento de Fragmentação
Tempo de Vida (TTL)		Protocolo da camada superior	Verificação de Cabeçalho (*checksum*)	
Endereço IP de origem				
Endereço IP de destino				
Opções (se houver)				
Dados				

Figura 2 – Formato do datagrama IP

Outro protocolo da camada de Rede é o protocolo ICMP que é responsável por mensagens de controle, principalmente por mensagens de erro, o qual é utilizado tanto por hospedeiros quanto por roteadores (KUROSE; ROSS, 2004). Nas subseções a seguir serão descritos alguns protocolos importantes da Camada Rede.

2.1.1 Protocolo ICMP

O protocolo ICMP não é considerado um protocolo de nível superior à camada rede. Ele é considerado parte obrigatória do IP. As mensagens ICMP são encapsuladas no datagrama IP porque podem ter de passar por vários roteadores até chegarem aos seus destinos, por isso o ICMP é separado do IP e tratado como conteúdo do datagrama IP (COMER, 2006). As mensagens ICMP são transportadas dentro do datagrama IP conforme visto na Figura 3:

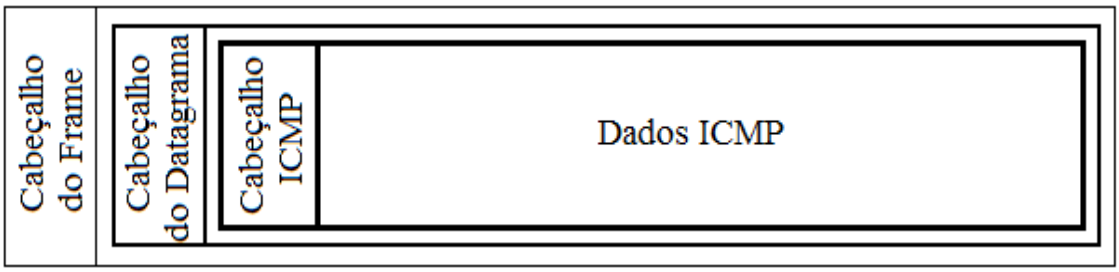


Figura 3 – Encapsulamento da mensagem ICMP

O cabeçalho ICMP contém três campos principais que podem ser encontrados em todas as mensagens ICMP: um campo TIPO de mensagem com 8 bits, que identifica do que se trata a mensagem; um campo CÓDIGO também com 8 bits, que contém detalhes do TIPO da mensagem; e um campo CHECKSUM de 16 bits, para conferência dos dados da mensagem ICMP. O cabeçalho ICMP pode conter mais campos, porém estes variam de acordo com o TIPO da mensagem (COMER, 2006). Na Tabela 1 são apresentados os principais valores do campo TIPO.

As mensagens *Echo Request* e *Echo Reply* do protocolo ICMP são constantemente utilizadas na área de gerência de redes para verificar se um determinado destino na rede está ativo e acessível (TANENBAUM, 2003). A ferramenta mais utilizada para esta tarefa é o *software* PING. Para exercer sua tarefa, a ferramenta PING envia uma mensagem *Echo Request* para um determinado destino e espera receber uma mensagem *Echo Reply*, caso a

receba o destino está ativo e acessível, caso não receba nada ou receba uma mensagem de erro constata-se que um possível problema está ocorrendo. Este problema pode ser uma simples perda de pacote, um problema de conectividade na rede ou o destino não está ativo. Normalmente a ferramenta PING envia uma mensagem Echo Request em um intervalo de tempo regular para facilitar uma possível estatística de pacotes perdidos.

Campo TIPO	Mensagem ICMP
0	Echo Reply
3	Destination Unreachable
4	Source Quench
5	Redirect
8	Echo Request
9	Router Advertisement
10	Router Solicitation
11	Time Exceeded
12	Parameter Problem

Tabela 1 – Principais mensagens ICMP descritas no campo TIPO (TANENBAUM, 2003) (COMER, 2006).

A mensagem *Destination Unreachable* é comumente utilizada por roteadores para indicar que o destino desejado não pode ser alcançado. Nesta mensagem, através do campo CÓDIGO, é detalhado o motivo deste problema (COMER, 2006). Os motivos mais comuns são: na tabela de encaminhamento não consta a rede de destino; a mensagem enviada não pode ser processada corretamente devido a ausência de um determinado protocolo; a porta de destino está inalcançável ou a mensagem enviada não pode ser fragmentada.

Para corrigir o problema da perda de pacotes em um *buffer* de entrada por falta de espaço, a mensagem *Source Quench* pode ser enviada à origem dos pacotes recebidos. Esta mensagem indica que a transmissão de pacotes para o respectivo destino deve ser desacelerada (TANENBAUM, 2003).

A mensagem *Redirect* é utilizada quando um roteador recebe um datagrama para encaminhar sendo que ele sabe que não é o roteador correto para encaminhar este datagrama, mas sim o seu vizinho. O roteador então envia uma mensagem *Redirect* à origem do datagrama pedindo para que este altere sua tabela de encaminhamento incluindo o seu roteador vizinho como destino para a respectiva rede de destino do datagrama. Como as

tabelas de roteamento raramente mudam e como os roteadores que usam protocolos de roteamento mantém suas rotas atualizadas, as mensagens *Redirect* raramente são usadas entre roteadores. Tais mensagens são comumente usadas entre um roteador e um *host* quando há mais de um roteador numa rede. Um *host* quando inicia seu serviço de rede, normalmente só tem conhecimento de um roteador na rede, o *default gateway*. Através de mensagens *Redirect* este *host* descobrirá novas rotas caso tenha mais de um roteador na sua rede local (COMER, 2006).

As mensagens *Router Advertisement* podem ser utilizadas por roteadores para divulgar periodicamente, se configurados assim, na sua rede local a sua presença e para responder às mensagens *Router Solicitation*. Nestas mensagens podem conter a divulgação de diversos outros roteadores presentes na sua tabela de roteamento (RFC 1256, 1991).

A mensagem *Router Solicitation* pode ser usada por um *host* sem rota *default*. Esta mensagem é normalmente enviada apenas algumas vezes no momento em que o *host* inicia seu serviço de rede. O *host* espera receber uma ou mais mensagens *Router Advertisement* em resposta (RFC 1256, 1991). As mensagens *Router Advertisement* e *Router Solicitation* são muito úteis para um *host* em casos onde o protocolo DHCP não está configurado. O protocolo DHCP será descrito na subseção a seguir.

A mensagem *Time Exceeded* é utilizada quando o campo Tempo de Vida (TTL) de um datagrama chega a zero ou quando um fragmento de um datagrama demora muito para chegar ao roteador que está fazendo a remontagem deste. O valor do campo TTL é decrementado a cada roteador para evitar que este possa vir a ficar eternamente dentro de um ciclo de roteamento. Esta mensagem é enviada à origem do datagrama informando que ele foi descartado (COMER, 2006).

A ferramenta de testes *Traceroute* é utilizada para descobrir por quais roteadores os datagramas percorrem até chegarem ao seu destino. Esta ferramenta envia um segmento UDP no datagrama IP, com o campo TTL de valor unitário, para o seu destino. Este datagrama ao chegar ao primeiro roteador é descartado e este roteador gera uma mensagem *Time Exceeded* para a origem do datagrama. Ao receber esta mensagem o *host* de origem gera um novo datagrama idêntico ao anterior, porém com o campo TTL incrementado em 1. Esta tarefa é executada sucessivamente até que o destino é alcançado. Conforme as mensagens *Time Exceeded* vão chegando é possível descobrir por quais roteadores os datagramas percorrem até chegar ao seu destino.

A mensagem *Parameter Problem* indica que foi encontrado um erro no cabeçalho IP. Esta mensagem é enviada à origem do datagrama (TANENBAUM, 2003).

2.1.2 Outros Protocolos importantes para a Camada Rede

Existem outros protocolos importantes para a camada Rede, entre eles está o DHCP (*Dynamic Host Configuration Protocol*) e o IGMP (*Internet Group Management Protocol*). O protocolo DHCP pertence à camada Aplicação por usar um modelo cliente-servidor. O protocolo IGMP é responsável pelo gerenciamento de grupos *multicast*.

O protocolo DHCP permite que um *host* obtenha seu endereço IP, o endereço de um roteador *default* e o endereço de um servidor DNS (Domain Name System). Para usar o DHCP um *host* cliente envia um pacote UDP em *broadcast* (para todos na rede local através do endereço IP de destino 255.255.255.255) na rede requisitando um servidor DHCP. Havendo um servidor DHCP na rede, este responde ao cliente alocando um endereço IP para ele e repassando a ele o endereço de um roteador *default* e o endereço de um servidor DNS (COMER, 2006).

Multicasting é uma forma de enviar uma mensagem para vários destinatários. Esta mensagem pode ser enviada apenas uma vez para o endereço IP *multicast* do grupo ao invés de enviar as várias mensagens iguais para os vários destinos. Através da técnica de *multicasting* tem-se um ganho significativo no tráfego da rede. O protocolo IGMP tem como sua principal tarefa gerenciar a entrada e saída de integrantes nos grupos multicast (STEVENS, 2008).

2.2 Roteamento e Encaminhamento

Duas tarefas muito importantes realizadas pela camada Rede, executadas por equipamentos de rede chamados roteadores, são o roteamento e o encaminhamento de pacotes. O roteamento consiste na descoberta de caminhos para transmitir os datagramas, normalmente utilizando algoritmos e protocolos de roteamento. O encaminhamento é o chaveamento ou comutação dos pacotes, que chegam aos enlaces de entrada dos roteadores, para os enlaces de saída apropriados. O roteador, equipamento indispensável em redes de

comutação de pacotes, é quem determina o caminho, ou seja, as rotas que os datagramas irão tomar.

O encaminhamento baseia-se no conceito de que o roteador, a cada datagrama que chega a um enlace de entrada, verifica o endereço destino deste e o compara com sua tabela de roteamento, também chamada de tabela de encaminhamento. Após encontrar um endereço de rede compatível com o destino o roteador encaminha o datagrama pela sua interface correspondente ao destino ou ao próximo roteador antes do destino (INÁCIO, 2002). Se o roteador não encontrar endereço de rede compatível com o destino o datagrama é descartado e uma mensagem de erro ICMP *Destination Unreachable* é gerada e enviada à origem. O processo de encaminhamento pode ser resumido em comutação de datagramas, ou seja, comutar um datagrama entrante a uma saída de acordo com a tabela de roteamento.

O processo de roteamento é responsável pela montagem e manutenção das tabelas de roteamento nos roteadores. Este processo é executado por um protocolo de roteamento, o qual pode ser dinâmico ou estático. O roteamento estático representa as ações humanas de alterar a tabela de roteamento "manualmente" e o roteamento dinâmico representa as ações de algum protocolo de roteamento na tabela de roteamento. Consequentemente, o roteamento estático é muito lento e raramente altera as tabelas de roteamento. Já o roteamento dinâmico é baseado em algoritmos que sempre estão atentos a mudanças na rede e podem ficar alterando as tabelas de roteamento de acordo com a necessidade (KUROSE; ROSS, 2004).

Na Figura 4 são ilustradas as tarefas de roteamento e encaminhamento dos roteadores. O valor do cabeçalho do pacote, que está chegando ao roteador com a tabela em evidência, é comparado com a tabela de encaminhamento e então é determinado à qual enlace de saída o mesmo deve ser encaminhado. No exemplo da Figura 4, de acordo com a tabela de encaminhamento, o pacote deve ser encaminhado pela saída 2 para alcançar o destino 0111.

Quando uma rede atinge um tamanho grande, onde não é mais possível que o administrador mantenha manualmente as informações sobre roteamento faz-se uso de protocolos de roteamento dinâmicos. Na Subseção 2.2.1 estão detalhados os algoritmos de roteamento dinâmicos.

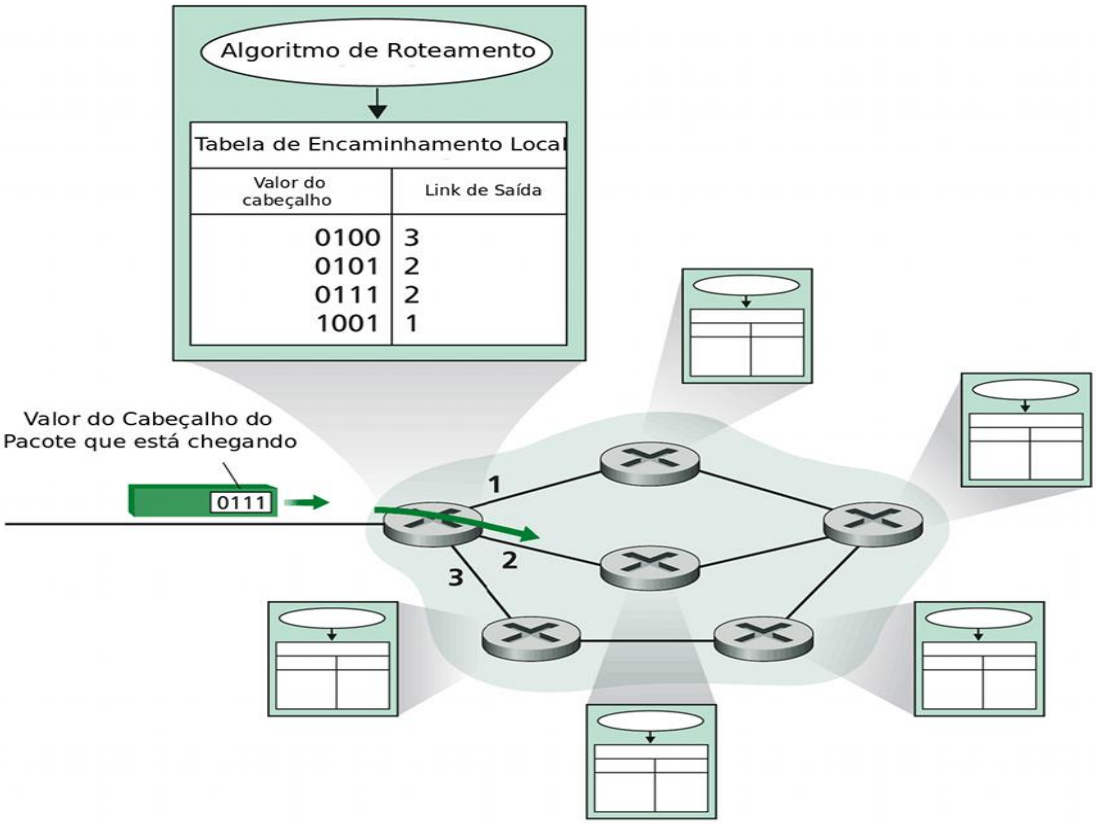


Figura 4 – Roteamento e Encaminhamento (KUROSE; ROSS, 2004).

2.2.1 Algoritmos de roteamento dinâmicos

Um algoritmo representa formalmente uma sequência finita de passos para que se obtenha a solução para um problema (LOPES, 1999). Portanto, um algoritmo de roteamento serve para solucionar o problema do roteamento, em uma malha de roteadores, através de uma sequência finita de passos que são executados por um protocolo. Mais à frente será descrito essa relação algoritmo protocolo.

Logo que um datagrama IP chega a um roteador, este deve saber por qual rota deve ser encaminhado tal datagrama IP. Neste caso tem-se como problema achar a melhor rota para cada possível rede de destino antes que qualquer datagrama IP chegue neste roteador.

A escolha da melhor rota é baseada em políticas e métricas, podendo variar de acordo com o(s) critério(s) escolhido(s). Uma rota pode ser determinada de acordo com algumas características, tais como número de saltos até o destino, atraso em milissegundos, número de pacotes a serem transmitidos por este enlace, entre outras características semelhantes (KUROSE; ROSS, 2004).

Os algoritmos de roteamento dinâmicos podem ser classificados em algoritmos globais ou descentralizados. Os algoritmos globais têm conhecimento completo e global da rede fazendo com que seus cálculos sejam melhores, porém necessitam de todas as informações necessárias sobre a rede antes de começar os cálculos. Os algoritmos descentralizados fazem o cálculo das rotas de maneira iterativa e distribuída, ou seja, o cálculo da melhor rota é realizado necessitando apenas ter o conhecimento de seus roteadores vizinhos (KUROSE; ROSS, 2004).

Nas subseções a seguir são descritos os dois principais algoritmos de roteamento, o algoritmo Estado de Enlace e o algoritmo Vetor de Distâncias.

Estado de enlace

Os algoritmos estado de enlace são algoritmos globais, sendo assim eles necessitam saber os custos de todos os enlaces entre roteadores de toda a rede. Para que todos os roteadores da rede obtenham tais informações cada roteador deve executar duas importantes tarefas. Primeiro, verificar o estado dos seus enlaces periodicamente, para tentar detectar qualquer mudança no estado do enlace que implique em alterações de custo para ele. Segundo, propagar informações periodicamente do estado de seus enlaces para todos os outros roteadores (COMER, 2006).

A tabela de roteamento de cada roteador é atualizada sempre que chega uma nova informação diferente da atual em sua tabela. Para realizar tal tarefa detalhar-se-á aqui o algoritmo de Dijkstra. Este algoritmo, dispondo de todos os custos de todas as rotas da rede, calcula a melhor rota para cada um dos *n* roteadores destino da rede em *n* iterações (KUROSE; ROSS, 2004). Com isso, define-se em cada uma das iterações a melhor rota para um possível roteador de destino.

Na Figura 5 é mostrado um exemplo de rede para o detalhamento da construção da tabela de roteamento, para o roteador A, através do algoritmo estado de enlace. As letras próximas aos nós da rede (roteadores) representam o nome dos roteadores e os números próximos aos enlaces representam os seus respectivos custos, cujos valores foram gerados aleatoriamente.

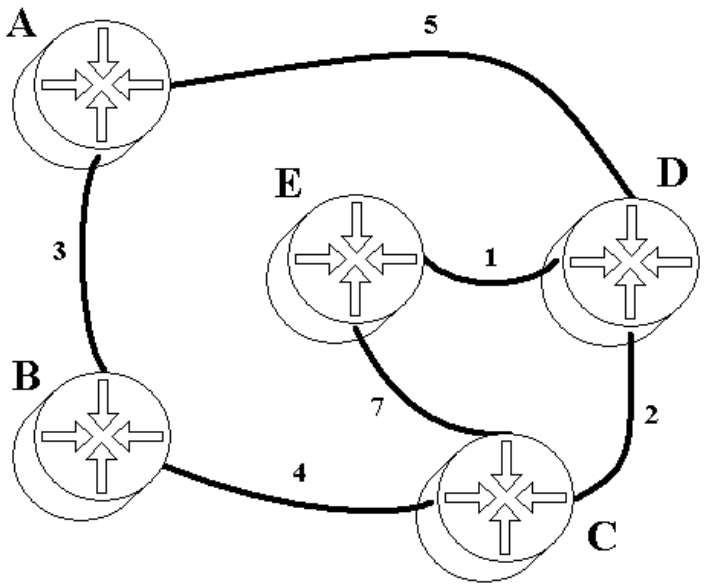


Figura 5 – Rede abstrata.

No estágio de inicialização, o algoritmo de Dijkstra descobre quais roteadores estão conectados diretamente no roteador A, e quais os custos dos enlaces até eles. Para os roteadores que não estão conectados diretamente subentende-se que o custo de chegar até eles é infinito. Tal estágio de inicialização pode ser visualizado na Tabela 2.

Como primeira tarefa de cada iteração, o algoritmo deve procurar por um provável destino com a menor métrica. Facilmente identifica-se o roteador B como o destino com a menor métrica através da Tabela 2.

Após o estágio de inicialização, o algoritmo define que o caminho de menor custo até o roteador B é através do enlace direto com o roteador B. Já para o roteador D não é possível definir o caminho de menor custo, pois pode haver um caminho de menor custo através do roteador B até o roteador D.

Destino	Roteador	Métrica
B	A	3
D	A	5
C	-	∞
E	-	∞

Tabela 2 – Estágio de inicialização para roteador A.

Na tarefa seguinte de cada iteração o algoritmo procura por custos menores, a partir de um roteador com rota já definida nesta mesma iteração, no caso desta primeira iteração utiliza-se o roteador B conforme previamente explicado, para todos os outros destinos possíveis de A através de enlaces diretos com B. O custo é calculado como sendo o custo de A até B mais o custo de B até o provável destino. Se este custo calculado for menor que o custo que já consta na tabela este valor é atualizado na mesma. Para os roteadores que não estejam acessíveis diretamente através de B não é alterado nada na tabela a respeito deste destino.

Na Tabela 3 é possível visualizar o resultado da primeira iteração. Os valores em negrito foram definidos no início da primeira iteração e não serão alterados nas iterações seguintes.

Destino	Roteador	Métrica
B	**A**	**3**
D	A	5
C	B	7
E	-	∞

Tabela 3 – Primeira iteração para o roteador A.

Após a primeira iteração é possível definir que o menor caminho até o roteador D é através do enlace direto de A até D, pois não há um caminho de menor custo através de B, já que o caminho A-B-C já tem um custo de 7.

Tendo mais uma rota definida é possível partir para a próxima tarefa da segunda iteração. Como na primeira iteração parte-se do próximo roteador, que teve sua rota definida previamente nesta primeira iteração, com métrica menor para tentar achar as melhores rotas para os outros roteadores. Na Tabela 4 é possível visualizar o resultado da segunda iteração.

Destino	Roteador	Métrica
B	**A**	**3**
D	**A**	**5**
C	B	7
E	D	6

Tabela 4 – Segunda iteração para o roteador A.

Na segunda iteração define-se que o custo dos enlaces de A até E passando por D: o custo de A até D mais o custo de D até E é 6. No início da terceira iteração o algoritmo define mais uma rota, a rota para o destino E. A partir deste roteador o algoritmo tenta novamente achar rotas com menor custo para o único roteador que ainda não tem rota definida, o roteador C.

Como o roteador E tem um enlace direto de custo 7 com o roteador C, o custo da rota de A até C através de E seria 6 + 7. O que resultaria em um custo maior para o destino C do que o atual na tabela, custo 7. Desta forma não há atualização alguma na tabela de roteamento durante a terceira iteração.

No início da quarta iteração é definido a rota para o roteador C, e como não há mais roteadores de destino sem rota definida, nada é alterado na tabela durante a quarta iteração. Terminada a n-iteração para os n-roteadores é finalizado o algoritmo estado de enlace.

Após o término do algoritmo estado de enlace é possível traçar através da tabela de roteamento resultante a provável rota até qualquer um dos possíveis destinos de A (KUROSE; ROSS, 2004). Sabe-se que uma informação a ser enviada, por exemplo, do roteador A para o roteador E deve ser encaminhada primeiro ao roteador D.

As tabelas de roteamento para todos os roteadores da Figura 5 podem ser conferidas nas Tabelas 5, 6, 7, 8 e 9 a seguir:

Destino	Roteador	Métrica
B	**A**	**3**
D	**A**	**5**
C	**B**	**7**
E	**D**	**6**

Tabela 5 – Tabela de roteamento final do roteador A.

Destino	Roteador	Métrica
A	**B**	**3**
C	**B**	**4**
D	**C**	**6**
E	**C**	**7**

Tabela 6 – Tabela de roteamento final do roteador B.

Destino	Roteador	Métrica
D	**C**	**2**
B	**C**	**4**
E	**D**	**3**
A	**D**	**7**

Tabela 7 – Tabela de roteamento final do roteador C.

Destino	Roteador	Métrica
E	**D**	**1**
C	**D**	**2**
A	**D**	**5**
B	**C**	**6**

Tabela 8 – Tabela de roteamento final do roteador D.

Destino	Roteador	Métrica
D	**E**	**1**
C	**D**	**3**
A	**D**	**6**
B	**D**	**7**

Tabela 9 – Tabela de roteamento final do roteador E.

O algoritmo estado de enlace define os custos dos enlaces de acordo com a situação dos mesmos no momento da averiguação. O tipo de característica que é averiguado no enlace pode variar de protocolo para protocolo e também depende do administrador da rede em questão.

Uma situação problemática comum surge, através do uso do roteamento estado de enlace, a partir do momento que a rede começa a ter grandes dimensões. Isso implica que o número de roteadores pode aumentar significativamente, consequentemente o número de enlaces também. Sabendo-se que quando se dispõe de mais enlaces, a frequência com que ocorrerão mudanças nos seus respectivos custos aumentará, resultando em um número maior de vezes em que cada roteador terá de executar todo o algoritmo novamente.

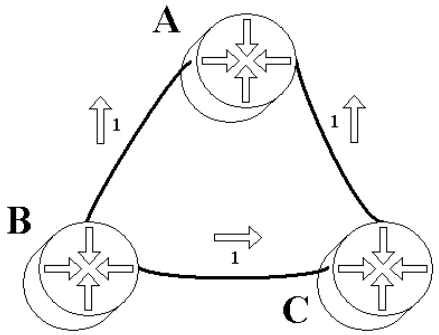
Além deste problema, um crescimento na rede implica também na necessidade de maior tempo para execução do algoritmo, em cada roteador, devido ao aumento proporcional de iterações, uma vez que o número de iterações é definido como sendo o número de roteadores na rede. Alguns protocolos que fazem uso deste algoritmo, como por exemplo, o OSPF, apresentam uma solução relativamente simples para resolver este problema, eles dispõe a organização da rede em áreas. Tal solução será abordada na Subseção 2.3.1 onde é descrito o protocolo OSPF (Open Shortest Path First).

Outro problema deste algoritmo surge quando é levado em consideração, para o cálculo do custo dos enlaces, o tráfego dos mesmos. Na Figura 6 a seguir é apresentado o problema das oscilações causado pela variação de tráfego nos enlaces descrito em (KUROSE; ROSS, 2004). Leva-se em consideração que o tráfego onde não há indicação é nulo. Por exemplo, o tráfego gerado pelo roteador A com destino ao roteador B.

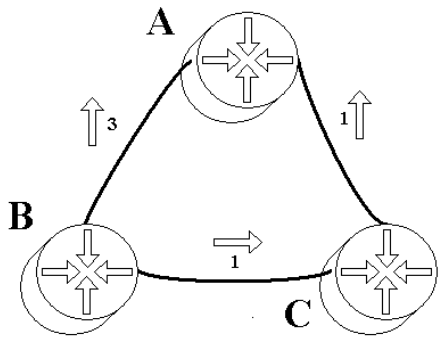
As condições iniciais, ou tráfego inicial, demonstradas na Figura 6 (a) indicam que o roteador A não gera tráfego algum. O roteador B gera um tráfego de 1 com destino ao roteador A e de 1 com destino ao roteador C. O roteador C apenas gera um tráfego de 1 com destino ao roteador A. É importante esclarecer que o tráfego do enlace AC não é o mesmo do enlace CA, ou seja o roteador C não está encaminhando o tráfego entrante ao roteador A. Na Figura 6 (b) é demonstrado o aumento de tráfego de 1 para 3 no enlace BA. Tal aumento provocará oscilações no roteamento da rede.

Ainda na Figura 6 (b), após o aumento de tráfego o roteador B nota-se que o custo do enlace BA agora é 3 e que o custo da rota para A através de C permanece 2. Rapidamente a sua tabela de roteamento é alterada e o tráfego de BA é desviado para BC resultando no tráfego da Figura 6 (c). Não há alterações nas tabelas dos demais roteadores devido à alteração de custo no enlace BA.

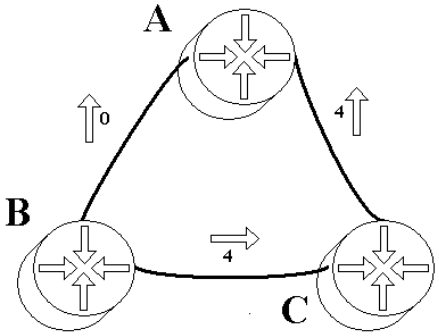
Depois de estabelecido o novo tráfego, mostrado na Figura 6 (c), novos custos são novamente definidos, isso leva a um novo cálculo da tabela de roteamento dos roteadores B e C. O roteador B nota que agora é possível transmitir para A com um custo 0 e para C através de A com um custo nulo também. Já o roteador C nota que é melhor transmitir para A através de B com um custo nulo também. Após tais mudanças de roteamento tem-se o tráfego demonstrado na Figura 6 (d).



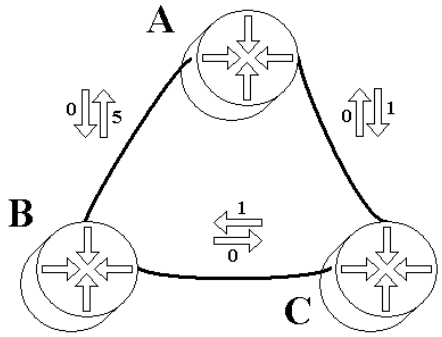
(a) Tráfego e roteamento inicial.



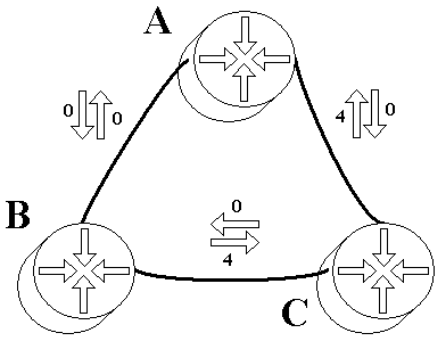
(b) Aumento de tráfego gerado pelo roteador B com destino ao roteador A.



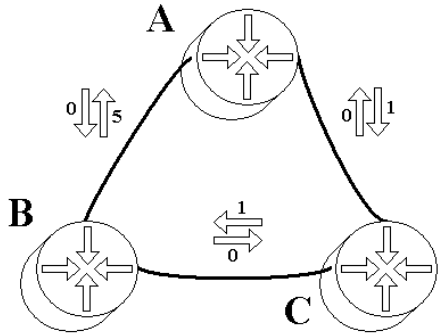
(c) Roteamento após o recálculo das rotas pelo algoritmo estado de enlace.



(d) Roteamento após recálculo.



(e) Roteamento após recálculo.



(f) Roteamento após recálculo.

Figura 6 – Oscilações causadas pela alteração de tráfego (KUROSE; ROSS, 2004).

Sucessivamente as tabelas de roteamento irão mudar devido às alterações de tráfego. As duas próximas alterações de tráfego podem ser visualizadas na Figura 6 (e) e (f). A tabela de roteamento do roteador A também sofreu mudanças devido às alterações de tráfego, porém tais mudanças não foram mencionadas devido ao roteador A não estar gerando tráfego algum na rede. O fato mais importante é que a Figura 6 (d) e a Figura 6 (f) são iguais, constatando assim o problema de oscilação.

Vetor de distâncias

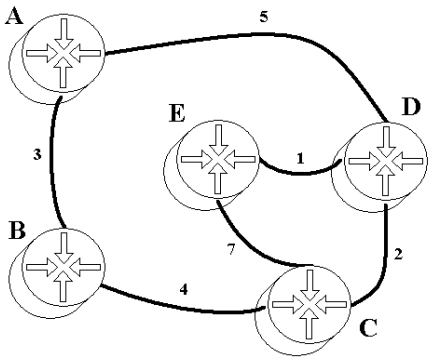
Os algoritmos vetor de distâncias, também conhecido como algoritmo de Bellman-Ford, ao contrário dos algoritmos estado de enlace (que são globais), são algoritmos distribuídos. O roteador que faz uso do algoritmo vetor de distâncias necessita apenas trocar mensagens entre seus vizinhos, pois necessita somente da informação contida neles para executar seus cálculos e transmiti-los (KUROSE; ROSS, 2004).

Ao contrário do algoritmo estado de enlace que precisa saber de todos os custos da rede para executar seus cálculos, o roteador que faz uso do algoritmo vetor de distâncias necessita apenas dos custos de seus enlaces para dar início aos seus cálculos de roteamento. Conforme seus vizinhos vão recebendo novas informações, processando-as e disseminando-as entre seus vizinhos, este roteador, anteriormente mencionado, vai descobrindo novos destinos (novas redes), calculando melhores caminhos e disseminando suas novas descobertas. Este processo continua até que não haja novas alterações nas tabelas de roteamento (KUROSE; ROSS, 2004).

Para entender, rapidamente, como o algoritmo vetor de distâncias calcula sua tabela de roteamento para o roteador A da Figura 5, representado também na Figura 7 (a), pode-se utilizar a tabela de distâncias representada na Figura 7 (b). A tabela de distâncias pode ser facilmente montada neste caso, onde dispomos visualmente de todos os custos de enlace da rede abstrata. A primeira coluna da tabela representa os possíveis destinos de A e as demais colunas representam os menores custos até estes destinos através de seus vizinhos. Cada linha representa um vetor de distâncias. Por exemplo, o vetor de distâncias do roteador A para o destino B é 3 e 11. Os custos sublinhados representam as rotas que o roteador deverá assumir para seus destinos em sua tabela de roteamento. Para o destino C onde os custos são iguais através dos dois vizinhos assume-se aqui que o roteador A recebeu de B a informação do custo para C antes de receber de D.

No exemplo da Figura 7 facilmente entende-se a tabela de distâncias, porém assumiu-se que o roteador A obteve todas as informações de custos da rede ao mesmo tempo, contudo, isso não acontece na prática. Na prática a tabela de distâncias é montada dinamicamente, ou seja, ela sofre alterações conforme chegam novas informações de seus vizinhos sobre custos (KUROSE; ROSS, 2004). Para entender como funciona essa dinâmica deve-se tentar montar a tabela de distâncias para o roteador A sem o conhecimento dos custos de enlaces que não sejam os seus. Assim o roteador A fará alterações na sua tabela de distâncias conforme for recebendo novas informações e para saber que informações este roteador receberá deve-se

calcular ao mesmo tempo as tabelas de distâncias dos outros roteadores.



(a) Rede abstrata.

Custo até o destino por:

Destino:	$D_A()$	B	D
	B	3	11
	C	7	7
	D	9	5
	E	10	6

(b) Tabela de distâncias para o roteador A.

Figura 7 – Rede abstrata e tabela de distâncias para o roteador A.

Na fase inicial do algoritmo de Bellman-Ford, todos os roteadores só conhecem os custos de seus enlaces. Portanto tais roteadores desconhecem outros roteadores que não são seus vizinhos diretamente. Na coluna da margem esquerda da Figura 8 é possível ver o estado inicial das tabelas de distâncias para os cinco roteadores da rede abstrata.

Definida a fase inicial em todos os roteadores, estes informam seus vizinhos sobre seus vetores de distâncias para cada possível destino (ASSIS; ALVES, 2001). As setas coloridas após a primeira coluna, na Figura 8, de tabelas de distâncias representam essa informação sendo transmitida de um roteador para seus vizinhos.

Na segunda coluna de tabelas estão as tabelas atualizadas após a chegada das informações dos vizinhos. Na tabela de distâncias do roteador A que a partir das informações recebidas do roteador B (seta azul), este agora dispõe de uma rota para C a custo 7 (coluna B e linha C). A partir das informações recebidas do roteador D (seta roxa), o roteador A descobre uma nova rota para C, também a custo 7 (coluna D e linha C). Ainda a partir de D, A descobre uma rota para E a custo 6 (coluna D e linha E). Adotou-se aqui que as informações de B chegaram instantes antes que as informações de D, sendo assim, o roteador A assume, por hora, a rota de custo 7 para C através de B e não de D.

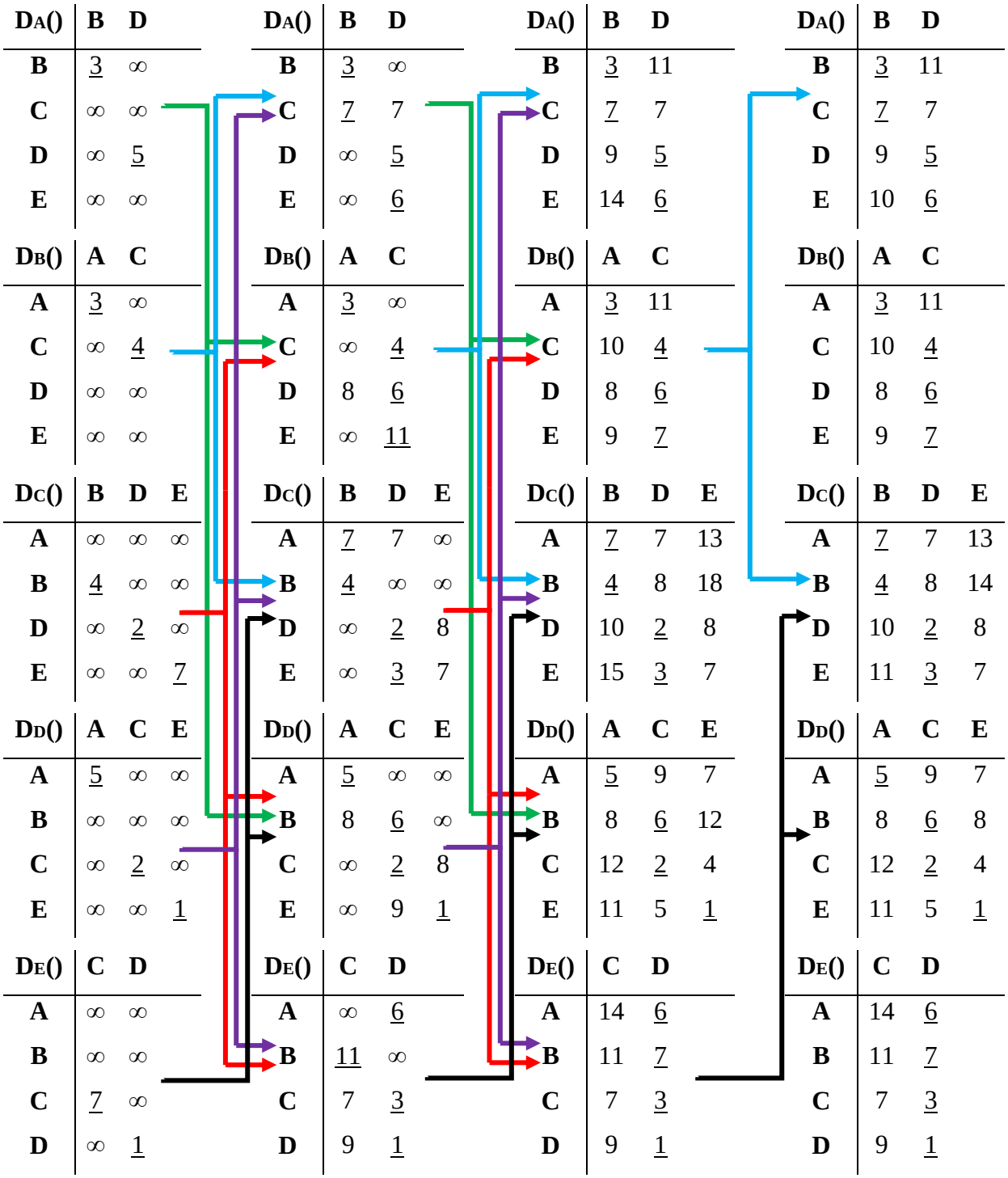


Figura 8 – Tabelas de distâncias para os roteadores da rede abstrata.

A tabela de roteamento gerada pela tabela de distâncias do roteador A, na terceira coluna da Figura 8, não sofreu alteração com a recente chegada de informações de seus vizinhos. Nesta tabela de distâncias houve mudanças nos vetores de distâncias para os destinos B, D e E, porém não houve mudança nos custos sublinhados. Sendo assim, não há a necessidade de informar seus vizinhos sobre tais descobertas.

Passado alguns instantes, o roteador A recebe novas informações do seu vizinho B e ocorre alteração no vetor de distâncias para o destino E, contudo, não acontece nenhuma alteração na tabela de roteamento. Nesta iteração do algoritmo vetor de distâncias, que formou a última coluna da Figura 8, todos os roteadores se estabilizam por não haver mais mudanças nas tabelas de roteamento. Nota-se agora que a tabela de distâncias da Figura 7, construída com conhecimentos gerais dos custos da rede, torna-se igual à tabela de distâncias final do roteador A na Figura 8, construída de forma iterativa com as informações dos outros roteadores.

Deste momento em diante o algoritmo se compromete a enviar periodicamente seus vetores de distâncias a seus vizinhos e os espera receber da mesma forma. Quando um enlace cai, quando o custo de um enlace até seu vizinho muda ou quando o roteador recebe uma informação que irá alterar a sua tabela de distâncias e a de roteamento, este deve sair do estado de inatividade (período em que o roteador fica enviando e recebendo vetores de distancias sem que haja alterações nas tabela de roteamento) e imediatamente comunicar seus vizinhos sobre as mudanças decorridas (KUROSE; ROSS, 2004).

Nota-se que a tabela de roteamento do roteador A construída com o algoritmo vetor de distâncias a partir da tabela de distâncias final do mesmo, disponível na Figura 7 ou na Figura 8 será igual à tabela de roteamento construída com o algoritmo estado de enlace disponível na Tabela 5. Isso comprova que o objetivo de ambos os algoritmos foi alcançado, ou seja, eles conseguiram calcular as rotas de menor custo para todos os destinos de A na rede abstrata (Figura 5).

O algoritmo vetor de distâncias converge para a resposta correta muito bem, porém esta resposta pode vir a ser calculada muito lentamente devido a um grande aumento do custo de um enlace. Depois de estabilizada todas as tabelas de distâncias o algoritmo responde muito bem e rapidamente à redução do custo de um enlace (KUROSE; ROSS, 2004). A Figura 9 (a) apresenta a rede abstrata 2 usada para descrever esta situação e a Figura 9 (b) demonstra como o algoritmo converge rapidamente após a mudança do custo do enlace AB de 5 para 1.

Como citado no parágrafo anterior, um problema pode acontecer quando um enlace sofre um grande aumento de custo. Na Figura 10 (a) é apresentado um aumento no custo do enlace AB de 1 para 50. Na Figura 10 (b) é apresentado o início do problema de *loop* de roteamento onde as tabelas de distâncias dos roteadores A e C não estabilizam até que 38 iterações sejam concluídas.

Este *loop* de roteamento é o período em que o roteador A e C ficam trocando vetores, e incrementando unitariamente o custo mínimo até B, até que C identifique que o custo de

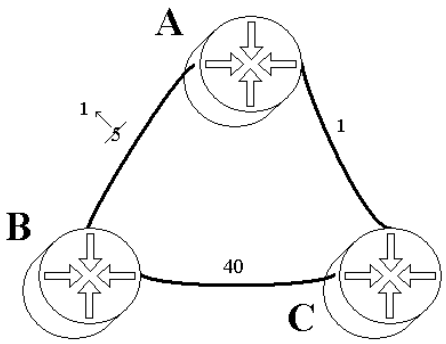
menor valor para B é através do enlace direto com o mesmo e não através de A. Este momento acontece quando o custo de C até B através de A atinge o valor 41.

Enquanto o problema de *loop* de roteamento persistir, todos os datagramas de A e C com destino a B ficam trafegando entre A e C até que o campo TTL do datagrama zere e este seja descartado ou até que os roteadores estabilizem suas tabelas de distâncias (KUROSE; ROSS, 2004).

Considerando a rede abstrata 2 disponível na Figura 10 e imaginando que o custo do enlace BC fosse 10^{10} e que o custo do enlace AB subisse de 1 para (10^{10} + 3), as atualizações levariam 10^{10} iterações. Devido a esta alta contagem de iterações este problema é, considerando determinados cenários, conhecido como problema de contagem até o infinito (KUROSE; ROSS, 2004).

O problema de *loop* de roteamento descrito através da rede abstrata 2 disponível na Figura 10 pode ser resolvido usando uma técnica chamada inversão envenenada. Esta técnica consiste em sempre anunciar ao roteador vizinho que seu custo até determinado roteador cuja rota passe por este mesmo vizinho é infinito (KUROSE; ROSS, 2004).

Na Figura 11 é apresentada como a inversão envenenada funciona. A tabela de distâncias do roteador A inicia com dois custos infinitos devido à inversão invenenada. Logo na primeira iteração após a alteração no custo do enlace o roteador B envenena C, pois a sua melhor rota para A é por C. Na última iteração A envenena C, pois a sua melhor rota para B é por C. Esta técnica não funciona com problemas de loop de roteamento que envolvam três roteadores ou mais (KUROSE; ROSS, 2004).



(a) Rede abstrata 2 com redução no custo do enlace AB.

$D_A()$	B	C
B	5	7
C	11	1

$D_B()$	A	C
A	5	41
C	6	40

$D_C()$	A	B
A	1	45
B	6	40

$D_A()$	B	C
B	1	7
C	7	1

$D_B()$	A	C
A	1	41
C	2	40

$D_C()$	A	B
A	1	45
B	6	40

$D_A()$	B	C
B	1	7
C	3	1

$D_B()$	A	C
A	1	41
C	2	40

$D_C()$	A	B
A	1	41
B	2	40

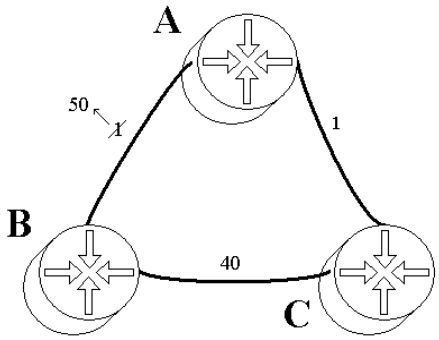
$D_A()$	B	C
B	1	3
C	3	1

$D_B()$	A	C
A	1	41
C	2	40

$D_C()$	A	B
A	1	41
B	2	40

(b) Modificações nas tabelas de distâncias da rede abstrata 2 devido à redução no custo do enlace AB.

Figura 9 – Rápidas atualizações devido à redução de custo num enlace.



(a) Rede abstrata 2 com aumento no custo do enlace AB.

$D_A()$	B	C
B	1	3
C	3	1

$D_B()$	A	C
A	1	41
C	2	40

$D_C()$	A	B
A	1	41
B	2	40

$D_A()$	B	C
B	50	3
C	52	1

$D_B()$	A	C
A	50	41
C	51	40

$D_C()$	A	B
A	1	41
B	2	40

$D_A()$	B	C
B	50	3
C	90	1

$D_B()$	A	C
A	50	41
C	51	40

$D_C()$	A	B
A	1	81
B	4	40

$D_A()$	B	C
B	50	5
C	90	1

$D_B()$	A	C
A	50	41
C	51	40

$D_C()$	A	B
A	1	81
B	4	40

$D_A()$	B	C
B	50	5
C	90	1

$D_B()$	A	C
A	50	41
C	51	40

$D_C()$	A	B
A	1	81
B	6	40

(b) Modificações nas tabelas de distâncias da rede abstrata 2 devido ao aumento no custo do enlace AB.

Figura 10 – Atualizações lentas devido ao aumento do custo num enlace.

$D_A()$	B	C
B	1	∞
C	∞	1

$D_B()$	A	C
A	1	41
C	2	40

$D_C()$	A	B
A	1	41
B	2	40

$D_A()$	B	C
B	50	∞
C	∞	1

$D_B()$	A	C
A	50	41
C	51	40

$D_C()$	A	B
A	1	41
B	2	40

$D_A()$	B	C
B	50	∞
C	90	1

$D_B()$	A	C
A	50	41
C	51	40

$D_C()$	A	B
A	1	∞
B	51	40

$D_A()$	B	C
B	50	41
C	90	1

$D_B()$	A	C
A	50	41
C	51	40

$D_C()$	A	B
A	1	∞
B	51	40

$D_A()$	B	C
B	50	41
C	90	1

$D_B()$	A	C
A	50	41
C	51	40

$D_C()$	A	B
A	1	∞
B	∞	40

Figura 11 – Técnica de inversão envenenada.

2.3 Protocolos de roteamento

Os protocolos de roteamento implementam os algoritmos de roteamento e são responsáveis pela comunicação entre os roteadores. Em roteadores que fazem uso do roteamento dinâmico, os protocolos de roteamento são fundamentais para a montagem e atualizações das tabelas de roteamento.

Com o aumento do número de roteadores no mundo para a escala de milhões implica que se somente um dos protocolos que implementam os algoritmos de roteamento, descritos anteriormente, estivesse em uso jamais se conseguiria utilizar a internet. O algoritmo estado de enlace saturaria a largura de banda com os seus pacotes de atualizações sobre milhares de rotas e o algoritmo vetor de distâncias não iria conseguir convergir devido ao número exorbitante de iterações necessárias (KUROSE; ROSS, 2004).

Outro problema comum é a necessidade de conectar redes diferentes que executam protocolos de roteamento diferentes e de conectar redes que necessitam de certa autonomia organizacional. Para resolver estes problemas pode-se criar AS's (*Autonomous Systems*), ou seja, agrupar roteadores em grupos da mesma região (KUROSE; ROSS, 2004).

Para possibilitar a comunicação entre os AS's criou-se a classificação de roteamento hierárquico, onde pode haver dois tipos de protocolos de roteamento: intra-domínio (intra-AS) ou inter-domínio (inter-AS).

Para que os computadores de um AS consigam acessar outros computadores de fora do seu AS é necessário que haja um ou mais roteadores de borda (*gateway routers*) em cada AS. Os roteadores de borda são responsáveis pela comunicação com outros roteadores de borda de outros AS's (KUROSE; ROSS, 2004). Tais roteadores executam os dois tipos de roteamento hierárquico, pois estão presentes dentro e fora do AS.

Nas subseções a seguir será descrito algumas características dos principais protocolos que implementam os algoritmos descritos na seção anterior de acordo com a classificação de roteamento hierárquico.

2.3.1 Protocolos de roteamento intra-domínio

Os roteadores de um domínio ou AS executam o mesmo protocolo de roteamento intra-domínio que pode utilizar qualquer um dos dois algoritmos tratados na seção anterior.

Nesta subseção será descrito o protocolo RIP (*Routing Information Protocol*) que implementa o algoritmo vetor de distâncias e o protocolo OSPF (*Open Shortest Path First*) que implementa o algoritmo estado de enlace.

O protocolo RIP estabelece um custo unitário para cada enlace tornando-o um protocolo que conta saltos para calcular a melhor rota para um destino. Esta definição resolve o problema da contagem até o infinito descrito na seção anterior. Sendo assim, só é necessário recalcular as rotas quando um enlace cai ou um roteador apresenta problemas, pois o custo dos enlaces nunca irá mudar (STEVENS, 2008).

O RIP define como métrica máxima 15 saltos. Esta definição limita o tamanho do AS onde este protocolo está sendo executado. Um roteador, para divulgar o seu *status* para os roteadores diretamente conectados a si, envia mensagens periodicamente a cada 30 segundos contendo informações sobre suas rotas. Se a mensagem de um vizinhonão é recebida em 3 minutos, o roteador aplica a métrica infinita (16) ao enlace em questão. Um roteador após definir a métrica infinita para um outro roteador vizinho, aguarda 60 segundos para apagar a rota da tabela de roteamento (STEVENS, 2008).

As mensagens RIP são transportadas através do protocolo UDP e destinadas à porta 520. Cada mensagem de atualização periódica pode comportar informações da tabela de roteamento de até 25 rotas (STEVENS, 2008).

O protocolo RIPv1 é definido no RFC 1058 e o RIPv2 é definido no RFC1723 (RFC 1058, 1988) (RFC 1723, 1994). As características descritas acima são comuns às duas versões.

A primeira versão do RIP não envia informações de máscaras de sub-redes nas suas atualizações enquanto que a segunda versão do RIP envia tais informações e permite o roteamento de sub-redes. O RIPv1 não acomoda autenticação, possibilitando que qualquer roteador conectado ao AS consiga obter as informações de roteamento de seus vizinhos. O RIPv2 permite autenticação do tipo texto ou com criptografia MD5 (*Message-Digest 5*). O protocolo RIPv1 transmite suas atualizações utilizando o endereço de *broadcast* na rede 255.255.255.255 enquanto que o protocolo RIPv2 transmite suas atualizações utilizando o endereço *multicast* 224.0.0.9 aumentando a eficiência em enlaces onde se encontram roteadores e computadores comuns (RFC 1058, 1988) (RFC 1723, 1994).

O protocolo OSPFv2 é definido no RFC 2178 e como foi concebido posteriormente ao RIP, ele é um protocolo com uma série de características avançadas. Um roteador que executa este protocolo envia o estado de seus enlaces periodicamente a todos os roteadores da rede (KUROSE; ROSS, 2004).

Todas as mensagens trocadas entre roteadores OSPF são certificadas. O OSPF permite que um destino possa ter mais de um caminho no caso de custo igual, possibilitando a distribuição da carga entre os diferentes caminhos (KUROSE; ROSS, 2004).

O OSPF permite diferentes tipos de métricas para diferentes tipos de tráfegos, ou seja, pacotes com ToS diferente com um destino comum podem ser enviados por rotas diferentes. Assim o OSPF pode definir caminhos com métricas baixas para mensagens que necessitam de altas taxas de transmissão e caminhos com métricas mais elevadas para mensagens que não necessitam de altas taxas de transmissão. O OSPF também tem suporte integrado para roteamento *unicast* e *multicast*. (KUROSE; ROSS, 2004).

Um dos maiores avanços do OSPF foi possibilitar a criação de uma hierarquia dentro de um AS resolvendo problemas de escalabilidade. Isso é possível através da criação de áreas dentro do AS onde cada área executa seu próprio algoritmo estado de enlace independentemente uma da outra. Cada área deve conter um ou mais roteadores de borda responsáveis por interagir com outros roteadores de borda de outras áreas. Tais roteadores também podem se comunicar com outros roteadores, que não pertencem a nenhuma área, conhecidos como roteadores de *backbone*, pois estão um nível acima dos roteadores inclusos em áreas. Interconectado com os roteadores de *backbone* deve estar um roteador de borda do AS responsável por trocas informações de roteamento com outros roteadores de borda de outros AS’s (KUROSE; ROSS, 2004).

Na
Tabela **10** é apresentada uma comparação das principais características entre os protocolos RIP e OSPF.

Característica	RIPv2	OSPFv2
Autenticação	Sim	Sim
Escalabilidade	Pequena pois pode abranger 15 enlaces de distância	Alta através da divisão do AS em áreas
Métrica	Unitária para cada enlace	Pode ser variável de acordo com a quantidade e tipo de tráfego
Tempo de convergência	Lento	Rápido
Tempo para atualização das rotas	A cada 30 segundos	Somente quando ocorre alguma alteração na rede
Consumo de banda	Alto	Baixo
Permite *multicast*	Sim	Sim

Tabela 10 – Tabela comparativa entre os protocolos RIP e OSPF.

2.3.2 Protocolo de roteamento inter-domínio

Para que seja possível a comunicação entre AS's que executam protocolos intra-domínio diferentes é necessário que os roteadores de borda dos AS's executem também um protocolo inter-domínio considerado atualmente padrão, o BGP v4 (*Border Gateway Protocol*) definido nos RFC's 1771, 1772, 1773 (KUROSE; ROSS, 2004).

O BGP foi o substituto do EGP (*Exterior Gateway Protocol*) que tinha diversas limitações. Provavelmente, o maior problema do EGP era a limitação da Internet em topologia tipo árvore, impedindo assim que outras formas de topologias mais genéricas fossem implantadas para interligar os AS's (PETERSON; DAVIE, 2004).

A principal preocupação do roteamento inter-domínio é encaminhar qualquer pacote destinado a qualquer lugar da Internet. Esta tarefa se torna difícil devido ao tamanho das tabelas de roteamento dos roteadores inter-domínio que podem ser imensas (PETERSON; DAVIE, 2004).

Como os AS's são independentes, as métricas internas são definidas pelos seus respectivos protocolos e podem levar diferentes questões para o cálculo das mesmas. Sendo assim, se um roteador inter-domínio precisasse calcular a distância para um destino longínquo seus cálculos poderiam estar muito errados. Isso levando em consideração que houvesse dois ou mais caminhos possíveis através de AS's diferentes, pois assim poderia-se

ter métricas que foram calculadas de formas diferentes. Para resolver este problema os roteadores divulgam entre si apenas a informação de que é possível acessar o destino específico através deste e daquele AS deixando assim o roteador de origem escolher a rota de destino através de questões políticas (PETERSON; DAVIE, 2004).

Uma política comum dos roteadores inter-domínio é a prevenção do tráfego em trânsito, ou seja, o tráfego originado em um AS vizinho com destino a outro AS vinho onde o AS local simplesmente faria a interligação dos dois AS's. Também há políticas para escolher o caminho de destino, como escolher o caminho que cruze menos AS's ou escolher, sempre que possível, um caminho que não cruze um AS em específico (PETERSON; DAVIE, 2004).

Com o BGP, o administrador de um AS deve escolher um ou mais "porta-vozes BGP". Tais roteadores devem estabelecer seções com outros porta-vozes em outros AS's para trocar informações de confiabilidade. Tais seções envolvem a troca de informações de capacidade de transmissão, pois nem sempre um caminho viável é um caminho que dará conta do tráfego necessário. Além dos porta-vozes os AS's devem ter os roteadores de borda, que não precisam ser necessariamente os mesmos roteadores porta-vozes (PETERSON; DAVIE, 2004).

O BGP tem características parecidas com um protocolo vetor de distância, porém ele é caracterizado como um protocolo vetor de caminho. Como vimos anteriormente, o BGP não propaga informações de custo, mas sim informações sobre o caminho até um destino, como a sequência de AS's a ser percorrida até um destino (KUROSE; ROSS, 2004).

O protocolo BGP utiliza a porta 179 com o protocolo TCP. Dois roteadores inter-domínio vizinhos formam um par BGP. As mensagens do BGP são propagadas entre pares BGP. Existem quatro tipos de mensagens (KUROSE; ROSS, 2004):

- OPEN: Esta mensagem é enviada por um roteador BGP para estabelecer contato pela primeira vez com seu par. Ela permite identificação e autenticação além de ajustar informações de temporização. Se tudo estiver correto, este roteador deve receber uma mensagem KEEPALIVE;
- UPDATE: Quando ocorre a necessidade de anunciar, ou remover um caminho, um possível caminho esta mensagem é utilizada;
- KEEPALIVE: Serve para anunciar que está tudo OK quando não há atualizações a serem feitas ou para responder à mensagem OPEN;
- NOTIFICATION: Notifica que um erro foi detectado em uma mensagem ou o remetente está encerrando a seção BGP.

3 *Simulação de funcionamento dos protocolos de roteamento*

O funcionamento dos protocolos de roteamento é muito difícil de ser acompanhado durante sua execução nas máquinas reais por ser um processo conjunto entre todos os roteadores envolvidos. Sendo assim é mais fácil utilizar um simulador para averiguar de forma rápida o funcionamento dos protocolos de roteamento.

O processo de simulação parte do princípio de imitar algum processo real de forma rápida para obter previsões de algum resultado (SCHNEIDER, 2004). Para observar as possíveis soluções de um protocolo de roteamento para uma determinada rede pode-se utilizar um simulador de redes apto a trabalhar com protocolos de roteamento.

Neste capítulo serão apresentadas na Seção 3.1 ferramentas de simulação para protocolos de roteamento e o na Subseção 3.1.1 o resultado o resultado de simulações utilizando a ferramenta OPNET.

3.1 Simulação em redes de computadores

Através de técnicas de modelagem e simulação é possível averiguar o funcionamento e desempenho de vários protocolos. Várias ferramentas de simulação vêm sendo desenvolvidas para permitir a modelagem e simulação em redes de computadores, entre elas está a ferramenta OPNET, o NS (*Network Simulator*) e o OMNET++ (*Objective Modular Network Testbed in C++*). A ferramenta OPNET foi inicialmente escolhida para o estudo dos protocolos de roteamento neste projeto.

Os resultados obtidos com a ferramenta escolhida não permitiram visualizar o que gostariamos. Tais resultados seriam a visualização das tabelas de roteamento após o processo de montagem das mesmas pelos protocolos de roteamento. Outras ferramentas de simulação de fonte aberta como o NS ou o OMNET++ não chegaram a ser testadas.

Na subseção a seguir está descrito o procedimento e os resultados obtidos com a ferramenta OPNET. Nesta mesma subseção também são expostos os problemas encontrados com esta ferramenta que ocasionaram a não utilização da mesma ao longo do projeto.

3.1.1 Simulação com OPNET

A ferramenta OPNET versão acadêmica foi escolhida para esta simulação devido a trabalhos anteriores já realizados com a mesma.

Com a esta ferramenta é possível criar o modelo de uma rede desejada, escolher quais estatísticas devem ser coletadas (de algum equipamento da rede, da rede como um todo ou de algum protocolo específico), realizar a simulação e visualizar os resultados.

Com o OPNET é possível definir qual o fabricante e o modelo para cada equipamento da rede de acordo com a lista disponível. O cenário da Figura 12 utilizado para esta simulação tem como componentes:

- Três roteadores interligados entre si;
- Um hub conectado a cada roteador;
- Dois computadores conectados a cada hub.

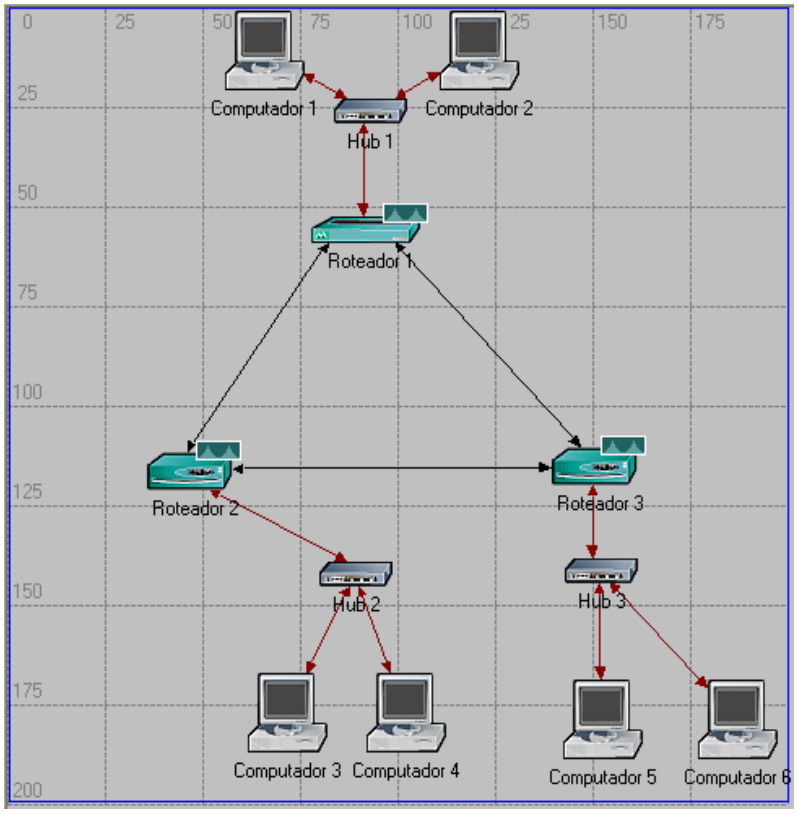


Figura 12 – Cenário Similar ao Laboratório de Redes I

Após a determinação do cenário e os equipamentos utilizados é necessário configurar cada equipamento e fazer as configurações necessárias para o protocolo de roteamento a ser utilizado.

Deve-se selecionar as estatísticas a serem coletadas na rede, executar a simulação e analisar os resultados.

Neste projeto foram feitas duas simulações, uma com o protocolo RIP e outra com o protocolo OSPF. Na Figura 13 e na Figura 14 é apresentado o tráfego dos protocolos RIP e OSPF na rede simulada.

É importante ressaltar que esses resultados são referentes aos três enlaces entre os roteadores juntos, ou seja, o tráfego obtido nestas simulações não é de um único enlace, mas sim dos três enlaces juntos. Sabe-se também que a simulação é iniciada nos três roteadores ao mesmo tempo.

Comparando-se os dois gráficos pode-se constatar que o protocolo RIP converge mais rápido que o protocolo OSPF, enquanto o RIP leva onze segundos e o OSPF leva quarenta e um (dos doze aos cinqüenta e três) segundos para convergir.

Também é possível observar as mensagens de troca de tabela sendo enviadas de trinta em trinta segundos pelo protocolo RIP. Já o protocolo OSPF troca mensagens "*Hello*" (mensagens para estabelecimento e status do enlace) de tempo em tempo entre os roteadores de forma mais espalhada no tempo, provavelmente devido a não convergência do protocolo ao mesmo tempo em todos os roteadores.

Mesmo o protocolo RIP enviando mensagens de forma síncrona, ocupando pouco tempo para transmitir, em todos os roteadores pode-se constatar que a média do tráfego deste protocolo após convergido, aproximadamente 443bits/s, é maior que o tráfego do protocolo OSPF, aproximadamente 320bits/s, que tem um tráfego mais distribuído no tempo.

Para um estudo mais aprofundado dos protocolos de roteamento através de simulações com a ferramenta OPNET seria necessário a versão completa da mesmo. Com a versão completa é possível analisar os dados importantes (a tabela de roteamento) de cada roteador.

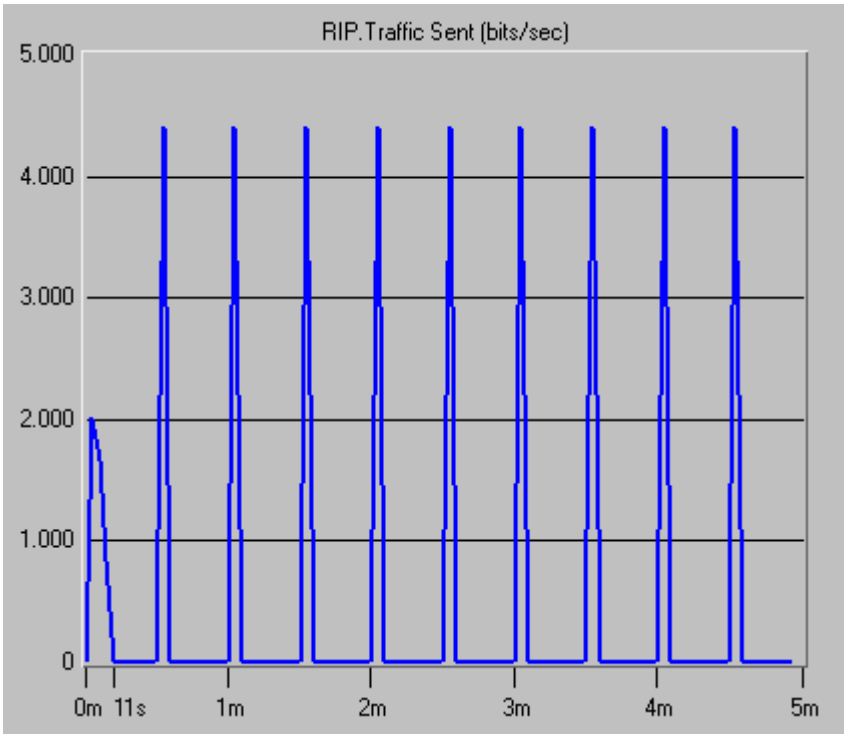


Figura 13 – Tráfego RIP

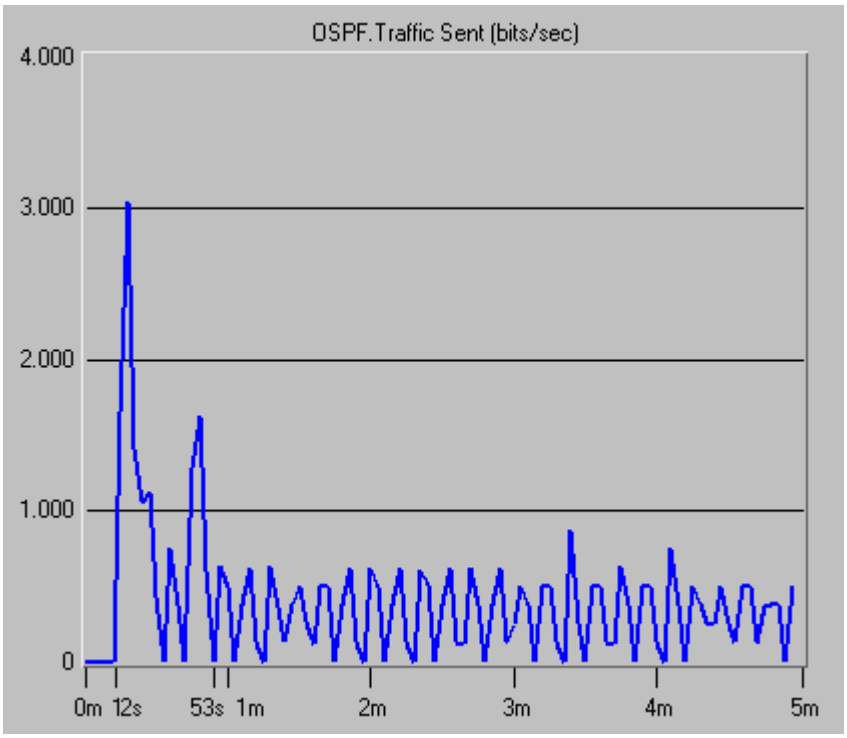


Figura 14 – Tráfego OSPF

4 *Análise de funcionamento dos protocolos de roteamento*

A análise do funcionamento de protocolos de roteamento pode ser realizada monitorando roteadores em operação em uma rede real. Entretanto, o uso de equipamentos reais para montagem de uma estrutura para testes demanda um trabalho árduo. É necessário montar todo o hardware incluindo conectores e cabeamento, além da conexão lógica de cada equipamento. Por outro lado, a montagem de uma rede real para testes pode ser facilmente executada utilizando máquinas virtuais. Já existem vários *softwares* que simulam *hardware* para as máquinas virtuais tais como o Virtual Box e o VMware, porém tais ferramentas exigem muito do sistema hospedeiro.

Optou-se pelas máquinas virtuais UML (*User-Mode Linux*) pela sua simplicidade, por ser um projeto *open-source* e por ser desenvolvido especificamente para o sistema operacional Linux.

Uma vez dispondo de uma "rede real" para testes pode-se utilizar protocolos de gerenciamento de redes, como o SNMP (*Simple Network Management Protocol*), para, remotamente, acompanhar e analisar o funcionamento dos protocolos de roteamento.

Neste capítulo será apresentado:

- Máquinas virtuais UML;
- Protocolo SNMP para análise de roteamento;
- O cenário utilizado;
- Análise dos resultados das observações.

4.1 Máquinas virtuais UML

O ambiente virtual tem toda a parte lógica (*software*), possivelmente, igual à de um ambiente real. A única diferença é que cada máquina do ambiente virtual não possui sua parte

física (*hardware*) própria. As máquinas virtuais compartilham o mesmo *hardware* da máquina hospedeira (RUSSEL; STEARNS, 2008). Sendo assim, este trabalho não parte do conceito de simulação, e sim do conceito de observação no qual o trabalho atua em cima do funcionamento lógico de roteadores reais.

O UML é ao mesmo tempo um *kernel* Linux e um processo Linux. Ambas as perspectivas são muito úteis. No entanto, para muitas pessoas, um *kernel* e um processo são duas coisas completamente diferentes. Na Figura 15 é ilustrada a relação entre uma instância UML, o *kernel* do *host* (computador hospedeiro), UML e processos. Para o *kernel* do *host*, a instância UML é um processo normal. Para os processos da UML, a UML é um *kernel*. Os processos interagem com o *kernel* abrindo Chamadas de Sistema (*System Calls*) (DIKE, 2006).

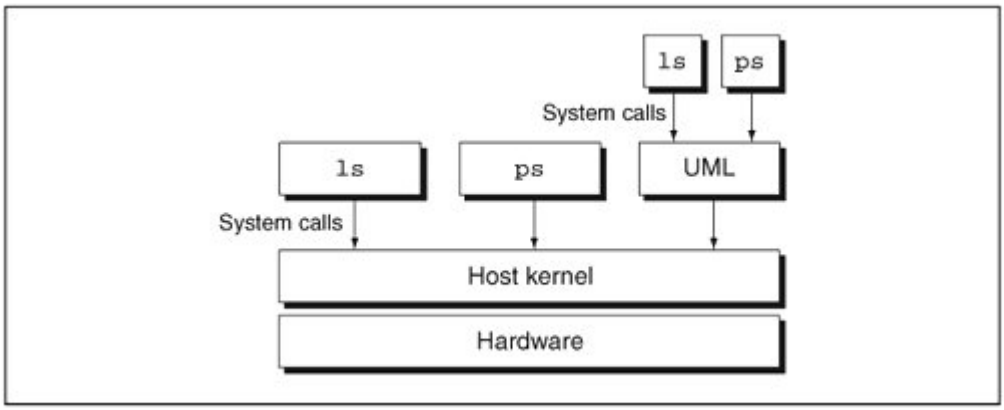


Figura 15 – Relação entre uma instância UML, o *kernel* do *host,* UML e processos (DIKE, 2006).

De acordo com testes inicialmente feitos envolvendo o protocolo RIP entre as máquinas virtuais e a máquina real, conclui-se que o funcionamento dos protocolos de roteamento nas máquinas virtuais será idêntico ao seu funcionamento nas máquinas reais. Sendo assim é perfeitamente possível combinar experimentos envolvendo máquinas reais e virtuais.

Através das máquinas virtuais UML é possível criar redes complexas dentro de um único computador real. Para conectar uma máquina virtual à outra utiliza-se um *switch* virtual chamado **UML Switch**. Para criar uma interface entre a máquina real e uma máquina virtual deve-se configurar uma interface **TUN/TAP** entre ambas as máquinas no momento inicial quando são configuradas todas as características de cada máquina virtual. A TAP simula um dispositivo Ethernet atuando na camada 2 com *frames*. A TUN simula um dispositivo de rede atuando na camada 3 com datagramas IP (KRASNYANSKY, 2000).

Utilizando as máquinas virtuais UML é possível iniciar várias máquinas virtuais a partir de um único sistema de arquivos. Isso é viável através da criação de COW's. A COW (*Copy-*

On-Write) é um mecanismo que permite que várias instâncias UML compartilhem um único sistema de arquivos. Uma COW é um arquivo secundário ao sistema de arquivos onde ficam gravadas todas as alterações feitas no sistema de arquivos referente a uma máquina virtual. Cada máquina virtual deve ter sua própria COW. O uso deste mecanismo torna o sistema de arquivos inalterável a cada reinicialização de uma máquina virtual (DIKE, 2006). Este mecanismo também pode ser facilmente removido para permitir uma alteração qualquer no sistema de arquivos, depois é possível reativar o mecanismo novamente.

Para se construir uma máquina virtual é necessária a disponibilidade de um computador com acesso à internet e com sistema operacional Linux, por exemplo com a distribuição Ubuntu 10.04, pois foi nessa distribuição que os últimos testes deste projeto foram realizados. Pode-se partir de alguns arquivos pré-definidos disponíveis na internet para a construção de uma máquina virtual UML. A seguir segue uma lista dos programas necessários e onde encontrá-los:

- Xterm: O Xterm é um programa que gera uma janela de linha de comando idêntica a um Terminal Linux utilizado para executar programas. O Xterm já vem incluso na distribuição Ubuntu 10.04;
- UML Kernel: Este *software* é o núcleo de processamento Linux da máquina virtual. Esta ferramenta também simula um *hardware* e interage com o sistema operacional do computador hospedeiro como um processo comum. A versão mais recente desta ferramenta pode ser atualmente encontrada no endereço 1 no final desta página;
- Sistema de arquivos: Este é o sistema de arquivos onde ficam gravados todos os arquivos de programas e de usuários da máquina virtual. Este simula um disco rígido para o UML Kernel com o formato ext3. Neste projeto foi utilizada a distribuição Ubuntu Server 9.10 para o sistema de arquivos das máquinas virtuais. As versões mais recentes dos sistemas de arquivos mais comuns, e também suas versões anteriores, podem ser atualmente encontradas no endereço 2 no final desta página;
- UML Utilities: Este *software* é um pacote de ferramentas para as máquinas virtuais UML. Para este projeto são importantes as ferramentas uml_switch, para viabilizar um enlace virtual entre duas máquinas virtuais, e tunctl, para viabilizar um enlace virtual entre uma máquina virtuail e a máquina real. Para instalá-lo no Ubuntu 10.04 basta executar a seguinte linha de comando: “`apt-get install uml-utilities`”.

1. http://uml.devloop.org.uk/index.html
2. http://fs.devloop.org.uk/

Disponibilizando-se das ferramentas enumeradas acima, que necessitam de *download*, e extraídas num mesmo diretório é possível iniciar uma máquina UML simples sem conexões através do seguinte comando:

```
“xterm -geometry 60x20 -T teste -e “”<diretório do UML Kernel> umid=brasil ubda=<diretório do sistema de arquivos> mem=<memória disponibilizada à máquina UML>””.
```

Podem ocorrer problemas devido às recentes compilações do UML Kernel ou com determinados sistemas de arquivos não testados em conjunto. Para evitar tais problemas e outros prováveis disponibiliza-se um pacote com todas as ferramentas necessárias para a execução das máquinas virtuais UML na rede local do campus São José do IF-SC.

4.2 Observação do funcionamento de algoritmos e protocolos de roteamento

Inicialmente, a idéia desse projeto era fazer com que o Observador Didático capturasse todas as mensagens do protocolo de roteamento em questão e a partir destas montar as tabelas de roteamento dos roteadores virtuais. Assim seria possível analisar como cada mensagem interfere na montagem de cada tabela de roteamento.

Na Figura 16 é apresentado um esboço do que seria o Observador Didático. O protocolo de roteamento utilizado deveria utilizar somente os enlaces das linhas contínuas entre os roteadores, na Figura 16, para efetuar a troca de mensagens de roteamento. Tais mensagens deveriam ser capturadas e enviadas a máquina real, onde as tabelas de roteamento da rede virtual, paralelamente, deveriam ser montadas, modificadas e apresentadas ao usuário. Para facilitar e simplificar a execução de testes decidiu-se trabalhar com apenas três roteadores virtuais no Observador Didático.

Estudos foram realizados com a ferramenta Tcpdump e sua pricipal biblioteca, libpcap.h, para a captura de pacotes na rede virtual. Na Figura 17 é possível visualizar parte dos resultados de capturas de pacote na rede virtual durante a sua inicialização. Nota-se que o roteador Brasil inicializou primeiro solicitando informações de roteamento para os roteadores

conectados, logo em seguida, ele divulga suas rotas atuais. Na sequência o roteador EUA inicializa e envia solicitações de informações de roteamento aos outros roteadores, o roteador Brasil responde e divulga suas rotas. Em seguida, o roteador EUA divulga suas rotas que não foram atualizadas ao roteador Brasil e divulga ao roteador Itália todas as suas rotas menos a rota para o roteador Itália. No pacote de ferramentas disponibilizado para a execução do Observador Didático, citado na seção anterior, está disponível uma planilha com a troca de mensagens completa durante a inicialização entre os roteadores virtuais.

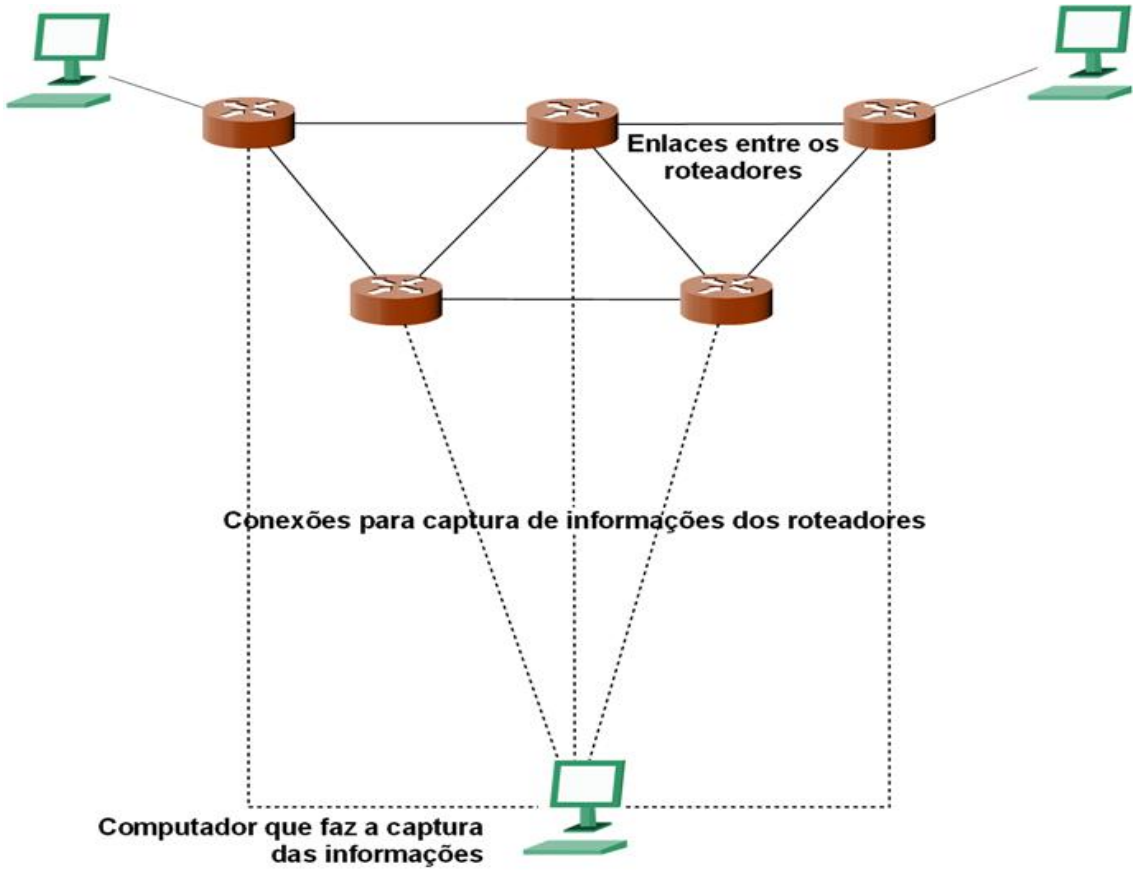


Figura 16 – Idéia inicial para o Observador Didático.

Estudos foram feitos para procurar um método mais simples para realizar a observação sobre algoritmos e protocolos de roteamento, uma vez que o método com a captura de mensagens se mostrou trabalhoso. Após o estudo realizado com o protocolo SNMP (*Simple Network Management Protocol*) para realizar a captura das tabelas de roteamento decidiu-se que através deste seria possível construir uma boa didática demonstrando como e quando as tabelas de roteamento se alteram. Na Seção 4.3 é descrito o protocolo SNMP.

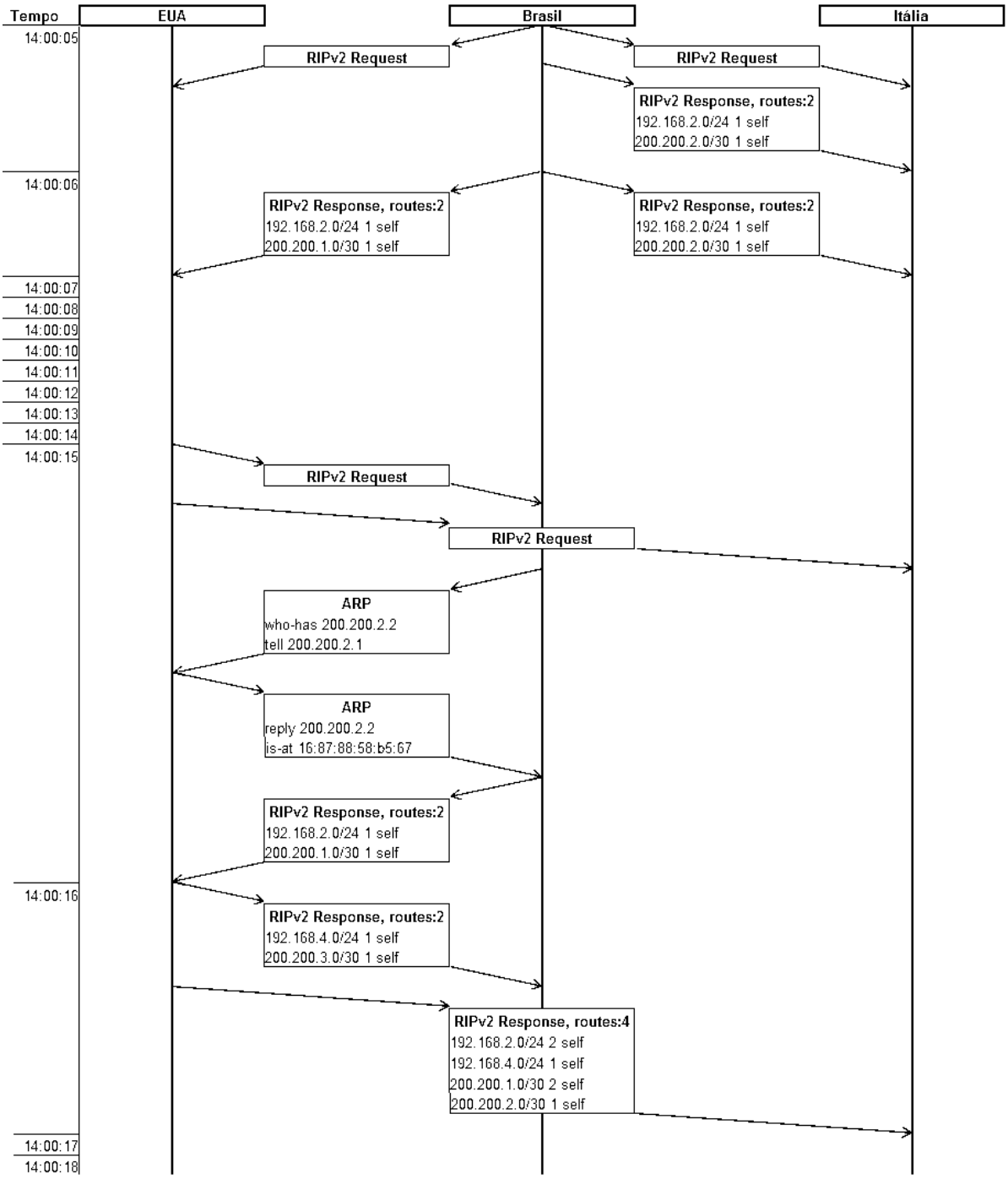


Figura 17 – Troca de mensagens RIP durante a inicialização dos roteadores Brasil e EUA.

4.3 Protocolo SNMP para análise de roteamento

O protocolo SNMP (*Simple Network Management Protocol*) visa facilitar o gerenciamento de redes através do monitoramento de equipamentos de rede e de servidores.

O protocolo SNMP é um protocolo da camada aplicação que não atua no modelo cliente-servidor, modelo muito comum em protocolos da camada aplicação. O modelo adotado é o gerente-agente onde o gerente, na maioria dos casos, inicia uma sessão com um agente para monitorar uma ou mais características importantes da rede (STEVENS, 2008). A versão 2 do SNMP definido na RFC 1905 foi adotada para este projeto.

As características da rede são nomeadas como objetos gerenciados, que são definidos e agrupados em módulos, na MIB (*Management Information Base*). Tais objetos podem ter diferentes permissões. O gerente pode ler e escrever em alguns objetos, somente ler em outros, ou ainda, criar determinados objetos. A versão atualmente usada é a MIB II, definida na RFC 1213. A MIB tem seus dados organizados hierarquicamente em árvore (KUROSE; ROSS, 2004).

A IETF (*Internet Engineering Task Force*) é responsável pela padronização de módulos MIB associados a roteadores e outros equipamentos de rede. Para dar nome e organizar tais módulos, a IETF resolveu adotar uma estrutura padronizada de identificação já publicada pela ISO (*International Organization for Standardization*). A estrutura de identificação MIB II constitui parte da árvore da linguagem de definição de objetos ASN.1 (*Abstract Syntax Notation 1*). Os módulos MIB estão inseridos nesta estrutura da ISO conforme demonstrado na Figura 18. Nesta figura são apresentados quatro importantes módulos MIB padronizados: System; IP; TCP e UDP. Existem muitos outros módulos MIB padronizados (KUROSE; ROSS, 2004).

O módulo MIB System é considerado um dos mais importantes, pois os objetos contidos neste módulo dispõem informações gerais sobre o dispositivo que está sendo gerenciado. Todos os agentes devem suportar os objetos deste módulo (KUROSE; ROSS, 2004).

O módulo MIB mais importante para este projeto é o módulo IP, pois nele está contido o objeto ipRouteTable, ou seja, a tabela de roteamento do agente (ou do roteador que está sendo gerenciado).

Todos os objetos da árvore ASN.1 possuem seu identificador. O identificador é uma sequência de números inteiros separados por pontos que representam os níveis da hierarquia. Por exemplo, o identificador 1.3.6.1.2.1.4.21 representa o objeto ipRouteTable (STEVENS, 2008).

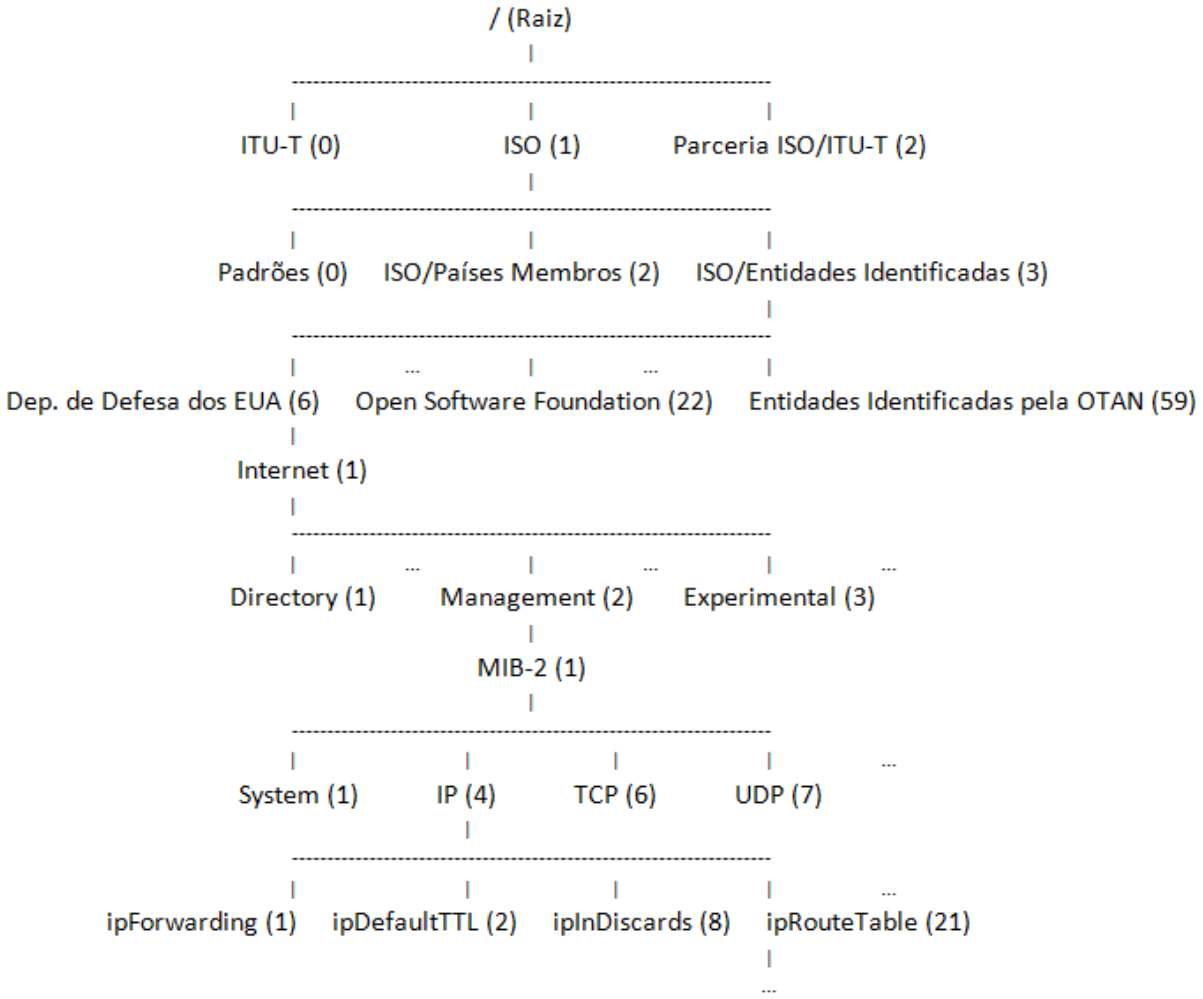


Figura 18 - Árvore de identificação de objetos da linguagem ASN.1.

Existem muitos tipos de dados entre os objetos dos módulos MIB. Alguns dos tipos de dados mais comuns são (KUROSE; ROSS, 2004) (STEVENS, 2008):

- Integer: Número inteiro de 32 bits. Exemplos: Métricas contidas na tabela de roteamento, ou seja, no ipRouteTable e os objetos ipForwarding e ipDefaultTTL;
- Counter: Contador de 32 bits. Exemplo: Objeto ipInDiscards;
- Endereço IP: Endereço Internet de 32. Exemplos: Endereços IP e máscaras de rede contidos no ipRouteTable;
- TimeTicks: Contador de tempo, em centésimos de segundo, transcorrido a partir de um evento. Exemplo: Objeto sysUpTime que define a quanto tempo o agente está ligado;
- Object Identifier (Object ID): Número identificador para um objeto da árvore ASN.1. Exemplo: A referência contida no ipRouteTable para as definições do protocolo de roteamento em vigor;

- Sequence: Contém mais de um objeto em sequência, não necessariamente do mesmo tipo. Exemplo: Uma linha da tabela de roteamento do ipRouteTable. Ela contém objetos como Endereços IP e métricas;
- Sequence Of: Define um vetor. Pode ser o agrupamento de objetos do tipo sequence ou uma tabela. Exemplo: Objeto ipRouteTable.

O protocolo SNMPv2 utiliza sete tipos de mensagens (KUROSE; ROSS, 2004):

- GetRequest: Utilizada pelo gerente para obter o conteúdo de um ou mais objetos;
- GetNextRequest: Utilizada pelo gerente para obter o conteúdo do próximo objeto de uma lista ou tabela;
- GetBulkRequest: Utilizada pelo gerente para obter objetos em grandes quantidades como o conteúdo de uma grande tabela;
- InformRequest: Utilizada por um gerente para informar a outro gerente remoto os dados da MIB requisitados de um agente local;
- Set Request: Utilizada por um gerente para definir um ou mais objetos em um agente;
- Response: Utilizada por um agente ou gerente para responder aos tipos de mensagens acima;
- SNMPv2-Trap: Utilizada por um agente para informar ao gerente um evento excepcional.

Existem sete tipos de TRAPs (STEVENS, 2008):

- coldStart: Informa que o agente está sendo inicializado;
- warmStart: Informa que o agente está sendo reinicializado;
- linkDown: Informa que uma determinada interface do agente mudou o estado de *up* para *down*;
- linkUp: Informa que uma determinada interface do agente mudou o estado de *down* para *up*;
- authenticationFailure: Informa que foi recebido uma mensagem de um gerente SNMP com autenticação inválida;
- egpNeighborLoss: Informa que um determinado par EGP ou enlace EGP caiu;
- enterpriseSpecific: Informa algum evento particular de um *software* proprietário.

O protocolo SNMP é de suma importância para o Observador Didático, pois é com este

protocolo que é feito toda a comunicação entre as máquinas virtuais e a máquina real. O *software* utilizado neste projeto é o Net-SNMP. Sua instalação na máquina real com o sistema operacional Ubuntu 10.04 é simples e dá-se através do comando "`apt-get install snmp snmpd snmptrapd`". A ferramenta instalada SNMP implementa um agente SNMP e é utilizada na máquina real apenas para testes locais.

O *daemon* Snmpd é responsável por fazer todas as requisições à tabela MIB de um agente SNMP. O *daemon* Snmptrapd é responsável por receber todas as mensagens TRAP e tratá-las e resolver o problema ou encaminhar a mensagem recebida a outro aplicativo quando necessário.

4.4 O cenário utilizado

Para a execução da ferramenta desenvolvida neste projeto, o Observador Didático, desenvolveu-se um cenário utilizando as máquinas virtuais descritas na Seção 4.1 UML. A estrutura do cenário, com suas conexões, pode ser visualizada na Figura 19.

No cenário apresentado na Figura 19 são utilizadas seis máquinas virtuais, sendo que três são roteadores e três são computadores comuns conectados individualmente a cada sub-rede de cada roteador. Na parte inferior direita da figura está representada a máquina real que provê sustentação física às máquinas virtuais. A máquina real está conectada aos três roteadores virtuais por enlaces diretos através das interfaces virtuais tap0, tap1 e tap2. A máquina real também serve como *gateway* para os roteadores virtuais provendo acesso à internet através da interface eth0.

Antes da execução do Observador Didático é necessário que algumas ferramentas estejam instaladas na máquina real:

- Snmpd e Snmptrapd: São responsáveis por implementar os *daemons* SNMP na máquina real descritos na Seção 4.3;
- Iptables: Responsável por fazer as configurações de rede necessárias para prover acesso à internet para as máquinas virtuais. Pode ser instalado com o comando "`apt-get install iptables`".

O Observador Didático é composto de *scripts* em Shell. A estrutura apresentada na Figura 19 é construída com a execução do *script* `redevirtual.sh` disponível no Anexo A. Os arquivos citados no início do *script*: Linux; Ubuntu9 e n3; uml_switch e tunctl. representam respectivamente as ferramentas UML Kernel, Sistemas de Arquivos e UML

utilities citadas na Seção 4.1.

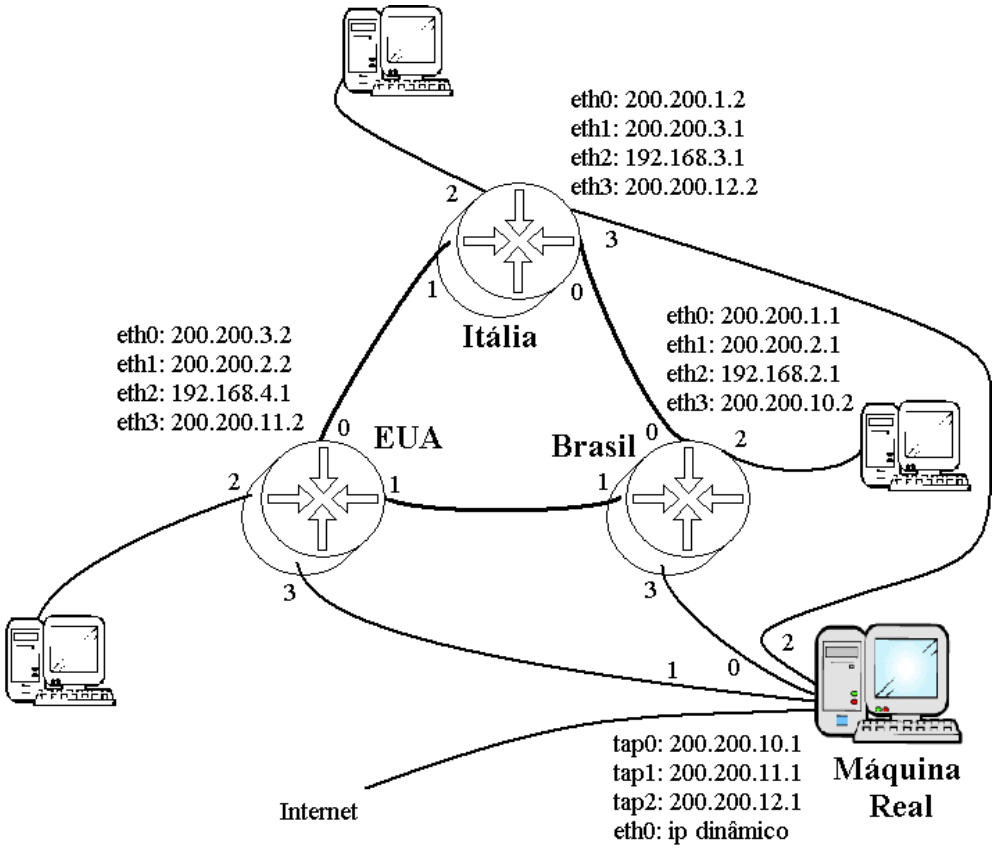


Figura 19 – Cenário desenvolvido para o Observador Didático.

O *script* `redevirtual.sh` inicialmente copia alguns arquivos e um *script* para o diretório /tmp. Tais arquivos poderiam também ser simplesmente criados, contudo, como em tais arquivos ficarão gravados os resultados após a execução desta ferramenta de forma bruta, ou simplesmente as tabelas de roteamento, concluiu-se que assim ficaria mais fácil gravar resultados posteriores num mesmo arquivo. No entanto, para tal é necessário que no final da primeira execução desta ferramenta tais arquivos devam ser copiados de volta manualmente do diretório /tmp para o diretório atual. Os arquivos de configuração snmpd.conf e snmptrapd.conf também são copiados do atual diretório para os seus respectivos diretórios /snmp/snmpd.conf e /etc/snmp/snmptrapd.conf.

Em seguida, o *script* `redevirtual.sh` cria e configura as interfaces virtuais tap0, tap1 e tap2 na máquina real com a ajuda da ferramenta Tunctl para mais tarde estabelecer enlaces virtuais com as máquinas reais. Feito isso, o script ativa o encaminhamento na máquina real para então torná-la um roteador. Logo em seguida são feitas configurações na máquina real para a tradução de endereços IP entre as interfaces virtuais e a interface real

através da ferramenta Iptables. Isso viabilizará o acesso à internet para as máquinas virtuais.

Com a ajuda das ferramentas Xterm e UML Switch, o então *script* `redevirtual.sh` inicializa os seis *switches* virtuais necessários para interconectar as máquinas virtuais. A seguir, este *script* executa sua principal tarefa: inicializar as seis máquinas virtuais para o Observador Didático com a ajuda das ferramentas descritas anteriormente na Seção 4.1.

Depois de inicializado o Observador Didático obtém-se as seguintes janelas demonstradas na Figura 20 onde cada uma representa uma máquina virtual.

É também responsabilidade do *script* `redevirtual.sh` encerrar o funcionamento do Observador Didático. Para tal o usuário deve simplesmente pressionar `<Ctrl>+<c>` no Terminal Linux onde este *script* foi executado. Automaticamente o *script* encerrará todos os processos que envolvem as máquinas virtuais e irá desfazer todas as configurações de rede feitas anteriormente na máquina real.

Dentro das máquinas virtuais está presente o *script* `autoconfiguracao.sh` apresentado no Anexo B. Ao serem inicializadas, as máquinas virtuais executam este *script*. As configurações de rede, em todas as máquinas virtuais, e as configurações do protocolo RIP e do protocolo SNMP, nas máquinas virtuais que são roteadores, são de responsabilidade do *script* `autoconfiguracao.sh`. Tais configurações são diferentes para cada máquina virtual. Como as máquinas virtuais, construídas aqui, compartilham o sistema de arquivos é necessário que o *script* `autoconfiguracao.sh` identifique primeiramente o nome da máquina virtual em vigor para que as configurações cabíveis sejam aplicadas.

O *script* `observador.sh` também está presente dentro das máquinas virtuais e é apresentado no Anexo C. Ele serve para monitorar as mudanças ocorridas na tabela de roteamento do roteador virtual em questão. Quando o script detecta qualquer modificação, este envia uma TRAP SNMP para a máquina real avisando o ocorrido.

Para detectar as mudanças na tabela de roteamento, o *script* `observador.sh` cria um *loop* infinito onde inicialmente, a tabela é copiada para um arquivo de texto. De ciclo em ciclo a atual tabela de roteamento é comparada com a tabela de roteamento presente no arquivo de texto. Quando ocorre uma alteração na tabela de roteamento, o arquivo de texto é atualizado e uma TRAP é enviada à máquina real contendo a informação temporal do momento em que a mudança foi detectada.

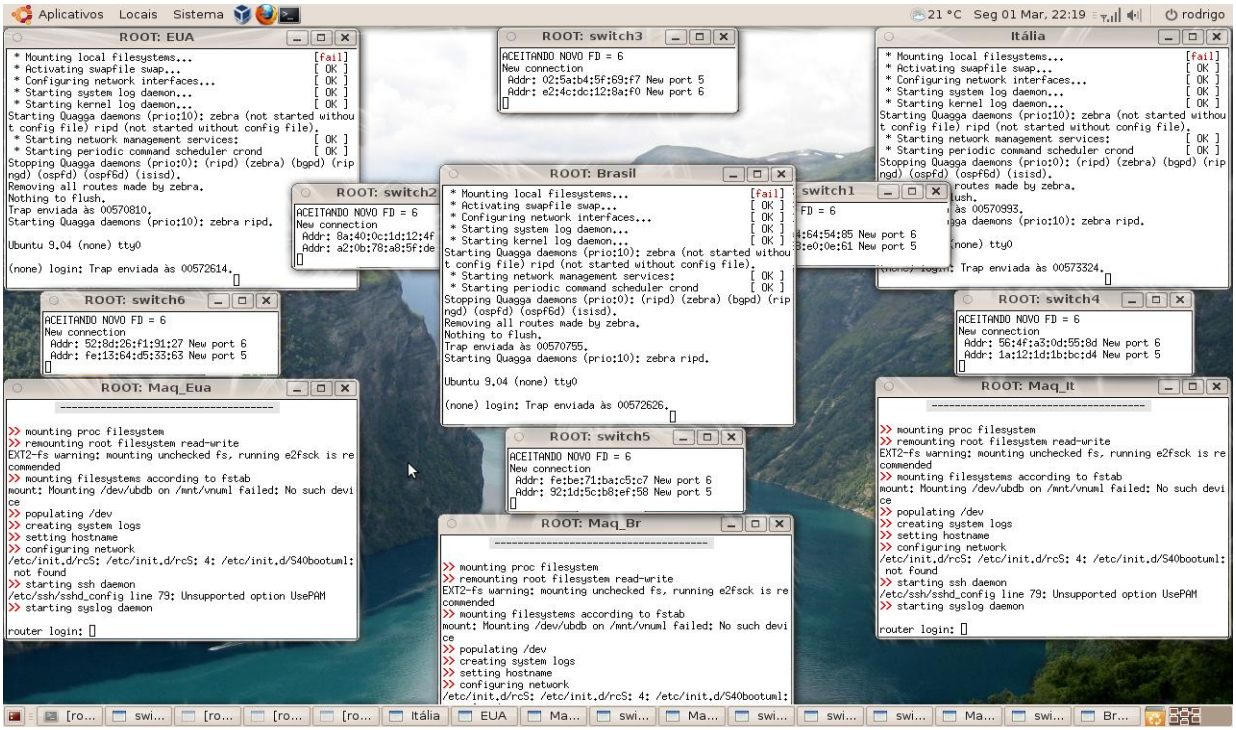


Figura 20 – Observador Didático em execução.

Na máquina real, o *script* `inFile.sh` é responsável por filtrar as TRAPs importantes para o Observador Didático, ou seja, as TRAPs geradas pelo *script* `observador.sh`.Ele é apresentado no Anexo D. Quando o *script* `inFile.sh` encontra uma TRAP importante para o Observador Didático este requisita a tabela de roteamento para o roteador em questão.

A captura da tabela de roteamento de um roteador virtual é feita na máquina real com a ajuda do protocolo SNMP. Após a execução do comando "`snmpnetstat -On -v2c -c alunos -Crn 200.200.10.2 >> /tmp/Brasil.out`" pelo *script* `inFile.sh` a máquina real requisita ao roteador virtual Brasil a tabela de roteamento presente na sua MIB. A tabela de roteamento chega à máquina real través de uma grande troca de pacotes, isto devido ao SNMP transferir um campo da tabela de roteamento por vez. Quanto maior a tabela de roteamento maior a troca de mensagens SNMP. Com este comando a tabela de roteamento é redirecionada e anexada no arquivo de texto Brasil.out disponível no diretório /tmp para análise do usuário.

4.5 Análise dos resultados das observações

As mudanças nas tabelas de roteamento dos três roteadores virtuais podem ser visualizadas nos arquivos de texto Brasil.out, Italia.out e EUA.out disponíveis no diretório

/tmp.

Para acompanhar as modificações nas tabelas de roteamento em tempo real do roteador virtual Brasil pode-se utilizar o comando "`tail -f /tmp/Brasil.out`". Na Figura 21 é apresentada a tabela de roteamento do roteador Brasil durante a sua inicialização e após a convergência do protocolo RIP. Sabe-se que houve estágios intermediários da tabela de roteamento antes da convergência do protocolo RIP. Tais estágios intermediários nem sempre são capturados pelo Observador Didático devido à velocidade com que a tabela de roteamento é atualizada.

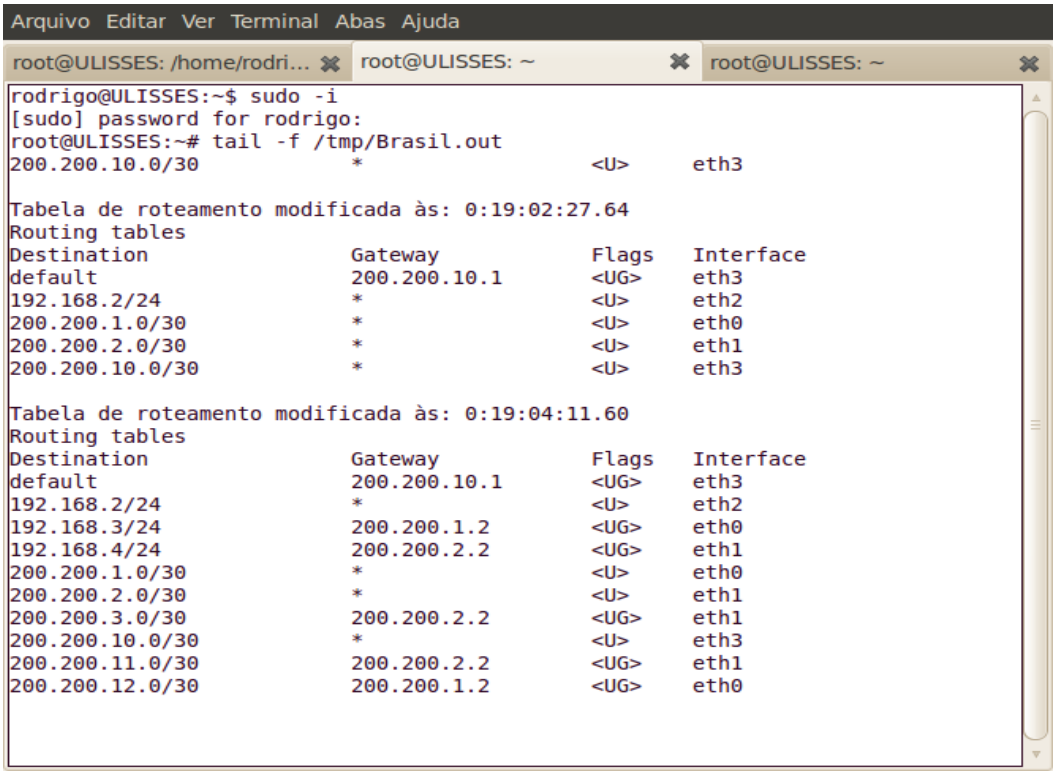


Figura 21 – Captura da tabela de roteamento em tempo real.

Na primeira tabela de roteamento, modificada às 19 horas e 02 minutos, da Figura 21 é possível visualizar as rotas da máquina virtual Brasil antes da inicialização do protocolo RIP. O *gateway* 200.200.10.1 para a rota de saída padrão (*default*) é a própria máquina real, que provê acesso à internet para as máquinas virtuais. A rede 192.168.2/24 é a rede local do roteador Brasil, que pode ser acessada diretamente através da interface eth2. As redes 200.200.1.0/30 e 200.200.2.0/30 são enlaces que conectam o roteador Brasil aos respectivos roteadores Itália e EUA. A rede 200.200.10.0/30 é o enlace que se conecta à máquina real.

Na segunda tabela de roteamento, modificada às 19:04, da Figura 21 é possível

visualizar outras cinco novas rotas da máquina virtual Brasil depois da inicialização do protocolo RIP. As redes 192.168.3/24 e 192.168.4/24 são redes locais dos respectivos roteadores Itália e Brasil. A rede 200.200.2.0/30 é o enlace que conecta os roteadores EUA e Itália. As redes 200.200.11.0/30 e 200.200.12.0/30 são enlaces que respectivamente conectam os roteadores EUA e Itália à máquina real. Tais conclusões sobre as tabelas de roteamento podem ser averiguadas na Figura 19.

Nem todas as etapas de evolução da tabela de roteamento podem ser capturadas devido à transferência da tabela de roteamento ser mais lenta do que a velocidade com que a tabela de roteamento é atualizada na máquina virtual. Ou seja, quando ocorre uma atualização na tabela de roteamento de um roteador virtal, uma mensagem trap é enviada para a máquina real e a tabela de roteamento do roteador virtual deve ser capturada pela máquina real. Contudo, se enquanto a máquina real está obtendo a tabela de roteamento ocorrer uma nova atualização na mesma perde-se o estágio anterior à atualização. Isso impede uma análise minuciosa da evolução do algoritmo vetor de distância.

É possível saber quando ocorre uma mudança na tabela de roteamento, pois aparece uma mensagem no terminal do mesmo indicando o momento que uma mensagem TRAP foi enviada. Nos arquivos de texto .out da máquina real é posto juntamente com cada tabela de roteamento o mesmo momento em que a TRAP, que gerou a captura da respectiva tabela, foi enviada. Assim é possível notar o atraso desde quando a mensagem da TRAP enviada aparece na máquina virtual até quando aparece a tabela de roteamento no arquivo .out. Se ocorrer alguma alteração na tabela de roteamento durante este período de transferência da mesma a tabela anterior à modificação se perde e é capturada a tabela de roteamento mais atual.

Com a ajuda da ferramenta Tcpdump é possível capturar todos os pacotes RIP que trafegam na rede virtual e compará-los as mudanças causadas nas tabelas de roteamento. Um exemplo está na Figura 17 onde é possível acompanhar as primeiras trocas de mensagens RIP entre os roteadores virtuais durante a sua inicialização. Nota-se que o roteador Brasil inicializou primeiro seguido do roteador EUA e que o roteador Itália ainda não inicializou.

Com o Observador Didático é possível realizar estudos de caso. Por exemplo, derrubar um enlace virtual entre dois roteadores e analisar as mudanças nas tabelas de roteamento da rede virtual ou ainda derrubar um enlace entre um roteador e a máquina real para analisar se as mensagens SNMP continuam a ser trocadas corretamente. Na Figura 22 é possível visualizar as mudanças decorridas na tabela de roteamento após o comando "`ip link set dev eth0 down`", na máquina virtual Brasil, para derrubar o enlace Brasil-Itália. Nota-se

que todas as rotas que utilizavam a interface eth0 passaram a utilizar a interface eth1.

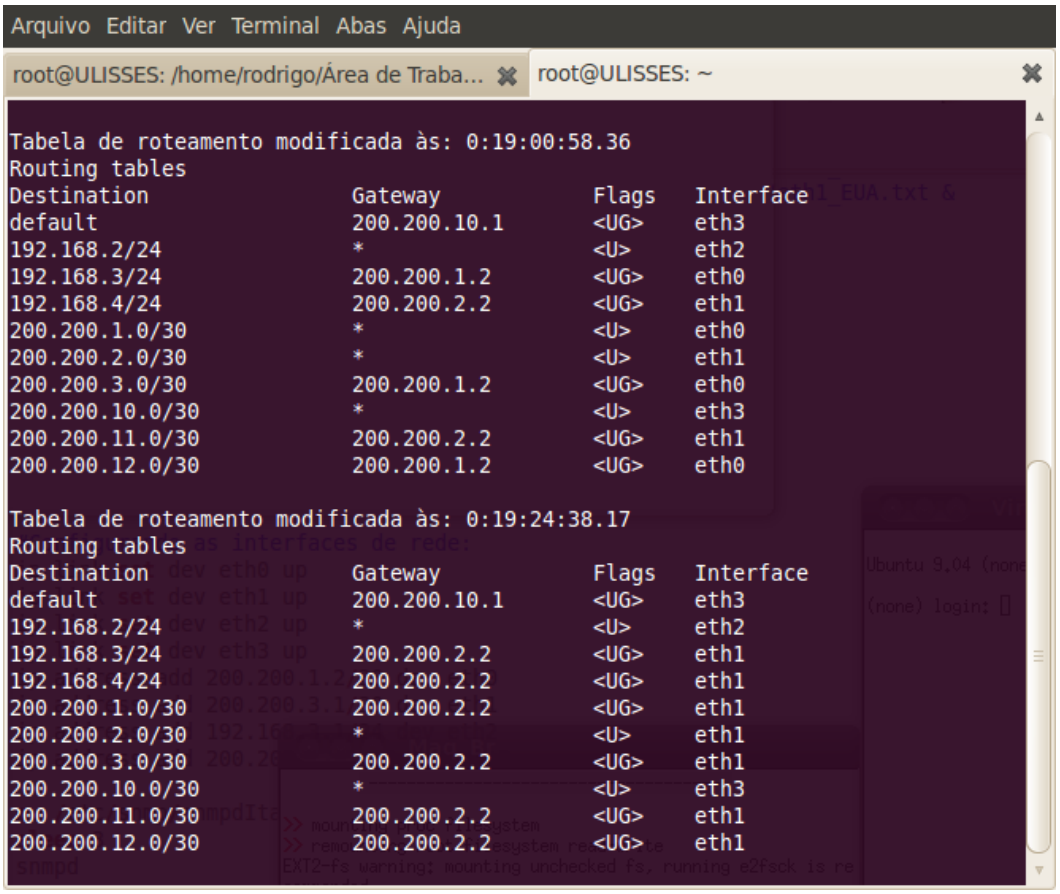


Figura 22 – Modificações na tabela de roteamento do roteador Brasil após a queda do enlace Brasil-Itália.

5 Conclusão

Com a ferramenta Observador Didático é possível fazer a captura e monitoramento das tabelas de roteamento dos três roteadores conforme verificado na Figura 19, possibilitando assim um auxílio para a interpretação prática do funcionamento dos protocolos. Com algumas mudanças nesta ferramenta é possível mudar a topologia da rede virtual para redes mais complexas.

Esta ferramenta também pode ser facilmente adaptada para o estudo de outros protocolos de roteamento, tais como o OSPF. Para tal seria necessário fazer alterações somente no *script* `autoconfiguracao.sh`.

Inicialmente, com os estudos realizados sobre os protocolos de roteamento e com os testes realizados através da ferramenta de simulação Opnet foi possível fixar e aprofundar o estudo realizado na área de roteamento. No decorrer do projeto, com os estudos e testes realizados com as máquinas virtuais UML, com a ferramenta Tcpdump e a sua biblioteca libpcap.h, com o protocolo SNMP e a ferramenta Net-SNMP, com o desenvolvimento de *scripts* em Shell e entre outros estudos foi possível fixar, aprofundar e aplicar conteúdos importantes para a área de gerência de redes.

Com o desenvolvimento deste projeto foi possível adquirir experiências muito importantes do cotidiano da gerência de redes. Muitos problemas foram encontrados e tiveram de ser superados ao longo do projeto, entre eles, manipulação de Shell scripts, configurações de rede em geral, montagem e configuração das máquinas virtuais e, finalmente, o uso do protocolo SNMP.

O objetivo inicial deste projeto era auxiliar o estudo de algoritmos e protocolos de roteamento a partir da captura de todas as mensagens trocadas por um protocolo de roteamento dentro de uma rede, contudo, o desenvolvimento de uma ferramenta baseada na biblioteca libpcap.h se mostrou muito trabalhosa conforme comentado na Seção 4.2. Então se pesquisou outros métodos tais como *sockets* e o protocolo SNMP. O protocolo SNMP foi escolhido por ser um protocolo padrão para o gerenciamento de redes, sendo assim, no futuro facilmente poder-se-ia migrar o Observador Didático para abranger equipamentos de rede

reais.

No decorrer do desenvolvimento do Observador Didático, principalmente na etapa da implantação do protocolo SNMP, muitas dificuldades foram encontradas, entre elas o problema das mensagens TRAPs não geradas automaticamente, no sistema operacional Ubuntu Server 9.10, quando um enlace era derrubado ou quando o mesmo era levantado. Para resolver tais problemas estudos aprofundados sobre a ferramenta Net-SNMP foram realizados e decidiu-se que as mensagens TRAPs deveriam ser geradas pelo *script* `observador.sh`. Testes recentes com o sistema de arquivos Ubuntu 10.04 concluíram que todas as mensagens TRAPs estão sendo geradas automaticamente, contudo, as TRAPs geradas pelo *script* `observador.sh` continuam sendo de grande valor pois elas indicam quando se descobre ou se perde uma rota. As TRAPs geradas automaticamente indicam somente quando um enlace é derrubado ou quando é levantado.

Uma perspectiva de continuidade deste projeto é implementar a apresentação das tabelas de roteamento de forma gráfica através da linguagem dot.

A linguagem dot é capaz de gerar gráficos a partir de um código após compilado. Ela é uma linguagem simples, no entanto pode gerar gráficos muito complexos de forma rápida (GANSNER; KOUTSOFIOS; NORTH, 2009).

A apresentação das modificações nas tabelas de roteamento de forma gráfica poderia ser gerida por um *software* independente deste projeto, facilitando assim o futuro desenvolvimento do mesmo.

Anexo A

Script **redevirtual.sh**:

```
#!/bin/bash

echo -e "Tenha certeza de que você está em modo root."
sleep 1
echo -e "\nOs arquivos linux, tunctl, Ubuntu9, n3 e uml_switch
devem estar no mesmo"
# O flag -e faz com que a '\' e o caractere a seguir sejam
interpretados e não, simplesmente, expostos na tela.
echo "diretório deste script que está sendo executado."
echo -e "\nPara parar este script pressione <Ctrl>+<c>"
sleep 2

echo -e "\nConfigurando as interfaces virtuais na máquina
real:"

# Obtendo o nome de usuário:
#USUARIO=`who | grep tty7 | awk '{ print $1 }'`
USUARIO=$(who -m | cut -d' ' -f1)

# Copiando arquivos para o diretório /tmp:
# Estes arquivos servem para a manipulação das traps que
chegam no gerente.
cp ./Brasil.out /tmp/Brasil.out
cp ./EUA.out /tmp/EUA.out
cp ./Italia.out /tmp/Italia.out
cp ./inFile.sh /tmp/inFile.sh
cp ./snmpd.conf /etc/snmp/snmpd.conf
cp ./snmptrapd.conf /etc/snmp/snmptrapd.conf

# Reiniciando daemons SNMP:
snmpd
snmptrapd

# Criando as interfaces virtuais tap0, tap1 e tap2:
./tunctl -u $USUARIO
./tunctl -u $USUARIO
./tunctl -u $USUARIO
```

```
ip link set dev tap0 up
ip link set dev tap1 up
ip link set dev tap2 up
ip address add 200.200.10.1/30 dev tap0 # Brasil
ip address add 200.200.11.1/30 dev tap1 # EUA
ip address add 200.200.12.1/30 dev tap2 # Itália

# Ativando roteamento do tráfego das máquinas virtuais para
fora da interface da máquina real:
# Ativando o encaminhamento:
echo "1" > /proc/sys/net/ipv4/ip_forward

# Salvando a política padrão da chain FORWARD:
POLITICA=`iptables -L FORWARD | grep policy | awk '{ print $4
}' | awk -F ')' '{ print $1 }'`

# Configurando a política padrão da chain FORWARD para ACCEPT:
iptables -P FORWARD ACCEPT

# Ativando o mascaramento das três redes:
iptables -t nat -A POSTROUTING -s 200.200.10.2/30 -j
MASQUERADE
iptables -t nat -A POSTROUTING -s 200.200.11.2/30 -j
MASQUERADE
iptables -t nat -A POSTROUTING -s 200.200.12.2/30 -j
MASQUERADE


echo -e "\nLançamento dos switches:"

xterm -geometry 40x5 -T switch1 -e "./uml_switch -unix
/tmp/switch1.ctl"  &
xterm -geometry 40x5 -T switch2 -e "./uml_switch -unix
/tmp/switch2.ctl"  &
xterm -geometry 40x5 -T switch3 -e "./uml_switch -unix
/tmp/switch3.ctl"  &
xterm -geometry 40x5 -T switch4 -e "./uml_switch -unix
/tmp/switch4.ctl"  &
xterm -geometry 40x5 -T switch5 -e "./uml_switch -unix
/tmp/switch5.ctl"  &
xterm -geometry 40x5 -T switch6 -e "./uml_switch -unix
/tmp/switch6.ctl"  &
sleep 2

echo -e "\nLançamento dos roteadores:"

xterm -geometry 60x20 -T Brasil -e "./linux umid=brasil
ubda=./cow-brasil,./Ubuntu9 mem=64M
eth0=daemon,,,/tmp/switch1.ctl eth1=daemon,,,/tmp/switch2.ctl
eth2=daemon,,,/tmp/switch5.ctl eth3=tuntap,tap0" &
```

```
xterm -geometry 60x20 -T EUA -e "./linux umid=eua  ubda=./cow-
eua,./Ubuntu9 mem=64M eth0=daemon,,,/tmp/switch3.ctl
eth1=daemon,,,/tmp/switch2.ctl eth2=daemon,,,/tmp/switch6.ctl
eth3=tuntap,tap1" &

xterm -geometry 60x20 -T Itália -e "./linux umid=italia
ubda=./cow-italia,./Ubuntu9 mem=64M
eth0=daemon,,,/tmp/switch1.ctl eth1=daemon,,,/tmp/switch3.ctl
eth2=daemon,,,/tmp/switch4.ctl eth3=tuntap,tap2" &
sleep 2

echo -e "\nLançamento das máquinas virtuais UML:"

xterm -geometry 60x20 -T Maq_Br -e "./linux umid=maq_br
ubda=./cow-maq_br,./n3 mem=32M eth0=daemon,,,/tmp/switch5.ctl"
&

xterm -geometry 60x20 -T Maq_Eua -e "./linux umid=maq_eua
ubda=./cow-maq_eua,./n3 mem=32M
eth0=daemon,,,/tmp/switch6.ctl" &

echo -e "\nPara finalizar pressione <Ctrl>+<c>"

xterm -geometry 60x20 -T Maq_It -e "./linux umid=maq_it
ubda=./cow-maq_it,./n3 mem=32M eth0=daemon,,,/tmp/switch4.ctl"


killall linux
#kill $(pidof linux)
#kill $(pidof xterm)
rm ./cow-*
rm /tmp/switch*

# Derrubando as interfaces virtuais tap0, tap1 e tap2:
ip link set dev tap0 down
ip link set dev tap1 down
ip link set dev tap2 down

# Desfazendo as configurações no firewall:
iptables -P FORWARD $POLITICA
iptables -t nat -D POSTROUTING -s 200.200.10.2/30 -j
MASQUERADE
iptables -t nat -D POSTROUTING -s 200.200.11.2/30 -j
MASQUERADE
iptables -t nat -D POSTROUTING -s 200.200.12.2/30 -j
MASQUERADE

# Apagando as interfaces virtuais:
./tunctl -d tap0
./tunctl -d tap1
./tunctl -d tap2
```

Anexo B

Script **autoconfiguracao.sh**:

```
#!/bin/bash

# Script de autoconfiguração das máquinas virtuais

# Este script deve ser executado após o script de
inicialização rcS.

# Ao rodar, o script identifica em qual máquina virtual ele se
encontra,
# configura as interfaces e, para o caso dos roteadores,
também configura o protocolo RIP.

# Comando que identifica o nome da máquina virtual:
NOME=`dmesg | grep mconsole | awk '{ print $6 }' | awk -F/ '{
print $4}'`
# dmesg mostra as mensagens da última inicialização do
sistema.

/etc/init.d/quagga stop

sleep 10

# Verifica o conteúdo de NOME para executar os comandos
específicos para cada máquina virtual:
case $NOME in

### Roteadores ###

brasil)

#Configurando as interfaces de rede:
ip link set dev eth0 up
ip link set dev eth1 up
ip link set dev eth2 up
ip link set dev eth3 up
ip address add 200.200.1.1/30 dev eth0
ip address add 200.200.2.1/30 dev eth1
ip address add 192.168.2.1/24 dev eth2
```

```
ip address add 200.200.10.2/30 dev eth3

mv /etc/snmp/snmpdBrasil.conf /etc/snmp/snmpd.conf
sleep 1
snmpd

sleep 1
/etc/init.d/monitor.sh 200.200.10.1 &

##Protocolos de roteamento##
#Copiando o arquivo modelo:
cp /usr/share/doc/quagga/examples/zebra.conf.sample
/etc/quagga/zebra.conf

#Removendo a última linha do arquivo ripd:
head -n 23 /usr/share/doc/quagga/examples/ripd.conf.sample >
/etc/quagga/ripd.conf

#Adicionando os parâmetros no ripd.conf:
echo redistribute connected >> /etc/quagga/ripd.conf
echo redistribute static >> /etc/quagga/ripd.conf
echo network eth0 >> /etc/quagga/ripd.conf
echo network eth1 >> /etc/quagga/ripd.conf
echo distribute-list private in eth0 >> /etc/quagga/ripd.conf
echo distribute-list private out eth0 >> /etc/quagga/ripd.conf
echo distribute-list private in eth1 >> /etc/quagga/ripd.conf
echo distribute-list private out eth1 >> /etc/quagga/ripd.conf
echo log stdout >> /etc/quagga/ripd.conf

#Iniciando captura de pacotes:
#tcpdump -i eth0 -s 0 -n -v -c 20 not ip6 and not igmp >>
./eth0_Brasil.txt &
#tcpdump -i eth1 -s 0 -n -v -c 20 not ip6 and not igmp >>
./eth1_Brasil.txt &

#Iniciando os serviços:
#/usr/sbin/zebra -d
#/usr/sbin/ripd -d
#/etc/init.d/quagga restart

sleep 3
ip route add default via 200.200.10.1
;;

eua)

#Configurando as interfaces de rede:
ip link set dev eth0 up
ip link set dev eth1 up
ip link set dev eth2 up
ip link set dev eth3 up
```

```
ip address add 200.200.3.2/30 dev eth0
ip address add 200.200.2.2/30 dev eth1
ip address add 192.168.4.1/24 dev eth2
ip address add 200.200.11.2/30 dev eth3

mv /etc/snmp/snmpdEUA.conf /etc/snmp/snmpd.conf
sleep 2
snmpd

sleep 1
/etc/init.d/monitor.sh 200.200.11.1 &

##Protocolos de roteamento##
#Copiando o arquivo modelo:
cp /usr/share/doc/quagga/examples/zebra.conf.sample
/etc/quagga/zebra.conf

#Removendo a última linha do arquivo ripd:
head -n 23 /usr/share/doc/quagga/examples/ripd.conf.sample >
/etc/quagga/ripd.conf

#Adicionando os parâmetros no ripd.conf:
echo redistribute connected >> /etc/quagga/ripd.conf
echo redistribute static >> /etc/quagga/ripd.conf
echo network eth0 >> /etc/quagga/ripd.conf
echo network eth1 >> /etc/quagga/ripd.conf
echo distribute-list private in eth0 >> /etc/quagga/ripd.conf
echo distribute-list private out eth0 >> /etc/quagga/ripd.conf
echo distribute-list private in eth1 >> /etc/quagga/ripd.conf
echo distribute-list private out eth1 >> /etc/quagga/ripd.conf
echo log stdout >> /etc/quagga/ripd.conf

#Iniciando captura de pacotes:
#tcpdump -i eth0 -s 0 -n -v -c 20 not ip6 and not igmp >>
./eth0_EUA.txt &
#tcpdump -i eth1 -s 0 -n -v -c 20 not ip6 and not igmp >>
./eth1_EUA.txt &

#Iniciando os serviços:
#/usr/sbin/zebra -d
#/usr/sbin/ripd -d
#/etc/init.d/quagga restart

sleep 3
ip route add default via 200.200.11.1
;;

italia)

#Configurando as interfaces de rede:
ip link set dev eth0 up
```

```
ip link set dev eth1 up
ip link set dev eth2 up
ip link set dev eth3 up
ip address add 200.200.1.2/30 dev eth0
ip address add 200.200.3.1/30 dev eth1
ip address add 192.168.3.1/24 dev eth2
ip address add 200.200.12.2/30 dev eth3

mv /etc/snmp/snmpdItalia.conf /etc/snmp/snmpd.conf
sleep 3
snmpd

sleep 1
/etc/init.d/monitor.sh 200.200.12.1 &

##Protocolos de roteamento##
#Copiando o arquivo modelo:
cp /usr/share/doc/quagga/examples/zebra.conf.sample
/etc/quagga/zebra.conf

#Removendo a última linha do arquivo ripd:
head -n 23 /usr/share/doc/quagga/examples/ripd.conf.sample >
/etc/quagga/ripd.conf

#Adicionando os parâmetros no ripd.conf:
echo redistribute connected >> /etc/quagga/ripd.conf
echo redistribute static >> /etc/quagga/ripd.conf
echo network eth0 >> /etc/quagga/ripd.conf
echo network eth1 >> /etc/quagga/ripd.conf
echo distribute-list private in eth0 >> /etc/quagga/ripd.conf
echo distribute-list private out eth0 >> /etc/quagga/ripd.conf
echo distribute-list private in eth1 >> /etc/quagga/ripd.conf
echo distribute-list private out eth1 >> /etc/quagga/ripd.conf
echo log stdout >> /etc/quagga/ripd.conf

#Iniciando captura de pacotes:
#tcpdump -i eth0 -s 0 -n -v -c 20 not ip6 and not igmp >>
./eth0_Italia.txt &
#tcpdump -i eth1 -s 0 -n -v -c 20 not ip6 and not igmp >>
./eth1_Italia.txt &

#Iniciando os serviços:
#/usr/sbin/zebra -d
#/usr/sbin/ripd -d
#/etc/init.d/quagga restart

sleep 3
ip route add default via 200.200.12.1
;;
```

```
### Hospedeiros ###

maq_br)
#Configurando a interface de rede:
echo 0 > /proc/sys/net/ipv4/ip_forward
ip link set dev eth0 up
ip address add 192.168.2.2/24 dev eth0
ip route add default via 192.168.2.1
;;

maq_eua)
#Configurando a interface de rede:
echo 0 > /proc/sys/net/ipv4/ip_forward
ip link set dev eth0 up
ip address add 192.168.4.2/24 dev eth0
ip route add default via 192.168.4.1
;;

maq_it)
#Configurando a interface de rede:
echo 0 > /proc/sys/net/ipv4/ip_forward
ip link set dev eth0 up
ip address add 192.168.3.2/24 dev eth0
ip route add default via 192.168.3.1
;;

esac

# Adicionando servidores DNS do IFSC campus São José ao
resolv.conf:
echo nameserver 200.135.37.65 > /etc/resolv.conf
echo nameserver 172.18.0.1 >> /etc/resolv.conf

# Adicionando um repositório de pacotes:
#echo deb http://security.debian.org/ etch/updates main
contrib > /etc/apt/source.list

#apt-get update
#apt-get --force-yes install snmpd snmp quagga

sleep 5

/etc/init.d/quagga start
```

Anexo C

Script **observador.sh**:

```
#!/bin/bash

# Script que monitora a tabela de roteamento e envia uma trap
quando acontece alguma alteração.


rm -rf /tmp/tabela.txt
rm -rf /tmp/tabela1.txt
touch /tmp/tabela.txt
touch /tmp/tabela1.txt


while :
do

     # Imprime a tabela de roteamento em /tmp/tabela.txt sem
as duas primeiras linhas de título:
     route -n | sed '1,2d' > /tmp/tabela.txt
     #cat /tmp/tabela.txt

     # Obtém o número de linhas da tabela de roteamento:
     linhas=$(cat /tmp/tabela.txt | wc -l)

     for i in $(seq $linhas)
     do

          for j in $(seq 8)
          do

               palavra1=$(cat /tmp/tabela.txt | head -n $i |
tail -n 1 | awk "{print $`echo $j`}")
               palavra2=$(cat /tmp/tabela1.txt | head -n $i |
tail -n 1 | awk "{print $`echo $j`}")

               if [ $palavra1 != ${palavra2:=a} ]
               then

                    enviatrap="sim"
```

```
                    break

               fi

          done

          if [ $enviatrap = "sim" ]
          then

               break

          fi

     done
     if [ $enviatrap = "sim" ]
     then

          enviatrap="nao"
          hora=$(date +%H%M%S%N | cut -c 1-10)
          echo -e "trap enviada às $hora"
          snmptrap -v 1 -c alunos $1 enterprises.0 $1 6 0 ''
hrSystemUptime.0 t $hora
          cat /tmp/tabela.txt > /tmp/tabela1.txt

     fi

done
```

Anexo D

Script **inFile.sh**:

```
#!/bin/sh

read host
read ip
read val
read oid

ipe=`echo "$ip" | awk -F '[' '{ print $2 }' | awk -F ']' '{
print $1 }'`

#Filtrando mensagens da máquina Brasil:
if [ $ipe = 200.200.10.2 ]
then
     echo >> /tmp/Brasil.out
     hora=$(echo $oid | awk '{ print $2 }')
     if [ `echo "$oid" | awk -F ':' '{ print $5 }'` !=
"coldStart" ]
     then
          echo "Tabela de roteamento modificada às: $hora" >>
/tmp/Brasil.out
          #echo "OID: $oid" >> /tmp/Brasil.out
          #echo "val: $val" >> /tmp/Brasil.out
          snmpnetstat -On -v2c -c alunos -Crn 200.200.10.2 >>
/tmp/Brasil.out
     fi
fi

#Filtrando mensagens da máquina EUA:
if [ $ipe = 200.200.11.2 ]
then
     echo >> /tmp/EUA.out
     hora=$(echo $oid | awk '{ print $2 }')
     if [ `echo "$oid" | awk -F ':' '{ print $5 }'` !=
"coldStart" ]
     then
          echo "Tabela de roteamento modificada às: $hora" >>
/tmp/EUA.out
          #echo "OID: $oid" >> /tmp/EUA.out
```

```
          #echo "val: $val" >> /tmp/EUA.out
          snmpnetstat -On -v2c -c alunos -Crn 200.200.11.2 >>
/tmp/EUA.out
     fi
fi

#Filtrando mensagens da máquina Itália:
if [ $ipe = 200.200.12.2 ]
then
     echo >> /tmp/Italia.out
     hora=$(echo $oid | awk '{ print $2 }')
     if [ `echo "$oid" | awk -F ':' '{ print $5 }'` !=
"coldStart" ]
     then
          echo "Tabela de roteamento modificada às: $hora" >>
/tmp/Italia.out
          #echo "OID: $oid" >> /tmp/Italia.out
          #echo "val: $val" >> /tmp/Italia.out
          snmpnetstat -On -v2c -c alunos -Crn 200.200.12.2 >>
/tmp/Italia.out
     fi
fi
```

Lista de abreviaturas e siglas

RIP – *Routing Information Protocol* (Protocolo de Informação de Roteamento)

OSPF – *Open Shortest Path First* (Primeira Rota Aberta mais Curta)

BGP – *Border Gateway Protocol* (Protocolo de Acesso de Borda)

EGP – *Exterior Gateway Protocol* (Protocolo de Acesso Exterior)

UML – *User Mode Linux* (Usuário Modo Linux)

COW – *Copy-On-Write* (Cópia da Escrita)

TCP – *Transmission Control Protocol* (Protocolo de Controle de Transmissão)

IP – *Internet Protocol* (Protocolo de Internet)

RFC – *Request for Comments* (Requisição de Comentários)

PDU – *Protocol Data Unit* (Unidade de Informação do Protocolo)

ICMP – *Internet Control Message Protocol* (Protocolo de Mensagens de Controle da Internet)

DHCP – *Dynamic Host Configuration Protocol* (Protocolo de Configuração Dinâmica de Hospedeiros)

IGMP – *Internet Group Management Protocol* (Protocolo de Gerenciamento de Grupos da Internet)

PPP – *Point-to-Point Protocol* (Protocolo Ponto-a-Ponto)

QoS – *Quality of Service* (Qualidade de Serviço)

ToS – *Type of Service* (Tipo de Serviço)

TTL – *Time to Live* (Tempo de Vida)

LAN – *Local Area Network* (Rede de Área Local)

AS – *Autonomous System* (Sistema Autônomo)

SNMP – *Simple Network Management Protocol* (Protocolo Simples de Gerência de Rede)

MIB – *Management Information Base* (Base de Informações de Gerenciamento)

IETF – *Internet Engineering Task Force* (Força-Tarefa de Engenharia na Internet)

ISO – *International Organization for Standardization* (Organização Internacional para Padronização)

ASN.1 – *Abstract Syntax Notation 1* (Notação de Sintaxe Abstrata 1)

DNS – *Domain Name System* (Sistema de Nomes de Domínio)

Referências bibliográficas

ASSIS, A. U. e ALVES, N. Protocolos de Roteamento RIP e OSPF, Setembro 2001. Disponível em: <http://www.rederio.br/downloads/pdf/nt01100.pdf>. Acessado em 16/06/2010.

CANTÚ, E. Elementos para o Fortalecimento da Mediação Docente na Educação Tecnológica: Aplicação no Ensino Aprendizagem de Redes de Computadores. Tese (Doutorado em Engenharia Elétrica), Universidade Federal de Santa Catarina, ago.2005.

CANTÚ, E. Redes de Computadores e a Internet. Instituto Federal de Santa Catarina, São José, fev.2009.

COMER, D. E. Interligação de Redes com TCP/IP, 5ª edição, Campus, 2006.

DIKE, J. User Mode Linux, 1ª edição, Prentice Hall, 2006.

GANSNER, E. R.; KOUTSOFIOS, E.; NORTH, S. "Drawing graphs with dot". Disponível em: <http://www.graphviz.org/pdf/dotguide.pdf>. Acessado em: 03/03/2010.

INÁCIO, F. C. MPLS Multiprotocol Label Switching, 2002. Disponível em: <http://www.gta.ufrj.br/grad/02_1/mpls/apres.html>. Acessado em: 29/03/2010.

KRASNYANSKY, M. Universal TUN/TAP device driver, 2000. Disponível em: <http://www.kernel.org/pub/linux/kernel/people/marcelo/linux-2.4/Documentation/networking/tuntap.txt>. Acessado em: 29/03/2010.

KUROSE, J. e ROSS, K. Redes de Computadores: Uma abordagem top-down, 3a edição, Campus, 2004.

LOPES, A. V. Estruturas de Dados para a Construção de Software, 1ª edição, ULBRA, 1999.

PETERSON, L. L. e DAVIE, B. S. Redes de Computadores: Uma abordagem de Sistemas, 3ª

edição, Campus, 2004.

RFC 1058, Routing Information Protocol, Junho 1988. Disponível em: <http://www.rfc-editor.org/cgi-bin/rfcdoctype.pl?loc=RFC&letsgo=1058&type=http&file_format=pdf>. Acessado em: 15/07/2010.

RFC 1256, ICMP Router Discovery Messages, S. Deering, Setembro 1991. Disponível em: <http://tools.ietf.org/html/rfc1256>. Acessado em: 20/05/2010.

RFC 1723, RIP Version 2 Carrying Additional Information, Novembro 1994. Disponível em: <http://www.rfc-editor.org/cgi-bin/rfcdoctype.pl?loc=RFC&letsgo=1723&type=http&file_format=pdf>. Acessado em: 15/07/2010.

RUSSEL, R. e STEARNS, B. User Mode Linux HOWTO, 2008. Disponível em: <http://user-mode-linux.sourceforge.net/old/UserModeLinux-HOWTO.html>. Acessado em: 29/03/2010.

SCHNEIDER, C. S. Utilização dos aspectos ergonômicos na simulação de sistemas de produção. Trabalho de conclusão do Curso de Mestrado Profissionalizante em Engenharia. Universidade Federal do Rio Grande do Sul, Porto Alegre, 2004.

STEVENS, W. R. TCP/IP Illustrated, Volume 1: The Protocols, 1ª edição, Addison Wesley, 2008.

TANENBAUM, A. S. Redes de Computadores, 4ª edição, Campus, 2003.